serie_05

March 31, 2018

1 Series 5

1.1 Aufgabe 5.1

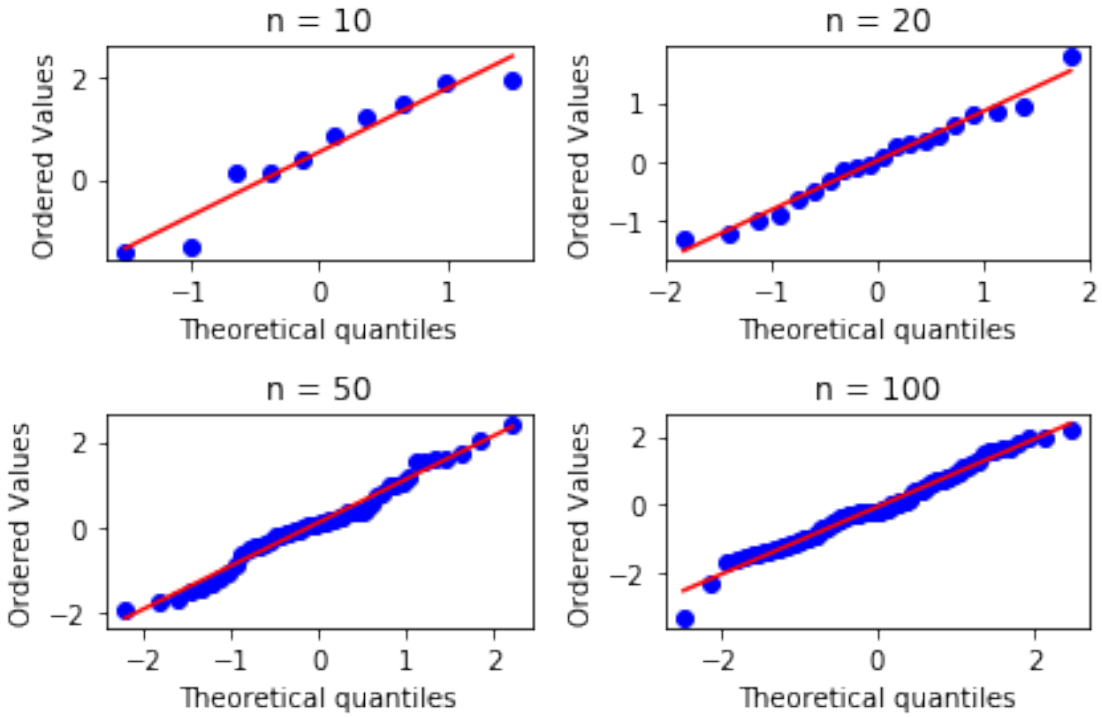
Normalverteile, t-verteilte, chiquadrat-verteile Zufallszahlen, Freiheitsgrade, QQ-Plot

---

a.) Normalverteilte Zufallszahlen simulieren und mit Normalplot betrachten. n wird als Parameter (size) der Methode rvs übergeben. n ist die Anzahl standard-normalverteilte Zufallszahlen. Wenn man die Simulation wiederholt, ändert sich die Streuung jedoch immer um die Winkelhalbierende des QQ-Plots. Je mehr Zufallszahlen generiert werden, desto stärker normalverteilt wird der QQ-Plot. (Bei einer uniformen Verteilung bewirken mehr Zufallszahlen, dass die Gerade immer stärker uniform verteilt wird. Die Summe aller Zufallszahlen wird immer stärker normalverteilt, je mehr Zufallszahlen dazukommen)

```
In [22]: import matplotlib.pyplot as plt
         import scipy.stats as st

         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()
```

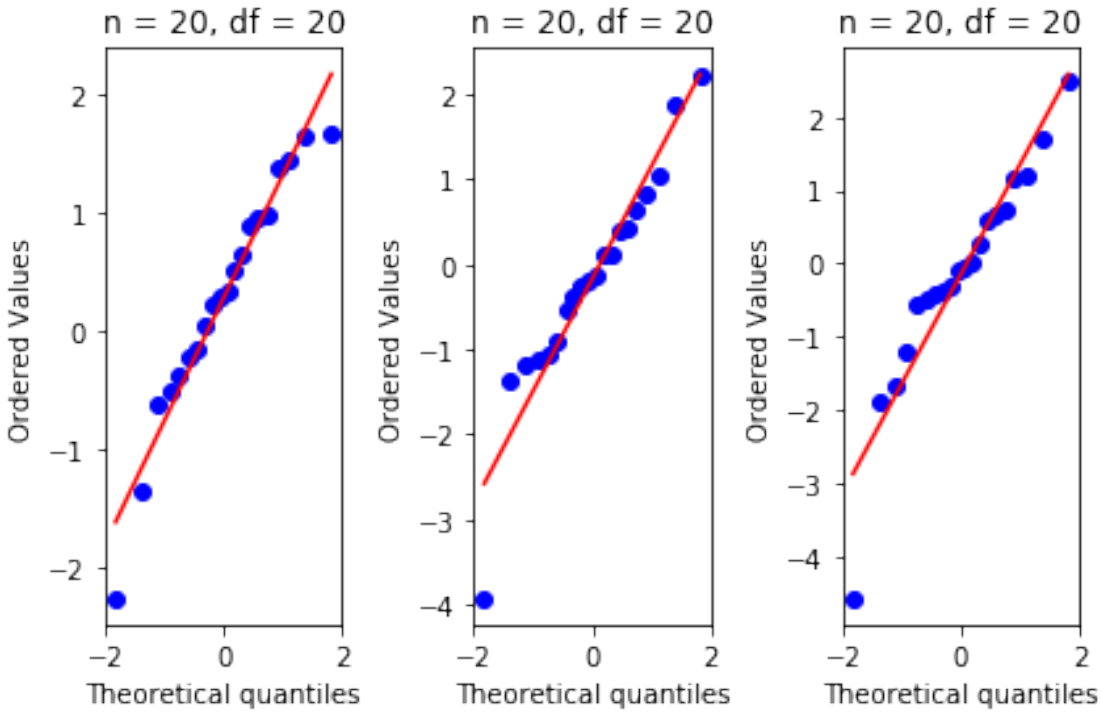


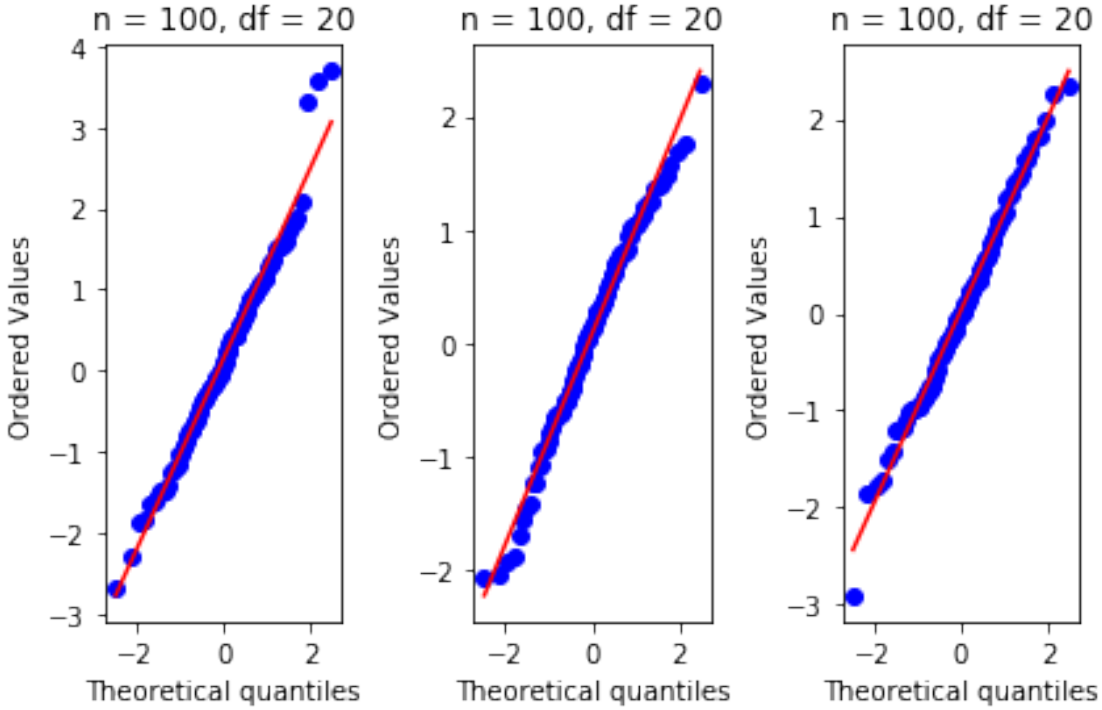
---

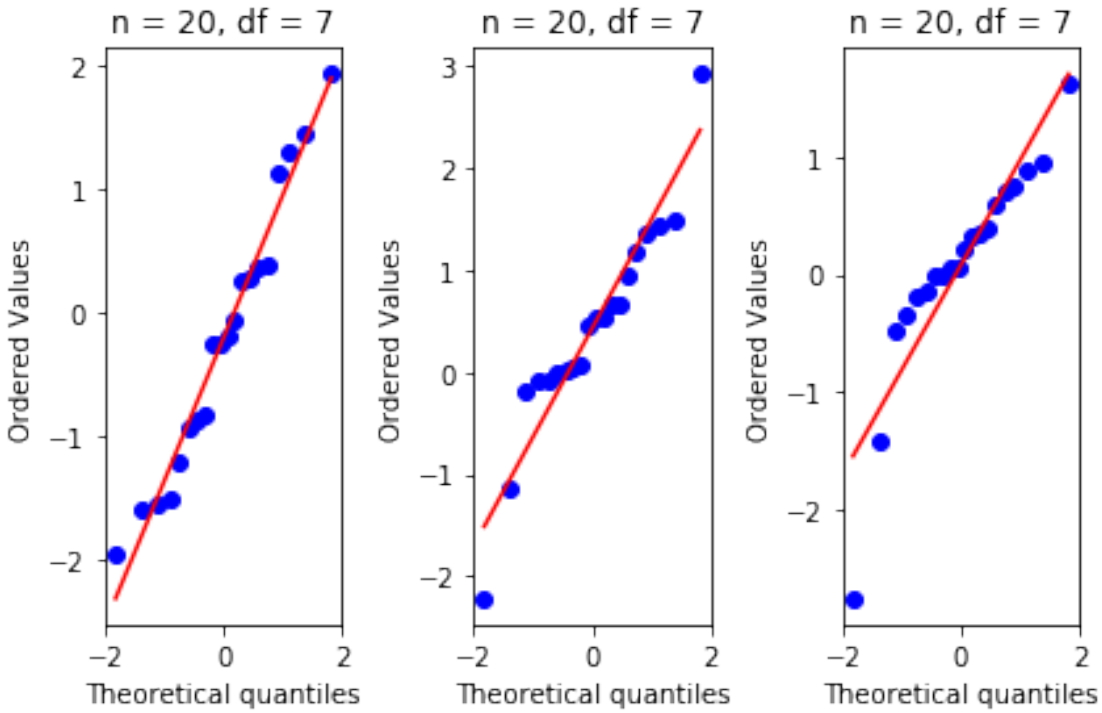
b.) Langschwänzige Verteilung: t-verteilte Zufallszahlen simulieren mit Freiheitsgraden. Die t-Verteilung näher sich der standard-Normalverteilung je mehr Zufallszahlen und je mehr Freiheitsgrade existieren. Das heisst die t-Verteilung mit 100 Zufallszahlen und 20 Freiheitsgraden nähert sich am meisten einer standard-Normalverteilung.

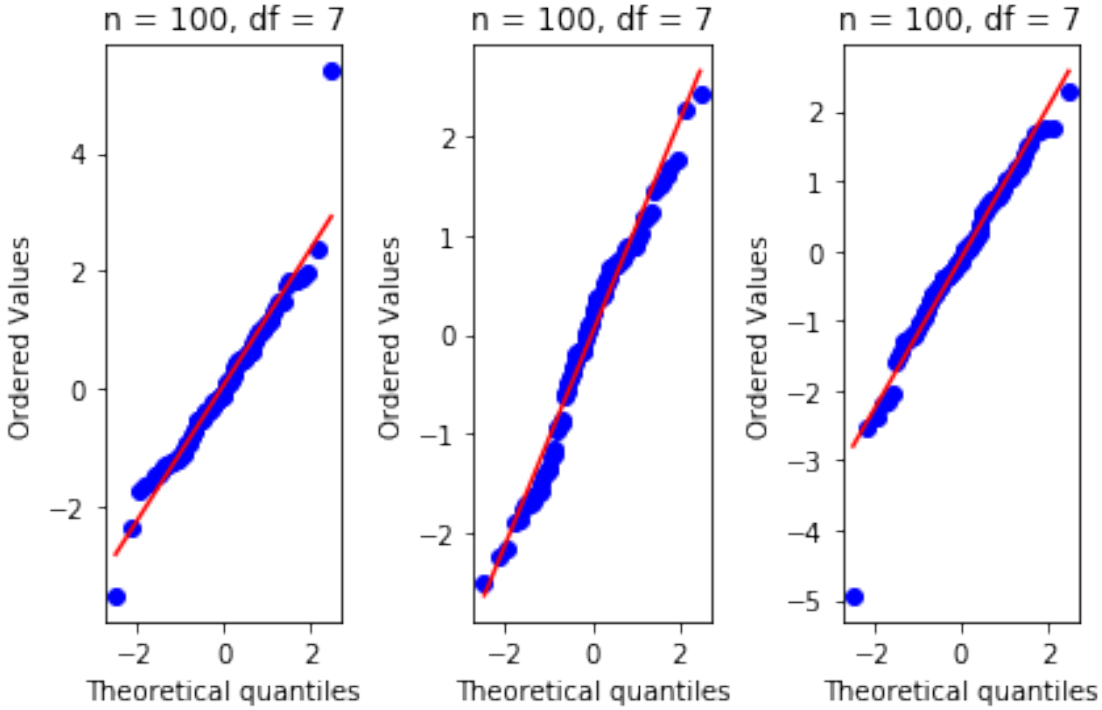
```
In [23]: import matplotlib.pyplot as plt
         import scipy.stats as st

         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):
```

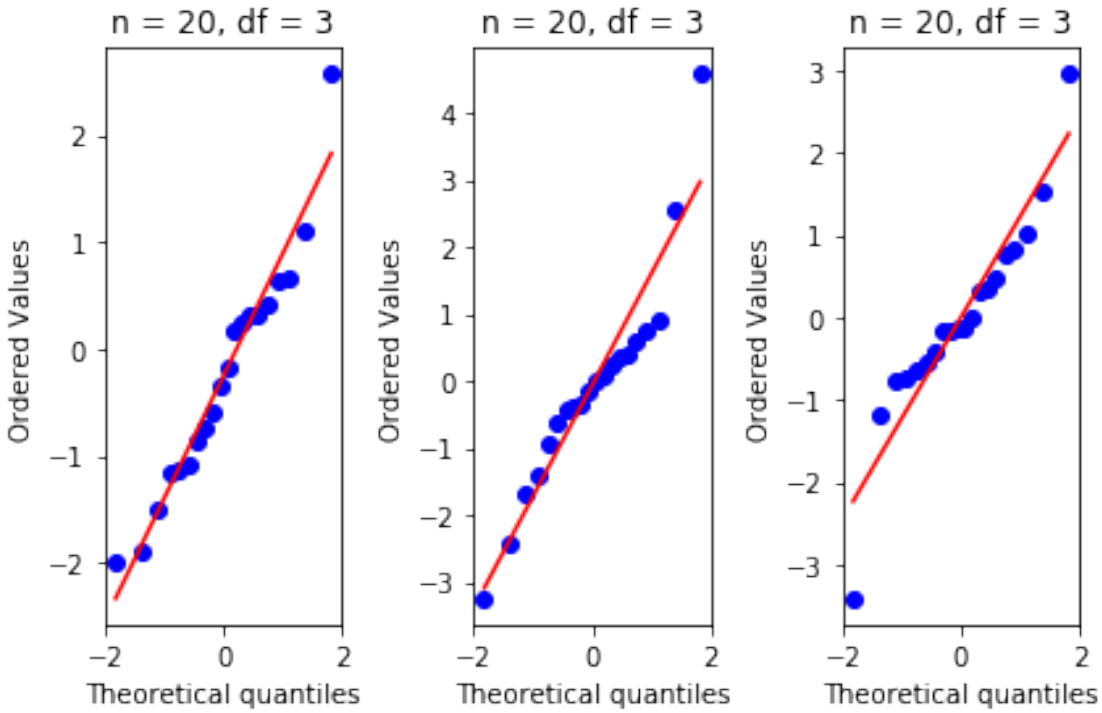
```
    plt.subplot(1, 3, i)
    x = st.t.rvs(size=100, df=20)
    st.probplot(x, plot=plt)
    plt.title("n = 100, df = 20")

plt.tight_layout()
plt.show()

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()

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

plt.tight_layout()
plt.show()

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

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

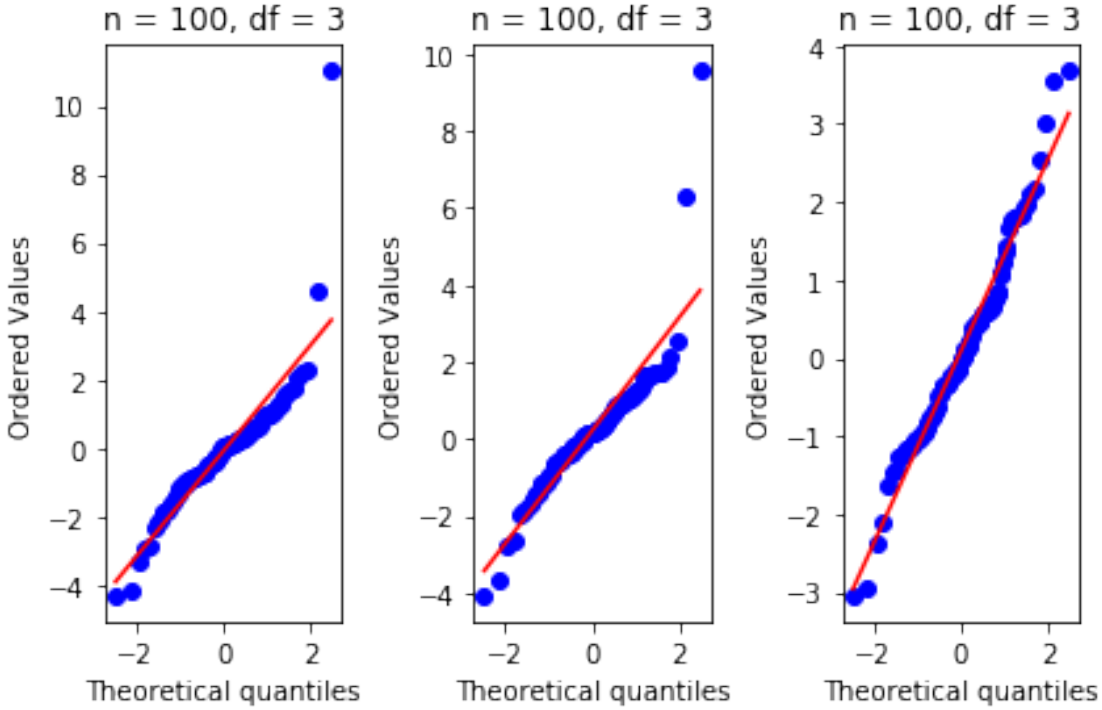

n = 20, df = 20
n = 20, df = 20
n = 20, df = 20
Ordered Values
Theoretical quantiles


n = 100, df = 20
n = 100, df = 20
n = 100, df = 20
Ordered Values
Theoretical quantiles


n = 20, df = 7
n = 20, df = 7
n = 20, df = 7
Ordered Values
Theoretical quantiles


n = 100, df = 7
n = 100, df = 7
n = 100, df = 7
Ordered Values
Theoretical quantiles

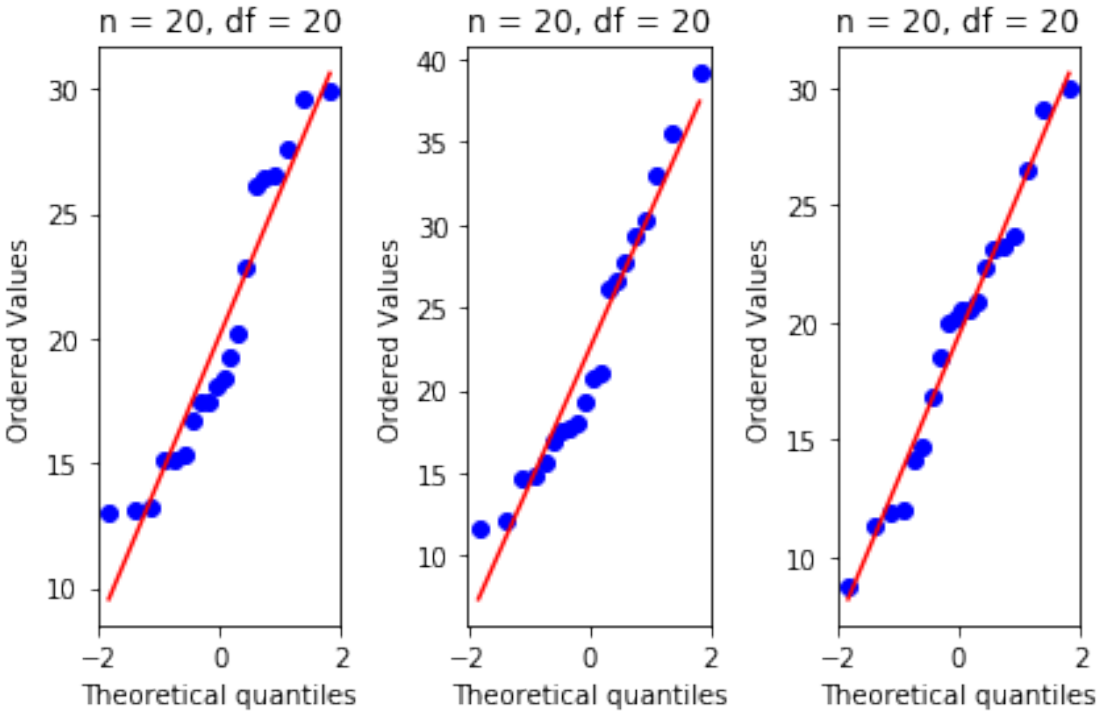

n = 20, df = 3
n = 20, df = 3
n = 20, df = 3
Ordered Values
Theoretical quantiles

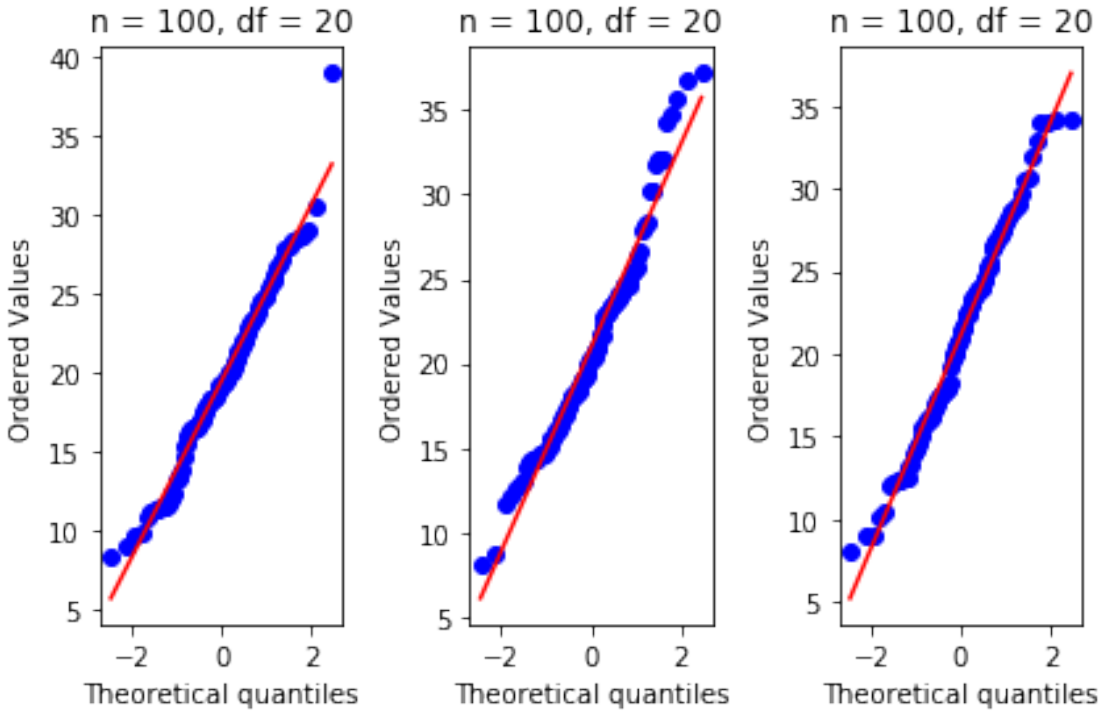

n = 100, df = 3
n = 100, df = 3
n = 100, df = 3
Ordered Values
Theoretical quantiles

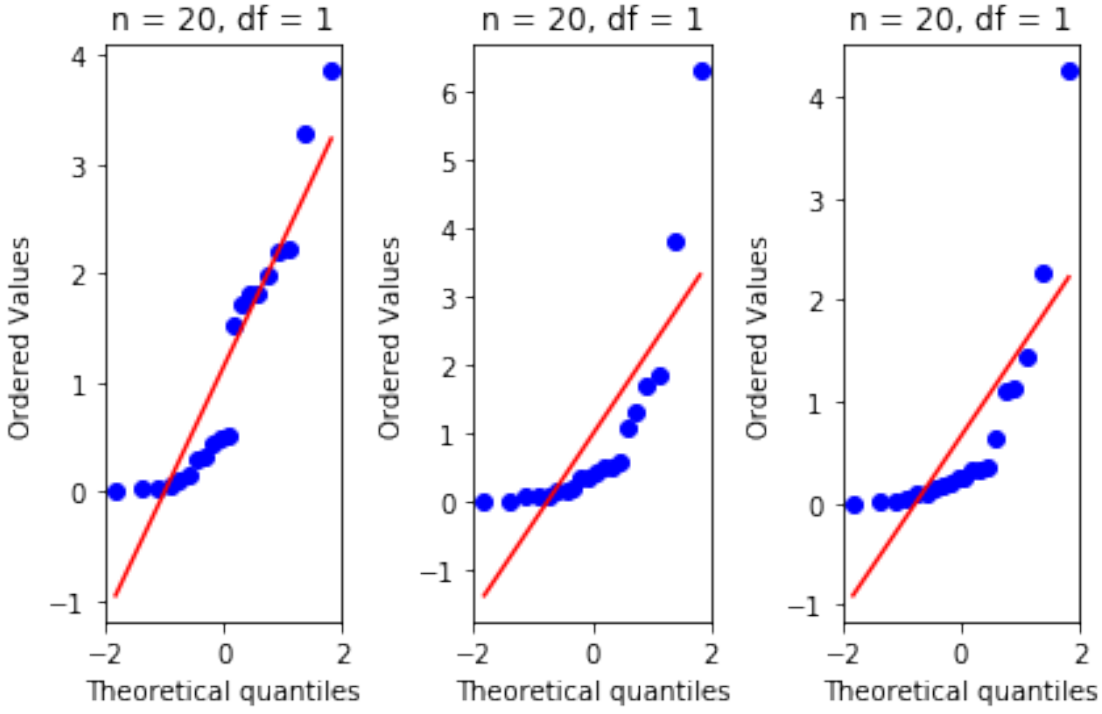
c.) Schiefe Verteilung: chiquadrat-verteilte Zufallszahlen mit Freiheitsgraden. Je mehr Freiheitsgrade die chiquadrat-Verteilung besitzt desto mehr formt sie sich zu einem Normalplot. Hier spielt die Anzahl Zufallsvariablen weniger bis gar keine Rolle im Gegensatz zu einer t-Verteilung.

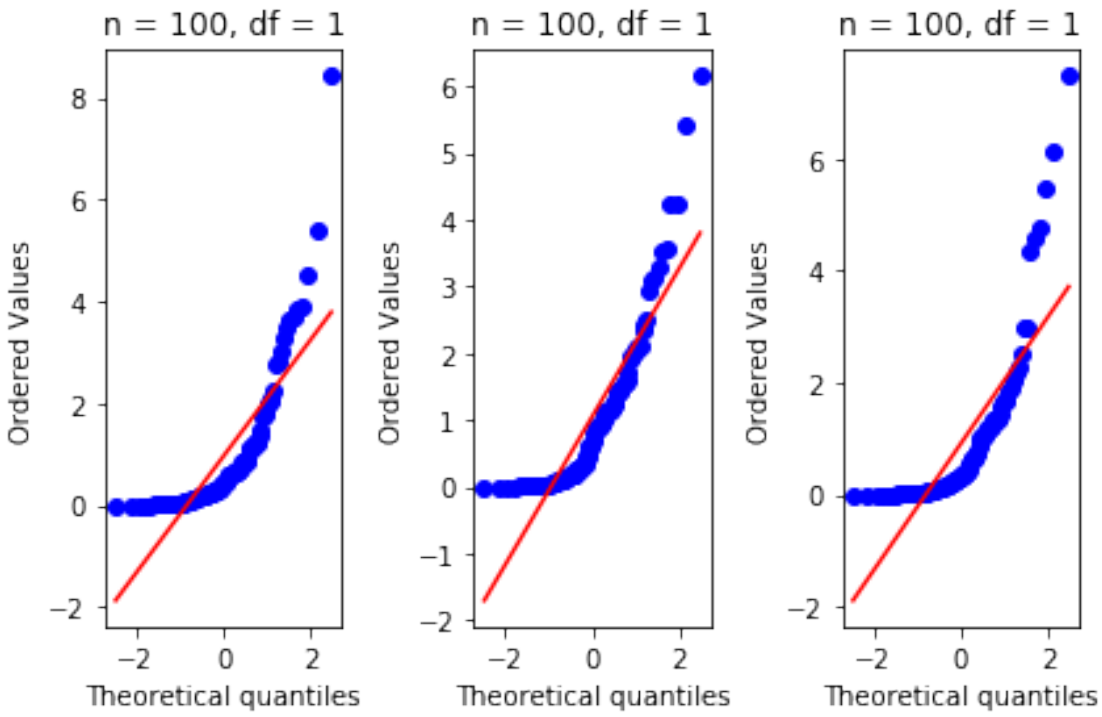
```
In [6]: import matplotlib.pyplot as plt
        import scipy.stats as st

        for i in range(1, 4):
            plt.subplot(1, 3, i)
            x = st.chi2.rvs(size= 20, df=20)
            st.probplot(x, plot=plt)
            plt.title("n = 20, df = 20")

        plt.tight_layout()
        plt.show()

        import matplotlib.pyplot as plt
        import scipy.stats as st

        for i in range(1, 4):
            plt.subplot(1, 3, i)
            x = st.chi2.rvs(size= 100, df=20)
            st.probplot(x, plot=plt)
            plt.title("n = 100, df = 20")

        plt.tight_layout()
        plt.show()

        import matplotlib.pyplot as plt
        import scipy.stats as st

        for i in range(1, 4):
            plt.subplot(1, 3, i)
            x = st.chi2.rvs(size= 20, df=1)
            st.probplot(x, plot=plt)
            plt.title("n = 20, df = 1")

        plt.tight_layout()
        plt.show()

        import matplotlib.pyplot as plt
        import scipy.stats as st

        for i in range(1, 4):
            plt.subplot(1, 3, i)
            x = st.chi2.rvs(size= 100, df=1)
```

```
    st.probplot(x, plot=plt)
    plt.title("n = 100, df = 1")

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




n = 100, df = 20
n = 100, df = 20
n = 100, df = 20
Ordered Values
Theoretical quantiles


n = 20, df = 1
n = 20, df = 1
n = 20, df = 1
Ordered Values
Theoretical quantiles



1.2 Aufgabe 5.2

Zentraler Grenzwertsatz, Histogramm, Normalplot

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. Das heisst, dass jede Zahl gleichwahrscheinlich angenommen werden kann.

Wir simulieren nun die Verteilung von X sowie die Verteilung des Mittelwerts $\overline{X}_n$ von mehreren X.

---

a.) Die Verteilung von X mittels eines Histogramms darstellen

1. mögliche Werte von X definieren
2. X simulieren mit Series(np.random.choice(werte, size, replace=True)
3. Subplot für Histogramm
4. Histogramm erstellen
5. Subplot für Normalplot
6. Normalplot erstellen

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

```
from pandas import Series, DataFrame
import scipy.stats as st

# 1. mögliche Werte von X definieren
werte = np.array([0, 10, 11])

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

# 3. Subplot für Histogramm
plt.subplot(1, 2, 1)

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

# 5. Subplot für Normalplot
plt.subplot(1, 2, 2)

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

```
<matplotlib.figure.Figure at 0x106fa0c50>
```

---

b.) Wir simulieren 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.

1. X_1, ..., X_n simulieren und in einer n-spaltigen Matrix (mit 1000 Zeilen) anordnen
2. In jeder Matrixzeile Mittelwert berechnen
3. Histogramm erstellen
4. QQ-Normalplot erstellen

Je mehr Beobachtungen wir durchführen, sprich je höher n ist, desto mehr wird das Histogramm und der QQ-Plot normalverteilt. Für das Verständnis:

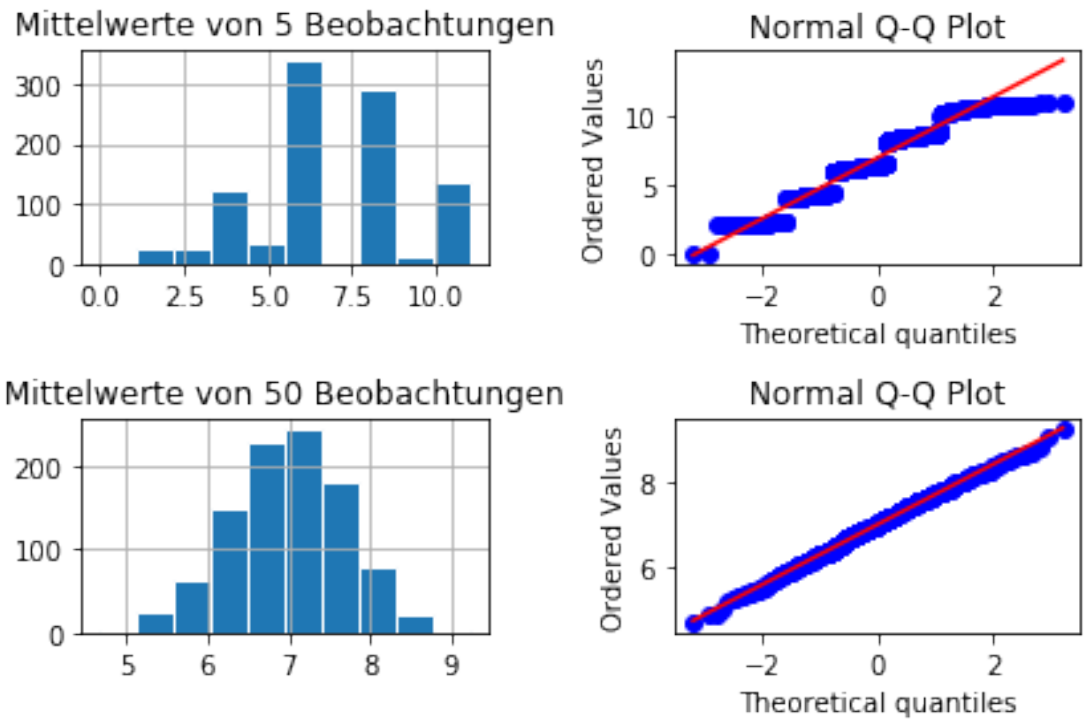
- Es kann die Zahlen 0, 10 oder 11 annehmen.
- Bei einem ersten Vorgang mit 1000 Realisierungen gab es zufällig 200 mal 0, 500 mal 10 und 300 mal 11.
- Der Mittelwert dieser Zahlen ist also $\frac{200*0+500*10+300*11}{1000} = 8.3$
- Bei einem weiteren Vorgang mit 1000 weiteren Realisierungen gab es zufällig 300 mal 0, 100 mal 10 und 600 mal 11.
- Dies gibt wieder einen neuen Wert (7.6)

- Die Hauptaussage ist, je mehr Wiederholungen dieses Vorgangs man vornimmt, desto normalverteilter wird die Streuung dieser Mittelwerte

  5 Vorgänge: $\frac{\frac{200*0+500*10+300*11}{1000}+\frac{200*0+700*10+100*11}{1000}+\frac{800*0+100*10+100*11}{1000}+\frac{500*0+500*10+0*11}{1000}+\frac{400*0+100*10+500*11}{1000}}{5} =$ 6

In [43]:
```
# Anzahl Beobachtungen definieren
n = 5

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

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

# 3. Histogramm erstellen
plt.subplot(2,2,1)
sim_mean.hist(edgecolor="white")
plt.title("Mittelwerte von 5 Beobachtungen")

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

# Das Ganze nochmals mit 50 Beobachtungen
n = 50
sim = Series(np.random.choice(werte, size=n*1000, replace=True))
sim = DataFrame(np.reshape(sim, (n, 1000)))
sim_mean = sim.mean()

plt.subplot(2,2,3)
sim_mean.hist(edgecolor="white")
plt.title("Mittelwerte von 50 Beobachtungen")

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

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

```
/Users/Christopher/anaconda3/lib/python3.6/site-packages/numpy/core/fromnumeric.py:52: FutureWar
  return getattr(obj, method)(*args, **kwds)
```



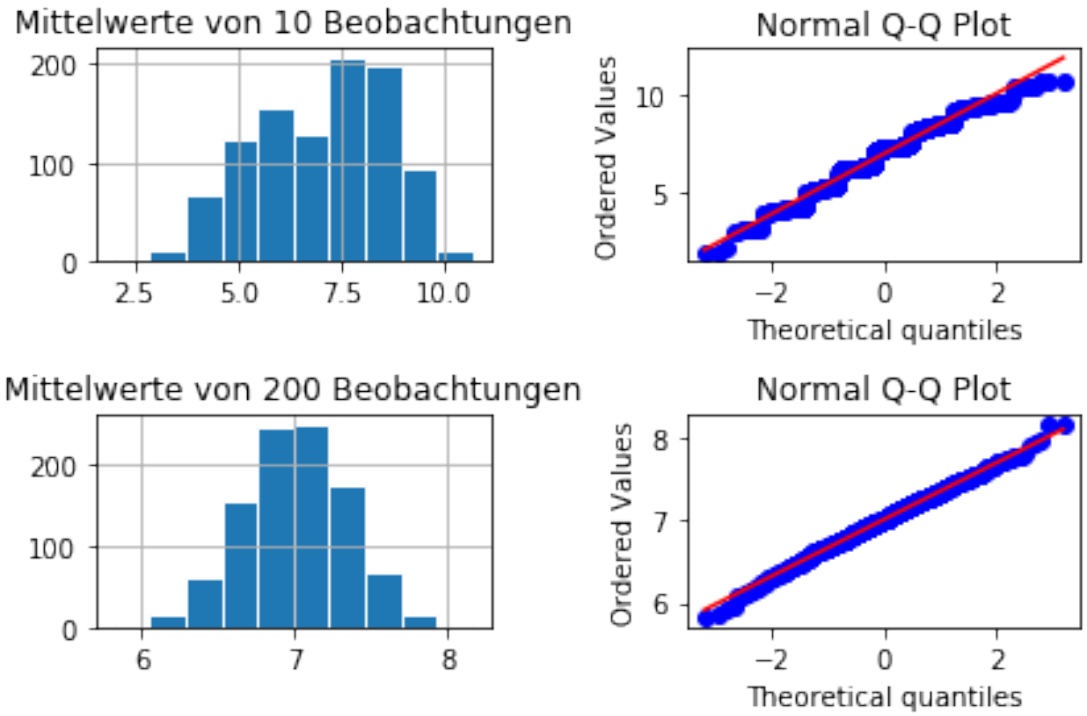
---

c.) Als nächstes simulieren wir 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.** Wie oben bereits erläutert nimmt die Verteilung immer mehr die Streuung der Normalverteilung an je mehr Beobachtungen durchgeführt werden.

```
In [48]: n = 10
         sim = Series(np.random.choice(werte, size=n*1000, replace=True))
         sim = DataFrame(np.reshape(sim,(n,1000)))
         sim_mean = sim.mean()

         plt.subplot(2,2,1)
         sim_mean.hist(edgecolor="white")
         plt.title("Mittelwerte von 10 Beobachtungen")

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

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

```
plt.subplot(2,2,3)
sim_mean.hist(edgecolor="white")
plt.title("Mittelwerte von 200 Beobachtungen")

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

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

```
/Users/Christopher/anaconda3/lib/python3.6/site-packages/numpy/core/fromnumeric.py:52: FutureWar
  return getattr(obj, method)(*args, **kwds)
```



Die obenstehenden 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 normalverteilt sind. An der x-Achse sieht man auch, dass die Varianz immer kleiner wird.

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

$\text{Var}[\overline{X}_n] = \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}(U_i) = \frac{\text{Var}(U_i)}{n} = \frac{1}{n}((0-7)^2 * \frac{1}{3} + (10-7)^2 * \frac{1}{3} + (11-7)^2 * \frac{1}{3}) = \frac{24.67}{n}$

Somit ist die Standardabweichung von $\overline{X}_n$, also der Standardfehler, gegeben durch: > $\sigma_{\overline{X}_n} = \sqrt{\frac{24.67}{n}}$

```
In [52]: import numpy as np
         from pandas import Series, DataFrame

         werte = np.array([0,10,11])
         n = 200
         sim = Series(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()
```

```
/Users/Christopher/anaconda3/lib/python3.6/site-packages/numpy/core/fromnumeric.py:52: FutureWar
  return getattr(obj, method)(*args, **kwds)
```

```
Out[52]: 0.36519431521159124
```

Experiment und Berechnung sind also in guter Uebereinstimmung. $\overline{X}_n$ folgt also der Verteilung $\mathcal{N}(7, 0.12)$.

1.3 Aufgabe 5.3

Boxplot, Transformation, μ, σ^2

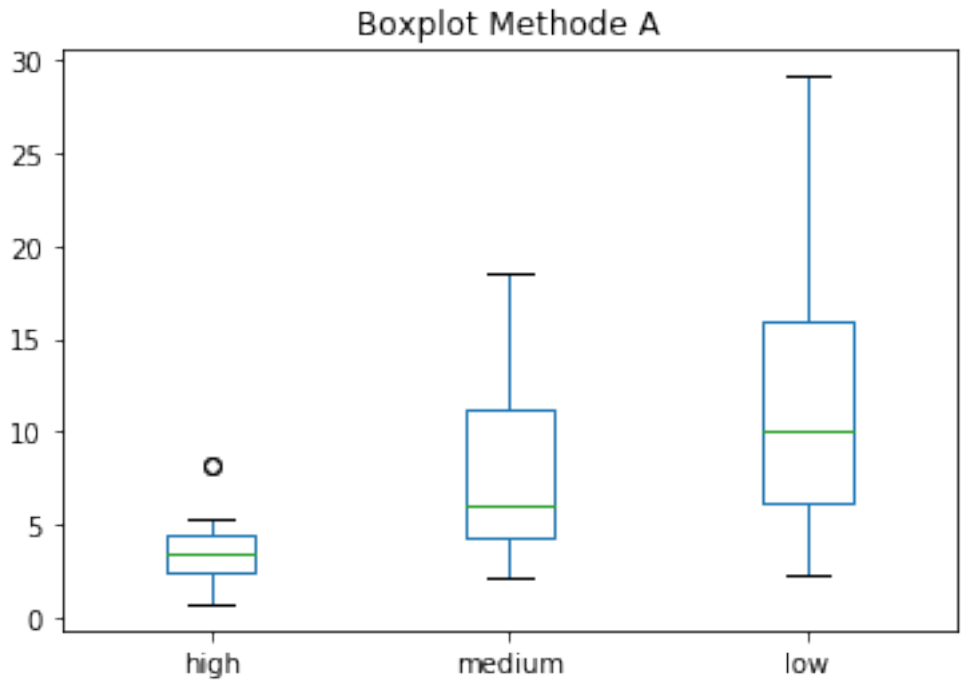
---

a.) Boxplot für jede Versuchtsbedingung erstellen. Die grösste normale Beobachtung ist die grösste Beobachtung, die höchstens 1.5 * Quantilsdifferenz vom oberen Quantil entfernt ist. Bei der hohen Dosis ist diese nicht weit vom oberen Quartil entfernt, bei der Medium-Dosis ein wenig weiter weg und bei der tiefen Dosis ist die grösste normale Beobachtung sehr weit vom oberen Quartil entfernt. Die hohe Dosis-Gruppe hat einen Ausreisser nach oben.

- Die hohe Dosis-Gruppe hat eher symmetrisch-verteilte Messwerte (normalverteilt)
- Die mittlere und tiefe Dosis-Gruppe hat eher rechtsschiefe Messwerte
- Je kleiner die Dosis, desto grösser ist die Streuung

```
In [59]: import pandas as pd

         # Daten einlesen
         iron = pd.read_table("ironF3.dat", sep=" ", index_col=False)
         # Boxplot erstellen
         iron.plot(kind="box", title="Boxplot Methode A")
```

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x1a18151fd0>



In [69]:
```
import matplotlib.pyplot as plt
import numpy as np
from pandas import DataFrame

# iron Daten beschreiben (optional als Übersicht)
DataFrame.describe(iron)
```

Out[69]:

	high	medium	low
count	18.000000	18.000000	18.00000
mean	3.698889	8.203889	11.75000
std	2.030870	5.447386	7.02815
min	0.710000	2.200000	2.25000
25%	2.420000	4.320000	6.10250
50%	3.475000	5.965000	9.98000
75%	4.472500	11.182500	15.99750
max	8.240000	18.590000	29.13000

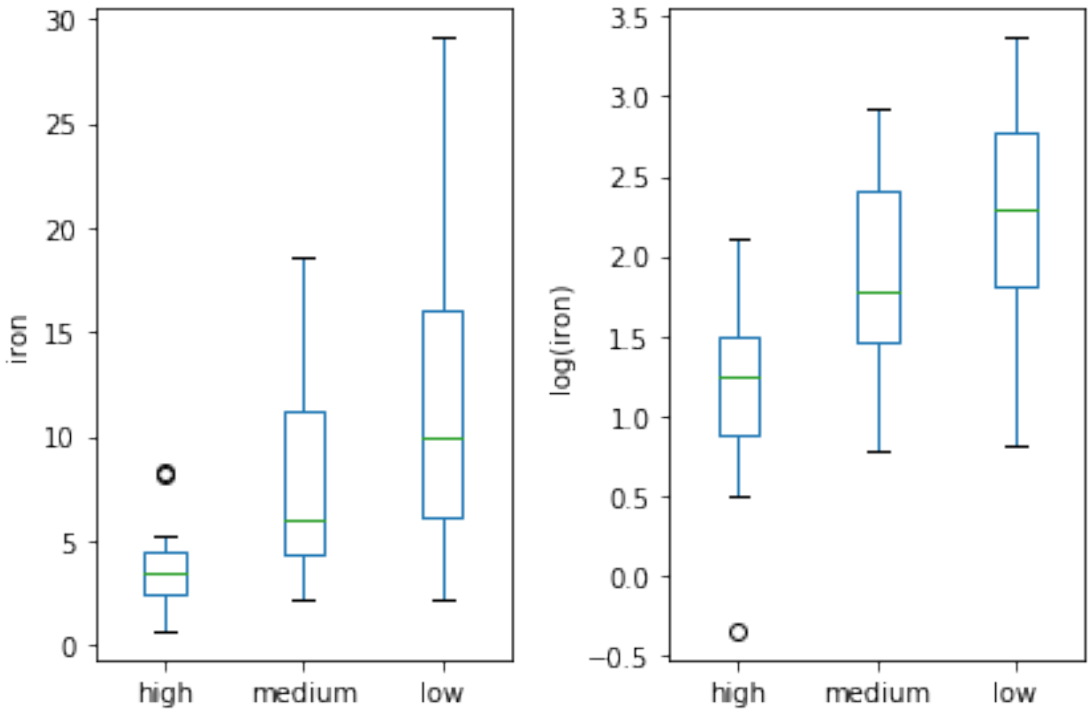
---

b.) Logarithmus-Transformation.

- 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.

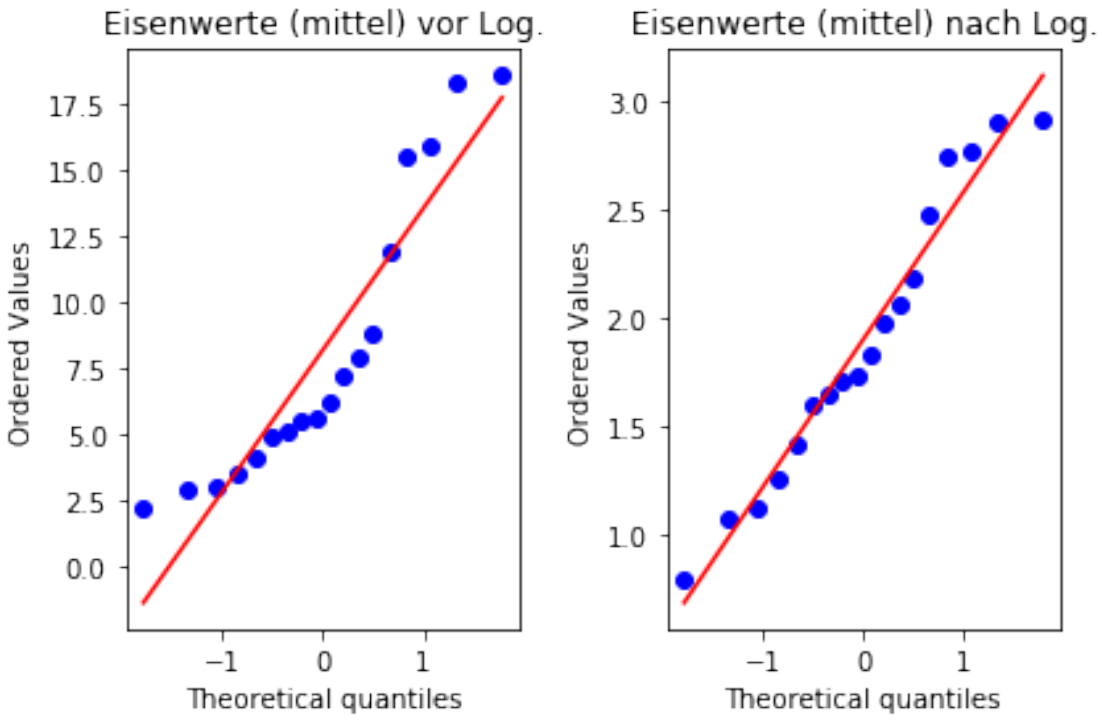
```
In [61]: import matplotlib.pyplot as plt

         # 1. Subplot erstellen (1 Reihe, 2 Zeilen, 1. Plot)
         plt.subplot(1, 2, 1)

         # iron Daten als Boxplot plotten
         iron.plot(kind="box", ax=plt.gca())
         plt.ylabel("iron")

         # 2. Subplot erstellen
         plt.subplot(1, 2, 2)

         # log(iron) Daten als Boxplot plotten
         np.log(iron).plot(kind="box", ax=plt.gca())
         plt.ylabel("log(iron)")

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



---

c.) Normalverteilung vor und nach Logarithmieren.

```
In [76]: import matplotlib.pyplot as plt
         import numpy as np
         import pandas as pd
         import scipy.stats as st

         # Plot vor dem Logarithmieren
         plt.subplot(1, 2, 1)
         st.probplot(iron["medium"], plot=plt)
         plt.title("Eisenwerte (mittel) vor Log.")

         # Plot nach dem Logarithmieren
         plt.subplot(1, 2, 2)
         st.probplot(np.log(iron["medium"]), plot=plt)
         plt.title("Eisenwerte (mittel) nach Log.")

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



---

d.) Parameter μ (Erwartungswert) und σ^2 (empirische Varianz) schätzen. Wie gross ist Wahrscheinlichkeit, dass 50% Eisen zurückgehalten wird?

- Erwartungswert durch empirischen Mittelwert der Daten bei mittlerer Dosierung berechnen

- Varianz durch `iron["medium"].var()` berechnen
- Beide Werte werden benötigt, um die Wahrscheinlichkeit zu berechnen

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 0.370805119544

```
In [77]: # Erwartungswert berechnen
         iron["medium"].mean()

Out[77]: 8.203888888888889

In [78]: # empirische Varianz berechnen
         iron["medium"].var()

Out[78]: 29.67401339869281

In [79]: import scipy.stats as st

         # Wahrscheinlichkeit mit der norm.cdf Methode
         1 - st.norm.cdf(x=10, loc=8.204, scale=np.sqrt(29.67))

Out[79]: 0.37080511954367934
```

1.4 Aufgabe 5.4

Poissonprozess, Momentenmethode, QQ-Plot, empirische/theoretische Quantile, Exponentialverteilung, Regressionsgerade

---

a.) Momentenmethode verwenden, um Parameter zu schätzen.

- Angler fängt in 2 Stunden 15 Fische
- Poissonprozess
- Mit welcher Wahrscheinlichkeit dauert es länger als 12 Minuten bis nächster Fisch anbeisst

Theorie Ablauf der Momentenmethode

1. Daten $x_1, x_2, ..., x_n$ als Realisierungen von Zufallsvariablen $X_1, X_2, ..., X_n$ auffassen (mit bekannter Verteilung)
2. Erwartungswert $E(X)$ berechnen
3. Gleichung nach unbekanntem Parameter auflösen
4. Wahrscheinlichkeit berechnen

Von der Poissonverteilung wissen wir folgendes:

$$E(X) = \lambda, Var(X) = \lambda, \text{d.h. } \mu = \lambda, \sigma^2 = \lambda$$

Also haben wir hier:

$P[X \leq x] \approx \phi(\frac{x-\lambda}{\sqrt{\lambda}})$

Lösung

- Als Wahrscheinlichkeitsverteilung der Wartezeit T (in Minuten) $\rightarrow$ Exponentialverteilung
- D.h. Wahrscheinlichkeitsdichte ist: $f(t) = \lambda \cdot e^{-\lambda t}$ für $t > 0$
- Wir verwenden jedoch die kummulative Verteilungsfunktion der Exponentialverteilung $F(t) = 1 - e^{\frac{-t}{8}}$ (1 - Stammfunktion)
- Parameter λ ermitteln (Momentenmethode)

1. Daten auffassen (sind bereits gegeben, 2 h - 15 Fische)
2. Erwartungswert berechnen:

   $E(T) = \frac{1}{\lambda}$ (Momentenmethode, beobachteten Wert für mittlere verstrichene Zeit gleich dem Erwartungswert setzen). Erwartungswert $= \frac{120 Minuten}{15 Fische} = \frac{1}{\lambda} = 8$
3. Gleichung nach unbekanntem Parameter auflösen:

   $\frac{1}{\lambda} = 8 \Leftrightarrow \hat{\lambda} = \frac{1}{8}$
4. Wahrscheinlichkeit berechnen:

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

---

b.) Wahrscheinlichkeit, dass genau 2 Fische in 12 Minuten anbeissen.

- X ist Anzahl Fische, die in nächsten 12 Minuten anbeissen
- X ist poissonverteilt mit $\lambda = 1.5$ (in 12 Minuten beissen durchschnittlich 1.5 Fische an)

$P(X = 12) = e^{-1.5} \cdot \frac{1.5^2}{2!} = 0.251$

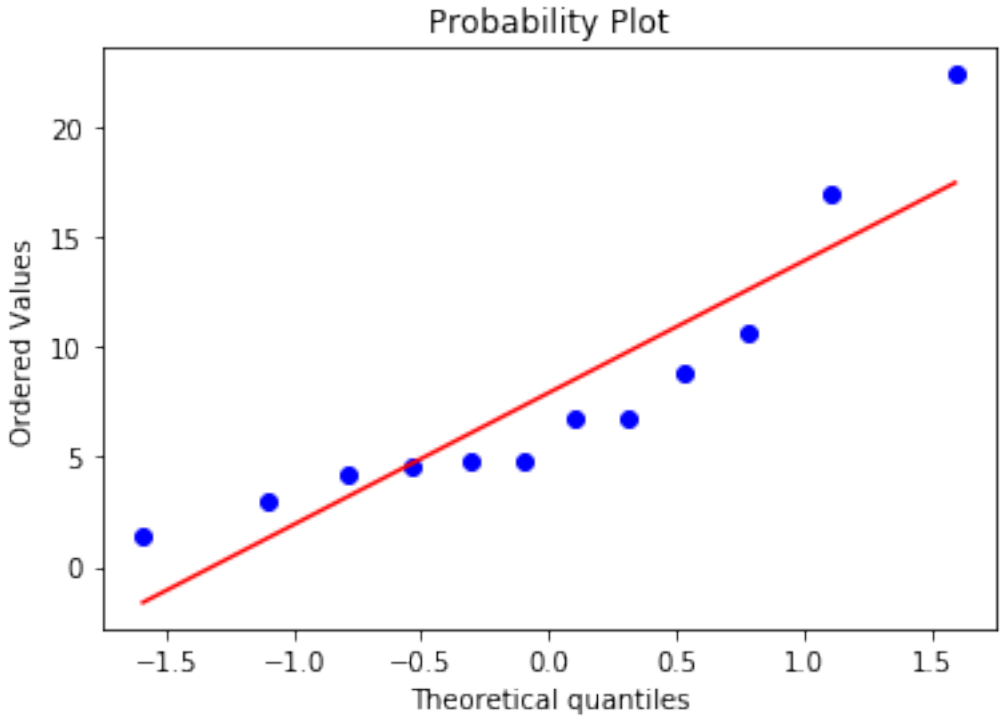
c.) Wartezeiten zwischen Fischfängen. QQ-Plot. Steigung der Regressionsgeraden.
Für die Exponentialverteilung haben wir:

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

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

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

```
In [5]: from pandas import Series
        import scipy.stats as st
        import matplotlib.pyplot as plt

        # 1. Messwerte eintragen und sortieren
        messungen = Series([16.9, 4.2, 6.7, 8.83, 10.7, 22.4, 1.37, 3, 4.82, 4.53, 6.77, 4.81])

        x = st.probplot(messungen, plot=plt)
        plt.show()
```



1.5 Aufgabe 5.5

Stetige Verteilung mit gegebener Dichte.

a. + b.) Likelihood- und Log-Likelihood-Funktion bestimmen. Integral muss 1 geben, damit es eine gültige Wahrscheinlichkeitskurve ist.

1. Integral berechnen
2. Gibt Integral 1, dann ist es eine gültige Wahrscheinlichkeitskurve
3. Alle x-Werte auf Funktion anwenden
4. $L(\alpha) = f(x_1;\alpha) * f(x_2;\alpha) * * * f(x_5;\alpha) = \frac{\alpha}{x_1^{\alpha+1}} + ... + = \sum_{i=1}^{5} \frac{\alpha}{x_i^{\alpha+1}}$
5. $L(\alpha) = log(L(\alpha)) => \frac{5}{log(12)+log(4)+...+log(15.4)}$

c.) Momentenschätzer für α bestimmen.

E[X] = Integral(-∞, ∞) von $xf(x, \alpha)dx = \overline{x_n}$

Die Likelihood-Methode ist eher zu vertrauen, wenns um die Genauigkeit geht.