Department of Computer Science
Algorithmic Bioinformatics

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Developing a Snakemake Workflow for Gene Set Enrichment Analysis

Lars Meinke

Bachelor's Thesis

Beginn der Arbeit:	17. February 2020
Abgabe der Arbeit:	17. April 2020
Gutachter:	Univ.-Prof. Dr. G. Klau
	Dr. phil. A. Dilthey

Abstract

Our thesis aims to create an easy to setup and efficient workflow that performs all necessary steps required to make a Gene Set Enrichment Analysis from RNA sequence reads. All steps to prepare high-throughput RNA reads can be automated to save time. Using Snakemake as the workflow manager we also save computational resources by re-using already calculated results. Other workflows only perform differential expression analysis, which means that the last step has to be performed manually and therefore making it not useful to be done in bulk. Our solution uses parts of an existing workflow and makes the necessary changes to add the Gene Set Enrichment Analysis to the end of the pipeline. We evaluated our workflow with built in benchmarking options to show that it scales with available computational resources and also works efficiently on repeated runs.

Contents

1 Introduction

Workflows are a commonly used methodology for processing vast amounts of data from biological experiments. A Workflow should be easy to setup, reproducible by others and ideally scale with growing input size. In this work, we are going to develop a workflow to analyze RNA sequencing data with Gene Set Enrichment Analysis (GSEA). We will be using Snakemake as the workflow management system together with the package manager anaconda and the GSEA Software provided by the Broad Institute [13] to create a workflow that is easy to install, replicate and scale.

GSEA is a pathway analysis of a ranked gene list. It is a threshold-free method that analyzes genes based on differential expression rank or other scores. GSEA searches for pathways that are over or under expressed (top or bottom of the ranked list), more so than by chance alone. To do so, GSEA calculates enrichment scores (ES) for every pathway, by scanning through the gene list from top to bottom. The running ES is increased for every pathway if the gene is part of the pathway and decreases otherwise. The ES is then weighted such that enriched genes in the top and bottom are amplified. It is then normalized relative to the pathway size, resulting in the normalized enrichment score (NES). Lastly, a permutation-based P value is computed based on a seed which is used to produce a permutation-based false-discovery rate (FDR) Q value that ranges from 0 (highly significant) to 1 (not significant).

The workflow can be downloaded from GitHub. In this thesis we will be referencing version `https://doi.org/10.5281/zenodo.3754598`.

To evaluate the workflow we used the data gathered in the study "RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells" [5].

2 Snakemake Workflow

2.1 Preliminaries

2.1.1 Conda

Conda [12] is a package and environment manager that makes it possible to install different sets of software and switch between them. Those sets are named environments. All of those are separate from each other, meaning, every one of them could have the same software installed in either the same or a different version. In our workflow, we define configuration files which will automatically create a new environment and install the given programs in the specified version. Conda can also install libraries - for example, R modules that would otherwise need to be installed with an R script. This makes the whole workflow easy to setup as many required programs get installed automatically and, by using the correct versions, it guarantees compatibility and produces consistent results.

2.1.2 Snakemake

For our thesis, we use the Python-based workflow manager Snakemake [7] which works comparably to GNU Makefiles. Snakemake allows us to split the workflow into smaller parts called rules. Each rule creates one or more output files from zero or more input files. The intermediate steps are fully transparent and can be listed by executing a dry-run, even showing the commands used. All files written by all rules do not get deleted at the end of the workflow, so intermediate files can be examined or used for other purposes. Rules can be chained together, taking the output file of one rule as the input of another. This creates a directed acyclic graph (see Figure 1 and 2) which is calculated on startup and determines the order in which the rules are executed. If required, a rule can be started on multiple input files which can also be done in parallel if the resources needed are available. They will only be executed if the output files are not already present or the input files are newer than the output files. This makes the workflow more efficient as only required files are generated and already present files are being reused.

When creating workflows one big aspect is the reproducibility. Snakemake lets us define Conda environments and load them on a per rule basis with the $conda$ directive. This can be used to specify the programs required to execute the rule including a specific version. It is also possible to load environment modules e.g. for cluster systems that often offer a better performance than standard packages. Those two features handle the used software, the $singularity$ directive gives control about the used operating system by loading a docker container. This can be done globally, making the whole workflow run inside of docker, or rule based. Defining singularities and Conda environments not only makes the workflow reproducible but also quick to set up which has been the main decision point to use Snakemake in our thesis. Another useful feature is wildcards. They allow rules to dynamically name output files and accept a filename pattern for input files, rather than a fixed name. For example, the input of a rule can be $samples/sample.fastq.gz$ and the output can be $mapped/sample.Aligned.out.bam$. In this case, if the requested output is

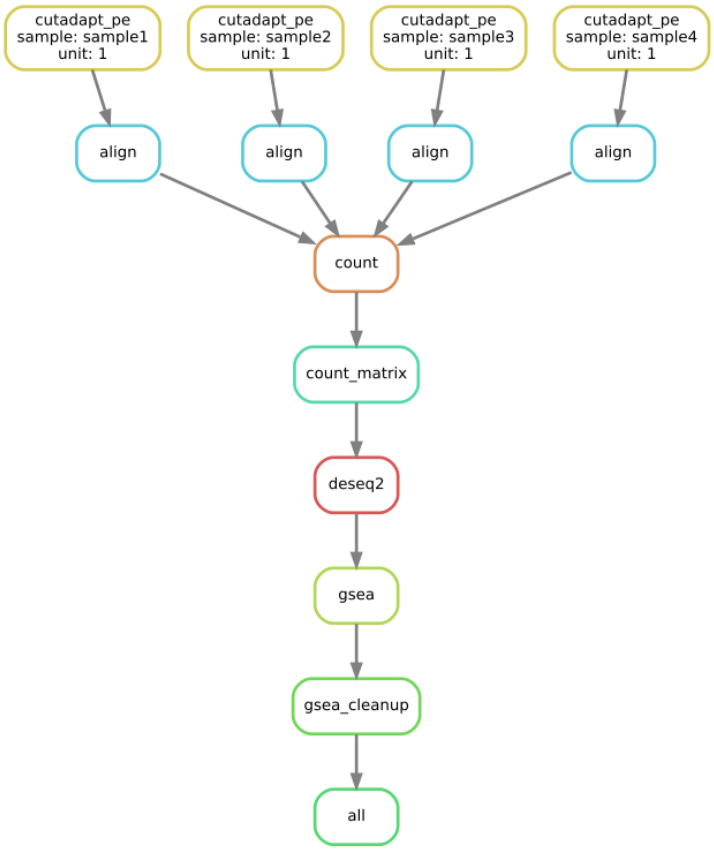

Figure 1: example Snakemake directed acyclic graph

$mapped/A.Aligned.out.bam$, Snakemake will replace the wildcard $sample$ with the value A and therefore look for the input file $samples/A.fast1.gz$.

2.2 RNA Trimming and Aligning

RNA sequence reads are stored in the FASTQ file format. It stores a quality score along with the nucleotide sequence and normally uses four lines per read. The first line always starts with a '@' character and is followed by a sequence identifier. Line two consists of only the raw sequence letters such as A, C, G, T. The 3rd line begins with a '+' character and is followed by the same sequence identifier used in line one. The fourth line encodes the quality score for the sequence in line two and must contain the same number of symbols as letters in the nucleotide sequence.

In next generation sequencing machines adapters are added to the sample RNA. Adapters are nucleotide sequences with varying length, depending on their function, used in high-throughput sequencing such as the Illumina, IonTorrent or SOLiD platform. They are required for the clonal amplification of the construct and used to initiate the sequencing reaction. Further they can be used as a barcode/index to identify samples that were sequenced in bulk with others, so not all samples belong to the same experiment.

Adapters often get added by polymerase chain reactions (PCR) but some platforms also use a ligation process to attach the adapter sequence to the insert [1]. It is important to remove any adapter or adapter fragments before other analyzing steps as that would introduce non-genome reads into the process and therefore the read alignment changes.

The first step in our workflow is to trim any adapters from the FASTQ files if present and then align the reads to a reference genome. Trimming can be turned off in the configuration file $config.yaml$ by setting the $skip$ boolean to $true$ if trimming is not skipped the used adapter sequence needs to be set.

To do the trimming we will use the program Cutadapt [11] which can remove fully or partly present 3'-end and 5'-end adapter sequences in an error tolerant way. To correctly start Cutadapt we have to distinguish between single-end and paired-end reads. The differentiation has to be made, because the forward reads might have a different adapter than the reverse reads. More importantly, Cutadapt will check that the paired-end reads are properly paired and throw an error if one of the files has more or less reads than the other. It also makes sure both files are synchronized, when a read has to be deleted in one file it will also get removed in the other. In some situations, Cutadapt can be run in single-end mode twice for both input files but this will disable any consistency checks Cutadapt would do otherwise. For single-end reads, it takes the adapter sequence with the $-a$ parameter. For paired-end reads, we additionally have to give the adapter sequence for the reverse read with the $-A$ parameter. To speed up the process, multi-core support can be turned on with the $-j$ parameter, where a value of 0 will use all available cores and any number greater than 0 will use the given count of cores. If one intends to use multi-core support and gzip compression, it is advised to have pigz (parallel gzip) installed as the compression task will otherwise throttle the other processes down. The trimmed FASTQ files are written to the filename given with the $-o$ parameter for single-end reads. The paired counterpart is written to the file from the $-p$ parameter. In Snakemake, we use a Python function to return the filename from the fq1 column and if present the fq2 column of the $units.tsv$ file where the sample and unit column match the current wildcard values. The wildcards $sample$ and $unit$ get set by the count rule. Depending on whether just fq1 is set or both fq1 and fq2 we start Cutadapt either in single-end or paired-end mode with two different Snakemake rules.

After the reads have been trimmed, they need to be aligned to a reference genome. Since we are working on human samples, we are going to use the genomic FASTA and GTF annotation files from the homo sapiens reference genome GRCh38.p13 [4]. For aligning the reads, we will be using the STAR (Spliced Transcripts Alignment to a Reference) [3] RNA-seq aligner. STAR requires the reference genome to be indexed before it can be used in the alignment process. This workflow assumes that this step has already been done with the project specific parameters, for our example, we will just specify the downloaded FASTA and GTF file along with the $--sjdbOverhang$ option. According to the manual, the $sjdbOverhang$ parameter should ideally be set to $ReadLength - 1$ or for variable length reads $max(ReadLength) - 1$. In most cases, the default value of 100 will give good results. However, since we know the read length in our study is 63, we can set the parameter to 62. In contrast to Cutadapt, we do not have to distinguish as much between single-end and paired-end reads with STAR. However, input of the align rule still depends on if trimming was done or not and if the reads are single-end or paired-end. Therefore we

use another python function, that returns the file paths depending on those conditions. In any case, the STAR command remains the same which makes it possible to cover all combinations with one Snakemake rule. The align rules output are the aligned reads, always in a single, unsorted BAM file. We have chosen the BAM over the SAM format, because it is the lossless, compressed binary version of the SAM format and therefore saves space without any disadvantages. As the align process is the most time consuming and the align rule the most repeated rule of the workflow, we also skip sorting of the BAM file. Due to the repeated execution, skipping the sorting saves a not insignificant amount of time.

2.3 Feature Counting and Differential Expression

The aligned reads can now be further processed and firstly the exons need to be counted. To do so, we will be using the $featureCounts$ function in the Rsubread [8] R package. $featureCounts$ is a efficient read summarization function that can count RNA-seq and DNA-seq reads. It needs an annotation file which can be provided in GTF or SAF format and one or more input files with aligned reads. The output is a table with each feature as a row and the corresponding count for each file in the columns (see table 1). To perform a differential expression analysis with DESeq2 [9], we need to further summarize the $featureCounts$ table. There are potentially multiple rows with the same GeneID in the count table, those need to be combined into one row for DESeq2 because they are later used as indices. To do so we use a small Python script that calculates the sum of all rows with the same GeneID, making them unique.

GeneID	GSM1275862	GSM1275863	GSM1275866	GSM1275867
DUSP1	664	5021	1213	4542
FKBP5	260	4651	380	3874
KLF15	71	1326	52	703
TSC22D3	376	4301	522	3088
PER1	179	1528	142	844
KCTD12	4750	831	4807	414

Table 1: featureCounts output shortened

DESeq2 [9] is an R package that normalizes the counts and performance, a differential expression analysis based on the model of a negative binomial distribution. Aside from other statistical values, it calculates the $\log_2$ fold change (LFC), which we can use for GSEA. DESeq2 takes the count matrix where each row represents a gene and each column a sample, the matrix entries indicate the number of reads that have been mapped. The table we created in the counting step represents exactly that. Additionally, we need the $samples.tsv$ file, so DESeq2 can map a treatment condition to each sample. After the function has processed the counts, we get a table with a row for each gene and a column for every statistical function. GSEA does normalization and shrinking of the ranked list, so we need a not normalized value, that is why we use the LFC. We therefore delete all columns except for $log2FoldChange$ which is the LFC, then sort for the LFC value and

save the table (see Table 2) to a file.

Gene_Name	log2FoldChange
DUSP1	-2.96
FKBP5	-3.95
KLF15	-4.57
TSC22D3	-3.34
PER1	-3.21
KCTD12	2.49

Table 2: DESeq2 output shortened

2.4 Gene Set Enrichment Analysis

Our last step in the workflow is to perform the Gene Set Enrichment Analysis (GSEA). The GSEA Software [13] is Java-based and comes in a GUI and a CLI version. In this workflow we will be using the CLI program, because it allows us to start the analysis with a simple, pre-defined script. It comes with different analysis modes and tools, that are also available with the CLI script, we will be using the $GSEAPreranked$ mode. In this operation mode, the program needs a pre-ranked gene list, which we created in section 2.3 as well as a gene set database and a chip platform.

Gene sets are a collection of genes, that together, create a biological pathway. They represent a chain of genes that need to be active, or that influence each other, to for example regulate metabolism in cells. Some gene sets are also responsible for diseases such as cancer. A chip file contains annotations for a microarray and maps features (e.g. probe sets) to gene symbols. While this is not directly used in the GSEA algorithm, it can be used to annotate the output and to collapse each probe set to a single gene vector. GSEA needs exactly one CHIP file and one or more gene sets, however duplicate gene sets will not be filtered out and may cause wrong statistical values in the report.

Running the software in a Snakemake workflow presents one challenge, the output folder name includes a timestamp that is determined upon start of GSEA and can not be known by Snakemake beforehand. Since the rule starting GSEA is expecting a certain directory to be created after the shell script has been executed, the rule will always fail because the folder name will not match. To solve this problem, we will be using a Snakemake featured called "Data-dependent conditional execution". This allows us to define checkpoint rules, which can only have a directory as the output and does not evaluate its contents. At such rules, the DAG is re-evaluated and a Python object is created which gives us access to all newly written file and folder names in the target directory. With this object we are able to rename the sub-folder containing the timestamp to produce a consistent output folder for possibly following rules or scripts.

2.5 Workflow Output

The main output of the workflow is a html report generated by GSEA which can be found in the folder $report/workflow.GseaPreranked$. The website can be opened locally with any browser without needing a server to do so. It consists of different html pages, tables and graphs that show the up and down regulated gene sets. The main page shows global statistics like information about the used gene and data set together with global ES and p-values vs. NES plots. There are also two phenotype sections, one for positive and one for negative enrichment scores.

The enrichment results pages show a table of all negative or positive enriched gene sets with their corresponding ES, NES, p-value and FDR q-value. For the top 20 gene sets there is an extra page available showing detailed information about the gene set. This includes a enrichment plot which shows the running ES score and positions of gene set member hits (see appendix A.1). Every list from the report is also available in a tab delimited text file and each graph is generated as a png and a svg image. This makes it easy to further process the tables or use the graphs in custom reports.

Apart from the html report, all intermediate files can also be used after the workflow is done. In particular, the generated count tables and ranked list can be used in other workflows or analysis. Furthermore, we use the $benchmark$ directive in all suitable rules to record the system usage of a rule's execution. Benchmarks get saved as a tab separated list, including run-time, memory used in Mib, harddisk I/O and mean CPU load. Memory usage is shown in four columns always referring to the maximum usage during execution. The four memory columns are RSS (Resident Set Size), VMS (Virtual Memory Size), USS (Unique Set Size) and PSS (Proportional Set Size). Harddisk I/O is split into bytes read (io_in) and bytes written (io_out). Lastly, the workflow also saves log files for each step. This helps in finding errors or checking program output.

3 Evaluation

Our workflow is similar to the rna-seq-star-deseq2 workflow in the Snakemake git repository. We used parts of that, namely trimming, aligning and DESeq2, to have a well known base for our workflow. Instead of using Snakemake wrappers, we build everything with Conda to keep the simple setup but allow for more flexibility in the commands. We changed the DESeq2 script to output a ranked list instead of the binary object and removed the step to export the complete list. From there we added our GSEA rules to create the html report. We are also using $featureCounts$ to count exons instead of STAR to have a more programmatic approach for this step. Using a programming language such as R gives us more control over the output.

3.1 Benchmark

All benchmarks and runs were performed using:

CPU Intel Core i9-7900X @ 3.3GHz

RAM 64GB DDR4-2133

Storage 2x SanDisk Ultra II SSD 960GB

Using Snakemake as the workflow manager grands a high amount of efficiency. It makes sure that only necessary files are being generated. While this is not useful on the first run, any consecutive executions of the workflow are much faster.

In our workflow the most time consuming steps are trimming and aligning (Figure 8). With our setup, these two rules took over 3 hours to complete. Since they can be run in parallel (see Figure 2) a second server, or a more powerful setup, would allow two rules to be run at the same time and therefore cut the time in half to 1.5 hours. Once a sample has been trimmed and aligned, Snakemake will re-use the aligned BAM file for future executions of the workflow so these two steps only have to be done once per sample. Since aligning uses the most RAM (33.45GB on average per execution, see Figure 9) only performing that once means re-runs of the same data e.g. to compare different conditions can be done on less powerful computers.

All rules after aligning can not be run in parallel, hence why only one server is needed. These remaining rules (counting, DESeq2, GSEA) took 30 minutes to finish with our example data (see Figure 10). Depending on which GSEA gene set one is using and how many samples need to be counted these steps might take longer, e.g. using gene set C7 instead of C2 takes 4 minutes longer. The whole workflow finished in 3.5 hours with our test setup, having the potential to save 1.5 hours when using a second server. After the workflow has finished at least once it can complete in 30 minutes or even less, if the counting tables can be re-used.

3.2 Result Comparison

To evaluate our results, we compare them as closely as possible with the paper [5] using the comparison untreated vs. dexamethasone. Starting with the read aligning we get an average of 90.07% uniquely mapped reads (range 88.54%-91.04%). That is a 6.71% increase, potentially because of the newer reference genome we used in our thesis. The increase in mapped reads and also the potential difference in annotated genes will make our count table differ from the paper and thus the ranked list. Table 3 compares our DESeq2 output with the values the paper computed with Cufflinks. As can be seen, the LFC values are close to the originals with a lower FDR. The biggest difference being C7 with a 0.13 higher LFC and a significantly higher FDR. This could be explained by the different reference genome as the FDR is calculated with the Benjamini-Hochberg approach in both DESeq2 and Cufflinks.

Gene Name	LFC	FDR	study LFC	study FDR
C7	-3.22	1.0e-05	-3.35	0
CCDC69	-2.90	4.0e-29	-2.92	0
DUSP1	-2.96	4.7e-46	-2.99	0
FKBP5	-3.95	8.4e-37	-3.95	0
TSC22D3	-3.34	2.6e-18	-3.27	2.5e-13
CRISPLD2	-2.71	1.1e-22	-2.70	6.9e-13
KCTD12	2.49	9.9e-41	2.52	1.1e-11

Table 3: LFC and FDR comparison

Looking at the results for a Gene Set Enrichment Analysis with the sample data we see, that the smooth muscle contraction pathway [10] is under expressed (Figure 6), as we would expect when treating asthma. Furthermore we see the asthma pathway [6] over expressed (see Figure 5). Checking the Q-Value, 424 gene sets are significantly under expressed with an FDR < 25% including the smooth muscle contraction pathway with 6.5%. Only 34 gene sets are significantly over expressed with an FDR < 25% with the asthma pathway at 42.7%. Given the high FDR value, the asthma pathway is probably not relevant or not fully expressed due to the medication given.

A significant gene set that is over expressed however is the "NFKB Targets Repressed by Glucocorticoids" pathway [2] (see Figure 7). This describes the use of glucocorticoids to treat autoimmune and inflammatory diseases such as asthma. Since dexamethasone, which is used in our example experiment, is an artificial glucocorticoids this article can be applied on our report. The pathway also includes, among other interleukins, IL6 which was mentioned in the example study to be elevated. Further interpretation from a biologist would be needed to evaluate the other pathways.

4 Outlook

A limitation in the workflow could be removed by reconsidering the $featureCounts$ function to count exons. The limitation here is, that when starting the $featureCounts$ function a parameter has to be set whether to count single-end or paired-end reads. Because we count all aligned RNA reads at the same time, it does not allow us to use a mixture of single-end and paired-end reads in one workflow run. We currently also do not have a way to automatically detect that for the counting step, so there is a configuration parameter that changes $featureCounts$ from single-end to paired-end mode. This could be fixed by starting $featureCounts$ for every sample instead of once per run or alternatively change to a different counting method.

The configuration could also be improved by offering more user friendly options for GSEA. This would be achievable by moving frequently used GSEA parameters to configuration variables and out of the $params$ variable. For example options such as $plot_top_x$, set_min and set_max are likely to be changed often, depending on the experiment that needs analyzing. Other options that should not be changed like rpt_label could be hard coded in the rule definition.

To further make the workflow easier to setup the "Between workflow caching" method from Snakemake could be implemented. This feature could be used to download annotation files and reference genomes. It can also create the STAR index required for the aligning process so that this step does not need to be done by hand before starting the workflow. However, this requires the maximum RNA read length to be known which the workflow should ideally compute on its own. In the future, caching might even save disk space because all annotations, reference genomes and indices are stored in a central location and being used by all workflows.

There are a couple of options to expand onto the workflow and further process the data. One could be, adding Cytoscape to the end of the workflow and generate aa enrichment map with the EnrichmentMap plugin (see Figure 11). Adding this step would show linked gene sets and leading edge genes. With DESeq2 we can create further statistical plots such as MA-plots (Figure 12) showing the LFC over the mean of normalized counts. This means, that genes with similar expression values in treated and untreated samples will be close to the $y = 0$ line, genes that are differently expressed will be further away. Genes with a adjusted p-value below a threshold (default 0.1) are shown in red.

5 Acknowledgments

I am grateful to Gunnar Klaur for making the thesis possible, even on a tight schedule. Thanks to Philipp Spohr for his excellent guidance on structure and content of the thesis.

A Appendix

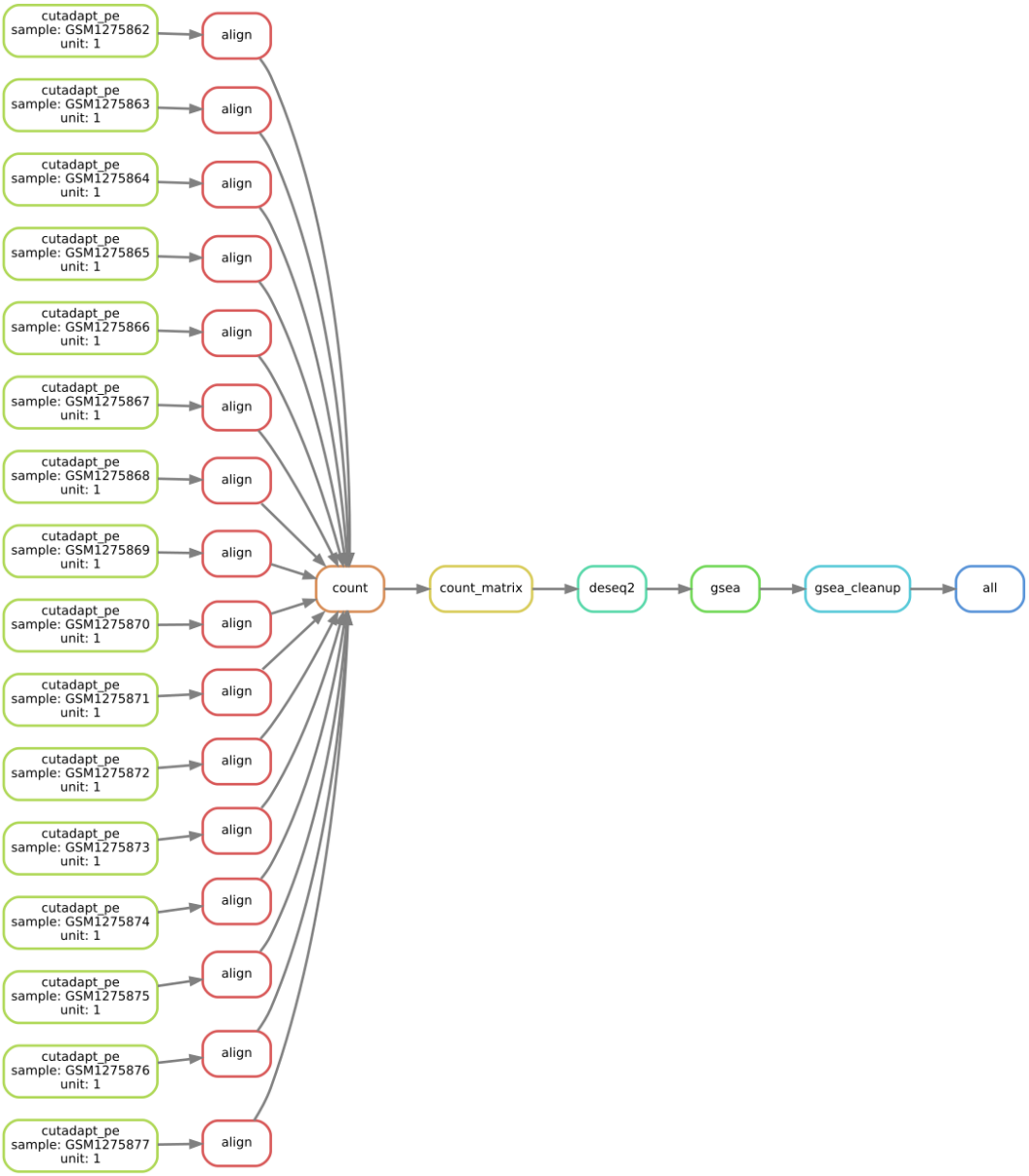

Figure 2: Snakemake directed acyclic graph from example data

A.1 Workflow Output

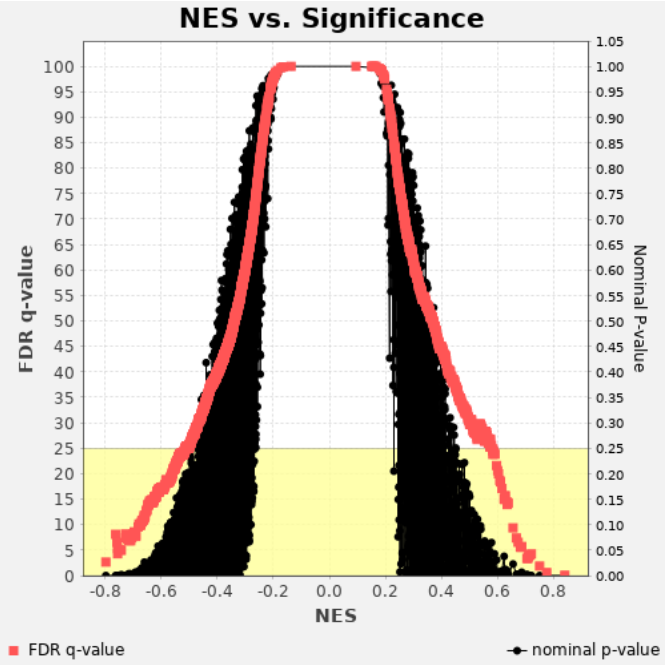

Figure 3: Normalized ES vs. FDR q-value - showing how many gene sets are relevant at a glance

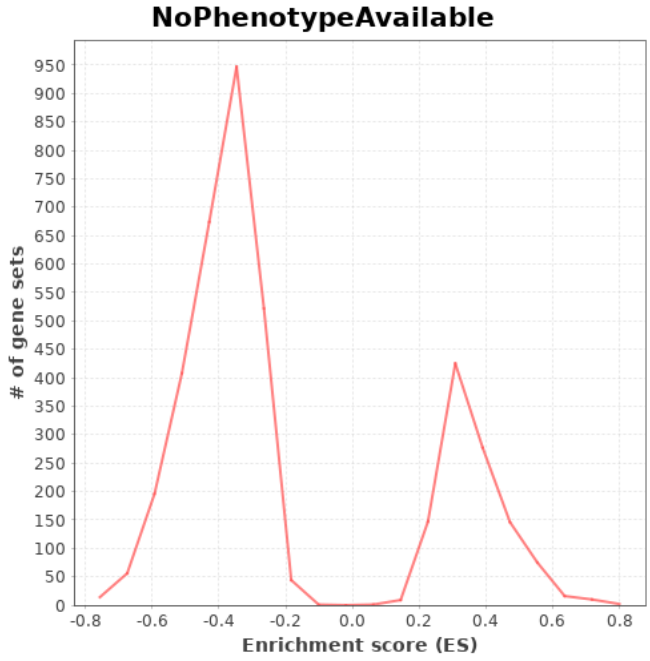

Figure 4: ES vs. number of gene sets - provides a quick, visual way to see the number of enriched gene sets

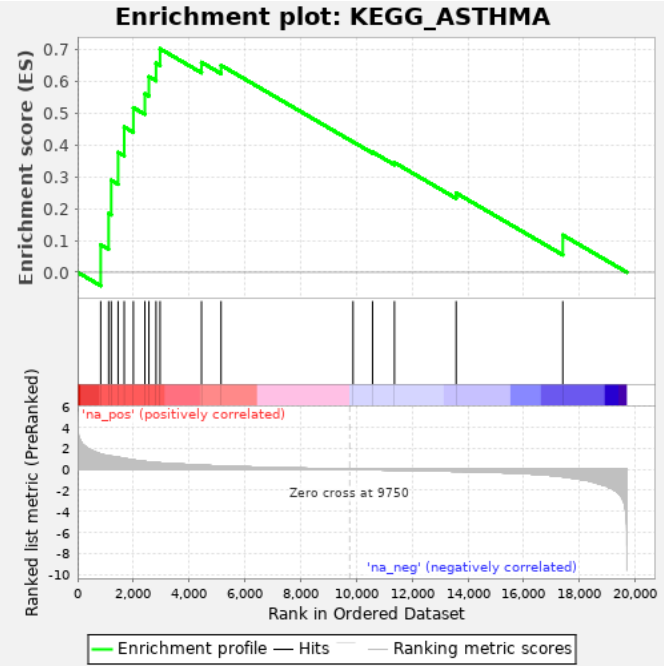

Figure 5: Top part shows the running ES, middle shows where the gene set members appear in the ranked list, bottom shows the value of the ranked metric

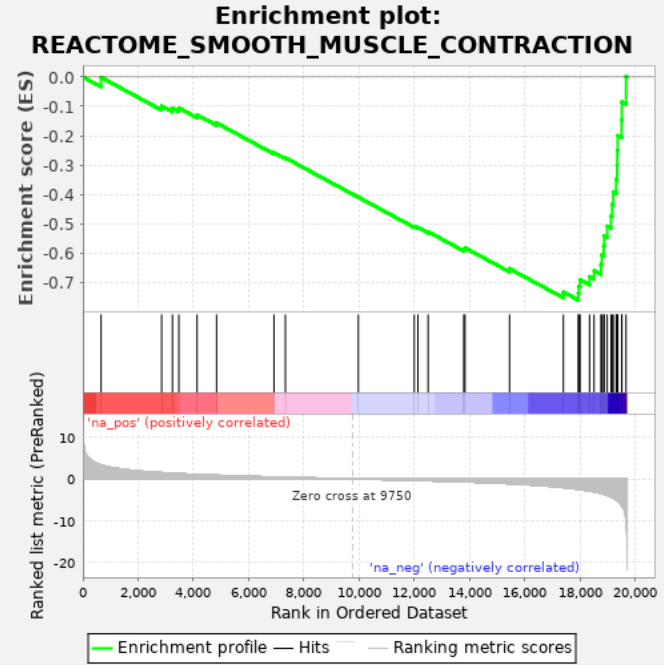

Figure 6: Top part shows the running ES, middle shows where the gene set members appear in the ranked list, bottom shows the value of the ranked metric

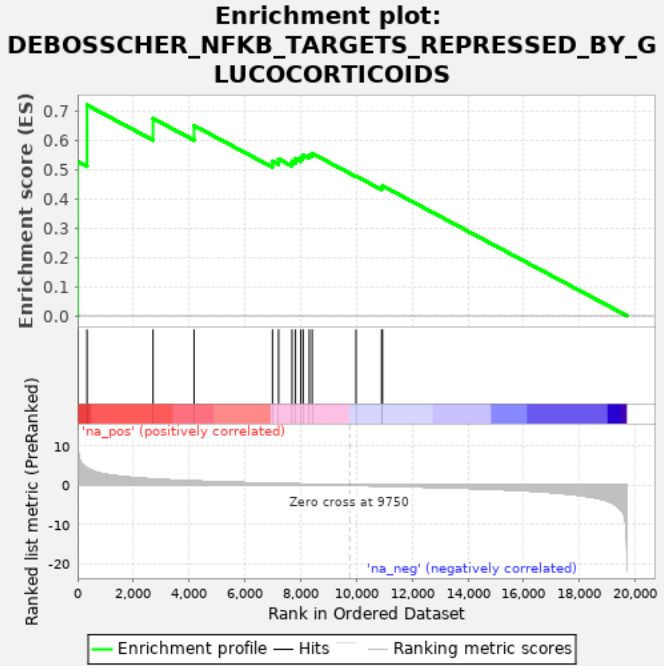

Figure 7: Top part shows the running ES, middle shows where the gene set members appear in the ranked list, bottom shows the value of the ranked metric

A.2 Benchmark

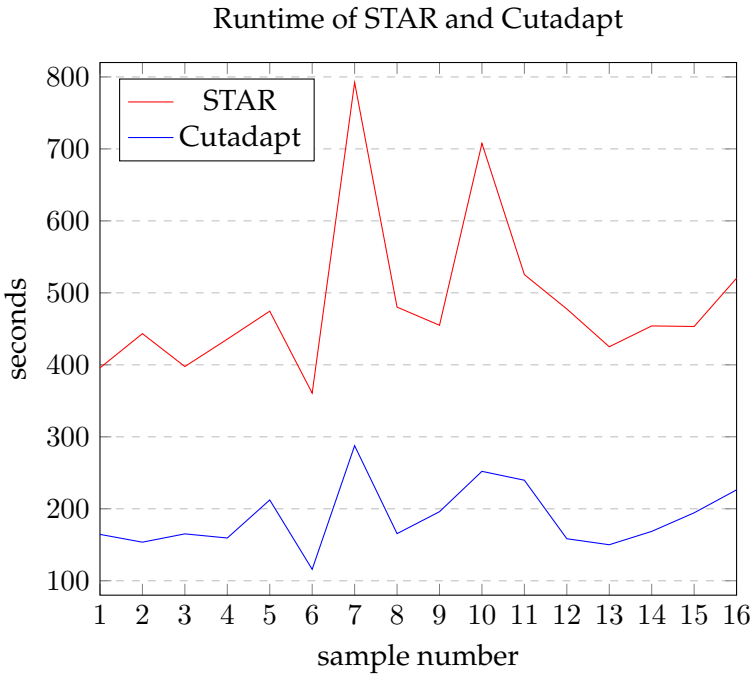

Figure 8: Runtime of STAR and Cutadapt for the example data

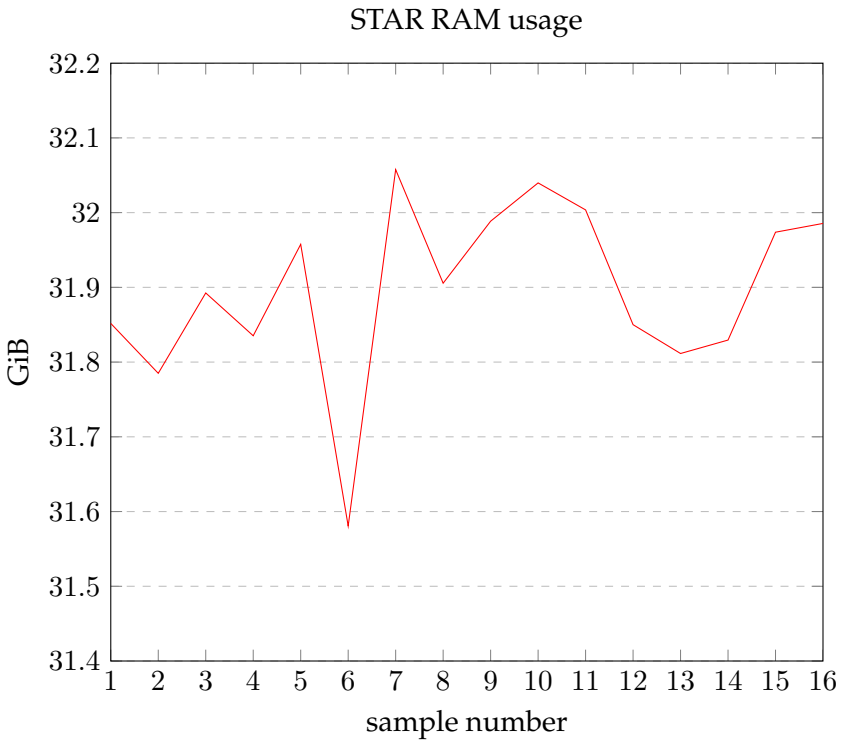

Figure 9: RAM used by STAR for the example data

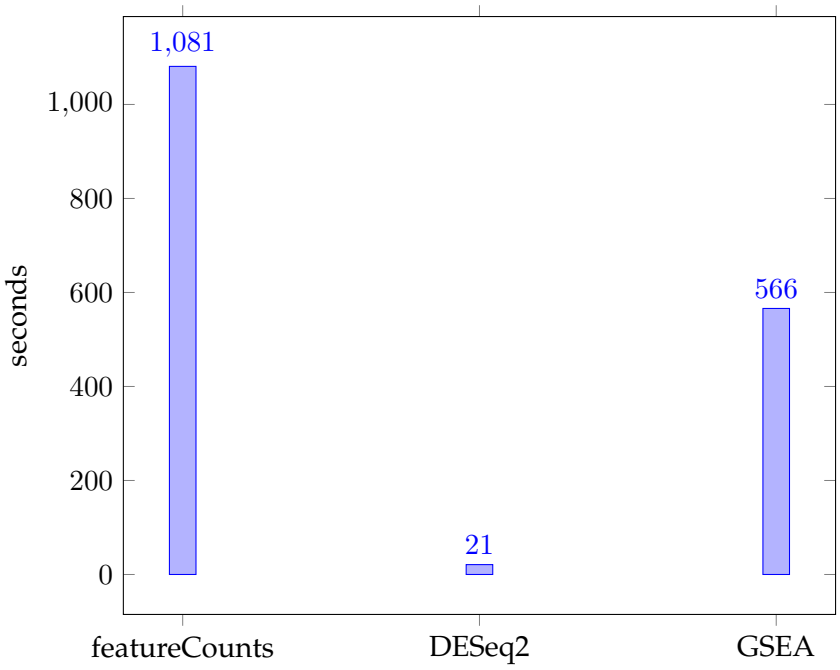

Figure 10: Runtime of featureCounts, DESeq2 and GSEA for the example data

A.3 Outlook

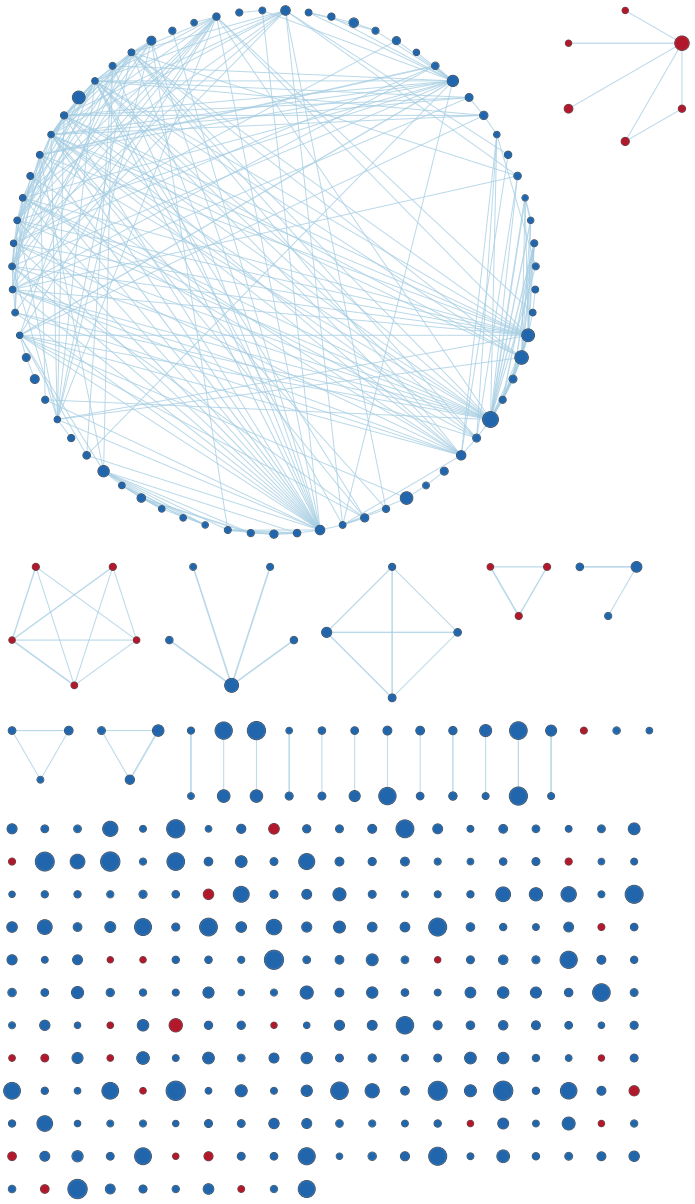

Figure 11: Enrichment Map showing linked gene sets

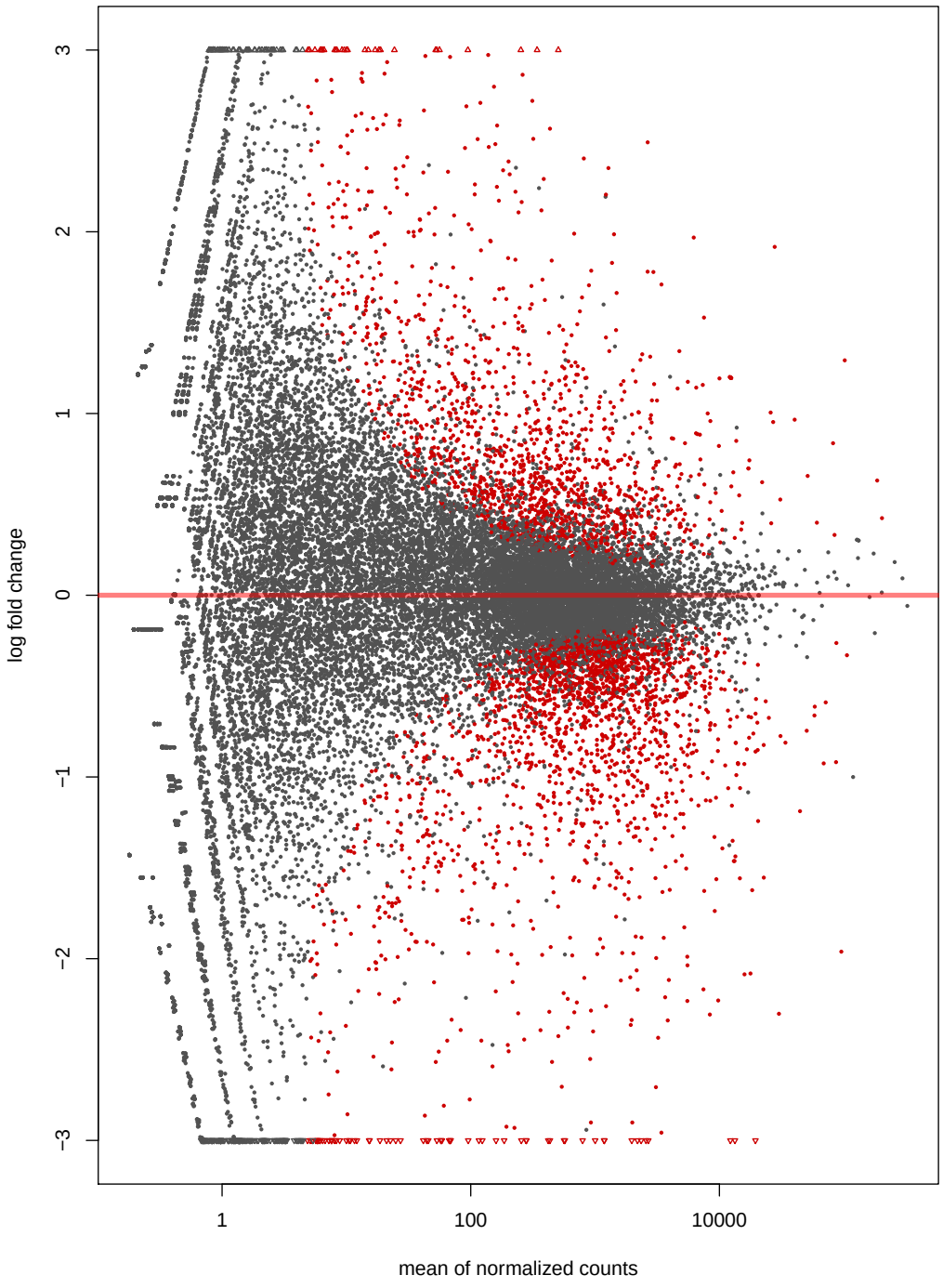

Figure 12: MA-plot showing LFC over the mean of normalized counts

References

[1] Cresko Lab, University of Oregon. *RNA-seqlopedia*. 2011. URL: https://rnaseq.uoregon.edu/#library-prep-sequencing-adapters (visited on 03/16/2020).

[2] K. De Bosscher, W. Vanden Berghe, and G. Haegeman. "Cross-talk between nuclear receptors and nuclear factor kB". In: *Oncogene* 25.51 (2006), pp. 6868–6886.

[3] Dobin, Alexander and Davis, Carrie A. and Schlesinger, Felix and Drenkow, Jorg and Zaleski, Chris and Jha, Sonali and Batut, Philippe and Chaisson, Mark and Gingeras, Thomas R. "STAR: ultrafast universal RNA-seq aligner". In: *Bioinformatics* 29.1 (Oct. 2012), pp. 15–21. eprint: https://academic.oup.com/bioinformatics/article-pdf/29/1/15/17101697/bts635.pdf.

[4] Genome Reference Consortium. *Genome Reference Consortium Human Build 38 patch release 13*. 2019. URL: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.39 (visited on 03/24/2020).

[5] Blanca E. Himes, Xiaofeng Jiang, Peter Wagner, Ruoxi Hu, Qiyu Wang, Barbara Klanderman, Reid M. Whitaker, Qingling Duan, Jessica Lasky-Su, Christina Nikolos, William Jester, Martin Johnson, Reynold A. Panettieri Jr, Kelan G. Tantisira, Scott T. Weiss, and Quan Lu. "RNA-Seq Transcriptome Profiling Identifies CRISPLD2 as a Glucocorticoid Responsive Gene that Modulates Cytokine Function in Airway Smooth Muscle Cells". In: *PLOS ONE* 9.6 (June 2014), pp. 1–13. eprint: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0099625&type=printable.

[6] KEGG. *Asthma - Homo sapiens*. July 2013. URL: https://www.genome.jp/kegg-bin/show_pathway?hsa05310 (visited on 04/15/2020).

[7] Johannes Köster and Sven Rahmann. "Snakemake—a scalable bioinformatics workflow engine". In: *Bioinformatics* 28.19 (Aug. 2012), pp. 2520–2522. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/19/2520/819790/bts480.pdf.

[8] Yang Liao, Gordon K Smyth, and Wei Shi. "The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads". In: *Nucleic Acids Research* 47.8 (Feb. 2019), e47–e47. eprint: https://academic.oup.com/nar/article-pdf/47/8/e47/28534862/gkz114.pdf.

[9] Michael I. Love, Wolfgang Huber, and Simon Anders. "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2". In: *Genome Biology* 15.12 (2014), p. 550.

[10] Marc E Gillespie. *Smooth Muscle Contraction*. Mar. 2009. URL: https://www.reactome.org/content/detail/R-HSA-445355 (visited on 04/14/2020).

[11] Marcel Martin. "Cutadapt removes adapter sequences from high-throughput sequencing reads". In: *EMBnet.journal* 17.1 (2011), pp. 10–12.

[12] *OS-agnostic, system-level binary package manager and ecosystem*. URL: https://conda.io/en/latest/ (visited on 03/27/2020).

[13] Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles". In: *Proceedings of the National Academy of Sciences* 102.43 (2005), pp. 15545–15550. eprint: https://www.pnas.org/content/102/43/15545.full.pdf.

List of Figures

List of Tables