Statistische Datenanalyse mit R Teil 5 online, Funktionen & Programme

Dr. Andrea Denecke
Leibniz Universität IT-Services

Funktionen

Eigene Funktionen in R haben den Aufbau `namefunction <- function (arguments){body of function}`

Beispiel: insbesondere Populationswachstumsdaten sind meist nicht linear. Um diese z.B. in einer Korrelationsanalyse verwenden zu können müssen diese linearisiert werden. Angenommen, die Wachstumsdaten folgen der power function $f(x)=ax^b$ mit a und b als konstante reelle Zahlen und x als Variable, können die Daten durch die Funktion log(f(x)=log(a)+b *log(x) linearisiert werden.

Der Testdatensatz „CanPop“ enthält Daten über das Populationswachstum der kanadischen Bevölkerung von 1851 bis 2001 in 10 Jahres Abständen. Die Populationsdaten sollen mit der power function linearisiert werden, die Parameter sind a=0.5 und b=2

Aufgabe: Öffnen Sie den Testdatensatz „CanPop“. Erzeugen Sie ein Liniendiagramm für die Populationsdaten. Linearisieren Sie die Daten mit der o.g. Funktion (neue Variable „pop.lin“)

Funktionen

Erster Schritt: Erzeugen der Funktion log(f(x))=log(a)+b *log(x) in der Konsole

```
lin.power <-  function(x,a,b){
f.x <- log(a) + b* log(x)
return(f.x)}      #Ausgabe des Wertes
```

Zweiter Schritt: Anwendung der Funktion auf die Daten, x ist die Variable „population“, a =0.5, b=2

```
CanPop$pop.lin <- lin.power(CanPop$population,
  0.5,2)
```

Dritter Schritt: Visuell untersuchen, ob die Linearisierung geeignet ist anhand eines Streudiagramms

3 Programmieren in R

R ist objektorientiert → alles im Workspace ist ein Objekt.

Datentypen:		
	logical	TRUE oder FALSE
	numerical	reelle Zahlen, z.B. 2, 1.75, -186
	complex	komplexe Zahlen, z.B. 3.5 -4i
	character	Zeichenfolgen, z.B. „männlich“

Datenstrukturen:

- **Vektor**, generelle Datenstruktur in R, ein Vektor „d“ kann erzeugt werden über `d <- c(1.03, 1.21, 1.58, 1.09, 0.98, 1.67)` dieser Vektor ist numerisch, ein Vektor mit character mode ist auch möglich.
- **Matrix**, kann über `matrix()` ,erzeugt werden, entweder nrow oder ncol muss angegeben werden, z.B. `matrix(d, nrow=3)`
- **Liste**, kann Daten von unterschiedlicher Struktur und Länge enthalten. Eine Liste kann über `list(vector1, vector2, matrix1)` erzeugt werden

4 Programmieren in R

Man kann in R Einschränkungen verwenden

```
if (Bedingung.1) {Ausdruck.1}
else if(Bedingung.2) {Ausdruck.2}
…
else {Ausdruck.k}
```

Ein (sinnfreies) Beispiel:

```
> k <- 1
> w <- 5
> if (k>2) {w-1} else {w+1}
[1] 6
```

Vorsicht! Die Bedingung darf immer nur die Länge 1 haben, also ein Element enthalten!

5 Programmieren in R

Das allgemeine Kommando für „for“-Schleifen ist

```
for (i in b) {Ausdruck}
```

„b“ ist ein Objekt in R (z.B. ein Vektor), für jedes „i“ wird der Ausdruck berechnet

Beispiel:

```
>b <- c(3,6)
>for (i in b) {print(sqrt(i*2))}
[1] 2.4494
[1] 3.464102
```

Meist können Schleifen vermieden werden, in diesem Beispiel erzeugt das Kommando

```
sqrt(b*2)
[1] 2.4494 3.464102
```

das gleiche Ergebnis

Programmieren in R

Bei der Programmierung von „while“-Schleifen besteht die Gefahr, Endlosschleifen zu programmieren. Die allgemeine Anweisung ist

```
while (Bedingung) {Ausdruck}
```

Ein Beispiel:

```
> k <- 4
> while (k>1)                  # solange k größer 1 ist
+ {print(k <- k-1)}            # Ausgabe der Werte und k
[1] 3                           wird jedesmal um 1 kleiner
[1] 2
[1] 1                          # die Schleife stoppt bei k=1
> k
[1] 1
```

Programmieren in R

Beispiel:

Für die Linearisierung der Daten CanPop wurden die Parameter a und b auf 0.5 bzw. 2 gesetzt. Man möchte nun wissen, wie sich eine ***Veränderung dieser Parameter*** auf den Kurvenverlauf auswirkt. Hierzu soll ein Programm geschrieben werden, dass den Parameter **a in 0.1er** Schritten von 0.1 bis 0.6 variiert und jeweils eine Grafik ausgibt, Parameter b soll konstant bei 2 bleiben.

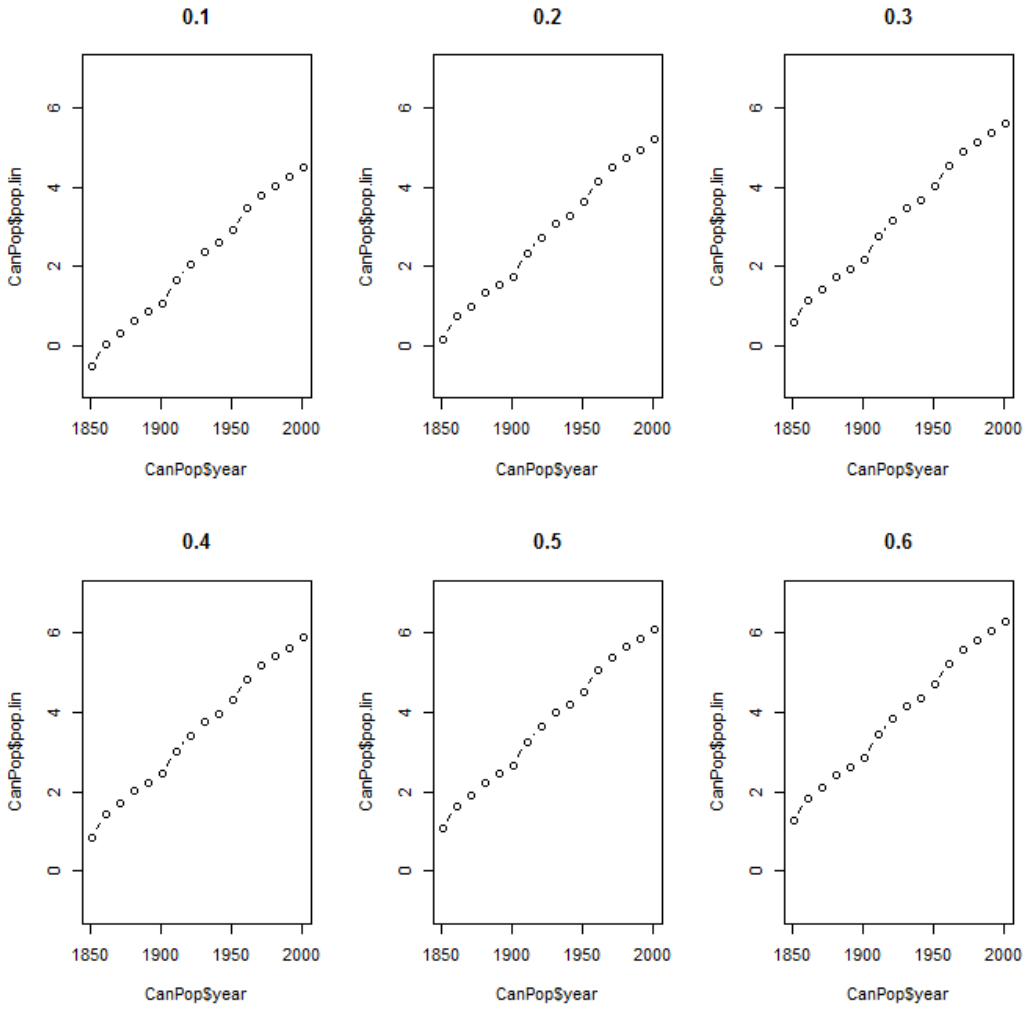
Im zweiten Schritt soll dann der Parameter **b von 1 bis 3 in 0.5er** Schritten variiert werden, Parameter a bleibt konstant bei 0.5.

Programmieren in R

Veränderung des Parameters „a“ mit einer „while“-loop

```
a <- 0.1                        # Startpunkt von a
b <- 2                          # konstanter Wert für b
par(mfrow=c(2,3))               # Unterteilung Grafik in sechs Teile
while (a <= 0.6) {              # Beginn der Schleife bis a 0.6 ist
CanPop$pop.lin <- lin.power (CanPop$population,
a, b)                           # Anwendung der Linearisierung
plot(CanPop$year,CanPop$pop.lin, type="b",
ylim=c(0,7))                    # Grafik year gegen pop.lin, „type=b“
                                (b=both) mit Punkten und Linien
title(a)                        # Der jeweilige Wert von a als Titel
print(CanPop$pop.lin)           # Ausgabe der Werte in Output Fenster
a <- a+0.1}                     # Erhöhung von a um 0.1
```

Programmieren in R



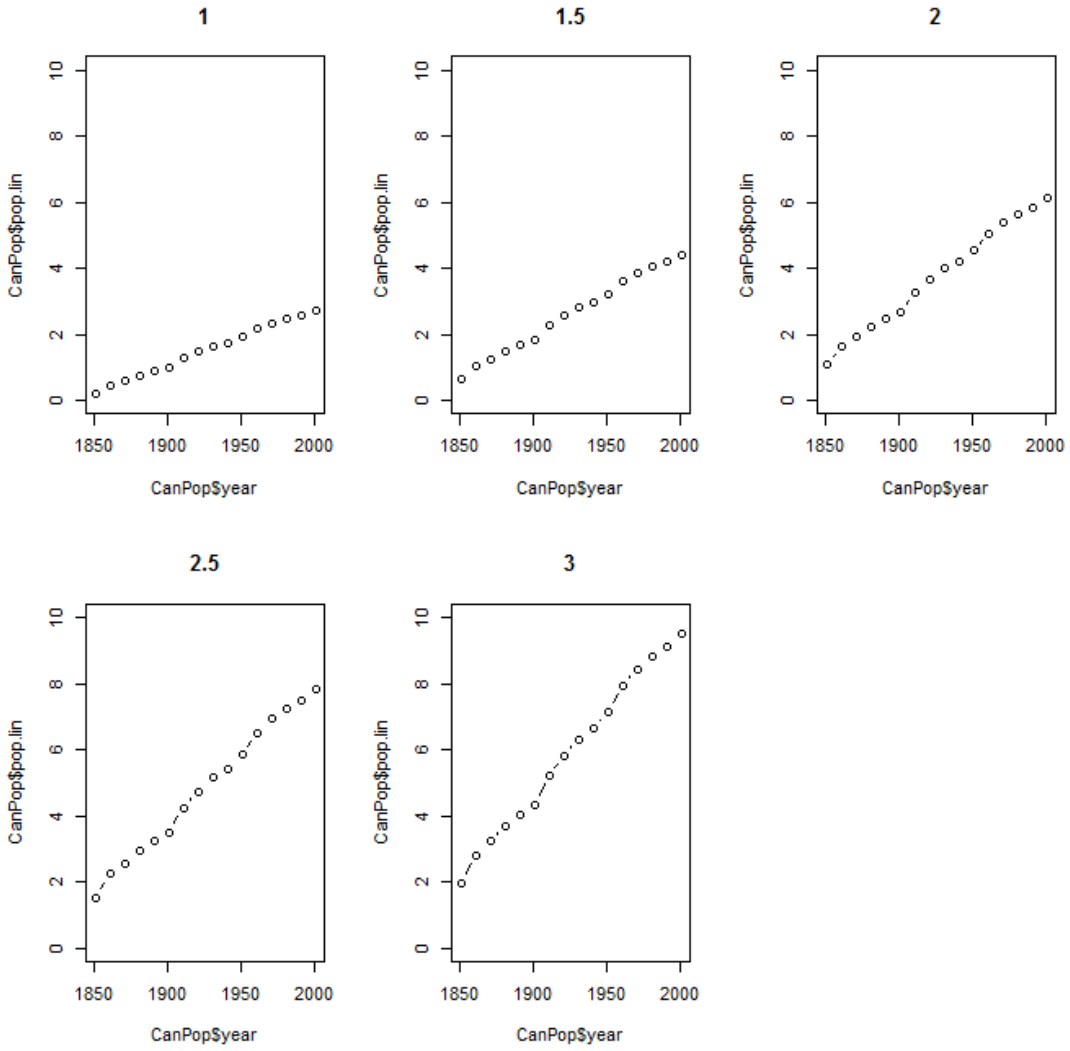
Die Erhöhung von „a“ führt zu einer parallelen Verschiebung der Kurve.

Programmieren in R

Veränderung des Parameters „b“ mit einer „for“-loop

```
a <- 0.5
b <- c(1,1.5,2,2.5,3) #Angabe der„b“-Werte
par(mfrow=c(2,3))     # Unterteilung Grafik in sechs Teile
for (i in b){         # Beginn der Schleife, jedes Element
                      des Vektors b soll Schleife durchlaufen
CanPop$pop.lin <- lin.power (CanPop$population,
a, i)                 # Anwendung der Linearisierung
plot(CanPop$year,CanPop$pop.lin , type="b",
ylim=c(0,10), main=i) #Grafik wie gehabt,“main“ ersetzt
                                „title()“
print(CanPop$pop.lin) # Ausgabe der Werte in Output Fenster
}
```

Programmieren in R



Die Veränderung des Parameters „b“ führt zu Veränderungen in der Steigung der Kurve.

Verteilungsfunktionen in R

In R sind diverse stetige sowie diskrete Verteilungsfunktionen bereits hinterlegt. Diese dienen z.B. der Erstellung von Modellen oder Berechnungen von Wahrscheinlichkeiten. Zugegriffen werden kann auf die jeweiligen Quantile und Wahrscheinlichkeiten unter Angabe der Verteilungsparameter, erstellt werden können auch Zufallsstichproben und Grafiken.

Beispiel: wir erstellen uns eine normalverteilte Stichprobe des Umfangs 15, per Voreinstellung ist MW=0, sd=1

`rnorm(15);` mit `rnorm(15,5,2)` erhalte ich eine Zufalls-stichprobe mit dem Mittelwert 5 und der Standardabweichung 2.
Beachten: jeder erneute Aufruf des Befehls ergibt eine andere (eben zufällige) Stichprobe.

Verteilungsfunktionen in R

Die Kommandos haben die folgende Struktur

erster Buchstabe:
- `d` → Dichtefunktion (density function)
- `p` → Wahrscheinlichkeit (probability)
- `q` → Quantil (quantile)
- `r` → Zufallsstichprobe (random)

Funktionsname:
- `norm` → Normalverteilung (mean, sd)
- `pois` → Poisson Verteilung (lambda)
- `binom` → Binomial Verteilung (size, prob)
- `weibull` → Weibull Verteilung (shape, scale)
- `chisq` → Chi-Square Verteilung (df, x-sq)
- `logis` → Logistische Verteilung (location, scale)
- `t` → t-Verteilung (df, t-value)
- …

Verteilungsfunktionen in R

Bei welchem Wert liegt das 5% und das 95% Quantil der Standardnormalverteilung (MW=0, sd=1) ?

`qnorm(0.05, 0,1); qnorm(0.95)` (-)1.645

Wie groß ist die Wahrscheinlichkeit, dass bei einer Standardnormalverteilung ein Wert von 2 oder größer auftritt?

Mit `pnorm(2)` wird uns die Wahrscheinlichkeit ausgegeben, dass der Wert kleiner als 2 ist, da wir den Kehrwert brauchen:
`pnorm(2, lower.tail=F)` 0.023 (oder `1-pnorm(2)`)

Grafisch darstellen: `plot(pnorm)`, da hier aber die Randbereiche nicht ausreichend dargestellt werden:

`plot(pnorm, xlim=c(-2.5,2.5)).`

Verteilungsfunktionen in R

Weiteres Beispiel: die Binomialverteilung beschreibt die Verteilung des Erfolgs/ Misserfolgs oder des Eintretens/ Nicht-Eintretens von Ereignissen.
Bei einem normalen Würfel ist die Wahrscheinlichkeit, eine bestimmte Zahl zu würfeln 1:6, also 0.167. Wie hoch ist die Wahrscheinlichkeit, zwei Mal hintereinander eine Sechs zu würfeln? (Beachten: um die Einzelwahrscheinlichkeit eines Ereignisses zu bekommen, muss man „d“ verwenden)
`dbinom(2, 2, 0.167)` also 0.028
Wie wahrscheinlich ist es, bei zwei Würfen keine Sechs zu werfen? `dbinom(0, 2, 0.167)` also 0.69
Dementsprechend ist `dbinom(1,2,0.167)` = 0.28 die Wahrscheinlichkeit, eine Sechs zu würfeln (im 1. oder 2. Versuch)

Übung 7

Dient nur zu Übungszwecken, kein wissenschaftlicher Hintergrund!

1. Erzeugen Sie eine Funktion zur Berechnung der Photosyntheseleistung pho der Pflanzen: pho=u*a*b
2. Wenden Sie diese Funktion an (Datensatz CO2)

u= Variable uptake
a= Faktor zu Type, bei Quebec=1, bei Mississippi=1.5
b= Faktor zu Treatment, bei chilled = 0.8, bei nonchilled=0.5

Tipp: die Faktoren zu durchlaufen geht mit ifelse()

Übung 7

3. Wie hoch ist nach der Binomialverteilung die Wahrscheinlichkeit, bei insgesamt fünf Kindern

- genau zwei Jungen zu bekommen?
- höchstens zwei Jungen zu bekommen?
- Stellen Sie dies grafisch dar