Imperial College London

Integration and visualisation of clinical-omics datasets for medical knowledge discovery

Thesis submitted by

Daniel Homola

For the degree of Doctor of Philosophy of Imperial College London
and the Diploma of Imperial College London

Supervisors: Professor Elaine Holmes,
Professor Jeremy Nicholson,
Professor Yike Guo

Division of Computational and Systems Medicine
Department of Surgery and Cancer
Imperial College London
2017

Abstract

In recent decades, the rise of various omics fields has flooded life sciences with unprecedented amounts of high-throughput data, which have transformed the way biomedical research is conducted. This trend will only intensify in the coming decades, as the cost of data acquisition will continue to decrease. Therefore, there is a pressing need to find novel ways to turn this ocean of raw data into waves of information and finally distil those into drops of translational medical knowledge. This is particularly challenging because of the incredible richness of these datasets, the humbling complexity of biological systems and the growing abundance of clinical metadata, which makes the integration of disparate data sources even more difficult.

Data integration has proven to be a promising avenue for knowledge discovery in biomedical research. Multi-omics studies allow us to examine a biological problem through different lenses using more than one analytical platform. These studies not only present tremendous opportunities for the deep and systematic understanding of health and disease, but they also pose new statistical and computational challenges. The work presented in this thesis aims to alleviate this problem with a novel pipeline for omics data integration.

Modern omics datasets are extremely feature rich and in multi-omics studies this complexity is compounded by a second or even third dataset. However, many of these features might be completely irrelevant to the studied biological problem or redundant in the context of others. Therefore, in this thesis, clinical metadata driven feature selection is proposed as a viable option for narrowing down the focus of analyses in biomedical research.

Our visual cortex has been fine-tuned through millions of years to become an outstanding pattern recognition machine. To leverage this incredible resource of the human brain, we need to develop advanced visualisation software that enables researchers to explore these vast biological datasets through illuminating charts and interactivity. Accordingly, a substantial portion of this PhD was dedicated to implementing truly novel visualisation methods for multi-omics studies.

Statement of Originality

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Acknowledgements

I would like to thank Professor Holmes and Professor Nicholson for offering me a place on the STRATiGRAD programme. I am very grateful to have joined the vibrant research community of the Computational System Medicine group. Meeting all the great minds here and working with the fantastic PhD students on this programme shaped me both intellectually and as a person in great and many ways.

I would like to thank my fiancé Andi for her patience and the limitless emotional support she has given me throughout the PhD, especially during the most stressful hours of the write-up period. Finishing my PhD while working full-time has proven to be more challenging than I had ever anticipated. Yet you managed to convince me repeatedly that I can power through and cross the finish line. I could not have done it without you, therefore like so many things in our lives, we share this achievement too. Thank you!

I also want to thank my wonderful parents for raising me to be curious and hard-working. Whatever goals or success I may reach in this life, ultimately, I owe them to you for leading by example. I could have easily chosen a more straightforward career path, but you have never questioned my choices. Instead, you supported me throughout the many years of my education, for which I will be forever grateful. I feel immensely proud and lucky to have you as my mother and father.

Personally but also in the name of the biomedical research community, I would like to express my deepest gratitude to the Wellcome Trust for funding not just my research but thousands of other PhDs and post-docs. The impact of your organisation is truly incredible and it is probably already beyond what entire nations have achieved in science. I will try hard throughout my professional life to spread the word about your amazing work and raise awareness about the need for private support of research.

Finally, a substantial part of this thesis was written on my daily commute in the morning and evening rush hours on the Northern line. Therefore, I would like to express my gratitude to TFL for running an excellent service and Alstom for designing carriages where one can sit in *relative* comfort while typing on their laptop.

Contents

Abstract **i**

Statement of Originality **ii**

Copyright Declaration **iii**

Acknowledgements **iv**

List of Figures **ix**

List of Tables **xi**

Abbreviations **xii**

1 Introduction **1**

1.1 Data driven life science 1

1.2 Omics fields with metabolism in focus 5

1.3 Overview of multi-omics data integration 9

1.4 The need for meta-tools 16

1.5 Challenges addressed by CorrMapper 18

1.5.1 The rise of the multi-omics study design 18

1.5.2 Clinical and other types of metadata 19

1.5.3 Large p small n datasets require feature selection 20

1.5.4 The potential of advanced interactive visualisation 22

1.5.5 Predictive and exploratory data analysis 23

1.5.6 Democratising advanced research tools 25

1.5.7 A meta-tool for data integration and visualisation 26

2 Datasets **28**

2.1 Bariatric surgery dataset 28

2.1.1 Data preprocessing of ^{1}H NMR data 30

2.1.2 Data preprocessing of 16S rRNA data 38

2.2 Breast cancer dataset 40

3 Methods **42**

3.1 Feature selection 42

3.1.1 Boruta 47

3.1.2 Joint Mutual Information 57

3.2 Further feature selection methods 62

3.2.1 Univariate methods 63

3.2.2 Recursive Feature Elimination 63

3.2.3 L1 norm based methods 65

3.2.4 Stability Selection 66

3.3 Feature selection benchmarking 67

3.3.1 Simulated datasets 67

3.3.2 Evaluation process 68

3.4 Graphical models 69

3.4.1 Directed and undirected graphical models 69

3.4.2 Marginal correlation networks 73

3.4.3 Partial correlation networks 76

3.4.4 Conditional independence networks 81

3.4.5 Estimating statistical relevance 87

3.5 Software development 91

4 ScienceFlask **92**

4.1 Overview 92

4.1.1 Why templates are useful? 93

4.2 Front-end architecture 94

4.3 Implemented components 97

4.4 Example application 101

4.5 Deployment notes and project documentation 102

4.6 Summary 103

5 CorrMapper - data integration **105**

5.1 Overview 105

5.2 Feature selection benchmarking 106

5.2.1 Variance filtering 111

5.3 CorrMapper's front-end 113

5.3.1 Index page and registration 113

5.3.2 Upload page 114

5.3.3 Profile page 117

5.3.4 Analysis page 120

5.4 CorrMapper's back-end 123

5.4.1 Checking uploaded files 123

5.4.2 Feature selection and graph estimation 126

5.4.3 Module finding 129

5.4.4 Preparing files for visualisation and download 130

5.4.5 CorrMapper with simulated data 131

5.5 Summary 135

6 CorrMapper - interactive data visualisation 141

6.1 Overview 141

6.1.1 Limitations of CorrMapper's visualisation modules 142

6.2 Metadata explorer 144

6.2.1 Components of the dashboard 145

6.2.2 Cross-filtering 149

6.2.3 Responsive layout 152

6.2.4 Metadata explorer in use 153

6.3 General network explorer 156

6.3.1 Components of the general network explorer 157

6.3.2 Module selection with interlinked components 163

6.3.3 General network explorer in use 166

6.4 Genomic network explorer 172

6.4.1 Components of the genomic network explorer 172

6.4.2 Interlinked components 178

6.4.3 Genomic network explorer in use 181

6.5 Summary 183

7 Conclusions 187

7.1 Future work 190

Supplementary materials 192

References 203

List of Figures

1.1 Number of publications in the omics era 3

1.2 Hierarchy of biological information 7

1.3 IDEs are great examples of meta-tools 17

2.1 Venn-diagrams of samples in the bariatric surgery study 29

2.2 Flowchart of NMR fitting pipeline. 32

2.3 Using the 2nd derivative for NMR peak fitting 33

2.4 Finding well-aligned and relevant peaks in NMR spectra 34

2.5 Examples of fitted peaks in NMR 37

2.6 Distribution of 16S rRNA copy numbers in bacterial species 39

3.1 Example of a decision tree 49

3.2 Bias variance trade-off 51

3.3 Evaluation metrics in FS benchmarking 68

3.4 Bayesian network example 71

3.5 Comparison of Pearson and Spearman correlation 74

3.6 Marginal correlation networks are sensitive to ϵ 75

3.7 Visual illustration of partial correlation 77

3.8 Network estimation in three different ways 78

3.9 Comparison of four methods for choosing λ 86

4.1 Architecture of ScienceFlask 94

4.2 Model View Controller web-service pattern 95

4.3 Profile page of ScienceFlask 101

5.1 Benchmarking results of feature selection methods 107
5.2 Benchmarking results of FS methods after variance filtering 111
5.3 Upload form of CorrMapper 116
5.4 Profile page of the guest account 118
5.5 Analysis form of CorrMapper 121
5.6 Flowchart of CorrMapper's front-end and back-end 124
5.7 Flowchart of CorrMapper's data integration pipeline 128

6.1 Components of the metadata explorer 148
6.2 Cross-filtering in the metadata explorer 151
6.3 Mapping oestrogen receptor status onto gene expression 155
6.4 General network explorer 160
6.5 Ring graph of the general network explorer 162
6.6 Selecting user defined modules in the general network explorer 165
6.7 Mapping time-point of samples onto 16S rRNA data 167
6.8 Boruta identifies relevant metabolites and OTUs 169
6.9 Interactions between urinary metabolome and gut microbiome 171
6.10 Genomic network explorer 173
6.11 Navigating the tables of genomic network explorer 179
6.12 Correlation between genomic amplification and overexpression 182

S1 Baseline correction example in NMR data 192
S2 Further examples of fitted peaks in NMR 193
S3 Single vs bagged decision trees 195
S4 Benchmarking results of seven FS methods, part 1. 198
S5 Benchmarking results of seven FS methods, part 2. 199
S6 Benchmarking results of seven FS methods, part 3. 200
S7 O-PLS coefficient plots of bariatric surgery NMR data 202

List of Tables

1.1 Feature and sample numbers in high-throughput datasets 21

2.1 Sample numbers in the bariatric surgery study 30

2.2 Statistics of NMR peak-fitting 38

3.1 Example dataset demonstrating feature interaction 60

3.2 Overview of the multiple correction problem 90

5.1 Summary of feature selection benchmarking results 108

5.2 Study types CorrMapper can analyse 115

5.3 Performance of CorrMapper on $R \geq 0.2$ simulated datasets 132

5.4 Comparing CorrMapper against other network estimators 133

6.1 Description of the breast cancer study's annotation fields 177

S1 Number of benchmarking experiments per ratio and algorithm 197

S2 Performance of CorrMapper on simulated datasets 201

S3 Comparing CorrMapper against other network estimators 201

S4 Differentially expressed OTUs in RYGB patients after operation . . . 202

Abbreviations

3-HBA	3-Hydroxybutyric acid
ACSN	Atlas of Cancer Signalling Network
AIC	Akaike Information Criterion
API	Application Programming Interface
BIC	Bayesian Information Criterion
BMI	Body Mass Index
CCA	Canonical Correlation Analysis
CDF	Cumulative Distribution Function
CLR	Centered Log Ratio
CNV	Copy Number Variation
CSS	Cascading Style Sheets
DAG	Directed Acyclic Graph
EBI	European Bioinformatics Institute
EMR	Electronic Medical Record
FDR	False Discovery Rate
FS	Feature Selection
FWER	Family-Wise Error Rate
GPD	Generalised Pareto Distribution
JMI	Joint Mutual Information
LLO	Leave-One-Out
LOH	Loss Of Heterozygosity
LSVC	Linear Support Vector Classifier
MbP	Megabase Pair
MCD	Multiple Concerted Disruption
MCIA	Multiple Co-Inertia Analysis

MCMC	Markov Chain Monte Carlo
MI	Mutual Information
MIFS	Mutual Information based Feature Selection
MIM	Mutual Information Maximisation
MRMR	Minimum Redundancy Maximum Relevance
MS	Mass Spectrometry
MSE	Mean Squared Error
MVC	Model View Controller
NCBI	National Center for Biotechnology Information
NMR	Nucleic Magnetic Resonance Spectroscopy
O-PLS	Orthogonal Projections to Latent Structures
OOB	Out Of Bag
OTU	Operational Taxonomic Unit
PAG	Phenylacetylglycine
PCA	Principal Coordinate Analysis
PLS	Partial Least Squares Regression
PLS-DA	PLS-Discriminant Analysis
RF	Random Forest
RFE	Recursive Feature Elimination
RFE-CV	Recursive Feature Elimination with Cross-Validation
RSWPA	Recursive Segment-Wise Peak Alignment
RYGB	Roux-en-Y Gastric Bypass
SF	ScienceFlask
SG	Savitzky-Golay
sGCCA	sparse Generalized Canonical Correlation Analysis
SL	Scikit-Learn
sMBPLS	sparse Multi-Block Partial Least Squares
SNF	Similarity Network Fusion
SS	Stability Selection
StARS	Stability Approach to Regularization
SVM	Support Vector Machine
TMA	Trimethylamine
TMAO	Trimethylamine N-oxide
VI	Variable Importance

1 Introduction

1.1 Data driven life science

> We are drowning in information but starved for knowledge.
>
> John Naisbitt

Life sciences and biological research are undergoing dramatic changes at the moment. This transformation is mainly driven by the exponentially growing high-throughput data coming from the different omics fields of biology such as genomics, proteomics, metagenomics and metabonomics. The ever-increasing growth in data volumes can be largely explained by the constant improvement in the efficiency of data acquisition technologies. This continued innovation and refinement has reduced the cost of applying these new analytical methods drastically, which in turn increased their wide-spread utilisation in the research community.

An example: the sequencing and assembly of the first human genome cost hundreds of millions of dollars, \$3 billion by the US government funded project[1] and \$300,000,000 by the private Celera initiative[2], while it took 11 and 3 years respectively. Yet less than 13 years later in 2014, we passed the \$1000 price point in human genome sequencing[3], with the Illumina HiSeq X Ten, which is capable of sequencing 18,000 human genomes per year, to the gold standard of 30× coverage[4].

Remarkably, the cost of sequencing is falling quicker than the cost of computation[5] and the prices of other analytical platforms are decreasing sharply as well[6]. Today,

the European Bioinformatics Institute (EBI) is storing and managing well over 100 Petabytes of biological data (this is an extrapolated figure, the true value was already 75 Petabytes in December 2015[7]). These data have been growing exponentially for the past decades, almost doubling in its size every year[8], while it not only has to be stored securely and managed properly, but also kept widely accessible, as researchers from around the world query this fundamental open resource 12 million times each month.

However, this amount of data is by no means unheard of these days; Facebook was reported to store 300 Petabytes of data in 2014[9] and physicists at CERN passed the 100 Petabyte mark in the same year[10]. All of these statistics are dwarfed however in comparison with Youtube's data load: 400 hours of videos are uploaded to its servers every single minute[11]. Biological data however, are much more heterogeneous than those in physics or technology. Life sciences collect data from hundreds of species across dozens of conditions and diseases, while the number of measurable variables can be orders of magnitude higher in some omics fields (e.g. metagenomics) than in physics.

Furthermore, biological data is often hopelessly complex, as it originates from organisms which were shaped by evolution's relentless trial and error process over millions of years, where new functions can never be developed through rational planning and design but only by building on and patching the ancestors' existing solutions. Lastly, the source and consequently the nature of biological data is really varied as well, ranging from small molecule profiles of metabonomics, through bacterial communities of metagenomics to sequences of genomics and structures of proteomics[12].

The trend of rapidly emerging omics fields is reflected in the exploding number of publications associated with these fields on PubMed, see Figure 1.1. Interestingly, "Data integration" and "Personalized medicine" seem to have grown in parallel with

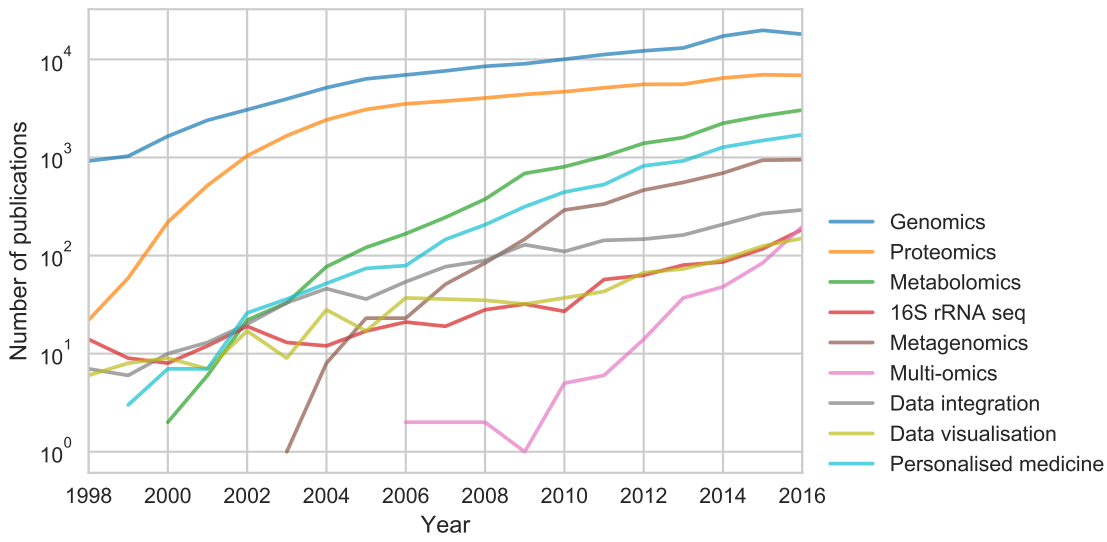


Figure 1.1: The advent of the omics era: number of publications mentioning the various omics fields have been growing exponentially on PubMed since the late 1990s.

these fields, suggesting an early need for the integration of various data sources and the extraction of translational medical knowledge.

Furthermore, with the dropping cost of omics technologies, the term "Multi-omics" has shown a strong upward trend in recent years too. As we will discuss later, this new study design represents a highly promising avenue for research, where multiple omics technologies are used simultaneously to gather information from the same patients or organisms. This approach has great potential for filling the gaps in our biomedical knowledge, as each of the analytical platforms offers a new angle to the studied problem, which can facilitate the discovery of novel connections between different compartments of living systems.

For example, in a multi-omics study, we might be interested in how the urinary metabolic profile and gut bacterial composition change in patients who have been prescribed a serious regimen of antibiotics. In this case, the urinary metabolites of the patients can be screened by one of the established metabonomics methods[6], while the gut microbiome would be profiled with 16S rRNA sequencing[13]. These data sources would be highly valuable on their own, but when combined, they could

help to discover completely novel interactions between the shifting microbial composition and the host's metabolism, and therein lies the real power of the multi-omics approach.

Despite these novel omics technologies and the truly unprecedented flood of biological data however, frustratingly, the majority of preclinical studies are irreproducible and by some estimates, 85% of all biomedical research might be wasted[14,15]. Nobel laureate Sydney Brenner cynically referred to this kind of top-down, data (rather than hypothesis) driven omics research as: "low input, high throughput, no output science"[16]. In light of these inefficiencies in biomedical research, delivering on the promises of systems biology might prove to be painfully difficult. Therefore, taking full advantage of the omics revolution, by turning raw biological data into translational medical knowledge hinges on several factors:

- Sufficient statistical power: this can be ensured by feature selection (based on a pilot study) or securing funds to acquire the necessary number of samples.
- Data warehousing and standardisation efforts that aim to maximise the cross-utilisation of data resources between projects.
- Transparency and reproducibility: utilisation of open-source tools, publication of raw data and sharing of notebooks containing all source-code of the analysis.
- Novel computational and statistical tools that are designed to work with noisy biological data.
- New visualisation methods that take advantage of the interactivity provided by modern web technologies.
- Data integration tools that enable researchers to take full advantage of their multi-omics studies by connecting the disparate data sources (including clinical metadata) in a meaningful way.

As we will see later, the work presented in this thesis makes small but hopefully valuable contributions to the last four points of this list.

1.2 Omics fields with metabolism in focus

The majority of research and development presented in this thesis was accomplished using a multi-omics dataset which combines metabonomics with 16S rRNA sequencing. This study recruited 97 patients who underwent various types of bariatric surgery to induce weight loss in them and consequently improve their health. To complement the omics datasets, extensive clinical metadata was also collected from these patients. This study is introduced in greater detail in Chapter 2. Therefore, here we will focus on the brief introduction of the two aforementioned omics methodologies.

Metabolomics aims to measure the concentrations and locations of all metabolites in a cell. Here, metabolites are defined as small molecules which are lighter than 1 kDa, i.e. substrates and products of enzymes. Metabonomics is a subset of metabolomics which is defined as the quantitative measurement of the metabolic responses of living systems to environmental changes, genetic modification, patho-pscychological stimuli or drugs[17]. Metabonomics is a well-established top–down systems biology approach, which places strong emphasis on the multivariate characterisation of population level differences in metabolic profiles across different physiological states[18].

It achieves this by collecting samples from biological fluids (urine, plasma, serum or faecal water), which are then analysed with high-throughput analytical chemistry technologies such as Nucleic Magnetic Resonance Spectroscopy (NMR) spectroscopy and/or Mass Spectrometry (MS)[19]. The resulting omics dataset contains the metabolites with their relative or absolute concentrations that were found in a patient's biofluid at the time of screening. Therefore, metabonomics captures a rich and colourful metabolic snapshot of the sampled organism with great potential for biomarker discovery[20].

With the cost of sequencing dropping, metataxonomics[21] emerged in the early 2000s as a culture independent way of doing microbiological taxonomy and ecology[22]. Trying to culture the hundreds of bacterial strains found in a stool or soil sample has proven to be impossible, because firstly most of these species are extremely sensitive to their environment (temperature, pH, concentrations of chemical elements), and secondly most strains have been coevolving for millions of years during which they have formed intricate inter-species metabolic pathways whereby they rely on each other's intermediary metabolites. Therefore, culturing these fragile ecosystems in the lab is extremely challenging.

Metataxonomics unties this Gordian knot by sequencing all 16S rRNA genes that could be found in a sample. By aligning and clustering the reads, one can identify Operational Taxonomic Units (OTUs), because the non-conserved variable regions of the 16S rRNA gene provide enough evolutionary information to precisely identify bacterial strains taxonomically. The omics dataset arising from this platform holds the count of each OTU for every patient in tabular form, acting as a taxonomic inventory of the sampled bacterial ecosystem. Since the bacterial composition shows strong inter-individual variation, many cells of this table are usually zero as rare species which only occur in one or two patients will be completely missing from others.

Both metabonomics and 16S rRNA sequencing use sophisticated data preprocessing pipelines and advanced mathematical modelling to transform the raw omics data into biological information. For this, dimensional reduction techniques such as Principal Coordinate Analysis (PCA) and multivariate statistical models like Orthogonal Projections to Latent Structures (O-PLS)[23] and PERMANOVA[24] are routinely used in these fields.

As shown in Figure 1.2, biological information is expressed in a remarkable hierarchical fashion. The inherent complexity of biology not only stems from the inter-

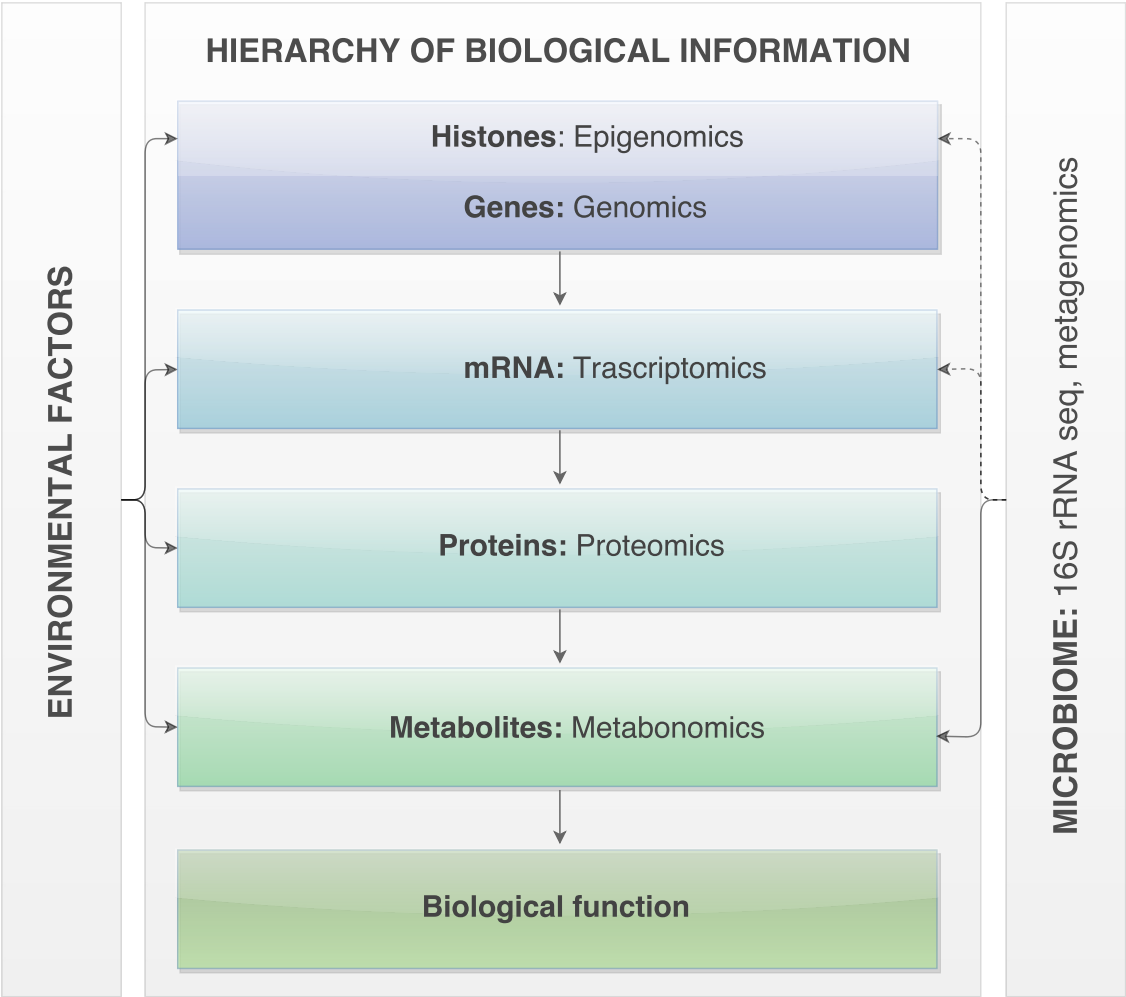


Figure 1.2: Hierarchy of biological information and its interaction with the environment and the microbiome. Solid arrows represent proven flow of information and/or interaction, while dashed arrows show hypothesised connections between the microbiome and certain compartments of the human body.

connectedness of these layers but also from the intricate and temporally changing interplay between us and our environment (shown as solid arrows in Figure 1.2).

Additionally, one of the greatest discoveries of the past decade was that the human microbiome, defined as the aggregate of micro-organisms (mainly bacteria) that live on and within our bodies, is just as important as the environment in modulating our biology. The gut microbiome became a particularly deeply researched subject, and in the last ten years, through a tremendous amount of research, it was proven to play a crucial role both in our health and diseases[25].

The time-scales at which the different biological information layers change shorten dramatically as we move from genome through proteome to metabolome. This is true both on the population and individual level. Although our genome is evolving quicker than it was previously believed[26], it still takes hundreds or thousands of years for new variants to appear and spread in a population. Our transcriptome and consequently proteome changes much more rapidly as we develop in the womb and also through life, but this can still be measured in years.

These time-scales are vastly longer than the swift changes observed in our metabolome[27] and microbiome[28]. They are influenced by our diet, lifestyle, a trip to another country and the drugs we take. Therefore, they capture a breathtaking amount of information about our current physiological state and well being. This makes these omics layers invaluable not just for biomedical research but also for clinical diagnostics and patient stratification.

Furthermore, it has been shown, that with the right study design and analytical techniques, a substantial amount of the variation found in metabonomics studies, could be attributed to individuals as opposed to the environment[29]. As the fields of 16S rRNA sequencing and metabonomics have evolved, the same conclusion was reached, i.e. inter-individual variability is much larger than intra-individual variability over time[30].

Finally, there is mounting evidence which shows that the gut microbiome is not only influenced by the host's metabolism but reciprocally it also modulates the metabolic pathways of the host in profound ways[31–33]. This two way communication was implicated in the progression of serious diseases such as coronary heart disease[34] and colorectal cancer[35]. Therefore, the combination of metabonomics and 16S rRNA sequencing represents a very promising mode of investigation for deepening our understanding and discovering clinically relevant, medically actionable biomarkers.

1.3 Overview of multi-omics data integration

The wealth of information generated by high-throughput omics technologies represent an unprecedented opportunity to assess molecular interactions at multiple functional levels and gain a better understanding of the biological pathways involved in different diseases. The continued decrease in the price of omics technologies has not only resulted in exponentially growing data volumes, but also gave rise to a multi-omics studies, see Figure 1.1.

Multi-omics studies represent a promising mode of investigation in life sciences as they attempt to capture a comprehensive snapshot of living systems by the collection of complimentary biological data from the same patients or organisms.

Therefore, each analytical platform provides a different viewpoint into the underpinnings of the studied disease or biological process. The aim of multi-omics projects is to integrate these disparate data sources originating from different omics levels and compartments of a biological organism, and uncover relevant interactions between their features.

These multi-level connections are crucial for identifying multi-omics signatures which can be predictive of a biological condition or provide novel mechanistic insights into our health and diseases. Furthermore, by the definition of multi-omics studies, their findings are often missed when we focus on a single omics type.

For example, one might investigate the gene expression of cancer patients while also measuring the Copy Number Variation (CNV) of their genomes. In this scenario, we would expect that the deletion or duplication of certain genomic regions result in an altered gene expression profile that can help with patient stratification and provide us with a better understanding of the disease such as breast cancer[36].

Alternatively, one might be interested in mapping out cross-species metabolic interactions that arise from the complex interplay between humans and their microbes[31]. 16S rRNA sequencing can capture the gut-microbial composition of patients, by quantitatively measuring the abundance of bacterial strains inhabiting their gastrointestinal tract. These data can be correlated with the metabolic urinary profiles of the same individuals, which can be obtained using metabonomics methodologies, such as NMR or GC-MS[6].

However, the integration of multi-omics data is highly challenging due to the heterogeneity of the various analytical platforms. As each omics data type captures a separate aspect of biology, they all have different dimensions, noise profiles and often modality: sequences, counts, floats, images, spectral information.

A quick glance at the literature of multi-omics data integration makes it clear, that there is a clear need to simultaneously explore the complex and correlated interplay between different data-types, and numerous avenues have been successfully explored to achieve this.

Bersanelli et al. recently reviewed 23 data integration tools[37] and divided them into four categories based on their mathematical formulation:

- network-free non-Bayesian (NF-NBY) - 8 methods,
- network-free Bayesian (NF-BY) - 5 methods,
- network-based non-Bayesian (NB-NBY) - 8 methods,
- network-based Bayesian (NB-BY) - 2 methods.

Here, the term “Bayesian” refers to methods that rely on prior distributions of the data, often originating from previous findings in the literature. These priors are updated using Bayes’ theorem based on the observed samples to obtain their posterior probability distribution. Network based methods estimate the relationship between omics features, which they map out and analyse as a graph.

Furthermore, multi-omics integration tools can also be categorised by their purpose:

- Uncover novel molecular interactions between different omics data types to expand our current knowledge about a certain disease or biological process.
- Discover clusters of patients or samples that are more similar to each other than the rest of the cohort. This strategy can lead to improved patient stratification and disease subtype detection.
- Build a predictive model that can for example classify new patients based on their omics data into "healthy" and "sick" phenotypes.

Bersanelli et al. found that the overwhelming majority of multi-omics integration methods aim to achieve one of the first two goals, while they most often specialise in two omics data types; DNA sequence data, gene and protein expression being the most prevalent ones. More than half of the reviewed tools are available as R or Matlab projects, but only two of them can be accessed through a webserver, which can limit their utilisation by clinicians and researchers without programming knowledge.

It is beyond the scope of this thesis to provide a thorough review of the dozens of published data integration methods. Therefore, the following pages simply provide a brief overview of some interesting, recently published multi-omics integration tools and their application. Section 5.5 will provide a more detailed comparison between CorrMapper (a data integration method developed as part of this PhD) and its closest "competitor", the mixOmics R package[38].

Charj et al. used Multiple Concerted Disruption (MCD) analysis of CNV, Loss Of Heterozygosity (LOH) and DNA methylation status to interpret changes in gene expression. They combined these three data types sequentially to explain the observed gene expression changes between breast cancer and non-cancer cell lines. Relying on our current model of genetics, they deemed a gene expression level as

"explained", if over-expression was coupled with gain in CNV, allelic imbalance and hypomethylation. Conversely, under-expression was expected to co-occur with a loss in CNV, LOH and hypermethylation. Their multi-omics approach has led to the identification of a higher number of explained gene expression changes, compared with standard single omics analysis[39].

Meng et al. developed Multiple Co-Inertia Analysis (MCIA), an exploratory data analysis tool which identifies relationships between high dimensional omics datasets. Based on a covariance optimization criterion, MCIA simultaneously performs the ordination of multiple omics data matrices within the same low-dimensional hyper-space, where features or samples exhibiting similar trends are projected closely to each other. This analysis can lead to the identification of biomarkers and clusters of samples. The authors used this method to integrate gene and protein expression of the NCI-60 cell line, and their analysis revealed pathways not uncovered by single-omics analyses[40].

Both MCD and MCIA are network-free non-Bayesian methods, aimed at uncovering novel molecular interactions between different omics datasets. However, unlike MCD and the majority of omics integration tools, MCIA is not specific to a set of omics data types, therefore it can be applied to any multi-omics dataset.

Several other NF-NBY methods have been proposed for omics data integration, which rely on the Partial Least Squares Regression (PLS) or Canonical Correlation Analysis (CCA) algorithms. Integromics[41] and its later incarnation mixOmics[38] both provide regularised CCA and sparse PLS, while sparse Multi-Block Partial Least Squares (sMBPLS)[42] extends sparse PLS to three input matrices.

Given two data matrices $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{n \times q}$, with p variables x_j and q variables y_k respectively, measured on the same n samples, CCA seeks H pairs of vectors a_h

and b_h such that the following objective function is maximised:

$$\underset{a,b}{\arg\max}\ cor(Xa_h, Yb_h), \qquad s.t.\ ||a_h||_2 = 1,\ ||b_h||_2 = 1, \qquad \text{Eq. 1.1}$$

where $h = 1, 2, ..., H$. In other words, CCA uses linear combination of the columns of X and Y to create the canonical variables $U = Xa$ and $V = Yb$, whose correlation is maximised. Once the first canonical variables U_1, V_1 are obtained, the second pair is found by maximising the same objective, but with the additional constraint that U_2, V_2 should be uncorrelated with the U_1, V_1. Although CCA loadings are not directly interpretable, the correlations $cor(X, a)$ and $cor(Y, b)$ can be visualised which has been shown to help with the explanation of CCA's results[43].

PLS is very similar to CCA but it maximises the *covariance* between U and V. Due to this close connection, PLS and CCA have been shown to perform similarly when used for discrimination between classes[44]. Both CCA and PLS simultaneously decomposes X and Y into co-varying latent variables (U_h, V_h) and their associated loading vectors (a_h, b_h). These methods assume that the majority of variation within a system could be explained by a few of these n-dimensional latent variables.

Integromics and mixOmics both use regularised versions of the PLS and CCA algorithm. These extensions add an L1 penalty to the above described objective functions (see Section 3.2.3) to obtain sparse loading vectors. Regularisation not only improves the numerical stability of CCA, but it also greatly increases the interpretability of both algorithms[45]. Nonetheless, relating the latent variables learned by these models to the effect of individual features on the outcome variable remains challenging. For example, given $H = 3$, a single feature x_j can have a positive loading in a_1, a negative loading in a_2 and zero in a_3.

Network based non-bayesian methods either rely on existing molecular interaction data (e.g.: metabolic or gene regulatory pathways) or build networks by analysing

the observed correlation between omics features. SteinerNet is a webserver which integrates transcriptional, proteomic and interactome data by searching for the solution to the prize-collecting Steiner tree problem[46]. It can search the interactome of humans (and four other model organisms) to find connections between experimentally detected proteins or genes, and to identify biologically meaningful pathways.

Endeavour is another online tool which prioritises genes for a given biological outcome or disease[47]. It calculates and combines gene-wise statistics from 75 different genome-wide data sources to rank candidate genes based on their biological relevance.

Unlike most network based methods Similarity Network Fusion (SNF) clusters patients with multi-omics data by building a correlation based sample-sample network from each omics matrix. Once these omics-specific similarity matrices are estimated, they are joined using an iterative diffusion processes which converges on a global similarity matrix. This weighted adjacency matrix defines clusters of patients across multiple omics modalities[48]. The authors applied their method to gene expression and DNA methylation data, which successfully discovered clusters of patients with different cancer subtypes and survival rates.

Finally, parametric Bayesian methods assume that the observed biological system could be modelled using a certain family of distribution parametrised by θ. The inference of $\hat{\theta}$ consists of using Bayes' theorem to update our prior belief about $\hat{\theta}$ with the observed data. A well-known limitation of such methods is their sensitivity to the choice of prior. Therefore, generally Bayesian methods require a lot more input and domain-specific knowledge from the user.

Provided that we have found an adequate distribution to model our system, if our chosen prior gives a reasonable distribution for θ (based on some previously obtained biological knowledge), then our model will capture the core information of a given dataset. Conversely, if the initial guess for the prior is hard or even impossible to

formalise or obtain (e.g. no experimental or theoretical results are available about the parameter), then given the small sample sizes in current biomedical research, Bayesian methods can lead to more biased estimates than traditional frequentist inference techniques[49].

Given the numerous multi-omics integration methods, the reader might legitimately ask why is another such tool needed? The answer to this is manifold and will be explicated in the following sections exhaustively, but the list below summarises some of the key arguments for the creation of CorrMapper:

- Most integration tools specialise in particular omics data types. While this enables these methods to leverage omics specific biological knowledge, it also restricts the applicability of these data integration algorithms to a predefined set of omics data types.
- Although several data fusion methods employ feature selection to make their results easier to interpret, this is almost exclusively done by penalising linear models with the L1 norm. While this a well studied and excellent form of feature selection, it does not account for non-linear interactions and more importantly correlated features[50].
- Several of the referenced tools have some form of visualisation capabilities. However, the complexity of multi-omics datasets require advanced visualisation tools that work in tandem with data integration algorithms and facilitate the exploration process through interactivity.
- The overwhelming majority of multi-omics integration algorithms are either available as an R or Matlab package, while Bersanelli et al. found that more than 20% of the 23 examined tools have no implementation at all. While programming languages are becoming more popular amongst life scientists, there is still a large fraction of researchers who do not possess any coding skills and therefore prefer graphical user interfaces.

1.4 The need for meta-tools

The word *thesis* comes from Ancient Greek and it means "proposition, statement, a thing laid down". A PhD thesis is supposed to be many things at once: a review of the field, a description of a problem or hypothesis, a detailed summary of a long-term research project and also an attempt to amalgamate the findings of this new piece of work into the collective body of knowledge of the field. Given the original meaning of the word, one could argue that a PhD thesis should also take a position, present it, and argue for its validity.

The main proposition of this thesis is that the robust and meaningful integration of different omics data types is one of the most pressing issues of life sciences currently, that can be best addressed by the development of composite software solutions, which will be referred to hereafter as meta-tools. This section and various parts of the thesis will argue that meta-tools have an incredible potential to tackle this multi-faceted challenge, and therefore the biomedical research community should invest more heavily in their development.

A meta-tool is an ensemble of simpler tools, which when combined together as a well-designed system, are far more powerful than the sum of individual components. Integrated Developer Environments (IDEs) are great examples of this concept in software engineering. IDEs consist of fairly simple tools such as text editor, version control (VC) system, syntax highlighter, file explorer, searchable help section of the programming language, debugger and a package manager. However, when combined together into a unified graphical interface they can increase the productivity of a developer by several folds. Figure 1.3 showcases an example of a modern IDE.

In bioinformatics, EBI's InterPro[51] is another great example of meta-tools. It comprises of several workhorse algorithms of protein sequence analysis: Gene3D, PAN-

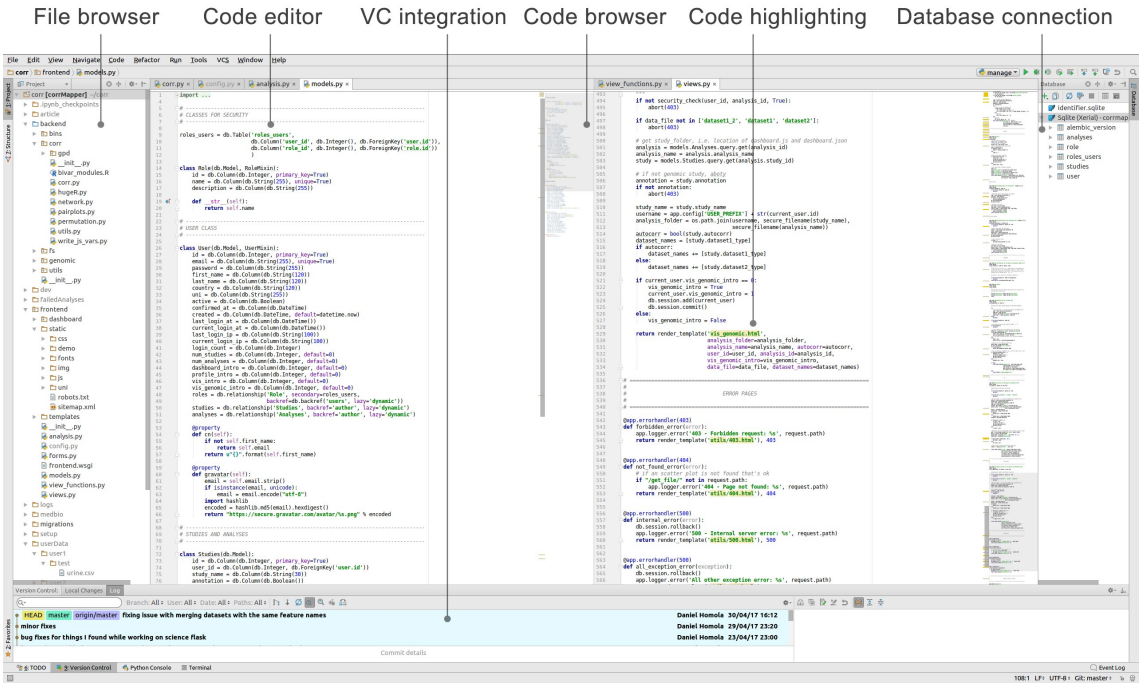


Figure 1.3: IDEs are great examples of meta-tools. PyCharm is a popular IDE for Python coders, offering all the must have features for productive development.

THER, Pfam, PIRSF, PRINTS, ProDom, PROSITE, SMART, SUPERFAMILY and TIGRFAMs. Each of these tools focuses on a different aspects of protein domain and functional site prediction and all of them have their own interface and website. However, when combined together as InterPro, they become a one-stop-shop for protein sequence analysis and an indispensable resource for life scientists.

Instead of checking dozens of online interfaces, researchers can simply input their sequence information into InterPro to utilize the above mentioned specialised tools all at once. Not even domain experts with extremely focused interests can keep up with all the results that are tangentially related to their field. Consequently, InterPro not only saves them time but can also spark new ideas by presenting complementary bits of biological information that might be new and relevant to researchers.

This thesis argues therefore, that gains in productivity and efficiency in biomedical research can not only come from completely novel algorithms and experimental techniques, but also from the ingenious combination and refinement of existing ones.

1.5 Challenges addressed by CorrMapper

> Progress in science depends on new techniques, new discoveries, and new ideas, probably in that order.
>
> Sydney Brenner[52]

The majority of this thesis is about CorrMapper, an online meta-tool for the integration and visualisation of complex multi-omics datasets. The problems that led to the inception of CorrMapper are outlined in the following sections. By enumerating these challenges it will become apparent how they tie into each other and why addressing them together with a meta-tool is a legitimate research problem.

1.5.1 The rise of the multi-omics study design

Even if we only consider the most popular omics methods (gene sequencing, SNP sequencing, RNA sequencing, transcriptomics, epigenomics, proteomics, bacterial 16S rRNA sequencing, metagenomics, targeted and untargeted metabonomics), there are 45 possible combinations one can choose to design a multi-omics study with two different data types. This number grows to 120 with three datasets.

There are numerous specialised multi-omics integration tools that concentrate on a particular pair[53,54] or a subset[55] of the aforementioned omics platforms. Often this sharp focus allows these methods to incorporate the various platform specific preprocessing steps or to adapt their methodology to the targeted omics data types specifically.

CorrMapper takes a different approach as it was designed to be a general purpose, platform agnostic omics data integration tool for any two omics datasets. To achieve this, it requires the user to perform all the omics field specific preprocessing *a priori*.

1.5.2 Clinical and other types of metadata

Healthcare systems around the world are increasingly becoming technology driven and digitised. Electronic Medical Records (EMRs) of hospital and doctor visits are routinely collected around the world not just by insurance companies in the US[56] but by government funded health agencies like the NHS as well[57]. These EMRs include the exact medical procedures that were carried out along with the resulting diagnosis, drug prescriptions and extensive phenotypic patient data as well. These vast population wide data sources allow researchers to reconstruct the full history of millions of patients with unprecedented precision, tracking key phenotypic traits longitudinally over several years or even decades.

The previously mentioned $1000 genome and EMR data will both pave the way to a new era of healthcare, that is often referred to as precision medicine[58]. If this becomes a reality (and numerous signs indicate it will), each patient will receive healthcare service that is tailored individually to their genetics, lifestyle and environment. For example, in the not very distant future our genes will dictate the kind of cancer treatment we receive[59], while the mutations that lead to altered drug efficiency will also be taken into account when designing the dosing regimen.

Furthermore, our health is now routinely being monitored outside of the hospital and doctor's office too. The wearable gadget industry is growing at an impressive rate and is expected to sell 900 million smart watches, bracelets and other monitoring devices by 2021[60]. This new and increasingly cheap technology is able to record essential health and activity metrics such as heart rate, body temperature and the number of steps walked, but it can also estimate more complex measures such as sleeping patterns, stress levels and calories burnt throughout the day.

Although the quality of this data is not yet comparable with the information collected by medical professionals, its volume and granularity makes these data sources

very promising. Millions of users track their health and vitality levels on a daily basis already which has helped to create a plethora of fitness-gadget startups and smartphone applications.

As with omics technologies however, these new data (whether it is coming from hospitals or wearable gadgets) not only present unprecedented opportunities for researchers but also very serious challenges. These phenotypic data can only complement and augment the omics datasets if they are mapped onto each other in an intelligent and sensible way, allowing for easy patient stratification and exploratory analysis.

CorrMapper provides a novel visualisation module that was designed to do exactly this. As we will see later, this interface helps researchers to maximise the potential of their clinical metadata, working in tandem with their omics datasets, to identify useful rules and strategies for patient stratification.

1.5.3 Large p small n datasets require feature selection

The human genome contains about 20,000 protein coding genes, while the collective genome of our gut microbiome may have a hundred times more than that[61]. Untargeted metabonomics studies can easily generate thousands of candidate features and the diverse bacterial communities inhabiting our gastrointestinal tract comprise of many hundred strains or OTUs.

Modern omics datasets are extremely feature rich in general, see Table 1.1. This presents tremendous opportunities for biomarker discovery but it is also a burden for statistical analysis. If the number of features p is vastly larger than the number of samples n, even the simplest models such as linear regression are too flexible and will lead to over-fitting. This means that our model will perform poorly on unseen samples, as it cannot generalise well from the training data[62].

	Typical feature number (p)	Samples (n)	Ratio $\frac{p}{n}$
SNP genotyping	$5 \times 10^5 - 2 \times 10^6$ probes	$10 - 500$	~392
Metagenomics	$2 \times 10^5 - 2 \times 10^6$ genes	$10 - 300$	~645
Transcriptomics	$20,000 - 40,000$ probes	$10 - 200$	~285
Genomics	$20,000$ human genes	$10 - 500$	~78
Metabonomics	$1000 - 4000$ peaks	$10 - 300$	~16
Metataxonomics	$1000 - 2000$ OTUs	$10 - 300$	~10

Table 1.1: High-throughput biological datasets are $p > n$ and often $p \gg n$.

Interestingly, most high-throughput studies are still underpowered due the price of omics technologies. Fortunately, as alluded to earlier, the price of omics technologies is showing a rapidly decreasing trend, but it might take another decade till we routinely see studies with tens of thousands of patients. Until then, depending on the omics field, we have to cope with 10-600 times more features than samples on average. This poses serious challenges and might also account for some of the reproducibility crisis biomedical research is experiencing right now[14].

Furthermore, since data analysis tends to be highly iterative in life sciences, working with such feature rich datasets will repeatedly incur a greater computational cost and therefore take noticeably longer. Finally, the interpretation of unregularised, complex models arising from such high-dimensional datasets, is impossibly difficult even for experts with relevant domain-specific knowledge.

Fortunately, however, many of the features collected by high-throughput omics platforms might be completely irrelevant to the studied biological problem, or redundant in the context of others (multicollinearity). For example, we would not expect to find all 20,000 human genes to have a direct influence on the progression of breast cancer. Neither would we anticipate to find every bacterial strain in our gut to be involved in irritable bowel syndrome.

Therefore, feature selection or prioritisation should be an integral part of any bioinformatics pipeline that is designed to work with high-throughput omics data. This

is especially true for multi-omics studies, where the large p small n problem is further exacerbated by using multiple analytical platforms, therefore collecting more features for the same number of patients.

Consequently, great attention was paid to the Feature Selection (FS) problem while designing CorrMapper and several state of the art methods were benchmarked to provide an optimal solution for the users.

1.5.4 The potential of advanced interactive visualisation

The reason why data visualisation is so important in data analysis is that a good plot can surprise us. By displaying characteristics of our data that we did not anticipate, graphs can provide us with immediate insight and spark new ideas. A single histogram or scatter plot can turn the course of exploration into a completely new direction or form the basis of a whole set of new experiments.

Although the cost of sequencing has been decreasing much quicker than the cost of computing[5], Moore's law has been holding since the early 70s and we often forget the incredible compounded advancement that has happened in computing. For example, our smart phones today are several orders of magnitude faster than the super computers of NASA that took the crew of Apollo 11 to the Moon[63].

As a consequence, we can run several complex applications simultaneously on our phones and tablets, play realistic looking games or watch a movie in high definition, enjoying the benefits of sophisticated real time decoding algorithms. If we can do all of this on a mid-priced smart phone, it is not surprising that modern laptops and desktop computers can be used to do almost anything: from composing film score with thousands of sampled musical instruments, through training complex machine learning algorithms to simulating three dimensional worlds with breathtaking reality.

Among many other things, this cheap computational power has revolutionised data

visualisation as well and has given rise to new software libraries with amazing capabilities. JavaScript packages such as d3.js[64] and sigma.js[65] can run in any web-browser and render complex vector graphics and networks with thousands of nodes fluidly while providing stunning interactivity and a smooth user experience. Some interesting example visualisations built with d3.js can be found here[66].

WebGL libraries such as three.js[67] take advantage of the dedicated graphical processor units of modern computers and consequently represent the cutting edge of what is achievable in terms of web-browser based advanced data visualisation. Some truly stunning examples of this technology are available here[68].

As we will see later, CorrMapper not only takes advantage of these modern software libraries but fully embraces the interactivity and complex visualisations they provide. Although certain sections of the scientific community dismiss data visualisation as a gimmick that lacks true rigour and scientific depth, hopefully by the end of this thesis, the reader will be convinced (if needed to be) that advanced data visualisations can be highly illuminating and useful in generating insight and therefore leading us to novel discoveries.

Finally, Section 6.5 enumerates several currently available visualisation tools that are tailored for omics data, and compares them to CorrMapper's solutions.

1.5.5 Predictive and exploratory data analysis

The adaptation of machine learning methods in bioinformatics has made it possible to build highly accurate predictive models which take into account the non-linearities and complex high-dimensional nature of omics datasets. However, this is rarely the final goal in biomedical research projects. Instead, basic research in life sciences is primarily interested in the mechanistic understanding of diseases and the accumulation of knowledge that can form the basis of new therapies and diagnostic tools.

Nonetheless, predicting patient outcome might be of interest in some scenarios. For instance, when we have real world evidence data, we can use the prescription and diagnostic history of millions of patients to build a predictive model that can detect a certain disease earlier than current best practices. Such a model is immensely valuable as it might improve patient outcome and also save money to the healthcare system by reducing the risk of comorbidities. Such a model however has to be trained on a very large sample to ensure statistical power and to minimise the chance of false negative diagnoses, which could have devastating consequences in real world application. Although the price of omics technologies has been decreasing steadily for decades, we are still not able to fund omics studies with thousands of patients, therefore currently this sort of predictive modelling is mainly done on EMR data in private companies.

Furthermore, a powerful enough predictive model will most likely not provide us with new knowledge about the fundamentals of a disease and therefore will not improve our capacity to target it in novel and more efficient ways. This is because generally speaking there is an inverse correlation between a model's performance and its interpretability in machine learning (see chapter 2.1.3 in James et al.[69]). Consequently, we know very little about the inner mechanics of models that perform astonishingly well, such as deep neural networks, gradient boosted trees or ensembles methods like random forests.

This frustrating phenomenon is one of the reasons machine learning practitioners often refer to these highly powerful and complex algorithms as "black-box" models. Several methods have been proposed[70,71] to alleviate this problem, but it is still very much an actively researched area of machine learning. Consequently, even if predictive modelling is part of a biomedical research project, it almost always has to be preceded with studies that gather and build knowledge regarding a particular condition or biological phenomenon.

Therefore, researchers often need to do exploratory analyses, dissecting their dataset in different and many ways. A frequently used key step in this process is the assessment of countless one-to-one associations between the various biological features. These links could be defined in numerous ways (correlation, mutual information, clustering) but ultimately one hopes to separate the noise from signal and find meaningful relationships that could be understood in a wider biological context of the studied disease or process. Due to the very high number of features however, these association networks are often extremely difficult to interpret as they are very hard to interact with or explore.

To remedy this unfortunate situation, CorrMapper was designed to be a data exploration and hypothesis generation tool that can spark new ideas and shine light on undiscovered novel associations in complex omics datasets.

1.5.6 Democratising advanced research tools

Although a number of advanced machine learning and statistical algorithms are available to solve the challenges outlined above, their highly technical nature prevents them from wide utilization in life sciences. Similarly, due to the abundance of free graphical plotting libraries, the creation of advanced interactive data visualisations has become easier in recent years, but these powerful libraries all come with steep learning curves that can deter many researchers without a background in programming.

Bioinformaticians tirelessly work to bridge these gaps by applying modern machine learning and advanced data visualisation to biological data but the output of such efforts is often another software package on Bioconductor[72]. While these projects offer fantastic tools to scientists that already posses some programming and scripting skills, a large portion of biomedical researchers lack these currently.

To address this issue, CorrMapper was developed as an online research tool that can be used without any coding experience. Although, this required an extensive amount of additional development, hopefully the online interface will encourage a lot more users (including non-technical ones) to incorporate CorrMapper into their research toolbox.

In recent years, the open-source and open-access agenda has really transformed the research community. Today, most software libraries are built using free software and are made available as open-source projects. There has never been a time when it was easier to access the source code of a project and get involved with its development through online version control systems such as GitHub.

Projects like Python's Scikit-learn[73] and JavaScript's d3.js[64] are extremely successful examples of this trend. By open-sourcing these projects, they grew quicker, became stronger and more feature rich than the authors would have ever imagined originally. They have been used by hundreds of companies and thousands researchers around the globe and consequently had a tremendous impact on the world we live in.

To align with this positive trend, the entire code-base of CorrMapper was open-sourced in the hope that not only will the science and developer community benefit from it, but they can also help to review and advance the project.

1.5.7 A meta-tool for data integration and visualisation

After carefully considering each of the above described challenges for several months, it became clear that these are all pieces from the same puzzle, hence they could be addressed together with a single meta-tool.

- Reflecting on the growing number of multi-omics studies, CorrMapper was designed to be a general data integration tool that can use any pair of preprocessed omics datasets. However, it works equally well with a single dataset.

- CorrMapper allows any kind of clinical metadata to be mapped onto the omics datasets. This is done in a novel and interactive way that can help with patient stratification and exploratory data analysis. Since the interface was designed to be used by a non-technical audience and run in a browser, it can easily help the work of doctors and other medical professionals.
- Clinical metadata driven feature selection is at the core of CorrMapper's pipeline. This can reduce the number of features dramatically to retain only the ones that are relevant to the studied biological problem. This not only saves computational time in subsequent analysis steps, but also increases the interpretability of the results.
- CorrMapper contains two advanced visualisation modules, which were developed to display and expose the complex correlation networks that are estimated from the selected features. These interfaces provide unparalleled interactivity that not only help with data exploration but can also spark new ideas and generate novel testable hypothesis for follow-up experiments.
- Given the large p small n nature of omics datasets and the black-box problem of powerful modern machine learning algorithms, CorrMapper focuses on data exploration, visualisation and ultimately knowledge discovery. Nonetheless, the features selected by CorrMapper can certainly enhance subsequent model building, by reducing the noise in the training data.
- Very importantly, CorrMapper is accessible online as a graphical interface without any programming knowledge. Hopefully, this will facilitate the wider utilisation of it among biomedical researchers.
- Finally, CorrMapper is also fully open-source. This will not only make the project more robust (by the community's helpful code checking process), but it will also help CorrMapper to grow and mature through the contributions of other researchers and developers.

2 Datasets

Most of the development and research presented in this thesis was carried out on the aforementioned bariatric surgery study, which was collected and organised by Dr. Jia Li in the CSM group at Imperial College London. Since this dataset has not been published yet, a lot of time and effort was spent on cleaning and organising the data for subsequent analysis. Furthermore, a novel NMR peak detection and extraction pipeline was also developed, so that the highly multicollinear NMR data can be used by conventional statistical and machine learning techniques.

The genomic visualisation module of CorrMapper (see section 6.4) was developed using a breast cancer study[36]. As this is a published dataset, no data preparation or preprocessing steps were taken with it. A brief overview of these multi-omics datasets and the preprocessing of bariatric surgery dataset is provided in the following sections.

2.1 Bariatric surgery dataset

Bariatric or weight loss surgery includes a variety of procedures(Roux-en-Y Gastric Bypass (RYGB), sleeve, gastric band) and it is performed on patients who are severely obese, with a Body Mass Index (BMI) over 35. Beyond the weight loss however, bariatric surgery was shown to have a number of beneficial health outcomes in previous research, such as reduced 5-year mortality (-89%) and risk of cardiovascular disease (-82%), but also resolved Type-2 Diabetes (T2D), sleep apnoea and asthma in 83%, 85% and 82% of patients respectively. In the same study bariatric surgery

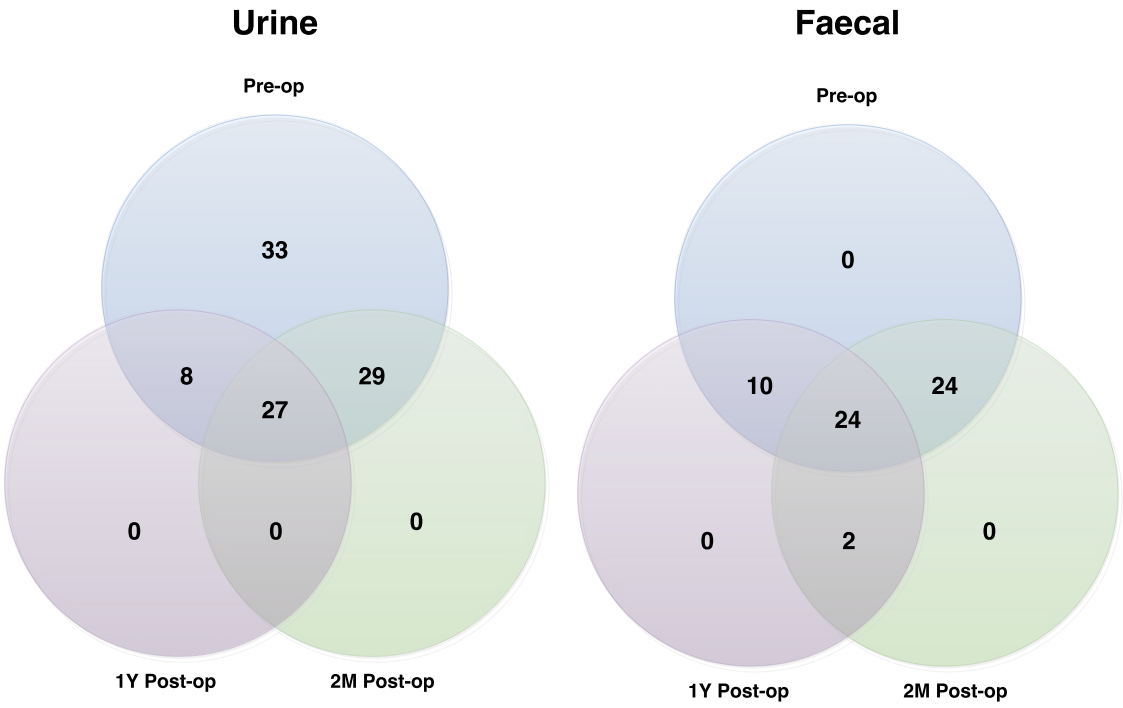


Figure 2.1: Venn-diagrams of the 97 patients, showing the collected NMR samples partitioned by time-points and sample source.

was also found to improve steatosis in 90% of patients with non-alcoholic fatty liver disease and the quality of life of the vast majority (95%) who undergo the surgery[74].

In the study used throughout this thesis, 97 severely obese patients underwent bariatric surgery (mostly RYGB). Their metabolic profiles and gut microbiome were measured by ^1H NMR and 16S rRNA sequencing at three different time-points: before operation, two months and a year after operation, referred to as T1, T2 and T3 hereafter, see Table 2.1. Figure 2.1 summarizes the collected NMR measurements by biofluids. Urine samples were profiled by NMR, while faecal samples were measured by NMR and 16S rRNA sequencing.

Additionally, a wide range of tests were carried out on the patients to monitor the physiological outcome of the surgery. This metadata contains 65 clinical and biochemical parameters including BMI, blood pressure, complete blood count, thyroid

	Pre-operation (T1)	2 months post-op (T2)	1 year post-op (T3)
NMR - urine	97	56	35
NMR - faecal	58	50	36
16S rRNA	38	38	10

Table 2.1: Number of samples per analytical platform and time-point in the bariatric surgery study.

function tests, lipid profile, iron profile, coagulation screen, urine and bone profile and concentrations of trace metals. Unfortunately, this metadata is quite sparse, on average 56% of the values are missing.

The clinical metadata for this study came in numerous Excel spreadsheets from our NHS partner hospital. Unfortunately, the data were inconsistently formatted and flooded with mistyped values. Furthermore, the linkage of samples across omics platforms was done manually relying on further Excel spreadsheets with inconsistent sample names.

To remedy this, the naming of the samples was unified, the metadata was cleaned using Python[75] scripts, and put into an SQLite database[76] to hold all sample and patient related information in three tables: Patients, NMR samples, and 16S rRNA samples. This database made it very easy to query the metadata and select any arbitrary sub-group of patients with their corresponding samples.

2.1.1 Data preprocessing of ^{1}H NMR data

An NMR spectrum obtained from metabolomic profiling of biofluids is one of the most feature rich omics data types currently in existence. Modern NMR machines produce 64K resolution spectra, meaning that once the raw data is Fourier transformed into the frequency domain, each sample is represented by 64,000 data-points.

The human urinary metabolome, on average, contains a few hundreds of metabolites[77]. Therefore, the 64,000 dimensional representation of this information is highly

redundant. NMR spectra of biofluids show extreme multicollinearity[78], as the datapoints next to each other are almost perfectly correlated and a given metabolite is represented with several peaks in the spectrum. Consequently, distant parts of the spectrum can be strongly correlated, as they hold information about the same metabolite. Traditionally, this problem was combated in metabonomics by using specialised algorithms such as PLS[79] and O-PLS[23]. However, in this work, the NMR spectra of the bariatric surgery dataset had to be integrated with another kind of omics dataset, therefore its dimensionality had to be reduced and its true features (metabolite concentrations) extracted.

To achieve this, a novel NMR peak-fitting pipeline was written in MATLAB, which finds well-aligned and relevant peaks in the data, then fits a Lorentzian curve to each one of them to estimate the peak's intensity[80]. Given n high resolution NMR spectra $X \in \mathbb{R}^{n \times 64000}$, the pipeline extracts around 1000 peaks across the n samples and produces $\hat{X} \in \mathbb{R}^{n \times\ 1000}$ whose multicollinearity is substantially lower than the original dataset's X.

In NMR profiles obtained from biofluids, the peaks of certain molecules are commonly shifted compared to where we find them in other samples, due to fluctuations in pH, temperature, instrument factors and ion content of the sample. To remedy this and to the make NMR spectra of different samples comparable i.e. aligned, certain peaks of each spectrum need to be moved or stretched. This is one of the most challenging parts in NMR spectra preprocessing, therefore numerous alignment algorithms have been devised to deal with it[81].

In the devised peak-fitting pipeline the Recursive Segment-Wise Peak Alignment (RSWPA) algorithm[82] was used to align the position of peaks across the samples. Although this is a powerful tool which corrects the misalignment of peaks demonstrably well, by shifting segments of each spectrum, it introduces breaks and stretches into the data that would make the peak-fitting a lot harder and inaccurate. To over-

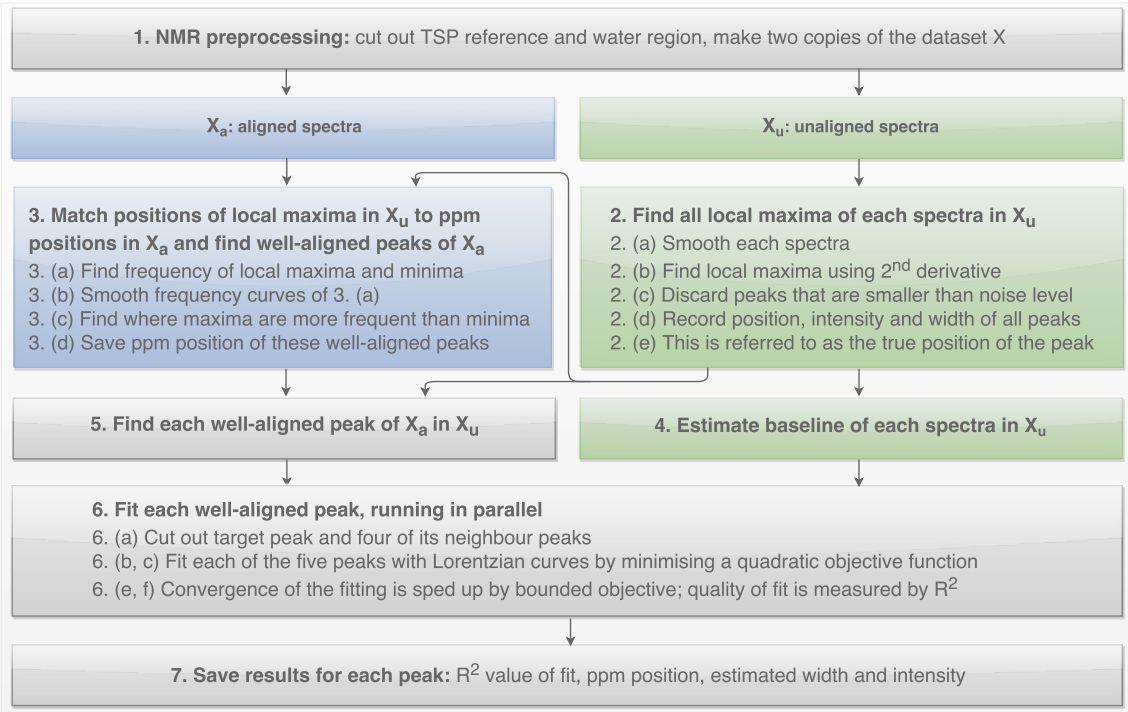


Figure 2.2: Flowchart of NMR fitting pipeline.

come this, the devised pipeline fits the peaks of unaligned spectra, but uses RSWPA aligned data to find the well-aligned and relevant peaks that need to be fitted. The following section provides a high-level overview of the peak-fitting algorithm. Figure 2.2 is intended to aid the reader while walking through the pipeline's description.

1. The imported NMR data $X \in \mathbb{R}^{n \times 64000}$ is cut to exclude the trimethylsilyl-propanoic acid (TSP) reference and water regions of the spectrum, resulting in X_c, which is then duplicated. One of the copies X_a is aligned using the RSWPA algorithm, while the other X_u is not.

2. In every spectrum s_u of X_u, all local maxima are identified:

   (a) Firstly, s_u is smoothed with a Savitzky-Golay (SG) filter[83] using a window size of thirteen and a degree four polynomial. The smoothed spectrum could be thought of as a function $\hat{s_u}(x)$, where x is the chemical shift expressed in parts per million (ppm) with reference to the standard chemical compound in the sample (TSP in most urine NMR datasets).

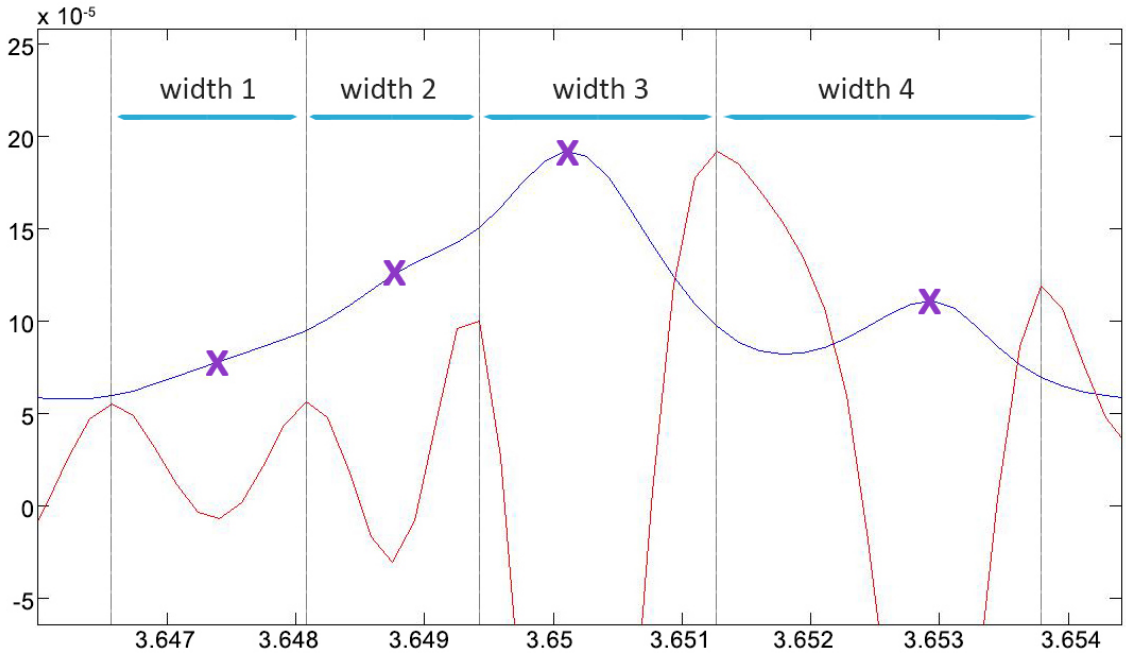


Figure 2.3: A section of an unaligned urinary NMR spectrum (black) and its SG smoothed 2nd derivative $\hat{s_u}''(x)$ (red). Purple crosses represent local maxima of the NMR section, which are found at $x = a$, where $\hat{s_u}'(a) = 0$ (not shown) and $\hat{s_u}''(a) < 0$. Each peak's position and intensity is saved along with its width, which is estimated as the ppm difference between the two nearest local positive maxima of $\hat{s_u}''(x)$ (grey vertical lines).

(b) It is well established in calculus and in NMR signal processing[84] that one can find a local maximum of $\hat{s_u}(x)$ at $x = a$, where $\hat{s_u}'(a) = 0$ and $\hat{s_u}''(a) < 0$, see Figure 2.3. All such points a of $\hat{s_u}(x)$ are found and collected in $\mathbf{m}$, using MATLAB's extrema function.

(c) $\mathbf{m}$ for which $\hat{s_u}(\mathbf{m}) < 8 * \sigma_{su}$ are discarded. Here σ_{su} is defined as the noise level of the spectrum s_u, which is found by segmenting s_u into 128 equally sized sections and then taking the smallest standard deviation of these spectral pieces[85]. This step eliminates very small peaks that are comparable to the noise level of the NMR spectrum.

(d) For each $\mathbf{m}$ that pass the noise-filtering step, we record the corresponding ppm position $\mathbf{m}_p$, intensity $\mathbf{m}_i$ and width $\mathbf{m}_w$. Given a peak $p \subseteq \mathbf{m}_p$, its intensity is defined as $\hat{s_u}(p)$ where $\hat{s_u}$ has been total area normalised so that $\int_{-\infty}^{\infty} s_u \approx \sum_{x=0}^{x=10} \hat{s_u}(x) = 1$, where $x \in [0, 10]$ is the ppm scale of the

NMR spectrum. This normalisation step is necessary because urine NMR spectra can differ widely in peak intensities for a given ppm position, due to the varying amount of water in the samples. Given a peak $p \subseteq \mathbf{m}_p$, its width is estimated as the ppm difference between the two nearest local positive maxima of $\hat{s_u}''(x)$, see Figure 2.3.

(e) For clarity, from herein, the position of a local maximum within X_u is referred to as the peak's true position, whereas the corresponding ppm position in the aligned spectrum X_a is referred to as aligned position.

3. In order to find well-aligned and relevant peaks across the spectra in X_a, the true position of each local maxima in X_u is matched with their aligned position in X_a. Then the pipeline identifies ppm positions in X_a of well-aligned local maxima (magenta spikes in Figure 2.4) using the following steps:

   (a) Firstly, the frequency of local maxima f_{max} and local minima f_{min} are calculated for every position along the ppm scale. This is defined as the number of local extrema found at a given ppm position divided by n, i.e. the number of spectra in X_a.

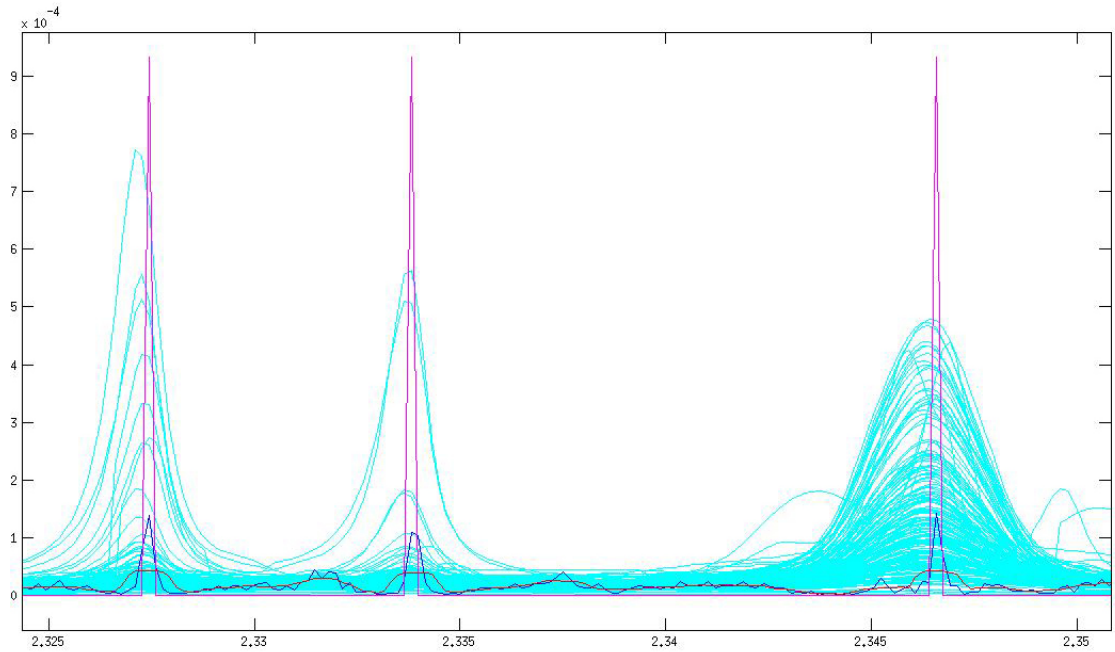


Figure 2.4: A section of the 188 aligned urinary NMR data (cyan) of the bariatric surgery dataset. The dark blue line represents the non-smoothed f_{max}, while the red one is the smoothed $\hat{f}_{max}$. The magenta spikes mark the ppm positions Π of the well-aligned peaks that will be fitted in the consequent steps.

(b) As alluded to before, since alignment of NMR spectra is an incredibly difficult task, it is not surprising that even with an advanced tool such as the RSWPA algorithm, the maxima of peaks will never be perfectly aligned across spectra. Consequently, both f_{max} and f_{min} are fairly noisy (see Figure 2.4) which is alleviated by smoothing them using a rolling average with a window size of seven, resulting in $\hat{f}_{max}$ and $\hat{f}_{min}$.

(c) Then, $\hat{f}_{max}$ is set to zero where $\hat{f}_{max} < 3 * \hat{f}_{min}$, i.e. where it is much more likely to find local minima than local maxima.

(d) Finally, $\hat{f}_{max}$ is total area normalised such that $\int_{\infty}^{\infty} \hat{f}_{max} \approx \sum_{x=0}^{x=10} \hat{f}_{max}(x) = 1$, where $x \in [0, 10]$ is the ppm scale of the NMR spectrum. This results in clearly identifiable spikes in $\hat{f}_{max}$ (show with magenta in Figure 2.4), which mark the ppm positions $\pi \subseteq \Pi$ of the well-aligned and relevant peaks in X_a, which will be fitted in the consequent steps.

4. The baseline b_u of s_u is estimated using a parametric smoothing model[86]. For an example of the fitted baseline by this algorithm see Figure S1.

5. Then, at every well-aligned peak position $\pi \subseteq \Pi$, the pipeline checks for each s_a of X_a if it has a local maximum. Note that in step 3. the local maxima $\mathbf{m}$ of each peak were matched between X_u and X_a, therefore we have $\mathbf{m}_p$, $\mathbf{m}_i$ and $\mathbf{m}_w$ for all local maxima of X_a. It is unlikely however, that any $\mathbf{m}_p$ of s_a will match $\pi \subseteq \Pi$ precisely, due to imperfect alignment. Therefore, the pipeline checks the closest peaks before $\mathbf{m}_{p-1}$ and after $\mathbf{m}_{p+1}$ the ppm position marked by π. If there is a peak found within 0.1 ppm of π, its intensity $\mathbf{m}_i$ and width $\mathbf{m}_w$ are compared to $\frac{1}{n}\sum_{i=1}^{n} X_a(\pi)$ and $\frac{1}{n}\sum_{i=1}^{n} \mathbf{m}_w(\pi)$, i.e. the average intensity and width of the peaks found in X_a at π. If these parameters of the neighbourhood peak $\mathbf{m}_{p+1/p-1}$ are within one standard deviation of the average intensity and width of $X_a(\pi)$, the pipeline accepts it.

6. Either a peak was perfectly well-aligned ($\pi = \mathbf{m}_p$) or was in the close neighbourhood and accepted ($\pi = \mathbf{m}_{p+1/p-1}$), it goes through the same fitting procedure in $s_u \subseteq X_u$.

   (a) Firstly a section q of the spectrum s_u is cut out, containing the peak of interest $s_u(\pi)$ and its neighbourhood, consisting of additional four peaks, two on each side.

   (b) Each of the five peaks within q is fitted simultaneously with a Lorentzian curve which is parametrised by the peak's position within the section $\mathbf{m}_p$, intensity $\mathbf{m}_i$ and width $\mathbf{m}_w$. For the optimisation, the starting values of these parameters are obtained earlier as described in 2.d.

   (c) The NMR section q is approximated with $\hat{q}$, by minimising the following quadratic objective function using MATLAB's `fminsearch`:

$$\underset{\mathbf{m}_p,\mathbf{m}_i,\mathbf{m}_w}{\arg\min}\ \mathbf{L}(q,\hat{q}) = (q - (b_u + \sum_{k=1}^{5} \mathcal{L}(\mathbf{m}_{p(k)}, \mathbf{m}_{i(k)}, \mathbf{m}_{w(k)}))^2,$$

   where $\hat{q}$ is made up of the baseline b_u estimated in step 4., and five Lorentzian curves $\mathcal{L}(\mathbf{m}_{p(k)}, \mathbf{m}_{i(k)}, \mathbf{m}_{w(k)})$, which are parametrised by their position, intensity and width parameter.

   (d) The result of the fitting process on two example sections are shown in Figure 2.5. As we can see, an excellent fit can be achieved with a fitting window size of five, i.e. with only four neighbour peeks. However, during the development of the peak-fitting pipeline, numerous experiments were carried out to find the optimal fitting-window size, which provides a good balance between fitting accuracy and computational time. For examples of different fitting-window sizes ($k = 5, k = 7, k = 11$), see Figure S2.

   (e) To speed up the convergence of `fminsearch`, $\mathbf{m}_p$, $\mathbf{m}_i$ and $\mathbf{m}_w$ are bounded, which turns this task into a constrained optimisation problem. The position of each peak $\mathbf{m}_p$ is bounded to allow for minimal shift up to five

data points in both directions along the ppm scale. The intensity $\mathbf{m}_i$ have upper and lower bounds of three times and 5% of the initial values respectively. The width parameter $\mathbf{m}_w$ has an upper bound of three times the initial value and a lower bound of one data point. The termination tolerance of `fminsearch` is set to 10^{-3}.

(f) When `fminsearch` has converged, the quality of the fit is estimated by the coefficient of determination, calculated as $R^2 = cor(q, \hat{q})^2$, i.e. the squared Pearson correlation coefficient of q and $\hat{q}$. Furthermore, for each well-aligned peak $\mathcal{L}(\pi)$ the pipeline saves its area as $\mathbf{m}_{a(\pi)} = \int_{-\infty}^{\infty} \mathcal{L}(\mathbf{m}_{p(\pi)}, \mathbf{m}_{i(\pi)}, \mathbf{m}_{w(\pi)})$. The end result of the fitting process is $\hat{X} \in \mathbb{R}^{n \times \Pi}$, where each row is a vector of $\mathbf{m}_{a(\pi)}$ values, representing the intensity of the fitted peaks within that particular sample.

7. Since the above described process is embarrassingly parallel, MATLAB's parallelised `parfor` loop is used to speed up the peak-fitting. Each peak's fitting time, R^2 value, and parameters $(\mathbf{m}_{a(\pi)}, \mathbf{m}_{p(\pi)}, \mathbf{m}_{i(\pi)}, \mathbf{m}_{w(\pi)})$ are saved. By the end of this pipeline, the initial dataset $X \in \mathbb{R}^{n \times 64000}$ is reduced to $\hat{X} \in \mathbb{R}^{n \times \Pi}$.

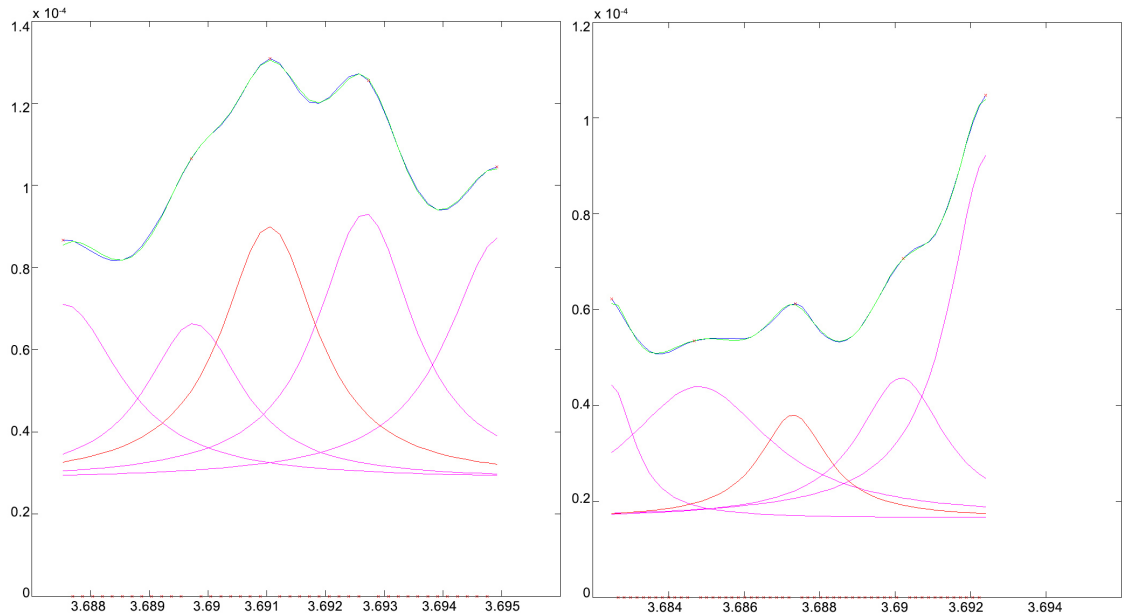


Figure 2.5: A pair of urinary NMR sections with their middle peaks fitted. The blue line is the real, unaligned spectrum, the green one is the sum of fitted Lorentzians plus the estimated baseline, the magenta ones are the fitted Lorentzians and the red one is the Lorentzian of the peak of interest.

	Urine NMR dataset	Faecal water NMR dataset
Number of samples	188	144
Number of features detected	1072	1236
Number of peaks fitted	81114	62569
Total CPU-time	11.9 hours	9.2 hours
Total wall time	4 hours	3 hours
Median of features per sample	441	435
Std of features per sample	66	68
Median R^2 values	0.996	0.998

Table 2.2: Summary statistics of peak-fitting the two NMR datasets of the bariatric surgery study.

Table 2.2 summarises the outcome of applying the NMR peak-fitting and feature extraction pipeline to the two NMR datasets of the bariatric surgery study. On average around 1100 well-aligned peaks were detected in the two datasets, and 440 of these were present in each sample.

The quality of peak-fitting was extremely high with a median R^2 value above 0.99 in both datasets. The computation took 3.5 hours on average per dataset and was carried out on a desktop machine with a four core Intel i5 CPU. Unfortunately, by the time of finishing this peak-fitting algorithm, the Focus pipeline got published[87], whose functionality overlaps considerably with this work.

2.1.2 Data preprocessing of 16S rRNA data

Metataxonomics works by sequencing all 16S rRNA genes in a given sample. Once the reads are clustered by similarity, these clusters can be mapped to a previously determined phylogenetic tree with thousands of OTUs such as the GreenGenes database[88].

This mapping will not only identify the OTUs that are present in a given sample but also quantify their relative abundance using the number of 16S rRNA reads as a proxy for the number of bacterial cells per taxa.

This relies on the key assumption however, that 16S rRNA genes are only present in a single copy in each organism and therefore a single 16S rRNA read can be equated to a single bacterial cell. Unfortunately this is known to be a false assumption and it is widely accepted that the copy numbers of the 16S rRNA gene shows strong variability across bacterial species[89].

Figure 2.6 shows the copy numbers of the 16S rRNA gene in 1073 fully sequenced bacterial species. The above described assumption does not hold in about 80% of bacterial species, as they all have more than a single copy of this gene. Given the surprisingly high number of species with more than one copy, recently it has been gaining recognition within the metatoxonomics community[90,91], that it is erroneous to treat all OTU counts as the equivalent number of organisms.

Although doing so is still common practice, it will distort the true relative abundance of the sampled bacterial community. Instead, the OTU counts should be normalised by the copy numbers of the given OTU, to better approximate the real abundance of a genus.

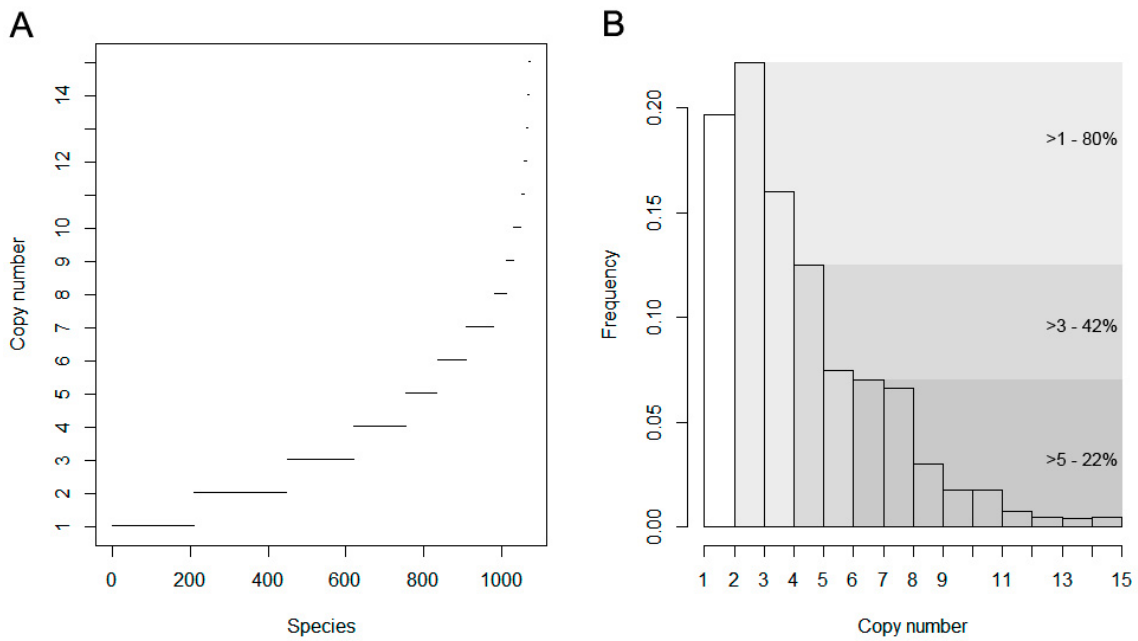


Figure 2.6: Distribution of the 16S rRNA gene copy numbers in 1073 fully sequenced bacterial species. **A:** Plot of the 16S rRNA copy numbers in ascending order for visual clarity. **B** Histogram of the same data. The shaded rectangles represent the percentage of species with more than one, three and five 16S rRNA gene respectively.

To overcome this problem, PI-CRUST[92] was used to normalise the OTUs by their species specific copy number. Following this, all singleton (OTUs (present in one sample with a single read) were removed. This filtering retained 192 OTUs in total and discarded extremely rare strains that would have made the OTU table very sparse. Finally, as suggested by several authors[93,94], Centered Log Ratio (CLR)[95] was applied to the data, to account for its compositional nature.

2.2 Breast cancer dataset

For the development of the genomic network explorer (see Section 6.4), a multi-omics breast cancer study was used[36]. This seminal paper represents one of the earliest application of the multi-omics approach in biomedical research. The authors successfully combined gene expression and CNV assays to show that stratification of patients according to outcome can be improved by measuring both of these omics data types.

CNV probes measure if a certain region of the genome is duplicated or deleted. Therefore, the biological rationale behind this study was to examine how copy number abnormalities of the genome can lead to over or underexpression of certain genes and consequently affect the disease progression and ultimately the clinical outcome.

The dataset contains an assay of 2149 CNV probes (mix of Scanning and OncoBAC arrays) for 145 patients and a gene expression assay with 22215 probes (Affymetrix High Throughput Array GeneChip system) for 118 patients. In total 108 patients had both omics data types. For CorrMapper's genomic network visualisation module, the chromosomal location and other biological annotation was obtained for each of these probes from the vendor's website.

The clinical metadata for this dataset included tumour staging, type of therapy, mortality over the studied period, progesterone and oestrogen receptor status, presence of family history, ethnicity, p53 status, ErbB2 status and age of patients.

The data was downloaded from the Gene Expression Omnibus[96], hosted by the National Center for Biotechnology Information (NCBI). As the data files were already preprocessed by their platform specific pipelines, no further transformation steps were applied to them. The two omics data files, their genomic annotation files and metadata can be downloaded from CorrMapper's website as an example dataset.

3 Methods

3.1 Feature selection

Feature selection (FS) is an integral and essential part of CorrMapper, therefore, a brief introduction to this subject is provided in the following section.

FS is a highly studied, central problem in machine learning, with a vast literature and dozens of published . By reducing the dimensionality of a dataset, FS methods decrease the chance of over-fitting while also improving prediction accuracy. Furthermore, they also make the interpretation of datasets easier, which is highly desirable in high-throughput systems biology.

Generally, the omics data of any biomedical research experiment can be recorded as a matrix $\hat{X} \in \mathbb{R}^{n \times p}$, which contains the measurements for n samples and p features, where usually $p \gg n$. Typical examples for X include:

- expression of a genes measured as relative fluorescent intensity,
- abundance of bacterial strains quantified by 16S rRNA sequencing,
- concentration of small molecules in a biological fluid, measured with ^{1}H NMR.

Each cell $x_{i,j}$ of X is a measurement for feature j in sample i. Throughout this thesis x_i refers to the row in X (i.e. sample), while x_j represents a column (i.e. feature) of the dataset. In biomedical research, FS consists of identifying a subset of features, which are most discriminative with respect to a particular biological response or outcome $y^{n \times 1}$, e.g. cancer vs healthy or BMI of patients.

More precisely, let us consider a classification problem with $y \in \{0, 1\}$, where 0 represents control cases while 1 denotes patients with cancer. A sample x_i in our cohort X could be characterised by a *feature vector* $x^{1\times p}$ and an outcome y. A classifier is a function $\Phi(x)$ that acts as a map from the feature space to the label space: $\Phi : \mathbb{R}^p \to \{0, 1\}$, meaning that y is to be predicted by $\Phi(X)$. As the classifier is fitted to the data, it finds a decision boundary within $\mathbb{R}^p$ that partitions this space into two subspaces: $\mathbb{R}_0 = \{x : \Phi(x) = 0\}$ and $\mathbb{R}_1 = \{x : \Phi(x) = 1\}$. The classifier's error $\varepsilon[\Phi]$ measures the probability of misclassification:

$$\varepsilon[\Phi] = P(\Phi(X) \neq y) \approx \frac{1}{n} \sum_{i=1}^{n} |\Phi(x_i) \neq y_i|.$$

The feature-label distribution $F_{x,y}(x^{1\times p}, y)$ quantifies the class conditional distributions of the *feature vectors* with respect to the outcome: $F_{x|1}(x)$ and $F_{x|0}(x)$. The accuracy of a binary classifier depends on how well $F_{x|1}(x)$ and $F_{x|0}(x)$ are separated by the partitions $\mathbb{R}_0 = \{x : \Phi(x) = 0\}$ and $\mathbb{R}_1 = \{x : \Phi(x) = 1\}$. Given the space of all possible binary functions B, the optimal classifier Φ_B (often called the *Bayes classifier*) is one which is minimising the Bayes error ε_B:

$$\varepsilon[\Phi_B] = \min\{\varepsilon(\Phi) : \Phi \in B\}.$$

Unfortunately, $F_{x,y}$ is unknown in real datasets, as it would require the whole population to estimate it. Instead, the classifier needs to learn from a random sample U_n of feature-label pairs $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} = X \in \mathbb{R}^{n\times p}, y \in \mathbb{R}^{n\times 1}$. Unless the fitted classifier Φ_n is the Bayes classifier Φ_B, its error ε_n will always be greater than the Bayes error[97]. This results in a so called *design cost* $\Delta_n = \varepsilon_n - \varepsilon_B$, where Δ_n and ε_n are sample dependent random variables. The expected error of Φ_n can be decomposed as

$$E[\varepsilon_n] = \varepsilon_B + E[\Delta_n],$$

where $E[\Delta_n]$ is the expected design cost over all possible random samples. The classification rule learned by the classifier is said to be *consistent* if $E[\Delta_n] \to 0$ as $n \to \infty$ relative to $F_{x,y}$, and *universally consistent* if the same holds for any feature-label distribution.

The reason why $p \gg n$ presents a tremendous challenge in the statistical analysis of omics datasets, is that $E[\Delta_n]$ is too high when the sample size is small. The reason why this occurs is that small sample data does not represent the full distribution of $F_{x,y}$ accurately. Therefore, one can easily design a classifier that will have low error on the training dataset $\varepsilon[\Phi_n(U_n)]$, but will generalise very poorly and produce large errors on unseen data pairs $\{(x_i, y_i)\}$ coming from the full feature-label distribution $F_{x,y}$. In machine learning and statistical modelling this phenomenon is called *overfitting*, which can be alleviated with either larger sample sizes that represent the true $F_{x,y}$ distribution more truthfully, or regularisation meaning various ways to restrict model flexibility.

FS constrains the classifier $\Phi(X)$ to a smaller class of potential classifiers by selecting a relevant and potentially non-redundant subset of features $S \subseteq \{x_1, x_2, ..., x_p\}$. By limiting the feature space to $|S| \ll p$, FS limits the complexity of decision boundaries and feature space partitions a classifier can learn, hence FS reduces the chance of overfitting. At the same time FS makes the resulting model easier to interpret while also reducing training time significantly. Finally, FS helps to avoid the curse of dimensionality, whereby even the closest neighbours of a data-point will appear to be highly dissimilar, making statistical inference very difficult[98].

Although it has clear advantages, FS also adds another level of complexity to data modelling, as we not only have to find the optimal parameters $\hat{\theta}$ of our model Φ fitted to our data $X^{n \times p}$ and outcome variable y, but we are also looking for an

optimal subset of features S such that:

$$\hat{\theta}, S = \underset{\hat{\theta}\in\Theta, S\subseteq p}{\arg\max}\, \mathcal{L}(\Phi_{\hat{\theta}}, S|X^{n\times p}, y),$$

where $\mathcal{L}$ is the likelihood function of $\hat{\theta}$ and S. A feature selection algorithm is the combination of a search technique which proposes new feature subsets, and an evaluation measure that is used to score these. Finding S might sound trivial at first: all we have to do is create increasingly large sets of features: $S = \{1, 2, ..., p\}$, and for every set size $k = |S|$ check all possible permutations $\binom{k}{p}$. As p grows above 10 however, this becomes computationally infeasible very quickly. Therefore all FS methods are estimating the optimal set as $\hat{S} \approx S$, using various heuristics and greedy search algorithms.

Saeys *et al.* provide a good overview of the different kinds of FS methods[99]. According to them and others, FS algorithms can be classified into four main categories:

Filter methods are classifier independent and capture the discriminating power of each feature by calculating their *relevance* statistic with respect to the outcome variable. Mutual information and Pearson's correlation are two commonly used relevance statistics that can quantify the association between each feature x_j and the outcome variable y. Once these statistics are calculated for all features, they could be ranked and filtered using a heuristic decision process, such as "keep the top k features". Filter methods are computationally cheap, as they do not require the training of any classifiers, but they are univariate, i.e. they cannot take crucial interactions between features into account, nor can they eliminate redundant features from X, as these will all get high relevance scores.

Wrapper methods rely on an external classifier and cross-validation. They assess the *usefulness* of a proposed features set $\hat{S}$ by fitting a model Φ_B to the data $X^{n\times\hat{S}}$ and measuring the cross-validated prediction accuracy of the classifier. This model

fitting has to be done for every proposed $\hat{S}$ and the best set is chosen as the one that minimises the cross-validated misclassification error. Therefore, they can be very expensive computationally, and their use could even become prohibitive with large datasets. They are also prone to over-fitting and their choice of *useful* features can be unstable, depending greatly on chosen classifier. Nonetheless, out of all FS methods, wrapper techniques are the most likely to find the optimal set $S \subseteq p$, given the chosen class of model $\Phi \in B$ is appropriate for the data.

Embedded methods are similar to wrapper techniques, but they improve on some of their shortfalls, as they perform FS *while* training the classifier. This inseparable coupling of learning and FS is as an inherent property of some classifiers such as the L1 norm penalised Support Vector Machines (SVMs)[100] and the LASSO regression[101]. They are generally computationally less intensive than wrapper methods.

Ensemble methods represent another classifier dependent way of FS. Instability issues are commonly observed in several FS techniques, whereby small perturbations in the training data could lead to quite different $\hat{S}$. Ensemble methods address this problem, by using subsampling strategies such as bootstrapping[102] to select a more robust set of features.

During the development of CorrMapper, several months have been spent on reviewing the literature of FS and finding an optimal set of methods that could be offered to the user. The lesser known Boruta and Joint Mutual Information (JMI) algorithm, described in then next sections, stood out from the list of well-known FS methods, because of their advantageous properties. However, they were not available in Python, and Boruta's R[103] implementation was fairly slow. Therefore, a fast, parallelised and flexible Python version was developed of both algorithms, which were benchmarked along with five other FS methods on hundreds of simulated datasets. In the next sections each of the benchmarked FS methods are introduced briefly, while the lesser known Boruta and JMI are described in greater detail.

3.1.1 Boruta

Boruta is an *all-relevant* FS algorithm[104], that aims to find every feature that has some useful information for the prediction of the outcome variable y. Wrapper methods iteratively prune the original feature space until they find a *minimal-optimal* set, which maximises the cross-validated prediction accuracy of a chosen classifier $\varepsilon[\Phi]$.

In basic biological research however, we generally want to know all the relevant features that bear some information regarding the response variable of interest. Furthermore, since Boruta is based on Random Forests (RFs)[105], it is robust against untransformed data and outliers, while it can also cope with correlated features and non-linearities.

The detailed description of the Boruta algorithm, has to be preceded by a brief introduction to decision trees, the bias variance trade-off and RFs, as they represent the core idea behind Boruta.

Decision trees

A random forest is an ensemble of decision trees, built on the old idea that a group of weak and uncorrelated learners can generally outperform a single strong learner. Decision trees work by learning how to partition the feature space $\mathbb{R}^p$ of a dataset X into k number of p-dimensional hypercubes, so that the outcome variable of samples within each of those regions are as homogeneous as possible. This is achieved through applying a series of binary splitting rules to the data. These rules form a tree structure, i.e. each one is taking samples as their input from the output of a previous splitting rule, see Figure 3.1. Decision trees could be applied to both regression ($y \in \mathbb{R}$) and classification problems ($y \in \{0, 1\}$) problems. Here we will briefly review how classification trees are built.

For this demonstration, a toy dataset $X \in \mathbb{R}^{10\times 2}$ with only two features x_1 and x_2, and a binary ($K = 2$) outcome variable $y = \{0, 1, 1, 1, 1, 1, 0, 0, 0, 1\}$ is used. Given a root node of the tree with the 10 samples, we need to find a splitting rule that will separate these 10 samples into two branches which are more homogeneous than their parent node was. The class-label impurity at a given node could be measured by the Gini criterion, as

$$Gini(T) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

where T represents a node in the decision tree, K is the number of classes in y (two in this example), and $\hat{p}_{mk}$ is the proportion of observations in the mth region that are from the kth class. In this example, at the root node, before any splitting, $Gini(T) = 0.479$. Let us consider splitting y based on first feature x_1, which produces two branches that separate the outcome variable as follows: $y_{T_1} = \{0, 1, 1, 1, 1\}$ and $y_{T_2} = \{1, 0, 0, 0, 1\}$. At the same time we find that splitting on x_2 gives us $y_{T_1} = \{0, 1, 1, 1, 1, 1\}$ and $y_{T_2} = \{0, 0, 0, 1\}$. We can quantify the impurity reductions of these splits as:

$$Gini_{split}(T) = \frac{N_1}{N} Gini(T_1) + \frac{N_2}{N} Gini(T_2),$$

where N_1 and N_2 are the number of samples that go into T_1 and T_2 branches respectively. Continuing with our example, $Gini_{split}(T_{x_1}) = 0.399$ and $Gini_{split}(T_{x_2}) = 0.316$, thus splitting on x_2 decreases the impurity of class labels more within y ($0.479 - 0.316 > 0.479 - 0.399$). Therefore, the decision tree will split the samples at the root node based on x_2. The exact split-point $\min(x_2) > s > \max(x_2)$ can be found using exhaustive search along the range of all features. This could be done highly efficiently in a parallelised manner, as does XGBoost[106], which is currently the most advanced tree based machine learning library.

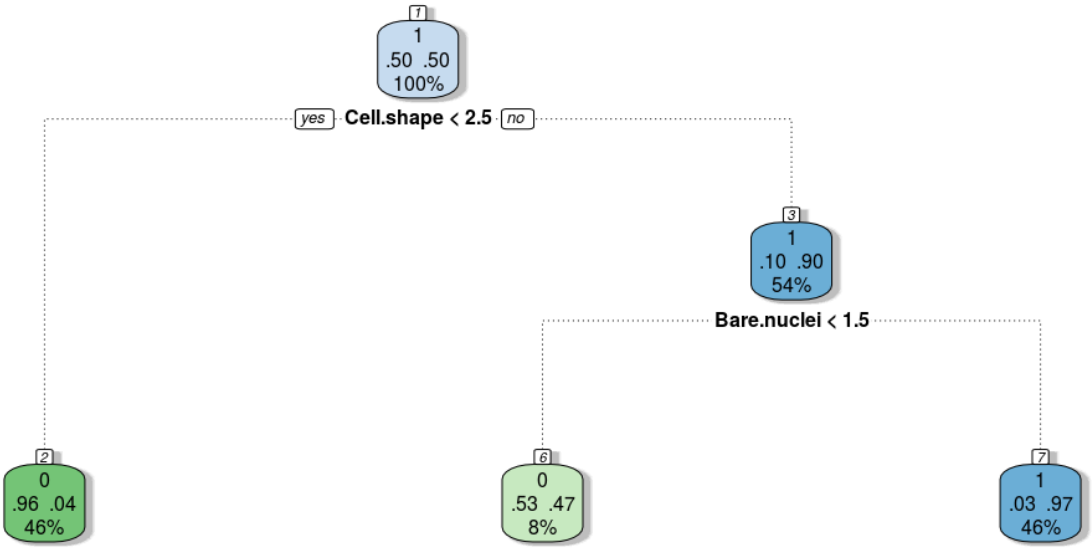


Figure 3.1: An example of a decision tree, fitted to a small (n=600, p=10) breast cancer dataset. Control cases are green, patients with cancer are blue, colours in-between represent a mix of the two classes. For each node the percentage of total patients is shown, along with the frequency of the negative and positive classes. With only two splits, 92% of all patients (46% in node 2 and 46% in node 7) are placed in almost perfectly pure nodes.

Decision trees have a number of beneficial properties: they are easy to interpret, they can handle untransformed categorical and numerical data, and take feature interactions into account. However, they also have high variance, meaning their predictions are highly sensitive to changes in the training data so they often do not generalise well to unseen data[107]. In the next section we take a short detour to introduce the bias-variance trade-off, which is essential for understanding RFs.

The bias-variance trade-off

In any regression learning problem the expected Mean Squared Error (MSE) of a learner $\hat{f}$ could be decomposed to irreducible and reducible parts[69]:

$$MSE = variance + bias^2 + error.$$

With some modification this holds for classification problems too.

Continuing with regression, more formally the above can be written as:

$$E\left[y - \hat{f}(X)\right]^2 = \underbrace{Var(\hat{f}(X)) + \left[Bias(\hat{f}(X))\right]^2}_{\text{reducible}} + \underbrace{Var(\varepsilon)}_{\text{irreducible}} \quad ,$$

where neither the variance nor the bias of the learner can be negative. Therefore, the MSE (that we would obtain if we repeatedly estimated f using a large number of training sets), can never be lower than the irreducible part of the error term $Var(\varepsilon)$.

The flexibility of a learner (often referred to as degrees of freedom in the statistical literature) depends on the class of functions a model can represent. For instance, linear regression is a fairly inflexible model, as it can only represent linear combinations of its input variables, resulting in additive, linear functions, whereas neural networks are called *universal approximators*, as they can approximate any continuous function in their input space to arbitrary precision[107] .

As alluded to earlier, *variance* refers to the amount by which $\hat{f}$ would change if we trained it on a different training dataset. A learner with low variance will extract very similar decision rules even from moderately different training datasets, hence the model's generalisability does not depend on the sampling of the data. Flexible models generally have higher variance. A well known drawback of decision trees is that they can produce very different models even with small perturbations to the training data.

Bias is the error introduced by modelling a very complex real life problem with a much simpler model. A linear regression model will assume that the outcome variable can be accurately modelled as a linear combinations of the input variables. Even if this is a good approximation, it is highly unlikely that such a simple model will capture all nuances within our data, and therefore such a model (as less flexible models in general) will have relatively higher bias compared to flexible models.

During any data analysis, a substantial part of the statistical modelling is spent of exploring the model space and finding the right trade-off between the bias and variance of various learners. Figure 3.2 illustrates this for a hypothetical learner whose λ parameter controls the learner's flexibility. More flexible models can represent richer functions and therefore can approximate a dataset better, but they are prone to overfitting, i.e. instead of extracting a generalisable rule from the training dataset that applies to unseen data, they memorise the training dataset's samples with all their non-general peculiarities.

Restricting a model's flexibility by regularisation, i.e. by shrinking some of its free parameters to zero is a common way to tackle this problem. As we will see in the next section about RFs, another way of reducing the chance of overfitting for a high variance learner is to average multiple of them as an ensemble.

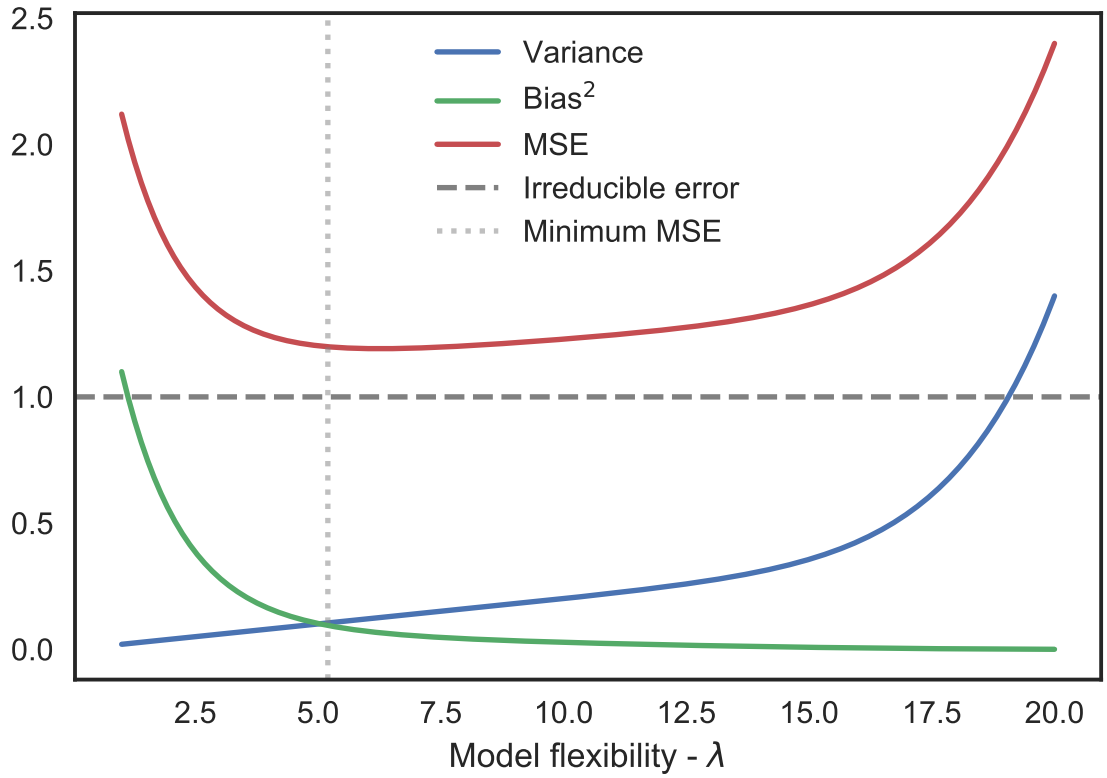


Figure 3.2: The overall mean squared error of the learner (red) can be decomposed into the irreducible error (dash grey horizontal line), reducible squared bias (green) and reducible variance (blue). The dotted vertical grey line represents λ at which MSE is minimised. Reproduced from page 36, James et al.[69]

Random Forests

Random forests use *bagging* and *random subspace* methods to combine decision trees and reduce their variance. Bagging stands for *bootstrap aggregation* and it is a general purpose procedure for reducing the variance of a statistical models. Given n independent observations $x_1, x_i, ..., x_n$ each with variance σ^2, the variance of the mean of the observations $\hat{x}$ is given by σ^2/n. In other words, averaging a set of observations reduces variance[69].

Therefore, one can increase the accuracy of a prediction method by taking many training datasets from a population, building a model with each of them, and finally averaging their predictions. This is not something we can do of course in most real life scenarios as the acquisition of each new training dataset costs substantial amount of money. However, we can use bootstrapping to sample from our only training dataset with replacement and get B bootstrapped datasets X_B. Then we can fit a model to each of these and our bagged estimator will take the form of:

$$\hat{f}_{bag} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(X_B).$$

The predictions of a bagged model can be obtained in several ways: if we combined regression trees, the average prediction for each sample across all $\hat{f}^b$ is a good approximation of the ensemble's prediction. If we used classification trees, taking the majority vote is a commonly used method. Applying bagging to decision trees increases their performance substantially because even though the *bias* of the bagged estimator is slightly higher, its *variance* is much lower[108]. This phenomenon is one of the reasons why RFs are more accurate and generalise better than decision trees. This phenomenon is demonstrated and explained in greater detail in Figure S3.

A very important benefit of bagged models is that they can estimate the generalisation error that we would obtain on a test or holdout dataset, without performing

cross-validation or other validation steps. Since each learner is trained on a bootstrapped sample, they all use approximately 2/3 of the data (this is a mathematical consequence of random sampling with replacement). Therefore, for each of the learners we have 1/3 of the dataset as Out Of Bag (OOB) observations which can be used to test the model and estimate its generalisation error. By averaging these OOB error estimates across the trees, RF produces a robust error estimate for the ensemble during training time at almost no additional computational cost.

Unfortunately by combining decision trees into an ensemble they loose one of their most appealing quality, their interpretability. To overcome this challenge, RFs produce Variable Importances (VIs), which are an internal estimate of the importance of each feature x_j. In a RF built with classification trees, they are calculated by summing the Gini impurity reductions of the class labels over all tree nodes where x_j appears. Large VI scores mean that splitting the samples on x_j purifies the class labels well in the two subsequent branches. By averaging these impurity reductions for each x_j over all decision trees of the ensemble, we get a reasonable estimate of how important each feature is.

Finally, RFs improve on bagged decision trees by restricting the features that a given decision tree is able to use during training to a random subset. By using the *random subspace* method, RFs decorrelate the bagged decision trees, which further increases the ensemble's performance. There are well established information theoretic reasons for why a diverse ensemble with uncorrelated predictions perform well leading to a substantial reduction in the model's variance[109].

Intuitively, without restricting the feature set in each tree, it might easily happen, that the same handful of features would always be used to split the samples in each tree, irrespective of training them on different bootstrapped samples. Therefore, the predictions of these trees would be extremely similar and consequently almost identical to a single decision tree.

Boruta algorithm

The main idea behind Boruta is to iteratively fit RF models to approximate the null distribution of VIs for truly irrelevant features. More concretely, Boruta trains a RF model on $[X|X_{sha}]$ and the outcome variable y, where $X^{n\times p}$ is our original data matrix, while $X_{sha}^{n\times p}$ is a duplicate of X where each feature (column of the matrix) is permuted randomly, rendering them useless for the model. Hence, these permuted columns are referred to as *shadow features*. The two matrices are simply concatenated and treated as single one $[X|X_{sha}]$. The permuted shadow features are essentially noise and therefore their VIs provide a good approximation of the importance scores that a truly irrelevant feature would receive from RF.

Once the RF is fitted to $[X|X_{sha}]$, Boruta compares the VIs of the real features to the maximum VI of shadow ones sha_{max}. If one of the real features receives higher VI than sha_{max} consistently across multiple iteration of Boruta, it is deemed relevant and kept within the dataset. Conversely, if a real feature has consistently lower VI than a noise feature, it is discarded from the dataset and is not considered in the subsequent iterations.

The high-level pseudo-code of the Boruta method is given in Algorithm 1, while the full pseudo-code could be found in the supplementary materials, see Algorithm S1.

Python implementation of Boruta

The R implementation of the RF algorithm, on which the Boruta R package was based until very recently, is one of the slowest available, as shown by Louppe[108] on slide 34. Furthermore, unlike this R version, the RF in the Scikit-Learn (SL) Python package[73] is parallelised, and it is one of the fastest available implementations, owing to the remarkable optimisation work the core developers of the SL package have done.

Algorithm 1 Boruta

```
function BORUTA(X^{n×p}, y^{n×1}, t_max)
    Set iteration number t ← 1
    while t < i_max or there are still undecided features do
        active ← Features that have not been discarded yet.
        X_sha^{n×active} ← X^{n×active}
        Permute all features in X_sha to remove their correlations with y.
        Join the two matrices as [X^{n×active} | X_sha^{n×active}].
        Train a RF classifier on the joined matrix.
        Get VI for all real and shadow features.
        sha_max ← Maximum VI of the shadow features.
        For each feature x_j ⊆ active test if VI_j ≥ sha_max.
        Remove features from active which have significantly lower VI than
        sha_max throughout the iterations.
        Keep features which have significantly higher VI than sha_max throughout
        the iterations, save them as confirmed.
    end while
    return Ŝ ←List of confirmed features.
end function
```

Consequently, to speed up the Boruta algorithm, it was reimplemented in Python and released as an open-source package under the name BorutaPy[110]. This implementation can be several orders of magnitude faster than the original R package, depending on the number of available cores for the RF training, and is now featured on the official website of the Boruta method[111].

BorutaPy is a fully tested and documented Python package that was designed to follow the general Scikit-Learn (SL) learner interface. SL is currently the most popular and comprehensive general-purpose machine learning library, used by thousands of researchers, data scientists and companies world-wide. One of the key strengths of SL is that it provides a unified interface to dozens of very high quality machine learning algorithms. Therefore, building production ready data science products or carrying out quick iterations during research becomes much simpler, as one can easily switch an algorithm for another with minimal change to the code.

This compatibility with SL has made BorutaPy quite popular (at the time of writing it has been starred by more than 180 people on GitHub) while it also granted a place

for it in the `scikit-learn-contrib` package, which is the official repository for collecting high quality machine learning packages that are 100% compatible with SL. Therefore, BorutaPy might make it into one of the upcoming releases of SL. This would be a great honour, given SL has been an invaluable tool in my work and research for years.

During the implementation of BorutaPy, several months worth of work and experimentation was spent on gaining a deep understanding of the Boruta method. This has led to two modifications of the original algorithm:

1. The number of decision trees used in a RF greatly determines the run-time of the training process. While it is claimed that increasing the number of trees in the ensemble does not cause the RF to over-fit the training data[105], it certainly increases the computational burden of the algorithm. Furthermore, the increase we get in prediction accuracy from greater number of trees, leads to diminishing returns and plateaus quickly[112].

   Therefore, it is inefficient to use the same number of trees for the consecutive iterations t in Boruta. For example in a gene expression dataset, the number of features could be in the range tens of thousands, which will get filtered to a few hundred after a couple dozens of iterations t of the Boruta algorithm.

   To address this issue, BorutaPy automatically adjusts the number of decision trees in the RF during the FS process, based on the number of active features. This modification resulted in further speed improvements compared to the original algorithm.

2. The correction for multiple testing (see Boruta paper or Algorithm S1) was split into a two step process, instead of the original original version's Bonferroni correction[113]. In each iteration t we need to correct for the fact that we test multiple features against the null hypothesis, i.e. does a feature perform better

than the shadow features. The Bonferroni correction, which is used in the original algorithm, is known to be too stringent[113,114] in such scenarios.

For this reason, the first step of correction in BorutaPy is the widely used Benjamini Hochberg FDR[115]. Following this however, we also need to account for the fact that we have been testing the same features over and over again in each iteration t with the same statistical test.

The Bonferroni correction is well suited for this task, thus it is applied as a second correction step in each iteration, by dividing the $\alpha = 0.05$ p-value threshold with the current iteration number t. This allows for finer control of the correction for multiple testing, which can lead to more relevant features being discovered.

Both of these modifications could be bypassed however if the user wishes to do so, which makes BorutaPy algorithmically indistinguishable from the original R implementation.

3.1.2 Joint Mutual Information

Before describing the Joint Mutual Information (JMI) method in detail, the next section provides a very brief introduction to information theory and some simple filter type FS methods that are based on it.

Information theory based FS methods

In the past twenty years, a large variety of information theory based filter methods were developed. In general, they all work by assessing the mutual information between the p features of $X^{n \times p}$ and y. Information theory is concerned about quantifying the uncertainty present in distributions, by measuring their entropy.

The entropy of the distribution of variable X, denoted as $H(X)$, is defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x),$$

where x denotes a possible value of variable X, that it can take from a set of values $\mathcal{X}$, and $p(x)$ is the distribution of X. This equation quantifies the uncertainty in X, and for discrete variables it could be computed by estimating $p(X)$ as the fraction of observations taking on value x from the total N:

$$\hat{p}(x) = \frac{\#x}{N}.$$

If $p(x)$ peaks around a certain value, than the entropy of it will be low, while if it is uniform, meaning all events in X are equally likely, it will be high. Furthermore, conditional entropy of two distributions X and Z could be defined as:

$$H(X|Z) = -\sum_{z \in \mathcal{Z}} p(z) \sum_{x \in \mathcal{X}} p(x|z) \log p(x|z),$$

which represents the amount of uncertainty remaining in X after we have seen Z. Finally, Mutual Information (MI)[116] between X and Z is then defined as:

$$\begin{aligned} I(X;Z) &= H(X) - H(X|Z) \\ &= \sum_{x \in \mathcal{X}} \sum_{z \in \mathcal{Z}} p(xz) \log \frac{p(xz)}{p(x)p(z)}. \end{aligned}$$

In this difference the first term represents the uncertainty *before* Z is known, while the second term captures the uncertainty *after* Z is known. Thus, mutual information could also be thought of as the amount of uncertainty in X that is removed by knowing Z. Mutual information is symmetric $I(X;Z) = I(Z;X)$ and is zero if and only if X and Z are statistically independent.

As described earlier, filter methods attempt to rank the features based on a chosen

relevance measure. Therefore, the simplest information theoretic filter techniques simply calculate the MI between all features and the outcome variable y independently, rank them, and select the k best ones to form the selected set $\hat{S}$. This is called Mutual Information Maximisation (MIM),which results in a selected set $\hat{S}$ that has *maximum relevance*, and it has been used in many early algorithms[117].

MIM is known to perform poorly however, when the features are interdependent, as it will end up selecting highly correlated thus *redundant* features. Furthermore, like all other univariate filter methods, these algorithms cannot take the interactions between predictors into account and therefore their selected $\hat{S}$ is often suboptimal.

One of the most successful FS methods that tried to address this issue is the Minimum Redundancy Maximum Relevance (MRMR) algorithm[118] which uses MI to select highly relevant features with respect to the outcome variable, while also ensuring that the selected features in $\hat{S}$ are mutually far away from each other, i.e. have low pair-wise MI and hence form a non-redundant set.

JMI algorithm

Using the JMI, Moody[119] and Meyer *et al.*[120] proposed a method that focuses on increasing the complementary information between the selected features:

$$J_{JMI}(x_j) = \sum_{\in \hat{S}} I(x_j, x_k; Y),$$

where $J_{JMI}(x_j)$ is the JMI score for feature j under consideration, and $x_k \subseteq \hat{S}$ represents all features that were selected in previous iterations of the algorithm. By calculating the joint mutual information $I(x_j, x_k; y)$ of the multivariate composite variable x_j, x_k with the outcome variable y, this selection criterion ensures, that the candidate feature x_j must not only be relevant for y, but also *complementary* to all previously selected features x_k, in order to be added to $\hat{S}$.

x_1	x_2	x_3	$x_2 \oplus x_3$	y
0	0	0	0	0
1	0	0	0	1
0	1	0	1	1
1	1	0	1	1
0	0	1	1	1
1	0	1	1	1
0	1	1	0	0
1	1	1	0	1

Table 3.1: Example dataset demonstrating interacting binary features through XOR function. Adapted from Vergara et al.[121].

Therefore, the JMI method can discover sets of features that are only relevant for the prediction of y when combined. A classic example of such an interaction is the XOR function. Table 3.1 displays a toy dataset, where the binary outcome variable is built through the function $y = x_1 + (x_2 \oplus x_3)$ from three binary features, where $+$ represents the OR logic function. In this example x_2 and x_3 are both equally useless for predicting y, but when combined through a XOR function $x_2 \oplus x_3$, they become highly predictive.

Brown *et al.*[122] in their extensive review, systematically benchmarked 17 information theoretic filter methods including the widely used Mutual Information Feature Selection[123] and MRMR[118] algorithms. They performed a large empirical study to rank these methods by their accuracy, stability and flexibility, and the JMI criterion based FS methods were picked as overall winners. The high level pseudo-code of the JMI method is given in Algorithm 2.

Python implementation of JMI

The only available implementation of the JMI algorithm was in the `FEAST` Feature Selection Toolbox[122], which was written in C and was only suitable for discrete data. To easily incorporate the JMI method into CorrMapper's pipeline and to make it available to a wider audience, as with Boruta, the algorithm was reimplemented

Algorithm 2 JMI algorithm

```
1: function JMI(X^{n×p}, y^{n×1}, k)
2:     active ← p
3:     for x_j ∈ p do
4:         I_MI ← MI(x_j, y), calculate MI between the outcome variable and all
           features.
5:     end for
6:     x_j^{I_max} ← arg max I_MI, find x_j with highest MI.
7:     Ŝ ← x_j^{I_max}
8:     Discard feature X_{I_max} from active.
9:     while |Ŝ| < k do
10:        for x_j ∈ active do
11:            for X_j ∈ Ŝ do
12:                I_JMI ← I(x_j, X_j; y), calculate joint mutual information between
                   all previously selected features within Ŝ and all remaining features
                   with respect to the outcome variable y.
13:            end for
14:            J_JMI(x_j) = Σ_{X_j∈Ŝ} I(x_j, X_j; Y), for each candidate feature x_j, calcu-
               late the JMI score as the sum of I_JMI across all X_j ⊆ Ŝ
15:        end for
16:        x_j^{J_max} ← arg max J_JMI, find x_j with highest JMI score.
17:        Ŝ ← x_j^{J_max}
18:        Discard feature x_j^{J_max} from active.
19:    end while
20:    return Ŝ
21: end function
```

in Python and was made open-source as the Mutual Information based Feature Selection (MIFS) package[124].

Uniquely amongst FS packages, MIFS implements both the JMI and MRMR feature selection algorithms in a parallelised fashion, which makes MIFS scale very well to large datasets. This computational efficiency and the `scikit-learn` like interface made the package fairly popular (at the time of writing almost 60 people starred the project on GitHub) and attracted contributors who provided valuable improvements to the code base.

As with BorutaPy, these information theory based algorithms were studied in great detail during their implementation, which resulted in two significant improvements:

1. Unlike the `FEAST` package, MIFS can calculate MI between continuous variables $X \in \mathbb{R}^{n \times p}$ and a continuous outcome variable $y \in \mathbb{R}^{n \times 1}$. Additionally, by utilising a recently proposed idea in information theory research[125], MIFS can also approximate MI between continuous variables $X \in \mathbb{R}^{n \times p}$ and a discrete outcome variable with C classes $y \in \{0, .., C\}$.

   This latter improvement is crucially important in biomedical research, where the majority of datasets have a discrete, binary outcome variable and therefore MI has to be estimated between continuous features and a discrete outcome.

2. MIFS can automatically select the optimal number of features $k = |\hat{S}|$ by monitoring the decrease in mutual information of the newly selected features $\hat{S} \leftarrow x_j^{J_{max}}$. Once the SG smoothed first derivative of this monotonically decreasing function reaches zero, MIFS terminates.

   Although this is a heuristic approach, it gives the user a rough idea about the size of $\hat{S}$ given their dataset. Once MIFS has found the optimal k_{opt} automatically, the algorithm can be rerun with several nearby values $k_{+1,-1}$, and the resulting sets $\hat{S}_{opt}, \hat{S}_{+1,-1}$ can be compared, by training a model on them on a holdout dataset.

3.2 Further feature selection methods

Since FS is an integral part of CorrMapper, the FS literature was studied extensively and numerous popular methods were tested on a wide variety of simulated datasets. The following subsections list the FS methods which were included in the benchmarking process besides BorutaPy and JMI. Within this section, all methods written in `typewriter font` are from the SL package, unless otherwise stated.

3.2.1 Univariate methods

Haury et al. have found after benchmarking numerous FS methods, that simple filter techniques that are based on univariate statistical tests, work remarkably well with high-dimensional biological data[126]. Inspired by this surprising finding, two univariate FS techniques were included:

- `SelectFDR` function was used with the ANOVA F-statistic (`f_classif`) to select features that passed the FDR[115] correction at $\alpha = 0.05$ threshold.
- `SelectPercentile` function was also used with `f_classif`, to find the top 10% of features based on their F-statistic.

These methods both rely on the F-test as their relevance statistic to rank the features. Subsequently they select $\hat{S}$ based on a cut-off value for either the corrected p-values (`SelectFDR`), or the percentile function (`SelectPercentile`).

3.2.2 Recursive Feature Elimination

Recursive Feature Elimination (RFE) works by selecting smaller and smaller sets of features until a user-defined k number is reached, see Algorithm 3.

Adding cross-validation to this process makes it much more computationally intensive, but removes the need of arbitrarily choosing k. Recursive Feature Elimina-

Algorithm 3 Recursive Feature Elimination

```
function RFE(X^{n×p}, y^{1×n}, C, f, k)
    active ← p
    while |active| > k do
        active ← Features that have not been discarded yet.
        Train a classifier C with X^{n×active} and y to assigns VI to each feature.
        Rank features by their VI, remove fraction f of the worst.
    end while
    return Ŝ ← k selected features.
end function
```

Algorithm 4 Recursive Feature Elimination with Cross-Validation

```
function RFE-CV(X^{n×p}, y^{1×n}, C, f, cv)
    t ← 1
    active ← p
    selected ← empty list
    accuracies ← empty list
    while |active| > 0 do
        Train classifier C on X^{n×active} and y to assigns VI to each feature.
        Remove fraction f of the worst features, based on their VI.
        active ← Features that have not been discarded yet.
        Train classifier C on X^{n×active} and y with cv-fold cross-validation.
        accuracy ← Mean cross-validated prediction accuracy of the classifier.
        selected_t ← active
        accuracies_t ← accuracy
        t ← t + 1
    end while
    k ← arg max_i accuracies
    return Ŝ ← selected_k
end function
```

tion with Cross-Validation (RFE-CV) finds the optimal k based on internal cross-validation rounds[127], see Algorithm 4.

Linear Support Vector Classifier (LSVC) (`LinearSVC`) was used as the learner within RFE-CV, and its C parameter was tuned for each separate dataset using a cross-validated grid-search (`GridSearchCV`) with values spanning the logarithmic scale ranging from 10^{-6} to 10^{3}.

The C parameter acts as a penalty term for the LSVC, setting the amount of regularisation. Larger C values yield smaller margins around the hyperplane that separates the classes, causing the classifier to eventually overfit the data and loose generalisation power. Conversely, small C values will under-fit the data and misclassify certain samples, by using larger margins around the decision-plane.

The weights that LSVC assigns to the features during training could be interpreted as VI measures. Based on these, RFE-CV can prune the least relevant features in each round. RFE-CV was set up to discard $f = 1\%$ of the features in each iteration.

3.2.3 L1 norm based methods

A standard way of estimating the coefficients of a linear regression model is to solve:

$$\hat{\beta} = (X^T X)^{-1} X^T Y.$$

This is the ordinary least squares estimator, which requires the inversion of $X^T X$. However, when $p \approx n$ or even $p > n$, the columns of X cannot all be linearly independent. Therefore, by the invertible matrix theorem, it can be shown that $X^T X$ cannot be inverted. Consequently, $\hat{\beta}$ and the linear regression model is not defined when $p > n$[107].

To alleviate this problem, Tibshirani proposed a novel regularisation method. Applying the L1 norm as a penalty to the linear regression model results in the LASSO regression[101], which unlike the L2 norm regularised ridge regression, is capable of shrinking certain coefficients to exactly zero, hence it performs feature selection at training time[69]. It achieves this by optimising the following objective function

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 ,$$

where the first term on the right hand side is the quadratic loss function of linear regression, while the second term is the regularisation or penalty term using the L1 norm. This ensures that the β coefficients are not getting too large and in fact some of them are shrunk to zero. Consequently, λ is a regularisation parameter that controls the size of the selected set, which is defined as $\hat{S} = \{k : \hat{\beta}_k \neq 0\}$. A large λ will increase the cost function, thus exert more shrinkage on the β coefficients, and hence reduce the size of $\hat{S}$. The optimal λ is dataset dependent and is usually found with cross-validation. In the FS benchmarking, scikit-learn's `LSVC` algorithm was used with L1 penalty and 3-fold cross-validation.

3.2.4 Stability Selection

Stability Selection (SS) is a fairly new ensemble FS method[102], which tries to address two major problems of many FS techniques: high variance of $\hat{S}$ and random selection in case of strong multicollinearity. The first problem means that small perturbations in the training data could lead to different sets of selected features. The second issue occurs when there are many highly correlated predictors in X, and several FS algorithms (particularly the ones penalised with the L1 norm) will randomly select one, while discarding the other correlated feature(s). This might be suboptimal or even erroneous in a biomedical setting, where highly correlated features often originate from the same biochemical pathway.

SS can work with any kind of FS method that has a regularisation parameter λ, like the Lasso regression. It assesses a range of values $\lambda \in \Lambda$, by bootstrap sampling $X^{n\times p}$, N times for each λ. It calculates the fraction of times a given feature was selected across the N bootstrap samples for all $\lambda \in \Lambda$. Finally, it constructs a stable set $\hat{S}_{stable}$, consisting of features that were selected more frequently than a predefined threshold π_{thr}, see Algorithm 5. Scikit-learn's `RandomizedLogisticRegression` was used with $\Lambda = \{10^{-3}, 10^{-2}, ..., 10^{2}, 10^{3}\}, N = 1000,$ and $\pi_{thr} = 0.7$.

Algorithm 5 Stability Selection

```
function SS(X^{n×p}, y^{n×1}, Λ, N, π_thr)
    for each λ ∈ Λ do
        for i ← 1 to N do
            X_sub ← Subsample X without replacement to get X^{n/2×p}.
            Run a FS method on X_sub with regularisation parameter λ to obtain
            a selected set Ŝ_i^λ.
        end for
        Calculate the empirical selection probability for each feature as
        ∏̂_k^λ = P{k ∈ Ŝ^λ} = (1/N) Σ_{i=1}^N I{k ∈ Ŝ_i^λ}
    end for
    Ŝ_stable ← {k : max_{λ∈Λ} ∏̂_k^λ ≥ π_thr}
    return Ŝ_stable
end function
```

3.3 Feature selection benchmarking

3.3.1 Simulated datasets

The benchmarking was carried out on large variety of simulated datasets, using the `make_classification` function of SL. This data simulation algorithm was used to generate the datasets for the famous NIPS FS challenge[128].

`make_classification` generates clusters of points, which are normally distributed around the vertices of a C sided hypercube, where C represents the number of different class labels in y. In this benchmarking experiment, datasets with two classes were simulated, with two sub-clusters within each class, to make them more heterogeneous and realistic. Furthermore, the following parameters were varied to generate a wide variety of classification datasets:

- the number of samples: $n \in \{100, 300, 500, 1000, 2000\}$, to model datasets from relatively small omics studies to large epidemiology cohorts,
- the number of features: $p \in \{100, 500, 1000, 5000, 10000, 20000\}$ to model 16S rRNA studies with a few hundred OTUs, metagenomics studies with around a thousand bacterial genes, and human genomics studies with several thousands of genes,
- the number of informative features: $informative \in \{0.1 \times p, 0.05 \times p\}$.

The number of redundant or highly correlated features was set to $redundant = 0.25 \times informative$ in all datasets. These redundant features are generated as a linear combination of the $informative$ ones.

Each dataset was generated ten times with a different random number seed. Out of the 36 $n \times p$ combinations, 24 were used, which satisfied: ($p \leq 5000$ `or` $n \leq 1000$) condition. This was done to avoid overly large datasets (e.g. $n = 2000, p = 20000$)

which do not represent the typical omics dataset size of today and which would have been computationally very expensive to work with. Given ten random seeds, and two values for the number of informative features, this simulation process resulted in 480 different datasets.

3.3.2 Evaluation process

After extensive literature review, the seven previously introduced FS algorithms were included in the benchmarking. As alluded to in the previous section, the simulated datasets contained redundant features, which were linear combinations of the relevant ones. These redundant features were treated as relevant in these experiments, due to the fact that in biology, strongly correlated features often originate from the same biochemical pathway.

Therefore, even though from an information theoretic viewpoint, two correlated features might carry the same predictive power with respect to the outcome (and hence one of them is redundant), they can be both highly important in understanding the studied biological process.

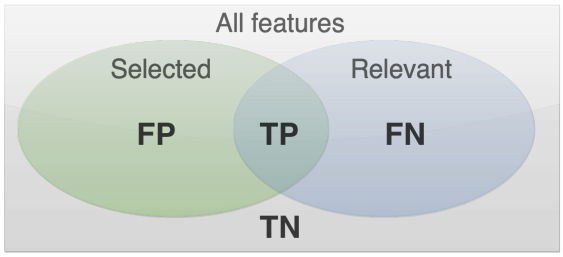


Figure 3.3: Evaluation metrics in FS benchmarking. TP = true positive, TN = true negative, FP = false positive, FN = false negative.

The effectiveness of the seven FS methods was assessed by applying them to each of the simulated datasets, then calculating their precision and recall:

$$prec = \frac{TP}{TP + FP} \qquad rec = \frac{TP}{TP + FN} \qquad \text{Eq. 3.1}$$

Here TP = true positive, TN = true negative, FN = false negative. Figure 3.3 displays these quantities in a FS context. The results of the benchmarking experiment are described in Section 5.2.

3.4 Graphical models

As we have seen in Section 1.5.5, biomedical research is often more interested in the mechanistic understanding of a certain biological condition or disease, than training highly accurate predictive models. Nonetheless, given sufficient samples to learn from (something we rarely have in omics research yet), predictive machine learning models can help with the early diagnosis of ill patients.

However, they cannot shed light on the genetic and biochemical underpinnings of the studied biological phenomenon and consequently they are less useful for knowledge discovery. Conversely, CorrMapper was designed to be a multi-omics data explorer that helps researchers to dissect their datasets and find biologically relevant associations between their features. For this, it uses undirected graphical models to uncover the conditional independence structure of omics datasets.

Correlation networks are a subtype of graphical models, which are routinely used in bioinformatics[129] for enumerating all possible pair-wise associations in biological datasets. In $p > n$ omics datasets however, the naive approach of estimating the covariance between features leads to noisy results with spurious correlations. As this is one of the central problems CorrMapper's pipeline is addressing, in the next sections, a brief introduction is provided to regularised covariance estimation and graphical models in general.

3.4.1 Directed and undirected graphical models

Given a dataset $X \in \mathbb{R}^{n \times p}$, *graphical models* provide a principled way of inferring the relationships between their p features. They were born out of the interplay between probability theory and graph theory, and they provide a framework for building parsimonious models for high-dimensional data. Learning a graphical model from

data requires the simultaneous estimation of a graph structure and of the probability distribution that factorizes according to this graph.

Within these models, the p features of $X \in \mathbb{R}^{n \times p}$ are represented as the graph's vertices or nodes, while the edges between them represent relationships. More precisely $G = (V, E)$, where $V = (V_1, ..., V_p)$ are vertices and E are edges. The edges of G can be described with a $p \times p$ adjacency matrix $E_{(j,k)}$, where a cell is 1 if there is an edge between feature j and k and zero otherwise.

Graphical models have two main types: directed and undirected. Directed graphical models are also called *Bayesian networks*, which are Directed Acyclic Graphs (DAGs). These graphs not only describe a network of associations between features but due to their directed nature, they also represent causal or temporal relationships between the variables.

As a simple example let us consider the directed graphical model of Figure 3.4, which describes how the slipperiness of grass in a garden is affected by both the weather and the sprinkler being on. The joint probability distribution of this system can be factored into the following conditional probability distributions:

$$\mathbb{P}(X_1, X_2, X_3, X_4, X_5) = \mathbb{P}(X_1)\mathbb{P}(X_2|X_1)\mathbb{P}(X_3|X_1)\mathbb{P}(X_4|X_2, X_3)\mathbb{P}(X_5|X_4).$$

Since variables $\{X_2, X_3, X_4, X_5\}$ are binary and X_1 has four values, the exact description of this system's joint probability distribution would require $2^4 * 2^2 = 64$ entries in a table. If we think of a real world problem with hundreds or thousands of variables, it becomes apparent that this approach is untenable.

Fortunately, there is often some structure in most real life phenomena, and not all variables depend on all others, i.e. some of them are independent, or conditionally independent given others. Graphical models exploit this structure, thus making the probabilistic specification of the problem much more compact.

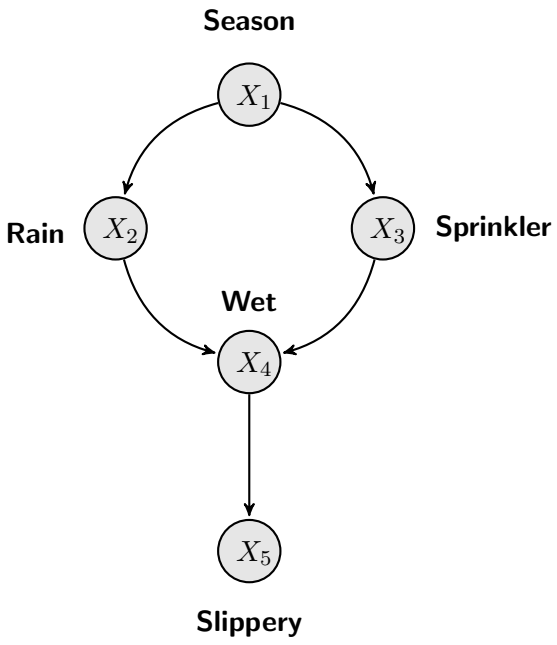


Figure 3.4: Bayesian network example. The graph encodes our intuition about the relationships of these variables: $X_5 \perp\!\!\!\perp \{X_1, X_2, X_3\}|X_4$ and $X_2 \perp\!\!\!\perp X_3|X_1$.

Independence and *conditional independence* are crucial concepts in graphical models. Given three variables X, Y and Z, X is said to be independent of Y or $X \perp\!\!\!\perp Y$ if: $\mathbb{P}(X, Y) = \mathbb{P}(X)\mathbb{P}(Y)$, and X is conditionally independent of Y given Z or $X \perp\!\!\!\perp Y|Z$ if: $\mathbb{P}(X, Y|Z) = \mathbb{P}(X|Z)\mathbb{P}(Y|Z)$.

In the example shown in Figure 3.4, naturally, the season will determine the frequency of rain and the times the sprinkler system is on. These two variables in turn will define when the grass is wet. However, once we know that the grass is wet, we know everything to determine if it is also slippery, and there is no need for us to know the season or the state of the sprinkler. Therefore, the slipperiness is conditionally independent of the season, rain and sprinkler given the wetness: $X_5 \perp\!\!\!\perp \{X_1, X_2, X_3\}|X_4$.

Bayesian networks are very attractive modelling tools as once they are estimated, one can answer very nuanced questions regarding the interactions of the dataset's

features probabilistically. Unfortunately however, the estimation of these directed networks for omics datasets is computationally very expensive as it consists of the following steps[129]:

1. Given p features of a dataset, choose a candidate DAG out of all possible DAG structures.
2. Calculate the posterior probability of the chosen DAG given the omics dataset.
3. Repeat step 1 and 2 to find the DAG with the highest posterior probability.

The difficulty of this Bayesian inference problem is further increased in omics datasets where p is often in the order of several thousand, and therefore the number of potential DAG structures to enumerate are truly enormous. Although many heuristics have been proposed to speed up the search process, the algorithms that estimate the posterior distribution are still based on Markov Chain Monte Carlo (MCMC), which is computationally very taxing.

Undirected graphical models or *Markov networks* merely estimate the association between the p features of X without inferring any directionality. Although this makes undirected graphical models less rich, they are still extremely helpful in identifying relevant associations and cliques of features that are linked through their biological function. Furthermore, undirected graphical models are generally easier to estimate than directed ones.

Undirected graphical models have three main types:

1. marginal correlation networks,
2. partial correlation networks,
3. conditional independence networks.

The ordering of this list represents the increasing complexity of these model types and also the chronological order of their invention. CorrMapper uses conditional independence networks to estimate the associations between features of omics datasets. Since these type 3. models rely on the concepts of marginal and partial correlation networks, in the next sections all three of these undirected graphical model are introduced briefly.

3.4.2 Marginal correlation networks

Marginal correlation networks are constructed by placing an edge between V_j and V_k if $|\rho(j,k)| \geq \epsilon$, where $\rho(j,k)$ is some measure of association. Therefore, the threshold ϵ determines the absolute strength of association that is deemed to be relevant and represented in the network. Importantly, even though $X \perp\!\!\!\perp Y$ implies that $\rho(X,Y) = 0$, in case of many association measure, the reverse is not true.

There are several methods one can use to measure $\rho(j,k)$: different correlation metrics (Pearson, Spearman, Dcorr) and mutual information are commonly used. Pearson correlation is a popular choice as it is easy to and quick compute. Its population and sample estimates are:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}, \qquad r(X,Y) = \frac{\sum_{i=1}^{n}(X-\bar{X})(Y-\bar{Y})}{S_X S_Y}, \qquad \text{Eq. 3.2}$$

where $\bar{X}$ is the sample mean, σ_X is the true population estimate and S_X is the sample estimate of the variance. Pearson correlation is sensitive to outliers, assumes the data is normally distributed and it can only detect strictly linear associations.

Spearman's rank correlation remedies these problems by applying Equation 3.2 to the column-ranked $X \in \mathbb{R}^{n \times p}$. As shown in Figure 3.5, it can capture non-linear monotonous associations, and is more robust to outliers.

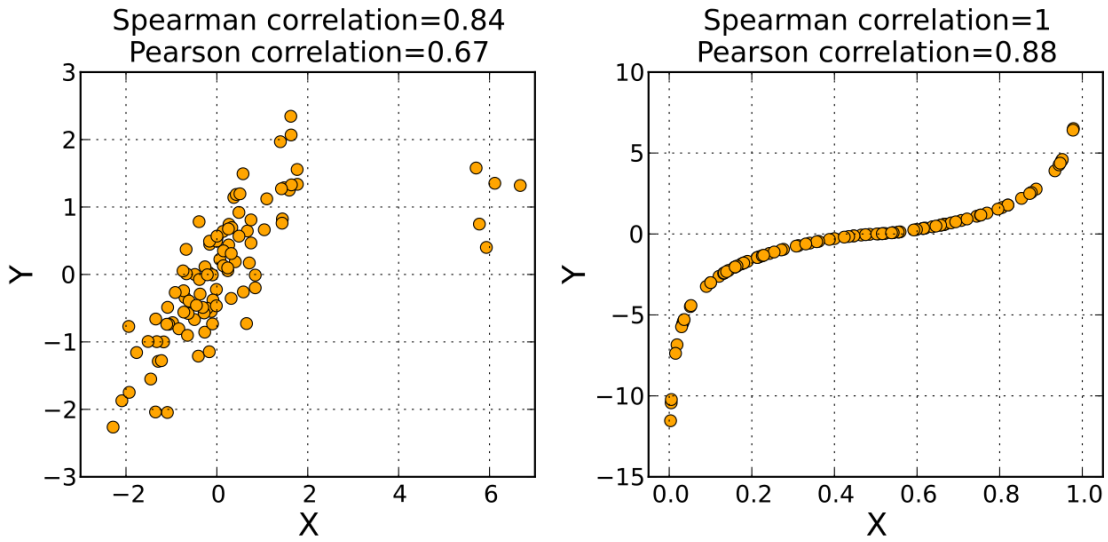


Figure 3.5: Comparison of Pearson and Spearman correlation. The Spearman correlation can capture non-linear monotonous associations and is more robust to outliers. Taken from Spearman's rank correlation Wikipedia page.

Both of these correlation measures range between -1 and 1. If X and Y are perfectly correlated $\rho(X, Y) = 1$, an increase in X always results in an increase in Y. Conversely, if $\rho(X, Y) = -1$, Y is always decreasing when X is increasing. Values between these extremes represent varying strength of association, while $\rho(X, Y) = 0$ means that X and Y are uncorrelated, but not necessarily independent (see later). Furthermore, both of these correlations have well established methods for estimating their null distributions and therefore calculating their statistical significance.

Unfortunately, neither Pearson nor Spearman satisfy the following property: $X \perp\!\!\!\perp Y$ if and only if $\rho(X, Y) = 0$, therefore, variable pairs which are not independent can be missed by these measures. The recently proposed distance correlation[130] (Dcorr) however, guarantees that if $\rho(X, Y) = 0$ then $X \perp\!\!\!\perp Y$. Although this is a very attractive property, Dcorr is a much more expensive to calculate computationally than Pearson or Spearman correlation, and it ranges between 0 and 1, hence does not show directionality. Furthermore, its statistical significance can only be obtained using permutation testing, for which one needs to repeat its moderately slow computation thousands of times for each $\binom{p}{2}$ variable pairs.

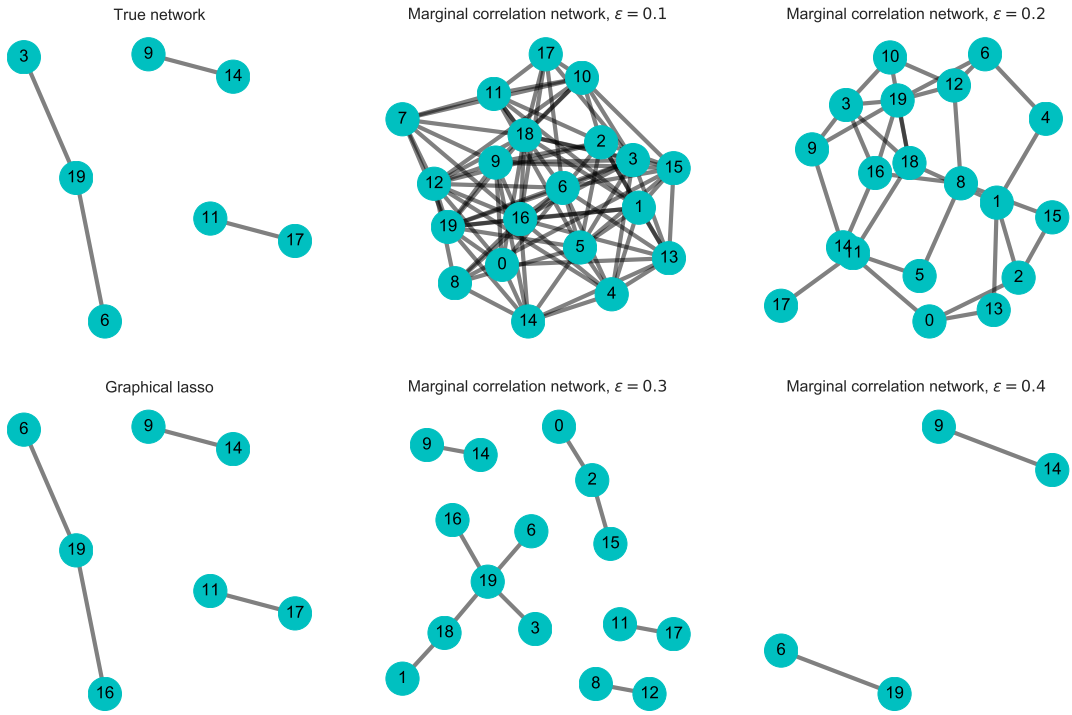


Figure 3.6: Marginal correlation network is sensitive to ϵ. In this example a small simulated dataset $X \in \mathbb{R}^{50\times 20}$ is used to reconstruct an undirected graphical network. Depending on the chosen ϵ, we arrive at different network topologies which can naturally lead to discordant conclusions. Conversely, graphical lasso is able to recover the true network almost perfectly.

Mutual information is an attractive measure of association as it can detect any non-monotonous relationship between two variables and as we have seen in Section 3.1.2, $MI(X,Y) = 0 \iff X \perp\!\!\!\perp Y$. However, MI does not necessarily outperform other correlation metrics[131], and its calculation is moderately expensive. Furthermore, and more importantly, unlike the Pearson or Spearman correlation coefficient, MI is not bound to a range of values and it does not indicate the directionality of association, which makes its interpretation more difficult.

In general, one of the main drawbacks of marginal correlation networks is that the cut-off threshold ϵ is a very sensitive parameter, which is often set arbitrarily. Therefore, several heuristic approaches have been proposed: $\epsilon = 0.95$, i.e. choosing the top 5% of positive and negative correlations[132], or trying a wide range of values till we find a network that displays the scale free graph property[133], something which

seems to be a universal attribute of biological networks[134]. Nonetheless, as we can see in Figure 3.6, the choice ϵ can severely affect the resulting network topology and consequently the conclusions of a study. As we will see in the next section, the graphical lasso often provides a better estimate of the real network.

3.4.3 Partial correlation networks

A common problem with marginal correlation networks is that they can become overly dense and hard to interpret. One of the principle reasons for this is that marginal correlation cannot distinguish partial correlation from real correlation. For example, given the following directed graph: $A \leftarrow B \rightarrow C$, the resulting marginal correlation network would put an edge between each three of these variables. Since $A \perp\!\!\!\perp C|B$, to help with the interpretation of this network, A and C should not be linked.

This problem is made much more severe in $p > n$ cases, such as omics datasets. As shown in Figure 3.6, marginal correlation networks will falsely identify a number of false positive links, which cannot be all eliminated regardless of how we set ϵ.

Given three variables X, Y and a controlling variable Z, partial correlation is defined as the association between X and Y after removing the effect of Z. More precisely $\rho(X, Y|Z)$ is the correlation between R_X and R_Y, where R_X and R_Y are the residuals we get after regressing X and Y onto Z respectively. Given the well-known problem of linear regression, if $\Pi_Z X = \beta^T X$ is the projection of X onto the linear space spanned by Z, where β minimises $\mathbb{E}[Z - \beta^T X]^2$, then the residual of X is given by $R_X = X - \Pi_Z X$.

As shown in Figure 3.7, in case of three variables, R_X and R_Y lie on a two dimensional plane which is perpendicular to Z, and geometrically the correlation between them can be expressed as $\rho(R_X, R_Y) = \cos(\varphi)$.

In a graphical model context, given a dataset $X \in \mathbb{R}^{n \times p}$, $\rho(j,k)$ denotes the partial correlation between x_j and x_k given all other features in X. In other words, $\rho(j,k)$ expresses the association between two features after removing the effects of *all the others*. This makes the interpretation of a partial correlation network much easier and mitigates the problem outlined in Figure 3.6.

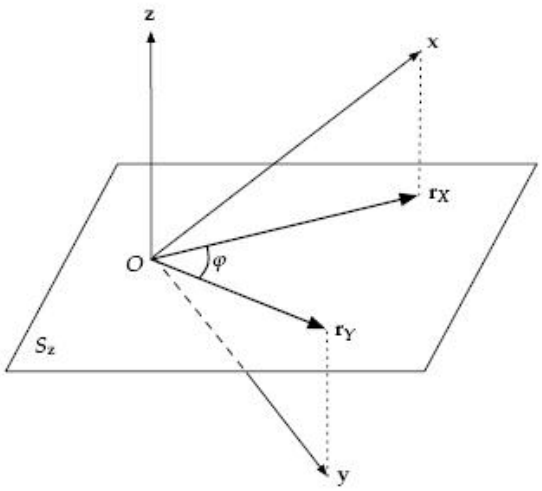


Figure 3.7: Visual illustration of partial correlation. Explanation in text. Taken from Partial correlation Wikipedia page.

Let $R(j,k)$ be a $p \times p$ matrix holding all partial correlations between the p features of X. The entries of this matrix can be estimated as $R(j,k) = -\Theta_{jk}/\sqrt{\Theta_{jj}\Theta_{kk}}$, where Θ is the precision matrix of X, defined as the inverse of covariance matrix $\Theta = \Sigma^{-1}$. The partial correlation graph G has an edge between j and k if $R(j,k) \neq 0$, i.e. if the partial correlation between two features is not zero.

In the low dimensional $p < n$ setting, R can be easily estimated as $\hat{\Theta} = S^{-1}$, where $S = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$ is the empirical covariance matrix of the data. In high dimensional setting however, this does not work, as $\hat{S}$ is rank-degenerate and consequently cannot be inverted.

Ledoit and Wolf proposed a solution to this problem[135], that is shrinking the diagonal elements of the covariance matrix as: $\hat{\Theta} = [(1-\epsilon)S + \epsilon D]^{-1}$, where D is a diagonal matrix with $D_{jj} = S_{jj}$. The authors derived a closed form solution for the optimal ϵ, which makes the use of this method fast and convenient. However, as shown in Figure 3.8, this diagonal shrinkage method is often outperformed by the graphical lasso algorithm. In the example shown in this figure, a small simulated dataset $X \in \mathbb{R}^{60 \times 20}$ is used to estimate an undirected graphical model by several methods. The graphical lasso outperforms both empirical covariance and Ledoit-Wolf estimators.

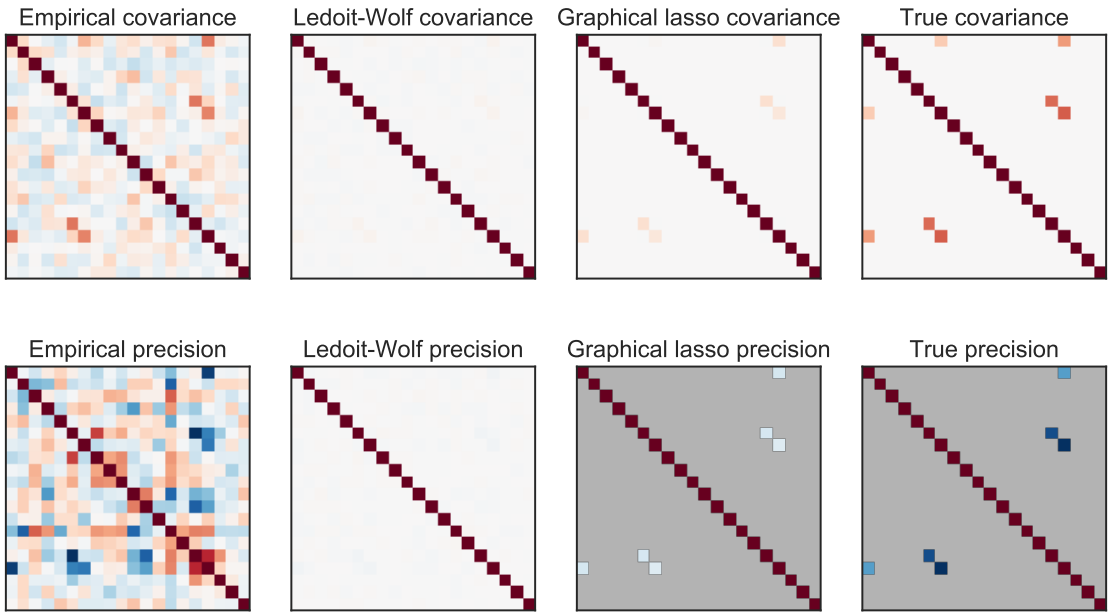


Figure 3.8: Network estimation in three different ways. Small simulated dataset $X \in \mathbb{R}^{60\times 20}$ is used to estimate an undirected graphical model by several methods. The true covariance and precision matrices of the simulated data are displayed on the right hand side of the figure. Further strengthening the points made previously (see Figure 3.6), we observe that the empirical covariance estimate shows a lot of spurious connections which would not be easily filtered out by changing the cut-off ϵ. This problem would become even more severe in $p > n$ datasets. The Ledoit-Wolf estimator shrinks the covariance matrix too much and therefore fails to recover the true connections in the network. Finally, the graphical lasso does a good job at recovering the true covariance structure and network topology of the data. This example was adopted from the `scikit-learn` documentation.

Graphical lasso algorithm

The graphical lasso algorithm[136] assumes that our p-dimensional data is normally distributed $X \sim \mathcal{N}(\mu, \sigma)$ with $\mu \in \mathbb{R}^p$ mean vector and $\Sigma \in \mathbb{R}^{p\times p}$ covariance matrix. In order to represent the multivariate normal as a graphical model, it is convenient to re-parametrise its classical definition[50]:

$$\mathbb{P}_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\},$$

which then becomes:

$$\mathbb{P}_{\gamma,\Theta}(x) = \exp\left\{\sum_{s=1}^{p}\gamma_s x_s - \frac{1}{2}\sum_{s,t=1}^{p}\theta_{st}x_s x_t - A(\theta)\right\},$$

where $\gamma \in \mathbb{R}^p$ and $\Theta \in \mathbb{R}^{p\times p}$ are the so-called canonical parameters, and $A(\Theta) = -\frac{1}{2}\log\det\left[\Theta/(2\pi)\right]$ so that $\int \mathbb{P}_{\gamma,\Theta}(x)dx = 1$. Any multivariate normal distribution with a strictly positive definite Σ can be represented in this form. The main benefit of this parametrisation is that it allows us to infer the graph factorisation properties of our data X directly from the sparsity patterns of Θ. More precisely, if X factorises according to graph G, we can be sure that $\Theta_{st} = 0$ for any pair $(s,t) \notin E$, i.e. G will only have edges where the precision matrix is non-zero.

Graphical lasso solves the graph selection problem and estimates Σ hence $\Theta = \Sigma^{-1}$ using a penalised log-likelihood method. The rescaled log-likelihood of a multivariate normal distribution is defined as:

$$\mathcal{L}(\Theta, X) = \frac{1}{n}\sum_{i=1}^{n}\log\mathbb{P}_{\Theta}(x_i) = \log\det\Theta - \mathrm{trace}(\hat{S}\Theta),$$

where the log-determinant function is defined on the space of symmetric matrices as:

$$\log\det(\Theta) = \begin{cases} \sum_{j=1}^{p}\log(\phi_j(\Theta)), & \text{if } \Theta \succeq 0 \\ -\infty, & \text{otherwise,} \end{cases}$$

where $\phi_j(\Theta)$ is the jth eigenvalue of Θ. The classical maximum-likelihood estimate Θ_{ML} converges to the true precision matrix as $n \to \infty$. However, as we have seen before, in the $p > n$ setting, the sample estimate of Σ ($\hat{S}$) is rank-degenerate and cannot be inverted, and hence this method will not lead to a solution.

Therefore, we need to introduce some form of regularisation to find Θ. Unfortunately, adding the L0 penalty to the log-likelihood estimator results in a highly non-convex optimisation problem[50]. The L0-norm is defined here as the number of

non-zero entries in Θ. Therefore, the graphical lasso uses the L1 penalty instead, and estimates the precision matrix as:

$$\hat{\Theta} \in \underset{\Theta \succeq 0}{\arg\max} \left\{ \log\det\Theta - \mathrm{trace}(\hat{S}\Theta) - \lambda \sum_{j \neq k} |\theta_{jk}| \right\}, \qquad \text{Eq. 3.3}$$

where the penalty term $\sum_{j \neq k} |\theta_{jk}|$ is simply the L1-norm of the off-diagonal entries of Θ. As the left side of this formula shows, we are looking for a strictly semi-definite precision matrix. This can be formulated as a convex log-determinant optimisation problem, which is well studied, and can be solved using the block coordinate descent algorithm[136].

As in lasso-regression, λ is a critical parameter of graphical lasso, which controls the amount of L1 penalty in the objective function, thus determining how sparse the precision matrix and resulting network will be. There is no closed form formula for estimating the optimal value for λ, therefore, it is generally chosen using K-fold cross-validation: we fit the graphical lasso model using a range of λ values on $K-1$ part of the data while evaluating the negative log-likelihood on a held-out K partition. This procedure is repeated K times for each value of λ. Then we choose the λ which minimises the average negative log-likelihood across the K folds. Unfortunately this method was shown to overfit, and as we will see in the next section, there exist more robust approaches for the selection of λ[137].

It is a fortunate coincidence that the sparsity assumption of the graphical lasso algorithm is not only a technical necessity which allows us to solve the graph finding problem in the $p > n$ scenario, but this sparsity condition is also adequate for biological networks in general, since it reflects their scale-free property, where some nodes have numerous edges but the majority of nodes only has a degree of less than four[134].

3.4.4 Conditional independence networks

Conditional independence networks represent the strongest type of undirected graphs as they place an edge between x_j and x_k if $x_j \perp\!\!\!\perp x_k | X_{\setminus\{j,k\}}$. In case of normally distributed data $X \sim \mathcal{N}(\mu, \Sigma)$, this definition for graph construction gives us the graphical lasso algorithm. However, in real datasets, normality is often an unrealistic assumption.

Liu et al. weakened this assumption by replacing the multivariate Gaussian with a semi-parametric Gaussian copula[137]. Copulas are essential tools in multivariate statistics, as they allow us to reconstruct the joint distribution of a complex multivariate system from its marginal distributions, provided we know how those variables interact. Sklar's theorem[138] provides the theoretical foundations for this remarkable statistical fact. It states, that any multivariate Cumulative Distribution Function (CDF):

$$F(x_1, ...x_p) = \mathbb{P}(X_1 \leq x_1, ..., X_p \leq x_p)$$

of a p-dimensional random variable $X = (X_1, ..., X_p)$ can be expressed in terms of its marginal cumulative distribution functions $F_i(x) = \mathbb{P}(X_i \leq x)$ and a copula C:

$$F(x_1, ...x_p) = C(F_1(x_1), ..., F_p(x_p), \theta). \qquad \text{Eq. 3.4}$$

Copulas combine the marginal probability distributions $F_i(x)$ into a joint probability distribution, using the dependence structure θ between the variables. The inputs of copulas are uniform distributions: in Equation 3.4, each input variable to the copula $F_i(x)$ is uniformly distributed, as each random variable X_i is sent through its own CDF. Therefore, copulas map from a p-dimensional unit cube to a single joint probability

$$C : [0, 1]^p \rightarrow [0, 1].$$

In the work of Liu et al., given a p-dimensional non-normal random variable $X = (X_1, ..., X_p)$, it is marginally transformed by p functions to get

$$f(X) = (f_1(X_1), ..., f_p(X_p)),$$

where $f(X)$ is multivariate Gaussian $f(X) \sim \mathcal{N}(\mu, \Sigma)$. This is a non-parametric extension of the multivariate normal, which the authors call the *nonparanormal* distribution, and write $X \sim \mathrm{NPN}(\mu, \Sigma, f)$. Although the transformation functions f_i, vector of means μ and covariance matrix Σ has to be estimated from the data, we get to represent a rich family of non-normal distributions this way, whose independence structure can still be estimated from their precision matrix as $\Theta = \Sigma^{-1}$.

If the f_i-s are monotone and differentiable, the joint probability density of $X \sim \mathrm{NPN}(\mu, \Sigma, f)$ can be written as:

$$\mathbb{P}_X(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(f(x)-\mu)^T\Sigma^{-1}(f(x)-\mu)\right\}\prod_{i=1}^{p}|f_i(x_i)|. \qquad \text{Eq. 3.5}$$

Liu et al. showed, that as a corollary of f_i-s being monotone and differentiable, this joint probability distribution of the nonparanormal could be expressed as a Gaussian copula:

$$\begin{aligned}
F(x_1, ..., x_p) &= C(F_1(x_1), ..., F_p(x_p)) \\
F(x_1, ..., x_p) &= \Phi_{\mu,\Sigma}(\Phi^{-1}(F_1(x_1)), ..., \Phi^{-1}(F_p(x_p))) \\
&\Downarrow \\
C(u_1, ..., u_p) &= \Phi_{\mu,\Sigma}(\Phi^{-1}(u_1), ..., \Phi^{-1}(u_p))
\end{aligned}$$

where $F(x_1, ..., x_p)$ is the joint CDF of the multivariate normal, F_i is the marginal CDF of the the ith random variable, $\Phi_{\mu,\Sigma}$ is the multivariate Gaussian CDF, and Φ^{-1} is the quantile function of the univariate Gaussian.

Furthermore, the authors also proved that if $X \sim \text{NPN}(\mu, \Sigma, f)$, and each f_i is differentiable, then $x_j \perp\!\!\!\perp x_k | X_{\setminus\{j,k\}}$ if and only if $\Theta_{jk} = 0$, where $\Theta = \Sigma^{-1}$. In other words, once we transform the marginal distributions of our variables as $h_i = \Phi^{-1}(F_i(x))$, any multivariate distribution can be expressed as a nonparanormal through the Gaussian copula above, and hence the graphical lasso can be applied to the empirical covariance matrix of $h(X)$.

The marginal transformation of each variable only needs to be done once before the application of the graphical lasso. Given a p-dimensional non-normal dataset with n samples $X \in \mathbb{R}^{n \times p}$, a natural choice for the estimator of F_i in $h_i(x) = \Phi^{-1}(F_i(x))$ is the empirical marginal distribution function:

$$\hat{F}_i(t) = \frac{1}{n} \sum_{j=1}^{n} \{X_i^{(j)} \leq t\}. \qquad \text{Eq. 3.6}$$

However, since the variance of $\hat{F}_i(t)$ is too large in high dimensional datasets, Liu et al. recommend the truncated or Winsorised version of it, which is defined as:

$$\hat{F}_i(x) = \begin{cases} \delta_n & \text{if } \hat{F}_i(x) < \delta_n \\ \hat{F}_i(x) & \text{if } \delta_n \leq \hat{F}_i(x) \leq 1 - \delta_n \\ (1 - \delta_n) & \text{if } \hat{F}_i(x) > 1 - \delta_n, \end{cases} \qquad \text{Eq. 3.7}$$

where

$$\delta_n = \frac{1}{4n^{1/4}\sqrt{\pi \log n}}.$$

According to the authors, this choice of δ_n provides the right bias-variance balance, so that we can achieve the desired rate of convergence in our estimate of Θ.

In summary, each random variable X_i in X is transformed marginally by applying Equation 3.6 and 3.7 to it, then obtaining $h_i(x) = \Phi^{-1}(\hat{F}_i(x))$ using the quantile function of the univariate Gaussian. Then $h(X) = (h_1(x_1), ..., h_p(x_p))$ is used to

compute the sample covariance matrix $\hat{S}$ of the normalised variables. Finally, $\hat{S}$ can be plugged into the graphical lasso algorithm to obtain $\Theta = \Sigma^{-1}$ and hence the conditional independence network G.

Importantly, if the data is truly normal, this marginal transformation will have negligible effects and will not harm the accuracy of the graph reconstruction. On the other hand, by marginally transforming variables to obtain a nonparanormal distribution, allows us to apply the graphical lasso algorithm to non-normal datasets, and estimate conditional independence networks from them.

Robust estimate for the regularisation parameter λ

CorrMapper uses the `huge` R package to apply the nonparanormal extension of the graphical lasso to omics datasets. The optimal value of the regularisation parameter λ (see Equation 3.3) is found using the Stability Approach to Regularization (StARS) algorithm[139]. As alluded to earlier, λ is a critical parameter of the graphical lasso algorithm, as it determines the overall sparsity of the resulting network and therefore the conclusions one can draw from the graph.

Interestingly, in biomedical research it is often more tolerable to have false positives than false negatives in a reconstructed network. This is because false positives can be relatively easily filtered out in subsequent experiments or by using already available biological knowledge. False negatives on the other hand are highly costly as the omission of a biologically relevant interaction might derail the line of investigation for years. Therefore, the StARS method aims to err on the side of under-regularisation to ensure that the selected network surely contains the true network.

StARS estimates the stability of each edge in the graph by drawing N random samples from X *without* replacement, each with b samples within. For each of these N samples StARS estimates a graphical lasso with a range of $\Lambda = 1/\lambda$ values. Taking

the inverse of λ allows us to talk about regularisation more naturally, as larger Λ values correspond to denser, while smaller ones to sparser networks. For a given value of Λ, StARS estimates the probability $\theta_{st}(\Lambda)$ of having an edge between node s and t, as the average amount of times there is an edge between the two features across N random samples.

Their instability is defined as $\xi_{st}(\Lambda) = 2\theta_{st}(\Lambda)(1 - \theta_{st}(\Lambda))$, which quantifies how often the N graphs disagree on the presence of an edge between s and t. Since $\theta_{st}(\Lambda)$ is an empirical frequency between 0 and 1, we have $0 \leq \xi_{st}(\Lambda) \leq \frac{1}{2}$. The total instability $\hat{D}(\Lambda)$ corresponding to a given Λ, is defined by averaging $\xi_{(.,.)}(\Lambda)$ across all $\binom{p}{2}$ edges.

StARS seeks to find the minimum amount of regularisation that still results in a sparse graph with an edge set that is reproducible under random subsampling. In order to achieve this it defines the optimal regularisation as $\hat{\Lambda} = \sup\{\Lambda : \hat{D}(\Lambda) < \beta\}$, where β is an interpretable quantity with a default value of $\beta = 0.05$. Regularisation is therefore determined by the concept of stability rather than regularisation strength.

The StARS method was demonstrated by its authors to show superior empirical performance on synthetic and real-world test cases, when compared against more traditional methods for choosing regularisation such as K-fold cross-validation, Akaike Information Criterion (AIC)[140] and Bayesian Information Criterion (BIC)[141,142].

Both AIC and BIC have deep statistical roots and are classical model selection tools which can also be applied to graphical models. Given a list of regularisation parameters $\mathcal{G} = \{\Lambda_1, ...\Lambda_K\}$, these measures could be used to score the fitted models corresponding to each Λ, and then compare them against each other to find the most parsimonious one. Both AIC and BIC aim to identify an optimal trade-off between goodness-of-fit (measured by the fitted model's log-likelihood) and model complexity (given by the degrees of freedom).

They are defined as:

$$\text{(AIC)} \quad \hat{\Lambda} = \arg\min_{\Lambda \in \mathcal{G}} \{-2\mathcal{L}(\hat{\Theta}(\Lambda)) + 2d(\Theta)\},$$

$$\text{(BIC)} \quad \hat{\Lambda} = \arg\min_{\Lambda \in \mathcal{G}} \{-2\mathcal{L}(\hat{\Theta}(\Lambda)) + d(\Theta) \cdot \log n\},$$

where $\mathcal{L}(\hat{\Theta}(\Lambda))$ is the log-likelihood of a graphical lasso model with the precision matrix $\hat{\Theta}$ corresponding to a particular Λ. The degree of freedom d is defined as the unique non-zero elements of $\hat{\Theta}$, which cannot be larger than $p * (p-1)/2$. However, despite their deep statistical foundations, these methods do not work well in $p \approx n$ and $p > n$ scenarios. As shown in Figure 3.9, they result in overly dense networks which are hard to interpret. Conversely, StARS is able to recover a good fraction of the true network topology with relatively high precision.

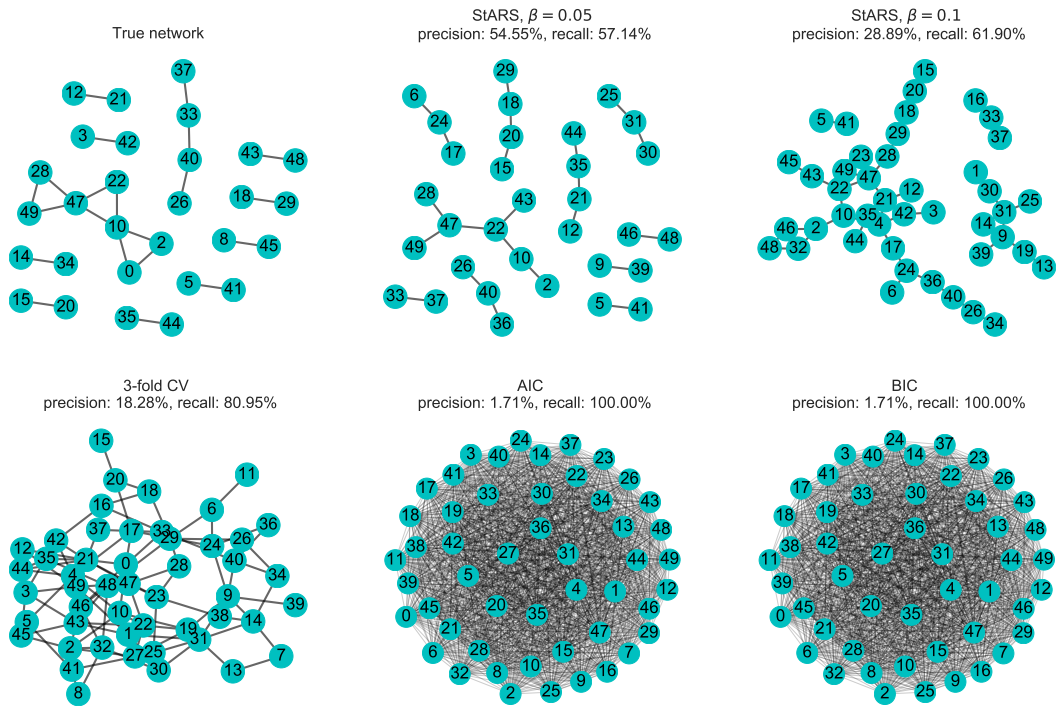


Figure 3.9: Comparison of four methods for choosing λ. The true network of a simulated dataset $X \in \mathbb{R}^{50\times 50}$ is estimated using the graphical lasso algorithm. The quality of recovered networks is assessed by both precision and recall when compared to the true network. AIC and BIC result in overly dense graphs that cannot be interpreted in any meaningful way. Out of the four methods StARS finds a good balance between recall and precision both with default $\beta = 0.05$ and also with $\beta = 0.1$. K-fold cross-validation ends up between the likelihood based methods and StARS both in terms of sparsity and precision.

3.4.5 Estimating statistical relevance

As we have seen in the previous section, CorrMapper uses non-parametric regularised conditional independence networks to estimate the sparse correlation structure between the features of omics datasets. Although this will provide a network topology, it will not inform users about the statistical significance of each edge in network. To alleviate this problem, CorrMapper offers users the option to obtain p-values for each edge, by permutation testing the Spearman rank correlation coefficients of the network.

Permutation testing is a general, non-parametric resampling technique to perform hypothesis testing and obtain p-values for almost any pair-wise statistics. It is most useful when we either do not have a closed form formula for the null distribution of our statistic, or when crucial assumptions of that formula for the null distribution are violated (e.g.: data is not normally distributed).

Permutation testing relies on the very intuitive idea, that if the null hypothesis is indeed true, and there is no association between the two variables, then the pairing of samples should not matter. Therefore, if we shuffle one of the variables and recalculate the statistic numerous times, we should obtain similar values to what we obtained originally with the un-shuffled data.

After the topology of the network is estimated in CorrMapper, given two features x_j and x_k with an edge between them $\hat{\Theta}_{jk} \neq 0$, their Spearman rank correlation $\rho(j,k)$ is calculated as discussed in Section 3.4.2. Then, by randomly permuting one of the two variables and recalculating the Spearman correlation $K = 10^4$ times, we obtain the empirical distribution of the Spearman statistic $\rho_{perm} \in \mathbb{R}^{1\times K}$. From this, an empirical p-value can be calculated as:

$$p_{\text{emp}} = \frac{|\rho(j,k) < \rho_{perm}|}{K}, \qquad \text{Eq. 3.8}$$

where the numerator is the number of permuted Spearman correlations that are more extreme than $\rho(j,k)$. Note, that if $\rho(j,k) < 0$, we have $|\rho(j,k) > \rho_{perm}|$. Although calculating p-values for the Spearman correlation this way is computationally more expensive than applying the closed form Fischer transformation[143], permutation testing automatically takes ties into account i.e. data points that have the same rank. Furthermore, this process is substantially sped up in CorrMapper using linear algebra and parallelisation.

The main drawback of permutation testing is that it ties the minimal obtainable p-value to the number of permutations K. Therefore, it requires a very large number of permutations if sufficiently small p-values are to be accurately estimated. This can be computationally expensive or even infeasible.

For example, given the 20000 protein coding genes of the human genome, obtaining permuted p-values which fall in the range of $10^{-5}-10^{-6}$, for all pair-wise correlations, would require at least $\frac{20000\times19999}{2} \times 10^5$ permutations. Even if we computed 10^5 permutations per second, and we parallelised this procedure over 100 CPUs, it would take 23 days to finish. Furthermore, as we would be performing $\approx 2 \times 10^8$ statistical tests, our type-I error rate would be substantially inflated and we would find numerous spurious correlations with significant p-values[144].

This problem is addressed in four separate ways within CorrMapper:

1. Feature selection ensures that our feature space is substantially reduced prior to the network estimation, and we only deal with variables which are biologically relevant to our investigation.

2. In agreement with our current knowledge of biological networks, CorrMapper assumes that the estimated conditional independence graph has to be sparse. Consequently, the number of edges for which we need to estimate p-values is substantially reduced.

3. The problem of requiring large K for small p-values stems from the rarity of extreme values. Since in permutation testing we are recalculating our statistic between shuffled variable-pairs, the overwhelming majority of Spearman correlations in ρ_{perm} will be around 0. Only very few correlations in ρ_{perm} will be above 0.5 or below -0.5. Consequently, our empirical null distribution will capture the true null distribution fairly accurately around 0, but it will get progressively worse as we move to its tails. As we can see from Equation 3.8, this is precisely the region of the empirical null distribution we need for accurate small p-values.

   This problem can be alleviated by approximating the tail of the empirical null distribution of ρ by fitting a Generalised Pareto Distribution (GPD) to the most extreme values of ρ_{perm}[145]. Once the GPD is fitted, we can use its probability density function to obtain p-values for $\rho(j, k)$. The authors of this method demonstrated that the number of sufficient permutations could be drastically reduced using this extreme value approximation technique, hence $K = 10^4$ in CorrMapper's pipeline.

4. After obtaining p-values for all edges in the recovered network, CorrMapper corrects them for multiple testing using one of the following methods (as requested by the user on the analysis page, see Section 5.3.4):

   (a) Bonferroni correction[146] is the oldest and simplest method to control the Family-Wise Error Rate (FWER). Given the definitions in Table 3.2, FWER $= \mathbb{P}(FP \geq 1)$, i.e. the probability of making one or more type-I errors when performing m hypothesis tests. For a given rejection threshold α, the Bonferroni correction is defined as:

   $$\text{reject } H_i \text{ if } p_i \leq \frac{\alpha}{m}.$$

   It is clear that as m grows to a few thousand, we require the p-values

to be extremely small in order to be deemed significant. Therefore, the Bonferroni correction is known to be underpowered and too stringent for life sciences datasets[114].

(b) Benjamini-Hochberg (BH) FDR[115] controls the False Discovery Rate (FDR). This is defined as (FDR) $= \mathbb{E}[FP/R]$, i.e. the expected number of false positives among the rejected hypotheses. It is less stringent than the Bonferroni, and it provides greater power for slightly elevated type-I error rate. Given a set of hypothesis tests $H_0, ..., H_m$ and their p-values $P_0, ..., P_m$, we list them in ascending order and denote them as $P_{(1)}, ..., P_{(m)}$. BH FDR is a step-wise procedure which iterates through the ordered p-values to find the largest k such that

$$P(k) \leq \frac{k}{m}\alpha.$$

Then it rejects hypotheses $H_{(i)}$ for $i = 1, .., k$. The BH FDR procedure assumes that the tests are all independent.

(c) Benjamini-Yekutieli FDR[147] is an improved version of BH FDR, which controls FDR under dependence assumptions. The threshold introduced above is modified to

$$P(k) \leq \frac{k}{m \cdot c(m)}\alpha,$$

where $c(m) = 1$ if the tests are positively correlated, and $c(m) = \sum_{i=1}^{m} \frac{1}{i}$ in case of negative correlation.

	Null is true (H_0)	Alternative is true (H_1)	Total
Test declared significant	FP	TP	R
Test declared non-significant	TN	FN	$m - R$
Total	m_0	$m - m_0$	m

Table 3.2: Overview of the multiple correction problem. TP: true positives, FP: false positives, TN: true negatives, FN: false negatives, m: number of tests, R: number of tests that are deemed significant.

3.5 Software development

The majority of this PhD was dedicated to the research and development of CorrMapper. In its current state, CorrMapper's code-base has more than 25,000 lines of code, combining 5 different languages: Python, R, JavaScript, HTML, CSS. It is predominantly a Python project, which is reflected in that it relies on 83 different Python libraries. CorrMapper's visualisation modules combine well known plot types in a novel and interactive way, while its modular data integration pipeline consist of 18 sub-modules. CorrMapper's open-source code-base is hosted on GitHub[148].

The reusable part of CorrMapper's front-end was further developed into ScienceFlask, an open-source template for scientific web applications. ScienceFlask is built with Python Flask, HTML and Bootstrap CSS. It is well documented and modular, therefore it can be used easily by other researchers to speed up the development of online scientific tools. Its architecture is introduced in Chapter 4.

CorrMapper's back-end is a robust data integration pipeline, that can work with any pairing of omics datasets. It uses cutting-edge feature selection methods, followed by regularised covariance estimation to uncover biologically relevant variables and their interaction networks. CorrMapper uses the standard data science libraries of Python: `numpy, scipy, scikit-learn, pandas, statsmodels, seaborn`. Chapter 5 describes the software architecture of CorrMapper's back-end.

Finally, CorrMapper's three advanced data visualisation modules can automatically generate complex dashboards from clinical metadata, display the intricate correlation networks of hundreds of features and visualise whole genomes in an intuitive manner. All three modules were developed using modern visualisation libraries of JavaScript: `d3.js, d3plus.js, dc.js, crossfilter.js, datatables.js`. Chapter 6 introduces these visualisation modules and their technology in great detail.

4 | ScienceFlask

4.1 Overview

Creating a secure, scalable, and modular online scientific tool requires the mastery (or at least working knowledge) of several programming languages and web development paradigms. Consequently, turning CorrMapper into an online research tool, brought up several unforeseen technical challenges that required the learning of a wide range of software skills. To ensure that other projects can benefit from this effort as well, CorrMapper's front-end architecture was developed into a template, to enable other researchers to turn their offline scientific tool or algorithm into an online web application.

ScienceFlask (SF) is a template project for scientific web applications, written in Python. It is built using modern, open-source web technologies such as Flask, SQLAlchemy, HTML5, JavaScript and Bootstrap Cascading Style Sheets (CSS). SF implements most of the basic components of an online scientific tool such as user management, admin panel, file upload, job submission, form validation and logging.

Since most of these components are reusable across projects, SF can substantially reduce the time needed to turn an offline prototype into a publication ready online tool. The source code of SF and a highly detailed tutorial about deploying a SF based project into the cloud is available at the project's GitHub repository[149]. Additionally, to showcase the various components of SF, an extensively documented and fully functional example project is available at https://scienceflask.com.

4.1.1 Why templates are useful?

Although most modern websites run on open-source software, the costs of developing an online scientific tool can be overwhelming. The reason for this becomes evident once we factor in the hundreds of hours one has to spend on learning various programming languages and web-technologies to build a cloud based research tool.

Due to this steep learning curve and because researchers are notoriously overworked, a lot of scientific tools are never made available online. This can reduce the visibility and utilisation of valuable projects, especially when the target audience is less willing to compile from source and use a tool locally from the command line. There are several benefits to providing an online interface for a scientific research tool: users are generally more willing to use it as no set-up or installation is required and computation is outsourced to the provider of the application, while the results could be accessed later from any machine as they are hosted in the cloud.

Unfortunately, even if a researcher has the necessary skills, the development of standard website functions and features which are not related to the scientific tool's algorithmic core, can take much longer than anticipated. These often overlooked but crucial parts involve: secure user management, admin panel for maintaining the application's database, robust form validation, uploading and checking of data files, asynchronous job submission and execution, logging, design of user interface and deployment to production server. SF provides all of these components and therefore can substantially reduce development time.

Although numerous bioinformatics pipelines are available as web applications, to my knowledge, there is no flexible, extendible and open-source template available for the rapid prototyping and development of such tools. Hopefully an active community will build around SF, improving its code-base and adding new features to it. SF is released under the GNU GPLv3 licence[150] as an open-source project.

4.2 Front-end architecture

Figure 4.1 demonstrates the simple idea behind SF. Everything in blue is non-specific to any given scientific web application. Therefore, these components and processes could be reused across projects. Since SF provides these, scientists do not need to work on anything else but the core algorithmic part of their tool, shown in green.

SF consists of two main parts: front-end and back-end. The majority of SF's code-base focuses on the front-end which implements all functionalities that are related to what users can see in their browser. The back-end components communicate with and execute the core algorithmic part of the scientific tool. Although SF is written mainly in Python, the core of a SF project can be written in any language.

This core can be plugged into SF with a few hours of work, reducing the development time by several orders of magnitude. Furthermore, by reusing the same open-source components across different projects, we can enhance the transparency of scientific

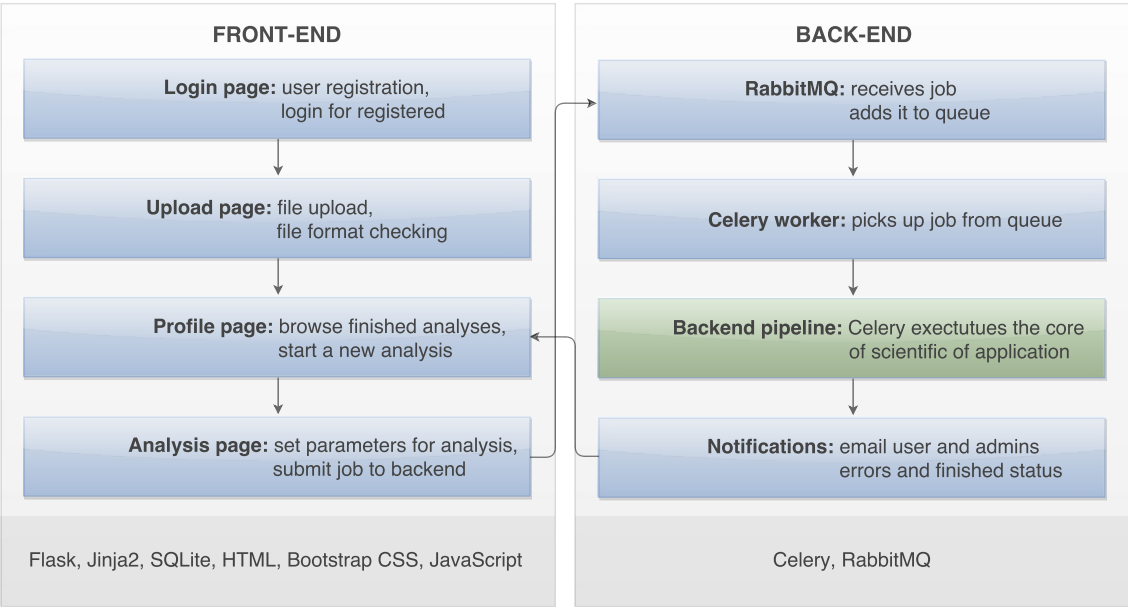


Figure 4.1: Architecture of ScienceFlask. All components and processes in blue are general, reusable parts of any online scientific tool. The core algorithmic part of most science tools (green) can be wrapped in SF's components. This reduces development time, and standardises code-bases across projects, which leads to better reproducibility.

tools, which in turn leads to easier code review process and better reproducibility.

The front-end of SF was mainly developed using Python's Flask[151] micro web-framework, which is highly modular and flexible, yet its minimalistic clean design makes it easy to learn. Like most modern web-frameworks, Flask greatly simplifies building web-services and handling HTTP request in a programmable, clean way.

Applications built in Flask follow the popular Model View Controller (MVC) design pattern. Within this paradigm, models describe the data tables and their relationships which ultimately define how the application stores and accesses its data. Views represent dynamically built web-pages that the web-service returns as a response to the user's request. Finally, the controller is our application's brain, which receives requests (e.g. a Uniform Resource Locator (URL) address) from the user's browser. Firstly, the controller parses the requested URL by matching it to its predefined list of routes. Once the route is found, the controller will follow the route specific build function to collect all necessary data from the models and build a view, i.e. a web-page which is then returned to the user's browser.

A schematic overview of the MVC pattern is shown in Figure 4.2. In this example, a user requests her profile page. Once Flask receives and parses the URL, it will match it against its routes. When it finds the build function of the profile page

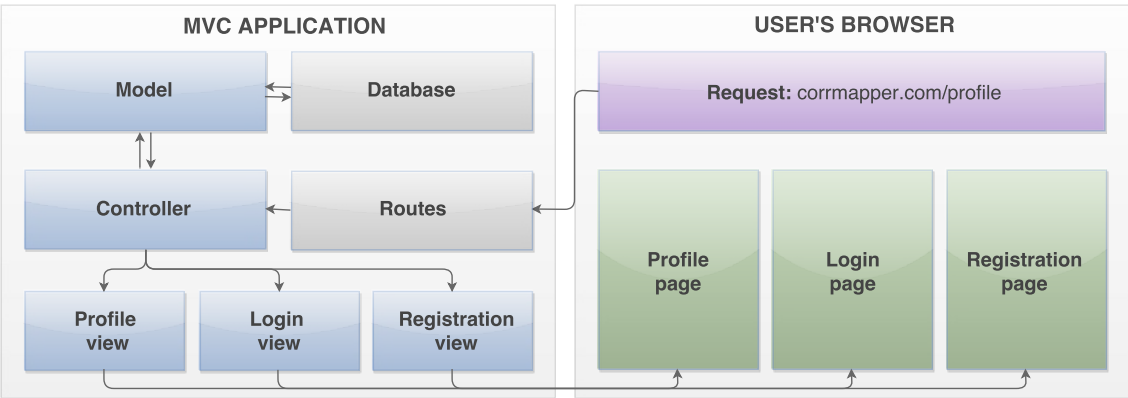


Figure 4.2: Model View Controller web-service pattern. A user requests her profile page. Based on whether she has registered and logged in the controller returns a different HTML page, which is built with the Jinja2 templating engine, using information from the database.

it will communicate with the model and retrieve the user's details from the `User` database table. Since the profile page displays a mix of information (name, email address, previously uploaded studies) the controller will also need to retrieve data from the `Studies` database table for this particular user. The model knows that users can have studies and therefore the `User` and `Studies` tables are linked, which consequently simplifies complex queries like this.

The retrieved pieces of information are combined to build a HTML web-page, which is returned to the user's browser. This web-page is built by taking the HTML template of the profile page, and injecting the user specific database fields into it. For this, Flask uses the Jinja2 templating engine, that allows developers to define scaffold pages with programmable placeholders, which could be filled in from the database.

The HTML pages of SF use Twitter's Bootstrap CSS library to make them responsive and therefore accessible on any device, including mobile phones and tablets with smaller resolution. Bootstrap also greatly simplifies the designing and layouting of websites, by providing nice looking and well-thought-out visual components.

Although the example in Figure 4.2 looks trivial, the application also needs to know if the request came from a user who is already registered and logged in. If not, the controller will need to return either a registration or login page. Web-frameworks allow developers to build logic like this in a systematic and programmable way, while keeping the code modular and reusable.

The model layer of SF is built using the SQLAlchemy library, which allows developers to use almost any relational database (MySQL, SQLite, PostgreSQL, Microsoft SQL, Oracle) without changing a single line of code in the definition of the application's models. By default, SF uses SQLite, which is a lightweight, single-file relational database. However, this could be changed very easily to any of the above mentioned flavours of SQL, by simply changing a single parameter in SF's configuration file.

4.3 Implemented components

SF implements a hierarchical data model: users can upload multiple studies, and each of those studies can have several analyses, each corresponding to a different set of pipeline parameters. To achieve this, SF implements a list of components that are essential for any online service, such as secure user management and a database layer, but it also provides numerous features to developers that are specific to scientific web applications.

User management: SF implements a registration process that is tailored to scientific tools. Users need to register with a valid email address from one of the ten thousand recognised academic institutions. This ensures that commercial use is restricted unless the administrator willingly adds a non-academic user to the application's database. However, this restrictive registration process can be easily disabled if needed, to allow everyone to sign up.

The user management is handled by the Flask-Security package, which allows the tracking of logged in users across the website (remember, at each URL point the application needs to know if a request is coming from someone who is already registered and logged in or not). Furthermore, Flask-Security allows users to reset or change their passwords safely. Finally, it stores the passwords of users after cryptographically hashing them, to ensure that not even the application's administrators can access them.

Upload page: Once registered, users can upload multiple data files to the server through a highly customisable form. The upload form is validated both in the client's browser using JavaScript and on the server side, where complex, application specific logic could be applied to the form's fields. For instance, we can ensure that a certain type of file can only be uploaded if another field were checked by the user.

To minimise the chance of subsequent runtime bugs, the uploaded files are only saved on the server if they pass a series of sanity checks. By default, these include checking if all cells of every uploaded data matrix are numeric, void of non-numeric characters and if the data files are within the application's dimension and file-size limits. The list of these checks could be easily extended to fit the project's needs. If a file does not pass any of these tests, the user is immediately notified and given a clear and informative error message on the upload page. Additionally, the uploaded files are deleted from the server to save disk space.

Profile page: Once saved, the files can be accessed by the user from their profile page, along with their account information, see Figure 4.3. Here, users can submit a job to the server using any of their uploaded datasets or reset their password. Furthermore, all previously finished analyses are listed at the user's profile page. The results of these runs can be downloaded as a .zip file or explored online in SF.

Analysis page: If users click any of their studies at the profile page, they are taken to the analysis page. By filling out a form, users can set parameters of the back-end pipeline, before submitting their job to the server. As with the upload page, server-side form validation ensures that all submitted values are sensible and correct according to the application's logic.

Following the submission of an analysis, the job is handed over to the back-end for processing. Users can leave the page at this time and an email notification will be sent to them once SF has finished running their job.

Bioinformatics pipelines are usually highly customisable. This flexibility is preserved in SF by exposing the pipeline's parameters to the user through the analysis page. Since SF saves these variables to a database for each submitted job, valuable data will accumulate over time, about the usage patters of the scientific tool. This data can inform future development directions or answer questions related to a project's necessary hardware configuration.

Asynchronous job handling: Once an analysis has been submitted by a user, it is passed to the back-end of SF where the core algorithmic part of the scientific tool is applied to the uploaded dataset with the specified parameters. At this point, the standard request-controller-model-view flow of the MVC paradigm needs to be broken, as users do not want to wait for minutes or hours till their analysis finishes, for a response HTML page. To break out of the MVC flow, SF uses RabbitMQ[152] as a message broker to communicate between the Flask application and Celery[153], which is an asynchronous job queue engine.

Celery runs a pool of workers which can pick up jobs from a queue, execute them and return the results. The job queue grows as different users submit their analyses to SF simultaneously. If all workers of the pool are already engaged, newly submitted jobs are added to the queue in order of the submission time.

The computational resources of each worker can easily be limited to ensure that the complete pool will never exceed a set amount memory and CPU usage. By default, in SF, Celery runs in the background as a daemon process on the same server where the application is hosted. Alternatively, one can host SF on one server, and delegate all computational load to another server. Both RabbitMQ and Celery are highly customisable, industry standard solutions that are routinely used by thousands of companies in production systems, handling millions of tasks daily.

Admin interface: SF uses Flask-Admin to automatically build a fully functional admin interface from the models of the application. Once the administrators are specified with their email addresses in SF's configuration file, their profile pages are extended with a link to the admin interface. Here, they can perform Create, Read, Update, Delete (CRUD) operations on any of the application's database tables.

Since this is a web based interface, the administrators can access the database and thus monitor their application's usage at any time, irrespective of where they are in the world. Furthermore, they can easily add new users who did not go through the

academic registration process, or delete users who violated any of the application's usage policies.

Notifications: Once an analysis has finished running, the user is notified via email. Conversely, if during the run of an analysis a bug is encountered, both the user and administrators are notified. The user receives an apologetic email, ensuring her that the developer team will debug the problem, while the administrators receive a detailed error message (see logging).

Furthermore, if an analysis has failed to finish because of the uploaded dataset or the chosen parameters did not yield any meaningful results, SF emails the user with a detailed explanation, that clarifies which part of the pipeline has failed, what is the most likely cause of this, and how the problem could be avoided when resubmitting the analysis.

Logging: SF comes with several levels of logging. All warning and error messages are kept and saved on disk, both from the Flask application and the Celery worker pool. This way, if an analysis does not finish or a bug is discovered, it is relatively easy to track down the source of failure. Furthermore, SF saves the data files of failed analyses so administrators can reproduce bugs easily.

By default, SF sends an email to all administrators if it encountered an error. The level of severity can be easily lowered however, to also notify administrators of warning or even information level messages.

Configuration: SF's configuration file allows developers to fine-tune much of the inner workings of their application from a single location. Some of the more obvious settings involve credentials to the email sending server or the configuration of Celery and RabbitMQ. But SF developers can also specify the maximum allowed file size and maximum dimensions of the uploaded datasets, or the maximum number of studies and analyses any user can have.

4.4 Example application

A demo application demonstrating each of the above described components and functionalities of SF is available at https://scienceflask.com. This simple scientific tool allows user to register with their academic email address and upload one or two numeric data files. The application then selects a user defined number of features with the highest variance from each dataset.

Next, it calculates the Spearman correlation matrix between these selected variables with their corresponding p-values. The correlation and p-value matrices are visualised as heatmaps, which can be either downloaded as a .zip file from the user's profile page or explored online in SF.

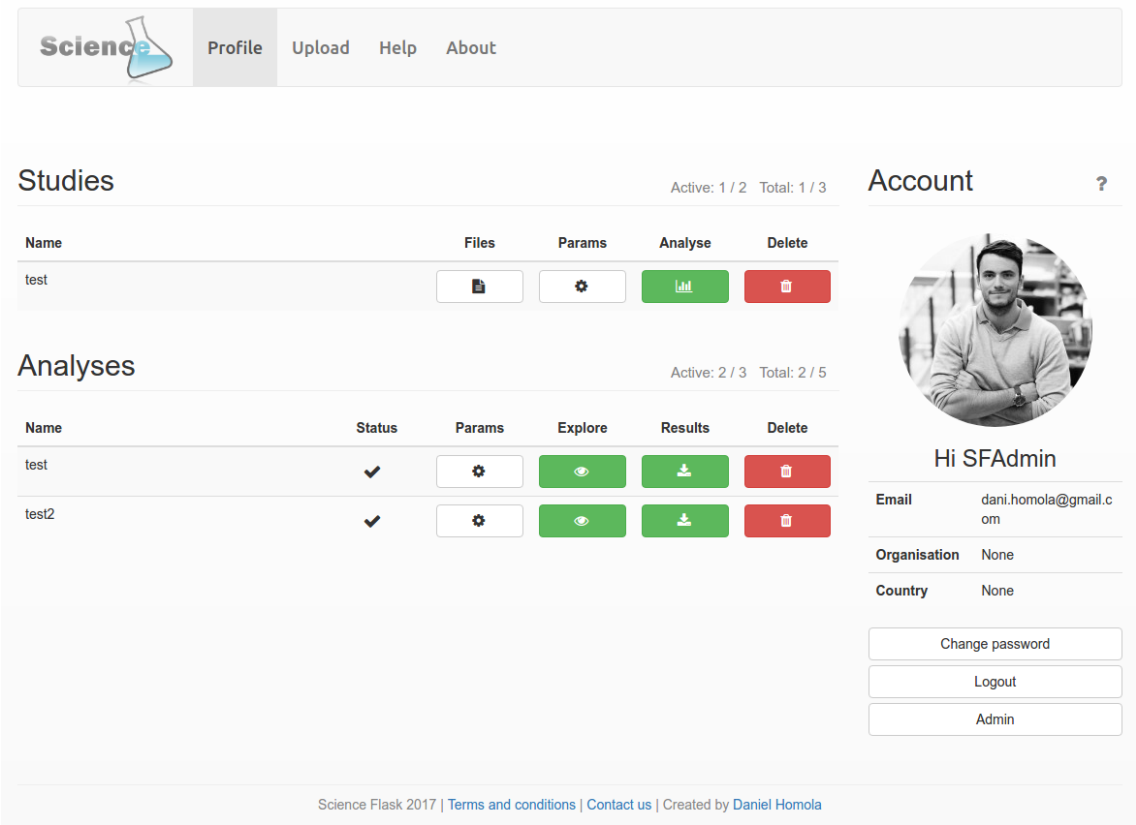


Figure 4.3: Profile page of ScienceFlask. Users can access their uploaded datasets and start new analyses based on each of them, while the results of previously run analyses are also accessible and downloadable from here. For the application's admin interface is also accessible for users with administrator rights.

4.5 Deployment notes and project documentation

With the proliferation of cost-effective computing (Google Cloud, Amazon AWS, Microsoft Azure), an increasing number of services and applications are deployed to cloud servers of these vendors. Infrastructure as a Service (IaaS) providers like Amazon can cut down hardware and production costs substantially: instead of the painstaking purchase-provision-install-maintain process that used to be the norm in server management, one can get a fully managed server or cluster with guaranteed up time and cutting edge software solutions for a fraction of the price.

However even in this current age of computational abundance, one of the most overlooked aspects of web application development is the deployment process. It can take a surprising amount of work and learning to transform a fully functioning application that is running in test mode on a laptop, into a live web-service.

To alleviate this problem, a deployment tutorial is available at the GitHub repository of SF, that explains in great detail, how to set up a secure web-server on Amazon's Elastic Compute (EC2) cloud. The notes provide help on every aspect of the deployment process: from signing up at Amazon AWS and starting a new EC2 instance, through obtaining SSL certificates for a secure connection between users and the application, to setting up and configuring an Apache web server.

Beyond these deployment notes, detailed documentation of the project's structure and overall architecture is also provided on SF's GitHub repository. These notes and the open-source nature of SF will hopefully encourage other scientists and developers to contribute to the project and make it more robust, feature rich and appealing to a wide researcher audience. At the time of writing, more than 70 people have starred the project on GitHub, which is a promising start.

4.6 Summary

Even though there exist numerous excellent free resources and tutorials online about the development of web-applications, the learning curve of the various technologies involved remain steep. Therefore, numerous scientific projects that would benefit from an online presence remain in the form of a distributed software package, which restricts their potential user base to researchers with programming skills.

The Galaxy project[154] provides fantastic resources for developing and publishing reproducible bioinformatics work-flows, which can be built from hundreds of pre-defined tools found in the Galaxy Tool Shed. Alternatively, new algorithms can be wrapped to be used in Galaxy and shared with the research community.

However, Galaxy is specialised for biological tools and as an enormous software project, it requires a significant amount of learning from new developers who plan to adapt and publish their algorithms within its ecosystem. Furthermore, there are limits to what can be incorporated into Galaxy as a new tool. For instance, advanced interactive visualisations are not supported currently.

ScienceFlask is an open-source, modular web-application template that was created to facilitate the rapid prototyping and deployment of online scientific tools. Owing to the detailed project description and deployment notes, researchers can turn their offline algorithms into an online research application within a few days.

ScienceFlask uses Python's Flask web framework along with numerous packages from the Flask ecosystem which are all well documented and carefully explained in countless online tutorials. Furthermore, ScienceFlask relies on SQLAlchemy to ensure that the platform can be used with any SQL like relational database. The simple, modular structure of the project guarantees that it is flexible, easy to extend and unlike larger projects such as Galaxy, it can be learned quickly.

ScienceFlask implements numerous front-end components that are essential for any scientific application: registration process, admin panel, user management, file upload, profile page, job submission page, logging, job queue and email notification system. At the same time it provides complete freedom for developers to build the core algorithmic part of their application in any language.

To my knowledge, there exists no open-source software solution or template that provides the same set of functionalities as ScienceFlask, therefore this work has been submitted for publication to Bioinformatics as an application note.

5 | CorrMapper - data integration

5.1 Overview

CorrMapper is an online research tool for the integration and visualisation of complex biomedical and omics datasets. It is available at www.corrmapper.com and www.medbio.imperial.ac.uk/corrmapper. As an open-source project, released under the GNU GPL 3 licence, its code-base is freely accessible, hosted on GitHub[148].

Since it is a multifaceted research tool, it will be described in two separate chapters. This chapter is introducing CorrMapper's data integration pipeline and the front-end components that support it, while Chapter 6 describes CorrMapper's advanced visualisation modules.

This chapter is organised as follows:

- The first section summarises the results of the feature selection benchmarking experiment, and how those influenced the design of CorrMapper's pipeline.
- The second section describes the Upload, Profile and Analysis page of CorrMapper. These are the main sites for user input, and even though their blueprint is based on ScienceFlask, they hold several important modifications that are essential for the understanding of CorrMapper's back-end.
- The third section describes the data integration pipeline in great detail and applies it to simulated data to assess its performance.
- The last section briefly compares CorrMapper to mixOmics, which is its closest "competitor" in multi-omics integration.

5.2 Feature selection benchmarking

As no previous work was found in the literature, which compared all seven of the feature selection methods introduced in Sections 3.1 and 3.2, extensive benchmarking experiments were carried out on 480 simulated datasets. As described in Section 3.3.1, a wide range of sample and feature numbers were simulated to assess how well these algorithms perform with different omics datasets and study types.

As mentioned earlier, CorrMapper is an open-source project. Consequently, for the sake of reproducibility, the `supp/fsTest` folder contains the code to reproduce every single result file of the benchmarking experiment and to collate them into tables and figures.

Figure 5.1 summarises the performance of the seven algorithms. The 480 different datasets were summarised by sample to feature ratios $R = n/p$. The varying sample and feature numbers resulted in 20 different ratios, of which eight are shown in Figure 5.1, while all the others could be examined in the Supplementary materials, see Figure S4, Figure S5 and Figure S6.

As described earlier in Section 3.3.2, precision and recall were used to measure the performance of these FS algorithms. In a FS context, recall measures the ratio of true positives found by the FS method out of all the relevant features, while precision quantifies the ratio of true positives among the features selected by the algorithm, see Figure 3.3. Therefore, an ideal FS method achieves 100% in both measures, meaning it identifies all relevant features but no more.

The error bars (horizontal black lines) in Figure 5.1 represent one standard deviation of the given metric across a number of datasets. The variability summarised by these error bars have several sources. Firstly, each dataset was re-generated with ten different random seeds. Secondly, distinct datasets can have the same R value.

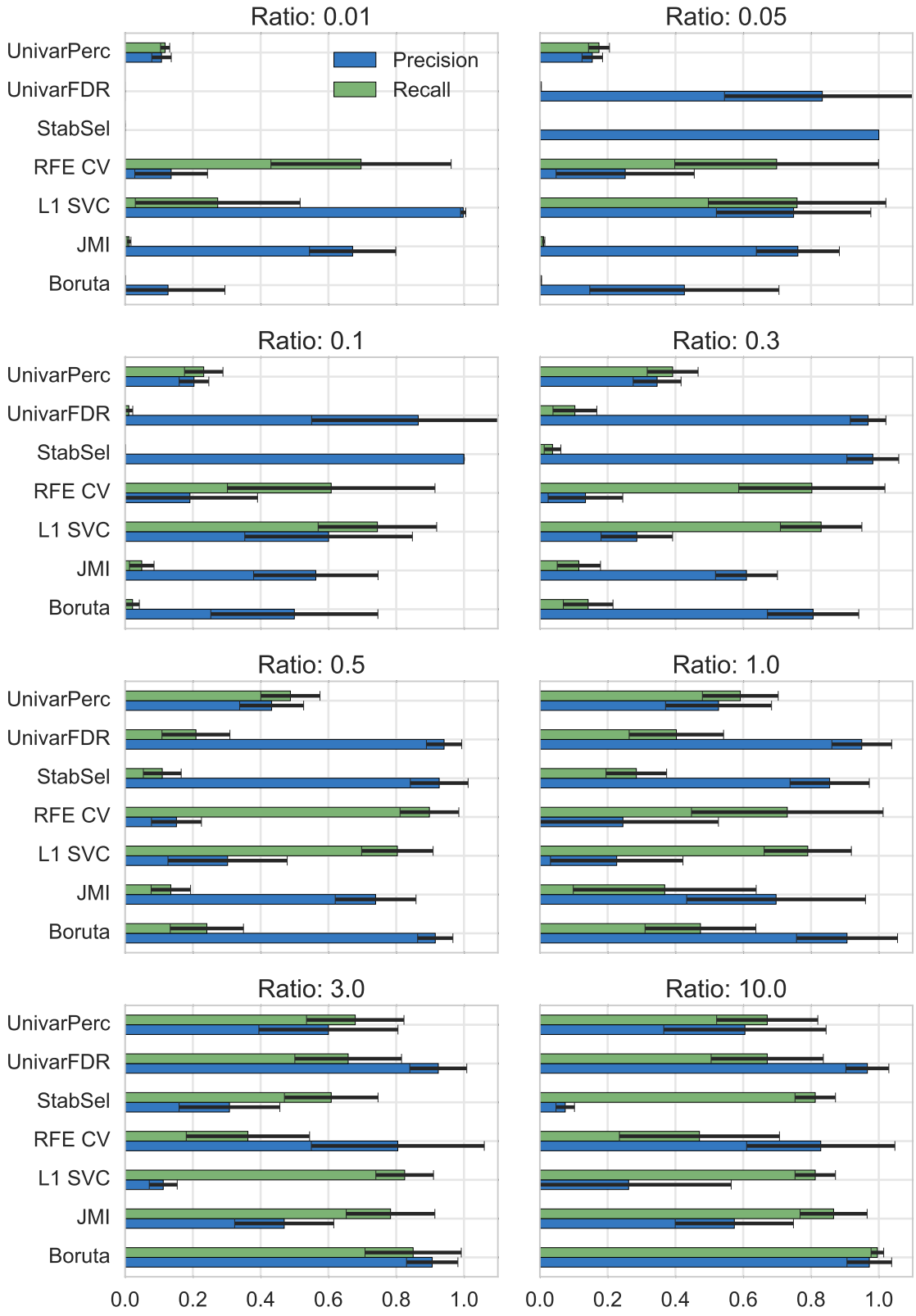


Figure 5.1: Benchmarking results of seven feature selection methods. The title of each sub-plot shows the ratio of samples to features $R = n/p$. Horizontal black lines represent standard deviations stemming from pooling the repeated runs with varying random seeds and different datasets having the same R value.

	UniPerc	UniFDR	RFE CV	L1 SVC	StabSel	Boruta	JMI
Mean R	0.4061	0.4091	0.6570	0.7121	0.2368	0.3519	0.2568
Std R	0.2421	0.2688	0.2884	0.2276	0.2947	0.3885	0.3147
Mean P	0.3633	0.9314	0.2940	0.4665	0.5678	0.6488	0.6914
Std P	0.2407	0.1637	0.3053	0.3464	0.3759	0.3691	0.2105
R + P	0.7694	1.3405	0.9510	1.1786	0.8047	1.0007	0.9482
RS std R	0.0517	0.0738	0.2396	0.1414	0.0323	0.0339	0.0465
RS std P	0.0441	0.1196	0.1318	0.1333	0.0647	0.1233	0.0936

Table 5.1: Summary of feature selection benchmarking results. R: recall, P: precision, Std: standard deviation, RS: random seed. Further explained in text.

For example $X_1 \in \mathbb{R}^{100\times1000}$ and $X_1 \in \mathbb{R}^{500\times5000}$ both have $R = 0.1$. Finally, some FS methods did not select any features from certain datasets. Table S1 in the Supplementary materials summarises for each FS algorithm, how many datasets were collated per ratio to calculate its performance.

Table 5.1 complements Figure 5.1, and presents several key statistics of the benchmarking experiment. The top four rows summarise the mean recall and precision of each FS methods across all 480 datasets, along with their standard deviation.

The fifth row is the sum of mean recall and mean precision, which is a combined measure of these methods' overall performance. The last two rows display the mean of standard deviations across the ten random seeds, i.e. how resilient these methods are to small perturbations of the input data.

There are several points to be made about these results:

- Not surprisingly, the higher R is, the better all algorithms performed, which is demonstrated by higher precision and recall values, coupled with shrinking error bars. However, there are notable differences between the performance of these FS methods: some of them seem to prioritise high recall (RFE CV, L1 SVC), while some of them err on the side of caution and favour high precision (Boruta, JMI, Univariate FDR).

- Despite its highly simplistic nature, the **univariate percentile** method performed surprisingly well on these datasets, but it still had the lowest combined recall and precision out of the seven methods. Furthermore, its percentile threshold was set to 10% in this experiment, which is coincidentally the same as how the number of informative features was defined in half of the datasets (in the other half it was set to 5%, see Section 3.3.1). This certainly helped the algorithm, and probably gave it an unfair advantage, as in real life scenarios, this fraction is an unknown hyperparameter, which can only be roughly estimated from the literature.
- The **univariate FDR** method chooses the number of features to select based on statistical testing, and thus it does not rely on such unmeasurable hyperparameter. Interestingly, and in concordance with previous findings[126], this univariate method performed similarly to more complex algorithms which are much more expensive computationally (Boruta, JMI), and it reached the highest combined precision and recall value. However, in real biological datasets, class membership might depend on the complex multivariate relationship between numerous features. Due to their design, univariate filter methods cannot discover these multivariate relationships, and therefore might perform worse than more complex FS algorithms. Finally, as Table S1 shows, univariate FDR either failed to or did not select any features from most $R < 0.1$ datasets.
- The performance of **stability selection** was mixed. On small R datasets it was very conservative and often did not select any features. As R grew, its recall started to increase, while maintaining relatively high precision. However, once $R > 1$, its precision decreased significantly and it turned into a high recall method which explains why it received the second lowest combined precision and recall score. As many other FS methods, stability selection seems to be sensitive to its hyperparameters, thus it performs best when these are tuned to one particular dataset.

- **Recursive feature elimination** produced high recall, similarly to L1 SVC, but its precision was the lowest among all methods. Furthermore, this method was by far the most sensitive to small perturbations of the input data, see last two rows of Table 5.1.
- Interestingly, the recall of **L1 SVC** method seems to increase as R grows, but this is coupled with the deterioration of its precision. Therefore, similarly stability selection, this method turned from high precision to high recall as R grew. Nonetheless, it had the second highest combined precision and recall.
- It is interesting to note, that with $R > 5$ datasets, most of these algorithms performed quite well and the univariate methods even outperformed the two LSVC based techniques(RFE CV and L1 SVC). These SVM based FS techniques are extremely sensitive to their regularisation parameter C. Although in both algorithms C was chosen automatically using cross-validation, it seems likely that these FS methods were over-regularised in the high R datasets, and with hand-tuned C values they could have achieved better results.
- Overall, the performance of **JMI** ranked in the middle. It produced similar precision and recall values to Boruta in $R < 0.5$ datasets. However, its performance could not match the random forest based method in $R > 1$ datasets. Nonetheless, the Python implementation's parallelised nature and automatic selection of optimal feature number makes it an attractive FS algorithm.
- All methods perform better when they are applied to higher R datasets, but this is especially true to **Boruta**. Once $R \geq 3$, its performance is unparalleled and reaches close to 90% recall and precision. Quite astonishingly, both performance metrics approximate 100% with higher R values, see Figure S6. However, in small R datasets its performance is on par with much simpler and less computationally taxing methods, such as the univariate FDR. On the whole, the only real advantage of Boruta method in $R < 0.5$ datasets is its lack of tunable hyper-parameters, and robustness against untransformed data.

5.2.1 Variance filtering

Since most biomedical studies today fall within the $R < 0.5$ range, it is alarming to see how all of these FS methods struggled with identifying the majority of relevant features in these $p \gg n$ datasets. To alleviate this problem CorrMapper's FS process starts with a variance filtering step. It is a common assumption in data analysis, that features with the highest variance are generally valuable, as it is along their dimensions, where the samples could be separated most successfully[69].

As we will see in Section 5.4, as a first step of crude dimensionality reduction, CorrMapper keeps $2n$ of the p features of uploaded omics dataset. The $2n$ are selected by choosing features with the largest variance. This effectively transforms all $R < 0.5$ datasets into a $R = 0.5$ one. This ratio was chosen as it represents a reasonable trade-off between the following two competing effects. The more features are filtered out based on their variance, the higher R gets, and the better all FS methods perform. But at the same time, the more we filter, the more we risk losing relevant but low variance features.

To assess the performance of the seven FS methods in this scenario as well, a second benchmarking experiment was run. If $2n < p$ was true for a simulated dataset (e.g.

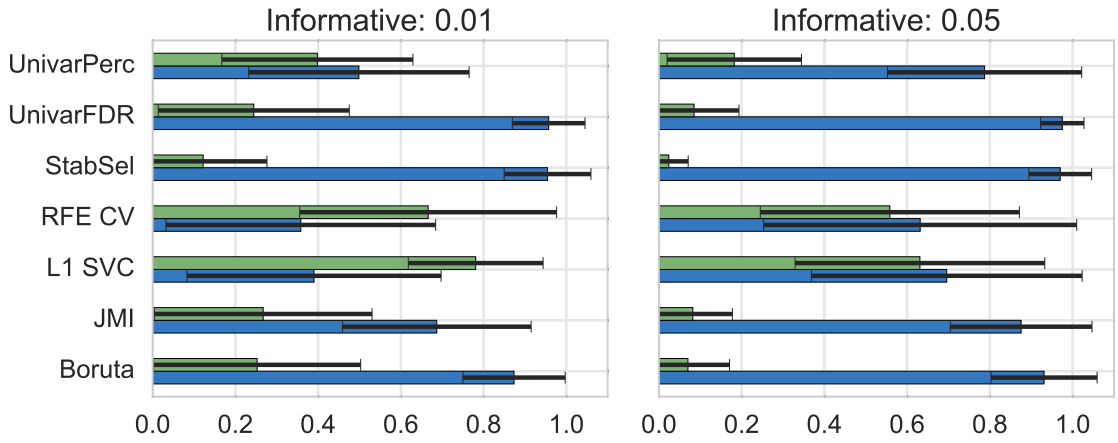


Figure 5.2: Benchmarking results after variance filtering, i.e. forcing $R < 0.5$ datasets to be $R = 0.5$. Blue: precision, green: recall. Number of informative features in the datasets were set to 1% and 5% of all features.

$n = 100, p = 500$), then it was first transformed to only retain $2n$ of its features with the highest variance. Then, the FS experiment proceeded as before. Datasets for which the $2n < p$ condition did not hold were excluded from the second benchmark experiment. This resulted in 260 datasets in total.

Furthermore, the number of informative features was set to be 1% and 5%. The reason for this is that scikit-learn's `make_classification` makes the variance of relevant features (i.e. informative or redundant) larger than that of noise features. Therefore, variance filtering always kept all relevant features in this experiment and only got rid off irrelevant ones. Therefore, had the ratio of informative features been kept at 10%, datasets like $X \in \mathbb{R}^{100\times1000}$ would have ended up with only relevant features after the filtering. This problem is mitigated by these lower percentages.

Figure 5.2 summarises the performance of the seven FS algorithms after variance filtering. As we can see in the left sub-plot, with lower number of informative features all methods achieve higher recall. More importantly however, all algorithms perform similarly to how they did on real $R = 0.5$ datasets, proving that variance filtering can increase the utility of FS methods even in $p \gg n$ datasets.

After carefully considering the results of both benchmarking experiments, univariate FDR, L1 SVC, JMI and Boruta were chosen to be part of CorrMapper's feature selection module. Univariate FDR is a fast and simple FS method that performed admirably and might work well for certain datasets. The performance of L1 SVC was above average around $R = 0.5$, while it is computationally cheaper than RFE CV. However, L1 SVC assumes linearity, which might not hold in biological datasets.

JMI and Boruta are hyperparameter free and can detect complex, non-linear multivariate interactions between features. Although Boruta is computationally the most expensive out of the seven benchmarked methods, it performed exceptionally for $n > p$ datasets. Finally, JMI's mid-tier performance is compensated by its information theoretic background and parallelised nature.

5.3 CorrMapper's front-end

Since ScienceFlask was developed from CorrMapper's front-end, understandably the two resemble each other. But ScienceFlask is more of a template or blueprint, whereas CorrMapper is a functioning scientific research tool. Therefore, the front-end of the latter is full with functionality that is missing from former.

These differences are outlined on the following pages in great detail. Nonetheless, the flowchart in Figure 4.1 depicting the connections between the front-end's components still holds for CorrMapper, therefore it might be useful to revisit it briefly before reading on. Additionally, Figure 5.6 in the next section displays the flowchart of CorrMapper, highlighting the relationship between its front-end and back-end.

5.3.1 Index page and registration

The opening or index page of CorrMapper is highly useful for several reasons. Firstly, it displays a 10 minute long demo video, which showcases CorrMapper's pipeline and visualisation modules with all their interactivity. The reader is strongly encouraged to watch this video before reading further or using the application, as it provides a great overview of the many functionalities CorrMapper has to offer.

Secondly, at the bottom of the page we find links to three visualisation demos, each displaying one of CorrMapper's modules, which are introduced in Chapter 6. These demos can be explored by anyone, without registering or uploading their data.

Finally, even though the required formatting of annotation, metadata and data files is explained in great detail on the Help page, the Chin et al. paper's multi-omics dataset can be downloaded from the index page for examination.

The registration process is identical to the one used by ScienceFlask. Therefore, by

default, only users with a valid academic email address can sign up to use CorrMapper. Thanks to the admin panel described in Section 4.3 however, the administrator of CorrMapper can easily add exceptions to this and let non-academic users access the application.

To help the reader get familiar with CorrMapper's visualisation modules, a guest account was created. This has both the bariatric surgery and the breast cancer datasets uploaded to it. Furthermore, several analyses were run on both of these datasets, the results of which can be accessed from the Profile page. Please use the credentials below to log in with the guest account:

username: guest@corrmapper, password: GraphicalLa$$o

Please note, that due to security reasons, guests are not allowed to upload or delete studies, and they cannot perform new analyses or delete the results of existing ones. Since the guest account can be used by anyone, these functionalities had to be restricted. Nonetheless, guests can explore both the Upload and Analysis pages along with their Profile page.

5.3.2 Upload page

CorrMapper was designed to be used with a wide range of omics and multi-omics study types. As Table 5.2 shows, it can handle eight different kinds of studies, as defined by three binary variables:

1. Multi-omics: does the study have one or two omics data files?
2. Genomic features: do the omics features have a genomic location that can be used to map them onto the chromosomal map of the studied species?

3. Metadata: does the study contain additional information about the samples or patients that could be used for feature selection?

Even though feature selection is at the heart of CorrMapper's pipeline, it is optional and can be disabled by users while analysing their dataset. Consequently, studies without metadata files are also permitted. Based on Table 5.2, the bariatric surgery study introduced in Section 2.1 is a type 7. dataset, while the breast cancer study from Section 2.2 is type 8. These datasets are radically different from a type 1. study for instance, which comprises of a single omics data file. Consequently, CorrMapper's back-end pipeline and upload form needs to reflect and accommodate this variety.

As Figure 5.3 shows, the upload form can seem a bit intimidatingly complex at first, but not all of its fields are mandatory for every study type, and helpful tooltips are located next to each one, which could be displayed by hovering over any of the questions marks.

To make the upload form flexible and adaptable to all eight types of studies, some of its fields and panels only appear once the user clicked a check-box corresponding to one of the above described three binary variables. To make the linkage between binary variables and form panels more explicit, Table 5.2 and Figure 5.3 are cross-referenced through the symbols: $\circ$, $\square$, $\triangle$.

Study type	Multi-omics ($\circ$)	Genomic features ($\square$)	Metadata ($\triangle$)
1.	-	-	-
2.	x	-	-
3.	-	x	-
4.	-	-	x
5.	x	x	-
6.	-	x	x
7. - bariatric surgery	x	-	x
8. - breast cancer	x	x	x

Table 5.2: Study types CorrMapper can analyse. Symbols next to the three binary variables correspond to boxes of the upload form in Figure 5.3.

More precisely, if the "More than one dataset" box is ticked, the fields marked with a ○ will appear.

Similarly, if "My features have genomic location" box is ticked, the "Species" panel will appear, along with the "Annotation file1" and "Annotation file2" fields, as marked by the □ next to them.

Finally, when the "Upload metadata file" box is ticked, the " Metadata file" field appears, as marked by △.

CorrMapper's genomic network explorer (see Section 6.4) provides an excellent data exploration interface for datasets with genomic features. In order to utilise this visualisation module, the ge-

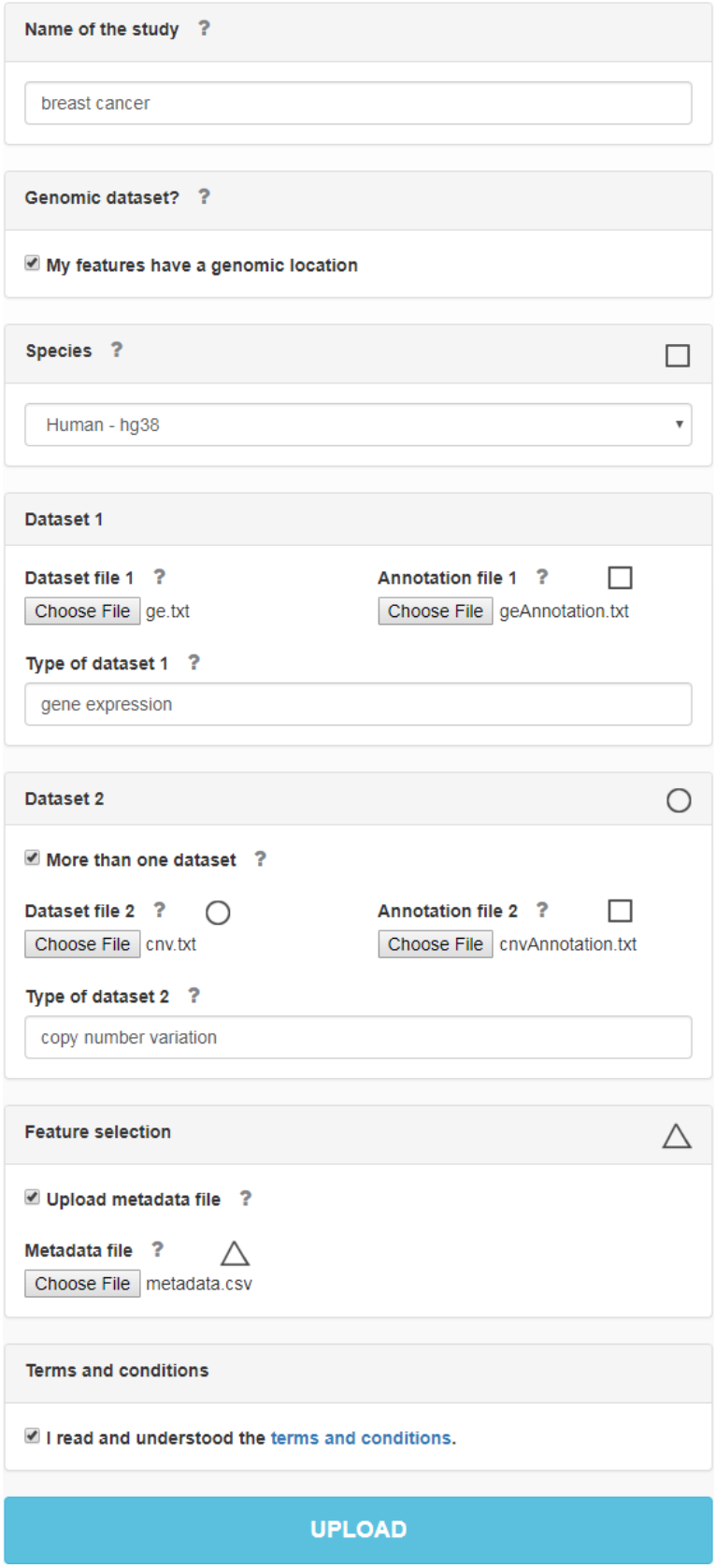


Figure 5.3: Upload form of CorrMapper, filled out for the breast cancer study. Further explained in the text.

nomic features have to mapped onto the studied species' chromosomes. As explained in Section 6.4.1, this process requires annotation files, which clearly define the genomic location of each feature in the omics data files.

As alluded to earlier, the required format of annotation, metadata and data files is explained on the Help page. Alternatively users can download the breast cancer dataset to check its files.

Hovering over the "Upload" button will temporarily disable it while a quick form validation is performed on the server side. This checks that all required files are attached and every necessary field is filled in. Furthermore, the validation process ensures that each file is attached only once, i.e. to one field, and that its size is within the permitted 100MB.

Finally, if all fields have passed the validation process, the "Upload" button becomes active again and once clicked, it initiates the file transfer. Once the files are uploaded, their format is immediately checked before being saved on the server. Section 5.4.1 describes this format validation process in detail. If any of the files fail to pass the validation, the user is informed with clear and explicit error messages which pop up right next to the problematic file. Conversely, if all the files are correctly formatted, the user is taken to her profile page.

5.3.3 Profile page

The profile page is similar to what we have seen in ScienceFlask, and it lists all the studies and analyses a user has uploaded or run. At the top, users can browse their uploaded studies and do one of the following:

- Hovering over the "Files" icon, a tooltip pops up, displaying the name of the study's uploaded files. This is especially useful, if the same study is uploaded

several times using different preprocessing steps. In such a scenario the files can be named accordingly to differentiate between the various versions.

- Hovering over the gear icon of the "Params" column will display the three aforementioned binary variables of the study.
- If a study has metadata, clicking the "Explore" button will generate an interactive dashboard that maps the metadata variables onto the lower dimensional representation of the omics data. Section 6.2 introduces and showcases this interface.
- Clicking the "Analyse" icon will redirect the user to the Analysis page, where parameters of CorrMapper's data integration pipeline can be specified (see Section 5.3.4), before submitting a new job to the cluster.
- The "Delete" button is disabled in the guest account for security reasons. As its name suggest however, clicking this button will delete a study with all of its corresponding analyses. This action cannot be undone.

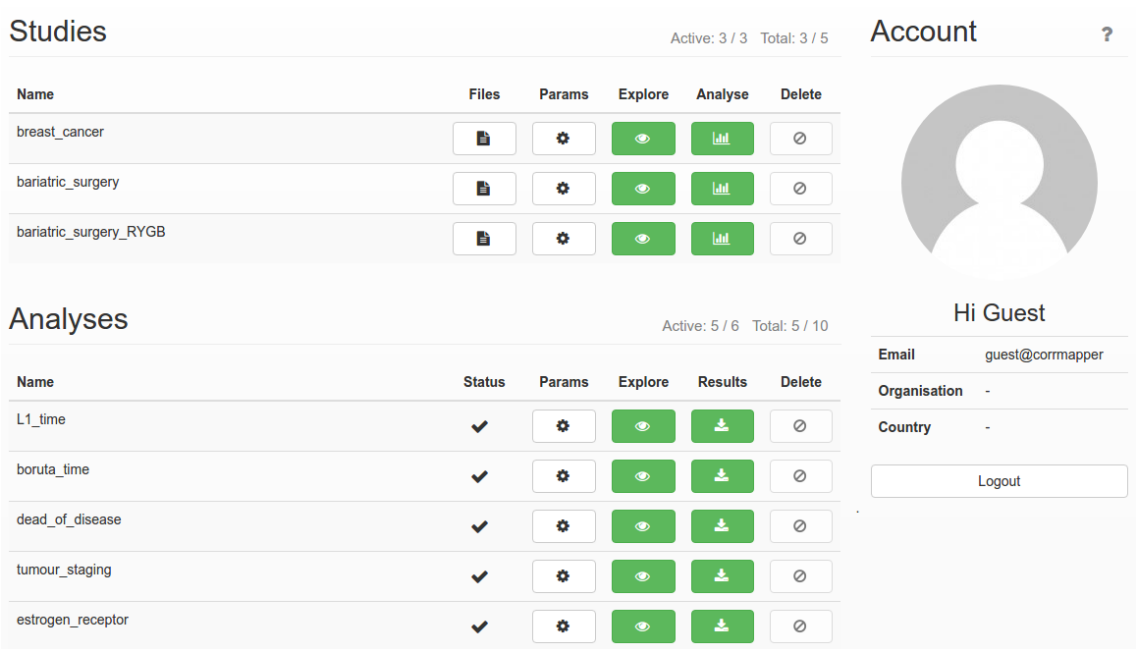


Figure 5.4: Profile page of the guest account. For security reasons guest cannot delete studies and analyses or change the account's password. Apart from these features however, the guest profile page is identical to a regular user profile page.

Below the “Studies” panel we find the finished analyses of the user. This panel offers the following functionalities:

- The “Status” column displays whether an analysis has finished running or not, in which case the tick is replaced with an hourglass icon.
- Hovering over the gear icon of the “Params” column displays the exact parameters that were used within CorrMapper’s pipeline to produce the results.
- Clicking the “Explore” button will redirect the user to the general network explorer. This interactive visualisation module is introduced in in Section 6.3.
- Clicking “Results” icon will offer all the resulting files and images of the analysis for download as a .zip archive.
- The “Delete” button is disabled in the guest account for security reasons, but in regular accounts it deletes a finished or running analysis. This latter option can be useful, as CorrMapper does not allow users to run multiple analyses simultaneously. Therefore, if a job is taking too long to finish, users can easily terminate it and start another analysis with more restrictive parameters.

The guest account has three uploaded studies, as the bariatric surgery dataset was uploaded twice: once with all samples, and once restricted to only contain patients who underwent RYGB type surgery. Furthermore, the account has five precomputed analyses, whose results will be discussed in Chapter 6.

The right hand side of the interface summarises the account’s information. At the bottom of this panel users can log out or change their passwords. This latter feature is disabled for the guest account for security reasons. Clicking the question mark at the top right corner of the “Account” panel, will start an interactive tour of the profile page interface.

To be mindful about the resource usage (disk space, memory, processing power) of CorrMapper on its host server, users are currently limited to have a maximum

of four uploaded studies at any given time and five altogether. Furthermore, they are allowed to store the results of six analyses at any given time, and perform ten jobs altogether. However, these limits can be easily changed both globally in CorrMapper's configuration file, and individually by altering the database through the admin panel. Finally, users can easily check their quota at the top right corner of the "Studies" and "Analyses" panels.

5.3.4 Analysis page

The analysis page allows users to set the parameters of CorrMapper's data integration pipeline before submitting their job to the server. Due to the different study types enlisted in Table 5.2, the analysis form has to be flexible, just like the upload form is. To stay consistent, the symbols used in Section 5.3.2 ($\circ$, $\square, \triangle$) are also used here in Figure 5.5. Furthermore, similarly to the upload form, hovering over the question marks will display tooltips with lots of useful and technical information about each parameter.

If the study has a metadata file, the "Feature selection" panel ($\triangle$) will appear. The top drop-down menu allows users to select one of CorrMapper's four feature selection algorithms, while the bottom one sets one of the metadata columns as target variable. As we will see in Section 5.4.1, each metadata variable has to pass a set of sanity checks in order to appear in this list. For example, in case of a categorical variable, CorrMapper requires at least 15 samples per each of its levels.

Although it is not recommended, feature selection can be skipped in CorrMapper's pipeline. This will result in much larger and denser networks, which are a lot more difficult to interpret. Moreover, CorrMapper uses JavaScript libraries for data visualisation which rely on CPUs and cannot utilise the power of graphics cards. Consequently, the fluid and seamless visualisation of such networks with thousands

of nodes requires top of the range multi-CPU machines. Finally, in an unfiltered network, potentially hundreds of features might need to be assessed, which are not necessarily relevant to the clinical outcome variable we are interested in.

The "Network selection" panel allows users to alter $\hat{\Lambda}$, i.e. the threshold in StARS. The default value ($\hat{\Lambda}$=0.1) represents the authors' recommendation.

The next panel sets the method for correction of multiple testing, as explained in Section 3.4.5. Very importantly, this step of the data integration pipeline, represents an additional filtering process of the network's edges that some researchers might find unnecessary. Therefore, it can be disabled, by setting $\alpha = 1$.

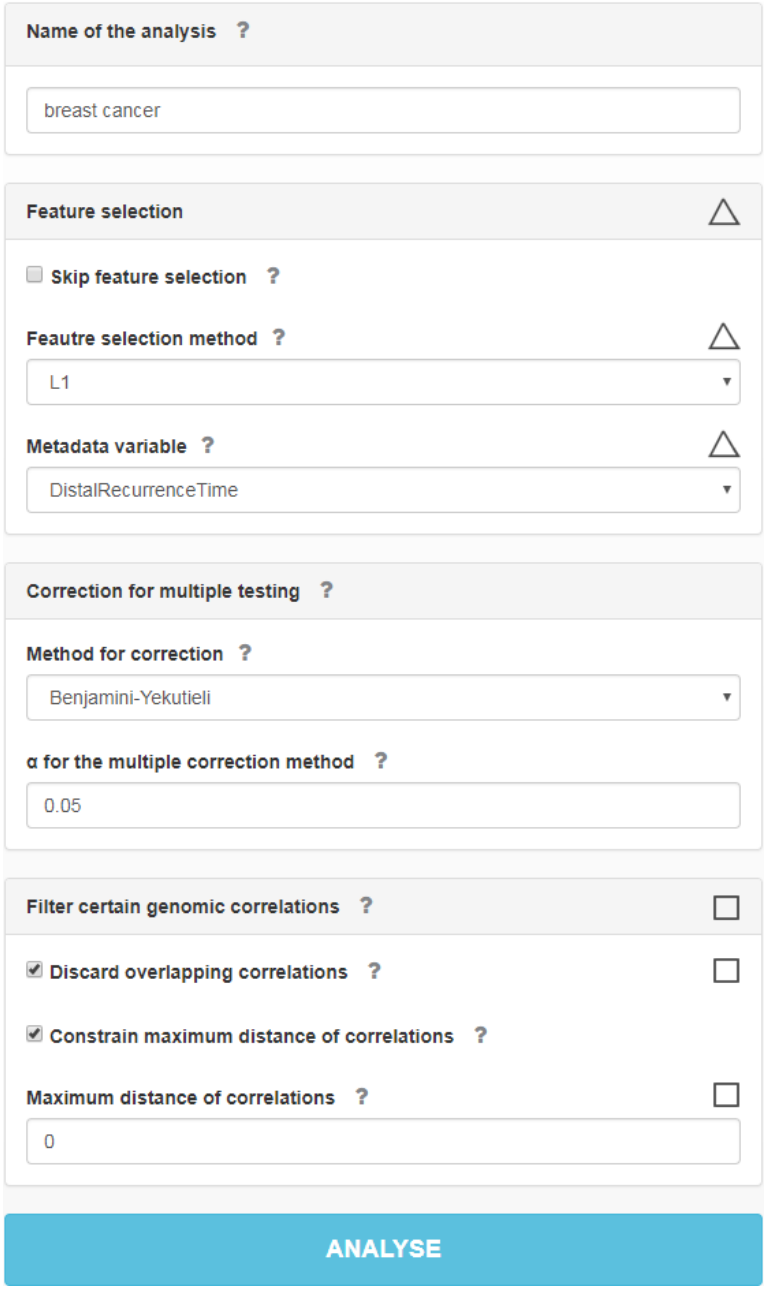


Figure 5.5: Analysis form of CorrMapper.

If the study has genomic features the panel and fields marked with a □ appear. These options represent different ways to filter out certain edges of the network, and therefore make them easier to explore and interpret.

If “Discard overlapping correlations” is ticked, CorrMapper will ignore correlations, whose two features overlap on a chromosome. For example, in case of the breast cancer study, it might be a sensible assumption, that if a certain genomic region is amplified (i.e. there are multiple copies of a gene), then its expression levels will be higher. This would result in a possibly significant and very strong positive correlation, which is biologically not too surprising or interesting.

Ticking the “Constrain distance of correlations” allows users to discard edges that are between overly distant genomic features. The number we put in the “Maximum distance of correlations” field are meant in megabases. Setting it to zero will constrain the maximum distance between correlated features to the length of chromosome they are located on. With this option we can focus on a short to mid-range genomic associations which might be important in some studies.

Similarly to the upload page, Hovering over the “Analyse” button will temporarily disable it and run the server side form validation process. If all fields pass, the button can be clicked, and then the analysis is added to the RabbitMQ message broker’s queue (see Section 4.3). As explained in the Chapter 4 about ScienceFlask, Celery will pick this job up from the task queue in order of submission time, and execute CorrMapper’s data integration pipeline with the user specified parameters.

The total time an analysis takes depends on multiple factors such as the number of jobs waiting in CorrMapper’s queue, the dataset’s size and the chosen feature selection method. For example, a typical multi-omics dataset with 10,000 features and 300 samples runs in under 15 minutes using Boruta and even quicker with L1 based feature selection. However, larger datasets or jobs of other users occupying the queue can expand this time frame to even an hour. To be economical with the host server’s resources, users cannot submit a new analysis while their previous job is still running. Once a job has finished running, CorrMapper notifies the user in email, and the results of the analysis become accessible from the profile page.

5.4 CorrMapper's back-end

Figure 5.6 summarises how CorrMapper's front-end and back-end interact. While the previous section introduced the user-facing components of the top box of the flowchart, this section focuses on the middle part, and describes what happens to the uploaded data. By the end of CorrMapper's data integration pipeline, all results become available for the user through the front-end's visualisation modules, which are introduced in Chapter 6.

5.4.1 Checking uploaded files

As alluded to in Section 5.3.2, once all files are transferred to the server from the user's computer, they are run through a series of checks to ensure CorrMapper's pipeline can use them. Generally, it is better to fail early in computational pipelines and let the user know as soon as possible about misformatted files. The Help page is a useful resource that describes the expected file formats. This section briefly summarises the checks CorrMapper uses to enforce these formatting rules.

Data files

Omics data files are expected to be preprocessed and cleaned, containing only numeric values. Each file must pass the following checks:

- All columns must be numeric. Those that contain non-numeric values are discarded. Each data file must have at least ten numeric features.
- The maximum sample size and feature number supported by CorrMapper currently are 500 and 25000 respectively. If a data file exceeds these dimensions, the script aborts.
- If two omics files were uploaded, they must have at least 15 samples in common.

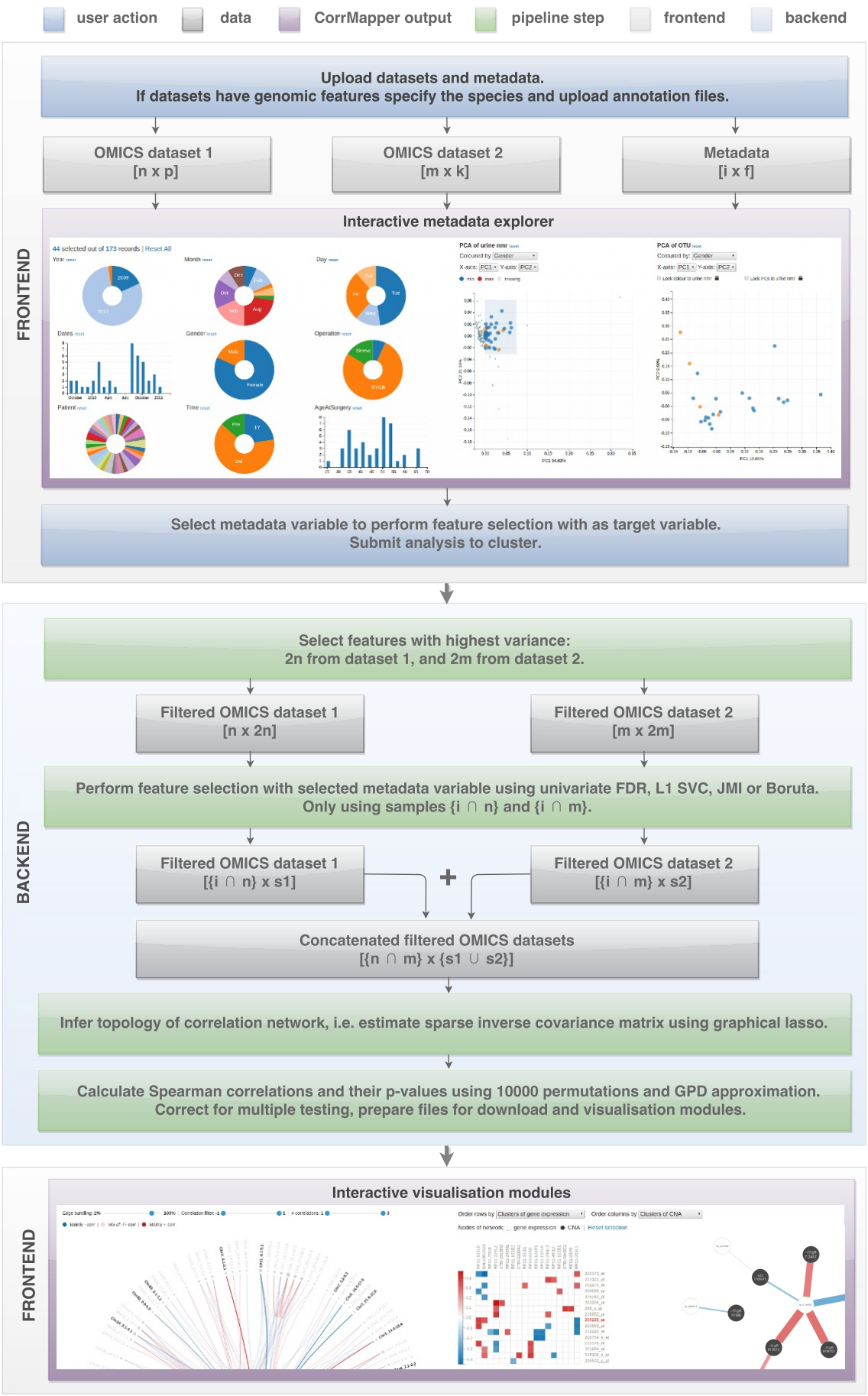


Figure 5.6: Flowchart of CorrMapper's front-end and back-end.

Annotation files

Annotation files hold chromosomal location of genomic features. This information is used by CorrMapper to build a special visualisation from genomic data. Each row of these files describes a single feature. Annotation files are checked line by line and a row can be discarded for several reasons:

- If it does not have the mandatory ProbeID, Chromosome, Start and End fields.
- If its probe is not present in the corresponding omics data file.
- If its chromosome number or name is not recognised by the species specific chromosomal map. For example, humans do not have a Z chromosome, therefore all annotation lines containing information about probes located on a Z chromosome would automatically get discarded.
- If its start or end position is smaller than zero or larger than the length of chromosome the probe is located on.

Metadata file

The metadata file is used both for feature selection and building the interactive metadata explorer described in Section 6.2. Therefore, its columns has to be checked thoroughly.

- The second row of the metadata file must contain the type of each variable. The accepted values include: categorical, continuous, date, sample and numerous abbreviations and synonyms of these.
- CorrMapper currently only supports 15 metadata columns. The reason for this is that these columns are later used to build the metadata explorer, which would get very dense visually with more variables.

- Categorical columns can contain any alpha-numeric characters to denote the levels of the variable. Very importantly for feature selection, each level has to have at least 15 samples in common with each omics dataset. For example the "Asian" level of the Ethnicity categorical variable would be discarded if the study's cohort would only have five patients with Asian ethnicity. If a categorical variable is left with only one level after this filtering step, it cannot be used for feature selection, and it will not show up in the analysis form.
- Continuous variables must contain numeric values only.
- The format of date variables should ideally comply with ISO8061, but most other commonly used formats are also automatically recognised and understood due to the highly advanced `time` module of `pandas` Python package. If the date variable cannot be parsed, it will be discarded.
- The sample variable type is a unique identifier that can be used to group patients within a study that collected repeated measurements from the same subjects: for example preoperative and postoperative samples.

If all checks are passed, the files are saved on the server and the study becomes accessible from the Profile page.

5.4.2 Feature selection and graph estimation

Figure 5.7 displays how the uploaded omics data files and metadata flow through CorrMapper's data integration pipeline, which was designed to be data type agnostic and make the most of $p > n$ datasets. It consists of the following steps:

1. The omics datasets are variance filtered: given a dataset $X \in \mathbb{R}^{n \times p}$, where $p > 2n$, it is filtered down to $X \in \mathbb{R}^{n \times 2n}$, where the remaining $2n$ features have the highest variance. As explained in Section 5.2, this ensures that FS methods perform closer to their optimal n/p ratio.

2. The user chosen feature selection method is applied to both omics datasets. Importantly the FS is carried out separately on both omics datasets. Although this means that the relevant features are not selected from the joint probability distribution of the two datasets, this way the optimal n/p ratio is ensured. Furthermore, if the two datasets were to be concatenated before the FS process, CorrMapper would need to take the intersection of their samples with the metadata file: $\{n \cap m \cap i\}$, see Figure 5.7. In most multi-omics datasets this is highly restrictive as we rarely have total coverage for a sample, i.e. omics data from both modalities and also metadata.

3. The two filtered datasets $X_1 \in \mathbb{R}^{n \times s1}$ and $X_2 \in \mathbb{R}^{m \times s2}$ are concatenated to form $X_c \in \mathbb{R}^{n \cap m \times s1 \cup s2}$. The features in X_c are standardised to have zero mean and a standard deviation of one. This is done to ensure that data coming from different platforms have the same range and variance.

4. As explained in Section 3.4.4, a nonparanormal extension of the graphical lasso algorithm is employed to infer the conditional independence network of the features within X_c. The `huge` R package is used both for network estimation and network regularisation with the StARS algorithm. Both algorithms are run with their default parameters, however the StARS algorithm's $\hat{\Lambda}$ threshold can be modified by the user through the analysis form.

5. As Figure 5.7 highlights, running the pipeline on two omics datasets will result in a matrix $N \in \mathbb{R}^{s1 \cup s2 \times s1 \cup s2}$ that holds three separate networks:
   - N_1: network of X_1, edges are formed between selected features of X_1.
   - N_2: network of X_2, edges are formed between selected features of X_2.
   - N_{12} : bipartite network of X_1 and X_2, where each edge is formed between selected features of X_1 and X_2.

6. Spearman correlations are calculated for each edge of the networks. Then, as explained in Section 3.4.5, 10^4 permutations and GPD approximation is used

estimate p-values for each. If the user set $\alpha < 1$, a correction for multiple testing is applied to the p-values, and those that did not pass are discarded.

7. Finally, networks are built from matrix cells of N which have non-zero precision ($\Omega_{ij} \neq 0$) and a p-value $< \alpha$ after correction for multiple testing.

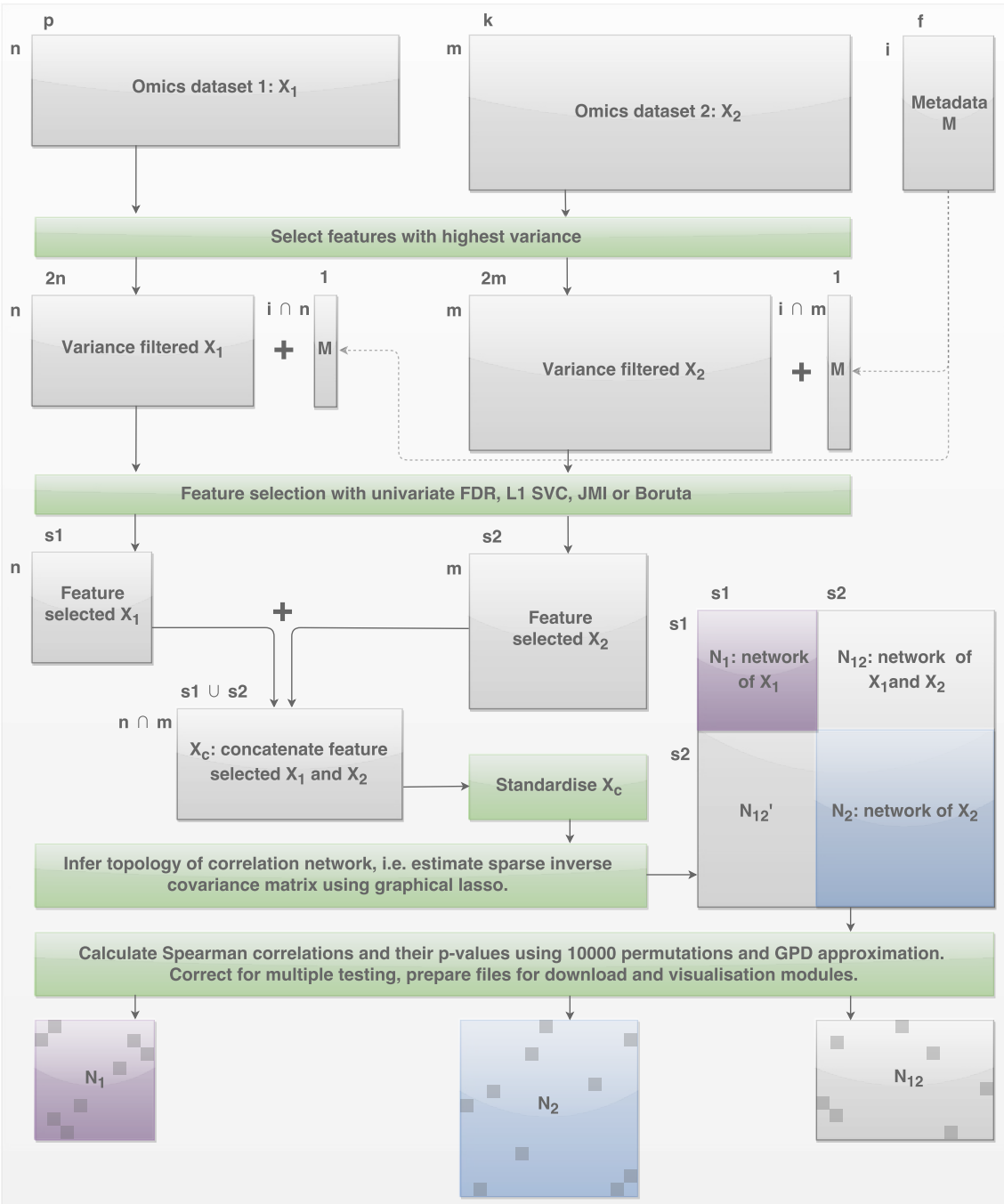


Figure 5.7: Flowchart of CorrMapper's data integration pipeline. The dimension of each matrix is displayed at its top left corner. Grey, purple and blue boxes represent data, while green ones are steps in the pipeline.

5.4.3 Module finding

Once the three conditional independence networks (N_1, N_2, N_{12}) are estimated, CorrMapper employs two different module finding algorithms to identify tightly connected groups of edges within each graph. Modularity is a mathematically well-defined measure of a graph's structuredness. It quantifies how strongly the network is divided into tightly connected sub-graphs, i.e. modules, or communities.

Modularity ranges between $[-1/2, 1)$, and it can be defined as the fraction of edges that fall within a given module minus the expected fraction if edges were distributed at random (i.e. in a random graph). A negative modularity score implies that fewer interactions occur within modules than expected in a random network. Conversely, networks with high modularity have densely linked communities which are more interconnected than it would be expected in a random graph.

Many real life problems can be modelled as a graph: food chains, metabolic pathways, neural networks, social networks, trade between companies. Detecting communities within these graphs can lead to a deeper understanding of them, as nodes within the same module are more strongly related, interact, transact and exchange information more often than with nodes outside of their community.

In biology, this closely knit nature of nodes from the same module, frequently arises from their similar biochemical function. Therefore, if we are unsure about the function of a node within a module, we can exploit the principle of guilt by association and learn something about it by examining its neighbours.

Modularity detection is an important problem in several scientific fields, which has led to the development of numerous algorithms[155]. Yang et al. performed extensive benchmarking experiments, comparing eight popular community detection algorithms, considering different aspects of these methods including their accuracy and

required computing time. Based on their results, CorrMapper uses a Python implementation of the widely cited Louvain method[156] from the `networkX` package[157], to identify modules within the correlation matrix of unipartite graphs N_1 and N_2. For bipartite graphs, the definition and discovery of modules is more difficult, as there are two kinds of nodes present in these networks. CorrMapper uses a recently published method[158] which was shown to outperform earlier bipartite modularity finding algorithms.

In both unipartite and bipartite networks the negative correlations are turned positive to make a weighted adjacency matrix as an input for the module finding algorithms. The modules found by these two algorithms in N_1, N_2, N_{12} are then exposed to the user through the general network explorer, which is introduced in Section 6.3.

5.4.4 Preparing files for visualisation and download

Once CorrMapper has finished running its data integration pipeline, it begins to produce all the necessary data files and figures for the visualisation modules. The networks N_1, N_2, N_{12}, their identified modules, the annotation files (in case of a genomic dataset), the Spearman ρ values and their p-vales are converted into JSON files for JavaScript. The networks are also saved in XML format to let users import them into Cytoscape[159], which is a popular offline biological network explorer.

Furthermore, as we will see in Section 6.3, CorrMapper has to produce scatter-plots for every single edge of the resulting networks, which are used in the general and genomic network explorer interfaces. These are also generated at this stage.

Finally, all of these result files are made accessible to the user through the profile page as a downloadable archive. This .zip file contains all the above listed items, along with the selected features per each dataset.

5.4.5 CorrMapper with simulated data

Similarly to the feature selection benchmarking experiments, the results shown in this section could be reproduced using the code in the `supp/cmTest` folder. Furthermore, as in case of the simulated datasets of the FS benchmarking, scikit-learn's `make_classification` function was used to generate artificial datasets with varying number of samples and features.

However, in order to test CorrMapper's ability of recovering the true conditional independence structure of these datasets, the `make_classification` function had to be modified to allow for the simulation of data with known correlation structure. This was done based on the following well established procedure for simulating multivariate normal data with known correlation structure :

- Given a set of p independent normal random variables $X = (X_1, ..., X_p) \sim \mathcal{N}(\mu, \Sigma)$, with a zero mean vector $\mu = 0$ and no pair-wise associations $\Sigma = I$,
- their correlation structure can be defined by first creating a positive semi-definite matrix $C \in \mathbb{R}^{p \times p}$, then decomposing it into a lower triangular form using the Cholensky factorisation, so that $C = LL^T$.
- Then, $X' = LX$ will give us back our normally distributed random variables with their correlation matrix set as C, $cor(X') = C$.

Scikit-learn's `make_sparse_spd_matrix` function was used to simulate a sparse symmetric definite positive matrix, which acted as the precision matrix Ω of the simulated data, with 98% of its cells set to zero. This was then inverted to obtain the correlation matrix $\Sigma = \Omega^{-1}$, which was used in the modified `make_classification` function to simulate a two class dataset with pre-defined correlation structure and known feature importances. Once a dataset was simulated this way, its feature space was randomly divided into two sets, to create two equally sized simulated omics data tables as an input to CorrMapper's data integration pipeline.

Numerous datasets were simulated with varying sample sizes $n \in \{100, 300, 500\}$, and feature numbers $p \in \{1000, 2000, 5000\}$, each with three different random seeds. Two values were used for the number of informative: 5% and 10% of p. Each dataset was run through CorrMapper with all four FS methods, and two different StARS thresholds: 0.05 and 0.1. To save computational time, the threshold of the multiple correction step was set to $\alpha = 1$, therefore this edge filtration part of the pipeline was switched off in these experiments.

Similarly to Figure 3.6, the performance of the pipeline was measured by precision and recall, as the number of correctly discovered true edges of the simulated dataset's conditional independence graphs (all three of them N_1, N_2, N_{12}). Table S2 shows the overall performance of CorrMapper's pipeline across all simulated datasets, while Table 5.3 displays the results obtained on $R \geq 0.2$ datasets.

Since a lower StARS threshold requires higher robustness under resampling for an edge to be kept, $\hat{\Lambda} = 0.05$ resulted in higher precision with all FS methods. Although the overall recall of all four methods seem disappointingly low, we need to keep in mind how hard the problem at hand is: we are aiming to not only correctly identify 5% of the features, but then also reconstruct the extremely sparse conditional independence network between these from noisy, $p \gg n$ data. As we will see later, this is very challenging for other network estimation methods as well.

	StARS	UnivarFDR	L1 SVC	Boruta	JMI
Rec	0.05	0.1182 ± 0.02	0.2808 ± 0.13	0.1146 ± 0.02	0.0429 ± 0.03
Prec		0.6113 ± 0.13	0.5348 ± 0.11	0.6146 ± 0.14	0.4905 ± 0.19
R + P		0.7295 ± 0.15	0.8156 ± 0.24	0.7293 ± 0.16	0.5334 ± 0.21
Rec	0.1	0.1356 ± 0.05	0.1032 ± 0.17	0.1298 ± 0.05	0.0462 ± 0.04
Prec		0.4984 ± 0.16	0.4127 ± 0.09	0.5353 ± 0.17	0.4496 ± 0.23
R + P		0.6341 ± 0.21	0.5158 ± 0.26	0.6652 ± 0.22	0.4958 ± 0.27

Table 5.3: Performance of CorrMapper on $R \geq 0.2$ simulated datasets. Rec: recall, Prec: precision, R + P: recall + precision. These performance metrics measure how many of the true network edges CorrMapper could reconstruct. Values to the left of the ± are means, while values on the right represent one standard deviation.

All methods performed worse in $R < 0.2$ datasets (see Table S2). This highlights the true difficulty in identifying meaningful relationships in $R < 0.2$ datasets. Nonetheless in $R \geq 0.2$ datasets with $\hat{\Lambda} = 0.05$, L1 SVC performed the best, whereas Boruta and univariate FDR were tied for the second place. This changed however with $\hat{\Lambda} = 0.1$, as in this scenario Boruta performed best, followed by univariate FDR and L1 SVC. Interestingly, all methods prioritised high precision over high recall.

To compare CorrMapper with other network estimation methods, a follow-up experiment was carried out on datasets with sample sizes $n \in \{300, 500\}$ and feature numbers $p \in \{1000, 2000\}$, each simulated with three different random seeds. The number of informative features was defined as 5% of p, while the threshold of StARS was set to $\hat{\Lambda} = 0.05$. Marginal correlation network estimation (see Section 3.4.2), graphical lasso (see Section 3.4.3), and sparse PLS from the mixOmics package (see Section 1.3 and 5.5) were used to benchmark CorrMapper's data integration pipeline.

Furthermore, to construct networks with varying density, marginal correlation networks and the network estimator of mixOmics (see Section 6.5) were used with a range of cut off values $\epsilon \in \{0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.8\}$. Finally, graphical lasso was used with the same $\hat{\Lambda} = 0.05$ StARS threshold.

Method	Recall	Precision	Rec + Prec
UnivarFDR	0.1455 ± 0.11	0.5845 ± 0.14	0.7300 ± 0.25
L1 SVC	0.3740 ± 0.20	0.5808 ± 0.14	0.9548 ± 0.34
Boruta	0.1525 ± 0.11	0.6113 ± 0.19	0.7638 ± 0.31
JMI	0.0852 ± 0.08	0.6462 ± 0.32	0.7315 ± 0.39
Marginal 0.05	0.3327 ± 0.10	0.0313 ± 0.01	0.3641 ± 0.11
Marginal 0.1	0.0672 ± 0.04	0.0494 ± 0.03	0.1166 ± 0.07
mixOmics 0.05	0.4432 ± 0.11	0.1086 ± 0.03	0.5518 ± 0.14
mixOmics 0.1	0.2558 ± 0.11	0.2153 ± 0.07	0.4711 ± 0.19
mixOmics 0.2	0.1339 ± 0.08	0.3840 ± 0.15	0.5180 ± 0.24
mixOmics 0.3	0.1065 ± 0.06	0.5371 ± 0.21	0.6437 ± 0.28
mixOmics 0.5	0.0632 ± 0.05	0.9061 ± 0.17	0.9693 ± 0.22

Table 5.4: Comparing CorrMapper against other network estimators. Values to the left of the ± are means, while values on the right represent one standard deviation.

Very importantly, since mixOmics only estimates the N_{12} network between two omics datasets (ignoring N_1 and N_2), all performance metrics were calculated on the edges of this particular network, for all methods.

Table 5.4 summarises the results of successful runs. Several rows were omitted from this table, as some cut off values resulted in selecting zero edges in the marginal and PLS networks. All results (including the ones omitted here) are listed in Table S3. Unfortunately, the graphical lasso algorithm did not yield any meaningful results on these 12 simulated datasets. Although $\hat{\Lambda} = 0.05$ is suggested by the authors of StARS as a default value for the resampling thresholds, it might not have been a suitable choice for graphical lasso on these datasets with $R \in \{0.15, 0.2, 0.25, 0.3\}$.

As expected from Section 3.4.2, regardless of the chosen ϵ, marginal correlation networks performed worse than other methods. Sparse PLS was used with three latent components in these experiments. Interestingly, despite its sophisticated nature, this algorithm also performed worse than CorrMapper with most choices of ϵ.

Although mixOmics achieved comparable results to CorrMapper with $\epsilon = 0.5$, it is very important to emphasise that ϵ is a hidden hyperparameter, that users cannot reliably estimate or know in a real dataset. Therefore, even though some optimal value of ϵ might result in comparable results, the probability of a user picking this out of all other values is negligible. Furthermore, mixOmics had a clear advantage over other methods, as in all experiments, the number of features to select (by the L1 regularisation), was set exactly to the correct one (i.e. half of the informative features in both datasets).

In summary, based on the last column of Table 5.4, and the fact that it requires minimal user input, and no prior knowledge about the number of features to select, CorrMapper performed notably well. These experiments demonstrate the overall utility of CorrMapper's data integration approach, including metadata driven feature selection and regularised conditional independence network estimation.

5.5 Summary

As we have seen in Section 1.3, there are numerous approaches to multi-omics data integration. CorrMapper is a network based, non-Bayesian data integration tool, which focuses on the discovery of novel biochemical interactions between different omics layers. As we will see in Section 6.2 it also performs a simple type of clustering through dimensionality reduction and metadata mapping. Furthermore, unlike other regularised data integration pipelines, it provides non-linear feature selection algorithms which can capture correlated groups of covarying features.

However, as it will be explicated exhaustively in Chapter 6, CorrMapper goes further and it offers advanced data visualisation modules which allow for the interactive exploration of complex correlation networks and clinical metadata. Because of this multifaceted nature, CorrMapper is probably closest to the mixOmics R package, which is a powerful and feature rich data integration and visualisation pipeline, developed as the joint effort of numerous researchers over the past 10 years[38,41,160–162]. Therefore, the following paragraphs briefly outline some of the main differences between the data integration pipelines of CorrMapper and mixOmics, while Section 6.5 will reflect on their visualisation capabilities.

Both CorrMapper and mixOmics rely on regularised multivariate statistical methods to discover meaningful relationships between different layers of multi-omics studies, but they use very different approaches. As alluded to in Section 1.3, mixOmics uses sparse versions of CCA and PLS, which are both unsupervised, linear projection based, latent variables models, that aim to maximise the correlation and covariance between two data matrices respectively. Additionally, mixOmics offers a sparse version[163] of PLS-Discriminant Analysis (PLS-DA)[164]. Similarly to PCA, PLS-DA projects a dataset into a low-dimensional space, but does this in a supervised fashion by maximising the separation between levels of a categorical outcome variable.

These latent variable models have several attractive properties. Firstly, they model two data matrices mathematically as a joint system with a single optimisation criterion. As we saw in Equation 1.1, the latent components ($H \ll p$) are formed by taking linear combinations of the original predictors, where the contribution of each feature is defined by the corresponding coefficient in the loading vectors. Therefore, these algorithms perform dimensionality reduction while capturing the largest sources of variation in multiple datasets simultaneously. Secondly, PLS is especially well suited to $p \gg n$ datasets, as evidenced by its widespread use in the chemometrics field[165].

However, there are some disadvantages of these techniques:

- The number of latent components H is an important hyper-parameter, which defines the dimensionality of the hyperspace into which all samples are projected. The optimal H can be found using cross-validation, however this means that the model selection is performed on less data, which can be detrimental in $p \gg n$ datasets. Instead of the traditional k-fold cross-validation, we can use Leave-One-Out (LLO) cross-validation to alleviate this problem, however that can increase training time substantially. Alternatively, in case of PLS-DA models, some authors offer a heuristic method where they define $H = G - 1$, where G is the number of levels in the categorical outcome variable[160].
- Latent variable models can be cumbersome to interpret. The samples are projected into a lower dimensional space, which is defined by orthogonal score vectors as linear combinations of thousands of variables (in case of omics datasets). Although the orientation of these dimensions are chosen such that they maximise the covariance or correlation between the two data matrices, similarly to PCA, two samples projected close to each other in a given pair of score vectors might be far away from each other in another pair of latent variables. Therefore, each pair of score vectors represent a new view of the dataset, that needs to be interpreted in conjunction with their loading vectors.

- The contribution or loadings coefficient of an individual omics feature can be positive in one score vector dimension and negative or zero in another. Therefore, it can be hard to determine how exactly one feature behaves overall and what is its relationship to others. In latent variable models, this can only be discussed in the context of score vectors, which represent orthogonal views of the data. Conversely, CorrMapper provides a single network view of the selected features, which might be easier to interpret in some scenarios.
- The regularised version of CCA and PLS use the L1 penalty to obtain sparse loading vectors and consequently perform feature selection while fitting the model. Although this can make the interpretation of loading vectors easier, the feature selection in this case is driven by the objective function's aim to maximise the covariance between the score vectors. Therefore, unlike in CorrMapper, the selected features do not possess a clear relationship with any clinical outcome variable. Furthermore, if two or more features are correlated, the L1 penalty will choose one of them randomly, while shrinking the coefficients of others to zero[50]. Therefore, a feature might deemed important in one loadings vector and unimportant in another. This can be problematic in biological datasets, where often, correlation between features encode a shared biological function or pathway, which can be lost using L1 regularisation. CorrMapper offers three FS methods (univariate FDR, Boruta, JMI) that can handle this situation and capture correlated and relevant feature sets.
- Interestingly, mixOmics request the user to specify the number of features to be selected for each latent variable. This can be very difficult to estimate *a priori*, especially if we are analysing a new high-throughput multi-omics study, that we have not interrogated before. Alternatively, users can supply a grid of values, and mixOmics will choose one of the grid's values using cross-validation. Conversely, CorrMapper aims to minimise the input required from users, and let the data determine the number of features to be selected.

Both CorrMapper and mixOmics are platform agnostic and can work with any type of numerical data. Furthermore, they both assume that the omics datasets have been preprocessed by their platform-specific pipelines.

Interestingly, until the very recent addition of sparse Generalized Canonical Correlation Analysis (sGCCA)[160] to mixOmics, it did not support the simultaneous integration of two omics datasets with an outcome variable. PLS-DA works with a single data matrix and a categorical outcome, while CCA and PLS either work with a matrix of outcome variable and an omics dataset, or with two omics datasets.

However, sGCCA solves this problem by extending sparse CCA algorithm to K number of normalised, centred and scaled datasets $X_1 \in \mathbb{R}^{n \times p}, ..., X_K \in \mathbb{R}^{n \times p_K}$. These datasets record the levels of $p_1, p_2, ...p_K$ omics features in the K dataset for the same n samples. sGCCA is fitted by optimising the following objective function:

$$\underset{a^1,...,a^K}{\arg\max} \sum_{k,j=1,k\neq j}^{K} c_{jk}\, cov(X_k a^k, X_j a^j), \qquad s.t. ||a^k||_2 = 1,\ ||a^k||_1 < \lambda_k,$$

where $c_{jk} = 1$ will maximize the covariance between the datasets X_j and X_k, while $c_{jk} = 0$ denotes that we do not presume any relationship between X_j and X_k. a^k denotes the loadings vector of dataset X_k. Similarly to other L1 regularised methods, λ_k is a non-negative parameter that controls the amount of shrinkage and thus the number of non-zero coefficients in a^k.

If one of the K datasets is chosen to contain a dummy-encoded categorical outcome variable with G levels as $X_k \in \mathbb{R}^{n \times G}$, then sGCCA will effectively perform sparse PLS-DA on $K - 1$ datasets, simultaneously maximising the covariance between $X_1, ...X_{K-1}$, while also maximising the discrimination between the omics datasets and the categorical outcome. Furthermore, sGCCA is a predictive model, which is capable of determining the class membership of new samples. These properties make sGCCA a very capable and promising data integration technique. In comparison,

CorrMapper cannot integrate more than two omics datasets with an outcome, but the chosen outcome variable can be continuous, which is not supported by sGCCA currently.

Additionally, the MINT pipeline within mixOmics[161] allows for the integration of several independent studies measured on the same p predictors. This is an orthogonal direction in data integration that is not supported in CorrMapper currently.

Similarly to CorrMapper, mixOmics is open-source and available as an R package under the GNU licence. Unfortunately its web interface at http://mixomics.qfab.org/ was not available during the writing of this thesis, therefore no comparison could be made between its online version and CorrMapper's capabilities. It is worth reiterating, that not all life-scientists posses programming skills and therefore the powerful capabilities of mixOmics might remain under-utilised in certain parts of the research community.

In summary, mixOmics leverages regularised latent variable models to project two or more omics datasets into a low-dimensional space, where most of the covariance of these matrices is captured. It builds feature selection into the optimisation criterion through the L1 penalty, and with the recent addition of sGCCA, mixOmics can integrate and build predictive models from more than two omics datasets in conjunction with a single categorical outcome.

Conversely, CorrMapper places metadata driven feature selection before the correlation analysis and builds robust conditionally independence networks from the biologically relevant features. However, CorrMapper's FS methods might be better suited to the correlated features of biomedical datasets. Furthermore, CorrMapper finds modules of tightly linked and therefore potentially biologically related features, using the topological properties of the estimated correlation network, but it cannot perform prediction on new samples.

To my knowledge, CorrMapper is the first multi-omics data integration pipeline that combines metadata driven feature selection with regularised conditional independence network estimation to discover multi-level interactions between omics datasets.

As we have seen in the benchmarking experiment of Section 5.4.5, CorrMapper offers a very capable alternative to existing methods, while requiring minimal input from users. Consequently, the data integration pipeline of CorrMapper and its visualisation modules (see Chapter 6) are currently in preparation for publication.

CorrMapper and mixOmics use entirely different and complimentary data integration strategies. Therefore, one should use both on multi-omics datasets and aim to combine their findings in a parsimonious way.

mixOmics's sGCCA presents an attractive one-stop shop solution for sparse integrative multi-omics modelling, with added predictive potential, while CorrMapper's robust conditional independence networks and powerful, interactive visualisation modules can provide illuminating insights into the complex inter-omics relationships of multi-modal biomedical studies.

Finally, both methods have extensive visualisation capabilities to facilitate the interpretation of their results. These software components will be compared at the end of the next chapter in Section 6.5.

6 | CorrMapper - interactive data visualisation

6.1 Overview

As alluded to earlier in Section 1.5.4, data visualisations have tremendous potential in data analysis, because a single well constructed chart has the power to take us by surprise, calling our attention to previously unknown relationships within our data. When combined with interactivity, data visualisations reach their full potential and turn from passive data presentation techniques into dynamic research tools.

This is especially valuable in biomedical research, given the complexity of omics datasets, which is compounded by the rich metadata that is collected with them. Interactivity allows us to dissect and subset these datasets in numerous ways iteratively, till we can answer our specific questions.

To capitalize on the value of interactive visualisation tools, CorrMapper implements three separate modules of them:

1. **Metadata explorer:** maps any form of metadata onto high-dimensional omics datasets. Its interactive dashboard interface allows users to stratify their cohort by any combination of metadata filters. For example: "show me the patients who are older than 50 years, received chemotherapy and have familial history of the cancer".

2. **General network explorer:** provides three different and complimentary views of the estimated conditional independence networks (N_1, N_2, N_{12}). Within this interface, two different network representations are cross-linked with an interactive heatmap. These three components together, enable users to explore and interrogate even the densest and most complex networks.

3. **Genomic network explorer:** is specifically designed for omics features which can be mapped onto the genome of a species, e.g. genes expression, epigenetic methylation or copy number variation probes. This module combines a circular genomic map with searchable and sortable tables. The former acts as a high level overview of the genome-wide correlations, while the latter provides very detailed information about each probe and their associations.

Due to the highly interactive nature of these modules, the static figures found in this chapter cannot fully demonstrate these tools. Therefore it is highly recommended to the reader to log in to CorrMapper with the guest account (username: guest@corrmapper, password: GraphicalLa$$o) and explore these modules interactively. As all three visualisation modules are fairly dense visually, a minimum screen resolution of 1920×1080 is required, although if possible, a higher one is recommended. Finally, in Section 6.5 a brief overview is given about the various currently existing omics visualisation tools and how they compare to CorrMapper's solutions.

6.1.1 Limitations of CorrMapper's visualisation modules

> When two things commonly occur together, the appearance of one will bring the other to mind.
>
> Aristotle c. 350 B.C.

The concept, that the co-occurrence of two things is a fundamental component in the creation of knowledge, was first recognised by Aristotle. Possibly owing to our

evolutionary past as hunters and gatherers, our minds are wired in such a way, that if we see pairs of things together, or following each other in quick succession, we quickly formulate causal links between them.

This curiosity and causality seeking behaviour is one of the foundations of modern science and human culture in general, but also the root cause of the damaging cognitive biases we all (even scientifically trained researchers) suffer from[166]. As Nobel laureate, Daniel Kahneman has put it, in his best selling book Thinking, Fast and Slow:

> "We are pattern seekers, believers in a coherent world, in which regularities appear not by accident but as a result of mechanical causality or of someone's intention."

As CorrMapper's network visualisation modules both build on the concept of co-occurrence, it is really important to admit the limitations of these tools, and to use them with caution. Due to their highly interactive nature, researchers can easily get lost in the exploration of their datasets, and cherry-pick correlations that support their prior beliefs or hypotheses. Therefore, a cautious attitude should be adopted when using these modules.

Furthermore, as we will see later, the metadata explorer allows us to stratify our cohort in any imaginable way, splitting it across multiple variables, therefore selecting an arbitrary subgroup of samples. This process, especially when carried out iteratively several times, can lead us to finding needles in a haystack which are not real but simply an artefact of the small sample size of the selection and repeated partitioning of the data.

Consequently one should spend ample time with these visualisations to develop a more holistic view of a dataset, simultaneously examining and interpreting all associations uncovered by CorrMapper.

6.2 Metadata explorer

CorrMapper's metadata explorer enables users to browse their multi-omics studies together with the collected metadata variables and stratify their samples in an interactive way. As discussed earlier in Section 5.3.3, all uploaded studies are listed under the profile page. To launch the metadata explorer, we need to click on the green "Explore" icon next to any of our uploaded studies. CorrMapper will then perform the following sequence of actions:

1. Omics datasets are first mean centred, and PCA is applied to them to reduce their dimensionality. The first five principal components are kept for each dataset. This allows CorrMapper to map the metadata variables onto the low dimensional representation of the omics data. Ultimately this mapping enables users to find clinically or biologically relevant connections between these disparate data sources, and make a decision on the metadata column they will use as target variable in CorrMapper's feature selection process.

2. The date column (if it is present in the uploaded metadata file), is parsed and split into three separate categorical variables: year, month and day of the week. This latter variable enables users to spot weekly temporal patterns. These three new date variables are appended to the metadata table.

3. Finally, CorrMapper builds a new dashboard from the metadata file's columns. This process is fully automated and uses algorithmic code generation, i.e. a different chart is generated for each metadata variable, which are then linked together to form an interactive dashboard. By the end of this process, a highly complex, study specific JavaScript file is produced, which is unique to the uploaded metadata and is often more than 700 lines of code. This process only takes a few seconds, and has to be done just once for each uploaded study.

6.2.1 Components of the dashboard

As we have seen in Section 5.4.1, the first cell of each metadata column, has to be the variable's type to ensure CorrMapper treats it properly. This becomes crucially important here, as CorrMapper uses these values to infer what type of chart to use for displaying a given metadata variable.

As shown in Figure 6.1, the metadata explorer is divided into two major parts: on the left we have the columns of the metadata file, represented as pie and bar charts, whereas on the right we have one or two scatter plots, displaying the lower dimensional representation of the uploaded omics dataset(s). First, we will look at the components that make up the left hand side.

As we will be referencing the metadata explorer of the bariatric surgery dataset throughout this section, it is recommended that the reader opens this dashboard from the guest account before proceeding. Furthermore, the demo video on CorrMapper's website can be helpful for new users, as it provides a brief introduction to the handling of all three visualisation modules. In the following section each dashboard variable type is briefly introduced along with its corresponding chart.

Categorical variables

Categorical variables are represented with pie-charts. The levels of each variable are clearly marked on the slices of these charts. If there is not enough space visually to display a certain label (as is the case with the Month chart), we can hover over a slice to find out the name of the level. Doing so will also display the number of samples that belong to this section of the dataset. If a label of a level is too long to be displayed on a pie-chart, CorrMapper will display the categorical variable as a horizontal bar-chart, see the Ethnicity chart for example of the breast cancer study.

Continuous variables

Continuous variables are represented using bar-charts. The y-axes of these plots clearly mark the number of samples belonging to each bar, while the x-axes display the range of each variable.

Patient/sample variable

The patient/sample metadata variable is optional, and it defines a grouping of samples. In the bariatric surgery study, samples which were obtained at different time-points for the same patient are grouped using this variable type, which is displayed as a rainbow coloured pie-chart.

Date variable

If there is a date variable present in the metadata file, as in the bariatric surgery dataset, the three derived categorical variables (Year, Month, Day), will always be displayed next to each other as pie-charts, followed by the bar-chart of Dates, which represents these three variables combined together.

PCA of omics datasets

On right hand side, CorrMapper displays the scatter-plots of two PCA components of each omics dataset. Every dot marks a sample, when projected into a two-dimensional space. Points that are closer together represent samples that are biologically more similar.

Although by default, the first two principal components are shown, users can easily chose any other from the precomputed five components, using the “X-axis” and “Y-axis” dropdown menus, located above each scatter-plot. This can be set separately

for each PCA scatter plot, or simultaneously by clicking the “Lock PCs” radio button, which links the axes of both charts together.

Furthermore, the colour of each dot is defined by one of he chosen metadata variables on the left. This can be changed by the “Colour by” dropdown menus. The colour could be set to both categorical and continuous variables.

When set to a categorical, the colour of each dot matches the hue of the level a sample belongs to, i.e. slice of the pie-chart. When set to a continuous, a blue (minimum) to white (median) to red (maximum) gradient is used to colour each sample according their metadata value. Similarly to the axes of the PCA scatter plots, the colouring of these charts can be linked together by clicking the “Lock colour” radio button.

Data table

CorrMapper also generates a table (located below the charts), which displays the metadata of each sample precisely, in tabular format. Although, given the pie and bar-charts, the information contained in this table is to some extent redundant, the tabular format enables users to quickly glance through the metadata values of a selected group of patients.

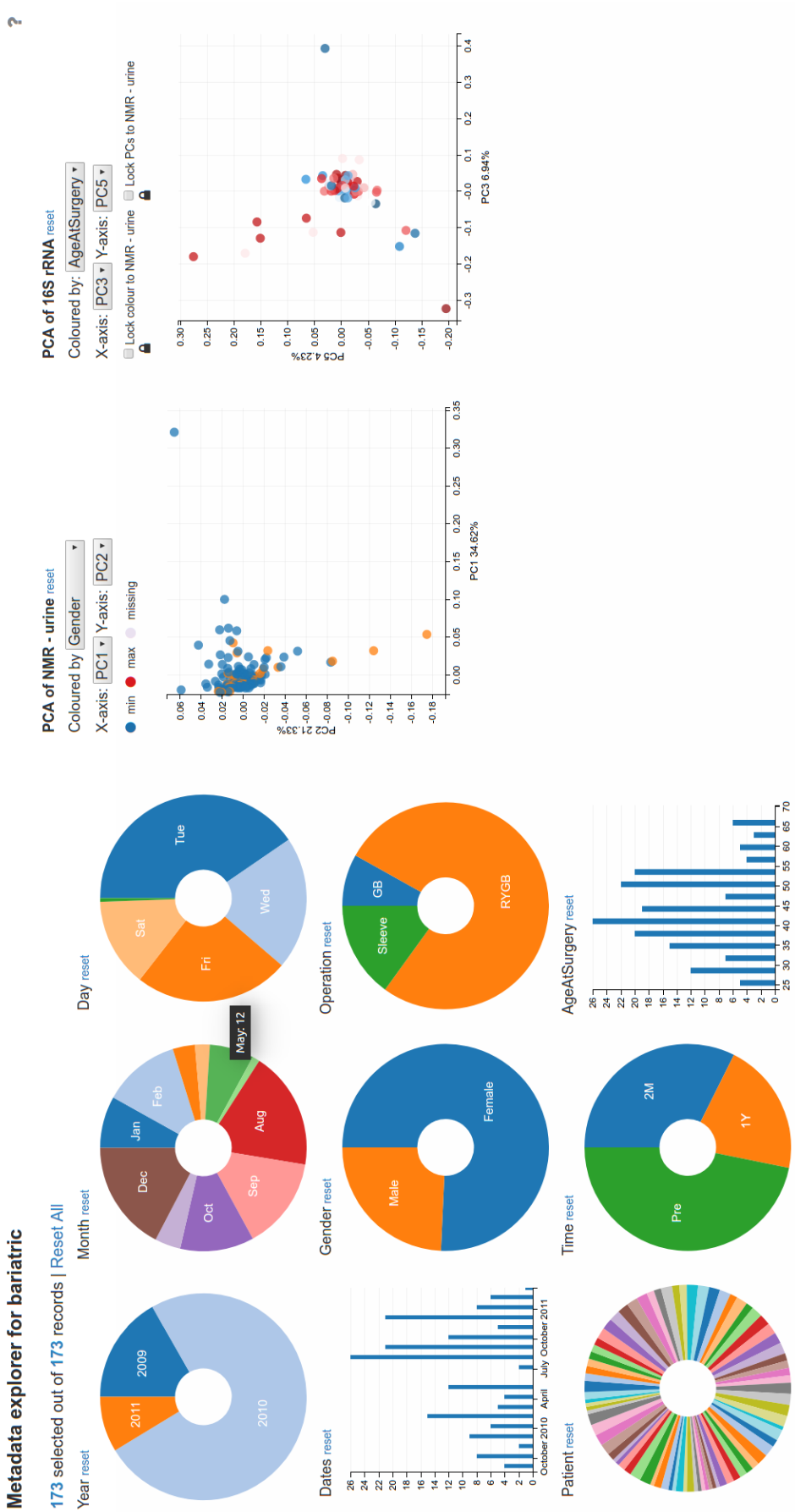


Figure 6.1: Components of the bariatric surgery study's metadata explorer. On the left: categorical and derived date variables (Year, Month, Day) are displayed as pie-charts. If the label of a pie-chart slice in not legible, hovering over it will display the text (May) and also the number of samples (12) belonging to that level of the categorical variable. Continuous variables are represented as bar-charts. On the right: two scatter-plots display the low dimensional representation of the omics datasets. Each dot marks a sample. The colour of these scatter plots can be set to any of the categorical (Gender) or continuous (Age) metadata variables.

6.2.2 Cross-filtering

As noted earlier, the metadata explorer is an algorithmically generated JavaScript program, which is produced for each study individually, using the uploaded metadata and omics files. To build the above described components of a dashboard, CorrMapper uses the `dc.js` JavaScript library. However, the real utility of these interactive charts stems from their interlinked nature, which is achieved with the `crossfilter.js` JavaScript library.

All components of CorrMapper's dashboard are interactive and filterable. Pie-charts, used to display categorical variables, can be filtered by clicking on one or more of their slices. This will select the samples which have the value represented by the clicked slice. In the top-left corner of the metadata explorer, a counter displays the size of our current selection and also the number of all samples. For example, we can check how many samples did not come from RYGB surgery, by clicking on the "Sleeve" and "GB" slices of the Operation pie-chart. After we have made this selection the count in the top-left corner shows 40 samples of the 173.

Very importantly however, when we clicked these two slices, all other pie and bar-charts have updated too along with the scatter-plots on the right. By selecting a subset of our full cohort, the distribution of some metadata variables changed too. For instance, if we now switch the "RYGB" slice on and off (by clicking on it repeatedly), we can see the histogram of BMI and Age change. Although, within this dataset, none of these changes seem striking, one can easily imagine scenarios where this interactive subsetting quickly and easily uncovers interesting associations between metadata variables.

Furthermore, filters can be applied to the bar-charts of continuous variables too. By clicking on any of the bar-charts and dragging our mouse (while keeping the left mouse button pressed down), we can select any range of values along the x-axis.

This range can be then fine-tuned by dragging the "ears" of the highlighted range on either side. The whole range can also be shifted, by clicking on it and dragging it along the x-axis, left or right.

The same logic applies to the PCA scatter-plots. Here we can draw a rectangle around a group of samples by pressing down on the scatter plot and dragging our mouse in the desired direction (while keeping the left mouse button pressed down). This is called a sweep selection, and the edge of a selected rectangle can be individually modified just as we did with the histograms of continuous variables. Alternatively, the whole selection can be dragged around as well. Doing so will immediately update the pie and bar-charts on the left. Finally, to my knowledge, CorrMapper is unique in offering cross-filtering through sweeping on two scatter-plots simultaneously, i.e. by drawing a selection rectangle on both PCA plots.

The true power of CorrMapper's metadata explorer shows however, when we start to combine selections on different variables. In the example, shown in Figure 6.2, four filters were applied to the bariatric surgery data, which ultimately filtered the 173 samples down to 36:

1. Firstly, the male participants of the study were excluded.
2. Only females who are older than 45 years,
3. and had RYGB type of bariatric surgery were retained.
4. Finally, two more samples were excluded (shown within the red ellipse in Figure 6.2), using the PCA scatter-plot of the urinary NMR omics dataset.

Any of these filtering criteria could be individually reset by pressing the blue "reset" button next to a chart's title, or by clicking next to the sweep selection area on the scatter-plot. Alternatively, all of them can be reset together, by pressing the "Reset all" button at the top-left corner of the interface.

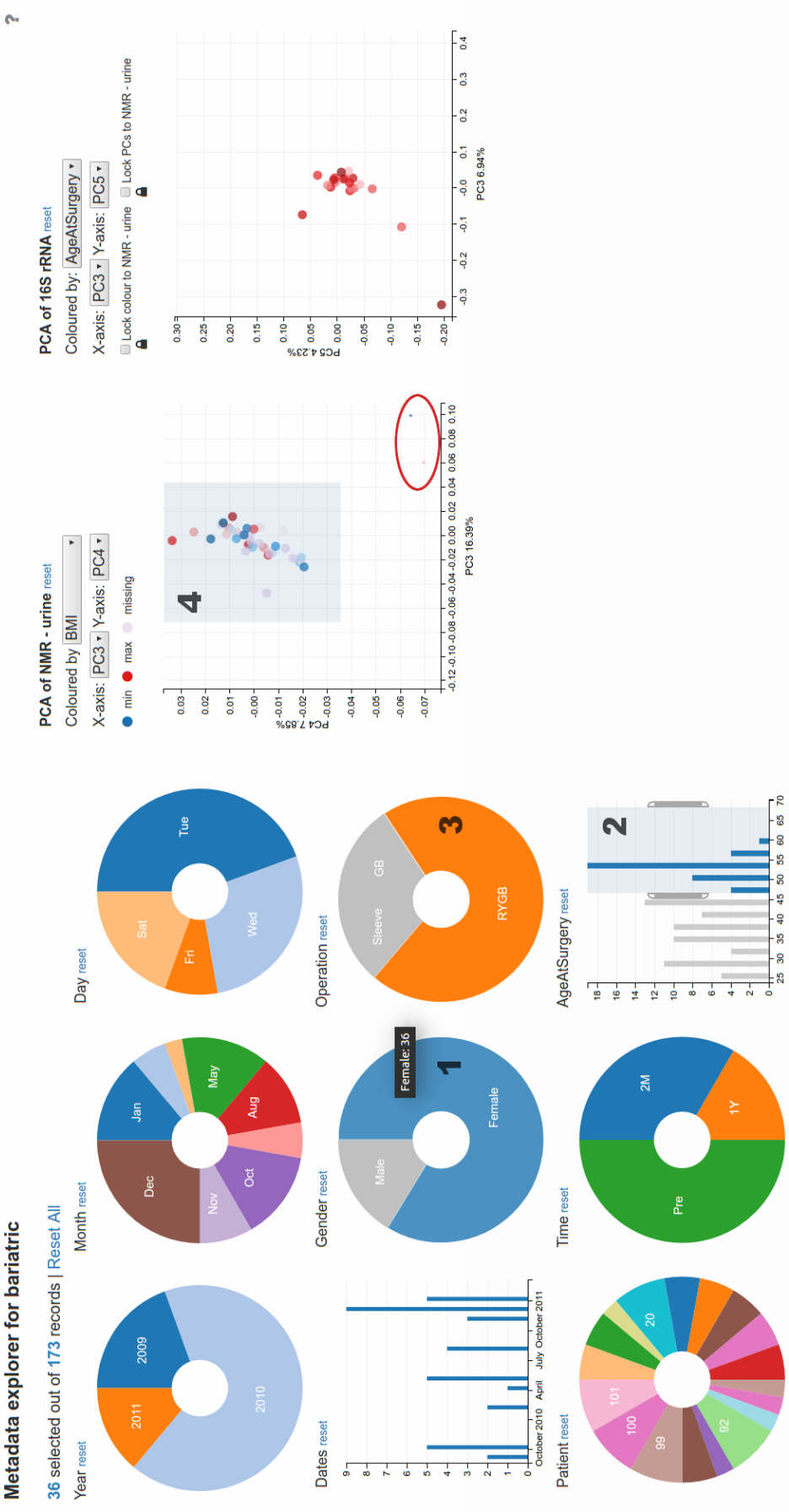


Figure 6.2: Cross-filtering in the metadata explorer. Across three different time-points, the bariatric surgery study has 173 samples. In this example, the total cohort was filtered down to 36 patients, by selecting females (1) who are older than 45 years (2), and had RYGB type of bariatric surgery (3). Additionally, using the sweep-selection (4) on the PCA scatter plot of the urinary NMR data, two more samples were also excluded (shown within the red ellipse in the bottom-right corner).

6.2.3 Responsive layout

One of the most attractive properties of interactive dashboards is their ability to represent a tremendous amount of information in a single, coherent, visual interface. In order to achieve this, dashboard designers have to strike a balance between visual density and clarity.

Too many components will result in a cluttered interface, with illegible legends and labels. Conversely, displaying too few components will waste valuable space on the user's screen, which will lead to an increased amount of scrolling, and a frustrating user experience.

Therefore it is imperative to consider the screen resolution of users' interfaces when we are layouting and rendering dashboards. CorrMapper achieves this, using the Bootstrap CSS library's grid system, which is able to render different column layouts of the dashboard, depending on the resolution of the receiving device.

Furthermore, CorrMapper combines Bootstrap with JavaScript, which allows it to redraw an already rendered dashboard without refreshing the page, if the resolution of the viewing port changes, e.g. when users rotates their tablets.

This can be easily tested by resizing the browser window, while browsing the metadata explorer. CorrMapper will automatically detect a new resolution and adjust the layout, size and position of each element in the dashboard.

All three of CorrMapper's visualisations modules were developed with this dynamic responsiveness in mind. Consequently, they can be rendered on smart phone screens or tablets, meaning users can explore CorrMapper's results even from their mobile devices while they are on the go. Nonetheless, feature rich omics datasets will naturally lead to visualisations with very high information content. Therefore, a full HD or larger resolution is generally recommended for an enjoyable user experience.

6.2.4 Metadata explorer in use

As shown in Figure 5.6, the metadata explorer plays a crucial part in the overall pipeline of CorrMapper. It enables users to explore their metadata file in conjunction with their omics datasets, through a unified dashboard interface. It helps to establish a meaningful mapping between (clinical) metadata and omics datasets, therefore it can inform users about which metadata column to use for feature selection in CorrMapper's data integration pipeline.

The interactive nature of the metadata explorer lends itself to iterative data exploration, which is ideal for getting to know a new dataset or finding novel associations in one that is already somewhat understood. However, as alluded to earlier in Section 6.1.1, the continued interrogation of a dataset through the interactive slicing and subsetting of the full cohort, can lead to spurious findings that are statistically under-powered and non-significant.

Although recently, a novel (and therefore barely cited) method was proposed to alleviate this problem via an adaptive multiple correction method[167], CorrMapper deals with it by purposefully not showing any statistics or corresponding significance level to the user. This simple strategy was also recommended via personal communications by Professor Emma Hall.

Despite these limitations, the metadata explorer is a valuable data exploration tool that could be used for the following:

- Check the number of samples belonging to a level of a categorical variable. By hovering our mouse over a slice of any pie-chart, the metadata explorer will display the number of samples belonging to that particular group.
- Check how extreme values of a continuous variable interact with other variables. In the bariatric surgery dashboard, we have multiple samples from the same patient, from different time-points (pre-surgery, two months and one

year post-surgery). A logical assumption of this study is that people will lose weight after the intervention. We can quickly check this, by selecting an extreme range of values on the BMI bar-chart (60-80), then slowly dragging this selection towards the minimum values (20-40). While doing so, we see that the Time pie-chart shows pre-surgery samples for high BMI values exclusively, but once the range of selected BMI values is dragged to the left, pre-surgery samples become the least frequent ones. With a few simple clicks, we managed to confirm that the overall majority of people in this cohort, do in fact lose weight after bariatric surgery.

- Discover associations between categorical variables. In the metadata explorer of the breast cancer dataset, African American females show a higher proportion of positive progesterone receptor status, when compared to Caucasian females. However, this can be due to the imbalanced sample sizes: 94 and 21 respectively.
- Check the distribution of continuous variables between different levels of a categorical variable. In the bariatric surgery study for instance, men have higher levels of haematocrit than females, which is in concordance with what we would expect[168].
- Discover interesting clusterings on the PCA scatter-plots by colouring the plots by different metadata variables. This can be made easier by locking the colour of the two PCA plots together. In the metadata explorer of the breast cancer dataset, the axis labels tell us that the first two principal components of the gene expression data explain 13.46% and 12.56% of the total variance respectively. As shown in Figure 6.3, when the colouring of this PCA plot is set to oestrogen receptor status, we see a clear separation between the positives and negatives, which means that samples with similar gene expression profiles tend to have the same oestrogen receptor status. This simple visual clue highlights the fact that these tumour types exhibit substantially different gene

expression patterns. Although this is something that has been well known for more than a decade[169], it is reassuring to see CorrMapper detecting it.

- The mapping of oestrogen receptor status onto gene expression data is the simplest connection CorrMapper can help to visualise. We could further stratify our cohort by selecting only the patients with a positive receptor status, then colour the remaining samples by any other metadata variable, therefore examining a joint relationship between two clinical features and an omics dataset.

Ultimately, the metadata explorer's utility is greatly determined by the depth of a researcher's domain specific knowledge and ability to place the interface's findings into a biological context meaningfully.

Users are encouraged to familiarise themselves with the interface's features before first using it, by clicking the question mark in the top-right corner, and going through the interactive help section. As pointed out earlier, this interface has its caveats and is not perfect. Hopefully the open-source community will extend it to provide more sophisticated statistical measures and hypothesis testing procedures. Nonetheless, as demonstrated above it can already help to identify biologically relevant associations between metadata and omics data. To my knowledge, CorrMapper is the first bioinformatics tool which does this through an algorithmically generated, interactive dashboard. This allows the metadata explorer to cater for the data exploration needs of a wide user base, including researchers without programming skills.

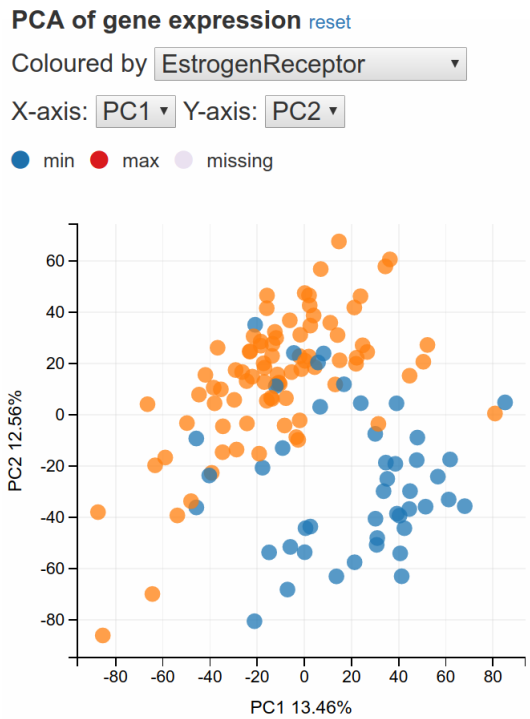


Figure 6.3: Mapping oestrogen receptor status onto gene expression. Orange: positive, blue: negative.

6.3 General network explorer

In the previous chapters we enumerated the statistical and computational tools that make up CorrMapper's back-end pipeline. To reiterate, CorrMapper estimates the conditional independence network of the features which were selected by the user's choice of FS method.

As explained in Section 5.4 and highlighted by Figure 5.7, while one uploaded omics dataset will result in a single network, two omics datasets yield three separate networks: a bipartite one (N_{12}) and two unipartite networks (N_1, N_2).

These networks are visualised by CorrMapper in a novel and engaging way. Since this visualisation module works with any kind of omics data, irrespective of whether the features have a genomic location or not, it is referred to as the general network explorer. It can be launched by clicking on the "Explore" button of a finished analysis on the profile page.

As we have seen, multiple steps of CorrMapper's pipeline were designed to alleviate the common problems arising from $p > n$ omics datasets: from filtering out low variance features, through using metadata driven feature selection to regularising the covariance estimation. Nonetheless, even with these preventive measures in place, we can arrive at very dense networks which are extremely hard to interpret.

The general network explorer was designed with these complex networks in mind, as it enables users to focus their investigation on subsections of large networks. These sub-modules can be either identified by the user or left to CorrMapper's clique finding algorithms to find them. Furthermore, to provide different viewpoints, the same network module can be examined through three different but complementary visualisation modes. These functionalities make the general network explorer an integral tool in CorrMapper's data exploration and integration pipeline.

6.3.1 Components of the general network explorer

As in case of the metadata explorer, due to the highly interactive and dynamic nature of the general network explorer, watching CorrMapper's demo video is strongly recommended before reading on. Furthermore, the guest account has several completed analyses, which will be referred to throughout the description of general and genomic network explorer modules.

Control panel

The general network explorer's control panel is located at the top left of the interface. If the analysed study has more than one omics dataset, CorrMapper will open its bipartite network (see N_{12} in Figure 5.7) by default, when we click the "Explore" button of an analysis on the profile page.

However, the "Load" drop-down menu of the control panel enables users to load the other two univariate networks (N_1 and N_2). These will open in a new browser windows that can be resized and positioned as needed, allowing for the simultaneous exploration of multiple networks. The other functionalities of the control panel will be introduced in the following sections, while describing the visualisation module.

Heatmap

Below the control panel, lies an interactive heatmap, which provides a quick overview of the network topology. Column and row labels display the names of the selected omics features. In case of a bipartite network, the row and column labels are different. Conversely, univariate networks are symmetrical, meaning they are mirrored over the heatmap's diagonal, therefore the column and row labels are matched.

Red cells represent positive correlations, while blue cells show negative ones. Left from the heatmap, a colour bar displays the full range of Spearman correlation values that could be found in the network. As we roll over any of the heatmap's cells, a triangle shaped tick indicates where exactly the given cell's correlation value lies within the range of all values. This might be easier and more accurate to read than shades of colours.

Importantly, once we hover over a heatmap cell, an information rich scatter-plot pops up displaying the following:

- Names of the two features that produced the correlation. Note, how simultaneously the corresponding column and row labels are also highlighted.
- Exact Spearman ρ and p-value found between these two features.
- A scatter plot of the actual data that went into producing the correlation. This scatter plot is coloured by the metadata variable that was used to select the features.
- If the metadata variable used for FS during the analysis is a categorical one, then the data points are divided into as many groups as many levels the variable has. Then a linear regression is fitted to each of the groups. For each group, the scatter plot display the fitted line, along with the bootstrapped 68% confidence interval, i.e. one standard error of the slope coefficient.

As we will see later, these scatter plots let us assess the quality of data behind each heatmap cell. They also help with the understanding and interpretation of individual associations within a network. These plots can be hidden with the "Hide scatters" button of the control panel.

The rows and columns of the heatmap can be reordered using the "Reorder" drop-down menu of the control panel. By default, the rows and columns of the heatmap are ordered so that the network modules identified by CorrMapper could be visualised easily (see more about this in Section 6.3.2). Nonetheless, other orderings of

the heatmap might be useful for identifying modules of tightly linked features. To facilitate this, CorrMapper offers three more options for reordering the columns and rows of the heatmap:

- Clusters of omics data type: sort rows/columns using hierarchical clustering, so that features with similar omics data values (e.g. gene expression) are rendered close to each other.
- Clusters of correlations: sort rows/columns using hierarchical clustering, so that features with similar correlation values rendered close to each other.
- Feature names: sort the rows/columns labels alphabetically or numerically. This option can be very useful for NMR datasets, where the feature names represent the position of each peak along the parts per million (ppm) scale. By ordering these NMR features names numerically we reconstruct the order of the peaks in which they appeared in the spectra, which makes the interpretation of the heatmap for a trained professional much easier.

Furthermore, if we hover over any of the row or column labels of the heatmap and keep our mouse there for half a second, both network visualisations will automatically redraw themselves to focus on the feature of interest. Doing so, immediately highlights a feature's location and neighbourhood in the network, which allows us to quickly gauge its relationships with other nodes and therefore possibly, infer its biological function or role.

Finally, by clicking on any of the row or column labels, we can sort the heatmap by the clicked feature's correlation values. Since hubs (i.e. a feature with a high number of connections) play a central role in scale-free biological networks[134], features that are negatively correlated with such a central feature should behave distinctively differently from features that are positively correlated. Therefore, this sorting feature might be very helpful, if we have a hub node, and we want to cluster the other features based on their positive or negative correlations with this central node.

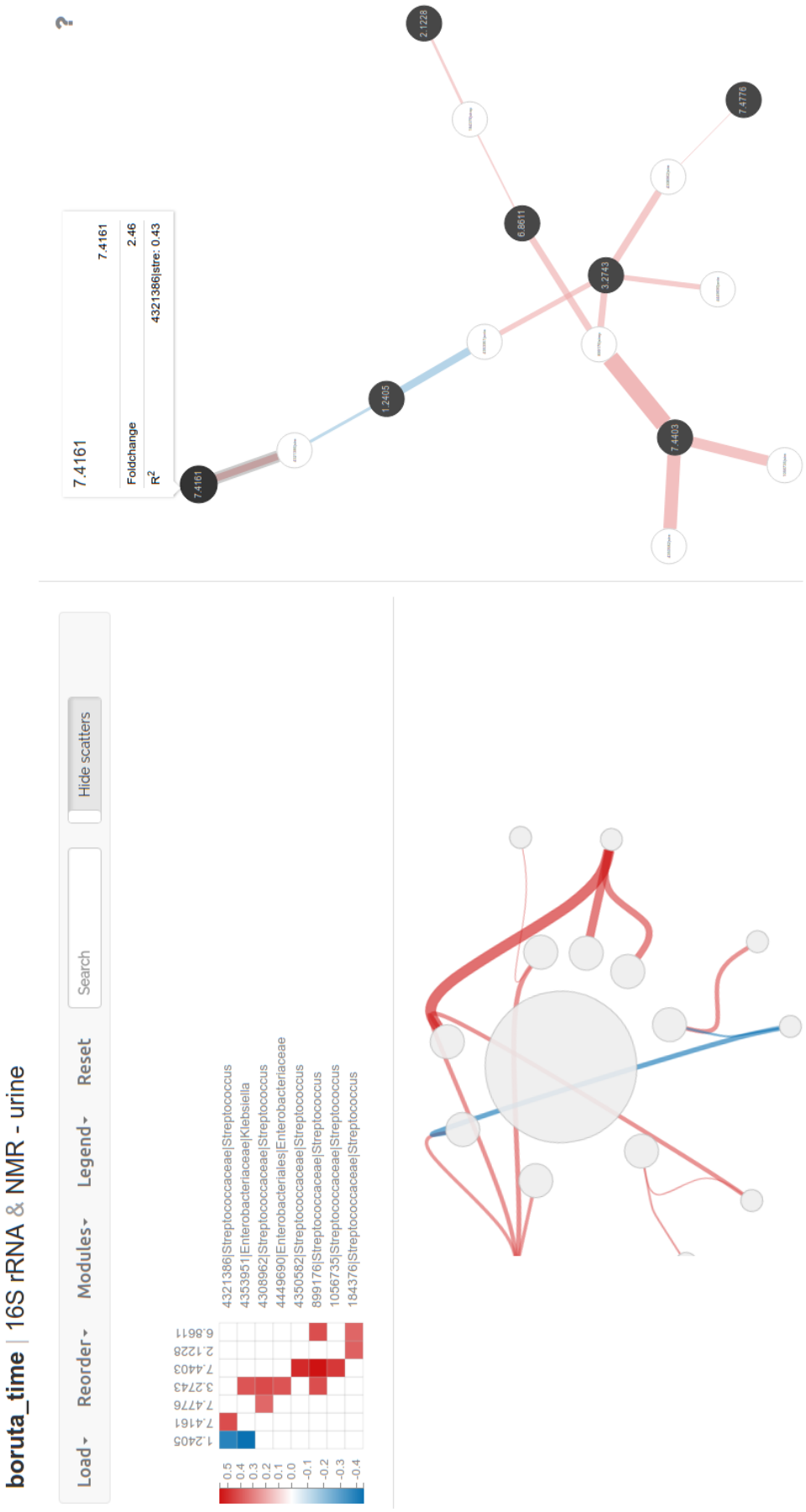


Figure 6.4: General network explorer of the bariatric surgery study. Boruta was used to select features using the time-point of samples as target metadata variable. At the top-left corner, a heatmap displays the bipartite network's topology in a tabular view, which can be filtered and ordered in several ways. At the right hand side, the same information is represented as a traditional network, while at the bottom a ring network is used to provide a third view-point. The colour of the network edges match the corresponding heatmap cell and correlation value. All three of these components are interactive and interlinked with each other.

Regular network

On the right hand side of the interface we find an interactive network, which - depending on the graph being visualised - is either bipartite or unipartite. In case of a bipartite network, the control panel's "Legend" drop-down menu clearly displays which node colour represents which omics data type.

Edges of the network match the heatmap's colouring scheme, and correspond to the strength and direction of correlations between features. Furthermore, this information is also emphasised by the width of each edge.

Although this network representation displays the same information as the heatmap, it does so through a different visual modality, which helps to identify larger connectivity patterns of the graph, including long, contiguous paths. Finding these by examining the matrix of a heatmap would be much harder.

By hovering over any of the nodes, a tooltip pops up with information about:

- The exact Spearman ρ values between this particular feature and its neighbours in the graph.
- If the metadata variable used for FS was a categorical one with two levels (e.g. Gender: female and male), this tooltip will also display the median fold-change between the two levels. The level which is alphabetically or numerically ranked first is used as the numerator, while the other level of the binary variable is the denominator. In our example: $medianFC = \frac{\text{median(female)}}{\text{median(male)}}$.

If we click on any of the nodes, the network zooms in to display only the clicked node along with its closest neighbours. Importantly, the non-neighbouring nodes that fall into this zoomed in display window are faded but still can be clicked.

Now, we can click any node within the display window (even non-neighbours) to pan the zoomed in region to that particular feature and its closest neighbours.

Alternatively, we can click the node we just clicked once again to reset the zoom. As we will see later, the zooming can be reset in two other ways as well.

Finally, by keeping our mouse over the network and scrolling with the mouse-wheel, we can zoom in on any part of the graph. Once zoomed in, we can pan to other sections of the network by clicking on its edges and dragging our mouse around.

This feature of the interface makes node labels legible in even extremely dense networks, while also helping with the exploration. The zoom can be reset to its default level, by clicking on the "Reset" button of the control panel, or pressing the Escape key on our keyboard.

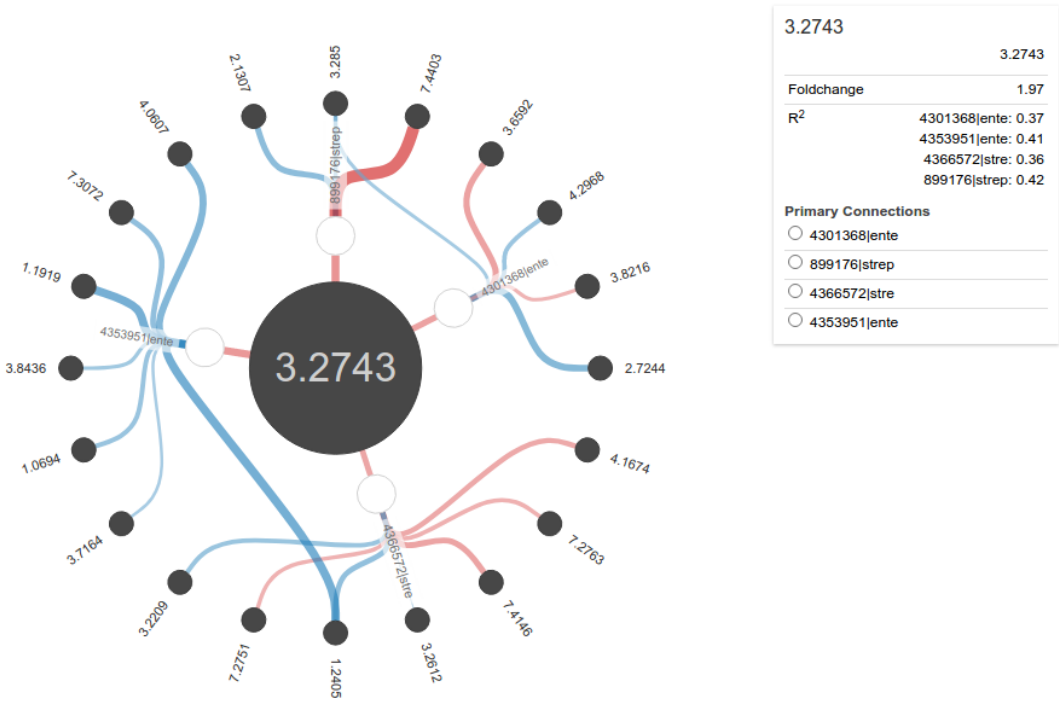


Figure 6.5: Ring graph of the general network explorer. At the centre of the graph we see the peak of Trimethylamine N-oxide (TMAO) from the "L1_time" analysis of the bariatric surgery dataset. This compound is positively correlated with four strains of bacteria, as listed by the legend in the top-right corner. These four OTUs are linked with further NMR peaks, displayed as the outer circle the graph. Each node can be clicked, which helps to quickly traverse the bipartite network.

Ring network

Below the heatmap we find a strange looking network, which comes to life once we click on any of its nodes. While the regular network displays only the closest neighbours of a clicked node, this ring network displays the closest and second closest neighbours of a clicked node. This is very valuable, especially in bipartite graphs, where the second closest neighbours of any node are by definition the same variable type as the clicked one.

Just as with the regular graph visualisation, the ring network displays a tooltip if we hover over any of its nodes. Importantly though, once a node is clicked, this tooltip gets fixed to the top right corner of this module, allowing users to hover over other nodes, while still seeing the exact Spearman ρ and fold change values of the previously clicked feature.

Finally, similarly to the regular network, this module also supports zooming and allows users to click any of the closest and second closest nodes once the network is zoomed in. This latter feature of the interface enables researchers to quickly traverse long paths of a graph with a few clicks.

6.3.2 Module selection with interlinked components

The real power of the general network explorer comes from the interlinked nature of the above described three components and from the interface's ability of focusing on any subsection of a graph. For instance, as mentioned earlier, if we click on any of the heatmap's row or column labels, both networks will focus themselves onto that particular feature. Furthermore, the clicked row or column will get reordered to display the Spearman ρ values of the selected feature in ascending or descending order (clicking on the same feature twice will toggle between the orderings).

In the following sections, we will see how the fluid integration of these three visualisation modules applies to sub-network selection, which is one of the key features of the general network explorer. Unfortunately, this cannot be captured by static figures, therefore the reader is strongly encouraged to use one of the guest account's analyses, to try out the user actions described in the following paragraphs.

Selecting modules found by CorrMapper

As alluded to before in Section 5.4.3, at the end of its pipeline, CorrMapper tries to identify interesting cliques or modules within the estimated conditional independence graphs. These modules represent closely linked features, which seem to belong together given the estimated network's connectivity patterns.

Note however, that these modules are derived using general graph theoretical considerations and no biological information or knowledge goes into their identification. Therefore, it is up to the user to verify the biological validity or value of these.

Modules identified by CorrMapper can be displayed by ordering the heatmap's columns and rows by modules (which is done by default when opening the general network explorer), and selecting any of the identified cliques from the "Modules" drop-down menu of the control panel.

Once a module is selected, its corresponding cells are highlighted within the heatmap by a black border. Furthermore, the labels of these features are also highlighted. At the same time, both networks are filtered to only show these connections and nothing else. If we now click on any of the regular or ring network's nodes we will be only exploring the edges that are within this module.

Given a complex and dense graph, this filterable aspect of the general network explorer becomes invaluable, as it allows researchers to quickly iterate over simpler and smaller modules of the network as opposed to assessing the whole graph at once.

Selecting user defined modules

The general network explorer also allows users to define any custom module and focus their investigation on this selected sub-graph. By clicking on any of the heatmap's cells and dragging our mouse to the right and down, we draw a rectangle which gets fixed once we release the left mouse button, see Figure 6.6.

Very importantly, we can add to this selection new rectangles by repeating the above described process while holding down the Shift key on our keyboard. This way, any combination of edges can be grouped together into a module or selection, to be explored separately from the rest of the network. Furthermore, by using the "Reorder" drop-down menu of the control panel, the heatmap can be rearranged in several ways to help with drawing these rectangular selections.

As in case of the modules found by CorrMapper, both networks are automatically filtered to only show the selected edges and features of the chosen sub-module. Any selection (including modules found by CorrMapper) can be cancelled either

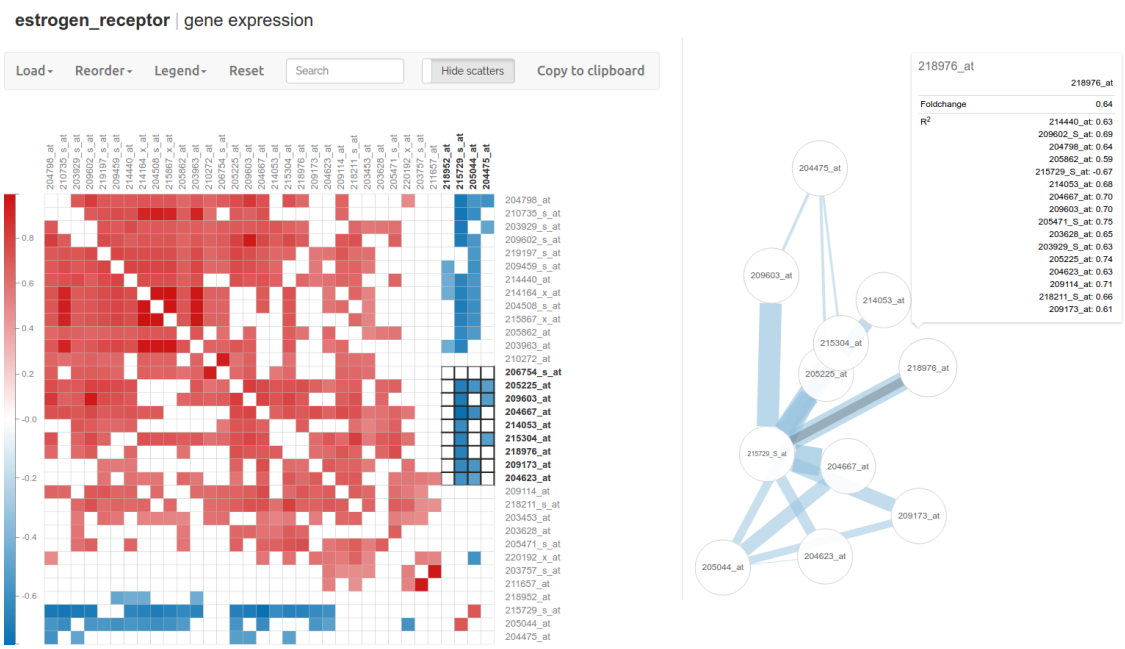


Figure 6.6: Selecting user defined modules in the general network explorer. A cluster of negative correlations are selected from the gene expression data of the "estrogen_receptor" analysis: see heatmap cells with black border. Consequently, the dense network is filtered and now this sub-module can be explored in isolation.

by clicking the control panel's "Reset" button, or pressing the Escape key on our keyboard.

Irrespective of whether we have selected a module identified by CorrMapper or a user defined one, by clicking the "Copy to clipboard" button, every feature in the clique will be pasted to our clipboard. This can be helpful in literature search, if we want to check whether these particular features have already been described together. Finally, the control panel's search bar allows users to quickly find any feature by their name. As we type in a few letters or numerals, all row and column labels which contain this search string will get highlighted instantaneously.

6.3.3 General network explorer in use

The aim of the bariatric surgery study was to increase our understanding of the biochemical changes that occur after weight-loss surgery, by capturing a comprehensive omics snapshot of obese individuals before and after their operation. As discussed in Section 2.1, bariatric surgery can provide numerous health benefits to patients, including resolving type 2 diabetic status and improving life-expectancy.

However, these procedures are fairly costly and can lead to postoperative complications. Therefore, a deeper understanding of the involved biochemical pathways is vital, as these can potentially lead to drugs and therapies which can induce the same metabolic changes and positive outcomes without surgery.

Obesity and bariatric surgery are both known to substantially alter the gut microbiota[74,170]. Consequently, this multi-omics study intended to characterise how the metabolism of patients change after weight-loss surgery, and identify cross-species metabolic interactions that might broaden our understanding of the complex interplay between the host and the gut microbiome.

The study included patients who underwent three radically different surgery types: RYGB, sleeve and gastric band. To reduce the cohort's heterogeneity for the analysis, only the RYGB patients were included, see "bariatric_rygb" dataset on the guest account's profile page. As we can see on Figure 6.7, when the samples are coloured by time-point, the 16S rRNA data already shows promising separation within the metadata explorer. Therefore, to assess the molecular and microbial changes that occur after the surgery, two analyses were run on the "Time" metadata variable using L1 SVC and Boruta for feature selection.

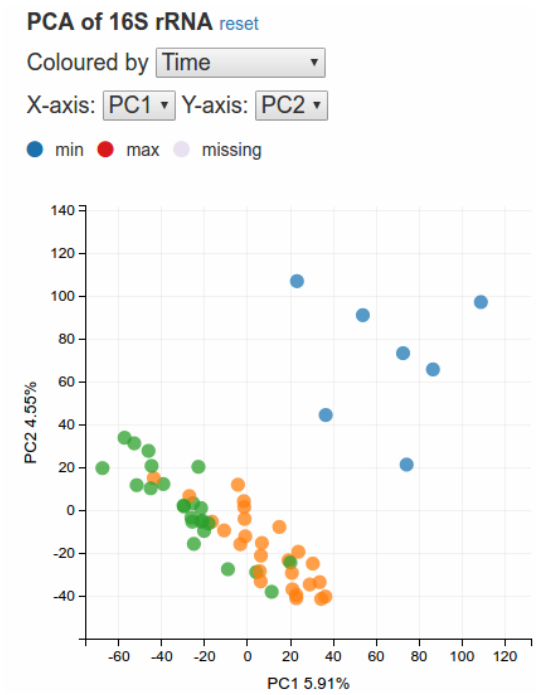


Figure 6.7: Mapping time-point of samples onto 16S rRNA data presents well formed clusters. This highlights the fundamental changes in microbial composition that occur after weight loss surgery. This plot only shows patients who underwent RYGB operation. Green: pre-op, orange: 2 months post-op, blue: 1 year post-op.

Please note that the bariatric dataset did not have enough (15 or more) postoperative samples from one year after the surgery. Therefore, CorrMapper's pipeline automatically excluded this level of the categorical outcome variable from the analysis. Consequently, all subsequent modelling was done on the preoperative and 2 months postoperative samples.

As expected from the benchmarking experiments (discussed in Section 5.2), L1 SVC selected a lot more features than Boruta. Therefore, opening the "L1_time" analysis from the guest account's profile page, leads us to a much larger and denser network than what we find by clicking the "boruta_time" analysis. Due to its network size, the former analysis can be used to test CorrMapper's module selection and network

filtering capabilities very effectively. However, the following paragraphs will focus on the results of the latter, as it is easier to interpret and navigate. Figure 6.4 displays the bipartite inter-omics network of the "bariatric_rygb" dataset with Boruta feature selection. The selected network is small and easy to navigate, yet as we will see it holds a surprising amount of biologically relevant information.

The features identified by Boruta as relevant are in agreement with platform specific modelling techniques. The DESeq2 algorithm[171] was used through the QIIME pipeline[172] to assess which OTUs are differentially abundant between the preoperative and postoperative samples. Seven out of eight OTUs identified by Boruta are in top twenty selection of DESeq2, see Table S4. This is remarkable, since DESeq2 is a highly sophisticated tool which is specifically designed for 16S rRNA data. It models the count data of OTU tables as negative binomial distribution and uses empirical Bayes shrinkage to estimate the fold-change between two group of samples. Boruta on the other hand is a general purpose feature selection method, yet it discovered several important OTUs that are in agreement with the results of DESeq2.

Furthermore, the NMR peaks selected by Boruta are in agreement with what was found by O-PLS modelling, see panel A of Figure S7. This O-PLS model found that the postoperative urine samples showed increased Phenylacetylglycine (PAG) and TMAO levels, while the concentration of creatinine and 3-Hydroxybutyric acid (3-HBA) was lower after surgery. This is in concordance with what has been reported in RYGB operated rats[173].

As shown in Figure 6.8, relevant peaks of PAG, TMAO and 3-HBA were correctly identified by Boruta. The peak at 2.1228 belonging to an unidentified compound was also indicated by O-PLS to be differentially abundant between preoperative and postoperative samples. This agreement of Boruta with O-PLS also highlights that the peak fitting pipeline (described in Section 2.1.1) has correctly identified and captured key spectral signatures of numerous relevant metabolites.

Figure 6.8: Boruta identifies relevant metabolites and OTUs. When assessing differentially abundant features between pre-op and postoperative samples, Boruta correctly identified (in agreement with platform specific modelling tools) six out of seven NMR peaks and seven out of eight OTUs.

Both PAG and TMAO have been indicated to be involved in the intricate metabolic interplay between human hosts and their microbiome. PAG has been previously described as a gut microbial cometabolite in mice[174], while TMAO is an oxidation product of Trimethylamine (TMA), which is produced exclusively by the gut microbiota in humans. TMA is derived from choline within these microbes.

TMAO is an osmolyte that the body uses to counteract the effects of increased concentrations of urea due to kidney failure, thus high levels of it can be indicative of kidney problems[175]. Therefore, the elevated levels of TMAO is surprising as weight loss surgery has been indicated to improve renal function in those with hyper filtration[176]. Furthermore, TMAO alters cholesterol metabolism as it increases its deposition within, and decreases its removal from peripheral cells such as those in the artery wall. This in turn explains its involvement in cardiovascular disease[177].

The high positive correlation between the OTU 4353951 and TMAO motivated a follow-up experiment, that was carried out by Dr. Jia Li, who is the lead investigator

on the bariatric surgery study. Dr. Li was able to culture *Klebsiella oxytoca* using a choline subtract, and repeated NMR profiling of the cell media demonstrated conclusively that this strain of bacteria can produce TMA, as the choline was used up and the concentration of TMA increased.

Consequently, we could hypothesise that the level of *Klebsiella* increases after bariatric surgery, which leads to higher TMA and ultimately TMAO production in patients. Given the above listed known biological effects of TMAO, one could aim to manipulate the gut microbiota of bariatric surgery patients to reduce TMAO production and ultimately lower their risk of postoperative cardiovascular diseases.

CorrMapper was designed to inspire novel testable hypotheses and follow-up experiments like the one described above. The fact that CorrMapper managed to identify biologically relevant features which are concordant with the findings of platform specific tools suggests that it might be a viable platform-agnostic data integration tool. Furthermore, CorrMapper managed to reduced the overwhelming complexity of this multi-omics study to a small network of cross-species metabolic interactions, of which several are biologically plausible and one has been experimentally verified.

Nonetheless, as panel B of Figure 6.9 shows, the positive correlation between OTU 4353951 and TMAO in Figure 6.8 stems from noisy biological data that does not fully conform with the above outlined hypothesis. The scatter plot is directly taken from the downloaded results of the "boruta_time" analysis, which is also automatically displayed if we hover over the relevant cell of the heatmap.

Even though the abundance of *Klebsiella* is clearly higher in postoperative samples, this is not necessarily true for the concentration of TMAO. In fact most postoperative samples seem to have similar levels of TMAO as the preoperative ones. This mismatch between our perception of the data (as informed by a single correlation statistic) and its real nature, highlights the importance of these scatter-plots and why colouring them by the target variable of feature selection can be illuminating.

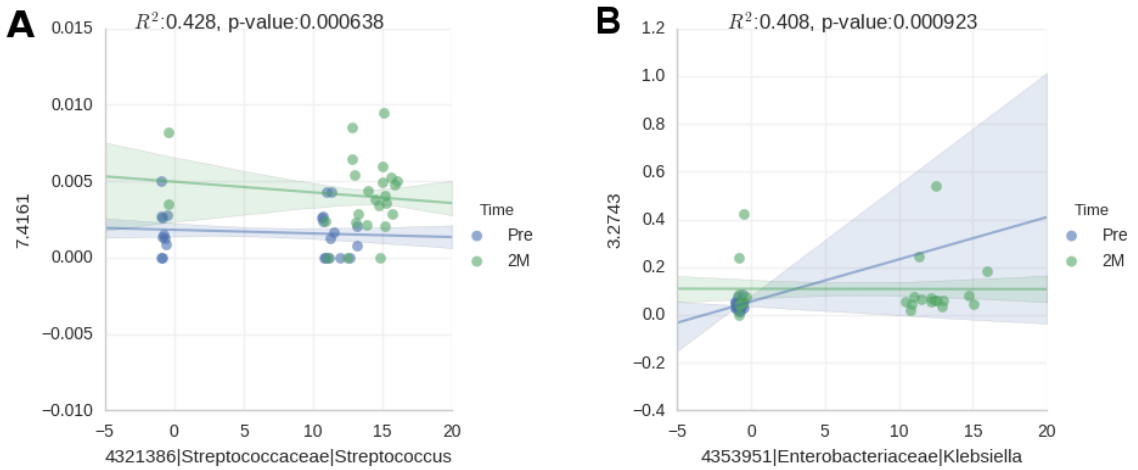


Figure 6.9: Interactions between urinary metabolome and gut microbiome in bariatric surgery patients. These scatter plots were taken from the downloaded results of the "boruta_time" analysis. **A:** Higher levels of this strain of Streptococcus is more prevalent in postoperative samples, which is coupled with higher concentrations of PAG. **B:** Although *Klebsiella* is more abundant in most postoperative samples, the concentration of TMAO does not follow this trend so clearly.

Panel A of Figure 6.9 shows another strongly positively correlated feature pair. However, in this instance, the association seems clearer as postoperative samples tend to have higher abundance of this particular *Straptococcus* strain and also higher concentration of PAG. The interpretation of these results is still ongoing and will be part of a publication describing the bariatric surgery study and its analysis.

In summary, the interactive graph interfaces of the general network explorer provide additional viewpoints that can lead to valuable insights. While the heatmap can be highly useful on its own, the bipartite network on the right hand side (see Figure 6.4) can expose interesting connections between features and modules that are seemingly distant when observed in a tiled, tabular view in a heatmap. This is especially true in larger networks such as the one produced by the "L1_time" analysis.

Although none of the components of the general network explorer are truly new or revolutionary, it could be argued that when combined together, they provide a powerful and novel way to interactively explore the complex correlation networks of omics datasets with great ease.

6.4 Genomic network explorer

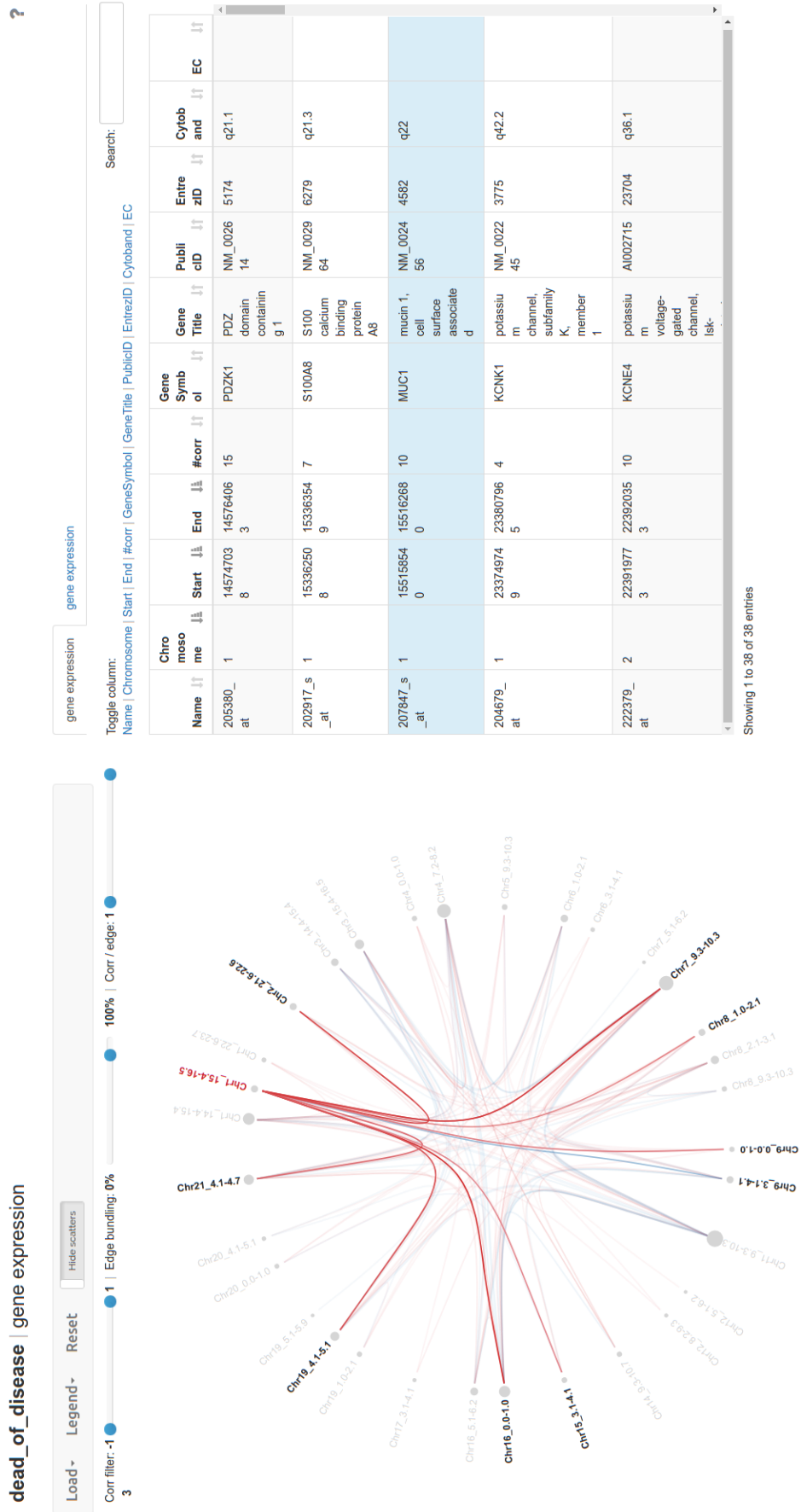
Although the general network explorer is a powerful data exploration interface, when omics features have genomic locations, they can be visualised in a much more engaging way by exploiting this information. By mapping the selected genomic features onto the studied species' circular chromosomal map, we not only obtain a visually pleasing chart that is laid out in a biologically meaningful way, but this arrangement also helps researchers (who have been often studying the given organism for years or decades) to quickly navigate and find certain features such as genes, methylation or single nucleotide polymorphism sites.

CorrMapper's genomic network explorer can be launched from the general network explorer's control panel by clicking the "Load" drop-down menu and selecting the "Genomic explorer". The following sections introduce the components of this interface and explain the close interplay between them. As with both previous visualisation modules, the reader is encouraged to watch the corresponding segment of CorrMapper's demo video and use the guest account to open the genomic network explorer for one of the analyses of the breast cancer study.

6.4.1 Components of the genomic network explorer

The genomic network explorer is composed of two main components which are fully interlinked. A circular graph is positioned on the left hand side, which displays the genome of the studied species. On the right, interactive, searchable and sortable tables complement this circular visualisation with annotation information.

The control panel in the top left corner of the interface is a simplified version of the general network explorer's menu, therefore to avoid redundancy, its functionalities are not reiterated here.



Name	Chromosome	Start	End	#corr	Gene Symbol	Gene Title	PubliclD	EntrezID	Cytoband	EC
205380_at	1	145747038	145764063	15	PDZK1	PDZ domain containing 1	NM_002614	5174	q21.1	
202917_s_at	1	153362508	153363549	7	S100A8	S100 calcium binding protein A8	NM_002964	6279	q21.3	
207847_s_at	1	155158540	155162680	10	MUC1	mucin 1, cell surface associated	NM_002456	4582	q22	
204679_at	1	233749749	233807965	4	KCNK1	potassium channel, subfamily K, member 1	NM_002245	3775	q42.2	
222379_at	2	223919773	223920353	10	KCNE4	potassium voltage-gated channel, Isk-	AI002715	23704	q36.1	

Figure 6.10: Genomic network explorer of the breast cancer study. The target for feature selection was the "DeadOfDisease" metadata variable. On the left hand side, the correlation network of gene expression probes is mapped onto a circular chromosomal map of the human genome. Red edges represent positively correlated genomic regions, while blue ones encode negative correlations. On the right, searchable and sortable tables provide detailed information about each probe. The two components are tightly interlinked and integrated: hovering over the MUC1 gene in the table highlighted its position in the chromosomal map on the left.

Circular network

The circular graph of the genomic network explorer displays a whole genome in clockwise fashion. As we have seen in Section 5.3.2, CorrMapper supports a number of commonly used model organisms, but for the remainder of this chapter we will assume that the reader is looking at data from a human study.

Since the human genome has more than 20,000 genes, we can end up with several hundred features to visualise even after employing feature selection. Furthermore, in a multi-omics study the number of selected genomic features can add up to much more than we can display in a circular plot without making it overly dense and therefore severely compromising the interface's visual clarity.

Therefore, CorrMapper bins the genomic features into 300 equally sized buckets. For instance, in case of the human genome, each bucket is 10 Megabase Pair (MbP) long. This binning process is done using a species specific chromosomal map that is pre-calculated based on the statistics found for the species' latest assembled genome in the UCSC Genome Browser[178].

Given the precise chromosomal location of a given genomic probe, CorrMapper decides which feature belongs to which bin based on these chromosomal maps. This is the reason why on the upload page, CorrMapper asks the users to specify the species of their study.

In the breast cancer study, we will find the buckets of the first chromosome starting at 12 o'clock. As we move clockwise around the circular plot, we see the buckets of human autosomes laid out in order, and at half 11 we arrive at the X chromosome's buckets. Although breast cancer affects men too, this particular study only recruited female participants, therefore we do not see any features mapping to the Y chromosome. Furthermore, to aid visual clarity, buckets of the same chromosome are grouped closer to each other.

The label of each bucket displays not just the chromosome's name or number (depending on species specific convention), but also the start and end position of the bucket in MbP. The size of each node represent the number of features that mapped to this particular genomic bucket. Note, that in case of a bipartite graph, features could map to the same region from both omics data type.

Due to the above described binning process, the network's edges most often do not represent a single correlation between two features, but instead numerous correlations between several features, mapped to two genomic regions. Therefore, the colour of each edge cannot be easily equated to a single correlation value.

In fact the edge colour is calculated by mapping the mean of correlation values within an edge onto the blue-red gradient that is used in the general network explorer. Consequently, red edges connect genomic regions (10 MbP long in case of humans) which are mostly positively correlated, while blue edges link negatively correlated regions. Purple edges on the other hand hold a mix of these. Therefore, the circular network provides a high level overview of the connectivity patterns between different regions of the genome.

Similarly to the general network explorer, this interface was designed with dense graphs in mind. Consequently, the visual clarity of an overly dense or complex genomic network can be increased in several ways:

- Hovering over any of the nodes will highlight its links, while all other edges become faded. This helps to visualise where exactly the connections of a certain genomic region lead.
- By using the "Corr filter" slider above the network, the edges can be filtered to only retain a certain range of mean correlation values. This simple tool enables users to assess which regions of the genome are positively or negatively correlated, by adjusting the left and right tick of the slider respectively.

- A circular plot with straight edges is very uneasy on the eye that gives a visually cluttered feeling. To alleviate this, CorrMapper uses the hierarchical edge bundling capabilities of `d3.js`, which bends the network's edges so that links heading in similar direction are drawn closer to each other. The "Edge bundling" slider can be used to adjust this: 0% corresponds to the straight edges, and 100% results in heavily bundled edges. We recommend users experiment with this until they find a visually pleasing representation of the graph.
- Finally, edges can be filtered based on the number of correlations contained in them. For instance, by filtering out the links with a single correlation, this functionality helps to focus our attention on strong edges which connect heavily connected and correlated genomic regions.

Interactive tables

While the circular graph on the left provides an excellent high level overview of the connectivity patterns between large stretches of genomic regions, the right hand side is occupied by two interactive, searchable and sortable tables that enable researchers to dig deeper and assess the individual correlations. These tables are automatically filled with the annotation uploaded by the user, therefore they can hold any type of valuable information about the features.

For the binning of genomic features CorrMapper requires users to provide a name, chromosome number, start and end position for each probe. These are mandatory fields, without which CorrMapper cannot bin, and consequently cannot map features onto the circular genomic network. Therefore, while checking the uploaded files, CorrMapper will discard any genomic feature from the omics data files which do not have a precise location within the provided annotation files.

However, as mentioned above, users can provide any sort of extra information for their features as additional columns of the annotation file. Table 6.1 describes the breast cancer dataset's annotation files.

Omics dataset	**Annotation column**	**Description**
GE	Name	Mandatory annotation field. Name of the gene expression probe.
	Chromosome	Mandatory annotation field. Chromosome on which the probe is located.
	Start	Mandatory annotation field. Start position (in basepairs) of the region to which the probe hybridises.
	End	Mandatory annotation field. End position (in basepairs) of the region to which the probe hybridises.
	GeneSymbol	The colloquially used nickname of the gene. Often the same as the gene's formal name.
	GeneTitle	The gene title comes from the `DEFINITION` line of the record within the GenBank database[179].
	PublicID	The accession number assigned to the gene by NCBI.
	EntrezID	The gene's unique identifier in the Entrez database.
	Cytoband	Name of the chromosome's cytoband to which the gene belongs to.
	EC	Enzyme Commission number for the numerical classification of enzymes.[180]
CNV	Name	Mandatory annotation field. Name of the copy number variation probe.
	Chromosome Start End Cytoband	See gene expression part of table.
	LinkedFeature	Genes that are linked to the CNV probe's genomic region.

Table 6.1: Description of the breast cancer study's annotation fields. Within this dataset we have two omics data types: Gene Expression (GE) and Copy Number Variation (CNV). The information for these fields were obtained from the vendor's website and the GenBank database. The annotation data was cleaned then collated together into two files using Python scripts. Mandatory annotation fields (Name, Chromosome, Start, End) are required by CorrMapper's feature binning process.

In case of a bipartite network, the two tables hold different omics data types. However, the reason for having two separate tables even for unipartite networks, will become clear in the next section. We can switch between these tables by either clicking on their tabs at the top (labelled as "gene expression" and "copy number variation" in the breast cancer study) or by simply using the left and right arrows of our keyboard.

By default, both tables are ordered according to their features' genomic locations using the Chromosome, Start and End columns. The ordering of tables can be easily changed however, by clicking the icon (depicting two arrows pointing in the opposite directions) next to any column header. Pressing the sort icon multiple times will switch between ascending and descending orderings. Furthermore, by holding down the Shift key, the tables can be sorted simultaneously by multiple columns.

The search bar, located at the top right corner of the interface, allows researchers to quickly find any genomic feature by their name or any other additional information field uploaded with the annotation files. As the two tables have separate search bars, they can be filtered completely independently using different queries.

Finally, if the user provided a lot of additional information columns in the uploaded annotation files, these tables can become quite dense and hard to navigate. This problem can be alleviated by toggling the visibility of certain columns, using the "Toggle column" menu bar above the tables. Columns which were set to be invisible are still searchable however.

6.4.2 Interlinked components

Similarly to CorrMapper's previously introduced other two visualisation modules, the genomic network explorer gains most of its power from the closely interlinked nature of its components. While the network provides a zoomed out overview of the

genome, displaying major connectivity patterns of different chromosomal regions, the tables complement this and act as magnifying glasses that allow researchers to zoom in on any region of the graph of genomic correlations.

Hovering over a row of the tables displays this selected feature's location in the genome by highlighting its node and connections. Since the tables are order accord-

gene expression | copy number variation

Toggle column: Name | Chromosome | Start | End | #corr | GeneSymbol | GeneTitle | PublicID | EntrezID | Cytoband | EC

Search:

Name	Chromosome	Start	End	#corr	GeneSymbol	GeneTitle
206373_at (1)	3	147127173	147132348	3	ZIC1	Zic family member 1 (odd-paired homolog, Drosophila)
205242_at	4	78432905	78532986	1	CXCL13	chemokine (C-X-C motif) ligand 13

gene expression | 206373_at (2)

Toggle column: Name | Chromosome | Start | End | #corr | GeneSymbol | GeneTitle | PublicID | EntrezID | Cytoband | EC

Search:

Name	Chromosome	Start	End	#corr	GeneSymbol	GeneTitle
206373_at	3	147127173	147132348	3	ZIC1	Zic family member 1 (odd-paired homolog, Drosophila)

Correlated feature	R value	P value
BAL12B2624	-0.439	0.0000057
RP11-109L8	-0.350	0.0001532
RP11-213L15	0.340	0.0003955

gene expression | 206373_at

Toggle column: Name | Chromosome | Start | End | #corr | LinkedFeature | Cytoband

Search:

Name	Chromosome	Start	End	#corr	LinkedFeature	Cytoband
BAL12B2624	12	79315815	79316148	6	G08936, SHGC-17533	12q14
RP11-109L8	12	63780799	63780948	5		
RP11-213L15	22	12465600	12465833	5		

Figure 6.11: Navigating the tables of genomic network explorer. **Top:** Clicking on the ZIC1 gene (denoted with the first red circle) will open its sub-table and show all CNVs that are correlated with it. **Middle:** At the same time, the right tab at the top of the tables changed from "copy number variation" to "20673_at". **Bottom:** Clicking on this tab (denoted with the second red circle) will open the other table which now lists all three of the correlated CNVs.

ing to genomic location by default, hovering over the first row will highlight the first chromosomal bucket, i.e. a node which is at almost 12 o'clock. Then, as we hover over rows further down, nodes and their edges light up in clockwise fashion.

Clicking on any of the genomic features X_s in a table T_1 causes the following:

- The features which are correlated with the selected one X_c, are displayed in a sub-table below the clicked row. This sub-table holds the names of X_c, along with the Spearman ρ and p-values of their correlation with X_s, see Figure 6.11.
- Hovering over any of the sub-table's rows displays the same information rich scatter-plot that was introduced in Section 6.3.1 while discussing the heatmap component of the general network explorer. This allows users to assess the data of each feature-pair and examine how the selected metadata variable maps onto them. The control panel's "Hide scatters" button could be used to hide these figures.
- The sub-table's rows cannot be clicked. Instead, clicking on X_s re-renders the other interactive table T_2, which gets populated with the annotation of X_c. This is also signified by the changed label of the other table's tab at the top, which now says X_s. We can easily access T_2 by either clicking on its tab at the top or by using the left or right arrow keys on our keyboard.

  Once we are at T_2 we can examine the annotation of each feature within X_c, or check their location in the genome by hovering over the rows if T_2. Furthermore, we can also click any features within T_2, which now will become X_{s2}, and consequently a sub-table will be draw below it with its correlated neighbours X_{c2}. The label of T_1 will change X_{s2} and the features of X_{c2} will be used to populate T_1.

  This process of jumping back and forth between the two tables by clicking on correlated features of X_s, can continue infinitely and it provides an easy way to traverse the circular graph one neighbouring variable-pair at the time.

Importantly, clicking on X_s permanently highlights its position and connections within the circular network. This means that if we now hover over other rows of T_1, even though their location is still highlighted within the network, but once we roll our mouse out of the table's area, X_s gets highlighted again automatically. The network can be brought back to its default state by either clicking the "Reset" button of the control panel, or by pressing the Escape key on our keyboard.

6.4.3 Genomic network explorer in use

Three analyses have been performed on the breast cancer study to demonstrate the genomic network explorer, using the following metadata variables as target for feature selection: "DeadOfDisease", "TumorStaging" and "EstrogenReceptor". These analyses could all be accessed from the guest account's profile page. The interpretation of these results is still in progress by our collaborators, and will be part of CorrMapper's paper. Therefore, in this section we only take a very brief look at one particular preliminary result, which also requires further investigation.

The "dead_of_disease" analysis revealed the Mucin 1 (MUC1) gene as a relevant contributor to patient outcome. This gene is a heterodimeric protein, formed by two subunits, which has been shown to be aberrantly overexpressed in human breast and other cancers. Its transmembrane MUC1 C-terminal subunit (MUC1-C) functions as a druggable oncoprotein, which contributes to the activation of several key cell signalling pathways (PI3K, AKT, MEK, ERK)[181]. As the genomic network explorer reveals, this gene is positively correlated with three amplified CNV probes on the first chromosome, while negatively correlated with two probes, located on the eleventh and sixteenth chromosomes. Figure 6.12 highlights the relationship between the MUC1 gene (probe 207847_s_at) and one of the amplified CNV probes (RP11-131M16). As we can see, patients who died during the study period, exhibited an even stronger positive correlation than the rest of the cohort.

Figure 6.12: Positive correlation between genomic amplification and overexpression of MUC1 gene. The MUC1 gene is a known oncoprotein, whose expression exhibits a strong positive correlation with three amplified CNV probes along the first chromosome. Here, the RP11-131M16 CNV probe is examined. By clicking on it in the "copy number variation" table, and hovering over the MUC1 gene, the two features' scatter-plot is displayed. This plot suggests that this positively correlated relationship was even stronger in patients who passed away during the study, hinting at the high relevance of this relationship.

6.5 Summary

CorrMapper was designed to be a multi-faceted research tool that focuses equally on multi-omics integration and advanced data visualisation. Although, in recent years, numerous novel visualisation tools have been proposed for biomedical research, few of them can match the level of interactivity and overall versatility of CorrMapper. The following paragraphs briefly introduce several interesting tools whose functionality partially overlaps with CorrMapper's visualisation modules.

Cytoscape is an offline network exploration and interactive visualisation tool, that has proven to be very popular in life science research[159]. Due to its numerous features and functionalities, Cytoscape is a fairly complex application, which requires substantial amount of learning from the user. Furthermore, Circos[182] is a very capable visualisation package that can be used to produce circular network plots, which are very information dense and resemble the chromosomal map of the genomic network explorer. However, Circos can only generate static, non-interactive figures, and requires programming knowledge from the user.

iCanPlot uses the Canvas element of the HTML5 standard to visualise gene expression data as an interactive scatter-plot in the web-browser. Given a dataset of thousands of genes, researchers can interactively select a subset of them and the tool will calculate gene set overlap statistics using these genes. The software is available online at www.icanplot.org[183]. Although the online availability and interactivity of iCanPlot is appealing, it is highly specialised for gene expression data and cannot be used for multi-omics studies.

jHeatmap is an open-source JavaScript library that allows for the interactive exploration, sorting and filtering of tabular biological data[184]. The package implements these functionalities as a set of JavaScript classes, which could be used to build

powerful and highly interactive heatmap visualisations. However, doing so requires deep expertise in programming and JavaScript.

ReconMap uses the Application Programming Interface (API) of Google Maps to display the genome-scale reconstruction of human metabolism in a web interface[185]. This manually curated and crafted metabolic network is available at http://vmh.uni.lu, and it represents an invaluable resource of information for biomedical researchers, that can be browsed interactively. Additionally, users can manually overlay their simulated or real omics data onto the graph, by specifying a different colour and thickness to each node and reaction. Therefore, ReconMap can be a very useful for human studies, once the exact multi-omics interactions we want to visualise are already identified, however, it cannot help with the discovery of these connections.

The recently published NaviCom[186] is a very capable multi-omics data visualisation platform, available at https://navicom.curie.fr. It relies on cBioPortal[187], which collects and catalogues large-scale cancer studies with expression data for mRNA, microRNA, proteins, mutation, gene copy number and methylation profiles. NaviCom does not require any programming knowledge, and allows users to select subgroups of patients based on their metadata. Once a multi-omics study is selected from cBioPortal, NaviCom will automatically map the disparate omics data types onto a manually created molecular network from the Atlas of Cancer Signalling Network (ACSN)[188] or the NaviCell database[189]. The mapping of omics data onto the chosen biochemical network is done using bar-charts, heatmaps and glyphs depending on the type of the visualised omics dataset.

Users can interactively traverse, search and explore this highly information-rich map, which contains omics data from multiple analytical platforms. Although NaviCom is a truly exceptional research tool and a fine example of advanced multi-omics visualisation, users are currently restricted to studies about human cancer, which have already been published and deposited in either ACSN or NaviCell.

Despite their numerous merits, all the above described methods are highly specialised for either certain data types, species, diseases or a distinct type of graphical outputs. Although limiting the scope of these projects allowed their creators to tailor their solution to a particular visualisation problem, naturally, this also restricted the overall utility and applicability of these tools in a wider context. For example, both ReconMap and NaviCom focus on human metabolism only, and consequently cannot incorporate metagenomics data or contribute to the actively researched field of host-microbial metabolic interactions[190].

Conversely, CorrMapper was designed to be data type agnostic, and this principle was also guiding the development of its visualisation modules. Therefore, CorrMapper is similar to mixOmics, whose authors worked hard in recent years to improve the interpretability of latent variable models such as CCA and PLS, through novel visualisation modes[162].

For instance, given the first two latent variable dimensions of a fitted PLS model U_1 and U_2, mixOmics uses correlation circle plots to visualise the similarity between features. This is done by projecting each feature X_j onto U_1 and U_2, and plotting it as a vector with $cor(X_j, U_1)$ and $cor(X_j, U_2)$ as x and y coordinates respectively.

The angle between these vectors can be interpreted as the correlation of two features within the U_1, U_2 space. The angle is sharp if two features are positively correlated, obtuse if the correlation is negative, while the angle is right if the correlation is zero.

Although this allows researchers to quickly identify positively and negatively correlated features from a single plot, the associations discovered in these graphs only pertain to the chosen latent dimensions. This might not be a problem if for example the first two score vectors capture the majority of variation in the datasets. However, if this is not the case, then the interpretation of the overall relationship between two biological features remains a challenging problem in latent variable models.

Similarly to CorrMapper, mixOmics can generate networks and hierarchically clustered heatmaps from the above described latent variable correlations. It relies on Cytoscape to visualise networks, and it can only build bipartite ones, potentially missing interesting associations between the features of a single omics dataset within the N_1 and N_2 networks. More importantly, mixOmics builds marginal correlation networks using a user specified threshold. As discussed earlier in Section 3.4.2, in $p > n$ datasets, this graph construction method leads to dense networks with a lot of spurious correlations, and as the benchmark experiment of Section 5.4.5 demonstrated, the performance of mixOmics was often inferior compared to CorrMapper.

Furthermore, due the above described mathematical definition of these correlations, users need to examine a separate network or heatmap for each pair of latent variables, which can make the interpretation of these models cumbersome. Finally, all graphical output of mixOmics is static, therefore dense networks and heatmaps cannot be explored, filtered, searched or understood in detail.

With the exception of mixOmics and NaviCom, all of the above mentioned solutions are primarily visualisation tools which leave the majority of data analysis and coding to the researcher. Furthermore, several of them are specific to certain omics data types and can only work with human studies. Conversely, CorrMapper's strength stems from its meta-tool nature and how it seamlessly combines its data integration pipeline and visualisation capabilities.

Although this is true for mixOmics as well, CorrMapper excels at providing engaging and interactive visualisations without any programming knowledge, while in some areas (automatically generated metadata explorer) its visualisation capabilities are currently completely unparalleled. Finally, as noted earlier, CorrMapper is currently in preparation for publication. In the meanwhile, the valuable feedback of numerous beta testers have been used to improve various parts of the user interface, and debug edge cases within the visualisation modules.

7 | Conclusions

Multi-omics studies present unprecedented opportunities to capture multi-modal molecular signatures arising from different compartments of biological organisms. Furthermore, they can provide a more holistic view of biochemical pathways and greatly enhance our mechanistic understanding of health and disease. Therefore, the research and development in this PhD was carried out recognising the need for novel tools that facilitate the exploration of such multifaceted and complex studies.

CorrMapper is a robust multi-omics data integration pipeline coupled with highly advanced, interactive visualisation modules, which enable researchers to interrogate intricate correlation networks with great precision. The project is fully open-source, and freely available as an online research tool for academic use.

CorrMapper was designed to solve or alleviate several problems commonly encountered during the integration and visualisation of complex biological datasets:

- In almost all cases, omics datasets have several times more features than samples ($p \gg n$). Extensive benchmarking experiments were run to choose an optimal set of feature selection algorithms, that can filter down the p candidate features to the biologically relevant ones, which are the most discriminative with respect to a clinical metadata variable.
- Unfortunately, even after feature selection, we are often left with $p > n$ or $p \approx n$datasets. This is especially true if we have to concatenate more than one omics datasets. Estimating the correlation network from such wide data leads to spurious correlations, imprecise networks and ultimately false discoveries.

Furthermore, traditional estimation techniques, such as marginal correlation networks cannot recover the true dependence structure of features, and often label conditionally independent feature pairs as correlated. This leads to overly dense networks with many false positive edges.

To alleviate this problem, CorrMapper uses the non-paranormal extension of the graphical lasso algorithm, to estimate the conditional independence network of the selected features. Additionally, the StARS algorithm is used to chose the optimal regularisation parameter for the graphical lasso. Finally, CorrMapper estimates the statistical significance of the network edges in a robust, non-parametric way by calculating Spearman correlations with permutation testing.

- Despite the feature selection, and regularised network estimation process, the graphs arising from biological datasets can still be extremely complex and hard to interpret. Therefore, CorrMapper employs interactive visualisation modules that allow users to subset these large networks to smaller sub-graphs or modules and interrogate them in isolation.

  Datasets with genomic probes can benefit from CorrMapper's genomic network explorer which exploits the location of these features and maps them onto a circular chromosomal layout.

- The prevalence of clinical metadata in biomedical research necessitates the development of novel tools that enable the exploration of these additional information sources in conjunction with omics datasets. CorrMapper's metadata explorer provides a novel dashboard interface that allows for metadata driven stratification of two omics datasets simultaneously.

- The overwhelming majority of bioinformatics tools are command line based and expect some form of programming knowledge from the user. Although this cuts down development time and can often lead to more modular software, it restricts the tool's audience. CorrMapper was designed to be accessible to a

wide range of users, from clinicians to bioinformatics researchers. Its codebase is open-source and modular enough to be easily modified and extended by power users, but at the same time, its graphical user interface enables the technically less capable users to benefit from its data integration pipeline and visualisation modules.

- Finally, turning a piece of scientific software into an online research tool can add months of overhead to a project, due to the steep learning curve of the technologies involved, and numerous technical hurdles one has to overcome.

  Therefore, to capitalise on the hundreds of hours of work that went into making CorrMapper's front-end, ScienceFlask was created, which is an open-source template for scientific web application development. This project allows researchers from all fields to turn their offline scientific tool into a feature rich online application within hours, and run it in the cloud.

Although CorrMapper attempts to alleviate the above listed problems, it is far from providing a perfect solution to all of them. However, as data acquisition costs decrease further, it is highly likely that we will see $p \gg n$ studies turning into $p > n$ or even $p \approx n$ datasets within the next ten years. This will naturally increase the performance of many data integration tools, including CorrMapper's.

Until then, it is highly recommended to not rely on a single method for multi-omics integration and visualisation, but use different and possibly complimentary ones (CorrMapper and mixOmics for instance), and attempt to derive parsimonious results from them.

7.1 Future work

While CorrMapper is already fully functional and hopefully useful for researchers, it is only in its first iteration, waiting to be developed further and made better. Although this PhD has come to its end, the project is open-source, thus it can be forked, altered, extended and improved by bioinformaticians all around the globe.

The following list outlines possible directions for future work:

- The general network explorer allows users to copy the features of a selected network module onto their clipboard, while, the genomic network explorer can display detailed information about each feature, based on the uploaded annotation files.

  However, the interpretation of the conditional independence networks would be much easier if CorrMapper would plug into the search APIs of EBI and NCBI to automatically gather and display information about features of interest. This extension would require the modification of the general and genomic network explorers' JavaScript files, and adding a new "Search EBI" tab to the HTML pages of both interfaces.

- The metadata explorer is currently using PCA to obtain a lower dimensional representation of the omics samples. Metadata features are then mapped onto the first five principal components of this lower dimensional projection to discover target variables for feature selection.

  Although this is effective, it is a suboptimal solution, as PCA was not designed to discriminate between classes. PLS-DA would be a more adequate choice, as it would not only project the omics data into a lower dimensional space, but also maximise the separation between the levels of a categorical metadata variable.

- Feature selection is currently carried out on each omics dataset individually, as this strategy was deemed the most economical and reasonable, given the limited number of intersecting samples in today's multi-omics studies. However, users could be given the freedom to concatenate the two datasets prior to feature selection and use their joint probability distribution for FS.
- With extensive development, a prediction layer could be added to CorrMapper, that enables users to train powerful machine learning algorithms on the selected features. However, to avoid over-fitting, this would require a complicated doubly nested cross-validation scheme. Therefore, probably only larger studies with $n > 500$ would benefit from this new predictive component.
- As noted in Section 5.3.4, visualising very large networks in CorrMapper currently can prove to be computationally demanding, as no graphics card acceleration is used to generate and dynamically render these complex graphical objects on the user's screen.

  However, this is an intensely researched topic and JavaScript libraries such as the recently published Stardust[191] can utilise GPU acceleration to dynamically render and interactively visualise large networks with reduced lag and a satisfying user experience.

  Owing to the feature selection procedures of CorrMapper, most of its visualised networks are small enough to be handled by JavaScript libraries relying on CPU only. However, in the absence of any feature selection, the visualisation of the resulting large and dense networks could be supported by the extension of CorrMapper's front-end with Stardust.

Supplementary materials

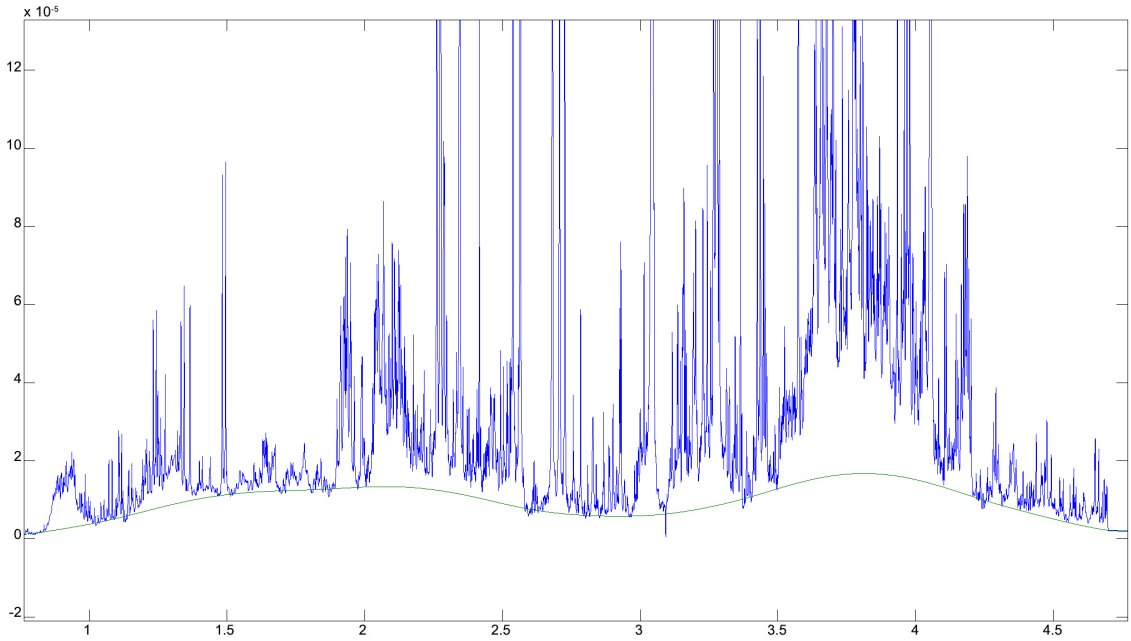


Figure S1: An example of fitted baseline (green), in one of the critical regions of a urine NMR spectrum.

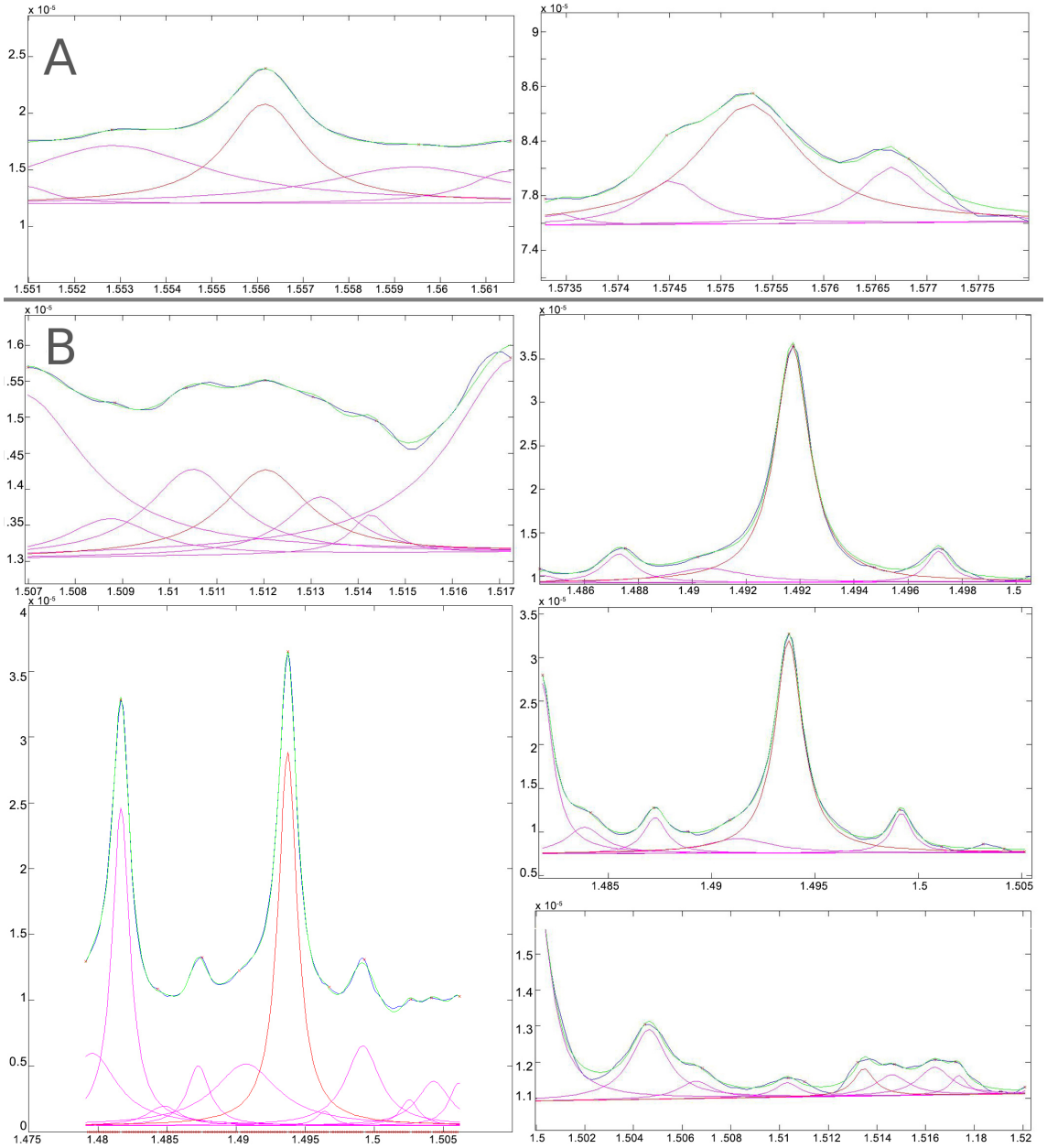


Figure S2: A collection of fitted urinary NMR sections. In each plot, the blue line is the real, unaligned spectra, the green one is the sum of fitted Lorentzians plus the estimated baseline, the magenta ones are the fitted Lorentzians and the red one is the Lorentzian of the peak of interest. A fitting-window size of seven (A) and eleven (B) are displayed. Several plots highlight the maxima detection method's excellence at detecting convolved multiplets and shoulder peaks.

Bias-variance decomposition

In this demonstration (see Figure S3) a single one dimensional regression problem is simulated. The ground truth is shown in blue $f(x)$. Sampling from this function gives us a noisy training set, shown as blue dots $LS \approx y = f(x) + noise$. Fitting a decision tree or an ensemble of bagged decision trees to this sampled dataset provides us with a learned representation, i.e. prediction of the truth, shown in red $\hat{y}(x)$. Finally, the average prediction of the learner across many instances of samples LS gives us $\mathbb{E}_{LS}\hat{y}(x)$, shown in cyan. The bottom figures demonstrate the point-wise decomposition of the learner's overall error into its three constituents as described in Section 3.1.1.

The upper left figure illustrates the predictions (dark red) of a single decision tree trained over a random dataset LS. It also shows the predictions of other single decision trees trained over other (and different) randomly drawn instances LS of the problem (light red). The *variance* of the learner can be though of as the width of the beam of predictions (in light red) of the individual estimators. The **bias** term corresponds to the difference between the average prediction of the estimator (cyan) and the ground truth (dark blue). This example demonstrates nicely that decision trees in general have low bias and high variance.

In the upper right figure we see the same problem approached with a bagged decision tree learner. The bias term is slightly larger than in the previous case, as the prediction of the learner is an average of $B = 10$ bagged decision trees, and therefore it cannot approximate the ground truth as closely as a single decision tree. In terms of variance however, the beam of predictions is narrower, which suggests that the variance is lower, which is also confirmed by the lower right figure. Therefore, the bias-variance decomposition is no longer the same, and the trade-off is better for bagging. More precisely, averaging several decision trees which are each fitted on bootstrap samples of the dataset, slightly increases the bias term, but allows for a larger reduction of the variance, which ultimately results in a lower overall mean squared error.

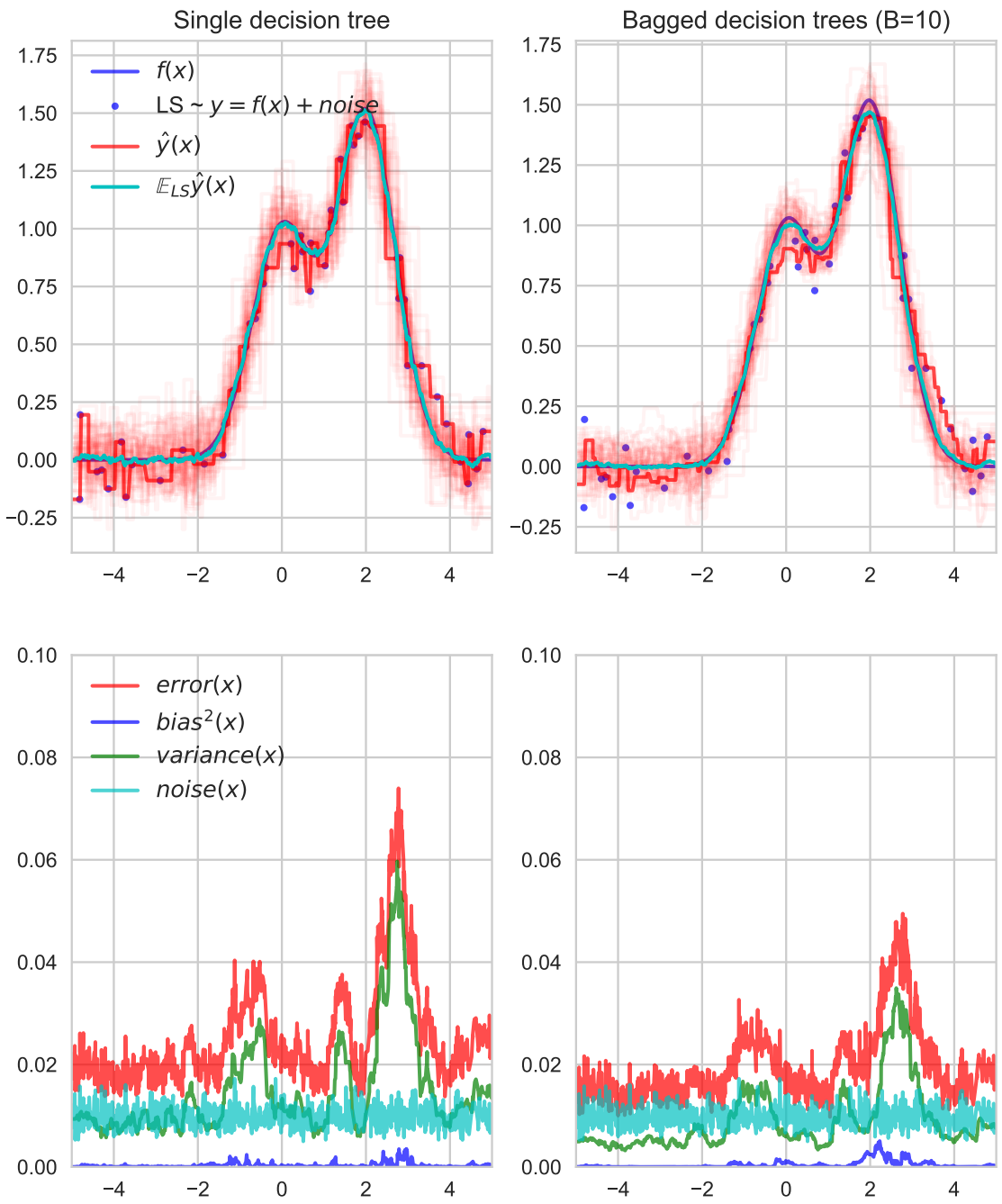


Figure S3: Figure generated from the official `scikit-learn` documentation[192] with slight modifications.

Algorithm S1 describes the full pseudo-code of the Boruta FS method[104].

Algorithm S1 Boruta, part 1.

```
function BORUTA(X^{n×p}, y^{n×1}, t_max)
    t ← 1
    D ∈ ℝ^{p×1} ← 0                    ▷ create list for storing decisions on features
    H ∈ ℝ^{p×1} ← 0                    ▷ create list for storing hits for each feature
    while t < t_max or Σ_{i=1}^{p} [D > 0] do          ▷ main loop of Boruta
        active ← indices where D ≥ 0                   ▷ still active features
        act_p ← |active|                               ▷ number of active features
        Z^{n×2act_p} ← ADDSHADOWS(X^{n×active}, D)     ▷ duplicate X
        Ir, Is ← GETIMP(Z, y)                          ▷ VI of real and shadow features
        H ← ASSIGNHITS(H, Ir, Is)
        D ← UPDATEDECISION(D, H, t)
        t ← t + 1
    end while
    C ← indices where D = 1
    return C
end function

function ADDSHADOWS(X^{n×active}, D)
    X2^{n×active} ← X^{n×active}                   ▷ copy matrix of active features
    shuffle all columns of X2^{n×active}           ▷ shuffle each shadow feature
    Z = [X^{n×active} | X2^{n×active}]             ▷ concatenate the two matrices
    return Z
end function

function GETIMP(Z, y)
    imp ← RF(Z, y)                                 ▷ train Random Forest classifier
    Ir ← VI of the real features                   ▷ from imp
    Is ← VI of the shuffled shadow features        ▷ from imp
    return Ir, Is
end function

function ASSIGNHITS(H, Ir, Is)
    sha_max = max(imp_shadow)                      ▷ find the shadow feature with highest VI
    h = indices where H > sha_max                  ▷ find real features better than sha_max
    H_h ← H_h + 1                                  ▷ increment the list of hits for these
    return H
end function
```

Continued on next page.

Algorithm S2 Boruta, part 2.

```
function updateDecision(D, H, t)
    for h ← 0 to |H| do
        P ∈ ℝ^{act_p×1} ← 0
        p ← binom(H_h, i, 0.5)        ▷ probability of randomly selecting the h^th feature H_h times out of t experiments
        P_h ← p
    end for
    Bonferroni(P)                     ▷ Correct p-values for multiple testing
    r = indices of rejected features
    c = indices of selected features
    D_r ← −1                          ▷ these features will get removed in the next round
    D_c ← 1
    return D
end function
```

The following table summarises how many datasets did run successfully for each of the FS methods. Univariate FDR and L1 SVC did not find any features relevant in 33% and 11% of the cases respectively.

Ratio	UnivarPerc	UnivarFDR	RFE CV	L1 SVC	StabSel	Boruta	JMI
0.005	20	1	20	11	20	20	20
0.01	20	0	20	15	20	20	20
0.015	20	0	20	14	20	20	20
0.02	20	1	20	17	20	20	20
0.025	20	3	20	16	20	20	20
0.03	20	1	20	18	20	20	20
0.05	20	3	20	15	20	20	20
0.06	20	3	20	17	20	20	20
0.1	40	21	40	38	40	40	40
0.2	20	16	20	19	20	20	20
0.3	20	20	20	20	20	20	20
0.5	20	20	20	20	20	20	20
0.6	20	20	20	20	20	20	20
1.0	60	59	60	60	60	60	60
2.0	40	40	40	40	40	40	40
3.0	20	20	20	20	20	20	20
4.0	20	20	20	20	20	20	20
5.0	20	20	20	20	20	20	20
10.0	20	20	20	20	20	20	20
20.0	20	20	20	20	20	20	20
Sum	480	308	480	440	480	480	480

Table S1: Number of benchmarking experiments per ratio and algorithm. We only count a dataset for a given algorithm, if the method selected at least one feature from it.

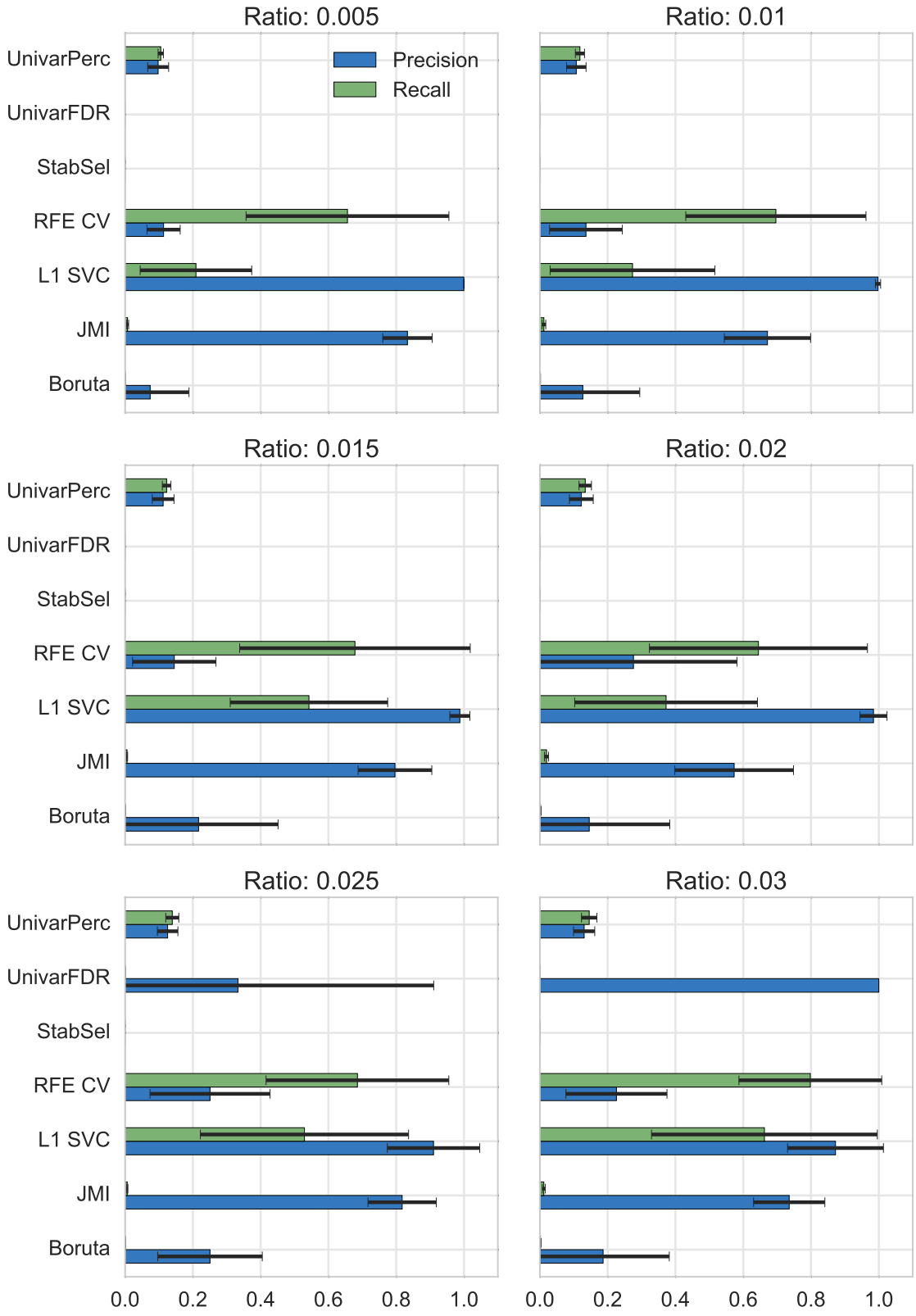


Figure S4: Benchmarking results of seven FS methods, part 1. The title of each sub-plot shows the ratio of samples to features: $R = n/p$. Horizontal black lines represent standard deviations stemming from pooling the repeated runs with varying random seeds and different datasets having the same R value.

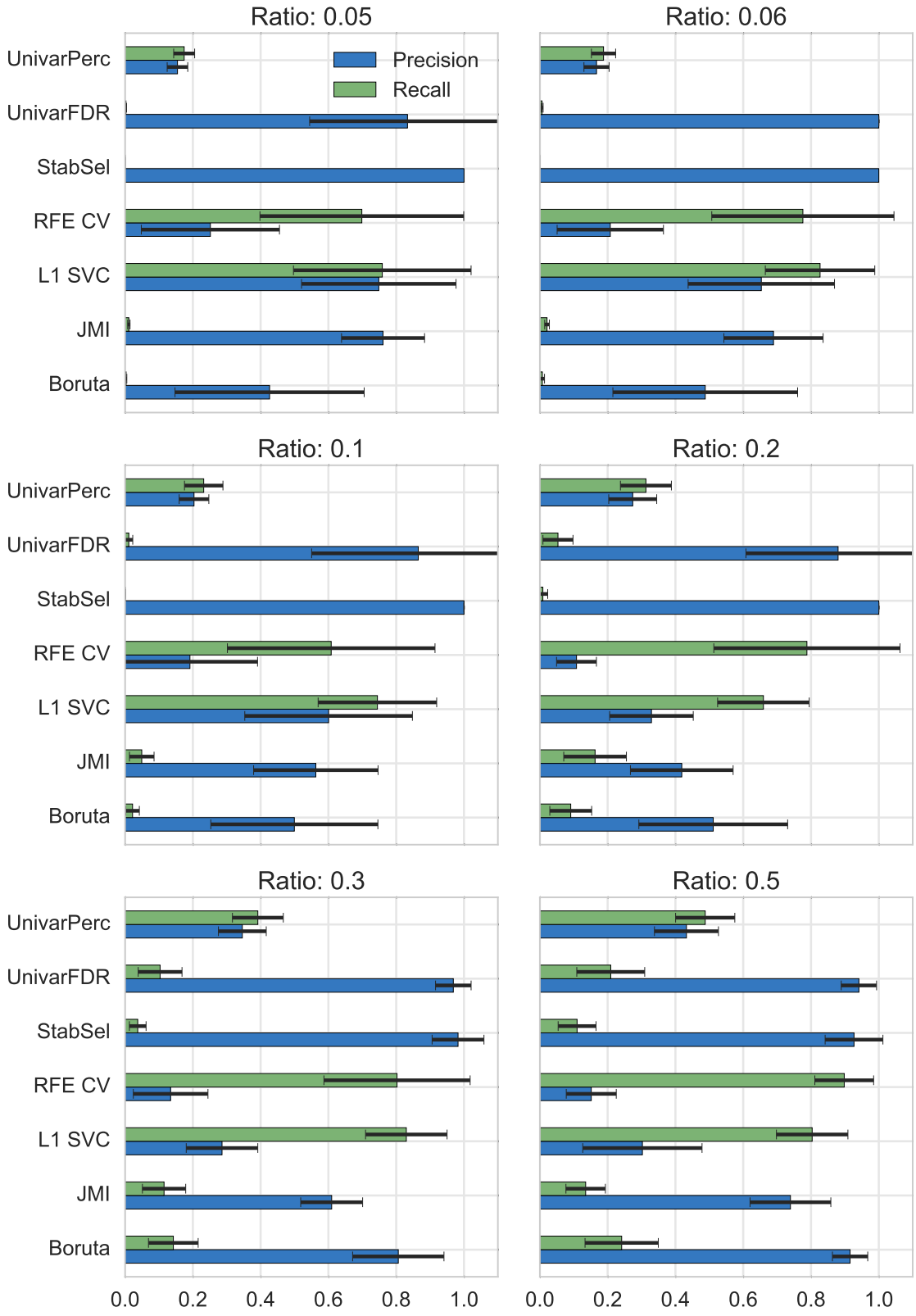


Figure S5: Benchmarking results of seven FS methods, part 2. The title of each sub-plot shows the ratio of samples to features: $R = n/p$. Horizontal black lines represent standard deviations stemming from pooling the repeated runs with varying random seeds and different datasets having the same R value.

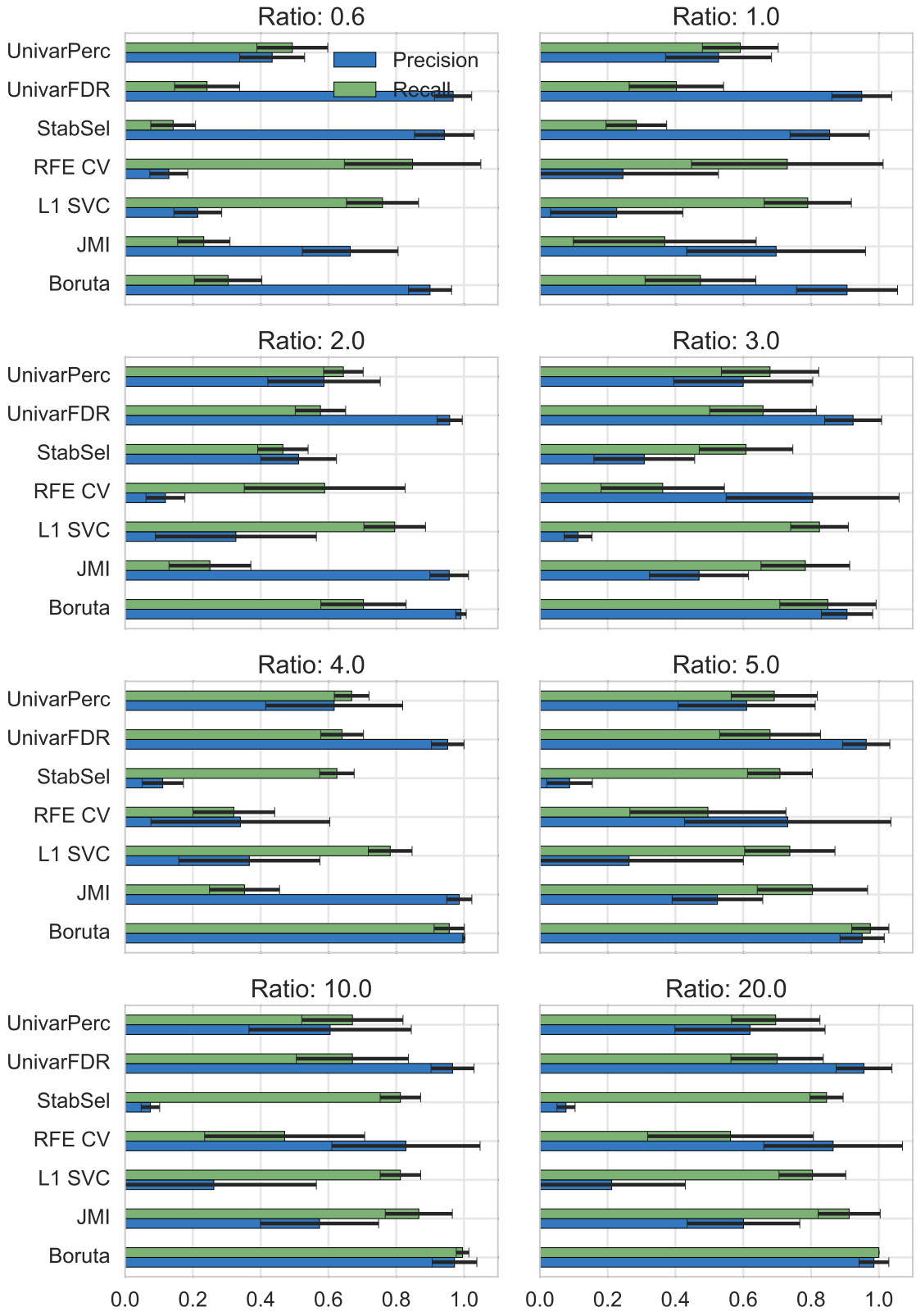


Figure S6: Benchmarking results of seven FS methods, part 3. The title of each sub-plot shows the ratio of samples to features: $R = n/p$. Horizontal black lines represent standard deviations stemming from pooling the repeated runs with varying random seeds and different datasets having the same R value.

	StARS	UnivarFDR	L1 SVC	Boruta	JMI
Rec	0.05	0.0571 ± 0.05	0.1458 ± 0.15	0.0480 ± 0.05	0.0177 ± 0.02
Prec		0.4409 ± 0.24	0.3992 ± 0.21	0.4221 ± 0.26	0.3976 ± 0.25
R + P		0.4979 ± 0.29	0.5449 ± 0.35	0.4701 ± 0.31	0.4153 ± 0.27
Rec	0.1	0.0622 ± 0.07	0.0467 ± 0.11	0.0562 ± 0.06	0.0212 ± 0.03
Prec		0.3837 ± 0.21	0.2901 ± 0.19	0.3656 ± 0.25	0.2772 ± 0.22
R + P		0.4458 ± 0.28	0.3368 ± 0.29	0.4218 ± 0.31	0.2984 ± 0.25

Table S2: Performance of CorrMapper on simulated datasets. Rec: recall, Prec: precision, R + P: recall + precision. These performance metrics measure how many of the true network edges CorrMapper could reconstruct. Values to the left of the ± are means, while values on the right represent one standard deviation.

Method	# success	Recall	Precision	Recall + Precision
UnivarFDR	12	0.1455 ± 0.11	0.5845 ± 0.14	0.7300 ± 0.25
L1 SVC	12	0.3740 ± 0.20	0.5808 ± 0.14	0.9548 ± 0.34
Boruta	12	0.1525 ± 0.11	0.6113 ± 0.19	0.7638 ± 0.31
JMI	12	0.0852 ± 0.08	0.6462 ± 0.32	0.7315 ± 0.39
GraphLasso	0	0.0000 ± nan	nan ± nan	nan ± nan
Marginal 0.05	12	0.3327 ± 0.10	0.0313 ± 0.01	0.3641 ± 0.11
Marginal 0.1	12	0.0672 ± 0.04	0.0494 ± 0.03	0.1166 ± 0.07
Marginal 0.2	3	0.0000 ± 0.00	0.0000 ± 0.00	0.0000 ± 0.00
Marginal 0.3	0	0.0000 ± 0.00	nan ± nan	nan ± nan
Marginal 0.5	0	0.0000 ± 0.00	nan ± nan	nan ± nan
Marginal 0.7	0	0.0000 ± 0.00	nan ± nan	nan ± nan
Marginal 0.8	0	0.0000 ± 0.00	nan ± nan	nan ± nan
mixOmics 0.05	12	0.4432 ± 0.11	0.1086 ± 0.03	0.5518 ± 0.14
mixOmics 0.1	12	0.2558 ± 0.11	0.2153 ± 0.07	0.4711 ± 0.19
mixOmics 0.2	12	0.1339 ± 0.08	0.3840 ± 0.15	0.5180 ± 0.24
mixOmics 0.3	12	0.1065 ± 0.06	0.5371 ± 0.21	0.6437 ± 0.28
mixOmics 0.5	11	0.0632 ± 0.05	0.9061 ± 0.17	0.9693 ± 0.22
mixOmics 0.7	0	0.0000 ± 0.00	nan ± nan	nan ± nan
mixOmics 0.8	0	0.0000 ± 0.00	nan ± nan	nan ± nan

Table S3: Comparing CorrMapper against other network estimators. Unlike Table 5.4, this table includes all cut off values for marginal correlation and mixOmics. Numerous thresholds resulted in zero edges identified (see # success column), and therefore a zero recall value and a non-interpretable precision, due to division by zero. Values to the left of the ± are means, while values on the right represent one standard deviation.

OTU	Base μ	Log_2 FC	Adj. p-val	Family	Genus
4321386	56.5732	-3.0799	0.0000	Streptococcaceae	Streptococcus
4366572	11.1821	-2.8802	0.0000	Streptococcaceae	Streptococcus
184376	34.8257	-3.2843	0.0000	Streptococcaceae	Streptococcus
899176	7.3268	-2.9705	0.0000	Streptococcaceae	Streptococcus
4353951	12.1208	-2.8175	0.0000	Enterobacteriaceae	Klebsiella
4430570	27.0982	-2.6345	0.0000	Enterobacteriaceae	
4308962	3.6592	-2.2360	0.0000	Streptococcaceae	Streptococcus
290612	17.4495	-2.8185	0.0000	Enterobacteriaceae	
4321	5.8048	-2.3544	0.0000	Enterobacteriaceae	
4449690	20.0906	-2.3589	0.0000	Enterobacteriaceae	
4301368	11.0270	-2.1611	0.0002	Enterobacteriaceae	Klebsiella
3474874	4.1883	-2.0261	0.0002	Enterobacteriaceae	
2999290	3.8586	-2.0419	0.0002	Enterobacteriaceae	
365704	6.1607	1.8978	0.0004	Erysipelotrichaceae	
4326711	2.7533	-1.8295	0.0004	Veillonellaceae	Veillonella
1056735	26.4205	-1.9074	0.0004	Streptococcaceae	Streptococcus
291164	6.9656	-2.0537	0.0006	Enterobacteriaceae	
174874	3.7439	1.6544	0.0012	Erysipelotrichaceae	
367867	5.4790	1.6945	0.0013		
321132	4.1840	1.7533	0.0013	Lachnospiraceae	

Table S4: Top twenty differentially expressed OTUs in RYGB patients after operation. Results were produced using DESeq2[171] through the QIIME pipeline[172]. Base μ: mean count OTU before surgery.

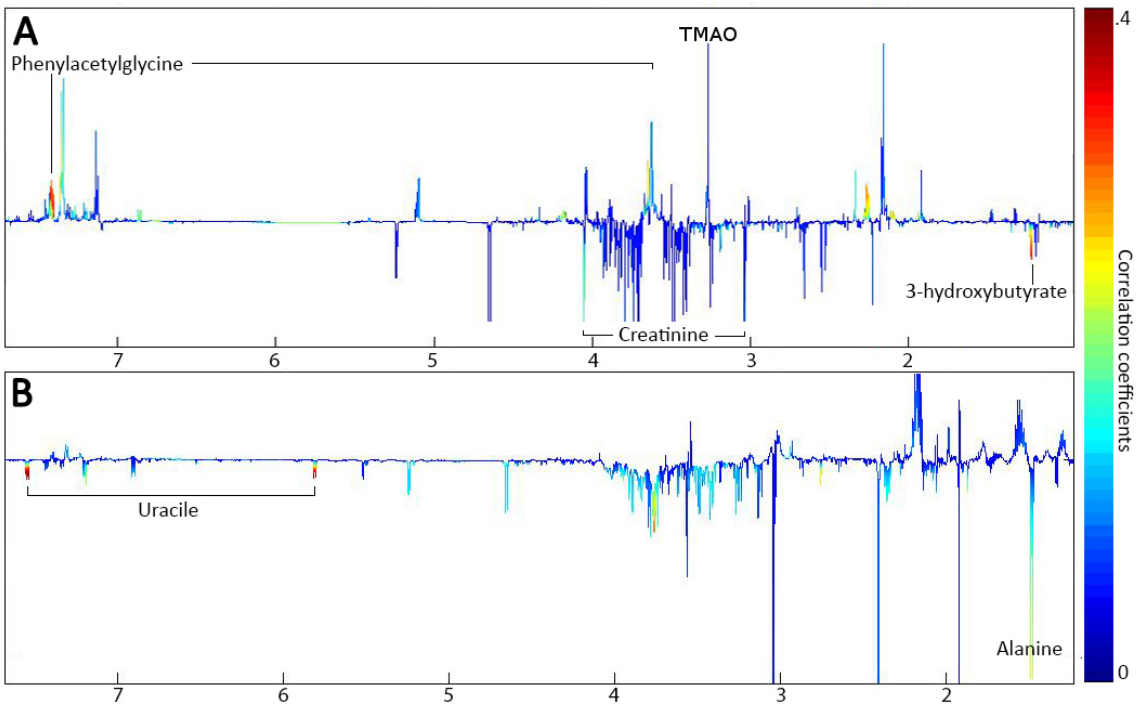


Figure S7: O-PLS coefficient plots of urinary (A) and faecal water (B) NMR spectral data. Datasets were regressed against time-points (1, 2 and 3), thus upward pointing peaks indicate metabolites whose concentrations increased after surgery, while downward pointing peaks mean the opposite.

References

1. Lander, E. S. *et al.* Initial sequencing and analysis of the human genome. *Nature.* (2001).
2. Venter, J. C. *et al.* The sequence of the human genome. *Science.* (2001).
3. Hayden, E. C. Technology: The $1,000 genome. *Nature.* (2014).
4. Check Hayden, E. Is the $1,000 genome for real? *Nature.* (2014).
5. National Human Genome Research Institute. *The Cost of Sequencing a Human Genome* URL: `https://www.genome.gov/sequencingcosts/`.
6. Tzoulaki, I., Ebbels, T., Valdes, A., Elliott, P. & Ioannidis, J. Design and analysis of metabolomics studies in epidemiologic research: A primer on-omic technologies. *American Journal of Epidemiology* (2014).
7. Cook, C. E. *et al.* The European Bioinformatics Institute in 2016: Data growth and integration. *Nucleic acids research.* (2016).
8. EMBL-EBI. *Statistics About the European Nucleotide Archive* URL: `http://www.ebi.ac.uk/ena/about/statistics`.
9. Facebook. *Scaling the Facebook data warehouse to 300 PB* URL: `https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/`.
10. CERN. *Data Centre passes 100 petabytes* URL: `http://home.web.cern.ch/about/updates/2013/02/cern-data-centre-passes-100-petabytes`.
11. Jarboe, G. *VidCon: Trends, Strategic Insights & Tactical Advice* URL: `http://tubularinsights.com/vidcon-2015-strategic-insights-tactical-advice/`.
12. Marx, V. Biology: The big challenges of big data. *Nature.* (2013).
13. Janda, J. M. & Abbott, S. L. 16S rRNA gene sequencing for bacterial identification in the diagnostic laboratory: pluses, perils, and pitfalls. *Journal of clinical microbiology.* (2007).
14. Begley, C. G. & Ioannidis, J. P. A. Reproducibility in science: improving the standard for basic and preclinical research. *Circulation research.* (2015).
15. Ioannidis, J. P. A. Why most published research findings are false. *PLoS medicine.* (2005).
16. Brenner, S. An interview with Sydney Brenner. *Nature reviews. Molecular cell biology.* (2008).
17. Goodacre, R. Metabolomics – the way forward. *Metabolomics.* (2005).
18. Holmes, E., Wilson, I. D. & Nicholson, J. K. Metabolic Phenotyping in Health and Disease. *Cell.* (2008).

19. Beckonert, O. *et al.* Metabolic profiling, metabolomic and metabonomic procedures for NMR spectroscopy of urine, plasma, serum and tissue extracts. *Nature protocols.* (2007).
20. Nagana Gowda, G. A. & Raftery, D. Biomarker Discovery and Translation in Metabolomics. *Current Metabolomics.* (2013).
21. Marchesi, J. R. & Ravel, J. The vocabulary of microbiome research: a proposal. *Microbiome.* (2015).
22. Clarridge, J. E. Impact of 16S rRNA Gene Sequence Analysis for Identification of Bacteria on Clinical Microbiology and Infectious Diseases Impact of 16S rRNA Gene Sequence Analysis for Identification of Bacteria on Clinical Microbiology and Infectious Diseases. *Medicine* (2004).
23. Trygg, J. & Wold, S. Orthogonal projections to latent structures (O-PLS). *Journal of Chemometrics.* (2002).
24. Anderson, M. J. A new method for non-parametric multivariate analysis of variance. *Austral Ecology.* (2001).
25. Kåhrström, C. T., Pariente, N. & Weiss, U. Intestinal microbiota in health and disease. *Nature.* (2016).
26. Helgason, A. *et al.* The Y-chromosome point mutation rate in humans. *Nature Genetics.* (2015).
27. Giskeødegård, G. F., Davies, S. K., Revell, V. L., Keun, H. & Skene, D. J. Diurnal rhythms in the human urine metabolome during sleep and total sleep deprivation. *Scientific Reports.* (2015).
28. Caporaso, J. G. *et al.* Moving pictures of the human microbiome. *Genome biology.* (2011).
29. Maitre, L. *et al.* Assessment of metabolic phenotypic variability in children's urine using 1H NMR spectroscopy. *Scientific Reports.* (2017).
30. Lloyd-Price, J., Abu-Ali, G. & Huttenhower, C. The healthy human microbiome. *Genome medicine.* (2016).
31. Holmes, E., Li, J. V., Athanasiou, T., Ashrafian, H. & Nicholson, J. K. Understanding the role of gut microbiome-host metabolic signal disruption in health and disease. *Trends in Microbiology.* (2011).
32. Holmes, E., Li, J. V., Marchesi, J. R. & Nicholson, J. K. Gut microbiota composition and activity in relation to host metabolic phenotype and disease risk. *Cell Metabolism.* (2012).
33. Marcobal, a. *et al.* A metabolomic view of how the human gut microbiota impacts the host metabolome using humanized and gnotobiotic mice. *The ISME journal.* (2013).
34. Feng, Q. *et al.* Integrated metabolomics and metagenomics analysis of plasma and urine identified microbial metabolites associated with coronary heart disease. *Scientific Reports.* (2016).
35. Weir, T. L. *et al.* Stool Microbiome and Metabolome Differences between Colorectal Cancer Patients and Healthy Adults. *PLoS ONE.* (2013).
36. Chin, K. *et al.* Genomic and transcriptional aberrations linked to breast cancer pathophysiologies. *Cancer cell* (2006).

37. Bersanelli, M. *et al.* Methods for the integration of multi-omics data: mathematical aspects. *BMC Bioinformatics.* (2016).

38. Rohart, F., Gautier, B., Singh, A. & Le Cao, K.-A. mixOmics: an R package for 'omics feature selection and multiple data integration. *bioRxiv.* (2017).

39. Chari, R., Coe, B. P., Vucic, E. A., Lockwood, W. W. & Lam, W. L. An integrative multi-dimensional genetic and epigenetic strategy to identify aberrant genes and pathways in cancer. *BMC Systems Biology.* (2010).

40. Meng, C., Kuster, B., Culhane, A. C. & Gholami, A. A multivariate approach to the integration of multi-omics datasets. *BMC Bioinformatics.* (2014).

41. Lê Cao, K.-A., González, I. & Déjean, S. integrOmics: an R package to unravel relationships between two omics datasets. *Bioinformatics.* (2009).

42. Li, W., Zhang, S., Liu, C.-C. & Zhou, X. J. Identifying multi-layer gene regulatory modules from multi-dimensional genomic data. *Bioinformatics.* (2012).

43. Gonzalez, I., Déjean, S., Martin, P. & Baccini, A. CCA: An R Package to Extend Canonical Correlation Analysis. *Journal of Statistical Software.* (2008).

44. Barker, M. & Rayens, W. Partial least squares for discrimination. *Journal of Chemometrics.* (2003).

45. Lê Cao, K.-A., Martin, P. G. P., Robert-Granié, C. & Besse, P. Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC bioinformatics.* (2009).

46. Tuncbag, N., McCallum, S., Huang, S.-s. C. & Fraenkel, E. SteinerNet: a web server for integrating 'omic' data to discover hidden components of response pathways. *Nucleic Acids Research.* (2012).

47. Tranchevent, L.-C. *et al.* Candidate gene prioritization with Endeavour. *Nucleic Acids Research.* (2016).

48. Wang, B. *et al.* Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods.* (2014).

49. McNeish, D. On Using Bayesian Methods to Address Small Sample Problems. *Structural Equation Modeling: A Multidisciplinary Journal.* (2016).

50. Hastie, T., Tibshirani, R. & Wainwright, M. *Statistical learning with Sparsity: the lasso and generalizations* (2016).

51. Hunter, S. *et al.* InterPro: the integrative protein signature database. *Nucleic acids research* (2009).

52. Brenner, S. Life sentences: Detective Rummage investigates. *Genome biology* (2002).

53. Chen, B.-J. *et al.* Harnessing gene expression to identify the genetic basis of drug resistance. *Molecular Systems Biology.* (2009).

54. Aure, M. R. *et al.* Identifying In-Trans Process Associated Genes in Breast Cancer by Integrated Analysis of Copy Number and Expression Data. *PLoS ONE.* (2013).

55. Louhimo, R. & Hautaniemi, S. CNAmet: an R package for integrating copy number, methylation and expression data. *Bioinformatics.* (2011).

56. Mues, K. *et al.* Use of the Medicare database in epidemiologic and health services research: a valuable source of real-world evidence on the older and disabled populations in the US. *Clinical Epidemiology.* (2017).

57. Herbert, A., Wijlaars, L., Zylbersztejn, A., Cromwell, D. & Hardelid, P. Data Resource Profile: Hospital Episode Statistics Admitted Patient Care (HES APC). *International Journal of Epidemiology.* (2017).

58. Hodson, R. Precision medicine. *Nature.* (2016).

59. Iyer, P. M., Karthikeyan, S., Sanjay Kumar, P. & Krishnan Namboori, P. K. Comprehensive strategy for the design of precision drugs and identification of genetic signature behind proneness of the disease—a pharmacogenomic approach. *Functional & Integrative Genomics.* (2017).

60. Statista. *Global connected wearable devices* URL: `https://www.statista.com/statistics/487291/global-connected-wearable-devices/`.

61. Consortium, T. H. M. P. Structure, function and diversity of the healthy human microbiome. *Nature.* (2012).

62. Subramanian, J. & Simon, R. Overfitting in prediction models – Is it a problem only in high dimensions? *Contemporary Clinical Trials.* (2013).

63. Cliff Saran. *Apollo 11: The computers that put man on the moon* URL: `http://www.computerweekly.com/feature/Apollo-11-The-computers-that-put-man-on-the-moon`.

64. Bostock, M. *D3.js* URL: `http://d3js.org/`.

65. Alexis Jacomy. *Sigma.js* URL: `http://sigmajs.org/`.

66. Bostock, M. *d3.js examples* URL: `https://github.com/d3/d3/wiki/Gallery`.

67. Ricardo Cabello. *three.js - Javascript 3D library* URL: `https://threejs.org/`.

68. Uber. *Advanced Data Visualisation WebGL - deck.gl* URL: `https://uber.github.io/deck.gl/%7B%5C#%7D/examples/overview`.

69. James, G., Witten, D., Hastie, T. & Tibshirani, R. *Introduction to Statistical Learning* 4th. (2014).

70. Ribeiro, M. T., Singh, S. & Guestrin, C. Why Should I Trust You?: Explaining the Predictions of Any Classifier. *arXiv.* (2016).

71. Zeiler, M. D. & Fergus, R. Visualizing and Understanding Convolutional Networks. *arXiv.* (2013).

72. Gentleman, R. C. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome biology.* (2004).

73. Pedregosa, F. & Varoquaux, G. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research.* (2011).

74. Brethauer, S., Chand, B. & Schauer, P. R. Risks and benefits of bariatric surgery: current evidence. *Cleveland Clinic journal of medicine.* (2006).

75. Guido van Rossum. *Python* URL: `https://www.python.org/`.

76. Hipp, D. R. *SQLite* URL: `https://www.sqlite.org/about.html`.

77. Bouatra, S. *et al.* The Human Urine Metabolome. *PLoS ONE.* (2013).

78. Cloarec, O. *et al.* Statistical total correlation spectroscopy: an exploratory approach for latent biomarker identification from metabolic 1H NMR data sets. *Analytical chemistry.* (2005).

79. Wold, S., Sjöström, M. & Eriksson, L. PLS-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems* (2001).

80. Cavanagh, J. *Protein NMR spectroscopy : principles and practice* (1996).

81. Vu, T. & Laukens, K. Getting Your Peaks in Line: A Review of Alignment Methods for NMR Spectral Data. *Metabolites.* (2013).

82. Veselkov, K. a. *et al.* Recursive segment-wise peak alignment of biological (1)H NMR spectra for improved metabolic biomarker recovery. *Analytical chemistry.* (2009).

83. Savitzky, A. & Golay, M. J. E. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry.* (1964).

84. Koh, H. W. *Feature Extraction in NMR Data Analysis* PhD thesis (2010).

85. De Brouwer, H. Evaluation of algorithms for automated phase correction of NMR spectra. *Journal of Magnetic Resonance.* (2009).

86. Xi, Y. & Rocke, D. M. Baseline correction for NMR spectroscopic metabolomics data analysis. *BMC bioinformatics.* (2008).

87. Alonso, A. *et al.* Focus: A robust workflow for one-dimensional NMR spectral analysis. *Analytical Chemistry* (2014).

88. DeSantis, T. Z. *et al.* Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Applied and environmental microbiology* (2006).

89. Větrovský, T. & Baldrian, P. The variability of the 16S rRNA gene in bacterial genomes and its consequences for bacterial community analyses. *PloS one.* (2013).

90. Větrovský, T. & Baldrian, P. The Variability of the 16S rRNA Gene in Bacterial Genomes and Its Consequences for Bacterial Community Analyses. *PLoS ONE.* (2013).

91. Kembel, S. W., Wu, M., Eisen, J. A. & Green, J. L. Incorporating 16S Gene Copy Number Information Improves Estimates of Microbial Diversity and Abundance. *PLoS Computational Biology.* (2012).

92. Langille, M. G. I. *et al.* Predictive functional profiling of microbial communities using 16S rRNA marker gene sequences. *Nature biotechnology.* (2013).

93. Auer, P. L. *et al.* Statistical Design and Analysis of RNA Sequencing Data. *Genetics.* (2010).

94. Filzmoser, P. *et al.* Principal component analysis for compositional data with outliers. (2007).

95. Aitchison, J. The Statistical Analysis of Compositional Data. *Journal of the Royal Statistical Society.* (1982).

96. Barrett, T. *Gene Expression Omnibus* (2013).

97. Dougherty, E. R., Hua, J. & Sima, C. Performance of feature selection methods. *Current genomics.* (2009).

98. Clarke, R. *et al.* The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature reviews. Cancer.* (2008).

99. Saeys, Y., Inza, I. & Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics* (2007).

100. Vapnik, V. *Statistical learning theory* (1998).

101. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B* (1996).

102. Meinshausen, N. & Peter, B. Stability selection. *Journal of the Royal Statistical Society. Series B* (2010).

103. R Core Team. *R: A Language and Environment for Statistical Computing* URL: `http://www.r-project.org`.

104. Kursa, M. B. & Rudnicki, W. R. Feature Selection with the Boruta Package. *Journal Of Statistical Software.* (2010).

105. Breiman, L. Random Forests. *Machine Learning.* (2001).

106. Chen, T. & Guestrin, C. XGBoost: A Scalable Tree Boosting System. *arXiv.* (2016).

107. Hastie, T., Tibshirani, R. & Friedman, J. H. *The elements of statistical learning: data mining, inference, and prediction* (2001).

108. Louppe, G. *Understanding Random Forests: From Theory to Practice* URL: `http://www.slideshare.net/glouppe/understanding-random-forests-from-theory-to-practice?`.

109. Bishop, C. M. *Pattern recognition and machine learning* (2006).

110. Homola, D. *BorutaPy - a Python implementation of the Boruta R package* URL: `https://github.com/scikit-learn-contrib/boruta_py`.

111. Kursa, M. B. *Boruta website* URL: `https://m2.icm.edu.pl/boruta/`.

112. Oshiro, T., Perez, P. & Baranauskas, J. *How Many Trees in a Random Forest?* (2012).

113. Noble, W. S. How does multiple testing correction work? *Nature Biotechnology.* (2009).

114. Glickman, M. E., Rao, S. R. & Schultz, M. R. False discovery rate control is a recommended alternative to Bonferroni-type adjustments in health studies. *Journal of Clinical Epidemiology* (2014).

115. Benjamini, Y. & Hochberg, Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society* (1995).

116. Shannon, C. E. A Mathematical Theory of Communication. *Bell System Technical Journal.* (1948).

117. Lewis, D. D. *Feature selection and feature extraction for text categorization* in *Proceedings of the workshop on Speech and Natural Language* (1992).

118. Peng, H. C., Long, F. H. & Ding, C. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *Ieee Transactions on Pattern Analysis and Machine Intelligence* (2005).

119. Howard Hua Yang, J. M. *Feature Selection Based on Joint Mutual Information* in *Proceedings of International ICSC Symposium on Advances in Intelligent Data Analysis* (1999).

120. Meyer, P. E., Schretter, C. & Bontempi, G. Information-Theoretic Feature Selection in Microarray Data Using Variable Complementarity. *IEEE Journal of Selected Topics in Signal Processing.* (2008).

121. Vergara, J. R. & Estevez, P. A. A review of feature selection methods based on mutual information. *Neural Computing and Applications.* (2014).

122. Brown, G., Pocock, A., Zhao, M.-J. & Lujan, M. Conditional Likelihood Maximisation: A Unifying Framework for Mutual Information Feature Selection. *Journal of Machine Learning Research.* (2012).

123. Battiti, R. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks* (1994).

124. Homola, D. *Parallelized Mutual Information based Feature Selection module* URL: `https://github.com/danielhomola/mifs`.

125. Ross, B. C., Leonenko, N., Carpena, P., Román-Roldán, R. & Oliver, J. Mutual Information between Discrete and Continuous Data Sets. *PLoS ONE.* (2014).

126. Haury, A. C., Gestraud, P. & Vert, J. P. The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. *PLoS ONE* (2011).

127. Guyon, I., Weston, J., Barnhill, S. & Vapnik, V. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning* (2002).

128. Guyon, I., Gunn, S., Ben-Hur, A. & Dror, G. Result Analysis of the NIPS 2003 Feature Selection Challenge. (2004).

129. Yu, D., Kim, M., Xiao, G. & Hwang, T. H. Review of biological network data and its applications. *Genomics & informatics.* (2013).

130. Székely, G. J. & Rizzo, M. L. Brownian distance covariance. *The Annals of Applied Statistics.* (2009).

131. Song, L., Langfelder, P. & Horvath, S. Comparison of co-expression measures: mutual information, correlation, and model based indices. *BMC Bioinformatics.* (2012).

132. Lee, H. K., Hsu, A. K., Sajdak, J., Qin, J. & Pavlidis, P. Coexpression Analysis of Human Genes Across Many Microarray Data Sets. *Genome Research.* (2004).

133. Bassel, G. W. *et al.* Genome-wide network model capturing seed germination reveals coordinated regulation of plant cellular phase transitions. *Proceedings of the National Academy of Sciences.* (2011).

134. Barabási, A.-L. & Oltvai, Z. N. Network biology: understanding the cell's functional organization. *Nature reviews. Genetics* (2004).

135. Ledoit, O. & Wolf, M. Honey, I Shrunk the Sample Covariance Matrix. *The Journal of Portfolio Management.* (2003).

136. Friedman, J., Hastie, T. & Tibshirani, R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics (Oxford, England).* (2008).

137. Liu, H., Lafferty, J. & Wasserman, L. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. *Journal of Machine Learning Research.* (2009).

138. Sklar, A. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Université de Paris.* (1959).

139. Liu, H., Roeder, K. & Wasserman, L. Stability Approach to Regularization Selection (StARS) for High Dimensional Graphical Models. *arXiv.* (2010).

140. Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control.* (1974).

141. Yuan, M. & Lin, Y. Model selection and estimation in the Gaussian graphical model. *Biometrika.* (2007).

142. Foygel, R. & Drton, M. Extended Bayesian Information Criteria for Gaussian Graphical Models. *arXiv.* (2010).

143. CHOI, S. C. Tests of equality of dependent correlation coefficients. *Biometrika.* (1977).

144. Noble, W. S. How does multiple testing correction work? *Nature Biotechnology.* (2009).

145. Knijnenburg, T. A., Wessels, L. F. A., Reinders, M. J. T. & Shmulevich, I. Fewer permutations, more accurate P-values. *Bioinformatics.* (2009).

146. Bonferroni, C. E. in *Studi in Onore del Professore Salvatore Ortu Carboni.* (1935).

147. Benjamini, Y. & Yekutieli, D. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics.* (2001).

148. Homola, D. *CorrMapper: an online research tool for the integration and visualisation of complex biomedical and omics datasets* URL: `https://github.com/danielhomola/CorrMapper`.

149. Homola, D. *Science Flask: an open-source template for scientific web-app developmen* URL: `https://github.com/danielhomola/science_flask`.

150. Free Software Foundation. GNU GPLv3 licence. (2007).

151. Ronacher, A. *Welcome to Flask - A Python Microframework* URL: `http://flask.pocoo.org/`.

152. Pivotal Software. *RabbitMQ - Messaging that just works* URL: `https://www.rabbitmq.com/`.

153. Ask Solem. *Celery: Distributed Task Queue* URL: `http://www.celeryproject.org/`.

154. Afgan, E. *et al.* The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Research.* (2016).

155. Yang, Z., Algesheimer, R. & Tessone, C. J. A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Nature.* (2016).

156. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment.* (2008).

157. Hagberg, A. A., Schult, D. A. & Swart, P. J. Exploring Network Structure, Dynamics, and Function using NetworkX. *SciPy Conference.* (2008).

158. Beckett, S. J. Improved community detection in weighted bipartite networks. *Royal Society Open Science.* (2016).

159. Shannon, P. *et al.* Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research.* (2003).

160. Singh, A. *et al.* DIABLO - an integrative, multi-omics, multivariate method for multi-group classification. *bioRxiv.* (2016).

161. Rohart, F., Eslami, A., Matigian, N., Bougeard, S. & Lê Cao, K.-A. MINT: a multivariate integrative method to identify reproducible molecular signatures across independent experiments and platforms. *BMC Bioinformatics.* (2017).

162. González, I., Cao, K.-A. L., Davis, M. J. & Déjean, S. Visualising associations between paired 'omics' data sets. *BioData Mining.* (2012).

163. Lê Cao, K.-A., Boitard, S. & Besse, P. Sparse PLS discriminant analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics.* (2011).

164. Nguyen, D. V. & Rocke, D. M. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics.* (2002).

165. Wold, S., Sjöström, M. & Eriksson, L. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems.* (2001).

166. Kahneman, D. *Thinking, Fast and Slow* (2011).

167. Zhao, Z. *et al.* Controlling False Discoveries During Interactive Data Exploration. (2016).

168. Murphy, W. G. The sex difference in haemoglobin levels in adults — Mechanisms, causes, and consequences. *Blood Reviews.* (2014).

169. Gruvberger, S. *et al.* Estrogen receptor status in breast cancer is associated with remarkably distinct gene expression patterns. *Cancer research.* (2001).

170. Duncan, S. H. *et al.* Human colonic microbiota associated with diet, obesity and weight loss. *International journal of obesity.* (2008).

171. Love, M. I., Huber, W. & Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology.* (2014).

172. Caporaso, J. G. *et al.* QIIME allows analysis of high-throughput community sequencing data. *Nature methods* (2010).

173. Li, J. V. *et al.* Metabolic surgery profoundly influences gut microbial-host metabolic cross-talk. *Gut.* (2011).

174. Claus, S. P. *et al.* Colonization-induced host-gut microbial metabolic interaction. *mBio.* (2011).

175. Bouatra, S. *et al.* The Human Urine Metabolome. *PLoS ONE.* (2013).

176. Neff, K. J. *et al.* The effect of bariatric surgery on renal function and disease: a focus on outcomes and inflammation. *Nephrology Dialysis Transplantation.* (2013).

177. Koeth, R. A. *et al.* Intestinal microbiota metabolism of l-carnitine, a nutrient in red meat, promotes atherosclerosis. *Nature Medicine.* (2013).

178. Kent, W. J. *et al.* The human genome browser at UCSC. *Genome research.* (2002).

179. Benson, D. A. *et al.* GenBank. *Nucleic Acids Research.* (2013).

180. IUBMB. *Enzyme nomenclature: recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes* (1992).

181. Kufe, D. W. MUC1-C oncoprotein as a target in breast cancer: activation of signaling pathways and therapeutic approaches. *Oncogene.* (2013).

182. Krzywinski, M. *et al.* Circos: an information aesthetic for comparative genomics. *Genome research.* (2009).

183. Sinha, A. U., Armstrong, S. A., Armstrong, S., Clark, T. & Das, S. iCanPlot: Visual Exploration of High-Throughput Omics Data Using Interactive Canvas Plotting. *PLoS ONE.* (2012).

184. Deu-Pons, J., Schroeder, M. P. & Lopez-Bigas, N. jHeatmap: an interactive heatmap viewer for the web. *Bioinformatics.* (2014).

185. Noronha, A. *et al.* ReconMap: an interactive visualization of human metabolism. *Bioinformatics.* (2016).

186. Dorel, M., Viara, E., Barillot, E., Zinovyev, A. & Kuperstein, I. NaviCom: a web application to create interactive molecular network portraits using multilevel omics data. *Database.* (2017).

187. Cerami, E. *et al.* The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data. *Cancer Discovery.* (2012).

188. Kuperstein, I. *et al.* Atlas of Cancer Signalling Network: a systems biology resource for integrative analysis of cancer data with Google Maps. *Oncogenesis.* (2015).

189. Kuperstein, I. *et al.* NaviCell: a web-based environment for navigation, curation and maintenance of large molecular interaction maps. *BMC Systems Biology.* (2013).

190. Sung, J. *et al.* Global metabolic interaction network of the human gut microbiota for context-specific community-scale analysis. *Nature Communications.* (2017).

191. Donghao Ren, Bongshin Lee, T. H. Stardust: Accessible and Transparent GPU Support for Information Visualization Rendering. *Computer Graphics Forum* (2017).

192. ScikitLearn. *Single estimator versus bagging: bias-variance decomposition* URL: `http://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html`.