Eine Einführung in R

8. März 2004

1 Erste Bemerkungen

R bietet verschiedene Hilfen an. Kennt man den Namen eines R Kommando (z.B. `solve`) und will darüber Auskunft, so liefert `help(solve)` oder `?solve` eine genaue Beschreibung zu diesem Befehl. Hilfe in HTML Format erhält man mit `help.start()`, wodurch ein Web Browser gestartet wird. Beispiele zu einem Thema sind in `example(solve)` angegeben. Für eine generelle Hilfe verwende man

```
> apropos("solve")
[1] "backsolve"     "forwardsolve"  "qr.solve"      "solve"
[5] "solve.default" "solve.qr"
```

Sind R Befehle auf einer externen Datei (z.B. `commands.R`), so können diese durch `source("commands.R")` eingelesen und ausgeführt werden. Die Default-Extension ist dabei `*.R`. Der Output kann auf eine externe Datei (z.B. `record.lis`) durch `sink("record.lis")` umgeleitet werden. Zurücksetzen auf die Konsole erfolgt mit `sink()`.
Während einer R-Session werden Objekte bei ihrem Namen abgespeichert. Der Befehl `objects()` oder `ls()` liefert eine Liste mit allen Namen der zur Zeit in R gespeicherten Objekte. Gelöscht werden Objekte (z.B. `temp` und `x`) mittels `rm(temp, x)`. Alle Objekte die in einer Session erzeugt wurden können permanent in eine Datei für spätere Sessions gespeichert werden. Am Ende einer Session (vor `q()`) hat man dafür die Möglichkeit, alle Objekte auf die Datei `.Rdata` im `working directory` (aktuelles Verzeichnis) zu schreiben. Bei einem späteren Start wird dann dieser `workspace` (wie auch das entsprechende `command history`) von dort geladen.
Oft möchte man mehrere unterschiedliche `workspaces` verwenden (z.B. für verschiedene Projekte). Dazu erzeugt man eigene shortcuts für jedes Projekt. Alle Pfade sind relativ zum `starting directory`. Setze daher das `Start in` Feld (in den jeweiligen shortcuts) für separate Projekte. Alternativ nutze drag-and-drop und bringe `.RData` auf den R shortcut.

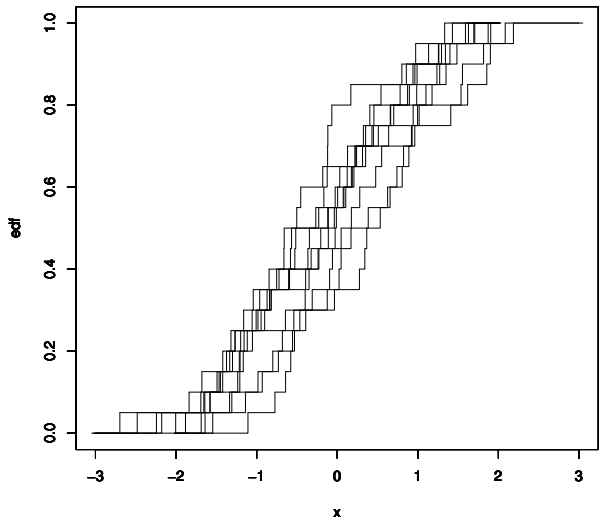
2 Momente und Quantile

```
> n <- 100; x <- rnorm(n, 5, 2)
> mean(x)         # arithmetic mean
[1] 4.991007
> (prod(x))^(1/n) # geometric mean
[1] 4.60635
> n/sum(1/x)      # harmonic mean
[1] 4.070126

> var(x)
[1] 3.123983
> sd(x)
[1] 1.767479
> sd(x) / mean(x) # coefficient of variation
[1] 0.3541328
```

```
> (sum((x - mean(x))^3)/n) / (sd(x)^3)     # Skewness
[1] -0.1052344
> (sum((x - mean(x))^4)/n) / (sd(x)^4) - 3 # Excess
[1] -0.5735405

> median(x)
[1] 5.041436
> quantile(x, probs = seq(0, 1, 1/4))
       0%       25%       50%       75%      100%
0.6125487 3.9130137 5.0414359 6.3409851 8.6175029
> fivenum(x)
[1] 0.6125487 3.8891774 5.0414359 6.3597095 8.6175029
> range(x) # min(x); max(x)
[1] 0.6125487 8.6175029

> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.6125  3.9130  5.0410  4.9910  6.3410  8.6180

> my.summary <- function(data) {  # write your own "summary" function
+   mys <- c(0, 0, 0, 0, 0) # initialize output
+   mys[1] <- mean(data)
+   mys[2] <- median(data)
+   mys[3] <- sd(data)
+   mys[4] <- (quantile(data, 0.75)-quantile(data, 0.25)) / 1.349
+   mys[5] <- length(data)
+   names(mys) <- c("mean", "median", "std.dev(mom)", "std.dev(iqr)", "n")
+   return(mys)
+ }
> my.summary(x)
        mean       median std.dev(mom) std.dev(iqr)            n
    4.991007     5.041436     1.767479     1.799831   100.000000
```

3 Empirische Verteilungsfunktion

```
> edf <- function(x, ...){
+   s <- sort(x);   n <- length(x)
+   Fn <- c(0, (1:n)/n, 1);  eps <- min(diff(s))
+   min <- floor(s[1]) - 2*eps;   max <- ceiling(s[n]) + 2*eps
+   plot(c(min, s, max), Fn, type = "s", ylab = "edf", ...)
+ }

> par(mfrow=c(3,4)) # series of several edf's on 1 page
> for (i in 1:12)
+   edf(rnorm(20), xlim=c(-3, 3), xlab="x", ylim=c(0, 1))

> par(mfrow=c(1,1)) # edf's in 1 plot on 1 page
> for (i in 1:10){
+   edf(rnorm(20), xlim=c(-3, 3), xlab="x", ylim=c(0, 1))
+   par(new = T) # don't open new graphic window
+ }
> par(new = F) # reset to new graphic window
```



4 Weitere Graphische Verfahren

Boxplot: `boxplot(fvc, notch=F, horizontal=T)`

Histogramm: `hist(fvc, breaks, freq=T)`
`breaks`: Vektor mit Intervallsgrenzen (`b <- seq(500, 600, 10)`), oder Stabanzahl (`b <- 10`), der Name des Algorithmus (Sturges, Scott, oder FD), oder der Name einer selbstdefinierten Funktion.
Falls `freq=TRUE` werden absolute Häufigkeiten gezeichnet, sonst relative Häufigkeiten.

Kernschätzer: `density(fvc, bw, kernel)`
`bw` ist der Glättungsparameter (bandwidth) oder der Name der zu verwendenden Regel (siehe `bw.nrd`)
`kernel` bezeichnet den Kern, z.B. Gaussian, epanechnikov, rectangular, triangular, biweight, cosine.
Der Aufruf von `density` generiert noch keine Graphik sondern führt nur Berechnungen durch. Die Graphik selbst wird durch `plot(density(...))` erzeugt.

Empirische Verteilungsfunktion: `ecdf(fvc)`
Diese Funktion befindet sich in der Bibliothek `library(stepfun)`, z.B.:

```
plot(ecdf(y), do.points=FALSE, verticals=TRUE)
```

Oft wird der Graph der Normalverteilungsfunktion zusätzlich eingezeichnet:

```
x <- seq(400, 800, 20); lines(x, pnorm(x, mean=mean(fvc),
sd=sd(fvc)))
```

Theoretischer Quantil-Quantil-Plot: `qqplot, ppoints, qqnorn, qqline`
Allgemeiner QQ-Plot, hier der Vergleich mit t_5-Quantilen

```
qqplot(qt((seq(1:79)-1/2)/79, df=5), fvc , xlab="t_5 Quantiles") #
oder einfacher mit ppoints(n, a) qqplot(qt(ppoints(n=length(fvc),
a=0.5), df=5), fvc , xlab="t_5 Quantiles")
```

Zwei spezielle Befehle werden für den Normal-QQ-Plot angeboten:

```
qqnorm(fvc) # und dazu noch eine Gerade durch das 1. und 3.
Datenquartil qqline(fvc)
```

5 Hypothesen-Tests

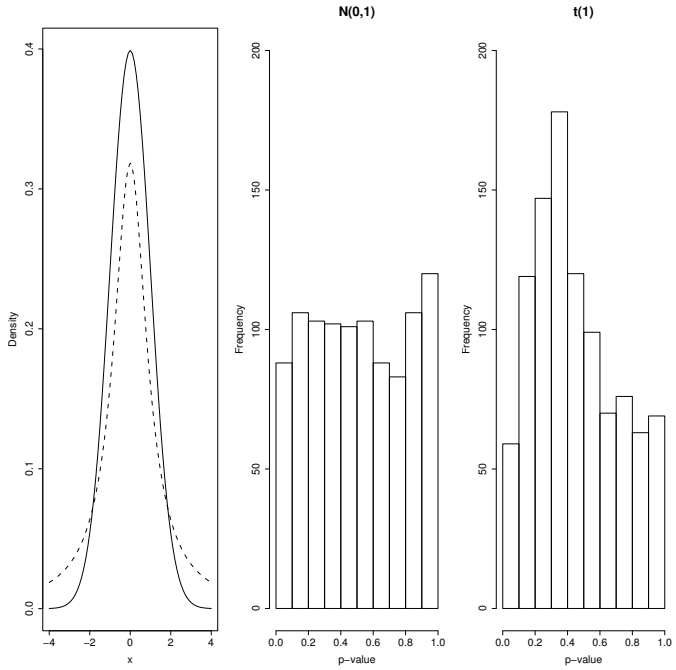
Folgende Verfahren sind in R (package:ctest) implementiert:

Alternative	1 Gruppe		2 Gruppen		k Gruppen	
	param.	nichtparam.	param.	nichtparam.	param.	nichtparam.
Allgemein	chisq.test	shapiro.test	chisq.test	ks.test	chisq.test	
Lokation	t.test	wilcox.test	t.test	wilcox.test	oneway.test	kruskal.test
Variabilität			var.test	ansari.test	bartlett.test	fligner.test
				mood.test		
Proportionen	binom.test		prop.test		prop.test	

5.1 t-Test

`t.test(x, y = NULL, alternative = "two.sided", mu = 0, paired = F, var.equal = F, conf.level = 0.95)`:
Student's t-Test für den Ein- und Zwei-Stichproben Fall. Getestet wird auf `mu`, den wahren Erwartungswert der `x` oder auf die wahre Differenz der Erwartungswerte von `x` und `y`. Der logische Indikator `paired` erlaubt im Zwei-Stichproben Fall den t-Test für verbundene Stichproben. Mittels `var.equal` werden gleiche oder verschiedene Varianzen angenommen.

```
> petrol <- c(11.5, 11.8, 12.0, 12.4, 12.5, 12.6, 12.8, 12.9, 13.0, 13.2)
> t.test(petrol, mu=12)

        One Sample t-test
data: petrol
t = 2.6932, df = 9, p-value = 0.02467
alternative hypothesis: true mean is not equal to 12
95 percent confidence interval:
 12.07522 12.86478
sample estimates:
mean of petrol
    12.47

> # MC estimate of type I error
> R <- 1000; n <- 50; pn <- pt <- 1:R
> for (r in 1:R) {
+   z <- rnorm(n);   t <- rt(n, 1)
+   pn[r] <- t.test(z)$p.value
+   pt[r] <- t.test(t)$p.value
+ }
> c(sum(pn > 0.95),  sum(pt > 0.95))
[1] 60  28

> par(mfrow = c(1, 3))
> plot(seq(-4, 4, .1), dnorm(seq(-4, 4, .1)), type="l", xlab="x", ylab="Density")
> lines(seq(-4, 4, .1), dt(seq(-4, 4, .1), 1), lty=2) # adds 'lines' to the plot

> hist(pn, xlab="p-value", main="N(0,1)", ylim=c(0,200)) # shows uniform behaviour
> hist(pt, xlab="p-value", main="t(1)", ylim=c(0,200))   # no longer uniformly distributed
```




```
> power.t.test(n=length(petrol), delta= 1, type = "one.sample")  # Power calculations

     One-sample t test power calculation
              n = 10
          delta = 1
             sd = 1
      sig.level = 0.05
          power = 0.803
    alternative = two.sided

> # alternatively, specify Power of test (1 minus Type II error probability)
> power.t.test(power=0.90, delta= 1, type = "one.sample")

     One-sample t test power calculation
              n = 12.6
          delta = 1
             sd = 1
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

5.2 Kolmogorov-Smirnov Test

`ks.test(x, y, exact = T, alternative = "two.sided")`:
Kolmogorov-Smirnov Test für den Ein- und Zwei-Stichproben Fall. `y` ist ein Vektor mit Daten (2 Gruppen Vergleich) oder bezeichnet den Namen einer Verteilung, z.B. `y = "pgamma", 2, 3` für den Vergleich mit der $\mathrm{N}(2, 3^2)$-Verteilung. Wenn keine Bindungen vorliegen und $n_x n_y < 10000$ ist, liefert `exact=T` den exakten p-Wert im zweiseitigen, 2 Stichproben Fall. `alternative` kann auch `"less"` oder `"greater"` sein.

```
> ks.test(petrol, "pnorm", 12, 1, alternative="two.sided") # Testing N(12, 1)

        One-sample Kolmogorov-Smirnov test
```

```
data:  petrol
D = 0.3554, p-value = 0.1598
alternative hypothesis: two.sided

> c(mean(petrol), sd(petrol))
[1] 12.470  0.552
# Testing N(12.47, 0.552) => KS-Test conservative
> ks.test(petrol, "pnorm", mean(petrol), sd(petrol), alternative="two.sided")

        One-sample Kolmogorov-Smirnov test
data:  petrol
D = 0.149, p-value = 0.9787
alternative hypothesis: two.sided
```

5.3 Shapiro-Wilk Test auf Normalverteilung

`shapiro.test(x)`:
Shapiro-Wilk's W- Test auf Normalverteilung. Der Vektor `x` beinhaltet die Daten.

```
> shapiro.test(petrol)

        Shapiro-Wilk normality test
data:  petrol
W = 0.9529, p-value = 0.7026
```

5.4 χ^2 Test

`chisq.test(x, y = NULL, correct = T, p = rep(1/length(x), length(x)), simulate.p.value = F, B = 2000)`:
Pearson's χ^2-Test für nichtnegative Zähldaten `x`, `y`. Hierbei ist `x` entweder ein Vektor oder eine Matrix. Falls `x` eine Matrix ist, wird `y` ignoriert.
Ist `x` eine Matrix mit nur einer Zeile oder einer Spalte, oder ist `x` ein Vektor und `y` nicht gegeben, so wird `x` als eindimensionale Kontigenztafel betrachtet und die Hypothese getestet, dass die Populationswahrscheinlichkeiten denen in `p` entsprechen. Hierbei hat `p` dieselbe Länge wie `x` und beinhaltet die hypothetischen Wahrscheinlichkeiten in den Klassen (Zellen der Kontingenztafel).
Ist `x` eine Matrix mit mindestens zwei Zeilen und Spalten, wird `x` als zweidimensionale Kontingenztafel gesehen und die Unabhängigkeitshypothese getestet. Bei `correct = T` wird bei 2×2 Tafeln die Stetigkeitskorrektur zur Berechnung der Teststatistik verwendet. Ist `simulate.p.value = T` wird der p-Wert durch Monte-Carlo Simulation mit `B` Replikationen berechnet. Diese Simulation basiert auf zufälliges Ziehen aus allen Kontingenztafeln mit gleicen Rändern und funktioniert daher nur falls alle Randhäufigkeiten positiv sind. Unter anderem liefert diese Funktion die `observed` und unter H_0 `expected` Häufigkeiten.

```
> x <- c(20, 30, 20, 25, 15, 10)  # Fairness of a die (Test Problem A)
> chisq.test(x)

        Chi-squared test for given probabilities
data:  x
X-squared = 12.5, df = 5, p-value = 0.02854
```

5.5 Binomialtest

```
> s <- 17; t <- 20 # or x <- (successes, failures)
> p <-  0.95 # success probability
> binom.test(s, t, p, alternative = "less", conf.level = 0.95)
```

```
        Exact binomial test
data:  s and t
number of successes = 17, number of trials = 20, p-value = 0.07548
alternative hypothesis: true probability of success is less than 0.95
95 percent confidence interval:
 0.0000000 0.9578306
sample estimates:
probability of success
                 0.85
```

5.6 Vorzeichentest

```
> h <- c(179, 177, 178, 174, 170, 185, 175, 179, 176, 169, 186, 189, 168, 170, 174)
> binom.test(sum(h - 180 > 0), length(h), p = 1/2, conf.level = 0.90)

        Exact binomial test
data:  sum(h - 180 > 0) and length(h)
number of successes = 3, number of trials = 15, p-value = 0.03516
alternative hypothesis: true probability of success is not equal to 0.5
90 percent confidence interval:
 0.05684687 0.43978444
sample estimates:
probability of success
                  0.2
```

5.7 Wilcoxon Vorzeichen-Rangtest für den Median

`wilcox.test(x, y = NULL, alternative = "two.sided", mu = 0, paired = F, exact = T, correct = T, conf.int = F, conf.level = 0.95)`:
Wilcoxon-Test für den Ein- und Zwei-Stichproben Fall. Zusätzlich zu den in `t.test` beschriebenen Parametern erlaubt `exact = T` die exakte Berechnung des p-Werts, und bei `correct = T` die Verwendung der Stetigkeitskorrektur in der Normalapproximation. Nur bei `conf.int = F` wird ein nichtparametrisches Konfidenzintervall berechnet. Ist nur `x` gegeben, oder ist `pairted =T` bei `x` und `y`, wird die Hypothese getestet, dass die Verteilung von `x` oder die von `x` − `y` symmetrisch um `mu` ist. Falls `x` und `y` gegeben sind und `pairted =F` gesetzt ist, wird auf einen Lokationsunterschied `mu` in den Verteilungen von `x` und `y` getestet.

```
> wilcox.test(h, mu=180)

        Wilcoxon signed rank test with continuity correction
data:  h
V = 26.5, p-value = 0.06053
alternative hypothesis: true mu is not equal to 180

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(h, mu = 180)

> a <- c(3.5, 4.5, 4.0, 0.5, 2.5, 7.0, 8.5, 8.0)  # another example w/o ties
> wilcoxon <- wilcox.test(a, mu=5)
wilcoxon

        Wilcoxon signed rank test
data:  a
V = 17, p-value = 0.9453
alternative hypothesis: true mu is not equal to 5
```

```
# Notice:  dsignrank, psignrank, qsignrank, rsignrank are also available
> wilcoxon$p.value
[1] 0.9453125
> qsignrank(wilcoxon$p.value, length(a))
[1] 29
> qsignrank(wilcoxon$p.value/2, length(a))
[1] 17

# recalculate the above Wilcoxon statistic by using vectors
> d <- a - 5
> r.absd <- rank(abs(d))
> z <- (d >= 0)
> w <- t(z) %*% r.absd;  w  # now a 1*1 matrix
     [,1]
[1,]   17
```

`oneway.test(formula, var.equal = F)`:
Einfache Varianzanalyse testet (zweiseitig) auf Gleichheit der Erwartungswerte von k normalverteilte Stichprobengruppen. Hier hat `formula` die Form `x` $\sim$ `g`, wobei der Vektor `x` die Daten aller k Gruppen beinhaltet und der Vektor `g` den Faktor mit der Gruppenzugehörigkeit bezeichnet, z.B. `fvc` $\sim$ `ort` für den Test auf gleiche Erwartungswerte in allen Ortsgruppen. Mit `var.equal = T` können gleiche Varianzen für allen Gruppen angenommen werden.

`kruskal.test(x, g)`:
(Zweiseitiger) Kruskal-Wallis Test auf Gleichheit der Lokationsparameter in allen k Gruppen. Der Vektor `x` beinhaltet alle Daten und der Faktor `g` deren Gruppenzugehörigkeit.

`var.test(x, y, ratio = 1, alternative = "two.sided", conf.level = 0.95)`:
F-Test zum Vergleich der Varianzen von zwei normalverteilten Stichproben. Unter der Nullhypothese ist der Quotient der beiden Populationsvarianzen gleich dem in `ratio` spezifizierten Wert.

`ansari.test(x, y, alternative = "two.sided", exact = T, conf.int = F, conf.level = 0.95)`:
Ansari-Bradley Test auf unterschiedliche Skalenparameter zweier Gruppen. Optional wird bei `conf.int = T` ein nichtparametrisches Konfidenzintervall für θ berechnet, dem Quotienten der beiden Skalenparameter. Hierbei würde θ^2 dem Varianzenquotient beim F-Test entsprechen.

`mood.test(x, y, alternative = "two.sided")`:
Mood's Zwei-Stichproben Test auf Unterschied in den Skalenparametern.

`bartlett.test(x, g)`:
Bartlett's Test auf Varianzhomogenität von k normalverteilte Gruppen.

`fligner.test(x, g)`:
Fligner-Killeen (Median) Test auf Varianzhomogenität von k Gruppen (Lineare Rangstatistik).

`binom.test(x, n, p = 0.5, alternative = "two.sided", conf.level = 0.95)`:
Exakter Binomialtest über die Erfolgswahrscheinlichkeit in einem Bernoulli-Experiment. `x` ist die beobachtete Anzahl von Erfolgen, oder ein zweielementiger Vektor mit der Anzahl an Erfolgen und der Anzahl der Versuche. `n` ist die Anzahl der Versuche und wird ignoriert, falls `x` Länge 2 hat. Die hypothetische Erfolgswahrscheinlichkeit ist `p`.

`prop.test(x, n, p = NULL, alternative = "two.sided", conf.level = 0.95, correct = T)`:
Damit testet man die Hypothese, dass die Erfolgswahrscheinlichkeiten in mehreren Gruppen gleich sind, oder dass sie bestimmten Werten entsprechen. Hier ist `x` ein Vektor mit den Erfolgsanzahlen oder eine Matrix mit 2 Spalten mit der Erfolgs- und Misserfolgsanzahl in den Gruppen. Der Vektor `n` beinhaltet die Anzahl an Versuchen und wird ignoriert, falls `x` eine Matrix ist. Die hypothetischen Erfolgswahrscheinlichkeiten sind im Vektor `p` gegeben, der dieselbe Länge wie `x` haben muss. `alternative` wird nur beim ein- oder zweiseitigen Test auf Gleichheit einer einzelnen Wahrscheinlichkeit mit einem speziellen Wert,

oder auf Gleichheit zweier Wahrscheinlichkeiten verwendet. In diesen Fällen wirkt auch `conf.level`, das sich dann auf ein Konfidenzintervall für die wahre Erfolgswahrscheinlichkeit eines Experiments, oder auf ein Konfidenzintervall für die Differenz zweier Erfolgswahrscheinlichkeiten bezieht.

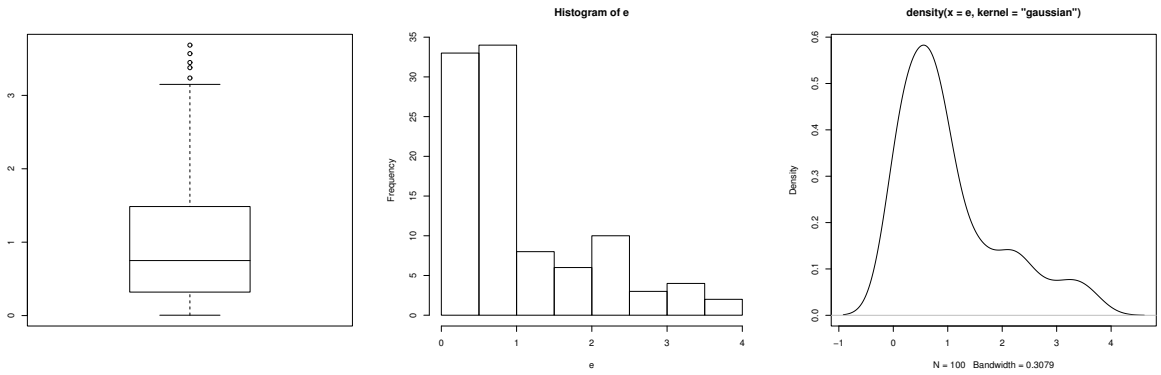
6 Vektoren und Matrizen

```
> A <- matrix(c(1, 3, 2, 5, 6, 4), byrow=T, ncol=2);  A  # Create a 3*2 matrix
     [,1] [,2]
[1,]    1    3
[2,]    2    5
[3,]    6    4
> B <- matrix(c(2, 2, 3, 1), byrow=T, ncol=2);  B # Create a 2*2 matrix
     [,1] [,2]
[1,]    2    2
[2,]    3    1

> A %*% B  # Matrix multiplication (operator '%*%')
     [,1] [,2]
[1,]   11    5
[2,]   19    9
[3,]   24   16
> B^2  # Square each element of a matrix
     [,1] [,2]
[1,]    4    4
[2,]    9    1
> B %*% B  # Multiply a square matrix by itself
     [,1] [,2]
[1,]   10    6
[2,]    9    7
> solve(B)  # Find the inverse of a square matrix
      [,1] [,2]
[1,] -0.25  0.5
[2,]  0.75 -0.5
> diag(B)  # Extract the diagonal of a square matrix
[1] 2 1
> sqrt(diag(B))
[1] 1.414214 1.000000
> c(A)  # Concatenate a matrix (i.e. stack the columns)
[1] 1 2 6 3 5 4
> t(A)  # Transpose a matrix
     [,1] [,2] [,3]
[1,]    1    2    6
[2,]    3    5    4
> cbind(t(A),B)  # Bind the columns of two matrices
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    6    2    2
[2,]    3    5    4    3    1
```

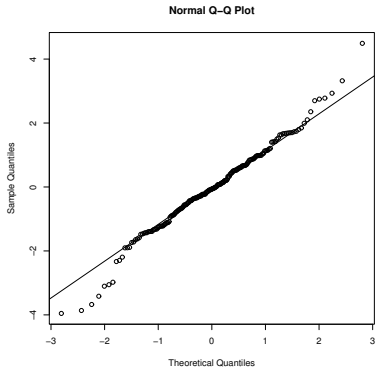
7 Explorative Verfahren

```
> e <- rexp(100, 1);  summary(e)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00507 0.32300 0.74900 1.03000 1.47000 3.68000
```

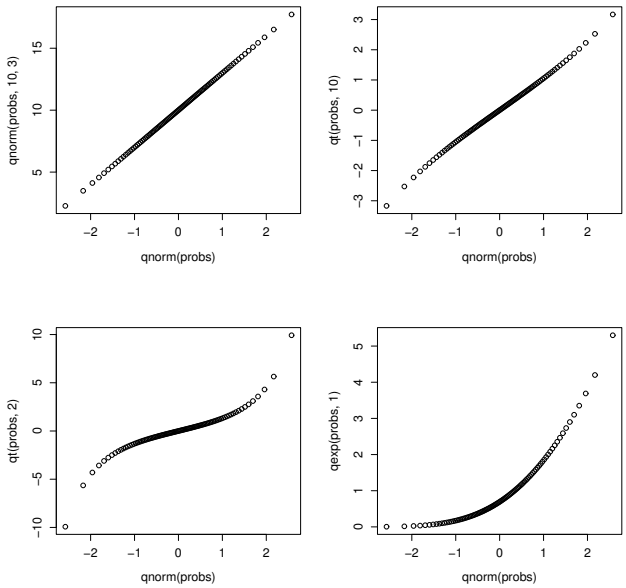
```
> boxplot(e) # Boxplot
> b <- pretty(e, 8);  b
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
> hist(e, breaks=b) # Histogram

> d <- density(e, kernel="gaussian");  d   # Kernel Density Estimates
Call:
       density(x = e, kernel = "gaussian")

Data: e (100 obs.);     Bandwidth 'bw' = 0.3079

       x                y
 Min.   :-0.919   Min.   :0.000206
 1st Qu.: 0.463   1st Qu.:0.051662
 Median : 1.845   Median :0.118579
 Mean   : 1.845   Mean   :0.180725
 3rd Qu.: 3.227   3rd Qu.:0.261892
 Max.   : 4.608   Max.   :0.582772

> plot(d)
```



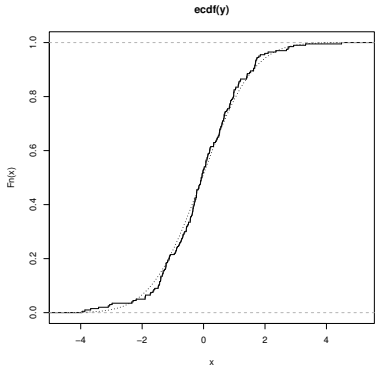

```
> y <- rt(200, df = 5)  # Quantile-Quantile-Plots
> qqnorm(y) # Normal QQ Plot
> qqline(y)  # adds a line to a normal QQ plot passing through Q(.25) and Q(.75)
```




```
> n <- 100;  probs <- (1:n - 1/2)/n; par(mfrow = c(2, 2))
> plot(qnorm(probs), qnorm(probs, 10, 3));  plot(qnorm(probs), qt(probs, 10))
> plot(qnorm(probs), qt(probs, 2));  plot(qnorm(probs), qexp(probs, 1))
```




```
> # Empirical Cumulative Distribution Function in Standard Package "Stepfun"
> library(stepfun)
> plot(ecdf(y), do.points=FALSE, verticals=TRUE)
> x <- seq(-4.5, 4.5, 0.1)
> lines(x, pnorm(x, mean=mean(y), sd=sd(y)), lty=3)
```



8 Daten auf einer externen Datei

Zuerst sollte man das Working Directory von `R` auf jenes, in dem die Daten zu finden sind, wechseln. Dazu klickt man im `RGui` auf `File`, folgt `Change dir ...` und wechselt in das entsprechende Verzeichnis. Angenommen, es befindet sich dort eine ASCII-Datei (mit den zu analysierenden Daten) namens `houses.dat`. Eingelesen könnnen diese $n = 93$ Beobachtungen mit $p = 5$ Variablen als Matrix werden.

```
> X <- matrix(scan("houses.dat"), byrow=T, ncol=5)
```

Zugriff erfolgt dann mittels Matrixsyntax:

```
> X[45,1]
[1] 87.9
> X[ ,1]
 [1]  48.5  55.0  68.0 137.0 309.4  17.5  19.6  24.5  34.8  32.0  28.0  49.9
```

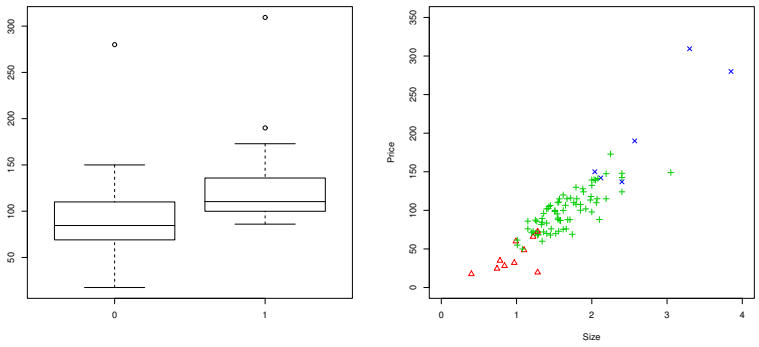
```
...

> mean(X[ ,1])
[1] 99.53333
> m <- apply(X, 2, mean);  m
[1] 99.5333333  1.6496774  3.1827957  1.9569892  0.3010753

> v <- apply(X, 2, var);  v
[1] 1952.2376812    0.2758988    0.3683964    0.1720430    0.2127162
> s <- sqrt(v);  s
[1] 44.1841338  0.5252607  0.6069567  0.4147807  0.4612117
> var(X)
            [,1]        [,2]        [,3]        [,4]       [,5]
[1,] 1952.237681 20.85983696 15.82971014 13.07971014 7.26594203
[2,]   20.859837  0.27589881  0.21332048  0.14433380 0.04270687
[3,]   15.829710  0.21332048  0.36839645  0.08403460 0.07480131
[4,]   13.079710  0.14433380  0.08403460  0.17204301 0.03482936
[5,]    7.265942  0.04270687  0.07480131  0.03482936 0.21271622
```

Günstiger ist es sicherlich einen Datensatz zu definieren

```
> HP <- read.table("house.dat", col.names=c("Price", "Size", "Bed", "Bath", "New"))
```

Jetzt kann man auf die Daten mit `HP$Price` zugreifen. Dazu sollten diese sich bereits in der `objects()` Liste aller vorhandenen Objekte befinden. Günstiger ist es, den Datensatz mittels (`attach(HP)`) in den Suchpfad zu geben. Jetzt ist `HP` ein Element der `search()` Liste.

```
> options(digits=3);  cor(HP)
      Price  Size   Bed  Bath   New
Price 1.000 0.899 0.590 0.714 0.357
Size  0.899 1.000 0.669 0.662 0.176
Bed   0.590 0.669 1.000 0.334 0.267
Bath  0.714 0.662 0.334 1.000 0.182
New   0.357 0.176 0.267 0.182 1.000
> summary(HP)
     Price            Size           Bed            Bath           New
 Min.   : 17.5   Min.   :0.40   Min.   :1.00   Min.   :1.00   Min.   :0.000
 1st Qu.: 72.9   1st Qu.:1.33   1st Qu.:3.00   1st Qu.:2.00   1st Qu.:0.000
 Median : 96.0   Median :1.57   Median :3.00   Median :2.00   Median :0.000
 Mean   : 99.5   Mean   :1.65   Mean   :3.18   Mean   :1.96   Mean   :0.301
 3rd Qu.:115.0   3rd Qu.:1.98   3rd Qu.:4.00   3rd Qu.:2.00   3rd Qu.:1.000
 Max.   :309.4   Max.   :3.85   Max.   :5.00   Max.   :3.00   Max.   :1.000
> boxplot(Price~New) # boxplots of both (new/old) groups

> plot(Size, Price, type="n", ylim=c(0,350), xlim=c(0,4)) # don't plot the points
> points(Size, Price, pch=Bath+1, col=Bath+1) # use plotting characters and colour
```



Häufigkeitstabellen von Faktoren:

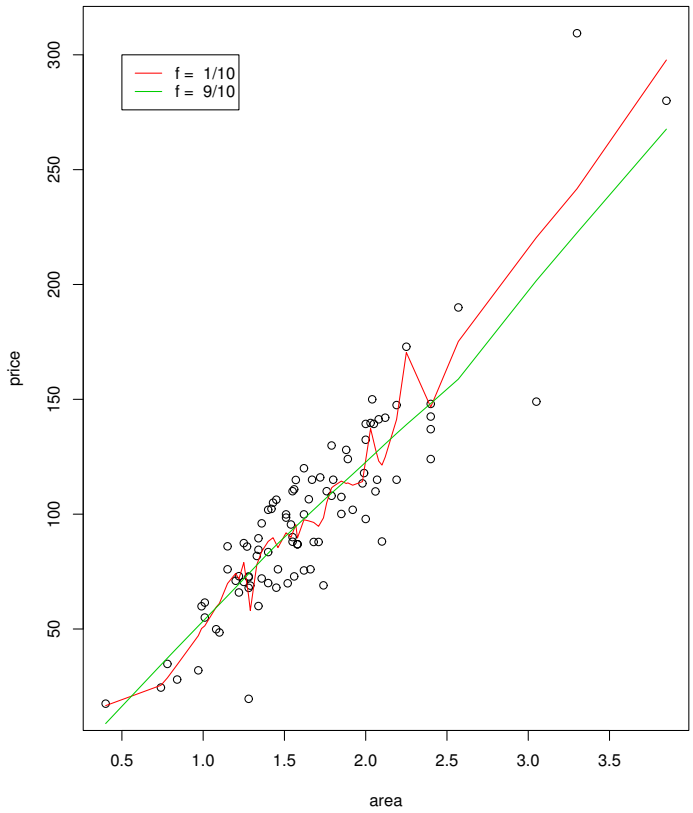
```
> Price
[1]  48.5  55.0  68.0 137.0 309.4  17.5  19.6  24.5  34.8  32.0  28.0  49.9
...
> cut(Price, breaks=15+60*(0:5))
[1] (15,75]   (15,75]   (15,75]   (135,195] (255,315] (15,75]   (15,75]
 ...
Levels:  (15,65] (65,115] (115,165] (165,215] (215,265] (265,315]
> Pricef <-cut(Price, breaks=15+60*(0:5))
> Sizef <-cut(Size, breaks=0.8*(0:5))
> table(Pricef, Sizef)
        Sizef
Pricef      (0,0.8] (0.8,1.6] (1.6,2.4] (2.4,3.2] (3.2,4]
  (15,75]         3        22         1         0       0
  (75,135]        0        25        27         0       0
  (135,195]       0         0        11         2       0
  (195,255]       0         0         0         0       0
  (255,315]       0         0         0         0       2
```

9 Statistische Abhängigkeiten

9.1 Die Lowess-Glättung

`lowess(x, y, f=2/3, iter=3)`: Berechnung des Locally WEighted Scatterplot Smoothers von**y** in Abhängigkeit von `x`. Der Glättungsparameter `f` bezieht sich auf den Anteil der Gesamtdaten, die in jedem Fenster berücksicht werden. Die Anzahl von Robustifizierungsschritte ist `iter`.
Siehe dazu auch die neuere, auf `formula` basierende Funktion `loess` im package `modreg` für einen *Local Polynomial Regression Fit*.

```
plot(area, price) lines(lowess(area, price, f=1/10, iter=0), col =
2) lines(lowess(area, price, f=9/10, iter=3), col = 3) legend(0.5,
300, c(paste("f = ", c("1/10", "9/10"))), lty = 1, col = 2:3)
```



9.2 Korrelation

Zur Berechnung von Korrelationsmatrizen nach der Momentenmethode (Pearson Korrelationskoeffizient) verwende man:
`cor(x, y = NULL`: Hierbei ist `x` (wie auch `y`) ein numerischer Vektor, eine Matrix oder ein `data.frame`.

```
cor(aimu[ , 3:5], aimu[ , 4:6])
          groesse   gewicht        fvc
alter   -0.2391290 0.1503169 -0.2914085 groesse  1.0000000
0.4939768  0.6829789 gewicht  0.4939768 1.0000000  0.4066760
```

Die Assoziation zwischen Stichprobenpaaren `x` und `y` schätzt und testet man mit:
`cor.test(x, y, alternative = "two.sided", method = "pearson", exact = T, conf.level = 0.95)`: `method` ist `"pearson"`, `"kendall"`, oder `"spearman"`. Für Kendall's Tau kann der exakte p-Wert mit `exact = T` für $n < 50$ und keine Bindungen berechnet werden. Bei `method = "pearson"` basiert das Konfidenzintervall auf Fisher's Z-Transformation.

```
cor.test(fvc, gewicht)
        Pearson's product-moment correlation
data:  fvc and gewicht t = 3.9062, df = 77, p-value = 0.0001999
alternative hypothesis: true correlation is not equal to 0 95
percent confidence interval:
 0.2039004 0.5759929
sample estimates:
     cor
```

```
0.406676

cor.test(fvc, gewicht, method = "spearman")
        Spearman's rank correlation rho
data:  fvc and gewicht S = 50350, p-value = 0.0004665 alternative
hypothesis: true rho is not equal to 0 sample estimates:
      rho
0.3871594 Warning message: p-values may be incorrect due to ties
in: cor.test.default(fvc, gewicht, method = "spearman")

cor.test(fvc, gewicht, method = "kendall")
        Kendall's rank correlation tau
data:  fvc and gewicht z.tau = 3.503, p-value = 0.00046
alternative hypothesis: true tau is not equal to 0 sample
estimates:
      tau
0.2685761
```

9.3 Kontingenzanalyse

```
alt <- rep(NA, length(alter)) for (i in 1:length(alter))
if(alter[i]<25) alt[i] <- "jung" else alt[i] <- "alt" table(ort,
alt)
   alt
ort alt jung
  A  19   15
  M  35   10

chisq.test(table(ort, alt), correct=F)
        Pearson's Chi-squared test
data:  table(ort, alt) X-squared = 4.2923, df = 1, p-value =
0.03829

chisq.test(table(ort, alt), simulate.p.value=T)
        Pearson's Chi-squared test with simulated p-value (based on 2000 replicates)
data:  table(ort, alt) X-squared = 4.2923, df = NA, p-value =
0.026

mytest <- chisq.test(table(ort, alt)); mytest
        Pearson's Chi-squared test with Yates' continuity correction
data:  table(ort, alt) X-squared = 3.3398, df = 1, p-value =
0.06762

mytest$observed
   alt
ort alt jung
  A  19   15
  M  35   10

mytest$expected
   alt
ort      alt     jung
  A 23.24051 10.75949
  M 30.75949 14.24051
```

```
fisher.test(table(ort, alt))
        Fisher's Exact Test for Count Data
data:  table(ort, alt) p-value = 0.05146 alternative hypothesis:
true odds ratio is not equal to 1 95 percent confidence interval:
 0.1209039 1.0667569
sample estimates: odds ratio
 0.3668412
```

10 Lineare Regression

Definition des Prediktors: Seie dazu `y` die (normalverteilte) abhängige Responsevariable, `x2, x3, x4` erklärende Variablen und `a1, a2` erklärende Faktoren. Jede erklärende Variable x_j hat im Modell genau einen unbekannten Parameter β_j. Hat ein Faktor k Stufen, z.B. die $k = 4$ Meßorte Graz, Klagenfurt, Wien und Linz, so kann man diese Information mittels $k - 1$ Dummyvariablen kodieren. Für die 4 Meßorte wäre dies

$$z_i^{\text{Graz}} = \begin{cases} 1 & y_i \text{ aus Graz,} \\ 0 & \text{sonst} \end{cases} \qquad z_i^{\text{Klag}} = \begin{cases} 1 & y_i \text{ aus Klagenfurt,} \\ 0 & \text{sonst} \end{cases} \qquad z_i^{\text{Linz}} = \begin{cases} 1 & y_i \text{ aus Linz,} \\ 0 & \text{sonst} \end{cases}$$

Der Prediktor hat somit die Form

$$\mathrm{E}(y_i) = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i2} + \beta_4 z_i^{\text{Graz}} + \beta_5 z_i^{\text{Klag}} + \beta_6 z_i^{\text{Linz}}.$$

Stammt eine Messung aus `Wien`, so sind alle drei obigen Dummies Null und es bleibt

$$\mathrm{E}(y_i) = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i2}.$$

Kommt y_i aus `Graz`, dann lautet das Modell

$$\mathrm{E}(y_i) = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i2} + \beta_4 z_i^{\text{Graz}}.$$

Der Unterschied liegt hier also in verschiedenen Intercepts. Dieser ist β_1 für `Wien`, und $\beta_1+\beta_4$ für `Graz`. Bei dummykodierten Faktoren beschreiben daher die Parameter gerade die Abweichungen der Faktorstufen von einer Referenzstufe (hier `Wien`). Als Referenzstufe wurde hier `Wien` verwendet. Defaultmässig wird dafür in `R` die lexikographisch erste verwendet.
Falls nur die Haupteffekte `x2` und `a1` ins Modell eingehen sollen, lautet `formula`

```
y ~ x2 + a1
```

Der Intercept ist defaultmäßig immer im Modell enthalten. Er kann durch `-1` auch entfernt werden.
Bei Interaktionseffekten schreibt man die rechte Seite der `formula` als `x2 + x3*a1`. Dies entspricht dem Modell `1 + x2 + x3 + a1 + x3:a1`. Durch den `*` Operator (`x3*a1`) werden sowohl die Haupteffekte (`x3, a1`) als auch die Interaktion (`x3:a1`) inkludiert.
Entsprechend reduziert der `-` Operator das Modell. So liefert `x2*x3*x4 - x2:x3:x4` das Modell `1 + x2 + x3 + x4 + x2:x3 + x2:x4 + x3:x4`, also ohne der dreifachen Interaktion `x2:x3:x4`.
Gefittet wird das Modell, z.B. anhand der Häuserpreise, mittels

```
attach(houses) price.lm <- lm(price ~ area * bath * new); price.lm

Call: lm(formula = price ~ area * bath * new)

Coefficients:
  (Intercept)          area          bath           new     area:bath
       3.5214       25.1216        0.3573     464.5467       14.1408
    area:new      bath:new  area:bath:new
    -156.1634     -233.1644       84.0603
```

```
summary(price.lm)

Call: lm(formula = price ~ area * bath * new)

Residuals:
    Min      1Q  Median      3Q     Max
-34.535  -8.404   0.989   8.704  26.821

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)      3.5214    15.1076   0.233  0.81625 area
25.1216    11.0178   2.280  0.02511 * bath             0.3573
7.7742   0.046  0.96345 new            464.5467   163.9892   2.833
0.00576 ** area:bath       14.1408     4.6259   3.057  0.00299 **
area:new      -156.1634    61.1701  -2.553  0.01247 * bath:new
-233.1644    79.8071  -2.922  0.00446 ** area:bath:new   84.0603
28.3933   2.961  0.00398 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.54 on 85 degrees of freedom Multiple
R-Squared: 0.9132,     Adjusted R-squared: 0.906 F-statistic:
127.7 on 7 and 85 DF,  p-value:     0


table(bath,new)
    new
bath  0  1
   1 10  0
   2 51 26
   3  4  2

# fitted models i10 <- bath==1 & new==0 i20 <- bath==2 & new==0;
i21 <- bath==2 & new==1 i30 <- bath==3 & new==0;  i31 <- bath==3 &
new==1 a <- c(min(area[i10]), max(area[i10]),
       min(area[i20]), max(area[i20]),   min(area[i21]), max(area[i21]),
       min(area[i30]), max(area[i30]),   min(area[i31]), max(area[i31]))
b <- c(1, 1, 2, 2, 2, 2, 3, 3, 3, 3) n <- c(0, 0, 0, 0, 1, 1, 0,
0, 1, 1) d <- data.frame(area = a, bath = b, new = n)

p <- predict(price.lm, d);  cbind(d, p)
   area bath new         p
1  0.40    1   0  19.58365 2  1.28    1   0  54.13459 3  1.01    2
0  58.17320 4  3.05    2   0 167.11579 5  1.15    2   1  77.61833
6  2.25    2   1 149.51490 7  2.04    3   0 142.38301 8  3.85    3
0 264.63771 9  2.57    3   1 190.00000 10 3.30    3   1 309.40000

anova(price.lm) Analysis of Variance Table

Response: price
             Df Sum Sq Mean Sq  F value    Pr(>F)
area          1 145097  145097 790.9001 < 2.2e-16 *** bath
1   4476    4476  24.3957 3.878e-06 *** new            1   6349
```

```
6349  34.6088 7.747e-08 *** area:bath      1   4850    4850
26.4349 1.720e-06 *** area:new       1   1628    1628   8.8750
0.003768 ** bath:new       1      4       4   0.0199  0.888237
area:bath:new  1   1608    1608   8.7649  0.003979 ** Residuals
85  15594     183
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
plot(area, price, xlim = c(0, 4.5)) lines(d$area[1:2], p[1:2]);
text(d$area[2], p[2], "1 bath, old", pos = 4, cex = .8)
lines(d$area[3:4], p[3:4]); text(d$area[4], p[4], "2 baths, old",
pos = 4, cex = .8) lines(d$area[5:6], p[5:6]); text(d$area[6],
p[6], "2 baths, new", pos = 4, cex = .8) lines(d$area[7:8],
p[7:8]); text(d$area[8], p[8], "3 baths, old", pos = 4, cex = .8)
lines(d$area[9:10], p[9:10]); text(d$area[10], p[10], "3 baths,
new", pos = 4, cex = .8)
```