A

Obtaining and installing R and the ISwR package

The way to obtain R is to download it from one of the CRAN (Comprehensive R Archive Network) sites. The main site is

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

It has a number of mirror sites worldwide that may be closer to you and give faster download times.

Installation details tend to vary over time, so you should read the accompanying documents and any other information offered on CRAN.

Binary distributions

As of this writing, the version for recent variants of Microsoft Windows comes as a single `R-2.6.2-win32.exe` file, on which you simply double-click with the mouse and then follow the on-screen instructions. When the process is completed, you will have an entry under Programs on the Start menu for invoking R, as well as a desktop icon.

For Linux distributions that use the RPM package format (primarily RedHat, Fedora, and SUSE), `.rpm` files of R and the recommended add-on packages can be installed using the `rpm` command and the respective system software management tools. Fedora now has R in its standard repositories, and it is also in the repository of openSUSE.org. Debian

packages can be accessed through APT, the Debian package maintenance tool, as can packages for Ubuntu (in both cases, make sure that you get the r-recommended package). Further details are in the FAQ.

For the Macintosh platforms, only OS X 10.2 and above are supported. Installation is by downloading the disk image R-2.6.2.dmg and double-clicking the "R.mpkg" icon found inside it.

Installation from source

Installation of Rfrom source code is possible on all supported platforms, although not quite trivial on Windows, mainly because the build environment is not part of the system. On Unix-like systems (Macintosh OS X included), the process can be as simple as unpacking the sources and writing

```
./configure
make
make install
```

The above works on widely used platforms, provided that the relevant compilers and support libraries are installed. If your system is more esoteric or you want to use special compilers or libraries, then you may need to dig deeper.

For Windows, the directory src/gnuwin32 has an INSTALL file with detailed information about the procedure to follow.

Package installation

To work through the examples and exercises in this book, you should install the ISwR package, which contains the data sets.

Assuming that you are connected to the Internet, you can start R and from the Windows and Macintosh versions use their convenient menu interfaces.

On other platforms, you can type

```
install.packages("ISwR")
```

This will give off a harmless warning and install the package in the default location.

On Unix and Linux systems you will need superuser permissions to install in the system location. Similarly, you may require administrator access on some Windows versions.

Otherwise you can set up a private library directory and install into that. Set the R_LIBS environment variable to use your private library subsequently. Further details can be found on the help page for library.

If your R machine is not connected to the Internet, you can also download the package as a file via a different computer. For Windows and Macintosh, you should get the binary package (.zip or .tgz extension) and then installation from a local file is possible via a menu entry. For Unix and Linux, you can issue the following at the shell prompt (the -l option allows you to give a private library if needed):

```
R CMD INSTALL ISwR
```

More information

Information and further Internet resources for R can be obtained from CRAN and the R homepage at

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

Notice in particular the mailing lists, the user-contributed documents, and the FAQs.

B

Data sets in the ISwR package[1]

IgM *Immunoglobulin G*

Description

Serum IgM in 298 children aged 6 months to 6 years.

Usage

```
IgM
```

Format

A single numeric vector (g/l).

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 3.2, Chapman & Hall.

[1] Reproduced with permission from the documentation files in the ISwR package.

Examples

```
stripchart(IgM,method="stack")
```

`alkfos` *Alkaline phosphatase data*

Description

Repeated measurements of alkaline phosphatase in a randomized trial of Tamoxifen treatment of breast cancer patients.

Usage

```
alkfos
```

Format

A data frame with 43 observations on the following 8 variables.

`grp` a numeric vector, group code (1=placebo, 2=Tamoxifen).
`c0` a numeric vector, concentration at baseline.
`c3` a numeric vector, concentration after 3 months.
`c6` a numeric vector, concentration after 6 months.
`c9` a numeric vector, concentration after 9 months.
`c12` a numeric vector, concentration after 12 months.
`c18` a numeric vector, concentration after 18 months.
`c24` a numeric vector, concentration after 24 months.

Source

Original data.

References

B. Kristensen et al. (1994), Tamoxifen and bone metabolism in postmenopausal low-risk breast cancer patients: a randomized study. *Journal of Clinical Oncology*, 12(2):992–997.

`ashina` *Ashina's crossover trial*

Description

The `ashina` data frame has 16 rows and 3 columns. It contains data from a crossover trial for the effect of an NO synthase inhibitor on headaches. Visual analog scale recordings of pain levels were made at baseline and at five time points after infusion of the drug or placebo. A score was calculated as the sum of the differences from baseline. Data were recorded during two sessions for each patient. Six patients were given treatment on the first occasion and the placebo on the second. Ten patients had placebo first and then treatment. The order of treatment and the placebo was randomized.

Usage

```
ashina
```

Format

This data frame contains the following columns:

`vas.active` a numeric vector, summary score when given active substance.

`vas.plac` a numeric vector, summary score when given placebo treatment.

`grp` a numeric vector code, 1: placebo first, 2: active first.

Source

Original data.

References

M.Ashina et al. (1999), Effect of inhibition of nitric oxide synthase on chronic tension-type headache: a randomised crossover trial. *Lancet* 353, 287–289

Examples

```
plot(vas.active~vas.plac,pch=grp,data=ashina)
abline(0,1)
```

`bcmort` *Breast cancer mortality*

Description

Danish study on the effect of screening for breast cancer.

Usage

```
bcmort
```

Format

A data frame with 24 observations on the following 4 variables.

`age` a factor with levels `50-54`, `55-59`, `60-64`, `65-69`, `70-74`, and `75-79`.

`cohort` a factor with levels `Study gr.`, `Nat.ctr.`, `Hist.ctr.`, and `Hist.nat.ctr.`.

`bc.deaths` a numeric vector, number of breast cancer deaths.

`p.yr` a numeric vector, person-years under study.

Details

Four cohorts were collected. The "study group" consists of the population of women in the appropriate age range in Copenhagen and Frederiksberg after the introduction of routine mammography screening. The "national control group" consisted of the population in the parts of Denmark in which routine mammography screening was not available. These two groups were both collected in the years 1991–2001. The "historical control group" and the "historical national control group" are similar cohorts from 10 years earlier (1981–1991), before the introduction of screening in Copenhagen and Frederiksberg. The study group comprises the entire population, not just those accepting the invitation to be screened.

Source

A.H. Olsen et al. (2005), Breast cancer mortality in Copenhagen after introduction of mammography screening. *British Medical Journal*, 330: 220–222.

bp.obese — *Obesity and blood pressure*

Description

The bp.obese data frame has 102 rows and 3 columns. It contains data from a random sample of Mexican-American adults in a small California town.

Usage

```
bp.obese
```

Format

This data frame contains the following columns:

sex a numeric vector code, 0: male, 1: female.
obese a numeric vector, ratio of actual weight to ideal weight from New York Metropolitan Life Tables.
bp a numeric vector,systolic blood pressure (mm Hg).

Source

B.W. Brown and M. Hollander (1977), *Statistics: A Biomedical Introduction,* Wiley.

Examples

```
plot(bp~obese,pch = ifelse(sex==1, "F", "M"), data = bp.obese)
```

caesarean — *Caesarean section and maternal shoe size*

Description

The table caesar.shoe contains the relation between caesarean section and maternal shoe size (UK sizes!).

Usage

```
caesar.shoe
```

Format

A matrix with two rows and six columns.

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 10.1, Chapman & Hall.

Examples

```
prop.trend.test(caesar.shoe["Yes",],margin.table(caesar.shoe,2))
```

`coking` *Coking data*

Description

The `coking` data frame has 18 rows and 3 columns. It contains the time to coking in an experiment where the oven width and temperature were varied.

Usage

```
coking
```

Format

This data frame contains the following columns:

`width` a factor with levels `4`, `8`, and `12`, giving the oven width in inches.

`temp` a factor with levels `1600` and `1900`, giving the temperature in Fahrenheit.

`time` a numeric vector, time to coking.

Source

R.A. Johnson (1994), *Miller and Freund's Probability and Statistics for Engineers*, 5th ed., Prentice-Hall.

Examples

```
attach(coking)
matplot(tapply(time,list(width,temp),mean))
detach(coking)
```

`cystfibr` *Cystic fibrosis lung function data*

Description

The `cystfibr` data frame has 25 rows and 10 columns. It contains lung function data for cystic fibrosis patients (7–23 years old).

Usage

```
cystfibr
```

Format

This data frame contains the following columns:

`age` a numeric vector, age in years.
`sex` a numeric vector code, 0: male, 1:female.
`height` a numeric vector, height (cm).
`weight` a numeric vector, weight (kg).
`bmp` a numeric vector, body mass (% of normal).
`fev1` a numeric vector, forced expiratory volume.
`rv` a numeric vector, residual volume.
`frc` a numeric vector, functional residual capacity.
`tlc` a numeric vector, total lung capacity.
`pemax` a numeric vector, maximum expiratory pressure.

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 12.11, Chapman & Hall.

References

O'Neill et al. (1983), The effects of chronic hyperinflation, nutritional status, and posture on respiratory muscle strength in cystic fibrosis, *Am. Rev. Respir. Dis.*, 128:1051–1054.

eba1977	*Lung cancer incidence in four Danish cities 1968–1971*

Description

This data set contains counts of incident lung cancer cases and population size in four neighbouring Danish cities by age group.

Usage

```
eba1977
```

Format

A data frame with 24 observations on the following 4 variables:

city a factor with levels Fredericia, Horsens, Kolding, and Vejle.

age a factor with levels 40-54, 55-59, 60-64, 65-69, 70-74, and 75+.

pop a numeric vector, number of inhabitants.

cases a numeric vector, number of lung cancer cases.

Details

These data were "at the center of public interest in Denmark in 1974", according to Erling Andersen's paper. The city of Fredericia has a substantial petrochemical industry in the harbour area.

Source

E.B. Andersen (1977), Multiplicative Poisson models with unequal cell rates, *Scandinavian Journal of Statistics*, 4:153–158.

References

J. Clemmensen et al. (1974), *Ugeskrift for Læger*, pp. 2260–2268.

`energy` *Energy expenditure*

Description

The `energy` data frame has 22 rows and 2 columns. It contains data on the energy expenditure in groups of lean and obese women.

Usage

```
energy
```

Format

This data frame contains the following columns:

`expend` a numeric vector, 24 hour energy expenditure (MJ).
`stature` a factor with levels `lean` and `obese`.

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 9.4, Chapman & Hall.

Examples

```
plot(expend~stature,data=energy)
```

`ewrates` *Rates of lung and nasal cancer mortality, and total mortality.*

Description

England and Wales mortality rates from lung cancer, nasal cancer, and all causes, 1936–1980. The 1936 rates are repeated as 1931 rates in order to accommodate follow-up for the `nickel` study.

Usage

```
ewrates
```

Format

A data frame with 150 observations on the following 5 variables:

year calendar period, 1931: 1931–35, 1936: 1936–40, ...
age age class, 10: 10–14, 15:15–19, ...
lung lung cancer mortality rate per 1 million person-years
nasal nasal cancer mortality rate per 1 million person-years
other all cause mortality rate per 1 million person-years

Details

Taken from the "Epi" package by Bendix Carstensen et al.

Source

N.E. Breslow, and N. Day (1987). *Statistical Methods in Cancer Research. Volume II: The Design and Analysis of Cohort Studies*, Appendix IX. IARC Scientific Publications, Lyon.

---

fake.trypsin *Trypsin by age groups*

Description

The `trypsin` data frame has 271 rows and 3 columns. Serum levels of immunoreactive trypsin in healthy volunteers (faked!).

Usage

```
fake.trypsin
```

Format

This data frame contains the following columns:

trypsin a numeric vector, serum-trypsin in ng/ml.
grp a numeric vector, age coding. See below.
grpf a factor with levels `1`: age 10–19, `2`: age 20–29, `3`: age 30–39, `4`: age 40–49, `5`: age 50–59, and `6`: age 60–69.

Details

Data have been simulated to match given group means and SD.

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 9.12, Chapman & Hall.

Examples

```
plot(trypsin~grp, data=fake.trypsin)
```

---

graft.vs.host *Graft versus host disease*

Description

The gvhd data frame has 37 rows and 7 columns. It contains data from patients receiving a nondepleted allogenic bone marrow transplant with the purpose of finding variables associated with the development of acute graft-versus-host disease.

Usage

```
graft.vs.host
```

Format

This data frame contains the following columns:

pnr a numeric vector patient number.
rcpage a numeric vector, age of recipient (years).
donage a numeric vector, age of donor (years).
type a numeric vector, type of leukaemia coded 1: AML, 2: ALL, 3: CML for acute myeloid, acute lymphatic, and chronic myeloid leukaemia.
preg a numeric vector code indicating whether donor has been pregnant. 0: no, 1: yes.
index a numeric vector giving an index of mixed epidermal cell-lymphocyte reactions.
gvhd a numeric vector code, graft-versus-host disease, 0: no, 1: yes.
time a numeric vector, follow-up time
dead a numeric vector code, 0: no (censored), 1: yes

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Exercise 12.3, Chapman & Hall.

Examples

```
plot(jitter(gvhd,0.2)~index,data=graft.vs.host)
```

`heart.rate` *Heart rates after enalaprilat*

Description

The `heart.rate` data frame has 36 rows and 3 columns. It contains data for nine patients with congestive heart failure before and shortly after administration of enalaprilat, in a balanced two-way layout.

Usage

```
heart.rate
```

Format

This data frame contains the following columns:

`hr` a numeric vector, heart rate in beats per minute.
`subj` a factor with levels `1` to `9`.
`time` a factor with levels `0` (before), `30`, `60`, and `120` (minutes after administration).

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 12.2, Chapman & Hall.

Examples

```
evalq(interaction.plot(time,subj,hr), heart.rate)
```

`hellung` *Growth of Tetrahymena cells*

Description

The `hellung` data frame has 51 rows and 3 columns. diameter and concentration of *Tetrahymena* cells with and without glucose added to growth medium.

Usage

```
hellung
```

Format

This data frame contains the following columns:

glucose a numeric vector code, 1: yes, 2: no.
conc a numeric vector, cell concentration (counts/ml).
diameter a numeric vector, cell diameter (μm).

Source

D. Kronborg and L.T. Skovgaard (1990), *Regressionsanalyse*, Table 1.1, FADLs Forlag (in Danish).

Examples

```
plot(diameter~conc,pch=glucose,log="xy",data=hellung)
```

---

intake *Energy intake*

Description

The intake data frame has 11 rows and 2 columns. It contains paired values of energy intake for 11 women.

Usage

```
intake
```

Format

This data frame contains the following columns:

pre a numeric vector, premenstrual intake (kJ).
post a numeric vector, postmenstrual intake (kJ).

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 9.3, Chapman & Hall.

Examples

```
plot(intake$pre, intake$post)
```

`juul` *Juul's IGF data*

Description

The `juul` data frame has 1339 rows and 6 columns. It contains a reference sample of the distribution of insulin-like growth factor (IGF-I), one observation per subject in various ages, with the bulk of the data collected in connection with school physical examinations.

Usage

```
juul
```

Format

This data frame contains the following columns:

`age` a numeric vector (years).
`menarche` a numeric vector. Has menarche occurred (code 1: no, 2: yes)?
`sex` a numeric vector (1: boy, 2: girl).
`igf1` a numeric vector, insulin-like growth factor (μg/l).
`tanner` a numeric vector, codes 1–5: Stages of puberty ad modum Tanner.
`testvol` a numeric vector, testicular volume (ml).

Source

Original data.

Examples

```
plot(igf1~age, data=juul)
```

`juul2` *Juul's IGF data, extended version*

Description

The `juul2` data frame has 1339 rows and 8 columns; extended version of `juul`.

Usage

```
juul2
```

Format

This data frame contains the following columns:

`age` a numeric vector (years).
`height` a numeric vector (cm).
`menarche` a numeric vector. Has menarche occurred (code 1: no, 2: yes)?
`sex` a numeric vector (1: boy, 2: girl).
`igf1` a numeric vector, insulin-like growth factor (μg/l).
`tanner` a numeric vector, codes 1–5: Stages of puberty ad modum Tanner.
`testvol` a numeric vector, testicular volume (ml).
`weight` a numeric vector, weight (kg).

Source

Original data.

Examples

```
plot(igf1~age, data=juul2)
```

`kfm` *Breast-feeding data*

Description

The `kfm` data frame has 50 rows and 7 columns. It was collected by Kim Fleischer Michaelsen and contains data for 50 infants of age approximately 2 months. They were weighed immediately before and

after each breast feeding. and the measured intake of breast milk was registered along with various other data.

Usage

```
kfm
```

Format

This data frame contains the following columns:

no a numeric vector, identification number.
dl.milk a numeric vector, breast-milk intake (dl/24h).
sex a factor with levels boy and girl.
weight a numeric vector, weight of child (kg).
ml.suppl a numeric vector, supplementary milk substitute (ml/24h).
mat.weight a numeric vector, weight of mother (kg).
mat.height a numeric vector, height of mother (cm).

Note

The amount of supplementary milk substitute refers to a period before the data collection.

Source

Original data.

Examples

```
plot(dl.milk~mat.height,pch=c(1,2)[sex],data=kfm)
```

---

lung — *Methods for determining lung volume*

Description

The lung data frame has 18 rows and 3 columns. It contains data on three different methods of determining human lung volume.

Usage

```
lung
```

Format

This data frame contains the following columns:

volume a numeric vector, measured lung volume.
method a factor with levels A, B, and C.
subject a factor with levels 1–6.

Source

Anon. (1977), Exercises in Applied Statistics, Exercise 4.15, Dept. of Theoretical Statistics, Aarhus University.

---

malaria *Malaria antibody data*

Description

The malaria data frame has 100 rows and 4 columns.

Usage

```
malaria
```

Format

This data frame contains the following columns:

subject subject code.
age age in years.
ab antibody level.
mal a numeric vector code, Malaria: 0: no, 1: yes.

Details

A random sample of 100 children aged 3–15 years from a village in Ghana. The children were followed for a period of 8 months. At the beginning of the study, values of a particular antibody were assessed. Based on observations during the study period, the children were categorized into two groups: individuals with and without symptoms of malaria.

Source

Unpublished data.

Examples

```
summary(malaria)
```

`melanom` *Survival after malignant melanoma*

Description

The `melanom` data frame has 205 rows and 7 columns. It contains data relating to the survival of patients after an operation for malignant melanoma, collected at Odense University Hospital by K.T. Drzewiecki.

Usage

```
melanom
```

Format

This data frame contains the following columns:

`no` a numeric vector, patient code.
`status` a numeric vector code, survival status; 1: dead from melanoma, 2: alive, 3: dead from other cause.
`days` a numeric vector, observation time.
`ulc` a numeric vector code, ulceration; 1: present, 2: absent.
`thick` a numeric vector, tumor thickness (1/100 mm).
`sex` a numeric vector code; 1: female, 2: male.

Source

P.K. Andersen, Ø. Borgan, R.D. Gill, and N. Keiding (1991), *Statistical Models Based on Counting Processes*, Appendix 1, Springer-Verlag.

Examples

```
require(survival)
plot(survfit(Surv(days,status==1),data=melanom))
```

`nickel` *Nickel smelters in South Wales*

Description

The data concern a cohort of nickel smelting workers in South Wales, with information on exposure, follow-up period, and cause of death.

Usage

```
nickel
```

Format

A data frame containing 679 observations of the following 7 variables:

`id` subject identifier (numeric).
`icd` ICD cause of death if dead, 0 otherwise (numeric).
`exposure` exposure index for workplace (numeric)
`dob` date of birth (numeric).
`age1st` age at first exposure (numeric).
`agein` age at start of follow-up (numeric).
`ageout` age at end of follow-up (numeric).

Details

Taken from the "Epi" package by Bendix Carstensen et al. For comparison purposes, England and Wales mortality rates (per 1,000,000 per annum) from lung cancer (ICDs 162 and 163), nasal cancer (ICD 160), and all causes, by age group and calendar period, are supplied in the data set `ewrates`.

Source

N.E. Breslow and N. Day (1987). *Statistical Methods in Cancer Research. Volume II: The Design and Analysis of Cohort Studies*, IARC Scientific Publications, Lyon.

nickel.expand *Nickel smelters in South Wales, expanded*

Description

The data concern a cohort of nickel smelting workers in South Wales, with information on exposure, follow-up period, and cause of death, as in the nickel data. This version has follow-up times split according to age groups and is merged with the mortality rates in ewrates.

Usage

```
nickel.expand
```

Format

A data frame with 3724 observations on the following 12 variables:

agr age class: 10: 10–14, 15: 15–19, ...
ygr calendar period, 1931: 1931–35, 1936: 1936–40, ...
id subject identifier (numeric).
icd ICD cause of death if dead, 0 otherwise (numeric).
exposure exposure index for workplace (numeric).
dob date of birth (numeric).
age1st age at first exposure (numeric).
agein age at start of follow-up (numeric).
ageout age at end of follow-up (numeric).
lung lung cancer mortality rate per 1 million person-years.
nasal nasal cancer mortality rate per 1 million person-years.
other all cause mortality rate per 1 million person-years.

Source

Computed from nickel and ewrates data sets.

philion *Dose response data*

Description

Four small experiments with the purpose of estimating the EC50 of a biological dose-response relation.

Usage

```
philion
```

Format

A data frame with 30 observations on the following 3 variables:

`experiment` a numeric vector; codes 1 through 4 denote the experiment number.

`dose` a numeric vector, the dose.

`response` a numeric vector, the response (counts).

Details

These data were discussed on the R mailing lists, initially suggesting a log-linear Poisson regression, but actually a relation like $y = y_{\max}/(1 + (x/\beta)^{\alpha})$ is more suitable.

Source

Original data from Vincent Philion, IRDA, Québec.

References

```
http://tolstoy.newcastle.edu.au/R/help/03b/1121.html
```

`react` *Tuberculin reactions*

Description

The numeric vector `react` contains differences between two nurses' determinations of 334 tuberculin reaction sizes.

Usage

```
react
```

Format

A single vector, differences between reaction sizes in mm.

Source

Anon. (1977), *Exercises in Applied Statistics*, Exercise 2.9, Dept. of Theoretical Statistics, Aarhus University.

Examples

```
hist(react) # not good because of discretization effects...
plot(density(react))
```

`red.cell.folate` *Red cell folate data*

Description

The `folate` data frame has 22 rows and 2 columns. It contains data on red cell folate levels in patients receiving three different methods of ventilation during anesthesia.

Usage

```
red.cell.folate
```

Format

This data frame contains the following columns:

`folate` a numeric vector, folate concentration (μg/l).

`ventilation` a factor with levels `N2O+O2,24h`: 50% nitrous oxide and 50% oxygen, continuously for 24 hours; `N2O+O2,op`: 50% nitrous oxide and 50% oxygen, only during operation; `O2,24h`: no nitrous oxide but 35%–50% oxygen for 24 hours.

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 9.10, Chapman & Hall.

Examples

```
plot(folate~ventilation,data=red.cell.folate)
```

rmr — *Resting metabolic rate*

Description

The rmr data frame has 44 rows and 2 columns. It contains the resting metabolic rate and body weight data for 44 women.

Usage

```
rmr
```

Format

This data frame contains the following columns:

body.weight a numeric vector, body weight (kg).
metabolic.rate a numeric vector, metabolic rate (kcal/24hr).

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Exercise 11.2, Chapman & Hall.

Examples

```
plot(metabolic.rate~body.weight,data=rmr)
```

secher — *Birth weight and ultrasonography*

Description

The secher data frame has 107 rows and 4 columns. It contains ultrasonographic measurements of fetuses immediately before birth and their subsequent birth weight.

Usage

```
secher
```

Format

This data frame contains the following columns:

bwt a numeric vector, birth weight (g).
bpd a numeric vector, biparietal diameter (mm).
ad a numeric vector, abdominal diameter (mm).
no a numeric vector, observation number.

Source

D. Kronborg and L.T. Skovgaard (1990), *Regressionsanalyse*, Table 3.1, FADLs Forlag (in Danish).
Secher et al. (1987), European Journal of Obstetrics, Gynecology, and Reproductive Biology, 24: 1–11.

Examples

```
plot(bwt~ad, data=secher, log="xy")
```

`secretin` *Secretin-induced blood glucose changes*

Description

The `secretin` data frame has 50 rows and 6 columns. It contains data from a glucose response experiment.

Usage

```
secretin
```

Format

This data frame contains the following columns:

gluc a numeric vector, blood glucose level.
person a factor with levels `A`–`E`.
time a factor with levels `20`, `30`, `60`, `90` (minutes since injection), and `pre` (before injection).
repl a factor with levels `a`: 1st sample; `b`: 2nd sample.
time20plus a factor with levels `20+`: 20 minutes or longer since injection; `pre`: before injection.
time.comb a factor with levels `20`: 20 minutes since injection; `30+`: 30 minutes or longer since injection; `pre`: before injection.

Details

Secretin is a hormone of the duodenal mucous membrane. An extract was administered to five patients with arterial hypertension. Primary registrations (double determination) of blood glucose were on graph paper and later quantified with the smallest of the two measurements recorded first.

Source

Anon. (1977), *Exercises in Applied Statistics*, Exercise 5.8, Dept. of Theoretical Statistics, Aarhus University.

`stroke` *Estonian stroke data*

Description

All cases of stroke in Tartu, Estonia, during the period 1991–1993, with follow-up until January 1, 1996.

Usage

```
stroke
```

Format

A data frame with 829 observations on the following 10 variables.

`sex` a factor with levels `Female` and `Male`.
`died` a Date, date of death.
`dstr` a Date, date of stroke.
`age` a numeric vector, age at stroke.
`dgn` a factor, diagnosis, with levels `ICH` (intracranial haemorrhage), `ID` (unidentified). `INF` (infarction, ischaemic), `SAH` (subarchnoid haemorrhage).
`coma` a factor with levels `No` and `Yes`, indicating whether patient was in coma after the stroke.
`diab` a factor with levels `No` and `Yes`, history of diabetes.
`minf` a factor with levels `No` and `Yes`, history of myocardial infarction.
`han` a factor with levels `No` and `Yes`, history of hypertension.
`obsmonths` a numeric vector, observation times in months (set to 0.1 for patients dying on the same day as the stroke).
`dead` a logical vector, whether patient died during the study.

Source

Original data.

References

J. Korv, M. Roose, and A.E. Kaasik (1997). Stroke Registry of Tartu, Estonia, from 1991 through 1993. Cerebrovascular Disorders 7:154–162.

`tb.dilute` *Tuberculin dilution assay*

Description

The `tb.dilute` data frame has 18 rows and 3 columns. It contains data from a drug test involving dilutions of tuberculin.

Usage

```
tb.dilute
```

Format

This data frame contains the following columns:

`reaction` a numeric vector, reaction sizes (average of diameters) for tuberculin skin pricks.
`animal` a factor with levels `1`–`6`.
`logdose` a factor with levels `0.5`, `0`, and `-0.5`.

Details

The actual dilutions were 1:100, 1:100$\sqrt{10}$, 1:1000. Setting the middle one to 1 and using base-10 logarithms gives the `logdose` values.

Source

Anon. (1977), *Exercises in Applied Statistics*, part of Exercise 4.15, Dept. of Theoretical Statistics, Aarhus University.

thuesen *Ventricular shortening velocity*

Description

The thuesen data frame has 24 rows and 2 columns. It contains ventricular shortening velocity and blood glucose for type 1 diabetic patients.

Usage

```
thuesen
```

Format

This data frame contains the following columns:

blood.glucose a numeric vector, fasting blood glucose (mmol/l).

short.velocity a numeric vector, mean circumferential shortening velocity (%/s).

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Table 11.6, Chapman & Hall.

Examples

```
plot(short.velocity~blood.glucose, data=thuesen)
```

tlc *Total lung capacity*

Description

The tlc data frame has 32 rows and 4 columns. It contains data on pretransplant total lung capacity (TLC) for recipients of heart-lung transplants by whole-body plethysmography.

Usage

```
tlc
```

Format

This data frame contains the following columns:

age a numeric vector, age of recipient (years).
sex a numeric vector code, female: 1, male: 2.
height a numeric vector, height of recipient (cm).
tlc a numeric vector, total lung capacity (l).

Source

D.G. Altman (1991), *Practical Statistics for Medical Research*, Exercise 12.5, 10.1, Chapman & Hall.

Examples

```
plot(tlc~height,data=tlc)
```

---

vitcap *Vital capacity*

Description

The vitcap data frame has 24 rows and 3 columns. It contains data on vital capacity for workers in the cadmium industry. It is a subset of the vitcap2 data set.

Usage

```
vitcap
```

Format

This data frame contains the following columns:

group a numeric vector; group codes are 1: exposed > 10 years, 3: not exposed.
age a numeric vector, age in years.
vital.capacity a numeric vector, vital capacity (a measure of lung volume) in liters.

Source

P. Armitage and G. Berry (1987), *Statistical Methods in Medical Research*, 2nd ed., Blackwell, p.286.

Examples

```
plot(vital.capacity~age, pch=group, data=vitcap)
```

---

vitcap2 *Vital capacity, full data set*

Description

The vitcap2 data frame has 84 rows and 3 columns. Age and vital capacity for workers in the cadmium industry.

Usage

```
vitcap2
```

Format

This data frame contains the following columns:

group a numeric vector; group codes are 1: exposed > 10 years, 2: exposed < 10 years, 3: not exposed.

age a numeric vector, age in years.

vital.capacity a numeric vector, vital capacity (a measure of lung volume) (l).

Source

P. Armitage and G. Berry (1987), *Statistical Methods in Medical Research*, 2nd ed., Blackwell, p.286.

Examples

```
plot(vital.capacity~age, pch=group, data=vitcap2)
```

---

wright *Comparison of Wright peak-flow meters*

Description

The wright data frame has 17 rows and 2 columns. It contains data on peak expiratory flow rate with two different flow meters on each of 17 subjects.

Usage

```
wright
```

Format

This data frame contains the following columns:

std.wright a numeric vector, data from large flow meter (l/min).
mini.wright a numeric vector, data from mini flow meter (l/min).

Source

J.M. Bland and D.G. Altman (1986), Statistical methods for assessing agreement between two methods of clinical measurement, *Lancet*, 1:307–310.

Examples

```
plot(wright)
abline(0,1)
```

---

zelazo — *Age at walking*

Description

The zelazo object is a list with four components.

Usage

```
zelazo
```

Format

This is a list containing data on age at walking (in months) for four groups of infants:

active test group receiving active training; these children had their walking and placing reflexes trained during four three-minute sessions that took place every day from their second to their eighth week of life.

passive passive training group; these children received the same types of social and gross motor stimulation, but did not have their specific walking and placing reflexes trained.

none no training; these children had no special training, but were tested along with the children who underwent active or passive training.

ctr.8w eighth-week controls; these children had no training and were only tested at the age of 8 weeks.

Note

When asked to enter these data from a text source, many students will use one vector per group and will need to reformat data into a data frame for some uses. The rather unusual format of this data set mimics that situation.

Source

P.R. Zelazo, N.A. Zelazo, and S. Kolb (1972), "Walking" in the newborn, *Science*, 176: 314–315.

C

Compendium

Elementary

Commands

`ls()` or `objects()`	List objects in workspace
`rm(object)`	Delete `object`
`search()`	Search path

Variable names

Combinations of letters, digits, and period. Must not start with a digit. Avoid starting with period.

Assignments

`<-`	Assign value to variable
`->`	Assignment "to the right"
`<<-`	Global assignment (in functions)

Operators

Arithmetic

`+`	Addition
`-`	Subtraction, sign
`*`	Multiplication
`/`	Division
`^`	Raise to power
`%/%`	Integer division
`%%`	Remainder from integer division

Logical and relational

`==`	Equal to
`!=`	Not equal to
`<`	Less than
`>`	Greater than
`<=`	Less than or equal to
`>=`	Greater than or equal to
`is.na(x)`	Missing?
`&`	Logical AND
`|`	Logical OR
`!`	Logical NOT

`&` and `|` are *elementwise*. See "Programming" (p. 336) for `&&` and `||`.

Vectors and data types

Generating

`numeric(25)`	25 zeros
`character(25)`	25 × `""`
`logical(25)`	25 × `FALSE`
`seq(-4,4,0.1)`	Sequence: −4.0, −3.9, 3.8, . . . , 3.9, 4.0
`1:10`	Same as `seq(1,10,1)`
`c(5,7,9,13,1:5)`	Concatenation: 5 7 9 13 1 2 3 4 5
`rep(1,10)`	1 1 1 1 1 1 1 1 1 1
`gl(3,2,12)`	Factor with 3 levels, repeat each level in blocks of 2, up to length 12 (i.e., 1 1 2 2 3 3 1 1 2 2 3 3)

Coercion

`as.numeric(x)`	Convert to numeric
`as.character(x)`	Convert to text string
`as.logical(x)`	Convert to logical
`factor(x)`	Create factor from vector `x`

For factors, see also "Tabulation, grouping, and recoding" (p. 331).

Data frames

`data.frame(height = c(165,185), weight = c(90,65))`	Data frame with two named vectors
`data.frame(height, weight)`	Collect vectors into data frame
`dfr$var`	Select vector `var` in data frame `dfr`
`attach(dfr)`	Put data frame in search path
`detach()`	— and remove it from path

Attached data frames always come *after* `.GlobalEnv` in the search path.

Attached data frames are copies; subsequent changes to `dfr` have no effect.

Numerical functions

Mathematical

`log(x)`	Logarithm of x, natural (base-e) logarithm
`log10(x)`	Base-10 logarithm
`exp(x)`	Exponential function e^x
`sin(x)`	Sine
`cos(x)`	Cosine
`tan(x)`	Tangent
`asin(x)`	Arcsin (inverse sine)
`acos(x)`	
`atan(x)`	
`min(x)`	Smallest value in vector
`min(x1,x2,...)`	Minimum over several vectors (one number)
`max(x)`	Largest value in vector
`range(x)`	Like `c(min(x),max(x))`
`pmin(x1,x2,...)`	Parallel (elementwise) minimum over multiple equally long vectors
`pmax(x1,x2,...)`	Parallel maximum
`length(x)`	Number of elements in vector
`sum(complete.cases(x))`	Number of nonmissing elements in vector

Statistical

`mean(x)`	Average
`sd(x)`	Standard deviation
`var(x)`	Variance
`median(x)`	Median
`quantile(x,p)`	Quantiles
`cor(x,y)`	Correlation

Indexing/selection

`x[1]`	First element
`x[1:5]`	Subvector containing first five elements
`x[c(2,3,5,7,11)]`	Element nos. 2, 3, 5, 7, and 11
`x[y<=30]`	Selection by logical expression
`x[sex=="male"]`	Selection by factor variable
`i <- c(2,3,5,7,11); x[i]`	Selection by numeric variable
`l <- (y<=30); x[l]`	Selection by logical variable

Matrices and data frames

`m[4,]`	Fourth row
`m[,3]`	Third column
`dfr[dfr$var<=30,]`	Partial data frame
`subset(dfr,var<=30)`	Same, often simpler

Input of data

`data(name)`	Built-in data set
`read.table("filename")`	Read from external file

Common arguments to `read.table`

`header=TRUE`	First line has variable names
`sep=","`	Data are separated by commas
`dec=","`	Decimal point is comma
`na.strings="."`	Missing value is dot

Variants of `read.table`

`read.csv("filename")`	Comma-separated
`read.delim("filename")`	Tab-delimited
`read.csv2("filename")`	Semicolon-separated, comma decimal point
`read.delim2("filename")`	Tab-delimited, comma decimal point

These all set `header=TRUE`.

Missing values

Functions

`is.na(x)`	Logical vector. `TRUE` where `x` has `NA`.
`complete.cases(x1,x2,...)`	Missing neither in `x1`, nor `x2`, nor....

Arguments to other functions

`na.rm=`	In statistical functions, remove missing if TRUE, return `NA` if FALSE.
`na.last=`	In `sort`; `TRUE`, `FALSE` and `NA` mean, respectively, "last", "first", and "throw away".
`na.action=`	In `lm`, etc., values `na.fail`, `na.omit`, `na.exclude`; also in `options("na.action")`.
`na.print=`	In `summary` and `print.default`; how to represent `NA` in output.
`na.strings=`	In `read.table()`; code(s) for `NA` in input.

Tabulation, grouping, and recoding

`table(f1,...)`	(Cross)-tabulation
`xtabs(~ f1 + ...)`	ditto, formula interface
`ftable(f1 ~ f2 + ...)`	"Flat" tables
`tapply(x,f,mean)`	Table of means
`aggregate(df,list(f),mean)`	Means for several variables
`by(df, list(f), summary)`	Summarize data frame by group
`factor(x)`	Convert vector to factor
`cut(x,breaks)`	Groups from cutpoints for continuous variable

Arguments to `factor`

`levels`	Values of x to code. Use if some values are not present in data or if the order would be wrong.
`labels`	Values associated with factor levels.
`exclude`	Values to exclude. Default `NA`. Set to `NULL` to have missing values included as a level.

Arguments to `cut`

`breaks`	Cutpoints. Note that values of `x` outside `breaks` give `NA`.
`labels`	Names for groups. Default is `(0,30]`, etc.
`right`	Right endpoint included? (`FALSE`: left)

Recoding factors

`levels(f) <- names`	New level names
`levels(f) <- list( new1=c("old1","old2") new2="old3")`	Combining levels

Statistical distributions

Normal distribution

`dnorm(x)`	Density
`pnorm(x)`	Cumulative distribution function, $P(X \leq x)$
`qnorm(p)`	p-quantile, $x : P(X \leq x) = p$
`rnorm(n)`	`n` (pseudo-)random normally distributed numbers

Distributions

`pnorm(x,mean,sd)`	Normal
`plnorm(x,mean,sd)`	Lognormal
`pt(x,df)`	Student's t
`pf(x,n1,n2)`	F distribution
`pchisq(x,df)`	χ^2
`pbinom(x,n,p)`	Binomial
`ppois(x,lambda)`	Poisson
`punif(x,min,max)`	Uniform
`pexp(x,rate)`	Exponential
`pgamma(x,shape,scale)`	Gamma
`pbeta(x,a,b)`	Beta

Same convention (d-q-r) for density, quantiles, and random numbers as for normal distribution.

Statistical standard methods

Continuous response

`t.test`	One- and two-sample t tests
`pairwise.t.test`	Pairwise comparisons
`cor.test`	Correlation
`var.test`	Comparison of two variances (F test)
`lm(y ~ x)`	Regression analysis
`lm(y ~ f)`	One-way analysis of variance
`lm(y ~ f1 + f2)`	Two-way analysis of variance
`lm(y ~ f + x)`	Analysis of covariance
`lm(y ~ x1 + x2 + x3)`	Multiple regression analysis
`bartlett.test`	Bartlett's test (k variances)
Nonparametric:	
`wilcox.test`	One- and two-sample Wilcoxon tests
`kruskal.test`	Kruskal–Wallis test
`friedman.test`	Friedman's two-way analysis of variance
`cor.test` variants:	
`method="kendall"`	Kendall's τ
`method="spearman"`	Spearman's ρ

Discrete response

`binom.test`	Binomial test (incl. sign test)
`prop.test`	Comparison of proportions
`prop.trend.test`	Test for trend in relative proportions
`fisher.test`	Exact test in small tables
`chisq.test`	χ^2 test
`glm(y ~ x1+x2+x3, binomial)`	Logistic regression

Models

Model formulas

`~`	Described by
`+`	Additive effects
`:`	Interaction
`*`	Main effects + interaction (`a*b = a + b + a:b`)
`-1`	Remove intercept

Classifications are represented by descriptive variable being a *factor*.

Linear, nonlinear, and generalized linear models

`lm.out <- lm(y ~ x)`	Fit model and save result
`summary(lm.out)`	Coefficients, etc.
`anova(lm.out)`	Analysis of variance table
`fitted(lm.out)`	Fitted values
`resid(lm.out)`	Residuals
`predict(lm.out, newdata)`	Predictions for new data frame
`glm(y ~ x, binomial)`	Logistic regression
`glm(y ~ x, poisson)`	Poisson regression
`nls(y ~ a*exp(-b*x), start=c(a=5, b=.2))`	Nonlinear regression

Diagnostics

`rstudent(lm.out)`	Studentized residuals
`dfbetas(lm.out)`	Change in β if obs. removed
`dffits(lm.out)`	Change in fit if obs. removed

Survival analysis

`S <- Surv(time, ev)`	Create survival object
`survfit(S)`	Kaplan–Meier estimate
`plot(survfit(S))`	Survival curve
`survdiff(S ~ g)`	(Log-rank) test for equal survival curves
`coxph(S ~ x1 + x2)`	Cox's proportional hazards model

Graphics

Standard plots

`plot()`	Scatterplot (and more)
`hist()`	Histogram
`boxplot()`	Box-and-whiskers plot
`stripplot()`	Stripplot
`barplot()`	Bar diagram
`dotplot()`	Dot diagram
`piechart()`	Cakes…
`interaction.plot()`	Interaction plot

Plotting elements

`lines()`	Lines
`abline()`	Line given by intercept and slope (and more)
`points()`	Points
`segments()`	Line segments
`arrows()`	Arrows (N.B.: `angle=90` for error bars)
`axis()`	Axis
`box()`	Frame around plot
`title()`	Title (above plot)
`text()`	Text in plot
`mtext()`	Text in margin
`legend()`	List of symbols

These are all *added* to existing plots.

Graphical parameters

`pch`	Symbol (*p*lotting *ch*aracter)
`mfrow, mfcol`	Several plots on one (*m*ulti*f*rame)
`xlim, ylim`	Plot limits
`lty,lwd`	Line type/width
`col`	Colour
`cex, mex`	Character size and line spacing in margins

See the help page for `par` for more details.

Programming

Conditional execution	`if(p<0.05)` `  print("Hooray!")`

— with alternative	`if(p<0.05)`
	`  print("Hooray!")`
	`else`
	`  print("Bah.")`

Loop over list	`for(i in 1:10)` `  print(i)`

Loop	`i <- 1`
	`while(i<10) {`
	`  print(i)`
	`  i <- i + 1`
	`}`

User-defined function	`f <- function(a,b,doit=FALSE){`
	`  if (doit)`
	`  a + b`
	`  else`
	`  0`
	`}`

In flow control, one uses `a && b` and `a || b`, where `b` is only computed if necessary; that is, `if a then b else FALSE` and `if a then TRUE else b`.	

D

Answers to exercises

1.1 One possibility is

```
x <- y <- c(7, 9, NA, NA, 13)
all(is.na(x) == is.na(y)) & all((x == y)[!is.na(x)])
```

Notice that `FALSE & NA` is `FALSE`, so the case of different `NA` patterns is handled correctly.

1.2 Factor `x` gets treated as if it contained the integer codes.

```
x <- factor(c("Huey", "Dewey", "Louie", "Huey"))
y <- c("blue", "red", "green")
x
y[x]
```

(This is useful, e.g., when selecting plot symbols.)

1.3

```
juul.girl <- juul[juul$age >=7 & juul$age < 14 & juul$sex == 2,]
summary(juul.girl)
```

1.4 The levels with the same name are collapsed into one.

1.5 `sapply(1:10, function(i) mean(rexp(20)))`

2.1 To insert 1.23 between `x[7]` and `x[8]`:

```
x <- 1:10
z <- append(x, 1.23, after=7)
z
```

Otherwise, consider

```
z <- c(x[1:7],1.23,x[8:10])
z
```

or, more generally, to insert `v` just after index `k` (the boundary cases require some care),

```
v <- 1.23; k <- 7
i <- seq(along=x)
z <- c(x[i <= k], v, x[i > k])
z
```

2.2 (First part only) Use

```
write.table(thuesen, file="foo.txt")
# edit the file
read.table("foo.txt", na.strings=".")
```

or

```
write.table(thuesen, file="foo.txt", na=".")
read.table("foo.txt", na.strings=".")
```

(Notice that if you do not edit the file in the first case, then the second column gets read as a character vector.)

3.1

```
1 - pnorm(3)
1 - pnorm(42, mean=35, sd=6)
dbinom(10, size=10, prob=0.8)
punif(0.9) # this one is obvious...
1 - pchisq(6.5, df=2)
```

It might be better to use `lower.tail=FALSE` instead of subtracting from 1 in (a), (b), and (e). Notice that question (c) is about a point probability, whereas the others involve the cumulative distribution function.

3.2 Evaluate each of the following. Notice that the standard normal can be used for all questions.

```
pnorm(-2) * 2
qnorm(1-.01/2)
qnorm(1-.005/2)
qnorm(1-.001/2)
qnorm(.25)
qnorm(.75)
```

Again, `lower.tail` can be used in some cases.

3.3 `dbinom(0, size=10, prob=.2)`

3.4 Either of the following should work:

```
rbinom(10, 1, .5)
ifelse(rbinom(10, 1, .5) == 1, "H", "T")
c("H", "T")[1 + rbinom(10, 1, .5)]
```

The first one gives a 0/1 result, the two others H/T like the `sample` example in the text. One advantage of using `rbinom` is that its `prob` argument can be a vector, so you can have different probabilities of success for each element of the result.

4.1 For example,

```
x <- 1:5 ; y <- rexp(5,1) ; opar <- par(mfrow=c(2,2))
plot(x, y, pch=15) # filled square
plot(x, y, type="b", lty="dotted")
plot(x, y, type="b", lwd=3)
plot(x, y, type="o", col="blue")
par(opar)
```

4.2 Use a filled symbol, and set the fill colour equal to the plot background:

```
plot(rnorm(10),type="o", pch=21, bg="white")
```

4.3 You can use `qqnorm` with `plot.it=F` and get a return value from which you can extract the range information (you could of course also get this "by eye").

```
x1 <- rnorm(20)
x2 <- rnorm(10)+1
q1 <- qqnorm(x1, plot.it=F)
q2 <- qqnorm(x2, plot.it=F)
xr <- range(q1$x, q2$x)
yr <- range(q1$y, q2$y)
qqnorm(x1, xlim=xr, ylim=yr)
points(q2, col="red")
```

Here, `qqnorm` is used for the basic plot to get the labels right. Then `points` is used with `q2` for the overlay.

Setting `type="l"` gives a messy plot because the values are not plotted in order. The remedy is to use `sort(x1)` and `sort(x2)`.

4.4 The breaks occur at integer values, as do the data. Data on the boundary are counted in the column to the left of it, effectively shifting the

histogram half a unit left. The `truehist` function allows you to specify a better set of breaks.

```
hist(react)
library(MASS)
truehist(react,h=1,x0=.5)
```

4.5 The thing to notice is the linear interpolation between data points:

```
z <- runif(5)
curve(quantile(z,x), from=0, to=1)
```

5.1 The distribution appears reasonably normal, with some discretization effect and two weak outliers, one at each end. There is a significant difference from zero ($t = -7.75, p = 1.1 \times 10^{-13}$).

```
qqnorm(react)
t.test(react)
```

5.2 `t.test(vital.capacity~group,conf=0.99,data=vitcap)`. The fact that `age` also differs by group may cause bias.

5.3 This is quite parallel to `t.test` usage

```
wilcox.test(react)
wilcox.test(vital.capacity~group, data=vitcap)
```

5.4 The following builds a post-vs.-pre plot, a difference-vs.-average) (Bland-Altman) plot, and a histogram and a Q-Q plot of the differences.

```
attach(intake) ; opar <- par(mfrow=c(2,2))
plot(post ~ pre) ; abline(0,1)
plot((post+pre)/2, post - pre,
    ylim=range(0,post-pre)); abline(h=0)
hist(post-pre)
qqnorm(post-pre)
detach(intake)
par(opar)
```

5.5 The outliers are the first and last observations in the (sorted) data vector and can be removed as follows

```
shapiro.test(react)
shapiro.test(react[-c(1,334)])
qqnorm(react[-c(1,334)])
```

The test comes out highly significant even with outliers removed because it picks up the discretization effect in the otherwise nearly straight-line `qqnorm` plot.

5.6 A paired *t* test is appropriate if there is no period effect. However, even with a period effect (assumed additive), you would expect the difference between the two periods to be the same in both groups if there were no effect of treatment. This can be used to test for a treatment effect.

```
attach(ashina)
t.test(vas.active, vas.plac, paired=TRUE)
t.test((vas.active-vas.plac)[grp==1],
       (vas.plac-vas.active)[grp==2])
```

Notice that the subtraction is reversed in one group. Observe that the confidence interval in the second case is for *twice* the treatment effect.

5.7 This is the sort of thing `replicate` is for. The plot at the end shows a P-P plot with logarithmic axes, showing that extreme *p*-values tend to be exaggerated.

```
t.test(rnorm(25))$p.value       #repeat 10x
t.test(rt(25,df=2))$p.value     #repeat 10x
t.test(rexp(25), mu=1)$p.value  #repeat 10x
x <- replicate(5000, t.test(rexp(25), mu=1)$p.value)
qqplot(sort(x),ppoints(5000),type="l",log="xy")
```

6.1 The following gives both elementary and more general answers. Notice the use of `confint`.

```
fit <- lm(metabolic.rate ~ body.weight, data=rmr)
summary(fit)
811.2267 + 7.0595 * 70 # , or:
predict(fit, newdata=data.frame(body.weight=70))
qt(.975,42)
7.0595 + c(-1,1) * 2.018 * 0.9776 # , or:
confint(fit)
```

6.2 `summary(lm(sqrt(igf1)~age,data=juul,subset=age>25))`

6.3 We can fit a linear model and plot the data as follows:

```
summary(lm(log(ab)~age, data=malaria))
plot(log(ab)~age, data=malaria)
```

The plot appears to show a cyclic pattern. It is unclear whether it reflects a significant departure from the model, though. Malaria is a disease with epidemic behaviour, so cycles are plausible.

6.4 (This could be elaborated by wrapping the random number generation in a function, etc.)

```
rho <- .90 ; n <- 100
x <- rnorm(n)
```

```
y <- rnorm(n, rho * x, sqrt(1 - rho^2))
plot(x, y)
cor.test(x, y)
cor.test(x, y, method="spearman")
cor.test(x, y, method="kendall")
```

You will most likely find that the Kendall correlation is somewhat smaller than the two others.

7.1

```
walk <- unlist(zelazo) # or c(..,recursive=TRUE)
group <- factor(rep(1:4,c(6,6,6,5)), labels=names(zelazo))
summary(lm(walk ~ group))
t.test(zelazo$active,zelazo$ctr.8w) # first vs. last
t.test(zelazo$active,unlist(zelazo[-1])) # first vs. rest
```

7.2 A and C differ with B intermediate, not significantly different from either. (The B–C comparison is not available from the summary, but due to the balanced design, the standard error of that difference is 0.16656 like the two others.)

```
fit <- lm(volume~method+subject, data=lung)
anova(fit)
summary(fit)
```

7.3

```
kruskal.test(walk ~ group)
wilcox.test(zelazo$active,zelazo$ctr.8w) # first vs. last
wilcox.test(zelazo$active,unlist(zelazo[-1])) # first vs. rest
friedman.test(volume ~ method | subject, data=lung)
wilcox.test(lung$volume[lung$method=="A"],
            lung$volume[lung$method=="C"], paired=TRUE) # etc.
```

7.4 (Only the square-root transform is shown; you can do the same for log-transformed and untransformed data.)

```
attach(juul)
tapply(sqrt(igf1),tanner, sd, na.rm=TRUE)
plot(sqrt(igf1)~jitter(tanner))
oneway.test(sqrt(igf1)~tanner)
```

The square root looks nice, logarithms become skewed in the opposite direction. The transformations do not make much of a difference for the test. It is, however, a problem that strong age effects are being ignored, particularly within Tanner stage 1.

8.1 With 10 patients, $p = 0.1074$. Fourteen or more are needed for significance at level 0.05.

```
binom.test(0, 10, p=.20, alt="less")
binom.test(0, 13, p=.20, alt="less")
binom.test(0, 14, p=.20, alt="less")
```

8.2 Yes, it is highly significant.

```
prop.test(c(210,122),c(747,661))
```

8.3 The confidence interval (from `prop.test`) is $(-0.085, 0.507)$

```
M <- matrix(c(23,7,18,13),2,2)
chisq.test(M)
fisher.test(M)
prop.test(M)
```

8.4 The following is a simplified analysis, which uses `fisher.test` because of the small cell counts:

```
tbl <- c(42, 157, 47, 62, 4, 15, 4, 1, 8, 28, 9, 7)
dim(tbl) <- c(2,2,3)
dimnames(tbl) <- list(c("A","B"),
                      c("not pierced","pierced"),
                      c("ok","broken","cracked"))
ftable(tbl)
fisher.test(tbl["B",,]) # slice analysis
fisher.test(tbl["A",,])
fisher.test(margin.table(tbl,2:3)) # marginal
```

You may wish to check that there is little or no effect of egg size on breakage, so that the marginal analysis is defensible. You could also try collapsing the "broken" and "cracked" categories.

8.5 The curve shows substantial discontinuities where probability mass is shifted from one tail to the other and also a number of local minima. A confidence region could be defined as those p against which there is no significant evidence at level α, but for some α that is not an interval.

```
p <- seq(0,1,0.001)
pval <- sapply(p,function(p)binom.test(3,15,p=p)$p.value)
plot(p,pval,type="l")
```

9.1 The estimated sample size is 6.29 or 8.06 per group depending on whether you use one- or two-sided testing. The approximate formula gives 6.98 for the two-sided case. The reduction in power due to the unbalanced sampling can be accounted for by reducing `delta` by the ratio of the two `SEDM`.

```
power.t.test(power=.8,delta=.30,sd=.20)
power.t.test(power=.8,delta=.30,sd=.20,alt="one.sided")
```

```
(qnorm(.975)+qnorm(.8))^2*2*(.2/.3)^2 # approx. formula
power.t.test(n=8, delta=.30, sd=.20)   # power with eq.size
d2 <- .30 * sqrt(2/8) / sqrt(1/6+1/10) # corr.f.uneq. size
power.t.test(n=8, delta=d2, sd=.20)
```

9.2 This is straightforward:

```
power.prop.test(power=.9, p1=.6, p2=.75)
power.prop.test(power=.8, p1=.6, p2=.75)
```

9.3 Notice that the noncentral t distribution is asymmetric, with a rather heavy right tail.

```
curve(dt(x-3, 25), from=0, to=5)
curve(dt(x, 25, 3), add=TRUE)
```

9.4 This causes the "power" at zero effect size (i.e., under the null hypothesis) to be *half* the significance level, in contradiction to theory. For any relevant true effect size, the difference is immaterial.

9.5 The power in that case is approximately 0.50; exactly so if the variance is assumed known.

10.1

```
attach(thuesen)
f <- cut(blood.glucose, c(4, 7, 9, 12, 20))
levels(f) <- c("low", "intermediate", "high", "very high")
```

10.2

```
bcmort2 <- within(bcmort,{
  period <- area <- cohort
  levels(period) <- rep(c("1991-2001","1981-1991"), each=2)
  levels(area) <- rep(c("Cph+Frb","Nat"),2)
})
summary(bcmort2)
```

10.3 One way is the following (for later use, we also make sure that variables are converted to factors):

```
ashina.long <- reshape(ashina, direction="long",
                       varying=1:2, timevar="treat")
ashina.long <- within(ashina.long, {
     m <- matrix(c(2,1,1,2),2)
     id <- factor(id)
     treat <- factor(treat)
     grp <- factor(grp)
     period <- factor(m[cbind(grp,treat)])
```

```
    rm(m)
})
```

Notice the use of *array indexing*. Alternatively, an `ifelse` construct can be used; e.g., the following (notice that `(3 - grp)` is 2 when `grp` is 1 and vice versa):

```
within(ashina.long,
  period2 <- ifelse(treat != "active",
             as.numeric(grp), 3 - as.numeric(grp))
)
```

Arithmetic involving `grp` does not work after it was converted to a factor, hence the conversion with `as.numeric`.

10.4 This can be done a little more easily than in the `nickel` example by using `subset` and `transform`. It also helps that all observation periods start at time zero in this case.

```
stroke.trim <- function(t1, t2)
   subset(transform(stroke,
                    entry=t1, exit=pmin(t2, obsmonths),
                    dead=dead & obsmonths <= t2),
          entry < exit)
stroke2 <- do.call(rbind, mapply(stroke.trim,
       c(0,0.5,2,12), c(0.5,2,12,Inf), SIMPLIFY=F))
table(stroke$dead)
table(stroke2$dead)
```

Notice the use of `mapply` here. This is like `sapply` and `lapply` but allows the function to have multiple arguments. Alternatively, one could arrange for `stroke.trim` to have a single `interval` argument and use `lapply` on a list of such intervals.

The tabulation at the end is a "sanity check" to show that we have the same number of deaths but many more censored cases after time-splitting.

11.1 The model with both diameters has a residual error of 0.107, compared with 0.128 using abdominal diameter alone and 0.281 with no predictors at all. If a fetus is scaled isotropically, a cubic relation with weight is expected, and you could speculate that this is reflected in the sum of coefficients when using log scales.

```
summary(lm(log(bwt) ~ log(bpd) + log(ad), data=secher))
summary(lm(log(bwt) ~ log(ad), data=secher))
```

11.2 If you use `attach(tlc)`, the `tlc` variable will mask the data frame of the same name, which makes it awkward to access the data frame if you need to. If the data frame is in the global environment rather than in

a package, you get the opposite problem, masking of the variable by the data frame. The simplest workaround is to avoid `attach`.

```
pairs(tlc)
summary(lm(log(tlc) ~ ., data=tlc))
opar <- par(mfrow=c(2,2))
plot(lm(log(tlc) ~ ., data=tlc), which=1:4)

drop1(lm(log(tlc) ~ ., data=tlc))
drop1(lm(log(tlc) ~ . - age, data=tlc))

par(mfrow=c(1,1))
plot(log(tlc) ~ height, data=tlc)
par(mfrow=c(2,2))
plot(lm(tlc ~ ., data=tlc), which=1:4) # slightly worse
par(opar)
```

Some new variations of model formulas were introduced above. A dot on the right-hand side in this context means "everything not used on the left-hand side" within the scope of the data frame. A minus term is removed from the model. In other words, `... ~ . - age` is the same as `... ~ sex + height`.

11.3 The regression coefficient describes a value to be added for females.

11.4 `age` is highly significant in the first analysis but only borderline significant ($p = 0.06$) in the second analysis after removing `height` and `weight`. You would expect similar results, but the number of observations differs in the two cases, due to missing observations.

```
summary(lm(sqrt(igf1) ~ age, data=juul2, subset=(age >= 25)))
anova(lm(sqrt(igf1) ~ age + weight + height,
           data=juul2, subset=(age >= 25)))
```

11.5 `sex` is treated as a binary indicator for girls. Notice that there are effects both of the mother's and the child's size. The reason why height rather than weight of the mother enters into the equation is somewhat obscure, but one could speculate that weight is an unreliable indicator shortly after pregnancy.

```
summary(lm(dl.milk ~ . - no, data=kfm))
summary(lm(dl.milk ~ . - no - mat.weight, data=kfm))
summary(lm(dl.milk ~ . - no - mat.weight - sex, data=kfm))
summary(lm(dl.milk ~ weight + mat.height, data=kfm))
```

The variations on model formulas used here were described in the solution to Exercise 11.2.

12.1 Using `ashina.long` from Exercise 10.3,

```
fit.ashina <- lm(vas ~ id + period + treat, data=ashina.long)
drop1(fit.ashina, test="F")
anova(fit.ashina)

attach(ashina)
dd <- vas.active - vas.plac
t.test(dd[grp==1], -dd[grp==2], var.eq=T)
t.test(dd[grp==1], dd[grp==2], var.eq=T)
```

Notice that the imbalance in group sizes makes the tests for period and treatment effects order-dependent. The t tests are equivalent to the F tests from `drop1` but not those from `anova`.

12.2

```
attach(tb.dilute)
anova(lm(reaction ~ animal + logdose))
ld <- c(0.5, 0, -0.5)[logdose]
anova(lm(reaction ~ animal + ld))
summary(lm(reaction ~ animal + ld))
4.7917 + 0.6039 * qt(c(.025,.975), 11)
# or:
confint(lm(reaction ~ animal + ld))["ld",]

slopes <- reaction[logdose==0.5] - reaction[logdose==-0.5]
t.test(slopes)

anova(lm(reaction ~ animal*ld))
```

Notice that the formula for the fitted slope is $\hat{\beta} = \sum xy / \sum x^2$ since $\bar{x} = 0$, which reduces to taking differences. (The calculation does rely on data being in the right order.)

The confidence interval is wider in the t test, reflecting that slopes may vary between rats and that there are fewer degrees of freedom for estimating the variation.

The final ANOVA contains a test for parallel slopes, and the F statistic is less than one, so in these data the slopes vary *less* than expected and the DF must be the important issue for the confidence interval.

12.3 This can be varied indefinitely, but consider these examples:

```
model.matrix(~ a:b)         ; lm(z ~ a:b)
model.matrix(~ a * b)       ; lm(z ~ a * b)
model.matrix(~ a:x)         ; lm(z ~ a:x)
model.matrix(~ a * x)       ; lm(z ~ a * x)
model.matrix(~ b * (x + y)) ; lm(z ~ b * (x + y))
```

The first model is singular because indicator variables are created for all four groups, but the intercept is not removed. R will only reduce the set of design variables for an interaction term between categorical variables

when one of the main effects is present. There are no singularities in either of the two cases involving a categorical and a continuous variable, but the first one has one parameter less (common-intercept model).

The last example has a "coincidental" singularity (`x` and `y` are proportional within each level of `b`) that R has no chance of detecting.

12.4 The models can be illustrated by plotting the fitted values against time with separate symbols for each person; e.g.,

```
tt <- c(20,30,60,90,0)[time]
plot(fitted(model4)~tt,pch=as.character(person))
```

With `model1` there is no imposed structure, `model2` is completely additive so that the individual traces are parallel to each other, `model3` allows the jump from the "pre" value to the value at 20 minutes to vary between individuals, and finally `model4` is like `model3` except that there is no change after 30 minutes (traces become horizontal). So `model3` is nested in `model1` and both `model2` and `model4` are nested in `model3`, but there is no nesting relation between `model2` and `model4`.

12.5

```
bp.obese <- transform(bp.obese,sex=factor(sex, labels=c("M","F")))
plot(log(bp) ~ log(obese), pch=c(20,21)[sex], data=bp.obese)
summary(lm(log(bp) ~ sex, data=bp.obese))
summary(lm(log(bp) ~ sex + log(obese), data=bp.obese))
summary(lm(log(bp) ~ sex*log(obese), data=bp.obese))
```

12.6

```
vitcap2 <- transform(vitcap2,group=factor(group,
                                          labels=c("exp>10",
                                          "exp<10", "unexp")))
attach(vitcap2)
plot(vital.capacity~age, pch=(20:22)[group])
vit.fit <- lm(vital.capacity ~ age*group)
summary(vit.fit)
drop1(vit.fit, test="F")
for (i in 1:3) abline(lm(vital.capacity ~ age,
                         subset=as.numeric(group)==i), lty=i)
legend(20, 3.5 ,legend=levels(group), pch=20:22, lty=1:3)
```

Notice that there is a significant interaction; i.e., the lines are *not* parallel.

12.7

```
juul.prepub <- subset(juul, tanner==1)

summary(lm(sqrt(igf1)~age, data=juul.prepub, subset= sex==1))
summary(lm(sqrt(igf1)~age, data=juul.prepub, subset= sex==2))
```

```
summary(lm(sqrt(igf1)~age*factor(sex), data=juul.prepub))
summary(lm(sqrt(igf1)~age+factor(sex), data=juul.prepub))
```

12.8

```
summary(fit.aicopt <- step(lm(dl.milk ~ . - no, data=kfm)))
opar <- par(mfrow=c(2,2))
plot(fit.aicopt, which=1:4)
kfm[32,]
summary(kfm)
summary(update(fit.aicopt, ~ . - sex))
plot(update(fit.aicopt, ~ . - sex - ml.suppl), which=1:4)
par(opar)
```

Observation 32 contains an extremely large value of `ml.suppl` and therefore has a large influence on its regression coefficient. Without `ml.suppl` in the model, the Cook's distances are much smaller.

12.9

```
juulyoung <- subset(juul, age < 25)
juulyoung <- transform(juulyoung,
                  sex=factor(sex), tanner=factor(tanner))
fit.untf <- lm(igf1 ~ age * sex * tanner, data=juulyoung,
               na.action=na.exclude)
plot(fitted(fit.untf) ~ age, data=juulyoung,
     col=c("red","green")[sex])
fit.log <- update(fit.untf, log(igf1) ~ .)
fit.sqrt <- update(fit.untf, sqrt(igf1) ~ .)
opar <- par(mfrow=c(2,2))
plot(fit.untf, which=1:4)
plot(fit.log, which=1:4)
plot(fit.sqrt, which=1:4)
par(opar)
```

13.1

```
summary(glm(mal~age+log(ab), binomial, data=malaria))
```

(You may also want to check for interaction.)

13.2

```
attach(graft.vs.host)
type <- factor(type,labels=c("AML", "ALL", "CML"))
m1 <- glm(gvhd~rcpage+donage+type+preg+log(index), binomial)
m1a <- glm(gvhd~rcpage+donage+type+preg+index, binomial)
summary(m1)
summary(m1a)
```

The coefficient to `log(index)` is more significant, but the model with `index` has a slightly better deviance. There is little hard evidence for either. The log-transform has the advantage that it reduces the influence of two very large values of `index`.

```
drop1(m1, test="Chisq")
drop1(update(m1, ~ . - rcpage), test="Chisq")
drop1(update(m1, ~ . - rcpage - type), test="Chisq")
drop1(update(m1, ~ . - rcpage - type - preg), test="Chisq")
summary(m2 <- glm(gvhd~donage + log(index), binomial))
```

Notice that except for `log(index)` it is essentially an arbitrary decision which variables to put in the final model. Altman (1991) treats the `type` classification as separate binary variables and gets a final model where ALL and AML are combined into one group and includes `preg` but not `donage`.

13.3 For example,

```
confint(m2)
## normal approximation:
est <-  coefficients(summary(m2))[,1]
se <-  coefficients(summary(m2))[,2]
est + cbind(qnorm(.025)*se, qnorm(.975)*se)
confint.default(m2)
```

Notice that the `confint`-generated intervals lie asymmetrically around the estimate. In this case, both ends of the interval are shifted away from zero, in accordance with the fact that the deviance-based tests from `drop1` have lower p-values than the approximate t tests in `summary`.

13.4 The model can be fitted as follows

```
counts <- c(13,40,157,40,21,61)
total <- c(108,264,375,310,181,162)
age <- gl(3,1,6)
type <- gl(2,3,6)
anova(glm(counts/total~age+type,weights=total, binomial),
      test="Chisq")
```

The effect of `type` vanished once `age` was included, suggesting that it really is the same disease, which has affected mostly younger (and fitter) subjects in the Eastern region.

13.5

```
juul.girl <- transform(subset(juul,age>8 & age<20 &
                              complete.cases(menarche)),
                       menarche=factor(menarche))
logit.menarche <- glm(menarche~age+I(age^2)+I(age^3),
                      binomial, data=juul.girl)
```

```
probit.menarche <- glm(menarche~age+I(age^2)+I(age^3),
                       binomial(probit), data=juul.girl)
summary(logit.menarche)
summary(probit.menarche)
Age=seq(8,20,.1)
newages <- data.frame(age=Age)
p.logit <- predict(logit.menarche,newdata=newages,type="resp")
p.probit <- predict(probit.menarche,newdata=newages,type="resp")
matplot(Age,cbind(p.probit,p.logit),type="l")
```

14.1

```
attach(graft.vs.host)
plot(survfit(Surv(time,dead)~gvhd))
survdiff(Surv(time,dead)~gvhd)
summary(coxph(Surv(time,dead) ~ gvhd)) # for comparison
summary(coxph(Surv(time,dead) ~
              gvhd + log(index) + donage + rcpage + preg))
```

Subsequent elimination suggests that `preg` might be a better predictor than `gvhd`.

14.2

```
attach(melanom)
cox1 <- coxph(Surv(days, status==1) ~
              log(thick) + sex + strata(ulc))
new <- data.frame(sex=2, thick=c(0.1, 0.2, 0.5))
svfit <-  survfit(cox1,newdata=new)
plot(svfit[2], ylim=c(.985, 1))
```

14.3

```
summary(coxph(Surv(obsmonths, dead)~age+sex, data=stroke))
summary(coxph(Surv(obsmonths, dead)~sex, data=stroke))
with(stroke, tapply(age,sex,mean))
```

The men were considerably younger than the women when they had their stroke, which may explain their apparently better survival.

14.4 Using `stroke2` from Exercise 10.4,

```
summary(coxph(Surv(entry, exit, dead)~age+sex, data=stroke2))
```

Notice that the result is essentially the same as in the unsplit analysis; only `n` and `Rsquare` are changed.

15.1 Using `bcmort2` from Exercise 10.2,

```
bcfit <- glm(bc.deaths ~ (age + period + area)^2, poisson,
             offset=log(p.yr), data=bcmort2)
```

```
summary(bcfit)
drop1(bcfit, test="Chisq")
confint(bcfit, parm="period1981-1991:areaNat")
```

15.2 Continuing with `stroke2` from Exercise 10.4, the only slight complication is to convert `entry` to a factor to specify the relevant time interval.

```
summary(glm(dead~sex+age+factor(entry), poisson,
       offset=log(exit-entry), data=stroke2))
```

Notice how similar the results are to the Cox analysis in Exercise 14.3.

16.1 To fit to the data for girls, just copy the procedure for boys. Even though the growth curves differ, there is no real reason to redo the starting value calculation, so we can fit the model to boys, girls, and both as follows

```
girls <- subset(juul2, age<20 & age>5 & sex==2)
boys <- subset(juul2, age<20 & age>5 & sex==1)
young <- subset(juul2, age<20 & age>5)
stval <- c(alpha=exp(5.3),beta=exp(0.42),gamma=0.15)
fit.boys <- nls(height~alpha*exp(-beta*exp(-gamma*age)),
       start=stval, data=boys)
fit.girls <- nls(height~alpha*exp(-beta*exp(-gamma*age)),
       start=stval, data=girls)
fit.young <- nls(height~alpha*exp(-beta*exp(-gamma*age)),
       start=stval, data=young)
```

To test whether we can use the same model for boys and girls, there are two approaches. One is to make an F test based on the three fits above:

```
ms.pooled <- (deviance(fit.boys) + deviance(fit.girls))/(499+625)
ms.diff <- (deviance(fit.young) -
            deviance(fit.boys) - deviance(fit.girls))/3
ms.diff/ms.pooled
```

This gives $F = 90.58$ on 3 and 1124 degrees of freedom, which is of course highly significant.

Alternatively, we can set up the joint model with separate parameters for boys and girls and test whether it fits the data better than the model with the same parameters, like this:

```
fit.young2 <- nls(height~(alpha+da*(sex==1))*
                    exp(-(beta+db*(sex==1))*
                    exp(-(gamma+dg*(sex==1))*age)),
           start=c(alpha=exp(5.3),beta=exp(0.42),gamma=0.15,
           da=0, db=0, dg=0), data=young)
summary(fit.young2)
anova(fit.young, fit.young2)
```

Notice that `da`, `db`, and `dg` represent differences between the parameters for the two genders. The term `sex==1` is 0 for girls and 1 for boys.

16.2 We consider experiment 1 only. Starting values can be eyeballed by using the observation at zero dose for $y_{\max}$ and the x (`dose`) value at approximately $y_{\max}/2$ as β. The value of α can be guessed as the fixed constant 1. Below, we use α via its logarithm, called `la`.

```
e1 <- subset(philion, experiment==1)
fit <- nls(sqrt(response) ~ sqrt(ymax / (1 + (dose/ec50)^exp(la))),
        start=list(ymax=28, ec50=.3, la=0), data=e1,
        lower=c(.1,.0001,-Inf), algorithm="port")
summary(fit)
confint(fit)
p <- profile(fit, alphamax=.2)
par(mfrow=c(3,1))
plot(p)
confint(p)
```

16.3 The alternative model has similar tail behaviour but behaves differently when x is close to zero. (In particular, the original model has a term proportional to $-x^{\alpha-1}$ in its derivative. At zero, this is $-\infty$ when $\alpha < 1$ and 0 when $\alpha > 1$, so the model describes curves that are either very steep or very flat near zero.) Notice that, in the modified model, β is no longer the EC50; the latter is now the solution for x of $(1 + x/\beta)^{\alpha} = 2$. The two are connected by a factor of $2^{1/\alpha} - 1$.

```
e1 <- subset(philion, experiment==1)
fit1 <- nls(sqrt(response) ~ sqrt(ymax / (1 + dose/b)^exp(la)),
        start=list(ymax=28, b=.3, la=0), data=e1,
        lower=c(.1,.0001,-Inf), algorithm="port")
summary(fit1)
fit2 <- nls(sqrt(response) ~ sqrt(ymax / (1 +
        dose/(d50/(2^(1/exp(la))-1)))^exp(la)),
        start=list(ymax=28, d50=.3, la=0), data=e1,
        lower=c(.1,.0001,-Inf), algorithm="port")
summary(fit2)
```

Here, `fit1` and `fit2` are equivalent models, except that the latter is parameterized in terms of `ec50`. We can compare the fitted curve with the model from the previous exercise as follows:

```
dd <- seq(0,1,,200)
yy <- predict(fit, newdata=data.frame(dose=dd))
y1 <- predict(fit2, newdata=data.frame(dose=dd))
matplot(dd,cbind(yy,y1)^2, type="l")
```

(Notice that the fitted values should be squared because of the square-root transformation in the models.)

Bibliography

Agresti, A. (1990), *Categorical Data Analysis*, John Wiley & Sons, New York.

Altman, D. G. (1991), *Practical Statistics for Medical Research*, Chapman & Hall, London.

Andersen, P. K., Borgan, Ø., Gill, R. D., and Keiding, N. (1991), *Statistical Models Based on Counting Processes*, Springer-Verlag, New York.

Armitage, P. and Berry, G. (1994), *Statistical Methods in Medical Research*, 3rd ed., Blackwell, Oxford.

Bates, D. M. and Watts, D. G. (1988), *Nonlinear regression analysis and its applications*, John Wiley & Sons, New York.

Becker, R. A., Chambers, J. M., and Wilks, A. R. (1988), *The NEW S Language*, Chapman & Hall, London.

Breslow, N. E. and Day, N. (1987), *Statistical Methods in Cancer Research. Volume II: The Design and Analysis of Cohort Studies*, IARC Scientific Publications, Lyon.

Campbell, M. J. and Machin, D. (1993), *Medical Statistics. A Commonsense Approach*, 2nd ed., John Wiley & Sons, Chichester.

Chambers, J. M. and Hastie, T. J. (1992), *Statistical Models in S*, Chapman & Hall, London.

Clayton, D. and Hills, M. (1993), *Statistical Models in Epidemiology*, Oxford University Press, Oxford.

Cleveland, W. S. (1994), *The Elements of Graphing Data*, Hobart Press, New Jersey.

Cochran, W. G. and Cox, G. M. (1957), *Experimental Designs*, 2nd ed., John Wiley & Sons, New York.

Cox, D. R. (1970), *Analysis of Binary Data*, Chapman & Hall, London.

Cox, D. R. and Oakes, D. (1984), *Analysis of Survival Data*, Chapman & Hall, London.

Everitt, B. S. (1994), *A Handbook of Statistical Analyses Using S-PLUS*, Chapman & Hall, London.

Hájek, J., Šidák, Z., and Sen, P. K. (1999), *Theory of Rank Tests*, 2nd ed., Academic Press, San Diego.

Hald, A. (1952), *Statistical Theory with Engineering Applications*, John Wiley & Sons, New York.

Hosmer, D. W. and Lemeshow, S. (2000), *Applied Logistic Regression*, 2nd ed., John Wiley & Sons, New York.

Johnson, R. A. (1994), *Miller & Freund's Probability & Statistics for Engineers*, 5th ed., Prentice-Hall, Englewood Cliffs, NJ.

Kalbfleisch, J. D. and Prentice, R. L. (1980), *The Statistical Analysis of Failure Time Data*, John Wiley & Sons, New York.

Lehmann, E. L. (1975), *Nonparametrics, Statistical Methods Based on Ranks*, McGraw-Hill, New York.

Matthews, D. E. and Farewell, V. T. (1988), *Using and Understanding Medical Statistics*, 2nd ed., Karger, Basel.

McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, 2nd ed., Chapman & Hall, London.

Murrell, P. (2005), *R Graphics*, Chapman & Hall/CRC, Boca Raton, Florida.

Siegel, S. (1956), *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill International, Auckland.

Venables, W. N. and Ripley, B. D. (2000), *S Programming*, Springer-Verlag, New York.

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S*, 4th ed., Springer-Verlag, New York.

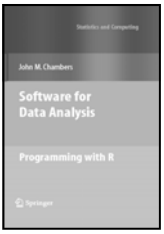
Weisberg, S. (1985), *Applied Linear Regression*, 2nd ed., John Wiley & Sons, New York.

Zar, J. H. (1999), *Biostatistical Analysis*, Prentice Hall, Englewood Cliffs, NJ.

Index

Springer
the language of science

springer.com



Software for Data Analysis
Programming with R

John M. Chambers

This book guides the reader through programming with R, beginning with simple interactive use and progressing by gradual stages, starting with simple functions. More advanced programming techniques can be added as needed, allowing users to grow into software contributors, benefiting their careers and the community. R packages provide a powerful mechanism for contributions to be organized and communicated.

2008. Approx. 510 pp. (Statistics and Computing) Hardcover
ISBN 978-0-387-75935-7

Time Series Analysis with Applications in R

Jonathan D. Cryer and Kung-Sik Chan

Time Series Analysis With Applications in R, Second Ed., presents an accessible approach to understanding time series models and their applications. Although the emphasis is on time domain ARIMA models and their analysis, the new edition devotes two chapters to the frequency domain and three to time series regression models, models for heteroscedasticty, and threshold models. All of the ideas and methods are illustrated with both real and simulated data sets. A unique feature of this edition is its integration with the R computing environment.

2008. 2nd Ed., 494 pp. (Springer Texts in Statistics) Hardcover
ISBN 0-387-75958-6

Data Manipulation with R

Phil Spector

This book presents a wide array of methods applicable for reading data into R, and efficiently manipulating that data. In addition to the built-in functions, a number of readily available packages from CRAN (the Comprehensive R Archive Network) are also covered. All of the methods presented take advantage of the core features of R: vectorization, efficient use of subscripting, and the proper use of the varied functions in R that are provided for common data management tasks.

2008. 164 pp. (Use R) Softcover
ISBN 978-0-387-74730-9
