Deforges 2019 rna-seq analysis

Marc Galland

2021-05-18

Contents

Setup

Add a table of contents

Link to R Markdown guide (click me)

Disabling warnings and messages

This will keep your final PDF report clean from execution alarms, unnecessary text, etc.
This code chunck sets global options for the execution of each code chunk. You can disable warnings and messages globally this way.

```r
knitr::opts_chunk$set(echo = TRUE,
                      warning = FALSE,
                      message = FALSE,
                      collapse = TRUE)
```

Introduction

Q1: The publication of Deforges et al. 2019 is aimed at discovering the extent of the regulated exerted by certain non-coding RNAs called "cis-natural antisense transcripts" (cis-NATs) on the coding messenger RNA located on the opposite DNA strand. cis-natural antisense RNAs are non-coding RNAs located on the opposite DNA strand relatively to a true coding messenger RNA. cognate sense RNAs are messenger RNA that code for a protein.

Q2: seedings were harvested after 3h of incubation with the different phytohormones (including auxin). Three biological replicates per condition were taken.

Q3: the Illumina HiSeq 2500 plaftorm was used.

> "The libraries were sequenced on a HiSeq 2500 Illumina sequencer and about 30 million of paired-end reads per sample were obtained."

Q4: the Hisat2 software was used for read mapping to the reference Arabidopsis thaliana genome.

> "Identification of cis-NATs and Analysis of their Coding Potential
> To identify cis-NATs, the paired-end reads from the 3 replicates were pooled together and uniquely mapped to the TAIR10 genome using Hisat2 (Kim et al., 2015)."

Exercise 1: data import

We first load the `tidyverse` package that contains most of the data transformation functions we will need.

```r
library("tidyverse")
```

Import gene counts and sample to conditions

```r
raw_counts <- read.csv("raw_gene_counts_arabidopsis_root_auxin.csv",
                       header = TRUE,
                       stringsAsFactors = FALSE) %>%
  # for DESeq subsequent data import
  column_to_rownames("gene")

# first five rows
knitr::kable(raw_counts[1:5,])
```

	root_control_1	root_control_2	root_control_3	root_auxin_1	root_auxin_2	root_auxin_3
AT1G01010	2029	1481	2694	2450	1767	2166
AT1G01020	1626	1608	1895	1816	2429	1716
AT1G01030	150	230	375	149	175	260
AT1G01040	3174	2599	4260	3753	2419	3838
AT1G01046	70	42	115	67	45	89

```r
sample2condition <- read.csv("sample2condition_arabidopsis_root_auxin.csv",
                             header = TRUE,
                             stringsAsFactors = FALSE)

# first five rows
knitr::kable(sample2condition)
```

sample	condition
root_control_1	control
root_control_2	control
root_control_3	control
root_auxin_1	auxin
root_auxin_2	auxin
root_auxin_3	auxin

Create the DESeqDataSet object

This `DESeqDataSet` object is used to store both data (gene counts) and metadata (sample to experimental condition correspondence) in one unique R object. Functions can be directly be applied to this object and corresponding results stored within the same object.

```
library(DESeq2)
dds <- DESeqDataSetFromMatrix(countData = raw_counts,
                              colData = sample2condition,
                              design = ~ condition)
```

You can have a quick peek at the number of genes, number of samples, etc. by calling the `dds` object.

```
dds
## class: DESeqDataSet
## dim: 28642 6
## metadata(1): version
## assays(1): counts
## rownames(28642): AT1G01010 AT1G01020 ... ATMG01400 ATMG01410
## rowData names(0):
## colnames(6): root_control_1 root_control_2 ... root_auxin_2
##   root_auxin_3
## colData names(2): sample condition
```

Calculate size factors and scale gene counts

The median ratio method is used to calculate the size factor correction for each sample (mainly corrects for library sequencing depth but not only).

```
dds <- estimateSizeFactors(dds)
sizeFactors(dds)
## root_control_1 root_control_2 root_control_3   root_auxin_1   root_auxin_2
##      0.9933574      0.9079246      1.1889831      1.0078218      0.9604149
##   root_auxin_3
##      0.9712745
```

Since most of these size factors are comparable, it suggests that the sequencing depth and other possible biases are not heavily affecting the gene count levels.

Let's scale (= correct) gene counts accordingly. Since we have already calculated and saved the size factors in the `dds` object, we can run:

```
scaled_counts = counts(dds, normalized = TRUE)
```

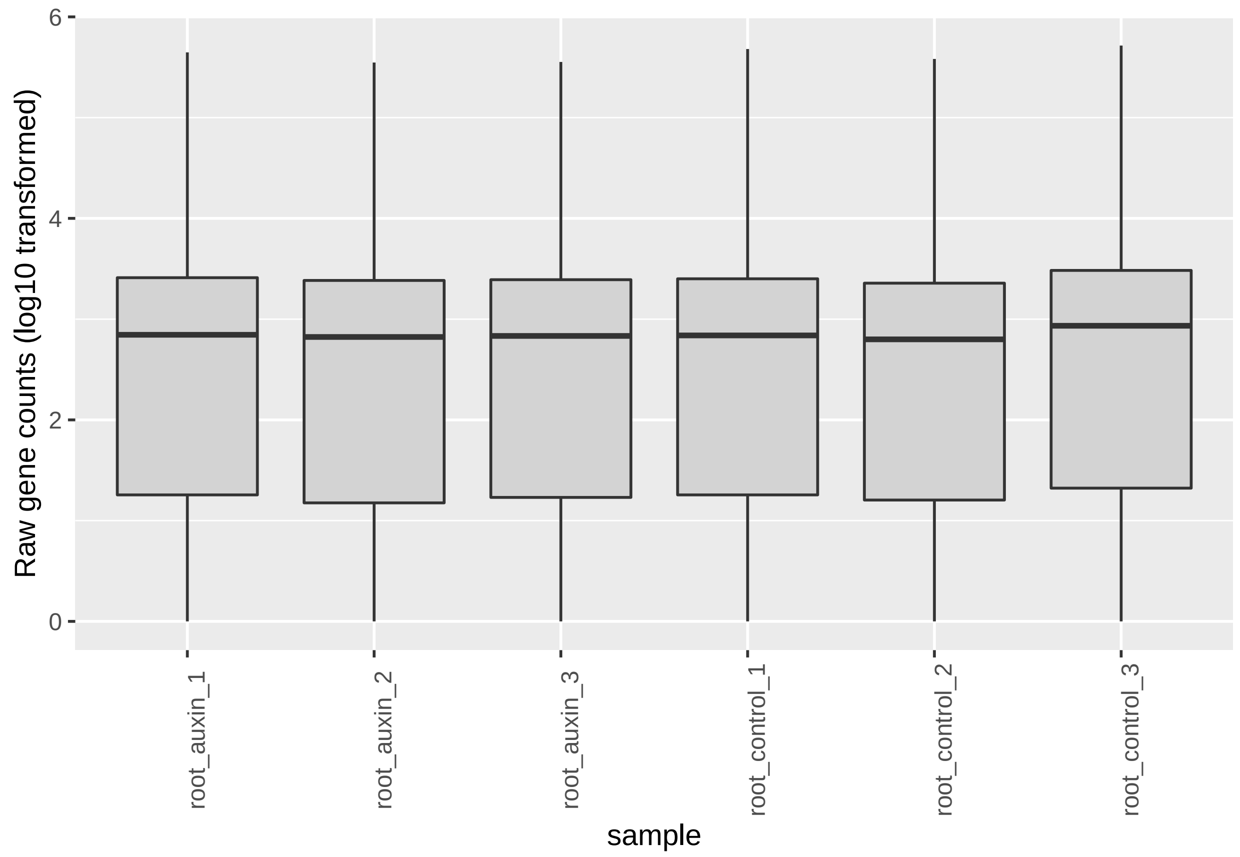
Save this data as a `.csv` file.

```
write.csv(x = scaled_counts,
          file = "../01.desforges2019/scaled_counts.csv",
          quote = FALSE,
          row.names = TRUE)
```

Plot the raw and scaled counts next to one another

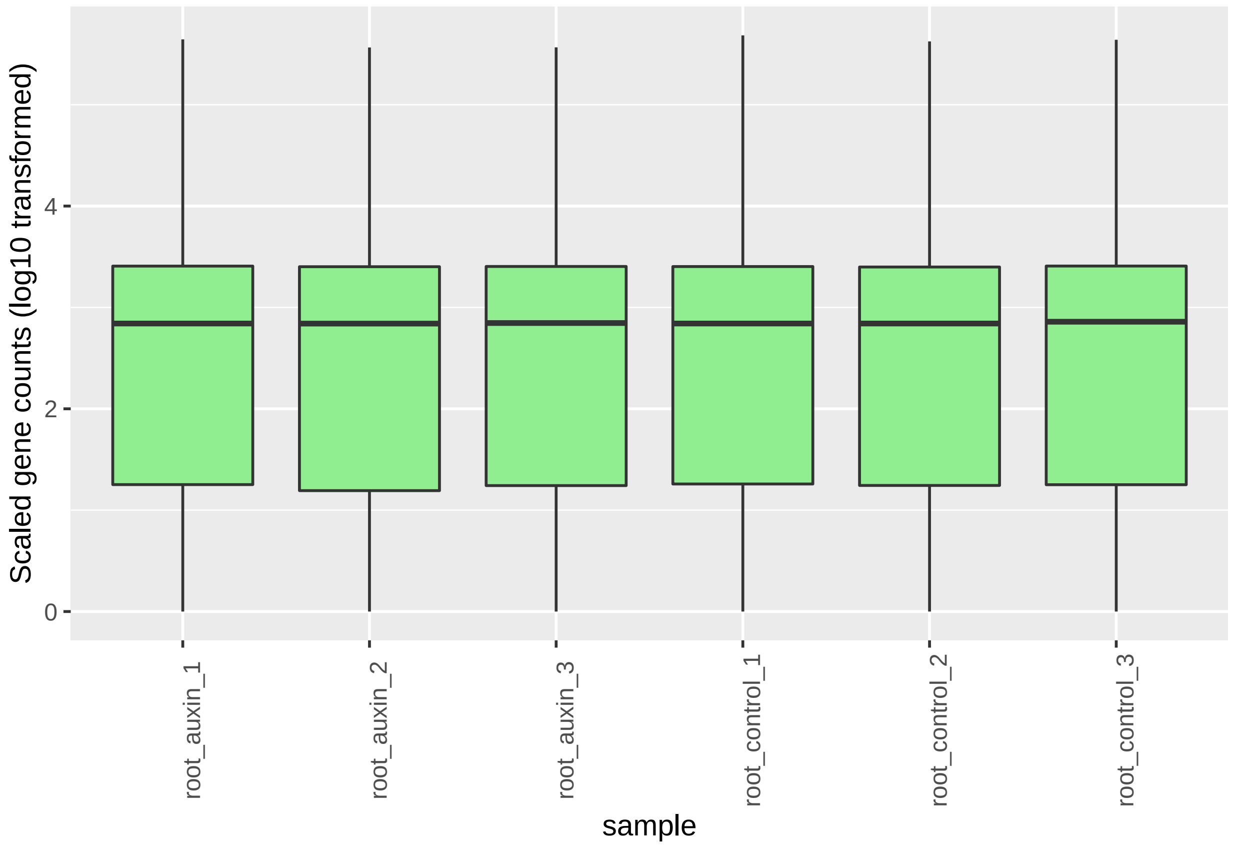
Plot the raw counts.

```
p_raw <-
  raw_counts %>%
  rownames_to_column("gene") %>%
  gather(key = "sample", value = "gene_counts", - gene) %>%
  mutate(gene_counts_log = log10(gene_counts + 1)) %>%
  ggplot(., aes(x = sample, y = gene_counts_log)) +
  geom_boxplot(fill = "lightgrey") +
  labs(y = "Raw gene counts (log10 transformed)") +
  theme(axis.text.x = element_text(angle = 90))
p_raw
```



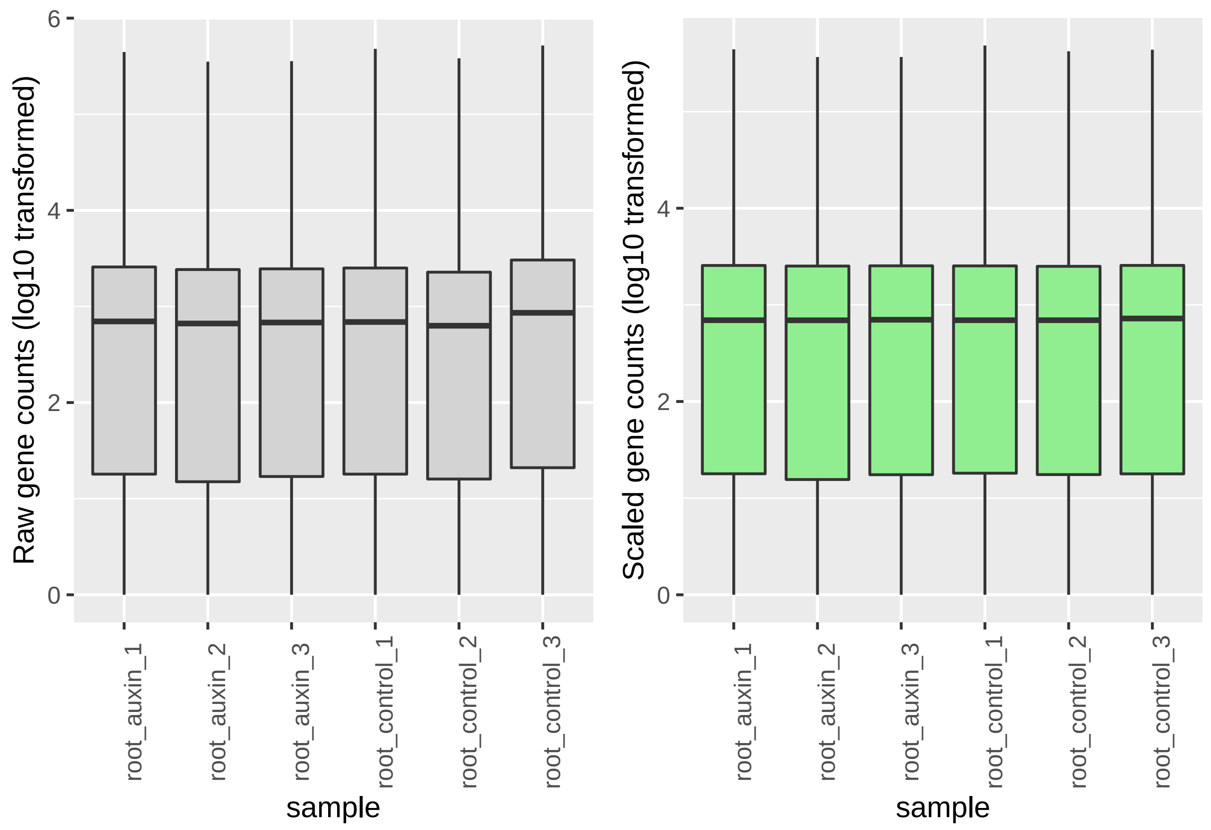
Plot the scaled counts.

```
p_scaled <-
  scaled_counts %>%
  as.data.frame() %>%
  rownames_to_column("gene") %>%
  gather(key = "sample", value = "gene_counts", - gene) %>%
  mutate(gene_counts_log = log10(gene_counts + 1)) %>%
  ggplot(., aes(x = sample, y = gene_counts_log)) +
  geom_boxplot(fill = "lightgreen") +
  labs(y = "Scaled gene counts (log10 transformed)") +
  theme(axis.text.x = element_text(angle = 90))
p_scaled
```



We can plot them side by side using the `patchwork` library.

```
library("patchwork")
p_raw + p_scaled
```



Q5: The scaling procedure ensures that gene counts are corrected for potential bias affecting the final gene counts e.g. different sequencing depth per sample. It uses the median of ratios method to calculate a correction factor: `estimateSizeFactors()` One can see on the last plot that gene count medians are more similar after scaling.

Exercise 2: Principal Component Analysis

The PCA analysis will be used to show the distance between samples. To do so, we will create a so-called score plot. The PCA will also tell us how much of the total variance can be explained by the first two principal components.

A PCA analysis acts as a sample-level quality check of our experiment: - Are the samples from the same condition grouped together? - Are the first two principal components (PC1, PC2) explaining a major percentage of the total variation present in the dataset? - Are the samples from different experimental conditions well separated by PC1 and PC2?

If "yes" is the answer to these questions, then the experiment can be considered a success.

Variance stabilisation

The PCA is very sensitive to mean-variance relationship. We first stabilise the variance so that it becomes independent from the mean.

```r
dds = estimateDispersions(object = dds,
                          fitType = "parametric",
                          quiet = TRUE)

vsd = varianceStabilizingTransformation(
  object = dds,
  blind = TRUE, # do not take the design formula into account.
                # best practice for sample-level QC
  fitType = "parametric")

# extract the matrix of variance stabilised counts
variance_stabilised_counts <- assay(vsd)
```

We need to have samples in rows and genes in columns. The `scaled_counts` object has to be transposed before computing the PCA.

```r
t_variance_stabilised_counts <- t(variance_stabilised_counts)
```

PCA function

Let's import the `mypca()` function.
Source: https://scienceparkstudygroup.github.io/rna-seq-lesson/05-descriptive-plots/index.html#53-the-iris-data-set

```r
source("../mypca.R")
```

PCA computation: screeplot

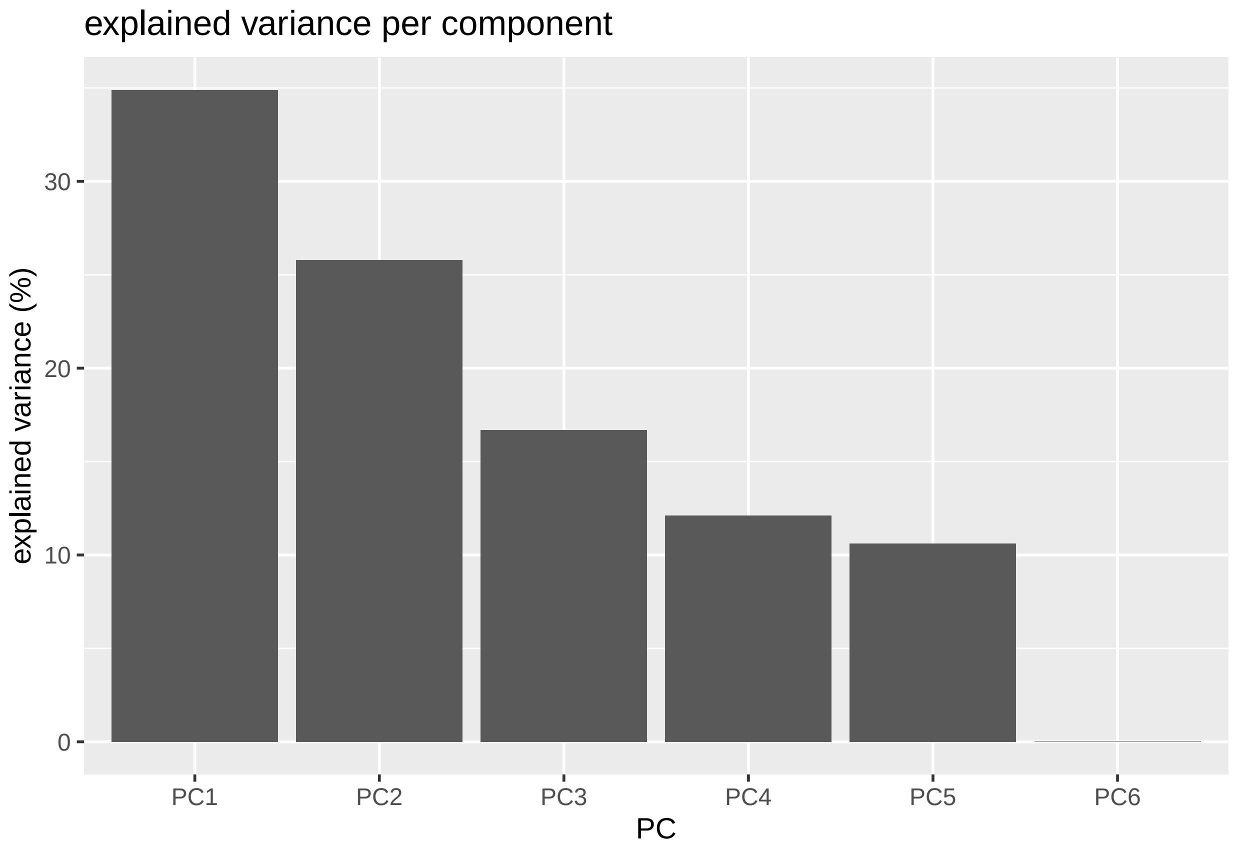
```r
pca_results <- mypca(t_variance_stabilised_counts,
                     center = TRUE,
                     scale = TRUE)
```

The `pca_results` contains the scores, loadings and explained variance.

Scree plot

The scree plot shows the percentage of the total variance explained by each principal components. Here, with 6 samples, we can compute a maximum of 6 PCs.

```r
percentage_variance <- as.data.frame(pca_results$explained_var)

# make the plot
scree_plot <-
  percentage_variance %>%
  rownames_to_column("PC") %>%
  ggplot(., aes(x = PC, y = exp_var)) +
    ylab('explained variance (%)') +
    ggtitle('explained variance per component') +
    geom_bar(stat = "identity")
scree_plot
```



Q6: PC1 and PC2 explain 34.9% and 25.8% of the total variance respectively.

Sample score plot

In the same dataframe, PCA sample scores are combined with their experimental condition.

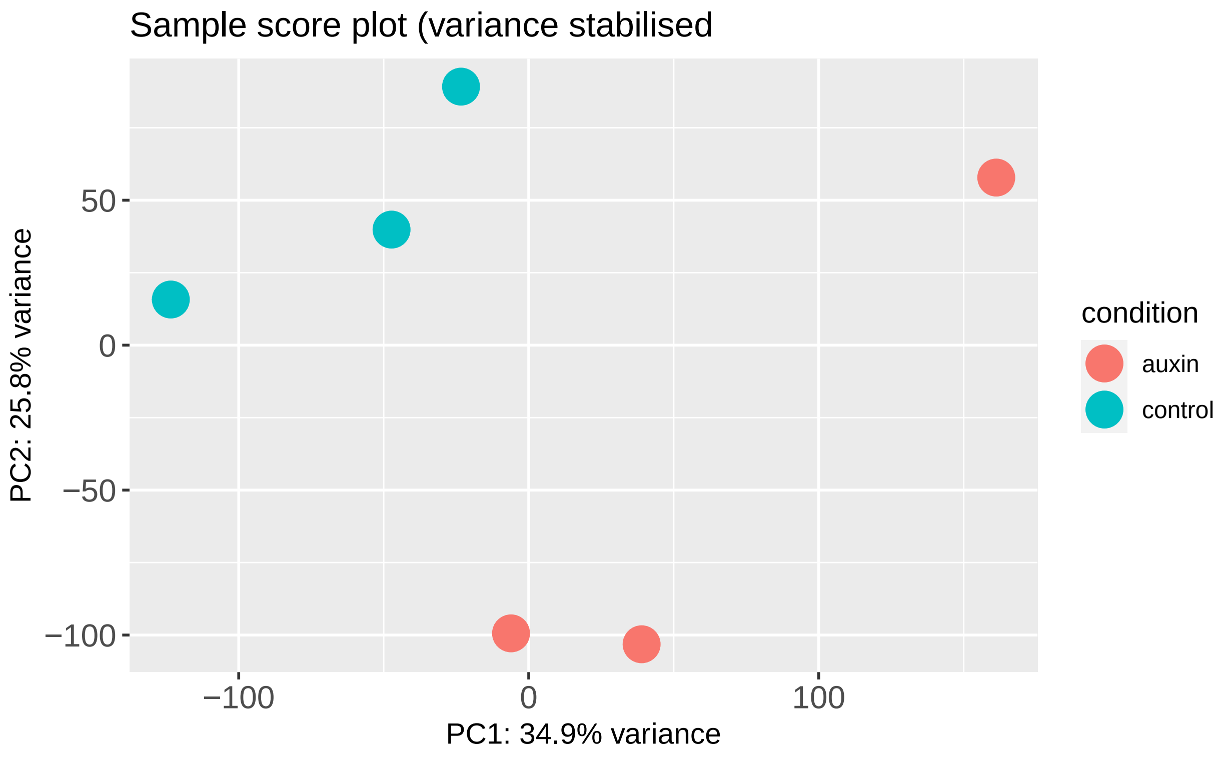
```r
scores <- pca_results$scores %>%
  rownames_to_column("sample") %>%
  left_join(., y = sample2condition, by = "sample")

knitr::kable(scores, digits = 5)
```

sample	PC1	PC2	PC3	PC4	PC5	PC6	condition
root_control_1	-47.29801	39.86358	-73.54597	71.02844	48.38594	0	control
root_control_2	-23.33957	89.18108	-22.91227	-93.02960	12.96244	0	control

sample	PC1	PC2	PC3	PC4	PC5	PC6	condition
root_control_3	-123.41854	15.74120	63.45075	25.08320	-60.88214	0	control
root_auxin_1	38.93312	-103.18006	-75.55200	-16.40258	-53.29818	0	auxin
root_auxin_2	161.22114	57.81208	44.35319	29.90570	-15.09479	0	auxin
root_auxin_3	-6.09813	-99.41788	64.20630	-16.58517	67.92674	0	auxin

Create the score plot.

```
p_score_plot <- ggplot(scores, aes(PC1, PC2, color = condition)) +
  geom_point(size = 6) +
  xlab(paste0("PC1: ",percentage_variance[1,],"% variance")) +
  ylab(paste0("PC2: ",percentage_variance[2,],"% variance")) +
  coord_fixed() +
  ggtitle("Sample score plot (variance stabilised") +
  theme(axis.text = element_text(size = 12))
p_score_plot
```



Q7: the samples from the control group are grouped together on both PC1 and PC2. That is not as clear for samples from the auxin-treated group since these are separated on both PC1 and PC2. Yet, this experiment can be considered good enough to continue since the experimental condition is clearly reflected by PC1. Since PC1 account for the majority of the variance, the differential gene expression should give consistent results.

Exercise 3: differential expression

Calling differential genes

```r
dds <- DESeq(dds)
diff_genes <- results(dds, contrast = c("condition", "auxin", "control")) %>%
  as.data.frame()
```

```r
diff_genes_signif <-
  diff_genes %>%
  filter(padj < 0.01) %>%
  nrow()
```

Q8: 1740 genes out of a total of 28642 genes.

Top 20 genes

By filtering first on the adjusted p-value (should be < 0.01) and then by ordering the genes by their absolute log2FoldChange, we can visualise the top 20 most significant differentially regulated genes.

```r
diff_genes %>%
  filter(padj < 0.01) %>%
  dplyr::arrange(desc(abs(log2FoldChange))) %>%
  head(n = 20) %>%
  knitr::kable(digits = 3)
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
AT3G23635	34.248	8.575	1.253	6.841	0	0.000
AT1G05650	160.789	-7.878	0.762	-10.338	0	0.000
AT3G58190	3921.939	7.674	0.314	24.415	0	0.000
AT2G17680	66.683	7.669	1.073	7.150	0	0.000
AT2G23170	66400.262	7.626	0.188	40.638	0	0.000
AT3G22250	15.477	-7.369	1.421	-5.187	0	0.000
AT1G05660	427.411	-6.962	0.365	-19.082	0	0.000
AT3G23637	9.870	6.779	1.421	4.769	0	0.000
AT3G17760	161.227	6.744	0.598	11.285	0	0.000
AT4G37390	38155.491	6.669	0.203	32.822	0	0.000
AT2G43860	8.822	6.619	1.739	3.807	0	0.002
AT3G49700	330.503	6.601	0.377	17.504	0	0.000
AT1G06920	145.807	6.477	0.511	12.673	0	0.000
AT4G12510	467.165	-6.448	0.711	-9.068	0	0.000
AT5G46900	198.485	-6.360	0.601	-10.589	0	0.000
AT2G03740	211.401	6.342	0.532	11.912	0	0.000
AT2G10608	7.084	6.302	1.530	4.120	0	0.001
AT1G15600	6.934	6.270	1.685	3.721	0	0.003
AT3G10870	3042.296	6.244	0.311	20.104	0	0.000
AT5G57520	804.028	6.007	0.260	23.065	0	0.000

Since “we called”auxin" was used as the numerator in the `results()` function, a positive log2FoldChange indicates that the gene is up-regulated by the application of auxin in Arabidopsis roots.

```r
genes_upregulated_by_auxin <-
  diff_genes %>%
  filter(padj < 0.01) %>%
  dplyr::arrange(desc(abs(log2FoldChange))) %>%
  head(n = 20) %>%
  dplyr::filter(log2FoldChange > 0)


knitr::kable(genes_upregulated_by_auxin,
             digits = 3)
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
AT3G23635	34.248	8.575	1.253	6.841	0	0.000
AT3G58190	3921.939	7.674	0.314	24.415	0	0.000
AT2G17680	66.683	7.669	1.073	7.150	0	0.000
AT2G23170	66400.262	7.626	0.188	40.638	0	0.000
AT3G23637	9.870	6.779	1.421	4.769	0	0.000
AT3G17760	161.227	6.744	0.598	11.285	0	0.000
AT4G37390	38155.491	6.669	0.203	32.822	0	0.000
AT2G43860	8.822	6.619	1.739	3.807	0	0.002
AT3G49700	330.503	6.601	0.377	17.504	0	0.000
AT1G06920	145.807	6.477	0.511	12.673	0	0.000
AT2G03740	211.401	6.342	0.532	11.912	0	0.000
AT2G10608	7.084	6.302	1.530	4.120	0	0.001
AT1G15600	6.934	6.270	1.685	3.721	0	0.003
AT3G10870	3042.296	6.244	0.311	20.104	0	0.000
AT5G57520	804.028	6.007	0.260	23.065	0	0.000

Q9: 15 are up-regulated by auxin.

Maximum and minimum log2 fold changes

What are the maximum and minimum log2 fold changes from these top 20 genes?

Maximum

```
max_logfc <-
  diff_genes %>%
  filter(padj < 0.01) %>%
  dplyr::arrange(desc(abs(log2FoldChange))) %>%
  pull(log2FoldChange) %>%
  max()
```

The maximum log2FoldChange in response to auxin is 8.5747053. This corresponds to a fold change of 381.2795337. (2^foldchange).

The expression of the AT3G23635 gene in auxin-treated seedlings is 381.2795337 higher than in control seedlings.

```
min_logfc <-
  diff_genes %>%
  filter(padj < 0.01) %>%
  dplyr::arrange(desc(abs(log2FoldChange))) %>%
  pull(log2FoldChange) %>%
  min()
```

The minimum log2FoldChange in response to auxin is -7.8777312. This corresponds to a fold change of 0.0042517.

The gene AT1G05650 is expressed 235.1978743 times higher (1/min) in control seedlings than in auxin-treated seedlings.

Exercise 4: volcano plot

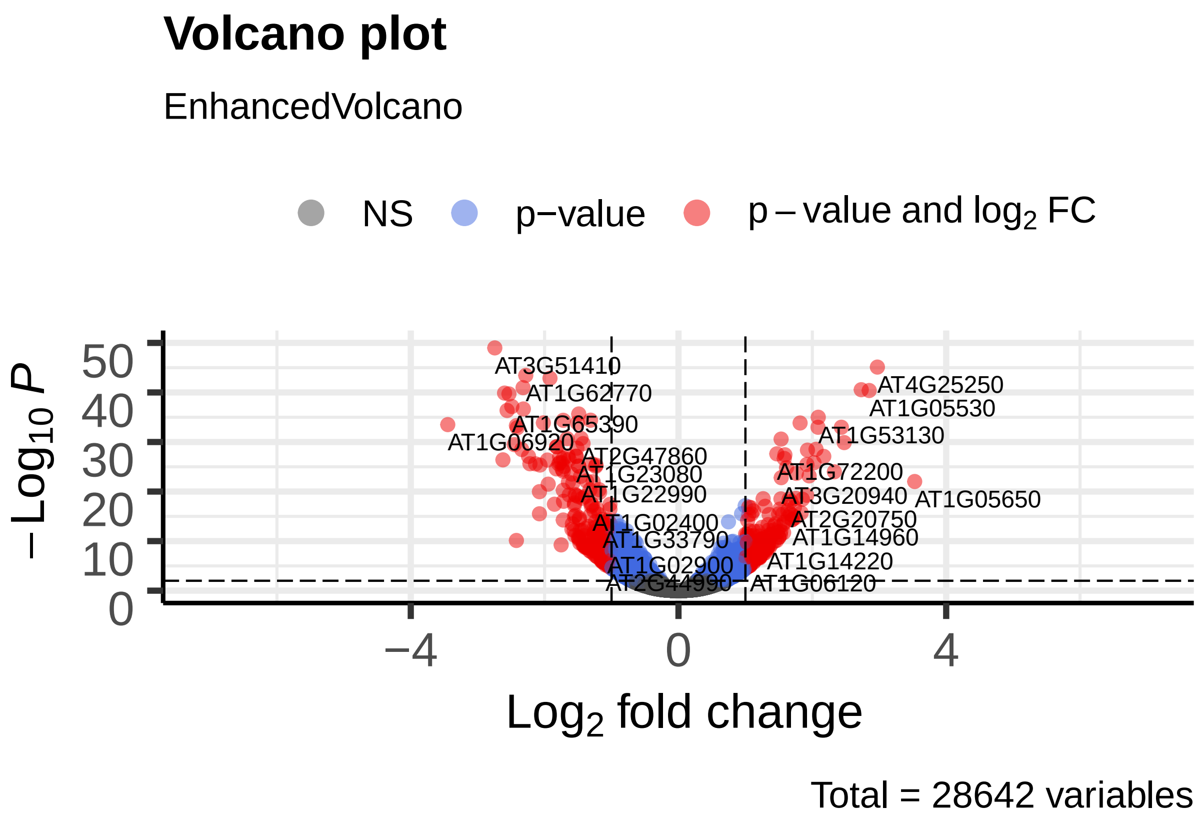
Shrinkage

Shrinkage prevents

```r
diff_genes <- results(dds, contrast = c("condition", "auxin", "control"))

resLFC <- lfcShrink(dds = dds,
                    res = diff_genes,
                    type = "normal",
                    coef = 2)
```

Volcano plot

```r
library("EnhancedVolcano")
EnhancedVolcano(toptable = resLFC,
                x = "log2FoldChange",
                y = "padj",
                lab = rownames(resLFC),
                xlim = c(-7, +7),
                ylim = c(0,50),
                pCutoff = 0.01,
                transcriptPointSize = 2.0,
                FCcutoff = 1,
                title = "Volcano plot",
                legend=c(
                  'Not significant',
                  'Log2 fold-change (but do not pass p-value cutoff)',
                  'Pass p-value cutoff',
                  'Pass both p-value & Log2 fold change')) +
  guides(legend = NULL)
```



Q11: - Most up-regulated genes: top right of the volcano plot. - Most down-regulated genes: top left of the volcano plot. - Most statistically significant genes are the highest on the y-axis.

Q12: the volcano plot is well balanced since it seems that the number of up- and down-regulated genes is comparable.

Q13: The minimum and maximum log2 fold changes have been altered (see below) by the shrinkage.
The minimum is now -6.2808438.
The maximum is now 4.6365042.
The limits of the volcano plot have been changed accordingly.

```
knitr::kable(broom::tidy(
  summary(
    resLFC$log2FoldChange)
  )
)
```

minimum	q1	median	mean	q3	maximum	na
-6.280844	-0.1043489	-0.0020778	0.0018927	0.1063977	4.636504	2513