Biostatistics in R I

Exercises

Veit Schwämmle

1 R Basics

EXERCISE 1.1 Read file *DataTable.csv* with `read.table()` and check whether it has been input correctly. Rownames and colnames should be set. Adjust them with `colnames()` if required.
Calculate the number of missing values for each column. Count number of complete rows (without missing values).

EXERCISE 1.2 Read the data frame `Pima.tr2` from the MASS library. Install package if not available (`install.package("MASS")`).

```
> library(MASS)
> data(Pima.tr2)
```

Get the dimensions of the data frame (`dim()`). Again, count the number of missing values for each column. Use `table()` to make statistics about how many rows do have how many missing values.

EXERCISE 1.3 Use the functions `mean()` and `range()` to find the mean and range of.

1. the numbers 1, 2, . . . , 21
2. the sample of 50 random values generated from a normal distribution with mean 0 and variance 1 using `rnorm(50)`. Repeat several times with new set of 50 random numbers to get a feeling about random numbers.
3. the columns `height` and `weight` in the data frame `women` (already available in R).

EXERCISE 1.4 Repeat exercise 3, now applying functions `median()`, `sum()` and `sd()`.

EXERCISE 1.5 Get dataset mammals that is part of package MASS. Plot `brain` versus `body`. Use the $\sim$ operator within the `plot` command. Additionally, do the same plot accessing the columns directly. Try to visualize the data on logarithmic scale applying the `log=` argument.

EXERCISE 1.6 Get the data set `trees` and investigate the relationships between `Girth`, `Height` and `Volume`. Try to plot all 3 in one figure by adding the missing relationship with `lines` or `points`.

EXERCISE 1.7 Get data set `genotype` from library(MASS) and sort the data by column `Wt` with `order`. Sort the data also by column `Mother`. Then sort by both `Mother` and then `Wt`.

EXERCISE 1.8

1. Create a `for` loop that, given a numeric vector, `print`s out each number of the vector on a separate line, with its square and cube alongside.
2. Look up `help(while)`. Show how to use a `while` loop to achieve the same result.
3. Show how to achieve the same result without the use of an explicit loop.

EXERCISE 1.9 Carry out the following commands

```
> paste("Leo","the","lion")
> paste("a","b")
> paste("a","b", sep="")
> paste(1:5)
> paste(1:5, collapse="")
```

What do the arguments `sep` and `collapse` do? Try out your own examples to find out.

EXERCISE 1.10 The following function calculates the mean and standard deviation of a numeric vector.

```
> MeanAndSd <- function (x) {
+   av <- mean(x)
+   sdev <- sd(x)
+   c(mean=av, sd=sdev)
+ }
```

Modify the function so that: (a) the default is to use `rnorm()` to generate 20 random numbers; (b) if there are missing values, the mean and standard deviation are calculated for the remaining values.

2 More advanced R

EXERCISE 2.1 Play with the class command such as `class("a")`. Load the `cabbages` data frame from the MASS library. Use `sapply(cabbages, class)` to find out which columns are numeric. Extract those columns into a separate data frame.

EXERCISE 2.2 Apply functions that may be used to get information about data frames, such as `class`, `dim`, `rownames`, `mean` and `str`. Get the data sets `DNase` and `cabbages`. The command `sapply()` provides information about the columns. Retrieve `table`, `mean` and `class` for the columns.

EXERCISE 2.3 Investigate the distribution of random samples from a normal distribution. Plot histogram and overlay a density plot. Put several plots into one window using `par(mfrow=c(2,2))`. Repeat the following commands several times.

```
> y <- rnorm(100)
> hist(y, probability=TRUE)
> lines(density(y))
```

Vary the number of random samples, taking sizes of 25, 100, 500 and 2000. Make several plots for each. Discuss when these plots are useful and how they could be improved.

EXERCISE 2.4 The function `sample()` randomly mixes the numbers of a vector. Apply the function on the vector 1:5. Repeat with the argument `replace=T`. What does it change?

EXERCISE 2.5 Load the data set `Pima.tr2` from the *MASS* package. The first seven columns classify diabetes according to `type`. Let's check the properties of the missing values of the individual columns.

1. Count the number of missing values per column. Investigate whether missing values occur systematically with respect to rows with specific `type`.

   ```
   > library(MASS)
   > # function to count missing values of a vector/column
   > count.na <- function(x) sum(is.na(x))
   ```

   The `by()` function allows to apply function on individual `types`, e.g.

   ```
   > by(Pima.tr2, Pima.tr2$type, function(x) sapply(x, count.na))
   ```

   What does this function do? Which column behaves differently for different `types` (count number of rows per `type`)?

2. Add new column `anymiss` to the data set that tells whether the row contains missing values. The function `complete.cases()` might be helpful in this task.

EXERCISE 2.6 Quantile-quantile plots are a powerful way to check probability distributions. In order to understand them better, we now combine them with density plots. Carry out the following commands and vary `mean` and `sd` of the random variables.

```
> a <- rnorm(100, mean=0, sd=1)
> b <- rnorm(200, mean=0, sd=1)
> plot(density(a))
> lines(density(b), col=2)
> qqplot(a,b)
> abline(0,1)
```

What do you observe?

EXERCISE 2.7 Get the data frame `mammals` from the *MASS* package. Plot the data on a double-logarithmic scale (argument `log="xy"` in the plot function). Do a linear regression on this scale:

```
> mam.lm <- lm(log10(brain) ~ log10(body), data=mammals)
> abline(mam.lm)
> coef(mam.lm)
```

What is the precise equation that we fitted to the data?

EXERCISE 2.8 Sometimes, calculations can take long times and it is a good idea to optimize the script to carry out the tasks faster. `system.time()` provides timings. The output is user cpu time, system cpu time and elapsed time. Hence, we can check how long each command takes and identify bottlenecks in a script.
We now use the timings to illustrate the difference between carrying out identical operations on matrices and data frames.

```
> my.matrix <- matrix(runif(10000000), ncol=100)
> my.datframe <- as.data.frame(my.matrix)
> system.time(rowMeans(my.matrix))
> system.time(rowMeans(my.datframe))
```

What does rowMeans calculate? Do you note the difference between matrices and data frames? Vary the size of the matrix.
We now compare the functions `apply` ad `sapply` with `for` loops.

```
> system.time(out <- colMeans(my.matrix))
> system.time(out <- colMeans(my.datframe))
> system.time(out <- apply(my.matrix, 2, mean))
> system.time(out <- apply(my.datframe, 2, mean))
> system.time({out <- numeric(ncol(my.matrix))
+   for(i in 1:ncol(my.matrix)) {
+     out[i] <- mean(my.matrix[,i])
+   }})
> system.time({out <- numeric(ncol(my.datframe))
+   for(i in 1:ncol(my.datframe)) {
+     out[i] <- mean(my.datframe[,i])
+   }})
> # matrix calculation
> system.time({colOnes <- rep(1, nrow(my.matrix))
+              out <-  t(my.matrix) %*% colOnes / nrow(my.matrix)})
```

Discuss the results. Check class of `out` for the different operations.

EXERCISE 2.9 Pick on of the calculations of exercise 8 and plot the system times for different matrix sizes. Do the same for the data frames. Compare to the other ones.

3 Distributions

EXERCISE 3.1

1. Plot the density (`dnorm()`) and the cumulative (`pnorm()`) probability distribution of a normal distribution with mean 2.5 and standard deviation 1.5.

2. Read the probability of having a number between 0.5 and 4 from the cumulative distribution. Check the number with

   ```
   > pnorm(4, 2.5, 1.5) - pnorm(0.5, 2.5, 1.5)

   [1] 0.7501335
   ```

   Do the same with other numbers

3. The relative number of observations per unit interval around $x = 2$ (between 1.5 and 2.5) is given by `dnorm(x=2, 2.5, 1.5)`. Hence

   (a) In a sample of 100 the expected number of observations per unit interval in the immediate vicinity of $x = 2$ is 25.16

   (b) In a sample of 1000 the expected number of observations per unit interval in the immediate vicinity of $x = 2$ is 251.6

   (c) The expected number of values from a sample of 1000, between 1.9 and 2.1, is approximately $0.2 \cdot 251.6 = 50.32$, or, more precisely,

   ```
   > pnorm(2.1, 2.5, 1.5) - pnorm(1.9, 2.5, 1.5)

   [1] 0.05028465
   ```

   Do the calculation repeatedly with other numbers. Do the same with other intervals.

EXERCISE 3.2

1. Plot the density and cumulative probability distribution (`dt()` and `pt` with argument `df=3`) for a t-distribution with 3 degrees of freedom. Plot the normal distribution over it with `lines()`.

2. Plot the density and cumulative probability distribution for an exponential distribution with a rate parameter equal to 1 (the default). Repeat with a rate parameter equal to 2. What happens when you do the plot on logarithmic (y-coordinate) and double-logarithmic scale?

EXERCISE 3.3 Use the function `rnorm()` to draw a random sample of 25 values from a normal distribution with a mean of 0 and a standard deviation equal to 1.0. Use a histogram, with `probability=TRUE` to display the values. Overlay the histogram with: (a) an estimated density curve; (b) the theoretical density curve for a normal distribution with mean 0 and standard deviation equal to 1.0. Repeat with samples of 100, 500 and 1000 values, showing the different displays in different panels on the same graphics page (`par(mfrow=...)`)

EXERCISE 3.4 Data whose distribution is close to lognormal are common. Size measurements of biological organisms often have this character. As an example, consider the measurements of body weight (`body`) in the data frame `Animals` (*MASS* package). Begin by drawing a histogram of the untransformed values, and overlay a density curve. Then

1. Draw an estimated density curve for the logarithms of the values.

2. Determine the mean and standard deviation of `log(Animals$body)`. Overlay the estimated density with the theoretical density for a normal distribution with the mean and standard deviation just obtained.

Does the distribution look like a normal distribution after transformation to a logarithmic scale?

EXERCISE 3.5 The following plots an estimated density curve for a random sample of 50 values from a normal distribution:

```
> plot(density(rnorm(50)), type="l")
```

1. Plot estimated density curves, for random samples of 50 values, from (a) the normal distribution; (b) the uniform distribution (`runif(50)`); (c) the t-distribution with 3 degrees of freedom. Overlay the three plots and different colors.

2. Repeat the previous exercise, but taking random samples of 500 and 5000 values

EXERCISE 3.6 There are two ways to make the estimated density smoother:

1. One is to increase the number of samples

2. The other is to increase the bandwidth. For example

   ```
   > plot(density(rnorm(50), bw=0.2), type="l")
   > plot(density(rnorm(50), bw=0.6), type="l")
   ```

Repeat each of these with bandwidths of 0.15, with default choice of bandwidth, and with the bandwidth set to 0.75.

EXERCISE 3.7 Here we experiment with the use of `sample()` to take a sample from an empirical distribution, i.e. from a vector of values that is given as argument. Here, the sample size will be the number of values in the argument. Any size of sample is however permissible.

```
> sample(1:5, replace=T)
> for (i in 1:10) print(sample(1:5, replace=T))
> plot(density(log10(Animals$body)))
> lines(density(sample(log10(Animals$body), replace=T)), col="red")
```

Repeat the final density plot several times, perhaps using different colors for the curve on each occasion. This gives an indication for the stability of the estimated density curve with respect to sample variation.

EXERCISE 3.8 The density estimation has the issue that it depends strongly on bandwidth and choice of kernel, making it sometimes not very useful to judge normality. A much better tool is the quantile-quantile plot, which works with cumulative probability distributions. Try

```
> qqnorm(rnorm(10))
> qqnorm(rnorm(15))
> qqnorm(rnorm(200))
```

See how the plot deviates when comparing the normal distribution with random variables from other distributions.

EXERCISE 3.9 Take the data sets `lh` and `Animals` and check for normality using `qqnorm`. Do the same on the logarithmic values. Which one is normal distributed? Additionally, use `boxplot()` to get an idea about how the boxplot of a normal distribution looks.

EXERCISE 3.10 Here, we will calculate the limit distribution of the mean of random variables. Note that the mean corresponds to the sum divided by the number of variables, and therefore the central limit theorem applies.
First take a random sample from the normal distribution, and plot the estimated density function

```
> y <- rnorm(100)
> plot(density(y), type="l")
```

Now, take the repeated samples of size 4, calculate the mean for each such sample, and plot the density function for the distribution of means:

```
> av <- numeric(100)
> for (i in 1:100) {
+   av[i] <- mean(rnorm(4))
+ }
> lines(density(av), col=2)
```

Additionally, use `qqnorm()` to estimate normality.
Repeat this code, using different sample numbers (e.g. 9 and 25) and numbers of averages larger than 100.

EXERCISE 3.11 In Exercise 10, we calculated the mean of normally distributed variables. The central limit theorem applies for almost arbitrary distributions. Show this by calculating the mean distribution of n uniformly distributed variables (`runif(n)`), log-normally distributed ones (`rlnorm(n)`) and exponentially distributed ones (`rexp(n,rate=1)`).

EXERCISE 3.12 It is also possible to take random samples, usually with replacement, from a vector of values, i.e. from an empirical distribution. This is the bootstrap concept. Again, it may of interest to study the sampling distributions of means of different sizes. Consider the distribution of heights of female Adelaide University students, in the data frame `survey` (*MASS* package). The following takes 1000 bootstrap samples of size 4, calculating the mean for each such sample:

```
> library(MASS)
> y <- na.omit(survey[survey$Sex == "Female", "Height"])
> av <- numeric(1000)
> for (i in 1:1000)
+   av[i] <- mean(sample(y, 4, replace=T))
```

Repeat, taking samples of size 9 and 16. In each case use a density plot and `qqnorm()` to display the (empirical) sampling distribution.

EXERCISE 3.13 Generate random numbers from a normal distribution with a sequential dependence.

```
> y1 <- rnorm(51)
> y <- y1[-1] + y1[-51]
```

Try to understand the definition of y. The autocorrelation function (`acf()` in R) calculates the dependence within a series (see also http://en.wikipedia.org/wiki/Autocorrelation). Apply this function on both data sets and check whether there is a consistent pattern for the correlated data set. Vary number of data points and repeat the experiment several times to get feeling of how a autocorrelation function can look like.

EXERCISE 3.14 Create a function that calculates the correlated data set of Exercise 13. The input of the function is the number of data points with a default value of 51.
Create a `for` loop that calculates the sum and the mean of the correlated data set 1000 times. Check whether the sum and the mean are normally distributed. What do the results suggest?

EXERCISE 3.15 Take the artificial count data for e.g. the number of tumors in 7 rats suffering from a certain type of cancer.

```
> dat <- c(87, 53, 72, 90, 78, 85, 83)
```

Enter the data and compute mean and variance. In order to check whether a Poisson model would be appropriate, calculate seven random values for the corresponding Poisson distribution (`lambda=78.3`). Take their mean and variance and compare them to the artificial data.
Calculate the distribution of mean and variance, plot their histograms and check whether mean and variance from the artificial data are within the main core of the distributions. Does the Poisson model fit well?

Data modeling

EXERCISE 3.16 Take the data frame `ChickWeight` and fit a linear dependency between `weight` and `Time`. Split the data into data frames containing the data for different `Diet` and do the linear regression again. Are the diets performing differently?
Check the distribution of the residuals. Does it look as it should? What about the `qqnorm()` of the residuals?
Try different regressions by data transformation of explanatory and response variables (try logarithm and inverse). Therefore, delete rows with `Time=0`. Which regression looks best?

EXERCISE 3.17 The data set `pressure` contains measurements of vapor pressure versus the absolute temperature. According to the theory (Clausius-Clapeyron equation), the logarithm of the pressure is approximately inversely proportional to the temperature. Transform the data, plot it and fit a linear regression line. Add the line to the graph. Is the fit adequate?

EXERCISE 3.18 Get data frame `oddbooks` from the package *DAAG* (You'll have to install the package).

1. Add a further column that gives the density
2. Use the function `pairs()`, or the *lattice* function `splom()`, to display the scatterplot matrix. Which variables show evidence for a strong relationship?
3. In each panel of the scatterplot matrix, record the correlation for that panel (use `cor()` to calculate correlations)
4. Fit the following regression relationships:
   (a) `log(weight)` on `log(thick)`, `log(height)` and `log(breadth)`.
   (b) `log(weight)` on `log(thick)` and 0.5 (`log(breadth)` + `log(height)`). Transform the data before using `lm()`. What feature of the scatterplot matrix suggests that it might make sense to use this form of equation?

EXERCISE 3.19 Plot `distance` versus `speed` of the `cars` dataset. Fit a line and a quadratic curve to the data. Does the quadratic fit improve the data model? Check residuals and use `anova()` for comparison of the two models.

EXERCISE 3.20 Create a function that generates a simulated data set for a linear regression based on the model
$y = 2 + 3x + \epsilon$.
The input of the function is assumed to consist of specific values of x (see below). The noise term should be normally distributed with mean 0 and standard deviation 1.
Create two datasets: a) from 10 points that are uniformly distributed; b) 10 points where half of them are 1 and the other half -1. Do a linear regression on both datasets and compare standard error estimates, intercept, slope, R squared and p-values. What are advantages and disadvantages between the two datasets?

EXERCISE 3.21 Get the dataset `hills2000` from the `DAAG` package. The set is an updated version of the `hills` data. Carry out a regression of the logarithm of `time` versus `log(dist)` and `log(climb)` for both data sets. Compare the results, check residuals etc. and plot them together.

EXERCISE 3.22 Take the dataset `litters` from the `DAAG` package. Carry out linear regressions of `brainwt` versus `bodywt` alone, versus `lsize` alone and versus both with and without interaction. Compare the results using `anova()`.

EXERCISE 3.23 An up-to-date version of the `ozone` data frame can be found at http://www.antarctica.ac.uk/met/jds/ozone/data/ZOZ5699.DAT.
Read the data directly with `read.table` (adjust attributes to read the file correctly). Substitute 0 values by NA. Calculate the number of missing values for each month.
Plot the ozone concentration of October over the years. Fit a smooth curve using the `lowess()` function. Adjust the attributes of the function to see the effect of changing smoother span. To do this, read the help page and generate the data points of the curve. Estimate the year at which the ozone concentration begins to decrease strongly. Do the same for the other months as well. Do the other months show a similar pattern of decline?

EXERCISE 3.24 The following table shows number of occasions when inhibition across a membrane occurred (yes/no), for different protein concentrations.

```
> inhib <- rbind(conc=c(0.1,0.5,1,10,20,30,50,70,80,100,150),
+                no=c(7,1,10,9,2,9,13,1,1,4,3),
+                yes=c(0,0,3,4,0,6,7,0,0,1,7))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
conc  0.1  0.5    1   10   20   30   50   70   80   100   150
no    7.0  1.0   10    9    2    9   13    1    1     4     3
yes   0.0  0.0    3    4    0    6    7    0    0     1     7
```

Use logistic regression to model the probability of inhibition as a function of protein concentration.
a) Transpose the table and add a column `inhfreq` containing the number of inhibitions over the number of trials.
b) Use `glm(inhfreq~conc, ..., family=binomial(link='logit'), ...)` with the attribute `weights=` where you put the number of trials
c) Plot the output of the data model (residuals, ...).

4 Statistical testing

EXERCISE 4.1 Draw figures that show, for degrees of freedom between 1 and 100, the change in the 5% critical value of the t-statistic (that is the value above and below which 5% of the density distribution are located, `qt(0.975)`). Do the plot on absolute and double-logarithmic scale. Which figure provides better visual information about the critical values? What does the decrease mean?

EXERCISE 4.2 Take an artificial data set of three groups

```
> stat.dat <- data.frame(x=rep(c("A","B","C"),each=10),
+                        y=c(rnorm(10), rnorm(10)+0.5, rnorm(10)-1))
```

a) Do an ANOVA to check whether B and C are significantly different from A
b) Do an ANOVA to check whether A, B and C are significantly different from 0
c) Do an ANOVA to check whether A and C are significantly different from B (you'll have to reorder the columns of `stat.dat`)
d) Calculate a t-test between C and A and compare the result to a)
e) Redo ANOVA on the data set containing only A and C and compare again

EXERCISE 4.3 a) Redo Exercise 2 using a sample size of 5 instead of 10.
b) Change the normal distribution to an exponential one and check whether you would accept any of the results as significantly different.

EXERCISE 4.4 Take the data `PlantGrowth` and reduce it to conditions `ctrl` and `trt2`. Carry out a standard two-sample t-test, the Wilcoxon rank test and a permutation test. Compare the results.
Write a function to get the p-values of the three tests. Use the function on normally distributed artificial data having the same number of values, the same mean and the same standard deviation as `ctrl` and `trt2`.
Check the distribution of the p-values and compare them to the one obtained for the `PlantGrowth` data.

EXERCISE 4.5 Redo the first part of Exercise 4 changing to paired tests. Compare and think about when to do a paired test.

EXERCISE 4.6 Correction for multiple testing: Write a function to get the p-value (t-test) between normally distributed (s.d. 1) artificial sets mutually shifted by S (set default to 0.5). Write a for loop to get 1000 p-values from the same comparison and plot them on a histogram. Count the number of p-values below 0.05.
Correct for multiple testing using Bonferroni, Benjamin-Hochberg (`p.adjust`)and `qvalue` (install the `qvalues` package). Count the number of corrected p-values below 0.05.
Repeat the same for a shift of $S = 1$ and $S = 0$. How many corrected p-values below 0.05 would one optimally get for $S = 0$?

5 High-throughput

EXERCISE 5.1 Get the data `geneData` from the `Biobase` package. Normalize the columns by a) mean, b) median, c) mean of log-values, d) median of log-values. Revise the results extensively by comparing the distributions with plots for histograms, density plots, ranked plots and qqnorm. Do also a direct comparison by scatter plots.

EXERCISE 5.2 Take the log-transformed `geneData` set and perform t-tests for all genes between sample groups (B, I, K, N, P, T) and (C, G, J, O, R, U, V). Correct for multiple testing. Plot histograms of p-values and generate a volcano plot.
Take randomly chosen samples of 6 versus 6 groups and redo the statistical tests.
Carry out a principal component analysis on the entire data set and look for the groups that you tested for significantly different genes (loading plot).

EXERCISE 5.3 Carry out principal component analysis and linear discriminant analysis (using `age` as discriminator) for the `possum` data. Rows with missing values need to be removed before. Plot the scores of the PCA with different colors for the locations the possums were trapped (defined by `site`). Compare the outcome to the scaling plot of the LDA.

EXERCISE 5.4 Read the file *Example.csv* and extract the column with the protein accessions.
a) Pick out one of the values and apply `strsplit` to separate database name (e.g. TREMBL, SWISS-PROT) from accession id.
b) Take a value with multiple protein accession and extract only the accession ids.
c) Operate `strsplit` on the entire column and try to extract the accession ids.

EXERCISE 5.5 Carry out hierarchical clustering, k-means and fuzzy c-means on `geneData`.
Extract the columns 114-117 from *ExampleFile.csv* and take the logarithm. Normalize the data to the median and do the clustering on the resulting data.