Grundlagen der Bioinformatik

15.06.2018

Recap Assignment 3: Homology & Hierarchical Clustering

Assignment 4: Basics of R

Introduction to R

Everything you always wanted to know about R
(but were afraid to ask)

Getting you started

What you need

- Get running R installation
  - http://cran.r-project.org/
  - Ubuntu apt-get
- Gruenau2 has R-installation
  - Login via ssh or Putty

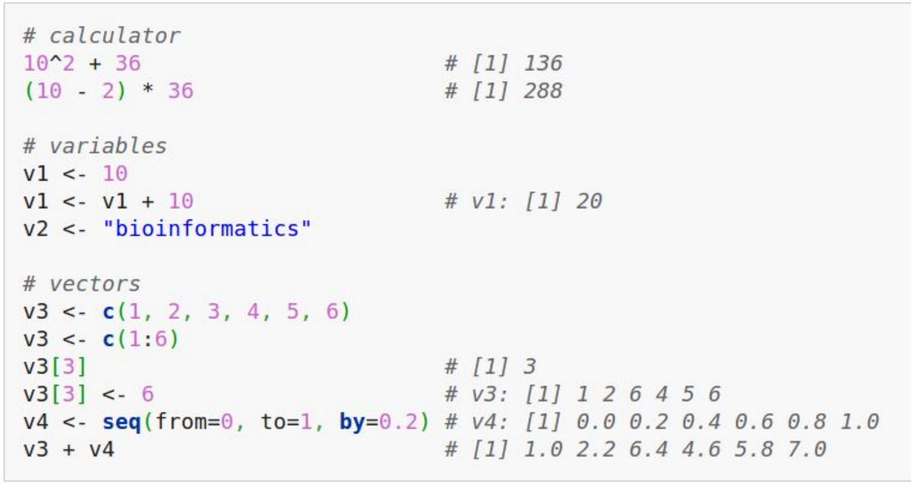
Introduction to R

- http://cran.r-project.org/doc/manuals/R-intro.pdf

Reference Card

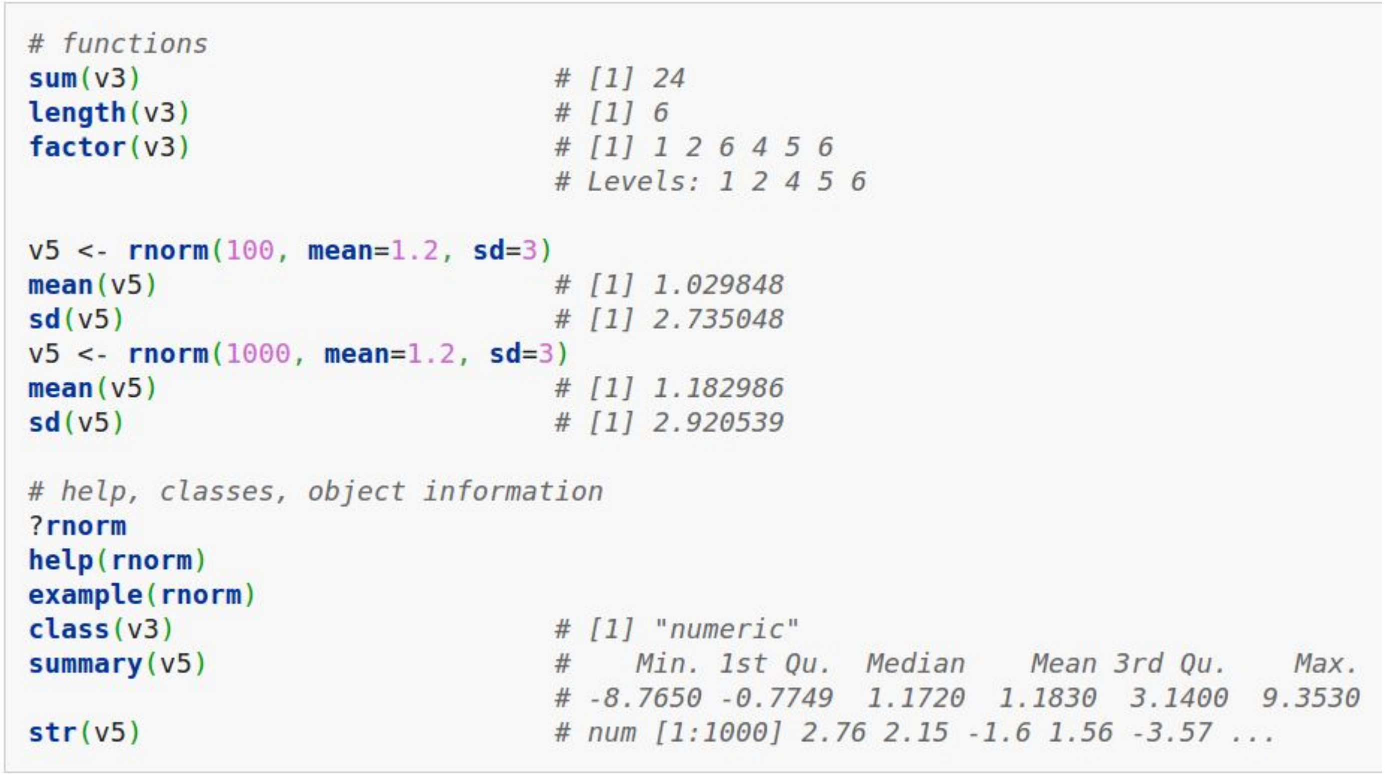
- Explanation of common commands
- http://cran.r-project.org/doc/contrib/Short-refcard.pdf

Calculator, Variables, Scalars, Vectors



```
# calculator
10^2 + 36                       # [1] 136
(10 - 2) * 36                   # [1] 288

# variables
v1 <- 10
v1 <- v1 + 10                   # v1: [1] 20
v2 <- "bioinformatics"

# vectors
v3 <- c(1, 2, 3, 4, 5, 6)
v3 <- c(1:6)
v3[3]                           # [1] 3
v3[3] <- 6                      # v3: [1] 1 2 6 4 5 6
v4 <- seq(from=0, to=1, by=0.2) # v4: [1] 0.0 0.2 0.4 0.6 0.8 1.0
v3 + v4                         # [1] 1.0 2.2 6.4 4.6 5.8 7.0
```

Functions, Help, Object Information



```r
# functions
sum(v3)                         # [1] 24
length(v3)                      # [1] 6
factor(v3)                      # [1] 1 2 6 4 5 6
                                # Levels: 1 2 4 5 6

v5 <- rnorm(100, mean=1.2, sd=3)
mean(v5)                        # [1] 1.029848
sd(v5)                          # [1] 2.735048
v5 <- rnorm(1000, mean=1.2, sd=3)
mean(v5)                        # [1] 1.182986
sd(v5)                          # [1] 2.920539

# help, classes, object information
?rnorm
help(rnorm)
example(rnorm)
class(v3)                       # [1] "numeric"
summary(v5)                     #    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
                                # -8.7650 -0.7749  1.1720  1.1830  3.1400  9.3530
str(v5)                         # num [1:1000] 2.76 2.15 -1.6 1.56 -3.57 ...
```

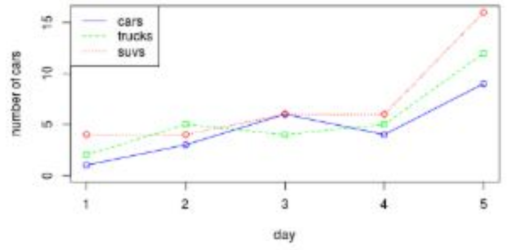
Matrices and Data Frames

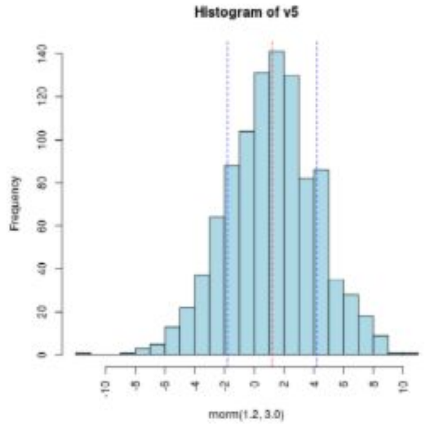
```r
# matrices
mat <- matrix(data=c(9,2,3,4,5,6), ncol=3)
mat[1,2]                        # [1] 3
mat[2,]                         # [1] 2 4 6
mean(mat)                       # [1] 4.833333
mean(mat[1,])                   # [1] 5.666667

# data.frames
d <- data.frame(cars=c(1,3,6,4,9), trucks=c(2,5,4,5,12), suvs=c(4,4,6,6,16))
d                               # cars trucks suvs
                                # 1    1      2    4
                                # 2    3      5    4
                                # 3    6      4    6
                                # 4    4      5    6
                                # 5    9     12   16
mean(d$trucks)                  # [1] 5.6
mean(d[,"trucks"])              # [1] 5.6
d[1,c("cars","trucks")]         # cars trucks
                                # 1    1      2
dim(d)                          # [1] 5 3
```

more data structures: lists, factors, tables, ...)

Default Plots

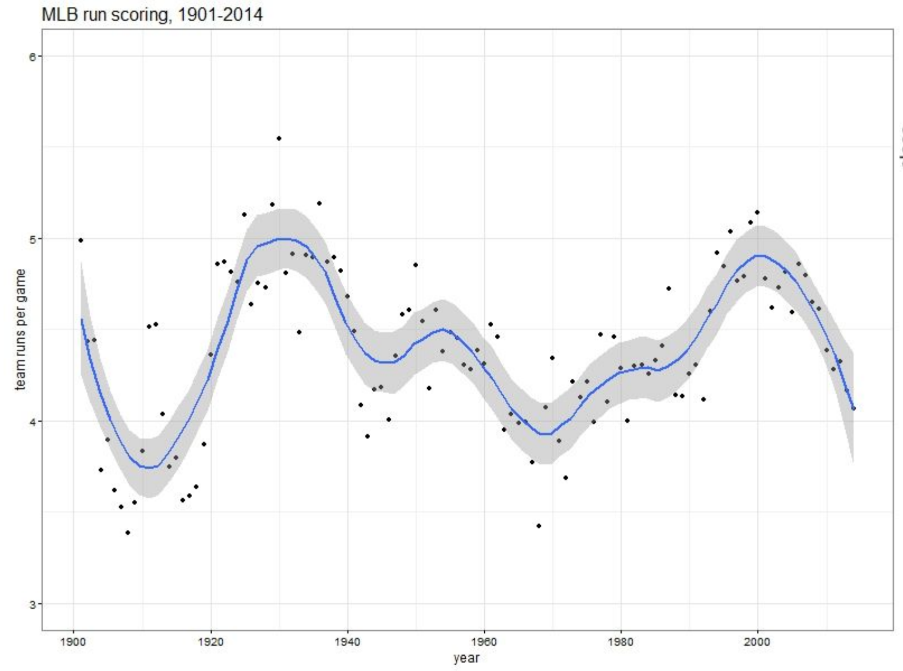
```r
png(file="cars.png", width=480, height=300)
plot(d[,"cars"], type="o", lty=1, col="blue",
     ylim=c(0,max(d)), ylab="number of cars", xlab="day")
lines(d[,"trucks"], type="o", pch=22, lty=2, col="green")
lines(d[,"suvs"], type="o", pch=23, lty=3, col="red")
legend("topleft", c("cars", "trucks", "suvs"),
       lty=c(1,2,3), col=c("blue", "green", "red"))
dev.off()

png(file="norm.png")
hist(v5, col="lightblue", breaks=20,
     xlab="rnorm(1.2, 3.0)", xaxt="n")
abline(v=1.2, col="red", lty=2)
abline(v=c(1.2-3, 1.2+3), col="blue", lty=2)
axis(1,seq(from=-10, to=10, by=2), las=2)
dev.off()
```

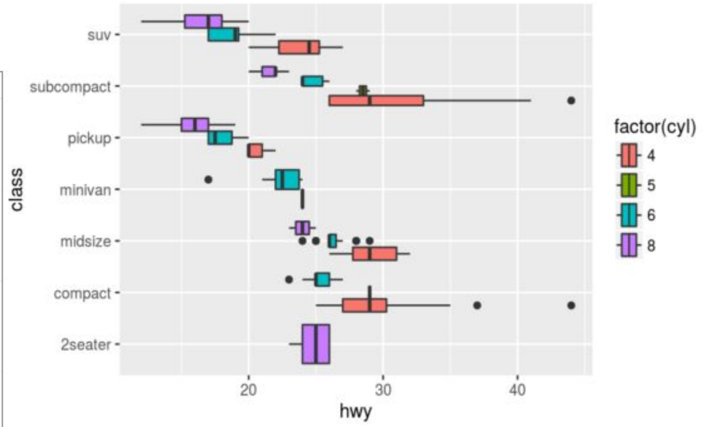


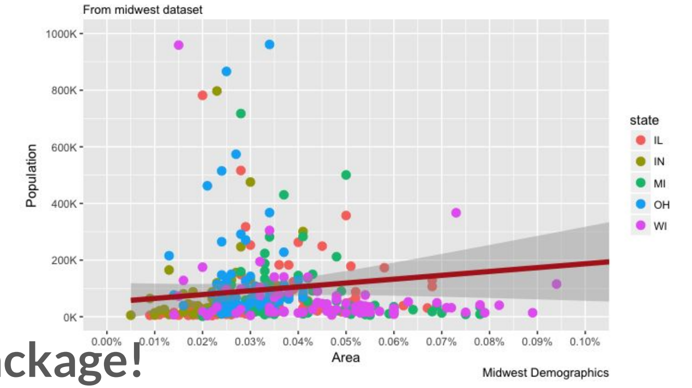


more plots: barplot, boxplot, pie, heatmap, …

Freestyler plots

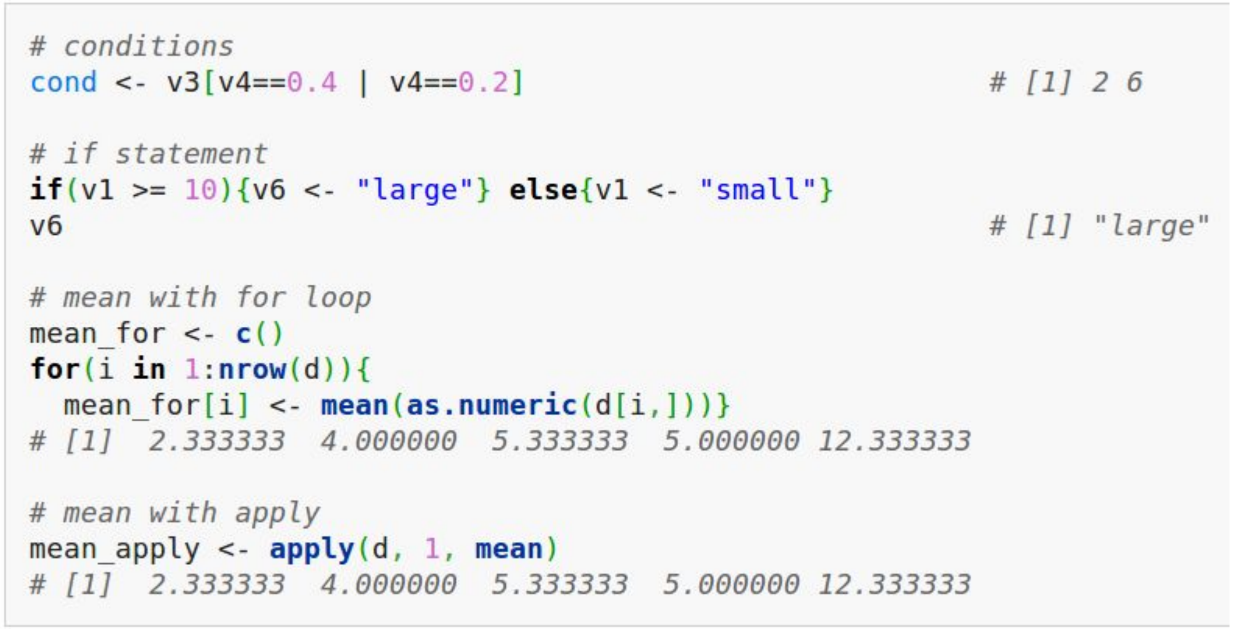






If you want to be one of the cool kids: ggplot2 package!

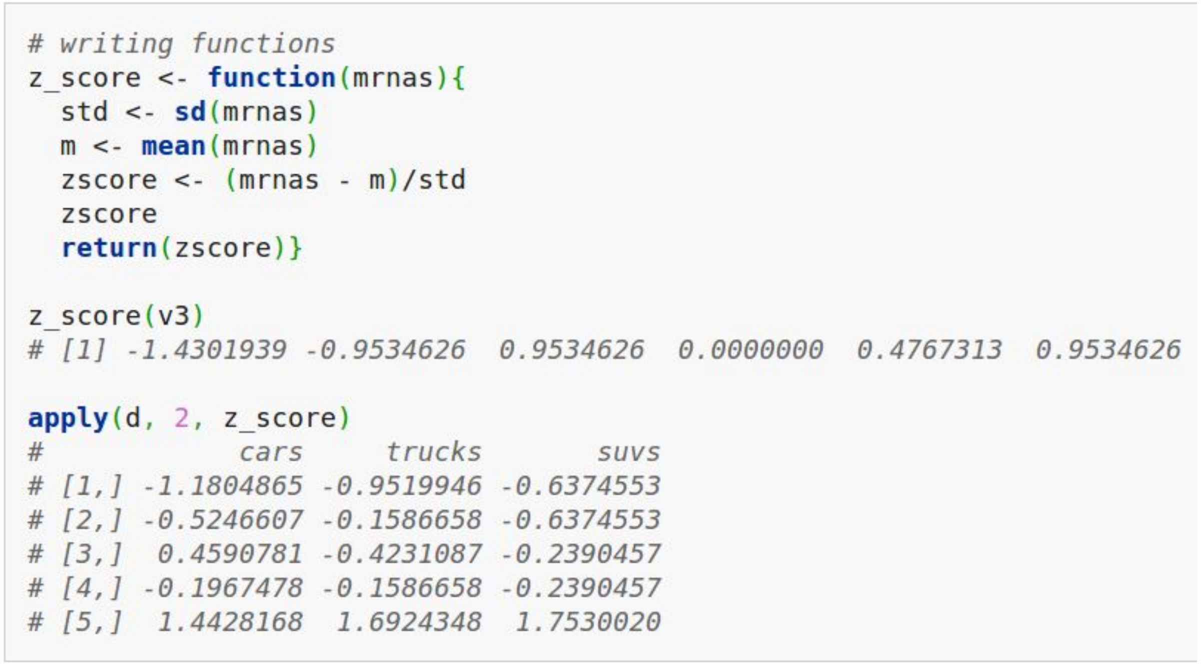
Programming – if, for, apply



```
# conditions
cond <- v3[v4==0.4 | v4==0.2]                          # [1] 2 6

# if statement
if(v1 >= 10){v6 <- "large"} else{v1 <- "small"}
v6                                                     # [1] "large"

# mean with for loop
mean_for <- c()
for(i in 1:nrow(d)){
  mean_for[i] <- mean(as.numeric(d[i,]))}
# [1]  2.333333  4.000000  5.333333  5.000000 12.333333

# mean with apply
mean_apply <- apply(d, 1, mean)
# [1]  2.333333  4.000000  5.333333  5.000000 12.333333
```

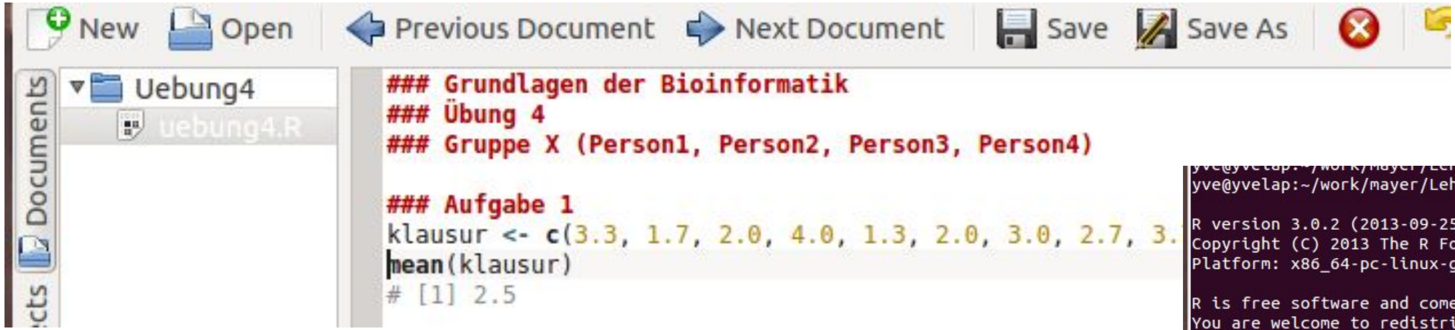
better use apply than for! similar functions are apply, sapply, tapply, ...

Functions

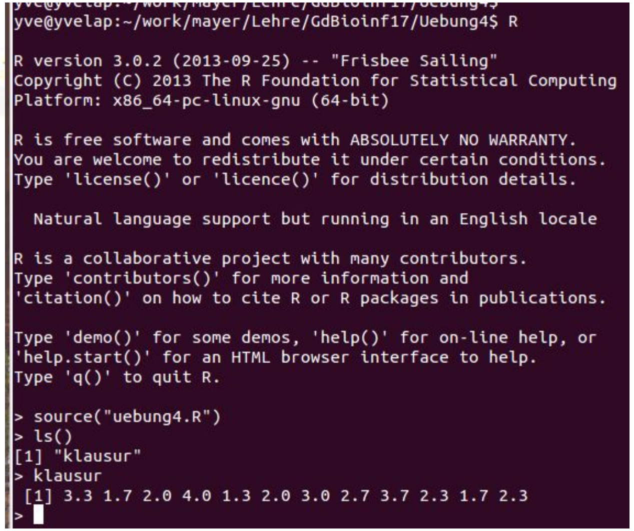


```
# writing functions
z_score <- function(mrnas){
  std <- sd(mrnas)
  m <- mean(mrnas)
  zscore <- (mrnas - m)/std
  zscore
  return(zscore)}

z_score(v3)
# [1] -1.4301939 -0.9534626  0.9534626  0.0000000  0.4767313  0.9534626

apply(d, 2, z_score)
#             cars     trucks       suvs
# [1,] -1.1804865 -0.9519946 -0.6374553
# [2,] -0.5246607 -0.1586658 -0.6374553
# [3,]  0.4590781 -0.4231087 -0.2390457
# [4,] -0.1967478 -0.1586658 -0.2390457
# [5,]  1.4428168  1.6924348  1.7530020
```

Loading R Scripts



```
### Grundlagen der Bioinformatik
### Übung 4
### Gruppe X (Person1, Person2, Person3, Person4)

### Aufgabe 1
klausur <- c(3.3, 1.7, 2.0, 4.0, 1.3, 2.0, 3.0, 2.7, 3.
mean(klausur)
# [1] 2.5
```



```
yve@yvelap:~/work/mayer/Lehre/GdBioinf17/Uebung4$ R

R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> source("uebung4.R")
> ls()
[1] "klausur"
> klausur
 [1] 3.3 1.7 2.0 4.0 1.3 2.0 3.0 2.7 3.7 2.3 1.7 2.3
>
```

Save code as .R file Load file with the source command

Assignment 4

Everything never wanted to code in R
(but have to now nonetheless)

1.0 Variables & functions (1.5 P)

Results of an examination

Mark	3.3	1.7	2.0	4.0	1.3	2.0	3.0	2.7	3.7	2.3	1.7	2.3

1. Create a variable ‘exam’ that contains the marks (0.25 P)
2. Calculate
   a. mean (0.25 P)
   b. variance and standard deviation (0.25 P)
   c. median (0.25 P)
3. Implement your **own** function to calculate the average/ mean over a vector (0.5 P)
   a. **Don’t** use the default mean() function!

2.0 Histograms & Boxplots (1P)

1. Create for the examination results
    a. A histogram (0.5 P)
    b. A boxplot (0.5P)
2. Add X and Y labels
    a. Else point deduction
3. Fill histogram and boxplot with a color of you choice

3.0 Dataframes, Correlations & Scatter plots (1.5 P)

- Dataset ***faithful*** contains information about the eruption patterns of the geysir *Old Faithful* of Yellowstone national park (USA)
- Two variable are provided
  - *Eruptions*
    - Length of eruption
  - Waiting
    - Time between eruptions

1. Load data set *faithful* (0.25 P)
   a. function *data*()
2. Calculate for both variables
   a. Variance
   b. Standard deviation (0.25 P)
   c. Average (0.25 P)
   d. Correlation (0.25P)
3. Plot
   a. Scatter-plot of both variables (0.5 P)
   b. Remember X and Y axis labels!

4.0 Vectors (1.0 P)

Two vectors

- X = c(3,7,1,10,15,8,11,2,12)
- Y = c(8,6,2,0,4,11,9,17,3)

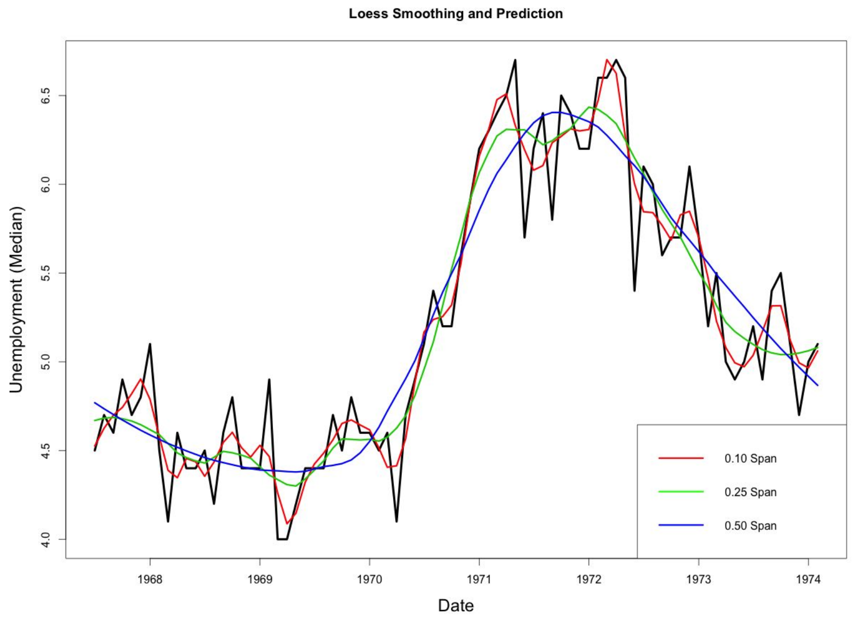
1. Create variables *X* and *Y* that contain the vectors listed above
2. Remove the last element of each vector (0.25 P)
3. Concatenate both vectors into the new vector *Z* (0.25 P)
4. Assign 9 to every number contained in Z which is greater than 9 (0.5 P)

5.0 Dataframes & Plots (1.5 P)

1. Load data set *'anorexia'* (0.25 P)
   a. Data set contained in package *MASS*, which you will have to load beforehand -> *library*()
2. Calculate the average of the variable ***Postwt*** only for those patients that underwent a **successful** therapy
   a. Note access to data set variables: ***anorexia$Postwt***
3. Weight-gain: Plot the weight of each patient before and after therapy
   a. For all cohorts together (0.25 P)
   b. And split into the three cohorts (0.5 P)
      i. Use only one plot, but color the different subgroups within the plot

6.0 Plots & Smoothing curves (3.0 P)

1. Load data set *‘airquality’*
2. Create a diagram that plots the temperature from May 1st to September 30th (1.0 P)
3. Add a smoothing curve to the plot (2.0 P)



Example of a smoothing curve

7.0 Normal-distribution (3.0 P)

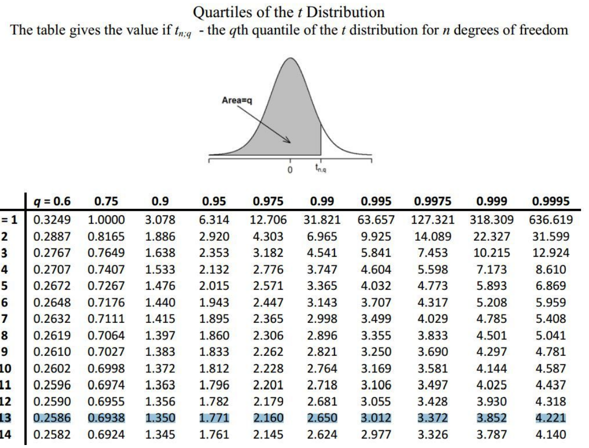
1. Create vectors of **normal-distributed** random samples
   a. Set **average** to 50
   b. Set **standard deviation** to 15
   c. Create 80 vectors for each of the following sizes (-> 240 overall)
      i. 10
      ii. 100
      iii. 1000
2. Calculate
   a. Average of each sample-size
   b. Standard deviation for each of the three sample-sizes
   c. Tip: Use sapply() and rnorm()
   d. As slow alternative to sapply() you can as well use a for-loop
3. Create a boxplot of the random-samples
   a. One box for each size -> Split and fill color

8.0 T-test (2.5 P)

You conduct a scientific experiment with 10 patients and measure **before** and **after** exposition to a new drug

1. Use a T-test statistic to decide whether the **before** and **after** vectors differ significantly
    a. Confidence level alpha of 0.01 (1.5 P)
    b. -> Show the test-statistic i.e. (1.0 P)
        i. Degrees of freedom
        ii. At what number the test would be significant
        iii. Any other parameter that you deem important

	1	2	3	4	5	6	7	8	9	10
Before	34	56	45	47	69	93	51	63	54	62
After	31	55	47	44	73	89	44	60	50	61



Quartiles of the *t* Distribution
The table gives the value if $t_{n;q}$ - the qth quantile of the t distribution for n degrees of freedom

	q **= 0.6**	**0.75**	**0.9**	**0.95**	**0.975**	**0.99**	**0.995**	**0.9975**	**0.999**	**0.9995**
n **= 1**	0.3249	1.0000	3.078	6.314	12.706	31.821	63.657	127.321	318.309	636.619
2	0.2887	0.8165	1.886	2.920	4.303	6.965	9.925	14.089	22.327	31.599
3	0.2767	0.7649	1.638	2.353	3.182	4.541	5.841	7.453	10.215	12.924
4	0.2707	0.7407	1.533	2.132	2.776	3.747	4.604	5.598	7.173	8.610
5	0.2672	0.7267	1.476	2.015	2.571	3.365	4.032	4.773	5.893	6.869
6	0.2648	0.7176	1.440	1.943	2.447	3.143	3.707	4.317	5.208	5.959
7	0.2632	0.7111	1.415	1.895	2.365	2.998	3.499	4.029	4.785	5.408
8	0.2619	0.7064	1.397	1.860	2.306	2.896	3.355	3.833	4.501	5.041
9	0.2610	0.7027	1.383	1.833	2.262	2.821	3.250	3.690	4.297	4.781
10	0.2602	0.6998	1.372	1.812	2.228	2.764	3.169	3.581	4.144	4.587
11	0.2596	0.6974	1.363	1.796	2.201	2.718	3.106	3.497	4.025	4.437
12	0.2590	0.6955	1.356	1.782	2.179	2.681	3.055	3.428	3.930	4.318
13	0.2586	0.6938	1.350	1.771	2.160	2.650	3.012	3.372	3.852	4.221
14	0.2582	0.6924	1.345	1.761	2.145	2.624	2.977	3.326	3.787	4.140

Test-statistic example

9.0 tapply & apply (3.0 P)

Utilize ***tapply*** and the data set ***'airquality'*** to

1. Calculate the average temperature per month (0.5 P)
2. Calculate the average ozone concentration per month (0.5 P)
   a. Note and deal with the fact, that ozone measurements are not available for each months

Utilize ***apply*** and the data set ***'Orange'*** to

1. Calculate the increase in diameter for each year (1.5 P)
2. Identify and print the minimal and maximal growth for each year (0.5 P)

10.0 Functions (2.0 P)

1. Write a functions that sums-up the first *n* numbers but in two different ways
    a. Iteratively (for or while loop) (0.25 P)
    b. Recursively (0.25 P)
2. Print the resulting sums (0.5 P)
    a. Make sure your functions runs on Gruenau2 with self-chosen example calculations
3. Briefly mention which implementation you think is faster and why (1.0 P)

Deadline & what we want

- Deadline for submission 27.06.2018 at 12:00 p.m.
- Submit here
  - https://box.hu-berlin.de/u/d/6529ea122a1f4ba2af86/

Zip **must** to contain

- R source code for tasks with comments
- R script has to run on Gruenau2
- PDF that contains all plots
  - See *pdf()*
  - All plots have to have X and Y axis labels
- GdBioinf_[Assignment_nr]_Gruppe_[Gruppen_nr].R