Data Visualization in R

Part of CEPLAS PhD Qualification Module 2020

Dominik Brilhaus

November 20, 2020

Contents

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

1 General Info

1.1 Goals

- Spike interest for code
- Understand and use ggplot2 grammar for nice graphs
- Compile RMarkdown (to .pdf or .html)
- Write a small interactive Shiny app

1.2 About this reader and workshop

1. The following notes are by no means a complete introduction to R nor ggplot2, RMarkdown or Shiny.
2. Also the exemplary analysis of metabobolomics, mRNASeq is by no means comprehensive, but just re-used here to have some nice plot examples and still reproduce the code part from-data-to-plot.
3. I am writing this on a Mac hoping that it will run just as smoothly run on a linux or PC.
4. R is very versatile, still somewhat intuitive and helpful. Compared to GUI statistic tools you write down all details of your analysis (-> computational analysis lab book).
5. There's never just one (correct) way. And programming languages are actively being developed.
6. The best way to learn is with your own data and problem.

1.3 Prerequisites

Please install the latest versions of R and RStudio Desktop.

1.4 Hands-on sessions

Exercise session 1
Exercise session 2
Exercise session 3
Exercise session 4
Exercise session 5
Exercise session 6

1.5 Safe points

Added to backup data or restore (from sciebo). In order to do copy them into your working directory and use the `load`.

Safe point 1
Safe point 2
Safe point 3
Safe point 4

---

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

2 Useful links

2.1 Basics

- CRAN: https://cran.r-project.org

  The repository for R software and libraries.

- Bioconductor: https://bioconductor.org

  Open source package / library repository (primarily R) for bioinformatics.

- Hands-on programming with R: https://rstudio-education.github.io/hopr/

  This is a good intro to R (Studio) for beginners.

- The R cookbook: http://www.cookbook-r.com

  Another (older but still valid) beginner's guide to R.

2.2 R for data science and visualization

- The R graphics cookbook https://r-graphics.org

  This used to be *the* introduction on gpplot. Probably a lot of the examples are now covered in the tidyverse guide(s) below.

- R for data science: https://r4ds.had.co.nz

  A closer look into how you can use R for data science, with a focus on the famous packages (e.g. ggplot2, dplyr, tidyr) of the Tidyverse (https://www.tidyverse.org).

- R (tidyverse) style guide: https://style.tidyverse.org

  Good coding (e.g. punctuation) practice to make your code more readable and intuitively comprehensible.

- Rstudio cheat sheets https://github.com/rstudio/cheatsheets or https://rstudio.com/resources/cheatsheets/

  Cheat sheets are well condensed pages of how to use certain packages. The most important one for data visualization using ggplot2: https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

2.3 Support

- Stackoverflow https://stackoverflow.com

  Basically if you google any question related to coding, R, python, data science, there's a good chance you'll end up at stackoverflow.

- That said, https://google.com is actually a very good starting point to answer your questions. The coding community is very motivated when it comes to solving puzzles and many people will have faced the issues you'll run into before.

2.4 RMarkdown

If you want to learn more on how to write nice Rmarkdown files that can be rendered to e.g. html or pdf output (like this workshop reader), check https://bookdown.org/yihui/rmarkdown/

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

2.5 Miscellaneous

- A good resource for color palettes: https://colorbrewer2.org/
- Some general notes on good scientific practice and data management:
  - https://github.com/CEPLAS-FAIRidise
  - https://datamanagement.hms.harvard.edu/hms-data-management-working-group/website-subgroup

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

3 Basics

3.1 Navigating / scripting in R

```
## Commenting

# Commented lines start with a hash (#)  (CMD+shift+C)
# Any jibber-jabber in here. This command won't be executed

## Keyboard shortcuts
# CMD+[enter] executes line / highlighted code
# CMD+shift+D duplicates line / highlighted code
# CMD+D deletes line / highlighted code
# Option+[ArrowKeys] moves line / highlighted code

# Use the [tab] key to display available functions, function arguments, files in your
# directory and to auto-fill basically anything.


## History
# use the arrow keys in the console to circle through commands that you executed

history() # will print the latest commands


## Other helpful basic commands

# head() / tail() print the first / last elements of an object (by default 6).

timestamp() # prints a nice timestamp
Sys.time()
```

3.2 The environment

The environment stores all your current variables, data.frames, etc.

```
ls() # will list the content of the current environment

# rm()
# rm(x, seq_vec3)
rm(list = ls()) # clean complete environment
```

3.3 Packages

You only need to `install.packages()` once. You can `update.packages()`. When you restart R, you just need to load the package with `library()`. Note, usually you don't need the *repos* argument, but you do in RMarkdown.

In order for you to run this script smoothly, I collect all required packages in a vector and then `for` loop through this vector to check whether they need to be installed prior to loading. Hope it works.

Note: `ggplot2` is now part of `tidyverse` so we do not need to specifically install and load it.

```
## Example
# install R packages from CRAN
# install.packages("tidyverse", repos = "https://cran.uni-muenster.de/")
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
# load / attach package
# library(tidyverse)

# update R packages from CRAN
# update.packages(ask = FALSE, checkBuilt = TRUE)


required_packages <-
  c('knitr', 'kableExtra', "rmarkdown", ## RMarkdown,
  "Rmisc", 'openxlsx', 'readxl', "plyr", "tidyverse", ## data import and shaping
  "RColorBrewer", "grid", "gridExtra")  ## plotting


for(package in required_packages)
{
  print(package)
  ## Check if package is installed. If not, install
  if(!package %in% row.names(installed.packages())){install.packages(package,
                          repos ="https://cran.uni-muenster.de/")}
  # ## Check if package is up to date. If not, update
  # update.packages(package, repos = "https://cran.uni-muenster.de/")
  ## Load package
  library(package, character.only = T)
}
```

```
## [1] "knitr"
## [1] "kableExtra"
## [1] "rmarkdown"
## [1] "Rmisc"
## [1] "openxlsx"
## [1] "readxl"
## [1] "plyr"
## [1] "tidyverse"
## [1] "RColorBrewer"
## [1] "grid"
## [1] "gridExtra"
```

3.4 Seeking help

```
help()
help(mean)
?mean

# browseVignettes(package = "ggplot2")

# If you seek help online and want to ask a question e.g. in a forum, it's a good idea to
# also provide information about your system and session (e.g. R version, OS, locale)

sessionInfo()
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

4 In-and-out

4.1 The working directory

```
# where am I?
getwd()
# I want to be there
setwd("~/Desktop/")
# getwd()

# dir()
# dir(recursive = T)
dir(pattern = 'xlsx')
dir(pattern = 'xlsx', recursive = T)
```

4.2 Import / export data

```
read.csv("example_data/exp_01_metabolomics/example_data.csv")
read.table("example_data/exp_01_metabolomics/example_data.csv", sep = ',')

assay01 <- read.table("example_data/exp_01_metabolomics/example_data.csv", sep = ',')

typeof(assay01)
class(assay01)
assay01$V1

assay01 <- read.table("example_data/exp01_sq/assay01.csv", sep = ',', header = T)

### Warning: depending on OS, this overwrites existing files
write.csv(file = "new.csv", x = assay01, row.names = F)
write.table(file = "new.txt", x = assay01, row.names = F, sep = '\t')

dir()
file.remove(c("new.csv", "new.txt"))

# Reading directly from Excel
## Note the . instead of _ !
## These two functions come from different packages, mostly do the same, but in an
## annpyingly different manner...

exp01A <- read_xlsx("example_data/exp_01_metabolomics/Praktikum_sample_list.xlsx")
exp01B <- read.xlsx("example_data/exp_01_metabolomics/Praktikum_sample_list.xlsx", sheet = 1)
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

4.3 Some example data

```
course_df <- data.frame()
cnames <- c("name", "height_cm", "age_a", "used_ggplot", "PC_Linux_Mac", "cat_dog")

course_df <- rbind.data.frame(stringsAsFactors = F,
  c("Karl", 180, 20L, "y", "pc", "dog"),
  c("Bjork", 120, 31, "y", "linux", "cat"),
  c("Fred", 195, 21L, "y", "linux", "cat"),
  c("Hella", 180, 25, "y", "mac", "dog"),
  c("Phine", 155, 26, "y", "mac", "cat"),
  c("Tim", 182, 26, "y", "mac", "cat"),
  c("Jose", 188, 21, "y", "pc", "cat"),
  c("Rajani", 172, 28, "y", "pc", "dog"),
  c("Joana", 172, 30, "y", "linux", "dog"),
  c("Judith", 166, 29, "y", "pc", "cat"),
  c("Lisa", 191, 22L, "n", "linux", "cat"),
  c("Max", 175, 24L, "n", "mac", "dog"),
  c("Miriam", 165, 28L, "y", NA, "cat")
)


colnames(course_df) <- cnames

course_df$PC_Linux_Mac <- toupper(course_df$PC_Linux_Mac)
course_df$cat_dog <- toupper(course_df$cat_dog)
course_df$height_cm <- as.numeric(course_df$height_cm)
course_df$age_a <- as.integer(course_df$age_a)
```

4.4 Printing plots to files

```
pdf(file = "plot01.pdf", width = 10, height = 10)
plot(course_df$age_a ~ course_df$height)
dev.off()

png(file = "plot01.png", res = 100, width = 100, height = 100)
plot(course_df$age_a ~ course_df$height)
dev.off()

tiff(file = "plot01.tiff", res = 100, width = 1000, height = 1000)
plot(course_df$age_a ~ course_df$height)
dev.off()

jpeg(file = "plot01.jpg", res = 100, width = 1000, height = 1000)
plot(course_df$age_a ~ course_df$height)
dev.off()
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

5 ggplot2 – Powerful data visualization

5.1 Types of geoms and stats

5.1.1 Dot plots

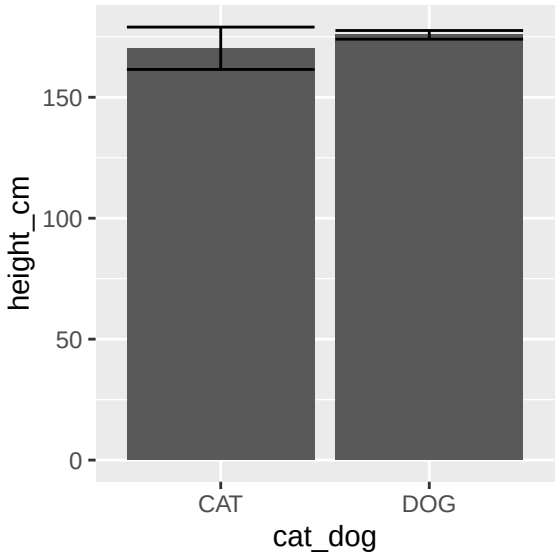
```
ggplot(data = course_df, aes(x = age_a, y = height_cm))

# Adding geom
ggplot(course_df, aes(x = age_a, y = height_cm)) +
  geom_point()

# Changing x/ y
ggplot(course_df, aes(x = cat_dog, y = age_a)) +
  geom_point()
```

5.1.2 Bar charts

```
ggplot(course_df, aes(x = cat_dog, y = age_a)) +
  geom_col()

## geom_bar() instead of geom_col()
ggplot(course_df, aes(x = cat_dog, y = age_a)) +
  geom_bar(stat = 'identity')

## aes_string() instead of aes()
ggplot(course_df, aes_string(x = "cat_dog", y = "age_a")) +
  geom_col()
```

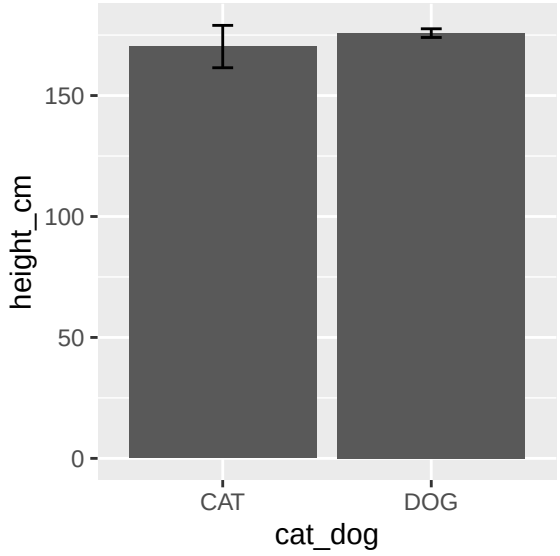
```
ggplot(course_df, aes(x = cat_dog, y = height_cm)) +
  stat_summary(fun = 'mean', geom = 'bar') +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar')
```

5.1.2.1 Adding error bars

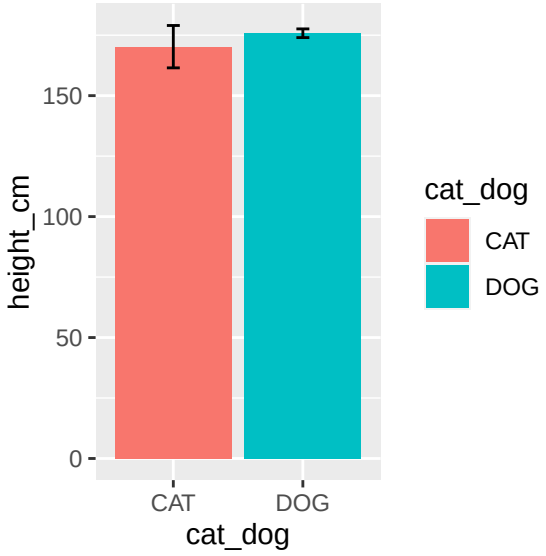


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
ggplot(course_df, aes(x = cat_dog, y = height_cm)) +
  stat_summary(fun = 'mean', geom = 'bar') +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1)
```



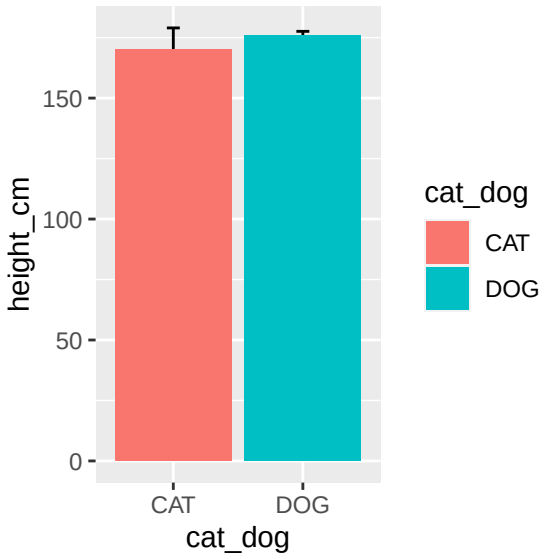

```
ggplot(course_df, aes(x = cat_dog, y = height_cm, fill = cat_dog)) +
  stat_summary(fun = 'mean', geom = 'bar') +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1)
```




```
ggplot(course_df, aes(x = cat_dog, y = height_cm, fill = cat_dog)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1) +
  stat_summary(fun = 'mean', geom = 'bar')
```

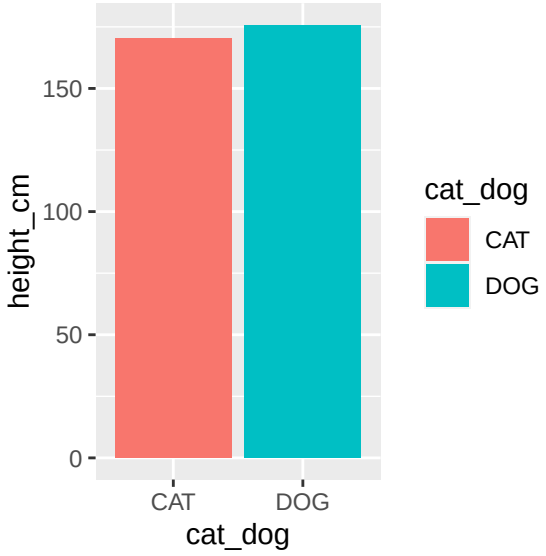
5.1.2.2 Order (of layers) matters

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



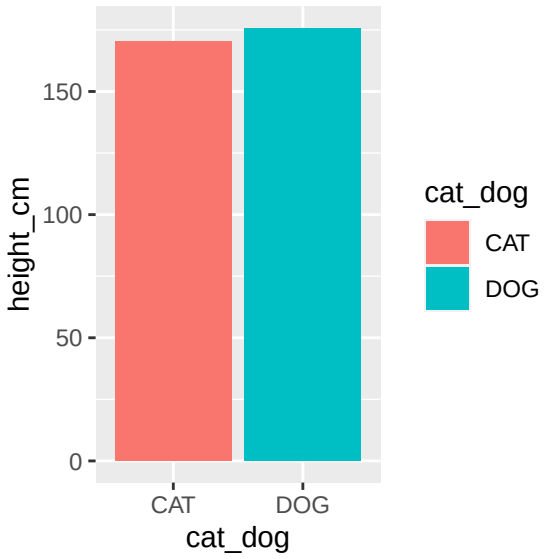
5.1.2.3 Another way to Rome Instead of letting ggplot calculate the means, sd, etc., do it yourself and plot the results.

```
height_summary <- summarySE(course_df, groupvars = 'cat_dog', measurevar = 'height_cm')

ggplot(height_summary, aes(x = cat_dog, y = height_cm, fill = cat_dog)) +
  geom_col()
```

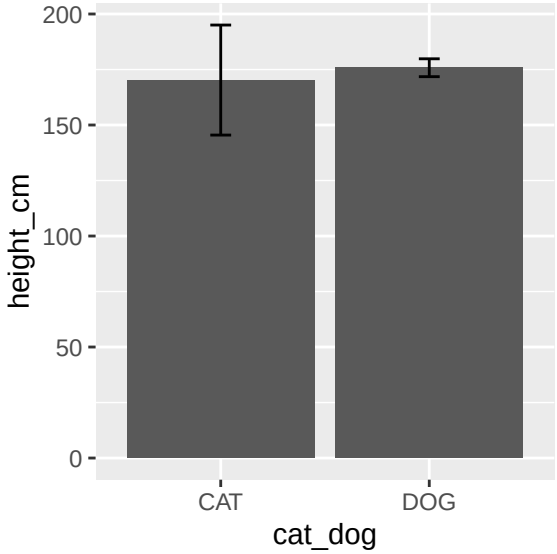



```
ggplot(height_summary) +
  geom_col(aes(x = cat_dog, y = height_cm, fill = cat_dog))
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197




```
ggplot(height_summary, aes(x = cat_dog, y = height_cm)) +
  geom_col() +
  geom_errorbar(aes(ymin = height_cm - sd, ymax = height_cm + sd), width = 0.1)
```




```
course_df$cat_dog <- factor(course_df$cat_dog, levels = c('DOG', 'CAT'))

plot1 <-  ggplot(course_df, aes(x = cat_dog, y = height_cm, fill = cat_dog)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1) +
  stat_summary(fun = 'mean', geom = 'bar')
```

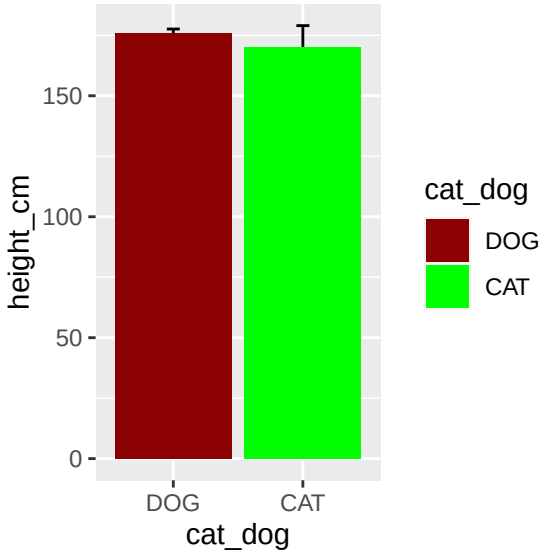
5.1.2.4 Factor levels matter

5.1.3 Adapting scales

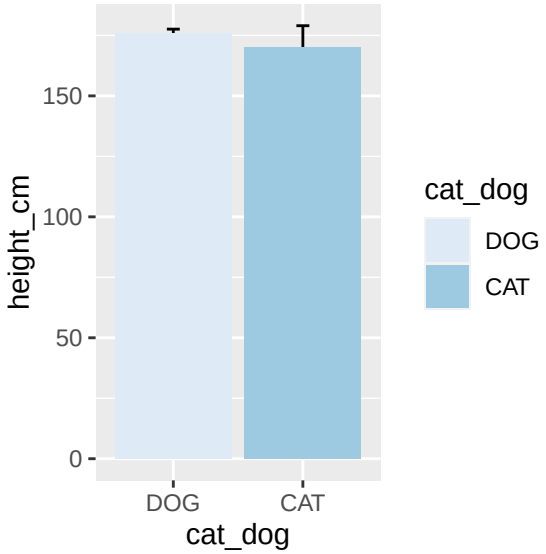
```
## to change color

plot1 +
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
  scale_fill_manual(values = c("DOG" = 'darkred', "CAT" = "green"))
```

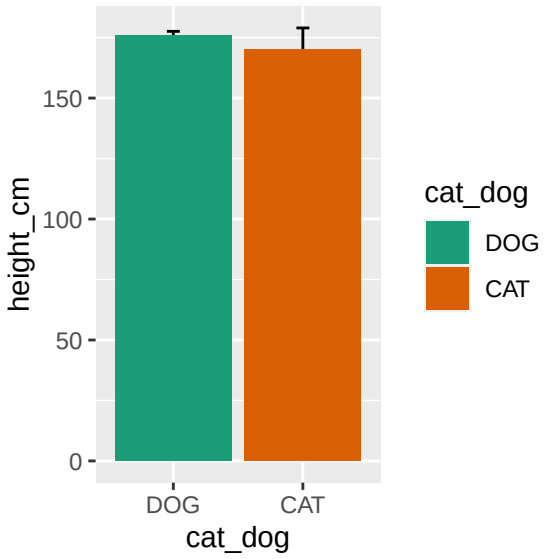



```
  # scale_fill_manual(values = c("DOG" = "#8dd3c7", "CAT" = "green"))

# See colors() for a selection of available colors

plot1 +
  scale_fill_brewer()
```

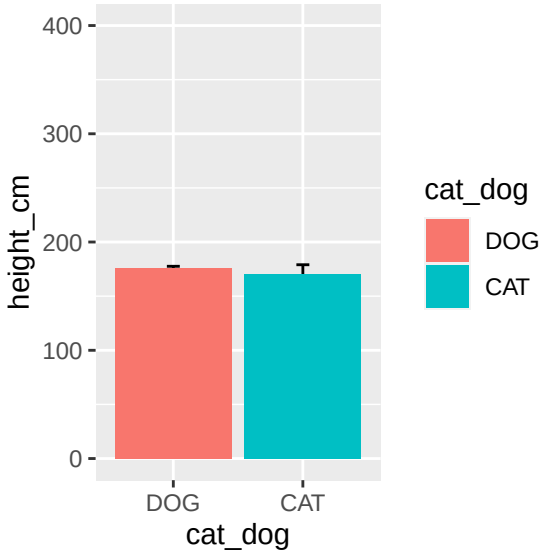



```
# See display.brewer.all() for a selection of available brewer palettes

plot1 +
  # scale_fill_brewer(palette = 'Set2')
  scale_fill_brewer(palette = 'Dark2')
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197




```
# scale_fill_brewer(palette = 'Paired')


## To adjust x or y

plot1 +
  scale_y_continuous(limits = c(0, 400))
```

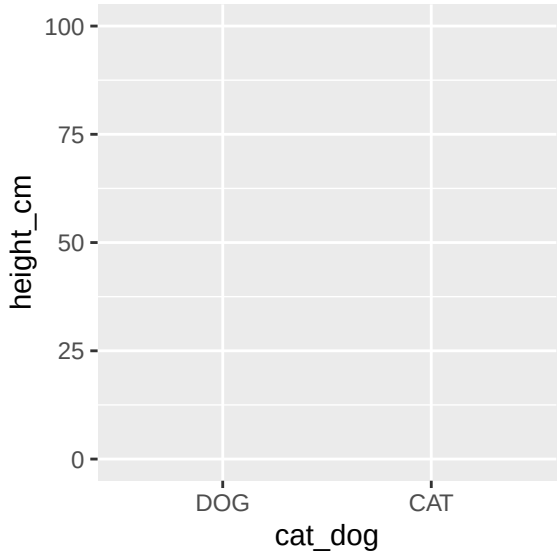



```
plot1 +
  scale_y_continuous(limits = c(0, 100))
```

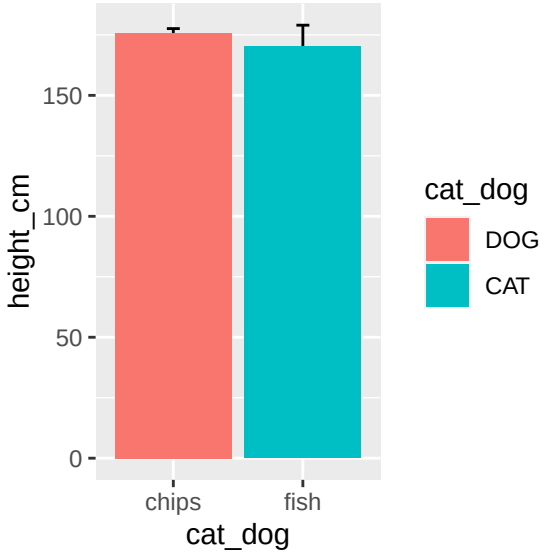
```
## Warning: Removed 13 rows containing non-finite values (stat_summary).

## Warning in max(f): no non-missing arguments to max; returning -Inf

## Warning: Computation failed in `stat_summary()`:
## argument must be coercible to non-negative integer

## Warning: Removed 13 rows containing non-finite values (stat_summary).

## Warning in max(f): no non-missing arguments to max; returning -Inf
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
## Warning: Computation failed in `stat_summary()`:
## argument must be coercible to non-negative integer
```



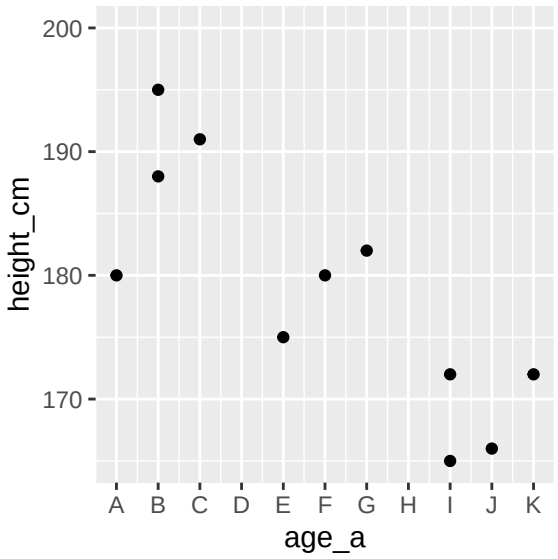

```
plot1 +
  scale_x_discrete(breaks = c("CAT", "DOG"), labels = c("fish", "chips"))
```




```
ggplot(course_df, aes(x = age_a, y = height_cm)) +
  geom_point() +
  scale_y_continuous(limits = c(165, 200)) +
  # scale_x_continuous(limits = c(20, 30))
  # scale_x_continuous(breaks = seq(20, 30, 1))
  scale_x_continuous(limits = c(20, 30), breaks = seq(20, 30, 1),
                     labels = LETTERS[1:11])
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```

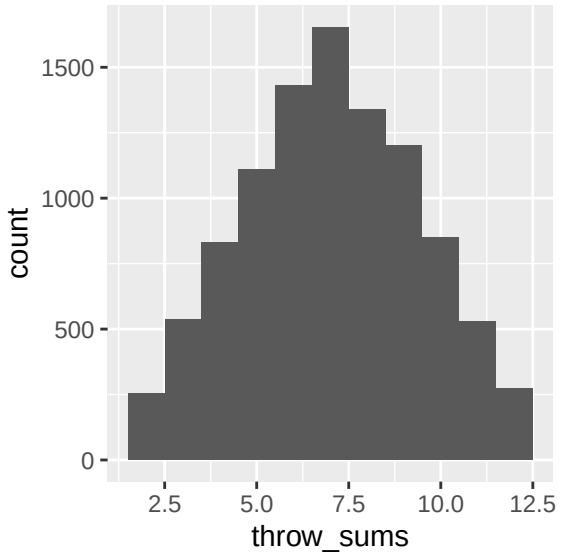

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



5.1.4 Histogram

Let's take a small detour on for loops

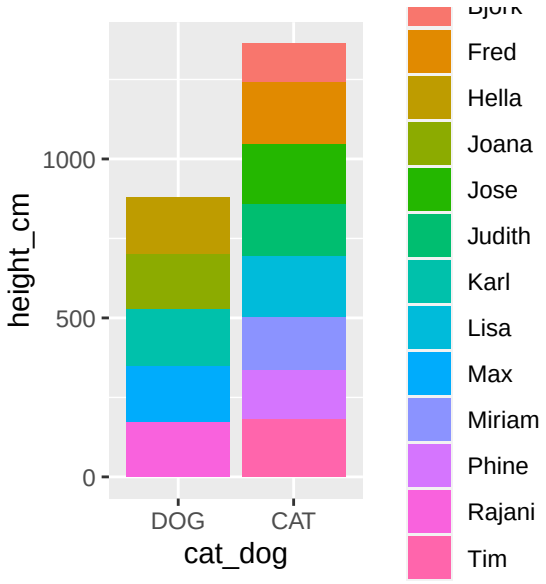
```
die <- 1:6

# sample(die, size = 2, replace = T)

throw_collect <- c()
for(i in 1:10000)
{
  throw_collect <- c(throw_collect, sum(sample(die, size = 2, replace = T)))
}

throw_df <- data.frame(throw_sums = throw_collect)

ggplot(throw_df, aes(x = throw_sums)) +
  geom_histogram(binwidth = 1)
```



dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

5.1.5 Adding color

```
ggplot(course_df, aes(x = cat_dog, y = height_cm, fill = name)) +
  geom_col()
```



Note, this shows us what ggplot (i.e. `geom_col`) actually does with the data (here: stacking and summing).

```
course_df[course_df$cat_dog == 'DOG', 'height_cm']
```

```
## [1] 180 180 172 172 175
```

```
course_df[course_df$cat_dog == 'CAT', 'height_cm']
```

```
## [1] 120 195 155 182 188 166 191 165
```

```
mean(course_df[course_df$cat_dog == 'DOG', 'height_cm'])
```

```
## [1] 175.8
```

```
mean(course_df[course_df$cat_dog == 'CAT', 'height_cm'])
```

```
## [1] 170.25
```

```
summary(course_df[course_df$cat_dog == 'DOG', 'height_cm'])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   172.0   172.0   175.0   175.8   180.0   180.0
```

```
summary(course_df[course_df$cat_dog == 'CAT', 'height_cm'])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   120.0   162.5   174.0   170.2   188.8   195.0
```

```
summarySE(course_df, groupvars = 'cat_dog', measurevar = 'height_cm')
```

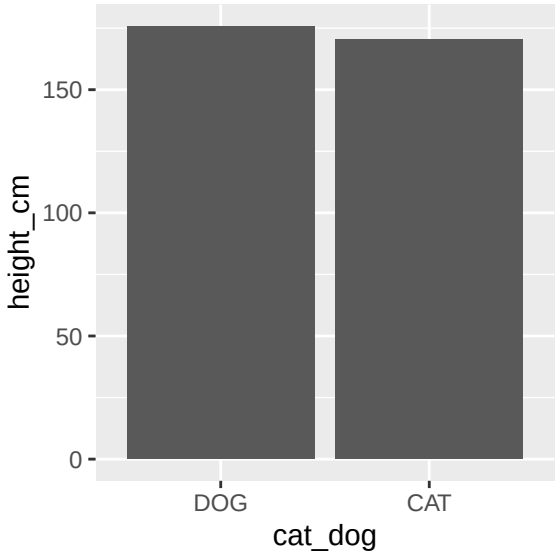
```
##   cat_dog N height_cm        sd       se        ci
## 1     DOG 5    175.80  4.024922 1.800000  4.997601
## 2     CAT 8    170.25 24.783347 8.762236 20.719397
```

```
summarySE(course_df, groupvars = 'cat_dog', measurevar = 'height_cm')
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
##   cat_dog N height_cm        sd       se        ci
## 1     DOG 5    175.80  4.024922 1.800000  4.997601
## 2     CAT 8    170.25 24.783347 8.762236 20.719397
```

```
ggplot(course_df, aes(x = cat_dog, y = height_cm)) +
  stat_summary(fun = 'mean', geom = 'bar')
```



5.1.6 Exercise session 1

- Plot "age" against "R usage"
- Color by "R usage"
- Reorder to show "yes" first
- Display "yes" in green and "no" in red

---

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

6 Real life examples

6.1 Example 1: Metabolomics

The data:

1. *GC-MS_QuantReport_ISTD_DB.xlsx* contains the **"raw" data**. This comes from a metabolomics course at the **Plant Metabolism and Metabolomics Lanboratories** at HHU in December 2019 tought by Philipp Westhoff. Marion Eisenhut designed the experiment and provided the samples.
2. *Praktikum_sample_list.xlsx* is basically the experimental **metadata** describing the experiment.

6.1.1 Prep the data

```
# exp1_data <- read.xlsx('example_data/exp_01_metabolomics/GC-MS_QuantReport_ISTD_DB.xlsx',
#                        sheet = 'Summary', startRow = 3)

# Alternatively:
exp1_data <-
  read_xlsx('example_data/exp_01_metabolomics/GC-MS_QuantReport_ISTD_DB.xlsx',
            sheet = 'Summary', skip = 2)

colnames(exp1_data) <- gsub(' ', '.', colnames(exp1_data))
exp1_data <- as.data.frame(exp1_data)


### Subset to remove annoying header lines
exp1_data_tidy <- exp1_data[exp1_data$Data.File != "Data File", ]
exp1_data_tidy2 <- subset(exp1_data, Data.File != "Data File")


exp1_metadata <-
  read_xlsx('example_data/exp_01_metabolomics/Praktikum_sample_list.xlsx')

colnames(exp1_metadata) <- gsub(' ', '.', colnames(exp1_metadata))
exp1_metadata <- as.data.frame(exp1_metadata)

### merge two tables

# head(exp1_data_tidy)
# tail(exp1_data_tidy)

# head(exp1_metadata)

exp1_plotdata <- merge(exp1_data_tidy, exp1_metadata,
                       by.x = 'Sample.Name',
                       by.y = "Sample.label.on.tube")

## unfortunately, this is character
head(exp1_plotdata$Resp.Ratio)

head(as.numeric(exp1_plotdata$Resp.Ratio))

# exp1_plotdata$Sample.weight.[mg] # won't work
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
head(exp1_plotdata$`Sample.weight.[mg]`)

## Add a new column with the normalized values
exp1_plotdata$relative_resp <-
  as.numeric(exp1_plotdata$Resp.Ratio) / exp1_plotdata$`Sample.weight.[mg]`

## Another way:
exp1_plotdata[ , 'relative_resp'] <-
  as.numeric(exp1_plotdata$Resp.Ratio) / exp1_plotdata$`Sample.weight.[mg]`
```

6.1.2 Safe point

```
save(exp1_plotdata, file = "exp1_plotdata.RData")
```

```
load(file = "exp1_plotdata.RData")
```

6.1.3 Let's plot some metabolites!

```
# unique(exp1_plotdata$Compound)
sort(unique(exp1_plotdata$Compound))
```

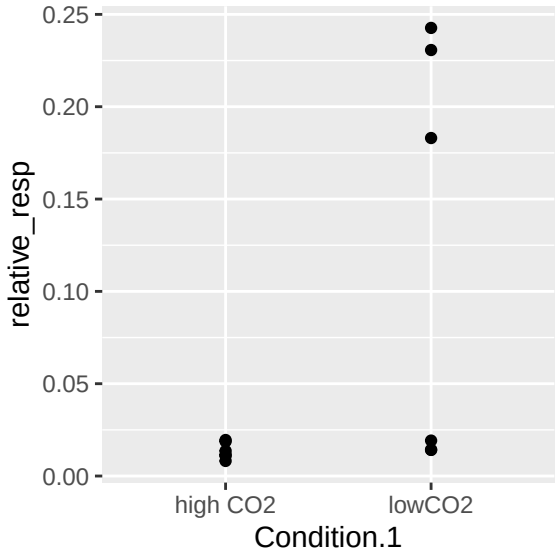
```
##  [1] "alpha-Alanine"       "alpha-Ketoglutarate" "Asparagine"
##  [4] "Aspartate"           "beta-Alanine"        "Citrate+Isocitrate"
##  [7] "Fumarate"            "Glucose"             "Glutamate"
## [10] "Glycerate"           "Glycerol"            "Glycine"
## [13] "Isoleucine"          "Leucine"             "Malate"
## [16] "Phenylalanine"       "Proline"             "Pyruvate"
## [19] "Serine"              "Succinate"           "Sucrose"
## [22] "Threonine"           "Tryptophan"          "Tyrosine"
## [25] "Valine"
```

```
# plyr::count(exp1_plotdata$Compound)

serine_subset <- subset(exp1_plotdata, Compound == "Serine")
```

6.1.4 Combining multiple variables

```
## points
ggplot(serine_subset, aes(x = Condition.1, y = relative_resp)) +
  geom_point()
```

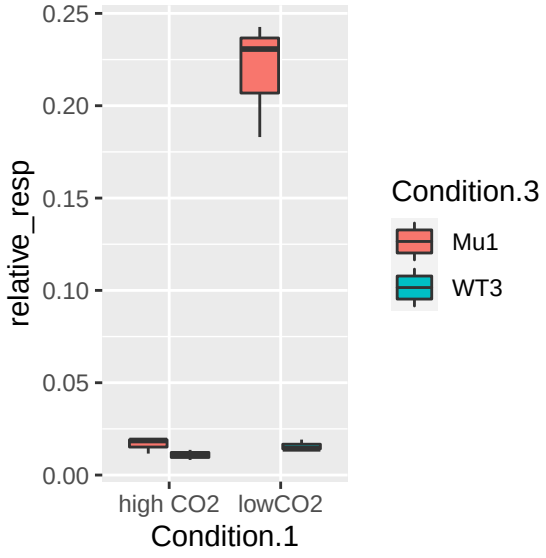

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197




```
summarySE(serine_subset, groupvars = c('Condition.1', 'Condition.3'),
          measurevar = 'relative_resp')
```

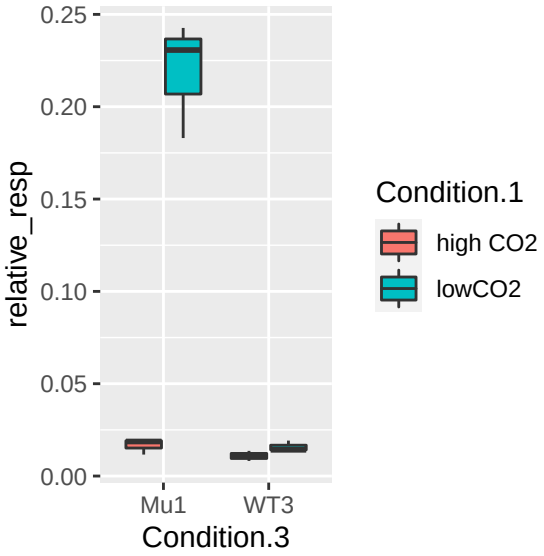
```
##   Condition.1 Condition.3 N relative_resp          sd          se          ci
## 1    high CO2         Mu1 3    0.01661487 0.004333649 0.002502033 0.010765380
## 2    high CO2         WT3 3    0.01089394 0.002729803 0.001576052 0.006781206
## 3      lowCO2         Mu1 3    0.21880569 0.031543119 0.018211428 0.078357452
## 4      lowCO2         WT3 3    0.01585002 0.002890422 0.001668786 0.007180207
```

```
## boxplot
ggplot(serine_subset, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
  geom_boxplot()
```

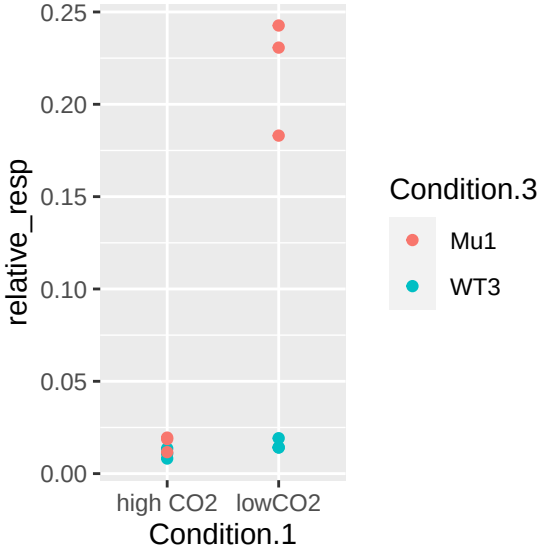



```
## boxplot opposite
ggplot(serine_subset, aes(x = Condition.3, y = relative_resp, fill = Condition.1)) +
  geom_boxplot()
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

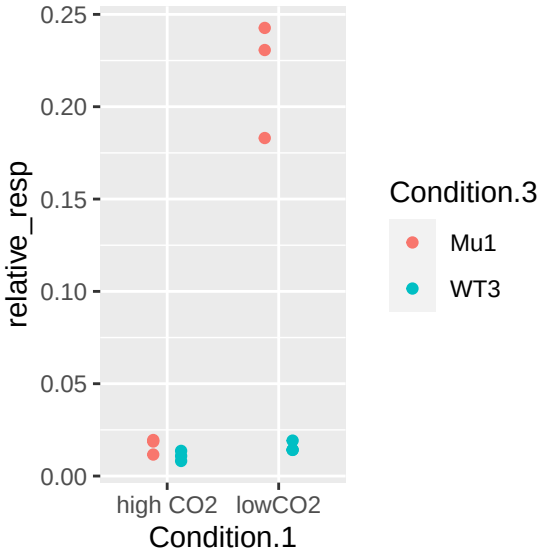



```
## points with color
ggplot(serine_subset, aes(x = Condition.1, y = relative_resp, col = Condition.3)) +
  geom_point()
```

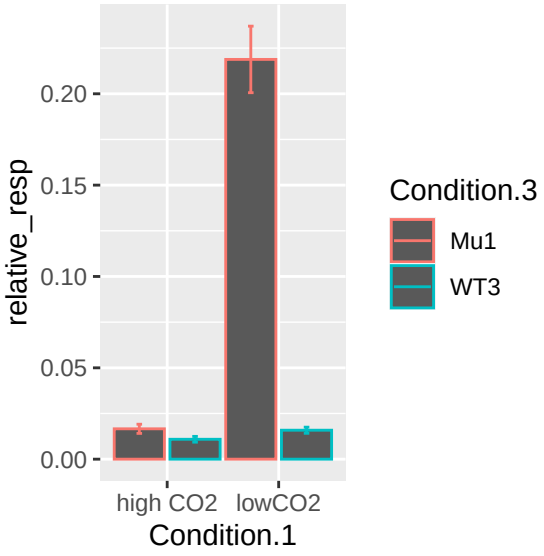



```
## points with positioning
ggplot(serine_subset, aes(x = Condition.1, y = relative_resp, col = Condition.3)) +
  # geom_point(position = position_dodge(width = 1))
  geom_point(position = position_dodge(width = 0.5))
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



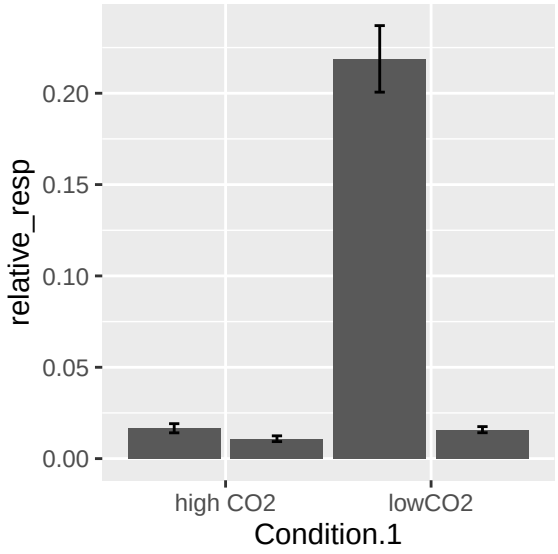

```
## bars with positioning
ggplot(serine_subset, aes(x = Condition.1, y = relative_resp, col = Condition.3)) +
  # geom_col(position = position_dodge(width = 1))
  stat_summary(fun = 'mean', geom = 'bar', position = position_dodge(width = 1)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
               position = position_dodge(width = 1))
```




```
## use fill for bars / boxplots, and col for points / lines
plot1 <- ggplot(serine_subset, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
  stat_summary(fun = 'mean', geom = 'bar', position = position_dodge(width = 1)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
               position = position_dodge(width = 1))


## group option

ggplot(serine_subset, aes(x = Condition.1, y = relative_resp, group = Condition.3)) +
  stat_summary(fun = 'mean', geom = 'bar', position = position_dodge(width = 1)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
              position = position_dodge(width = 1))
```



6.1.5 Exercise session 2

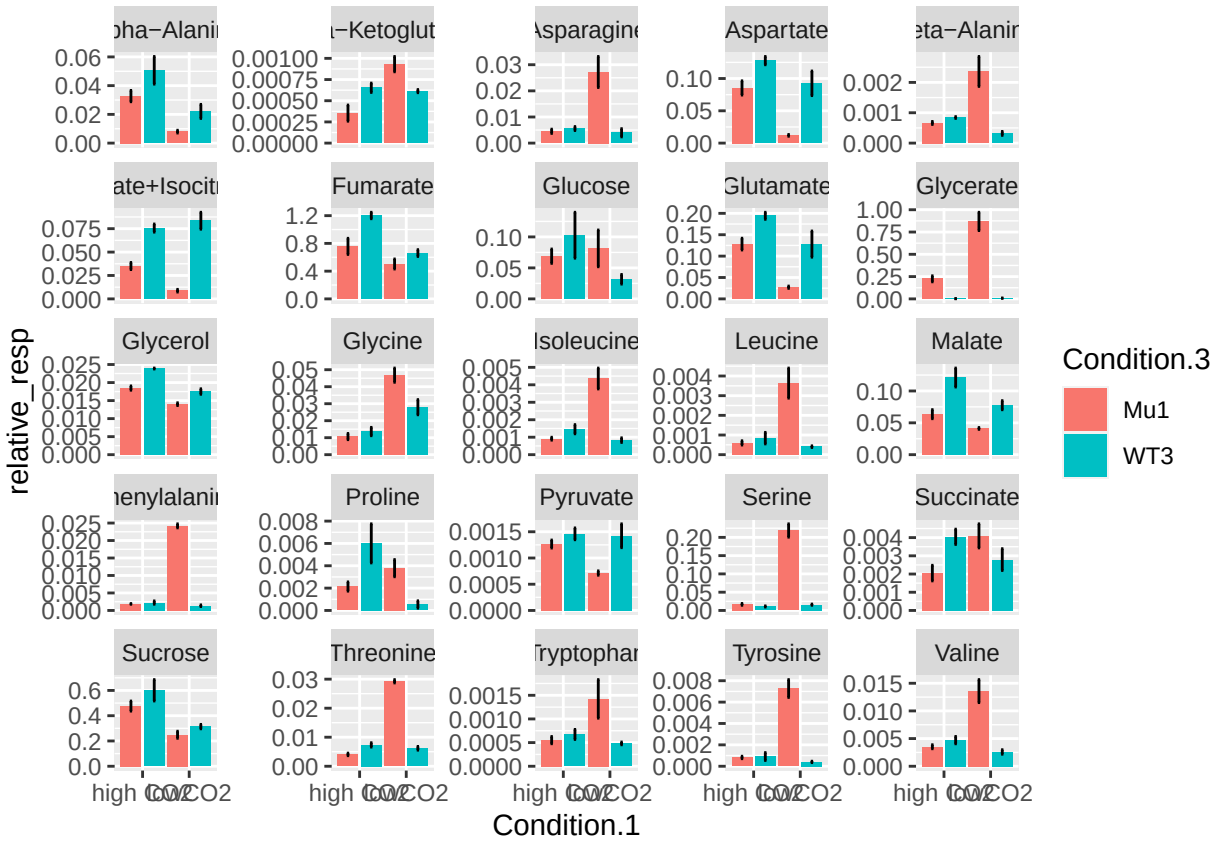
From the metabolomics data: - Select your favorite compound and subset the data to only that compound (check `sort(unique(exp1_plotdata$Compound))`)

- Plot the amount (`Resp.Ratio`) against the genotype (`Condition.3`) as a **dotplot** and color by CO2 treatment(`Condition.1`)
- Plot the response of the internal standard (`ISTD.Resp`) against the measurement number (`Data.File`). Note: `ISTD.Resp` is stored as characte - you need to convert with (`as.numeric`)

6.1.6 Combining multiple plots (facet)

```
ggplot(exp1_plotdata, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
  stat_summary(fun = 'mean', geom = 'bar', position = position_dodge(width = 1)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
               position = position_dodge(width = 1)) +
  # facet_grid(~Compound, scales = 'free_y')
  # facet_grid(Compound~., scales = 'free_y')
    facet_wrap(~Compound, scales = 'free_y')
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



6.1.7 Shaping your plot (the theme function)

```
ggplot(serine_subset, aes(x = Condition.3, y = relative_resp, shape = Condition.1,
                          col = Condition.3)) +
    geom_point(size = 10, position = position_dodge(width = 0.8)) +
    scale_color_brewer(type = "qual") +
    theme(panel.background = element_rect(color = "red", size = 5, linetype = "dashed")) +
    theme(axis.ticks.x = element_line(color = "darkblue", size = 2))+
    theme(aspect.ratio = 1) +
    # xlab("Whatever") +
    # ylab("Whatever there") +
    # ggtitle(" Serine ") +
    theme(plot.title = element_text(vjust = 3, face = "bold", size = 20)) +
    labs(x = "somtehing", y = "respo", title = "Serine", caption = "Fig. 1 Serine data") +
    # theme(legend.position = "bottom", legend.direction = "vertical")
    theme(legend.position = c(0.9, 0.9), legend.direction = "vertical",
          legend.background = element_rect(fill = "green")) +
    # theme(axis.title.x = element_text(size = 20),
    # axis.title.y = element_text(size = 30, face = "italic"))
    theme(axis.title = element_text(size = 20)) +
    theme(axis.text = element_text(size = 10, face = "bold"))



exp1_plot <-
  ggplot(exp1_plotdata, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
  stat_summary(fun = 'mean', geom = 'bar', position = position_dodge(width = 1)) +
  stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
               position = position_dodge(width = 1)) +
  # facet_wrap(~Compound)
  facet_wrap(~Compound, scales = 'free_y') +
  scale_fill_brewer(palette = 'Dark2') +
  theme_bw() +
  theme(strip.background = element_rect(fill = NA)) +
  # theme(strip.background = element_rect(fill = 'orange')) +
  # theme(plot.background = element_rect(fill = 'lightblue')) +
  # theme(panel.background = element_rect(fill = 'yellow')) +
  # theme(axis.title = element_text(colour = 'darkblue')) +
  theme(axis.title = element_text(colour = 'darkblue', face = 'bold.italic')) +
  theme(axis.text = element_text(colour = 'pink')) +
  theme(axis.ticks = element_line(colour = 'darkred')) +
  theme(legend.position = 'bottom', legend.title = element_blank()) +
  # xlab("CO2 treatment") +
  # ylab('Relative metabolite level') +
  # ggtitle('My super cool metabolite plots') +
  labs(x = "CO2 treatment", y = 'Relative metabolite level',
       title = 'My super cool metabolite plots',
       subtitle = '...well ok cool',
       caption = "Really? This is what I wasted my time on today?")


png(file = "exp1_metabolomics_plot.png", res = 300, width = 3000, height = 3000)
exp1_plot
dev.off()
```

6.1.8 Exercise session 3

Based on the course data (`course_df`): Go nuts and try to map as many variables as possible into one plot (single or facetted) by choice of scales, geom, color, fill, shapes, groups, facet…

---

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

6.2 Example 2: RNASeq

PP2015-01076R2_Supplemental_Dataset_S1.xlsx is the Supplemental Dataset 1 from Brilhaus, Dominik et al. (2016) https://doi.org/10.1104/pp.15.01076.

6.2.1 Preparing a theme

I like to store some favorite parameters, that I repeatedly use for different plots in the same context in my own `theme_dominik`.

```
theme_dominik <-
  theme(panel.grid = element_blank(), panel.background = element_blank(),
        panel.border = element_rect(fill = NA)) +
  theme(axis.text = element_text(color = "black"),
        axis.ticks = element_line(color = "black"),
        axis.title = element_text(color = "black")) +
  theme(plot.title = element_text(face= "bold")) +
  theme(aspect.ratio = 1) +
  theme(strip.background = element_blank(),
        strip.text = element_text(size = 7, face = "bold")) +
  theme(legend.position = "bottom",  legend.title = element_blank())
```

6.2.2 Preparing the data

This is the original supplemental data as downloaded from plant phys. It's mostly well described for human readability. However, it's not really directly readable by R or ggplot. So we need to shape it up a bit.

```
exp2_original <-
  read.xlsx('example_data/exp_02_rnaseq/PP2015-01076R2_Supplemental_Dataset_S1.xlsx',
            startRow = 21)

# exp2_url_data <-
# "http://www.plantphysiol.org/highwire/filestream/11813/field_highwire_adjunct_files/
# 1/PP2015-01076R2_Supplemental_Dataset_S1.xlsx"
# download.file(exp2_url_data, destfile = "example_data/exp_02_rnaseq/exp2_fromURL")
# exp2_original <- read.xlsx(exp2_url_data, startRow = 21)


## Extract and prep expression data
exp2_rpm <- exp2_original[, c(1, grep('rpm', ignore.case = T, colnames(exp2_original)))]
colnames(exp2_rpm) <- c('AGI', paste(rep(c(0,4,9,12,'re2'), each = 6),
                                     rep(c('MD', 'MN'), each = 3),
                                     c('rep1', 'rep2', 'rep3'), sep = '_'))


## Filter non-expressed genes
exp2_rpm <- exp2_rpm[!rowSums(exp2_rpm[, 2:ncol(exp2_rpm)]) == 0, ]

exp2_rpm_long <- exp2_rpm %>%
  pivot_longer(-AGI, values_to = "rpm")

exp2_rpm_long <- separate(data = exp2_rpm_long, col = name, sep = '_',
                          into = c("trtmt", "time", "replicate"))
exp2_rpm_long$trtmt <- factor(exp2_rpm_long$trtmt, unique(exp2_rpm_long$trtmt))

## Extract and prep statistics
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
exp2_stats <- exp2_original[, c(1, grep('q-value',
                                        ignore.case = T, colnames(exp2_original)))]
colnames(exp2_stats) <- c('AGI', paste(rep(c(4,9,12,'re2'), each = 2),
                                       rep(c('MD', 'MN')), sep = '_'))

exp2_stats_long <- exp2_stats %>%
  pivot_longer(-AGI, values_to = 'qvalue')

exp2_stats_long$asterisks <- cut(as.numeric(exp2_stats_long$qvalue),
                                 breaks = c(0,0.001,0.01, 0.05, 1),
                                 labels = c('***', '**', '*', ''))

exp2_stats_long[is.na(exp2_stats_long$asterisks), 'asterisks'] <- ''

exp2_stats_long <- separate(data = exp2_stats_long, col = name, sep = '_',
                            into = c("trtmt", "time"))

exp2_stats_long$trtmt <- factor(exp2_stats_long$trtmt, unique(exp2_stats_long$trtmt))


## Extract and prep log2-FC (fold changes)
exp2_lfc <- exp2_original[, c(1, grep('log2',
                                      ignore.case = T, colnames(exp2_original)))]

colnames(exp2_lfc) <- c('AGI', paste(rep(c(4,9,12,'re2'), each = 2),
                                       rep(c('MD', 'MN')), sep = '_'))

exp2_lfc_long <- exp2_lfc %>%
  pivot_longer(-AGI, values_to = 'log2fc')

exp2_lfc_long$log2fc <- as.numeric(exp2_lfc_long$log2fc)

exp2_lfc_long <- separate(data = exp2_lfc_long, col = name, sep = '_',
                          into = c("trtmt", "time"))

exp2_lfc_long$trtmt <- factor(exp2_lfc_long$trtmt, unique(exp2_lfc_long$trtmt))


## Extract and prep functional annotation

exp2_annotation <- exp2_original[, 1:3]
colnames(exp2_annotation) <- c('AGI', 'gene.name', 'category')

exp2_annotation <- separate(data = exp2_annotation, col = category, remove  = F, sep = "[\\.]",
                    into = paste0(rep("BIN", 8), 1:8), extra = "merge", fill = "right")
```

6.2.3 Safe point

```
save(exp2_rpm, exp2_rpm_long, exp2_stats, exp2_stats_long,
     exp2_lfc_long, exp2_annotation, file = "exp2.RData")
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

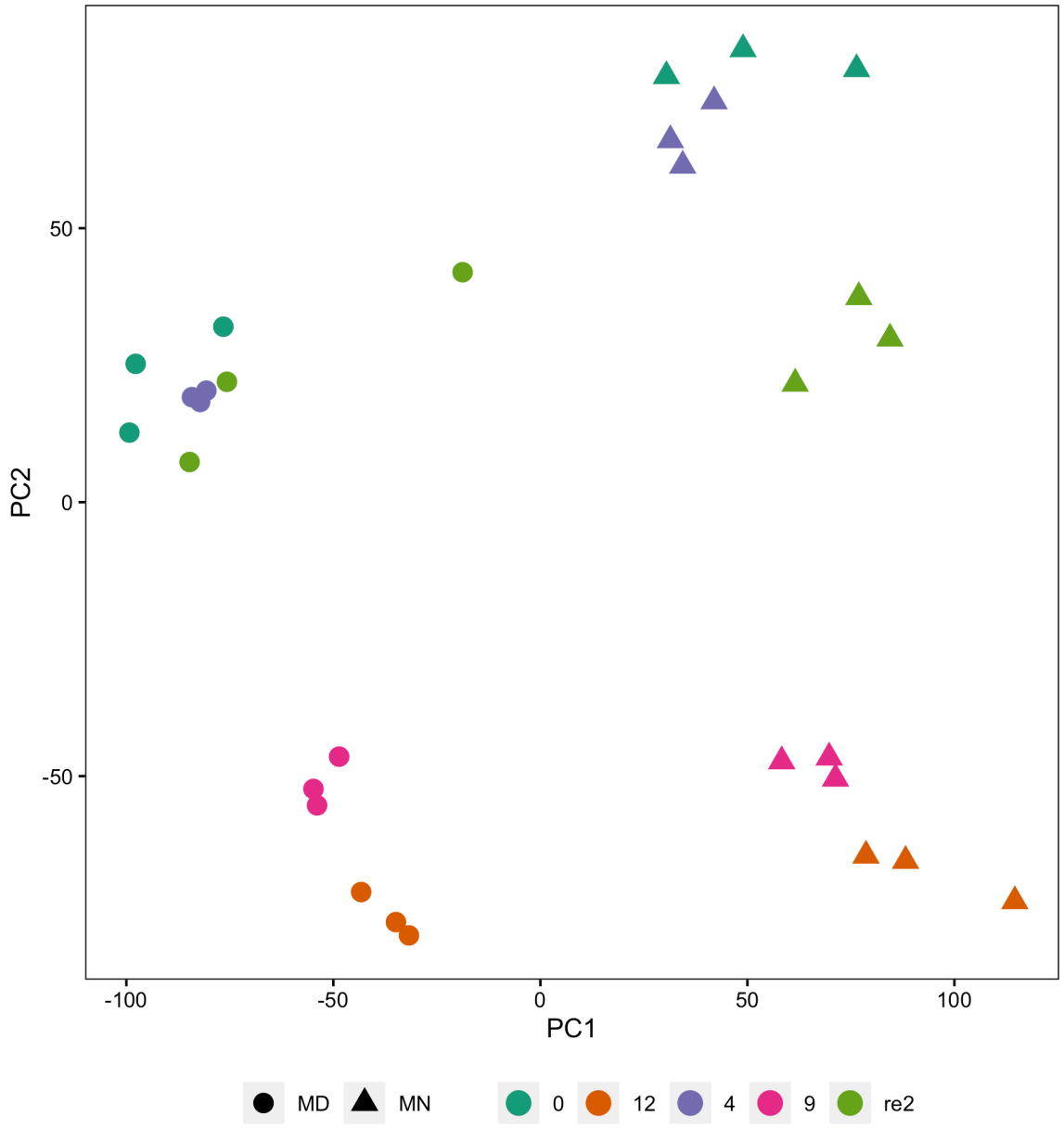
```
load(file = "exp2.RData")
```

6.2.4 Calculate and draw a PCA to get an overview of the dataset

```
pca_data <- as.data.frame(pivot_wider(exp2_rpm_long,
                                      names_from = AGI,
                                      values_from =  rpm,
                                      id_cols = c(trtmt, time, replicate)))

pca_data <- unite(pca_data,  trtmt, time, replicate, col = 'merger', sep = '_')
rownames(pca_data) <- pca_data$merger
pca_data <- pca_data[, -1]

pca_data <- pca_data[, apply(pca_data, 2, function(x) {sum(x) != 0})]

pca <- prcomp(pca_data, scale = T)
pcaPlotData <- as.data.frame(pca$x)
pcaPlotData$merger <- rownames(pcaPlotData)
pcaPlotData <- separate(data = pcaPlotData, col = merger, sep = '_',
                        into = c("trtmt", "time", "replicate"))

exp2_pca <- ggplot(pcaPlotData, aes_string(color = 'trtmt', shape = 'time', x = 'PC1', y = 'PC2')) +
  geom_point(size = 3,  stroke = 1.5) +
  coord_equal() +
  theme_dominik +
  scale_color_brewer(palette = 'Dark2')

png(file = "exp2_rnaseq_pca.png", res = 300, width = 2000, height = 2000)
exp2_pca
dev.off()
```

```
## pdf
##   2
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



6.2.5 Plot some genes of interest

```
# plyr::count(exp2_annotation$category)
# plyr::count(exp2_annotation$BIN1)

## Choose a gene set to plot

# grep('hormone', unique(exp2_annotation$category), value = T)

aba_genes <- subset(exp2_annotation, grepl("hormone metabolism.abscisic acid.signal", category))

## Combine the gene set with the data
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

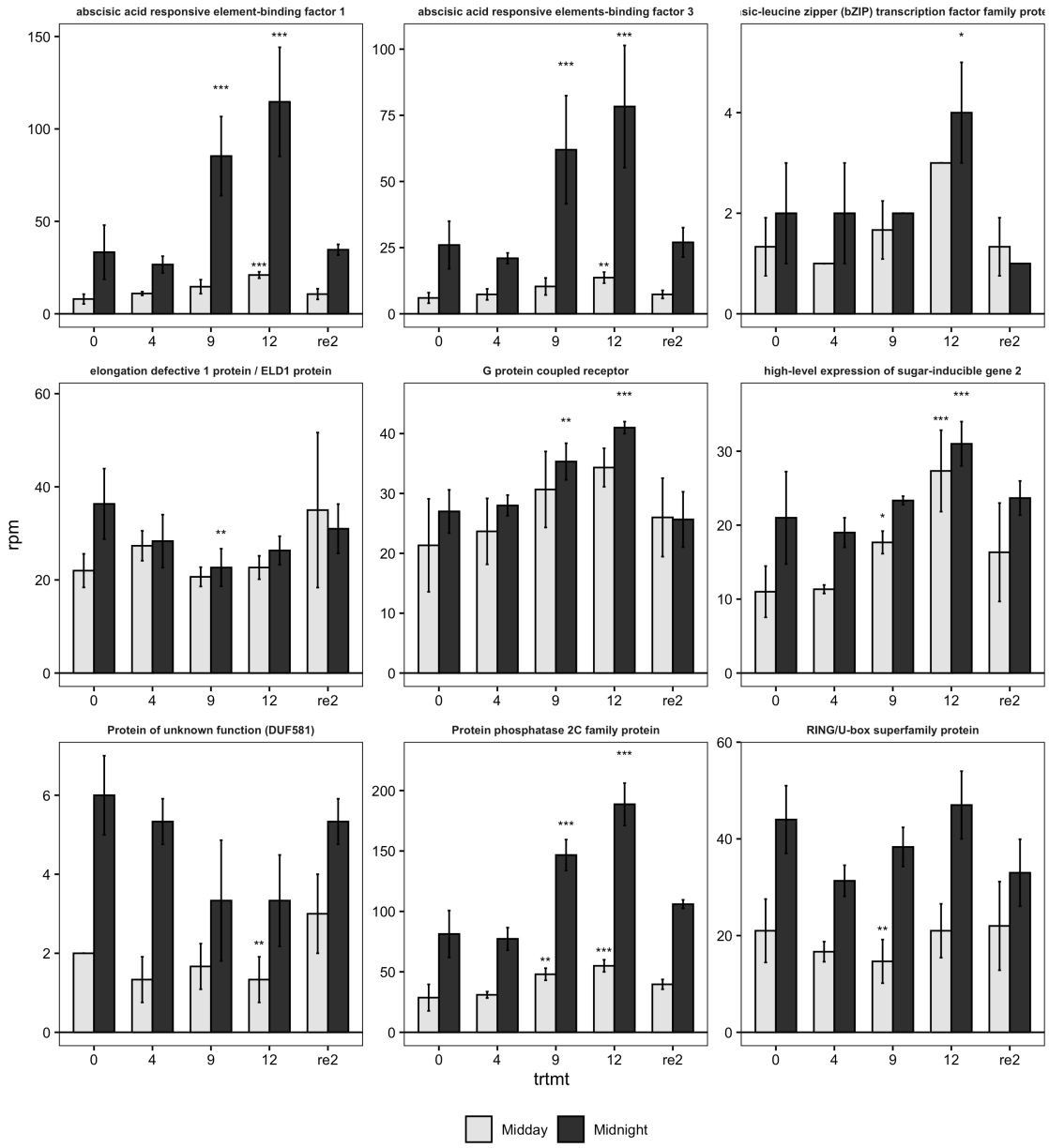
```
aba_rpm <- merge(aba_genes, exp2_rpm_long)
aba_stats <- merge(aba_genes, exp2_stats_long)

## Calculate the y position for the asterisks

aba_ymax <- aba_rpm %>%
  group_by(gene.name, trtmt, time) %>%
  dplyr::summarise(ymax = max(rpm))

aba_stats <- merge(aba_stats, aba_ymax)

aba_plotdata <- summarySE(aba_rpm, groupvars = c('trtmt', 'time', 'gene.name'),
                          measurevar = 'rpm')
aba_plotdata <- merge(aba_plotdata, aba_stats, all.x = T)

exp2_plot <- ggplot(aba_plotdata, aes(x = trtmt, y = rpm, fill = time)) +
  geom_hline(yintercept = 0, col = 'black') +
  facet_wrap(~gene.name, scales = 'free') +
  geom_col(position = position_dodge(width = 0.7), width = 0.7, col = 'black') +
  geom_errorbar(aes(ymin = rpm - sd, ymax = rpm + sd), #
                position = position_dodge(width = 0.7), width = 0.1, col = 'black') +
  geom_text(aes(label = asterisks, y= 1.1*ymax),
            position = position_dodge(width = 0.7), size = 3, col = 'black') +
  theme_dominik +
  scale_fill_manual(values = c(MD = 'grey90', MN = 'grey20'), labels = c('Midday', 'Midnight')) +
  scale_y_continuous(limits = c(0,NA))


png(file = "exp2_rnaseq_geneset.png", res = 300, width = 3000, height = 3000)
exp2_plot
```

```
## Warning: Removed 18 rows containing missing values (geom_text).
```

```
dev.off()
```

```
## pdf
##   2
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



6.2.6 Kmeans clustering

```
md_rpm <- subset(exp2_rpm_long, time == 'MD')

md_rpm_means <- md_rpm %>%
      group_by(AGI, trtmt) %>%
      summarise(mean_rpm = mean(rpm))

kmeans_data <- as.data.frame(pivot_wider(md_rpm_means,
                                         names_from = trtmt,
                                         id_cols = AGI,
                                         values_from =  mean_rpm))
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
row.names(kmeans_data) <- kmeans_data$AGI
kmeans_data <- kmeans_data[, -1]

# Filter
km_to_cluster <- kmeans_data[apply(kmeans_data, 1,
                        function(x) min(x) > 0 & max(x) > 10), ]
# log-transform
km_to_cluster <- log2(km_to_cluster)
#scale
km_to_cluster <- t(scale(t(km_to_cluster)))

km_results <- kmeans(km_to_cluster, centers = 10, iter.max = 500, nstart=100)
cluster_mship <- as.data.frame(km_results$cluster)
colnames(cluster_mship) <- 'km'
cluster_mship$AGI <-  rownames(cluster_mship)


plyr::count(cluster_mship$km)

km_plotdata <- as.data.frame(km_to_cluster)
km_plotdata$AGI <- rownames(km_plotdata)
km_plotdata <- pivot_longer(km_plotdata, -AGI,
                            values_to = "z_scores", names_to = "trtmt")


km_plotdata <- merge(km_plotdata, cluster_mship, by = "AGI")

km_plotdata$trtmt <- factor(km_plotdata$trtmt, levels = unique(km_plotdata$trtmt))
```

6.2.7 Safe point

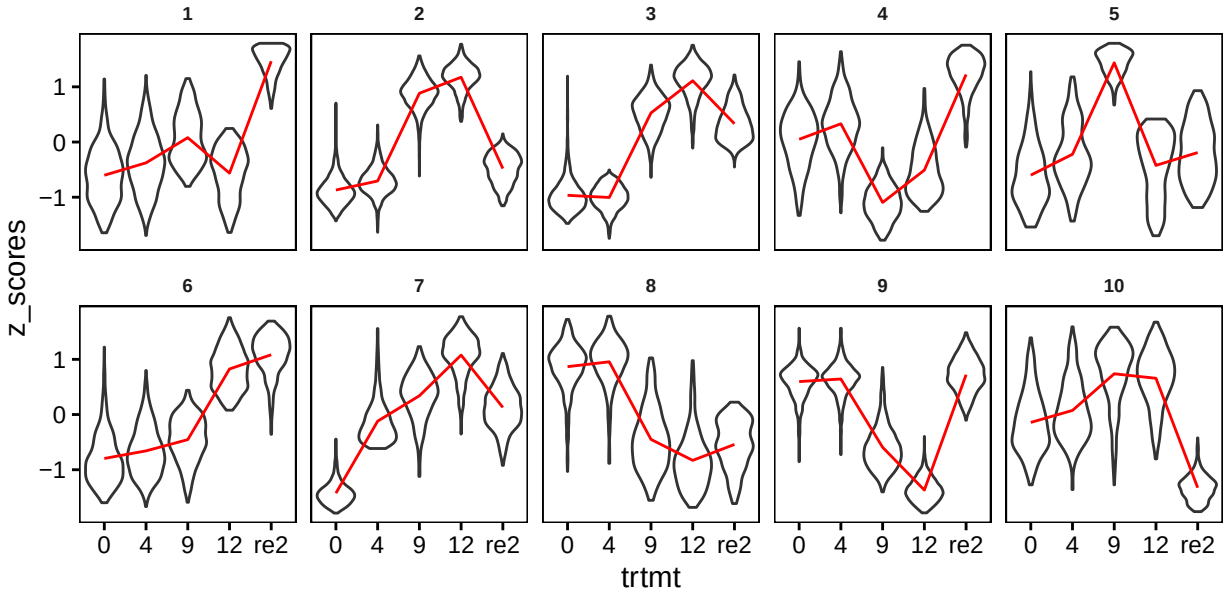
```
save(cluster_mship, km_plotdata, file = "exp2_kmeans.RData")
```

```
load(file = "exp2_kmeans.RData")
```

6.2.8 Plot kmeans results

```
km_plot <- ggplot(km_plotdata, aes(x = trtmt, y = z_scores)) +
  geom_violin(aes(group = trtmt), scale = 'width') +
  # stat_summary(fun = mean, geom = "line", col = "grey", aes(group = AGI)) +
  stat_summary(fun = mean, geom = "line", col = "red", aes(group = 1)) +
  facet_wrap(~ km, ncol = 5) +
  theme_dominik

km_plot
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



6.2.9 Run fisher's exact test on kmeans results + functional categorization

```
km_fishers <- merge(cluster.mship, exp2_annotation, "AGI")
km_fishers <- unite(km_fishers, c("BIN1", "BIN2"), col = "test_term", sep = "_", remove = F)

fish_list <- list()
for(i in unique(km_fishers$km))
{

  fishers_results <-  data.frame(matrix(NA, 0,3))
  for(category_term in unique(km_fishers$test_term))
  {
    gene.count <- length(unique(km_fishers[km_fishers$test_term == category_term,"AGI"]))

    InIn <- sum(km_fishers$km == i & km_fishers$test_term == category_term)
    InOut <- sum(km_fishers$km == i & km_fishers$test_term != category_term)
    OutIn <- sum((km_fishers$km != i & km_fishers$test_term == category_term))
    OutOut <- sum((km_fishers$km != i & km_fishers$test_term != category_term))

    p <- fisher.test(matrix(c(InIn, InOut, OutIn, OutOut), nrow = 2))$p.value

    if(InIn/InOut > OutIn/OutOut){represented  = "over"}else{represented  = "under"}

    fishers_results <- rbind.data.frame(fishers_results,
                                        cbind.data.frame(category_term, p,
                                                         represented, gene.count,
                                                         InIn, InOut, OutIn, OutOut))

  }

  fishers_results$km <- i

  fish_list[[as.character(i)]] <- fishers_results
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
}

### Rbind all results together
final_fish <- do.call(rbind.data.frame, fish_list)
### Correction for multiple hypothesis testing
final_fish$q <- p.adjust(final_fish$p, method = "BH")
### Remove non-significant and under-represented
final_fish <- subset(final_fish, represented == "over" & q <= 0.05)
```

6.2.10 Safe point

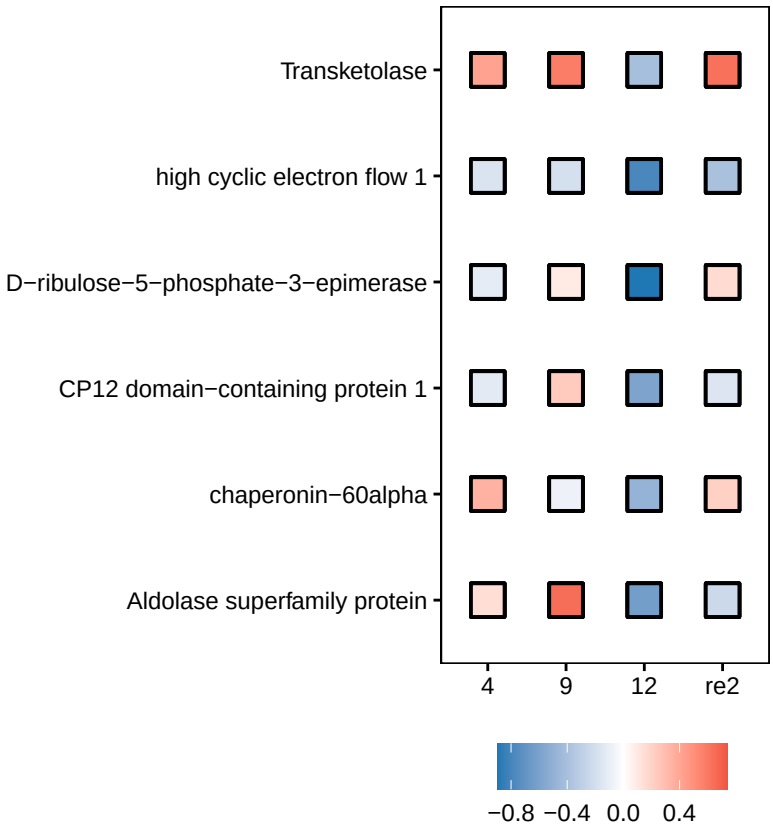
```
save(final_fish, km_fishers, file = "exp2_fishers.RData")
```

```
load(file = "exp2_fishers.RData")
```

6.2.11 Heat map

```
geneset <- subset(km_fishers, km == 5 & test_term == "PS_calvin cycle")

plotset <- merge(geneset, exp2_lfc_long)

# ggplot(plotset, aes(x = trtmt, y = gene.name, fill = log2fc)) +
#       geom_tile() +
#       scale_fill_gradient2(low =  "#1f78b4", high = "#e41a1c", midpoint = 0) +
#       theme(aspect.ratio = 7) +
#       theme_dominik

ggplot(plotset, aes(x = trtmt, y = gene.name, fill = log2fc)) +
      scale_fill_gradient2(low =  "#1f78b4", high = "#e41a1c", midpoint = 0) +
      geom_point(color = "black", stroke = 1, fill = NA, size = 6, shape = 22) +
      geom_point(alpha = 1, stroke = 1, size = 6, shape = 22) +
      # geom_text(size = 3, color = "black", vjust = 0.7) +
      theme_dominik +
      theme(aspect.ratio = 2) +
      theme(axis.title = element_blank())
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197



6.2.12 Exercise session 4

- Pick your favorite pathway from the annotation and plot a log-FC heat map

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

7 RMarkdown

- markdown
- YAML header
- References
- Code chunks (bash, python) and options
- Benefit: Reproducibly package data with code
- Output formats

7.0.1 Exercise session 5

Compile your code leading to one of the plots in exercises 2 or 4. (incl. data loading, shaping, etc.) in a nice RMarkdown.

---

8 Shiny

Note: the following does not work in this pdf reader, but needs to be copied and compiled in a new RMarkdown Shiny html document.

8.1 Interactive documents

8.1.1 Load data and package

```
load("exp1_plotdata.RData")
library(tidyverse)
library(shiny)
```

8.1.2 Single selection

```
available_compounds <- sort(unique(exp1_plotdata$Compound))

selectInput("compound", label = "Compound",
            choices = available_compounds,
            selected = sample(available_compounds, size = 1))

renderPlot({

  selected_subset <- subset(exp1_plotdata, Compound == input$compound)

  ggplot(selected_subset, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
    stat_summary(fun = 'mean', geom = 'bar',
                 position = position_dodge(width = 0.8), width = 0.7) +
    stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
                 position = position_dodge(width = 0.8)) +
    scale_fill_brewer(palette = 'Dark2') +
    theme_bw() +
    theme(strip.background = element_rect(fill = NA)) +
    theme(legend.position = 'bottom', legend.title = element_blank()) +
    theme(aspect.ratio = 1)

})
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

8.1.3 Multiple compounds

```
available_compounds <- sort(unique(exp1_plotdata$Compound))

inputPanel(
  selectInput(multiple = T, "compound", label = "Compound",
              choices = available_compounds,
              selected = sample(available_compounds, size = 5))
)


renderPlot({

  selected_subset <- subset(exp1_plotdata, Compound %in% input$compound)

  ggplot(selected_subset, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
    stat_summary(fun = 'mean', geom = 'bar',
                 position = position_dodge(width = 0.8), width = 0.7) +
    stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
                 position = position_dodge(width = 0.8)) +
    facet_wrap(~Compound, scales = 'free_y') +
    scale_fill_brewer(palette = 'Dark2') +
    theme_bw() +
    theme(strip.background = element_rect(fill = NA)) +
    theme(legend.position = 'bottom', legend.title = element_blank()) +
    theme(aspect.ratio = 1)

})
```

8.1.4 Sidebar layout

```
available_compounds <- sort(unique(exp1_plotdata$Compound))

sidebarLayout(

  sidebarPanel(

    selectInput(multiple = T, "compound", label = "Compound",
                choices = available_compounds,
                selected = sample(available_compounds, size = 5))
  ),

  mainPanel(

    renderPlot({

      selected_subset <- subset(exp1_plotdata, Compound %in% input$compound)

      ggplot(selected_subset, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
        stat_summary(fun = 'mean', geom = 'bar',
                     position = position_dodge(width = 0.8), width = 0.7) +
        stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
                     position = position_dodge(width = 0.8)) +
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

```
        facet_wrap(~Compound, scales = 'free_y') +
        scale_fill_brewer(palette = 'Dark2') +
        theme_bw() +
        theme(strip.background = element_rect(fill = NA)) +
        theme(legend.position = 'bottom', legend.title = element_blank()) + theme(aspect.ratio = 1)

    })


  )
)
```

8.1.5 Exercise session 6

Compile your code leading to one of the previous plots (incl. data loading, shaping, etc.) in an interactive shiny document.

---

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

8.2 Shiny App

```
load("exp1_plotdata.RData")
library(tidyverse)
library(shiny)

available_compounds <- sort(unique(exp1_plotdata$Compound))

# Define UI for application
ui <- fluidPage(

    # Application title
    titlePanel("My favorite Metabolite Plot App"),

    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            selectInput(multiple = T, "compound", label = "Compound",
                        choices = available_compounds,
                        selected = sample(available_compounds, size = 5))
        ),

        mainPanel(
           plotOutput("thePlot")
        )
    )
)


# Define server logic
server <- function(input, output) {

    output$thePlot <-  renderPlot({

        selected_subset <- subset(exp1_plotdata, Compound %in% input$compound)

        ggplot(selected_subset, aes(x = Condition.1, y = relative_resp, fill = Condition.3)) +
            stat_summary(fun = 'mean', geom = 'bar', position = position_dodge(width = 0.8), width = 0.
            stat_summary(fun.data = 'mean_se', geom = 'errorbar', width = 0.1,
                         position = position_dodge(width = 0.8)) +
            facet_wrap(~Compound, scales = 'free_y') +
            scale_fill_brewer(palette = 'Dark2') +
            theme_bw() +
            theme(strip.background = element_rect(fill = NA)) +
            theme(legend.position = 'bottom', legend.title = element_blank()) +
          theme(aspect.ratio = 1)

    })
}

# Run the application
shinyApp(ui = ui, server = server)
```


dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197

References

Brilhaus, Dominik, Bräutigam, Andrea, Mettler-Altmann, Tabea, Winter, Klaus, and Weber, Andreas P M (2016). Reversible Burst of Transcriptional Changes during Induction of Crassulacean Acid Metabolism in Talinum triangulare. Plant Physiology **170**: 102–122.

dominik.brilhaus@hhu.de
ORCID: 0000-0001-9021-3197