HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTER THESIS

Ms.
Karolin Wiedemann

Detection of secondary structure motifs in long non-coding RNAs

2014

HELMHOLTZ
CENTRE FOR
ENVIRONMENTAL
RESEARCH – UFZ

Faculty **Mathematics, Natural Sciences and Computer Science**

MASTER THESIS

Detection of secondary structure motifs in long non-coding RNAs

author:
Karolin Wiedemann

course of studies:
Molecular Biology / Bioinformatics

seminar group:
MO12w1-M

first examiner:
Prof. Dr. rer. nat. Dirk Labudde

second examiner:
Dr. Kristin Reiche, Dr. Jörg Hackermüller

Mittweida, November 2014

Bibliographic description

Wiedemann, Karolin: Detection of secondary structure motifs in long non-coding RNAs, 87 pages, 3 figures, Hochschule Mittweida, University of Applied Sciences, Faculty Mathematics, Natural Sciences and Computer Science

master thesis, 2014

Abstract

The almost complete transcription of the human genome yield in a high number of transcripts, that do not encode proteins. However, the functional elucidation of especially long non coding RNAs is still difficult. Secondary structure analysis is assumed to be a possible method to detect functional relationships of lncRNAs on a large scale, but it is still time consuming and error-prone. GRAPHCLUST, the currently most suitable clustering tool based on RNA secondary structure analysis, lacks mainly in an efficient method for the interpretation of its results. Hence, an independent and interactive RNA clustering interpretation tool was developed to allow visualisation and an efficient analysis of RNA clustering results.

I. Contents

II. List of Figures

III. List of Tables

IV. Nomenclature

3D	three-dimensional
A	adenine
app	application
B cells	lymphocytes, which mature in bone marrow
C	cytosine
CD	clusters of differentiation
CLP	common lymphoid progenitor
CM	covariance model
DN T cells	double negative T cells ($CD4^-CD8^-$)
DP T cells	double positive T cells ($CD4^+CD8^+$)
ETP	early thymic progenitors
G	guanine
GO	gene ontology
GTF	general transfer format
HPC-cluster	high-performance computing cluster
I	inosine
ID	identifier
IL	interleukin
kb	kilo-base pair, 1000 nucleotides
lncRNA	long non-coding RNA
MFE	minimum free energy

MHC ..... major histocompatibility complex

MPI ..... mean pairwise identity

NK cell ..... Natural killer cells

NSPDK ..... neighbourhood subgraph pairwise distance kernel

RNA ..... ribonuclein acid

SCFG ..... stochastic context free grammars

SCI ..... structure conservation index

SGE ..... Sun Grid Engine

SP T cells ..... single positive T cells ($CD4^+CD8^-$ / $CD4^-CD8^+$)

SQL ..... structured query language

suppl. ..... supplemental/supporting information

T cells ..... lymphocytes, which mature in thymus

T_{reg} ..... regulatory T cells

TCR ..... T cell receptor

TGF ..... transforming growth factors

Th ..... T helper cells

widget ..... window gadget

Acknowledgements

Simply go on climbing. It's like writing a novel. Everything's quite straightforward at first - the early chapters go with a tremendous swing, but sooner or later you begin to tire. You look back and see you're only halfway through. You look ahead and see you still have as much again to write. If you lose heart at that stage, you've had it. It's easy enough to start something. Finishing it is the hard part.
(Walter Moers - City of Dreaming Books)

With this in mind, I wish to thank all people who have helped me to overcome the challenges of the Bachelor and the Master degree program.

First of all, I would like to express my thanks and great appreciation to Dr. Kristin Reiche of the Helmholtz Centre for Environmental Research (UFZ), Leipzig, for her encouraging and comprehensive supervision during the entire duration of my master thesis. Special thanks also to Dr. Jörg Hackermüller, UFZ Leipzig, for his advice and comments as well as to the entire Young Investigators Group Bioinformatics and Transcriptomics (YIGBT).

On the part of the university of Mittweida, I am particularly grateful for the sedulous support given by Prof.Dr. Dirk Labudde, who is a highly dedicated and motivating person.

Furthermore, I want to thank Steffen Heyne, Max Planck Institute of Immunobiology and Epigenetics in Freiburg, for the support in the context of GRAPHCLUST as well as Ben Langenberg and Christian Krause, UFZ Leipzig, for the technical support.

Last but not least, without the help of my family and my friends, I wouldn't have reached this aim.

1 Introduction

Research like the ENCODE-Project showed that eukaryotic genomes are transcribed almost entirely, with only 1.5-2% of the transcripts encode proteins. Thus, the majority belong to non-protein coding RNAs (ncRNAs) [The07]. The smaller part of these are short ncRNAs, whereas the mass is named long ncRNA (lncRNA), because they are of at least 200 nucleotides (nt). In contrast to short ncRNAs, lncRNAs are less well understood. The few analysed lncRNAs show involvement in transcriptional or postranscriptional gene regulation, including epigenetic processes like reorganizing chromatin structure by protein interaction. Nevertheless, for the majority of lncRNAs the function is still unknown [WC11, WKG14].
While protein-coding genes are often elucidated by sequence conservation, this approach yield no or just poor results in case of long non-coding RNAs, because of quite distinct sequences. However, the three-dimensional structure is believed to be evolutionarily conserved to maintain essential functions on RNA level, because destroying this structure means losing the function and decreasing the fitness of the organism. But determining 3D-structures is a complex problem and not yet applicable for a comprehensive similarity search. Instead, the secondary structure is used, which is only defined by base pairings, and which forms the basis for 3D interactions.
Finally sequence and structure similarities (motifs) allow to divide ncRNAs into families, clans and classes. This classification provides a quick overview to elucidate relationships and functions. For example similar structure motifs hint at similar binding partners, as it is observed for protein-binding short ncRNAs [LRB$^+$12].
While short ncRNAs can be classified well by their secondary structures, it is still open whether common sequence and/or structure motifs among lncRNAs exist. In order to infer such motifs one could use clustering approaches (unsupervised pattern recognition for classification). But current clustering methods either need a known secondary structure as input, which do not allow to find clusters of unknown structures, or they are critical in time requirement. Heyne et al. [HCRB12] introduced 2012 a new clustering tool named GRAPHCLUST, which is said to be applicable to 'hundreds of thousands of sequences'.

The aim of this work, is to establish an complete pipeline based on GRAPHCLUST, that allows to analyse data of a transcriptome wide study to identify secondary structure motifs in and relations between lncRNAs. This includes firstly the adaptation of GRAPHCLUST to the given technical conditions and to the requirement of large dataset processing. Secondly, the development of an framework for visualisation and interpretation of RNA clustering results is needed to efficiently conclude biologically functional motifs from the derived clusters. Finally, the pipeline should be exemplarily applied to a transcriptome-wide dataset of long-non-coding RNAs differentially expressed during the T cell development.

2 Background

2.1 T cell development

T cells belong together with B cells and Natural killer cells (NK cells) to the group of lymphocytes. These are white blood cells, which are responsible for immune defence to impurities and pathogens. While T and B cells, also referred to as T or B lymphocytes, are important for the adaptive (specific) immune response, NK cells are involved in the innate (unspecific) immune response. B cells release antibodies into the body fluids (humoral immunity), whereas T and NK cells mainly function through direct cell contact (cell mediated immunity). T cells express T cell receptors (*TCR*) to discover changes on other cell surfaces by antigen recognition. These receptors are developed during maturation in the thymus. The overall T cell development can be divided in several stages, while it starts with the progenitor cells deriving from the bone marrow.

Progenitor cells
All T cells have their origin in the bone marrow. Here ***hematopoietic pluripotent stem cells*** differentiate stepwise into ***thymus settling progenitor cells*** [KR11, LK06]. It is still controversial, how much these progenitors are lymphoid committed respectively T cell committed [KR11]. Different progenitor populations are discussed. For instance ***early thymic progenitors*** (ETP), which already have moved into the thymus, were observed in vitro and showed potential for T cells and myelocytes (non lymphatic blood cells). Koch and Radtke [KR11] mention on the other side a study, which tracks lymphocyte development in vivo. They did not recognise significant myeloid potential at the ETP stage. Thus, the authors confirm the existence of classical ***common lymphoid progenitor*** (CLP), a cell with T, B and NK cell potential. They further assume myeloid potential might be repressed by the physiological micro-environments. These are for instance the conditions of the different zones within the thymus, where further T cell differentiation takes place (see Fig.2.1).
Beside the local assignment, the differentiation process of T cells within the thymus is usually described by specifically expressed cell surface markers [KR11]. Such markers compose the cells immune phenotypes and are used to classify cells and to track their increasing differentiation (clusters of differentiation, short CD) [Kau14, ZSB$^+$07]. The thymic T cell lineage can be characterised by the expression patterns of the two co-receptors CD4 and CD8 (for further details see Fig.2.1):

Double Negatives (DN)
The early thymic progenitor lacks both co-receptors ($CD4^-CD8^-$), thus it is referred to as Double Negatives. This stage is divided in four sub-stages:
DN1 cells are the most undifferentiated DN cells within the thymus. They are located within the thymic corticomedullar zone, where they seem to be stimulated by cortical

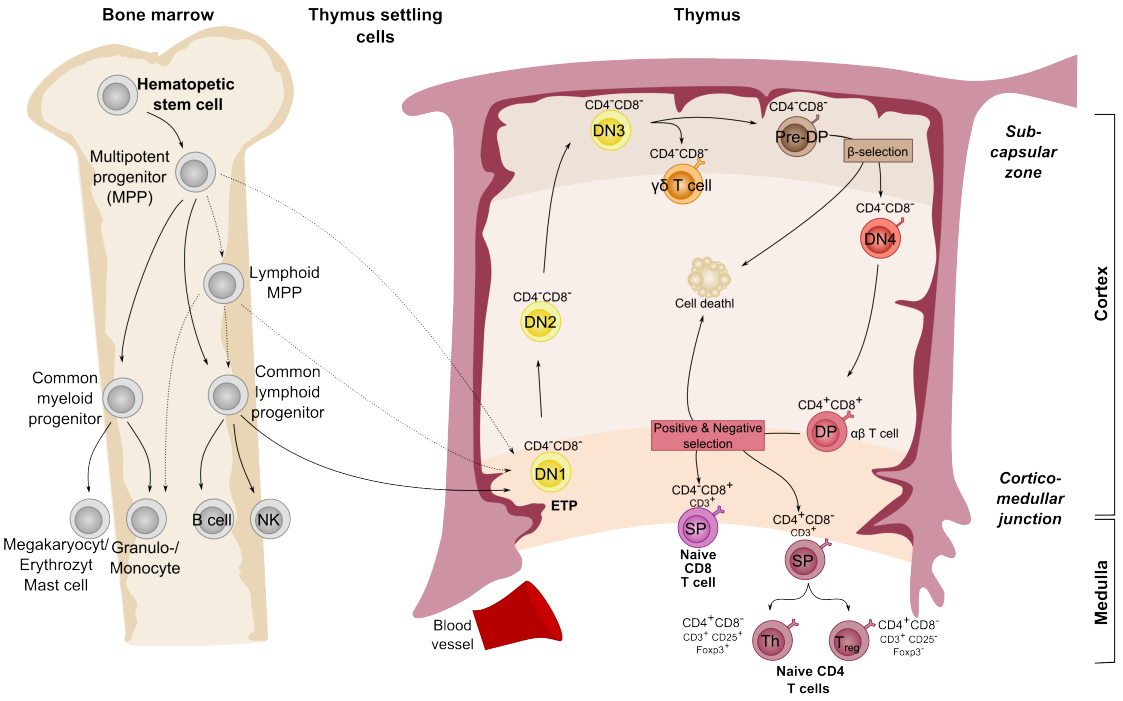

Figure 2.1: **T cell development.** Adapted from [BM04, RMY08, LK06, MCG+13, RMTC13]. T cells have their origin in bone marrow. Here hematopoietic pluripotent stem cells differentiate in thymus settling cells (left). However, it is unclear how much these are committed to the T cell lineage. Different models are discussed (dashed arrows). In general (full arrow) a common lymphoid progenitor is assumed, which has also potential for B cells and Natural killer cells (NK) (full arrow). The cells finally maturate within the thymus (left). This can be tracked by T cell specific surface markers (CD), while +/- describes their expression. CD4 and CD8 are mainly used, so that three subtypes are clearly distinct: Double Negative (DN), Double Positive (DP) and Single Positive (SP). The early thymic progenitor (ETP) belongs to the DN fraction. These T cells divide in $\alpha\beta$ or $\gamma\delta$ types depending on the expressed T cell receptor. Only the $\alpha\beta$ types reach the DP stage. Cells of defective TCRs are discarded by triggering apoptosis during one of the three selections steps (β, positive, negative). Finally, mature, but inactive (naive) SP T cells leave the thymus, CD4 cells are mostly T helper cells (Th) or regulatory T cells (T_{reg}), while CD8 cells usually function as cytotoxic cells.

thymic epithelial cells [KR11, KFB+08]. These cells carry the transmembrane protein *DLL4*. This is a ligand to the *Notch1* receptor on the DN cell surface. Removing DLL4 seems to cause B cell development in thymus, instead.

DN2 stage follows by moving trough the cortex into the thymic subcapsular zone, where the development of the T cell receptor (TCR) starts [KR11]. This is a process of gene rearrangements of the *TCR* loci, that generate one of two *TCR* types. Mature T cells carry finally either a *TCR*$\alpha\beta$ or a *TCR*$\gamma\delta$. During phase DN2 the genes of the β-, γ- and δ-chains are rearranged [KR11, YR14]. However, first evidence to one of the two fates is given during this stage [KR11]. High expression of the Interleukin-7 receptor (*IL-7R*) points to future $\gamma\delta$ cells, while low expression correspond to future $\alpha\beta$ cells.

The expression of genes, which are significant for the $\alpha\beta$ T cell lineage (e.g. *Ptrca, Rag1*), does not increase until the end of stage DN2 .
DN3 cells still remain within the subcapsular zone, while the *TCR* rearrangements proceed [KR11]. This stage ends with the result that some cells express functional $TCR\gamma\delta$, whereas the others express a pre-*TCR* with a functional $TCR\beta$-chain [YR14]. While the former remain in the DN stage and mainly function as cytotoxic cells in derma, intestine and lung epithelium [CHS00, Kau14, Spe99], the latter enter a fourth DN stage and finally a Double Positive stage. However, before changing the stage, pre-*TCR* cells are subject to the β-selection, where apoptisis is induced to cells with defective $TCR\beta$-chains [MZP02].
DN4 are conform $TCR\beta$-cells, that start to re-migrate into the corticomedullar zone of the thymus and up-regulate CD4 and CD8 expression [KR11].

Double Positives (DP)
$TCR\beta$-T-Lymphocytes within this stage show highest expression rates of the two co-receptors CD4 and CD8 ($CD4^+CD8^+$) and initiate rearrangement of $TCR\alpha$ [KR11,YR14]. Finally non-functional $TCR\alpha\beta$ cells are eliminated by positive selection of functional ones. For this purpose thymic epithelium cells present the major histocompatibility complex (*MHC*) to these cells [YR14]. This is an endogenous component, which is used to present antigens on the cell surface and which is required for antigen recognition by $TCR\alpha\beta$ lymphocytes. Non-funtional T cells do not bind to MHC and perish within the thymus.

Single Positives (SP)
By the help of the positive selection process the $TCR\alpha\beta$ T cells differentiate into two distinct subclasses, which now express only one of both co-receptors CD4 or CD8. This division depends on the class of the recognised MHC, which exist as MHC class I and MHC class II [CHS00]. T cells, which show only CD4 co-receptors ($CD4^+$) mainly associate with MHC class II, whereas CD8 single positives ($CD8^+$) prefer MHC class I. A further selection step rejects auto-reactive cells by negative selection within the medulla [KR11, YR14]. This means cells binding MHC stronger than required remain within the thymus, which leads them to apoptosis. Correctly functioning SP cells finally increase expression of the sphingosine-1-phosphat receptor 1 ($S1P_1$) to leave thymus and enter the peripheral lymphatic system with higher $S1P_1$ concentration. Here they are able to move through the organisms body as naive (mature, but inactive) T-Lymphocytes, which express CD3 [CHS00, Kau14]. They become active by interaction with antigens and are called effector T cells then.
$CD4^+$ cells differentiate in numerous effector types [CHS00, ZCL09, Kau14]. Their differentiation fate is mainly induced by cytokines (growth and differentiation regulating proteins) in the microenvironment, which were produced by antigen-presenting cells, that interact with the $CD4^+$ cells (cf. Fig.2.1). The majority of them becomes ***T helper (Th)*** cells, which can be divided again in several subclasses. *Th1* cells were stimulated by the cytokines Interleukin 12 (IL-12) and IL-γ. They are finally a part of the protec-

tion against intracellular microorganisms. In contrast, Th cells, that are stimulated by IL-4, differentiate into *Th2* cells. These are involved in the elimination of extracellular pathogens. Another prominent example is the subset of *Th17* cells, which is required for controlling extracellular bacteria and fungi. This subset emerges trough the influence of TGF-β (transforming growth factor β) together with IL-6, IL-21 and IL-23.
SP CD4$^+$ cells can also occur as ***regulatory T cells (T_{reg})***, if they were stimulated by TGF-β and IL-2. This class differs from the Th class in the production of CD25 and the transcription factor *Foxp3*. Thus, T_{reg}'s can be described by CD4$^+$CD8$^-$CD25$^+$Foxp3$^+$, while Th cells show the signature CD4$^+$CD8$^-$CD25$^-$Foxp3$^-$. T_{reg} cells suppress immune reactions in order to finish them after infection elimination and to prevent autoimmune reactions.
CD8$^+$ cells, instead, mainly have cytolytic effects on virus-infected endogenous cells by cell-cell interaction [Kau14]. But also CD8$^+$ Th cells are known [Rie94]. These additionally express CDw60.

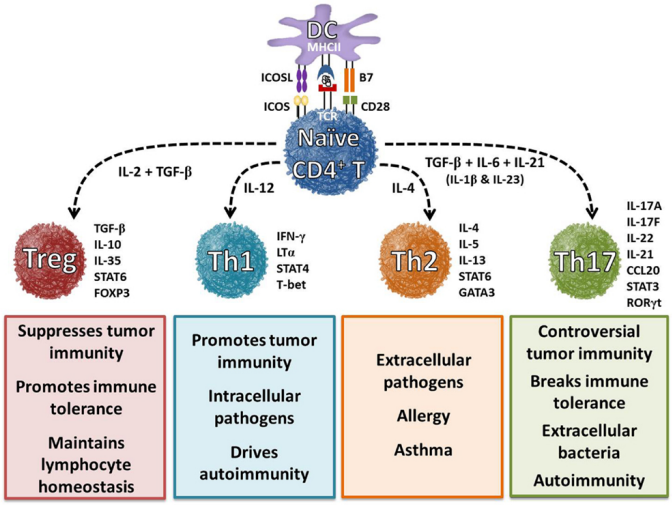

Figure 2.2: **Differentiation of CD4$^+$ T cells depends on cytokine microenvironments.** Taken from [BNH$^+$14]. Antigen-presenting cells (DC) produce, during interaction with naive CD4$^+$ T cells, proteins that regulate cell growth and differentiation (cytokines). The specific cytokines, i.e. interleukins (IL) and transforming growth factors (TGF), determine the T cells differentiation fate. Thus, four major subset are known: T-helper cells Th1, Th2 and Th17 as well as the regulatory T cells (T_{reg}). These cells produce, finally, cytokines by themselves to stimulate specific immune responses to pathogenic cells.

In summary the T cell differentiation is a complex process with many influences to identify and to clarify. Thus, it is still controversial, how the fate of a cell is determined.

2.2 Long non-coding RNAs

The DNA sequence of an organism consists of just four nucleotides (Adenin, Cytosin, Guanin, Thymin), but still it has to encode all information required to define its phe-

notype. Hence, one of the most fundamental tasks in biological research is to detect functional elements and their regulatory mechanisms, which interpret the organisms phenotypic complexity within this simple four letter code. For many decades, protein-coding genes were the best known functional genomic elements. However, the number of these genes does not correlate with the complexity of the organism, i.e. the number of cell types. For example, thread worms of the species *Caenorhabditis elegans* own 20,541 genes, whereas in human genome 20,805 are known (Ensembl release 75). According to current research, most genomic loci are actively transcribed, with just a small fraction of transcripts ($\approx$2%) translated into proteins [Mat03]. Thus, the vast majority of the genomic regions does not follow the classical central dogma of molecular biology, which only a one-way flow of genetic information. According to that, DNA is read and transcribed into intermediary RNA (messengerRNA, mRNA), which is finally translated into functional proteins (DNA$\rightarrow$RNA$\rightarrow$protein). Most genomic regions are transcribed, but the transcripts are not translated into proteins. Consequently such transcripts are called non(-protein) coding transcripts (ncRNAs). Furthermore, ncRNAs are believed to form an additional layer of genomic regulation [Mat03]. Hence, the intergenic space is not longer termed 'junk DNA'.
During the last decade knowledge has been growing fast especially for small/short ncRNAs with less than 200 nucleotides. Many subgroups are now known beside transfer RNAs (tRNAs), like for example microRNA (miRNAs) or small nucleolar RNAs (snoRNAs). In contrast, the description of the significantly larger group of ncRNAs longer than 200nt, so called long non-coding RNAs (lncRNAs), is still in its infancy. Some examples of lncRNAs are associated with regulatory functions, but for most of them it is unclear whether they are biologically functional or if they are just transcriptional noise. Although, first long non-coding RNAs *H19* and *Xist* were already described in the early 1990 [KCL13], new transcripts are still described and characterised individually. However, no general sequence or structure characteristics have currently been identified for lncRNAs. Hence, the division of lncRNAs into subclasses is hindered, not to mention systematic functional elucidation.

2.2.1 Epigenetic and genomic characteristics of lncRNA

The genomic loci of lncRNAs show similar characteristics to the genomic loci of protein-coding genes [WKG14, GR12, KCL13]. They are also often encoded in exon-like structures, which are spliced into several isoforms. Even transcription by the enzyme *RNA polymerase II* and polyadenylation of the transcripts were found in many cases [GR12]. Another common characteristic is the epigenetic labeling. This is no attribute of the DNA by itself but of DNA associated proteins named histones. Histones build complexes, which are used to roll up the DNA and form a higher condensation level. Modifications of these histones regulate DNA activity by en- or disabling its accessibility [SGE09, p.333,382]. Interestingly, both protein- and non-coding loci exhibit upstream, at their promoter site, a short series of histones with a trimethylated (me3) aminoacid Lysin (K)

at position 4 in protein H3 (H3K4me3). Furthermore, their transcriptional active sites are covered by histones with a methylation (me) of Lysin (K) at position 36 in protein H3 (H3K36me). Thus, this epigenetic labeling has been used to identify numerous members of the prominent class of long intergenic ncRNAs (lincRNAs). The difference to protein-coding sequences is, in particular, the missing protein encoding and on average a lower number of exons [GR12].

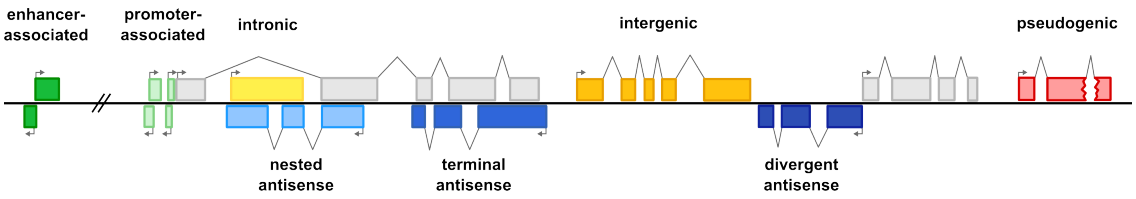

Figure 2.3: **Genomic locations of lncRNAs.** Adapted from [KCL13], this figure illustrates the genomic regions, in which lncRNAs are encoded in. Many lncRNAs have been found within intergenic regions (orange). However, they may also arise from regulatory elements (green) of protein-coding genes (grey) or from their introns (yellow). Even some defective copies of protein-coding genes, so called pseudogenes, are transcribed and can function as ncRNAs (red).
Additionally, for many coding and non-coding regions the antisense strand also encodes transcripts (blue). These antisense regions are mostly located around the sense promoter or terminator region, but completely nested or divergent sense-antisense pairs are known, too.

In addition to this, a prominent number of non-coding transcripts can be found in antisense reading direction to the known loci. Moreover, it does not matter, whether these loci are coding or non-coding [KCL13, KTK+05], but mRNA-lncRNA pairs have been observed more frequently than lncRNA-lncRNA pairs. Corresponding instances are the mRNA *Igf2r* together with the lncRNA *Air* [KCL13] and the lncRNA pair *Xist* and *Tsix* [KCL13]. The genomic locations of sense-antisense pairs may be divergent, overlapping or nested. More often the antisense transcripts tend to be located either at the 3' or the 5' region of the sense gene (Fig.2.3). By the way it can be mentioned, antisense transcripts are not limited to lncRNAs, even mRNAs occur as antisense transcripts. *WRAP53*, for example, is the antisense mRNA to the mRNA of the protein *P53* [MHC+09, MHF+11]. Apart from intergenic transcripts, other ncRNAs have been described, which originate from protein-coding regions. As shown in Fig.2.3, they can be assigned to introns, but also to enhancers or promoters nearby transcription start sites [KCL13]. Especially enhancer-associated transcripts (eRNAs) differ from the above mentioned lncRNAs, because of a distinct characteristic epigenetic label (H3K4me1) and the involvement of the transcriptional coactivator *p300* [GR12, KCL13, OrS13].
Finally, imperfect copies of protein-coding genes, which lost their messenger function, have been described to act as functional lncRNAs [KCL13, PWC+11]. These so called pseudogenes are caused by accumulated mutations within a copy [SGE09, p.338].

Independent of the previous mentioned characteristics, many lncRNAs have evolutionarily conserved promoters, but in general their expression patterns seem to be high spe-

cific regarding cell-type, tissue and developmental stage. Thus, *H19* was found to be involved in mouse liver development, whereas *Xist* showed involvement in inactivation of one of the two X chromosomes in female mammals [GR12]. Moreover, Hackermüller et al. assume: lncRNAs seem to be transcribed pathway specifically [HRO+14]. Accordingly, many different functions for lncRNAs have been described until now [GR12].

2.2.2 Functions and molecular mechanisms

LncRNAs are described to be involved in several cellular processes, like establishing epigenetic modifications, regulation of transcription and RNA processing, or even in translation [PRP+13, KCL13]. Hence, they have crucial influence on gene expression and can play important roles in diseases [PRP+13].
For example, instances of transcribed pseudogenes control the expression of corresponding parent genes by epigenetic changes. Such an instance is the ATP-binding cassette (ABC) transporter *ABCC6*, whose mRNA expression is decreased by its pseudogene *ABCC6P1* [KCL13, PWC+11]. The eRNAs, instead, have been associated with differential expression of neighbouring protein-coding genes [OrS13], while few examples of promoter-associated ncRNAs showed interactions with epigenetic factors [KCL13]. In addition, Reiche et al. [RKS+14] revealed the role of antisense lncRNAs within breast cancer.

LncRNA effects are primarily distinguished in cis and trans activities. Cis-acting means 'on this side' and is the regulation of genes, which are located on the same DNA molecule. Whereas, trans denotes 'on the other side' and defines effects on genes of other alleles or chromosomes [AAA+14, p.164] [GR12]. All in all, for lncRNAs cis activity has been observed less frequently than trans activity until now [GR12]. However, an lncRNA is not necessarily limited to one of both types [KCL13]. Some transcripts, e.g. *Xist*, may act as cis regulator as well as trans regulator [JL11].
It is assumed that lncRNAs interact with proteins as well as RNA and DNA. Based on guilt-by-association and loss-of-function experiments several models have been developed:

Molecular guides have been described from the observation, that some lncRNAs bind chromatin associated proteins and carry them to their specific loci. They were firstly observed for an lncRNA of the inactivated X chromosome of female mammals [GR12]. The X-inactive specific transcript, short *Xist*, is highly expressed before X chromosome inactivation to cover its host chromosome. This covering enables to catch silencing factors like the Polycomb repressing complex 2 (*PRC2*) and bring them close to the chromatin. *PRC2*, for example, is reponsible for establishing the histone modification H3K27me3 of inactive chromatin [KCL13, ZSE+08]. This case clearly demonstrates the cis-activity of *Xist*.

Molecular scaffolds are another type of linkers. They bind diverse molecules to assemble a functional complex [GR12, PRP+13, KCL13]. This model was characterised by the DNA synthesising enzyme *Telomerase*, which maintains the chromosomal ends. It comprises a telomerase RNA component (*TERC*), that interacts with the *polymerase* enzyme and several accessory proteins. Simultaneously, *TERC* is used as template for the missing telomeric repeats [GR12, WC11]. In addition, it is proposed that lncRNAs can also be flexible modular scaffolds, that own discrete domains for various specific molecules [GR12]. Particular combinations of these components may then form unique functional complexes. Thereby, other RNAs and also DNA are not excluded.

Molecular decoys, instead, bind Proteins or RNA to prevent their activities. One example is the lncRNA *Tsix*, which is an antisense transcript of *Xist*. Zhao et al. [ZSE+08] proposed, that *Tsix* binds *PRC2* as well and that it seems to protect the prospective active X chromosome by competitive inhibition of *PRC2*. More prominent is *Gas5*, the growth arrest-specific 5 lncRNA. It disables hormone receptors for glucocorticoids, by reserving its DNA-binding site, that usually interacts with hormone response elements within the genome [GR12, KCL13].

miRNA sponges are RNAs with multiple binding sites for miRNAs [KCL13, ENS07, CCL+11]. These binding sites are short sequences, that are complementary to the characteristic miRNAs seed sequences of around seven nucleotides. Hence, whole miRNA families can be inhibited [ENS07]. Thus, miRNA sponges are a special type of 'molecular decoys', but they are more efficient in inhibiting mRNA repression by miRNAs [KCL13, CCL+11]. The *linc-MD1*, a muscle-specific lncRNA, has 36 highly conserved miRNA binding sites, including two for *miR-135* and one for *miR-133* [CCL+11]. These miRNAs repress the expression of muscle-specific genes by inactivation of their transcription factors. The sponge activity of *linc-MD1* is based on a lower binding energy. Consequently the lincRNA-miRNA complex is a more favourable energetic state than the mRNA-miRNA complex.

Precursors are lncRNAs, which host small ncRNAs, like miRNAs or snoRNAs [KCL13]. *Gas5*, as such an example, contains ten highly conserved snoRNAs, whereas *H19* comprises *miR-675*.

To sum up, the activities of lncRNAs are derived from single examples, which were studied intensively in wet lab. Nevertheless only a little is known about the functionality and function of most lncRNAs. Experimental elucidation of lncRNA functions in wet labs is still time consuming, thus not suitable on a large scale.

2.2.3 Sequence and secondary structure

A well established and fast method to elucidate functions is based on computational identification of similarities on sequence level. Unfortunately this requires well con-

served sequences like they are observed for mRNAs or amino-acid sequences. LncRNAs, instead, are less well conserved [WKG14, GR12, KCL13]. Although sequence constraints have already been found for lncRNAs, the approach of sequence similarity often does not yield a result, because they are related, in particular, by short conserved pieces within large non-conserved areas [WKG14].
However, it is assumed that the function of lncRNAs is particularly influenced by their three-dimensional (3D) structure, since lncRNAs are single stranded and able to fold back and bind themselves [TUL71]. Furthermore, in case a structure is essential for the function of an RNA, it is likely to be conserved, because destroying this structure means losing the function and decreasing the fitness of the organism [ED94]. But determining 3D-structures is a complex problem and not yet applicable for a comprehensive similarity search [WWH$^+$12]. Alternatively the secondary structure can be used. It is only defined by base pairings, which form the basis of 3D interactions. The most popular example of a secondary structure is the cloverleaf structure of tRNAs [WWH$^+$12]. Such structural similarities (motifs) can hint at similar binding partners, as it is observed for protein-binding short ncRNAs [ED94, LRB$^+$12]. While short ncRNAs can be classified well by their binding motifs, it is still open whether common sequence and/or structure motifs among lncRNAs exist.

2.3 lncRNAs in T cells

There are already some few studies about lncRNAs in T cells [PDM$^+$09, HTS$^+$13, XDY$^+$14]. The lncRNA of *TMEVPG1* is probably the first mentioned example in this context. Other names are *NEST*, *IFNG-AS1* or LincR-*Ifng*-3'AS. It was firstly described by Vigneau et al. [VRBB03], who studied resistance to Theiler's virus infections, a mouse model system for multiple sclerosis. *TMEVPG1* expression was only observed in resistant unstimulated CD4$^+$ T cells, CD8$^+$ T cells and in NK cells. It is transcribed in 'convergent' antisense (see Fig.2.4) to the gene of interferon gamma (*IFNG*), a cytocine expressed in response to intracellular infections and tumors. After antigen stimulation these two genes showed negatively correlated expression for mouse and human. Vigneau et al. [VRBB03] concluded down-regulated expression of *IFNG* by *TMEVPG1*. Collier et al. [CCW$^+$12] took up this issue, but they described *TMEVPG1* as specific lncRNA of Th1 cells and as beneficial to *IFNG* expression. In addition, the expression of both genes is based on the presence of the Th1-specific transcription factors *STAT-4* and *T-bet*. Thus, they concluded the regulation of the Ifng transcription by *TMEVPG1* is linked to the Th1 differentiation fate [CCW$^+$12].

Pang et al. [PDM$^+$09] were the first, who focused on a genome-wide lincRNA characterisation in CD8$^+$ cells. Several hundred/thousand lncRNAs were identified for the human/mouse, while about 1% is both lymphoid-specific and differentially expressed in CD8$^+$ cells. Many of these lncRNAs are nearby protein-coding genes, that are associated with T cell differentiation and activation. About a fifth of them seem to be organised

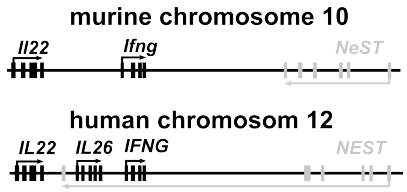

Figure 2.4: **Genomic context of *NEST*,** taken from patent US 2014/0056929 A1 [KBGC14]. The lncRNA *NEST*, also known as *TMEVPG1*, IFNG-AS1 or LincR-*Ifng*-3'AS, is located in antisense to the cytocine gene interferon gamma (*IFNG*). But both genes do share a common promoter. Instead, their transcription sites converge. This applies to both mouse (murine) and human.

in antisense to protein-coding genes. Two examples are the antisense lncRNAs of the lymphoid enhancer binding factor 1 (*Lef1*) and the protein tyrosine phosphatase receptor type E (*Ptpre*). Both of them overlap at least one isoform of its protein-coding gene. Thus, the authors suggest a regulative function for these antisense lncRNAs. But also intronic lncRNAs, for example within the interleukin 2 receptor *IL2Rα*, as well as promoter overlapping or small RNA overlapping lncRNAs were found.
Since promoters of mammalian protein-coding genes are often related to a high CpG-content in case of 'housekeeping function', respectively to a low CpG-content in case of tissue-specificity, a CpG-analysis was done for promoters of lncRNAs expressed in $CD8^+$ cells. One third showed a low content and one fifth a hight content. The authors consider this result to be a confirmation of the fact, that lncRNAs mostly have tissue-specific effects. Finally primary sequences and secondary structures were studied. Corresponding to the lncRNA knowledge the primary sequences were weakly conserved as a whole, but show some conserved elements. Furthermore, secondary structures of small ncRNAs and novel secondary structures have been detected within lncRNAs. For instance a non-coding isoform of the chemokine *Ccl4*, which is critical for T cell adhesion and migration, seems to form a large, double-stranded hairpin. All in all this study proves the existence of lncRNAs within the T cell lineage and confirms the properties of them. However, no specific functional elucidation was made.

Hu et al. [HTS^+13] focused on transcriptome analysis during T cell development from early T cell progenitors to Th cells. Here more than 1524 lincRNAs have been identified, with 73% not unnotated. The comparison of the expression patters of the different T cell stages showed stage and lineage specificity for more than 50% of that lncRNAs. Additionally, it was shown, that lncRNAs are often polyadenylated, spliced and dynamically regulated. Even in this study the T cell lncRNAs are close to genes, which are associated with functions of immune-regulatory functions. Interestingly many of these lncRNAs show regulation through transcription factors (TF), which are significant for T cell lineage. For instance, *GATA-3* and *STAT-4* are critical for Th1 differentiation, while *T-bet* and *STAT-6* are critical for Th2 cell differentiation. A stronger binding between such a TF and a lncRNA seems to point out in which T cell lineage the lncRNA is expressed. For example, *LincR-Ccr2-5' AS*, an antisense lincRNA cluster of the chemokine recep-

tor gene Ccr2, is preferentially expressed in Th2 cells and is controlled by *GATA-3*. This lncRNA was then further investigated and showed a correlation to protein-coding genes involved in chemokine-mediated signalling pathways. Artificial depletion of *LincR-Ccr2-5' AS* led to significant impairment in tissue migration, down-regulation of genes enriched in gene ontology (GO) terms like cell cycle and nuclear division, but up-regulation of genes known for immune system processes and defence response. The positive regulated genes are often co-expressed ones. The assumption *LincR-Ccr2-5' AS* causes a epigenetic regulation to them, could not be confirmed.

Most recently Xia et al. [XDY+14] published a study about lncRNA within T cell development. The four stages DN, DP, naive $CD4^+$ and activated $CD4^+$ were compared. They confirm the previous summarised general assertions, but found three times more lincRNAs. In addition to a GO analysis of close mRNAs and mRNAs with correlating expression patterns for functional elucidation of the lncRNAs, a sequence similarity analysis was performed by using the BLAT Algorithm (UCSC). For lncRNA, which matches an mRNA or its flanking region of 1kb, regulation of that gene is assumed. This allowed to create regulatory networks of lncRNAs and mRNAs for each differentiation step, e.g. DN→DP. These depicted genes are controlled by many lncRNAs, while lncRNAs may have several target genes (see Fig.2.5).

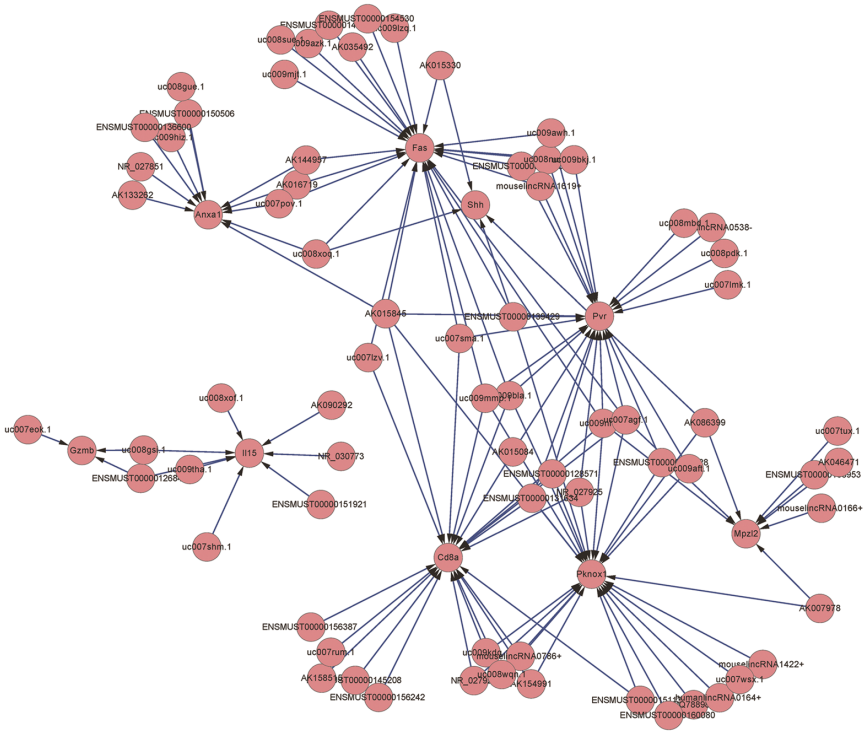

Figure 2.5: **Exemplary regulatory network of lncRNAs within T cells.** Taken from Xia et al. [XDY+14]. The authors of this study assume differentially expressed lncRNAs, which matches a correlated mRNA or its flanking region of 1kb, regulate this gene. This regulatory network depicts the relations between the lncRNAs and mRNAs differentially expressed during differentiation step DN→DP.

2.4 Secondary structure based classification

2.4.1 Secondary structure of RNAs

A secondary structure is a first approximation to model the complex problem of an RNA three-dimensional structure. It describes the folding of the sequence according to the biological background of hydrogen bonds between complementary bases of the same sequence. These bindings are results of reactions, which run spontaneously, so that energy is set free and an energetically more favourable state is reached. The biologically strongest and most probable pairs are formed by cytosine (C) and guanine (G) as well as adenine (A) and uracil (U). Additionally, to these Watson-Crick base pairs RNAs often contain G:U pairs [TUL71], which are known as wobble base pairs. In some few cases, as for instance in tRNAs, also inosine(I):A, I:C or I:U wobble base pairs have been observed. However, in tRNAs these pairs are not formed within the secondary structure, but within the binding of an mRNA [AVG07]. Hence, the inosine pairs are not considered in most bioinformatic approaches:

Definition 2.1 (RNA secondary structure [MW09])
Let $R := r_1 r_2 ... r_n$ with $r_i \in \Sigma$ be a sequence over the alphabet $\Sigma := \{A, C, G, U\}$.
Then, an RNA secondary structure is defined as a set S of pairs $(r_i, r_j) \in \{(A:U), (C:G), (G:U), (U:G), (U:A), (G:U)\}$, provided that $1 \leq i < j \leq n$ and $j - i > 4$.

The fundamental importance of base pairings causes two basic elements of an RNA secondary structure [TUL71, WWH$^+$12]: double stranded regions of base pairings, so called *Stem/Stacking regions*, and unpaired regions named *loops* (cf. Fig.2.6 A). Loops are diverse. One immediate consequence of 'self-binding' is the existence of an unpaired subsequence located at the end of stem region, where sequence changes direction. These so called *hairpin loops* are at least three free bases long, otherwise steric hindrances prevent the folding. *Bulges*, instead, disrupt a stem region by forming a loop at only one side of the double strand. If two bulges arise oppositely at both strands, they are denoted as *interior loops* and if they link more than two stem regions, *multi loops*.
Longer RNA sequences are able to form additional bindings between basic loops (Fig.2.6 B, top), so called pseudoknots:

Definition 2.2 (Pseudoknots [MW09])
A pseudoknot occurs in an RNA secondary structure S, if two pairs $(r_i, r_j) \in S$, $(r_k, r_l) \in S$ exist, with $i < k < j < l$.

These can be visualised by drawing the sequence as straight-lined chain with bindings as edges, while crossings of bindings correspond to pseudoknots (see Fig.2.6 B, bottom). Missing empirical data of these structures and insufficient computational solutions result in the fact that pseudoknots are ignored in most secondary structure prediction algorithms [WWH$^+$12].

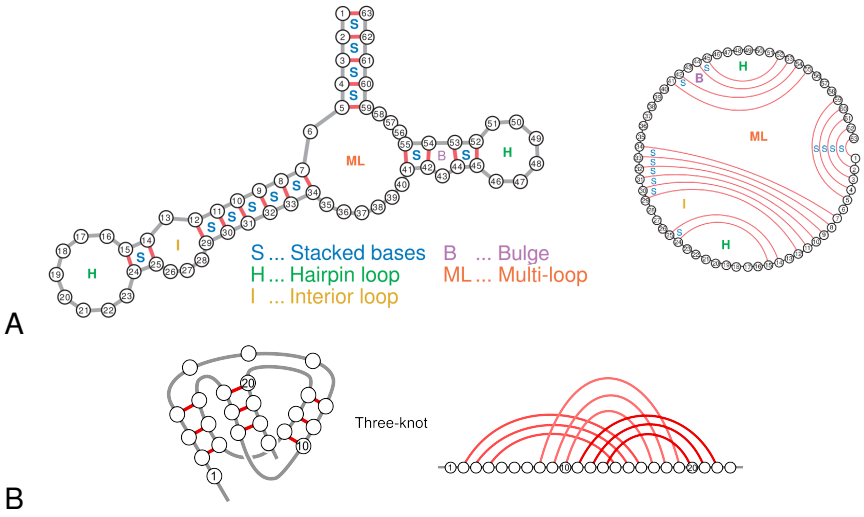

Figure 2.6: **Elements of RNA secondary structures** taken from Washietl [WWH+12]. Example (A) includes basic elements. Example (B) depicts pseudoknots.

Prediction

Predicting secondary structures of RNAs is an optimisation problem, that tries to find the most energetically favourable solution from the set of all feasible foldings, given a particular energy model. A standard solution process for bioinformatic folding problems is dynamic programming. It describes the partitioning of an optimisation problem in several similar subproblems. By combining the optimal subsolutions stepwise, one can receive the complete optimal solution in an efficient way [NPGK78, WWH+12]. This strategy mirrors quite well the biological RNA folding process, since the whole structure arise step-by-step out of single local foldings [TUL71].Thus, the optimal structure of an RNA sequence can be determined by combining the optimal structures of its subsequences. In this case the word 'optimal' basically means the most favourable energetic state, which is defined by the minimal free energy value (MFE).

First approaches, like the NUSSINOV ALGORITHM, just maximise the number of base pairs to simplify the MFE minimisation. This is due to the fact that base pairs decrease this value, while loops increase it. However, it is shown that the maximal number of base pairs does not necessarily determine biologically relevant structures, because number and size of bulges and loops as well as length of stems are not restricted. Instead, stacking interactions were postulated [TUL71]. In this way single base pairs are more unlikely, because initial base pairs promote further pairings of immediate neighbours. Furthermore, other thermodynamic conditions influence the result, as for instances temperature, pressure, the presence of other binding partners or the ion strength. Hence, Zuker and Stiegler [ZS81] published a more adequate approach, known as *Zuker algorithm*. It uses empirical thermodynamic data of single RNA molecule analysis and minimises the MFE to predict the optimal secondary structure of an single RNA sequence up to 600nt. Nevertheless in many cases the results are still far away from the biological relevant folding. On the one hand most folding algorithms ignore the existence of pseudo-

knots [WWH+12]. On the other hand RNAs are dynamic molecules, which belong to a dynamic changing system. Numerous parameters have an effect on the shape of an RNA, so that more than one structure can be formed [McC90, WWH+12]. This ensemble of possible structures exists also physically and can be determined with the McCaskill algorithm [McC90]. It calculates dynamically a matrix of the base pairing probabilities to identify all thermodynamic stable secondary structures (equilibrium partition function) for an RNA sequence. Thereby, its complexity of $\mathcal{O}(n^3)$, with n as length of the sequence, is the same like that of the folding algorithms of Nussinov and Zuker.

An efficient implementation of these algorithms is provided by the ViennaRNA package, a well known collection of several RNA secondary structures prediction and comparison tools [LBH+11]. Its tool RNAfold allows to compute both the MFE (Zuker) as well as the equilibrium partition function (McCaskill) and as an alternative option one can also choose the maximisation of base pairs (Nussinov) [HFS+94].

Anyway, the prediction of a biological relevant structure for just a single RNA sequence remains error-prone [GVR04, WWH+12], but it improves remarkably in case evolutionary related RNA sequences are taken into account. Thus, a sequence can be compared to another, whose structure is known, or if both structures are unknown, the comparison can restrict the solution space.

Comparative Prediction

In consequence of the assumption, that essential secondary structures have to be evolutionarily conserved, further information can also be taken from multi-sequence alignments. These may reveal compensatory mutations, which change the sequence while maintaining the structure [WWH+12].
In such a way RNAalifold predicts a MFE secondary structure for a multiple sequence alignment on the basis of the Zuker algorithm mentioned above. The crucial difference is the adaptation of the energy model. It incorporates information of base pair conservation and computes the energetic average over all sequences [BHW+08]. Another approach is the use of stochastic context free grammars. They combine production rules and machine learning to conclude a secondary structure of a single RNA from the knowledge of homologous RNAs [WWH+12, BB07, 330ff.]. The production rules describe, how RNA secondary structure elements derive from the primary sequence. Then, machine learning is used to support this rules with statistical probabilities determined by a training set of sequence-structure pairs. This is the basic idea of for example Pfold [KH03].

An further alternative is given by the Sankoff algorithm [San85]. Here two sequences are simultaneously aligned and folded, so that the consensus is optimised in both the alignment and the minimal free energy, despite of low sequence similarity. The original algorithm has a time requirement of $\mathcal{O}(L^{3\times N})$ for N sequences of length L, so that it is impractical for multiple comparison, but feasible for two sequences of rea-

sonable length.
Only ten years later Gorodkin et al. [GHS97] come up with a simplified derivative included in FOLDALIGN. It uses the SANKOFF ALGORITHM progressively for just two sequences. A following greedy algorithm select the best alignments and restart the Sankoff step for aligning each alignment with a new sequence or another alignment. Thus, sequences, that do not match the rest and might sophisticate the result, are discarded. FOLDALIGN is, furthermore, restricted to a maximal number of sequences per alignment and to local alignments. Thus, nothing but the most significant common pattern is presented. As a result, the time requirement is reduced to $\mathcal{O}(n^4)$ [GHS97]. The first version, additionally, even maximised the number of base pairs and neglected multibranches [GHS97], while the current version allows all pseudoknot-free substructures and adapts the scoring function by involving energetic values and substitution rates [HLSG05]. Anyway, it predicts only one recurring secondary structure pattern (motif).

2.4.2 Clustering / Unsupervised classification

In order to infer several motifs within a pool of secondary structures, clustering approaches could be used. Clustering is an unsupervised learning method that recognises patterns common in subsets of the input data [MW09, p.105].
Most suitable previous clustering tools for RNA secondary structures are based on the same Sankoff-like concept. It was introduced by Hofacker et al. [HBS04] as PMCOMP and PMMULTI and captured, for example, by the derivatives LOCARNA [WRH$^+$07] and FOLDALIGN [THG07].

Instead of searching for the MFE structure, PMCOMP directly compares two base pair probability matrices computed by the MCCASKILL ALGORITHM. Comparing whole equilibrium partition functions would go beyond the time constraints again, as it is an NP-complete problem [HBS04]. To avoid this problem, PMCOMP is limited to the search for the secondary structure of maximal 'weight', which both matrices have in common. Additionally, the process is speeded up by restricting the alignment to substructures of comparable length (differing in length less than a predefined value) [HBS04]. Thus, PMCOMP reaches a computational time of $\mathcal{O}(n^4)$.
PMMULTI allows progressive multiple alignments, which uses PMCOMP first for all-against-all pairwise comparison. Through hierarchical clustering by the weighted pair group method (WPGMA) , the closest possibility/consensus matrices are arranged stepwise. The resulting tree is finally used to guide the progressive multiple alignment.

FOLDALIGNM additionally allows, instead of McCaskill matrices, the usage of pair probability score matrices generated by pairwise runs of FOLDALIGN (global). In this way, they are able to use the SANKOFF ALGORITHM, and thus a real simultaneously aligning and folding, already at the pre-step. Since the SANKOFF ALOGRITHM is efficient only on

short sequences, first multibranch points are located to just align unbranched regions in the following. Furthermore, the matrix is reduced to only branch points and significant base pairs, while base pair scores below a given threshold are discarded. As PMCOMP, FOLDALIGNM also restricts the alignments by a maximal length difference of the subsequences and apply a progressive alignment base on a WPGMA generated guide tree. All in all, the time and memory complexity of $\mathcal{O}(n^4)$ is reached again. In addition, FOLDALIGNM provide a clustering of significant pairwise results (threshold). Here, single linkage is used to collect alignments that share a sequence step-by-step [THG07]. Although FOLDALIGNM is based in global alignments, one could use the local alignment mode of FOLDALIGN to finally cluster semi-locally. This is more sensible for prediction of secondary structure motifs since functional sites may be surrounded by flanking regions, that may disturb the prediction.

Contemporary LOCARNA was developed. It also focuses on local alignments of two probability matrices, while it is, again, only based on McCaskill's probability matrices, which are computed by RNAFOLD of the ViennaRNA package. Similar to FOLDALIGNM, LOCARNA reduces the matrix to base pairs above a probability cutoff, but goes one step further by saving only matching base pairs. It does not compute multibranch points separately and is not restricted to alignments of sequences of similar length. Instead, the dynamic programming is even more efficiently exploited by using an temporary auxiliary matrix to prevent redundant computation. Progressive semi-local multiple alignment can then be done by the version MLOCARNA, which is similar to PMMULTI. Nevertheless, PMCOMP sets the score of mismatches to zero, which lowers the base pair probabilities substantially and leads to an increasing loss of structural information, while MLOCARNA uses for mismatches the probability expected for a random occurrence of the considered base pair. In this way, even the LOCARNA package provides an efficient basis for further clustering approaches, like it is proposed by Will et al. [WRH$^+$07].

These clustering methods are still too time consuming, and thus unsuitable, for clustering of big data sets like whole transcriptomes in reasonable time. Heyne et al. [HCRB12] published the first clustering method, which captures this problem. Their approach GRAPHCLUST is a pipeline of different tools, that compare and cluster RNAs by sequence and secondary structure similarities. Thereby, it accelerates the comparison process by the use of mathematical representation for RNA structures. GRAPHCLUST is said to be applicable to 'hundreds of thousands of sequences' in linear time. Thus, it may be probable to analyse data of a transcriptome wide study to identify secondary structure motifs in and relations between lncRNAs.

2.4.3 GRAPHCLUST

GRAPHCLUST provides a new alignment-free clustering method, that involves both sequence and structure information, with the biggest advantage of linear computational

time [HCRB12]. Thus, it is the first approach, which is eminently suitable for big data. The speed-up is mostly achieved at the overall comparison process by using graph-theoretical representation of RNA structures and hashing techniques. The graph representation allows to observe inner local structure relationships by a NEIGHBOURHOOD SUBGRAPH PAIRWISE DISTANCE KERNEL (NSPDK). Here, pairs of near small subgraphs define 'inner relation motifs'. These 'motifs' are transferred into a hash-code and for each RNA structure it is counted, how often each 'motif' is present. This leads to high-dimensional vectors, that are easily comparable.
This novel idea is part of a pipeline of different well known RNA and cluster analysing tools. These reduce nearly identical sequences (BLASTCLUST [LG06]), determine abstracted representative secondary structures (RNASHAPE [GVR04]) or model, scan and refine clusters (LOCARNA, CMFINDER [YWR06], INFERNAL [NKE09]). All in all GRAPHCLUST consists of 9 steps:

Phase 1 - Preprocessing
This phase is responsible for splitting all input sequences in overlapping fragments of similar length. They are then cleaned of duplicates and almost identical replicates by the tool BLASTCLUST. The remaining fragments are passed to the next phase, where the structural analysis starts.

Phase 2 - Structure determination
Each fragment is processed by RNASHAPE, which performs an abstract shape analysis [GVR04]: Here, the complete folding space of a sequence is explored and divided into classes of characteristic motifs. In detail, all possible folding structures of natural base pairings are computed and compared to each other by using a pre-defined abstraction level. Thus, the first abstraction, for example, ignores the length of stacking regions. These abstraction levels ensure a minimum of aggregation and depend on the sequence length. Finally, two folding structures belong to the same class if they have the same shape at the same abstraction level. For each class the structure with the minimal free energy is chosen as the shape representative (shrep) of this class.
In GRAPHCLUST this abstract shape analysis is done twice. First, the fragment is analysed in its full length to detect large multiloop structures. Second, a smaller window size allows to obtain local bulges or hairpins. For each of each window the most representative shape is further processed [HCRB12].
For chosen shapes, further secondary structures feature are determined in the following phase in order to easily compare their secondary structures.

Phase 3 - Feature determination and Encoding
For that purpose the sets of representative shapes have to be transformed into graph representation of vertices and edges first. For each fragment a single labelled graph is built, which has n disconnected components namely one for each shape. The shapes by itself are connected subgraphs. Here, nucleotides are the vertices, whereas nucleotide adjacencies and bindings of base pairing are the edges. This models the basic

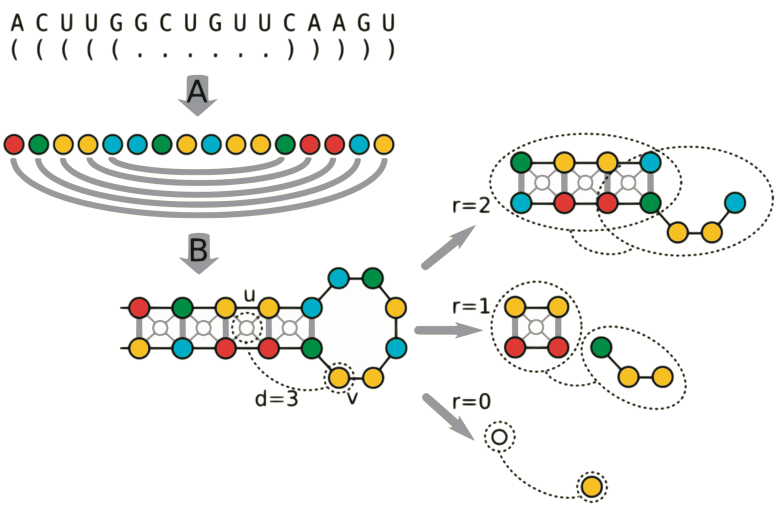

Figure 2.7: **RNA secondary structure encoding and Graph Kernel Features.** This picture, taken from Heyne et al. [HCRB12], illustrates in step (A) how a nucleotide sequence and one of its secondary structures are transferred into a simple graph representation. Vertices and its labels (here colours) symbolise nucleotides, whereas edges represent base pairings. In GRAPHCLUST additional vertices (light grey) together with 4 edge are assigned to the graphs (B). These insert stacking information of two base pairs. Furthermore, neighbourhood subgraphs up to radius $r = 3$ are shown exemplarily for vertex v on the right within dashed ovals. Graph kernel features are then determined by comparing this neighbourhood subgraphs of vertex v to those of others vertices, which are within a pre-defined distance d, like for example vertex u.

elements of biological RNA folding. Furthermore, each pair of adjacent base pairs, so called quadruplet, receives an additional connection. It is made by one vertex, that has one edge to each of the four involved nucleotide vertices (Fig.2.7.B). The quadruplet connection should emphasise the important role of stacking regions.
With these sparse graphs the actual feature determination can be done. Here, a graph kernel is used. Graph kernels are inner product functions, which are used 'to capture the long range relationships between data points induced by the local structure of the graph' [KL02]. So one can get an independent feature set for each data point as well as for the entire graph. These feature sets can then be used to compare graphs.

GRAPHCLUST applies the NEIGHBOURHOOD SUBGRAPH PAIRWISE DISTANCE KERNEL (NSPDK) published by [CG03] for each fragment graph. In this kernel two vertices are compared by their neighbourhood subgraphs of radius $\leq r$ (Fig.2.7, right). A neighbourhood subgraph of a vertex v with radius r is denoted as $(N_r)^v$. By definition it is rooted in v and includes all vertices, which are connected to v by a shortest path (minimal number of edges) $\leq r$, and all induced edges. For two vertices the comparison is made for all $(N_r)^v$ of radius $r \leq r*$. Each pair of such small neighbourhood subgraphs defines a feature of the vertex v.
Such a complete pairwise vertex comparison is computationally not feasible for data space of huge graphs. Hence, the comparisons are restricted to vertices with a dis-

tance $d \leq d*$ (Fig.2.7, lower left). Thus, the number of features and their information content can be adjusted by the upper boundary values $d*$ and $r*$.

The basic principle of this subgraph comparison is an isomorphism test, meaning that a structure- and label-preserving, bijective mapping has to be found, which transforms two graphs into each other. At graph representation level this is still a gigantic computational effort. Hence, NSPDK maps the features into an integer code, 'such that two isomorphic graphs can be reduced to an identical string' [HCRB12]. The encoding is made by three relabelling functions (Fig.3.7):
The first one $\mathscr{L}^v$ renames the vertices. Here, for each vertex v of a rooted neighbourhood subgraph a sorted list is created, which contains for all other vertices u of the subgraph a double label. Such a double label is made up of $D(v,u)$, the distance between root v and node u, and the original nucleotide label $L(u)$. The list is then lexicographically sorted by the double labels $\langle D(v,u), L(u)\rangle$. Furthermore, the double label $\langle D(v,root), ROOT\rangle$ is added at the first position of the list. Finally the list is read as a string, which is then transformed into an integer code by a hash-function.
The second relabelling function $\mathscr{L}^e$ uses the new vertex labelling to change the edge labels to triplet labels. These are of the sorted hash-codes of the two nodes followed by the original label of the edge $\langle \mathscr{L}^v(u), \mathscr{L}^v(v), L(uv)\rangle$.
The last relabelling function $\mathscr{L}^g(G)$ sorts all edge labels of the rooted subgraph G and encodes this string by an additional hashing into another integer code [CG03, suppl.].

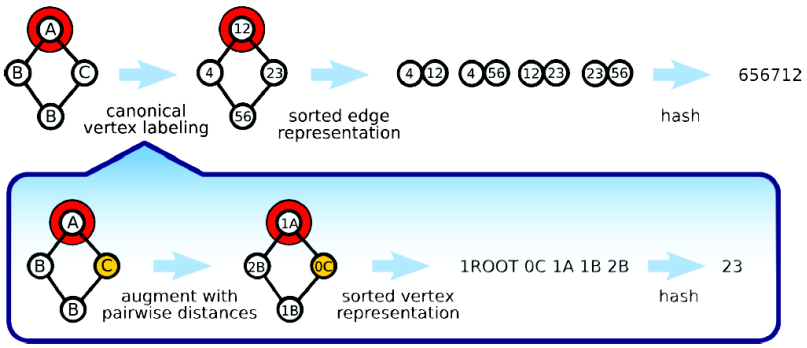

Figure 2.8: **Graph Kernel Feature encoding.** An encoding of neighbouring subgraphs into integer codes (hashing) simpifies and precipitate the comparison. Differend functions are used to compute such hash codes. 1. Each node of of a subgraph is relabelled by a first hash code, which is determined from the ordered list of the distances to each node of that subgraph plus distance to the root node (bottom). 2. Edges are labelled by the hash codes of the vertexes they connect and are sorted in increasing order (top, second from right). 3. Another hash function determines the subgraphs hash code (top, right). Taken from Costa [Cos11].

All in all the NSPDK calculates the such a hash code for each subgraphs und summarises for each of them, for all radii $\leq r*$ and all distances $\leq d*$, the amount of identical neighbouring subgraphs. [CG03]. By dividing each count by the number of

overall counts, one receives equally weighted features. These are collected in a high-dimensional sparse vector $\in \mathscr{R}^m$, with m as the number of the features. Such a vector is then used to compare the fragments secondary structures in linear time.

Phase 4 - Candidate clusters

The fragments comparison is finally computed by an approximate nearest neighbour search, which clusters the resulting sparse vectors into sequences with identical features. But according to the high-dimensional feature space a pairwise comparison in linear time is still too time consuming for large datasets, because the number of features is exponential [HCRB12]. That is why the min-hash technique is used to reduce the number of features and to reach a sublinear computational time in this phase.
The min-hash technique approximates the Jaccard index $J = \frac{|A \cap B|}{|A \cup B|}$, an exact similarity measure. However, for this the sparse vectors have to be transformed into the binary system: $\{0,1\}^m$, with 1 for non-null features. Additionally, the vector position i is assigned to each vector element $v_i \in v = (v_1, ..., v_m)^T$ as hash key, for example $h(v_1) = 1, h(v_2) = 2, ...$. After re-arranging the vector elements randomly, the hash key of the first binary 1 ($h_{min}(v)$) of each vector v is returned. Vectors with the same h_{min} value are assumed to be similar: $u, v \in 0, 1^m$. Since min-hashing is locality sensitive it is likelier to map closer objects onto the same cluster than distinct objects. An example, which summarise this method, is shown in Fig.2.9. To improve the similarity

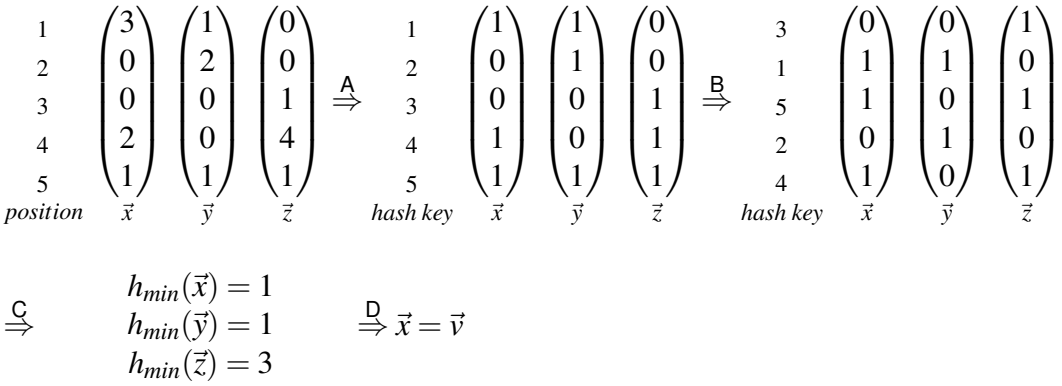

Figure 2.9: **Example of min-hashing.** This example depicts the min-hashing technique, which is used to compute a approximated similarity measure of secondary feature vectors. Therefore, the vector entries are mapped to zero if the feature does not occure in the vector, otherwise it is mapped to one. Furthermore, the position number is assigned as hash key (A). In the following the vector entries are re-arranged by chance (B). The first hash key, whose value is one, (h_min) is returned for each vector (C). Similar h_min hash codes of different vectors hint at the similarity of these vectors (D). These procedure is applied several times, while the average is an approximation of the complete comparison (not shown here).

estimation the re-arranging step is done with a pre-defined number of different permutation functions. The average of these single similarity estimators results in sufficient similarity estimator for each pair of vectors. GRAPHCLUST follows a different strategy. It collects for each vector v and each permutation function all similar vectors as approxi-

mate neighbourhood. Then the most frequent neighbours are sorted according to their NSPDK similarity to v. The k-closest of this are the k-neighbourhood. The average pairwise similarity of a vector v to its k neighbours, so called density, is used to choose the candidate clusters. These are at least the most compact neighbourhoods of a randomly chosen subset, whose overlap is beneath a specified threshold. The choice by chance is necessary here for a further speed-up.

Phase 5 to 8 - Cluster refinement, model, scanning and Iteration
A local sequence-structure Alignment with the tool LocARNA is applied to each cluster to incorporate domain specific information and improve the quality of the cluster model. By using an UPGMA created cluster tree the subcluster with the best average pairwise alignment score and at least three sequences is used to build the model. But before modelling an additional alignment with LocARNA-P is applied. This probabilistic version of LocARNA offers local and global reliability scores. Thus, sequence length will be refined to trusted seed regions. These regions are finally used to determine a covariance model (CM) by the tool Infernal. Such a covariance model of RNAs is ”based on pairwise covariation in multiple alignments”, so that it ”clearly describes both the secondary structure and the primary sequence consensus”, like first remarked by [ED94]. Furthermore, it is qualified for searching databases. Hence, the CM of a cluster is used to detect cluster members in the entire dataset. After all clustered sequences are removed from the dataset and the clustering pipeline is started again at phase 4. The repetition is stopped if a iteration or time limit is reached or all sequences are clustered. Since, several clusters can finally contain fragments of the same input sequences and since these may overlap, a post-processing step is performed at the end of GraphClust.

Phase 9 - Post-processing
Lastly strong overlapping clusters are merged. This depends on the amount of identical input sequence IDs and the size of the fragments overlap. For example, two clusters are merged, if they have more than 60% of the sequence IDs in common, with a fragment overlap of more than 50%. If a sequence belongs to different clusters at the end, the CM bit score decides on the final cluster assignment.

3 Framework for efficient biological interpretation of functional RNA motifs

Due to its linear computing time, GRAPHCLUST is currently the most suitable clustering tool allowing secondary structure analysis of whole transcriptome data. However, some of its pre- and post-processing steps as well as large memory requirements are not suitable to process large datasets. For these reasons, GRAPHCLUST was adapted in some points first to ensure an error-free usage in feasible time and under the given condition of large datasets. Afterwards an additional post-processing method was developed to improve interpretability of the results.

3.1 Methods

The clustering process of GRAPHCLUST is affected by three major hindrances in case of large datasets: First, the pre-processing tool BLASTCLUST is highly time-intensive. Second, the data structure of feature vectors is highly memory-intensive. And third, the parallel processing parameters have to be adjusted at both steps. More details and corresponding adaptations are described in the following together with the complete RNA clustering pipeline, that is finally available.

3.1.1 Preprocessing - Detecting sequence similarity in transcriptome data

One essential step in pre-processing is to filter sequences with high sequence similarity to reduce the number of sequence-structure alignments and thus decrease computational time and memory requirements.
To discard redundancies on sequence level GRAPHCLUST uses BLASTCLUST [AMS$^+$97] in its initial phase, which clusters sequences by pairwise all-against-all sequences alignments. Pairs of high sequence similarity are connected in a similarity graph to finally detect clusters of linked sequences (single-linkage).
The underlying alignment algorithm is BLAST [AMS$^+$97] in case of amino acid sequences, respectively MEGABLAST [ZSWM00], a greedy version of BLAST, in case of (long, slightly different) nucleotide sequences. BLAST is a tool for scanning databases. It first extracts all substrings of length k (k-*mers*/short word filter) of a query sequence to filter then matching k-*mers* (hits) from the database sequences. If two hits occur in the query sequence as well as in a database sequences in the same order (same diagonal in the alignment matrix) within a predefined distance, they are merged to a high-scoring segment pair. These pairs are used to rank the database sequences to finally align the

database sequence of the highest score against the query sequence [MW09, ZB08]. The BLASTCLUST standalone version uses the input FASTA file to generate such a database and allows in this way an all-against-all comparison. BLASTCLUST has to be called iteratively, because the clustering procedure assesses similarity just for the best-scoring alignment for each input sequence. Hence, high-scoring alignments with slightly smaller scores than the best-scoring alignment are not included in the neighbour graph used for subsequent single-linkage clustering. Restricting the neighbour graph to only the best-scoring alignment for each in input sequence results in a sparse graph missing many less-optimal alignments. Hence, especially for large datasets, this feature of BLASTCLUST may result in many false negatives.

Nevertheless BLASTCLUST is known to be highly time consuming in case of large input data [LG06, ABM+12], because it is based on all-against-all comparison. Even if multi-core processing (threads) is used, BLASTCLUST needs over 6 hours for a test dataset of 5,000 sequences of an average length of 138nt (see Tab. 3.1). Hence, BLASTCLUST is not suitable for large datasets. In addition, the integrated alignment algorithm BLAST expects only one input database, but in BLASTCLUST large datasets are by default split into several databases by FORMATDB, the software used to build BLAST databases. As a result, it was decided to replace BLASTCLUST by a more recent and faster method.
As an alternative tool, CD-HIT-EST [LG06] was chosen. It is less restrictive than BLASTCLUST and thus allows the processing of large datasets in reasonable time [LG06]. This is achieved by reducing the number of pairwise comparisons. CD-HIT-EST, first sorts input sequences by length and aligns them top down with a short word filter. In detail, the longest sequence is chosen as the representative of the very first cluster. The second sequence is aligned to this first representative. If it matches well, it is assigned to the first cluster, otherwise it becomes the representative of a new cluster. All following sequences are aligned stepwise against the representatives, while they are assigned to the first matching cluster (by default) or optional to the best matching cluster found. Since both variants reduce the dataset, but the latter is slower and time saving is the most crucial factor, CD-HIT-EST is applied by default. Furthermore, in contrast to BLASTCLUST, CD-HIT-EST is not used iteratively, as the sequences/representatives always remain sorted and thus no improvement of clusters is expected.

To test the performance of CD-HIT-EST compared to BLASTCLUST, four different scenarios are feasible: (i) BLASTCLUST is used non-iteratively, (ii) BLASTCLUST is used iteratively, (iii) CD-HIT-EST is used non-iteratively or (iv) CD-HIT-EST is used in combination with BLASTCLUST as post-processing (see Tab.3.1). As discussed above, an iterative use of BLASTCLUST is not reasonable.
For testing the non-iterative versions, BLASTCLUST was turned off in the GRAPHCLUST process by setting the configuration file parameter `input_blastclust` to 0 and stop GRAPHCLUST at the end of the stage 1 with the command-line argument `-stage-end 1`. In this step all input sequences are fragmented and the fragments are written into a FASTA file. This FASTA file was used as input for both tools. Their performance was

assessed by computation time and the number of remaining fragments (Tab.3.1). The iterative way of using BLASTCLUST (ii) is the default implementation in GRAPHCLUST (`input_blastclust` 1). For the combination of both tools (iv), CD-HIT-EST (iii) was added in the source code of GRAPHCLUST prior to BLASTCLUST (ii). GRAPHCLUST was then started in the same way as before.
The results show, that BLASTCLUST (ii) is more sensible in detecting sequence clusters than CD-HIT-EST (iii), while CD-HIT-EST (iii) is, in case of large input files, more sensitive than BLASTCLUST (i). This confirms, that BLASTCLUST has to be used iteratively. However, it is also apparent, that the iterative BLASTCLUST (ii) shows the longest runtime (cf. Tab.3.1). CD-HIT-EST (iii), instead, is faster no matter of file input size (cf. Tab.3.1). In contrast to CD-HIT-EST (iii), the minor improvements, if any, achieved by an additional BLASTCLUST run (iv) are associated with a drastically increased runtime (cf. Tab. 3.1). Hence, combining CD-HIT-EST and BLASTCLUST is not feasible for large datasets. Thus, scenario (iii) was chosen.

	GRAPHCLUST input	**test input** (fragments)	**BLASTCLUST 1 run** (i)	**BLASTCLUST interative** (ii)	**CD-HIT-EST 1 run (iii)**	**combination** (iv)
#seq	100	12,419	2,079	3,516	2,079	3,514
runtime			*00:00:20*	*00:00:20*	*00:00:04*	*00:00:07*
#seq	1,000	148,159	133,534	30,730	40,638	40,592
runtime			*00:04:05*	*00:14:21*	*00:00:55*	*00:02:50*
#seq	5,000	689,997	675,256	159,676	187,906	187,674
runtime			*00:48:35*	*06:10:31*	*00:14:09*	*01:08:54*
#seq	10,000	1,4 mio	—	—	372,076	371,668
					00:50:30	*4:34:45*

Table 3.1: **Performance comparison of BLASTCLUST and CD-HIT-EST.** To avoid redundant computation, GRAPHCLUST frees the input set of RNA sequences from near identical entries. The tools BLASTCLUST and CD-HIT-EST are used for this task, while four different scenarios were tested (col. 3-6) to measure runtime (hh:mm:ss, grey) and the number of remaining sequences (#seqs, black). '1 run' depicts non-iterative versions of the tested tools (col.3,5), while 'iterative' means repeated application in GRAPHCLUST (col.4) and 'combination' denotes CD-HIT-EST runs once prior to the iterative BLASTCLUST version. Since GRAPHCLUST splits input sequences (col.1) in overlapping fragments prior to this filtering, the `FASTA` file of the fragments is used as 'test input' (col.2) for BLASTCLUST and CD-HIT-EST. Both columns quote the number of sequences. '—' indicates the test failed, because BLASTCLUST cannot process large input files (see text for details).

3.1.2 Memory requirements and parallel processing

The advantage of GRAPHCLUST is its gain of time, which is a result of the encoding of sequences and their structure features. This encoding generates high-dimensional

vectors, that include information about the number of features (also of other sequences) occurring in a specific sequence. The computation of these feature vectors for all sequences causes a huge amount of additional data, which is loaded as a whole into the memory (RAM).
GRAPHCLUST produces, during the feature encoding, one file, that contains vectors of all fragments of each input sequence. As mentioned above, this file has to fit into the RAM, but it easily reaches the region of several GB for only 1000 sequences of reasonable length [HCRB12]. For example, an input `FASTA` file of 13MB (≈12,000 sequences) results in a vector file of 30GB, whereas an input `FASTA` file of 61MB (≈120,000 sequences) causes a vector file of already 148GB. Thus, a high performance system is definitely needed to analyse large-scale input datasets. If a vector file goes beyond the scope of available RAM, one possible solution is to restrict the input file. For instance, if lncRNAs in transcriptome data shall be analysed, one may use lncRNAs of intergenic regions only.

In case of large input files, the required computation time can be decreased by parallel processing on a high-performance computing cluster (HPC-cluster). GRAPHCLUST provides options to use threads (multiple core computing) and/or the SUN GRID ENGINE (SGE) to speed up the RNA clustering process. SGE is a system for distributing and managing processes on a HPC-cluster.
Since individual HPC-clusters come with individual SGE configurations, the corresponding options were tested and finally adapted. At the available HPC-system it is required to declare obligatory parameters prior to submitting a job to the cluster. This parameters are:

- `-l h_rt` the upper hard runtime boundary
- `-binding linear:1` the core binding, required in case threads are disabled,
- `-l centos6` to define the operating system,
- `-S /bin/bash` to define the shell type and furthermore
- `-l highmem` in case of large memory requirements (>59GB).

However, GRAPHCLUST needs to be started on a local resource and submits jobs on it's own. But not all obligatory parameters are declared in the source code of GRAPHCLUST. Hence, it was necessary to adapt the code. This explicitly includes all SGE job submitting scripts, but also the main script `MASTER_GraphClust.pl`. Into the latter, the code '`$qsub_opst .= "$-l highmem" if($size>59)`' was added at line 643 to dynamically insert the option for jobs, that require high memory. Thus, GRAPHCLUST can load the complete vector file into the RAM in case of large input files. Further, the SGE parameter `-l h_vmem` (hard memory limit), given in the GRAPHCLUST job submitting scripts, had to be increased stepwise for large input files. The same is true for `-l h_rt`, the upper hard runtime boundary. All in all, the GRAPHCLUST job submitting scripts are potential sources of processing errors, if the underlying computer/software system changes or the input increases.

3.1.3 Final clustering pipeline for transcriptome data

With all the adaptations described above, a ready-to-use RNA clustering pipeline for transcriptome-wide datasets is available. The final pipeline is summarised in Fig.3.1, including the generation of input files by GTF2FASTA.R (A), the new GRAPHCLUST-internal pre-processing by CD-HIT-EST (C.1) and the adaptations within the GRAPHCLUST-scripts needed for parallel processing on HPC-clusters (C.2).

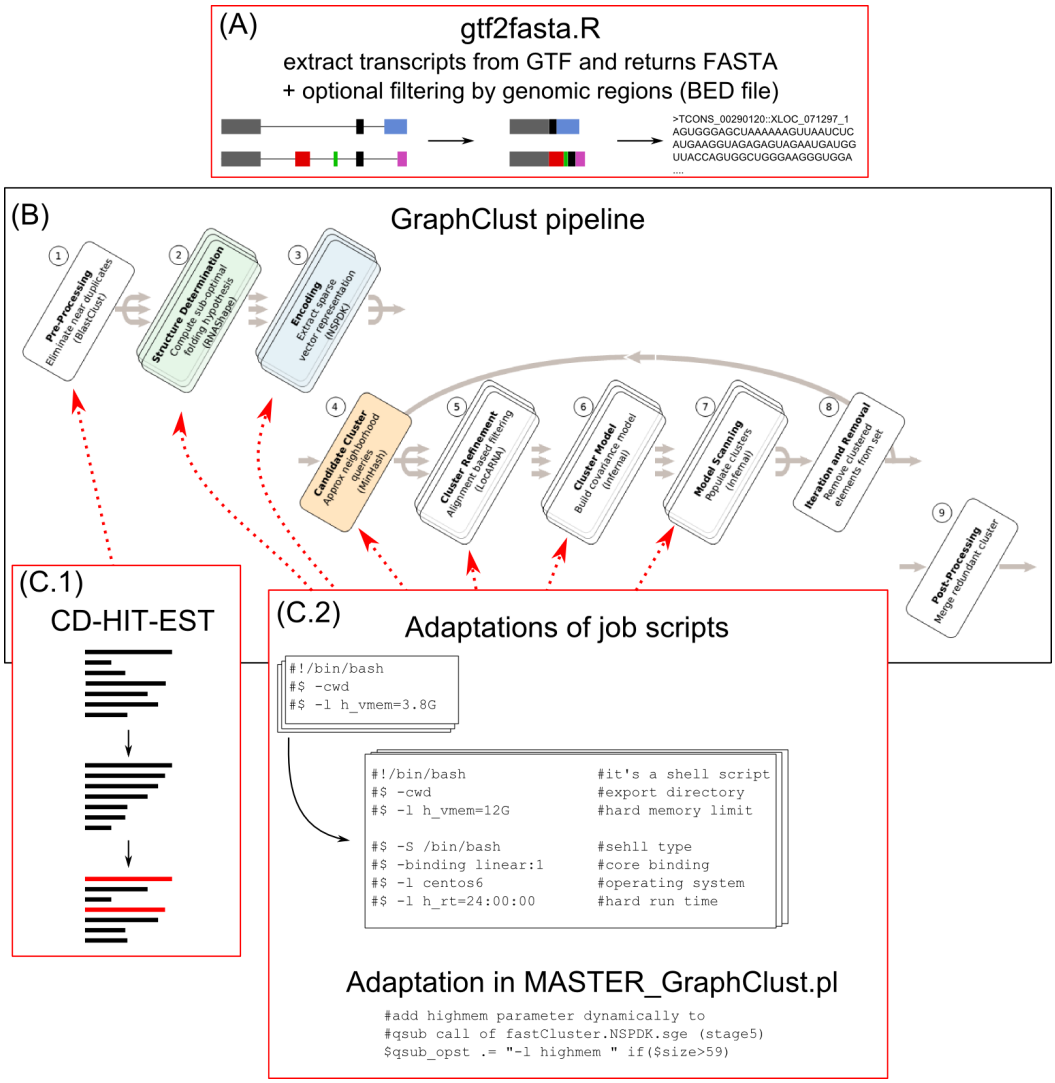

Figure 3.1: **Final clustering pipeline for transcriptome data.** (A) Pre-processing prior to GRAPHCLUST allows to generate the input FASTA file from a GTF assembly file. (B) GRAPHCLUST pipeline of 9 steps [HCRB12]. (C) Adaptations: (C.1) *CD-HIT-EST* replaces BLASTCLUST for efficiency reasons. (C.2) Example of obligatory parameters of SGE submit scripts adapted to large datasets and the specific high-performance computing cluster.

To generate input files for GRAPHCLUST, transcript annotation must be exported to FASTA files (Fig.3.1). One format for transcript annotation is, for example, the General Transfer Format (GTF). It is a widely used flat file format, that describes genomic annota-

tion by 9 tabulator-separated features: `chromosome, data source, feature type (e.g. CDS, exon or intron), start, end, score, strand (+,-), reading frame` and `further attributes`. Genes respectively transcripts and exons are characterised by separate lines. This information has to be combined to get the full transcript sequences for the subsequent clustering process by GRAPHCLUST. Alternatively, one can also use single feature types only, e.g. 3'UTRs or exons, but this would lose sequence information at splice sites and is thus not reasonable for RNA secondary structure analysis.

For generating a `FASTA` file from a `GTF` files, the script GTF2FASTA.R was developed (see Fig.3.2). It is based on R [R C14] and BIOCONDUCTOR [GCB+04], since both provide packages for handling standard file formats for transcript annotation and biological sequences.

The script GTF2FASTA.R basically reads a `GTF` file and generates a transcript database by the BIOCONDUCTOR package GENOMICFEATURES [LHP+13], while redundant information is filtered. Alternatively, GTF2FASTA.R allows the direct input of an existing transcript database to save time. Beside the obligatory input file, an optional `BED` file is excepted. `BED` files are also tab-delimited flat files describing genomic regions, but in contrast to `GTF` without gene/transcript type information. Such a file can be used to discard database transcripts overlapping genomic regions, that are defined in the `BED` file in order to restrict the final `FASTA` dataset. For reading the `BED` file the package RTRACKLAYER [LGC09] is used. After defining the locations of all final transcripts, the corresponding sequences are extracted by the package BSGENOME [Pag14] with respect to a provided genome package, e.g. BSGENOME.HSAPIENS.UCSC.HG19 [Tea14]. These sequences are finally exported into a `FASTA` file by using the package RTRACKLAYER.

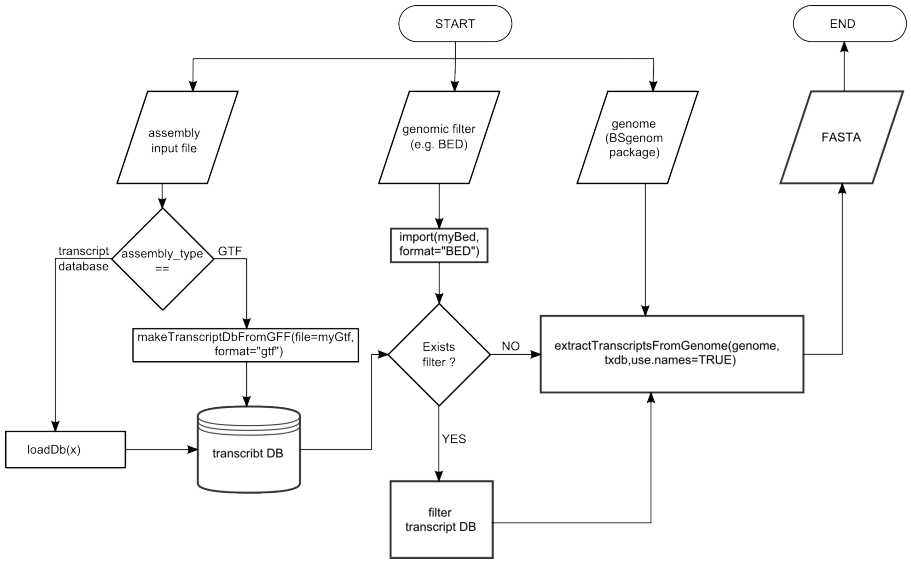

Figure 3.2: **Flowchart of GTF2FASTA.R.** This R script generates a `FASTA` file from transcripts given by an assembly file in `GTF` format or as transcript database of the GENOMICFEATURES package. An optional filter of genomic regions reduces the number of transcripts.

3.1.4 Interpretation of clustering results of transcriptome data

To reach the main goal of detecting secondary structure motifs in RNA sequences, the most important step is to finally interpret the clustering results. GRAPHCLUST provides all results as flat text files, complicating the biological interpretation of clusters. In detail, during the start of GRAPHCLUST the command-line parameter `-root` specifies the directory of output. This includes after a successful run a `RESULT` directory, which contains one directory per final cluster and the file `cluster.final.stats`, that summarises all clusters in tab-limited lines.

In the summary file each cluster is described by the number of its sequences, the number of identical sequence names as well as by two widely used similarity measures of RNA sequences: the mean pairwise identity (MPI) and the structure conservation index (SCI). While the MPI denotes similarity on the sequence level, the SCI assesses similarity on the structure level:

$$SCI(A) = \frac{MFE_{consensus}(A)}{\frac{1}{|A|}\sum_{i \in A} MFE(i)}. \tag{3.1}$$

The SCI is defined as ratio of the minimal free energy of the consensus structure $MFE_{consensus}$ of an alignment A and the average of the minimal free energies of the corresponding sequences $i \in A$, where $|A|$ denotes the number of sequences of the alignment A [WHS05]. Both values, MPI and SCI, together indicate the biologically functional relevance of a cluster.
A cluster seems to be a potential functional structure motif, if it owns a high SCI together with an MPI of about 60% – 90% [WHS05, HLSG05, RKS08]. If the MPI is larger in this context, it is not clear which conservation level is crucial, because highly similar RNA sequences likely fold into the same MFE structure. The interpretation, whether a particular cluster consists of a structural motif, is also problematic, if the MPI is less than 50%. The underlying alignment algorithms require a minimum of sequence conservation, because they perform for cost reasons a multiple sequence alignment prior to the folding, instead of aligning sequences and predicting structure simultaneously [BSG10].

All sequences of a cluster are described in detail in the text file `cluster.all`, which is located within the cluster directory of the corresponding cluster number. Such a cluster directory, furthermore, contains several other files and directories, which are vector graphics of the alignment and the secondary structure of the top five as well as the top ten sequences in `PostScript` format (generated by LOCARNA and RNAALIFOLD). In addition it contains the related covariance models and `FASTA` files. These files are supposed to be a first visualisation and starting point for analyses. But only `cluster.all` describes all sequences in detail, that belong to the cluster and thus define the structural motif (see Fig.3.3 bottom).
In the file `cluster.all` the motif sequences are labelled by an internal transcript iden-

tifier (ID), consisting of its local start and end position as well as the strand. The original name of the source transcript is saved in the columns `ORIGID` and if the name is a standardised `FASTA` header also in `ORIGHEAD`. Additionally, for each sequence it is saved during which phase it was assigned to the cluster (column 6). In this case `BLASTCLUST` denotes the sequence was filtered by sequence similarity (phase 1), `MODEL` marks an assignment through vector similarity (the cluster origin phase, phase 5), while `CMSEARCH` hints at an successful search by the covariance model (CM). Aside from this fact, for all sequences the `CM_SCORE` is computed, which denotes how well a sequence finally fits the clusters covariance model [EHH13].

(A) `cluster.final.stats`

CLUSTER	1	SEQS	33	IDS_UNIQUE	1	MODELS	1	MPI_TOP5	84.74	SCI_TOP5	0.90
CLUSTER	2	SEQS	122	IDS_UNIQUE	2	MODELS	1	MPI_TOP5	100.00	SCI_TOP5	1.00
CLUSTER	3	SEQS	68	IDS_UNIQUE	55	MODELS	1	MPI_TOP5	95.39	SCI_TOP5	1.01
CLUSTER	4	SEQS	13	IDS_UNIQUE	1	MODELS	1	MPI_TOP5	86.29	SCI_TOP5	0.68
CLUSTER	5	SEQS	18	IDS_UNIQUE	2	MODELS	1	MPI_TOP5	72.20	SCI_TOP5	0.56
CLUSTER	6	SEQS	60	IDS_UNIQUE	60	MODELS	1	MPI_TOP5	95.31	SCI_TOP5	0.90
CLUSTER	7	SEQS	237	IDS_UNIQUE	30	MODELS	1	MPI_TOP5	71.00	SCI_TOP5	0.26
CLUSTER	8	SEQS	75	IDS_UNIQUE	75	MODELS	1	MPI_TOP5	92.83	SCI_TOP5	0.92
CLUSTER	9	SEQS	27	IDS_UNIQUE	12	MODELS	1	MPI_TOP5	87.23	SCI_TOP5	0.48
CLUSTER	10	SEQS	232	IDS_UNIQUE	193	MODELS	1	MPI_TOP5	50.94	SCI_TOP5	0.25

(B) `cluster.all` of a single cluster

SEQ7628#425#526#+	RESULT	3	CM_SCORE	125.96	MODEL	1.3	ORIGID	TCONS_00076165::XLOC_019425_7628	ORIGHEAD
SEQ3346#103#204#+	RESULT	3	CM_SCORE	124.16	MODEL	1.3	ORIGID	TCONS_00218829::XLOC_052073_3346	ORIGHEAD
SEQ2706#2360#2461#+	RESULT	3	CM_SCORE	122.82	MODEL	1.3	ORIGID	TCONS_00317259::XLOC_079332_2706	ORIGHEAD
SEQ55#632#733#+	RESULT	3	CM_SCORE	120.41	MODEL	1.3	ORIGID	TCONS_00216979::XLOC_051526_55	ORIGHEAD
SEQ4663#858#959#+	RESULT	3	CM_SCORE	119.11	MODEL	1.3	ORIGID	TCONS_00315353::XLOC_078734_4663	ORIGHEAD
SEQ1938#1547#1648#+	RESULT	3	CM_SCORE	126.51	BLASTCLUST	1.3	ORIGID	TCONS_00194767::XLOC_045511_1938	ORIGHEAD
SEQ1427#1930#2031#+	RESULT	3	CM_SCORE	125.03	BLASTCLUST	1.3	ORIGID	TCONS_00231581::XLOC_055672_1427	ORIGHEAD
SEQ953#154#255#+	RESULT	3	CM_SCORE	123.58	CMSEARCH	1.3	ORIGID	TCONS_00092692::XLOC_023241_953	ORIGHEAD
SEQ7683#4295#4396#+	RESULT	3	CM_SCORE	121.55	CMSEARCH	1.3	ORIGID	TCONS_00301169::XLOC_074577_7683	ORIGHEAD
SEQ1064#823#924#+	RESULT	3	CM_SCORE	119.13	BLASTCLUST	1.3	ORIGID	TCONS_00341834::XLOC_086231_1064	ORIGHEAD

Figure 3.3: **GRAPHCLUST output.** The resulting clusters are basically described by two types of tab-limited text files, whose head is shown here. Odd columns are the headers of even columns. (A) The `cluster.final.stats` file, with `MPI_TOP5` as Mean pairwise identity of the top 5 sequences and `SCI_TOP5` as corresponding structure conservation index, summarises all clusters. (B) The `cluster.all` file depicts the sequences of a single cluster. Here, each sequence is named by the source sequence, start and end position plus strand. Column 3 quotes the cluster number, `CM_SCORE` denotes the significance of the search by the covariance model and column 4 hints at how the sequence was assigned to the cluster. For more details see main text.

Taken as a whole, the output of GRAPHCLUST is well structures and easily readable – in case of some few clusters. But, large input files may result in several thousand clusters and since the corresponding files are substantially text based, reading manually will be well-nigh impossible, not to mention interpreting these data. By the way, GRAPHCLUST provides just a low-level post-processing, with files of secondary structures and covariance models, limited to top matching sequences (top five/ten). Hence, the user is bound to this post-processing initial files and can not change them without great effort. In addition, each cluster also includes inherently much more information, that can be helpful to interpret the motif biologically, but which is currently not well presented to the GRAPHCLUST user:

- location of the motif in the transcript

- conservation on sequence and/or structure level
- number of a motif copies in a transcript
- number of different motifs in a transcript
- and many more ... .

Such details require graphical representation.

Another point is, that the true number of clusters (granularity) is unknown. Clustering approaches are learning techniques, that try to extrapolate grouping features from a given set of data points [MW09]. Since they are based on limited knowledge defining which information is used and how this is done, they are just approximations, which do not have the right balance between details and relations. Accordingly, it might be possible, that some clusters should be joined, while others can be further divided (see Fig.3.4). For these reasons also the clusters themselves should be compared.

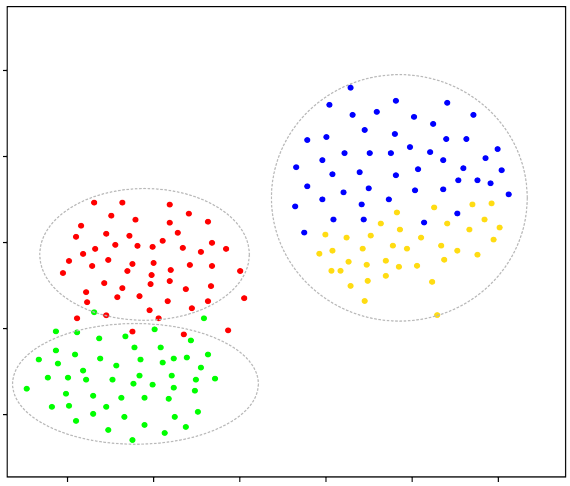

Figure 3.4: **Clustering and granularity.** The true number of clusters is often not known. Red, green, blue and yellow dots denote data points and their 'real' relations as an example. Grey, dashed circles mark a possible result of an clustering process.

One reasonable solution is the development of an additional, user-friendly post-processing tool, which can be used for an easy visualisation, filtering and further meta-analysis of the clustering results.

3.2 Results

In order to detect secondary structures motifs within RNA sequences, RNA clustering pipelines need a further post-processing step, especially if they provide just text-based output. Such a additional post-processing should visualise the data in a appropriate manner including the incorporation of biological knowledge and computation of a further meta-analysis. Thus, it should support the interpretation of the clustering result. Hence, the development of such an additional post-processing method is based on some requirements, for example adequate visualisation methods. Corresponding to these requirements the framework for efficient biological interpretation of functional RNA motifs was implemented.

3.2.1 Requirement analysis

Each user has to develop its own automatic approach to post-process the the text-based output of GRAPHCLUST. Alternatively, a general and reusable post-processing tool for RNA clusters may simplify and speed up the cluster analysis pipeline. However, such a tool has to meet some requirements, which have to be covered during the development. These requirements can be broadly divided in three crucial aspects (see Fig.3.5): increasing interpretability (I), quality control (II) and an adequate software design (III) depending on (I) and (II).

Interpretability (I)
One of the most important points in data analysis is the improvement of the interpretability, which inherently means, firstly, *visualisation* by illustrative plots and graphics. This includes, secondly, the representation of data by different *summary statistics*. Such statistics are quite informative, especially if, thirdly, *biological ancillary data* are added. For example, if novel RNA transcripts fall into the same cluster of known transcripts, knowledge about the function of the known transcripts can be projected to novel transcripts. An additional biological indispensable information is the transcripts nucleotide sequence, which is needed to display the motifs secondary structure independently from the clustering tool. Fourth, interpretability can also be increased by a *meta-analysis* of the clustering. Since the true number of clusters (granularity) is unknown, single cluster analysis do not detect relations between clusters. Comparative analyses based on cluster similarity measures (pairwise distances) may hint at a more appropriate granularity.

Quality control (II)
In order to assess whether the chosen parameters for the clustering process yield reliable results, the quality of the overall clustering is controlled. Sequences of known RNA classes (here referred as `spike-ins`) are included in the input file and assessed if they have been grouped into separate clusters. Such a quality control allows to evaluate

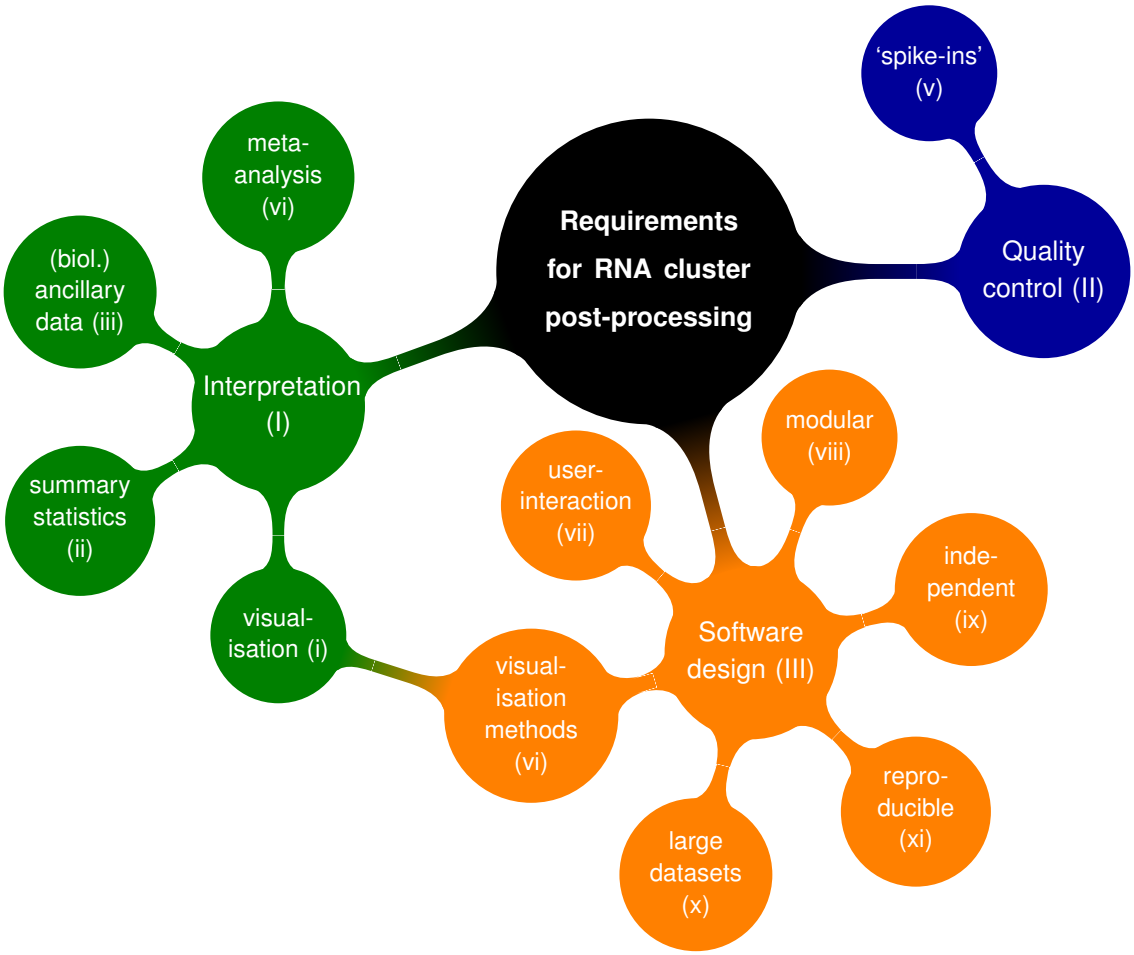

Figure 3.5: **Mind map of requirements for a RNA cluster analysis tool.**

if the overall clustering result is of reasonable granularity with respect to known RNA classes.

Software design (III)

Beside pursuing easy interpretation of clusters as well as a reasonable quality control, the third major issue is to decide on an adequate software design. According to the first requirement, of course, *straightforward visualisation methods* are required sixthly to support interpretability. In addition, meta-analysis of the clustering requires the choice of user-defined parameters (e.g. selecting a subset of clusters as input for the meta-clustering process), hence in turn the seventh requirement is *user-interaction*. To allow, furthermore, easy and fast extension of the software to new analysis concepts, eighthly, *modularity* is needed, because new questions may arise during the analysis. Ninthly, *independence* from the clustering tool used is important to increase the area of application of the proposed RNA cluster analysis software. This leads directly to requirement ten: the tool must also be able to handle *large dataset*. Last but not least, a common fact of good research is *reproducibility* [Xie14]. Hence, it is required to ensure reliability of the whole method.

3.2.2 A development environment complying with the requirements

It is widely accepted, that R is a popular framework for summary statistics *(ii)* and plotting *(i,vi)* [R C14]. R is complemented by a range of packages *(viii)*, whose number and functional diversity grow continually. By now, several packages are available for biological and bioinformatical topics, which are available as BIOCONDUCTOR packages [GCB+04]. Thus, standard file formats as `FASTA` or `GTF` files can be handled easily, while the corresponding data can be analysed and depicted *(iii)*.
Although R is inherently not suitable for large data, because data is loaded completely into the RAM [IG96], there are several packages, that provide solutions to this problem *(x)*. The package ANNOTATIONDBI [PCFL], for example, utilises the indexing of data and reduces duplicates by using the relational database concept and SQL (structured query language). The packages SNOW [TRLS13] and PARALLEL [R C14], instead, can be used to decrease time requirements by parallel computing.
Hence, by using the programming language R all requirements described above are met. This is even true for reproducibility, since the packages KNITR [Xie14] and SHINY [RI14] aim at dynamic reproducible documentation *(xi)*.

KNITR

The R package KNITR allows to create static documentations as `PDF` files or websites. These display the source code in combination with the analysis results in form of tables, plots or text. Thus, results can be published and reconstructed by other researchers. However, KNITR has some drawbacks:
Firstly, static documentations require that all computations are run at once. Hence, all processing steps and parameters have to be previously defined. However, meta-analysis, that compares single clusters, depend on the results of the summary statistics and have to be adapted accordingly. Alternatively, both steps, summary statistics and meta-analysis, can be computed separately, but this produce several files for a analysis in total and is error-prone, because the source code has to be changed manually each time. Hence, user-interaction (vii) is needed, but not provided.
Secondly, the resulting `PDF` files or web pages are easily shareable, but the method cannot be shared easily. The only solution would be to make the source code available. However, still all required R packages and third-party software have to be installed. This hinders users with low computer science background to use the method at all.
All in all, KNITR is an option, but it meets the requirements only partially.

SHINY

An alternative to KNITR is given by the R package SHINY [RI14]. SHINY is used, similar to KNITR, to generate reproducible reports, but in contrast, it is based on websites only. However, this restriction allows to utilise user-interaction *(vii)*, including file up- and downloads as well as reactive pictures. Even parameter settings for meta-computations can be defined by the user.

So called 'interactive web applications (apps)' [RI14] are created by a modular system *(viii)* of layout features, interfaces and reactive methods, so that all basics of an user-friendly web page are already implemented. By providing such an web application and all necessary software on a server the method is suitable for sharing, because the user is independent from soft- and hardware. As a result different groups may use the same analysis process and do not need to develop a project-related solution. Hence, SHINY offers a good basis for an independent analysis tool of RNA secondary-structure clusters.

3.2.3 Independent standard input file format for RNA clusters

The last two requirements are: independence from the clustering tool (ix) and a quality control of the overall clustering by infiltrating and observe known RNA classes, so called 'spike-ins' (v). Both depend on the file format and the data structure, which are used to import the cluster data into the final analysis tool. To achieve interdependency from the clustering tool used, the input file of the cluster exploration tool must contain standard cluster features (ix), like a cluster identifier (ID), the motif-presenting sequences and scores. To detect 'spike-ins', i.e. control sequences of known RNA classes, the data structure has to include an additional feature that tags selected sequences as control sequence (v).
The clustering tool GRAPHCLUST provides only the basic information and these in one text-based file per cluster and in one summary file. This is unreasonable for analysing the data as well as unhandy for uploading them onto a server. Hence, the whole cluster information should be available in one single file of a standardised format.
A, standard RNA cluster file has to combine metadata, like cluster ID and quality, as well as the essential cluster data – the motif-presenting sequences – together with the control tag. In this way, a tabular representation as often used in R is not suitable, because it would include dozen copies of the metadata and thus unnecessary inflate the file. A more adequate format is, instead, proposed by Ben-Kiki, Evans and Ingerson [BKEI04], who developed `YAML` a language for data formats based on lists.

YAML
`YAML` was known as the abbreviation of 'Yet Another Markup Language' [IEBK01], but the authors renamed it to 'YAML Ain't Markup language' [BKEI04]. The reason is, that a markup language is used to structure a document – like `HTML` structures an web page. `YAML`, instead, is a structured language for serialisation of object-oriented data.

To put it in a nutshell, object-orientation is a concept of programming, that models a complex system by classes and their instances. Classes define data structures of attributes and methods. An instance of a class is a realisation, that assign data to variables according to the data structure of a specified class [FH11, p.625f.,701]. Such instances or objects, usually exist only during the execution of the corresponding program. The

objects and their current states are not persistently save. Serialisation is a method, that transfers an object into a byte stream, together with all objects that refer to this objects, meaning the whole object graph. In this way the data are machine readable and can be written to a file on a persistent data storage device. Thereby, the object exists at least twice: once in memory and once as persistent file, while changing the former does not effect the latter. Accordingly, persistent data can also be read again (deserialisation) and reused in the same state as before [FH11, p.804f.,667] [SB05, p.429ff.].

Thus, YAML is a language, that allows to persistently save and reload object-oriented data. Further advantages are: First, YAML utilises the printable international standardised character code, called Unicode, which is both machine readable as well as human readable. Second, YAML is based on the assumption, that all data structures can be reduced to combinations of only three basic structures (see. Fig.3.6): scalars (i.e. strings and numbers), mappings ('key: value' pairs) as well as unordered sequences (e.g. lists) [BKEI04]. Hence, YAML is reasonable for sharing data between several programming languages working on these three data structures. This also includes R, where mappings are named lists and series are unnamed list or vectors. Furthermore, YAML just requires a small set of structural characters in this way.
Taken together, both facts allow to easily create and control YAML files. In addition, since lists are well parallelisable data structures in R, YAML is well suitable for storing and sharing large data, as e.g. resulting from an RNA clustering process. YAML is available for R through a package with the same name, i.e. YAML [Ste14].

sequence of scalars

```
- mean pairwise identity
- SCI
- MFE_consensus
```

mapping scalars to scalars

```
value:  92.3
min:  0
max:  100
```

mapping of sequences

```
cluster scores:
  - mean pairwise identity
  - SCI
  - MFE_consensus
sequence scores:
  - CM_SCORE
```

sequence of mappings

```
-
  score:  mean pairwise identity
  value:  84.74
-
  score:  MFE_consensus
  value:  -65.13
```

Figure 3.6: **Selected basics of YAMLs structural characters.** YAML is defined on only three basic data structures: scalars (strings, numbers), mappings (hashes/directories) and unsorted sequences (lists,arrays). Its authors assume, that all other data structures can be reduced to combinations of the three basic structures. This picture is based on [BKEI04].

A YAML based standard input file format for RNA clusters

By using the list-based YAML format a standard RNA cluster file can be defined by a list of clusters (cf. Fig.3.7). Thereby, a cluster is made up of two items: the metadata and the set of motif-presenting sequence data. Both items are mappings, whose tags map onto the corresponding data. The former is tagged by the key `META`, the latter by the key `CLUSTER`.

```
-
  META:
    CLUSTER_ID: <character>
    SCORES:
      <scalar>:  <numeric> (+)
  CLUSTER:
    TRANSCRIPT_ID: <list of N scalars>
    ASSIGN_TYPE: <list of N scalars∈{"CORE", "SEQUENCE", "STRUCTURE"}>
    IS_CONTROL: <list of N boolean values>
    MOTIF_START: <list of N numerics>
    MOTIF_END: <list of N numerics>
    SCORES:
      <scalar>:  <list of N numerics> (+)
```

Figure 3.7: **A YAML-based data structure for a standard RNA cluster file.** The standard RNA cluster file is a list of clusters, that are defined by metadata (`META`) and a motif-representing sequence set (`CLUSTER`). '-' denotes an unnamed list element. ':' represents a mapping of a key to a value (data). '<>' labels the basic data structure. '(+)' denotes, that the element occurs at least once. For further details see text.

Metadata consists of at least two entries. The first is the cluster ID, where the key `CLUSTER_ID` maps onto a string, which names the specific cluster uniquely. The second element is a set of scores, which describe the clusters biological relevance. It is named by the key `SCORES`. The scores themselves are defined by user-defined score names, that map to numerical values. This subdivision allows to use one, but also several different scores. This enables a more exact interpretation, like it was shown in the GRAPHCLUST results by the mean pairwise identity (MPI) and the secondary structure index (SCI). Furthermore, inconvenient placeholders are not needed, since score names are marked as scores and thus can be directly handled by their names.
Further metadata are possible, for example the consensus structure in dot-bracket notation, but they are not needed yet or can be easily determined from the existing cluster data.

The set of motif-presenting sequences tagged by the key `CLUSTER` is specified by six features: the transcript ID, the local start and end position (in relation to the transcript), the similarity level used to assign the motif-presenting sequence to the cluster, the control tag (requirement ix) and scores, that denote membership qualities.
To save all these information efficiently, each feature maps to a vector of length N, with N as number of cluster-specific, motif-presenting sequences. Hence, the transcript infor-

mation is distributed over different vectors, where the position number within the vectors denotes the information of a specific transcript.
The first three items are tagged by the keys `TRANSCRIPT_ID`, `MOTIF_START` and `MOTIF_END`. They allow to conclude annotation and biological interpretation, if a widely used ID type is employed. This could be for instance the *Ensembl* ID type (ENST00000421406.1). Alternatively, it was thought about the usage of the gene ID instead, but this may lose information of active isoforms, whereas the transcript ID inherently includes the gene id.
The fourth element – the similarity level – has the key `ASSIGN_TYPE`. This information denotes which sequence level was used to assign the specific motif-presenting sequence to the cluster. Three ways are possible: by similarity on sequence level, by similarity on structure level or by an another method, that depends on the clustering tool. Accordingly the values `'SEQUENCE'`, `'STRUCTURE'` and `'CORE'` are used. This information may hint at clusters of duplicates, if for example all sequences are added by sequence similarity.
The fifth entry within `CLUSTER` is tagged by the key `IS_CONTROL`. Thereby the boolean value `true` is used to denote control sequences, which were mixed into the input file for control reasons, and `false` otherwise. This is necessary in order to check clusters of known structural motifs for correct granularity.
Of course, also the motif-presenting sequences are scored to give a quality measure. Such a score list, is constructed similarly to the score list within the metadata. The only difference is, that these scores must map to a list of length N in order to associate each transcript with a value.

This basic data structure contains all necessary information about a cluster. Further data, especially a possible secondary structure, can be computed from them. Additional biological information, like for example sequences and annotation should be handled separately, because these may be needed across different clusters. For the nucleotide sequence information, the `FASTA` file initially committed to the clustering process is supposed to be used again. In this way, the user works on the same data in each step. Furthermore, it ensures, that all nucleotide sequences are available, and independence from public databases. The annotation is based on `GTF` files, that include information about gene type and transcript type. These are available from public annotation databases, like *Ensembl* or *Gencode*. This gives each user the ability to choose the database source, the organism and a version number. Thus, it is only required, that all three files, `YAML`, `FASTA` and `GTF`, contain the same transcript IDs.

Parser for the standard RNA cluster files

Transferring a file type into another requires a translator method – a so-called parser. A parser reads the input file, categorises the data to pieces and creates the new file format by reassembling the data in accordance with the specific structure [FH11, p.655].
Since each RNA clustering tool provides a specific output format, specific parsers are needed. A exemplary parser was implemented to transfer output files of the clustering

tool GRAPHCLUST to the standard RNA cluster file format mentioned above.
The parser is called by the R function `GraphClust2YAML(path)`. It requires the argument `path`, a character vector of a path name to a directory. This directory has to be specified before, during the start of GRAPHCLUST by the parameter `-root`, and has to contain a subdirectory called `RESULTS`, if the GRAPHCLUST run finished successfully. `GraphClust2YAML` tries to read file names from this `RESULTS` directory with in the given path `"path"`. It has to find both file types of the GRAPHCLUST output: a `cluster.final.stats` file as well as the `cluster.all` cluster files.
First, the summary file `cluster.final.stats` is read and transformed into a data frame, a easy manageable tabular data structure of R. This 'meta data frame' is restricted to three columns: the `CLUSTER_ID` and the scores `MPI_TOP5` as well as `SCI_TOP5`. By going through the data frame row by row, the corresponding cluster files `cluster.all` are read iteratively. Thereby, each of these cluster files is also transformed into a data frame, where the column names are adapted to the keys of the `YAML`-based standard RNA cluster structure:
The `TRANSCRIPT_ID` is extracted from the columns `ORIGID` and `ORIGHEAD`, while the `MOTIF_START` and `MOTIF_END` are extracted from the GRAPHCLUST-internal fragment ID. The `ASSIGN_TYPE` is extracted from the column including the information which similarity level was used to assign a specific sequence to the cluster. Thereby, `BLASTCLUST` is transformed to `SEQUENCE`, `CMSEARCH` to `STRUCTURE` and `MODEL` to `CORE`. To allow, then, to detect control sequences, the user has to add the suffix 'RNA' to the transcript IDs already within the inital `FASTA` file. In this way, for all sequences, whose `TRANSCRIPT_ID` includes this suffix, `IS_CONTROL` can be set to `true`, while all other are characterised by `false`. It is also allowed to use suffixes like 'miRNA', 'tRNA' or 'snoRNAcd' in order to distinguish between different classes of control RNAs. Finally the scores list is generated by extracting the `CM_SCORE` values together with this header as score name.
By summarising all required cluster information, it is possible to create a `YAML`-based list entry, that is structured in the required standard RNA cluster format. The whole list of clusters is then generated by the concatenation of all list entries. The final `YAML` code is finally created by using the R package YAML and saved within a file by the R standard function '`cat`'. The final `YAML` file is the input file of the independent RNA cluster analysis tool, described in the following.

3.2.4 Implementation

The framework for RNA cluster interpretation is finally based on SHINY web applications. This application requires an `YAML`-based input file in standard RNA cluster format, that contains the clusters resulting from an RNA clustering tool. Several summary statistics are provided to visualise the results, while potentially functional motifs can be recognised by filtering the clusters by their scores. A `GTF` and a `FASTA` file are optionally accepted to allow biological interpretation. Moreover an exemplary meta-analysis for comparing potentially functional clusters is provided by computing the pairwise dis-

tances and displaying them in a heatmap. The entire application is structured in a modular way, so that further elements, i.e. summary statistics or meta-analyses, can be easily integrated.

Basics of SHINY interactive web applications

The R package SHINY provides methods to create user-friendly, interactive web pages (applications). Since, SHINY is based on R, this package is reasonable to present results and share the methods to allow other users to analyse own data in the same way. Thus, SHINY is taken as a basis for the required framework for efficient biological interpretation of functional RNA motif clusters.

SHINY applications (app) are basically made up of two files: an user-interface script (`ui.R`) and a server script (`server.R`) (cf. Fig.3.8). The former describes the composition of the app, the latter contains the background computations required for reactive plots and other elements. Both files have to be in the same directory, because otherwise the app is not able to start. To run an app, one has to call R , load the SHINY package first by `library(shiny)` and use the command `runApp(path)`, with the path being a character vector of the full path name of the SHINY app directory [RI14].

In more detail, the user-interface script `ui.R` consists of two nested functions (cf. Fig.3.8 (A)): The outer function `shinyUI(ui)` declares an new user interface `ui` within SHINY. The inner function, the argument of `shinyUI()`, instantiate a user interface of a predefined page layout. This can be a simple page of a fluid or fixed layout or a page of more complex layout. In the former, panels (sub-windows) are located side by side, while objects are either adjustable to the browser (fluid) or not (fixed). Instead, a more complex page, like `navbarPage()`, provides a fluid layout together with a navigation menu. Such a menu allows to switch between different panels. Within this a inner function all further elements of the web application are listed and specified, for example a title or a panel by itself. These elements are again functions, that contemporary declare and instantiate the specific objects. In this way, also classical HTML structure statements are implemented. Thus, for instance, a header of level three is defined by `h3(text)` instead of the HTML tag `<h3>text<\3>`. In addition, control elements (widgets resp. window gadgets) can be placed into the page. They are already implemented to take user-inputs and thus to allow reactivity. A corresponding example is `sliderInput()`. It yields a slider, shown in Fig.3.8, that enables the input of a number of a predefined range. Since, reactivity inherently also includes output, output functions are available, too. `plotOutput()`, for example, can be called to display 'plot object' like histograms (cf. in Fig.3.8). However, additional background computations are needed to process the user input to generate and display a reactive output. Hence, the second file of a SHINY app – `server.R` – requires modifications.

The basic server script file `server.R` is again structured by two nested main functions `shinyServer(function(input,output [,session]){})`. Here, the outer function

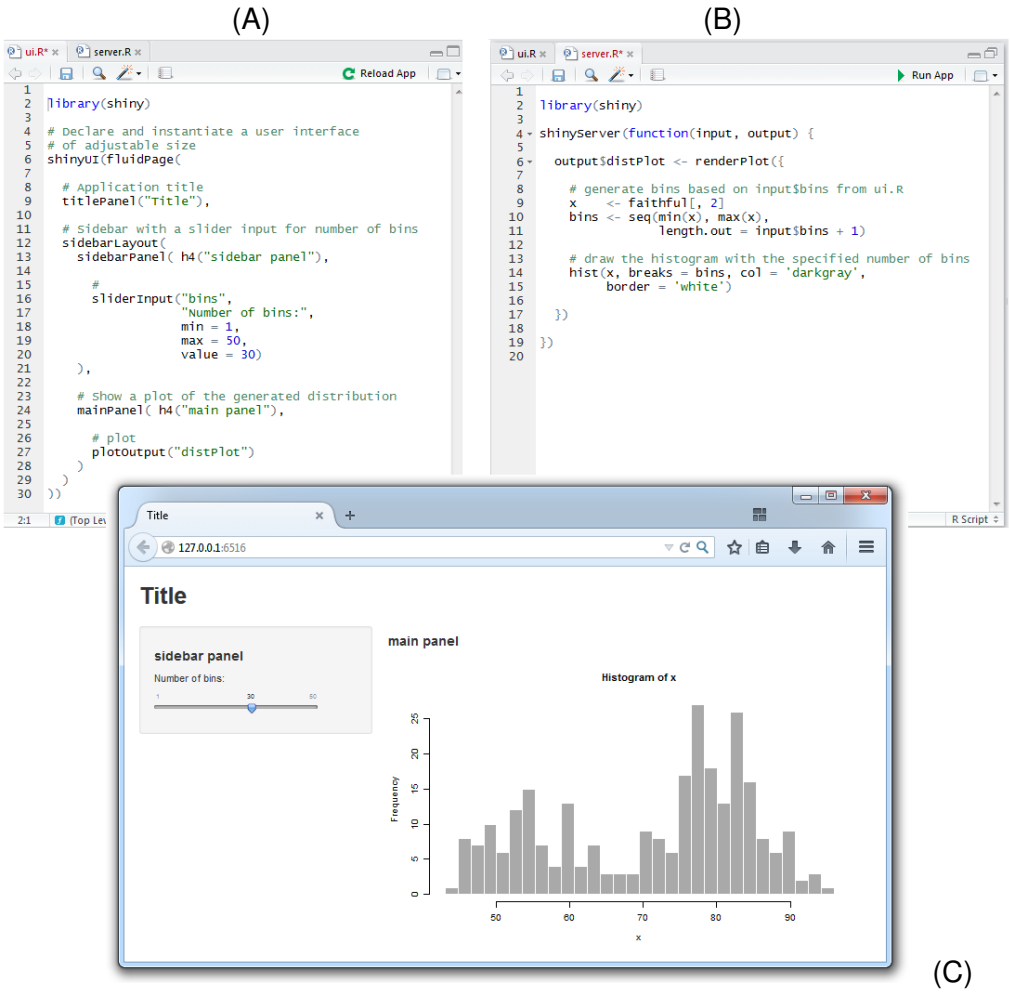

Figure 3.8: **Basis structure of SHINY applications.** This figure depicts the basic example from the RSTUDIO documentation of the SHINY package: a reactive histogram, whose bar size (number of bins) can be changed by a slider. [RI14]. (A) The user-interface script `ui.R` describes the basic composition. (B) The server script `server.R` defines the underlying computations needed for reactivity. (C) Final reactive web application.

`shinyServer(func)` realises the server functionality by processing the inner function `func`. This inner function is unnamed, but has two mandatory parameters (`input`, `output`) plus an optional one (`session`) (see Fig.3.8(B)). The parameters `input` and `output` are lists, that transfer all `ui.R` inputs elements respectively output elements to the server functions. The optional argument `session` is an environment object, that transfers client data, and thus allows multi-user services. Return values of `func` are not expected and thus ignored by `shinyServer()`. However, all further functions, that are needed to build the specific app, have to be declared within the body of the inner server function `func` (cf. Fig.3.8(B)).

There are two types of functions: non-reactive and reactive functions. Non-reactive functions need to be re-called manually, if a parameter changes, to re-compute the output. Reactive functions, instead, are notified by their reactive input parameters in case of changes, and re-execute their internal code automatically. Since, all reactive functions require reactive inputs, the parameter `input` of the inner server function (`func`)

is inherently available to all of them. Hence, required reactive input values can be observed and used directly by calling `input$<input_object_ID>`. Reactive functions, that do not produce output by itself, are called 'observers' and are used to start side-computations. All other reactive functions return their results by changing/adding an element of/to the list of output values within the parameter `output`, which is also the argument of the inner server function `func`. In this way, reactive functions can be used as links between input elements and output elements of the user-interface. For example, `renderPlot({})` (see. Fig.3.8) can create reactive plot objects. Therefore, it has to observe an input object , e.g. a slider named 'slider1'. by calling `input$slider1`. This variable can be used then during the generation of a plot, e.g. a histogram. If the value of this variable changes, the plot is re-build and a new picture is displayed in the app.

It should be noticed here, that apart from the two basic files `ui.R` and `server.R`, SHINY also accepts the additional file `global.R`. This file can be used to define non-reactive variables and functions, that should be available within the global environment. In this way they are known to all functions within the app and within all sessions, in case of multi-user service.
Following these simple structure of files and functions, a basic web application can be created quickly, while the numerous reactive elements and further options also provide more complex approaches.

Layout and functions of the RNA cluster interpretation framework
For reasons of clarity and comprehensibility, as basic layout of the RNA cluster interpretation tool a page with a navigation bar (`navbarPage()`) was chosen. This allows to define different panels (sub-windows), that can be displayed separately. Four panels were created: 'Data', 'Single Cluster', 'All Cluster' and 'Meta-analysis' (see Fig.3.9). Each of these panels is again made up of two sub-panels to optically separate input elements from output elements.

The 'Data' panel is the first panel the user can see (cf. Fig.3.9). It allows file uploads and data control. Three different file types are required to compute visualisations, analysis and interpretation of an RNA clustering result. First, a file in `YAML`-based standard RNA cluster format is needed to upload clusters determined by a RNA clustering tool. Second, a `FASTA` file is supposed to provide the corresponding RNA sequences and third a `GTF` supplies biological annotations including the information of transcript type and gene type. Corresponding to these file types three upload buttons are available. The output area is likewise organised in three panels. Here, the uploaded data are presented separately in tables or respectively as text, in case of nucleotide sequences. Each of these output panels is equipped with reactive input elements in order to allow data selection. Thus, the user can choose a specific cluster by its identifier from a drop-down menu and can restrict the number of cluster elements by using a slider. For searching sequence and annotation data a text input element expects a transcript ID, while a submit button starts the search.

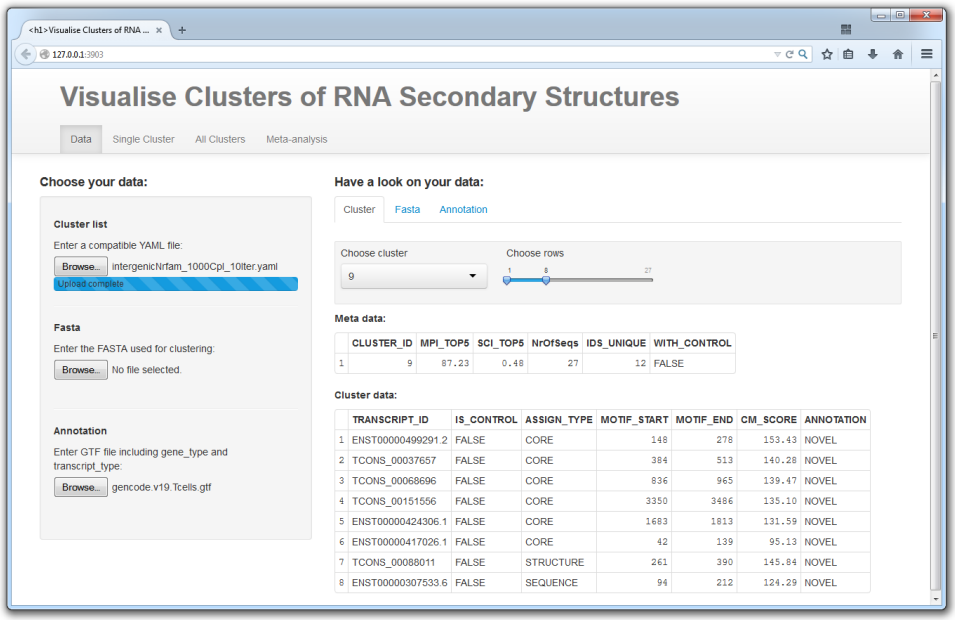

Figure 3.9: **Layout of the RNA cluster interpretation framework.** A page with navigation bar allows to use four separate panels. 'Data' allows file upload and data control. 'Single Cluster' provides summary statistics and figures of single clusters. 'All clusters' depicts all clusters by their metadata. 'Meta-analysis' allows further comparative analyses. Each main panel is again divided into two sub-panels: an input and an output panel.

The summary statistics of a single cluster are shown in the main panel 'Single Cluster' (see Fig.3.10). By activating this panel, the plots are computed for all clusters first. Thereby, a progress bar indicates the computation process. Afterwards a single cluster can be selected by a drop-down menu again, while the single pictures are chosen by activating radio buttons. The app is able to run with a `YAML`-based file only, this yields summary statistics of the raw cluster data. If an annotation (`GTF`) is available, it completes the summary statistics by including the biological information of transcript type and gene type. In addition, sequence information (`FASTA`) is used for generating a picture of the secondary structure, while the transcript length is used to draw a picture of the motif locations within the transcripts. Hence, all three files have to be uploaded, to utilise the visualisation methods in complete an thus allow increased interpretability.

The third main panel 'All Cluster' contains a single plot, that displays all clusters by their cluster scores on a two-dimensional dot plot. This enables an initial comparison and offers first indications of biologically functional motifs. Therefore two drop-down menus, one per axis, allow to select the score names used to plot the cluster data points. The size of the data points denotes the number of motif-presenting sequences in a cluster, while transparency of data points correspond to the ratio of the number of transcript IDs to the number of motif-presenting sequences within a specific cluster. Hence, big and pale coloured dots indicate clusters with many motif-presenting sequences, from which several share a transcript ID, so that single transcripts contain multiple copies of

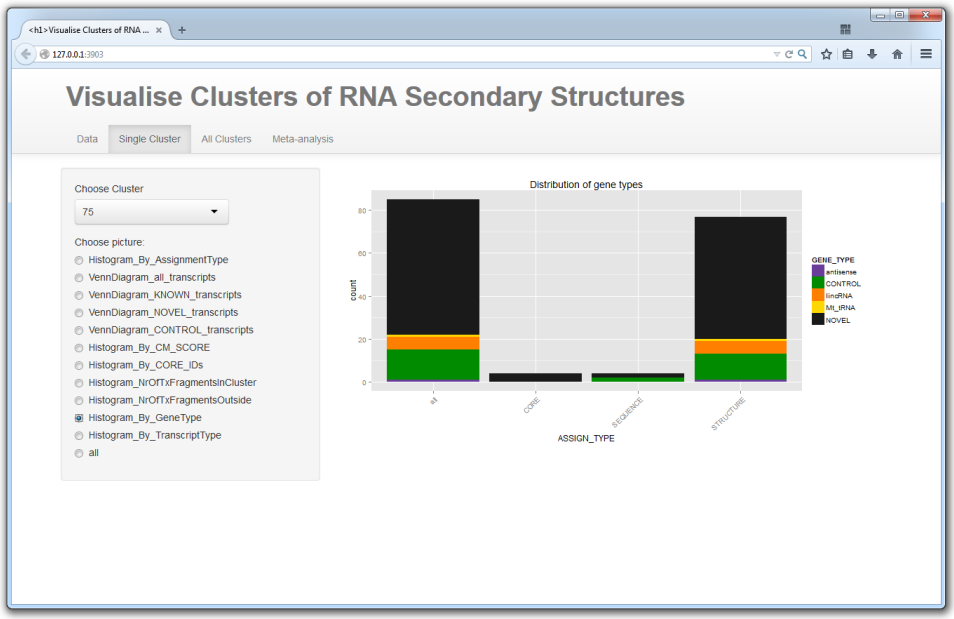

Figure 3.10: **'Single Cluster' panel of the RNA cluster interpretation framework.** Visualisation by summary statistics and further plots are available for each cluster. Thereby, the user can choose between a single plot or an overview, that entails all picturesm but of smaller size. Nevertheless biological information like gene type or transcript type is included, only if an annotation is available (`GTF`). Otherwise all transcripts are tagged as novel transcripts.

the specific motif. Small, intense coloured dots, in contrast, signalise clusters consisting of few transcripts, that contain a motif on average once. Furthermore, the dot plot highlights clusters of control sequences ('spike-ins') by colouring dots red and labelling them by the corresponding cluster IDs. Thus, the user gets a fast overview whether the expected granularity of known RNA classes, which where mixed in the initial set of sequences, was determined correctly. This knowledge can then be used for further comparative analysis (meta-analysis).

The last main panel 'Meta-analysis' provides further analysis methods. Its input box is made up of several reactive elements that allow to restrict the meta-analysis to a subset of clusters as well as a subset of sequences per cluster. This restriction is necessary for cost reasons, especially to save time. Clusters of interest can be chosen by at least one double slider, while each slider represents a cluster score and its range observed within the cluster data. The restriction of sequences is achieved by, firstly, three check boxes, which include respectively exclude sequences in/from the meta-analysis, that were assigned to the cluster by sequence similarity or structure similarity or by an another method based on the clustering tool(i). Secondly, a score is usable for shrinking the set of sequences. Therefore the user has to select one of the available scores by activating the corresponding radio button. Additionally, to select the most reliable sequences, the user has to define the direction of the score optimisation, since some scores – the minimum free energy for example – are optimised by decreasing values,

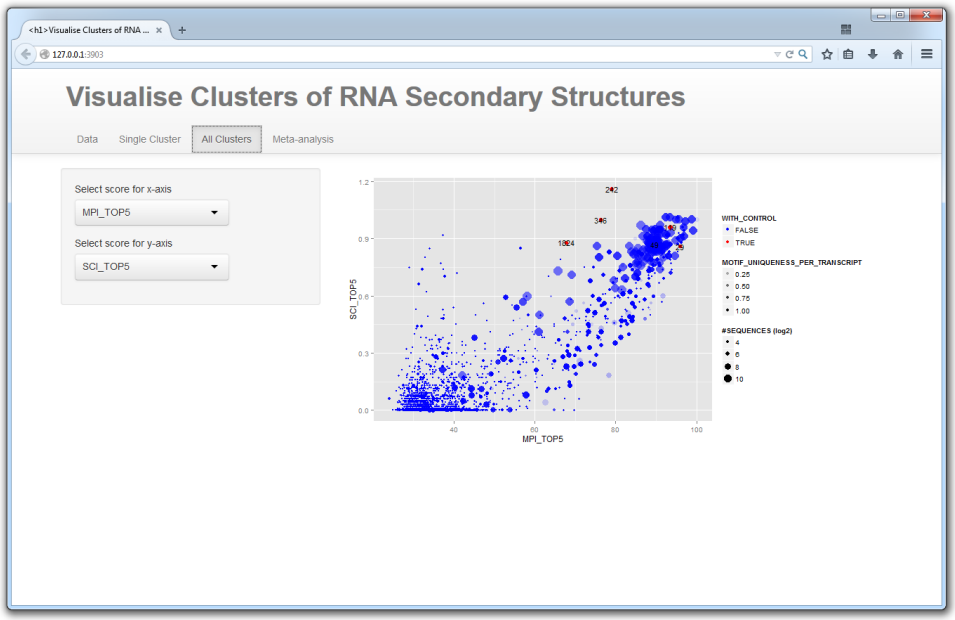

Figure 3.11: **'All Cluster' panel of the RNA cluster interpretation framework.** The plot allows an initial comparison of the clusters in accordance to their scores. Two scores can be selected on the left by their names, one for each axis.

while most other are optimised by increasing values. Finally, sequences meeting both requirements, similarity type (i) and score (ii) , can be further restricted by using exclusively a certain percentage. After restricting the data, the analysis is started by clicking the 'GO!' button. In this way a matrix of the pairwise cluster distances is computed, which can be used for further analyses. Exemplarily, a heatmap was choosen to depict the result of the meta-clustering together with a distance tree. A heatmap transfers the values of a distance matrix into a colour code. The matrix elements of the closest clusters are coloured red, while elements of increasing distance tend to blue. A distance tree supports the visualisation by fitting the distances into a hierarchical order, with the most distant cluster pair as root and the most closest as leafs.

Background computations of the RNA cluster interpretation framework

One essential point of the implementation of the RNA cluster interpretation framework is the development of input elements of reactive choices. Many panels described in the layout require input elements of selection, whose option lists (values) depend on the uploaded data. For example drop-down menus are used at the 'All Cluster' panel. They allow to choose the specific scores for the axes of clustering dot plot and thus need to include score names as options. Hence, the option lists have to be reactive by themselves, if the score names change. But, input elements integrated in the user-interface script `ui.R` do not allow reactive choices. Thus, most of the input elements have to be defined within the server script `server.R`, like all other reactive functions. However, SHINY provides methods to adapt user-interface objects within the server script: `renderUI({})` and `uiOutput()`. The function `renderUI({})` allows to create HTML objects in the server script, albeit they are usually defined in `ui.R`. The second func-

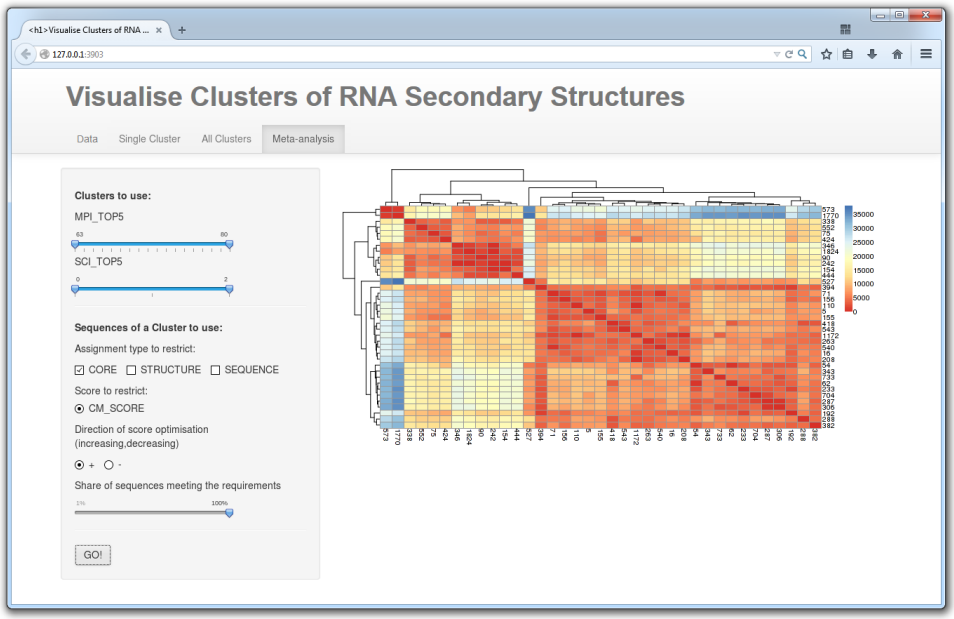

Figure 3.12: **'Meta-analysis' panel of the RNA cluster interpretation framework.** This analysis allows to compare clusters by computing pairwise distances. The user has to restrict the data set to clusters of interest and subsets of sequences (left). Finally a heatmap with a guide tree is computed to visualise the distance matrix (right).

tion `uiOutput()` defines then the user-interface object. Thus, reactive values can be extracted from the uploaded data, like for instance the score names, and can be used to rebuild the specific input element according to the new option list. In this way the application becomes more user-friendly, because it automatically adjusts itself to the data.

However, before choosing data, the upload and integration is needed. Three different file types (`YAML`, `FASTA`, `GTF`) are required by the RNA cluster interpretation framework. Although the R packages YAML [Ste14], RTRACKLAYER [LGC09] and GENOMICFEATURES [LHP+13] provide functions for reading these three file types, only the sequence information from the `FASTA` file is afterwards directly usable. The cluster data (`YAML`) and the annotation data (`GTF`), instead, need some adaptations.
The imported `YAML` object has to be checked for the standard RNA cluster data structure firstly, because the `YAML` format enables to transfer structured objects of any kind. To verify the required RNA cluster format, a set of functions was developed, that depend on each other. The function `is.validClusterList()`, for example, checks whether imported `YAML` objects are lists of valid clusters, that in addition share the same score names. Thereby, the function `is.validClusterSet()` is called to test each cluster for the two required sub-lists '`META`' and '`CLUSTER`', which describe the metadata and the motif-presenting sequences of a cluster. In accordance to the further substructure of the standard RNA cluster data structure, verification functions are available for all sub-elements, similar to an object-oriented approach. After verification of the `YAML` object, the score names are saved globally to allow the use of the scores by

their true names. Furthermore, metadata and cluster data of each cluster are transformed into data frames to make access as well as filtering more comfortable. Thereby, further columns are added. Thus, the 'META' data frame gets the entries `NrOfSeqs`, that describes the number of motif-presenting sequences within the specific cluster, `IDs_UNIQUE`, that depict the number of unique transcript IDs, and `WITH_CONTROL` to show whether the cluster entails control sequences. In the 'CLUSTER' data frame only one column is added. It is named `ANNOTATION` and contains information about the sequences annotation state. Three states are possible: 'CONTROL, NOVEL and KNOWN. The last value is set in case of an available annotation, that contains the specific transcript ID.
After all, if a valid cluster list is available, a drop-down menu, that uses the cluster IDs as options, is built at the 'Data' panel. The user is now able to select a cluster by its ID. This cluster ID is then used to extract the cluster specific data. Since 'META'- and 'CLUSTER' data are still data-frame objects, they can be easily rendered to a HTML table object and finally printed. In contrast to the 'META'-data table, which is directly printed, the 'CLUSTER data is linked to another condition. The user has, in particular, the option to select a subset of entries that should be printed by using a double slider. Hence, the data frame is supposed to react to these selections. Accordingly, a separate reactive function is needed for rendering and displaying the 'CLUSTER data frame.
The annotation data are imported from a GTF file by the R package GENOMICFEATURES. The corresponding function `makeTranscriptDbFromGFF()` reads the file and generates a SQLite database. Unfortunately, the package is designed for protein-coding sequences only, so that entries are just filtered by the feature types 'gene', 'mRNA', 'exon' and 'CDS'. Hence, some informations of non-coding transcripts may be lost during the import process. Furthermore, the ninth column 'further attributes' of the GTF is ignored completely, because these information does not belong to the standardised part of the format. This column provides data like the transcript type, the gene type or the gene symbol. However, the transcript and gene type is required to import biological knowledge to the RNA cluster interpretation tool. Thus, on the one hand the uploaded GTF has to be checked for these required elements, while on the other hand the SQLite database has to be complemented by an additional filtering for the feature type 'transcript' as well as by an additional table including gene type and transcript type. To solve all three problems at once, a new import function was implemented called `gtf2txdbWithGnTtypes()`. This function is based on the function `makeTranscriptDbFromGFF()` from the package GENOMICFEATURES. This standard function, firstly uses the package `rtracklayer` to import the data into a data frame, while rows are simultaneously filtered by the feature types mentioned above. Finally it builds the database structure. The new function `gtf2txdbWithGnTtypes()` trades on these intermediate steps. It adds the feature type 'transcript' to the filtering, which is used during the initial import of the data, so that it does not lose non-coding transcripts. If the data frame is successfully imported, it is checked for the column names 'gene_type' and 'transcript_type'. If they are missing, the computation is cancelled with an error message. Otherwise, the SQLite database is build by

default. Afterwards an additional table is added by using another new function called `addGnTtypesTable()`. These function takes a data frame as well as an existing transcript database and creates a SQLite conform table, that is added to the given database. Thereby, the data frame has to contain the columns '`transcript_id`', '`gene_type`' and '`transcript_type`', where the first becomes the foreign key, that link the additional annotation to the transcripts. However, a third function is needed, to get this information back from the database. For this reason, the function `getGnTtypes()` is provided. It takes as mandatory argument a transcript database, that includes the 'gene/transcript type' table. If this parameter is given only, the complete 'gene/transcript type' table is returned. If additionally the optional argument 'transcripts' – a vector of transcript IDs – is defined, the subset of the corresponding entries is returned, instead. Since all returned annotations are again data frame objects, the usage and the output is handy. The imported annotation information is afterwards used to adapt the current 'CLUSTER' data frames of the clusters. Thereby, the annotation state is adapted by searching the database for all transcript IDs marked as `NOVEL`, i.e. non-annotated. The annotation level is changed to `KNOWN`, if the transcript ID has been found within the database. In this case, the columns `GENE_TYPE` and `TRANSCRIPT_TYPE` are added additionally.

The initial adaptations of the input data enables finally to visualise the clusters by several summary statistics and further plots as well as the computations of the meta-analysis. Since visualisation is spread over three panels of the user-interface ('Single Cluster', 'All Cluster', 'Meta-analysis') and, furthermore, the meta-analysis depends on the results of the summary statistics, the generation of the corresponding plots and figures is computed separately for each panel.
The visualisation of single clusters is needed for individual interpretation. Hence, the majority of the pictures has to be computed at the panel 'Single Cluster'. For reasons of effectiveness, all pictures are computed at once. In this way one can take advantage of the internal list structure, which is used for the cluster data. On the one hand, it allows to apply a single function easily stepwise to each cluster, so that for all clusters finally the same pictures are available. On the other hand, R provides packages, that distribute the list entries over several compute nodes and thus allow parallel processing. Such packages are, for example, PARALLEL [R C14] and SNOW [TRLS13]. Afterwards, all pictures are available and the user can switch quickly between them. For computing the pictures the non-reactive function `visualiseCluster()` was implemented, that defines all kind of required plots and pictures by calling their specific functions. These functions are implemented separately. This allows to add further plot or pictures in a simple way, because only the single function `visualiseCluster()` has to be adapted. `visualiseCluster()` itself is called by a reactive function out of the server script, so that it is recomputed just in case of exchanged data. At the moment seven histograms and four Venn diagrams are provided as well as a motif location plot and a picture of the secondary consensus structure:

`Histogram_By_AssignmentType` — This histogram shows the distribution of the motif-presenting sequences to the similarity levels used to assign the sequences to the cluster: `CORE`, `STRUCTURE`, `SEQUENCE`. It may hint at true secondary structure clusters, if most sequences are assigned to the cluster by the secondary structure or by the 'core' method depending on the clustering tool (see Fig.A.1).

`Histogram_By_GeneType` / `Histogram_By_TranscriptType` — Both histograms are similar to the `Histogram_By_AssignmentType`. They just split the bars additionally by using the biological knowledge of gene type respectively transcript type. Thus, the user can distinguish, for example, clusters of protein-coding genes from clusters of non-coding RNAs (see Fig.A.2/A.3).

`Histogram_By_CORE_IDs` — This bar plot just focuses on motif-presenting sequences used initially to build the cluster, i.e. they were assigned to the cluster by the 'CORE' method. This set is more differentiated, so that the the sequences are grouped by the corresponding transcript IDs. In this way, one could see if the initial cluster model is based on either a single transcript or on several transcripts (see Fig.A.4).

`Histogram_By_<score>` — In this histogram the distribution of a sequence score (<score>) is depicted. Since the score name depends on the uploaded data, the histogram name is adapted, correspondingly. Furthermore, several scores are allowed, that is why a histogram is computed for each score name given for a motif-presenting sequence (see Fig.A.5).

`Histogram_NrOfTxFragmentsInCluster` — `NrOfTxFragmentsInCluster` is a abbreviation for the number of transcript fragments occurring within the specific cluster. It counts for each transcript ID the number of motif-presenting sequences, that occur in this cluster. This may hint at repeating motifs (see Fig.A.6).

`Histogram_NrOfTxFragmentsOutside` — `NrOfTxFragmentsOutside` is a abbreviation for the number of fragments, whose transcript IDs overlap with other clusters. It counts the number of motif-presenting sequences of other clusters, that share a transcript ID, which occur in the current cluster. This may hint at transcripts of several different motifs (see Fig.A.7).

`VennDiagram_<all>_transcripts` A Venn diagram is another method to display overlap between distributions. Here, the transcript IDs are assigned to the similarity level, that was used to assign a motif-presenting sequence to the cluster (`CORE`, `STRUCTURE`, `SEQUENCE`). The diagram finally depicts, whether all transcripts were assigned by a specific similarity level. Such a Venn diagram is available for all sequences of a cluster or for the subsets `CONTROL` sequences, `NOVEL` sequences or `KNOWN` sequences. Hence, <all> is replaced by the specific tag for each Venn diagram (see Fig.A.8-A.11).

`2DStructure_TOP5` This graph displays the consensus secondary structure of the motif based on the top five sequences of this cluster. The graph includes a colour code, that depicts both the structural conservation and the conservation on sequence level. Thereby, each each consensus base pair is considered, where red denotes for all aligned sequences the same base-pair at this position, yellow encodes two different base-pairs, cyan three, green four, blue five and purple all six different Watson-Crick base-pair constellations. The colours become paler if incompatible base-pairs are included [LBH$^+$11]. This means, bright red emphasises conservation on sequence level, while bright blue to purple emphasise compensatory mutations and thus secondary structure conservation. This picture provide a prompt overview of the conservation degree and arrangement (see Fig.A.13).

`Locations_Of_Motif_In_TXs` In this figure the motif-presenting sequences are mapped to their corresponding transcripts. Hence, the local positions are shown, which may point to dominant regions within the transcripts (see Fig.A.12).

All histograms are generated by the R package GGPLOT2 [Wic09], while the picture of the motif location within the transcripts is build by the package `ggbio` [YCL12], which is based on `ggplot`.
The Venn diagrams, instead, were computed by the R package VENNERABLE [Swi13]. But, additional functions were needed to catch the exception, that a set of sequences is defined by just one group. In this case, VENNERABLE builds the Venn diagram object, but throws an error if the object should be printed. Hence, the function `vennDiagrams()` was reimplemented. It re-uses the generation of the basic Venn diagram object, but afterwards this object is integrated into a new class, that checks the object for the single group error and catches this error. If a single-group Venn diagram occurs, a simple circle is drawn, similar to the VENNERABLE Venn diagrams, based on R package GRID [R

C14]. The corresponding generic functions `print()` and `plot()` are now able to plot all Venn diagrams up to a size of three groups.
The picture of the secondary structure of a motif, is computed externally (outside of R) by using LOCARNA [WRH+07] and RNAALIFOLD [BHW+08]. For that reason, the function `motif2fasta()` was implemented to extract the motifs sequences and export them to a `FASTA` file. The function `get2Dstructure()`, in turn, calls firstly `motif2fasta()`, secondly LOCARNA to generate the matrices of base pair probabilities, and thirdly RNAALIFOLD to generate the pictures. Finally, the function imports the pictures, which are in Postscript format, by using the R package GRIMPORT [Mur09].

However, the visualisation of the cluster data by summary statistics and motif pictures is still time consuming, because the user has to check all clusters. Hence, the overview plot (see Fig.3.11) is supposed to point out clusters of possibly biological functionality, so called clusters of interest. Based on this plot the user is able to consider more specifically single clusters. The overview plot is a simple dot plot of the given cluster scores, generated by `ggplot`.
Last but not least, by using the knowledge about clusters of interest, a meta-analysis allows to recognise further relationships of the clusters, so that granularity can be corrected to get a more credible result, and aims to detect structural similarities between clusters which have not been found by GRAPHCLUST. The meta-analysis is based on the tools LOCARNA and RNACLUST [Rei10]. The pipeline, again, starts with the function `motif2fasta()`, that generates a `FASTA` file for each cluster of interest corresponding to the user-defined restrictions. Each FASTA file is then used to align and fold the sequences by LOCARNA to get the consensus base pair probability matrix of each cluster. Afterwards, these probability matrices are compared pairwise by RNACLUST to get the distance matrix of the clusters of interest. The distance matrix can be used finally in different ways. A basic method is to represent the matrix by a hierarchical cluster tree or by a heatmap. Both concepts are realised at once by using the R package PHEATMAP [Kol13].

All in all the RNA cluster interpretation framework provides a web application that is able to handle standardised clustering results together with corresponding sequence and annotation data. Several functions are implemented to import, visualise and further analyse these data. Hence, a basic approach complements the clustering tools and thus allows to detect secondary structure motif within these data.

3.3 Application by using a T cell transcriptome data set

The complete pipeline of the GRAPHCLUST clustering tool and the RNA cluster interpretation tool was applied to transcriptome data of human T cells to identify functional secondary structure motifs within differentially expressed transcripts. However, before

starting the overall process, the parameters of GRAPHCLUST have to be adjusted to the data to get the most reliable result. Afterwards these resulting clusters can be easily analysed and interpreted.

3.3.1 Data

Transcriptome data

The data used for cluster analysis derives from an transcriptome analysis of human T cell samples. This analysis aims at the elucidation of the differentiation process of naive T-helper cells (CD4 single positive) to Th1-effector cells. This process was induced by activation through antibody interaction, while the antibodies αCD3 and αCD28 were used. CD3 antibodies are needed, because the T cell receptor (TCR) is associated with the CD3 receptor, which in turn is essential to transmit the activation signal into the cell. The CD28 antibodies, instead, cause a co-stimulatory signal, that is needed by the cell to recognise foreign cells and preserve auto-reactivity [Kau14].
The transcriptome analysis is applied to the naive state, the effector state and the Th1 state as well as at different time points after activation: 0h, 2h, 24h, 72h. After sequencing and assembling differential expressed transcripts [LHA14] are extracted that show a false discovery rate of less than 0.01.
The dataset was, furthermore, restricted to just intergenic transcripts, i.e. sequences, which to not overlap protein-coding regions. The final dataset consists of 9166 sequences of an average length of 1.4 kb.

Control sequences

The Rfam database [LRB$^+$12] collects information about non-coding RNAs that share similar secondary structures motifs and classifies them into families, clans and classes. Thereby, RNA families are defined as sets of well alignable, homologous sequences of the same function. Clans are RNA families, that own the same ancestor, but differ strongly either in sequence or in function. Classes instead collect families of common characteristic sequence and/or structure features, while evolutionary relation is not necessarily required [LRB$^+$12]. This classification provides a quick overview and thus elucidation of relationships and functions. While short ncRNAs can be classified well by their secondary structures, it is still open whether common sequence and/or structure motifs among lncRNAs exist. To assess the overall quality of the GraphClust output, a set of structurally well-characterised short ncRNAs classes was downloaded from the Rfam database and used as control sequences, so called ‘spike-ins’, during the clustering process. Thereby, the quality is tested by comparing the expected number of control clusters (number of used RFAM classes) to the number of the clusters computed by GRAPHCLUST.

Finally, a set of four RNA families was chosen as control data. It includes *SNORNA73* (RF00045), a family of the small nucleolar RNAs H/ACA class, *snoCD11* (RF00538)

a family of the snoRNA C/D class, *tRNA* (RF00005) and the *miRNA let-7 precursor* (RF00027). From each but one family 20 members were randomly chosen. Since, family *snoCD11* (RF00538) contains only 26 members, the whole family was used.
To recognise these RNAs as control sequences, the sequence identifiers (FASTA headers) were adapted by the suffixes 'snoRNAcd', 'snoRNAhaca', 'miRNA' and 'tRNA'.

Annotation
The annotation data of the human genome was taken from the GENCODE project [HFG+12], where release 19 (Ensembl release 73, 09.2013) was the most recent version at the beginning of this project.

3.3.2 Parameter setting of GRAPHCLUST

The RNA clustering tool GRAPHCLUST is a collection of different tools, resulting in a range of parameters (>35), which need to be defined. To verify the installation of GRAPHCLUST and to adjust the parameters in accordance to the data, that should be analysed, the intergenic T cell data was mixed with the Rfam control sequences.
Initially, GRAPHCLUST was started by default. Unfortunately, no control sequences occurred in the computed clusters. However, testing each parameter and their combinations is highly time consuming. That is why, five important parameters were selected:
The first parameter varied was the number of iterations (1), it allows to find new cluster

	parameter	description	default	variation
(1)	`GLOBAL_iterations`	number of GRAPHCLUST iterations	2	5, 10
(2)	`GLOBAL_num_clusters`	number of new cluster candidates per iteration	10	100, 1000, 2000, 5000
(3)	`nspdk_knn_center`	number of densest vectors, that initially define a cluster within the NSPDK	20	5
(4)	`OPTS_nspdk -D 3`	the distance of root nodes of features within the NSPDK	3	1, 6
(5)	`OPTS_nspdk -R 3`	radius of the subgraphs of features within the NSPDK	3	1, 4

Table 3.2: **Five important parameters of GRAPHCLUST varied during the parameter configuration.** 'parameter' – GRAPHCLUST internal name of the parameter. 'default' – default value. 'variation' – values tested additionally. NSPDK – NEIGHBOURHOOD SUBGRAPH PAIRWISE DISTANCE KERNEL

candidates and further cluster members for all clusters computed up to this point. By

changing this value from 2 to 5, the Rfam families of both snoRNAs types (RF00045, RF00538) were found in complete and grouped in well separated clusters. Members of the tRNA family (RF00005) was just found in a mixed cluster, with six missing tRNA members and additional non-control sequences, by applying 10 iterations. The miRNAs (RF00027) could not be found up to this point. For that reason `GLOBAL_num_clusters` (2) was firstly altered into 100 and 1000. Both values increased the number of tRNAs (to 18 resp. 19), but the miRNAs were still missing.
Furthermore, the combination of both parameters (1) and (2) was checked. The result remained roughly unchanged. Only the parameter setting of `GLOBAL_iterations`=10 and `GLOBAL_num_clusters`=1000 was able to find all four of the Rfam families, while all but the tRNAs were again grouped in well separated clusters. To refined the result, this parameter setting was tested in combination with `nspdk_knn_center` (3). This parameter allows to in-/decrease the number of vectors, that define a candidate cluster. Hence, lower values yield slightly specific clusters in most cases, so that more members, especially in case of tRNAs, can be found. That is why this parameter (5) was decreased from 20 to 5. As a result each of the control clusters contained several non-control sequences and the tRNA cluster was split in five single clusters. Hence, the default value yielded a more reliable granularity (true number of clusters).
An alternative is to change the parameters NSPDK distance (4) and the NSPDK radius (5), that allow to refine the feature vectors used to find candidate clusters. In case of higher values more and larger subgraphs are compared to each other. Hence, the information content is more specific. If the value is decreased, less and smaller subgraphs are compared and the information content of the feature vectors is lower. Both values were altered into the minimal and maximal values supposed by the GRAPHCLUST authors, so that all combinations of radius $r \in \{1,3,4\}$ and distance $d \in \{1,3,6\}$ were checked. For time reasons, the parameter `GLOBAL_num_clusters`=100 (2) was basically used in this test, because it was the quickest test yielded both snoRNA clusters and the tRNAs. However, decreasing the NSPDK distance (4) and the NSPDK radius (5) led to the loss of all control clusters, while by increasing the result remained unchanged again. Thus, refinements are not expected by these parameters.
To finally reduce computation time, but preserve clusters of all four Rfam families, the `GLOBAL_num_clusters` (2) was altered again to 2000 and 5000 clusters per iteration, while all other parameters are used by default. As a result, 2 iterations of each 5000 clusters yield the same result as 10 iterations of each 1000 clusters, but is above one day faster (1d 14h $\leftrightarrow$ 2d 23 h).

These results depend on three values per Rfam family: the number of clusters containing the control sequences, the number of the overall sequences of the specific clusters and the number of control sequences in these clusters. They were controlled manually for cluster, that contains sequences of an Rfam family. However, a objective measure of the overall accuracy of the results can be statistically computed by the F-measure:

$$F = 2 \cdot \frac{\frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}}{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}} = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{3.2}$$

where TP and TN denotes the true positive respectively true negative sequences, while FP and FN denotes false positive and false negative sequences. Hence, the F-measure allows to consider both the recall (true positive rate) and the precision (true negative rate) of a classification test in a combined manner [WRH+07].
Since each control cluster should just contain the sequences of a specific Rfam family, TPs are clustered control sequences of one Rfam family, while FPs are sequences of other Rfam families as well as non-control sequences. In addition, FNs are sequences of the Rfam familie, which where not found, while TN is a empty set, because no other sequences are allowed. Thus, the F-measure of the Rfam families is given by:

$$F = 2 \cdot \frac{\frac{|Rfam\_familiy_x \bigcap cluster|}{|Rfam\_familiy_x|} \cdot \frac{|Rfam\_familiy_x \bigcap cluster|}{|cluster|}}{\frac{|Rfam\_familiy_x \bigcap cluster|}{|Rfam\_familiy_x|} + \frac{|Rfam\_familiy_x \bigcap cluster|}{|cluster|}} \tag{3.3}$$

The F-measures are presented in Fig.3.13. Thereby, the results are the average F-measures of all parameter test, that use the specific parameter given at the x-axis with the specific value. The results confirm the manually result mentioned above. Hence, the most accurate parameter setting is given by `GLOBAL_iterations`=10 and `GLOBAL_num_clusters`=1000, while all other parameters are used by default.

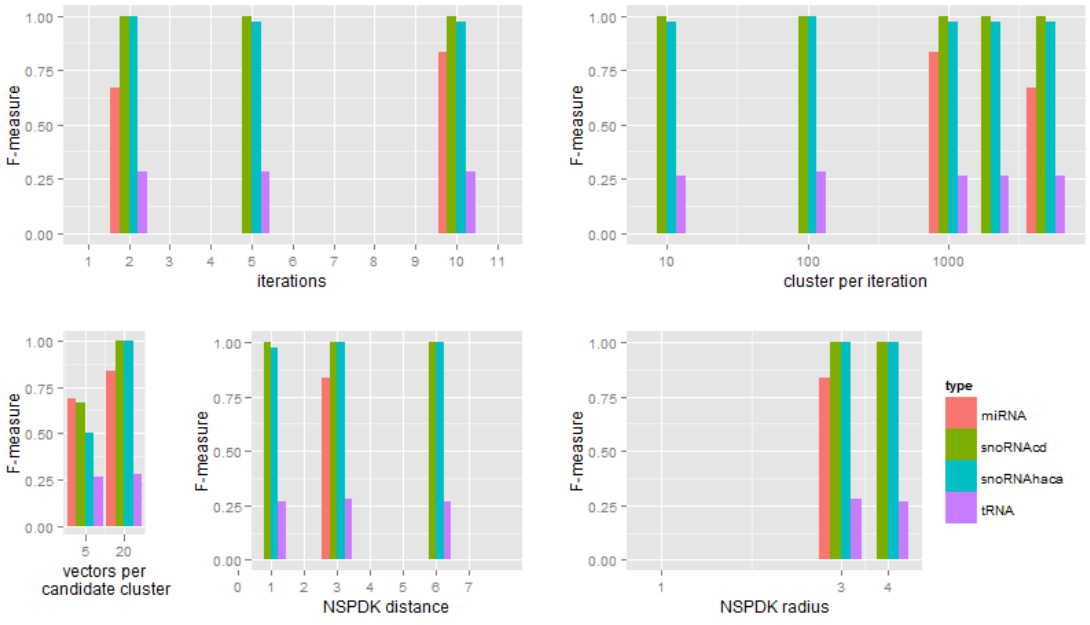

Figure 3.13: **Results of GRAPHCLUST validation with 4 Rfam RNA families.** Five important parameters were selected to set the parameters of GRAPHCLUST in accordance to the transcriptome data.

3.3.3 Characterisation of derived clusters

To finally show the application of the RNA cluster analysis tool developed during this work, the result of the most reliable parameter setting (10 iterations of 1000 cluster) should be exemplarily analysed. Thus, the dataset consists of the sequences of the four Rfam families as well as of the T cell transcripts. They are grouped GRAPHCLUST in 3225 clusters.

After transformation of the clustering output into the standard RNA cluster format, the dataset can be uploaded to the RNA cluster analysis tool. In addition, two files can be imported: the `FASTA` file used initially to run the clustering GRAPHCLUST, and the GENCODE annotation as `GTF` file.

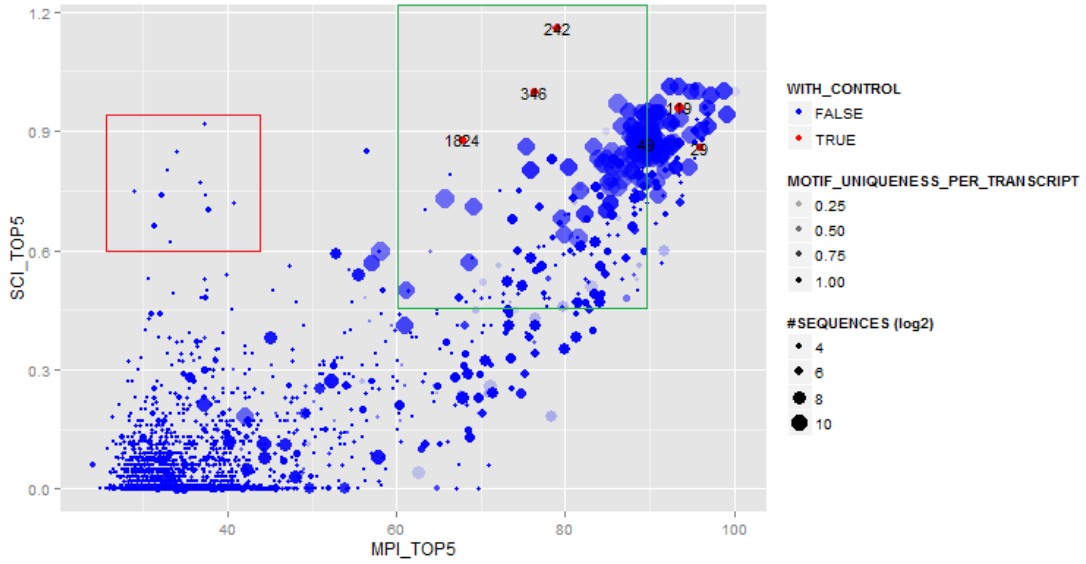

Figure 3.14: **Dot plot 'All Clusters' – Overview of an exemplary clustering result.**

To get an first overview of the clusters, the 'All Cluster' panel is chosen, which shows a dot plot of the distribution of the clusters by their scores. Since GRAPHCLUST provides exactly two cluster scores, the mean pairwise identity (MPI_TOP5) and the structure conservation index (SCI_TOP5) of the top five sequences, these scores are automatically chosen for the x- and the y-axis of the plot. The plot, shown in Fig.3.14, depicts a cloud of data points that runs from bottom left to top right. The dot size and the transparency seem to increase in the same direction. Hence, the clusters on the left consists of comparatively few motif-presenting sequences, which all belong to different transcripts, so that each transcript entails the motif just once. The clusters on the right, instead, are made up of numerous motif-presenting sequences, while on average each transcript exhibit multiple copies of this motif.
Nevertheless, for interpretation, one has to consider the cluster scores. Since structural motifs are wanted, the SCI has to be above 50%, because it assesses the similarity of RNAs on structure level. The MPI, which describes the RNA similarity on sequence

level, should be in the range of 60-90%, because a minimum of sequence similarity is needed by the underlying alignment algorithms, while the maximum should not be exceeded, because in this case it is not clear which feature is the important one. Thus, the set of potentially, biologically functional clusters can be reduced to this range. These clusters of interest are marked manually in Fig.3.14 by a green frame. However, it is easy recognisable, that the similarity on sequence level is either in a higher level or in a lower level, while in the space between only a few clusters exist.
In addition to the cluster of interest, mentioned above, a second group, marked in the dot plot by a red frame, may be of biological importance. These dots have, in particular, a very low MPI, but a relatively high SCI of above 60% und thus are quite interesting.
A further nice feature of the overview dot plot is the identification of clusters, that contain control sequences, i.e. sequences of the Rfam families. These clusters are coloured red and labelled by the cluster ID. In this way, it is easy to see that the obtained granularity of control clusters does not match the expected granularity, i.e. the number of Rfam families. By switching to the 'Data' panel, one can see that cluster 29 includes the sequences of the snoRNA C/D class, cluster 49 the members of the snoRNAs HACA class and cluster 1824 the miRNA sequences. The other three control clusters (242, 346, 119) contain tRNAs, while cluster 119 entails most of them. All control clusters show both high SCI and high MPI, and on average only one motif per transcript. Hence, the parameter setting is all right.

All of these facts can be used for the meta-analysis of the clusters, which allows to find further relationships. The restriction of the overall dataset is sensible, because unnecessary background alignments are prevented and thus computation time is reduced. The main fraction of the clusters of interest (green frame) is further restricted to clusters of a MPI between 60% and 80% and of a SCI >50%, since this is the most reliable range of possibly, biologically functional structures. Each cluster was then reduced to all 'CORE' sequences, that defined the cluster initially. The result of this meta-analysis is pictured in Fig.3.15. The heatmap in combination with a guide tree shows five reddish main areas, which denote relationships between the clusters. One can see that, for example, the clusters 346 and 242 belong to same reddish main area. This is quite interesting, because these are two of the three clusters that contain sequences of the tRNA family. The third one is not included, because its MPI is higher than 80%. However, a better trend is given by the guide tree. It signalise, that the cluster 346 is much closer to cluster 1824, which entails the miRNA precursor sequences. To consider this result, one can switch to the 'Single Cluster' and have look at the summary statistics and other plots. The comparison of both secondary structures shows that both of them are made of a long and a short hairpin loops (cf. App.B.1). This means the meta-analysis seems to find only rough similarities.The reason is, that LocARNA, the tool which is uses to compare the clusters by their base probability matrices, just performs a global alignment, so that mismatches at the sequence ends influence the result. Furthermore, their pictures of the motif location, does not show a trend of the motif location (cf. App.B.1). All in all a biological relationship can not be concluded.

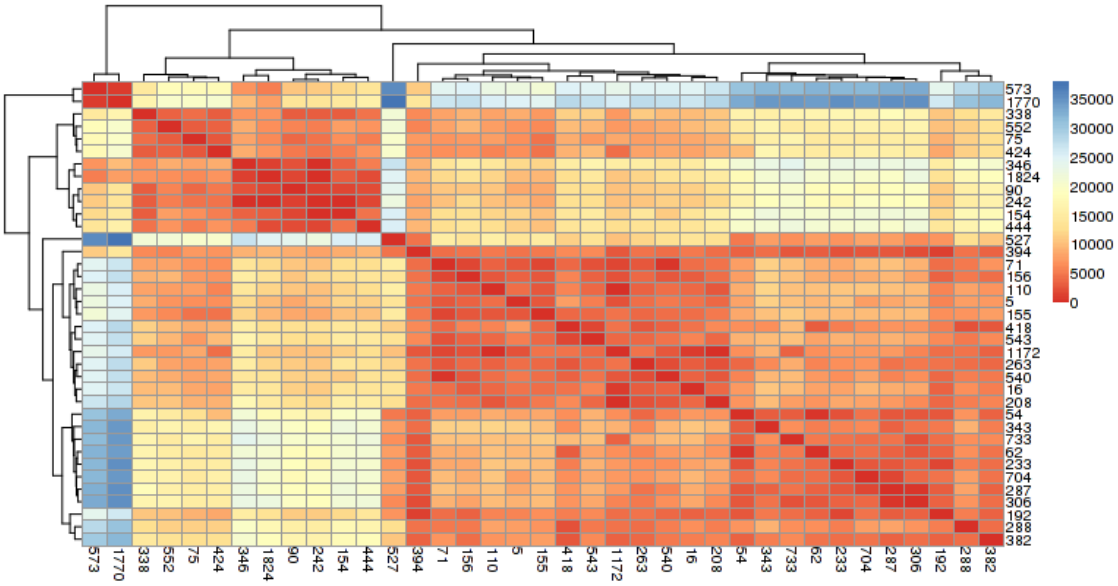

Figure 3.15: **Meta-analysis of potentially, biologially functional clusters.** The heatmap depicts five reddish main groups, while guide tree facilitates comparisons. Bright red fields denote stronger similarity, while blue fields mark quite distant clusters.

Another pair of clusters highlighted by the guide tree is build by cluster 444 and cluster 154. Both clusters show again a quite different secondary structure, which entails two hairpin loops, similar to clusters 346 and 1824, but the hairpins are less different in length. Hence, it is not surprising, that all four of these clusters occur in one of the reddish main areas of the heatmap. In contrast to the former cluster pair of tRNAs and miRNAs, the clusters 154 and 444 contain non-control sequences only.
Their summary statistic plots, that depict the distribution of the transcript types in accordance to the similarity level (used to assign a motif-presenting sequence to the cluster), show that both clusters are mainly defined by structural similarity, while lincRNAs and sense-intronic RNAs are included. By searching external databases like Ensemble, on can now search for the now transcript IDs. For example ‘ENST00000580048’ of the cluster 444 is an transcript, which is located in antisense to the zinc finger protein 407 (ZNF407). ‘ENST0000606434’ is located in antisense to the Steroid 5-alpha-reductase (SRD5A1).

To sum up, the RNA cluster interpretation tool provides a simple method to visualise and analyse a clustering result. The user can quickly filter clusters of possibly biological function and is able to apply an meta-analysis to detect further relationships between these clusters of interest. These are well represented and easy to recognise within in a heatmap and the corresponding guide tree. The single cluster plots allow then to compare such cluster pairs in detail, to search for preferences of the location of a motif or to find functional similarities by using known transcripts IDs of the clusters. Hence, it simplifies the overall result interpretation of an RNA clustering tool.

4 Discussion and Outlook

It is well known, that the human genome is transcribed nearly entirely, while many of these transcripts do not encode proteins. It is assumed, that the so called non-coding RNAs represent an additional layer of genomic regulation. However, for the majority of these transcripts the biological relevance and function still remain unclear. This is particularly true for the large group of long ncRNAs (>200nt), because they often show quite distinct sequences. Hence, functional elucidation by sequence similarity yield no result. Instead, the three-dimensional structure of RNA molecules is expected to be evolutionary conserved in case its function is essential and thus has to be preserved. Since the three-dimensional structure is complex, prediction algorithms use as approximation the secondary structure. Nevertheless, the clustering of RNAs based on their secondary structure is still time consuming respectively impossible on a large scale, like it is required for transcriptome data. Heyne et al. published with GRAPHCLUST the first RNA clustering tool, that is intended to process large datasets in reasonable time.
Anyhow, during this work, it was shown, that GRAPHCLUST has three crucial drawbacks in case of large dataset processing: 1. it uses BLASTCLUST to reduce sequence similarity a priori, albeit this is known to be highly time consuming, 2. it generates one large file of structure features, that becomes highly memory consuming, and 3. the text-based output is not manually interpretable. As a first result of this work GRAPHCLUST was adapted by replacing BLASTCLUST with CD-HIT-EST, a less sensitive, but considerably faster clustering method based on sequence similarity. The advantage was demonstrated. In contrast to the first drawback, the second is a fundamental problem, which depends on the inherent data structure of feature vectors. Thus, the user still has to restrict the dataset, if the vector file becomes to large. However, as main result, a framework for efficient biological interpretation of functional RNA motifs was introduced together with a standardised file format for RNA clustering results. These approach allows to analyse and to interpret clustering results semi-automatically and independently from the clustering tool in a visualised and interactive manner.
The standardised RNA cluster format is based on `YAML`, a human- and machine readable language for saving and sharing data in combination with their object-oriented data structures. Thus, a simple flat file format is proposed, that entails all basically required information of RNA clusters.
By using R and SHINY as basis of the framework, it was possible to realise a interactive and reproducible web application, that can be provided via the internet. However, R loads data in complete into the memory, which is quite complicate in case of large data. On the one hand the use of databases was suggested, on the other hand parallelisation of the computations, but the computation of several summary statistics and plots goes beyond the scope of memory. Hence, plots of single clusters have be restricted firstly to clusters of potentially biological relevance or have to be saved externally. Nevertheless, these plots together with the 'All Cluster' overview dot plot as well as the 'Meta-analysis'

heatmap with guide tree supply a simple, but efficient method to get an fast overview of the clustering result, to filter relevant information quickly and detect further relationships. The framework is implemented in a modular way, so that further methods can be integrated easily. For example, it is sensible to extend the meta-analysis by further methods, like computing the average silhouette width – a quality measure, that compares the cluster internal distances with the intra-cluster distances [Rou87]. In addition, a bootstrapping meta-analysis would yield more reliable results of relationships [MW09].
Beside the meta-analysis, the available biological information has to be enlarged. Knowing the transcript ID and the transcript type is a first step, but for functional elucidation further information is needed. Thus, the annotation database can be completed by adding gene names (symbols), which are not accepted yet. Alternatively, an external database can be used, for example, by using the R package BIOMART [DSBH09]. But in this case data independence is lost. Last but not least, the covariance models could be used to search the Rfam database for related, known structural RNA families, classes or clans. The tool CMCOMPARE [EHH13] provides such comparisons.

Taken all together, the framework for efficient biological interpretation of functional RNA motifs provides a basic method, which is required urgently to visualise, analyse and interpret RNA clustering results, especially in case of large data. It is a user-friendly method with high expandability, that generally complements clustering pipelines and even so annotation pipelines.

Appendix A: Plots of a single RNA cluster

Histograms

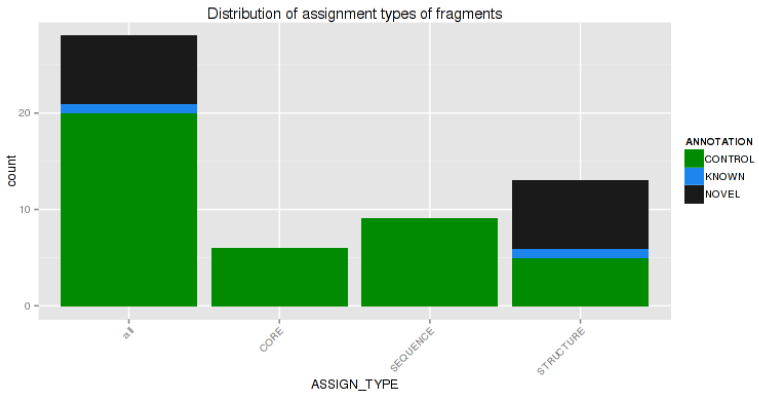

Figure A.1: `Histogram_By_AssignmentType`.

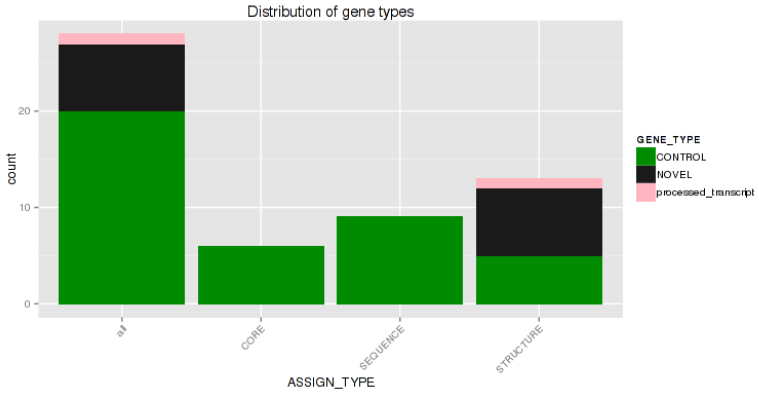

Figure A.2: `Histogram_By_GeneType`.

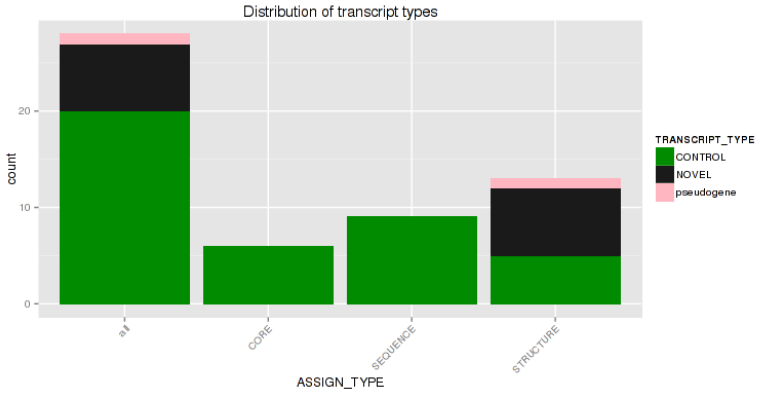

Figure A.3: `Histogram_By_TranscriptType`.

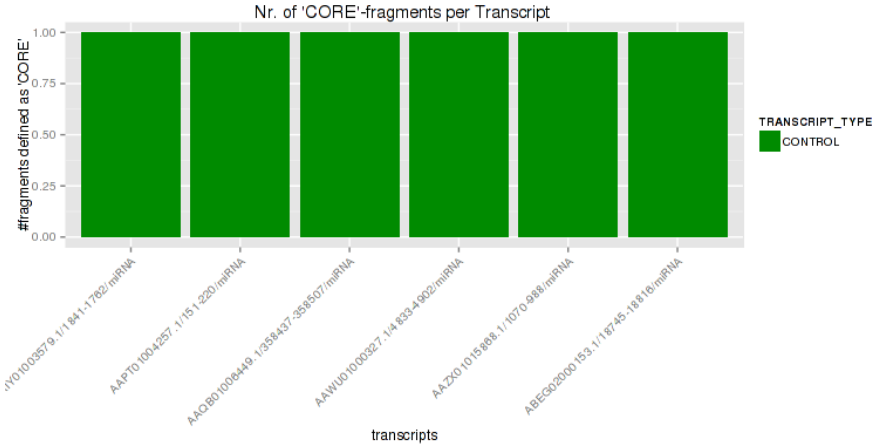

Figure A.4: `Histogram_By_CORE_IDs`.

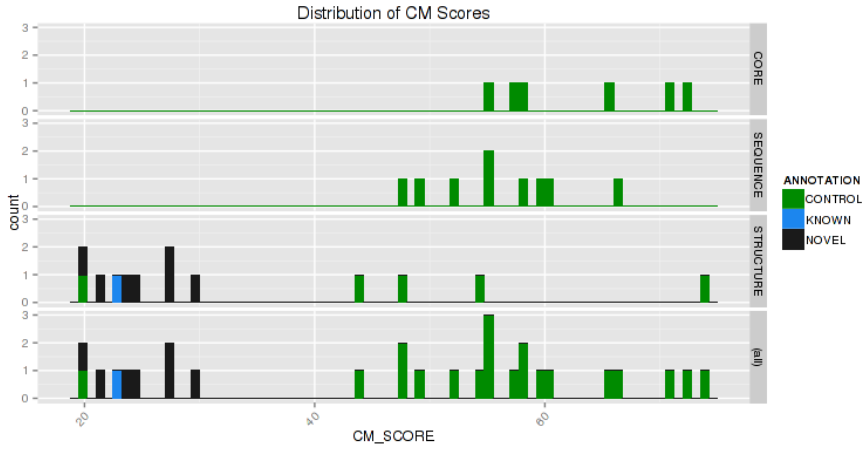

Figure A.5: `Histogram_By_CM_SCORE`.

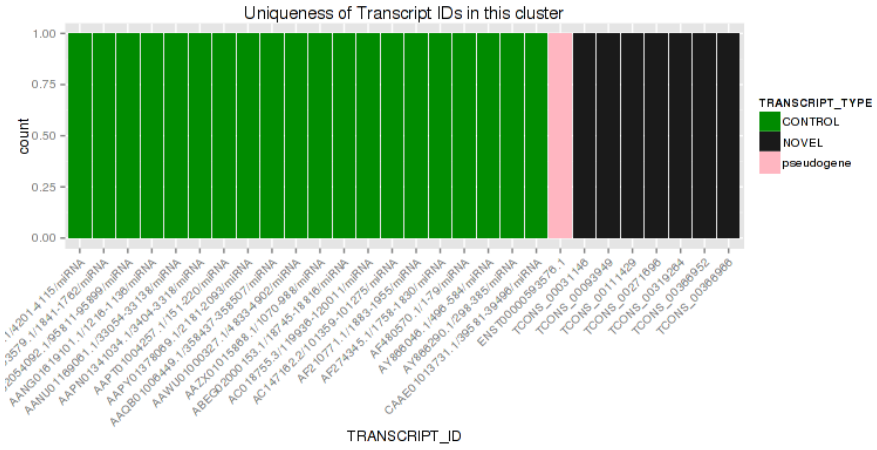

Figure A.6: `Histogram_NrOfTxFragmentsInCluster`.

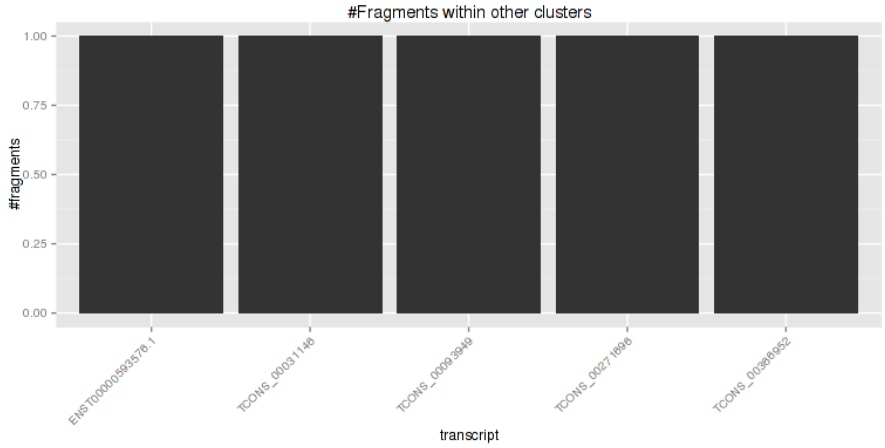

Figure A.7: `Histogram_NrOfTxFragmentsInOutside`.

Venn diagrams

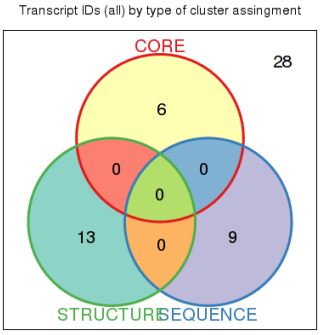

Figure A.8: `VennDiagram_all_transcripts`.

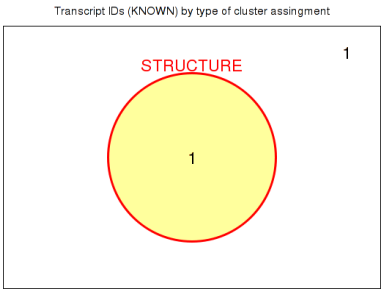

Figure A.9: `VennDiagram_KNOWN_transcripts`.

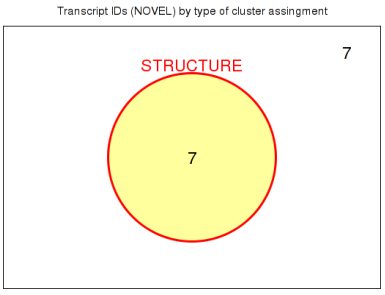

Figure A.10: `VennDiagram_NOVEL_transcripts`.

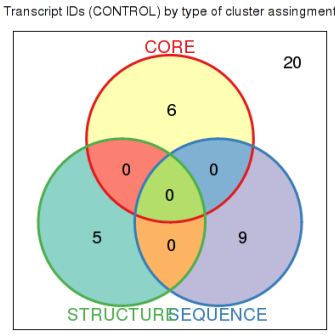

Figure A.11: `VennDiagram_CONTROL_transcripts`.

Further graphics

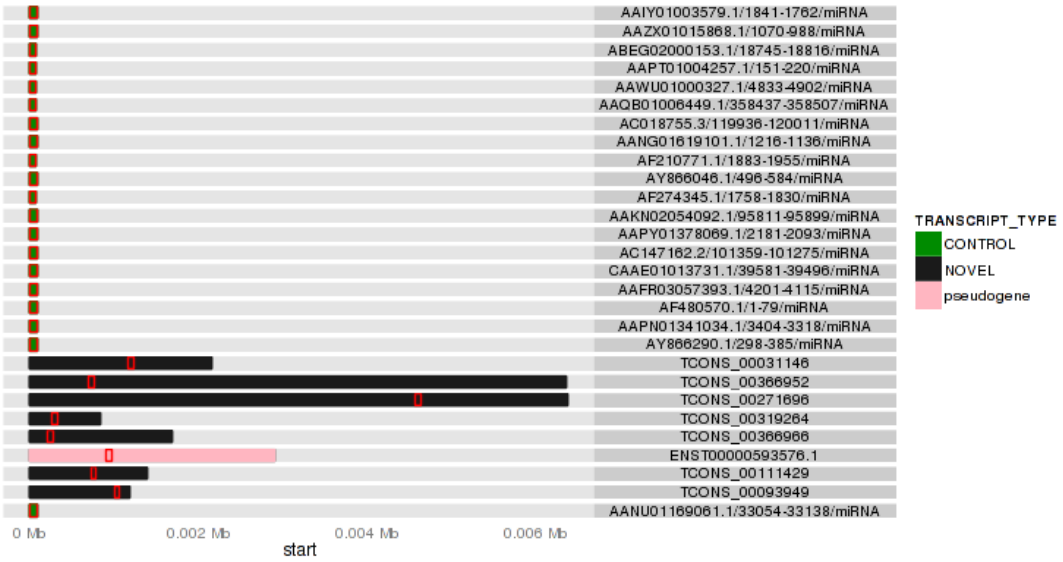

Figure A.12: `2DStructure_TOP5`.

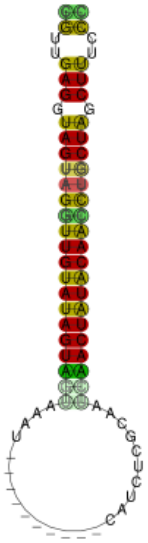

Figure A.13: `Locations_Of_ Motif_In_TXs`.

Appendix B: Comparison of clusters

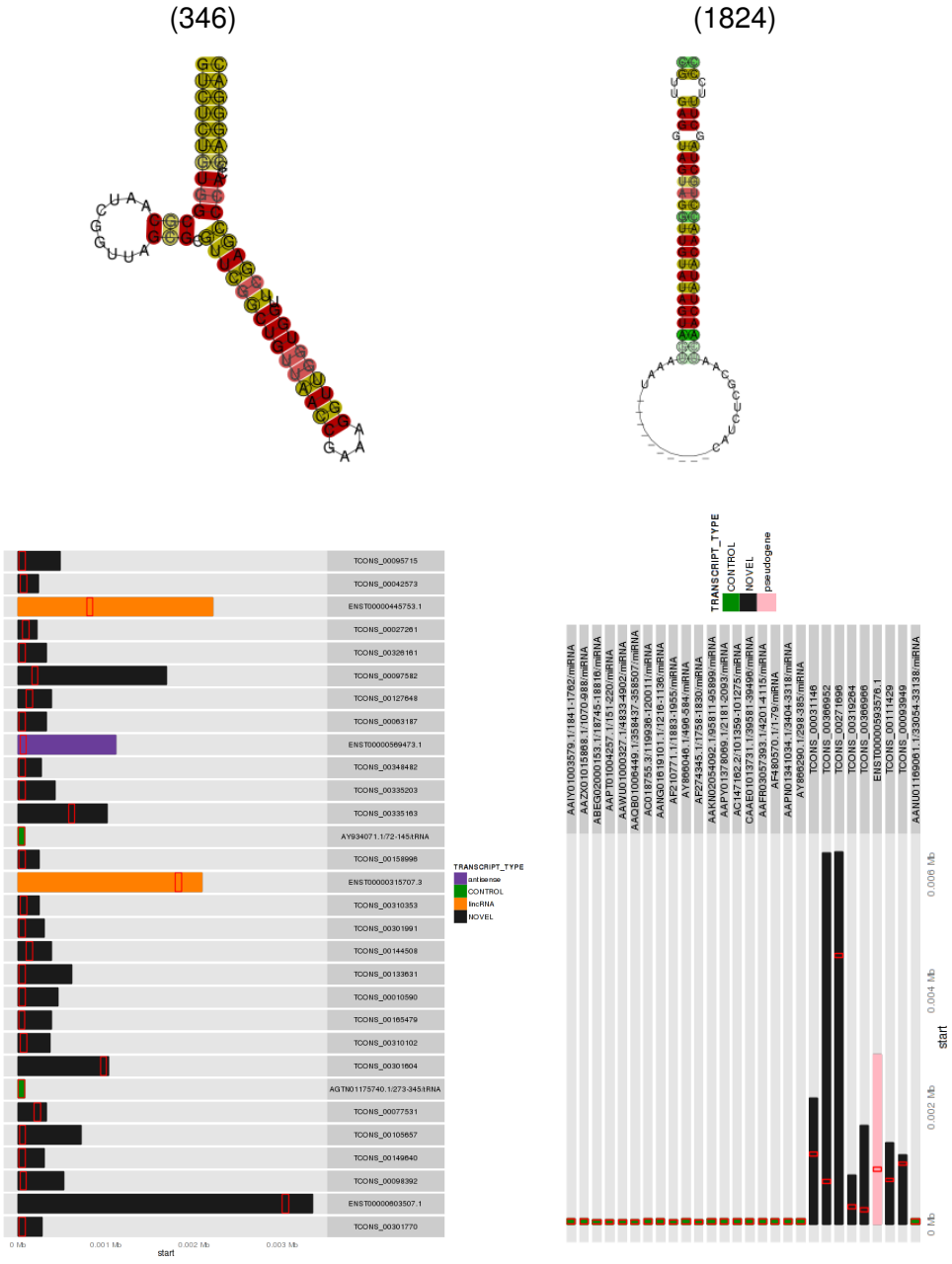

Figure B.1: **Comparison of cluster 346 and 1824 by their secondary structures and the motif location.** Both structures show a long and a short hairpin loop, while a location trend is not apparent from both motif location pictures.

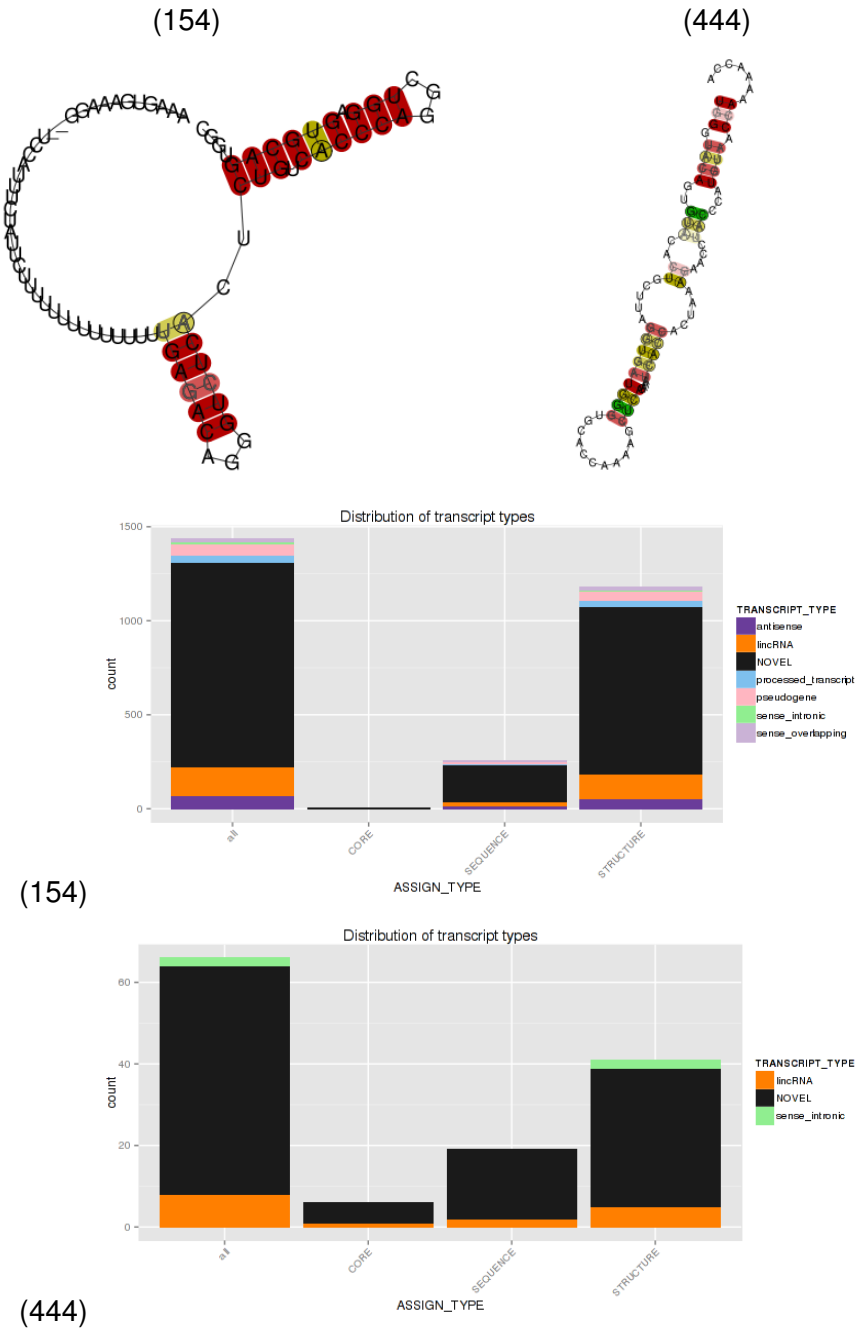

Figure B.2: **Comparison of cluster 154 and 444 by their secondary structures and their transcript types.** Both structures show roughly similar structures of hairpin loops. The motif-presenting sequences are assigned to the cluster mainly by structure similarity, while in particular no-coding RNAs are included.

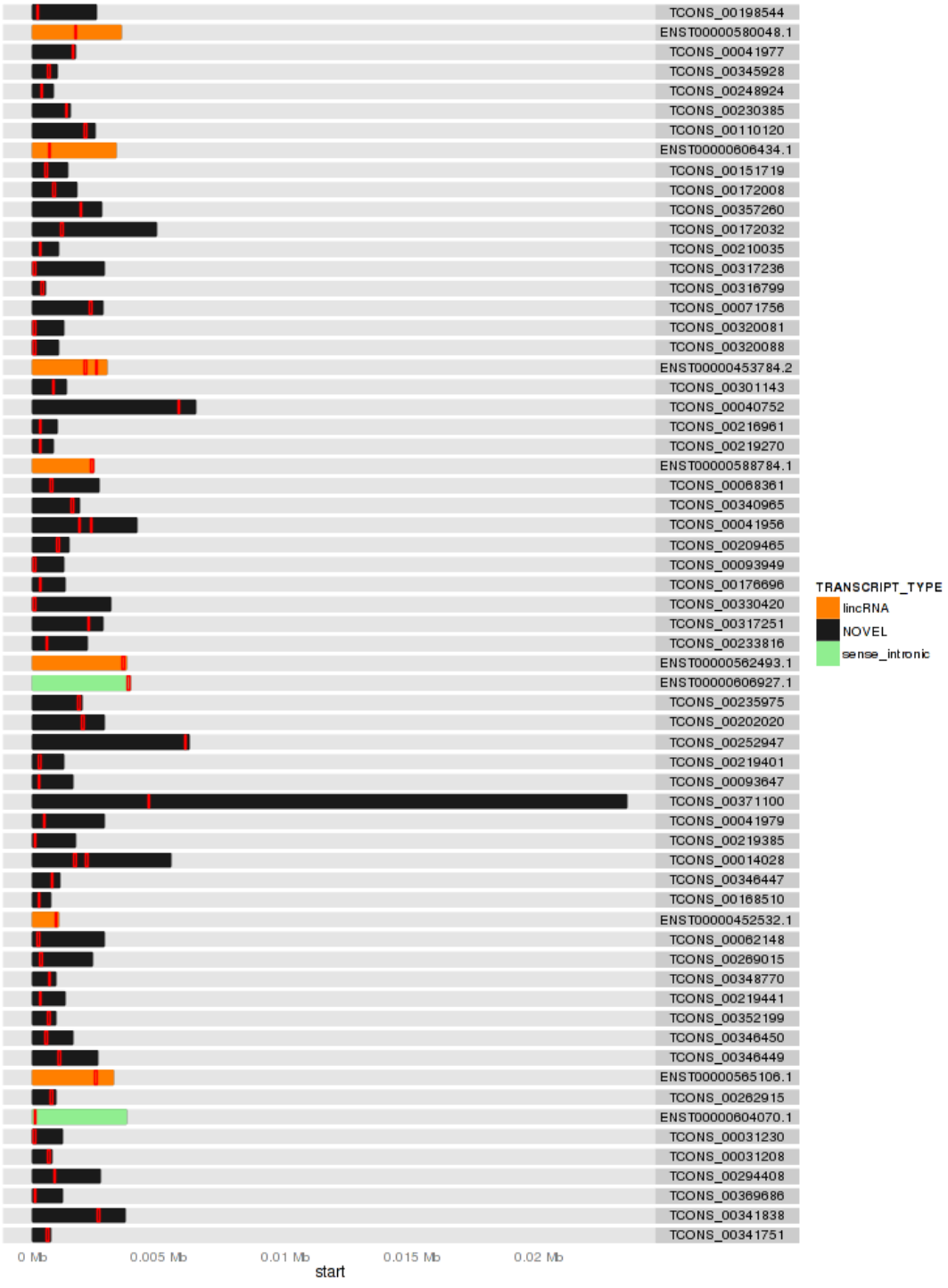

Figure B.3: **[Motif locations of cluster 444.** The locations of the motif 444 show no preference to the 3' or 5' end. The transcript IDs of annotated sequences can be used to search external databases for functional relationships.

Bibliography

[AAA+14] ACKERMANN, Wilfried ; ANSPACH, Birger ; APPEL, Bernd ; BORNSCHEUER, Uwe ; DECKWER, Monika ; FALBE, Jürgen ; FRANK, Ronald ; GOLLMER, Klaus-Uwe ; HAMPEL, Jürgen ; KRÄMER, Reinhard ; MICHEEL, Burkhard ; NIEMANN, Heiner ; PFISTER, Herbert ; POSTEN, Clemens ; PÜHLER, Alfred ; RAEDER, Ute ; REGITZ, Manfred ; RIEDEL, Klaus ; RUTTLOFF, Heinz ; SCHMID, Rolf D. ; SCHÜTTE, Horst ; WEHLAND, Jürgen ; WEHLMANN, Hermann ; WEILAND, Peter ; WILKE, Detlef ; WOBUS, Ulrich ; WOHLLEBEN, Wolfgang ; ZENG, An-Ping ; SCHELL, Thomas von: *RÖMPP Lexikon Biotechnologie und Gentechnik, 2. Auflage, 1999*. Thieme, 2014. – 878 S. http://books.google.com/books?id=MoyZAwAAQBAJ\&pgis=1. – ISBN 3131793325

[ABM+12] ARFI, Yonathan ; BUÉE, Marc ; MARCHAND, Cyril ; LEVASSEUR, Anthony ; RECORD, Eric: Multiple markers pyrosequencing reveals highly diverse and host-specific fungal communities on the mangrove trees Avicennia marina and Rhizophora stylosa. In: *FEMS microbiology ecology* 79 (2012), Februar, Nr. 2, 433–44. http://dx.doi.org/10.1111/j.1574-6941.2011.01236.x. – DOI 10.1111/j.1574–6941.2011.01236.x. – ISSN 1574–6941

[AMS+97] ALTSCHUL, S F. ; MADDEN, T L. ; SCHÄFFER, A A. ; ZHANG, J ; ZHANG, Z ; MILLER, W ; LIPMAN, D J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. In: *Nucleic acids research* 25 (1997), September, Nr. 17, 3389–402. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=146917\&tool=pmcentrez\&rendertype=abstract. – ISSN 0305–1048

[AVG07] AGRIS, Paul F. ; VENDEIX, Franck A P. ; GRAHAM, William D.: tRNA's wobble decoding of the genome: 40 years of modification. In: *Journal of molecular biology* 366 (2007), Februar, Nr. 1, 1–13. http://dx.doi.org/10.1016/j.jmb.2006.11.046. – DOI 10.1016/j.jmb.2006.11.046. – ISSN 0022–2836

[BB07] BÖCKENHAUER, Hans-Joachim ; BONGARTZ, Dirk: *Algorithmic Aspects of Bioinformatics*. Springer, 2007. – 406 S. http://books.google.com/books?id=_DMvZC6zJqAC\&pgis=1. – ISBN 354071913X

[BHW+08] BERNHART, Stephan H. ; HOFACKER, Ivo L. ; WILL, Sebastian ; GRUBER, Andreas R. ; STADLER, Peter F.: RNAalifold: improved consensus struc-

ture prediction for RNA alignments. In: *BMC bioinformatics* 9 (2008), Januar, Nr. 1, 474. http://dx.doi.org/10.1186/1471-2105-9-474. – DOI 10.1186/1471–2105–9–474. – ISSN 1471–2105

[BKEI04] BEN-KIKI, Oren ; EVANS, Clark ; INGERSON, Brian: *YAML Ain't Markup Language (YAML^{TM}) Version 1.1*. http://yaml.org/spec/current.pdf. Version: 2004

[BM04] BLACKBURN, C. C. ; MANLEY, Nancy R.: Developing a new paradigm for thmus organogenesis. In: *Nature Review Immunology* 4 (2004), 278–289. http://www.nature.com/nri/journal/v4/n4/fig_tab/nri1331_F1.html

[BNH+14] BAILEY, Stefanie R. ; NELSON, Michelle H. ; HIMES, Richard A. ; LI, Zihai ; MEHROTRA, Shikhar ; PAULOS, Chrystal M.: Th17 cells in cancer: the ultimate identity crisis. In: *Frontiers in Immunology* 5 (2014), Nr. 276. http://dx.doi.org/10.3389/fimmu.2014.00276. – DOI 10.3389/fimmu.2014.00276. – ISSN 1664–3224

[BSG10] BREMGES, Andreas ; SCHIRMER, Stefanie ; GIEGERICH, Robert: Fine-tuning structural RNA alignments in the twilight zone. In: *BMC bioinformatics* 11 (2010), Januar, Nr. 1, 222. http://dx.doi.org/10.1186/1471-2105-11-222. – DOI 10.1186/1471–2105–11–222. – ISSN 1471–2105

[CCL+11] CESANA, Marcella ; CACCHIARELLI, Davide ; LEGNINI, Ivano ; SANTINI, Tiziana ; STHANDIER, Olga ; CHINAPPI, Mauro ; TRAMONTANO, Anna ; BOZZONI, Irene: A long noncoding RNA controls muscle differentiation by functioning as a competing endogenous RNA. In: *Cell* 147 (2011), Oktober, Nr. 2, 358–69. http://dx.doi.org/10.1016/j.cell.2011.09.028. – DOI 10.1016/j.cell.2011.09.028. – ISSN 1097–4172

[CCW+12] COLLIER, Sarah P. ; COLLINS, Patrick L. ; WILLIAMS, Christopher L. ; BOOTHBY, Mark R. ; AUNE, Thomas M.: Cutting Edge: Influence of Tmevpg1 , a Long Intergenic Noncoding RNA, on the Expression of Ifng by Th1 Cells. In: *The Journal Of Immunology* 189 (2012), Nr. 5, 2084–8. http://dx.doi.org/10.4049/jimmunol.1200774. – DOI 10.4049/jimmunol.1200774

[CG03] COSTA, Fabrizio ; GRAVE, Kurt D.: Fast Neighborhood Subgraph Pairwise Distance Kernel. In: *Computer* 54 (2003), Nr. v, 255–262. http://dx.doi.org/10.1016/j.neuropharm.2007.07.003. – DOI 10.1016/j.neuropharm.2007.07.003. – ISSN 00283908

[CHS00] COUPLAND, S.E. ; HUMMEL, M. ; STEIN, H.: Lymphatisches System und Differenzierung von B- und T-Lymphozyten. In: *Der Pathologe* 21 (2000), 106–112. http://download.springer.com/static/pdf/263/art\%3A10.1007\%2Fs002920050378.pdf?auth66=1406736152_d48aed2442b8f1364a9214dda4e0ea30\&ext=.pdf

[Cos11] COSTA, Fabrizio: Beyond Base Pair Probabilities and Single Nucleotide Accessibility - The Way of Graph Kernels. In: *TBI Winterseminar in Bled*. Bled, 2011, 13

[DSBH09] DURINCK, Steffen ; SPELLMAN, Paul T. ; BIRNEY, Ewan ; HUBER, Wolfgang: Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. In: *Nature protocols* 4 (2009), Januar, Nr. 8, 1184–91. http://dx.doi.org/10.1038/nprot.2009.97. – DOI 10.1038/nprot.2009.97. – ISSN 1750–2799

[ED94] EDDY, Sean R. ; DURBIN, Richard: RNA sequence analysis using covariance models. In: *Nucleic acids research* 22 (1994), Juni, Nr. 11, 2079–88. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=308124\&tool=pmcentrez\&rendertype=abstract. – ISSN 0305–1048

[EHH13] EGGENHOFER, Florian ; HOFACKER, Ivo L. ; HÖNER ZU SIEDERDISSEN, Christian: CMCompare webserver: comparing RNA families via covariance models. In: *Nucleic acids research* 41 (2013), Juli, Nr. Web Server issue, W499–503. http://dx.doi.org/10.1093/nar/gkt329. – DOI 10.1093/nar/gkt329. – ISSN 1362–4962

[ENS07] EBERT, Margaret S. ; NEILSON, Joel R. ; SHARP, Phillip A.: MicroRNA sponges: competitive inhibitors of small RNAs in mammalian cells. In: *Nature methods* 4 (2007), September, Nr. 9, 721–6. http://dx.doi.org/10.1038/nmeth1079. – DOI 10.1038/nmeth1079. – ISSN 1548–7091

[FH11] FISCHER, Peter ; HOFER, Peter: *Lexikon Der Informatik*. Springer, 2011 http://books.google.de/books?id=8wZUvN5af_AC. – ISBN 9783642151262

[GCB+04] GENTLEMAN, Robert C. ; CAREY, Vincent J. ; BATES, Douglas M. ; BOLSTAD, Ben ; DETTLING, Marcel ; DUDOIT, Sandrine ; ELLIS, Byron ; GAUTIER, Laurent ; GE, Yongchao ; GENTRY, Jeff ; HORNIK, Kurt ; HOTHORN, Torsten ; HUBER, Wolfgang ; IACUS, Stefano ; IRIZARRY, Rafael ; LEISCH, Friedrich ; LI, Cheng ; MAECHLER, Martin ; ROSSINI, Anthony J. ; SAWITZKI, Gunther ; SMITH, Colin ; SMYTH, Gordon ; TIERNEY, Luke ; YANG, Jean Y H. ; ZHANG, Jianhua: Bioconductor: open software development

for computational biology and bioinformatics. In: *Genome biology* 5 (2004), Januar, Nr. 10, R80. http://dx.doi.org/10.1186/gb-2004-5-10-r80. – DOI 10.1186/gb–2004–5–10–r80. – ISSN 1465–6914

[GHS97] GORODKIN, J ; HEYER, L J. ; STORMO, G D.: Finding common sequence and structure motifs in a set of RNA sequences. In: *Proceedings / ... International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology* 5 (1997), Januar, 120–3. http://www.ncbi.nlm.nih.gov/pubmed/9322025. – ISSN 1553–0833

[GR12] GUTTMAN, Mitchell ; RINN, John L.: Modular regulatory principles of large non-coding RNAs. In: *Nature* 482 (2012), Februar, Nr. 7385, 339–46. http://dx.doi.org/10.1038/nature10887. – DOI 10.1038/nature10887. – ISSN 1476–4687

[GVR04] GIEGERICH, Robert ; VOSS, Björn ; REHMSMEIER, Marc: Abstract shapes of RNA. In: *Nucleic acids research* 32 (2004), Januar, Nr. 16, 4843–51. http://dx.doi.org/10.1093/nar/gkh779. – DOI 10.1093/nar/gkh779. – ISSN 1362–4962

[HBS04] HOFACKER, Ivo L. ; BERNHART, Stephan H F. ; STADLER, Peter F.: Alignment of RNA base pairing probability matrices. In: *Bioinformatics (Oxford, England)* 20 (2004), September, Nr. 14, 2222–7. http://dx.doi.org/10.1093/bioinformatics/bth229. – DOI 10.1093/bioinformatics/bth229. – ISSN 1367–4803

[HCRB12] HEYNE, Steffen ; COSTA, Fabrizio ; ROSE, Dominic ; BACKOFEN, Rolf: GraphClust: alignment-free structural clustering of local RNA secondary structures. In: *Bioinformatics (Oxford, England)* 28 (2012), Juni, Nr. 12, i224–32. http://dx.doi.org/10.1093/bioinformatics/bts224. – DOI 10.1093/bioinformatics/bts224. – ISSN 1367–4811

[HFG+12] HARROW, Jennifer ; FRANKISH, Adam ; GONZALEZ, Jose M. ; TAPANARI, Electra ; DIEKHANS, Mark ; KOKOCINSKI, Felix ; AKEN, Bronwen L. ; BARRELL, Daniel ; ZADISSA, Amonida ; SEARLE, Stephen ; BARNES, If ; BIGNELL, Alexandra ; BOYCHENKO, Veronika ; HUNT, Toby ; KAY, Mike ; MUKHERJEE, Gaurab ; RAJAN, Jeena ; DESPACIO-REYES, Gloria ; SAUNDERS, Gary ; STEWARD, Charles ; HARTE, Rachel ; LIN, Michael ; HOWALD, Cédric ; TANZER, Andrea ; DERRIEN, Thomas ; CHRAST, Jacqueline ; WALTERS, Nathalie ; BALASUBRAMANIAN, Suganthi ; PEI, Baikang ; TRESS, Michael ; RODRIGUEZ, Jose M. ; EZKURDIA, Iakes ; BAREN, Jeltje van ; BRENT, Michael ; HAUSSLER, David ; KELLIS, Manolis ; VALENCIA,

Alfonso ; REYMOND, Alexandre ; GERSTEIN, Mark ; GUIGÓ, Roderic ; HUBBARD, Tim J.: GENCODE: the reference human genome annotation for The ENCODE Project. In: *Genome research* 22 (2012), September, Nr. 9, 1760–74. http://dx.doi.org/10.1101/gr.135350.111. – DOI 10.1101/gr.135350.111. – ISSN 1549–5469

[HFS+94] HOFACKER, I. L. ; FONTANA, W. ; STADLER, P. F. ; BONHOEFFER, L. S. ; TACKER, M. ; SCHUSTER, P.: Fast folding and comparison of RNA secondary structures. In: *Monatshefte für Chemie Chemical Monthly* 125 (1994), Februar, Nr. 2, 167–188. http://dx.doi.org/10.1007/BF00818163. – DOI 10.1007/BF00818163. – ISSN 0026–9247

[HLSG05] HAVGAARD, Jakob H. ; LYNGSØ, Rune B. ; STORMO, Gary D. ; GORODKIN, Jan: Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. In: *Bioinformatics (Oxford, England)* 21 (2005), Mai, Nr. 9, 1815–24. http://dx.doi.org/10.1093/bioinformatics/bti279. – DOI 10.1093/bioinformatics/bti279. – ISSN 1367–4803

[HRO+14] HACKERMÜLLER, Jörg ; REICHE, Kristin ; OTTO, Christian ; HÖSLER, Nadine ; BLUMERT, Conny ; BROCKE-HEIDRICH, Katja ; BÖHLIG, Levin ; NITSCHE, Anne ; KASACK, Katharina ; AHNERT, Peter ; KRUPP, Wolfgang ; ENGELAND, Kurt ; STADLER, Peter F. ; HORN, Friedemann: Cell cycle, oncogenic and tumor suppressor pathways regulate numerous long and macro non-protein-coding RNAs. In: *Genome biology* 15 (2014), Januar, Nr. 3, R48. http://dx.doi.org/10.1186/gb-2014-15-3-r48. – DOI 10.1186/gb–2014–15–3–r48. – ISSN 1465–6914

[HTS+13] HU, Gangqing ; TANG, Qingsong ; SHARMA, Suveena ; YU, Fang ; ESCOBAR, Thelma M. ; MULJO, Stefan A. ; ZHU, Jinfang ; ZHAO, Keji: Expression and regulation of intergenic long noncoding RNAs during T cell development and differentiation. In: *Nature immunology* 14 (2013), November, Nr. 11, 1190–8. http://dx.doi.org/10.1038/ni.2712. – DOI 10.1038/ni.2712. – ISSN 1529–2916

[IEBK01] INGERSON, Brian ; EVANS, Clark ; BEN-KIKI, Oren: *Yet Another Markup Language (YAML) 1.0*. http://yaml.org/spec/history/2001-08-01.html. Version: 2001

[IG96] IHAKA, Ross ; GENTLEMAN, Robert: R: A Language for Data Analysis and Graphics. In: *Journal of Computational and Graphical Statistics* 5 (1996), September, Nr. 3, 299–314. http://dx.doi.org/10.1080/10618600.1996.10474713. – DOI 10.1080/10618600.1996.10474713. – ISSN 1061–8600

[JL11] JEON, Yesu ; LEE, Jeannie T.: YY1 tethers Xist RNA to the inactive X nucleation center. In: *Cell* 146 (2011), Juli, Nr. 1, 119–33. http://dx.doi.org/10.1016/j.cell.2011.06.026. – DOI 10.1016/j.cell.2011.06.026. – ISSN 1097–4172

[Kau14] KAUFMANN, Stefan H. E.: *Basiswissen Immunologie*. Springer Berlin Heidelberg, 2014. – 9–11,63–83,117 S. http://dx.doi.org/10.1007/978-3-642-40325-5. http://dx.doi.org/10.1007/978-3-642-40325-5. – ISBN 978–3–642–40325–5

[KBGC14] KIRKEGAARD, Karla A. ; BRAHIC, Michel ; GOMEZ, J. A. ; CHANG, Howard Yuan-Hao: *IMMUNOMODULATION BY CONTROLLING INTERFERON-GAMMA LEVELS WITH THE LONG NON-CODING RNA NeST*. http://www.google.com/patents/US20140056929. Version: Februar 2014

[KCL13] KUNG, Johnny T Y. ; COLOGNORI, David ; LEE, Jeannie T.: Long noncoding RNAs: past, present, and future. In: *Genetics* 193 (2013), März, Nr. 3, 651–69. http://dx.doi.org/10.1534/genetics.112.146704. – DOI 10.1534/genetics.112.146704. – ISSN 1943–2631

[KFB+08] KOCH, Ute ; FIORINI, Emma ; BENEDITO, Rui ; BESSEYRIAS, Valerie ; SCHUSTER-GOSSLER, Karin ; PIERRES, Michel ; MANLEY, Nancy R. ; DUARTE, Antonio ; MACDONALD, H R. ; RADTKE, Freddy: Delta-like 4 is the essential, nonredundant ligand for Notch1 during thymic T cell lineage commitment. In: *The Journal of experimental medicine* 205 (2008), Oktober, Nr. 11, 2515–23. http://dx.doi.org/10.1084/jem.20080829. – DOI 10.1084/jem.20080829. – ISSN 1540–9538

[KH03] KNUDSEN, Bjarne ; HEIN, Jotun: Pfold: RNA secondary structure prediction using stochastic context-free grammars. In: *Nucleic acids research* 31 (2003), Juli, Nr. 13, 3423–8. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=169020\&tool=pmcentrez\&rendertype=abstract. – ISSN 1362–4962

[KL02] KONDOR, Risi I. ; LAFFERTY, John: Diffusion kernels on graphs and other discrete structures. In: *In Proceedings of the ICML* (2002). http://dx.doi.org/10.1.1.128.4966. – DOI 10.1.1.128.4966

[Kol13] KOLDE, Raivo: *pheatmap: Pretty Heatmaps*, 2013. http://cran.r-project.org/package=pheatmap

[KR11] KOCH, Ute ; RADTKE, Freddy: Mechanisms of T cell development and transformation. In: *Annual review of cell and developmental bi-*

ology 27 (2011), Januar, 539–62. http://dx.doi.org/10.1146/annurev-cellbio-092910-154008. – DOI 10.1146/annurev–cellbio–092910–154008. – ISSN 1530–8995

[KTK+05] KATAYAMA, S ; TOMARU, Y ; KASUKAWA, T ; WAKI, K ; NAKANISHI, M ; NAKAMURA, M ; NISHIDA, H ; YAP, C C. ; SUZUKI, M ; KAWAI, J ; SUZUKI, H ; CARNINCI, P ; HAYASHIZAKI, Y ; WELLS, C ; FRITH, M ; RAVASI, T ; PANG, K C. ; HALLINAN, J ; MATTICK, J ; HUME, D A. ; LIPOVICH, L ; BATALOV, S ; ENGSTRÖM, P G. ; MIZUNO, Y ; FAGHIHI, M A. ; SANDELIN, A ; CHALK, A M. ; MOTTAGUI-TABAR, S ; LIANG, Z ; LENHARD, B ; WAHLESTEDT, C: Antisense transcription in the mammalian transcriptome. In: *Science (New York, N.Y.)* 309 (2005), Oktober, Nr. 5740, 1564–6. http://dx.doi.org/10.1126/science.1112009. – DOI 10.1126/science.1112009. – ISSN 1095–9203

[LBH+11] LORENZ, Ronny ; BERNHART, Stephan H. ; HÖNER ZU SIEDERDISSEN, Christian ; TAFER, Hakim ; FLAMM, Christoph ; STADLER, Peter F. ; HOFACKER, Ivo L.: ViennaRNA Package 2.0. In: *Algorithms for molecular biology : AMB* 6 (2011), Januar, Nr. 1, 26. http://dx.doi.org/10.1186/1748-7188-6-26. – DOI 10.1186/1748–7188–6–26. – ISSN 1748–7188

[LG06] LI, Weizhong ; GODZIK, Adam: Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. In: *Bioinformatics (Oxford, England)* 22 (2006), Juli, Nr. 13, 1658–9. http://dx.doi.org/10.1093/bioinformatics/btl158. – DOI 10.1093/bioinformatics/btl158. – ISSN 1367–4803

[LGC09] LAWRENCE, Michael ; GENTLEMAN, Robert ; CAREY, Vincent: rtracklayer: an R package for interfacing with genome browsers. In: *Bioinformatics* 25 (2009), 1841–1842. http://dx.doi.org/10.1093/bioinformatics/btp328. – DOI 10.1093/bioinformatics/btp328

[LHA14] LOVE, M. I. ; HUBER, W. ; ANDERS, S.: Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2. Version: Februar 2014. http://dx.doi.org/10.1101/002832. Cold Spring Harbor Labs Journals, Februar 2014. – Forschungsbericht. – 002832 S.

[LHP+13] LAWRENCE, Michael ; HUBER, Wolfgang ; PAGÈS, Hervé ; ABOYOUN, Patrick ; CARLSON, Marc ; GENTLEMAN, Robert ; MORGAN, Martin T. ; CAREY, Vincent J.: Software for computing and annotating genomic ranges. In: *PLoS computational biology* 9 (2013), Januar, Nr. 8, e1003118. http://dx.doi.org/10.1371/journal.pcbi.1003118. – DOI 10.1371/journal.pcbi.1003118. – ISSN 1553–7358

[LK06] LAI, Anne Y. ; KONDO, Motonari: Asymmetrical lymphoid and myeloid lineage commitment in multipotent hematopoietic progenitors. In: *The Journal of experimental medicine* 203 (2006), August, Nr. 8, 1867–73. http://dx.doi.org/10.1084/jem.20060697. – DOI 10.1084/jem.20060697. – ISSN 0022–1007

[LRB+12] LESSA, Felipe A. ; RAIOL, Tainá ; BRIGIDO, Marcelo M. ; MARTINS NETO, Daniele S. B. ; WALTER, Maria Emília M. T. ; STADLER, Peter F.: Clustering Rfam 10.1: Clans, Families, and Classes. In: *Genes* 3 (2012), Juli, Nr. 4, 378–390. http://dx.doi.org/10.3390/genes3030378. – DOI 10.3390/genes3030378. – ISBN 10.3390/genes3030378

[Mat03] MATTICK, John S.: Challenging the dogma: the hidden layer of non-protein-coding RNAs in complex organisms. In: *BioEssays : news and reviews in molecular, cellular and developmental biology* 25 (2003), Oktober, Nr. 10, 930–9. http://dx.doi.org/10.1002/bies.10332. – DOI 10.1002/bies.10332. – ISSN 0265–9247

[McC90] MCCASKILL, J S.: The equilibrium partition function and base pair binding probabilities for RNA secondary structure. In: *Biopolymers* 29 (1990), Nr. 6-7, 1105–19. http://dx.doi.org/10.1002/bip.360290621. – DOI 10.1002/bip.360290621. – ISSN 0006–3525

[MCG+13] MORCEAU, Franck ; CHATEAUVIEUX, Sébastien ; GAIGNEAUX, Anthoula ; DICATO, Mario ; DIEDERICH, Marc: Long and short non-coding RNAs as regulators of hematopoietic differentiation. In: *International journal of molecular sciences* 14 (2013), Januar, Nr. 7, 14744–70. http://dx.doi.org/10.3390/ijms140714744. – DOI 10.3390/ijms140714744. – ISSN 1422–0067

[MHC+09] MAHMOUDI, Salah ; HENRIKSSON, Sofia ; CORCORAN, Martin ; MÉNDEZ-VIDAL, Cristina ; WIMAN, Klas G. ; FARNEBO, Marianne: Wrap53, a natural p53 antisense transcript required for p53 induction upon DNA damage. In: *Molecular cell* 33 (2009), Februar, Nr. 4, 462–71. http://dx.doi.org/10.1016/j.molcel.2009.01.028. – DOI 10.1016/j.molcel.2009.01.028. – ISSN 1097–4164

[MHF+11] MAHMOUDI, S ; HENRIKSSON, S ; FARNEBO, L ; ROBERG, K ; FARNEBO, M: WRAP53 promotes cancer cell survival and is a potential target for cancer therapy. In: *Cell death & disease* 2 (2011), Januar, e114. http://dx.doi.org/10.1038/cddis.2010.90. – DOI 10.1038/cddis.2010.90. – ISSN 2041–4889

[Mur09] MURRELL, Paul: Importing Vector Graphics: The {grImport} Package for

{R}. In: *Journal of Statistical Software* 30 (2009), Nr. 4, 1–37. http://www.jstatsoft.org/v30/i04/

[MW09] MERKL, Rainer ; WAACK, Stephan: *Bioinformatik Interaktiv: Grundlagen, Algorithmen, Anwendungen*. 2. WILEY-VCH, 2009. – 596 S. http://books.google.com/books?id=A0FVAgAAQBAJ\&pgis=1. – ISBN 987–3–527–32594–8

[MZP02] MICHIE, Alison M. ; ZUNIGA-PFLÜCKER, Juan C.: Regulation of thymocyte differentiation: pre-TCR signals and β-selection. In: *Seminars in Immunology* 14 (2002), Oktober, Nr. 5, 311–323. http://dx.doi.org/10.1016/S1044-5323(02)00064-7. – DOI 10.1016/S1044–5323(02)00064–7. – ISSN 10445323

[NKE09] NAWROCKI, Eric P. ; KOLBE, Diana L. ; EDDY, Sean R.: Infernal 1.0: inference of RNA alignments. In: *Bioinformatics (Oxford, England)* 25 (2009), Mai, Nr. 10, 1335–7. http://dx.doi.org/10.1093/bioinformatics/btp157. – DOI 10.1093/bioinformatics/btp157. – ISSN 1367–4811

[NPGK78] NUSSINOV, Ruth ; PIECZENIK, George ; GRIGGS, Jerrold R. ; KLEITMAN, Daniel J.: Algorithms For Loop Matchings. In: *SIAM Journal on Applied Mathematics* 35 (1978), Nr. 1, 68–82. http://rci.rutgers.edu/~piecze/GriggsNussinovKleitmanPieczenik.pdf

[OrS13] Ø ROM, Ulf A. ; SHIEKHATTAR, Ramin: Long noncoding RNAs usher in a new era in the biology of enhancers. In: *Cell* 154 (2013), September, Nr. 6, 1190–3. http://dx.doi.org/10.1016/j.cell.2013.08.028. – DOI 10.1016/j.cell.2013.08.028. – ISSN 1097–4172

[Pag14] PAGES, Herve: *BSgenome: Infrastructure for Biostrings-based genome data packages*, 2014

[PCFL] PAGES, Herve ; CARLSON, Marc ; FALCON, Seth ; LI, Nianhua: *AnnotationDbi: Annotation Database Interface*

[PDM+09] PANG, Ken C. ; DINGER, Marcel E. ; MERCER, Tim R. ; MALQUORI, Lorenzo ; GRIMMOND, Sean M. ; CHEN, Weisan ; MATTICK, John S.: Genome-wide identification of long noncoding RNAs in CD8+ T cells. In: *Journal of immunology (Baltimore, Md. : 1950)* 182 (2009), Juni, Nr. 12, 7738–48. http://dx.doi.org/10.4049/jimmunol.0900603. – DOI 10.4049/jimmunol.0900603. – ISSN 1550–6606

[PRP+13] PAGANI, Massimiliano ; ROSSETTI, Grazisa ; PANZERI, Ilaria ; CANDIA, Paola de ; BONNAL, Raoul J P. ; ROSSI, Riccardo L. ; GEGINAT, Jens

; ABRIGNANI, Sergio: Role of microRNAs and long-non-coding RNAs in CD4(+) T-cell differentiation. In: *Immunological reviews* 253 (2013), Mai, Nr. 1, 82–96. http://dx.doi.org/10.1111/imr.12055. – DOI 10.1111/imr.12055. – ISSN 1600–065X

[PWC+11] PINK, Ryan C. ; WICKS, Kate ; CALEY, Daniel P. ; PUNCH, Emma K. ; JACOBS, Laura ; CARTER, David Raul F.: Pseudogenes: pseudo-functional or key regulators in health and disease? In: *RNA (New York, N.Y.)* 17 (2011), Mai, Nr. 5, 792–8. http://dx.doi.org/10.1261/rna.2658311. – DOI 10.1261/rna.2658311. – ISSN 1469–9001

[R C14] R CORE TEAM ; R FOUNDATION FOR STATISTICAL COMPUTING (Hrsg.): *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2014. http://www.r-project.org/

[Rei10] REICHE, Kristin: *RNAclust*. http://www.bioinf.uni-leipzig.de/~kristin/Software/RNAclust/. Version: 2010

[RI14] RSTUDIO ; INC.: *shiny: Web Application Framework for R*, 2014. http://cran.r-project.org/package=shiny

[Rie94] RIEBER, E. P.: CDw60: a marker for human CD8+ T helper cells. In: *Journal of Experimental Medicine* 179 (1994), April, Nr. 4, 1385–1390. http://dx.doi.org/10.1084/jem.179.4.1385. – DOI 10.1084/jem.179.4.1385. – ISSN 0022–1007

[RKS08] RABANI, Michal ; KERTESZ, Michael ; SEGAL, Eran: Computational prediction of RNA structural motifs involved in posttranscriptional regulatory processes. In: *Proceedings of the National Academy of Sciences of the United States of America* 105 (2008), September, Nr. 39, 14885–90. http://dx.doi.org/10.1073/pnas.0803169105. – DOI 10.1073/pnas.0803169105. – ISSN 1091–6490

[RKS+14] REICHE, Kristin ; KASACK, Katharina ; SCHREIBER, Stephan ; LÜDERS, Torben ; DUE, Eldri U. ; NAUME, Bjø r. ; RIIS, Margit ; KRISTENSEN, Vessela N. ; HORN, Friedemann ; DALE, Anne-Lise Bø r. ; HACKERMÜLLER, Jörg ; BAUMBUSCH, Lars O.: Long Non-Coding RNAs Differentially Expressed between Normal versus Primary Breast Tumor Tissues Disclose Converse Changes to Breast Cancer-Related Protein-Coding Genes. In: *PloS one* 9 (2014), Januar, Nr. 9, e106076. http://dx.doi.org/10.1371/journal.pone.0106076. – DOI 10.1371/journal.pone.0106076. – ISSN 1932–6203

[RMTC13] RADTKE, Freddy ; MACDONALD, H R. ; TACCHINI-COTTIER, Fabienne:

Regulation of innate and adaptive immunity by Notch. In: *Nature reviews. Immunology* 13 (2013), Juni, Nr. 6, 427–37. http://dx.doi.org/10.1038/nri3445. – DOI 10.1038/nri3445. – ISSN 1474–1741

[RMY08] ROTHENBERG, Ellen V. ; MOORE, Jonathan E. ; YUI, Mary A.: Launching the T-cell-lineage developmental programme. In: *Nature reviews. Immunology* 8 (2008), Januar, Nr. 1, 9–21. http://dx.doi.org/10.1038/nri2232. – DOI 10.1038/nri2232. – ISSN 1474–1741

[Rou87] ROUSSEEUW, Peter J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. In: *Journal of Computational and Applied Mathematics* 20 (1987), November, 53–65. http://dx.doi.org/10.1016/0377-0427(87)90125-7. – DOI 10.1016/0377–0427(87)90125–7. – ISSN 03770427

[San85] SANKOFF, David: Simultaneous Solution of the RNA Folding, Alignment and Protosequence Problems. In: *SIAM Journal on Applied Mathematics* 45 (1985), Nr. 5, S. 810 – 825

[SB05] SIERRA, K ; BATES, B: *Head First Java:*. 2. O'Reilly Media, Incorporated, 2005 (Head first series). – 688 S. http://books.google.de/books?id=uIVJiAPlBq0C. – ISBN 9780596009205

[SGE09] SCHARTL, Manfred (Hrsg.) ; GESSLER, Manfred (Hrsg.) ; ECKARDSTEIN, Arnold von (Hrsg.): *Biochemie und Molekularbiologie des Menschen*. 1. München : Elsevier GmbH, Urban \& Fischer Verlag, 2009. – 1056 S. – ISBN 978–3–437–43690–1

[Spe99] SPEKTRUM AKADEMISCHER VERLAG: *gamma-delta-T-Zellen*. http://www.spektrum.de/lexikon/biologie/gamma-delta-t-zellen/68232. Version: 1999

[Ste14] STEPHEN, Jeremy: *Methods to convert R data to YAML and back*, 2014. http://cran.r-project.org/web/packages/yaml/index.html

[Swi13] SWINTON, Jonathan: *Vennerable: Venn and Euler area-proportional diagrams*, 2013. http://r-forge.r-project.org/projects/vennerable/

[Tea14] TEAM TBD: *BSgenome.Hsapiens.UCSC.hg19: Full genome sequences for Homo sapiens (UCSC version hg19)*, 2014

[The07] THE ENCODE PROJECT CONSORTIUM: Identification and analysis of functional elements in 1% of the human genome by the ENCODE pi-

lot project. In: *Nature* 447 (2007), Juni, Nr. 7146, 799–816. http://dx.doi.org/10.1038/nature05874. – DOI 10.1038/nature05874. – ISSN 1476–4687

[THG07] TORARINSSON, Elfar ; HAVGAARD, Jakob H. ; GORODKIN, Jan: Multiple structural alignment and clustering of RNA sequences. In: *Bioinformatics (Oxford, England)* 23 (2007), April, Nr. 8, 926–32. http://dx.doi.org/10.1093/bioinformatics/btm049. – DOI 10.1093/bioinformatics/btm049. – ISSN 1367–4811

[TRLS13] TIERNEY, Luke ; ROSSINI, A J. ; LI, Na ; SEVCIKOVA, H: *snow: Simple Network of Workstations*, 2013. http://cran.r-project.org/package=snow

[TUL71] TINOCO, Ignacio ; UHLENBECK, Olke C. ; LEVINE, Mark D.: Estimation of Secondary Structure in Ribonucleic Acids. In: *Nature* 230 (1971), April, Nr. 5293, 362–367. http://dx.doi.org/10.1038/230362a0. – DOI 10.1038/230362a0. – ISSN 0028–0836

[VRBB03] VIGNEAU, Soline ; ROHRLICH, Pierre-Simon ; BRAHIC, Michel ; BUREAU, Jean-François: Tmevpg1, a candidate gene for the control of Theiler's virus persistence, could be implicated in the regulation of gamma interferon. In: *Journal of virology* 77 (2003), Mai, Nr. 10, 5632–8. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=154023\&tool=pmcentrez\&rendertype=abstract. – ISSN 0022–538X

[WC11] WANG, Kevin C. ; CHANG, Howard Y.: Molecular mechanisms of long noncoding RNAs. In: *Molecular cell* 43 (2011), September, Nr. 6, 904–14. http://dx.doi.org/10.1016/j.molcel.2011.08.018. – DOI 10.1016/j.molcel.2011.08.018. – ISSN 1097–4164

[WHS05] WASHIETL, Stefan ; HOFACKER, Ivo L. ; STADLER, Peter F.: Fast and reliable prediction of noncoding RNAs. In: *Proceedings of the National Academy of Sciences of the United States of America* 102 (2005), Februar, Nr. 7, 2454–9. http://dx.doi.org/10.1073/pnas.0409169102. – DOI 10.1073/pnas.0409169102. – ISSN 0027–8424

[Wic09] WICKHAM, Hadley: *ggplot2: elegant graphics for data analysis*. Springer New York, 2009 http://had.co.nz/ggplot2/book. – ISBN 978–0–387–98140–6

[WKG14] WASHIETL, Stefan ; KELLIS, Manolis ; GARBER, Manuel: Evolutionary dynamics and tissue specificity of human long noncoding RNAs in six

mammals. In: *Genome research* (2014), März, gr.165035.113–. http://dx.doi.org/10.1101/gr.165035.113. – DOI 10.1101/gr.165035.113. – ISSN 1549–5469

[WRH+07] WILL, Sebastian ; REICHE, Kristin ; HOFACKER, Ivo L. ; STADLER, Peter F. ; BACKOFEN, Rolf: Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. In: *PLoS computational biology* 3 (2007), April, Nr. 4, e65. http://dx.doi.org/10.1371/journal.pcbi.0030065. – DOI 10.1371/journal.pcbi.0030065. – ISSN 1553–7358

[WWH+12] WASHIETL, Stefan ; WILL, Sebastian ; HENDRIX, David A. ; GOFF, Loyal A. ; RINN, John L. ; BERGER, Bonnie ; KELLIS, Manolis: Computational analysis of noncoding RNAs. In: *Wiley interdisciplinary reviews. RNA* 3 (2012), Nr. 6, 759–78. http://dx.doi.org/10.1002/wrna.1134. – DOI 10.1002/wrna.1134. – ISSN 1757–7012

[XDY+14] XIA, Fei ; DONG, Fulu ; YANG, Yi ; HUANG, Anfei ; CHEN, Si ; SUN, Di ; XIONG, Sidong ; ZHANG, Jinping: Dynamic Transcription of Long Non-Coding RNA Genes during CD4+ T Cell Development and Activation. In: *PloS one* 9 (2014), Januar, Nr. 7, e101588. http://dx.doi.org/10.1371/journal.pone.0101588. – DOI 10.1371/journal.pone.0101588. – ISSN 1932–6203

[Xie14] XIE, Yihui: *knitr: A general-purpose package for dynamic report generation in R.*, 2014. http://yihui.name/knitr/

[YCL12] YIN, Tengfei ; COOK, Dianne ; LAWRENCE, Michael: ggbio: an R package for extending the grammar of graphics for genomic data. In: *Genome Biology* 13 (2012), Nr. 8, S. R77

[YR14] YUI, Mary A. ; ROTHENBERG, Ellen V.: Developmental gene networks: a triathlon on the course to T cell identity. In: *Nature Reviews Immunology* 14 (2014), Juli, Nr. 8, 529–545. http://dx.doi.org/10.1038/nri3702. – DOI 10.1038/nri3702. – ISSN 1474–1733

[YWR06] YAO, Zizhen ; WEINBERG, Zasha ; RUZZO, Walter L.: CMfinder–a covariance model based RNA motif finding algorithm. In: *Bioinformatics (Oxford, England)* 22 (2006), Februar, Nr. 4, 445–52. http://dx.doi.org/10.1093/bioinformatics/btk008. – DOI 10.1093/bioinformatics/btk008. – ISSN 1367–4803

[ZB08] ZVELEBIL, M J. ; BAUM, J O. ; HOLDSWORTH, Dom (Hrsg.): *Understanding Bioinformatics*. New York and London : Garland Science, 2008. –

772 S. http://books.google.de/books?id=dGayL_tdnBMC. – ISBN 9780815340249

[ZCL09] ZHOU, Liang ; CHONG, Mark M W. ; LITTMAN, Dan R.: Plasticity of CD4+ T cell lineage differentiation. In: *Immunity* 30 (2009), Mai, Nr. 5, 646–55. http://dx.doi.org/10.1016/j.immuni.2009.05.001. – DOI 10.1016/j.immuni.2009.05.001. – ISSN 1097–4180

[ZS81] ZUKER, M ; STIEGLER, P: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. In: *Nucleic acids research* 9 (1981), Januar, Nr. 1, 133–48. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=326673\&tool=pmcentrez\&rendertype=abstract. – ISSN 0305–1048

[ZSB+07] ZOLA, Heddy ; SWART, Bernadette ; BANHAM, Alison ; BARRY, Simon ; BEARE, Alice ; BENSUSSAN, Armand ; BOUMSELL, Laurence ; D BUCKLEY, Chris ; BÜHRING, Hans-Jörg ; CLARK, Georgina ; ENGEL, Pablo ; FOX, David ; JIN, Bo-Quan ; MACARDLE, Peter J. ; MALAVASI, Fabio ; MASON, David ; STOCKINGER, Hannes ; YANG, Xifeng: CD molecules 2006–human cell differentiation molecules. In: *Journal of immunological methods* 319 (2007), Januar, Nr. 1-2, 1–5. http://dx.doi.org/10.1016/j.jim.2006.11.001. – DOI 10.1016/j.jim.2006.11.001. – ISSN 0022–1759

[ZSE+08] ZHAO, Jing ; SUN, Bryan K. ; ERWIN, Jennifer A. ; SONG, Ji-Joon ; LEE, Jeannie T.: Polycomb proteins targeted by a short repeat RNA to the mouse X chromosome. In: *Science (New York, N.Y.)* 322 (2008), Oktober, Nr. 5902, 750–6. http://dx.doi.org/10.1126/science.1163045. – DOI 10.1126/science.1163045. – ISSN 1095–9203

[ZSWM00] ZHANG, Z ; SCHWARTZ, S ; WAGNER, L ; MILLER, W: A greedy algorithm for aligning DNA sequences. In: *Journal of computational biology : a journal of computational molecular cell biology* 7 (2000), Nr. 1-2, 203–14. http://dx.doi.org/10.1089/10665270050081478. – DOI 10.1089/10665270050081478. – ISSN 1066–5277

Statement of authorship

I hereby declare that I have written these master thesis on my own, with nothing more than the sources of information and aids stated above. Passages from sources, which are used litarally or analogously, where marked in such a way.

Furthermore, these master thesis has never been submitted in any kind of examination proceedings before.

Mittweida, 28.11.2014