

Lab 3: Data Visualization for Gene Expression Data

[Software needed: web access]

After nucleotide sequence data, gene expression data are the most prevalent type of genomics data. Advances in the early 2000s in microarray technology led to the generation of hundreds of thousand of gene expression datasets, which were further augmented by the advent of next generation sequencing technologies around 2010, which vastly increased sequencing throughput and decreased cost, as discussed in the lecture. RNA sequencing (RNA-seq) has enabled the easy quantification of expression levels, even in organisms without a sequenced genome. Further advances in this technology have enable the generation of transcriptomic data from single cells. We'll touch on scRNA-seq in the last lab, in the context of dimensionality reduction.

In today's lab, we will explore how we can visualize significantly differentially expressed genes using volcano plots via an online platform called Galaxy. We'll also generate heatmaps and clustergrams in R, and last, we'll generate an "electronic fluorescent pictograph" to help in the interpretation of gene expression data.

Table 1: Tools covered in this lab (see end of lab for references)

Tool	Notes
Galaxy	An online suite of tools for analyzing genomic data. Acts as a wrapper for many R scripts. Register to create a free account.
R	Generating heatmaps and clustergrams.
My eFP Browser	Create an "electronic fluorescent pictograph" to aid in the interpretation of gene expression data.

Galaxy

1. We'll be using a data set from the Provart Lab to create a volcano plot. Download the *K20dvs.w.csv* file from the Coursera website. The data set is the analysis of differentially expressed genes from RNA isolated from guard cells of Arabidopsis plants (isolated using the INTACT system) at 20% soil water content (i.e., quite a drought situation!) relative to guard cells from well watered plants at the same developmental stage. We cover how to perform read mapping, summarization, and identification of differentially expressed genes with DESeq2 in a different course (Bioinformatic Methods II). This week, we'd just like to highlight the most significantly differentially expressed genes in a nice figure. We might consider plotting the expression value for every gene (~20,506 genes are expressed in the samples) in both droughted and well-watered conditions as a scatter plot, but with such a plot we can't see what level of statistical significance a gene has.

If the drought didn't cause any change in gene expression, what would the scatter plot look like? If there were changes in gene expression caused by drought, how would these appear on the scatter plot?

2. A volcano plot is an alternative way of visualizing single-factor perturbation type of gene expression profiling experimental data. As touched on in the lecture, volcano plots display not only the fold-change in expression level of a gene in the treated versus control condition (e.g., droughted vs. well-watered plants in our case), but also the value for the statistical test used to call a gene as being differentially expressed. As we will be using a workflow to generate our volcano plot, we'll need to register for an account at usegalaxy.org. This part of the lab is based on a Galaxy tutorial: <https://galaxyproject.github.io/training-material/topics/transcriptomics/tutorials/rna-seq-viz-with-volcanoplot/tutorial.html>
3. Download the “rna-seq-viz-with-volcanoplot.ga” workflow from the course website. Go to <https://usegalaxy.org/> and click on the Workflow tab at the top of the page, as shown in **Figure 1**. Use the Import button to upload the workflow to your Galaxy workspace.

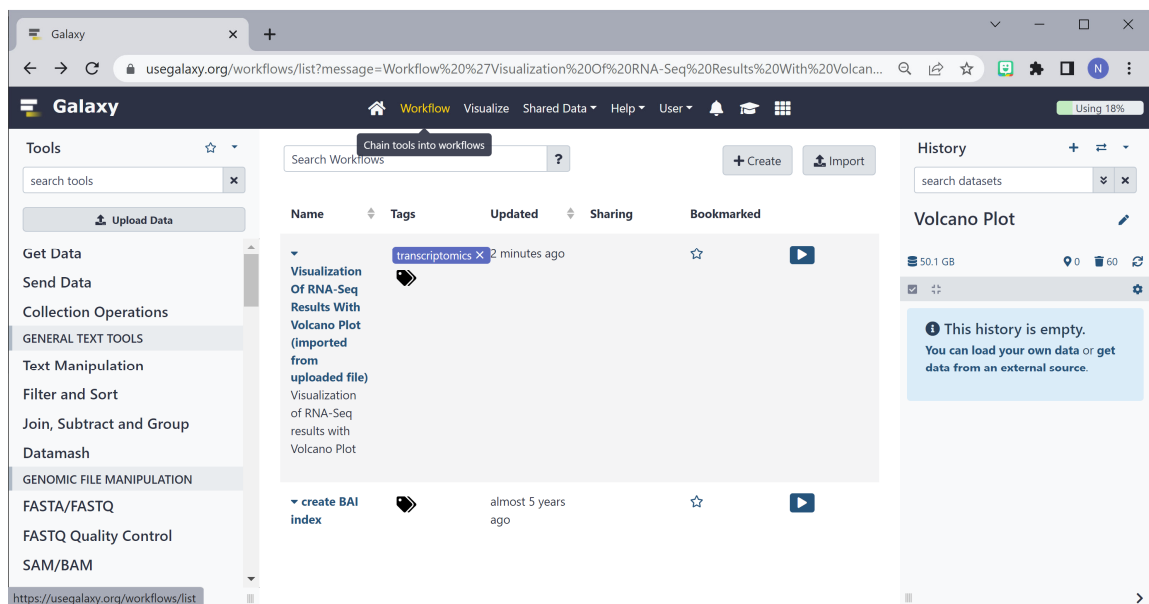



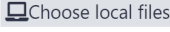





Figure 1. Galaxy platform with the “rna-seq-viz-with-volcanoplot.ga” workflow loaded.

4. Click the run icon  in the “Visualization of RNA-Seq Results With Volcano Plot”. We will need to provide two files: one that contains data on all genes in terms of fold-change, log₂-transformed – see the lecture as to why we work in log₂ space for fold-change data, p-values/FDR values and gene names. The second data file contains the genes we would like to highlight.

Upload the two files in the “DE results” area and in the “Volcano genes” area, respectively. This is a bit cumbersome: click the folder open icon , then the upload icon , then “Choose local files” with the  button. Locate your downloaded “K20dvsw.csv” file and select it. Click , then cancel out of the upload screen once upload has occurred. Then click the folder icon again to select the “K20dvsw.csv” dataset in the upload list. Do the same for the “Volcano genes”, by uploading the “volcano_genes_K20dvsw_top10.txt” file.

5. Now click **Run Workflow**.

What happens when we do this?

6. The issue is that we have used the workflow provided in the tutorial, and the columns provided with the tutorial example don't match those from our own "K20dvsw.csv" dataset. Go back to the workflow by clicking on the Workflow tab. Your screen will appear as in **Figure 1**. Click on the "Visualization of RNA-Seq Results With Volcano Plot" workflow name, then on the Edit () option. Click on the "3: Volcano Plot, highlighting significant" card, as shown in **Figure 3**, and scroll down the available options on the right. Change the columns to 6, 5, 2, and 1, respectively. Click the Save icon ()

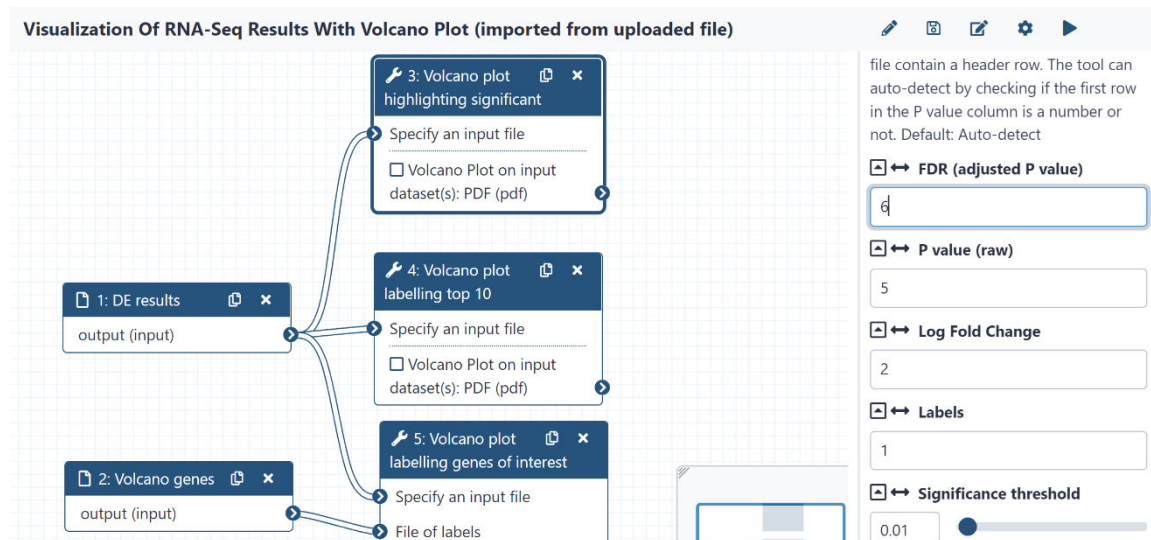




Figure 2. Changing the workflow column parameters to be compatible with our uploaded dataset. These changes correspond to the columns in our dataset as shown in **Figure 3**.

Column	①	②	③	④	⑤	⑥
	A	B	C	D	E	F
1		logFC	logCPM	F	PValue	FDR
2	AT4G15540	1.136016	7.222997	200.8955	2.33E-15	4.78E-11
3	AT1G62310	0.789006	7.705722	189.2937	5.30E-15	5.44E-11
4	AT4G37540	-2.47711	3.434612	169.9941	2.31E-14	1.30E-10
5	AT1G20490	1.899211	4.007012	168.7801	2.54E-14	1.30E-10

Figure 3. Column headings in K20dvsw.csv – the columns must match those in the workflow.

7. You'll have to change the columns in cards 4 and 5, too. When you click the Run  icon, you'll be asked to choose the datasets again, but you'll see a new object created in your History area, called something like "Volcano Plot on data x and data y". Click on the eye  icon to see the volcano plot!

Examine the plot for differentially expressed genes. The plot was generated with ggplot in R, which we used in Labs 1 and 2. Can you figure out how to access the R code used to generate the plot? Hint: view the tool source for the 5th card shown in Figure 2. Also, unhide files in the history section!

Lab Quiz
Question 1

R

We will explore generating some heatmaps, and heatmaps with clustergrams in this part of the lab. We'll use a "translatome" RNA-seq dataset from a part of a developing Arabidopsis flower as a test dataset, with RNA-seq data from the whole flower translatome as a control sample. See the lab discussion video for a bit more background on this dataset if you're interested!

1. Upload the "rpkmDFeByg_new.xlsx" and "DEGs.xlsx" to your Jupyter notebook as per Step 3 of the R part of Lab 1, or use the pre-populated version available on Coursera.
2. As for last week's lab, we'll load some packages that will help us with our goal of creating some plots this week.

```
# Install these packages onto JupyterHub
# Packages to help tidy our data
library(tidyverse)
# Packages for the graphical analysis section
library(repr)
# packages used for loading data from Excel
library(readxl)
```

You will receive some messages that these packages have been attached successfully.

3. Load the datasets from your working directory, which should be '/home/jovyan/work' if you're using Coursera's Jupyter Hub.

```
# read in FPKM-summarized expression data from an Excel file
expression_data.df <- read_excel("rpkmDFeByg_new.xlsx", sheet=1)
str(expression_data.df)

# read in an output from DESeq2 for our experiment
DEGs.df <- read_excel("DEGs.xlsx", sheet=1)
str(DEGs.df)
```

New names:

```
• `` -> ``.1`
```

```
tibble [145 × 5] (S3: tbl_df/tbl/data.frame)
 $ ...1      : chr [1:145] "AT1G01010" "AT1G01020" "AT1G01030" "AT1G01040" ...
 $ AP3_fl4a: num [1:145] 52.57 141.22 4.57 131.86 67.24 ...
 $ AP3_fl4b: num [1:145] 24.082 68.744 0.854 87.88 62.842 ...
 $ Tl_fl4a : num [1:145] 393.1 520.9 80.9 500.1 57.2 ...
 $ Tl_fl4b : num [1:145] 369.2 294.3 57.5 451.1 244.1 ...
```

New names:

```
• `` -> ``.1`
```

```
tibble [145 × 7] (S3: tbl_df/tbl/data.frame)
 $ ...1      : chr [1:145] "AT1G01010" "AT1G01020" "AT1G01030" "AT1G01040" ...
 $ AP3_TRL_baseMean: num [1:145] 50.42 87.67 7.98 345.81 5.57 ...
 $ AP3_TRL_logFC   : num [1:145] -0.873 0.509 -2.28 0.369 1.241 ...
 $ AP3_TRL_lfcSE   : chr [1:145] "0.53971517532179103" "0.44626080509107102"
"1.36532230617628" "0.245600414864767" ...
 $ AP3_TRL_stat    : chr [1:145] "-1.61801685846127" "1.14098613980372" "-
1.6695858376403001" "1.5015181645092699" ...
 $ AP3_TRL_pvalue  : chr [1:145] "0.105658964567623" "0.25387569083294897"
"9.50013343606564E-2" "0.13322159208748199" ...
 $ AP3_TRL_FDR     : num [1:145] 0.154 0.302 0.148 0.181 0.462 ...
```

4. In order to be able to filter genes based on significance values, we're going to merge the data frame containing expression values with the one containing significance levels (generated DESeq2, as discussed in the lecture) using the AGI ID as a "key" to do so. The AGI IDs are found in an unlabeled column, which was renamed to "...1" when the data were imported from Excel into the respective data frames in the previous step.

```
# merging signifance values (FDRs) in DEG list into our
# expression_data data frame
expression_data_w_FDR.df <- merge(expression_data.df, DEGs.df,
by.x = "...1", by.y = "...1", all.x = TRUE, all.y = TRUE)
str(expression_data_w_FDR.df)
```

```
'data.frame':      145 obs. of  11 variables:
 $ ...1           : chr  "AT1G01010" "AT1G01020" "AT1G01030" "AT1G01040" ...
 $ AP3_fl4a       : num  52.57 141.22 4.57 131.86 67.24 ...
 $ AP3_fl4b       : num  24.082 68.744 0.854 87.88 62.842 ...
 $ Tl_fl4a        : num  393.1 520.9 80.9 500.1 57.2 ...
 $ Tl_fl4b        : num  369.2 294.3 57.5 451.1 244.1 ...
 $ AP3_TRL_baseMean: num  50.42 87.67 7.98 345.81 5.57 ...
 $ AP3_TRL_logFC   : num  -0.873 0.509 -2.28 0.369 1.241 ...
 $ AP3_TRL_lfcSE   : chr  "0.53971517532179103" "0.44626080509107102" "1.36532230617628" "
0.245600414864767" ...
 $ AP3_TRL_stat    : chr  "-1.61801685846127" "1.14098613980372" "-1.6695858376403001" "1.
5015181645092699" ...
 $ AP3_TRL_pvalue  : chr  "0.105658964567623" "0.25387569083294897" "9.50013343606564E-2"
"0.13322159208748199" ...
 $ AP3_TRL_FDR     : num  0.154 0.302 0.148 0.181 0.462 ...
```

5. Often, we would like to generate lists of genes depending on some sort of cut-off. In the following two cases, we're generating lists of genes and their associated significance values such that the two lists have an FDR better than 10% or 1%, respectively, and a fold-change cut-off of greater than 2-fold "up" or "down".

```
# create a subset of genes that have an FDR rate of less
# than 10% and 2-fold change or more

FDR10.df <- subset(expression_data_w_FDR.df, AP3_TRL_FDR < .1 &
abs(AP3_TRL_logFC) > 1)

str(FDR10.df)

# create a subset of genes that have an FDR rate of less
# than 1% and 2-fold change or more

FDR1.df <- subset(expression_data_w_FDR.df, AP3_TRL_FDR < .01 &
abs(AP3_TRL_logFC) > 1)

str(FDR1.df)
```

What do you notice about all the AP3_TRL_FDR values in the FDR10.df and FDR1.df data frames? Hint: use `head(FDR10.df, 50)` or `head(FDR1.df, 50)` to examine the values in the FDR columns of those two data frames.

6. We'll now prune down our much larger data frame to remove all the extra information we added about FDRs, p-values etc.

```
# create new data frame with just expression values of FDR10 or
FDR1 genes

top_FDR10.df <- FDR10.df[ , c(1,2,3,4,5)]

str(top_FDR10.df)

top_FDR1.df <- FDR1.df[ , c(1,2,3,4,5)]

str(top_FDR1.df)
```

```
'data.frame':      39 obs. of  5 variables:
 $ ...1      : chr  "AT1G01050" "AT1G01060" "AT1G01070" "AT2G01008" ...
 $ AP3_fl4a: num  506.3 117.8 9.6 37.9 411532.7 ...
 $ AP3_fl4b: num  4.30e+02 6.64e+01 4.49 4.72e+01 4.43e+05 ...
 $ Tl_fl4a : num  9.14e+03 1.26e+03 3.68e+02 9.56 3.72e+05 ...
 $ Tl_fl4b : num  8016.3 1231.8 307.8 27.2 374844.6 ...
'data.frame':      29 obs. of  5 variables:
 $ ...1      : chr  "AT1G01050" "AT1G01060" "AT1G01070" "AT2G01008" ...
 $ AP3_fl4a: num  506.3 117.8 9.6 37.9 411532.7 ...
 $ AP3_fl4b: num  4.30e+02 6.64e+01 4.49 4.72e+01 4.43e+05 ...
 $ Tl_fl4a : num  9.14e+03 1.26e+03 3.68e+02 9.56 3.72e+05 ...
 $ Tl_fl4b : num  8016.3 1231.8 307.8 27.2 374844.6 ...
```

7. In order to generate a heatmap, we can use the built-in heatmap function of R. It requires a data matrix, not a data frame, and we can generate that with the `as.matrix()` function. The output is shown in **Figure 4**.

```
# the heatmap function requires a data matrix

m <- (as.matrix(top_FDR10.df[, -1]))

rownames(m) <- top_FDR10.df$...1

heatmap(m, Colv=NA, Rowv=NA, col=rev(heat.colors(256)), mar =
c(8, 6))
```

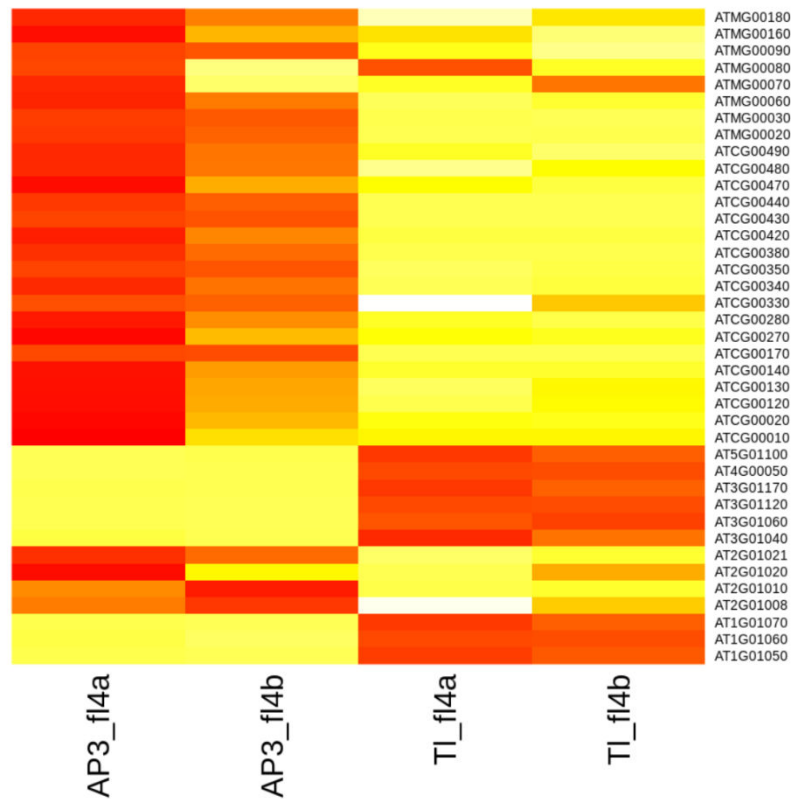


Figure 4. Heatmap of genes with an FDR of 10%.

8. We can generate a similar heatmap for the FDR 1% genes:

```
m <- (as.matrix(top_FDR1.df[, -1]))
rownames(m) <- top_FDR1.df$...1
heatmap(m, Colv=NA, Rowv=NA, col=rev(heat.colors(256)), mar =
c(8, 6))
```

9. We might want to cluster both the rows and the columns according to expression pattern similarity and depict the similarity with a clustergram. This kind of clustering is called hierarchical clustering, and the most similar expression profiles (either across all genes, or across all samples for a given gene as specified by omitting the `Colv = NA` or `Rowv = NA` parameters, respectively) are grouped with branch lengths reflecting how similar the profiles are: the shorter, the more similar. See **Figure 5**.

Check out some additional parameters here <https://r-graph-gallery.com/215-the-heatmap-function.html>. Can you figure out how to use different palettes or row side colours?

We've added another column in the DEGs.xlsx file, which is a log10-transformation of the baseMean column, which is a column that is included in DESeq2 output and that represents the average of the normalized count values, dividing by size factors, taken over all samples. You could imagine, however, that the column could contain other information, like Gene Ontology category for the gene in question...

We'll use that additional column to create an additional visual cue as to the overall level of expression of our genes, as the default behaviour of the heatmap function is to normalize the levels in each row – this means within a row you can easily see high and low expression levels, but you can't directly compare expression levels between rows.

```
# let's add a sidebar with colours denoting baseMean (~average)
expression level, after log10-transformation
# We'll cluster by rows by excluding the Rowv = NA parameter
library(RColorBrewer) # we explored this library in Lab 1!
my_group <- as.numeric(FDR1.df[FDR1.df$...1 %in% rownames(m),
"Log_level"])

# You need to add 1 when selecting colour groups as our
log_level values can range from 0 .. n
colSide <- brewer.pal(9, "Greys")[my_group + 1]
heatmap(m, Colv=NA, RowSideColors=colSide,
col=rev(heat.colors(256)), scale = "row")

# Calculate the min, median, and max values
val_range <- c(min(m), median(m), max(m))

# Change the margins of the plot (the first is bottom margin)
par(mar = c(6, 4.1, 4.1, 2.1), xpd=TRUE)

# Plot a corresponding legend for colour range
legend(x=-0.1, y = 0.2, legend=c("min", "med",
"max"), fill=rev(heat.colors(3)), bty="n")

# Make the legend for the log_level of base mean expression
log_legend <- data.frame(log_level = unique(my_group),
                        log_col = unique(colSide),
                        stringsAsFactors = FALSE) %>%
# Sort ascending by log level
arrange(log_level)

legend(x=-0.1, y=0.05,
      legend=log_legend$log_level, fill=log_legend$log_col,
      title = "Log10 level", bty="n", ncol = 2)
```

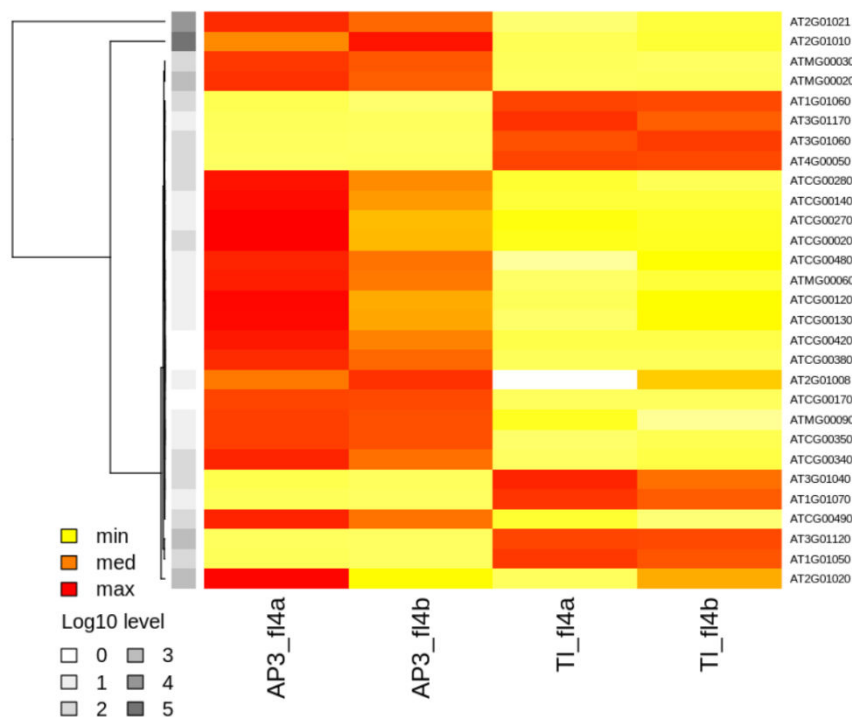



Figure 5. Heatmap of genes with an FDR level of 1% or better, and exhibiting a 2-fold change or higher when comparing the AP3 samples to the T1 samples. Rows are clustered by expression similarity, colours in first column denote the log10 of the baseMean value in the DEG file, which represents ~the average expression level across all samples.

Try turning off the default row-wise normalization to be able to visualize absolute expression levels by slightly changing the heatmap function to `heatmap(m, Colv=NA, RowSideColors=colSide, col=rev(heat.colors(256)), scale = "none")`

Check out <https://www.datanovia.com/en/lessons/heatmap-in-r-static-and-interactive-visualization/#r-base-heatmap-heatmap> for lots of additional tips and tricks to use with the heatmap function!

My eFP Browser

The Provart Lab's eFP Browser tool (Winter et al., 2007) is a widely used and highly cited tool in plant biology. It provides an intuitive interface for “electronic fluorescent pictograph” generation, which is a style of heatmap where each spot in the heatmap is represented by a depiction of the sample from which the expression data were generated. Let's look at it in action first before we generate our own eFP Browser view.

1. View the expression pattern of *ABI3* in “unstressed” tissues of Arabidopsis using the “electronic Fluorescent Pictograph (eFP)” Browser by going to <http://bar.utoronto.ca/efp/> and entering At3g24650 (the AGI ID for *ABI3*) in the Primary Gene ID box and clicking GO). This gene and its gene product are known to be involved in seed maturation.

Is the gene you're looking at expressed at a high level in a particular tissue (examine the colour scale and/or mouse over the tissue that exhibits the most expression – see **Figure 6**)? Hint: the eFP Browser will tell you in which “Data Source” the expression level is the highest. In addition, the small histogram shows how your gene is expressed relative to the average expression level distribution of all genes on the ATH1 microarray for this series. If you switch to the Klepikova Atlas, you can examine expression patterns based on RNA-seq data.

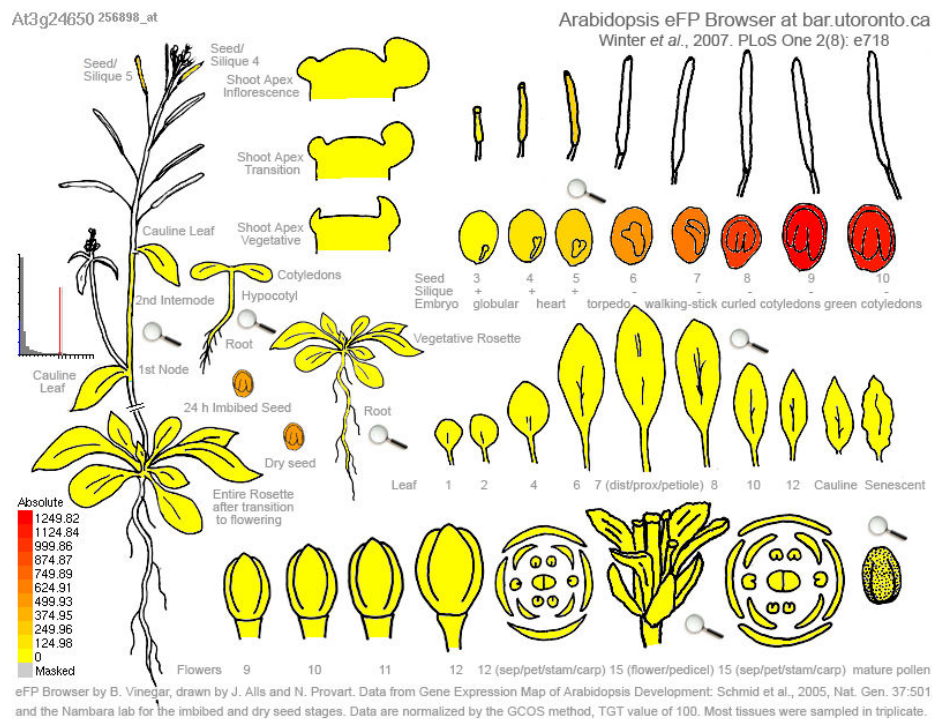


Figure 6: “Typical” eFP Browser output for the *ABI3* gene, showing strong expression in maturing seeds, where it is known to be involved in helping seeds prepare for desiccation.

- Now let's try our hand at making our own eFP image. We'll use a new interface developed by the Provart Lab that uses SVG images, instead of bitmap images as in the original eFP Browser. These have the advantage of not becoming pixelated if they're magnified. Go to https://bar.utoronto.ca/~dev/svg_editor_lite/ (Firefox is recommended) and click through to the CSV upload modal. Upload the “Lab3_test_data.csv” CSV file from the course website.

It is possible to use the (somewhat limited selection of) existing images in the interface to create an image that describes your experiment, or to upload your own SVG images. If you do this, there are guidelines as to how the SVG image must be created in terms of the <g> tags that are used to label parts of the image. In our test example, we'll use a derived dataset that looks at development of the carpel part of Arabidopsis in two different developmental stages and genetic backgrounds (one called *spt-12*, which contains a T-DNA insertional mutation in the *SPATULA* [*SPT*] gene or At4g36930, see more

information at <https://www.arabidopsis.org/servlets/TairObject?name=AT4G36930&type=locus>, and which has abnormal carpels). Transcriptomic data were generated and a small subset of these have been provided in the “Lab3_test_data.csv” file.

Use the interface to drag a “carpel” image to the canvas (Canvas > Add Element > Others). Use the Canvas > Add Text option to label your images. You can right click and “clone” to create additional copies of images if they’re active. For each image, use the Element Metadata area to specify the which samples should be associated with which element, and which samples are the controls (here the WT Col-0 samples are controls), as shown in **Figure 7**.

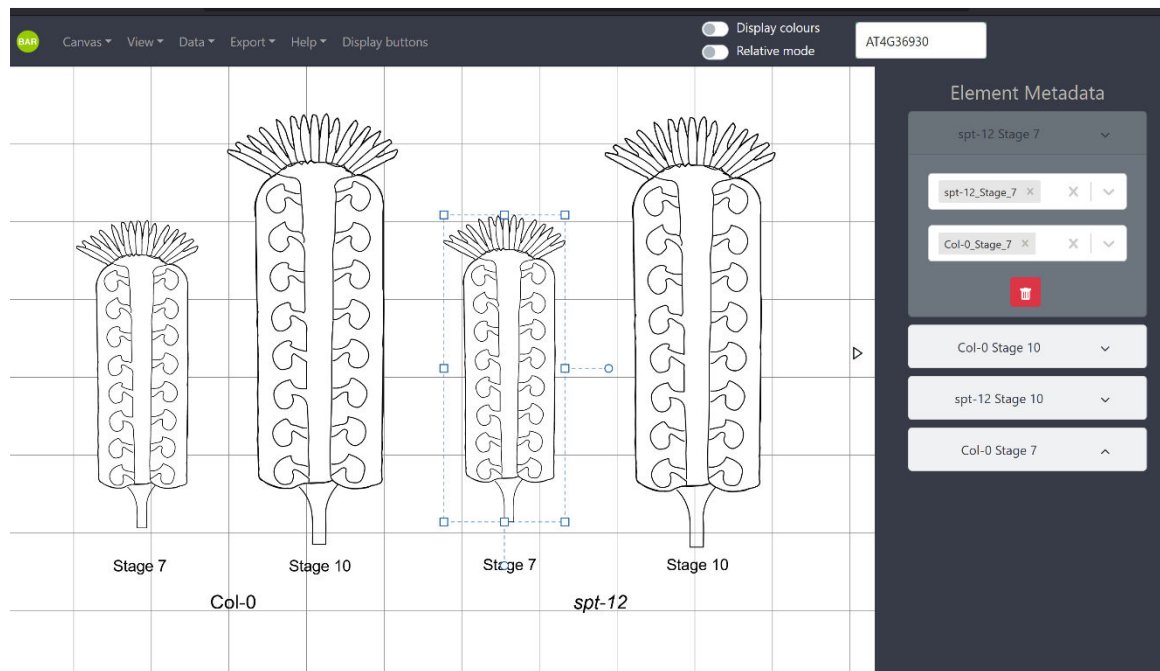


Figure 7. Creating a custom eFP image with the BAR’s SVG Editor. Element metadata are shown for the *spt-12*, Stage 7 sample pictograph, whereby the sample has been set to be “spt-12_Stage_7” and its corresponding control has been set to be “Col-0_Stage_7”.

- For the data that you uploaded in the provided CSV file, it is now possible to generate an eFP image. Simply choose the corresponding gene name in “Enter gene name here...” box. For instance, if we enter At4g36930 (i.e., *SPT*) in the box, and then turn on the “Display colours” option, we can generate an eFP image that looks like **Figure 8**.

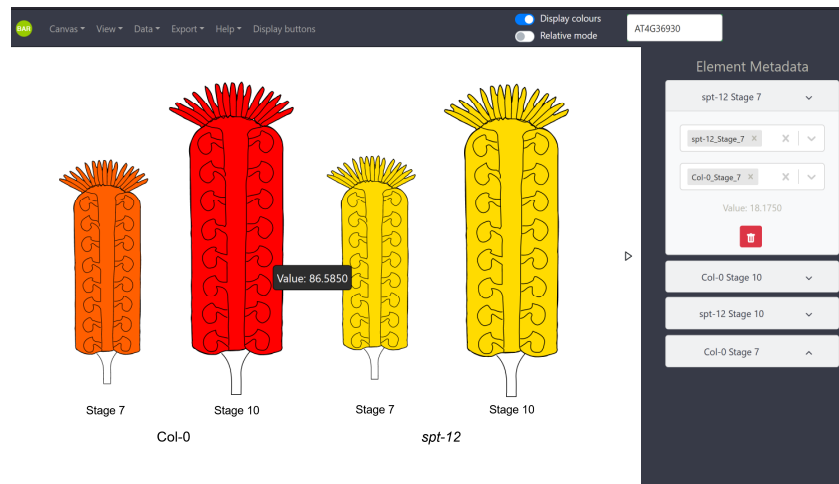


Figure 8. eFP image of *SPT* (At4g36930) expression in WT Col-0 carpels and the *spt-12* mutant, illustrating nicely the reduction in expression of *SPT* in carpels of the *spt-12* mutant.

Examine the expression patterns of the four genes provided in the “test_data.csv” file. *Do any genes exhibit interesting patterns?*

Lab Quiz
Question 3

End of Lab!

Lab 3 Objectives

By the end of Lab 3 (comprising the lab including its boxes, and the lecture), you should:

- understand design considerations for effective data visualization;
- be able to create a volcano plot with some gene expression data with Galaxy;
- be able to generate heatmaps with clustergrams in R;
- be able to create an eFP image for some gene expression data.

Do not hesitate to check with the Forums for this course on Coursera if you do not understand any of the above after reading the relevant material.

Additional Resources

Winter D, Vinegar B, Nahal H, Ammar R, Wilson GV, Provart NJ (2007). An 'Electronic Fluorescent Pictograph' Browser for Exploring and Analyzing Large-Scale Biological Data Sets. *PLoS One*. 2(8), e718.

Sections 16.1-16.4 in Chapter 16 “Clustering Methods and Statistics” in *Understanding Bioinformatics* by Marketa Zvelebil and Jeremy Baum, Garland Science, 2008. pp. 625-659.

Section 15.1 “Analysis of Large-scale Gene Expression Data” in Chapter 15 “Proteome and Gene Expression Analysis” in *Understanding Bioinformatics* by Marketa Zvelebil and Jeremy Baum, Garland Science, 2008. pp. 601-612.