Statistik 5

Übung 1

Einleitung und Ziele der Übung

Herzlich Willkommen zur Übung Statistik 5. Diese erste Übung dient der Einführung und Wiederholung von Funktionen im Bereich Zufall in R. In der zweiten Hälfte soll es um Schleifen gehen.

Agenda

- Kurze Vorstellung Ihres Übungsleiter*in
- set.seed() und sample()
- rnorm() und verwandte Funktionen
- replicate()
- for-Schleife und while-Schleife

1. Set.seed und sample - Einführung 1

Zufallszahlen auf dem Computer sind eigentlich Pseudozufallszahlen. Sie sehen zufällig aus, werden aber durch deterministische Algorithmen berechnet. Diese Algorithmen brauchen einen (internen) Startwert. Gibt man diesen nicht vor, kann man nicht sicher sein, womit der Algorithmus startet und Analysen und Methoden, die auf Zufallswerten basieren, sind nicht reproduzierbar.

Mit set.seed(SEED) legen wir einen Startwert fest. Das kann eine beliebige Ganzzahl sein.

Will man, dass diese Ganzzahl auch “zufällig ist”, nimmt man häufig das Systemdatum (vergangene Sekunden seit dem 1. Januar 1970). Das sollte man sich aber dann für die (gleiche) Analyse speichern und mit diesem Wert weiterarbeiten.

```
#Beispiele für set.seed()
set.seed(544) # Der Zufallszahlengenerator startet mit dem internen Wert 544

#Mit Systemzeit, aber die ist in jedem Durchgang anders
set.seed(as.integer(Sys.time())) # der Zufallszahlengenerator startet mit der Systemzeit

#besser
Startwert <- as.integer(Sys.time())
Startwert #(z.B. 1788947601) Diesen Wert rausschreiben und die beiden Zeilen auskommentier
set.seed(1788947601)
```

1. Set.seed und sample - Übung 1

Führen Sie folgende Aufgaben durch, um sich mit dem set.seed()-Befehl vertraut zu machen

- Setzen Sie den Startwert des Zufallgenerators auf 100
- Lassen Sie sich die aktuelle Systemzeit (als Datum) ausgeben
- wandeln Sie die Systemzeit in eine Ganzzahl um und lassen sich diese ausgeben
- Setzen Sie den Startwert des Zufallgenerators basierend auf der Systemzeit

1. Set.seed und sample - Einführung 2

Die erste Zufallsfunktion ist sample(). Mit sample() können wir aus vorgebenen Werten eine Stichprobe ziehen.

Dies kann sowohl mit als auch ohne Zurücklegen passieren.

Dabei können wir annehmen, dass alle Werte die gleiche Wahrscheinlichkeit haben, oder selber Wahrscheinlichkeiten festlegen.

Probieren Sie die nachfolgenden Beispiele aus, dann können Sie mit der nächsten Übung starten.

```
#Beispiele für sample() - Werfen von Münzen
MoeglicheErgebnisse <- c("Kopf","Zahl") #"" weil es sich um Wörter handelt
sample(MoeglicheErgebnisse,3, replace = TRUE) #Werfen von drei idealen Münzen mit Zurückle
sample(MoeglicheErgebnisse,3, replace = FALSE) #scheitert, da Möglichkeiten < Würfe

Wahrscheinlichkeit <- c(0.2,0.8) # Zahl ist 4x so wahrscheinlich wie Kopf
sample(MoeglicheErgebnisse,3, replace = TRUE, Wahrscheinlichkeit) #gezinkte Münze
```

1. Set.seed und sample - Übung 2

Führen Sie folgende Aufgaben mittels sample() durch

- Werfen Sie einen idealen Würfel 10 mal
- Passen Sie den Würfel so an, dass
  - die Wahrscheinlichkeit für die Zahl 1 nur 1/12 beträgt
  - die Wahrscheinlichkeit für die Zahlen 2-5 nur 1/6 beträgt
  - die Wahrscheinlichkeit für die Zahl 6 3/12 beträgt
- Was für einen Unterschied macht es, wenn der gezinkte Würfel entweder
  - 6x innerhalb eines sample()-Befehls ohne Zurücklegen oder
  - mittels 6 sample()-Befehlen jeweils 1x geworfen wird?
- Bei vielen Spielen werden am Anfang eines Zugs 2 Würfel gleichzeitig geworfen und die Summe addiert. Können Sie einen Code schreiben, der das für einen Zug simuliert?

Auf der nächsten Seite finden Sie eine Übung zum Zusammenspiel von set.seed() und sample()

1. Set.seed und sample - Übung 3

Führen Sie folgende Aufgaben durch. Vergleichen Sie die Werte der ersten 3 Würfe mit den nächsten 3. Können Sie das Ergebnis erklären?

- Setzen Sie den Startwert 16 für den Zufallsgenerator
  - Werfen Sie einen idealen Würfel 1x
  - Werfen Sie erneut einen idealen Würfel
  - Werfen Sie erneut einen idealen Würfel
- Setzen Sie den Startwert 16 für den Zufallsgenerator
  - Werfen Sie erneut einen idealen Würfel 1x
  - Werfen Sie erneut einen idealen Würfel
- Setzen Sie den Startwert 16 für den Zufallsgenerator
  - Werfen Sie erneuteinen idealen Würfel

- Probieren Sie das ganze nochmals mit dem Startwert 13 statt 16

2. rnorm() und verwandte Funktionen - Einführung 1

In der Statistik brauchen wir nicht nur Zufallszahlen, die zum Urnenmodell passen, sondern häufig Zufallszahlen, die einer bestimmten Verteilung entsprechen. Die Befehlsnahmen dieser Funktionen beginnen immer mir r (für random) gefolgt von einer Abkürzung für die jeweilige Verteilung. Häufig runden wir die Zahlen, wenn wir zum Beispiel Grössenangaben simulieren wollen (und 165.564645342 cm macht wenig Sinn).

```
# Ziehen von 10 Werten aus einer Normalverteilung mit
# dem Mittelwert 100 und der Standardabweichung 15
rnorm(10,100,15) #rnorm werden wir neben sample am häufigsten brauchen

# Ziehen von 10 Werten aus einer Gleichverteilung mit
# dem Minimum 0 und dem Maximum 100
runif(10,0,100)

# Ziehen von 10 Werten aus einer t-Verteilung mit 15 Freiheitsgraden
rt(10,15)
```

Auf der nächsten Seite finden Sie ein weiteres Beispiel.

2. rnorm() und verwandte Funktionen - Einführung 2

```r
#Simulieren von IQ-Test Ergebnissen von 20 Personen aus der Bevölkerung der Schweiz
#IQ-Scores haben in der Regel keine Nachkommastellen
ungerundeteWerte <- rnorm(20,100,15)
gerundeteWerte <- round(ungerundeteWerte,0)

#Histogramm ausgeben
hist(gerundeteWerte)
```

2. rnorm() und verwandte Funktionen - Übung

Führen Sie folgende Aufgaben durch.

- Ziehen Sie 10 Werte aus einer Normalverteilung mit dem Mittelwert 180 und der Standardabweichung 7
- Häufig werden bei Fragebögen T-Werte angegeben. Das sind ganzzahlige, normalverteilte Werte mit dem Mittelwert 50 und der Standardabweichung 10. Simulieren Sie die Gewissenhaftigkeitswerte von 100 Personen
- Ziehen Sie 5 Werte aus einer *F*-Verteilung, wobei df1 = 3 und df2 = 20 sei
- Ziehen Sie 10000 Werte aus einer Standardnormalverteilung und lassen Sie sich Ihre Werte als Histogramm ausgeben
- Setzen Sie den Zufallsgenerator auf den Startwert 64 und simulieren Sie 10 Werte aus einer Normalverteilung mit dem Mittelwert 10 und der Standardabweichung 2. Bestimmen Sie anschliessend den Mittelwert und die Standardabweichung ihrer 10 Werte. Was stellen Sie fest? Können Sie Ihre Beobachtung erklären?

3. replicate() - Einführung 1

Am letzten Beispiel wurde klar, dass unsere Werte eine Stichprobe aus einer Population sind, deren Verteilung (+ zugehörige Parameter) wir festlegen. Häufig werden wir mehrere Stichproben ziehen wollen. Dafür ist der Befehl replicate() sehr nützlich.

```r
# Ziehen von 10 Stichproben mit je 5 Werten aus einer Normalverteilung mit
# dem Mittelwert 15 und der Standardabweichung 2
set.seed (100)
replicate(10, rnorm(5,15,2))
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
[1,] 13.99562 15.63726 15.17977 14.94137 14.12382 14.12310 14.81777 14.55641
[2,] 15.26306 13.83642 15.19255 14.22229 16.52812 13.55956 18.51475 15.36582
[3,] 14.84217 16.42907 14.59673 16.02171 15.52392 15.46189 14.72414 15.83465
[4,] 16.77357 13.34948 16.47968 13.17237 16.54681 12.68454 14.77761 17.13080
[5,] 15.23394 14.28028 15.24676 19.62059 13.37124 15.49415 13.61997 16.94040
         [,9]    [,10]
[1,] 14.79674 17.64446
[2,] 17.80641 14.27312
[3,] 11.44645 17.63813
[4,] 16.24573 15.08756
```

Auf der nächsten Seite finden Sie ein weiteres Beispiel.

3. replicate() - Einführung 2

```
1 #Wichtig: Folgendes funktioniert nicht (alle Stichproben haben die gleichen Werte)!
2 set.seed (100)
3 Stichprobe <- rnorm(5,15,2)
4 replicate(10,Stichprobe)
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
[1,] 13.99562 13.99562 13.99562 13.99562 13.99562 13.99562 13.99562 13.99562
[2,] 15.26306 15.26306 15.26306 15.26306 15.26306 15.26306 15.26306 15.26306
[3,] 14.84217 14.84217 14.84217 14.84217 14.84217 14.84217 14.84217 14.84217
[4,] 16.77357 16.77357 16.77357 16.77357 16.77357 16.77357 16.77357 16.77357
[5,] 15.23394 15.23394 15.23394 15.23394 15.23394 15.23394 15.23394 15.23394
         [,9]    [,10]
[1,] 13.99562 13.99562
[2,] 15.26306 15.26306
[3,] 14.84217 14.84217
[4,] 16.77357 16.77357
```

3. replicate() - Übung

Führen Sie folgende Aufgaben durch.

- Simulieren Sie 10 Stichproben mit je 20 Personen deren Verträglichkeit (T-Werte) erhoben wurden.
- Häufig werden bei Spielen 2 Würfel gleichzeitig geworfen. Simulieren Sie 20 Runden (eine Runde = jeder Spielende war einmal am Zug) eines Spiels, welches in einer Vierergruppe mit 2 Würfeln gespielt wurde.

4. for-Schleife und while-Schleife - Einführung 1

replicate() macht Wiederholungen sehr einfach, ist aber im Einsatz beschränkt. Alternativ können wir Wiederholungen mit Schleifen umsetzen. Diese sind in R etwas langsamer, aber flexibler. Letzteres werden wir später im Semester benötigen. Zunächst die for-Schleife. Sie vollzieht der Reihe nach eine Wiederholung für alle Anweisungen in den {} jedes Element eines Vektors,welches wir angeben.

```
Werte = 1:3 # (Vektor mit den Zahlen von 1 bis 3)
for (i in Werte) { #i enthält in jeder Wiederholung das aktuelle Element des angegebenen V
  print(i) #Zeilen innerhalb der Schleife werden nicht ausgegeben, ausser wir verlangen es
  print("Hallo")
}
```

```
[1] 1
[1] "Hallo"
[1] 2
[1] "Hallo"
[1] 3
[1] "Hallo"
```

Auf der nächsten Seite finden Sie weitere Beispiele.

4. for-Schleife und while-Schleife - Einführung 2

```
Werte = 1:3
Werte2 = c("Apfel", "Birne", "Pflaume")

for (i in Werte) {
  print(Werte2[4-i])
}
```

```
[1] "Pflaume"
[1] "Birne"
[1] "Apfel"
```

```
Werte = 1:6
for (i in Werte) {
  print(i^2)
}
```

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
```

4. for-Schleife und while-Schleife - Übung 1

Führen Sie folgende Aufgaben durch.

- Schreiben Sie eine for-Schleife, die 10x das Wort “Test” ausgibt
- Schreiben Sie eine for-Schleife mit 5 Durchgängen, wobei jeweils eine Münze geworfen und das Ergebnis ausgegeben werden soll
- Schreiben Sie eine for-Schleife, die für jeden Monat des Jahres den Namen und die Anzahl an Tagen ausgibt (Also 12 Durchgänge).

4. for-Schleife und while-Schleife - Einführung 3

Im Gegensatz zu for-Schleife, wird die while-Schleife sooft durchgeführt, bis eine Bedingung nicht mehr erfüllt ist. Die Bedingung wird bereits vor derm ersten Durchgang geprüft. Es kann also sein, dass Sie nie ausgeführt wird. Es kann aber auch sein, dass die Bedingung nie erfüllt wird. Dann würde Sie endlos weiterlaufen.

```
#Schleife die vor dem ersten Durchgang schon beendet ist
Wert = TRUE
while (Wert==FALSE){
  print("Hallo")
}
```

```
#Eine Münze werfen bis wir Kopf bekommen
set.seed(55)
Wert = "egal"
MoeglicheErgebnisse <- c("Kopf","Zahl")
while (Wert!="Kopf"){
   Wert <- sample(MoeglicheErgebnisse, 1, replace = TRUE)
   print(Wert)
}
```

```
[1] "Zahl"
[1] "Zahl"
[1] "Zahl"
[1] "Kopf"
```

Auf der nächsten Seite finden Sie weitere Beispiele.

4. for-Schleife und while-Schleife - Einführung 4

```
#Zahlen von 1 bis 5 ausgeben
Wert = 0
while (Wert<5){
   Wert <- Wert+1
   print(Wert)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

```
#Oder
Wert = 1
while (Wert<6){
   print(Wert)
   Wert <- Wert+1
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

4. for-Schleife und while-Schleife - Übung 2

Führen Sie folgende Aufgaben durch.

- Schreiben Sie eine while Schleife, die 2 Würfel wirft und deren Summe ausgibt. Die Schleife soll enden, wenn die Summe 10 betragen hat.
- Schreiben Sie eine while Schleife, die Quadratzahlen berechnet und aufhört, sobald eine Quadratzahl grösser als 1000 wäre (also 1, 4, 9 usw. ausgibt)