Solutions to Exercise Series 1

1. Vectors

```
x <- c(5, 2, 1, 4)
xx <- c(1, 10, 15, 18)
y <- rep(1, 5)
z <- c(TRUE, FALSE, TRUE, TRUE)
w <- c("Marie", "Betty", "Peter")
```

a)
```
sum(x)
## [1] 12
range(x)
## [1] 1 5
length(y)
## [1] 5
sum(y)
## [1] 5
```

b)
```
c(x, y, 13)
##  [1]  5  2  1  4  1  1  1  1  1 13
```

c)
```
xx - x
## [1] -4  8 14 14
c(x, 12) * y
## [1]  5  2  1  4 12
1:6 + 1
## [1] 2 3 4 5 6 7
1:9 + 1:2
## Warning in 1:9 + 1:2: longer object length is not a multiple of shorter
object length
## [1]  2  4  4  6  6  8  8 10 10
```

For the last calculation R gives a warning.

d)
```
x <= 2
## [1] FALSE  TRUE  TRUE FALSE
x <= 2 & z
## [1] FALSE FALSE  TRUE FALSE
```

e)
```
substring(w, 2, 4)
## [1] "ari" "ett" "ete"
paste(substring(w, 1, 2), substring(w, 5, 5), sep = "...")
## [1] "Ma...e" "Be...y" "Pe...r"
```

f)
```
cbind(x, xx)
##      x xx
## [1,] 5  1
## [2,] 2 10
## [3,] 1 15
## [4,] 4 18
cbind(2, 6:1, rep(c(3, 1, 4), 2), seq(1.1, 1.6, by = 0.1))
##      [,1] [,2] [,3] [,4]
## [1,]    2    6    3  1.1
## [2,]    2    5    1  1.2
## [3,]    2    4    4  1.3
## [4,]    2    3    3  1.4
## [5,]    2    2    1  1.5
## [6,]    2    1    4  1.6
```

2. Sequences of numbers

a)
```
v1 <- 1:9
# or
v1 <- seq(1,9)
# or
v1 <- seq(from = 1, to = 9, by =1 )
# or
v1 <- seq(from = 1, length = 9, by = 1)
```

b)
```
v2 <- rep(c("m","w"), 5)
```

c)
```
v3 <- rep(1:4, 3)
```

d)
```
v4 <- rep(4:1, each = 3)
```

e)
```
v5 <- rep(1:5, 1:5)
```

f)
```
v6 <- rep(seq(1, 11, by = 2), each = 2)
```

3. Matrices and data frames

a) First matrix:

```
M0 <- matrix(c(1:5, 101:105, 201:205, 301:305), nrow = 5, ncol = 4)
```

Second matrix:

```
M1 <- diag(5, 3)
```

b)
```
d.f <- data.frame(Namen = c("Bob", "Alice", "Kim", "Julia", "Robert"),
                  Age = c(27, 34, 21, 25, 29),
                  Blind = c(TRUE, FALSE, FALSE, TRUE, TRUE))
```

c) We examine the first matrix:

```
class(M0)  # Object Type
## [1] "matrix"
```

```
dim(M0)    # Number of Rows and Columns
## [1] 5 4
str(M0) # Structure
##  int [1:5, 1:4] 1 2 3 4 5 101 102 103 104 105 ...
summary(M0) # Summary per column
##        V1           V2            V3            V4
##  Min.   :1   Min.   :101   Min.   :201   Min.   :301
##  1st Qu.:2   1st Qu.:102   1st Qu.:202   1st Qu.:302
##  Median :3   Median :103   Median :203   Median :303
##  Mean   :3   Mean   :103   Mean   :203   Mean   :303
##  3rd Qu.:4   3rd Qu.:104   3rd Qu.:204   3rd Qu.:304
##  Max.   :5   Max.   :105   Max.   :205   Max.   :305
```

4. **a)** See exercise sheet.

b)
```
vog.df <- d.vogel[, 3:4]
```

Alternatives:

```
vog.df <- d.vogel[, c("Feld.Nr", "Anzahl")]
vog.df <- d.vogel[, -c(1, 2, 5, 6)]
```

```
vog.mean <- mean(d.vogel[, "Anzahl"])
```

On average 22 birds were counted.

c) Create a data frame with the information on meadow 1413:

```
ok <- d.vogel[, "Feld.Nr"] == 1413
df <- d.vogel[ok, ]
```

d)
```
anz <- d.vogel[, "Anzahl"][ok]
anz
## [1] 15 44 41  2
```

Alternatives:

```
anz <- d.vogel[ok, "Anzahl"]
anz <- df["Anzahl"]
anz <- d.vogel[, "Anzahl"][d.vogel[, "Feld.Nr"] == 1413]
```

e) There are three observations of feeding birds:

```
sum(d.vogel[, "Taetigkeit"] == "fr")
## [1] 3
```

Total number of feeding birds:

```
sum(d.vogel[, "Anzahl"][d.vogel[, "Taetigkeit"] == "fr"])
## [1] 114
```

Corresponding observations/rows:

```
which(d.vogel[, "Taetigkeit"] == "fr")
## [1] 3 4 6
```

Alternatively you could use

```
(1:dim(d.vogel)[1])[d.vogel[, "Taetigkeit" ] == "fr"]
## [1] 3 4 6
```

f)
```
d.vogel[8, 4] <- 6          #Change number of lapwings of 8th row to 6
d.vogel[8, "Anzahl"] <- 6       #Gives the same as first command
d.vogel <- d.vogel[-c(3,7), ]    #Deletes third and seventh observations
```

5. Getting to know data: iris blossoms

a) `iris` is a data frame with 150 observations and 5 variables.

```
nrow(iris)
ncol(iris)
dim(iris)
str(iris)
```

b) In the summary, for each variable the quartiles, the maximum and minimum values and the mean and the median are provided.

c)
```
t.x <- iris$Sepal.Length
t.s <- c(min(t.x), quantile(t.x, 0.25), median(t.x), mean(t.x),
         quantile(t.x, 0.75), max(t.x))
names(t.s) <- c("min", "25%", "median", "mean", "75%", "max")
t.s
##      min      25%   median     mean      75%      max
## 4.300000 5.100000 5.800000 5.843333 6.400000 7.900000
```

6. Missing values

a)
```
d.iris[2, 3:4] <- NA
d.iris[5, c(1, 2, 4)] <- NA
```

b) The functions `class()`, `nrow()`, `ncol()`, `dim()`, `str()` do not care whether `NA`s exist.

```
d.iris[1:4, ]
class(d.iris)
nrow(d.iris)
ncol(d.iris)
dim(d.iris)
```

A line is added to the summary giving the total number of `NA`s for the respective variable.

```
summary(d.iris)
```

c) Either the result itself is also an `NA` (typically an undesired outcome for further calculation steps) or R returns an error.

```
t.x <- d.iris[, "Sepal.Length"]
min(t.x)
## [1] NA
quantile(t.x, 0.25)
## Error in quantile.default(t.x, 0.25): missing values and NaN's not allowed
if 'na.rm' is FALSE
median(t.x)
## [1] NA
mean(t.x)
```

```
## [1] NA
quantile(t.x, 0.75)
## Error in quantile.default(t.x, 0.75): missing values and NaN's not allowed
if 'na.rm' is FALSE
max(t.x)
## [1] NA
```

d) Add `na.rm = TRUE` to all functions as an argument.

```
t.s <- c(min(t.x, na.rm = TRUE), quantile(t.x, 0.25, na.rm = TRUE),
         median(t.x, na.rm = TRUE), mean(t.x, na.rm = TRUE),
         quantile(t.x, 0.75, na.rm = TRUE), max(t.x, na.rm = TRUE))
names(t.s) <- c("min", "25%", "median", "mean", "75%", "max")
t.s
##      min      25%   median     mean      75%      max
## 4.300000 5.100000 5.800000 5.848993 6.400000 7.900000
```

e) The missing values must not be filled with entries that could theoretically be possible in the data set. Example: If the age of a patient has not been observed and is entered as "zero" into the data frame, the resulting `mean(age)` will be wrong. Better: Use an impossible value (such as 999 for "age") or `NA`. Just leaving the field empty is not so advisable, because you cannot be sure afterwards whether the values was available and just forgotten by mistake.
Note that when reading the data into R, you should always make sure that the missing values are stored as `NA`s:

```
read.table(..., na.strings = c("999", "."))
```

f) The functions for variance and standard deviation also can take the argument `na.rm = TRUE`.

```
var(c(1, 2, NA))
## [1] NA
var(c(1, 2, NA), na.rm = TRUE)
## [1] 0.5
sd(c(1, 2, NA))
## [1] NA
sd(c(1, 2, NA), na.rm = TRUE)
## [1] 0.7071068
```

Calculating the correlation of two variables requires pairs of observations, meaning that the input for the `cor()` function should be two vectors of the same length. Simply removing the `NA` values of one vector will disrupt the pairings, and the `cor()` function will throw an error. One therefore needs to remove the pair of observations, and not just the missing value. Instead of the `na.rm` argument, `cor()` has the `use` argument, which specifies which pairs of points to keep (find out more by reading the help file with `?cor`).

```
## Will return NA with default settings
cor(c(1, 2, 3, NA), c(3, 7, 6, 8))
## [1] NA
cor(c(1, 2, 3, NA), c(3, 7, 6, 8), na.rm = TRUE) ## na.rm is not an argu-
ment
## Error in cor(c(1, 2, 3, NA), c(3, 7, 6, 8), na.rm = TRUE): unused argument
(na.rm = TRUE)
## of cor()
cor(c(1, 2, 3), c(3, 7, 6)) # Correlation based on complete pairs
```

```
## [1] 0.7205767
cor(c(1, 2, 3, NA), c(3, 7, 6, 8), use = "complete.obs") # Same as above
## [1] 0.7205767
dd <- na.omit(d.iris[,1:2])
cor(dd[, 1], dd[, 2])  ## or  cor(dd)
## [1] -0.109964
```

g)

```
is.na(d.iris$Sepal.Length)
which(is.na(d.iris$Sepal.Length))
is.na(d.iris$Petal.Length)
which(is.na(d.iris$Petal.Length))
d.iris[is.na(d.iris$Sepal.Length) | is.na(d.iris$Petal.Length), ]
```

h)

```
d.iris2 <- na.omit(d.iris)
d.iris3 <- na.omit(d.iris[, 1:3])
nrow(d.iris2)
## [1] 148
nrow(d.iris3)
## [1] 148
```