# Problem Set 3. Due Thurs March 2 5pm

*Etienne Danis*

*Feb 23, 2017*

## Submission

1. Fork this repository to your own account
2. Make changes to the `README.Rmd` file (including the author field above).
3. Knit the file to HTML format and add it to your git repository (`git add`)
4. Submit a pull request with your Rmd and knitted HTML files.

## Overview

You will examine a time-course of mRNA expression of barley powdery mildew fungus in immunocompromised plants (*Arabidopsis*). The fungus Blumeria graminis f. sp. hordei (called *Bgh* from now on)

Counts were generated using the `maSigPro` software and deposited in NCBI GEO.

Some important identifiers for the data set:

| Name | Description |
|---|---|
| `A6` | *Bgh* isolate expressing other AVRA effectors |
| `K1` | *Bgh* isolate expressing the cognate AVRA1 effector for MLA1 |
| `pps` | *Arabidopsis* plants |
| `B12` | *Arabidopsis* plants expressing MLA1-HA |

We will download tables from GEO containing counts of mRNA abundance from both the fungal sample.

## Raw data

First we need load the data.

```
library(tidyverse)

# bgh data
bgh_url <- 'http://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE43163&format=file&file=GSE43163_CompleteCo

# 2 info and blank lines at top, skip them
raw_data <- read_tsv(bgh_url, skip = 2)
```

```
#> Warning: Missing column names filled in: 'X1' [1]
```

```
# the header for the first column is "NA", set it manually
names(raw_data)[1] <- 'gene.name'

raw_data <- as_data_frame(raw_data)
raw_data
```

```
#> # A tibble: 6,477 × 49
#>              gene.name B12_A6_6hpi_1 B12_A6_6hpi_2 B12_A6_6hpi_3
#>                  <chr>         <int>         <int>         <int>
#> 1             bgh04079            13            15            13
#> 2             bgh01634            31            10            24
#> 3  bghG000012000001001           121           119           145
#> 4  bghG000012000002001             3             2             0
#> 5             bgh00757           253           191           207
#> 6             bgh01273            45            27            48
#> 7             bgh01274            32            22            24
#> 8             bgh01277             9             5             5
#> 9             bgh06140            47            31            32
#> 10            bgh05774            42            24            34
#> # ... with 6,467 more rows, and 45 more variables: B12_A6_12hpi_1 <int>,
#> #   B12_A6_12hpi_2 <int>, B12_A6_12hpi_3 <int>, B12_A6_18hpi_1 <int>,
#> #   B12_A6_18hpi_2 <int>, B12_A6_18hpi_3 <int>, B12_A6_24hpi_1 <int>,
#> #   B12_A6_24hpi_2 <int>, B12_A6_24hpi_3 <int>, B12_K1_6hpi_1 <int>,
#> #   B12_K1_6hpi_2 <int>, B12_K1_6hpi_3 <int>, B12_K1_12hpi_1 <int>,
#> #   B12_K1_12hpi_2 <int>, B12_K1_12hpi_3 <int>, B12_K1_18hpi_1 <int>,
#> #   B12_K1_18hpi_2 <int>, B12_K1_18hpi_3 <int>, B12_K1_24hpi_1 <int>,
#> #   B12_K1_24hpi_2 <int>, B12_K1_24hpi_3 <int>, pps_A6_6hpi_1 <int>,
#> #   pps_A6_6hpi_2 <int>, pps_A6_6hpi_3 <int>, pps_A6_12hpi_1 <int>,
#> #   pps_A6_12hpi_2 <int>, pps_A6_12hpi_3 <int>, pps_A6_18hpi_1 <int>,
#> #   pps_A6_18hpi_2 <int>, pps_A6_18hpi_3 <int>, pps_A6_24hpi_1 <int>,
#> #   pps_A6_24hpi_2 <int>, pps_A6_24hpi_3 <int>, pps_K1_6hpi_1 <int>,
#> #   pps_K1_6hpi_2 <int>, pps_K1_6hpi_3 <int>, pps_K1_12hpi_1 <int>,
#> #   pps_K1_12hpi_2 <int>, pps_K1_12hpi_3 <int>, pps_K1_18hpi_1 <int>,
#> #   pps_K1_18hpi_2 <int>, pps_K1_18hpi_3 <int>, pps_K1_24hpi_1 <int>,
#> #   pps_K1_24hpi_2 <int>, pps_K1_24hpi_3 <int>
```

# Problems

## Problem 1

Now that the raw data are loaded, your first task is to tidy the data with `tidyr`. The `cleaned_data` should look something like this:

```
# A tibble: 310,896 × 6
             gene.name plant fungus time.point   rep value
*                <chr> <chr>  <chr>      <chr> <chr> <int>
1             bgh04079   B12     A6       6hpi     1    13
2             bgh01634   B12     A6       6hpi     1    31
3  bghG000012000001001   B12     A6       6hpi     1   121
4  bghG000012000002001   B12     A6       6hpi     1     3
5             bgh00757   B12     A6       6hpi     1   253
6             bgh01273   B12     A6       6hpi     1    45
7             bgh01274   B12     A6       6hpi     1    32
8             bgh01277   B12     A6       6hpi     1     9
9             bgh06140   B12     A6       6hpi     1    47
10            bgh05774   B12     A6       6hpi     1    42
# ... with 310,886 more rows
```

The key steps are to `gather()` the data into key-value pairs and then `separate()` the information from the

key column into new columns.

```
# add tidying code here
library(tidyverse)
raw_data %>% gather(key,value,-gene.name) %>% separate(key, into = c('plant', 'fungus', 'time.point', '
cleaned_data
```

```
#> # A tibble: 310,896 × 6
#>              gene.name plant fungus time.point   rep value
#> *                <chr> <chr>  <chr>      <chr> <chr> <int>
#> 1             bgh04079   B12     A6       6hpi     1    13
#> 2             bgh01634   B12     A6       6hpi     1    31
#> 3  bghG000012000001001   B12     A6       6hpi     1   121
#> 4  bghG000012000002001   B12     A6       6hpi     1     3
#> 5             bgh00757   B12     A6       6hpi     1   253
#> 6             bgh01273   B12     A6       6hpi     1    45
#> 7             bgh01274   B12     A6       6hpi     1    32
#> 8             bgh01277   B12     A6       6hpi     1     9
#> 9             bgh06140   B12     A6       6hpi     1    47
#> 10            bgh05774   B12     A6       6hpi     1    42
#> # ... with 310,886 more rows
```

```
# report the cleaned_data by just naming it, uncomment the following line:
# cleaned_data
```

## Problem 2

You need to translate what you learned on the command line into the R world.

Which plant has the highest expression of any gene in the 6hpi time point?

```
raw_data | awk '$4 == "6hpi"' | sort -k6nr | head -n 1 | cut -f2
```

```
# translate to dplyr code

filter(cleaned_data, time.point=="6hpi") %>% arrange(desc(value)) %>% select(plant) %>%  filter(row_numb
```

```
#> # A tibble: 1 × 1
#>   plant
#>   <chr>
#> 1   pps
```

Which plant / fungus pair has the highest expression in the 18hpi time point?

```
raw_data | awk '$4 == "18hpi"' | sort -k6nr | head -n 1 | cut -f2,3
```

```
# translate to dplyr code

filter(cleaned_data, time.point=="18hpi") %>% arrange(desc(value)) %>% select(plant, fungus) %>% filter
```

```
#> # A tibble: 1 × 2
#>   plant fungus
#>   <chr>  <chr>
#> 1   pps     A6
```

## Problem 3

Identify the top 3 most consistently differentially expressed genes between the earliest and latest time points for each combination of `plant` and `fungus` strains.

- "Differential expression"" is the difference between `value` (i.e., gene expression level) between time points.
- "Consistency" is the smallest variance in `value` between replicates.

### Strategy

1. Create a new table from the cleaned data by moving each `hpi` value to a new column name, with counts for each in the column (hint: use a `tidyr` verb). It is helpful to reformat the `hpi` values by converting from e.g. `6hpi` to `hpi.6`. You can use `mutate` to do this, i.e.:

```
library(stringr)
# Version_1
cleaned_data %>% mutate(time.point = str_replace(time.point, 'hpi', ''), time.point = str_c('hpi.', time
cleaned_data_hpi
# Version_2: cleaned_data %>% mutate(time.value = str_replace(time.point, 'hpi', ''), time.hpi = str_c(
```

2. Create a new column containing the expression difference between the relevant time points.

```
library(stringr)
# Version_1
cleaned_data %>% mutate(time.point = str_replace(time.point, 'hpi', ''), time.point = str_c('hpi.', time
cleaned_data_hpi %>% spread(time.point,value) %>% mutate(expr.diff = abs(hpi.24 - hpi.6), expr.diff.var
expr_data
```

```
#> # A tibble: 77,724 × 10
#>    gene.name plant fungus   rep hpi.12 hpi.18 hpi.24 hpi.6 expr.diff
#>        <chr> <chr>  <chr> <chr>  <int>  <int>  <int> <int>     <int>
#> 1   bgh00001   B12     A6     1    128    265    261   218        43
#> 2   bgh00001   B12     A6     2     53    191    323   153       170
#> 3   bgh00001   B12     A6     3     78    177    251   180        71
#> 4   bgh00001   B12     K1     1     70    188    107   202        95
#> 5   bgh00001   B12     K1     2     64    286    184    99        85
#> 6   bgh00001   B12     K1     3     52    157    163   160         3
#> 7   bgh00001   pps     A6     1    153    259    115   185        70
#> 8   bgh00001   pps     A6     2     71    308    113   102        11
#> 9   bgh00001   pps     A6     3    106    212    204   154        50
#> 10  bgh00001   pps     K1     1     29    127    138    53        85
#> # ... with 77,714 more rows, and 1 more variables: expr.diff.var <dbl>
```

3. Calculate summary statistics (mean and variance) of the expression differences by grouping (hint) the `gene.name`, and `plant` columns.

```
# Using substraction to determine the differentially expressed genes
expr_data %>% group_by(plant, fungus, gene.name) %>% summarise(expr.diff.mean = mean(expr.diff), expr.di
expr_data_mean
```

```
#> Source: local data frame [25,908 x 5]
#> Groups: plant, fungus [?]
#>
#>    plant fungus gene.name expr.diff.mean expr.diff.var
#>    <chr>  <chr>     <chr>          <dbl>         <dbl>
```

```
#> 1    B12     A6  bgh00001       94.66667    4452.33333
#> 2    B12     A6  bgh00002       95.66667    4024.33333
#> 3    B12     A6  bgh00003       15.00000      49.00000
#> 4    B12     A6  bgh00004        3.00000       7.00000
#> 5    B12     A6  bgh00005       16.33333     226.33333
#> 6    B12     A6  bgh00006       38.00000     729.00000
#> 7    B12     A6  bgh00007      379.66667   10770.33333
#> 8    B12     A6  bgh00008      154.66667    1850.33333
#> 9    B12     A6  bgh00009       13.00000      12.00000
#> 10   B12     A6  bgh00010       33.66667      44.33333
#> # ... with 25,898 more rows
```

4. Sort by these statistics and use the `dplyr` verb `slice` to pull the ones you want (i.e., the top 3). Note you will have to remove `gene.name` from the grouping so that sorting works.

```
# Previous method used: I sorted first by expr.diff.mean and then by expr.diff.var. This gave me very s
#expr_data_mean %>% arrange(desc(expr.diff.mean)) %>%  arrange(expr.diff.var) %>% slice(1:3) %>% select
#top3_genes %>% tbl_df() %>% print (n=12)
```

```
# Sort by expr.diff.mean last
expr_data_mean %>% arrange(expr.diff.var) %>% arrange(desc(expr.diff.mean)) %>%  slice(1:3) %>% select(
top3_genes %>% tbl_df() %>% print (n=12)
```

```
#> # A tibble: 12 × 5
#>    plant fungus gene.name expr.diff.mean expr.diff.var
#>    <chr>  <chr>     <chr>          <dbl>         <dbl>
#> 1    B12     A6  bgh01942       4988.333     726785.33
#> 2    B12     A6  bgh00377       2539.667     236402.33
#> 3    B12     A6  bgh05405       2310.667     367852.33
#> 4    B12     K1  bgh01942       5202.333    2197082.33
#> 5    B12     K1  bgh05405       3309.333    2173654.33
#> 6    B12     K1  bgh00377       2902.000     836091.00
#> 7    pps     A6  bgh01942       5105.000    2448477.00
#> 8    pps     A6  bgh00377       2801.667     678297.33
#> 9    pps     A6  bgh05405       2706.000     759601.00
#> 10   pps     K1  bgh01942       2149.333      20658.33
#> 11   pps     K1  bgh00377       1106.000      16624.00
#> 12   pps     K1  bgh02634       1092.667     422292.33
```

## Problem 4

Now examine the above final data frame above and write a few sentences putting inline code in least 3 places. For example, There are 32 rows of data in `mtcars`.

My tidy dataset has 5 columns and 12 rows. The name of the columns are plant, fungus, gene.name, expr.diff.mean, expr.diff.var.

The highest difference in expression of a gene in the B12 plant expressing the immune receptor MLA1-HA treated with other effectors than the AVRA1 effector is: 4988.333.

The highest difference im expression of a gene in the pps plant (partially immunocompromised) treated with effectors different from the AVRA1 effector is: 5105.0.

The highest difference in expression of a gene in the B12 plant expressing the immune receptor MLA1-HA treated with the AVRA1 effector is: 5202.333.

The highest difference in expression of a gene in the pps plant (partially immunocompromised) treated with effectors different from the AVRA1 effector is: 2149.333.

The top 3 genes for the pair (B12 plant + A6 fungus) are: bgh01942, bgh00377, bgh05405.

The top 3 genes for the pair (B12 plant + K1 fungus) are: bgh01942, bgh05405, bgh00377.

The top 3 genes for the pair (B12 plant + A6 fungus) are: bgh01942, bgh00377, bgh05405.

The top 3 genes for the pair (B12 plant + K1 fungus) are: bgh01942, bgh00377, bgh02634.

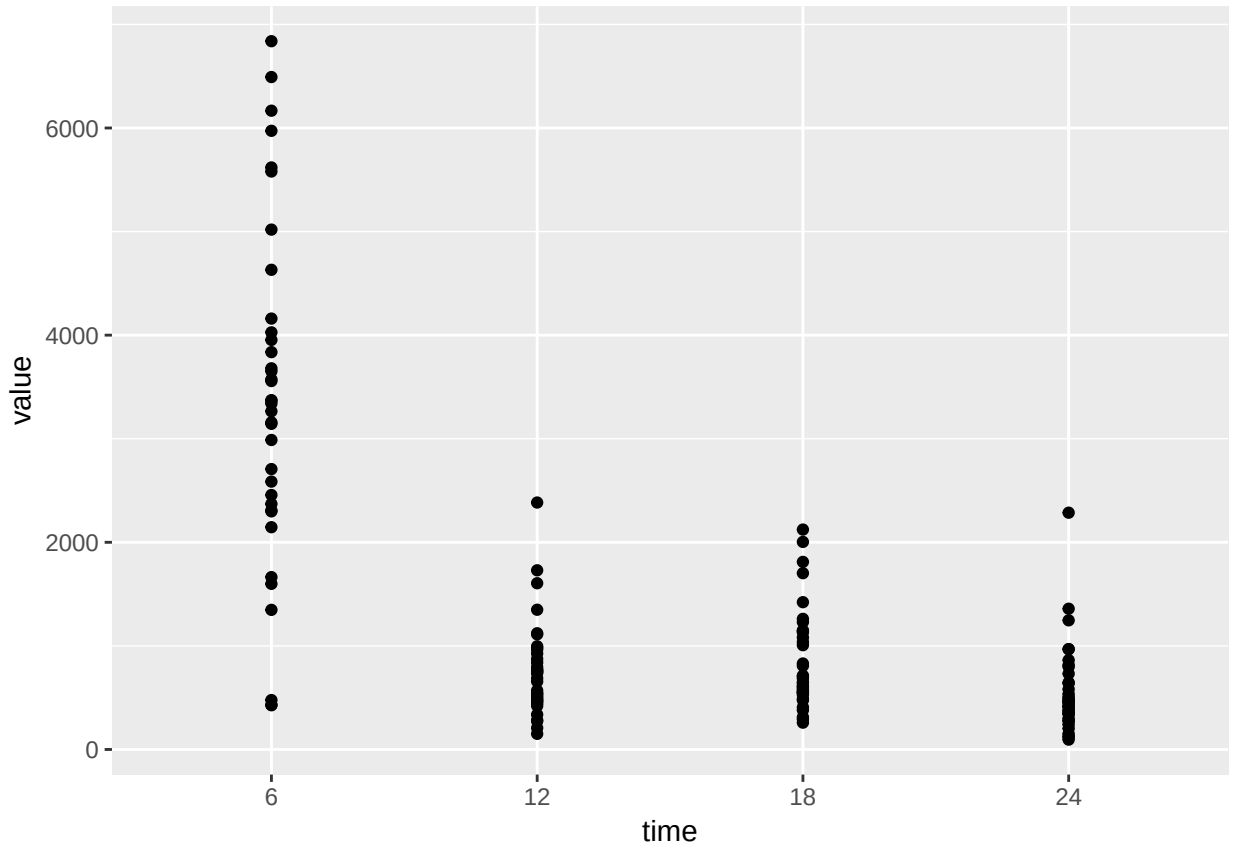
## Problem 5

Plot the expression (`value`) by time (`hpi`) for the above genes. Format the plot as follows:

```
# 3 top genes per pair plant/fungus, 4 pairs, triplicates, 4 time points = 3 x 4 x 3 x 4 = 144
# First, create for each pair plant/fungus, the list of top 3 genes differentially expressed and save t
cleaned_data %>% mutate(time.point = str_replace(time.point, 'hpi', '')) %>%  select(gene.name, plant, 

top3_genes %>% subset(., plant =="B12" & fungus =="A6", select=c(plant, fungus, gene.name)) -> top3_gene
top3_genes %>% subset(., plant =="B12" & fungus =="K1", select=c(plant, fungus, gene.name)) -> top3_gene
top3_genes %>% subset(., plant =="pps" & fungus =="A6", select=c(plant, fungus, gene.name)) -> top3_gene
top3_genes %>% subset(., plant =="pps" & fungus =="K1", select=c(plant, fungus, gene.name)) -> top3_gene

# Collect the time points and replicate values for the top 3 genes for each pair plant/fungus
# For the B12 plant and the A6 fungus
filter(data_selected, gene.name %in% top3_genes_for_B12_A6$gene.name) -> data_filtered_by_gene_name_B12_
filter(data_filtered_by_gene_name_B12_A6, plant %in% top3_genes_for_B12_A6$plant) -> data_filtered_by_ge
filter(data_filtered_by_gene_name_and_plant_B12_A6, fungus %in% top3_genes_for_B12_A6$fungus) -> data_fe

# For the B12 plant and the K1 fungus
filter(data_selected, gene.name %in% top3_genes_for_B12_K1$gene.name) -> data_filtered_by_gene_name_B12_
filter(data_filtered_by_gene_name_B12_K1, plant %in% top3_genes_for_B12_K1$plant) -> data_filtered_by_ge
filter(data_filtered_by_gene_name_and_plant_B12_K1, fungus %in% top3_genes_for_B12_K1$fungus) -> data_fe

# For the pps plant and the A6 fungus
filter(data_selected, gene.name %in% top3_genes_for_pps_A6$gene.name) -> data_filtered_by_gene_name_pps_
filter(data_filtered_by_gene_name_pps_A6, plant %in% top3_genes_for_pps_A6$plant) -> data_filtered_by_ge
filter(data_filtered_by_gene_name_and_plant_pps_A6, fungus %in% top3_genes_for_pps_A6$fungus) -> data_fe

# For the pps plant and the K1 fungus
filter(data_selected, gene.name %in% top3_genes_for_pps_K1$gene.name) -> data_filtered_by_gene_name_pps_
filter(data_filtered_by_gene_name_pps_K1, plant %in% top3_genes_for_pps_K1$plant) -> data_filtered_by_ge
filter(data_filtered_by_gene_name_and_plant_pps_K1, fungus %in% top3_genes_for_pps_K1$fungus) -> data_fe

#Combine all the dataframes in 1 for the graphs
all_data_for_graphs <- rbind(data_for_B12_and_A6_pair, data_for_B12_and_K1_pair, data_for_pps_and_A6_pai
```

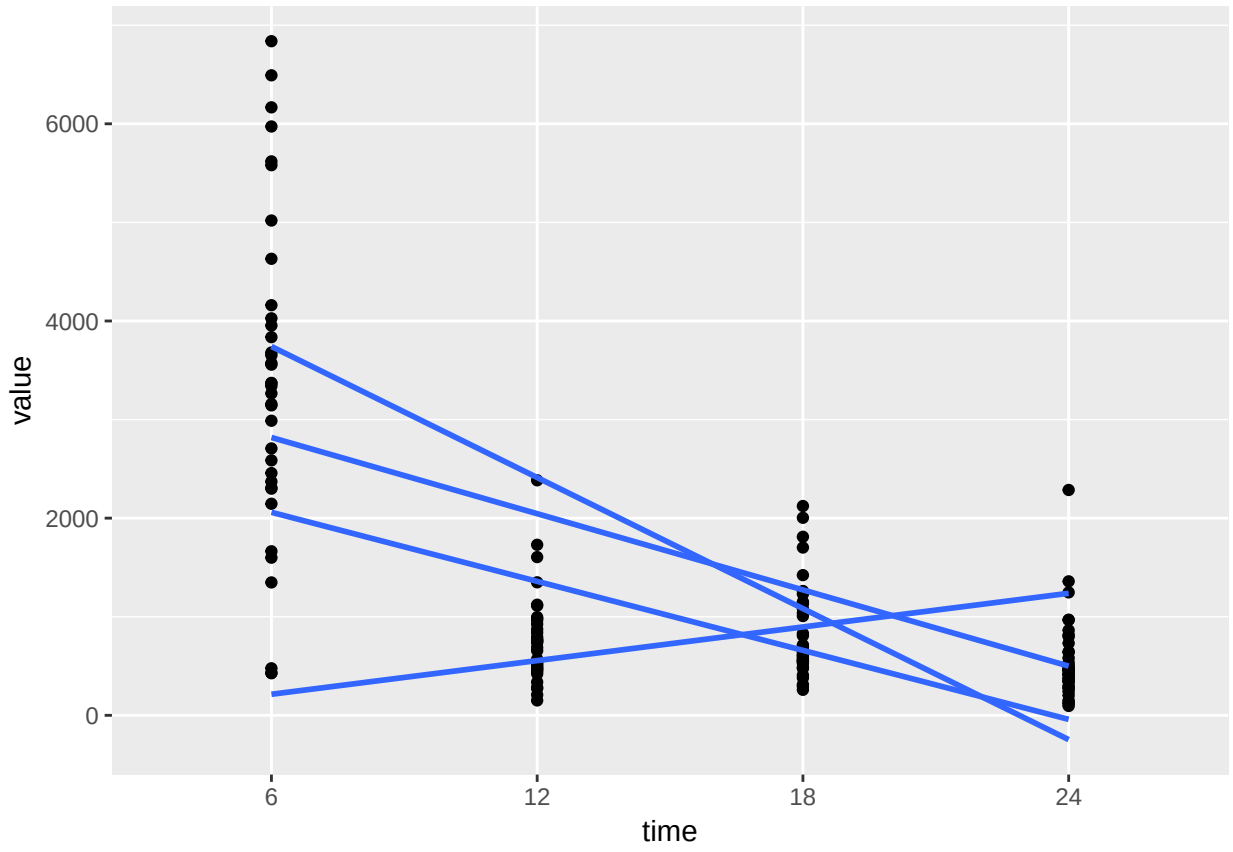
1. Plot each point.

```
library(ggplot2)
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value)) + geom_point() + xlab("time")
```



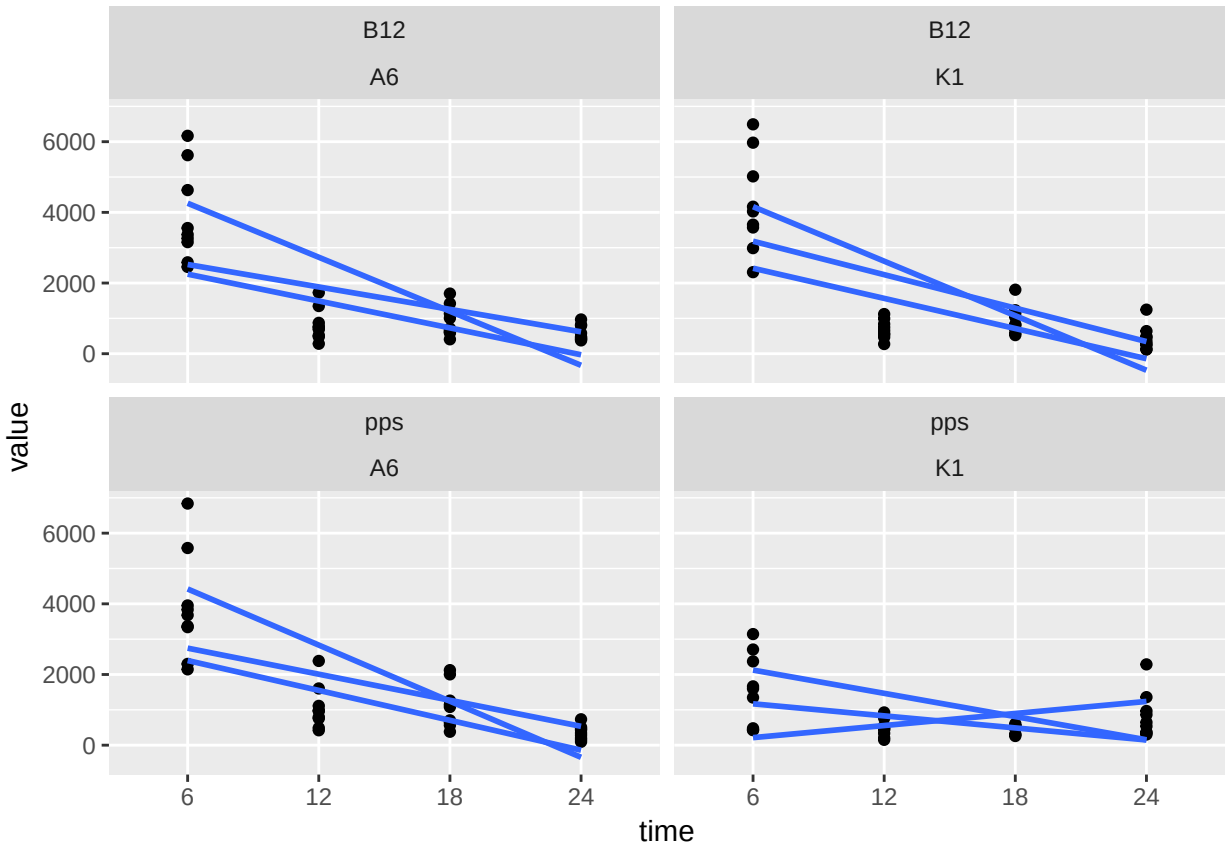

2. Connect the points with a smooth line

```
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value, group = gene.name)) +
 geom_point()  + xlab("time") + geom_smooth(method = 'lm', se = F)
```



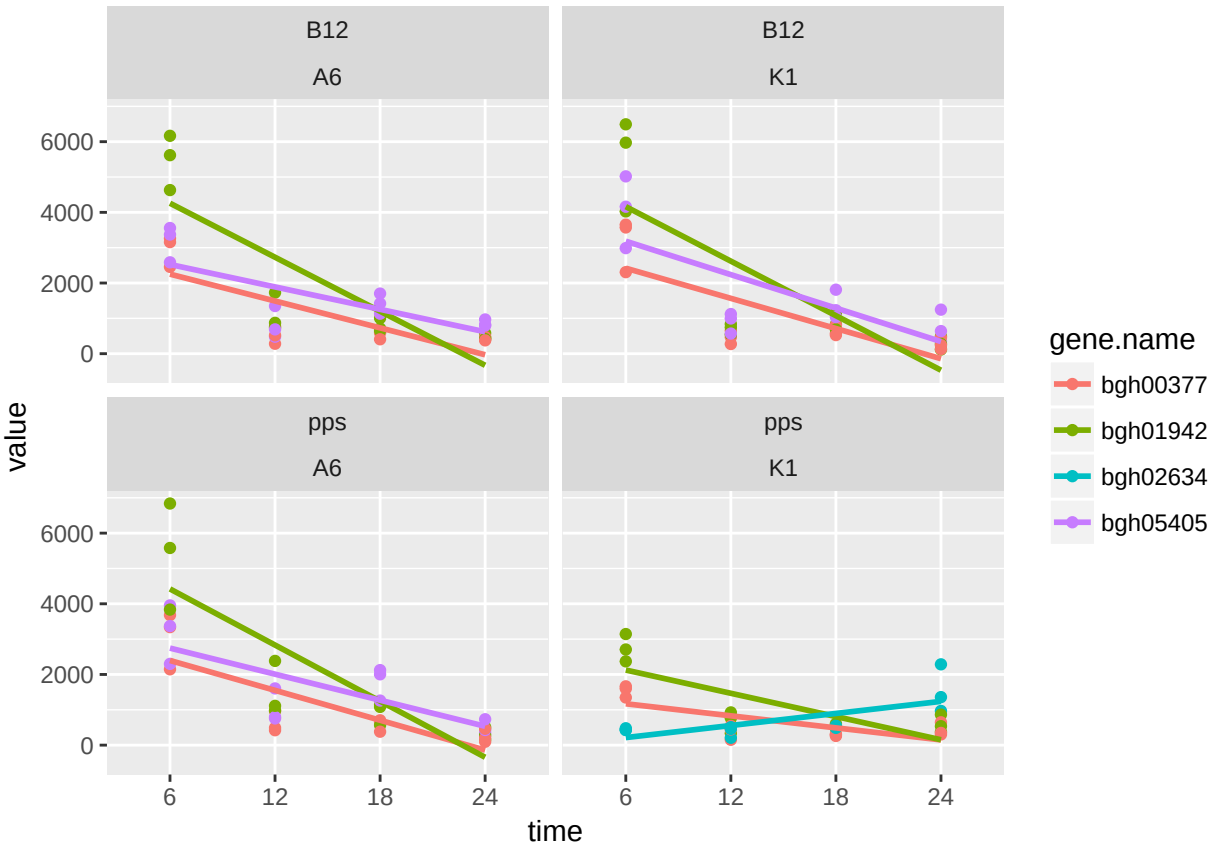

3. Create a facet for each `plant` / `bgh` pair

```
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value, group = gene.name)) +
  geom_point() + xlab("time") + facet_wrap(~plant~fungus) + geom_smooth(method = 'lm', se = F)
```



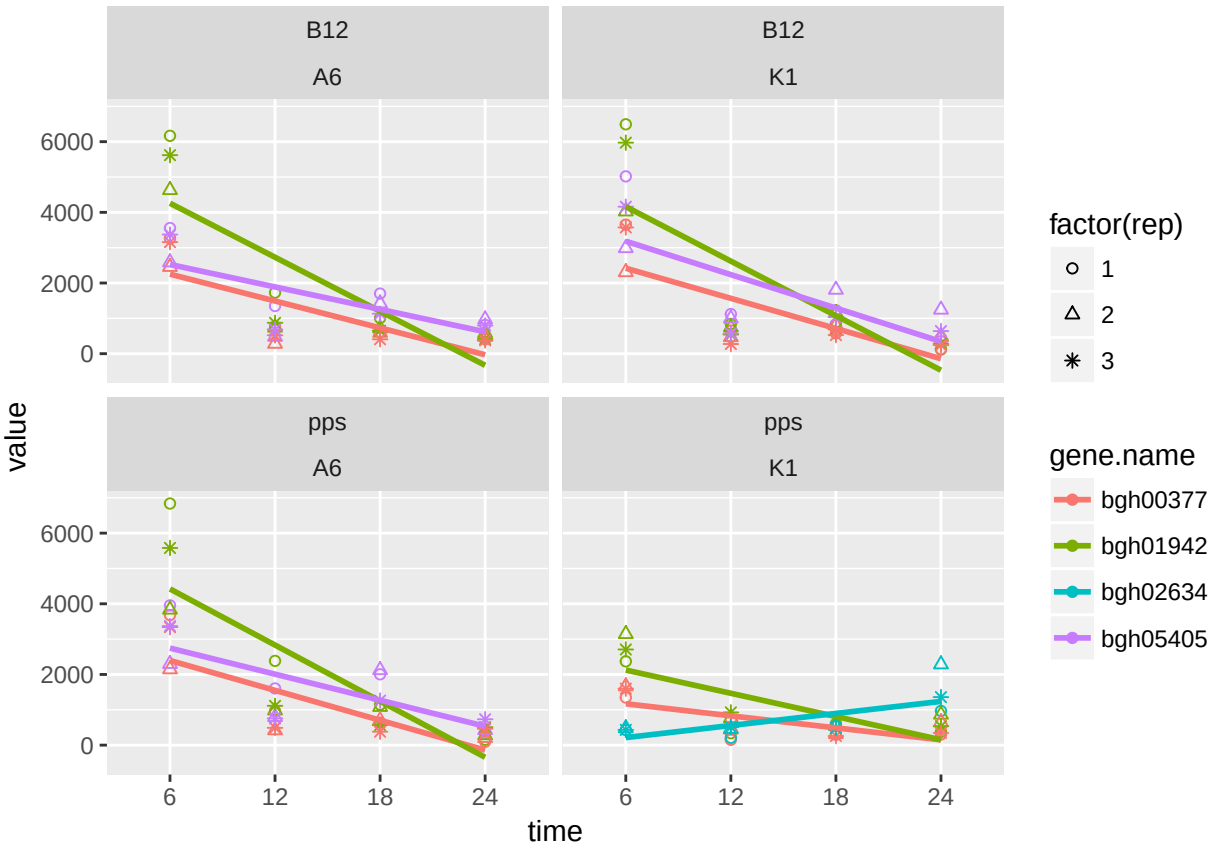

4. Assign a different color to each gene

```
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value, color = gene.name, group = gene.name
  geom_point()  + xlab("time") + facet_wrap(~plant~fungus) + geom_smooth(method = 'lm', se = F)
```



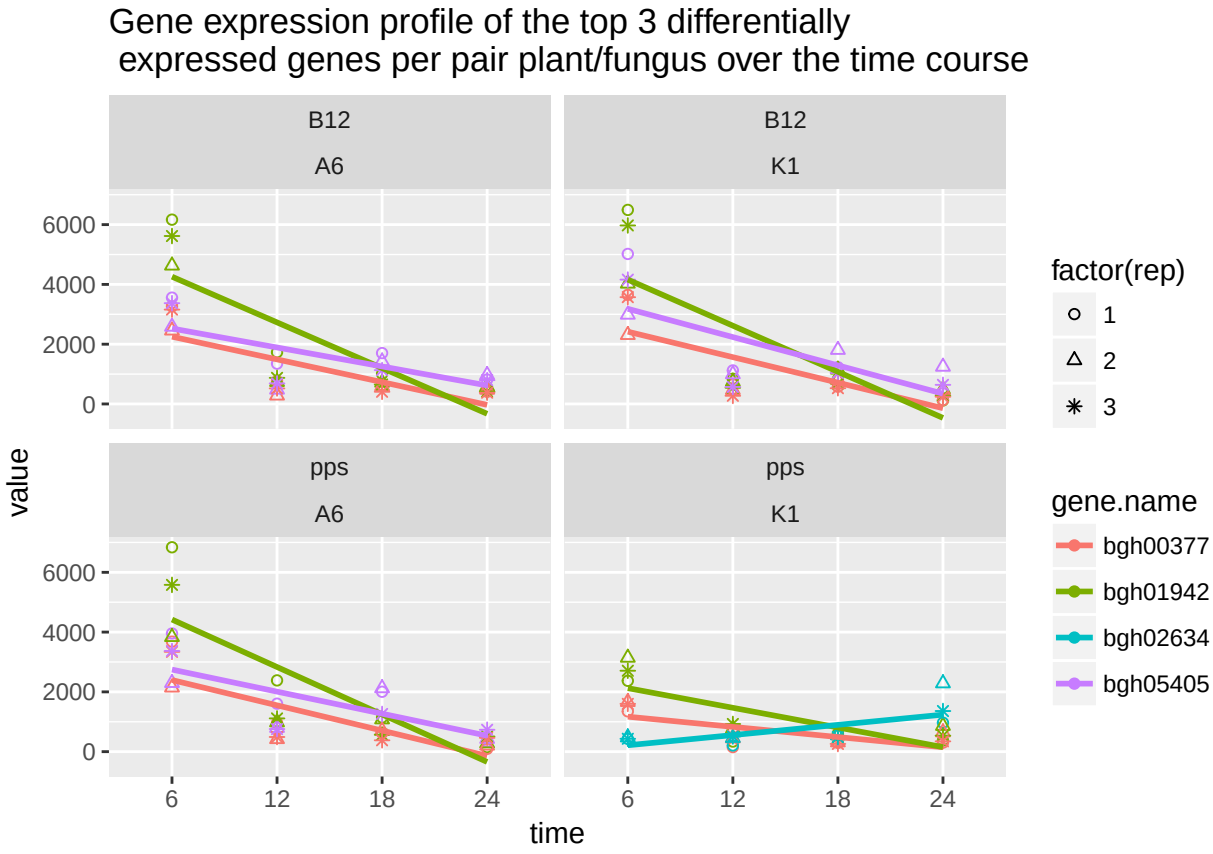

5. Assign a different shape to each replicate

```
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value, color = gene.name, group = gene.name
  geom_point()  + xlab("time") + facet_wrap(~plant~fungus) +  scale_shape_manual(values=c(21,24,8)) +
  geom_smooth(method = 'lm', se = F)
```




6. Add a descriptive title to the plot

```
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value, color = gene.name, group = gene.name
  geom_point()  + xlab("time") + facet_wrap(~plant~fungus) + scale_shape_manual(values=c(21,24,8)) +
  geom_smooth(method = 'lm', se = F) +
  ggtitle('Gene expression profile of the top 3 differentially \n expressed genes per pair plant/fungus
```

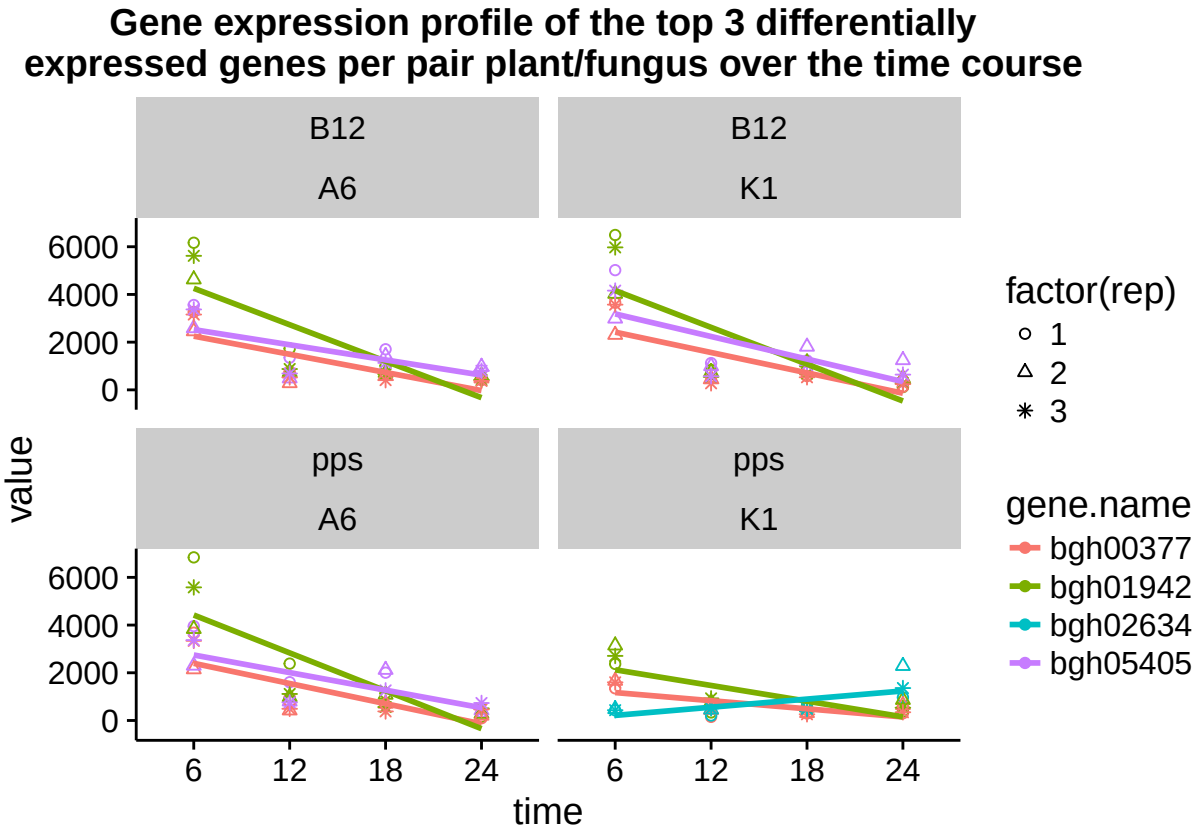



7. Use the "black & white" theme

```
# theme black and white (and theme cowplot)
library(cowplot)
```

```
#>
#> Attaching package: 'cowplot'
#> The following object is masked from 'package:ggplot2':
#>
#>     ggsave
```

```
all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
ggplot(all_data_for_graphs, aes(x = factor(time.point), y = value, color = gene.name, group = gene.name
  geom_point()  + xlab("time") + facet_wrap(~plant~fungus) + scale_shape_manual(values=c(21,24,8)) +
  geom_smooth(method = 'lm', se = F) +
  ggtitle('Gene expression profile of the top 3 differentially \n expressed genes per pair plant/fungus
  theme_bw() + theme_cowplot()
```




Extra credit: add error bars to the plot (use `geom_errorbar`).

```
#expr_data %>% group_by(plant, fungus, gene.name) %>% summarise(expr.diff.mean = mean(expr.diff), expr.
#expr_data_mean_for_sd %>% arrange(desc(expr.diff.mean)) %>%  arrange(expr.diff.var) %>% slice(1:3) %>%
#top3_genes_for_sd
#all_data_for_graphs %>% group_by(gene.name, plant, fungus, time.point) %>% summarise(mean.data = mean(
#all_data_for_graphs_with_sd
#ggplot(all_data_for_graphs_with_sd , aes(x = factor(time.point), y = standard.deviation)) + geom_point
#all_data_for_graphs$time.point <- as.numeric(all_data_for_graphs$time.point)
#ggplot(all_data_for_graphs_with_sd, aes(x = factor(time.point), y = value, color = gene.name, group = 
```