Data-Frames, Faktoren, Deskriptive Statistik

Jonathan Harrington

Siehe auch:
R/Rpad Reference Card: http://rpad.googlecode.com/svn/Rpad_homepage/Rpad-refcard.pdf
Für Abbildungen: http://addictedtor.free.fr/graphiques/

0. Daten einlesen, Data-Frames erzeugen

```
# Numerische Daten aus einer Matrix einlesen
# pfad: Das Verzeichnis, in dem sich die Text-Dateien befinden
werte = scan(file.path(pfad, "werte.txt"))
lok = read.table(file.path(pfad, "lok.txt"))
# oder
lok = read.delim(file.path(pfad, "lok.txt"), sep=" ")

# Erforderlich da Leerstellen in einigen Elementen vorhanden sind
kj = read.delim(file.path(pfad, "kj.txt"))
```

1. Faktoren und Data-Frames erzeugen

```
class(lok)
head(lok)
names(lok)
# Namen ändern
names(lok)[1] = "Neigungen"

# Faktor erzeugen. Kann mit gl() gemacht werden; sicherer mit rep() und factor()
Dial = c(rep("m", 10), rep("w", 10))
Dial = factor(Dial)
Alt = c(rep("j", 5), rep("a", 5))
Alt = rep(Alt, 2)
Alt = factor(Alt)
# oder in einer Zeile
Alt = factor(rep(c(rep("j", 5), rep("a", 5)), 2))

# Meistens sich Faktoren nicht geordnet (daher ist der Default ordered=F),
# da die Reihenfolge unwichtig ist. Wenn man für einen Trend prüfen will, dann sollte
# die Reihenfolge festgelegt werden mit ordered=T
vec = rep(c("sehr jung", "jung", "mittel", "alt", "sehr alt"), each=10)
vec = factor(vec, ordered=T, levels=c("sehr jung", "jung", "mittel", "alt", "sehr alt"))
is.ordered(Alt)
is.ordered(vec)

# Data-Frame erzeugen
d.df = data.frame(werte, Dial, Alt)
# oder mit anderen Namen
d.df = data.frame(F2 = werte, Dialekt = Dial, Alter = Alt)

# Data-Frame speichern
```

```
write.table(d.df, file.path(pfad, "namen.txt"))

# Data-Frame anschauen
head(lok)
with(lok, slopes)
with(lok, mean(slopes))
with(lok, table(G))
gender = with(lok, table(G))
usw.
```

2. Kontinuierliche Variablen (Ein Parameter)

2.1 Numerische Berechnung

mean(), quantile(), median(), IQR(), (auch sd() aber das wird später im Zusammenhang mit der Normalverteilung diskutiert).

```
with(lok, mean(slopes))
with(lok, tapply(slopes, Kons, mean))
with(lok, tapply(slopes, Kons, range))
# siehe unten
with(lok, tapply(slopes, Kons, quantile))
with(lok, tapply(slopes, paste(Kons, P), mean))
```

2.2 Boxplot-Abbildungen

Quantal p ($0 < p < 1$) befindet sich in Position 1+p*(n-1) in der nach Reihenfolge sortierten Daten.

```
# 11 randomisierte Ganzzahlen zwischen 0 und 100
g = sample(0:100, 11)

# Median
median(g)

# Das gleiche
# Der sechste Werte (da 11 Werte und wegen 1+p*(n-1) ) in der sortierten Reihenfolge
g.s  = sort(g)
g.s[6]

# min, 25%  Quantal, Median, 75% Quantal, Maximum
quantile(g)

# die 37% und 68% Quantale
quantile(g, c(0.37, 0.68))

# IQR() : interquartile-range
quantile(g, .75) - quantile(g, .25)
# das gleiche
IQR(g)
```

Boxplot-Abbildung. Der dicke Strich in der Mitte = der Median. Die oberen und unteren Kanten vom Rechteck sind die .25 und .75 Quantalen.

```
boxplot(slopes ~ Kons, data=lok)
# oder with(lok, boxplot(slopes ~ Kons))

with(lok, tapply(slopes, Kons, median))
abline(h = 0.74, col="red")
with(lok, tapply(slopes, Kons, quantile, c(.25, .75)))
abline(h = c(0.6125, 0.7875), col="green")

# * bedeutet 'gekreuzt mit' oder 'Interaktion mit'.
boxplot(slopes ~ Kons * P, data = lok)
with(lok, interaction.plot(Kons, P, slopes))
```

2.3 Histogramme

```
par(mfrow=c(1,3))
xlim = c(0, 1)
with(lok, tapply(slopes,  Kons, hist, xlim=xlim, col="steelblue"))

temp = with(lok, Kons == "g")
with(lok, hist(slopes[temp], xlim=xlim, col="steelblue"))
with(lok, hist(slopes[temp], xlim=xlim, col="steelblue", freq=F))
```

Der Unterschied zwischen diesen Histogrammen: die Gesamt-Fläche vom Histogramm (rechts) = 1.

Genauer:

Density = Frequency/(n * Balkenbreite) (1)

Die Balkenbreite ist 0.1, *n* ist 30 (sum(temp)) und zB ist die Häufigkeit des zweiten Balkens 6.0 in der Abbildung links

Daher ist die Höhe vom entsprechenden Balken in der Abbildung rechts:

```
6/(30 * 0.1)
2
```

Die **Fläche** im Histogramm (rechts) ist eine Verschlüsselung der **Proportionen**.

Fläche = Balkenbreite * Density (2)

Für den zweiten Balken von links ist die Fläche daher:

```
0.1 * 2
0.2
```

Also ca. 20% der Werte liegen zwischen 0.1 und 0.2. Stimmt das?

```
with(lok, sum(slopes[temp]  > 0.1 & slopes[temp] < 0.2))
6
6/30
0.2
```

Eine Gaußglocke auf die rechte Abbildung überlagern
with(lok, curve(dnorm(x, mean(slopes[temp]), sd(slopes[temp])), add=T, lwd=2))

3. Tabellarische Daten (count data)
Es handelt sich hier um Aufzählungen von **kategorialen Daten**.
Kategorial: die Anzahl von /I, E, a/.
wie oft wird /r/ im Vergleich zu /R/ in Bayern verwendet? (Kategoriale Daten, weil die Wahl nur zwischen /r/ oder /R/ besteht).

Kontinuierlich: Die F2-Werte von /I/ im Vergleich zu /E/.
Die Zungenposition und -konfiguration in der Erzeugung von /r/-Lauten.

Nützliche Funktionen in der deskriptiven Statistik von kategorialen Daten
table(), prop.table(), barplot()

with(kj, table(str, age))

Ein **Barplot** ist eine graphische Darstellungen der Informationen, die man auch mit table() bekommt. (Manche verwenden Barplots für kontinuierliche Daten, **aber dies ist im allgemeinen nicht zu empfehlen** - Boxplots sind dafür viel besser geeignet).

```
o = with(kj, table(str, age))
barplot(o, beside=T)

# als Proportionen der Gesamtanzahl der Stimuli
barplot(prop.table(o), beside=T)

# Die Abbildung schöner gestalten (siehe auch help(barplot) und example(barplot) ).
col = c("blue", "red")
barplot(o, beside=T, col=col)
# NB: Mausklick in der Abbildung
legend(locator(1), c("/S/", "/s/"), fill=col)
```

4. Zwei kontinuierliche Parameter
4.1 Die plot() Funktion

```
x = 1:10
plot(x)
y = x^2
plot(x, y)
```

plot() können mit mehreren Argumenten ergänzt werden. Diese können auch für barplot(), boxplot(), hist() usw. eingesetzt werden. Hier sind die wichtigsten davon (siehe help(par) für alle):

```
# die Farbe. Entweder Ganzzahl oder eine aus color() auswählen
col="blue"

# Um die Farb-Möglichkeiten zu sehen
farben = colors()  .

# Siehe auch
```

```
pie(rep(1,30), col=rainbow(30))
pie(rep(1, 6), col=1:6)

# Vier  Graustufen-Abbildungen zwischen schwarz und weiß:
gray(0:3/4)

# Daher:
# 21 Graustufen
g = gray(0:20/20)
# 21 Bars mit diesen Graustufen
barplot(rep(2, 21), col=g)

# Mit Linie, ("l"),  Linie + Punkte ("b"), Keine Werte ("n").
plot(x, type="l")

# Solid, dotted, dashed...(default lty=1)
plot(x, type="l", lty=1)
# das gleiche
plot(x, type="l", lty="solid")

# Liniendichte
plot(x, type="l", lty="solid", lwd=2)

# Symbole für die Abbildung
plot(x, pch=2)
plot(x, type="b", pch=1)

# Weitere Symbole
Die Funktion pchShow() in help(points) laden

dann
pchShow()

# Beschriftung
plot(x, type="b", pch=1, main="Überschrift", xlab = "etwas", ylab="etwas")
# Keine Achsen-Beschriftung
plot(x, type="b", pch=1, main="Überschrift", xlab = "", ylab="")

# Bereich setzen
x = y = 0:10

ylim = xlim = c(-20, 20)
plot(x, y, type="b", pch=1, main="Überschrift", xlim = xlim, ylim=ylim)

# Ohne Achsen
plot(x, y, type="l", axes=F)

# x-Achse hinzufügen
axis(side=1)

# x-Achse oben  (axis(side = 2) und axis(side=4) für die y-Achsen)
```

```
axis(side=3)

# /a:/ (gefüllte Kreise) und /a/ (offene Kreise) in der langsamen Geschwindigkeit
# farb-kodiert nach Versuchsperson

vdata = read.table(file.path(pfad, "vdata.txt"))
temp = with(vdata, Rate=="a" & V == "A")
cols = with(vdata, as.numeric(Subj))
pch = c(1, 16)[with(vdata, as.numeric(Tense))]
with(vdata, plot(X[temp], Y[temp], col=cols[temp], pch=pch[temp]))
```

5. Überlagerungsfunktionen

```
x = y = 0:10
# Die Linie y = -x + 5
plot(x, y, type="l")
abline(5, -1, col="red")

# Linie von Koordinaten überlagen. Hier von [2, 8] nach [4, 1]
lines(c(2, 4), c(8, 1), type="l", col="green", lty="dotted")

# Ein oder mehrere Werte überlagern
points(5, 9,  pch=5)

# Text überlagern; cex ist character expansion
text(6, 3, "Hallo", cex=2)

# locator(1) gibt die Koordinaten vom Mausklick zurück
# Daher text per Mausklick einfügen
text(locator(1), "mein text")

# Polynom überlagern. Hier y = 0.1x^2 + 3 zwischen x = 4 und x = 9
curve(0.1 * x^2 + 3, xlim=c(4, 9), col="blue", add=T)

# curve() kann auch alleine ohne add=T eingesetzt werden
curve(cos(x) + sin(x), xlim = c(-20, 20))

# y = cos (x) e^-0.1x
curve(cos(x) * exp(-.1*x), xlim=c(0, 50), main = "A decaying sinusoid", ylab="Amplitude")
```

6. Abbildungen überlagern, oder nebeneinander...

```
# Eine Abbildung auf eine andere überlagern: par(new=T)

plot(0:10, 0:10, type="l")
par(new=T)
plot(0:20, 20:0, type="l", col=2)
par(new=F)

# besser: Bereiche setzen; und die x- und y-Achsen Beschriftungen in
# der ersten Abb. weglassen
xlim = c(0, 20)
```

```
ylim = c(0, 20)
plot(0:10, 0:10, type="l", xlim=xlim, ylim=ylim, xlab="", ylab="")
par(new=T)
plot(0:20, 20:0, type="l", xlim=xlim, ylim=ylim, col=2, xlab="x", ylab="y")
par(new=F)

# Mehrere Abbildungen nebeneinander
# mfrow(m, n) m = Reihen, n = Spalten. Hier 4 Abbildungen in 2 Reihen und 2 Spalten
# (Es gibt auch mfcol(m, n) )
par(mfrow=c(2,2))
# oben links
plot(1:10)
# oben rechts
curve(cos(x) + sin(x), from = -20, to=20)
# unten links
curve(cos(x) * exp(-.1*x), from = 0, to= 50, main = "A decaying sinusoid",
ylab="Amplitude")

# Zurücksetzen auf 1 x 1
par(mfrow=c(1,1))
plot(1:10)
```