Computational Science for Undergraduate Biologists via QUT.Bio.Excel

*Lawrence Buckingham[1] and James M. Hogan[1]**
[1] Queensland University of Technology, Brisbane, Australia.
l.buckingham@qut.edu.au, j.hogan@qut.edu.au

Abstract
Molecular biology is a scientific discipline which has changed fundamentally in character over the past decade to rely on large scale datasets – public and locally generated - and their computational analysis and annotation. Undergraduate education of biologists must increasingly couple this domain context with a data-driven computational scientific method. Yet modern programming and scripting languages and rich computational environments such as R and MATLAB present significant barriers to those with limited exposure to computer science, and may require substantial tutorial assistance over an extended period if progress is to be made. In this paper we report our experience of undergraduate bioinformatics education using the familiar, ubiquitous spreadsheet environment of Microsoft Excel. We describe a configurable extension called QUT.Bio.Excel, a custom ribbon, supporting a rich set of data sources, external tools and interactive processing within the spreadsheet, and a range of problems to demonstrate its utility and success in addressing the needs of students over their studies.

Keywords: Bioinformatics, Molecular Biology, Computational Thinking, Education.

1 Introduction

Molecular biology is one of a number of scientific disciplines which has changed fundamentally over recent years through an explosion in data availability. From the mid-1990s onwards, well-funded public sequencing centres and projects generated a substantial number of annotated reference sequences, data which formed the foundations of curated collections such as GenBank [1]. Publicly accessible reference collections enabled scientific work beyond that of the individual laboratory, and researchers picked carefully the low hanging genomic and proteomic fruit through standard – often web-hosted – toolsets such as BLAST [2] and Clustal [3]. More recently, successive waves of Next Generation Sequencing (NGS) technologies [4] have taken data generation from the national and trans-national sequencing consortia and placed it firmly in the hands of the individual laboratory. The

* Author to whom all correspondence should be addressed. This work was supported in part by a grant to the second author from Microsoft Research (Redmond), and the QUT Teaching and Learning Grant Schemes.

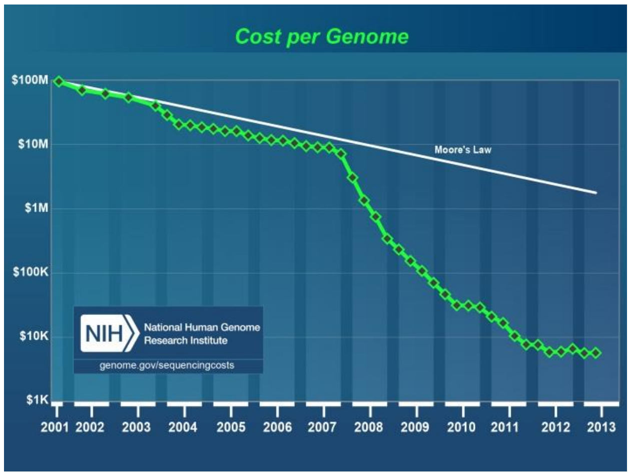


Figure 1: Decline in the cost of sequencing since the original Human Genome Projects, illustrating the relative flat-lining that would have resulted had the trend followed Moore's Law.

Source: National Human Genome Institute Sequencing Costs: http://www.genome.gov/sequencingcosts/

dramatic nature of these changes – in sequencing cost and consequent data availability and computational challenges – is compellingly illustrated in Figure 1.

As has been argued elsewhere [5], such revolutionary changes in technology and data availability present complex challenges to conventional scientific enquiry, with the late Jim Gray arguing persuasively for the emergence of a data-driven science. This *Fourth Paradigm* relies on targeted exploration and discovery from large data sets – data usually collected not to validate a specific hypothesis, but as an umbrella resource supporting a scientific community. The idea is not in itself new, being analogous to the taxonomic reference collections of a natural history museum, but operates more dynamically and at far greater scale. The approach complements earlier methods based on theory, experiment and simulation, retaining many of the characteristics of its predecessors, yet differing markedly in its operation and in the mechanisms for generating hypotheses.

Data driven science is necessarily experimental, but fundamentally a computational paradigm: each stage of the process, from the germination of the idea, through the selection and refinement of an hypothesis and the data to test it, through to the confirmation or falsification itself depends upon the computational facility of the researcher. If scientists of the future are to deal with the increasingly data intensive nature of their disciplines, then their education must incorporate patterns of computational thinking [6] in their approach to scientific questions, and practical training in the use of flexible computational tools and environments. Elements of Exploratory Data Analysis (EDA [7]), data filtration and visualisation must also be introduced.

These challenges are especially pronounced in molecular biology, a discipline in which many students commence with limited exposure to mathematics and computer science, and one in which the computational focus has changed rapidly as data generation has moved to the individual laboratory. Training in computational science for molecular biologists must allow access to existing repositories and the standard tools which accompany them, but provide sufficient flexibility in selection, transformation and display to support novel analysis and annotation of local data sets. To be successful in the undergraduate environment, these facilities must present few barriers to adoption, and ideally they should be hosted within an environment already familiar to the students.

In this work, we introduce a developing environment for bioinformatics education built upon the widely available Microsoft Excel spreadsheet product, using the add-in mechanism to provide a ribbon called QUT.Bio.Excel, hosting algorithms and data structures from the established .NET Bio bioinformatics libraries [8] and elsewhere. The toolset allows lightweight manipulation of large and complex bioinformatic data sets and has been successfully used in undergraduate bioinformatics classes at QUT for three semesters at various levels of maturity.

This paper is organised as follows. In section 2 we provide some additional background to our design choices and a technical overview of the system and the relationship between the computational environment (Excel), the bioinformatics library (.NET Bio) and the external data and computational services made available. Section 3 is concerned with examples of the exercises and their use within undergraduate classes. We conclude in section 4 with a brief discussion of our experiences and plans for further development of the system and its underlying ideas.

2 Background and System Design Principles

The arguments for embedding computational approaches to problem solving across the broader curriculum are well summarized by Wing [6] and have found broad support in the computer science community – through both academic and industrial initiatives. The need for a new breed of biologist – fluent or at least comfortable conversing in the language of mathematics and computer science – has been recognized for more than a decade [9], and scientists themselves were quick to recognize that graduate coursework programmes in bioinformatics, however welcome, could not address the fundamental gap in the skill base of the next generation of scientists. Hack and Kendall [9] noted in 2005 that:

> *"Teaching of the life sciences at undergraduate level has not yet adapted to [the changes], and graduates with good first degrees often lack the skills required to succeed in the new data-driven environment".*

and there is abundant anecdotal evidence that little has changed in many universities. These authors suggested that biology departments should adopt learning outcomes based on those of the physical sciences, with recommendations closely aligned to those of Isbell et al [10]:

> *"At a minimum, the curriculum of existing courses should be revisited to inculcate computationalist thinking—specifically, core competencies in modelling, scales and limits, simulation, abstraction, and automation".*

Yet these authors were also cognizant of the limitations of school education in preparing students outside the specialist mathematics, physics and computer science programmes for these challenges, and this was a key constraint on our approach. Our intention was to develop an environment that could cater readily to the needs of first and second year undergraduate biology students, while offering sufficient power and flexibility that it could support them in more sophisticated work over honours and graduate study, and act as an interface between biology and computer science students in joint projects. Sophisticated parsing and algorithms for search and pattern discovery were thus abstracted away, provided by the .NET Bio project, but data and annotations are then available at scale within the Excel environment.

More specifically, our goals were to:

1. Embed computational thinking directly within a tangible scientific context, supported with practical manipulation of meaningful scientific data;
2. Enable practical facility with external data sets using standard tools while supporting post processing and visualization; and
3. Avoiding the programming roadblock through the use of a familiar, yet flexible and computationally rich environment. Specifically, we took the view that the spreadsheet environment provided by Excel and equivalent systems was accessible to the biologists than the more sophisticated facilities provided by MATLAB or R. Scripting languages such as Python and Perl were not considered for similar reasons.

For computer science students we aimed to introduce them to the algorithmics of computational science in a realistic setting, while allowing them to build new functionality on top of a sophisticated open source library. From this perspective, Excel is a considerable burden, but its ubiquity and

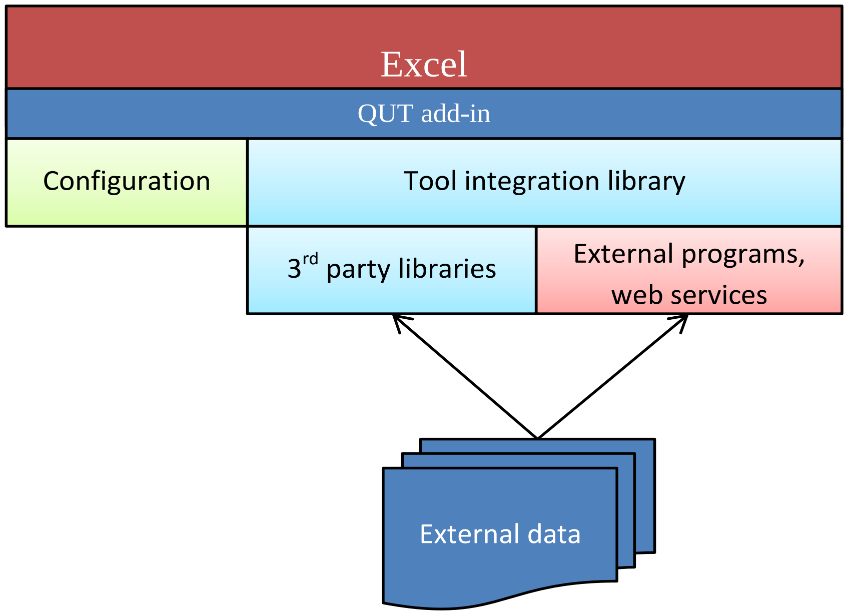


Figure 2: System architecture

familiarity to the student body, and its pervasive use as a cheap electronic lab note book offer compelling advantages.

3 Architecture and Operation

In this section we describe the system in more detail, with a focus on the integration of Microsoft Excel with external software and data sources to provide an accessible entry point for computational biology. Many researchers utilize Excel to capture and process research data, particularly for exploration, tabulation and visualisation. However, Excel does not readily process structured data files of the types used in bioinformatics – typically genome definitions and annotations or result datasets produced by software tools. Thus, if a user wishes to process this kind of data in Excel they are forced into a laborious document formatting task. Significant efficiencies can be obtained by hosting tools directly within the environment, and using worksheets as an input and output medium, removing the need to copy and reformat data and switch between applications.

To this end we introduce a custom Excel tool ribbon which provides a simplified API for programmatic interaction with Excel along with a standard procedure for integration of external programs and datasets with Excel. The tool ribbon together with a suitable collection of external tools allows an Excel workbook to be used as an executable scientific diary. As noted earlier, the principal interactions take place using the .NET Bio libraries, but the approach is far more general.

A simplified schematic view of the system architecture is shown in Figure 2. The tool ribbon is displayed as part of the Excel menu to provide access to external tools. Each tool is implemented by a small collection of cooperating classes residing in a tool integration assembly deployed alongside the add-in. When Excel starts, the add-in queries a configuration file to determine the identities and logical groupings of tools to be added to the ribbon. Each action that can be performed through the ribbon is represented by a button; buttons are laid out in visual groups according to the logical groupings specified in the configuration file.

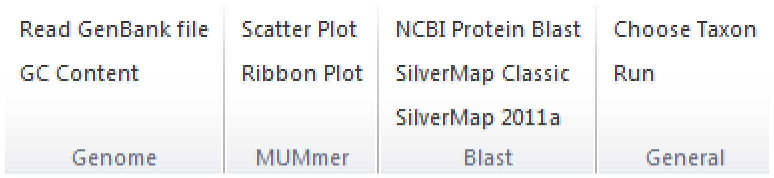


Figure 3: The QUT Bioinformatics Tool Ribbon

Figure 3 shows a view of the tool ribbon which has been configured to provide access to several functions exposed by the .NET Bio [8] library plus visualisation software developed in-house at QUT. The “Genome” group provides actions related to the processing of whole genomes: “Read GenBank File” allows the user to load the details of an annotated bacterial genome from a GenBank-formatted text file and store the content – DNA and gene annotations – into worksheets within the current workbook; “GC Content” calculates the distribution of the nucleotides guanine and cytosine along a selected strand of DNA. The actions in the group labelled “MUMmer” both perform a pairwise comparison between two selected strands of DNA using the MUMmer algorithm [11] to locate short stretches of identical DNA and plot the results. “Blast” actions allow the user to perform sequence similarity searches with NCBI Web BLAST [2] and visualise the results using QUT’s SilverMap software. The final group, “General” covers miscellaneous actions: “Choose Taxon” is a utility dialog which allows the user to browse the NCBI Taxonomy database and select a taxonomic ID; “Run” is used to execute tools.

The central ideas behind the add-in are:

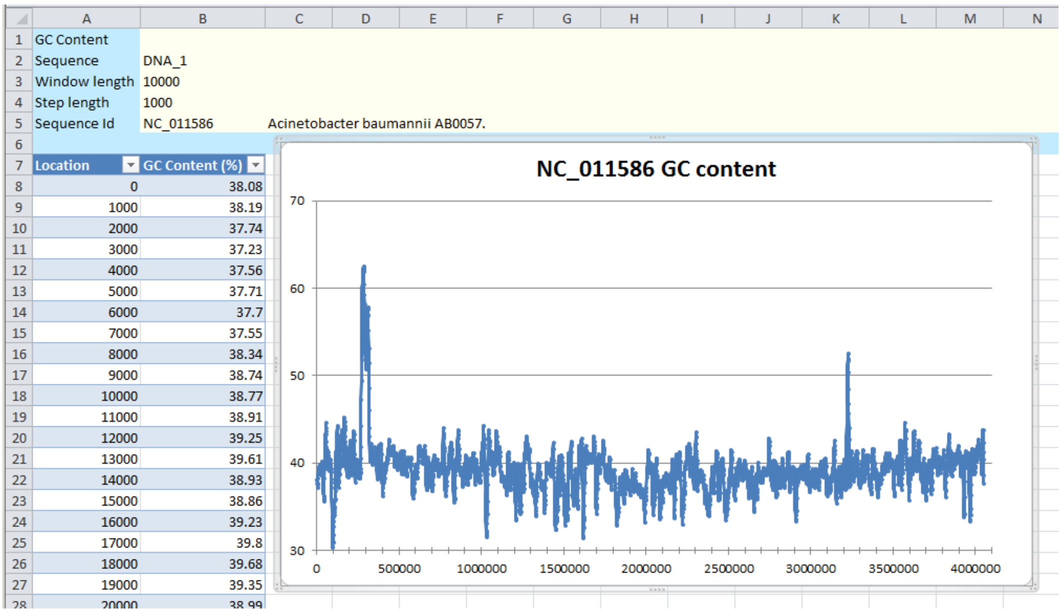
1. Special purpose worksheets called Pages mediate interaction with tools. Pages are created by clicking tool buttons in the ribbon.
2. Each page contains a metadata annotation which specifies the identity of the tool with which the page is associated. The value used is the fully qualified name of the .NET class that implements the tool.
3. Each page contains a table having rows which are records of a particular type associated with the tool. Initially the table is empty. The table may be used as a data entry instrument which is populated in some manner by the user, either via formulae or by hand, or the table may be used as an output device; this depends on the tool. This arrangement allows the output of one tool to be filtered in place then used as the input to another tool.
4. Most pages contain a simple input form where the user sets up parameters for the tool.
5. The user executes the tool associated with a page by clicking the “Run” button.

The initial release of the bioinformatics add-in provides five page types. A DNA page contains DNA records, each of which stores the name, length, ID and full DNA sequence of a bacterial chromosome. A CDS page contains CDS records in which details of protein coding sequences are stored: gene ID, gene function, location, orientation, symbolic name, locus tag and amino acid translation. A MUMmer page is used to execute the MUMmer sequence alignment tool and tabulate the results. Blast pages are used to perform protein homology searches via NCBI Web BLAST and tabulate the results, while GC pages let the user compute the GC content of a DNA sequence.

When the user clicks the “Run” button the identity of the tool is obtained from the page and an instance of the tool connected to the page is obtained. The tool then carries out the following generic sequence of operations:

1. Fetch parameter values from the page; read data from any input tables to which the page is connected.
2. Perform the operation to generate results.
3. Write the data back into the table. Optionally, the tool may also generate one or more Excel charts to display the data.

As an example, each time the user clicks the “GC Content” button the system adds a new GC page to the workbook. The GC page is a worksheet which has been formatted into two main areas. The top



	A	B	C
1	GC Content		
2	Sequence	DNA_1	
3	Window length	10000	
4	Step length	1000	
5	Sequence Id	NC_011586	Acinetobacter baumannii AB0057.
6			
7	Location	GC Content (%)	
8	0	38.08	
9	1000	38.19	
10	2000	37.74	
11	3000	37.23	
12	4000	37.56	
13	5000	37.71	
14	6000	37.7	
15	7000	37.55	
16	8000	38.34	
17	9000	38.74	
18	10000	38.77	
19	11000	38.91	
20	12000	39.25	
21	13000	39.61	
22	14000	38.93	
23	15000	38.86	
24	16000	39.23	
25	17000	39.8	
26	18000	39.68	
27	19000	39.35	
28	20000	38.99	

Figure 4: GC Content of *Acinetobacter baumannii* strain AB0057.

region of the document is set up as a data entry form where the user enters the identity of a DNA page that contains the DNA strand to be analysed and the numeric parameters for the calculation. The remainder of the spreadsheet is formatted as a table which will be populated with a list of GCRecord objects. When the "Run" button is clicked the system locates two tool objects, one of type GCPage which is connected to the GC page and another of type DnaPage which is connected to the DNA page in the workbook. The GCPage obtains the first visible DNA record from the DnaPage, computes the GC content then emits the results as a series of GC records in the GC table. Finally, the tool adds a chart to the worksheet which displays a plot of the GC content of the target DNA strand as depicted in Figure 4.

The page-and-table conceptual framework provided by the add-in lends itself naturally to a data-driven work pattern where the user executes a sequence of tools based on leads picked up by examining the data while maintaining a trail of intact worksheets which capture the inputs and outputs of each operation. To illustrate this, consider an extension of the example illustrated in Figure 4. Having obtained the results of the GC scan, the user is free to explore the result set using built-in Excel functionality to filter the table and observe the impact on the chart. In bacterial DNA, regions in which the GC content diverges significantly from the median tend to harbour biologically interesting genes: in the case of *A. baumannii*, a region of abnormally high GC content coincides with a large pathogenicity island [12]. Having identified a region of interest, the user might then switch to the CDS page to examine the genes located in that region. Excel's built-in research functionality might be used to discover more information about particular genes. The user could then select a few genes of interest and conduct a BLAST search to discover homologous genes in other organisms.

The add-in is designed to make creation and addition of new tools simple and non-invasive – computer science students in the third year of their degree should easily be equal to the task of encapsulating existing tools and libraries, as should some microbiologists, albeit with some degree of coaching on certain aspects of software development that may not be covered by in traditional scientific computation courses.

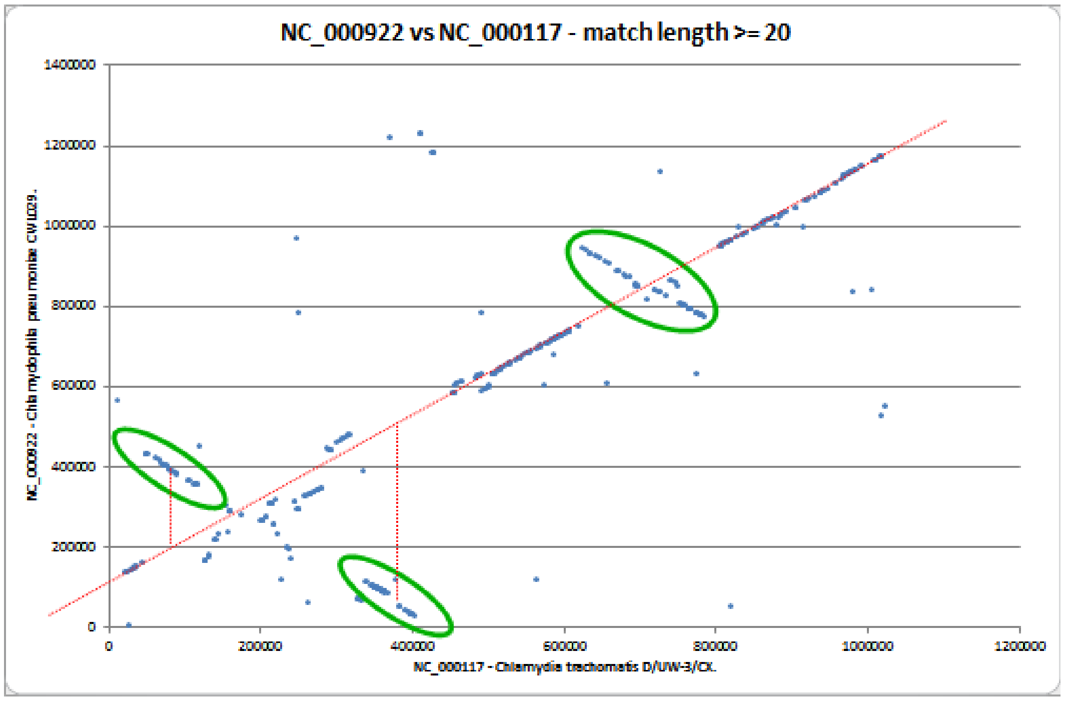


Figure 5: Large-scale symmetrical inversions between genomes of *C. trachomatis* and *C. pneumoniae* identified by the MUMmer tool.

Tools are programmed in a modern .NET language such as C#, F# or Visual Basic, and typically consist of up to four classes. It is mandatory to provide a class which can activate the tool when the user clicks the corresponding button in the tool ribbon. This class implements a very simple interface called IButtonHandler which is defined in the add-in support assembly. The button handler may perform arbitrary actions such as triggering a visualisation but one of the more common functions carried out is the creation of a new page which serves as the user interface for an external tool. Two classes are needed to define a page: one class is used to define the record type saved in the page's embedded table while another class implements the operations of the tool itself, responding when the user clicks the "Run" button. All information required to build the Excel worksheet is embedded in these classes via metadata annotations on the classes and their properties. The button handler and any ancillary classes are compiled into an assembly which is published along with the add-in. In the final step, the programmer adds an entry to the application configuration file to make the new tool available at run time.

4 Learning Activities

A suite of learning activities has been created to apply the QUT Excel extension in the setting of an undergraduate computational biology course. The activities, which are available as self-paced learning tasks at [13], are designed to provide useful educational outcomes for students while stimulating the development of new tools and learning activities by faculty. The initial set of activities aims to bring students up to speed with the facilities provided by modern versions of Excel and then move on to comparative genomics. The learning activities are described below: the first two activities should be completed in the order of presentation, while subsequent activities may be attempted in any order.

The first activity, "Work with GenBank Files," demonstrates basic functionality of the add-in: parsing an annotated bacterial genome from a GenBank-formatted text file using routines made

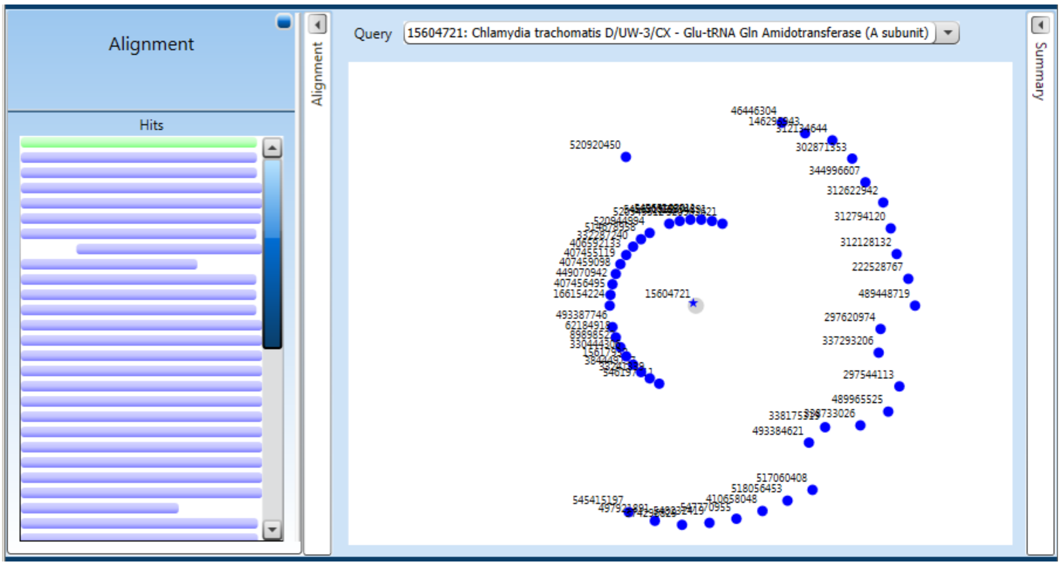

Alignment
Hits
Alignment
Query
15604721: Chlamydia trachomatis D/UW-3/CX - Glu-tRNA Gln Amidotransferase (A subunit)
Summary
15604721

Figure 6: Genes exhibiting a high degree of similarity to *C. trachomatis gatA*.

available by the .NET Bio library. The user selects one or more GenBank files and the contents are added to pages in the current workbook: the full DNA sequence of the organism is added to the DNA page while a list of records, each of which contains details of a protein coding gene, are added to the CDS page. The activity also demonstrates how to download bacterial genomes from the NCBI FTP server. Although rudimentary in nature, this action is the precursor to all subsequent activities.

A second activity, “Use Tables and Pivot Tables to Process Genomic Data,” makes students aware of some of Excel’s data management capabilities. Students learn how to create tables, how to sort and filter the contents of tables to organise information, and how to apply formulae to generate new information from the content of a table. Students also learn how to use a simple pivot table to summarise the contents of a collection of genomic information. This activity should bring students’ knowledge of Excel to a level where they are able to engage effectively with subsequent activities.

The activity “Produce a GC Map of a DNA Sequence” introduces the concept of GC Content – the probability that a nucleotide selected at random from that sequence is guanine (G) or cytosine (C). The lesson follows the process outlined in the extended example above to teach students how to plot the distribution of G+C nucleotides across a genome and analyse the results. Students use Excel’s filtering operations to identify regions of extreme GC content and identify the genes that occupy those regions.

In “Compare two DNA sequences with MUMmer” the .NET Bio implementation of MUMmer [11], is used to compare the genomes of two closely related bacterial species. Students create a scatter plot showing the relative locations of identical regions within a pair of sequences. By creating simple formulae and working with Excel's built-in data sorting and filtering operations they zoom in and explore regions of potential biological interest identified from the scatter plot. Students learn to recognise a range of distinctive evolutionary events such as insertions, deletions, transpositions and reversals by examining the scatter plots. An example which illustrates the effects of symmetrical DNA inversions is shown in Figure 5.

The “Search NCBI sequence databases with BLAST” activity and its sequel, “Analyse BLAST results with Excel and SilverMap” follow on from either of the two preceding activities. Having identified one or more protein coding genes of interest the student uses the NCBI Protein BLAST tool to search for genes which encode similar proteins. The tool tabulates the result set, a list of gene subsequences similar to the query gene, with numerical goodness-of-fit values, the location and extent of each matching subsequence and copies of the aligned subsequences. The integrated SilverMap visual analysis tool augments the tabular result display with a graphical view which shows the way subsequences align against the query sequence and a radar view which provides a tangible overview of sequence similarity. Figure 6 shows the outcome of a search for the gene *gatA* in *Chlamydia trachomatis* viewed in SilverMap. Aligned subsequences are shown on the left hand side of the display. The radar view appears on the right hand side: the query gene is placed at the centre of the map; icons representing the aligned subsequences are laid out so that the distance from the centre reflects the measure of similarity – sequences similar to the query are positioned close to the centre while less similar sequences are placed further away. A third panel at the extreme right of the display – collapsed in Figure 6 for clarity – presents the full detail of each match.

5 Experience and Conclusions

In this work we have presented a new approach to embedding a computational mindset into the education of undergraduate biologists while avoiding traditional barriers of programming experience and the need to learn complex data representations and languages. The toolset is freely available and builds on a virtually ubiquitous software tool in Microsoft Excel, a rich open source bioinformatics library in the .NET Bio system, and, through web service connections, a very broad range of publicly accessible data sets and established software tools in the field. The approach has been trialed in undergraduate bioinformatics practical classes at second year level for three semesters in collaboration with our colleagues Professor Peter Timms and Dr. Adam Polkinghorne. The approach has significantly enhanced the richness of the explorations possible in the timeframe. Even in these classes, however, the student experience may be hampered by differences in laboratory configuration, and in the difficulties some students experience in using cell reference formulae.

Much of this work will continue as a project associated with .NET Bio and our work in bioinformatics visualisation. It is expected that additional exercises will be developed through faculty activity and through ongoing student projects over the coming years.

References

[1] Benson D.A., Karsch-Mizrachi I., Lipman D.J., Ostell J., Wheeler D.L. (2006) *GenBank*, Nucleic Acids Res. 2006 Jan 1; 34(Database issue):D16-20.

[2] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990) *Basic local alignment search tool*. J. Mol. Biol. 215:403-410.

[3] Larkin M.A., Blackshields G., Brown N.P., Chenna R., McGettigan P.A., McWilliam H., Valentin F., Wallace I.M., Wilm A., Lopez R., Thompson J.D., Gibson T.J. and Higgins D.G. (2007) *ClustalW and ClustalX version 2*. Bioinformatics 2007 23(21): 2947-2948. doi:10.1093/bioinformatics/btm404

[4] Metzker M.L. (2010) *Sequencing technologies — the next generation*. Nature Rev. Genetics, Vol 11, pp.31-46

[5] Szalay, A.S., Blakely, J.A. (2009) *Gray's Laws: Database-centric Computing in Science*. In: Hey, A., Tansley, S., Tolle K. (eds) *The Fourth Paradigm: Data-Intensive Scientific Discovery*, pp.5-10. Microsoft

[6] Wing, J. (2006) *Computational Thinking*. Communications of the ACM March 2006/Vol. 49, No. 3, pp33-35.

[7] Tukey, John W. (1977). Exploratory Data Analysis. Addison-Wesley

[8] The .NET Bio Project, retrieved from http://bio.codeplex.com

[9] Hack, C. and Kendall, G. (2005) *Bioinformatics: current practice and future challenges for life science education*. Biochemistry and Molecular Biology Education, vol 33, no. 2, pp82-85.

[10] Isbell, C.L., Stein, L.A., Cutler, R., Forbes, J., Fraser, L., Impagliazzo, J., Xu, Y. (2010). *(Re)defining computing curricula by (re)defining computing*. SIGCSE Bull., 41(4), 195-207. doi: 10.1145/1709424.1709462

[11] Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., Antonescu, C. Salzberg, S.L. (2004) *Versatile and open software for comparing large genomes*. Genome Biology Vol 5, Issue 2, Article R12. http://genomebiology.com/2004/5/2/R12

[12] Smith, M.G., Gianoulis, T.A., Pukatzki, S., Mekalanos, J.J., Ornston, L.N., Gerstein, M. and Snyder, M. (2007) *New insights into Acinetobacter baumannii pathogenesis revealed by high-density pyrosequencing and transposon mutagenesis*. http://genesdev.cshlp.org/content/21/5/601.full

[13] The QUT.Bio.Excel Project http://bio.mquter.qut.edu.au/qut.bio.excel