# Computational Statistics

## *An Introduction to* 

Günther Sawitzki

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

# Computational Statistics

# An Introduction to 

# Exercises

**April 30, 2014 DRAFT**

---

Günther Sawitzki
StatLab Heidelberg
April 30, 2014

*in preparation*

*http://sintro.r-forge.r-project.org/*

# Contents

CHAPTER 1

# Basic Data Analysis

## 1.1 R Programming Conventions

## 1.2 Generation of Random Numbers and Patterns

### *1.2.1 Random Numbers*

| **Exercise 1.1** | |
|---|---|
| | Try experimenting with these plots and `runif()`. Do the plots show images of random numbers?<br>To be more precise: do you accept these plots as images of 100 independent realisations of random numbers, distributed uniformly on $(0, 1)$?<br>Repeat your experiments and try to note as precisely as possible the arguments you have for or against (uniform) randomness. What is your conclusion?<br>Walk through your arguments and try to draft a test strategy to analyse a sequence of numbers for (uniform) randomness. Try to formulate your strategy as clearly as possible.<br>*Hint:* For comparison, you can keep several plots in a window. The code<br>`par(mfrow = c(2, 3))`<br>parametrises the graphics system to show six plots simultaneously, arranged row wise as a $2 \times 3$ matrix (2 rows, 3 columns).<br>The function `par` is the central function to control graphics parameters. For more information, see `help(par)`. |

### *1.2.2 Patterns*

| **Exercise 1.2** | |
|---|---|
| | Use<br>`plot(sin(1:100))`<br>to generate a plot of a discretised sine function. Use your strategy from Exercise 1.1. Does your strategy detect that the sine function is not a random sequence?<br>*Hint:* If you do not recognise the sine function at first sight, use `plot(sin (1:100), type = "l")` to connect the points. |

## 1.3 Case Study: Distribution Diagnostics

### *1.3.1 First Pass for Example* ??*: Distribution Functions*

### *1.3.2 First Pass for Example* ??*: Histograms*

| **Exercise 1.3** | |
|---|---|
| | Use `runif(100)` to draw random numbers and generate histograms with 5, 10, 20, 50 cells of equal size. Use repeated samples.<br>Do the histogram plots correspond to what you expect from independent uniform random variates? Try to note your observations in detail.<br>Repeat the experiment with two cells $(0, 0.5], (0.5, 1)$.<br>`hist(runif(100), breaks = c(0, 0.5, 1))`<br>Repeat the experiment with random numbers generated by `rnorm(100)` and compare the results from `runif(100)` and `rnorm(100)`. |

| **Exercise 1.4** | |
|---|---|
| | Modify Example ?? (page ??) to include the kernel name and the bandwidth used in the kernel density estimation.<br>You have to store the result from `density()` and access its components in analogy to Example ?? (page ??). |

*Barcharts*

### *1.3.3 Statistics of Distribution Functions; Kolmogorov-Smirnov Tests*

| **Exercise 1.5** | |
|---|---|
| | Using `help(rbeta)` you get information about the functions available to work with beta distributions. Generate plots for the densities of the beta distribution for $n = 16, 32, 64, 128$ and $i = n/4, n/2, 3n/4$. Use the function `curve()` to generate the plots. For more information, see `help(curve)`. |

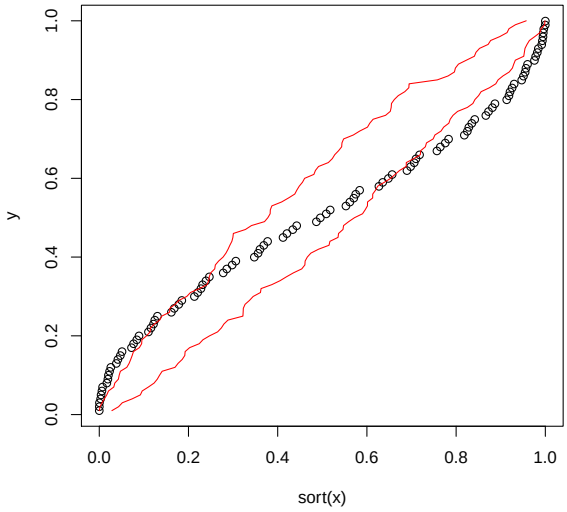
| **Exercise 1.6** | |
|---|---|
| | Draw the distribution function with the corrected reference line. |
| * | We use the graphical display for a single sample, not for a run of samples. Is the expected value of $X_{(i)}$ an adequate reference? Are there alternatives that can serve as references?<br>If you see alternatives, give an implementation. |

*Monte Carlo Confidence Bands*

***Example* 1.1: Monte Carlo Confidence Bands**

*Input*

```
x <- (sin(1:100)+1)/2                    # demo example only
y <- (1:length(x))/length(x)
plot(sort(x), y)
nrsamples <- 19                          # no of simulations
samples <- matrix(data = runif(length(x)* nrsamples),
    nrow = length(x), ncol = nrsamples)
samples <- apply(samples, 2, sort)
envelope <- t(apply(samples, 1, range))
lines(envelope[, 1], y, col = "red")
lines(envelope[, 2], y, col = "red")
```




| **Exercise 1.7** | |
|---|---|
| | Make use of the `help()`-function and comment on Example 1.1 step by step. Take special note of the new functions that are introduced here. |

| R ***Iterators*** | |
|---|---|

(cont.)→

| R *Iterators* (cont.) | |
|---|---|
| `apply()` | applies a function to the rows or columns of a matrix.<br>*Example:* `samples <- apply(samples, 2, sort)` sorts by column. |
| `outer()` | generates a matrix of all pair-wise combinations of two vectors, and applies a function to all pairs. |

| **Exercise 1.8** | |
|---|---|
| * | Why 19?<br>*Hint:* Try to take an abstract simplified view of the problem first: let $T$ be a measurable function and $X_0, X_1, \ldots, X_{nrsamples}$ independent samples with a common distribution function.<br>What is $P(T(X_0) > T(X_i))$ for all $i > 0$?<br>In a second step, give an abstract formulation for the example above. Then consider the special case `nrsamples = 19`. |

| **Exercise 1.9** | **Monte Carlo Coverage** |
|---|---|
| * | Estimate the coverage probability of the Monte Carlo band by first generating a band as above. (How can you draw the band without first making a plot for a special sample?)<br>Next, generate *sim* simulation samples of uniform random numbers of sample size 100. Count how many simulations give a sample within the band. You have to make your choice of the number *sim* of simulations (100? 1000? 999?) for this step.<br>Use this information to estimate the coverage probability.<br>*Hint:* `any()` can be used to evaluate a comparison for a full vector. |

**Theorem 1.1** *For all integer $n$ and any positive $\lambda$, we have*

$$P(\sqrt{n}\sup|F_n - F| > \lambda) \leq 2e^{-2\lambda^2}.$$

*Proof.* [6], Corollary 1 □

This inequality is valid even if $F$ is not continuous.

| **Exercise 1.10** | **Finite Sample Bounds** |
|---|---|
| | Use the inequality given in Theorem 1.1 to calculate bounds for $\sqrt{n}\sup\|F_n - F\|$. |
| | Add finite sample bands to the empirical distribution function. |

| **Exercise 1.11** | |
|---|---|
| | Using *help(ks.test)* you get information on how to invoke the function *ks.test*.<br>Which results do you expect if you test the following vectors for a uniform distribution?<br>*1:100*<br>*runif(100)*<br>*sin(1:100)*<br>*rnorm(100)*<br>Perform these tests and discuss the results. For the test, scale the values so that they fall into the interval $[0, 1]$, or use a uniform distribution on an interval that is adapted to the data. |

### *1.3.4 Statistics of Histograms and Related Plots; $\chi^2$-Tests*

| **Exercise 1.12** | |
|---|---|
| | Use *help(chisq.test)* to see the calling structure for $\chi^2$ tests. Apply it to test the hypothesis $(p_j = 1/J),\ J = 5$ on the following vectors of bin counts:<br>(3 3 3 3 3) (1 2 5 3 3) (0 0 9 0 6). |

| **Exercise 1.13** | |
|---|---|
| | Which results do you expect if you use a $\chi^2$ test to check the following vectors for a uniform distribution?<br>*1:100*<br>*runif(100)*<br>(cont.)→ |

| **Exercise 1.13** | (cont.) |
|---|---|
| | `sin(1:100)`<br>`rnorm(100)`<br>Perform these tests and discuss the results.<br>*Hint:* The function `chisq.test()` expects a frequency table as input. The function `table()` can be used to generate a frequency table directly (see `help(chisq.test)`). But you can also use the function `hist()`, which gives `counts` as one component of its result. |

| **Exercise 1.14** | |
|---|---|
| * | Sketch comparable test environments for fixed and adaptive choice of histogram cells.<br>For fixed and for adaptive choice of histogram cells draw `s = 1000` samples of size 50 from `runif()`. Calculate in both settings the formal $\chi^2$ statistics and plot its distribution functions.<br>Compare the distribution functions. |

| **Exercise 1.15** | |
|---|---|
| | For $n = 10, 50, 100$, draw 300 samples using `runif(n)`. For each sample, calculate the $\chi^2$ and the Kolmogorov-Smirnov statistic.<br>You have to choose a $\chi^2$ test. What is your choice?<br>Plot the distribution functions of these statistics and compare them to the theoretical (asymptotic) distributions.<br>Are there any indications against the assumption of independent uniform random numbers?<br>*Hint:* The functions for the $\chi^2$ and Kolmogorov-Smirnov test keep their internal information as a list. To get the names of the list elements, you can create a sample object. For example, use<br>`names(chisq.test(runif(100)))`. |

| **Exercise 1.16** | |
|---|---|
| ** | Analyse the power of the Kolmogorov-Smirnov test and the $\chi^2$ tests. Select values for $n, m$ and $\alpha$, and choose 9 pairs for $(a, b)$. What are your arguments for your choices?<br>Use your chosen parameters to draw samples from `rbeta()`.<br>Apply the Kolmogorov-Smirnov test and a $\chi^2$ test with 10 cells of equal size on $(0, 1)$. |
| | Choose alternative parameters $(a, b)$ so that you can compare the decision rules along the following lines:<br>i) a = b<br>ii) b = 1<br>iii) a = 1<br>and run these simulations. |
| | Choose alternative parameters $(a, b)$ so that you can compare the decision rules over the range $0 < a, b < 5$.<br>Your conclusions?<br>*Hint:* `outer(x, y, fun)` applies a function `fun()` to all pairs of values from $x, y$ and returns the result as a matrix.<br>Using<br>`contour()`<br>you can generate a contour plot.<br>See `demo("graphic")`. |

| **Exercise 1.17** | |
|---|---|
| ⋆⋆ | Design a test strategy to unmask "pseudo-random numbers".<br>Test this strategy using simple examples<br>i) $x \quad x = 1..100 \mod m$ for convenient $m$<br>ii) $\sin(x)$ $x = 1..100$<br>iii) ...<br>Do you tag these sequences as "not random"?<br>Now try to unmask the random number generators provided by R. Can you identify the generated sequences as "not random"? |

## 1.4 Moments and Quantiles

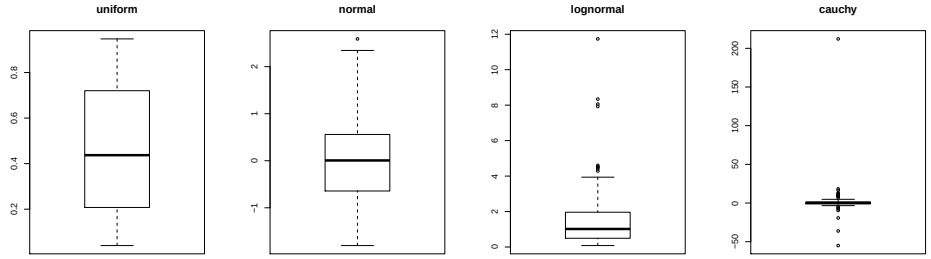
| **Exercise 1.18** | |
|---|---|
| | Generate a sample of random variables with sample size 100 from the distributions with the following densities:<br>$$p(x) = \begin{cases} 0 & x < 0 \\ 1 & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases}$$<br>and<br>$$p(x) = \begin{cases} 0 & x \leq 0 \\ 2 & 0 < x \leq 1/4 \\ 0 & 1/4 < x \leq 3/4 \\ 2 & 3/4 < x \leq 1 \\ 0 & x > 1 \end{cases}$$<br>Estimate the mean, variance and standard deviation in each of these.<br>Repeat the estimation for 1000 samples. Analyse the distribution of estimated mean, variance and standard deviation for repeated samples. |

| **Exercise 1.19** | |
|---|---|
| | Generate a sample of 100 random variables from the distributions of Exercise 1.18.<br>Estimate the median, and the lower and upper quartiles.<br>Repeat the estimation for 1000 samples. Analyse the distribution of the estimated median, lower and upper quartiles from repeated samples. |

***Example* 1.2: Box-and-Whisker Plot**

*Input*

```
oldpar <- par(mfrow = c(1, 4))
boxplot(runif(100), main = "uniform")
boxplot(rnorm(100), main = "normal")
boxplot(exp(rnorm(100)), main = "lognormal")
boxplot(rcauchy(100), main = "cauchy")
par(oldpar)
```




| **Exercise 1.20** | |
|---|---|
| | Modify Example 1.2 so that the plots are comparable: adjust the location so that the medians are at the same height. Adjust the scales so that the inter-quartile ranges have same length. |

| **Exercise 1.21** | |
|---|---|
| | For continuous distributions and the median $X_{med}$ we have $$P(X_i \geq X_{med}) = 0.5.$$ Hence we can find a $k$ such that $$k = min\{k : P(X_{(k)} \leq X_{med}) < \alpha\}$$ and $X_{(k)}$ as an upper bound for the median with confidence level $1 - \alpha$.<br>Use this idea to construct a confidence interval for the median with confidence level $1 - \alpha = 0.9$. |
| | Modify the box-and-whisker plot to show this interval.<br>(cont.)→ |

| **Exercise 1.21** | (cont.) |
|---|---|
| | *Hint:* You need the distribution function $F_X$, evaluated at the position marked by the order statistic $X_{(k)}$. The distributions of $F_X(X_{(k)})$ are discussed in Theorem **??**. |
| | The box-and-whisker plot offers an option `notch = TRUE` to mark confidence intervals. Try to use the documentation to find out how a `notch` is calculated. Compare your confidence intervals with those marked using `notch`. |
| $*$ | Use an analogous strategy to get a distribution-independent confidence interval for the inter-quartile range. |
| $***$ | Augment the box-and-whisker plot so that it gives information about the scale in a way that is statistically reliable.<br>*Hint:* Why is it not sufficient to mark confidence intervals for the quartiles? |

## 1.5 R Complements

### *1.5.1 Random Numbers*

### *1.5.2 Graphical Comparisons*

| **Exercise 1.22** | |
|---|---|
| | Generate a $PP$ plot of the $t(\nu)$ distribution against the standard normal distribution in the range $0.01 \leq p \leq 0.99$ for $\nu = 1, 2, 3, \ldots$. |
| | Generate a $QQ$ plot of the $t(\nu)$ distribution against the standard normal distribution in the range $-3 \leq x \leq 3$ for $\nu = 1, 2, 3, \ldots$. |
| | How large must $\nu$ be so that the $t$ distribution is barely different from the normal distribution in these plots? |
| | How large must $\nu$ be so that the $t$ distribution is barely different from the normal distribution if you compare the graphs of the distribution functions? |

| **Exercise 1.23** | |
|---|---|
| | Use $PP$ plots instead of distribution functions to illustrate the $\chi^2$- and Kolmogorov-Smirnov approximations. |

| **Exercise 1.24** | |
|---|---|
| | Use $QQ$ plots instead of distribution functions.<br>Can you add confidence regions to these plots with the help of the $\chi^2$- resp. Kolmogorov-Smirnov statistics? |

| **Exercise 1.25** | |
|---|---|
| | Generate a matrix of dimensions $(nrow * ncol - 1), length(x)$ with random numbers and use `apply()` to avoid the loop.<br>*Hint:* See Example 1.1 (page 3). |

| **Exercise 1.26** | |
|---|---|
| | Use `rnorm()` to generate with pseudo-random numbers for the normal distribution for sample size $n = 10, 20, 50, 100$.<br>For each sample, generate a $PP$ plot and a $QQ$ plot, using the theoretical normal distribution as a reference. |
| | Add Monte Carlo bands from the envelope of 19 simulations.<br>Instead of the uniform distribution, you have to use the normal distribution to generate the Monte Carlo bands. Then you have to represent the results in the coordinate system of the $QQ$ plots, that is, the x axis represents the quantiles of the normal distribution.<br>*Hint:* Inspect the source of `qqnorm()`. |
| * | The bands are initially bands for the standard normal distribution. Find bands adjusted in scale and location of the data at hand. |

### *1.5.3 Complements: Functions*

| **Exercise 1.27** | |
|---|---|
| | Rework your programming exercises and write reusable parts as functions. |

| **Exercise 1.28** | |
|---|---|
| | Write as functions:<br>• A function *ehist* showing an augmented histogram.<br>• A function *eecdf* showing the empirical distribution.<br>• A function *eqqnorm* showing a $QQ$ plot with the standard normal distribution as comparison.<br>• A function *eboxplot* showing a box-and-whisker plot.<br>and<br>• A wrapper function *eplot* showing a plot matrix with these four plots.<br>Your functions should call the standard functions (or modify them, if necessary) and guarantee that the plots have an adequate complete annotation. |

*Vectorisation*

I

| **Exercise 1.29** | **Vectorisation** |
|---|---|
| | Write *sqrt0()* as a vectorised function using *ifelse()*. |

*Compilation*

*1.5.4 Complements: Enhancing Graphical Displays*

| **Exercise 1.30** | |
|---|---|
| | Use *help(plot)* to inspect the possibilities of customising the plot function. Information on details of the parameters is only available if you use *help(plot.default)*. Modify your latest plot so it has a correct main title. |

*1.5.5 Complements:* R *Internals*

*Executing Files*

*1.5.6 Search Paths, Frames and Environments*

## 1.6 Additional Exercises

| **Exercise 1.31** | **Feature Detection** |
|---|---|
| | This series of example tries to judge feature detection sensitivity of various displays for univariate data. for a preparation, find (and fix) a display arrangement that is convenient for your display, (for example `par( mfrow=c(5,4))`. Select a false detection rate you are willing to tolerate (for example, $2/20 = 10\%$). |
| * | Write a function `plotdens <- function(n)` that draws $n$ normal random numbers for each of the display frames.<br>Find a number $n_{sym}$ so that with $n \geq n_{sym}$ observations most results appear symmetric (i.e. the non-symmetric samples are below tolerance rate.<br>Find a number $n_{unimodal}$ so that with $n \geq n_{unimodal}$ observations most results appear unimodal (i.e. the multimodal samples are below tolerance rate. |
| | Modify your function by adding an additional parameter `plotdens <- function(n, generator=rnorm)` that allows to select a random number generator.<br>For the following distributions, find a sample size that allow detection of the given features reliably within tolerance.<br>Generator \| Features<br>uniform \| flat density<br>lognormal \| unimodal, skewed, long tail<br>Cauchy \| symmetric, frequent outliers, long tail. |

| **Exercise 1.32** | **Distribution** |
|---|---|
| | Prepare a plot with one display frame showing a test sample, the others showing uniform random samples. What is the required sample size to identify a normal distribution with an error rate below tolerance level? With this sample size, what is the false detection rate if you start with a uniform sample. |
| | Exchange the roles. Use a uniform sample as a test sample, and normal samples for comparisons. What is the required sample size to identify a uniform distribution with an error rate below tolerance level? With this sample size, what is the false detection rate if you start with a gaussian sample. |

| **Exercise 1.33** | |
|---|---|
| | Repeat the example above using a QQ-Plot. |
| | Repeat the example above using a histogram. |

*1.6.1 Complements: Packages*

*Using Packages*

*Building Packages*

| **Exercise 1.34** | |
|---|---|
| | Install the functions from Exercise 1.28 as a package. You can prepare the package with `package.skeleton()`, if you have already defined the functions.<br>Load the package. Verify that you can still load the package with `library()` if you have restarted the R system.<br>*Hint:* For an object $x$, the statement `prompt(x)` generates a skeleton upon which you can build a documentation for `x`. |

*Compilation*

### 1.7 Statistical Summary

### 1.8 Literature and Additional References

[9] R Development Core Team (2000–2008): Writing R Extensions.
See: <http://www.r-project.org/manuals.html>.

[12] Shorack, G. R.; Wellner, J. A.: *Empirical Processes with Applications to Statistics.* Wiley, New York, 1986.

[3] Gänßler, P.; Stute, W.: *Wahrscheinlichkeitstheorie.* Springer, Heidelberg, 1977.

[4] Gentleman, R.; Ihaka, R.: Lexical Scope and Statistical Computing. *Journal of Computational and Graphical Statistics* 9 (2000) 491–508.

CHAPTER 2

# Regression

## 2.1 General Regression Model

## 2.2 Linear Model

### *2.2.1 Factors*

### *2.2.2 Least Squares Estimation*

For this example data set, we get the least squares estimator using

***Example* 2.1: Least Squares Estimator**

*Input*

```
lm(y ~ x)
```

Output

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     -1.295        2.541
```

| **Exercise 2.1** | |
|---|---|
| | When we generated the data, we did not use a constant term. The model specified for estimation, however, did not exclude the constant term. Repeat the estimation using the model without a constant term. Compare the results. |

***Example* 2.2: Linear Model Summary**

*Input*

```
summary(lm( y ~ x))
```

Output

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-28.2790  -6.3290   0.8087   7.3920  28.3508

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.2947     1.9719  -0.657    0.513
x             2.5410     0.0339  74.958   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.785 on 98 degrees of freedom
Multiple R-squared:  0.9829,        Adjusted R-squared:  0.9827
F-statistic:  5619 on 1 and 98 DF,  p-value: < 2.2e-16
```

| **Exercise 2.2** | |
|---|---|
| | Analyse the output of `lm()` shown in Example 2.2. Which of the terms can you interpret? Write down your interpretations. For which terms do you need more information?<br>Generate a commented version of the output. |

### *2.2.3 Regression Diagnostics*

| **Exercise 2.3** | |
|---|---|
| | Let<br>`yy <- 2.5*x +0.01 * x^2 +  err`<br>What are the results you get if you do a regression using the (incorrect) regression model `yy ~x`? Do you get any hints that this model is not adequate? |

| **Exercise 2.4** | |
|---|---|
| | Use `plot()` to inspect the results of Exercise 2.3. Does it give you indications that the linear model is not appropriate? Which indications? |

### *2.2.4 More Examples for Linear Models*

### *2.2.5 Model Formulae*

| **Exercise 2.5** | |
|---|---|
| | Write the four models from Section 2.2.4 using the R formula notation. |
| | For each of these models, generate an example data set by simulation, and apply `lm()` to the example. Compare the estimators returned by `lm()` with the parameters you have used in the simulations. |

| **Exercise 2.6** | |
|---|---|
| | Generate three vectors of random variables with an $N(\mu_j, 1)$ distribution, $\mu_j = j,\ j = 1, 3, 9$, each of length 10, and combine these into a vector $y$.<br>Generate a vector $x$ with the values $j,\ j = 1, 3, 9$, each repeated 10 times.<br>Calculate the Gauss-Markov estimator in the linear models<br>`y~x` and `y~factor(x)`.<br>Inspect the results as a table using `summary()` and graphically using `plot()`. Compare the results, and give a written report. |

### *2.2.6 Gauss-Markov Estimator*

| **Exercise 2.7** | |
|---|---|
| | What is the distribution of $\|R_X(Y)\|^2 = \|Y - \widehat{Y}\|^2$, if $\varepsilon$ has a $N(0, \sigma^2 I)$ distribution? |

| **Exercise 2.8** | |
|---|---|
| | Modify the output of `plot.lm()` for the linear model so that instead of the Tukey-Anscombe plot the studentised residuals are plotted against the fit. |
| * | Enhance the *QQ*-Plot by Monte Carlo bands for independent normal errors.<br>*Hint:* You cannot generate the bands directly from a normal distribution — you need the distribution of the residuals, not the distribution of the errors. |

| **Exercise 2.9** | |
|---|---|
| | Write a procedure that calculates the Gauss-Markov estimator for the simple linear regression<br>$y_i = a + bx_i + \varepsilon_i \quad$ with $x_i \in \mathbb{R}, a, b \in \mathbb{R}$<br>and shows four plots:<br>• response against regressor, with estimated straight line<br>• studentised residuals against fit<br>• distribution function of the studentised residuals in a *QQ* plot with confidence bands<br>• histogram of the studentised residuals |

## 2.3 Variance Decomposition and Analysis of Variance

| **Exercise 2.10** | |
|---|---|
| | What is the distribution of $F$, if $E(Y) \in \mathscr{M}_{X'}$ applies and $\varepsilon$ is distributed as $N(0, \sigma^2 I)$? |

| **Exercise 2.11** | |
|---|---|
| | Give an explicit formula for the $F$ statistics for analysis of variance in the one-way layout<br>$y_{ij} = \mu + \alpha_j + \varepsilon_{ij}$<br>in comparison to the homogeneous model<br>$y_{ij} = \mu + \varepsilon_{ij}$. |

The analysis of variance gives another representation and interpretation of linear models. For example, the regression result from (Example **??**) gives the following analysis of variance representation:

***Example* 2.3: Linear Model ANOVA Summary**

*Input*

```
summary(aov(lmres))
```

Output

```
            Df Sum Sq Mean Sq F value Pr(>F)
x            1 538022  538022    5619 <2e-16 ***
Residuals   98   9384      96
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

`anova()` can be used to compute analysis of variance tables for one or more fitted model objects.

***Example* 2.4: Linear Model ANOVA**

*Input*

```
anova(lm(y~x), lm(y~0+x))
```

Output

```
Analysis of Variance Table

Model 1: y ~ x
Model 2: y ~ 0 + x
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     98 9384.0
2     99 9425.2 -1   -41.283 0.4311  0.513
```

| **Exercise 2.12** | |
|---|---|
| | Analyse the output of `lm()` shown in Example 2.2 (page 17). Which terms can you interpret now? Give a written report. For which terms do you need more information? |

| **Exercise 2.13** | **One-Way Anova** |
|---|---|
| * | Write a function `oneway()` which takes a data table as an argument and performs a one-way analysis of variance as a test on difference between the columns. |
| * | Enhance `oneway()` by adding the necessary diagnostic plots. Which diagnostics are necessary? |

| **Exercise 2.14** | **Kiwi Hopp** |
|---|---|
| | The industrial enterprise Kiwi Inc.[1] wants to develop a new helicopter for the market. The helicopter design is rated by the time it stays in air before it touches ground[2] from a fixed starting height (ca. 2m). Figure 2.1, page 26, shows a design drawing. What are the factors that can affect the variability of the flight (sink) time? What are the factors that can affect the mean flight duration? |
| | Perform 30 test flights with a prototype and measure the time in 1/100s. (You will have to cooperate in pairs to carry out the measurements.) Would you consider the recorded times as normally distributed?<br>The requirement is that the mean flight duration reaches at least 2.4s. Does the prototype satisfy the requirement? |
| | Your task is to select a design for production. The variants under discussion are:<br>rotor width 45mm<br>rotor width 35mm<br>rotor width 45mm with an additional fold for stabilisation<br>rotor width 35mm with an additional fold for stabilisation<br>(cont.)→ |

[1] Following an idea of Alan Lee, Univ. Auckland, New Zealand.
[2] Kiwis cannot fly.

| **Exercise 2.14** | **Kiwi Hopp** (cont.) |
|---|---|
| | Your budget allows for about 40 test flights. (If you need more test flights, you should give good arguments for this.) Build 4 prototypes, perform the test flights and record the times. Find the design that achieves maximum flight duration. Generate a report. The report should contain the following details:<br>• a list of the observed data and a description of the experimental procedure<br>• suitable plots of the data for each of the designs<br>• an analysis of variance<br>• a clear summary of your conclusions<br><br>*Additional hints:* Randomise the sequence of your experiments. Reduce the variation by providing uniform conditions for the experiment (same height, same launch technique, etc.). |
| | The fold will result in additional production cost. Give an estimate of the gain that can be achieved by this additional cost. |

## 2.4 Simultaneous Inference

*2.4.1 Scheffé's Confidence Bands*

*2.4.2 Tukey's Confidence Intervals*

*Case Study: Titre Plates*

## 2.5 Beyond Linear Regression

*Transformations*

*2.5.1 Generalised Linear Models*

| **Exercise 2.15** | |
|---|---|
| | Write the normal densities in the exponential form (**??**). What is the natural statistics? What is the natural parametrisation? |

*2.5.2 Local Regression*

## 2.6 R Complements

*2.6.1 Complements: Discretisation*

*2.6.2 Complements: External Data*

*2.6.3 Complements: Testing Software*

| **Exercise 2.16** | |
|---|---|
| | For this series of exercises, let $y_i = a+bx_i+\varepsilon_i$ with $\varepsilon_i$ iid $\sim N(0,\sigma^2)$ and $x_i = i, i = 1,\ldots,10$. |
| | Choose a strategy to inspect `lm()` with regard to the parameter space $(a, b, \sigma^2)$.<br>Are there apparent cellular decompositions for the parameters $a, b, \sigma^2$?<br>What are the trivial cases? What are the asymptotics that apply?<br>Choose test points in the interior of each cell and on the boundaries.<br>Perform these tests and summarise the results. |
| | What are the symmetries/anti-symmetries that apply?<br>Check for these symmetries. |
| | Which invariant or covariate behaviour applies?<br>Check for these invariant or covariate behaviour. |

| **Exercise 2.17** | |
|---|---|
| | For this series of exercises, let $y_i = a + bx_i + \varepsilon_i$ with $\varepsilon_i$ iid $\sim N(0,\sigma^2)$. |
| | What are the extremal designs $(x_i)$? Check the behaviour of `lm()` for four extremal designs. |
| | Perform the tests from the last exercise, now with variable design. Summarise your results. |

| **Exercise 2.18** | |
|---|---|
| | For this series of exercises, let $y_i = a + bx_i + \varepsilon_i$ with $\varepsilon_i$ iid $\sim N(0, \sigma^2)$. |
| | Modify `lm()` to give a fail-safe function for simple linear models that checks deviations from the model assumptions as well. |

*2.6.4* R *Data Types*

*2.6.5 Classes and Polymorphic Functions*

*2.6.6 Extractor Functions*

**2.7 Statistical Summary**

**2.8 Literature and Additional References**

[1] Chambers, J.M.; Hastie, T.J. (eds.) (1992): *Statistical Models in S*. Chapman & Hall, New York.

[5] Jørgensen, B. (1993): *The Theory of Linear Models*. Chapman & Hall, New York.

[8] R Development Core Team (2004–2008): The R language definition.

[10] Sawitzki, G. (1994): Numerical Reliability of Data Analysis Systems. *Computational Statistics & Data Analysis* 18.2 (1994), 269–286. <http://www.statlab.uni-heidelberg.de/reports/>.

[11] Sawitzki, G. (1994):Report on the Numerical Reliability of Data Analysis Systems. *Computational Statistics & Data Analysis/SSN* 18.2 (1994) 289–301. <http://www.statlab.uni-heidelberg.de/reports/>.

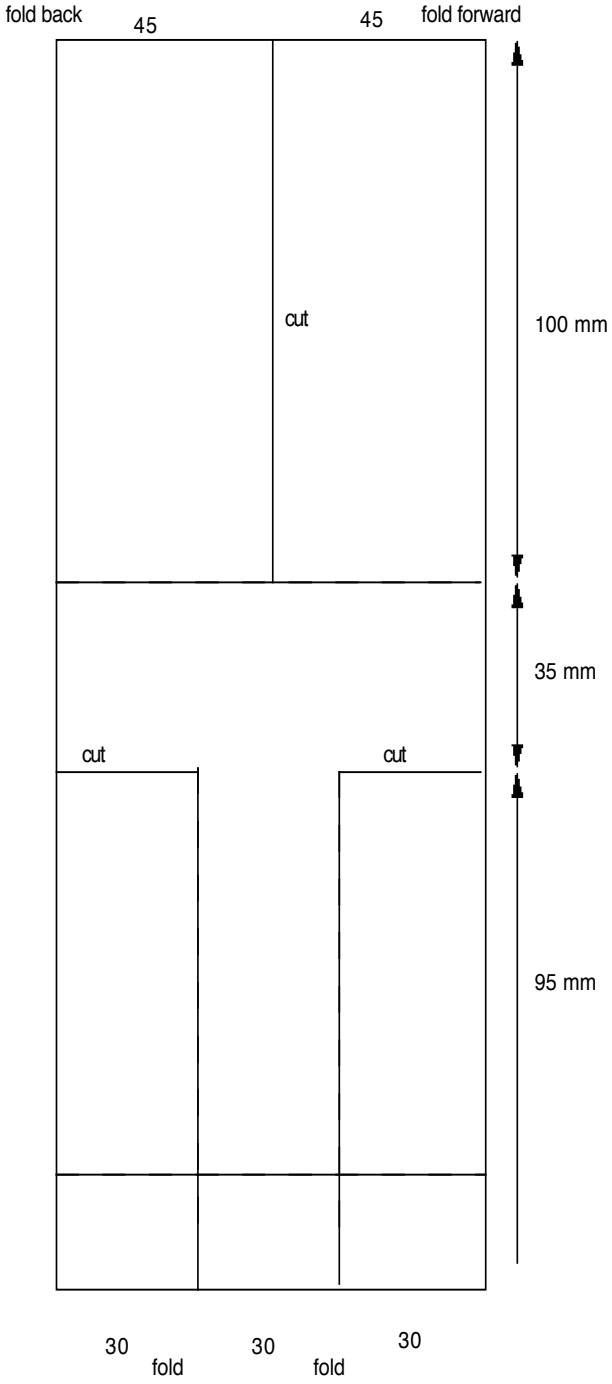



Figure 2.1 *KiwiHopp*

CHAPTER 3

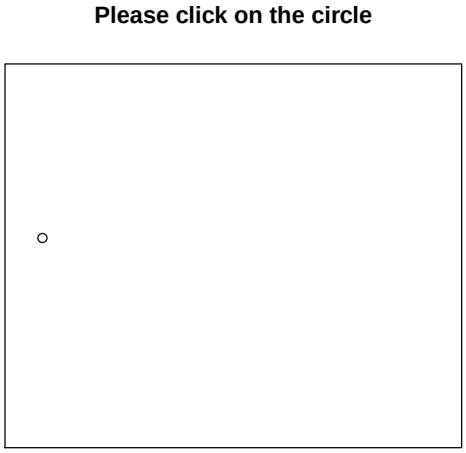
# Comparisons

We begin with the construction of a small gadget that will provide us with example data. The base is a reaction tester. We present a "random" point, wait for a mouse click on that point and record the position of the mouse pointer. To get a stable image for repeated activations, we fix the coordinate system.

***Example* 3.1: Interactive Location**

*Input*

```
plot(x = runif(1), y = runif(1),
     xlim = c(0, 1), ylim = c(0, 1),
     main = "Please click on the circle",
     xlab = '', ylab = '',
     axes = FALSE, frame.plot = TRUE)
xclick <- locator(1)
```




Now we wrap up the base function in a timer. We record the coordinates, try to measure the reaction time, and return the results as a list.

***Example* 3.2: Click Timing**

*Input*

```
click1 <- function(){
    x <- runif(1);y <- runif(1)
    plot(x = x, y = y, xlim = c(0, 1), ylim = c(0, 1),
        main = "Please click on the circle",
        xlab = '', ylab = '',
        axes = FALSE, frame.plot = TRUE)
    clicktime <- system.time(xyclick <- locator(1))
    list(timestamp = Sys.time(),
        x = x, y = y,
        xclick = xyclick$x, yclick = xyclick$y,
        tclick = clicktime[3])
}
```

For later processing we can integrate the list in a `data.frame` and extend this `data.frame` stepwise using `rbind`.

***Example* 3.3: Sequential Recording**

*Input*

```
dx <- as.data.frame(click1())
dx <- rbind(dx, data.frame(click1()))
dx
```

Output

```
                   timestamp      x      y xclick yclick tclick
elapsed  2014-04-23 21:54:22 0.9619 0.6249 0.9609 0.6250  1.944
elapsed1 2014-04-23 21:54:24 0.5477 0.2259 0.7410 0.4125  2.212
```

| **Exercise 3.1** | **Click Timing** |
|---|---|
| | Define a function `click(runs)` that repeats `click1()` a chosen number *runs* plus one times and returns the result as a `data.frame`. The additional first timing should be considered as a "warming up" and is not included in the following evaluations.<br>Select a number `runs`. Give reasons for your choice of `runs`. Execute `click(runs)` and store the result in a file using `write.table()`.<br>Display the distribution of the component `tclick` with the methods from Chapter 1 (distribution function, histogram, box-and-whisker plot). |

| Exercise 3.2 | Click Comparison |
|---|---|
| | Perform Exercise 3.1 using the right hand and then again using the left hand. Compare the empirical distributions of the timing data returned by `tclick()` for the right and left hand. |
| | The recorded data also contain information about the positions. Define a distance measure `dist` for the deviation. Give reasons for your definition. Perform a right/left comparison for `dist`.<br>For later analysis, store the results for the right hand and for the left hand in files. named `"clickright-xxxx"` and `" "clickright-xxxx"`, where xxxx is an identification of you choice. For example, use your initials, the date and some sequential number, such as in `"clickright-cs20050416-1"`. |

We concentrate on the comparison of two distributions only, for example, that of the results of two treatments. And we take a simple case: we assume that the observations are independent and identically distributed for each treatment. We use the index notation that is usual for the comparison of treatments in the two sample case.

$Y_{ij}$ independent identically distributed with distribution function $F_i$

$i = 1, 2$ treatments

$j = 1, \ldots, n_i$ observations in treatment group $i$.

How do we compare the observations in the treatment groups $i = 1, 2$? The (simple) linear models

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

consider only the case where the difference amounts to a shift $\Delta = \alpha_1 - \alpha_2$.

**Notation:** For a distribution with distribution function $F$ the family

$$F_a(x) = F(x - a)$$

is called the ***shift family*** for $F$. The parameter $a$ is called the shift or location parameter.

Speaking in terms of probabilities, the treatment can shift probability mass in quite different ways from what can be achieved by an additive shift term. We need more general ways to compare distributions. Shift families are not the only framework to consider.

**Notation:** A distribution with distribution function $F_1$ is ***stochastically smaller*** than a distribution with distribution function $F_2$ (in symbols, $F_1 \prec F_2$), if a variable distributed as $F_1$ takes rather smaller values than a variable distributed as $F_2$. This means that $F_1$ increases sooner.

$$F_1(x) \geq F_2(x) \; \forall x$$

and

$$F_1(x) > F_2(x) \text{ for at least one } x.$$

For shift families we have: If $a < 0$, then $F_a \prec F$. The shift results in a parallel shift of the distribution functions.

A typical result of the click comparison experiment (Exercise 3.2) is given in Figure 3.1. The response times for the right side are stochastically smaller than those for the left side. But the distributions do not belong to a common shift family, since the distribution functions are not parallel.

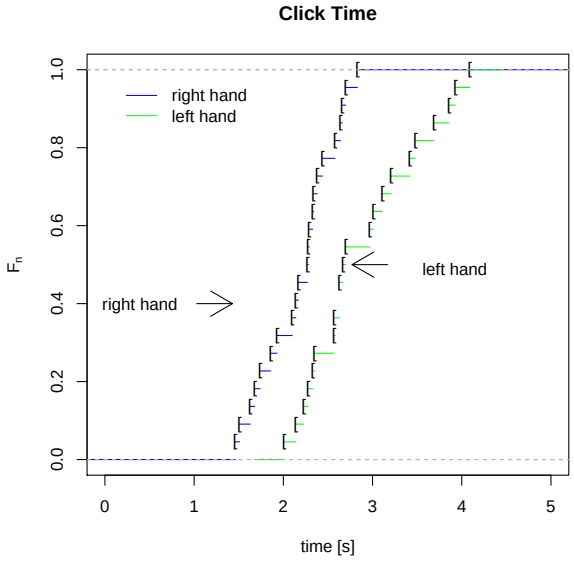



Figure 3.1 *Distribution functions for the right/left click time (samples from one person).*

| **Exercise 3.3** | **Stochastic Order** |
|---|---|
| | What does a $PP$ plot for $F_1$ against $F_2$ look like if $F_1 \prec F_2$? |
| | What does a $QQ$ plot for $F_1$ against $F_2$ look like if $F_1 \prec F_2$? |

| **Exercise 3.4** | |
|---|---|
| | The scale shift family for the $N(0,1)$ distribution are the $N(\mu, \sigma^2)$ distributions. Which $N(\mu, \sigma^2)$ distributions are stochastically smaller than the $N(0,1)$ distribution? Which are stochastically larger? Which distributions have an undefined order relation to $N(0,1)$? |

### 3.2 $QQ$ Plot, $PP$ Plot, and Comparison of Distributions

| **Exercise 3.5** | |
|---|---|
| | Use the $QQ$ plot to compare the results of the right/left `click` experiments. Summarise the results. |
| | Combine the right/left `tclick` data to a vector. Compare the $QQ$ plot with that of Monte Carlo samples taken from the joined vector.<br>*Hint:* You can draw random samples with `sample()`. With `par(mfrow = c(2, 2))` you arrange the display area so that it shows four plots at a time. |
| ** | For `sample()` use `replace = FALSE`. How do you have to apply `sample()` now to split the joint vector into two vectors with Monte Carlo samples? What differences do you expect in comparison to `replace = TRUE`? |

| **Exercise 3.6** | |
|---|---|
| | Find scale and shift parameters for the right/left `click` data so that, after using these parameters for transformation, the groups match as well as possible. Describe the differences using these parameters. Use a model formulation in terms of a linear model. |
| | Use the function `boxplot()` to display quartiles and tail behaviour.<br>Compare the information with the information you derived from the scale and shift parameters.<br>*Hint:* What corresponds to the shift (or location) parameter? What corresponds to the scale parameter? |

If representations such as visual representations in displays or numeric representations in summary statistics are affine invariant, scale and shift parameters can be ignored. If representations are not affine invariant, it is often helpful to estimate scale and shift

parameters first, then standardise the distributions, and only then to inspect the standardised distributions.

The potential problem with this is that we have to take into account the stochastic behaviour of the scale and shift parameter estimation. The usual way out is to be cautious and use "conservative" tests and robust estimators. The following function tries to transform scale and location to match a standard normal distribution.

```
ScaleShiftStd <- function (x) {
   xq <- quantile(x[!is.na(x)], c(0.25, 0.75))
   y <- qnorm(c(0.25, 0.75))
   slope <- diff(y)/diff(xq)
   (x-median(x, na.rm = FALSE)) * slope
   }
```

| **Exercise 3.7** | **Scale/Shift Standardisation** |
|---|---|
| | This algorithm is only appropriate for symmetric distributions.<br>Combine it with a power transformation as in Section 2.5 (page 23) to symmetrise a distribution and give an algorithm that can be applied to asymmetric transformations. |

| **Exercise 3.8** | **Two-Sample Monte Carlo Bands** |
|---|---|
| * | Modify the functions for the $PP$ plot and the $QQ$ plot so that Monte Carlo bands for the comparison of two samples are added. (Use a scale/shift standardisation for the $PP$ plot.)<br>For the bands, you can use an overlay of line plots.<br>*Hint:* Use the function `sample()` to generate random permutations. |

| **Exercise 3.9** | |
|---|---|
| ** | Augment the $PP$ plot and $QQ$ plot for the `click` experiments by permutation bands that cover 95% of the permutations. |
| ** | Generate new plots from the $PP$ plots and $QQ$ plots by adding Monte Carlo bands from permutations. Use an envelope of 19 Monte Carlo samples.<br>*Hint:* Use function `sample()` to draw a random sample of sample size $n_1$ from $x = (Y_{11}, \ldots, Y_{1n_1}, Y_{12}, \ldots, Y_{1n_2})$. |
| | *Hint:* See `help(sample)`. |

| **Exercise 3.10** | |
|---|---|
| * | Try to compare the properties of permutation bands, Monte Carlo bands and bootstrap bands on the hypothesis where $F_1 = F_2$. |

If not the distributions, but only single specified parameters are to be compared, an analogous strategy can be used. For example, if we focus on shift alternatives (that is $F_1$ and $F_2$ are from a shift family, $F_1(x) = F_2(x - a)$ for some $a$, we can take the mean (or the median) as the parameter of interest. The procedure given above can be used analogously to test the hypothesis that the distributions are not different ($a = 0$), based on the data.

| **Exercise 3.11** | |
|---|---|
| * | Formulate the strategies given above for intervals of single test statistic (example: mean) instead of bands.<br>*Hint:* Instead of the two mean values for both groups, can you use a single one-dimensional statistic? |

### *3.2.1 Kolmogorov-Smirnov Tests*

We also can use simulation to determine bands. In contrast to the one-sample case we do not have a given distribution from which to simulate. Under the hypothesis that the distributions $F$ and $G$ do not differ for independent observations the joined vector $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ is the vector of $n + m$ independent random numbers with identical distribution $F = G$. Given a data set, this relation can be used for simulation. Using a permutation $\pi$ of the indices from the vector $Z = (X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ a new vector $Z'$ with $Z'_i = Z_{\pi(i)}$ is generated. The first $n$ components are used as simulated values $(X'_i)_{i=1,\ldots,n}$, the remaining $m$ components as simulated values $(Y'_j)_{j=1,\ldots,m}$.

| **Exercise 3.12** | |
|---|---|
| * | Implement this algorithm and enhance the $PP$ plot by adding simulated $PP$ plots generated by a small number (19?) of permutations. |
| | Determine the permutation distribution of $\sup \lvert F_n - G_m \rvert$ from the simulation and calculate this statistic for the original data. Can you use this comparison to define a test procedure? |
| | The Kolmogorov-Smirnov test as implemented uses an approximation for the two-sample case. In our simulation we know that we simulate under the hypothesis. So any rejection we get is a false rejection, i.e., an error. Inspect the distribution of the error level under the simulated conditions. |

| **Exercise 3.13** | |
|---|---|
| | Use the $QQ$ plot for a pair-wise comparison of the results of the helicopter experiment from Chapter 2. Summarise your results. |

| **Exercise 3.14** | |
|---|---|
| | Inspect the implementation of `qqnorm()`. Implement an analogous function for the $PP$ plot and apply it to the helicopter data. |

### 3.3 Tests for Shift Alternatives

| **Exercise 3.15** | |
|---|---|
| * | Use a simulation to inspect the distribution of $\overline{Y}$, $\widehat{Var}(Y)$ and the $t$ statistic for $Y$ from a uniform distribution $U[0,1]$ with sample size $n = 1, \ldots, 10$. Compare the distributions from the simulation with the corresponding normal, $\chi^2$ resp. $t$ distribution. |
| | Use a simulation to inspect the distribution of $\overline{Y}$, $\widehat{Var}(Y)$ and the $t$ statistic for $Y$ from a mixture, consisting at 90% from an $N(0,1)$- and at 10% from an $N(0,10)$ distribution, with sample size $n = 1, \ldots, 10$. Compare the distributions from the simulation with the corresponding normal, $\chi^2$ resp. $t$ distribution. |

| **Exercise 3.16** | |
|---|---|
| | Use the Wilcoxon test to compare the results of the right/left `click` experiment.<br>Use both variants, the approximative test `wilcox.test()` and the exact Wilcoxon test `wilcox_test()`. |

| **Exercise 3.17** | **Click Project** |
|---|---|
| *** | In the the right/left `click` experiment several effects contribute to the response time. Some problems:<br>• The response time comprises reaction time, time for the large scale movement of the mouse, time for fine adjustment, etc.<br>• For the right/left movement in general a swivel of the hand is sufficient. For forward/backward movement in general a movement of the arm is necessary. It is not to be expected that both movements have a comparable statistical behaviour.<br>• Subsequent records may be affected by a training effect, or by a tiring effect.<br>Can you modify the experiment or the evaluation so that differences in the reaction time components can be investigated?<br>Can you modify the experiment or the evaluation so that differences in the precision of the position of the click can be investigated? |
| *** | Inspect and document for yourself the right/left differences in reaction time and precision. Summarise your results as a report. |

| **Exercise 3.18** | **Power Comparison** |
|---|---|
| | Use the shift/scale families of $N(0,1)$ and $t(3)$ and design a setting to compare the performance of the Wilcoxon test with that of the $t$-test for each of these families.<br>Perform the comparison in a simulation with sample sizes $n_1 = n_2 = 10, 20, 50, 100$ and summarise your results. |
| | Do an analogous comparison using simulation data from the log-normal distribution. |

## 3.4 A Road Map

## 3.5 Power and Confidence

### *3.5.1 Theoretical Power and Confidence*

### *3.5.2 Simulated Power and Confidence*

### *3.5.3 Quantile Estimation*

| **Exercise 3.19** | **Interquartile Interval** |
|---|---|
| | Write a function that calculates the coverage probability $n \mapsto P(med(X) \in I_Q)$ where $I_Q$ is the empirical interquartile interval.<br>*Hint:* Use (**??**). |
| | What is the minimal sample size so that the interquartile box covers the median with at least 90% confidence? |

### 3.6 Qualitative Features of Distributions

### 3.7 Statistical Summary

### 3.8 Literature and Additional References

[14] William N. Venables and Brian D. Ripley, B (2002): *Modern Applied Statistics with S.*
Springer, Heidelberg.

[13] William N. Venables, W.N.; and Brian D. Ripley (2000): *S Programming.*
Springer, Heidelberg.

[7] Rupert G. Miller (1981): *Simultaneous Statistical Inference.*
Springer, Heidelberg.

CHAPTER 4

# Dimensions 1, 2, 3, . . . , ∞

## 4.1 R Complements

## 4.2 Dimensions

## 4.3 Selections

## 4.4 Projections

### *4.4.1 Marginal Distributions and Scatter Plot Matrices*

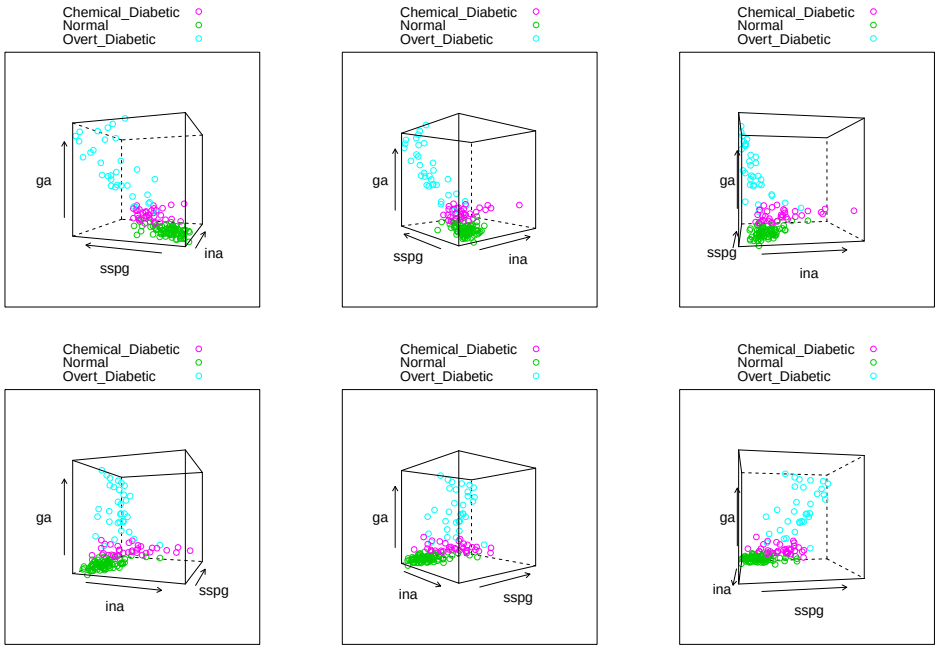
| **Exercise 4.1** | |
|---|---|
| | Generate a scatterplot matrix for the diabetes data set that shows a histogram of the variables in the diagonal panels.<br>*Hint:* See `help(pairs)`. |

### *4.4.2 Projection Pursuit*

*Input*

```
library("lattice")
diabcloud <- function(y, where, more = TRUE, ...) {
    print(cloud(ga ~ ina + sspg, data = chemdiab, groups = cc,
        screen = list(x = -90, y = y), distance = .4, zoom = .6,
        auto.key = TRUE, ...),
        split = c(where, 3, 2), more = more)
}
supsym <- trellis.par.get("superpose.symbol")
supsymold <- supsym
supsym$col = c("magenta", "green3", "cyan")
trellis.par.set("superpose.symbol" = supsym)
diabcloud(y = 70, where = c(1, 1))
diabcloud(y = 40, where = c(2, 1))
diabcloud(y = 10, where = c(3, 1))
```

```
diabcloud(y = -20, where = c(1, 2))
diabcloud(y = -50, where = c(2, 2))
diabcloud(y = -80, where = c(3, 2), more = FALSE)
trellis.par.set("superpose.symbol" = supsymold)
rm(diabcloud, supsymold, supsym)
```




*See Colour Figure* **??**.

| **Exercise 4.2** | |
|---|---|
| | Modify this example so that you get an impression of the three-dimensional structure. Try to use an animated sequence. You can use `sys.wait()` if it is available on your system to control the time sequence, or use `devAskNewPage()` to give interactive control for new images.<br>What is the difference between open diabetes and chemical diabetes?<br>How does the normal group compare to both diabetes groups? |

*4.4.3 Projections for Dimensions 1, 2, 3, ... 7*

*4.4.4 Parallel Coordinates*

| **Exercise 4.3** | |
|---|---|
| | For the `chemdiab` data set, prepare a (written!) report about the relation between the variables that you can recognise in the parallel coordinate plot. |
| | Instead of using `chemdiab[2:5]` you can specify the variables explicitly as `chemdiab[c(2, 3, 4, 5)]`. This gives you control over the order of the variables. Compare two different sequences of the variables and note (in writing!) your observations.<br>Which sequence of variables gives the simpler display?<br>Which relations between the variables are visible in both?<br>Which relations appear only in one of the arrangements? |

## 4.5 Sections, Conditional Distributions and Coplots

From an abstract point of view, sections are conditional distributions of the type $P(\cdot \mid X = x)$. But they are only reliable where the section defines a condition that has positive probability. To make the idea of restricting the view on conditional distributions applicable to data, we thicken the sections. Instead of considering conditional distributions of the type $P(\cdot \mid X = x)$ we consider $P(\cdot \mid \|X - x\| < \varepsilon)$, where $\varepsilon$ possibly can vary with $x$. In graphical representations of data this requires a series of plots showing only the part of the data set specified by the condition.

Statistically, projections lead to marginal distributions and sections to conditional distributions. In a certain sense, sections and projections are complementary: projections show structural features of low dimension. Sections are helpful to detect structural features of low codimension. For data analysis, both can be combined. The interplay of projections and sections is discussed in [2]. Like the dimension boundaries for projections there are boundaries for the codimension when using sections. We can only catch structures of small codimension. If the codimension is too large, a typical section is empty, hence it has no information.

As a first tool, R provides the possibility to analyse two variables ***conditioned*** on one or more additional variables. As a graphical display `coplot()` serves for this purpose. It is a variant of the plot matrix and shows in each panel the scatterplot of two variables, given the condition.

The coplot can be inspected for patterns. If the variables shown are stochastically independent of the conditioning variables, all plot elements show the same shape. The variables shown and the conditioning variables can then be de-coupled.

If the general shape coincides, but location and size vary, this hints at a (not necessarily linear) shift/scale relation. Additive models or variants of these can be used to model the relation between the variables shown and conditioning variables.

If the shape changes with varying condition, a major dependency structure or interaction may apply that needs more precise modelling.

**help(coplot)**

`coplot` *Conditioning Plots*

*Description*

This function produces two variants of the **conditioning plots discussed in the reference below.**

*Usage*

```
coplot(formula, data, given.values, panel = points, rows, columns,
       show.given = TRUE, col = par("fg"), pch = par("pch"),
       bar.bg = c(num = gray(0.8), fac = gray(0.95)),
       xlab = c(x.name, paste("Given :", a.name)),
       ylab = c(y.name, paste("Given :", b.name)),
       subscripts = FALSE,
       axlabels = function(f) abbreviate(levels(f)),
       number = 6, overlap = 0.5, xlim, ylim, ...)
co.intervals(x, number = 6, overlap = 0.5)
```

*Arguments*

| | |
|---|---|
| `formula` | a formula describing the form of conditioning plot. A formula of the form `y ~ x \| a` indicates that plots of `y` versus `x` should be produced conditional on the variable `a`. A formula of the form `y ~ x\| a * b` indicates that plots of `y` versus `x` should be produced conditional on the two variables `a` and `b`.<br>All three or four variables may be either numeric or factors. When `x` or `y` are factors, the result is almost as if `as.numeric()` was applied, whereas for factor `a` or `b`, the conditioning (and its graphics if `show.given` is true) are adapted. |
| `data` | a data frame containing values for any variables in the formula. By default the environment where `coplot` was called from is used. |
| `given.values` | a value or list of two values which determine how the conditioning on `a` and `b` is to take place.<br>When there is no `b` (i.e., conditioning only on `a`), usually this is a matrix with two columns each row of which gives an interval, to be conditioned on, but is can also be a single vector of numbers or a set of factor levels (if the variable being conditioned on is a factor). In this case (no `b`), the result of `co.intervals` can be used directly as `given.values` argument. |

| | |
|---|---|
| `panel` | a `function(x, y, col, pch, ...)` which gives the action to be carried out in each panel of the display. The default is `points`. |
| `rows` | the panels of the plot are laid out in a `rows` by `columns` array. `rows` gives the number of rows in the array. |
| `columns` | the number of columns in the panel layout array. |
| `show.given` | logical (possibly of length 2 for 2 conditioning variables): should conditioning plots be shown for the corresponding conditioning variables (default `TRUE`). |
| `col` | a vector of colors to be used to plot the points. If too short, the values are recycled. |
| `pch` | a vector of plotting symbols or characters. If too short, the values are recycled. |
| `bar.bg` | a named vector with components `"num"` and `"fac"` giving the background colors for the (shingle) bars, for **numeric and factor conditioning variables respectively.** |
| `xlab` | **character; labels to use for the x axis and the first conditioning variable. If only one label is given, it is used for the x axis and the default label is used for the conditioning variable.** |
| `ylab` | **character; labels to use for the y axis and any second conditioning variable.** |
| `subscripts` | **logical: if true the panel function is given an additional (third) argument `subscripts` giving the subscripts of the data passed to that panel.** |
| `axlabels` | **function for creating axis (tick) labels when x or y are factors.** |
| `number` | **integer; the number of conditioning intervals, for a and b, possibly of length 2. It is only used if the corresponding conditioning variable is not a `factor`.** |
| `overlap` | **numeric $< 1$; the fraction of overlap of the conditioning variables, possibly of length 2 for x and y direction. When overlap $< 0$, there will be *gaps* between the data slices.** |
| `xlim` | **the range for the x axis.** |
| `ylim` | **the range for the y axis.** |
| `...` | **additional arguments to the panel function.** |
| `x` | **a numeric vector.** |

*Details*

In the case of a single conditioning variable `a`, when both `rows` and `columns` are unspecified, a 'close to square' layout is chosen with `columns >= rows`.
In the case of multiple `rows`, the *order* of the panel plots is from the bottom and from the left (corresponding to increasing `a`, typically).

A panel function should not attempt to start a new plot, but just plot within a given coordinate system: thus `plot` and `boxplot` are not panel functions.
The rendering of arguments `xlab` and `ylab` is not controlled by `par` arguments `cex.lab` and `font.lab` even though they are plotted by `mtext` rather than `title`.

*Value*

`co.intervals(., number, .)` returns a (`number` $\times$ 2) `matrix`, say `ci`, where `ci[k,]` is the `range` of `x` values for the `k`-th interval.

*References*

Chambers, J. M. (1992) *Data for models.* Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.
Cleveland, W. S. (1993) *Visualizing Data.* New Jersey: Summit Press.

*See Also*

`pairs`, `panel.smooth`, `points`.

*Examples*

```
## Tonga Trench Earthquakes
coplot(lat ~ long | depth, data = quakes)
given.depth <- co.intervals(quakes$depth, number = 4, overlap = .1)
coplot(lat ~ long | depth, data = quakes, given.v = given.depth, rows = 1)

## Conditioning on 2 variables:
ll.dm <- lat ~ long | depth * mag
coplot(ll.dm, data = quakes)
coplot(ll.dm, data = quakes, number = c(4, 7), show.given = c(TRUE, FALSE))
coplot(ll.dm, data = quakes, number = c(3, 7),
       overlap = c(-.5, .1)) # negative overlap DROPS values

## given two factors
Index <- seq(length = nrow(warpbreaks)) # to get nicer default labels
coplot(breaks ~ Index | wool * tension, data = warpbreaks,
       show.given = 0:1)
coplot(breaks ~ Index | wool * tension, data = warpbreaks,
       col = "red", bg = "pink", pch = 21,
       bar.bg = c(fac = "light blue"))

## Example with empty panels:
with(data.frame(state.x77), {
coplot(Life.Exp ~ Income | Illiteracy * state.region, number = 3,
       panel = function(x, y, ...) panel.smooth(x, y, span = .8, ...))
## y ~ factor -- not really sensible, but 'show off':
coplot(Life.Exp ~ state.region | Income * state.division,
```

```
        panel = panel.smooth)
})
```

| **Exercise 4.4** | **Earthquakes** |
|---|---|
| | Analyse the "quakes" data set.<br>Summarise your results in a report.<br>Try to specify a formal model. |
| | How is the geographic position related to the depth? |
| | Can you identify relations between depth and magnitude of the earthquake? (You may have to choose a different model formula for the plots.) |

| Exercise 4.5 | Iris Classification |
|---|---|
| | Use the methods from Section 4.4 and 4.5 to inspect the data set. Can you see classification rules that give a classification of the three species to a large extent? |

## 4.7 Higher Dimensions

### *4.7.1 Linear Case*

*Partial Residuals and Added Variable Plots*

| Exercise 4.6 | Added Variables |
|---|---|
| | Modify the following function `pairslm()` so that it calculates the residuals of the regression of all original variables in matrix `x` by regression after the new variable `x$fit`, and produces a scatterplot matrix of these residuals.<br>`pairslm <- function(model, x, ... )`<br>`{ x$fit <- lm(model, x)$fit`<br>`pairs(x, ...) }`<br>Add title, legends, etc.<br>Use the "trees" data set as an example. |

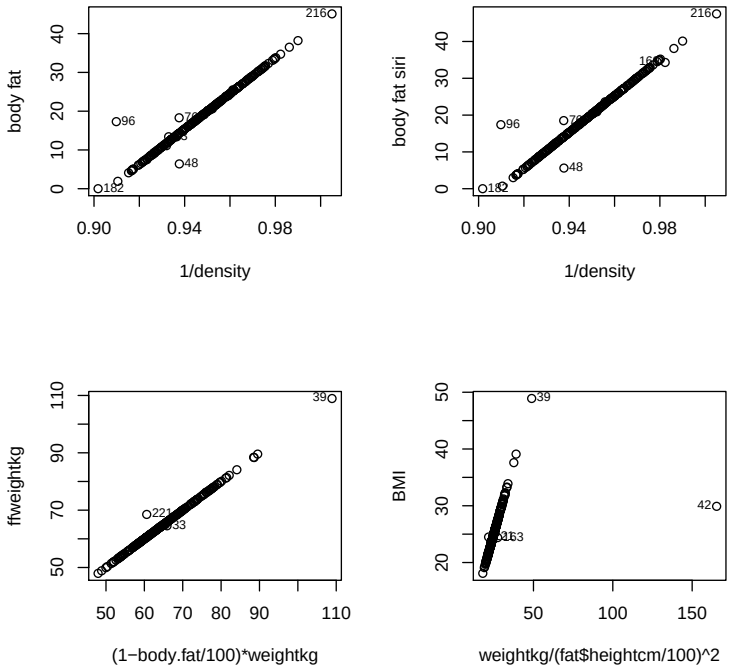
### *4.7.2 Non-Linear Case*

*Example: Cusp Non-Linearity*

### *4.7.3 Case Study: Melbourne Temperature Data*

### *4.7.4 Curse of Dimensionality*

### *4.7.5 Case Study: Body Fat*

| **Exercise 4.7** | |
|---|---|
| | Use functions *plot()*, *identify()*, and *text.id()* to generate the following output:   |

| **Exercise 4.8** | |
|---|---|
| * | Use<br>`library(leaps)`<br>`lm.reg <- regsubsets(body.fat ~ age + BMI + neck + chest + abdomen + hip + thigh + knee + ankle + bicep + forearm + wrist + weightkg + heightcm, data = fat)`<br>and inspect the result with<br>`summary(lm.reg)`<br>`plot(lm.reg, scale = "r2")`<br>`plot(lm.reg, scale = "bic"`<br>`plot(lm.reg, scale = "Cp")`<br>*Hint:* See *help(plot.regsubsets)*. |
| * | Use the function *leaps()* for model selection. |

| **Exercise 4.9** | |
|---|---|
| | Remove the obvious outliers and rearrange the variables starting from the body volume so that on average the correlation between subsequent variables is maximized. |

| **Exercise 4.10** | **Think !** |
|---|---|
| | Draw a sketch of a member doll that shows which body geometry features are represented by the next principal component *PC4*, . . . , *PC10*.<br>For a start, you can concentrate on the signs of the variable weights. |

| **Exercise 4.11** | |
|---|---|
| * | Extend the variables by other volume-related variables in the model given above. Do you gain precision? |
| ** | Try to include the variable *age* in the model. How exactly do you include *age* in the model? |
| ** | The function *mvr()* in *library(pls)* [15] is available to perform a regression based on principal components. Use this function for regression. What is the difference between this estimation and the usual least squares regression? |

For model construction, we used only the training part of the data. The quality of the model derived now can be checked using the evaluation part. This can be done using function *predict.lm()*, which applies a model estimated with *lm()* to a new data set with analogous structure, for example:

*Input*

```
fat.eval <- fat[fat$train == FALSE, ]
pred <- predict.lm(lm.volf, fat.eval, se.fit = TRUE)
```

| **Exercise 4.12** | |
|---|---|
| * | Estimate the precision of the model using the evaluation part of the data. |
| * | Carry out a regression diagnostics of the model derived, using the evaluation part of the data. |

**4.8 High Dimensions**

**4.9 Statistical Summary**

# References


[1] John M. Chambers and Trevor J. Hastie. *Statistical Models in S*. Chapman & Hall, London, 1992.

[2] George W. Furnas and Andreas Buja. Prosection views: Dimensional inference through sections and projections. *J. Comput. Graph. Statist.*, 3(4):323–385, 1994.

[3] Peter Gänßler and Winfried Stute. *Wahrscheinlichkeitstheorie*. Springer, Heidelberg, 1977.

[4] Robert Gentleman and Ross Ihaka. Lexical scope and statistical computing. *Journal of Computational and Graphical Statistics*, 9:491–508, 2000.

[5] Bent Jørgensen. *The Theory of Linear Models*. Chapman & Hall, New York, 1993.

[6] Paul Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability*, 18(3):1269–1283, July 1990.

[7] Rupert G. Miller. *Simultaneous Statistical Inference*. Springer, New York, 1981.

[8] R Development Core Team. *The R language definition*, 2008.

[9] R Development Core Team. *Writing R Extensions*, 2008.

[10] Günther Sawitzki. Numerical reliability of data analysis systems. *Computational Statistics & Data Analysis*, 18(2):269–286, 1994.

[11] Günther Sawitzki. Report on the numerical reliability of data analysis systems. *Computational Statistics & Data Analysis*, 18(2):289–301, 1994.

[12] Galen R. Shorack and Jon A. Wellner. *Empirical Processes with Applications to Statistics*. Wiley, New York, 1986.

[13] William N. Venables and Brian D. Ripley. *S Programming*. Statistics and Computing. Springer, New York, 2000.

[14] William N. Venables and Brian D. Ripley. *Modern Applied Statistics with S*. Springer, Heidelberg, fourth edition, 2002.

[15] Ron Wehrens and Bjørn-Helge Mevik. *pls: Partial Least Squares Regression (PLSR) and Principal Component Regression (PCR)*, 2007. R package version 2.1-0.

# Subject Index