&

University College London & the Ohio State University

Module Code
Module Name

microRNA expression profiles in human brain autopsy tissues

Christoph Sadée (15084362) May 12, 2016

Abstract

Divo.R is a R package used to analyse the diversity and similarity overlap between populations applicable to biological systems. Next generation sequencing produces large scale data often beyond the capabilities of divo.R using a standard desktop. A re-implementation of the current R extension, python into C reduced the run time for a 6208×1024 data table from +2hrs to 137seconds. The validity of the result was tested against previous published data. The new implementation was then used to analyse micro RNA data from ten brain regions from ten people. Dendrograms highlighted a particular microRNA, with potential implications of stroke therapy. This microRNA clustered with 2 other microRNAs which might also be important.

Acknowledgements

Dr. Maciej Pietrzak

Thanks to Dr. Pietrzak and his great help in explaining the python version of divo.R and also with many other things.

Ms. Katherinne Hartmann

Thanks to Ms. Hartmann, for taking the time and explaining the biological data in depth and helping to post process my results.

Prof. Grzegorz Rempala

Thanks to Prof. Rempala for inviting me to OSU, suggesting this project to me and acting as my mentor.

Prof. Peter Scambler

Thanks to Prof. Scambler for acting as my supervisor in UCL and his helpful attitude.

Contents

1 Introduction **4**

2 MicroRNA (miRNA) **4**
2.1 MiRNA biogenesis 4
2.2 MiRNA mechanism 6

3 MiRNA data **6**

4 Mathematical Method **7**

5 Coding **9**
5.1 R extension to C 10
5.2 Structure 11

6 Results and Discussion **14**
6.1 Validation 14
6.2 Speed improvement 14
6.3 miRNA 15

7 Conclusion **17**

8 Afterword **18**

References **18**

1 Introduction

MicroRNAs were recently discovered to be important in regulating gene expression, adding an additional control layer. MicroRNAs are well expressed in the brain, suggesting a link between the brain complexity and usage of additional control of protein regulation. Their length of 19 to 22 nucleotides is sufficient to target specific mRNAs for degradation. Importantly miRNAs often target multiple mRNAs and each mRNA is in turn regulated by multiple microRNAs, making the system quite complex and requiring mathematical modelling. To analyse the expression levels in different brain regions several mathematical overlap functions were implemented into an R package, called divo.R, allowing the user to group co-expressed microRNAs that may be functionally related to one another. Divo.R had been implemented in R and was then transfered to Python with a R wrapper for improved speed. Still unable large, next generation sequencing, data sets, a third version is implemented here in C with a R wrapper, promising a significant decrease in computing time.

2 MicroRNA (miRNA)

MicroRNAs (miRNA) are a type of non-coding RNAs, with a length of 19 to 25 nucleotides. Non-coding RNAs do not code for proteins but have a variety of different functions. MiRNA were observed to up- or down regulate protein coding genes, mainly at the translational level from mRNA to protein, by degradation of mRNA. Regulation was shown to affect embryonic development, differentiation, cell proliferation, stem cell renewal apoptosis and metabolism[1], hence the occurrence of aberrant miRNA or abnormal levels of miRNA is linked to several human diseases such as schizophrenia, psoriasis, diabetes and obesity[1]. Dysfunction of miRNA can be observed in Gliomas, a type of brain tumor, categorised into four different grades, based on their malignancy (I,II,III,IV). MiRNAs with pro-oncogenic properties, miR-21 and miR-23a, and anti-oncogenic properties, miR-7 and miR-137, were observed to increase and decrease respectively in glioma with increasing severity [11]. MiR-650 was found to increase in invading glioma cells and is therefore a suggested prognostic indicator for glioma[21]. About 70% of known miRNAs are observed to be expressed in the brain, but are often not brain specific. The complexity of the brain and the abundance of miRNA suggests another regulatory layer and indeed studies have shown that microRNAs are involved in development and function of the brain and controlling neuronal proliferation [22].

2.1 MiRNA biogenesis

MiRNAs are encoded in introns and exons of protein coding or non-coding genes (such as long non-coding RNAs; lncRNA), and intergenic regions. A sequence of DNA is transcribed by polymerase II and III to pre-mRNA and contains intronic and exonic regions. Introns are spliced out by the splicosome [7] as seen in Figure 2.1 and as previously mentioned can form primary transcripts miRNA (pri-miRNA).

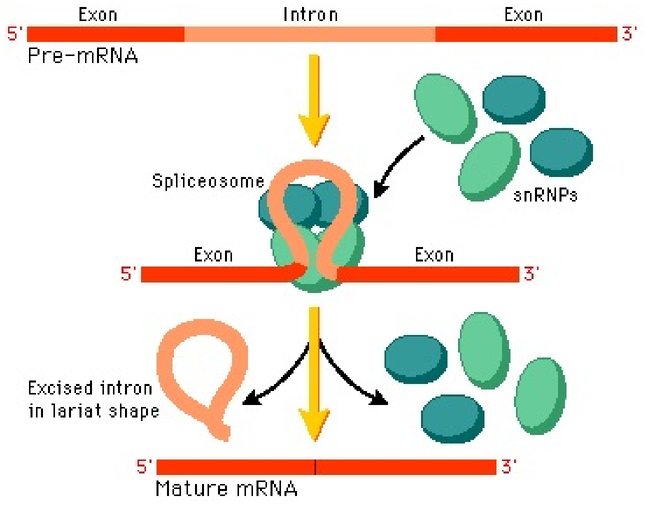

Figure 2.1: Splicing of pre-mRNA to mRNA [3]

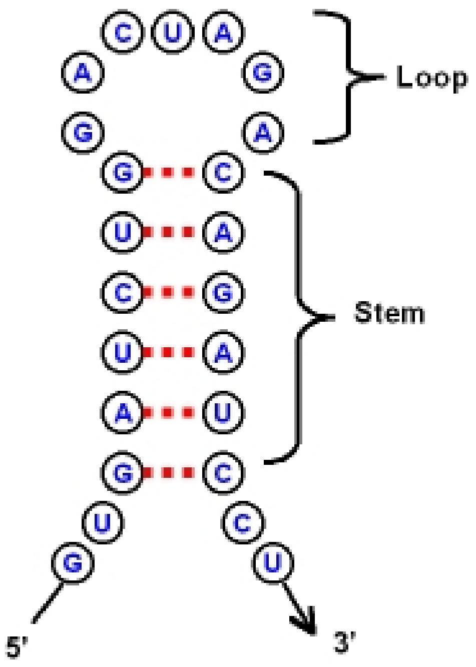

Figure 2.2: Stem loop structure [4]

Pri-mRNA has a length of several kilobases and still possesses a 5' CAP (start of intron, with a guanine) and a 3' polyadenylated tail (end of intron, with several adenosine monophosphates). It is looped into a stem loop of which a short representation is shown in Figure 2.2. Nuclear cleavage takes place by the Drosha RNase III endonuclease enzyme. Endonuclease indicates an enzyme that cuts within a sequence of nucleotides whereas exonucleases cut either at the 5' or the 3' end. Drosha is guided by binding protein DGCR8 [10], which recognises the double stranded RNA (dsRNA) of the stem loop, and cleaves its overhang, leaving a staggered cut with a 5' phosphate at one end and a 2 nucleotide 3' extension at its other end [6][14], and a resemblance to a hairpin in its structure. It's now referred to as the precursor miRNA (pre-mRNA) with about 70 to 100 nucleotides and exported to the cytoplasm by Exportin-5 and Ran-GTP for further processing [17]. In the cytoplasm the loop of the pre-mRNA is cleaved by another RNase III endonuclease, Dicer, leaving a dsRNA or by an Argonaute protein, Ago2, an essential component of the RISC complex discussed in more length in the next section. In the case of Dicer, cleavage is guided by another binding protein TRBP (trans-activation response RNA-binding protein) [13], allowing Dicer to cut both ends two helical turns away from the base of the stem loop [5]. The remaining dsRNA contains the mature miRNA with a length of 19 to 25 nucleotides and its opposing arm, labeled miRNA*. The miRNA:miRNA* is short lived and brakes down to two separate single stranded RNAs. miRNA* is then further decomposed which can be deduced from its much lower frequency in libraries* as shown in [15]. An overview of the full process is given in Figure 2.3

*In this context libraries refer to the prepared sample, ready for sequencing. RNA sequencing is a very fragile process since it is readily decomposed by proteases, commonly found in the surrounding. Out of that reason RNA is first converted to it's complementary cDNA, a much more stable compound for sequencing.

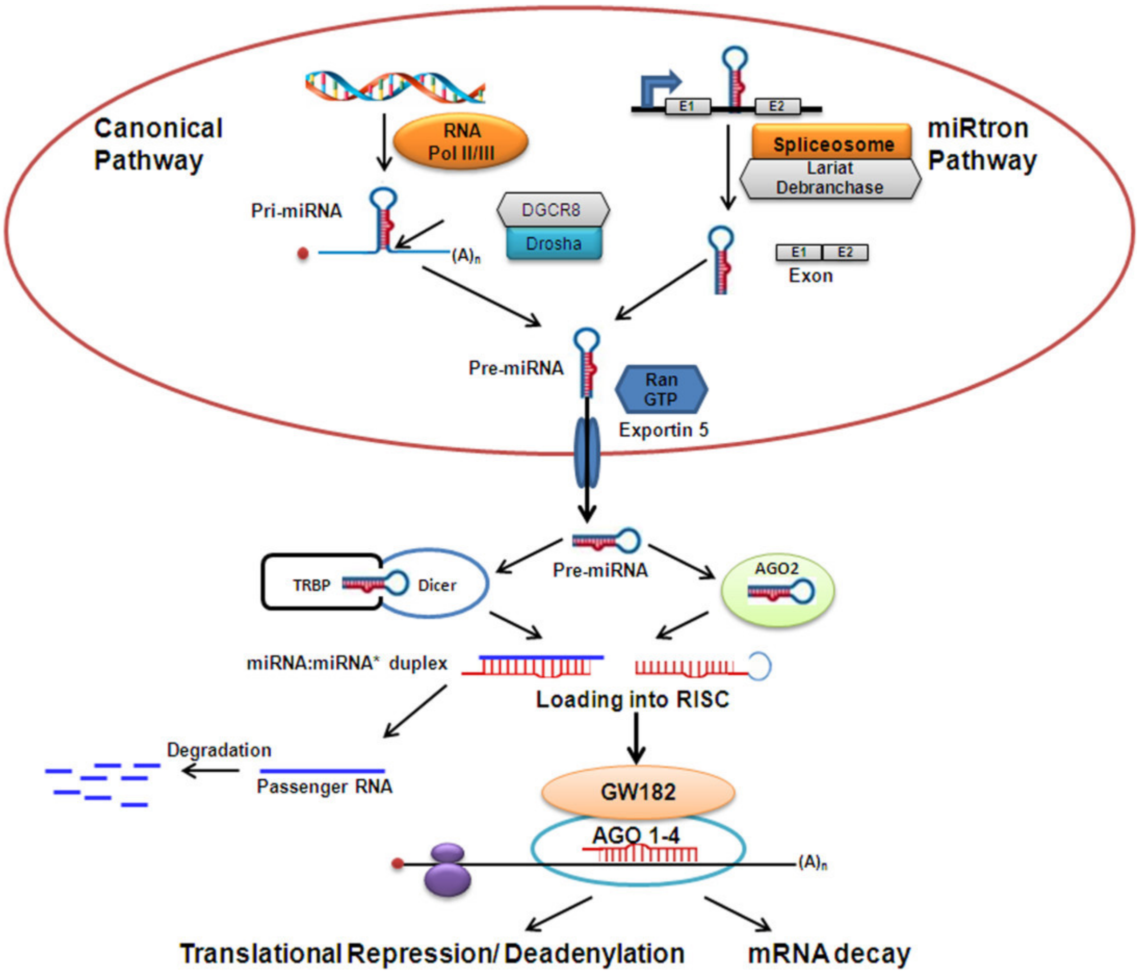

Figure 2.3: MiRNA biogenesis pathway and function [1]

2.2 MiRNA mechanism

MiRNA primarily down regulates protein coding genes by degrading the protein at the translational stage, such as through cleavage of mRNA, sequestration of mRNA into P bodies from ribosomes and translational repression although other mechanisms are also suggested such as transcriptional silencing [16][18][5]. MiRNA is loaded onto a RISC protein with the aid of Argonaute (AGO) proteins and glycine- tryptophan proteins, forming the miRISC compound [23][12]. In the case of cleavage, miRISC will bind to the 3'UTRs of mRNA and cleave the protein. The mRNA fragments are then further degraded. There have been reports about binding to 5'UTRs and the mRNA coding regions as well. For translational repression, miRISC also bind to the 3'UTRs in the mRNA. These will inhibit ribosomal translation or immediately degrade the newly formed polypeptide [5].

3 MiRNA data

MiRNA data was obtained from the Pharmacogenetics lab in OSU. It contained precursor miRNA data as well as miRNA data. It was collected from ten patients, five non smokers, five non smokers. For each patient ten brain regions were analysed of which the following nine were important for miRNA analysis BA10, BA22, Insula, Amygdala, Hippocampus, P. Putamen, Raphe Nucleus, Cerebellum. Further information about the data set can be found in [?], where an overview of all the collected RNA is given.

4 Mathematical Method

The mathematical methods implemented to analyse the overlap of species in a population such as miRNA in a human population are primarily covered in [?].
Consider n populations c_1, c_2, c_3, ...c_n, corresponding to the individual brain regions analysed for each patient in the miRNA data. There were ten regions analysed for 10 patients, yielding a hundred populations (n=100). Each population has m species associated with it. For the miRNA data, species refers to miRNAs detected per brain region per patient. Note that the data can also be transposed, allowing the analysis of the miRNA populations with different brain regions per patient as species. The overlap of each column with each other column is computed using an overlap index, i.e. column c_1 vs c_2. An overlap index, calculates the similarity between two data columns. This can be done by the widely accepted Jaccard index (JI) and Sorensen index (LI), widely used in ecology and immunology. Both are very similar but the Sorensen index can be consider to be a semimetric version of the Jaccard index. First consider the pair of populations (c_1, c_2), J and L are given by:

$$JI(c_1, c_2) = \frac{\sum min(c_{i,1}, c_{i2})}{\sum(c_{i1} + c_{i2}) - \sum(c_{i1}, c_{i2})} \tag{4.1}$$

$$LI(c_1, c_2) = \frac{2\sum min(c_{i,1}, c_{i2})}{\sum(c_{i1} + c_{i2}))} \tag{4.2}$$

Next consider the Renyi divergence for normalised populations.

$$RI(c_1, c_2) = \frac{2\sum \frac{c_{k1}}{\sum c_{i1}} \frac{c_{k2}}{\sum c_{i2}}}{\sum_k \left(\frac{c_{k1}}{\sum c_{i1}}\right)^2 + \sum_k \left(\frac{c_{k2}}{\sum c_i 2}\right)^2} \tag{4.3}$$

Two population columns can also be considered as vectors, having direction and magnitude. A large angle between the two vectors (hence pointing in different directions), indicates dissimilarity between the populations columns and therefore small overlap between the populations based on their species content. The Morisita-Horn index uses the cosine of the angle between the standardised population vectors p_1 and p_2 and is given by:

$$MH(p_1, p_2) = \frac{2p_1 p_2}{p_1^2 + p_2^2} \tag{4.4}$$

Unfortunately Equation (4.4) is very sensitive to frequent species within the columns. Therefore data with rare species is not well suited for the Morisita-Horn index.

The I-index is the most suited for measuring the overlap across multiple populations such as in the miRNA data. It measures the strength of dependence between marginals P_0 and P^0 of the population table C.

$$INP(C) = 1 - \frac{H_1(P_0) + H_1(P^0) - H_1(P)}{H_1(P_0)} \tag{4.5}$$

where $P_0 = (p_{01}, ...p_{0n})$ and $P^0 = (p_{10}, ...p_{n0})$ are elements of the normalised population table. H_1 stands for the Shannon entropy given by

$$H_1(p) = -\sum p_i \log p_i \tag{4.6}$$

Note that although overlap is computed across multiple population in INP, its input still only requires two population columns at a time of the population table.

5 Coding

Coding was the main focus of this project, implementing the mathemtical methods for diversity overlap as stated in Section 4 and improving on the already existing code divo.R"[19]. So far divo.R was implemented in R and then transferred from R to the Python language. This was done in order to improve on speed from a much slower implementation purely in R. The user does not directly interact with Python but calls Python through a complicated and time expensive interface Rcpp and Rcppnpy. In C extensive use of C pointers to arrays and pointer to row of pointers to arrays was made. A pointer saves stores an address, the address in turn is pointing to the first element of an array. Since every array element is saved in order, one can access each element of an array by advancing the pointer. Knowledge of heap and stack was used to create and free memory efficiently. A table of all custom R functions used is given below

Library	Source	Used for
nrutil.h	Numerical Recipes	Allocating mathematical structures in memory with pointer to pointer structure
Rdefines.h	R standard library	Import for R specific objects, includes Rinternals.h
R.h	R standard library	Required for any R to C extension
Rmath.h	R standard library	Includes standard math.h library and rmultnom.h
string.h	C standard library	strcmp.h for comparing strings
stdlib.h	C standard library	For memmory allocation
stdio.h	C standard library	printf.h for printing variables to the screen

5.1 R extension to C

The R language is written to 50% in C, 30% in Fortran and 20% in it's own language. This is done since C and Fortran are much faster at processing data than R. An example is the rmultnom function, that draws random numbers that are obeying the multinomial distribution in R. When executing this command, R will call a C file called "rmultnom.c", which will do the computation and then dump the result to R. The user will not realise that C was called and used to compute the output, which makes the whole process user friendly. Due to the large amount of R functions written in C, a very efficient interface or R extension was written for C. Allowing variables, arrays and other R objects to be passed to C. The library used for this extension is provided by R, when installing R. A list of all custom libraries[†] The documentation of these functions is sparse and only overviews of the function names were found. The R extension manual[9] is the basic manual for any extension such as Fortran, Cpp, C, Python, Java and further but is lacking sometimes in useful examples. The advantage of importing data from R to C compared to R to python, is that no actual data is passed. Only a pointer, pointing to the first element of the data structure, such as a string (string means an array of characters, i.e. a word), vector or matrix is passed to C. This is highly efficient since only the pointer and therefore the variable storing the address of the pointer has to be passed to C instead of the whole data structure. To fully understand, consider a pointer of type "double", which can store an address for a double value such as any decimal number with up to 15 decimal places. The pointers size in memory is 4bytes (depended on system), a very small amount of memory. Comparing this to importing the actual data matrix which can have thousands of entries, each with a double value of size 8bytes. This would take a long time to import since memory has to be allocated and then the values transferred, one at a time. This is done for the R to Python extension, with an additional caveat. The data structure first has to be stored in storage (not RAM but i.e. hard drive), by saving the data in a file on R and then reading this file into Python. This process is slower by up to a magnitude as compared when only storing in RAM. But also the increased efficiency of using C from R has a caveat, concerning the usage of the imported variables. Since a pointer is passed to C and not the actual data, any change to the data will result also in a change in the variable in R. This is not an issue, one simply has to remember that any change should be saved in a new location, if saved at all.
An abstract of an R imported variable is given in Listing 1.

```c
//Reading in X variable—————————————————————————
//Protect variables so that R doesn't delete them
PROTECT(X = AS_INTEGER(X)); nprot++;
//Defining pointers and assigning them to variables
int* ptrX; ptrX = INTEGER(X);
//Defining pointers and assigning them to variables
SEXP dimX; PROTECT(dimX = allocVector(INTSXP,2)); nprot++;
int* ptrdimX; ptrdimX=INTEGER(dimX);
ptrdimX[0]=INTEGER(GET_DIM(X))[1]; ptrdimX[1]=INTEGER(GET_DIM(X))[0];
//length of X array
int nX; nX = ptrdimX[0] * ptrdimX[1];
```

Listing 1: Importing R objects

Note that each imported variable and created array has to be protected in C from R's garbage collector and later unprotected before returning to R. The imported objects from R are labeled as SEXP, which stands for S-EXPression, the precursor to R. These objects contain the address to the first element of the array and their dimensions in case of a matrix.

[†]Here library means a collection of c files that are clustered into one library that one can download.

miRNA \ ID	1_1	1_2	1_4	1_5	1_6	1_7	1_8
pre-miRNA 29b-1	13409	6340	1526	7924	2186	6908	5026
miRNA 29b-3p	12931	5804	1314	7103	1666	5873	3989
miRNA 103a-3p	8765	8214	5936	14405	6586	10964	10887
miRNA mir-138	6947	7972	1862	8135	4763	13139	3167

Table 5.1: Sample Table data

5.2 Structure

Divo.R was fully restructured for speed and memory efficiency. The I-index function (INP) will serve as an example, it measures the overlap between populations as stated in the math section. In the case of the miRNA data, consider humans and their brain regions (population) with their corresponding measured miRNAs level (species)[‡] as given in the sample table Table 5.1.

The data is first read into R and then passed to C. In C a pointer is assigned to the first element of the data. In case of table Table 5.1 corresponding to "13409". R saves it's variables column wise, hence meaning that in memory the end of the first column is joint to the start of the next column. The data is saved in memory as represented in Figure 5.1.

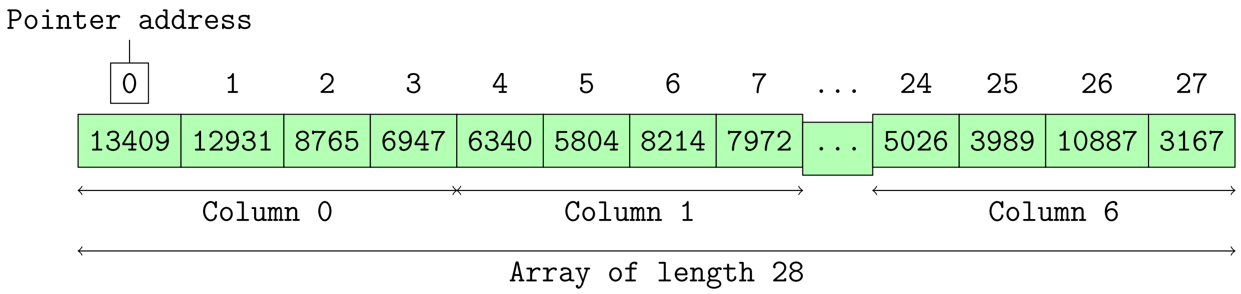

Figure 5.1: R Matrix saved in memory

At first glance, this seems non beneficial since C usually saves multi dimensional arrays row wise when initialised in C and not column wise. A possible solution is to transpose the matrix. But this requires the reallocation of each data point to a new address with two for-loops, a time consuming process. Instead, realising that when performing I-index analysis (or any other analysis, i.e. MH or JI) that only two columns are compared at a time, concludes that the structure is beneficial for the analysis. A very useful pointer to pointer to column structure was

[‡]Note that this can also be inverted, with miRNAs as population and human brain regions as species

adopted. This allows access to the first element of each column and therefore the possibility to select columns individually as seen in Figure 5.2.

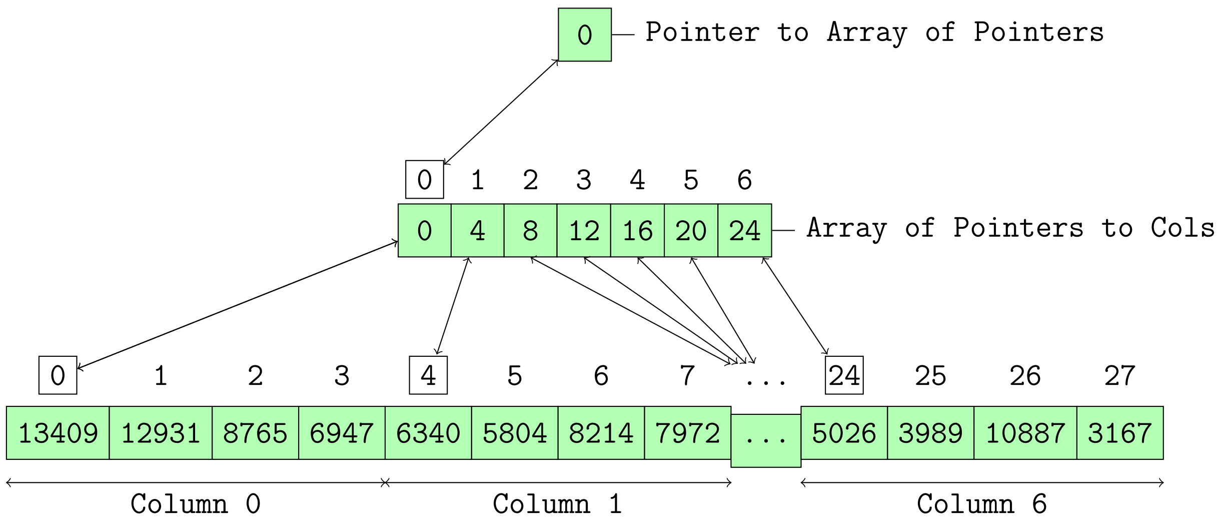

Figure 5.2: Pointer to Pointer to Column

The structure allows one to select one column at a time without unnecessary data allocation. The structure of pointers access the same memory slots as initially allocated in R. This is very efficient and a column can now be passed to a function via its pointer. Next a resample plugin is implemented, which allows the data to be resampled. This is done by computing the probability of each population value and then drawing them randomly from a multinomial distribution. The probabilities are evaluated by summing the whole data and then dividing each value by the total sum. This is taken as input for the "rmultinom.c" function, part of the "Rmath.h" library, which will redraw the data values. Resampling has the benefit that for a large numbers of resample runs (min. of 500 with no upper limit) the upper and lower quantile can be computed, showing if the clustering is stable, where clustering is the final result of the I-Index function or any other analysis function.

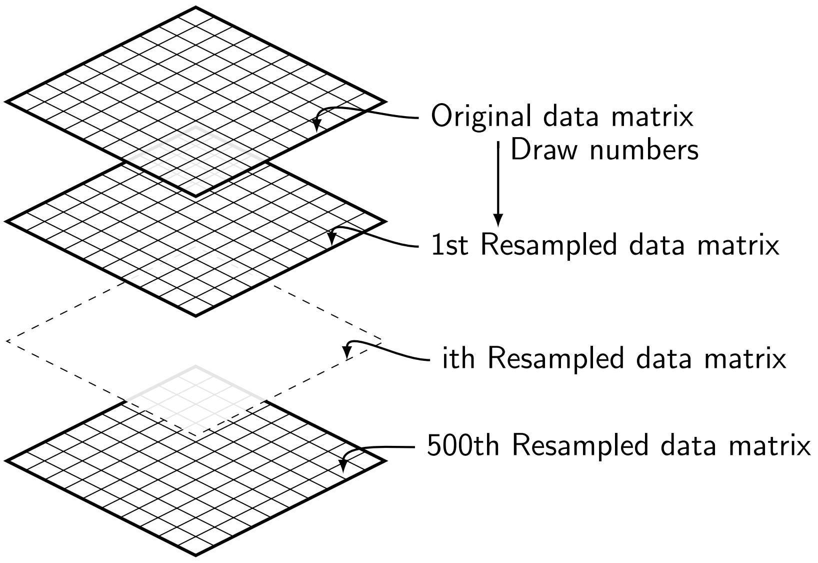
Original data matrix
Draw numbers
1st Resampled data matrix
ith Resampled data matrix
500th Resampled data matrix

Figure 5.3: Resampled data matricies

Figure 5.3 shows the resampled data matricies. Since each column of a matrix is analysed one at a time and each matrix individually, it is not necessary to save all matricies at the same time. For using as little memmory as possible, a single data matrix is drawn (or resampled), then analysed and it's result saved before a new data matrix is drawn. Each new matrix is saved in the same memory location as the first matrix by overwriting it.
How does this compare to the previous version of divo and it's python implementation? After a data file is read into python the values are transposed to bring them into the correct order. Then all matrices are drawn at the same time, yielding a 3D matrix. This is inefficient in regards to speed and memory.
Making use of the new implemented data structures, individual columns of a single data matrix can be selected and analysed using any of the analysis functions. Each column has to be compared with each other column. This is computationally intensive analysis, since any analysis performed for two columns has to be repeated for every possible combination of columns times the amount of resampled matrices. For a matrix of four columns, there are six possible column combinations for analysis $[1,2][1,3][1,4][2,3][2,4][3,4]$, if then resampled 500 times it will result in 30.000 analysis' that have to be performed. Most realistic data sets compare far more than four columns. The miRNA data has roughly 400 columns to be analysed. According to the binomial coefficient and 500 resamples this would accumulate to:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \Rightarrow \quad \binom{400}{2} \times 500 = 39.900.000 \tag{5.1}$$

Now considering that each analysis between two columns by the I-index function or any of the other analysis functions will take several internal steps, will result in an even greater number of operations that have to be performed. Hence an efficient C implementation is critical as genomic data is increasing in scale.

Two pointers, pointing to two different columns within a data matrix are passed to any analysis function such as I-index. Each of the functions after analysis of the two input columns will return a single value, a measure of how similar both columns are (total similarity of species in the population). Each of these values has to be placed into a column×column table. A sample of such a matrix is given below.

Column	0	1	2	3	4	5	6
0	0	V_0	V_1	V_2	V_3	V_4	V_5
1	0	0	V_6	V_7	V_8	V_9	V_{10}
2	0	0	0	V_{11}	V_{12}	V_{13}	V_{14}
3	0	0	0	0	V_{15}	V_{16}	V_{17}
4	0	0	0	0	0	V_{18}	V_{19}
5	0	0	0	0	0	0	V_{20}
6	0	0	0	0	0	0	0

Figure 5.4: Column table

The matrix in Figure 5.4, is half filled with zeros and half filled with V's indicating that values from an analysis function will be placed into these cells. Row index 1 and column index 2 represents a comparison between column 1 and column 2. Row index 1 and column index 2 would yield the same data as row index 2 and column index 1 (since comparing column 1 with 2 is the same as comparing column 2 with 1) and is therefore excluded and filled with a zero. Figure 5.4 will be the final structure returned from C to R. Therefore it is important to fill this table as efficient as possible when using many resampled matrices. Each resampled matrix can be grouped into a table such as Figure 5.4. All column tables are then added at their corresponding cells and averaged. This will yield the mean values in each V cell node and the final output column table. Additionally two other column tables can be extracted, corresponding to the lower and upper quantile, representing the values that diverge the most from the mean table in the lower and upper limit[§].
Instead of grouping each resampled matrix into a column table and then averaging, a more memory efficient array arrangement was used. All values from every analysed resampled matrix is saved in a long array and therefore all zero values can be left out, saving half of the memory.

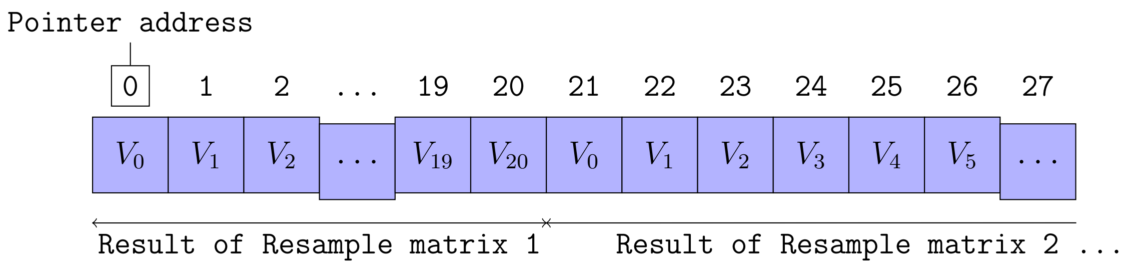
Pointer address
0 1 2 ... 19 20 21 22 23 24 25 26 27
V_0 V_1 V_2 ... V_{19} V_{20} V_0 V_1 V_2 V_3 V_4 V_5 ...
Result of Resample matrix 1
Result of Resample matrix 2 ...

Figure 5.5: Results from Resample matrices

All V_0 values from each resample matrix result can then be added and averaged. This is repeated with every V value. The result will be an array containing the mean values of all V values. The array is then filled into a column table and returned to R.

6 Results and Discussion

6.1 Validation

The new C implementation was tested against a dataset prior analysed with divo.R and the python extension. The data set covered naive and regulatory T-cell receptors from different tissue types in a specific mouse model. The different populations were Naive and Regulatory T-cell receptor type from colon, thymus, mesenteric lymph node and peripheral lymph node. The analysis yielded the same data and an identical dendrogram Figure 6.1 as in [8], validating the correctness of the code.

6.2 Speed improvement

A 1.3 GHz Intel Core i5 and 8GB of RAM computer was used to analyse the TCR data with eight columns and 6208 rows. To test the improved speed the data was appended to itself seven times, resulting in a 6208 rows to 1024 column. No resampling was performed. Hence yielding

[§]If all 3 column tables give rise to widely different plots in post-processing then this is an indication for data that is similar and will form clusters due to small variations. Care has to be taken for such data sets.

523,776 analysis operations. The run time analysis gave: 137.2 seconds, which is marginally above two minutes. Compared to the python version which was aborted after 2hrs of run time. This clearly shows how important good code design is and C compared to high level languages.

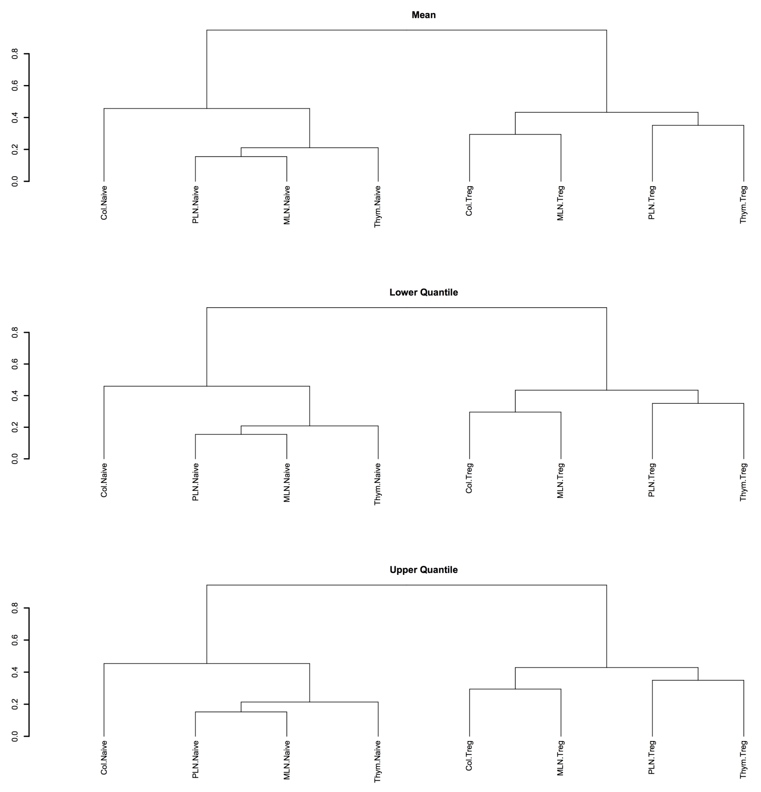

Figure 6.1: TCR data dendrogram

6.3 miRNA

The miRNA data was analysed with the I-index to cluster similarity between the populations of patients across brain regions and resampled 500 times to test for stability. The data had 406 columns for analysis resulting in 41,107,500 analysis operations. This took about 20 minutes to run on a laptop with 1.3 GHz Intel Core i5 and 8GB, using the newly implemented version of divo.R. The output column tables for the mean, lower quantile and upper quantile were plotted in R using the "hclust" function to graph a dendrogram Figure 6.2. The y scale indicates distance, increase indicating dissimilarity between populations. Clusters are formed between co-expressed populations. hence forming clusters between species that are closely co-expressed.

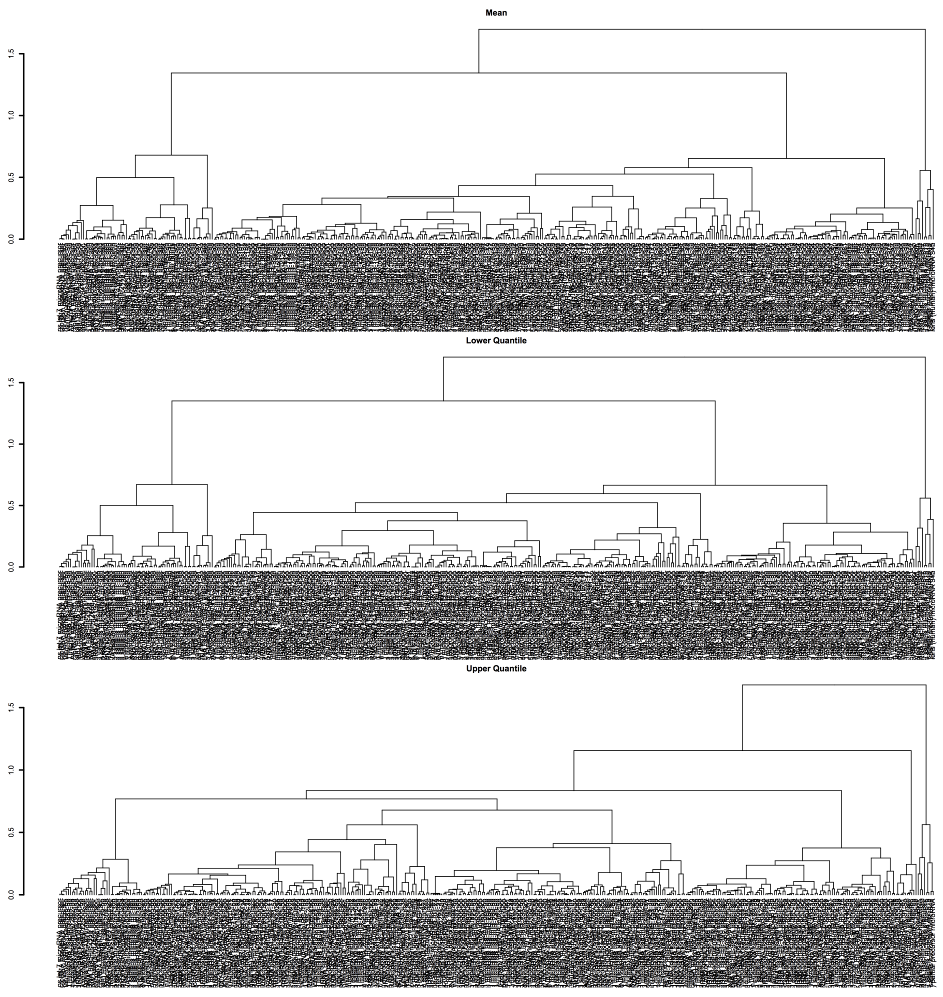

Figure 6.2: miRNA, mean, upper and lower quantile for 500 resamples

Note that the clustering between upper and lower quantile majorly change in Figure 6.2, and hence are diverging. This can mean that small changes (introduced to resampling) can alter clustering and should therefore be carefully monitored. Indicating that difference between species in the population is small.

For further analysis precursor miRNA was excluded and the miRNA expression data reanalysed with the I-index and no resamples, Figure 6.3.

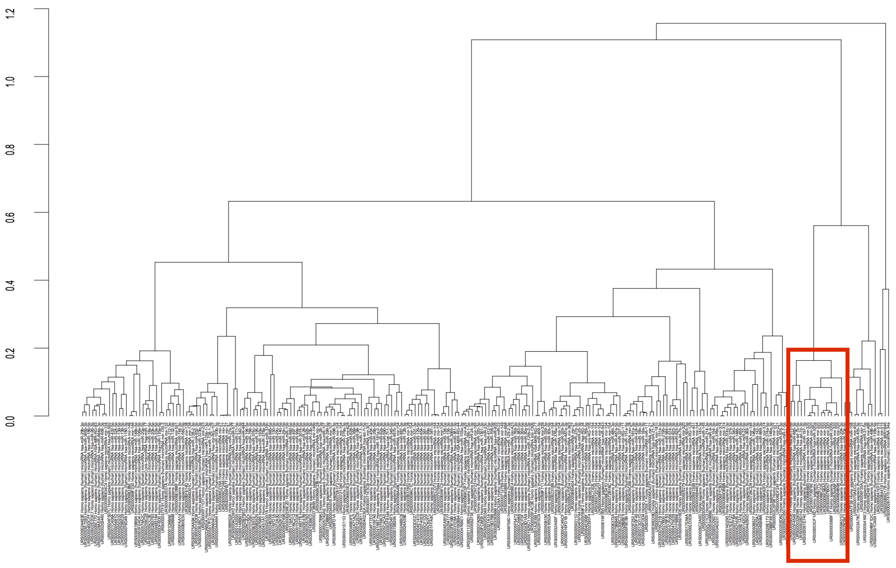

Figure 6.3: MiRNA clusters, URS-es are different database IDs referring to the same miRNA.

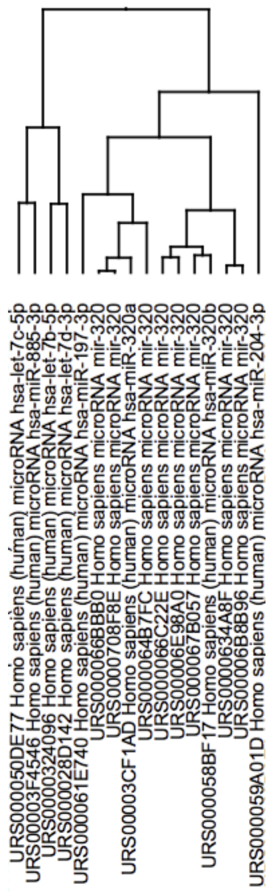

Figure 6.4: MirRNA sub-cluster

A small subsection of Figure 6.3, having a distinct cluster with a large separation to other clusters, was chosen and replotted in Figure 6.4. The cluster contains three distinct miRNAs. MiRNA-197-3p, miRNA-320 and miRNA-204-3p. Each of the three miRNA has several hundred recorded mRNA targets, however only 4 are shared among these miRNAs. MED7, NABP1, ONECUT2, PAPD5. These were obtained using Targetscan[2]. Of interest among the three miRNAs in this cluster is miRNA-320, which is differentially expressed in stroke patients and a rat model of ischemia[20]. In vitro analysis shows this relationship may be attributable to regulation of aquaporins, that control movement of water into the cell. This miRNA has been suggested as a possible drug target for treatment of stroke. For all 100 targets of miRNA-320 the R package biomaRt was used to identify the associated GO IDs. Among the top ten most frequent GO IDs was nervous system development. MiRNA-320 is clearly associated with nervous system physiology and disease. Understanding miRNA clustered with miRNA-320 could provide important insight into neurological disease. Further analysis is required.

7 Conclusion

Divo.R was improved in several ways. It's structure was adapted to be more memory efficient, allowing it to handle large data tables. The total amount of operations per analysis function was reduced and all unnecessary data transformations in the python structure excluded. The speed improvement is by a factor of 60, from 2hrs to 137 seconds, allowing the usage of large data sets. Clustering performed by divo.R can be used to add additional miRNAs to already recognised miRNAs as was found with miRNA-197-3p and miRNA-204-3p with miRNA-320. MiRNA-320 was shown by in-vitro analysis as a potential drug target for stroke. Future work is required whether the two additional miRNAs provide additional insight into stroke or serve as drug targets.

8 Afterword

The project at Ohio State University was exciting although hard. Coding in a low level language such as C with only minor prior knowledge in C++ was very time consuming. C does not provide many build in functions, adding to its difficulty. The package will be soon ready to be send to CRAN for analysis. If accepted I will be mentioned as co-author of the divo.R package, a great outcome for such a short project. Possibly the miRNA data analysed, will contribute towards a paper.

References

[1] Yogita K Adlakha and Neeru Saini. Brain micrornas and insights into biological functions and therapeutic potential of brain enriched mirna-128. *Molecular cancer*, 13(1):1, 2014.

[2] Vikram Agarwal, George W Bell, Jin-Wu Nam, and David P Bartel. Predicting effective microrna target sites in mammalian mrnas. *Elife*, 4:e05005, 2015.

[3] Unknown Author. Splicing of pre-mrna to mrna. http://www.phschool.com/science/biology_place/biocoach/transcription/images/eusplice.gif.

[4] Unknown Author. Splicing of pre-mrna to mrna. http://www.mikeblaber.org/oldwine/BCH4053/Lecture21/stemloop.jpg.

[5] David P Bartel. Micrornas: genomics, biogenesis, mechanism, and function. *cell*, 116(2):281–297, 2004.

[6] Eugenia Basyuk, Florence Suavet, Alain Doglio, Rémy Bordonné, and Edouard Bertrand. Human let-7 stem–loop precursors harbor features of rnase iii cleavage products. *Nucleic acids research*, 31(22):6593–6597, 2003.

[7] JM Berg, JL Tymoczko, L Stryer, and GJ Gatto. Biochemistry (7ma. ed., p. 773), 2012.

[8] Anna Cebula, Michal Seweryn, Grzegorz A Rempala, Simarjot Singh Pabla, Richard A McIndoe, Timothy L Denning, Lynn Bry, Piotr Kraj, Pawel Kisielow, and Leszek Ignatowicz. Thymus-derived regulatory t cells contribute to tolerance to commensal microbiota. *Nature*, 497(7448):258–262, 2013.

[9] CRAN. Writitng r extensions. https://cran.r-project.org/doc/manuals/r-release/R-exts.html.

[10] Jinju Han, Yoontae Lee, Kyu-Hyun Yeom, Young-Kook Kim, Hua Jin, and V Narry Kim. The drosha-dgcr8 complex in primary microrna processing. *Genes & development*, 18(24):3016–3027, 2004.

[11] PA Koshkin, DA Chistiakov, AG Nikitin, AN Konovalov, AA Potapov, DY Usachev, DI Pitskhelauri, GL Kobyakov, LV Shishkina, and VP Chekhonin. Analysis of expression of micrornas and genes involved in the control of key signaling mechanisms that support or inhibit development of brain tumors of different grades. *Clinica chimica acta; international journal of clinical chemistry*, 430:55, 2014.

[12] Jacek Krol, Inga Loedige, and Witold Filipowicz. The widespread regulation of microrna biogenesis, function and decay. *Nature Reviews Genetics*, 11(9):597–610, 2010.

[13] Ho Young Lee, Kaihong Zhou, Alison Marie Smith, Cameron L Noland, and Jennifer A Doudna. Differential roles of human dicer-binding proteins trbp and pact in small rna processing. *Nucleic acids research*, page gkt361, 2013.

[14] Yoontae Lee, Chiyoung Ahn, Jinju Han, Hyounjeong Choi, Jaekwang Kim, Jeongbin Yim, Junho Lee, Patrick Provost, Olof Rådmark, Sunyoung Kim, et al. The nuclear rnase iii drosha initiates microrna processing. *nature*, 425(6956):415–419, 2003.

[15] Lee P Lim, Nelson C Lau, Earl G Weinstein, Aliaa Abdelhakim, Soraya Yekta, Matthew W Rhoades, Christopher B Burge, and David P Bartel. The micrornas of caenorhabditis elegans. *Genes & development*, 17(8):991–1008, 2003.

[16] Jidong Liu. Control of protein synthesis and mrna degradation by micrornas. *Current opinion in cell biology*, 20(2):214–221, 2008.

[17] Elsebet Lund, Stephan Güttinger, Angelo Calado, James E Dahlberg, and Ulrike Kutay. Nuclear export of microrna precursors. *Science*, 303(5654):95–98, 2004.

[18] Ramesh S Pillai, Suvendra N Bhattacharyya, and Witold Filipowicz. Repression of protein synthesis by mirnas: how many mechanisms? *Trends in cell biology*, 17(3):118–126, 2007.

[19] Grzegorz A Rempala and Michal Seweryn. Methods for diversity and overlap analysis in t-cell receptor populations. *Journal of mathematical biology*, 67(6-7):1339–1368, 2013.

[20] Sugunavathi Sepramaniam, Arunmozhiarasi Armugam, Kai Ying Lim, Dwi Setyowati Karolina, Priyadharshni Swaminathan, Jun Rong Tan, and Kandiah Jeyaseelan. Microrna 320a functions as a novel endogenous modulator of aquaporins 1 and 4 as well as a potential therapeutic target in cerebral ischemia. *Journal of Biological Chemistry*, 285(38):29223–29230, 2010.

[21] Boqian Sun, Bo Pu, Dake Chu, Xiaodan Chu, Wei Li, and Dun Wei. Microrna-650 expression in glioma is associated with prognosis of patients. *Journal of neuro-oncology*, 115(3):375–380, 2013.

[22] Davide De Pietri Tonelli, Jeremy N Pulvers, Christiane Haffner, Elizabeth P Murchison, Gregory J Hannon, and Wieland B Huttner. mirnas are essential for survival and differentiation of newborn neurons but not for expansion of neural progenitors during early neurogenesis in the mouse embryonic neocortex. *Development*, 135(23):3911–3921, 2008.

[23] Luke A Yates, Chris J Norbury, and Robert JC Gilbert. The long and short of microrna. *Cell*, 153(3):516–519, 2013.