Musterlösungen zu Serie 7

Lösung 7.1

a) Für den Mittelwert gilt

```
import numpy as np


x = np.loadtxt("../../../Themen/Statistik_Messdaten/Uebungen_de/Daten/oldfa

np.mean(x)
print(np.mean(x))

## 209.2683823529412
```

Setzen **`nboot=1000`**:

```
import numpy as np

x = np.loadtxt("../../../Themen/Statistik_Messdaten/Uebungen_de/Daten/oldfa

n = np.size(x)

nboot = 1000

tmpdata = np.random.choice(x, n*nboot, replace=True)

bootstrapsample = np.reshape(tmpdata, (n, nboot))

xbarstar = np.mean(bootstrapsample, axis=0)

d = np.percentile(xbarstar, q=[2.5, 97.5])

print('Vertrauensintervall: ',d)

## Vertrauensintervall:  [200.49954044 217.30891544]
```

Das Vertrauensintervall ist etwa

$$I = [201,\ 217]$$

Da der Bootstrap vom Zufall abhängt, ergeben sich auch leicht verschiedene Resultate bei jedem Durchlauf.

b) Für den Median gilt

```
import numpy as np

x = np.loadtxt("../../../Themen/Statistik_Messdaten/Uebungen_de/Daten/oldfa

np.median(x)

## 240.0
```

Setzen **nboot=1000**:

```
import numpy as np

x = np.loadtxt("../../../Themen/Statistik_Messdaten/Uebungen_de/Daten/oldfa

n = x.size

nboot = 1000

tmpdata = np.random.choice(x, n*nboot, replace=True)

bootstrapsample = np.reshape(tmpdata, (n, nboot))

xbarstar = np.median(bootstrapsample, axis=0)

d = np.percentile(xbarstar, q=[2.5, 97.5])

print('Vertrauensintervall: ',d)

## Vertrauensintervall:  [230.  246.5]
```

Das Vertrauensintervall ist (etwa)

$$I = [230,\ 247]$$

c) Code:

```
import numpy as np
x = np.loadtxt("../../../Themen/Statistik_Messdaten/Uebungen_de/Daten/oldfa

n = x.size

nboot = 1000

tmpdata = np.random.choice(x, n*nboot, replace=True)
```

```
bootstrapsample = np.reshape(tmpdata, (n, nboot))

xbarstar = np.mean(bootstrapsample, axis=0) - np.mean(x)

l = np.sum(xbarstar < -5)

u = np.sum(xbarstar > 5)

ratio = (l+u)/nboot


## 0.226
```

Die Wahrscheinlichkeit ist also etwa 0.23, dass der Durchschnitt der Messungen mehr als 5 Einheiten vom wahren Mittelwert abweicht.

Lösung 7.2

a) Für die uniforme Verteilung $X \sim \text{Uniform}([0, 10])$ gilt $\text{E}(X) = \mu_X = 5$, $\sigma_X = \frac{10-0}{\sqrt{12}} = \frac{5}{\sqrt{3}}$ (siehe Kennzahlen für die uniforme Wahrscheinlichkeitsverteilung). Dann folgt aufgrund des Zentralen Grenzwertsatzes

$$\overline{X}_n \sim \mathcal{N}(\mu_X, \sigma_X^2/n)$$

Wird der arithmetische Mittelwert $\overline{X}_n$ der Stichprobe vom Umfang n standardisiert, so gilt für die standardisierte Zufallsvariable

$$Z_n = \frac{\overline{X}_n - \mu_X}{\sigma_X/\sqrt{n}} = \frac{\overline{X}_n - 5}{5/\sqrt{3n}} \sim \mathcal{N}(0, 1)$$

Es gilt dann für das gesuchte Intervall $[\mu - e, \mu + e]$, dass

$$\begin{aligned} P(\mu_X - e \leq \overline{X}_n \leq \mu_X + e) &= P\left(-\frac{e}{5/\sqrt{3n}} \leq \frac{\overline{X}_n - 5}{5/\sqrt{3n}} \leq \frac{e}{5/\sqrt{3n}}\right) \\ &= P\left(-\frac{e}{5/\sqrt{3n}} \leq Z_n \leq \frac{e}{5/\sqrt{3n}}\right) \\ &= \Phi\left(\frac{e}{5/\sqrt{3n}}\right) - \Phi\left(-\frac{e}{5/\sqrt{3n}}\right) \\ &= 0.95 \end{aligned}$$

Aufgrund der Symmetrie der Standardnormalverteilung genügt es, eine Seite der Verteilung zu betrachten: $\Phi\left(\frac{e}{5/\sqrt{3n}}\right)$ soll also 97.5 % der Fläche unter der

Gesamtkurve entsprechen. Das 97.5 %-Quantil $q_{0.975}$ der Standardnormalverteilung ist

$$\frac{e}{5/\sqrt{3n}} = q_{0.975} = \Phi^{-1}(0.975) = 1.96\,.$$

Also ist

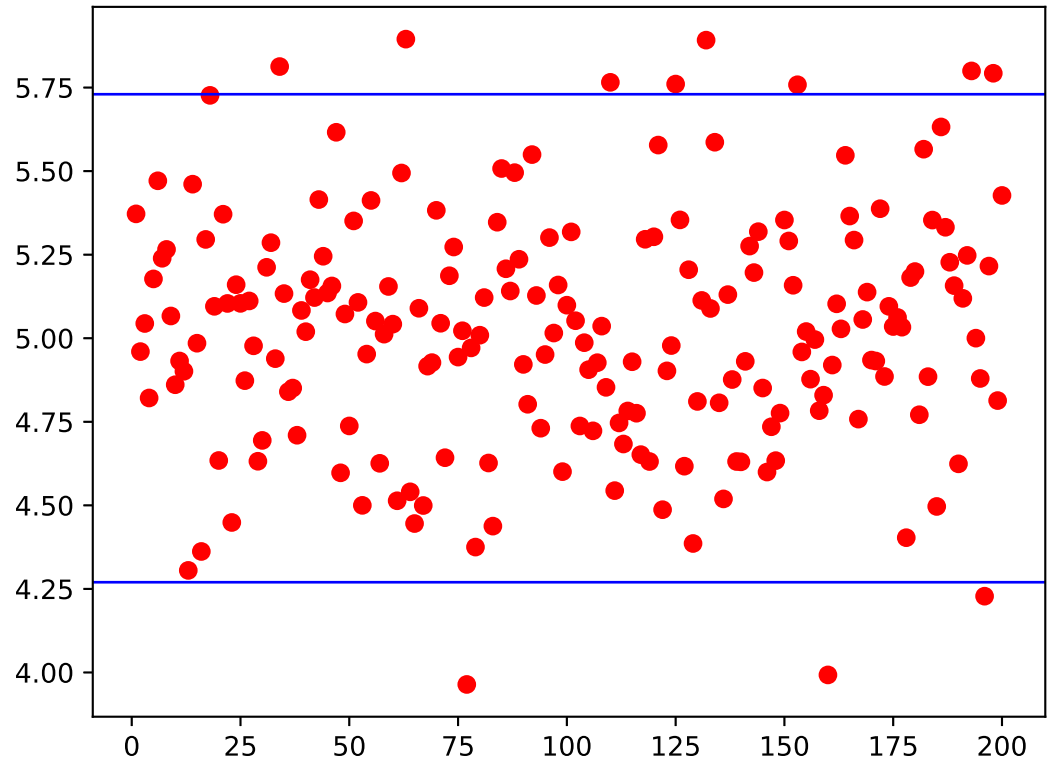
$$e = \frac{5}{\sqrt{3n}} \cdot q_{0.975} = \frac{5}{\sqrt{3 \cdot 60}} \cdot 1.96 = 0.73$$

Somit lautet das Prognoseintervall: $I = [5 - 0.73, 5 + 0.73] = [4.27, 5.73]$.

b) Auflösen der Gleichung $e = 0.2 = \frac{5}{\sqrt{3n}} \cdot 1.96$ nach n liefert $n = 800$.

c) $I = [5 - 0.73, 5 + 0.73]$, $n = 200$. Man erwartet ca. $0.05 \cdot 200 = 10$ Datenpunkte, die ausserhalb des Prognoseintervalls liegen. (zu **R**)

```
from scipy.stats import uniform
import matplotlib.pyplot as plt
import numpy as np
n = 60  # Anzahl Stichproben
# X_1,...,X_n simulieren und in einer
# n-spaltigen Matrix (mit 200 Zeilen) anordnen
sim = uniform.rvs(size=200*n, loc=0, scale=10)
sim = sim.reshape(200, n)
#In jeder Matrixzeile Mittelwert berechnen
sim_mean = sim.mean(axis=1)
plt.plot(np.arange(1,201,1),sim_mean)
axhline(y=5.73, linewidth=4, color='b')
axhline(y=4.27, linewidth=4, color='b')
```

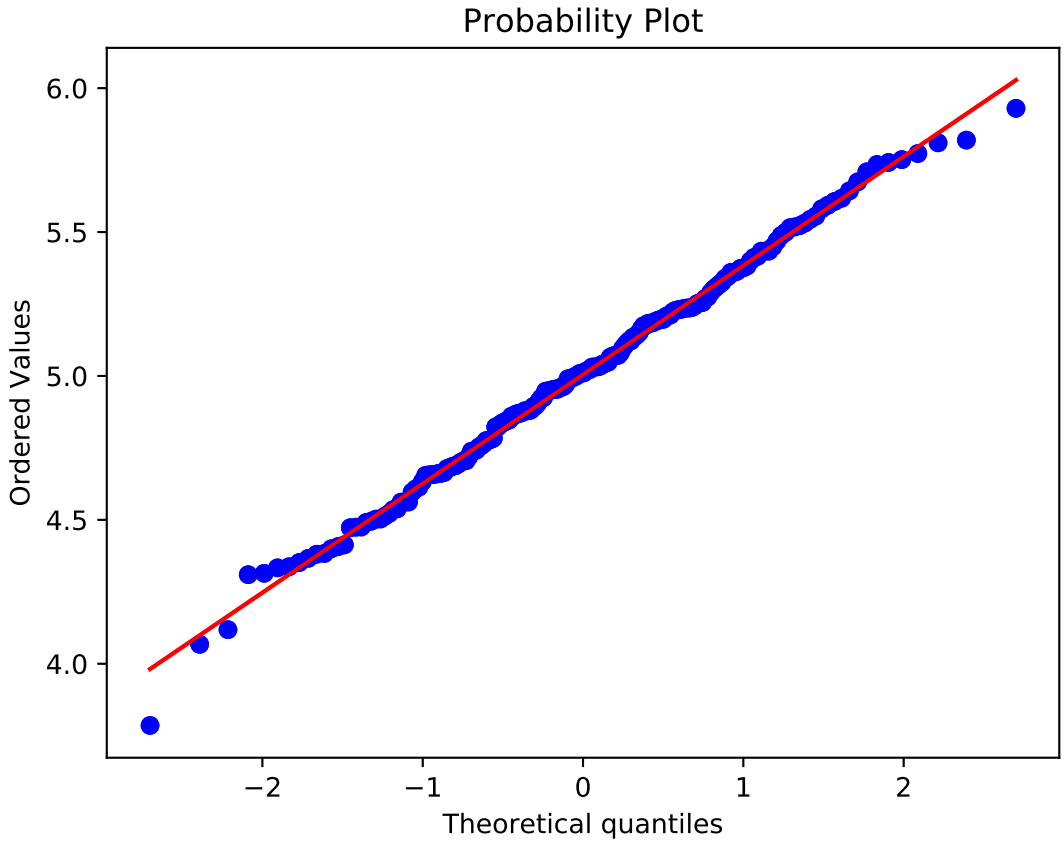



```
from scipy.stats import uniform
import matplotlib.pyplot as plt
import numpy as np
n = 60
sim = uniform.rvs(size=200*n, loc=0, scale=10)
sim = sim.reshape(200, n)
d = np.sum(sim_mean>5.73) + np.sum(sim_mean<4.27)
print(d)

## 4
```

Weiter bestätigt sich auch der Zentrale Grenzwertsatz: (zu **R**)

```
from scipy.stats import probplot
probplot(sim_mean, plot=plt)
```



Lösung 7.3

a) Wir bezeichnen mit X_i den Bleigehalt in der i-ten Bodenprobe. Es gilt

$$X_i \sim \mathcal{N}(\mu, \sigma^2)\,.$$

μ ist der wahre Mittelwert der Verteilung, wobei wir diesen Wert in der Praxis natürlich nicht kennen. $\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ ist auch normalverteilt, allerdings mit Standardabweichung $\sigma/\sqrt{n}$, dem sogenannten *Standardfehler*

$$\overline{X}_n \sim \mathcal{N}(\mu, \sigma^2/n)$$

Nun betrachten wir in dieser Aufgabe $\overline{X}_{10}$, den Mittelwert von 10 Stichproben. $\overline{X}_{10}$ ist also normalverteilt mit Varianz $\sigma^2/n = \frac{36}{10} = 3.6$ und μ unbekannt. Der Fall, dass σ bekannt ist, μ aber nicht, ist natürlich nicht realistisch. Wir werden in einer späteren Teilaufgabe den Fall sehen, wo sowohl σ wie μ unbekannt sind. Da $\Phi(2.58) = 0.995$, liegen 99 % aller Beobachtungen von der standardisierten

Zufallsvariable

$$\frac{\overline{X}_{10} - \mu}{\sigma/\sqrt{n}} = \frac{\overline{X}_{10} - \mu}{6/\sqrt{10}}$$

in dem Intervall $[-2.58, 2.58]$. Es gilt also

$$P\left(\frac{\overline{X}_{10} - \mu}{6/\sqrt{10}} \in [-2.58, 2.58]\right) = 0.99$$

D. h. angenommen wir kennen das wahre μ, und wir bestimmen in 100 Messreihen mit jeweils 10 Stichproben $\overline{x}_{10}$, dann erwarten wir, dass der Wert $\frac{\overline{x}_{10}-\mu}{6/\sqrt{10}}$ in 99 Messreihen im Intervall $\in [-2.58, 2.58]$ liegt. Also liegen 99 % aller Beobachtungen von $\overline{X}_{10} - \mu$ im Intervall $[-2.58 \cdot 6/\sqrt{10}, 2.58 \cdot 6/\sqrt{10}]$. Somit gilt einerseits $\overline{X}_{10} - \mu \leq 2.58 \cdot 6/\sqrt{10}$ und andererseits $-2.58 \cdot 6/\sqrt{10} \leq \overline{X}_{10} - \mu$. Das 99 % Vertrauensintervall für μ ist demnach gegeben durch

$$\left[\overline{X}_{10} - 2.58 \cdot \frac{6}{\sqrt{10}}, \overline{X}_{10} + 2.58 \cdot \frac{6}{\sqrt{10}}\right],$$

d. h. die Intervallgrenzen sind abhängig von der Zufallsvariablen $\overline{X}_{10}$. In unserem Fall ist die Realisierung von $\overline{X}_{10}$: $\overline{x}_{10} = 31$. Man erhält also für ein 99 % Vertrauensintervall (für jede Realisierung von $\overline{X}_{10}$ erhält man ein leicht anderes Vertrauensintervall):

$$[26.1, 35.9]$$

Mit **Python**: (zu **R**)

```
from scipy.stats import norm, t
import numpy as np
norm.interval(alpha=0.99, loc=31, scale=6/np.sqrt(10))

## (26.112707522188142, 35.887292477811854)
```

b) Aus Teilaufgabe a) sieht man, dass die Breite des Vertrauensintervalls wie $1/\sqrt{n}$ abfällt mit der Anzahl n von Beobachtungen. Also sind viermal so viele Beobachtungen, $4 \cdot 10$ nötig, um die Breite des Vertrauensintervalls zu halbieren. Wie aus Teilaufgabe a) ersichtlich ist die Breite des 99 % Vertrauensintervalls

$$2 \cdot 2.58 \cdot \frac{\sigma}{\sqrt{n}}.$$

Um die Breite des Vertrauensintervalls kleiner als 1 ppb zu erhalten, muss die Anzahl n der Beobachtungen entsprechend gross werden:

$$2 \cdot 2.58 \cdot \frac{\sigma}{\sqrt{n}} \leq 1$$

$$30.96 \leq \sqrt{n}$$

$$n \geq 959$$

Es müssen mindestens 959 Beobachtungen vorliegen, um ein 99 % Vertrauensintervall von weniger als 1 ppb Breite zu erhalten.

c) Da σ_x unbekannt, verwenden wir eine t-Verteilung mit 9 Freiheitsgraden.

Mit **Python**: (zu **R**)

```
from scipy.stats import norm, t
import numpy as np
t.interval(alpha=0.99, df=9, loc=31, scale=6/np.sqrt(10))

## (24.833870595963425, 37.166129404036575)
```

Für $\widehat{\sigma} = 6$, $n = 10$ und $\overline{x}_{10} = 31$ ergibt sich das Vertrauensintervall

$$[24.8, 37.2]$$

Durch Vergleich mit a) findet man, dass das Vertrauensintervall einen Faktor 3.25/2.58, also um 26 % grösser geworden ist.

Lösung durch Standardisierung:

Die standardisierte Zufallsvariable

$$\frac{\overline{X}_n - \mu}{\widehat{\sigma}/\sqrt{n}}$$

mit Schätzwert $\widehat{\sigma}$ für σ ist nicht mehr normal-verteilt (wie für ein bekanntes, festes σ), sondern folgt einer t-Verteilung mit 9 Freiheitsgraden.

Das 99.5 % Quantil dieser Verteilung ist bei (zu **R**)

```
from scipy.stats import t

t.ppf(q=0.995, df=9)

## 3.2498355440153697
```

Somit fallen 99 % der Beobachtungen von

$$\frac{\overline{X}_n - \mu}{\widehat{\sigma}/\sqrt{n}}$$

in das Intervall $[-3.25, 3.25]$. Es gilt also

$$P\left(\frac{\overline{X}_n - \mu}{\widehat{\sigma}/\sqrt{n}} \in [-3.25, 3.25]\right) = 0.99$$

Ein Vertrauensintervall für μ ist daher gegeben durch

$$\left[\overline{x}_{10} - 3.25 \cdot \frac{\widehat{\sigma}}{\sqrt{n}}, \overline{x}_{10} + 3.25 \cdot \frac{\widehat{\sigma}}{\sqrt{n}}\right]$$

Für $\widehat{\sigma} = 6$, $n = 10$ und $\overline{x}_{10} = 31$ ergibt sich das Vertrauensintervall

$$[24.8, 37.2]$$

Durch Vergleich mit a) findet man, dass das Vertrauensintervall einen Faktor 3.25/2.58, also um 26 % grösser geworden ist.

Lösung 7.4

a) $\left[-403 \pm t_{9-1;97.5\,\%} \cdot \frac{3.127}{\sqrt{9}}\right] = [-403 \pm 2.31 \cdot 1.042] = [-405.4, -400.6]$

Mit **Python**: (zu **R**)

```
from scipy.stats import norm, t
import numpy as np

t.interval(alpha=0.95, df=8, loc=-403, scale=3.127/np.sqrt(9))

## (-405.40362497674977, -400.59637502325023)
```

b) Da -400.0 nicht im 95 %-Vertrauensintervall liegt, würde die Nullhypothese $H_0 : \mu = -400.0$ zu Gunsten der Alternative $H_A : \mu \neq -400.0$ auf dem 5 %-Signifikanzniveau verworfen werden.

Die Beobachtungen und die Hypothese $H_0 : \mu = -400.0$ passen also nicht gut zusammen und daher ist die wahre Differenz wohl nicht -400.0.

Lösung 7.5

a) Wir wählen

$$H_0: \quad \mu = 500$$

als Nullhypothese und

$$H_A: \quad \mu > 500$$

als Alternativhypothese.

Man möchte auf jeden Fall den Fehler „Es handelt sich um 500er Schrauben, doch wir denken, es seien die neuen verbesserten Schrauben." vermeiden, da

dies im schlimmsten Fall zum Brückeneinsturz führen kann. Daher wird dies der Fehler 1. Art.

Die Geschichte schliesst $\mu < 500$ als Teil der Alternative aus. Die Alternative $H_A : \mu > 500$ führt zu einer grösseren Macht als $H_A : \mu \neq 500$ (eines der beiden „Argumente" reicht).

b) Mit **Python** (zu **R**)

```
import numpy as np
from scipy.stats import norm, t
from pandas import Series

x = Series([520, 512, 499, 524, 505])

x.mean()
x.var()

t.ppf(q=0.95, df=x.size-1, loc=500, scale=x.std()/np.sqrt(x.size))

## 512.0
## 106.5
## 509.8388828578604
```

Der Verwerfungsbereich ist also

$$I = [509.8, \infty)$$

Der Wert 512 für den Mittelwert liegt im Verwerfungsbereich, somit wird die Nullhypothese verworfen. Die Schrauben sind also statistisch signifikant stärker.

Berechnung von Hand:

Die Teststatistik berechnet sich als

$$T = \sqrt{5}\frac{\overline{X}_n - 500}{\hat{\sigma}}$$

wobei $\overline{x}_5 = \frac{1}{5}\sum_{i=1}^{5} x_i = 512$ der empirische Mittelwert und

$$\hat{\sigma}^2 = \frac{1}{4}\sum_{i=1}^{5}(x_i - \bar{x}_5)^2 = 106.5$$

die empirische Varianz ist.

Aus unserem Datensatz ergibt sich $t = \frac{512-500}{\sqrt{106.5/5}} \approx 2.6$ T ist t-verteilt mit vier Freiheitsgraden. Der Verwerfungsbereich auf dem 5 % Niveau ist gegeben durch

(zu **R**)

$$K = [2.132, \infty)$$

```
from scipy.stats import t

t.ppf(q=0.95, df=4)

## 2.13184678133629
```

Die Nullhypothese wird demnach abgelehnt.

c) Das zweiseitige Vertrauensintervall ist gegeben durch (zu **R**)

$$\text{VI} := \left[\overline{x}_5 - \frac{\hat{\sigma} t_{n-1;1-\alpha/2}}{\sqrt{5}}; \overline{x}_5 + \frac{\hat{\sigma} t_{n-1;1-\alpha/2}}{\sqrt{5}}\right]$$

```
import numpy as np
from scipy.stats import norm, t
from pandas import Series

x = Series([520, 512, 499, 524, 505])

t.ppf(q=[0.025,0.975], df=x.size-1, loc=x.mean(),
scale=x.std()/np.sqrt(x.size))

## [499.18617192 524.81382808]
```

Für obige Daten ergibt sich $VI \approx [499.1882; 524.8118]$.

d) Setzt man die Streuung σ als bekannt voraus, so ist das zweiseitige Vertrauensintervall gegeben durch

$$\text{VI} := \left[\hat{\mu} - \frac{\sigma z_{1-\alpha/2}}{\sqrt{5}}; \hat{\mu} + \frac{\sigma z_{1-\alpha/2}}{\sqrt{5}}\right]$$

wobei $z_{1-\alpha/2}$ das $1-\alpha$-Quantil einer Standard-Normalverteilung ist. (zu **R**)

```
import numpy as np
from scipy.stats import norm, t
from pandas import Series

x = Series([520, 512, 499, 524, 505])

norm.ppf(q=[0.025,0.975], loc=x.mean(), scale=x.std()/np.sqrt(x.size))
```

```
## [502.9543893 521.0456107]
```

Für obige Daten ergibt sich dann VI $\approx [502.9544; 521.0456]$.

e) R,R,R,R,R

R-Code

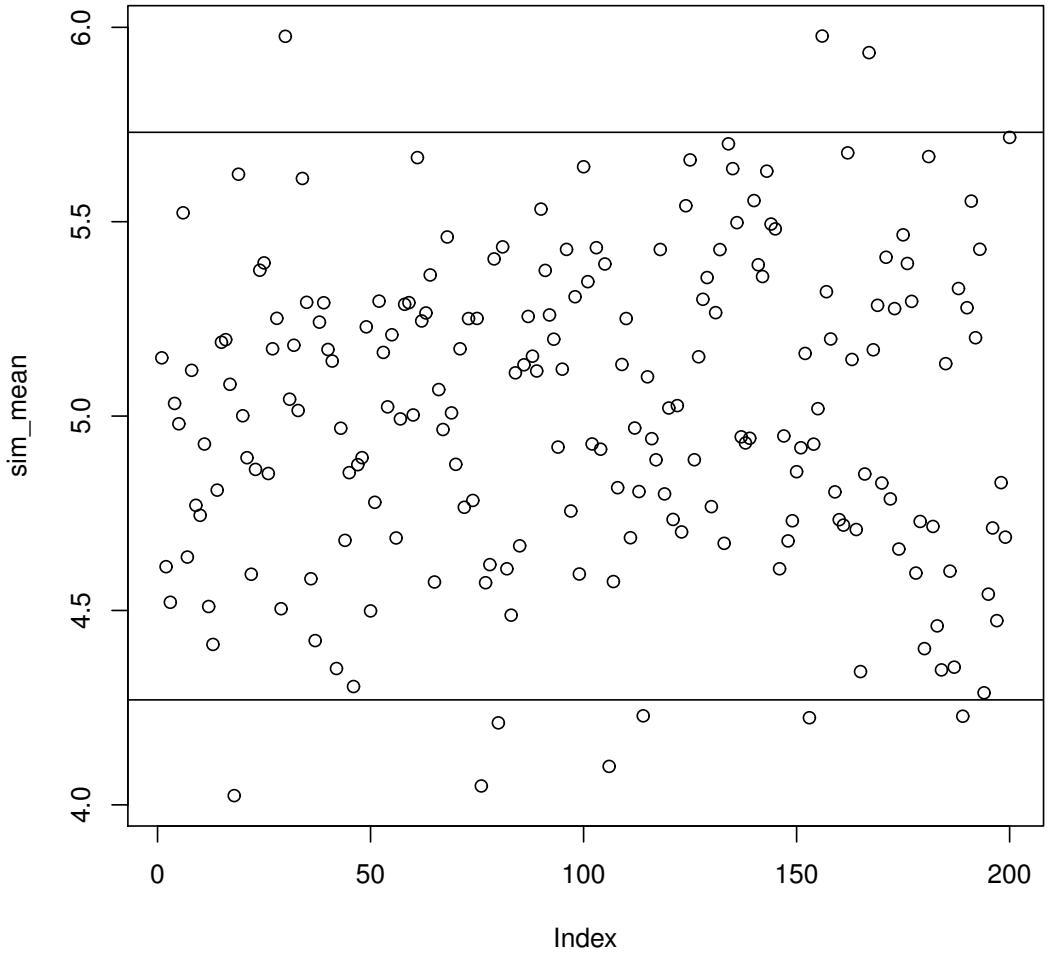
Aufgabe 7.2

a)

b)

c) (zu **Python**)

```r
n <- 60
sim <- matrix(runif(n * 200, min = 0, max = 10), ncol = n)
sim_mean <- apply(sim, 1, "mean")
plot(sim_mean)
abline(h = 5.73)
abline(h = 4.27)
```

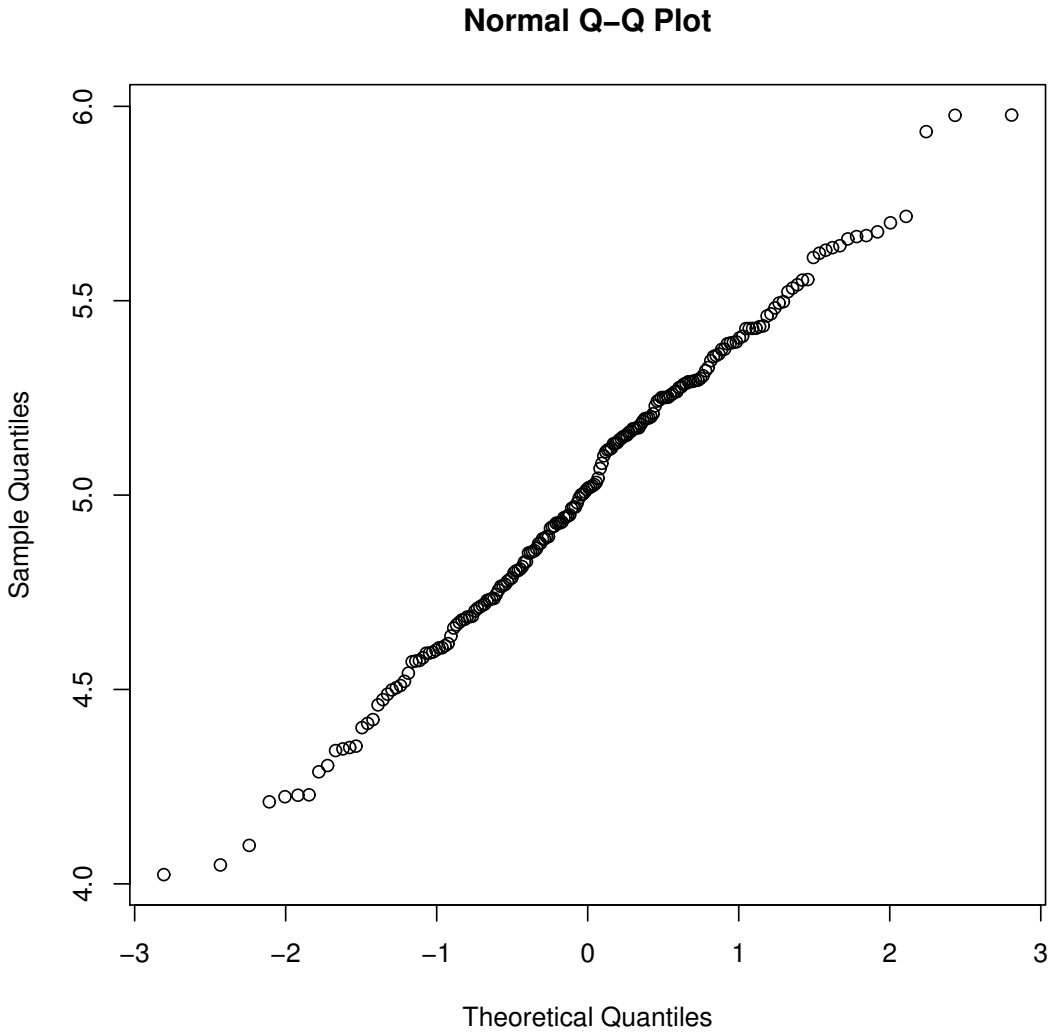



```r
d <- sum(sim_mean > 5.73) + sum(sim_mean < 4.27)
```

In unserem Beispiel sind es 10.

Weiter bestätigt sich auch der Zentrale Grenzwertsatz: (zu **Python**)

```r
qqnorm(sim_mean)
```



Aufgabe 7.3

a) (zu **Python**)

```
qnorm(c(0.005, 0.995), 31, 6/sqrt(10))

## [1] 26.11271 35.88729
```

b)

c) (zu **Python**)

Diese Option gibt es bei **R** nicht.

(zu **Python**)

```r
qt(0.995, 9)

## [1] 3.249836
```

Aufgabe 7.4

a) (zu **Python**)

Dieser Befehl existiert in **R** nicht.

Aufgabe 7.5

a)

b) (zu **Python**)

```r
x <- c(520, 512, 499, 524, 505)

mean(x)

## [1] 512

var(x)

## [1] 106.5

length(x) - 1

## [1] 4

qt(0.95, df = length(x) - 1) * (sd(x)/sqrt(length(x))) +
    500

## [1] 509.8389
```

(zu **Python**)

```r
qt(0.95, df = 4)

## [1] 2.131847
```

c) (zu **Python**)

```r
x <- c(520, 512, 499, 524, 505)

qt(c(0.025, 0.975), df = length(x) - 1) * (sd(x)/sqrt(length(x))) +
    512

## [1] 499.1862 524.8138
```

d) (zu **Python**)

```r
x <- c(520, 512, 499, 524, 505)

qnorm(c(0.025, 0.975), mean = mean(x), sd = sd(x)/sqrt(length(x)))

## [1] 502.9544 521.0456
```