Introduction to R

Daniel Münch

May 12, 2014
v1.1.0

1 Links

These are some useful links for learning R or looking up stuff. While learning R the Internet is your friend, so when you have a problem try googling it, there are always people having the same issue and at least one will have started a discussion in some forum already.

1.1 Getting R

- CRAN, here you find the R downloads
  http://cran.r-project.org/
- RStudio
  http://rstudio.org/

1.2 R documentation

- An introduction to R, from CRAN
  http://cran.r-project.org/doc/manuals/R-intro.pdf
- The R Reference Card
  http://cran.r-project.org/doc/contrib/Short-refcard.pdf
- The R help files (you also reach them when typing `?function`) in R
  http://stat.ethz.ch/R-manual/R-devel/doc/html/
- The R wikibook, a huge R guide (German)
  http://de.wikibooks.org/wiki/GNU_R
- A R lecture including exercises (German)
  http://www.uni-leipzig.de/~zuber/teaching/ws09/r-kurs/index.html

1.3 R tipps & tricks & getting help

- The R Cookbook, a very good source for tips&tricks, providing a lot of hints for ggplot2.
  http://www.cookbook-r.com/

- Quick-R
  http://www.statmethods.net/

- getting help
  http://stackoverflow.com/questions/tagged/r

1.4 Plotting with ggplot2

- The ggplot2 homepage
  http://ggplot2.org

- Documentation for the latest `ggplot2` version
  http://docs.ggplot2.org/

- Hadley Wickham's introductory R scripts
  http://had.co.nz/stat480/r/

1.5 More links

- R styleguide, a guide on how to write R code
  http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html

- ggobi, a system for interactively viewing and modifying multivariate data, also available as R package `rggobi` from CRAN
  http://www.ggobi.org/

2 Code

This is the code we used in the lecture.

```
# Part I, Basics ------------------------------------------------------------

# calculating stuff: + - / * <-

  1-5
  2*4
  8+.2
  9^2
  sqrt(25)

  a <- 10                    # assign values to variables
  b =  20                    #  - same effect
  assign("c",30)             #  - same effect

  b^a                        # calculate with variables
  d <- b^c                   # assign results of calculations to variables

  c(1,2,3)                   # numerical vector
  c("a", "b", "c")           # create a character vector
  vector <- c(2, "b", "c")   # mixed vectors get transformed

  vector[1]

  v <- c(a,b,c)              # create vector from variables
  v/2
  v[1] / 3
  v <- c(v,v/2,v-10)

  ls()                       # display workspace content
  rm(v)                      # remove object; rm(list=ls()) for removing everything
  v                          # Fehler: Objekt 'v' nicht gefunden
  v <- c(a,b,c)


# base functions: mean, max, min, median, ...

  length(v)
  summary(v)
  mean(v)
  max(v)
  min(v)
  median(v)
  quantile(v)
  rep(v,7)
  sort(v, decreasing=TRUE)
  order(v, decreasing=TRUE)

  fun1 <- function(x){x*10}          # a simple function
    fun1(5)
    fun1(v)

```

```
fun2 <- function(x=1,y=5)           # more complex function, z will not appear in
    wokspace
  {
  z <- x * y
  w <- z*2
  return(w)
  }

  fun2(x=v,y=2)                     # write "fun2(" and hit TAB to get a list of
      possible options


# objects

  # vectors (character, numeric, factor)

  cv <- "this is a test text"
  cv <- c("a", "b", "c")
  cv <- rep(cv,10)

  fv <- factor(cv)                  # transform character vector into factor
  fv
  levels(fv)                        # show levels/factors


  # data.frames
  # - most used data format in R (at least for me),
  # - can take all kinds of data,
  # - think of it as a spreadsheet (like in excel)

  nv <- 1:length(cv)

  df <- data.frame(fv, cv, nv)

  names(df)
  colnames(df)
  rownames(df)
  str(df)

  df <- data.frame("factor" = fv, "character" = cv, "numeric" = nv, stringsAsFactors =
      FALSE)   # give custom names, don't convert strings to factors

  str(df)
  head(df)                  # get only first rows of a dtata.frame
  summary(df)                   # get structure of an object
  length(df)
  dim(df)

  df$character              # get column by name, safest way if you are not 100% sure
      about the column order
  df[,2]                    # same as above, indexed by number
  df[2,]                    # show 2nd row
  df[8,2]                   # show content of row 8, column 2
  df[3:8, c(1,3)]           # show row 3 to 8 of columns 1 and 3
```

```
    df[3:8,"factor"]          # show


#     # matrix
#
#       matrix(v)
#       matrix(v, nrow=9, ncol=3)
#
#
#     # lists
#
#       li <- list(a,b,v)
#       li <- list("a"=a, "b"=b,"v"=v)
#
#       li[[2]]
#       li$v
#       li$v[3]


# exporting & importing data

    getwd()
    setwd()  # !! if you are on windows and want to reference subfolders, remember to use "/
        " instead of "\"

    write.csv(df, "df.csv", row.names=F)

    df2 <- read.csv("df.csv")


# installing packages
    install.packages("ggplot2")
    library(ggplot2)

    head(diamonds)
    summary(diamonds)

    # logical operators
    #     a == b
    #     a != b
    #     a <  b
    #     a >  a
    #     a >= a
    #

    subset(diamonds, cut == "Premium")
    subset(diamonds, cut == "Premium" & carat >= 4)

    #- take a look at the diamonds dataset: head, summary, str, dim...
    head(diamonds)
    dim(diamonds)
    summary(diamonds)

    #- create a subset of "diamonds" named "onecarat" containing all diamonds that:
```

```
#  - weight between 1 and 2 carat
#  - cut is "Good"
onecarat <- subset(diamonds, carat >= 1 & carat <= 2 & cut == "Good")
dim(onecarat)

#- create two new data frames as subset of "onecarat", called "D1G" and "J1G" containing
      diamonds of color "D" or "J"
D1G <- subset(onecarat, color=="D")
J1G <- subset(onecarat, color=="J")
dim(D1G); dim(J1G)


# Part II, Testing ------------------------------------------------------

# parametric test
t.test(D1G$price,J1G$price)
t.test(D1G$price,J1G$price, alternative="greater", var.equal=T)
t <- t.test(D1G$price,J1G$price,alternative="greater", var.equal=T)
names(t)

#  not normally distributed
wilcox.test(D1G$price, J1G$price)
wilcox.test(D1G$price, J1G$price,alternative="greater")
wil <- wilcox.test(D1G$price, J1G$price,alternative="greater")
names(wil)

# check normal distribution
shapiro.test(D1G$price)


# Part II, Plotting -----------------------------------------------------

# ggplot2

  # http://ggplot2.org/
  # http://docs.ggplot2.org/      # ggplot2 documentation

  boxplot(D1G$price, J1G$price)

  qplot(x=color, y=price, data=onecarat)
  qplot(x=color, y=price, data=onecarat, geom="boxplot")

  ggplot(aes(y=price, x=color), data=onecarat) + geom_boxplot()
  p <- ggplot(aes(y=price, x=color), data=onecarat)
  p
  p + geom_boxplot()
  p + geom_boxplot() +  geom_jitter(alpha=.2, color="red")
  p + geom_boxplot() +  geom_jitter(alpha=.5, aes(color=color))

  p2 <- ggplot(data=subset(diamonds, cut == "Good" & color %in% c("D", "J") & carat >= 1 &
      carat <= 2), aes(x=color, y=price))
  p2 + geom_boxplot()
  p2 + geom_jitter()
  p2 + geom_boxplot() + geom_jitter()
```

```
p2 + geom_violin(aes(fill=color))
p2 + geom_violin(aes(fill=color), trim=T) + geom_boxplot(alpha=.4)
p2 + geom_violin(aes(fill=color), trim=T) + geom_boxplot(alpha=.4) + geom_jitter()
p2 + geom_violin(aes(fill=color), trim=T) + geom_boxplot(alpha=.4) + geom_jitter(color="green")
p2 + geom_violin(aes(fill=color), trim=T) + geom_boxplot(alpha=.4) + geom_jitter(alpha=.6)
p2 + geom_violin(aes(fill=color), trim=T) + geom_boxplot(alpha=.4) + geom_jitter(alpha=.6, aes(size=carat))


p3 <- ggplot(data=subset(diamonds, cut == "Good"), aes(x=carat, y=price))
p3 + geom_point()
p3 + geom_point(aes(color=clarity))
p3 + geom_point(aes(color=clarity)) + geom_smooth(method="lm")
p3 + geom_point(aes(color=clarity)) + geom_smooth(method="lm", aes(group=clarity))
p3 + geom_point(aes(color=clarity)) + geom_smooth(method="lm", aes(group=clarity, color=clarity))

p3 + geom_point(aes(color=clarity)) + geom_smooth(method="lm", aes(group=clarity, color=clarity)) + facet_wrap( ~ clarity)
p3 + geom_point(aes(color=clarity)) + geom_smooth(method="lm", aes(group=clarity, color=clarity)) + facet_grid(color ~ clarity)


# Loops ------------------------------------------------------------------

if (a == b ) print("richtig!") else print("falsch!")

if (a > 1)
{
  x <- a + b
  print(x)
} else
  {
  x <- a - b
  print(x)
  }

x <- c()
for (i in 1:dim(df)[1])
{
  x <- c(x, df$numeric[i]*i)
}


# Exercise (if time left) -------------------------------------------------

# use the dataset "mtcars" to test whether cars with 4 cylinders ("cyl") can drive a
# longer distance ("mpg" - miles per gallon) than cars with 8 cylinders
# try to plot this data in a useful way
#
# ! the "cyl" column is a numeric vector, not a factor, try to convert it when plotting
```