Serie 5

Aufgabe 5.1

Mit folgender Simulationsaufgabe sollen Sie mit dem Normalplot vertraut werden.

a) Simulieren Sie $n = 10, 20, 50$ und 100 (standard-) normalverteilte Zufallszahlen und betrachten Sie die jeweilige Folge mit einem Normalplot.

Wiederholen Sie diese Simulationen einige Male, bis Sie abschätzen können, wie weit zufällige Abweichungen von einer Geraden im Normalplot üblich sind.

***Python**-Hinweise:*

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as st

x = st.norm.rvs(size=10)
st.probplot(x,plot=plt)
```

b) *Langschwänzige Verteilung*: Simulieren Sie je $n = 20$ und 100 t-verteilte Zufallszahlen mit $\nu = 20,7$ und 3 Freiheitsgraden. Wiederholen Sie diese Simulationen einige Male, bis Sie abschätzen können, wie gross Abweichungen von einer Geraden im Normalplot üblich sind.

***Python**-Hinweise:*

```
x = st.t.rvs(size=10, df=20)
```

c) *Schiefe Verteilung*: Simulieren Sie je $n = 20$ und 100 chiquadrat-verteilte Zufallszahlen mit $\nu = 20$ und 1 Freiheitsgraden. Wiederholen Sie diese Simulationen einige Male, bis Sie abschätzen können, wie gross Abweichungen von einer Geraden im Normalplot üblich sind.

***Python**-Hinweise:*

```
x = st.chi2.rvs(size=10, df=20)
```

Aufgabe 5.2

In dieser Aufgabe untersuchen Sie die Wirkung des Zentralen Grenzwertsatzes mittels Simulation. Gehen Sie von einer Zufallsvariablen X aus, die folgendermassen

verteilt ist: die Werte 0, 10 und 11 werden je mit einer Wahrscheinlichkeit $\frac{1}{3}$ angenommen. Simulieren Sie nun die Verteilung von X sowie die Verteilung des Mittelwerts $\overline{X}_n$ von mehreren X.

a) Simulieren Sie X. Stellen Sie die Verteilung von X mittels eines Histogramms von 1000 Realisierungen von X dar, und vergleichen Sie sie mittels des Normalplots mit der Normalverteilung.

```
# moegliche Werte von X
werte = np.array([0,10,11])

# X simulieren
sim = Series(np.random.choice(werte, size=1000, replace=True))


# Mehrere Grafiken neben- und untereinander
plt.subplot(4,2,1)

# Histogramm erstellen
sim.hist(bins=[0,1,10,11,12], edgecolor="black")
plt.title("Original")

plt.subplot(4,2,2)

# Normalplot erstellen
st.probplot(sim,plot=plt)
plt.title("Normal Q-Q Plot")
```

b) Simulieren Sie nun $\overline{X}_5 = \frac{X_1+X_2+X_3+X_4+X_5}{5}$, wobei die X_i die gleiche Verteilung haben wie X und unabhängig sind. Stellen Sie die Verteilung von $\overline{X}_5$ anhand von 1000 Realisierungen von $\overline{X}_5$ dar, und vergleichen Sie mit der Normalverteilung.

```
n = 5

# X_1,...,X_n simulieren und in einer n-spaltigen
# Matrix (mit 1000 Zeilen) anordnen
sim = np.random.choice(werte, size=n*1000, replace=True)
sim = DataFrame(np.reshape(sim,(n,1000)))

#In jeder Matrixzeile Mittelwert berechnen
sim_mean = sim.mean()

plt.subplot(4,2,3)
sim_mean.hist(edgecolor="black")
plt.title("Mittelwerte von 5 Beobachtungen")
```

```
plt.subplot(4,2,4)
st.probplot(sim_mean,plot=plt)
plt.title("Normal Q-Q Plot")
```

c) Simulieren Sie nun die Verteilung von $\overline{X}_n$ auch für die Fälle, wo $\overline{X}_n$ das Mittel von $n = 10$ resp. $n = 200$ X_i ist.

d) Geben Sie die Verteilung von $\overline{X}_{200}$ an, zusammen mit den Werten der Verteilungsparameter.

Aufgabe 5.3

In einer Studie wurde untersucht, wie bei Mäusen die Aufnahme von Eisen (Fe^{3+}) von der Dosis abhängt. Dazu wurden 54 Mäuse zufällig in 3 Gruppen zu je 18 Mäusen eingeteilt und jeweils mit Dosis hoch, mittel und tief gefüttert (hoch = 10.2 millimolar, mittel=1.2 millimolar, tief=0.3 millimolar). Mittels radioaktiver Markierung wurde der Anteil des zurückgehaltenen Eisens in Prozent nach einer gewissen Zeit bestimmt. Die Daten sind auf Ilias in der Datei `ironF3.dat` abgelegt; Sie können sie einlesen mit dem Befehl

```
iron = pd.read_table("*/ironF3.dat",sep=" ",index_col=False)
```

Für * muss der gesamte Pfad stehen, wo Sie ihre Datei abgespeichert haben.

a) Erstellen Sie für jede der 3 Versuchsbedingungen einen Boxplot, am besten gerade nebeneinander. Wie unterscheiden sich die Daten der verschiedenen Versuchsbedingungen?

b) Transformieren Sie alle Werte mit dem Logarithmus und erstellen Sie wieder die 3 Boxplots wie bei Aufgabe a). Was hat sich durch die Transformation geändert?

c) Erstellen Sie einen Normalplot der Daten bei mittlerer Dosis vor und nach dem Logarithmieren. Wann passt die Normalverteilung besser? Verwenden Sie die **Python**-Funktion

```
st.probplot(...)
```

d) Unter der Annahme, dass die Daten bei mittlerer Dosis normalverteilt sind, schätzen Sie die Parameter μ und σ^2. Wie gross ist die Wahrscheinlichkeit, dass eine Maus mehr als 10 % Eisen zurückhält.

e) (*Zusatzaufgabe*) Unter der Annahme, dass die Daten bei mittlerer Dosis log-normalverteilt sind, schätzen Sie die Parameter μ und σ^2. Wie gross ist die Wahrscheinlichkeit, dass eine Maus mehr als 10 % Eisen zurückhält.

Hinweis: Ist $Y = \log(X)$ normalverteilt ist, so heisst X log-normalverteilt.

Aufgabe 5.4

Ein Statistiker beobachtet, dass ein Angler innerhalb von 2 Stunden 15 Fische fängt. Er nimmt an, dass es sich um einen Poissonprozess handelt und überlegt sich:

a) Mit welcher Wahrscheinlichkeit dauert es länger als 12 Minuten, bis der nächste Fisch anbeisst? *Hinweis:* Benützen Sie die Momentenmethode, um Parameter zu schätzen.

b) Mit welcher Wahrscheinlichkeit beissen innerhalb der nächsten 12 Minuten genau 2 Fische an?

c) Ein Fischer hat die Wartezeiten zwischen zwei Fischfängen aufgeschrieben. Erstellen Sie einen QQ-Plot für die angegebenen Zeitdifferenzen. Tragen Sie dazu die empirischen Quantile der Messungen gegen die theoretischen Quantile der $\mathrm{Exp}(1)$-Verteilung auf. Passt die Exponentialverteilung zu den Messdaten? Bestimmen Sie die Steigung der Regressionsgeraden im QQ-Plot. Was ist die Bedeutung von der Steigung der Regressionsgeraden in diesem QQ-Plot? Schätzen Sie aufgrund der Steigung den Parameter λ der $\mathrm{Exp}(\lambda)$-Verteilung.

i	$x_{(i)}$ in Minuten
1	16.9
2	4.20
3	6.70
4	8.83
5	10.7
6	22.4
7	1.37
8	3.00
9	4.82
10	4.53
11	6.77
12	4.81

Aufgabe 5.5

Für grossangelegte Simulationen müssen im allgemeinen **pseudozufällige** Zahlen generiert werden; diese Zahlen heissen pseudozufällig, da sie mit Hilfe eines Algorithmus erzeugt werden und daher nicht "'wirklich"' zufällig sind. Angenommen wir möchten die Performance von Warteschlangennetzwerken beurteilen mit Hilfe einer Simulation, dann müssen wir zufällige Zeitintervalle zwischen den Ankünften

von Kunden generieren. Wir nehmen an, diese Zeitintervalle folgen einer Exponentialverteilung. Verfügen wir über keinen Zufallsgenerator für exponentialverteilte Zufallszahlen, dann können wir exponentialverteilte Zufallszahlen mit Hilfe von gleichmässig im Intervall $[0,1]$ verteilten Zufallszahlen erzeugen, und zwar mit folgender Überlegung: Sei U eine uniform auf dem Intervall $[0,1]$ verteilte Zufallsvariable, und sei $X = F_X^{-1}(U)$, wobei $F_X(x) = 1 - \mathrm{e}^{-\lambda\, x}$ die kumulative Verteilungsfunktion der Exponentialverteilung ist. Dann gilt

$$P(X \leq x) = P(F_X^{-1}(U) \leq x) = P(U \leq F_X(x)) = F_U(F_X(x)) = F_X(x)\,,$$

wobei wir im letzten Schritt benutzt haben, dass die kumulative Verteilungsfunktion der uniformen Verteilung gegeben ist durch $F_U(u) = \frac{u-0}{1-0} = u$, falls $u \in [0,1]$. Die Zufallsvariable, die durch $X = F_X^{-1}(U)$ definiert wurde, folgt also einer Exponentialverteilung.

a) Lösen Sie $F_X(x) = 1 - \mathrm{e}^{-\lambda\, x} = u$ nach x auf, d. h., bestimmen Sie die Funktion $F_X^{-1}(U)$. Generieren Sie nun $n = 1000$ Zufallszahlen $X_i \sim \mathrm{Exp}(\lambda = 2)$ $(i = 1, \ldots, n)$ mit $F_X^{-1}(U)$, wobei U uniform auf $[0,1]$ verteilt ist. **Hinweis:** Benützen Sie die **`Python`** -Funktion **`uniform.rvs()`**.

b) Vergleichen Sie das Histogramm der in Aufgabe a) generierten 1000 exponentialverteilten Zufallszahlen mit dem Histogramm der mit Hilfe des **`Python`** -Befehls **`expon.rvs()`** erzeugten Zufallszahlen.

c) Erstellen Sie einen QQ-Plot, um zu überprüfen, dass die 1000 generierten Zahlen exponentialverteilt sind.

Aufgabe 5.6

In dieser Aufgabe geht es um Parameterschätzung. Wir betrachten eine stetige Verteilung mit folgender Dichte:

$$f(x) = \begin{cases} \dfrac{\alpha}{x^{\alpha+1}} & x \geq 1 \\ 0 & x < 1 \end{cases}$$

wobei $\alpha > 0$ ein unbekannter Parameter ist. Wir wollen den Parameter α aus einer Stichprobe schätzen.

a) Bestimmen Sie die Likelihood- und die Log-Likelihood-Funktion basierend auf n unabhängigen identisch verteilten Beobachtungen $x_1, \ldots, x_n$ einer Zufallsvariablen mit obiger Dichte.

b) Bestimmen Sie den zugehörigen Maximum-Likelihood-Schätzer für α. Schreiben Sie zuerst die allgemeine Formel für n Beobachtungen hin und berechnen Sie

den Schätzer dann für die folgende konkrete Stichprobe:

x_1	x_2	x_3	x_4	x_5
12.0	4.0	6.9	27.9	15.4

c) Bestimmen Sie den Momentenschätzer für α, wieder zuerst allgemein basierend auf n unabhängigen Beobachtungen $x_1, \ldots, x_n$ und dann für obige Stichprobe. *Hinweise*: Der Erwartungswert einer Zufallsvariablen X mit obiger Dichte f ist

$$\mathrm{E}[X] = \frac{\alpha}{\alpha - 1} \text{ für } \alpha > 1$$

Für $\alpha \leq 1$ ist der Erwartungswert gleich ∞ und der Momentenschätzer ist nicht definiert. Sie müssen für diese Teilaufgabe also annehmen, dass $\alpha > 1$.

d) Vergleichen Sie den Maximum-Likelihood und den Momentenschätzer für obige Stichprobe. Ist der Momentenschätzer hier sinnvoll?

Aufgabe 5.7

Die Inkubationszeit eines Virus ist definiert als die Zeitspanne (gemessen in Tagen) von der Ansteckung bis zum Ausbruch der Krankheit. Diese Inkubationszeit wird mit folgender Dichte beschrieben:

$$f(x) = \begin{cases} 0 & x \leq 0 \\ \dfrac{1}{x\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(\log(x)-1)^2}{2\sigma^2}} & x > 0 \end{cases}$$

wobei wir mit $\log(\cdot)$ den natürlichen Logarithmus bezeichnen. Hier ist $\sigma > 0$ der unbekannte Parameter, welchen es zu schätzen gilt. Wir möchten nun den Parameter der Verteilung kennen, dazu nehmen wir eine Stichprobe. Das Ziel ist, diesen unbekannten Parameter daraus zu schätzen, um somit zum Beispiel die Dauer von Quarantänemassnahmen für ein neuartiges Virus abschätzen zu können. Wir betrachten die Daten (in Tagen):

x_1	x_2	x_3	x_4	x_5
1.62	5.29	3.93	1.49	1.57

Hinweis: Für eine Zufallsvariable X mit obiger Dichte gilt

$$\mathrm{E}[X] = \mathrm{e}^{1+\sigma^2/2}$$

a) Bestimmen Sie die Likelihood- und die log-Likelihood-Funktion basierend auf n unabhängigen identisch verteilten Beobachtungen $x_1, \ldots, x_n$ einer Zufallsvariable mit obiger Dichte. Berechnen Sie daraus den Maximum-Likelihood Schätzer für den unbekannten Parameter σ. Die konkreten Werte brauchen Sie nicht einzusetzen.

b) Bestimmen Sie den Momentenschätzer für σ, sowohl basierend auf n unabhängigen Beobachtungen $x_1, \ldots, x_n$, wie auch den realisierten Wert für die gegebene Stichprobe.

c) Nehmen wir für diese Teilaufgabe an, dass wir folgende Daten beobachtet haben: Welcher Schätzer (Momentenschätzer oder Maximum-Likelihood Schätzer) ist

x_1	x_2	x_3
2.12	1.63	4.21

bei diesen Daten zu bevorzugen? Begründen Sie.

Kurzlösungen vereinzelter Aufgaben

A 5.3:

d) 0.371 e) 0.271

A 5.4:

a) 0.223 b) 0.251 c) Aus dem QQ-Plot folgt $\lambda = 1/7.4$.

A 5.5: $X_i = F_X^{-1}(U) = \frac{-\log(1-U)}{\lambda}$ mit $i = 1, \ldots, 1000$.

A 5.7:

a) $\log l(\sigma; x_1, \ldots, x_n) = \log(1) - \frac{n}{2}\log(2\pi) - n\log(\sigma) - \sum_{i=1}^{n}\log(x_i) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\log(x_i) - 1\right)^2$ und $\hat{\sigma}_{\text{MLE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(x_i) - 1\right)^2}$

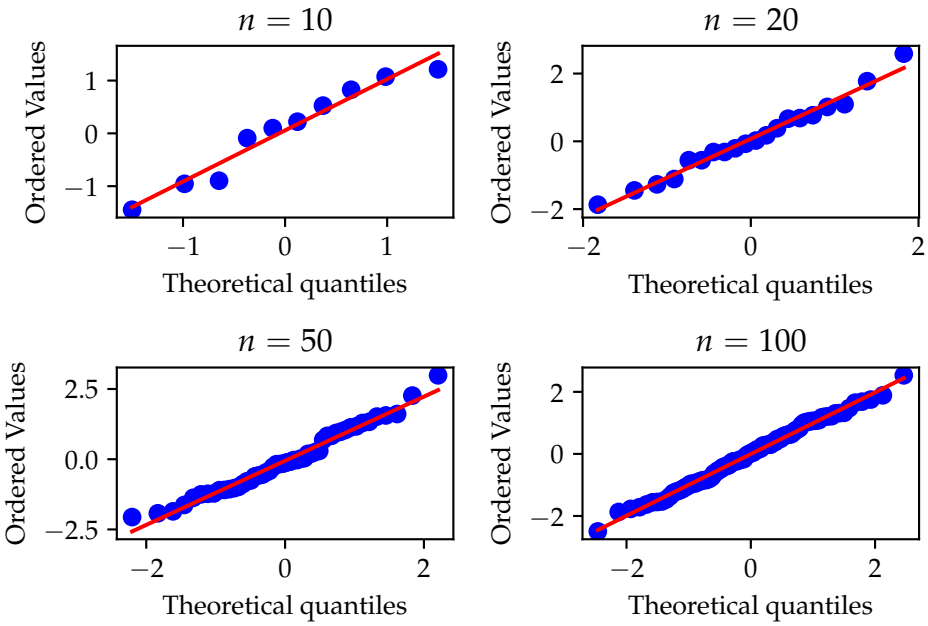
b) $\hat{\sigma}_{\text{MoM}} = \sqrt{\log\left(\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)^2\right) - 2} = 0.2119$

Musterlösungen zu Serie 5

Lösung 5.1

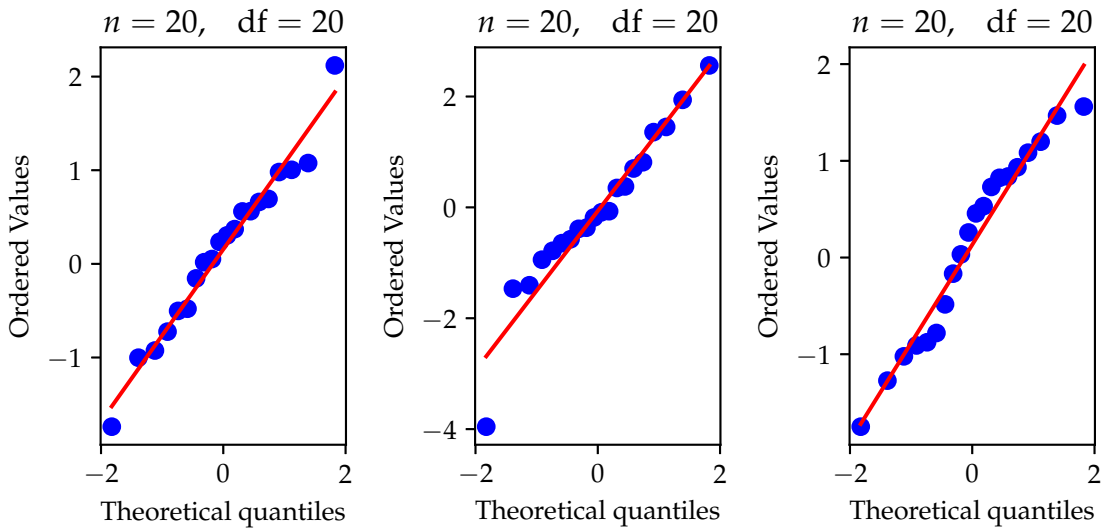
a) (zu R)

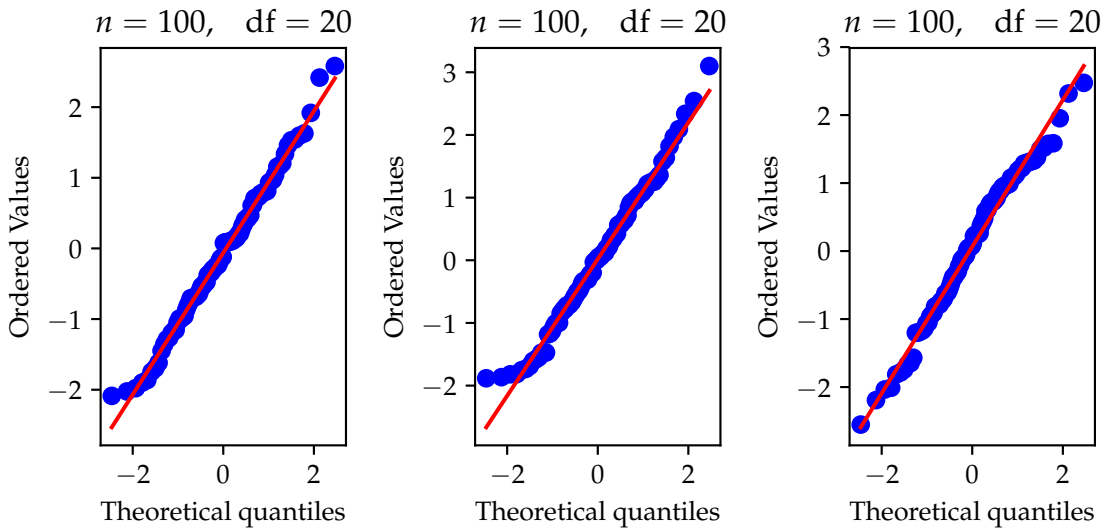
```
plt.subplot(2, 2, 1)
x = st.norm.rvs(size = 10)
st.probplot(x, plot = plt)
plt.title("n=10")

plt.subplot(2, 2, 2)
x = st.norm.rvs(size = 20)
st.probplot(x, plot = plt)
plt.title("n=20")

plt.subplot(2, 2, 3)
x = st.norm.rvs(size = 50)
st.probplot(x, plot = plt)
plt.title("n=50")

plt.subplot(2, 2, 4)
x = st.norm.rvs(size = 100)
st.probplot(x, plot = plt)
plt.title("n=100")

plt.tight_layout()
plt.show()
```



b) (zu R)

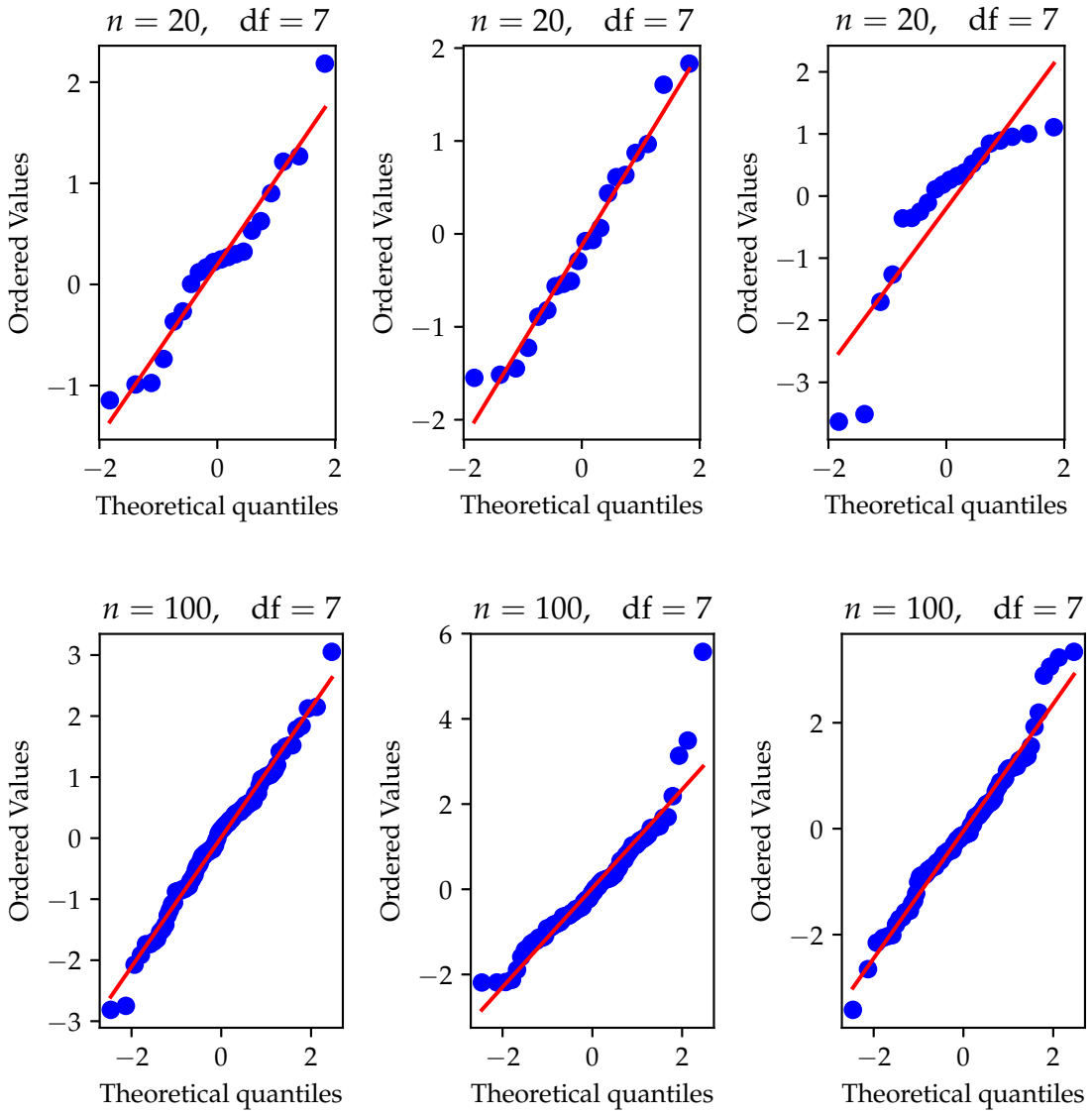
```
for i in range(1,4):
    plt.subplot(1,3,i)
    x = st.t.rvs(size=20, df=20)
    st.probplot(x,plot=plt)
    plt.title("n=20, df=20")

plt.tight_layout()
plt.show()

for i in range(1,4):
    plt.subplot(1,3,i)
    x = st.t.rvs(size=100, df=20)
    st.probplot(x,plot=plt)
    plt.title("n=100, df=20")

plt.tight_layout()
plt.show()
```





(zu R)

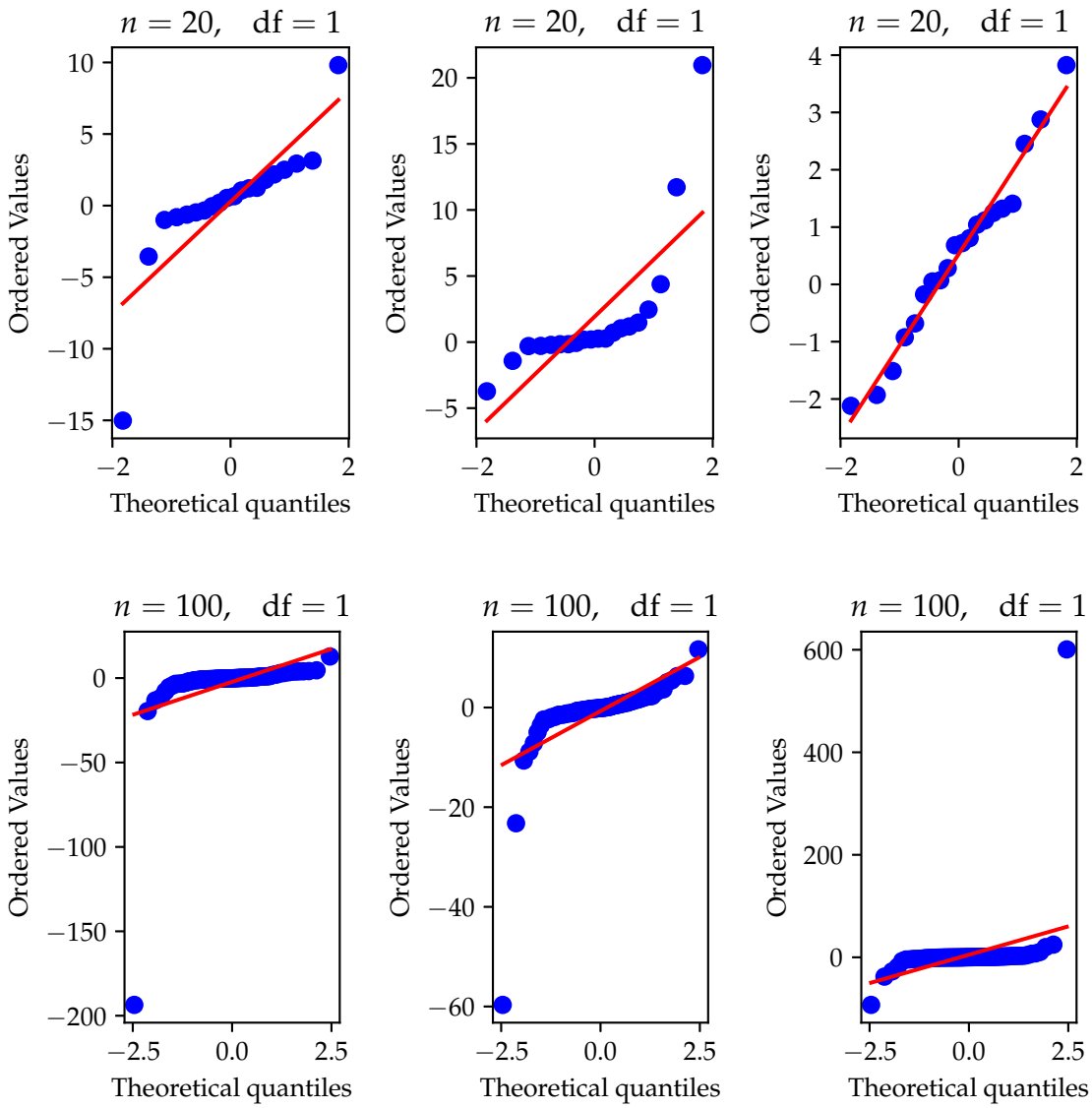
```
for i in range(1,4):
    plt.subplot(1,3,i)
    x = st.t.rvs(size=20, df=7)
    st.probplot(x,plot=plt)
    plt.title("n=20, df=7")

plt.tight_layout()
plt.show()

for i in range(1,4):
    plt.subplot(1,3,i)
    x = st.t.rvs(size=100, df=7)
    st.probplot(x,plot=plt)
    plt.title("n=100, df=7")

plt.tight_layout()
plt.show()
```



(zu **R**)

```
for i in range(1,4):
    plt.subplot(1,3,i)
    x = st.t.rvs(size=20, df=1)
    st.probplot(x,plot=plt)
    plt.title("n=20,df=1")

plt.tight_layout()
plt.show()
```

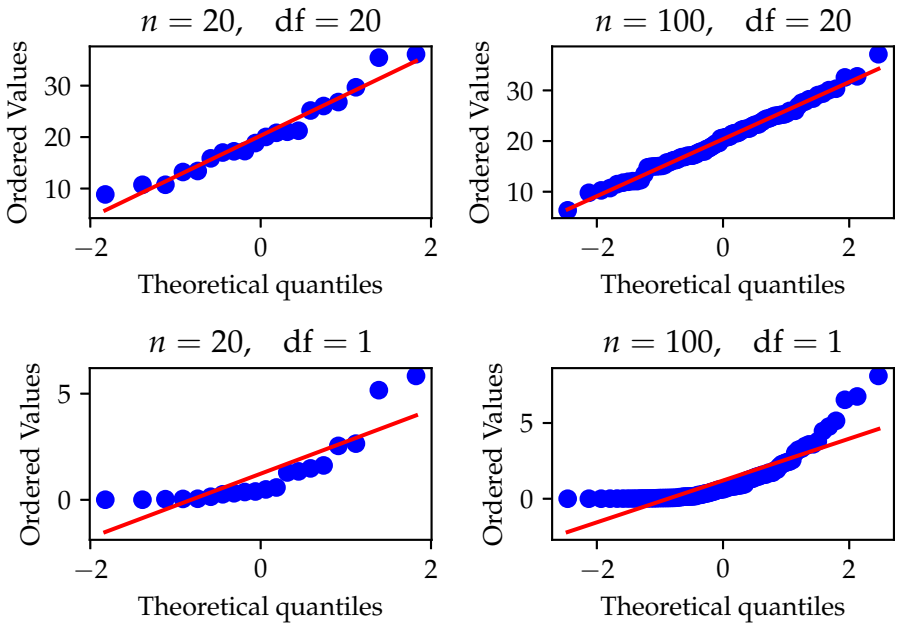
```
plt.figure(figsize=(6,3))

for i in range(1,4):
    plt.subplot(1,3,i)
    x = st.t.rvs(size=100, df=1)
    st.probplot(x,plot=plt)
    plt.title("n=100,df=1")

plt.tight_layout()
plt.show()
```



Wir beobachten, dass je grösser n, desto klarer ist die Langschwänzigkeit sichtbar.

c) (zu **R**)

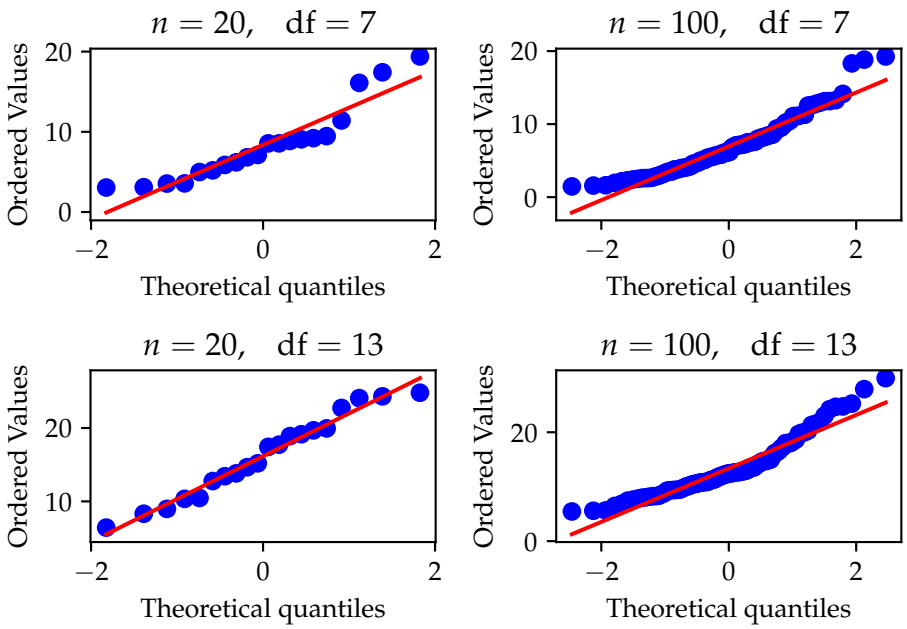
```
plt.subplot(2, 2, 1)
x = st.chi2.rvs(size = 20, df = 20)
st.probplot(x, plot = plt)
plt.title("n=20, df=20")

plt.subplot(2, 2, 2)
x = st.chi2.rvs(size = 100, df = 20)
st.probplot(x, plot = plt)
plt.title("n=100, df=20")

plt.subplot(2, 2, 3)
x = st.chi2.rvs(size = 20, df = 1)
st.probplot(x, plot = plt)
plt.title("$n=20, df=1")

plt.subplot(2, 2, 4)
x = st.chi2.rvs(size = 100, df = 1)
st.probplot(x, plot = plt)
plt.title("n=100, df=1")


plt.tight_layout()
plt.show()
```



Bei einem Freiheitsgrad ist die Rechtsschiefe klar sichtbar. Bei 20 Freiheitsgraden sind bloss noch Hinweise auf eine Rechtsschiefe. (zu R)

```
plt.subplot(2, 2, 1)
x = st.chi2.rvs(size = 20, df = 7)
st.probplot(x, plot = plt)
plt.title("n=20, df=7")

plt.subplot(2, 2, 2)
x = st.chi2.rvs(size = 100, df = 7)
st.probplot(x, plot = plt)
plt.title("n=100, df=7")

plt.subplot(2, 2, 3)
x = st.chi2.rvs(size = 20, df = 13)
st.probplot(x, plot = plt)
plt.title("n=20, df=13")

plt.subplot(2, 2, 4)
x = st.chi2.rvs(size = 100, df = 13)
st.probplot(x, plot = plt)
plt.title("n=100, df=13")


plt.tight_layout()
```

```
plt.show()
```



Bei 7 Freiheitsgraden und $n = 100$ ist die Rechtsschiefe klar sichtbar. Ansonsten ist kaum erkennbar, dass die Verteilung rechtsschief ist.

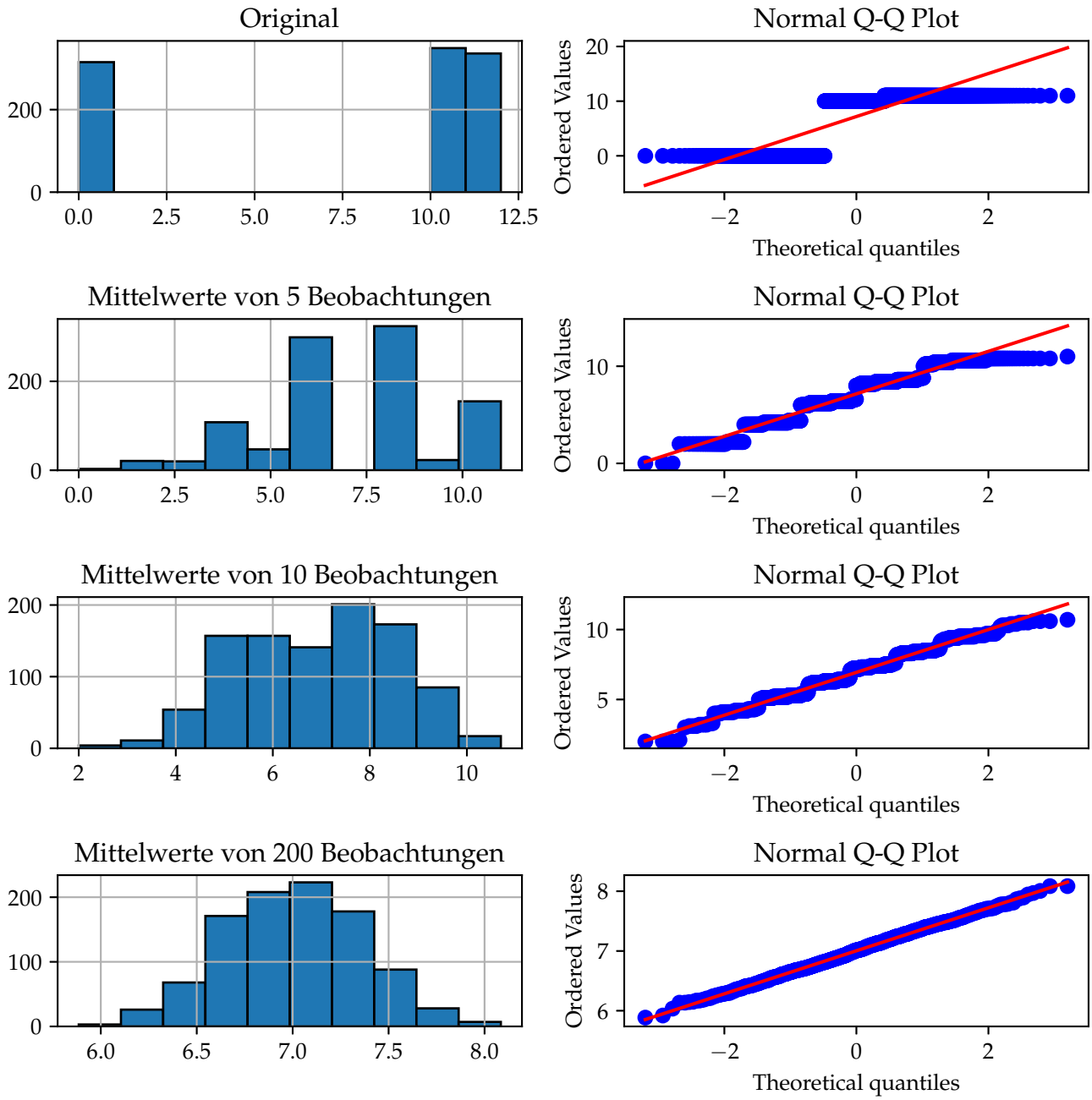
Lösung 5.2

Die untenstehenden Graphiken zeigen, dass die Form der Verteilung des Mittelwerts von unabhängigen Zufallsvariablen auch dann der Normalverteilung immer ähnlicher wird, wenn die Variablen selber überhaupt nicht normal-verteilt sind. An der x-Achse sieht man auch, dass die Varianz immer kleiner wird. (zu **R**)

```
import matplotlib.pyplot as plt
import numpy as np
from pandas import Series, DataFrame
import scipy.stats as st

werte = np.array([0,10,11])
sim = Series(np.random.choice(werte, size=1000, replace=True))

plt.subplot(4,2,1)
sim.hist(bins=[0,1,10,11,12],edgecolor="black")
plt.title("Original")

plt.subplot(4,2,2)
st.probplot(sim,plot=plt)
```

```
plt.title("Normal Q-Q Plot")

n = 5
sim = np.random.choice(werte, size=n*1000, replace=True)
sim = DataFrame(np.reshape(sim,(n,1000)))
sim_mean = sim.mean()

plt.subplot(4,2,3)
sim_mean.hist(edgecolor="black")
plt.title("Mittelwerte von 5 Beobachtungen")

plt.subplot(4,2,4)
st.probplot(sim_mean,plot=plt)
plt.title("Normal Q-Q Plot")


n = 10
sim = np.random.choice(werte, size=n*1000, replace=True)
sim = DataFrame(np.reshape(sim,(n,1000)))
sim_mean = sim.mean()

plt.subplot(4,2,5)
sim_mean.hist(edgecolor="black")
plt.title("Mittelwerte von 10 Beobachtungen")

plt.subplot(4,2,6)
st.probplot(sim_mean,plot=plt)
plt.title("Normal Q-Q Plot")


n = 200
sim = np.random.choice(werte, size=n*1000, replace=True)
sim = DataFrame(np.reshape(sim,(n,1000)))
sim_mean = sim.mean()

plt.subplot(4,2,7)
sim_mean.hist(edgecolor="black")
plt.title("Mittelwerte von 200 Beobachtungen")

plt.subplot(4,2,8)
st.probplot(sim_mean, plot=plt)
plt.title("Normal Q-Q Plot")
```

```
plt.tight_layout()

plt.show()
```



Wir stellen also fest, dass $\overline{X}_n = \frac{U_1+U_2+\ldots+U_n}{n}$ einer Normalverteilung folgt. Der Mit-

telwert $\overline{X}_n$ ergibt sich aus:

$$\mathrm{E}[\overline{X}_n] = \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}(U_i) = \mathrm{E}(U_i) = \frac{1}{3}(0+10+11) = 7$$

Die Standardabweichung von $\overline{X}_n$ folgt aus

$$\begin{aligned}
\mathrm{Var}[\overline{X}_n] &= \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(U_i) \\
&= \frac{\mathrm{Var}(U_i)}{n} \\
&= \frac{1}{n}\cdot\left((0-7)^2\cdot\frac{1}{3} + (10-7)^2\cdot\frac{1}{3} + (11-7)^2\cdot\frac{1}{3}\right) \\
&= \frac{24.67}{n}
\end{aligned}$$

Somit ist die Standardabweichung von $\overline{X}_n$, also der Standardfehler, gegeben durch

$$\sigma_{\overline{X}_n} = \sqrt{\frac{24.67}{n}}$$

Im Falle von $n = 200$ ergibt dies $\sigma_{\overline{X}_{200}} = 0.35$. Wir überprüfen dies in unserem Simulationsbeispiel mit $n = 200$: (zu **R**)

```
import numpy as np
from pandas import Series, DataFrame

werte = np.array([0,10,11])
n = 200
sim = np.random.choice(werte, size=n*1000, replace=True)
sim = DataFrame(np.reshape(sim,(n,1000)))

sim_mean = sim.mean()

sim_mean.mean()

sim_mean.std()



## 7.00967
## 0.35694677768835803
```

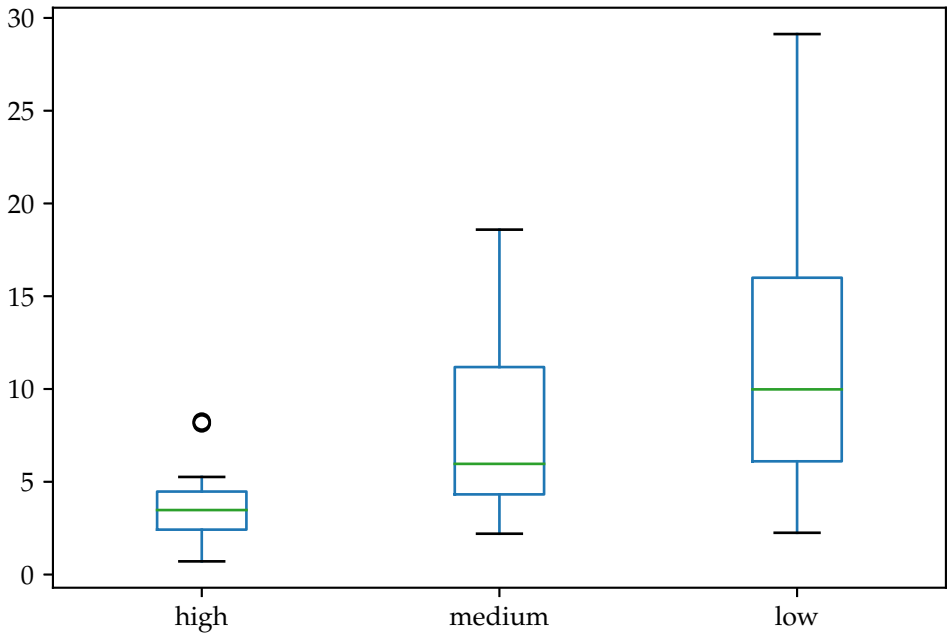
Experiment und Berechnung sind also in guter Übereinstimmung. $\overline{X}_n$ folgt also der

Verteilung $\mathcal{N}(7, 0.12)$.

Lösung 5.3

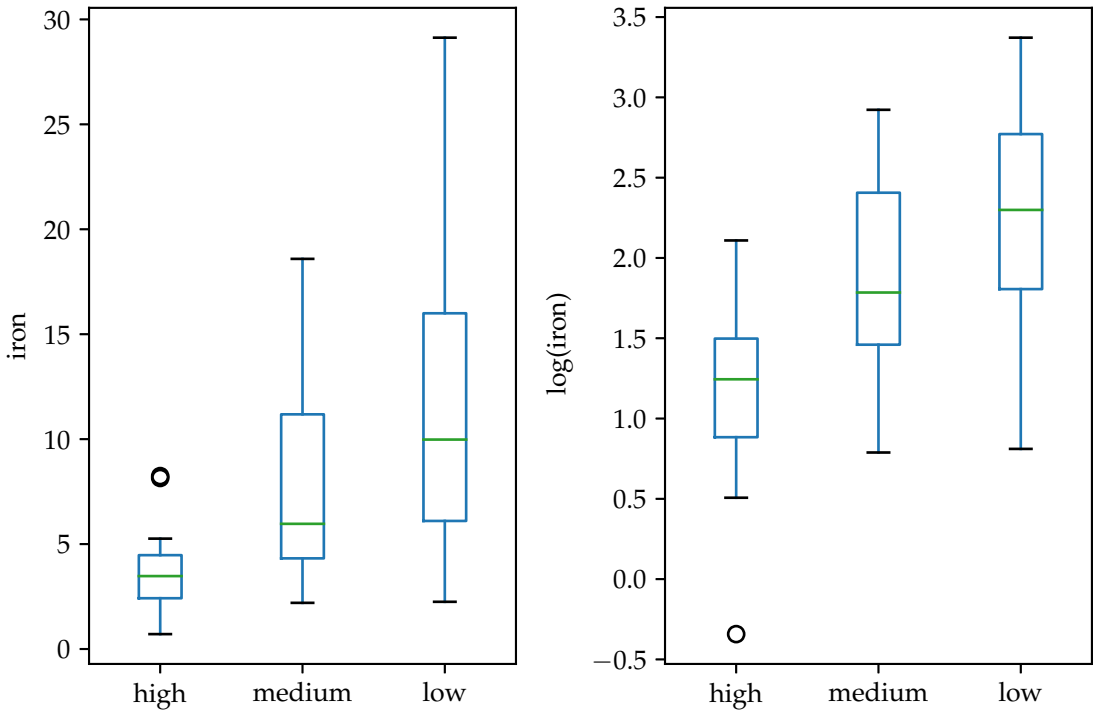
a) Die drei Gruppen unterscheiden sich sowohl in der Lage wie auch in der Streuung. Bei tiefer Dosis ist der Anteil des zurückgehaltenen Eisens höher als bei hoher Dosis. Je kleiner die Dosis, desto grösser wird die Streuung. (zu R)

```
import matplotlib.pyplot as plt
import numpy as np
from pandas import Series, DataFrame
import pandas as pd
import scipy.stats as st

iron = pd.read_table("../../../Themen/Statistik_Messdaten/Daten/ironF3.dat"

iron.plot(kind="box")

plt.show()
```



b) Wenn man die Daten logarithmiert, so wird die Varianz „stabilisiert", d. h. alle Gruppen zeigen jetzt eine ähnlich grosse Streuung. Der Unterschied in der Lage ist immer noch ersichtlich. (zu R)

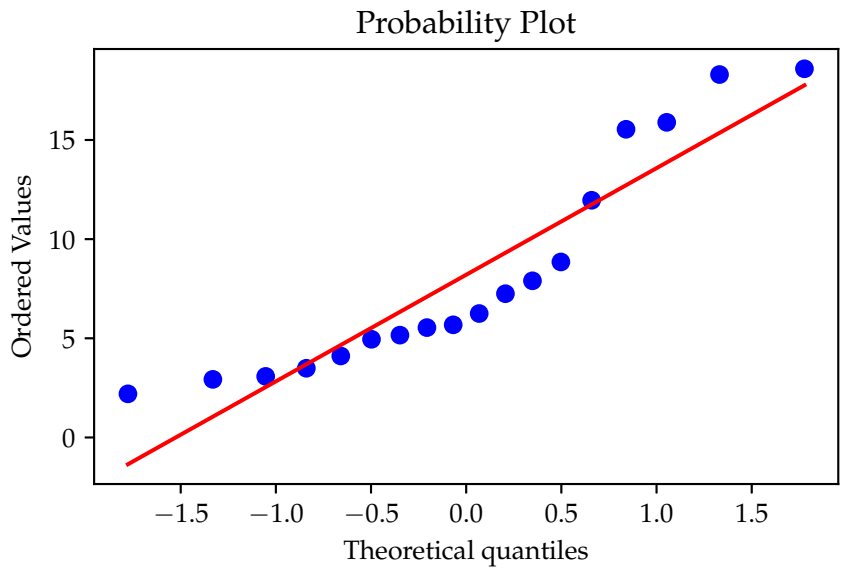
```
plt.subplot(1,2,1)

iron.plot(kind="box",ax=plt.gca())
plt.ylabel("iron")

plt.subplot(1,2,2)
np.log(iron).plot(kind="box",ax=plt.gca())
plt.ylabel("log(iron)")

plt.tight_layout()

plt.show()
```



c) Bei normalverteilten Daten sollte man im Normalplot ungefähr eine Gerade erhalten. Aus den Plots sieht man, dass dies bei den ursprünglichen Daten (oben) nicht der Fall ist, während man in der Abbildung mit den logarithmierten Daten (unten) eher eine Gerade erkennt. Die Lognormalverteilung scheint also besser zu passen. (zu **R**)

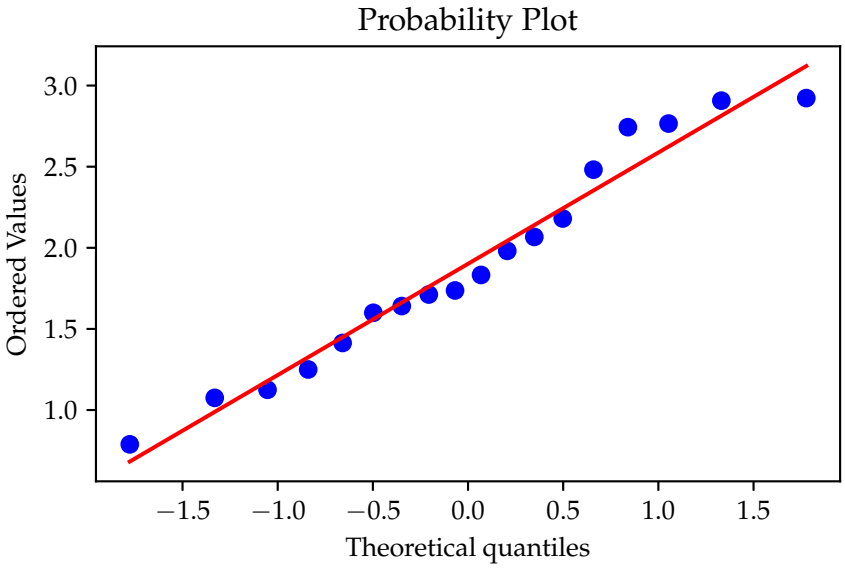
```
st.probplot(iron["medium"], plot=plt)
```

```
plt.show()
```



(zu **R**)

```
st.probplot(np.log(iron["medium"]), plot=plt)

plt.show()
```



d) Wir schätzen den Erwartungswert μ durch den empirischen Mittelwert der Daten

bei mittlerer Dosierung (zu **R**)

```
iron["medium"].mean()

## -c:7: FutureWarning: read_table is deprecated, use read_csv instead.
## 8.20388888888889
```

und die empirische Varianz durch (zu **R**)

```
iron["medium"].var()

## -c:7: FutureWarning: read_table is deprecated, use read_csv instead.
## 29.674013398692814
```

Wenn X den zurückgehaltenen Prozentsatz Eisen bei mittlerer Dosierung bezeichnet, dann ist

$$X \sim \mathcal{N}(\hat{\mu} = 8.20, \hat{\sigma}^2 = 29.7)\,.$$

Die Wahrscheinlichkeit $P(X > 10) = 1 - P(X \leq 10)$ ergibt (zu **R**)

```
1 - st.norm.cdf(x=10, loc=8.204, scale=np.sqrt(29.67))

## 0.37080511954367934
```

e) Bezeichnen wir mit X nach wie vor den zurückgehaltenen Prozentsatz Eisen bei mittlerer Dosierung, so definieren wir die Zufallsvariable $Y = \log(X)$, die normalverteilt ist, wie wir aus dem Normal-Plot erkennen konnten. In diesem Fall heisst X **lognormalverteilt**. Wir schätzen den Erwartungswert von Y durch den arithmetischen Mittelwert (zu **R**)

```
np.log(iron["medium"]).mean()

## -c:7: FutureWarning: read_table is deprecated, use read_csv instead.
## 1.901224840075921
```

und die Varianz durch die empirische Varianz (zu **R**)

```
np.log(iron["medium"]).var()

## -c:7: FutureWarning: read_table is deprecated, use read_csv instead.
## 0.43363756126085784
```

Somit folgt Y der Verteilung

$$Y \sim \mathcal{N}(\hat{\mu} = 1.90, \hat{\sigma}^2 = 0.434)\,.$$

Somit ist die Wahrscheinlichkeit, dass die Mäuse mehr als 10 % des Eisens zurückhalten (zu **R**)

```
1-st.norm.cdf(np.log(10), loc=1.901, scale=np.sqrt(0.4336))

## 0.27097597476867175
```

Lösung 5.4

a) Als Wahrscheinlichkeitsverteilung der Wartezeit T (in Minuten) wählen wir die Exponentialverteilung, d. h. die Wahrscheinlichkeitsdichte ist gegeben durch $f(t) = \lambda \cdot e^{-\lambda t}$ für $t > 0$. Den Parameter λ ermitteln wir mit der Momentenmethode, d. h. wir setzen den beobachten Wert für die mittlere verstrichene Zeit zwischen zwei Fischfängen gleich dem Erwartungswert der Exponentialverteilung

$$E(T) = \frac{1}{\lambda}$$

also $\frac{1}{\hat{\lambda}} = \frac{120}{15} = 8$. Somit ist $\hat{\lambda} = \frac{1}{8}$. Also ist

$$P(T > 12) = 1 - P(T \leq 12) = 1 - (1 - e^{-12/8}) = e^{-1.5} = 0.223\,,$$

wobei wir benützt haben, dass die kumulative Verteilungsfunktion der Exponentialverteilung gegeben ist durch $F(t) = 1 - e^{-\lambda\, t}$.

b) X sei die Anzahl Fische, die in den nächsten 12 Minuten anbeissen; X ist poissonverteilt mit $\lambda = 1.5$ (in 12 Minuten beissen durchschnittlich 1.5 Fische an), also

$$P(X = 2) = e^{-1.5} \cdot \frac{1.5^2}{2!} = 0.251$$

c) Für die Exponentialverteilung haben wir

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{falls } x \geq 0 \\ 0 & \text{falls } x < 0 \end{cases}$$

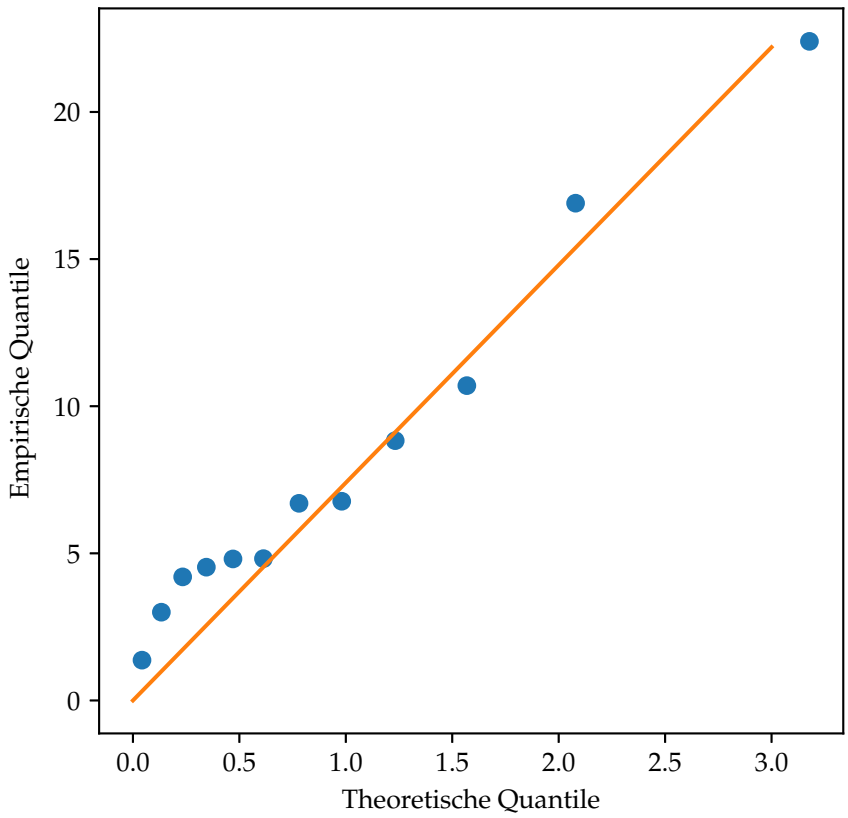
Das $\alpha-$Quantil ist gegeben durch die Beziehung $F(q_\alpha) = \alpha$, also

$$q_\alpha = F^{-1}(\alpha) = -\frac{1}{\lambda}\log(1-\alpha)$$

Die geordneten Beobachtungen $x_{(k)}$ sind gerade die empirischen α_k-Quantile, wobei $\alpha_k = \frac{k-0.5}{n}$. Wir zeichnen daher $(-\log(1-\alpha_k), x_{(k)})$ auf. (zu **R**)

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as st

y = np.sort(np.array([16.9, 4.20, 6.70, 8.83, 10.7 ,22.4, 1.37, 3.00, 4.82,
alphak = (np.arange(1,13)-0.5)/12
x = -np.log(1-alphak)
plt.plot(x,y,"o")
plt.xlabel("Theoretische Quantile")
```

```
plt.ylabel("Empirische Quantile")

x1 = np.linspace(0,3)
y1 = x1*7.4
plt.plot(x1,y1)

plt.show()
```



Wir verwenden hier als Referenzverteilung die $\text{Exp}(1)$ Verteilung. Wenn die Daten wirklich $\text{Exp}(\lambda)$ verteilt sind, erwarten wir ungefähr eine Gerade mit Steigung $1/\lambda$. Denn in diesem Fall sollte gelten:

$$x_{(k)} \approx q_{\alpha_k}$$

für alle k. Folglich ist dann

$$\frac{x_{(k)}}{-\log(1-\alpha_k)} \approx \frac{1}{\lambda}$$

Dies entspricht aber gerade der Steigung der Geraden in der Abbildung, die

durch die Punkte $(-\log(1-\alpha_k), x_{(k)})$ geht. Wir schätzen die Steigung der Geraden mit der Methode der Kleinsten Quadrate und finden für die Steigung der Regressionsgeraden 7.4. (zu **R**)

```
x = x[:,np.newaxis]
a, _, _, _ = np.linalg.lstsq(x,y)

print(a)

## -c:9: FutureWarning: `rcond` parameter will change to the default of mac
## To use the future default and silence this warning we advise to pass `rc
## [7.42075136]
```

Die letzten Befehle erreichen, dass die Regressionsgerade durch den Ursprung gehen soll (im Gegensatz zu z.B. **`np.polyfit(x, y, deg=1)`** für die ursprünglichen Daten `x` und `y`).

Lösung 5.5

a) Die Zufallsvariable X ist definiert als $X = F_X^{-1}(U)$, wobei $U \sim \text{Uniform}([0,1])$ und $F_X(x) = 1 - \lambda e^{-\lambda x}$ die kumulative Verteilungsfunktion der Exponentialverteilung ist. Dann ist

$$X = F_X^{-1}(U) = \frac{-\log(1-U)}{\lambda},$$

Definieren wir die Zufallsvariable

$$V = 1 - U,$$

so gilt $V \sim \text{Uniform}([0,1])$. Die Zufallsvariable X kann äquivalent folglich auch definiert werden als

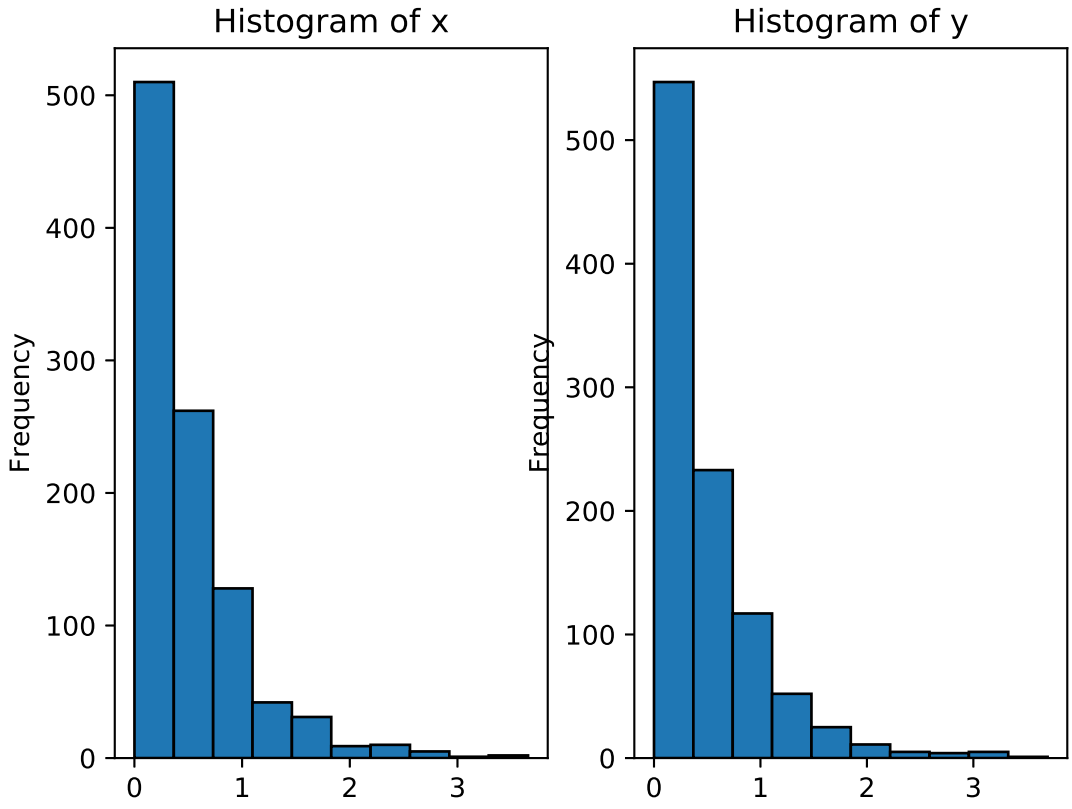
$$X = \frac{-\log(V)}{\lambda}.$$

Wir erzeugen nun aufgrund der obigen Vorschrift mit `R` $n = 1000$ exponentialverteilte Zufallszahlen $X_1, X_2, \ldots, X_{1000}$. (zu **R**)

```
from scipy.stats import uniform
import numpy as np
# Generiere 1000 gleichmaessig verteilte Zufallszahlen
# im Intervall [0,1]
v = uniform.rvs(size=1000, loc=0, scale=1)
# Generiere 1000 exponentialverteilte Zufalsszahlen mit
# Hilfe der gleichmaessig verteilten Zufallszahlen
x = -np.log(v)/2
```

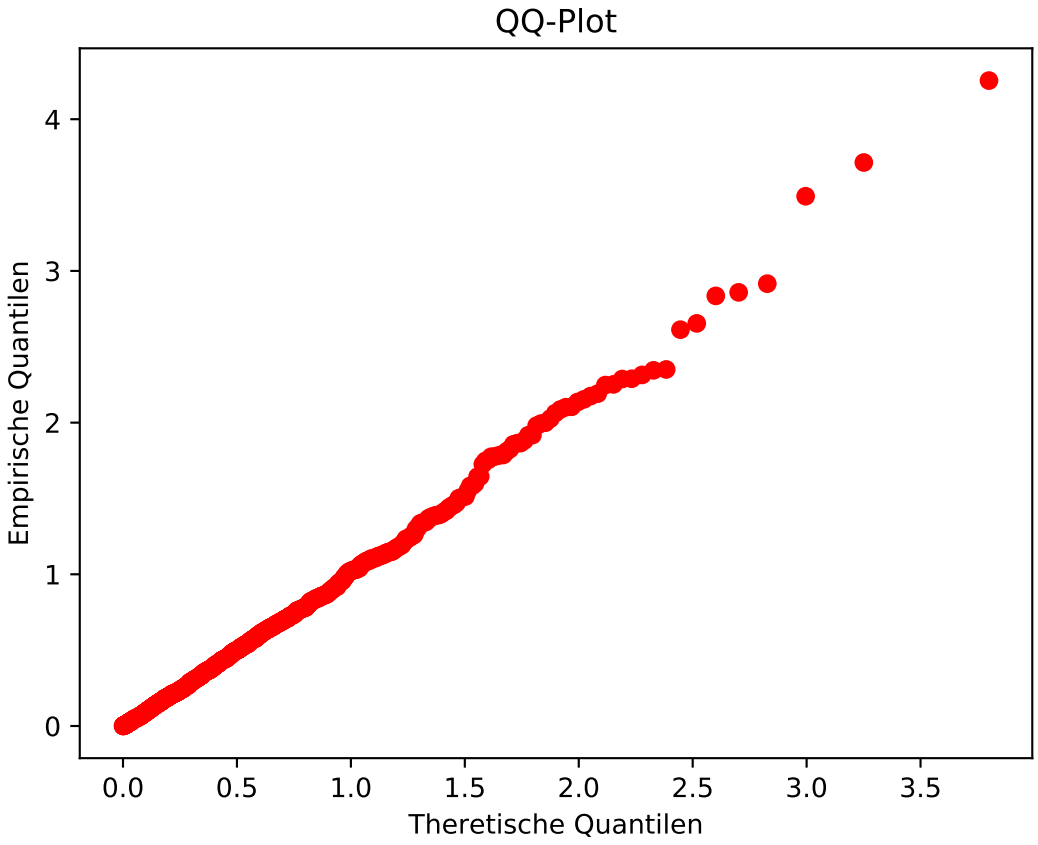
b) (zu **R**)

```python
from scipy.stats import uniform, expon
import numpy as np
import matplotlib.pyplot as plt
from pandas import Series
v = uniform.rvs(size=1000, loc=0, scale=1)
x = Series(-np.log(v)/2)
plt.subplot(121)
x.plot(kind="hist", edgecolor="black")
plt.title("Histogram of x")
y = Series(expon.rvs(size=1000, scale=1/2))
plt.subplot(122)
y.plot(kind="hist", edgecolor="black")
plt.title("Histogram of y")
```



c) Um zu überprüfen, ob die generierten Zahlen X_i ($i = 1, \ldots, 1000$) exponentialverteilt sind, fertigen wir einen qq-Plot an. (zu R)

```
from scipy.stats import uniform, expon
import numpy as np
import matplotlib.pyplot as plt
from pandas import Series
import pandas as pd
v = uniform.rvs(size=1000, loc=0, scale=1)
x = -np.log(v)/2
# qq-Plot
# Berechne die theoretischen Quantilen
n = 1000
theor_quantilen = expon.ppf((np.linspace(1,n,n)-0.5)/n, scale=1/2)
# Berechne die empirischen Quantilen
empir_quantilen = np.sort(x)
plt.plot(theor_quantilen, empir_quantilen,'ro', linewidth=0.5)
plt.xlabel("Theretische Quantilen")
plt.ylabel("Empirische Quantilen")
plt.title("QQ-Plot")
```



Die empirischen Quantile können als eine lineare Funktion der theoretischen Quantilen mit Ordinatenabschnitt 0 betrachtet werden, woraus wir schliessen, dass die generierten Zufallszahlen tatsächlich exponentialverteilt sind.

Lösung 5.6

a)

$$\begin{aligned} l(\alpha; x_1, \ldots, x_n) &= \alpha \frac{1}{x_1^{\alpha+1}} \cdot \alpha \frac{1}{x_2^{\alpha+1}} \cdots \alpha \frac{1}{x_n^{\alpha+1}} \\ &= \alpha^n \frac{1}{\left(\prod_{i=1}^n x_i\right)^{\alpha+1}} \\ \log l(\alpha; x_1, \ldots, x_n) &= n \log \alpha - (\alpha + 1) \sum_{i=1}^n \log x_i \end{aligned}$$

b) Ableiten und Null setzen der Loglikelihood gibt

$$\begin{aligned} \hat{\alpha}_{\text{MLE}} &= \frac{n}{\sum_{i=1}^n \log x_i} \\ &= \frac{5}{\log(12.0) + \log(4.0) + \log(6.9) + \log(27.9) + \log(15.4)} \\ &\approx 0.4 \end{aligned}$$

(zu **R**)

```
import numpy as np
5/(np.log(12.0)+np.log(4.0)+np.log(6.9)+np.log(27.9)+np.log(15.4))

## 0.4213820504741322
```

c) Gleichsetzen von Erwartungswert und Stichprobenmittel, $\mathrm{E}[X] = \frac{1}{n} \sum_{i=1}^n x_i$, ergibt

$$\hat{\alpha}_{\text{MoM}} = \frac{\overline{x}}{\overline{x} - 1} \approx 1.01$$

(zu **R**)

```
import numpy as np
x = np.array([12.0,4.0,6.9,27.9,15.4])
np.mean(x)/(np.mean(x)-1)

## 1.0816993464052287
```

d) Der Momentenschätzer setzt $\alpha > 1$ voraus, was unplausibel erscheint, da der MLE deutlich kleiner als 1 ist.

Lösung 5.7

a) Die Likelihoodfunktion ist gegeben durch

$$l(\sigma; x_1, \ldots, x_n) = \frac{1}{(2\pi\sigma^2)^{n/2}} \cdot \left(\prod_{i=1}^{n} x_i \right)^{-1} \cdot \exp\left(- \sum_{i=1}^{n} \frac{(\log(x_i) - 1)^2}{2\sigma^2} \right)$$

Die log-Likelihood-Funktion ist dann

$$\begin{aligned} \log l(\sigma; x_1, \ldots, x_n) = \log(1) - \frac{n}{2}\log(2\pi) \\ - n\log(\sigma) - \sum_{i=1}^{n} \log(x_i) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\log(x_i) - 1)^2 \end{aligned}$$

Um nun den Maximum-Likelihood Schätzer für σ zu erhalten leiten wir die log-Likelihood-Funktion nach σ ab und setzen das Resultat gleich null.

$$\left. \frac{\mathrm{d}f}{\mathrm{d}\sigma} \right|_{\hat{\sigma}} = -\frac{n}{\hat{\sigma}} + \frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} (\log(x_i) - 1)^2 = 0$$

Wenn wir das auflösen nach $\hat{\sigma}$ erhalten wir den Schätzer

$$\hat{\sigma}_{\mathrm{MLE}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\log(x_i) - 1)^2}$$

b) Mit dem Tipp $\mathrm{E}[X] = \mathrm{e}^{1+\sigma^2/2}$ erhalten wir durch Gleichsetzen der Momente $\mathrm{E}[X] = \frac{1}{n}\sum_{i=1}^{n} x_i$ und auflösen nach σ den Momentenschätzer

$$\hat{\sigma}_{\mathrm{MoM}} = \sqrt{\log\left(\left(\frac{1}{n} \sum_{i=1}^{n} x_i \right)^2 \right) - 2}$$

Der Schätzer nimmt mit den Daten approximativ den Wert 0.2119 an.

c) Der Maximum-Likelihood Schätzer ist zu bevorzugen, da dieser in allen Situationen definiert ist. Der Momentenschätzer hingegen ist nicht definiert für die gegebenen Daten, da wir haben, dass:

$$\log\left(\left(\frac{1}{3} \sum_{i=1}^{3} x_i \right)^2 \right) < 2$$

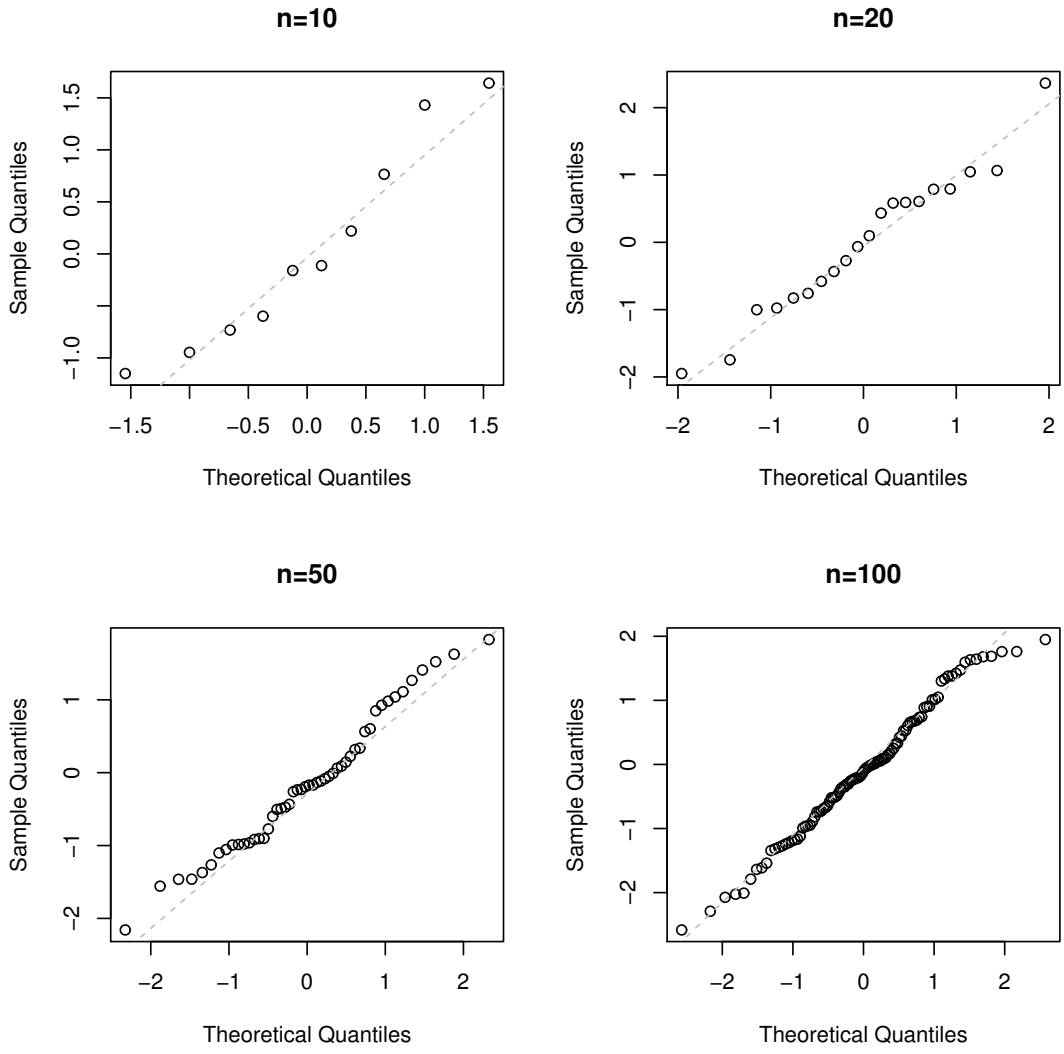
Der Term unter der Wurzel wird negativ.

R-Code

Aufgabe 5.1

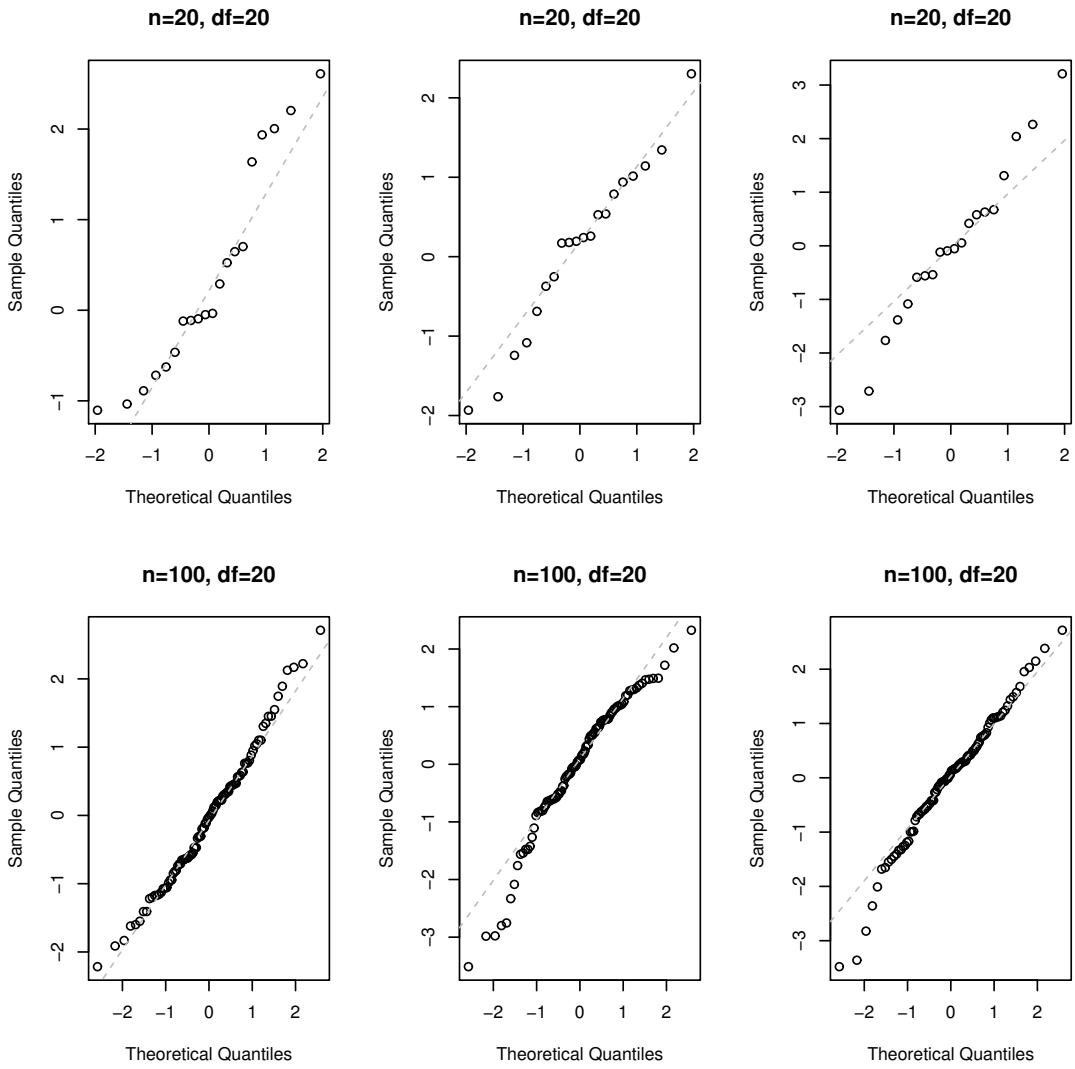
a) (zu **Python**)

```r
par(mfrow = c(2, 2))
x <- rnorm(10)
qqnorm(x, main = "n=10")
qqline(x, col = "grey", lty = 2)
x <- rnorm(20)
qqnorm(x, main = "n=20")
qqline(x, col = "grey", lty = 2)
x <- rnorm(50)
qqnorm(x, main = "n=50")
qqline(x, col = "grey", lty = 2)
x <- rnorm(100)
qqnorm(x, main = "n=100")
qqline(x, col = "grey", lty = 2)
```



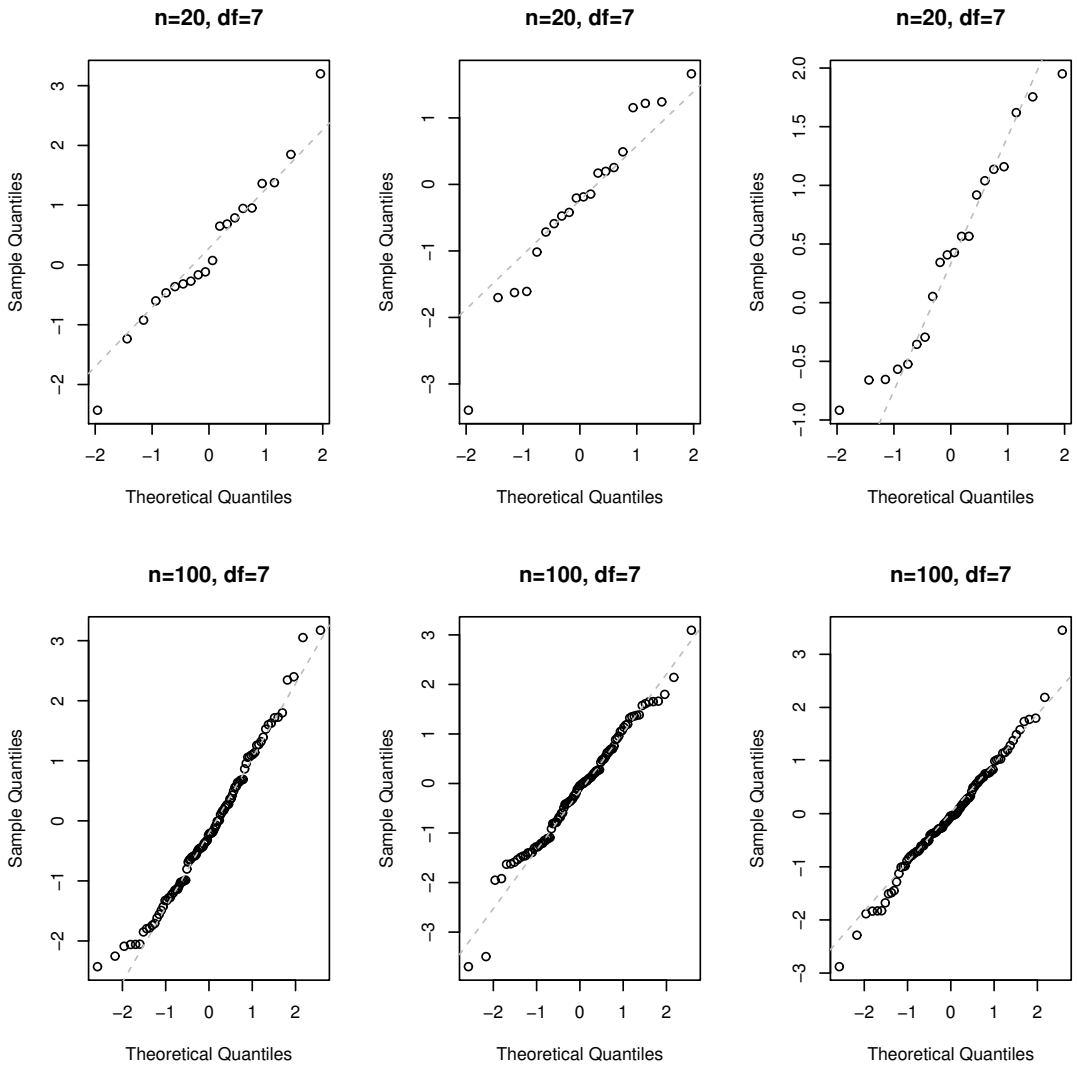
b) (zu **Python**)

```
par(mfrow = c(2, 3))
for (i in 1:3) {
    x <- rt(20, df = 20)
    qqnorm(x, main = "n=20, df=20")
    qqline(x, col = "grey", lty = 2)
}
for (i in 1:3) {
    x <- rt(100, df = 20)
    qqnorm(x, main = "n=100, df=20")
    qqline(x, col = "grey", lty = 2)
}
```



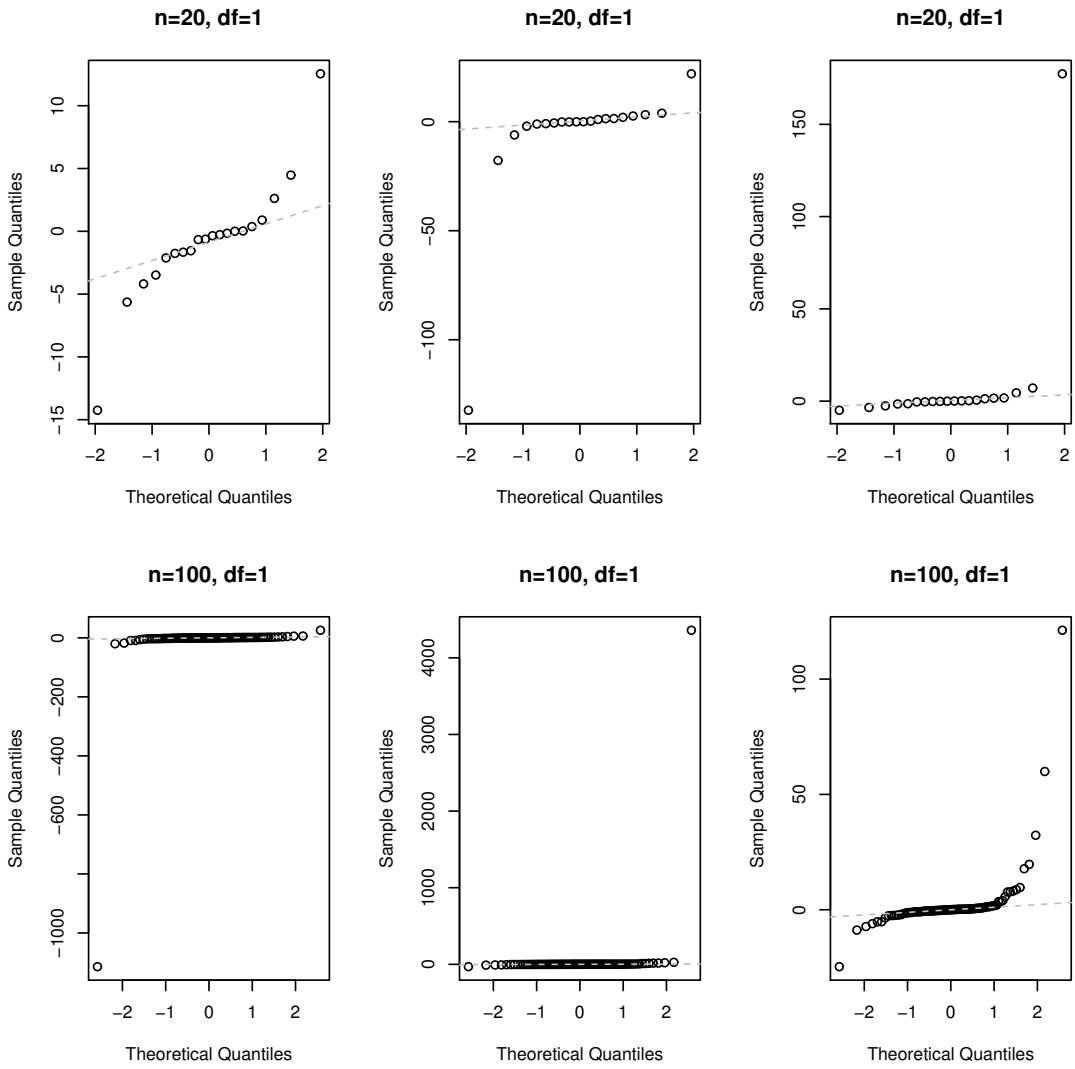
(zu **Python**)

```r
par(mfrow = c(2, 3))
for (i in 1:3) {
    x <- rt(20, df = 7)
    qqnorm(x, main = "n=20, df=7")
    qqline(x, col = "grey", lty = 2)
}
for (i in 1:3) {
    x <- rt(100, df = 7)
    qqnorm(x, main = "n=100, df=7")
    qqline(x, col = "grey", lty = 2)
}
```



(zu **Python**)

```r
par(mfrow = c(2, 3))
for (i in 1:3) {
    x <- rt(20, df = 1)
    qqnorm(x, main = "n=20, df=1")
    qqline(x, col = "grey", lty = 2)
}
for (i in 1:3) {
    x <- rt(100, df = 1)
    qqnorm(x, main = "n=100, df=1")
    qqline(x, col = "grey", lty = 2)
}
```

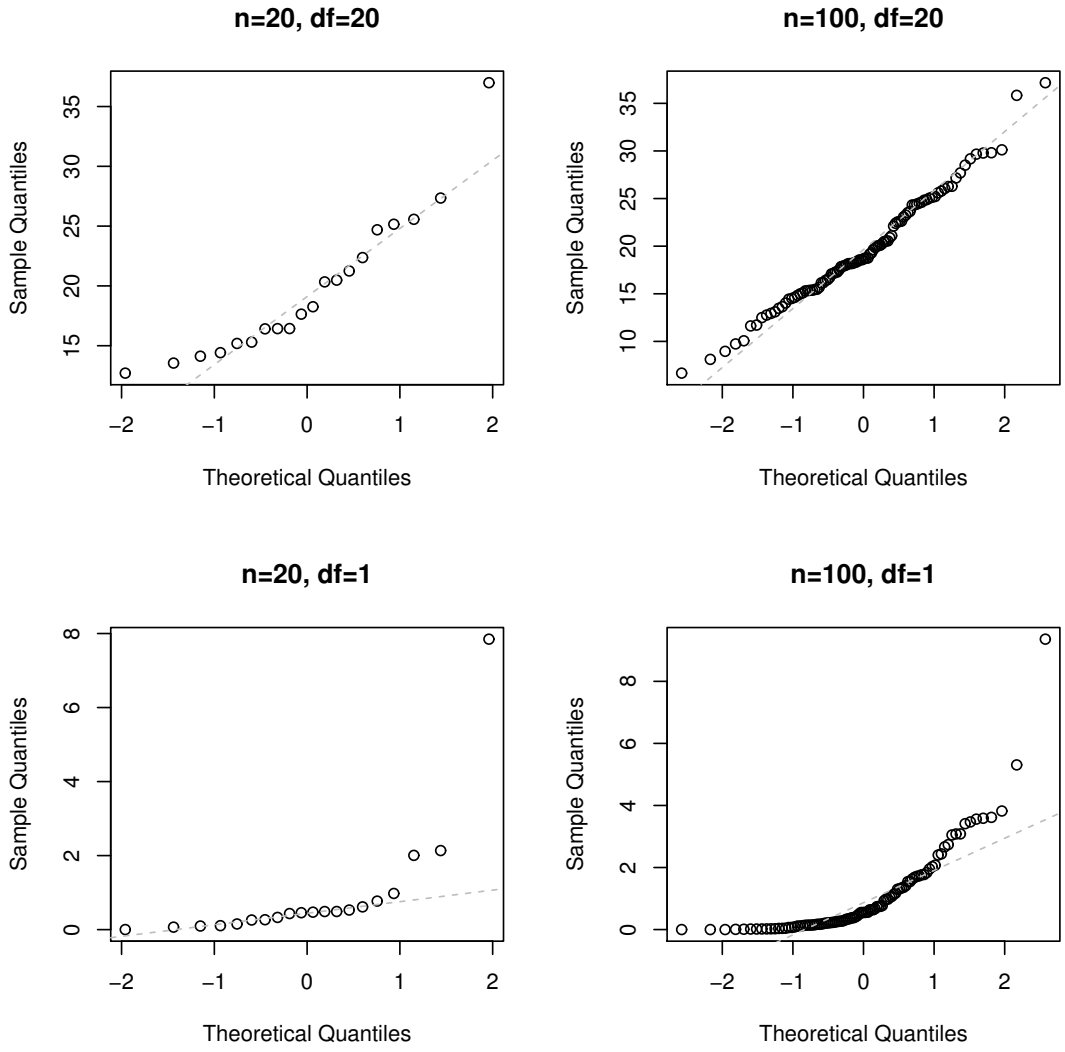


Wir beobachten, dass je grösser n, desto klarer ist die Langschwänzigkeit sichtbar.

c) (zu **Python**)

```
par(mfrow = c(2, 2))
x <- rchisq(20, df = 20)
qqnorm(x, main = "n=20, df=20")
qqline(x, col = "grey", lty = 2)
x <- rchisq(100, df = 20)
qqnorm(x, main = "n=100, df=20")
qqline(x, col = "grey", lty = 2)
x <- rchisq(20, df = 1)
qqnorm(x, main = "n=20, df=1")
qqline(x, col = "grey", lty = 2)
```

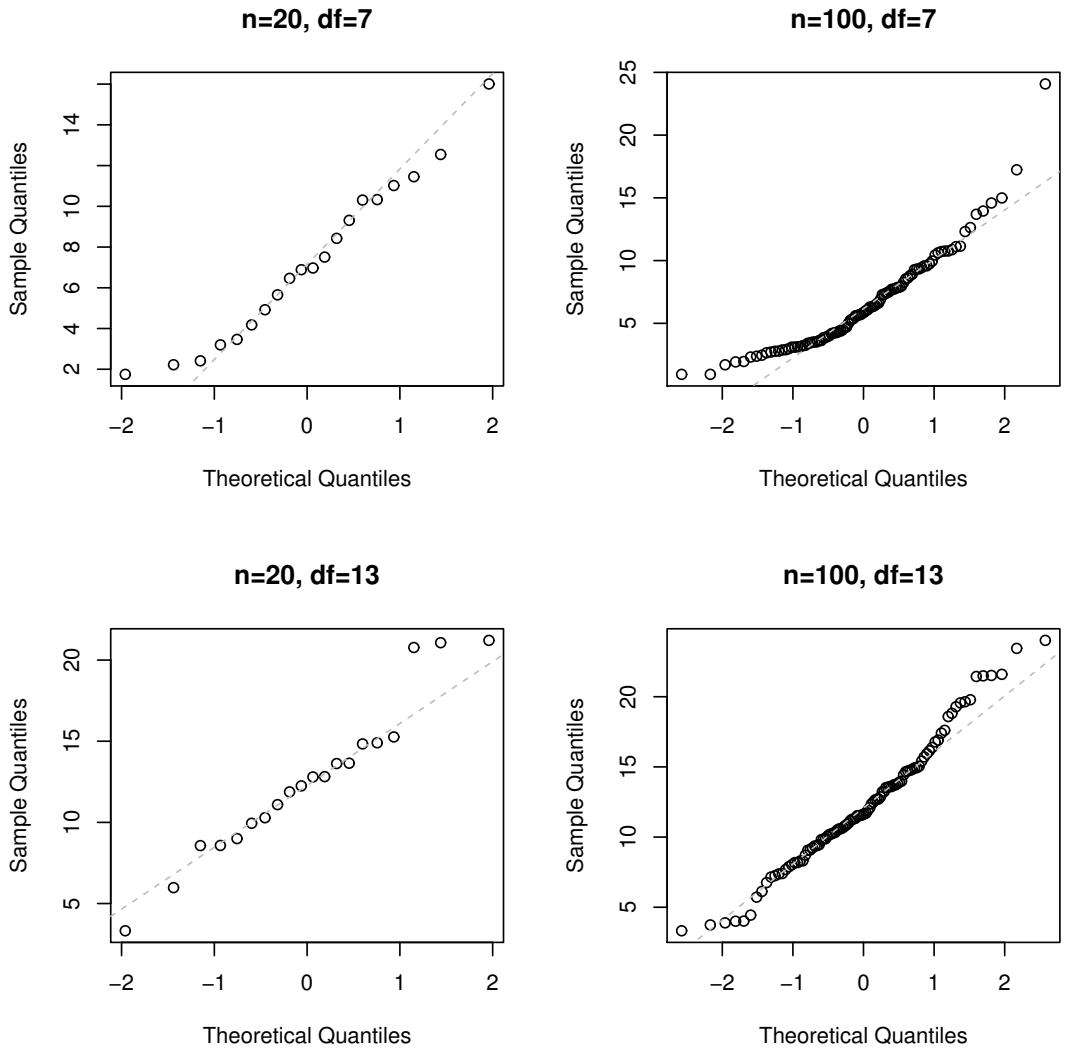
```r
x <- rchisq(100, df = 1)
qqnorm(x, main = "n=100, df=1")
qqline(x, col = "grey", lty = 2)
```



Bei einem Freiheitsgrad ist die Rechtsschiefe klar sichtbar. Bei 20 Freiheitsgraden sind bloss noch Hinweise auf eine Rechtsschiefe. (zu **Python**)

```r
par(mfrow = c(2, 2))
x <- rchisq(20, df = 7)
qqnorm(x, main = "n=20, df=7")
qqline(x, col = "grey", lty = 2)
x <- rchisq(100, df = 7)
qqnorm(x, main = "n=100, df=7")
qqline(x, col = "grey", lty = 2)
x <- rchisq(20, df = 13)
```

```r
qqnorm(x, main = "n=20, df=13")
qqline(x, col = "grey", lty = 2)
x <- rchisq(100, df = 13)
qqnorm(x, main = "n=100, df=13")
qqline(x, col = "grey", lty = 2)
```

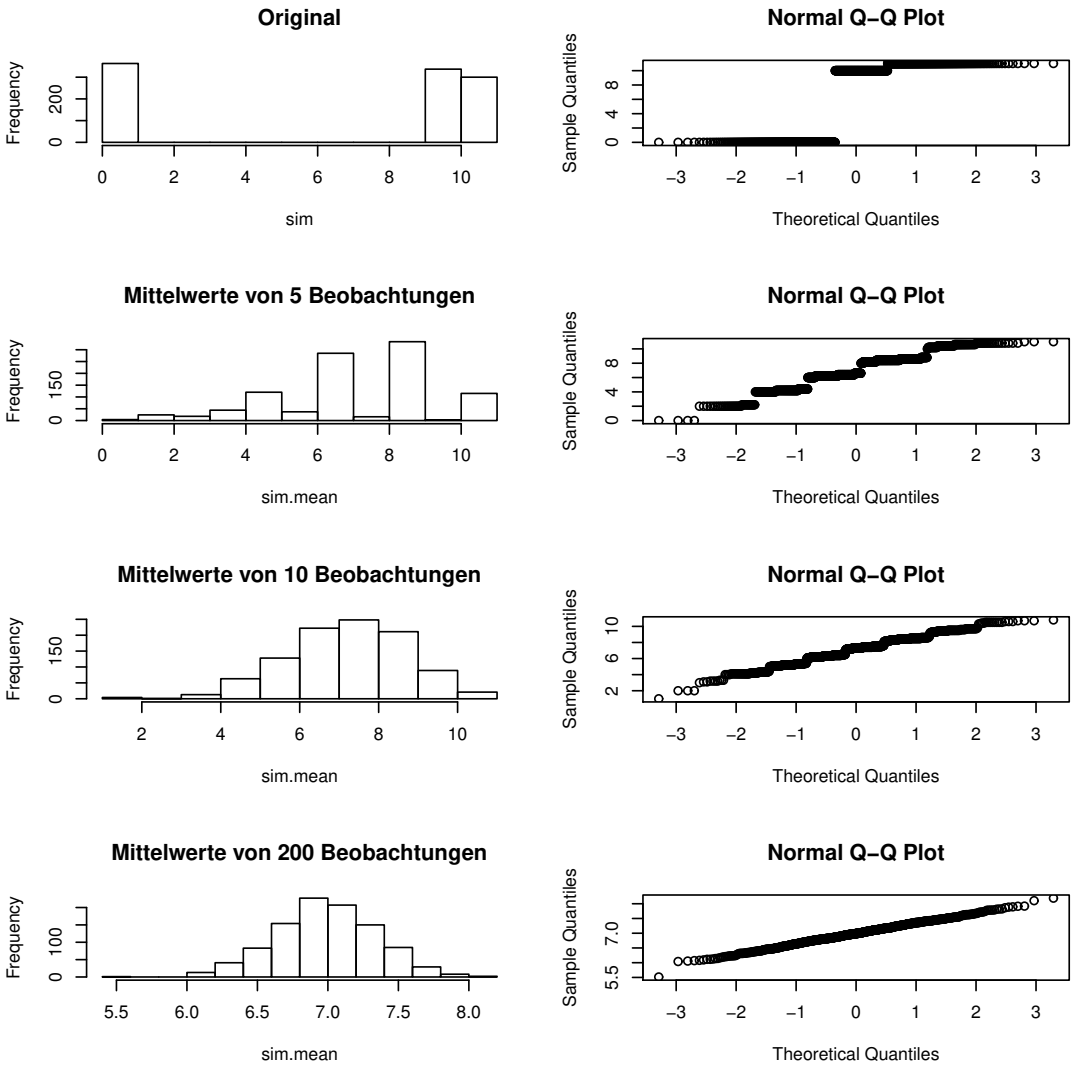


Bei 7 Freiheitsgraden und $n = 100$ ist die Rechtsschiefe klar sichtbar. Ansonsten ist kaum erkennbar, dass die Verteilung rechtsschief ist.

Aufgabe 5.2

a)-d) (zu **Python**)

```r
# Mehrere Grafiken neben- und untereinander
par(mfrow = c(4, 2))
# moegliche Werte von X
werte <- c(0, 10, 11)
# X simulieren
sim <- sample(werte, 1000, replace = TRUE)
# Histogramm erstellen
hist(sim, main = paste("Original"))
# Normalplot erstellen
qqnorm(sim)
n <- 5
# X_1,...,X_n simulieren und in einer n-spaltigen
# Matrix (mit 1000 Zeilen) anordnen
sim <- matrix(sample(werte, n * 1000, replace = TRUE),
    ncol = n)
# In jeder Matrixzeile Mittelwert berechnen
sim.mean <- apply(sim, 1, "mean")
hist(sim.mean, main = paste("Mittelwerte von", n,
    "Beobachtungen"))
qqnorm(sim.mean)
n <- 10
# X_1,...,X_n simulieren und in einer n-spaltigen
# Matrix (mit 1000 Zeilen) anordnen
sim <- matrix(sample(werte, n * 1000, replace = TRUE),
    ncol = n)
# In jeder Matrixzeile Mittelwert berechnen
sim.mean <- apply(sim, 1, "mean")
hist(sim.mean, main = paste("Mittelwerte von", n,
    "Beobachtungen"))
qqnorm(sim.mean)
n <- 200
# X_1,...,X_n simulieren und in einer n-spaltigen
# Matrix (mit 1000 Zeilen) anordnen
sim <- matrix(sample(werte, n * 1000, replace = TRUE),
    ncol = n)
# In jeder Matrixzeile Mittelwert berechnen
sim.mean <- apply(sim, 1, "mean")
hist(sim.mean, main = paste("Mittelwerte von", n,
    "Beobachtungen"))
qqnorm(sim.mean)
```



d) (zu **Python**)

```
n <- 200
# X_1,...,X_n simulieren und in einer n-spaltigen
# Matrix (mit 1000 Zeilen) anordnen
sim <- matrix(sample(werte, n * 1000, replace = TRUE),
    ncol = n)
sim.mean <- apply(sim, 1, "mean")
mean(sim.mean)

## [1] 7.01895

sd(sim.mean)

## [1] 0.3606576
```

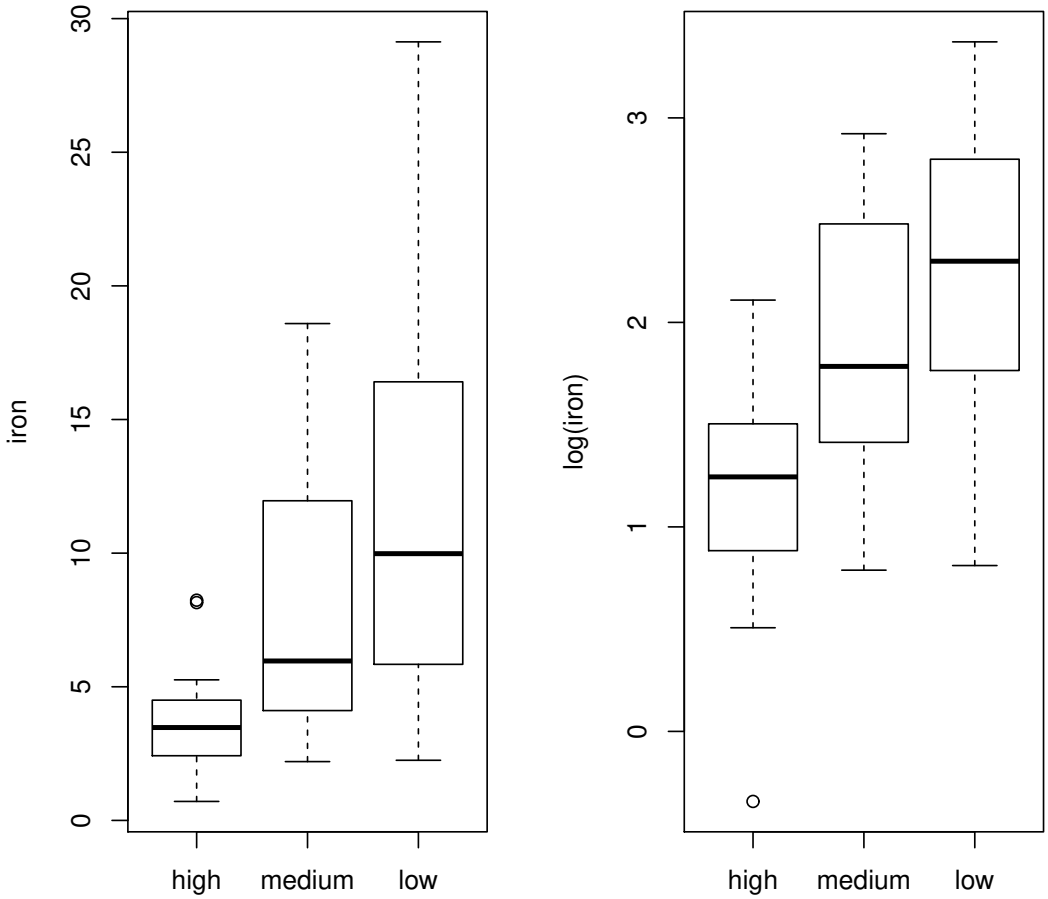
Aufgabe 5.3

a) (zu **Python**)

Siehe b) links

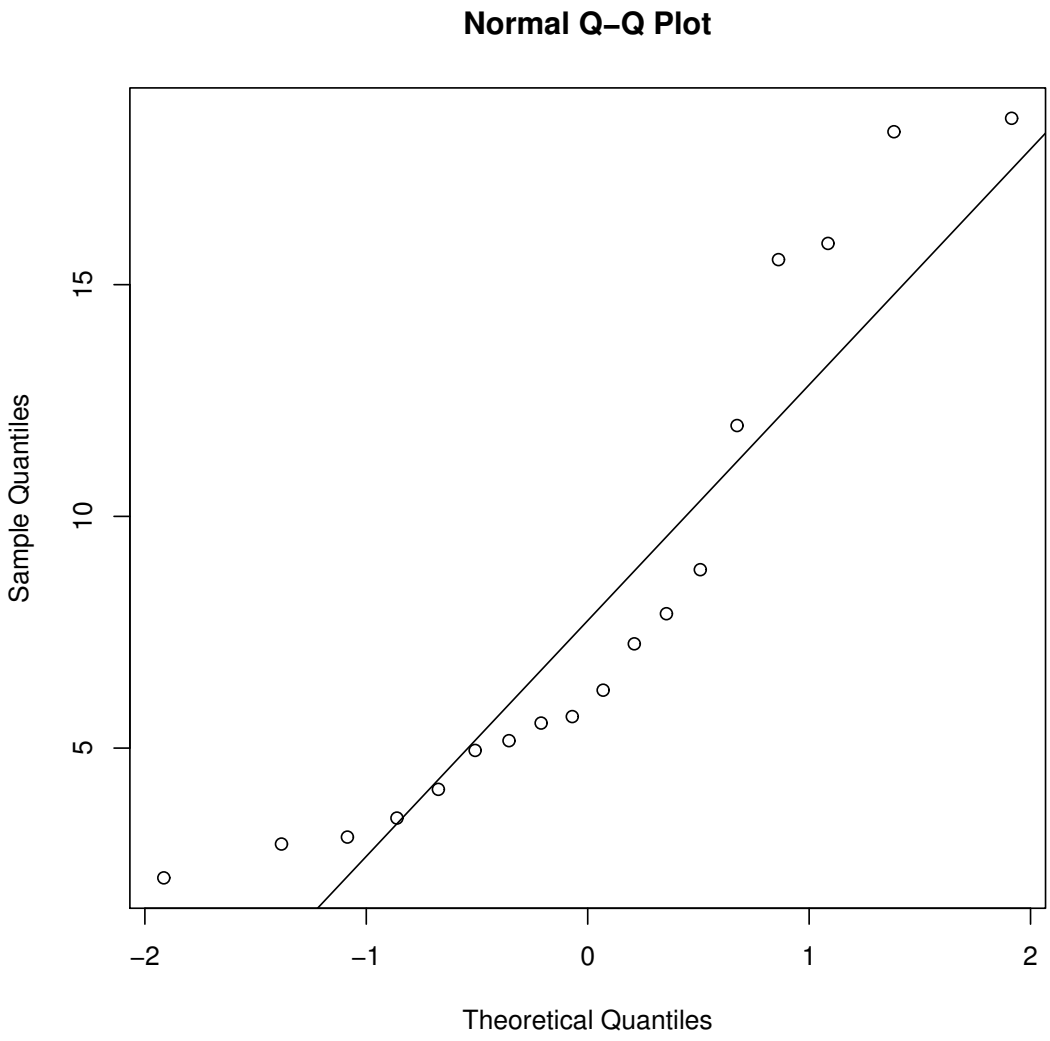
b) (zu **Python**)

```
iron <- read.table("./Daten/ironF3.dat", header = TRUE)
par(mfrow = c(1, 2))
boxplot(iron, ylab = "iron")
boxplot(log(iron), ylab = "log(iron)")
```



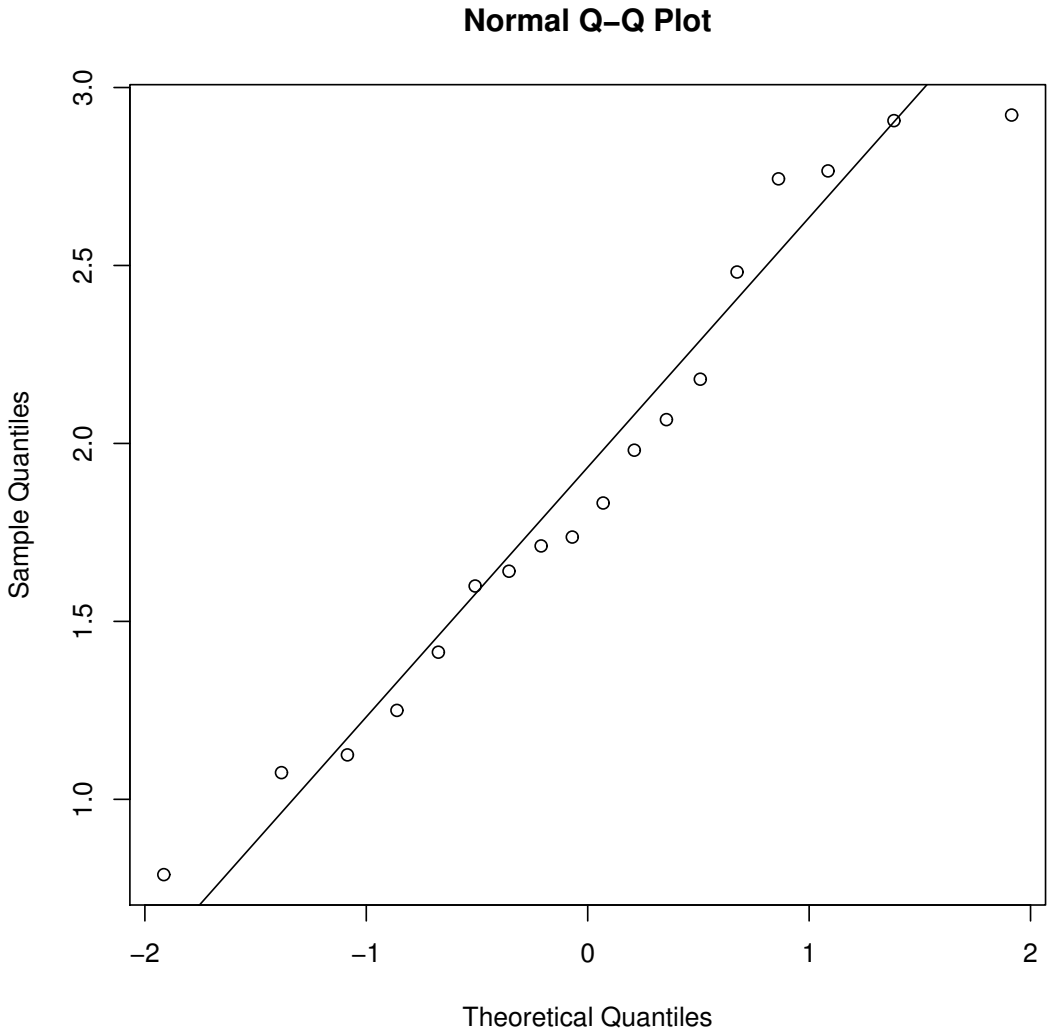
c) (zu **Python**)

```
qqnorm(iron[, "medium"])
qqline(iron[, "medium"])
```



(zu **Python**)

```
qqnorm(log(iron[, "medium"]))
qqline(log(iron[, "medium"]))
```



d) (zu **Python**)

```
mean(iron[, "medium"])

## [1] 8.203889
```

(zu **Python**)

```
var(iron[, "medium"])

## [1] 29.67401
```

(zu **Python**)

```
1 - pnorm(10, mean = 8.204, sd = sqrt(29.67))

## [1] 0.3708051
```

e) (zu **Python**)

```
mean(log(iron[, "medium"]))

## [1] 1.901225
```

(zu **Python**)

```
var(log(iron[, "medium"]))

## [1] 0.4336376
```

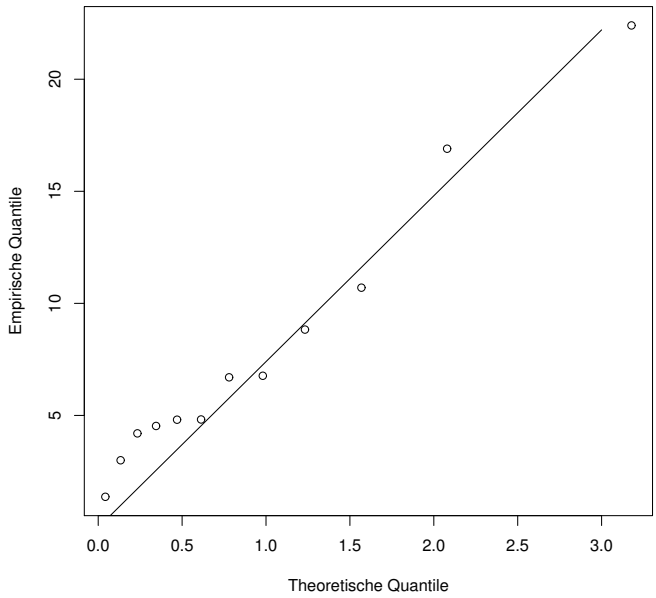
(zu **Python**)

```
1 - pnorm(log(10), mean = 1.901, sd = sqrt(0.4336))

## [1] 0.270976
```

Aufgabe 5.4

(zu **Python**)

```
x <- c(16.9, 4.2, 6.7, 8.83, 10.7, 22.4, 1.37, 3, 4.82,
    4.53, 6.77, 4.81)
alphak <- (seq(1, 12, by = 1) - 0.5)/12
qqplot(-log(1 - alphak), sort(x), xlab = "Theoretische Quantile",
    ylab = "Empirische Quantile")
lines(seq(0, 3, by = 1), 7.4 * seq(0, 3, by = 1), type = "l")
```



(zu **Python**)

```
empir.quantile <- sort(x)
theor.quantile <- -log(1 - alphak)
## alternativ mit theor.quantile <-
## qexp(alphak,rate=1)
lm(empir.quantile ~ 0 + theor.quantile)


##
## Call:
## lm(formula = empir.quantile ~ 0 + theor.quantile)
##
## Coefficients:
## theor.quantile
##          7.421
```

Die Darstellung `0+theor.quantile` im Befehl

```
lm()
```

bedeutet, dass die Regressionsgerade durch den Ursprung gehen soll.

Aufgabe 5.5

a) (zu **Python**)

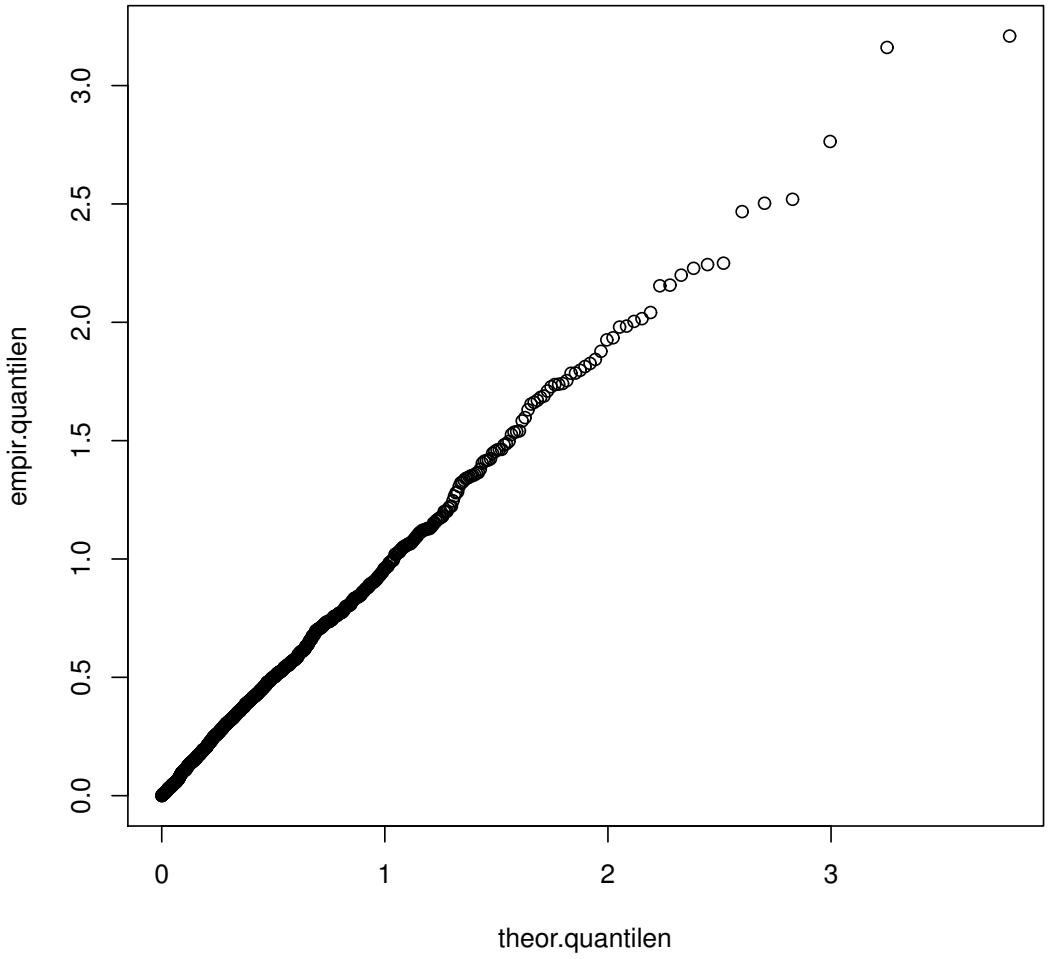
```
# Generiere 1000 gleichmaessig verteilte
# Zufallszahlen im Intervall [0,1]
v <- runif(1000, min = 0, max = 1)
# Generiere 1000 exponentialverteilte Zufalsszahlen
# mit Hilfe der gleichmaessig verteilten
# Zufallszahlen
x <- -log(v)/2
```

b) (zu **Python**)

```
par(mfrow=c(1,2))
v <- runif(1000, min=0, max=1)
x <- -log(v)/2
hist(x)
box()
hist(rexp(1000,rate=2), main="Histogram of x",
     xlab="x")
box()

## Traceback (most recent call last):
##   File "<string>", line 1, in <module>
## NameError: name 'par' is not defined
```

(zu **R**) @

```
# qq-Plot Berechne die theoretischen Quantilen
n <- 1000
theor.quantilen <- qexp((seq(1, n, by = 1) - 0.5)/n,
    rate = 2)
# Berechne die empirischen Quantilen
empir.quantilen <- sort(x)
qqplot(theor.quantilen, empir.quantilen)
```



Aufgabe 5.6

a)

b) (zu **Python**)

```
5/(log(12) + log(4) + log(6.9) + log(27.9) + log(15.4))

## [1] 0.4213821
```

c) (zu **Python**)

```
x <- c(12, 4, 6.9, 27.9, 15.4)
mean(x)/(mean(x) - 1)
```

```
## [1] 1.081699
```