Grundlagen der Bioinformatik
Übung 5
Einführung in R

Ulf Leser, Yvonne Mayer

Introduction to R

Einführung in R

- Voraussetzung: funktionsfähige R Installation (http://cran.r-project.org/)
- auf gruenau2 existiert eine R Installation (Login via ssh oder Putty)
- Einführung in R: http://cran.r-project.org/doc/manuals/R-intro.pdf
- Übersicht häufiger Befehle (Reference Card): 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

```
# 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 ...
```

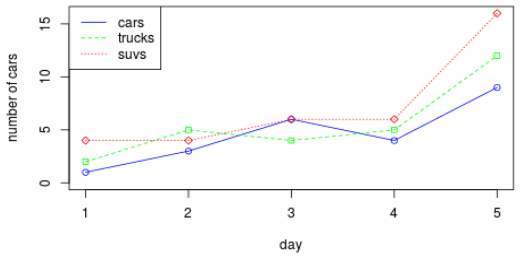
Matrices and Data Frames

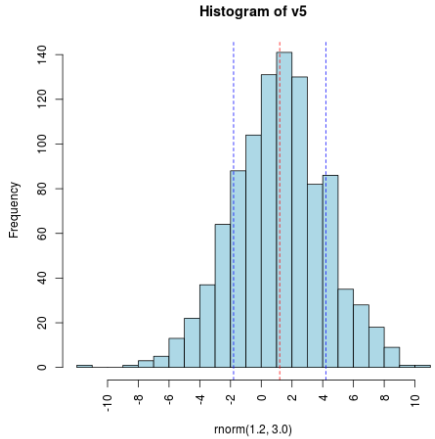
```
# matrices
mat <- matrix(data=c(9,2,3,4,5,6), ncol=3)
mat[1,2]                          # [1] 3
mat[2,]                           # [1] 2 4 6
mean(mat)                         # [1] 4.833333
mean(mat[1,])                     # [1] 5.666667

# 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, ...)

Plots

```
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, …
- more parameters: cex, cex.axis, lwd, pch, mfrow, mar, ...

Programming – if, for, apply

```r
# 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, ...

Programming – writing 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
```

Assignment 5: Introduction to R

(Tasks mostly taken from
http://www.staff.uni-marburg.de/~gruener/lehre/r.w05/welcome.html)

(1) Variablen, Funktionen (1.5P)

In der folgenden Tabelle sind die Ergebnisse einer Klausur wiedergegeben:

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

(a) Erstellen Sie eine Variable *klausur* mit den Ergebnissen (0.25P)

(b) Berechnenen Sie den Mittelwert (0.25P)

(c) Berechnen Sie die Standardabweichung (0.25P)

(d) Berechnen Sie den Median (0.25P)

(e) Berechnen Sie den Mittelwert ohne die eingebaute Funktion *mean()* (0.5P)

(2) Histogramm, Boxplot (1P)

(a) Fertigen Sie ein Histogramm für die Ergebnisse der Klausur an (0.5P)

(b) Fertigen Sie einen Boxplot für die Ergebnisse der Klausur an (0.5P)

Denken Sie an x-Achsen und y-Achsen Beschriftung! Färben Sie das Histogram/den Boxplot mit einer beliebigen Farbe.

(3) Dataframes, Korrelation, Streudiagramm (1.5P)

Der eingebaute Datensatz 'faithful' enthält Daten über aufeinanderfolgende Eruptionen des Geysirs Old Faithful im Yellowstone-Nationalpark (USA). Im Datensatz sind zwei Variablen enthalten: *eruptions* (Dauer des Ausbruchs) und *waiting* (Zeit bis zum Ausbruch).

(a) Laden sie den Datensatz 'faithful' (Funktion: data()) (0.25P)

(b) Berechnen Sie Standardabweichung und Mittelwert für beide Variablen (0.5P)

(c) Berechnen Sie die Korrelation der beiden Variablen (0.25P)

(d) Fertigen Sie ein Streudiagramm der beiden Variablen an. Denken Sie an x-Achsen und y-Achsen Beschriftung! (0.5P)

(4) Vektoren (1P)

Gegeben seien die Vektoren
x=(3,7,1,10,15,8,11,2,12) und y=(8,6,2,0,4,11,9,17,3)

(a) Verketten Sie die beiden Vektoren zu einem neuen Vektor z, nachdem Sie das letzte Element von x und das letzte Element von y gestrichen haben (0.5P)

(b) Weisen Sie allen Werten von z, die größer als 9 sind, den Wert 9 zu (0.5P)

(5) Dataframes, Plots (1.5P)

(a) Machen Sie sich den Anorexia-Datensatz aus der MASS-Library verfügbar (0.25P)

(b) Berechnen Sie den Mittelwert der Variable 'Postwt' für die Patientinnen mit erfolgreicher Therapie (Gewichtszunahme) (0.25P)

(c) Plotten Sie das Gewicht vor der Behandlung gegen jenes nach der Behandlung. Die drei Behandlungsgruppen sollen in dem Plot unterscheidbar sein (1P)

(6) Plots, Glättungskurven (3P)

(a) Zeichnen Sie ein Diagramm für den Verlauf der Temperatur zwischen dem 1. Mai und dem 30. September (Datensatz 'airquality') (1P)

(b) Zeichnen Sie eine Glättungskurve in das Diagramm ein (2P)

(7) Normalverteilungen (3P)

Erzeugen Sie je 80 Zufallsstichproben mit einem Stichprobenumfang von

(a) 10
(b) 100
(c) 1000

aus einer Normalverteilung mit einem Mittelwert von 50 und einer Standardabweichung von 15. Berechnen Sie die Mittelwerte der 80 Zufallsstichproben und fertigen Sie einen Boxplot an.

Tipp: Nützlich sind hier die Funktionen sapply() und rnorm(). Alternativ zu sapply kann man auch eine for-Schleife verwenden.

(8) t-Test (2.5P)

10 Versuchspersonen (VP) wurden vor und nach einem Trainingsprogramm einem Test unterzogen. Man erhält folgende Testergebnisse:

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

Testen Sie, unter der Voraussetzung, dass die Testwerte normalverteilt sind, mit einem Signifikanzniveau von alpha = 0.01, ob die Testleistungen vorher/nachher nur zufällig voneinander abweichen. Suchen Sie dafür die Funktion für den t-Test heraus.

(9) tapply (1P)

Nutzen Sie tapply um

(a) die durchschnittliche Temperatur pro Monat zu berechnen (Datensatz 'airquality') (0.5P)

(b) den durchschnittlichen Ozonwert pro Monat zu berechnen (Datensatz 'airquality'). Beachten Sie, dass nicht für jeden Tag Ozonwerte zur Verfügung stehen (0.5P)

(10) apply (2P)

Laden Sie den Datensatz 'Orange'
(Wachstum von Orangenbäumen).

(a) Nutzen Sie die Funktion apply() um pro Eintrag (Zeile) die Zunahme des Umfangs pro Jahr zu berechnen (1.5P)

(b) Was ist der maximale und was der minimale Umfangwachstum pro Jahr? (0.5P)

(11) Funktionen (2P)

Schreiben Sie eine Funktion, die die ersten n Zahlen aufaddiert auf zwei verschiedene Arten:

(a) iterativ (mit for oder while Schleife) (1P)

(b) rekursiv (1P)

Abgabe

- Abgabe bis Mittwoch den 29.06.2016 um 23:59 Uhr
- Abgabe per Email an: yvonne.lichtblau@informatik.hu-berlin.de (gerne auch Fragen zur Übung per Email)
  - R Skript
  - PDF Datei mit allen erzeugten Plots (z.B. mit Funktionen pdf() oder png()) und ggf. den Lösungen zu den Aufgaben. Alle Plots müssen eine x- und y-Achsen Beschriftung aufweisen.