
COMPUTER SETUP

- If possible, use a computer/laptop on which you have **administrator rights**. Otherwise, you might run into problems when installing the software.
- **Make sure you have the necessary software installed and up to date on your laptop.** We will code in RStudio, which requires both R and RStudio installed. For installation, follow the instructions here: https://datacarpentry.org/r-socialsci/#setup-instructions . If you have those already installed, make sure you have R version 4.0 or newer (and, if necessary, update to the latest version).
- If you have any trouble, **send us an email by Monday (26 Feb) morning,** so we can help you sort this out **before** the course starts on Tuesday. If need be we can set up a call to help you sort out any issues.
- The lesson also requires the **“BiocManager" package installed**, and all necessary data downloaded and saved on your computer. You can copy and execute the code in R/RStudio (https://carpentries-incubator.github.io/high-dimensional-stats-r/setup.html) before the workshop. However, as we still work on ways to best deliver this workshop, we will also provide the setup code a day before the workshop starts in our collab document https://pad.carpentries.org/2024-02-27_ed-dash_high-dim-stats. Additionally, the first workshop hour will deal with the setup and possible issues stemming from it.

ZOOM METING DETAILS

- We will be using **Zoom** for this workshop. If you do not have it already installed, please download **Zoom** from here: https://zoom.us/download.
- **Zoom meeting link**: https://ed-ac-uk.zoom.us/j/85716381279
- Meeting ID: 857 1638 1279
- Time: 10:00 AM - 1:00 PM (UK TIME)
- Passcode: in the email

R CODE NEEDED TO INSTALL ALL PACKAGES AND DOWNLOAD DATA

```
# last minute change in the first download link used, instead of gh-pages (as in the website) we use main
install.packages("BiocManager")
download.file(
    "https://raw.githubusercontent.com/carpentries-incubator/high-dimensional-stats-r/main/dependencies.csv",
    destfile = 'dependencies.csv')
table <- read.table('dependencies.csv')
BiocManager::install(table[[1]])
dir.create("data", showWarnings = FALSE)
data_files <- c(
    "cancer_expression.rds",
    "coefHorvath.rds",
    "methylation.rds",
    "scrnaseq.rds",
    "prostate.rds")
for (file in data_files) {
    download.file(
        url = file.path(
            "https://raw.githubusercontent.com/carpentries-incubator/high-dimensional-stats-r/gh-pages/data",
            file
        ),
        destfile = file.path("data", file)
    )}

# additonally, we will be using some elements of tidymodels that may not be in the original libraries list
install.packages("tidymodels")
```

ALTERNATIVE INSALLATION METHOD

1. Create a new project in RStudio
2. Install renv with install.packages or the RStudio GUI
3. Download the DESCRIPTION file https://drive.google.com/file/d/1UVEiiHe4wVWXGuclyKEA6nCYN7dMbXp8/view?usp=drive_link and drop it the top level of your R project directory
4. Run renv::init(bioconductor = TRUE)
5. Select option 1 to install from the description file
6. run the code to get remaining data packages:

```
dir.create("data", showWarnings = FALSE)data_files <- c(
    "cancer_expression.rds",
    "coefHorvath.rds",
    "methylation.rds",
    "scrnaseq.rds",
    "prostate.rds")
for (file in data_files) {
    download.file(
        url = file.path(
            "https://raw.githubusercontent.com/carpentries-incubator/high-dimensional-stats-r/gh-pages/data",
            file
        ),
        destfile = file.path("data", file)
    )}
```

WEBSITES

WORKSHOP ORGANIZATIONAL WEBSITE: https://edcarp.github.io/2024-02-27_ed-dash_high-dim-stats/
THE LESSON: https://carpentries-incubator.github.io/high-dimensional-stats-r/
EDCARP PAD: https://pad.carpentries.org/2024-02-27_ed-dash_high-dim-stats

Day 4 (1st March)

ATTENDANCE:

Andrzej Romaniuk (Host)
Aleksandra Chybowska (helper)
Alan O'Callaghan (helper)
Karim Abu Nahia
Nneka Nnadi
Siva kumar
Josie Robertson
Manal Alharbi
Juan Zeng
Shikha Vashisht
Maciej Migdał

LIVE CODE

```
#How to perform PCA
#first, get the data
prostate <- readRDS(here("data/prostate.rds"))
#the data rows represent unique patients
head(prostate)
#creating a substed of data we are interested in
pros2 <- prostate[, c("lcavol", "lweight", "lbph", "lcp", "lpsa")]
head(pros2)

#Are there are differences in variances betwen data?
```

```
apply(pros2, 2, var)

par(mfrow = c(1, 2))
hist(pros2$lweight, breaks = "FD")
hist(pros2$lbph, breaks = "FD")

#When perofming pcy, we state the data source (pros2), with scaling = TRUE due to values having
differences in variances, centering on 0
pca.pros <- prcomp(pros2, scale = TRUE, center = TRUE)
pca.pros

# how many axes we need? What percent of variability each one explains?
varExp <- (pca.pros$sdev^2) / sum(pca.pros$sdev^2) * 100
# calculate percentage variance explained using output from the PCA
varDF <- data.frame(Dimensions = 1:length(varExp), varExp = varExp)
plot(varDF)

#Cteating a biplot
stats::biplot(pca.pros, xlim = c(-0.3, 0.3))



#
library("PCAtools")

library("SummarizedExperiment")
cancer <- readRDS(here::here("data/cancer_expression.rds"))
mat <- assay(cancer)
metadata <- colData(cancer)
View(mat)
View(metadata)
#Whether columns match
all(colnames(mat) == rownames(metadata))

#CHALLENGE 3
pc <- pca(mat, metadata = metadata)
#Many PCs explain a very small amount of the total variance in the data
#Remove the lower 20% of PCs with lower variance
pc <- pca(mat, metadata = metadata, removeVar = 0.2)
#Explore other arguments provided in pca
pc$rotated[1:5, 1:5]

#Exploring loadings
pc$loadings[1:5, 1:5]
which.max(pc$loadings[, 1])
pc$loadings[49, ]
pc$loadings[27, ]

#Challenge 4
screeplot(pc, axisLabSize = 5, titleLabSize = 8)
```

DAY 3 (29 Feb)

ATTENDANCE:

Andrzej Romaniuk (Host)
Aleksandra Chybowska (helper)
Alan O'Callaghan (helper)
Karim Abu Nahia
Nneka Nnadi
siva kumar
Maciej Migdał
Shikha Vashisht
Manal
Juan Zeng
Josie Robertson

LIVE CODE

```
# best to start from loading libraries we previously used
library("here")
library("minfi")

#Setup for the Regularised Regression
coef_horvath <- readRDS(here::here("data/coefHorvath.rds"))
methylation <- readRDS(here::here("data/methylation.rds"))
library("SummarizedExperiment")
age <- methylation$Age
methyl_mat <- t(assay(methylation))
coef_horvath <- coef_horvath[1:20, ]
features <- coef_horvath$CpGmarker
horvath_mat <- methyl_mat[, features]
#Generate an index to split the data
set.seed(42)train_ind <- sample(nrow(methyl_mat), 25)

#CHALLENGE 3

#first, we split data
train_mat <- horvath_mat[train_ind, ]
train_age <- age[train_ind]
test_mat <- horvath_mat[-train_ind, ]
test_age <- age[-train_ind]
```

```
# as.data.frame() converts train_mat into a data.frame
fit_horvath <- lm(train_age ~ ., data = as.data.frame(train_mat))
mean(residuals(fit_horvath)^2)


#We use glmnet to show how things change as we restrict/increase pentalty
library("glmnet")
## glmnet() performs scaling by default, supply un-scaled data:
horvath_mat <- methyl_mat[, features]
# select the first 20 sites as before
train_mat <- horvath_mat[train_ind, ]
# use the same individuals as selected before
test_mat <- horvath_mat[-train_ind, ]


ridge_fit <- glmnet(x = train_mat, y = train_age, alpha = 0)
plot(ridge_fit, xvar = "lambda")
abline(h = 0, lty = "dashed")


#Challenge 4
#ridge regression works better on unseen data
min_err_ridge
err_lm
#ridge shows lesser spread
all <- c(pred_lm, test_age, pred_min_ridge)
lims <- range(all)par(mfrow = 1:2)
plot(test_age, pred_lm,
    xlim = lims, ylim = lims,
    pch = 19)
abline(coef = 0:1, lty = "dashed")
plot(test_age, pred_min_ridge,
    xlim = lims, ylim = lims,
    pch = 19)
abline(coef = 0:1, lty = "dashed")


#Challenge 5
fit_lasso <- glmnet(x = methyl_mat, y = age, alpha = 1)


#Challenge 6
#fitting an elastic model is like fitting a lasso model
elastic <- glmnet(methyl_mat[, -1], age, alpha = 0.5)
plot(elastic)

#model selection similar
elastic_cv <- cv.glmnet(methyl_mat[, -1], age, alpha = 0.5)
plot(elastic_cv)
```

# coefficients similar with lasso, but more conservative for elastic
coefe <- coef(elastic_cv, elastic_cv$lambda.1se)
sum(coefe[, 1] == 0)
sum(coefl[, 1] == 0)

plot(
    coefl[, 1], coefe[, 1],
    pch = 16,
    xlab = "LASSO coefficients",
    ylab = "Elastic net coefficients")abline(0:1, lty = "dashed")

DAY 2 (28 Feb)

ATTENDANCE:

Andrzej Romaniuk (Host)
Aleksandra Chybowska (helper)
Alan O'Callaghan (helper)
Karim Abu Nahia
Siva Kumar
Nneka Nnadi
Maciej Migdał
Manal
Shikha Vashisht
Juan Zeng
Josie Robertson

LIVE CODE

# tidy() helps extracting info on coef and assoc hypoth. details
# remember to run code from previous day if you do not have an object lm_age_methyl1 in your envrionment
library("broom")
tidy(lm_age_methyl1)

# we are interested in coefficients
coef_age_methyl1 <- tidy(lm_age_methyl1)[2,]
coef_age_methyl1
# however, obtained p-values are qite high

## HOW DOES HYPOTHESIS TESTING FOR A SUCH A MoODEL WORK?

# saving all the tidy data as a separate object

```
table_age_methyl1 <- tidy(lm_age_methyl1)

# creating t-statistic & estimating standard error using all.equal()
tvals <- table_age_methyl1$estimate / table_age_methyl1$std.error
all.equal(tvals, table_age_methyl1$statistic)

# same, but for p-values
pvals <- 2 * pt(abs(tvals), df = lm_age_methyl1$df, lower.tail = FALSE)
all.equal(table_age_methyl1$p.value, pvals)

## SHARING INFORMATION ACROSS OUTCOME VARIABLES

# we will use a package called limma
library("limma")
design_age <- model.matrix(~age)
dim(design_age)
head(design_age)

# how data() function works, on cars dataset
data(cars)
head(cars)

# fit regression model using speed as a predictor (from cars)
mod1 <- lm(dist ~ speed, data=cars)
# the model matrix contains two columns: intercept and speed
head(model.matrix(mod1))

fit_age <- lmFit(methyl_mat, design = design_age)
fit_age <- eBayes(fit_age)

toptab_age <- topTable(fit_age, coef = 2, number = nrow(fit_age))
# order by effect size (absolute log-fold change)
orderEffSize <- rev(order(abs(toptab_age$logFC)))
head(toptab_age[orderEffSize, ])

plot(toptab_age$logFC, -log10(toptab_age$P.Value),
    xlab = "Effect size", ylab = bquote(-log[10](p-value)),
    pch = 19
)

#CHALLENGE 4
#First, we extract a matrix from methylation data
design_smoke <- model.matrix(~methylation$smoker)
#Fitting data
fit_smoke <- lmFit(methyl_mat, design = design_smoke)
fit_smoke <- eBayes(fit_smoke)
toptab_smoke <- topTable(fit_smoke, coef = 2, number = nrow(fit_smoke))
#finally, we create a plot
plot(toptab_smoke$logFC, -log10(toptab_smoke$P.Value),
    xlab = "Effect size", ylab = bquote(-log[10](p)),
    pch = 19
)

#THE PROBLEM OF MULTIPLE OUTCOMES
age_perm <- age[sample(ncol(methyl_mat), ncol(methyl_mat))]design_age_perm <-
model.matrix(~age_perm)
fit_age_perm <- lmFit(methyl_mat, design = design_age_perm)fit_age_perm <-
eBayes(fit_age_perm)toptab_age_perm <- topTable(fit_age_perm, coef = 2, number
= nrow(fit_age_perm))
plot(toptab_age_perm$logFC, -log10(toptab_age_perm$P.Value),
    xlab = "Effect size", ylab = bquote(-log[10](p)),
    pch = 19)abline(h = -log10(0.05), lty = "dashed", col = "red")

#BONFERRONI CORRECTION
#most conservative way of adjusting multiple tests to take into account false positvies and negatives
p_raw <- toptab_age$P.Valuep_fwer <- p.adjust(p_raw, method = "bonferroni")
#we will usse ggplot for visualisation
library("ggplot2")
ggplot() +
    aes(p_raw, p_fwer) +
    geom_point() +
    scale_x_log10() + scale_y_log10() +
    geom_abline(slope = 1, linetype = "dashed") +
    geom_hline(yintercept = 0.05, linetype = "dashed", col = "red") +
    geom_vline(xintercept = 0.05, linetype = "dashed", col = "red") +
    labs(x = "Raw p-value", y = "Bonferroni p-value")

# CHALLENGE 6
#adjusting p values
p_fdr <- p.adjust(p_raw, method = "BH")ggplot() +
    aes(p_raw, p_fdr) +
    geom_point() +
    scale_x_log10() + scale_y_log10() +
    geom_abline(slope = 1, linetype = "dashed") +
    geom_hline(yintercept = 0.05, linetype = "dashed", color = "red") +
    geom_vline(xintercept = 0.05, linetype = "dashed", color = "red") +
    labs(x = "Raw p-value", y = "Benjamini-Hochberg p-value")

#plotting the results from previous p-adjusting
ggplot() +
    aes(p_fdr, p_fwer) +
    geom_point() +
    scale_x_log10() + scale_y_log10() +
    geom_abline(slope = 1, linetype = "dashed") +
    geom_hline(yintercept = 0.05, linetype = "dashed", color = "red") +
    geom_vline(xintercept = 0.05, linetype = "dashed", color = "red") +
    labs(x = "Benjamini-Hochberg p-value", y = "Bonferroni p-value")
```

DAY 1 (27 Feb)

ATTENDANCE:

Andrzej Romaniuk (Host)
Aleksandra Chybowska (helper)
Alan O'Callaghan (helper)

Nneka Nnadi
Siva Kumar
Josie Robertson
Maciej Migdał
Karim Abu Nahia
Juan Zeng
Shikha Vashisht
Manal

LIVE CODE

```
# CHALLENGE 2
# loading library "here"
library("here")

# reading a previously saved rds file
Prostate <- readRDS(here("data/prostate.rds"))

dim(Prostate)       # print the number of rows and columns
names(Prostate)  # examine the variable names
head(Prostate)     # print the first 6 rows
names(Prostate)  # examine column names
pairs(Prostate)     # plot each pair of variables against each other

#CHALLENGE 3
# use cor() to examine correlations
cor(Prostate)
# rounding helps to visualise the correlations
round(cor(Prostate), 2)

# create a simple (univariate) regression model exploring the impact of
# highly correlated variables (here: gleason and pgg45 are 0.75)
# lm() to create a model
model1 <- lm(age ~ gleason, data = Prostate)
model2 <- lm(age ~ pgg45, data = Prostate)

# summary() to explore the results
```

```
summary(model1)
summary(model2)

# create a mutivariate regression model using both variables (gleason + pgg45)
model3 <- lm(age ~ gleason + pgg45, data = Prostate)
# check the results (no actual impact if using both as covariates)
summary(model3)

#USING BIOCONDUCTOR
# example package 'minfi', specifically for analysing
# Illumina Infinium DNA methylation arrays
library("minfi")

# we can explore this package using browseVignettes()
browseVignettes("minfi")

# example setup to explore methylation dataset
library("here")
library("ComplexHeatmap")

methylation <- readRDS(here("data_input/methylation.rds"))
# Oryginal dataset is complex, and not realy a table, so to view it we will use colData() and head()
head(colData(methylation))

# assay() function creates a matrix-like (table) object where rows represent
# probes for genes and columns represent samples.
methyl_mat <- t(assay(methylation))

# example usage
# calculate correlations between cells in matrix
cor_mat <- cor(methyl_mat)
# print the top-left corner of the correlation matrix
cor_mat[1:10, 1:10]

#DNA METHYLATION DATA
# starting from methylation data, see previous day
# methylation is a GenomicRatioSet object, a specific type of object
methylation
dim(methylation)
# 5000 fatures x 37 columns

# assay() for extracting this data
methyl_mat <- assay(methylation)

# distribution of m-values
hist(methyl_mat, breaks = "FD", xlab = "M-value")

# exploring data, using knitr:kable
knitr::kable(head(colData(methylation)), row.names = FALSE)
```

```
# similar to head(colData(methylation)) , but can be better in some environments

# In this episode, we will focus on the association between
# age and methylation

# extracting data
age <- methylation$Age

# example of using a Heatmap() function, to show how many possible relationships are there
library("ComplexHeatmap")
order <- order(age)
age_ord <- age[order]
methyl_mat_ord <- methyl_mat[, order]

Heatmap(methyl_mat_ord,
        name = "M-value",
        cluster_columns = FALSE,
        show_row_names = FALSE,
        show_column_names = FALSE,
        row_title = "Feature",
        column_title =  "Sample",
        top_annotation = columnAnnotation(age = age_ord))
```

#REGRESSION WITH MANY OUTCOMES

```
# getting age data
age <- methylation$Age
# fitting model
# methyl_mat[1, ] indicates that the 1st CpG will be used as outcome variable
lm_age_methyl1 <- lm(methyl_mat[1, ] ~ age)
lm_age_methyl1

# plotting the results with the fitting line
plot(age, methyl_mat[1, ], xlab = "Age", ylab = "Methylation level", pch = 16)
abline(lm_age_methyl1)
```

THOUGHTS ABOUT THE COURSE:

WHAT YOU LIKED:
I enjoy the teaching approach of the main instrctor, and the explanation is quite straightforward.
Clear explanations, helpful to have pre-prepared data to use

WHAT COULD BE IMPROVED
we can spend more time on breakout room. This will give us more time to discuss
We could spend less time on the breakout rooms, imo there is no improvement to the amount of discussion than what we would have on an open forum plus our group is quite small anyway. It also feels like the breakout rooms limit our chances to interact with the main instructor. I would also like to see a

biger challange to solve at the end of each episode.
it is better if we spend more time with the main instructor to discuss the challenges rather than using the breakout rooms
Overall too much time spent on the first two topics
Would have been good to focus more on the later sections & cover more of these