Overview Script Contents

(1) Starting a new session	Create projects directories and set your working-directory to this path. Open a new script and write a header. Clean up the memory and the global environment and load your Rdata if it was previously saved.
(2) Installing and Activating New Packages	Automatic installation and installation from source. Activating packages and its functions
(3) Getting Help	See the package descriptions inside R. Methods for further help and how to solve problems refer to the handout.
(4) Object Types and Simple Operations	(4.1) Use of Simple Mathematical Operators and object creation (4.2) Vectors (4.3) Different data types (4.3.1) numerical data (4.3.2) Character Strings and Factor (=Categorical) data (4.3.4) Logical data (4.3.5) Missing Values (4.4) Matrices and dataframes (4.1.1) Matrices (4.4.2) Dataframes (4.4.3) Side Topic: Adding new categories/ Factor Level Creation. (4.5) Lists (4.6) Useful Functions to get Informations on Objects
(5) Indexing Specific Parts of Data	(5.1) Indexing Vectors (index single and multiple values, change specific values) (5.2) Indexing Matrices and dataframes (index single and multiple values, whole column, all but one column, whole row) (5.3) Indexing Lists (objects in a list, specific values inside an object in a list) (5.4) Finding specific values (addresses) in objects
(6) Importing and Exporting Data	(6.1) CSVs (6.2) xlsx (Excel) (6.3) Import and Export R objects with lists
(7) Functions	(7.1) Functions operating on vectors (7.2) Create own functions (7.3) Apply functions several times (7.4) Nested functions (7.5) Side Topic: Using masked functions
(8) Control Structures	(8.1) loops (only for loops are demonstrated) (8.2) if else statements
(9) Plotting	(9.1) Common types of plots (9.2) Styling a plot (9.3) Plotting with ggplot2 - Example
(10) Closing a Session	Remove everything unnessery from your Global Environment and save all the objects for reuse in the next session.

```r
# ---------- header ----------
## Introduction to R programming
# Author: Johannes Schielein
# Date: 2016-04-13
## Contents of the script:
# (1) Starting a new session
# (2) Installing and Activating New Packages
# (3) Getting Help
# (4) Object Types and Simple Operations
# (5) Indexing Specific Parts of Data
# (6) Importing, Exporting
# (7) Functions
# (8) Control Structures (loops, if-else)
# (9) Plotting
# (10) Closing a Session
# Additional Remarks and Data Sources

#---------- (1) Starting a new session ----------

## create a new working directory for R and your R projects (do this only once!)
# Note: Generally avoid using whitespace and special characters, use only lowercase letters!
# You can do this step as well outside R

dir.create("/home/jschielein/R/projects/intro-classes")
dir.create("/home/jschielein/R/projects/intro-classes/input")
dir.create("/home/jschielein/R/projects/intro-classes/output")
dir.create("/home/jschielein/R/projects/intro-classes/scripts")
dir.create("/home/jschielein/R/projects/intro-classes/rdata")

## set your workingspace to the new directory
setwd("/home/jschielein/R/projects/intro-classes/")
# check if it worked
getwd()

## clean up your environment and memory
rm(list=ls()) # remove eventuall remaining objects from the global environment
gc(reset=TRUE) # free the memory from eventuall load

## load your workspace (Only if previously saved at the end of the session)
load("rdata/lookup-script-2016.Rdata")

#---------- (2) Installing and Activating New Packages ----------
## installing a new package (only once). You can also do this in R-Studio under the tab "Packages".
install.packages("ggplot2")
install.packages("xlsx")
install.packages("highlight")

## installing packages from source (if they can't be installed due to compatability issues)
# First locate and download the package source on http://cran.r-project.org/web/packages/
# For windows use the Windows binaries and locate the download link under "r-release" (ZIP file)
install.packages("/home/jschielein/R/packages-source/raster_2.3-24.zip",repos=NULL, type="source")

# activate the package and its functions for the current session (doe this every session that you n
library("ggplot2")
library("xlsx") # known java issues on windows 7 can be solved by installing and activating  xlsx,
library("highlight")

#---------- (3) Getting Help ----------
# show help for any function
help(boxplot)
# alternativly
?boxplot

# note: for further tipps on how to solve problems refer to the handout of the class!
```

```
#---------- (4) Object Types and Simple Operations ----------
## (4.1) Use of Simple Mathematical Operators and object creation
3*(3+1)-6/3
# create a new object from this opeation
result.example<-3*(3+1)-6/3

# show the result it in the command prompt
result.example

# we can further operate on our object
result.example + 4


### (4.2) Vectors
# are a combination of observations that can be concannated e.g. like this
vector.example<-c(1,2,3,4,5)

# show it
vector.example

# R operates on vectors
vector.example*5

# matrices are a combination of several vectors (as columns) -> details see below

# (4.3) Different data types
# (4.3.1) numerical data
# integer (full rounded numbers)
integer.vector.example<-1:5

# double precision numbers
double.vector.example<-c(1.32,2.54,3.72,4.89,5.01)

# How numeric numbers are stored influeces processing speed.

## (4.3.2) Character Strings and Factor (=Categorical) data
# simple character string
string.example<-"Fischer Fritz fischt frische Fische" # thats a german tongue twister by the way!

# several strings in a vector
string.vector.example<-c("R is great! I love it",
                         "R seems like chinese to mee!",
                         "R? Isn't that the letter before S?",
                         "I want an ice-cream!!",
                         "I want an ice-cream!!",
                         "I use R every day. I even write my emails in R!")

# show it in the console
string.vector.example

# convert the string data to factor data/categorical data
factor.example<-as.factor(string.vector.example)
# show it
factor.example

# show the unique factor levels/ categories
levels(factor.example)

# Important: if you work with categorical data also see the sidetopic 4.5.1 below

## (4.3.4) Logical data
logical.vector.example<-double.vector.example>2 # which elements of the numeric vector are greater
# show it
logical.vector.example
# note: ">" is the logical operator for greather than. Other important ones are: ,>=,<,<=,=,!=,&,|
```

```r
## (4.3.5) Missing Values
# are denominated by NA
missings.example<-c(1,2,3,NA,5,6,7,NA,9,10)

# if we want to operate on the string we might remove the missings in some cases for the operation
mean(missings.example) # may not be the desired result
mean(missings.example,na.rm=TRUE) # works!

## (4.4) Matrices and dataframes
## (4.4.1) Matrices
# first we create a vector ranging from 1 to 12
vector.2<-1:12

# from this we can create a simple matrix where we split up the vector in four columns
matrix.example<-matrix(vector.2, ncol = 4)
# have a look
matrix.example

# we can assign column names to the matrix (e.g. variable names)
colnames(matrix.example)<-c("variable.a","variable.b","variable.c","variable.d")
# Note: matrices work only with numeric values.
# if you need to work with other data types (e.g. categorical data) use dataframes

# we can assign rownames as well (e.g. observations)
rownames(matrix.example)<-c("Peter","Gabriela","Fritz")

# show it
matrix.example

## (4.4.2) Dataframes
# we convert our matrix to a dataframe like this
dataframe.example<-as.data.frame(matrix.example)

# another way to create a data.frame from scratch is to construct vectors and combine them with the
dataframe.example.scratch<-data.frame(
  variable.1=1:10,
  variable.2=c("a","b","c","d","e","f","g","h","i","j"),
  variable.3=rnorm(10)) # rnorm creates randomized normally distributed numbers between 0 and 1, in

# show it
dataframe.example.scratch

# add a new COLUMN the easy way
dataframe.example.scratch$variable.4<-1:10

# show it
dataframe.example.scratch

# add a new COLUMN with cbind (works for multiple columns from e.g. two different matrices)
dataframe.example.scratch<-cbind(
  dataframe.example.scratch,
  variable.5=11:20) # bind a column do the dataframe.

# show it
dataframe.example.scratch

# add a new ROW to the dataframe
dataframe.example.scratch<-rbind(
  dataframe.example.scratch,
  c(11,"a",0.5,11,21)
)

# show it
dataframe.example.scratch
```

```r
# Rotate/transpose a dataframe
dataframe.example.transposed<-t(dataframe.example.scratch)

# have a look
dataframe.example.transposed

## (4.4.3) Side Topic: Adding new categories/ Factor Level Creation.
# lets say we want to add a new category inserting data like this:
dataframe.example.scratch<-rbind(
  dataframe.example.scratch,
  c(11,"xyz",0.5,11,21)
)
# doesnt work instead of the new observation xyz a missing value was introduced because the categor
# show the unique factor levels/ categories
levels(dataframe.example.scratch$variable.2)
# add the new category as a new factor level
levels(dataframe.example.scratch$variable.2)<-c(
  levels(dataframe.example.scratch$variable.2),
  "xyz")
# add a new entry to the vector that corresponds to this category
dataframe.example.scratch<-rbind(
  dataframe.example.scratch,
  c(11,"xyz",0.5,11,21)
)
# now it worked
dataframe.example.scratch

## (4.5) Lists
# serve like archives and can help us to store various objects
list.example<-list(double.vector.example, vector.2)

#show it
list.example

# unlist: if we want to use both elements of the list as a single vector we have to unlist it
unlist(list.example)

## (4.6) Useful Functions to get Informations on Objects
str(dataframe.example) # gives information on datatype and other
summary(dataframe.example) # gives some summary statistics
length(vector.2) # shows the total length of a vector
head(dataframe.example) # shows the first rows of a dataframe
View(dataframe.example) # shows data in a seperate window. only Rstudio!

# ---------- (5) Indexing Specific Parts of Data ----------
## We use the built-in datasets to show how indexing works
# Mean annual flow of the  Nile river in cubic meters per second measure at Ashwan from 1871-1970
nile.flow<-as.vector(Nile)
# check out
nile.flow
# Dataframe on height, age and seed of Loblolly Pine Trees
pinetrees<-Loblolly
#check out
head(pinetrees)

## (5.1) Indexing Vectors
# single value
nile.flow[3]
# Change the value e.g. to correct a transcription
nile.flow[3]<-1300

# several values
nile.flow[c(1,3,5)]
# or if they are in sequence
```

```r
nile.flow[3:5]

## (5.2) indexing matrices and dataframes
# single value e.g. second row, first column
pinetrees[2,1] # matrix notation: first the row than the column

# several values
pinetrees[2:5,1]

# whole column
pinetrees[,1] #note: nothing means all. Seems strange but is like that

# Excluding a column
pinetrees[,-1] #note: nothing means all. Seems strange but is like that

# whole row
pinetrees[1,]

# if you have a dataframe you can index columns as well with the $ sign
pinetrees$height # will return the whole column as a vector

## (5.3) indexing lists
list.index.example<-list(pinetrees,nile.flow)
# requires two brackets [[]]
list.index.example[[2]] # returns the whole vector

# indexing something inside this vector is straightforward
list.index.example[[2]][3] # just put one index after another

## (5.4) Finding specific values (addresses) and logical subscripts
# finding the maximum and minimum
min(nile.flow)
max(nile.flow)

# finding values within a range with the which function
which(nile.flow>800 & nile.flow<1000) # returns the index of the observation in the vector

# do the same but return the value of the observation
nile.flow[nile.flow>800 & nile.flow<1000] # returns the values of the observation

## use indexing to subset dataframes
# example 1: return all rows where the pinetree seed is 301
pinetrees[pinetrees$Seed==301,]

# example 2: return all rows where the pinetree seed is not 301
pinetrees[pinetrees$Seed!=301,]

# example 3: return only the first column (height) where the pinetree seed is not 301
pinetrees[pinetrees$Seed!=301,1]

# example 4: combining return all rows where the pinetree seed is either 301 or 303
pinetrees[pinetrees$Seed==301 | pinetrees$Seed==303,]

# example 5: return all rows of pinetrees where the Seed is equal 301 and the age is equal to 5
pinetrees[pinetrees$Seed==301 & pinetrees$age==5,]

# (5.6) Side Topic: subsetting dataframes using the subset function
# subset based on two criteria, only returning one column
subset(pinetrees,Seed==301 & age==5, select = height)

# ---------- (6) Importing, Exporting  ----------
## (6.1) CSVs
# write csv
write.table(dataframe.example,"output/dataframe.example.csv",sep=",")
```

```r
## read csv
dataframe.example.reimported<-read.table("output/dataframe.example.csv", sep=",",header = TRUE)
# note since the first row of the csv contains the variable names we set header = TRUE
View(dataframe.example.reimported) # have a look

## (6.2) xlsx
# write out some data to excel format
write.xlsx(dataframe.example.scratch,
           "output/dataframe.example.scratch.xlsx")

# import the data again
dataframe.example.scratch.reimported<-read.xlsx("output/dataframe.example.scratch.xlsx",
                                                sheetIndex = 1)

## (6.3) SPSS
# activate the library
library("foreign")

## Angry Birds example
# import the Angry Birds data-set
honesty.lab<-read.spss("input/honesty.lab.sav",to.data.frame = TRUE)

## (6.4) Import and Export R objects with lists
# save a single object
save(dataframe.example.scratch,
     file= "output/dataframe.example.scratch.Rdata")

# reimport it again
load(file = "output/dataframe.example.scratch.Rdata")

## Save several objects (and functions) with lists
# Save everything you need first in a list
my.list<-list(dataframe.example.scratch,
                dataframe.example,
                vector.2)

# export the list with saveRDS
saveRDS(my.list,file="output/my.list.Rdata")

# reimport the list
my.list.reimported<-readRDS("output/my.list.Rdata")

# save all objects from your environment (do this at the end of your session)
save.image("rdata/lookup-script-2016.Rdata")

# ---------- (7) Functions ----------
## (7.1) Functions operating on vectors
# very simple
mean(dataframe.example$variable.b)

## (7.2) Create custom functions
# create with two parameters
function.example<-function(a,b){
  tmp.data<-mean(a)/mean(b)
  return(tmp.data)}

# usage
function.example(pinetrees$height,pinetrees$age)

## (7.3) Apply functions several times
# on different columns in a matrix/dataframe
apply(dataframe.example, 2 , summary) # example with the summary function

# on different rows
apply(dataframe.example, 1 , mean) # example with the mean
```

```r
## (7.4) nested functions (calculates from inside to outside)
sqrt(mean(dataframe.example$variable.a))

# you can go on as you like
round(sqrt(mean(dataframe.example$variable.a)),digits=2)

## (7.5) Side Topic: Using masked functions
# sometimes functions from different packages have the same name so they are masked from the global
# In order to use them you will have to write: package::function e.g.
base::sum(1:15)

# ---------- (8) Control Structures ----------
## (8.1) loops
# for loops serve for iterations.
for(i in 1:10){print(i+5)}

# Storing loop results
# if we want to store the results of a loop in a new object,
# we will first have to create an empty object to receive the results and then use a clever indexin

# create empty vector
loop.results<-vector()

# loop and store results
for (i in 1:10){loop.results[i]<-i+5}

# show it
loop.results

# General note: avoid loops, if possible because they are slow. have a lock at the plyr package for
# other types of loops (e.g. while loops) see e.g. here: https://blog.udemy.com/r-tutorial/

## (8.2) if else statements
# use it to define multiple outcomes of a test
summary(pinetrees)
ifelse(pinetrees$height>50,
       "big",
       "small")

# you can nest it as well to check for multiple conditions
ifelse(pinetrees$height<20,
       "small",
       ifelse(
         pinetrees$height>=20 &
           pinetrees$height<50,
         "medium",
         "big"
       ))
# you can use it to create a new column
pinetrees$height.class<-as.factor(
  ifelse(pinetrees$height<20,
         "small",
         ifelse(
           pinetrees$height>=20 &
             pinetrees$height<50,
           "medium",
           "big"
         )))

# ---------- (9) Plotting ----------
# (9.1) Common types of plots
# barplot
barplot(pinetrees$height)
# histogram
```

```
hist(pinetrees$height)
# boxplot
boxplot(pinetrees$height)
# scatterplot
plot(pinetrees$age,
     pinetrees$height)
# scatterplot with smoothed curve
scatter.smooth(pinetrees$age,
               pinetrees$height)

# (9.2) Styling a plot
scatter.smooth(pinetrees$age,
               pinetrees$height,
               xlab = "Age of pinetrees",
               ylab = "Tree height in meters",
               main = "Loblolly Pinetrees growth and age",
               col = "blue"
              )

# for more information on how to style your plot use the help function of each plot

# (9.3) Plotting with ggplot2 - Example
# NOTE: ggplot two is more complex to learn but later far more powerfull to make sophisticated grap
# easy example: scatterplot with smoothed line
xy.plot.example<-ggplot(data=pinetrees,
                        aes_string(x="age", y="height"))+ # defines the basics
  geom_point()+ # add a layer with the points
  geom_smooth() # add a layer with the smoothed curve

#plot
xy.plot.example

# styled ggplot
xy.plot.example<-ggplot(data=pinetrees,
                        aes_string(x="age", y="height"))+ # defines the basics
  geom_point(color="darkgreen", # layer specific styles go into the layer itself
             size=5)+
  geom_smooth()+
  ggtitle("Loblolly Pinetrees growth and age")+
  ylab("Tree height in meters")+
  xlab("Age of pinetrees")+
  theme( # general styles relating the plot go into this theme box
    plot.title=element_text(color="red", size=16, vjust=2), # adjust the title
    axis.title.x=element_text(color="blue", size=11, vjust=0.5), # adjust the x-axis title
    axis.title.y=element_text(color="#A86BA7", size=11, vjust=1), # adjust the y-axis title with a
    panel.border = element_rect(colour = "black",fill=NA) # add a boarder to the plot area
  )

#plot
xy.plot.example

# ---------- (10) Closing a Session ---------
# remove all unneeded objects from your environment e.g. the first created vector from above
rm(result.example) # if you want to remove several objects seperate them with commas.

# save your environment
save.image("rdata/lookup-script-2016.Rdata")
# next time you open your project load the data by: load("introduction.data.1.Rdata")
# Insert the load command at the beginning of your script


## exporting your script as a html file for printing
library("highlight")
highlight( "scripts/lookup-script-2016.R",
           output = "scripts/lookup-script-2016.html",
```

```
            renderer = renderer_html(
              stylesheet = "/home/jschielein/R/costum_functions/css/costum_1.css") # only needed if
)

# ---------- Additional Remarks ----------

## Important packages

# foreign: to import and export several file formats. Comes with basic installation. Just call libr
# ggplot2: Advanced plotting and mapping tool. Package needs to be installed.
# plyr: Tools for splitting up datasets and applying functions on several parts of it (automation)
# and combining the parts again. Package needs to be installed
# rgdal: to use gdal libraries to work with spatial data. Package needs to be installed
# raster: most extensive data-set to work with raster data. Package needs to be installed
# reshape2: For reshaping all kinds of tables e.g. from wide to longformat.

## Other important topics you might consider studing more in depth:
# merge, union and intersect data
# rank, sort and order dataframes
# pattern matching in string with grep
```