Frankfurt University of Applied Sciences
Prof. Dr. Christina Andersson

R Tutorial

1 Introduction

1.1 What is R?

`http://www.r-project.org.`

1.2 A 1-Minute Inroduction

- To start R using Linux, type in the command line:

  ```
  > R
  ```

- If you use Windows, you can after installation find R as a programme.
- To finish R:

  ```
  > q()
  ```

- Comments in R:

  ```
  > #
  ```

- Start the built-in help system:

  ```
  > help.start()
  ```

2 Manipulation of Vectors and Numbers

2.1 Vectors and Assignments

To assign the values 4, 5 and 6 to the vector x, write:

```
> x=c(4,5,6)
```

If you then want to take a look at the current content of the vector x, then just type x and the current value will be displayed:

```
> x
[1] 4 5 6
```

Here [1] means that the display starts with the first observation in the vector.
Let us now also create the vector y with the content 1, 2 and 3:

```
> y=c(1,2,3)
```

To add two vectors elementwise, proceed as follows:

```
> x+y
[1] 5 7 9
```

The values obtained in the vector addition are shown at once, since we didn't give a name to the new vector. If we want to store the result of the vector addition in a new vector, called z, proceed as follows:

```
> z=x+y
```

2.2 Extraction of Elements from Vectors

In order to display the third element of the vector, write:

```
> x[3]
[1] 6
```

If we want to display both the second and the third value of the vector x, we type:

```
> x[c(1,3)]
[1] 4 6
```

Here, it would not work with $x[1,3]$, since this would be a matrix manipulation (see next section)! If we want to display the elements from the first to the third, we type:

```
>  x[1:3]
[1] 4 5 6
```

2.3 Matrices

A matrix is a rectangular structure of numbers. The following examle shows how we can create a 2×3 matrix:

```
> g = matrix(c(4,5, 6, 7, 8, 9), nrow = 2, ncol = 3)
```

To display the matrix, type:

```
> g
     [,1] [,2] [,3]
[1,]    4    6    8
[2,]    5    7    9
```

If we want to enter the data values per row instead if per column, use the additional option *byrow* = *T*:
matrix(vector of data, numrows, numcols, byrow=T), i.e. this would in our example be:

```
> gt = matrix(c(4,5, 6, 7, 8, 9), nrow = 2, ncol = 3, byrow=T)
```

which is displayed with

```
> gt
     [,1] [,2] [,3]
[1,]    4    5    6
[2,]    7    8    9
```

To display the first row of the matrix g, use:

```
> g[1,]
[1] 4 6 8
```

Analogously, to show the element located at 2,3 (i.e. row 2 and column 3), use:

```
> g[2,3]
[1] 9
```

2.4 Basic Manipulations

In this section, we use the example vector $x = c(1, 3.5, 7.2222)$.

- Round:
  To round the values of the vector to contain only two numbers after the decimal point, we use:

  ```
  > round(x,2)
  [1] 1.00 3.50 7.22
  ```

- Reverse the order:
  If we want to reverse the order of the elements of a vector, use

  ```
  > rev(x)
  [1] 7.2222 3.5000 1.0000
  ```

- Sort:
  If we want to sort the elements of the vector in ascending order:

  ```
  > sort(x)
  [1] 1.0000 3.5000 7.2222
  ```

  or in descending order:

  ```
  > sort(x, decreasing=T)
  [1] 7.2222 3.5000 1.0000
  ```

- Determine the number of elements:
  If we want to determine the number of elements of the vector, we use

  ```
  > length(x)
  [1] 3
  ```

- Smallest element:
  If we want to determine the minimum of the vector:

  ```
  > min(x)
  [1] 1
  ```

- Construct a sequence:
  To construct a sequence of numbers, we use

  ```
  > seq(1,3, length=5)
  [1] 1.0 1.5 2.0 2.5 3.0
  ```

  As an alternative, the same sequence can be generated with:

  ```
  >  seq(1, 3, by = 0.5)
  [1] 1.0 1.5 2.0 2.5 3.0
  ```

  A sequence that starts at 3 and stops at 1 and from one number to the next is reduced by 0.5 can be obtained with:

```
> seq(3,1, by=-0.5)
[1] 3.0 2.5 2.0 1.5 1.0
```

- Draw random sample with replacement:
  We can draw a random sample with replacement, containing two numbers from the vector x with the following command:

  ```
  >  sample(x, 2, replace = T )
  [1] 1 1
  ```

- Draw random sample without replacement:
  To draw a random sample without replacement, containing three numbers from a sequence of numbers, we proceed as follows:

  ```
  > sample(1 : 500, 3, replace = F )
  [1] 423 139 221
  ```

  If we draw with replacement, it means that the same number can be drawn again. If we draw without replacement, a number can only be drawn once and is then put aside, i.e. each number cannot be used more than once.

2.5 The Data Frame

It is often useful to summarize different variables in a common data frame.

- Construction of a data frame:
  Let us start with the variables x and y.

  ```
  > x = c(1, 2, 3)
  > y = c(4, 5, 6)
  > x
  [1] 1 2 3
  > y
  [1] 4 5 6
  ```

  We construct the data frame *mydata*:

  ```
  > mydata = data.frame(x, y)
  > mydata
    x y
  1 1 4
  2 2 5
  3 3 6
  ```

- Using data in a data frame:
  To reach the variable x in the data frame *mydata* and (as an example) calculate the arithmetic mean of x: ¿ mydata$x ¿ mydata$x [1] 1 2 3 ¿ mean(mydata$x) [1] 2

3 Tables and Graphs

3.1 Tables

- Table of absolute frequencies:
  We use the command *table* to produce a table of absolute frequencies:

```
> No_of_cats = c(0,0,1,1,1,2,3,3,3,2,3)
> abs_freq = table(No_of_cats)
No_of_cats
0 1 2 3
2 3 2 4
```

- Table of relative frequencies:
  We divide the absolute frequencies with the total number of elements.

```
> rel_freq = table(No_of_cats)/length(No_of_cats)
> rel_freq
No_of_cats
        0         1         2         3
0.1818182 0.2727273 0.1818182 0.3636364
```

3.2 Graphs

3.2.1 General about Graphs

Here we just use a very simple graph, the scatter plot *plot(x, y)*, to illustrate some features about graphs.

- In order to use labels for the axis, proceed as here:

```
> plot(x, y, xlab = ''Age'', ylab = ''Income'')
```

- A main title for the diagram is done with *main*:

```
> plot(x, y, main = ''My title'')
```

- We can determine the limits of the axes:

```
> plot(x, y, ylim = c(0, 40))
```

- If you want to create two diagrams above each other in the same graphics window:

```
> par(mf row = c(2, 1))
> plot(x, y)
> plot(x2, y2)
```

- If you want to create two diagrams beside each other in the same graphics window:

```
> par(mf row = c(1, 2))
> plot(x, y)
> plot(x2, y2)
```

- If you use Linux and want to save a graphics as an .eps-file:

  ```
  > postscript(''test.eps'')
  > plot(x, y)
  > dev.off()
  ```

  With *postscript*, you open a device for printing. Then you should proceed with all graphic commands (in the example only *plot*). In the end, close the device with *dev.off()*.

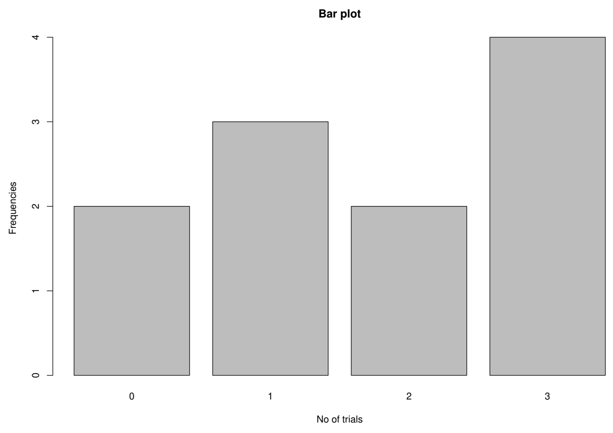
- If you use Windows and want to save a graphics:
  Create the graphics, right-click and copy & paste.

3.2.2 Bar Plot

To construct a bar plot, proceed as follows:

```
> barplot(abs_freq,names.arg=c("0","1","2","3"),main="Bar plot",
xlab="No of trials",ylab="Frequencies")
```

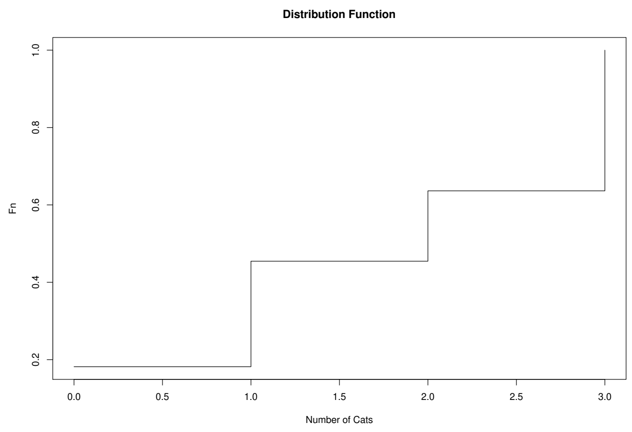
Here we used the variable *abs_freq* from the example above.



3.2.3 Cumulative Distribution Function

To draw the cumulative distribution function, proceed as follows (data based on previous examples about relative frequency table):

```
fn <- cumsum(rel_freq)
plot(sort(unique(exam)),fn,type="n",xlab="Number of Cats",
ylab="Fn",main="Distribution Function")
lines(sort(unique(exam)),fn,type="s")
```



3.2.4 Boxplot

To create a boxplot, use:

```
boxplot(x)
```

3.2.5 Histogram

To create a histogram, use:

```
hist(x)
```

3.2.6 Pie Chart

To create a pie chart, use:

```
pie(x)
```

4 Measures of Central Tendency

In this section, we use the example vector $x = c(5, 6, 7)$.

- Arithmetic mean:
  In order to compute the arithmetic mean we use

  ```
  > mean(x)
  [1] 6
  ```

- Median:
  To compute the median, we use:

  ```
  >  median(x)
  [1] 6
  ```

- Summary:
  Use the function *summary()* in order to at once obtain some common measures:

```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    5.0     5.5     6.0     6.0     6.5     7.0
```

The results of summary show the minimum value, the first quartile, the median, the arithmetic mean, the third quartile and the maximum value of the data.

- Quantile:
  The function *quantile()* can be used to calculate specific quantiles of the data:

  ```
  > quantile(x, 0.5)
  50%
    6
  ```

  and

  ```
  > quantile(x, 0.78)
   78%
  6.56
  ```

5 Measures of Spread

In this section, we use the example vector $x = c(5, 6, 7)$.

- Standard deviation:
  To compute the standard deviation, we can use

  ```
  > sd(x)
  [1] 1
  ```

- Variance:
  For the variance, we use

  ```
  > var(x)
  [1] 1
  ```

- Range:
  The maximum and the minimum value, which can be used to calculate the range, are obtained with the function *range()*:

  ```
  > range(x)
  [1] 5 7
  ```

  This means that in our exampe the range is 7-5=2.

6 Correlation

- Covariance:
  To calculate the covariance between two vectors, we use

  ```
  cov(x, y)
  ```

- Pearson correlation:
  For the Pearson correlation we need

  ```
  cor(x, y)
  ```

- Spearman correlation:
  Finally, for the Spearman correlation we used

  ```
  cor(x, y, method="spearman")
  ```

7 Regression

7.1 Simple Linear Regression

If we use Y as the dependent variable and x as the independent variable, then we perform a simple linear regression analysis in R with the command

```
> our_model = lm(y ~ x)
```

where we store the resulting model in the variable *our_model*. The function *lm* can also be used to perform more complicated regression analysis, but let us in this section start with a simple linear regression model.
We can look at the result of the regression analysis, by giving the command:

```
> summary(our_model)
```

Let us as an example perform a simple linear regression analysis for the data set *women* in the package *datasets*. We use the variable *weight* as dependent variable and the variable *height* as independent variable. We start with the following command:

```
> our_model = lm(women$weight ~ women$height)
```

Continuing with

```
> summary(our_model)
```

gives the output:

```
Call:
lm(formula = women$weight ~ women$height)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7333 -1.1333 -0.3833  0.7417  3.1167

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  -87.51667    5.93694  -14.74 1.71e-09 ***
women$height   3.45000    0.09114   37.85 1.09e-14 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.525 on 13 degrees of freedom
Multiple R-squared:  0.991,     Adjusted R-squared:  0.9903
F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

From this tablle, we can see that the estimated regression line in this case is

```
women$weight = -87.51667 + 3.45 * women$height
```

We can also see that the variable *height* is significant in the model, since the p-value, in the row where woman$height is, is very low, namely 1.09e-14.
Finally, the value for the multiple R-squared is 0.991. The multiple R-squared is another name for the coefficient of determination.

7.2 Multiple Linear Regression

If we include more than one independent variable (also called explanatory variable) in the analysis, we talk about multiple linear regression.
In the following example, we use these data:

```
y=c(8,8,8,9,9,7,6,5,5,5,6,7,8,7,5,5,5,8,8,8,9,8,7,6,5)
x1=c(7,7,7,7,8,7,6,6,6,6,6,7,8,4,4,5,5,6,8,8,7,7,7,6,6)
x2=c(1,2,9,7,2,3,4,4,6,1,1,2,1,3,3,1,1,1,8,8,3,3,3,4,4)
```

If we want to use both $x1$ and $x2$ as independent variables in the model with y as dependent variable, we use the following code:

```
> model1=lm(y ~ x1 + x2)
```

and we take a look at the result with

```
> summary(model1)
```

The output looks in our case like this:

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4914 -0.4845 -0.3055  0.6188  2.3262

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.049126   1.292678   0.812 0.425721
x1          0.903595   0.207041   4.364 0.000248 ***
x2          0.003447   0.095417   0.036 0.971505
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
Residual standard error: 1.085 on 22 degrees of freedom
Multiple R-squared:  0.4887,    Adjusted R-squared:  0.4422
F-statistic: 10.51 on 2 and 22 DF,  p-value: 0.000624
```

We can from the output see that the estimated function is

```
y = 1.049126 + 0.903595 * x1 + 0.003447 * x2
```

From the values in the column $Pr(>|t|)$, we can see that the variable $x1$ shows significance, whereas the variable $x2$ is not significant.

If we want to determine the residuals, we type

```
residuals(our_model)
```

which here would result in this list with residuals:

```
> residuals(model1)
         1          2          3          4          5          6          7
 0.6222605  0.6188131  0.5946811  1.6015760  0.7152179 -0.3846343 -0.4844866
         8          9         10         11         12         13         14
-1.4844866 -1.4913814 -1.4741443 -0.4741443 -0.3811869 -0.2813346  2.3261512
        15         16         17         18         19         20         21
 0.3261512 -0.5705491 -0.5705491  1.5258557 -0.3054666 -0.3054666  1.6153657
        22         23         24         25
 0.6153657 -0.3846343 -0.4844866 -1.4844866
> residuals(our_model)
          1           2           3           4           5           6
 2.41666667  0.96666667  0.51666667  0.06666667 -0.38333333 -0.83333333
          7           8           9          10          11          12
-1.28333333 -1.73333333 -1.18333333 -1.63333333 -1.08333333 -0.53333333
         13          14          15
 0.01666667  1.56666667  3.11666667
```

8 Probability Distributions

8.1 Simulation of Normally Distributed Data

In order to simulate n data points from a normal distribution with expectation m and standard deviation s, we use

```
rnorm(n, m, s)
```

In this example, we draw a random sample with size 10 from a normal distribution with expectation 0 and standard deviation 1:

```
> x=rnorm(10, 0, 1)
> x
 [1]  0.9788789 -1.2472658  0.9007622  0.2054623 -0.2446444  0.5971050
 [7] -0.1390959 -1.6645601  0.4239374 -0.3431368
```

Analogously to how you use *rnorm* to simulate observations from a normal distribution, you can use *rbeta* to simulate observations from a beta distribution, *rbinom* to simulate observations from a binomial distribution, *rchisq* to to simulate observations from a chisquare distribution.

8.2 The *p*, *q* and *d* Functions

For a continuous distribution (like the normal distribution), the most useful functions for solving problems that include probability calculations are the *p* and *q* functions (corresponding to the cumulative distribution function and the inverse cumulative distribution function, respectively).
For discrete distributions, we analogously have the the *d* and *q* functions.

8.2.1 Calculation of Probabilities under the Normal Curve

We can use the command *pnorm(x, mean, sd)* to calculate the probability of obtaining a value less than x under the normal distribution. The arguments *mean* and *sd* give the mean and standard deviation of the applied normal distribution.
Assume that you want to know the probability that X is less than 50, where X is normally distributed with expectation 49 and standard deviation 2. The following command gives us the desired probability:

```
> pnorm(50, mean = 49, sd = 2)
[1] 0.6914625
```

If we instead need $P(X > 52)$, then we use:

```
> 1 - pnorm(52, mean = 49, sd = 2)
[1] 0.0668072
```

Be careful with the parameter *sd*! R really wants the standard deviation as the parameter, not the variance. If we start with the variance in the problem statement, we'll need to take the square root!

8.2.2 *qnorm*

qnorm is the R function that calculates the inverse cumulative distribution function of the normal distribution. So let a number p between zero and one be given, then *qnorm* looks up the p-th quantile of the normal distribution.
Here is an example. Which x-value corresponds to the probability 0.95 of a normal distribution with expectation 5 and standard deviation 2?

```
> qnorm(0.95, mean=100, sd=15)
[1] 124.6728
```

8.2.3 The functions *dbinom* and *qbinom*

For a discrete distribution (e.g. the binomial distribution), the *d* function (i.e. *dbinom*) calculates the density (i.e. the probability function), which for the discrete case is a probability

$$f(x) = P(X = x)$$

and thereful is useful in calculating probabilities.
For example, let us calculate the probability $P(X \leq 27)$ when X is Bin(100, 0.25)-distributed. This is done with

```
> dbinom(27, size=100, prob=0.25)
[1] 0.08064075
```

or shorter

```
> dbinom(27, 100, 0.25)
[1] 0.08064075
```

qbinom is the R function that calculates the ”inverse cumulative distributive function” of the binomial distribution.
As an example, we want to know the 10th quantile of the Bin(10, 0.2)-distribution. This is solved with

```
qbinom(0.1, 10, 0.2)
```

9 Hypothesis Tests

9.1 Test of the mean, if big sample

We consider independently, identically distributed random variables $X_1, \ldots, X_n$. If n is big enough (how big?), we can use the test statistics

$$Z = \frac{\bar{X} - \mu}{s/\sqrt{n}} \text{ approx. } N(0,1)$$

for a test of the expectation μ.

Critical values for the rejection region for the standard normal distribution can immediately be obtained with the function *qnorm(a)*, which gives the quantile at a.
If we want the critical values for a two-sided test at significance level 5%, we type:

```
> qnorm(0.975)
[1] 1.959964
```

This means that the critical values are approx. -1.96 and +1.96.
For a corresponding one-sided test, we would used

```
> qnorm(0.95)
[1] 1.644854
```

or

```
> qnorm(0.05)
[1] -1.644854
```

depending on the direction of the alternative hypothesis.
The p-values can also be calculated with *pnorm(z)*, where z is the observed value of the test statistics Z. If we, e.g. get the value $z = 1.72$ for the observed test statistic, then we obtain the following p-value for a two-sided test:

```
> 2*(1-pnorm(1.72))
[1] 0.08543244
```

and for a one-sided test:

```
> 1-pnorm(1.72)
[1] 0.04271622
```

Observe that we have to calculate *1-pnorm(z)* to obtain the correct value, since *pnorm* is the distribution function of the normal distribution.

9.2 Test of the Mean, if Normal Distribution and Variance Known

If we use normally distributed data with known variance, the results in the previous section holds, but exactly, i.e. without approximation.

9.3 Test of the Mean, if Normal Distribution and Variance Unknown

If we can assume that the sample comes from a normal distribution, but that the variance of the sample has to be estimated, we should use a t-distribution. Then our starting point is the following test statistics:

$$T = \frac{\bar{X} - \mu}{s/\sqrt{n}}\; t_{n-1}$$

In R the function *t.test* exists. This function can be used to perform different kinds of t-tests. A lot of different options can be specified, e.g. if it should be a one- or two-sided two, the significance level (or confidence level) etc.
If we have the sample data in the vector x and want to perform the following test at significance level 10%:

$$\begin{array}{ll} H_0\text{:} & \mu \leq 4.5 \\ H_1\text{:} & \mu > 4.5 \end{array}$$

we use

```
> t.test(x, alternative = greater, mu = 4.5, conf level = 0.90)
```

If we want the alternative hypothesis, but the other way around, we change *greater* to *less*. If we want the alternative hypothesis to be two-sided, we use the option *two.sided*.

9.4 Test of Independence

To test if two variables are independent we use the following command:

```
chisq.test(x)
```

where x is a matrix containing the data.
The hypothesis test has the following null hypothesis and alternative hypothesis:

$$\begin{array}{ll} H_0\text{:} & \text{The variables are independent.} \\ H_1\text{:} & \text{The variables are not independent.} \end{array}$$

10 Confidence Intervals

If we construct the confidence interval based on the t-distribution, we can use the function *t.test*, as described in section 9.3. In addition to the test result, also the corresponding confidence is automatically shown.
If we base the construction of the confidence interval on the normal distribution, we can obtain the value of the normal distribution ($u_{1-\alpha}$), as described in section 9.1, and use this value in the following formula:

$$\bar{x} \pm u_{1-\alpha}\frac{\sigma}{\sqrt{n}}$$