

## Solutions to Exercise Series 4

1. a) iris is a data frame with 150 observations and 5 variables. This can be deduced directly from `str(iris)`.

```
> class(iris)
[1] "data.frame"
> nrow(iris)
[1] 150
> ncol(iris)
[1] 5
> dim(iris)
[1] 150  5
> str(iris)
'data.frame':  150 obs. of  5 variables:
 $ SPECIES: int  1 1 1 1 1 1 1 1 1 1 ...
 $ SEP.L  : num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ SEP.W  : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ PET.L  : num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ PET.W  : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
```

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

```
> summary(iris)
      SPECIES      SEP.L      SEP.W      PET.L      PET.W
Min.   :1   Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
1st Qu.:1   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
Median :2   Median :5.800   Median :3.000   Median :4.350   Median :1.300
Mean   :2   Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:3   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :3   Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

```
c) t.x <- iris$SEP.L
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
```

2. 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 NAs exist.

```

> d.iris[1:8,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1          3.5          1.4          0.2  setosa
2          4.9          3.0          NA           NA  setosa
3          4.7          3.2          1.3          0.2  setosa
4          4.6          3.1          1.5          0.2  setosa
5           NA           NA          1.4          NA  setosa
6          5.4          3.9          1.7          0.4  setosa
7          4.6          3.4          1.4          0.3  setosa
8          5.0          3.4          1.5          0.2  setosa
> class(d.iris)
[1] "data.frame"
> nrow(d.iris)
[1] 150
> ncol(d.iris)
[1] 5
> dim(d.iris)
[1] 150  5

```

A line is added to the summary giving the total number of NAs for the respective variable.

```

> summary(d.iris)
  Sepal.Length   Sepal.Width   Petal.Length   Petal.Width   Species
Min.   :4.30   Min.   :2.00   Min.   :1.00   Min.   :0.10   setosa   :50
1st Qu.:5.10   1st Qu.:2.80   1st Qu.:1.60   1st Qu.:0.30   versicolor:50
Median :5.80   Median :3.00   Median :4.40   Median :1.30   virginica :50
Mean   :5.85   Mean   :3.05   Mean   :3.77   Mean   :1.21
3rd Qu.:6.40   3rd Qu.:3.30   3rd Qu.:5.10   3rd Qu.:1.80
Max.   :7.90   Max.   :4.40   Max.   :6.90   Max.   :2.50
NA's   :1.00   NA's   :1.00   NA's   :1.00   NA's   :2.00

```

- 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...
> 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...
> max(t.x)
[1] NA

```

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

```

t.s <- c(min(t.x, na.rm=T),quantile(t.x,0.25, na.rm=T),median(t.x, na.rm=T),
         mean(t.x, na.rm=T),quantile(t.x,0.75, na.rm=T),max(t.x, na.rm=T))
names(t.s) <- c("min","25%","median","mean","75%","max")

```

```
> t.s
      min      25%  median   mean    75%    max
4.30    5.10    5.80    5.85    6.40    7.90
```

- 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:** When **reading the data into R**, always make sure, the missing values are stored as NAs: `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))
Error in var(c(1, 2, NA)) : missing observations in cov/cor
> var(c(1,2,NA), na.rm=TRUE)
[1] 0.5

> sd(c(1,2,NA))
Error in var(x, na.rm = na.rm) : missing observations in cov/cor
> sd(c(1,2,NA), na.rm=TRUE)
[1] 0.7071
```

To calculate the correlation of data containing missing values is not useful. Because always **pairs** of numbers will be necessary (the vectors have to be of equal length!). It is possible however to remove the value of the paired variable if the “partner” variable contains an NA. For the correlation between two variables in a data frame, this corresponds to the deletion of an observation as soon as one of the two correlated variables contains a NA.

```
> cor(c(1,2,NA),c(3,7,8))
Error in cor(c(1, 2, NA), c(3, 7, 8)) : missing observations in cov/cor
> cor(c(1,2,3,NA), c(3,7,6,8), na.rm=T)
Error in cor(c(1, 2, 3, NA), c(3, 7, 6, 8), na.rm = T): unused argument(s)
(na.rm=TRUE)  ## na.rm gibt es nicht bei cor()!
> cor(c(1,2,3), c(3,7,6))
[1] 0.7205767

> dd <- na.omit(d.iris[,1:2])
> cor(dd[,1], dd[,2])  ## oder cor(dd)
[1] -0.11
```

- g) `is.na(d.iris$SEP.L); which(is.na(d.iris$SEP.L))`  
`is.na(d.iris$PET.L); which(is.na(d.iris$PET.L))`

```
> d.iris[is.na(d.iris$Sepal.Length) | is.na(d.iris$Petal.Length) , ]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
2           4.9           3           NA           NA  setosa
5            NA           NA           1.4           NA  setosa
```

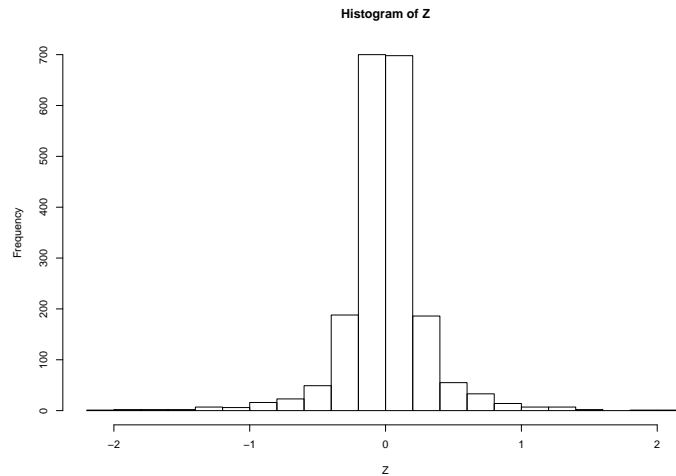
### 3. Simulation in R

a) You can create the variables  $X_1$ ,  $X_2$  and  $Z = \frac{X_1}{X_2}$  using:

```
> X1 <- rnorm(2000, mean = 0, sd = 1)
> X2 <- runif(2000, min = 1, max = 11)
> Z <- X1/X2
```

For the histogram, do not use overly wide bins.

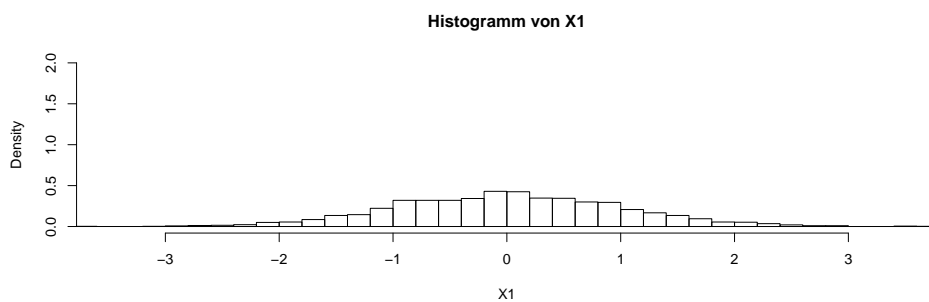
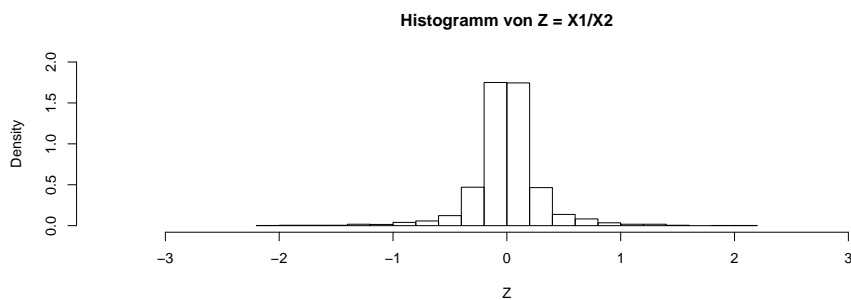
```
> hist(Z, nclass=30, main="Histogram of Z = X1/X2")
```



b) By explicit specification of the axis limits, you ensure that the two plots are comparable. The first command instructs R to split the plot window into 2 rows (and 1 column).

```
> par(mfrow=c(2,1))
```

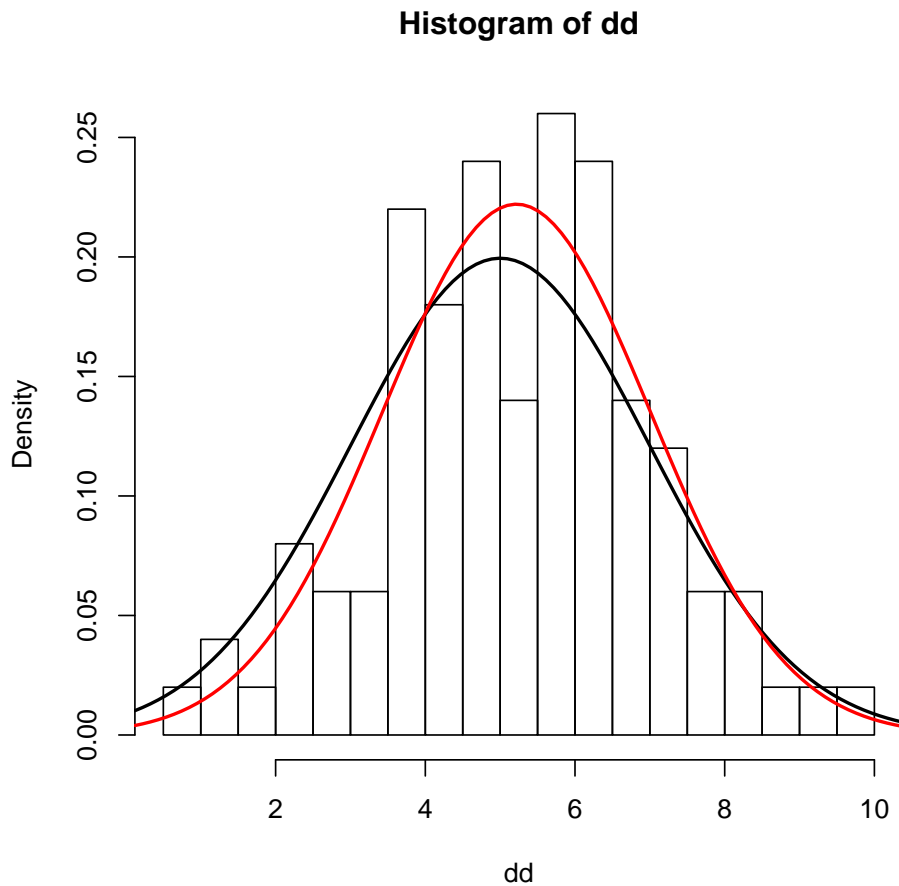
```
> hist(Z, freq=FALSE, nclass=30, xlim=c(-3.5,3.5), ylim=c(0,2),
+ main="Histogram of Z = X1/X2")
> hist(X1, freq=FALSE, nclass=30, xlim=c(-3.5,3.5), ylim=c(0,2),
+ main="Histogram of X1")
```



#### 4. Histograms and Normal-Distribution

```
a) > # Generate data from normal distribution
> set.seed(1)
> dd <- 5+2*rnorm(100)

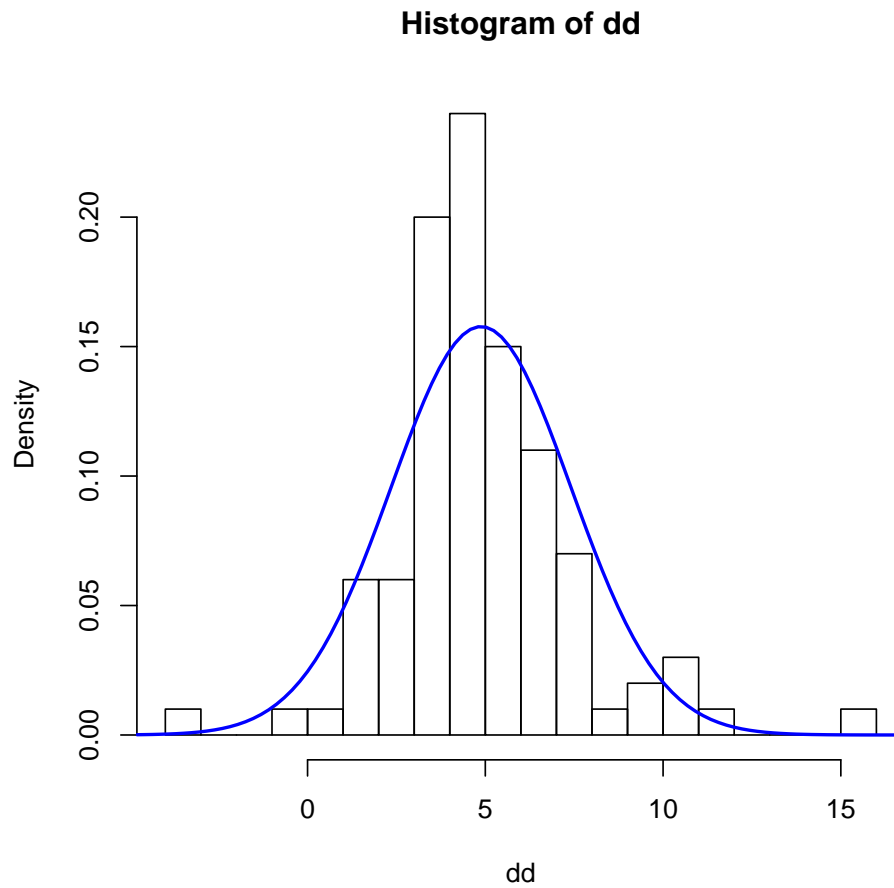
> # Plot Histogram, freq=FALSE switches on density-Scale
> hist(dd, freq=FALSE, nclass=15)
> #
> # (i):
> # Since we know the expected value and variance in the design, we can
> # compare those values to the mean and variance from the theoretical
> # normal distribution:
> t.mn <- 5 #Mean
> t.sd <- 2 #Standarddeviation
> #
> t.xl <- par('usr')[1:2] # get range of plot
> t.x <- seq(t.xl[1],t.xl[2], length=100) # equally spaced base points
> lines(t.x, dnorm(t.x, t.mn, t.sd), col='black', lwd=2)
> #
> # (ii):
> # characteristic values
> t.mn <- mean(dd); t.mn #Mean
[1] 5.217775
> t.sd <- sd(dd); t.sd #Standarddeviation
[1] 1.796399
> #
> # Overlay Density function
> lines(t.x, dnorm(t.x, t.mn, t.sd), col='red', lwd=2)
```



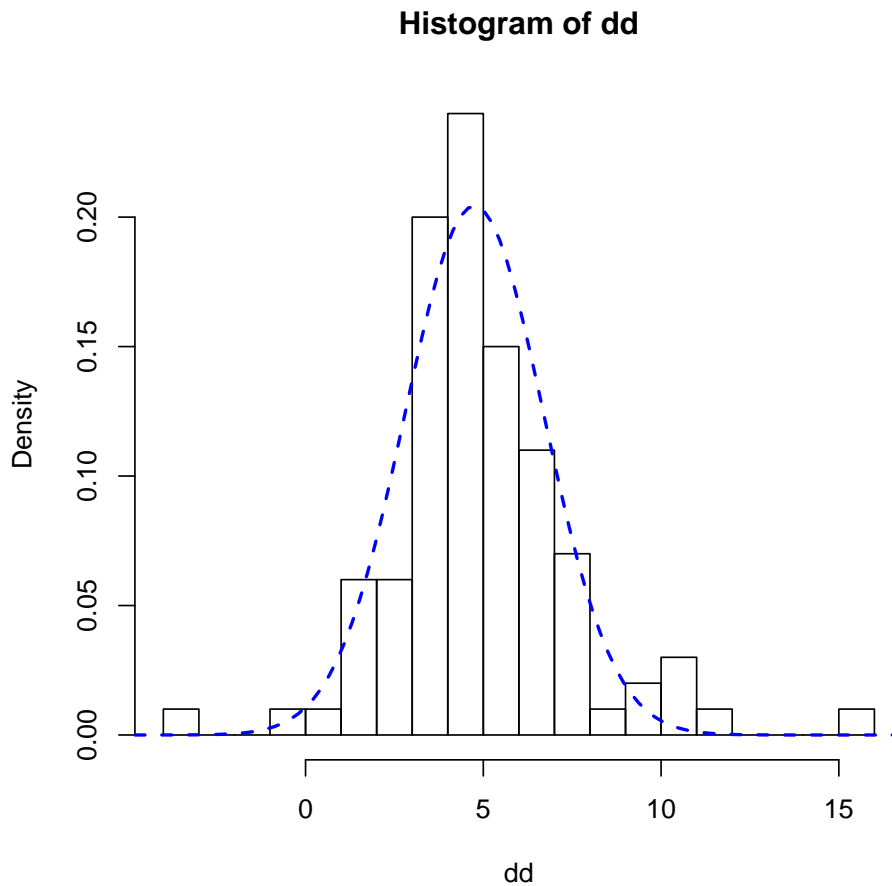
```

b) > #
  > # Generate data from t-distribution
  > dd <- 5+2*rt(100,5)
  > #
  > # Start again with ``hist'' etc.
  > # Plot Histogram, freq=FALSE
  > hist(dd, freq=FALSE, nclass=15)
  > # characteristic values
  > t.mn <- mean(dd); t.mn #Mean
  [1] 4.879302
  > t.sd <- sd(dd); t.sd #Standarddeviation
  [1] 2.529243
  > #
  > t.xl <- par('usr')[1:2] # Get range of plot
  > t.x <- seq(t.xl[1],t.xl[2], length=100) # equidistant base points
  > # Overlay density function
  > lines(t.x, dnorm(t.x, t.mn, t.sd), col='blue', lwd=2)

```



```
> #  
> # Now with the robust measures  
> hist(dd, freq=FALSE, nclass=15)  
> # characteristic values  
> t.med <- median(dd); t.med #Median  
[1] 4.746771  
> t.mad <- mad(dd); t.mad # Median Absolute Deviation  
[1] 1.952765  
> #  
> # Overlay density function  
> lines(t.x, dnorm(t.x, t.med, t.mad), col='blue', lwd=2, lty=2)
```



**Interpretation:**

The t-distribution produces more *extreme* observations than the normal distribution: it is long-tailed.

Extreme observations overly influence the arithmetic mean and the standard deviation. The median and the MAD in contrast are not disturbed much more by *outliers* than by moderately extreme observations: they are *robust* measures of position and variability/scatter. The corresponding curve fits the data well in the middle of the distribution that make up most of the data – but the extreme observations are less well captured here than by the curve that is based on mean and standard deviation.

Generally, the goal of robust statistics is to represent the major part of the data well and to react as little as possible to *extreme errors*. Such errors can hence be also better identified.