Computer Exercise 3:
RNA-seq data analysis

Introduction to Bioinformatics (MVE510)
Autumn, 2020

Introduction

Welcome to computer exercise 3, in which we will take a close look at the analysis of transcriptome sequencing data generated in RNA-seq experiments. In this computer exercise, you will analyze a large-scale dataset where RNA has been extracted from biopsies from 80 patients with and without a disease. The aim of the exercise is to perform all the necessary steps in the data analysis in order to identify the genes that are differentially expressed between the groups of patients.

Similar to previous exercises, all steps of the analysis should be done in R. The computer exercise should be performed in groups of maximum two students and the results will be examined through a written report that describes the different steps you took, the generated figures and your conclusions. The written code should be added as an appendix. The reports should be handed in through the course home page in Canvas, latest 20th December. Note that there are specific guidelines on the home page regarding the structure formats in which the report should be handed in.

Background and description of the data

In this exercise, you are working as a bioinformatician at a hospital where you are involved in a project investigating the molecular causes behind Chron's disease, a type of inflammatory bowel disease associated with several sever symptoms including abdominal pain, anemia, and arthritis. The cause of Chron's disease is today unknown and an important aim of the project is to investigate differences in gene expression in ileum (tissue from the final section of the small intestine) between healthy and sick patients.

In the project, a cohort of 80 children are studied. The cohort includes 40 children with Chron's disease and 40 children without any symptoms of inflammatory bowel disease (IBD), which are used as controls in the study. IBD is an umbrella term used to describe chronic inflammation of the digestive tract and includes both Ulcerative colitis (UC) and Chron's disease (CD), where the latter is a more severe form of inflammation. Your task, as a bioinformatician, is to investigate which gene expression patterns that are associated with the disease and have therefore conducted a large transcriptome sequencing study. You are also interested in if there are any correlations between age, gene expression and the disease.

Biopsies of the ileum of the 80 patients were used to extract RNA which then were sent for sequencing using an Illumina Hiseq 2000 machine. The raw reads were quality checked and mapped to the human reference genome Hg19 (GRCh37), resulting in around 58,000 genes (transcripts) being identified and quantified. The raw gene counts (number of mapped reads per gene and sample) are used as input for your analysis, together with a data file containing information about the 80 patients involved in the study.

To do this exercise, you will need two datafiles. The file 'count_matrix.txt' contains the raw gene counts for each gene and each sample. The file 'metadata.txt' contains metadata (i.e. data for the data) for each sample, for example, the gender and age of each patient. Both files

are available at http://bioinformatics.math.chalmers.se/courses/MVE510/ and can be downloaded by using your favorite browser.

Exercise 1: Load and filter the data

Start with loading the data into R. The counts matrix is a tab-separated text file and can be loaded into R using the function **read.table**, and stored in the variable ***x***.

> x = read.table('counts_matrix.txt')

How many genes and how many samples are present in the counts matrix?

Continue with loading the metadata file. Here you need to specify that it is a tab separated file, and that the first line is a header.

> metadata = read.table('metadata.txt',sep='\t',header=TRUE)

Explore the counts matrix and the metadata using e.g. **head** and **summary** to get an overview of the dataset. How many male and female patients are there? How many have the disease?

You can also use the **View** command to explore the data inside a matrix or data frame.

> View(x)

In this exercise, you will need to extract information from the metadata. The following commands might be useful for this. What are they doing?

> rownames(x)
> colnames(x)
> metadata$patient.id

Also, for convenience, patients in the metadata file and the counts matrix should match and thus be stored in the same order. To verify that this is the case, write

> colnames(x) == metadata$patient.id

It is sometimes useful to extract the gene expression data for a particular gene. Remember that you can subset a matrix using x['ENSG00000004809',] if 'ENSG00000004809' is present in rownames(x). Use this to look up the raw expression levels (i.e. the counts) for the two genes ENSG00000002586 (CD99) and ENSG00000004809 (SLC22A16). Describe the expression patterns for these two genes.

The next step is to perform a filtering where genes only represented by a small number of counts are removed. If the expression level of a gene is measured by only a few sequence reads, its expression level is uncertain and the statistical power to identify the gene as differentially expressed can be low. In this computer exercise, we will use the criterion that at least 20 of the samples (i.e. 25% of the total samples) should have a non-zero count. Otherwise the gene will be removed. Write a function called **count.nonzero** that takes the counts for one gene (one row in the counts matrix) and counts how many samples that have a count larger than 0 for this gene.

Hint: remember that check xrow>0 returns a vector with TRUE or FALSE values for each element in xrow. If you use **sum** on this vector it will count 1 for TRUE and 0 for FALSE. Use **apply** or a **for**-loop to apply your function to all rows in the count matrix, resulting in the number of non-zero entries for each gene. If you use a for-loop, remember to initiate the variable before the loop.

Create a new matrix called **x.filtered** containing only genes that have non-zero values in at least 25% of the samples. How many genes are left after the low expression filtering?

Exercise 2: Normalization and transformation of the data (logCPM)

The next step is to normalize your data across samples. Write your own code or function that calculates counts per million (CPM). The CPM count is the original count (number of reads) per 1,000,000 total mapped reads in that sample. In this way the data is normalized across samples by dividing by the total count of each sample. However, start with adding an offset (pseudocount) of 1 to each count to avoid taking the logarithm of 0. Remember to use the filtered data, i.e. x.filtered. Why is it important to normalize the data across samples?

Continue with log transformation of the CPM values to get logCPM. What is the main reason to transform the counts using the logarithm?

Exercise 3: Explore the data

Examine distribution of the data after the preprocessing in the previous exercise. Use **boxplot** to plot the distribution of gene expression each sample (next to each other in the same plot). Add a title to the plot and a label to the y-axis. For comparison, create boxplots also for the log-transformed non-normalized data. How does the distribution of counts look after normalization? Does it look as expected?

Use **plot** to draw a scatter plot for two of the samples. Plot the the logCPM counts (of all genes) for sample 1 (SRR1782694) on the x-axis and the logCPM counts for sample 41 (SRR1782687) on the y-axis. The genes that are above the line x=y have a higher expression in sample 41 than in sample 1, and the genes below the line have a lower expression. How do you explain the ‘stripes’ of genes that you can see at the bottom and left of the plot?

Exercise 4: Linear models to find differentially expressed genes

In this part we will perform statistical analysis of one single gene to see if it is differentially expressed when comparing different groups. In the next exercise we will generalize this analysis to all genes. Start with extracting the counts over the 80 samples for the first gene: ENSG00000000003 (TSPAN6). Plot the normalized and log transformed counts grouped by their diagnosis, i.e. Chron’s disease and controls, either by using boxplots or by plotting the mean and standard deviations of each group. Can you see from the plot if there is a difference in expression between the two groups for the gene ENSG00000000003? Remember that age and gender dependence are not accounted for in this grouping.

Test if the gene is differentially expressed when comparing the two disease groups. Start with fitting a linear model, using **lm**, that only models the disease groups and call the results **fit1.** Then, fit another model that takes three factors into account, namely age, gender and disease,

and save the results in the variable **fit2**. The age factor should be continuous and the other two categorical. For information on how to use the **lm** command see the lecture notes or use the **help** command.

For the categorical variables it is useful to convert them into factors in R before specifying the model. If you have a vector called diagnosis, first convert it to a factor using

>diagnosis=as.factor(diagnosis)

Check which levels the factor has using **levels(diagnosis)**. The first level will be used as a reference and the other levels will be compared to that level. If you want to use another condition as reference you can use the **relevel** command.

Is the gene ENSG00000000003 differentially expressed when comparing the two diagnosis groups, when using the first linear model (fit1)? What is the p-value? Is the gene significantly differentially expressed using the second linear model (fit2)? Is the gene up-regulated or down-regulated in the disease group compared to the controls? What is the effect size, i.e. the value of the parameter associated with the independent variable specifying if the patient is sick or healthy? Is the expression of the gene influenced by age or gender? What is the difference of using the model in fit2 compared to fit1?

Hint: Remember that you can use **summary** to generate additional information about the fitted linear model and **summary(fit1)$coefficients** to extract p-values.

Exercise 5: Linear models to find differentially expressed genes, all genes

Apply the analysis above to all the genes. Write a for-loop that goes through all genes and saves the p-value and estimated coefficient value for each gene for the disease vs. control comparison. Try with both models, i.e. the models in fit1 and fit2 from the previous exercise. Fitting the model for all genes may take a few minutes, so be patient. After the for-loop, adjust the p-values for multiple testing using **p.adjust**. Use the False Discovery Rate (FDR) method. Why is it, in this case, important to adjust the p-values for multiple testing?

How many genes are significantly differentially expressed when using the first model (only one factor)? What false discovery rate cutoff did you use? How many genes are differentially expressed (comparing disease to control) when using the second model (three factors)? Out of the significant genes, how many are up-regulated and down-regulated respectively, when comparing disease to control? Explain also the reason for calculating the FDR in this case and how it can be used to ensure that the results do not contain a large number of false positives.

Hint: use **cbind** to combine the vectors with coefficients and adjusted p-values, add rownames (make sure that they are in the same order), and then subset the resulting matrix twice, or use **&** (have a look at help('&')).

Which gene is the most significant when comparing disease to control? Is it up- or down-regulated in the disease group? What is the effect size (log fold-change)? Explain the role of this gene and try to explain why it is differentially expressed in the patients with Chron's disease.

The file geneAnnotation_GRCh37.87.txt, available at http://bioinformatics.math.chalmers.se/courses/MVE510/, contains descriptions of the genes present in the dataset. Have a look at the other top 5 most significant genes. Are they up- or down-regulated? What are they doing? Can you explain their role based on the biological question in this study?

How many genes are significantly associated with age and gender respectively? Which gene is the most significant for the gender factor? Why do you think it is differentially expressed?

Exercise 6: Hierarchical clustering of the most significant genes

In this part we will perform hierarchical clustering of the top 100 most significantly differentially expressed genes (considering the diagnosis factor). The **heatmap.2** function in the gplots package calculates a matrix of Euclidian distances and performs a hierarchical clustering of the genes and draws a heatmap of the clustered counts matrix. Hierarchical clustering is an unsupervised clustering and the heatmap is a way to visualize the clustering of the genes and samples. Start with extracting the logCPM counts over all 80 samples for the 100 most significant genes regarding the diagnosis (using the fit2 model**)** and save it in a matrix called xsig.

We will use the package RColorBrewer to produce a nice color scheme for the heatmap. Start by loading the packages gplots and RColorBrewer.

Define the colors using the color scheme 'RdYlBu', interpolate to get more colors and save the colors in reverse order, using the following code:

```
> mypalette = brewer.pal(11,"RdYlBu")
> morecols = colorRampPalette(mypalette)
> mycols=rev(morecols(255))
```

Also create a vector with colors that we will use to label the columns of the logCPM matrix according to the diagnosis of the samples, for example:

```
> column.cols=c("purple","orange")[metadata$diagnosis]
```

The function **heatmap.2** plots a heatmap, and automatically clusters both the rows and the columns of the input matrix, behind the scenes. Plot the heatmap of the xsig matrix and save it as a pdf, using the following code:

```
> pdf("top100sigGenesHeatmap.pdf",height=9,width=6)
> heatmap.2(xsig,trace='none',col=mycols,main='The 100 most significant
genes',ColSideColors=column.cols)
> dev.off()
```

Did the samples cluster as expected? Describe the clustering of the genes, how do the genes group together in the clustering?

Exercise 7: Principal component analysis

Principal component analysis (PCA) is another way to visualize the data. It is often used as a tool for visualization and quality assessment of the data. Here we will visualize the samples based on the expression (logCPM) of all genes. PCA is a mathematical technique that transforms and projects the variables onto linearly uncorrelated and orthogonal principal components.

Perform a PCA on the transposed logCPM matrix, to get a representation of the samples, based on the expression of all genes. This can be done using the **prcomp** function,

> pca=prcomp(t(logcpm))

Note that the matrix here needs to be transposed using the function **t**, otherwise **prcomp** will perform the PCA on the genes (i.e. the rows in the matrix) instead of the samples.

Explore the **pca** variable by using **summary(pca).** The values of the principal components are in **pca$x**. How much of the variability in the data is explained by the first two principal components (PC1 and PC2)?

Plot the first two principal components against each other in a scatter plot. Color the samples based on their diagnosis. Here you can reuse the column.cols vector from the previous exercise. Are the first two components sufficient to separate the Chron's disease samples from the controls? Try to include the third principal component as well, for example by plotting PC1 against PC3 and PC2 against PC3. Do you see any separation in any of the groups? Are any of the samples problematic? Why could that be? Color the samples based on gender as well. Do the groups separate in the plot?