Serie 3

Aufgabe 3.1

Edwin Hubble untersuchte seit 1920 am Mount Wilson Observatory die Eigenschaften von Galaxien ausserhalb der Milchstrasse. Mit Überraschung bemerkte er einen Zusammenhang zwischen der Distanz einer Galaxie zur Erde und dessen Geschwindigkeit, sich von der Erde fortzubewegen (Fluchtgeschwindigkeit). Hubbles ursprüngliche Daten von 24 galaktischen Nebeln (E. Hubble, *Proceedings of the National Academy of Science* 15 (1929): 168-73) sind in Tabelle 1 gezeigt. Die Fluchtgeschwindigkeit ist in Kilometer pro Sekunde angegeben und konnte aufgrund der Rotverschiebung im Lichtspektrum der Galaxien mit grosser Genauigkeit bestimmt werden. Die Distanz einer Galaxie zur Erde wird in Megaparsec (Mpc) gemessen: ein Megaparsec entspricht etwa 3.09×10^{10} m. Die Distanzen werden durch Vergleich der mittleren Luminosität von Galaxien mit der Luminosität von bestimmten bekannten Sternen bestimmt, wobei diese Methode relativ ungenau ist.

a) Erstellen Sie von den Daten in Tabelle 1 ein Streudiagramm, in dem Sie die Distanz versus Fluchtgeschwindigkeit aufzeichnen.

   Lesen Sie dazu die Datei ein:

```
hubble = pd.read_csv("*/hubble.txt", sep=" ")
```

b) Bestimmen Sie mit dem Befehl **`np.polyfit(...)`** (siehe Skript) die Koeffizienten β_0 und β_1 für die Regressionsgerade

$$y = \beta_0 + \beta_1 x$$

   wobei y die Distanz und x die Fluchtgeschwindigkeit bezeichnet.

c) Bestimmen Sie noch den Korrelationskoeffizienten und interpretieren Sie diesen.

Aufgabe 3.2

Wir betrachten eine Studie, die 1979 in den Vereinigten Staaten durchgeführt wurde (National Longitudinal Study of Youth, NLSY79): von 2584 Amerikanern im Jahr 1981 wurde die Intelligenz (gemäss AFQT - armed forces qualifying test score) gemessen; 2006 wurden dieselben Personen nach ihrem jährlichen Einkommen im Jahr 2005 und der Anzahl Jahre Schulbildung befragt. Uns interessiert hier natürlich, ob hohe Intelligenz oder eine lange Schulbildung zu einem höheren Einkommen führen. In der auf Ilias abgelegten Datei `income.dat` finden Sie den Datensatz mit dem Einkommen,

Nebel	Geschwindigkeit (km/s)	Distanz (Mpc)
S. Mag.	170	0.032
L. Mag. 2	290	0.034
NGC 6822	-130	0.214
NGC 598	-70	0.263
NGC 221	-185	0.275
NGC 224	-220	0.275
NGC 5457	200	0.450
NGC 4736	290	0.500
NGC 5194	270	0.500
NGC 4449	200	0.630
NGC 4214	300	0.800
NGC 3031	-30	0.900
NGC 3627	650	0.900
NGC 4626	150	0.900
NGC 5236	500	0.900
NGC 1068	920	1.000
NGC 5055	450	1.100
NGC 7331	500	1.100
NGC 4258	500	1.400
NGC 4151	960	1.700
NGC 4382	500	2.000
NGC 4472	850	2.000
NGC 4486	800	2.000
NGC 4649	1090	2.000

Table 1: Zusammenhang zwischen Distanz und Fluchtgeschwindigkeit von Galaxien.

der Anzahl Jahre abgeschlossener Schulbildung und den ermittelten Intelligenzquotienten von 2584 Amerikanern.

a) Lesen Sie den Datensatz `income.dat` ein

```
income = pd.read_csv("*income.dat", sep=" ")
```

und generieren Sie zwei Streudiagramme, in welchen das Einkommen versus Anzahl Jahre Schulbildung bzw. Einkommen versus AFQT aufgetragen sind.

b) Bestimmen Sie die Parameter a und b des linearen Modells $y = a + bx$, wobei y das Einkommen bezeichnet und x die Anzahl Jahre Schulbildung bzw. AFQT. Zeichnen Sie die Regressionsgerade wie im Skript

```
x = np.linspace(... , ...)

plt.plot(x, a+b*x, c="orange")
```

Wie interpretieren Sie jeweils die Parameter a und b?

c) Berechnen Sie die Korrelation zwischen Einkommen und Anzahl Jahre Schulbildung bzw. AFQT. Wie angebracht ist jeweils das Regressionsmodell?

Aufgabe 3.3

In dieser Aufgabe betrachten wir 4 Datensätze, die von Anscombe konstruiert wurden.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5])
y1 = np.array([8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68])
y2 = np.array([9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74])
y3 = np.array([7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73])
x4 = np.array([8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8])
y4 = np.array([6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89])
```

In jedem der Datensätze gibt es eine Zielvariable y und eine erklärende Variable x.

a) Stellen Sie jeden der 4 Datensätze als Streudiagramm dar, zeichnen Sie die Regressionsgerade ein und kommentieren Sie die Ergebnisse. Verwenden Sie wieder **`plt.subplot(...)`** und **`plt.scatter(...)`**.

b) Vergleichen Sie die Schätzungen von β_0 und β_1, wobei $y = \beta_0 + \beta_1 x$.

Die Schätzungen für die Koeffizienten β_0 und β_1 des linearen Regressionsmodells kann man mit **`np.polyfit(...)`** berechnen und numerisch auswerten.

c) Berechnen Sie die Korrelationskoeffizienten mit **`np.corrcoef(...)`**.

Aufgabe 3.4

Sie werfen zusammen einen blauen und einen roten Würfel.

a) Bestimmen Sie die Wahrscheinlichkeitsverteilung der geworfenen Augensumme.

b) Berechnen Sie den Erwartungwert. Interpretieren Sie diesen Wert. Verwenden Sie dazu **`Python`**, indem Sie zwei Vektoren **`x`** und **`p`** erzeugen, die beiden multiplizieren und den Befehl **`np.sum(...)`** benützen.

Aufgabe 3.5

Berechnen Sie den Erwartungswert der folgenden Wahrscheinlichkeitsverteilung.

x_k	-5	-4	1	3	6
p_k	0.3	p_2	0.1	0.2	0.3

Kurzlösungen vereinzelter Aufgaben

A 3.2:

b) $a = -40'200$ und $b = 6451$ bzw. $a = 21'182$ und $b = 518.68$

c) $r = 0.346$ bzw. $r = 0.308$

A 3.3:

b) $\widehat{\beta}_0 \approx 3.00$ und $\widehat{\beta}_1 \approx 0.500$

c) Alle ungefähr 0.816 (auf dritte Stelle nach Komma)

A 3.4: Erwartungswert ist 7.

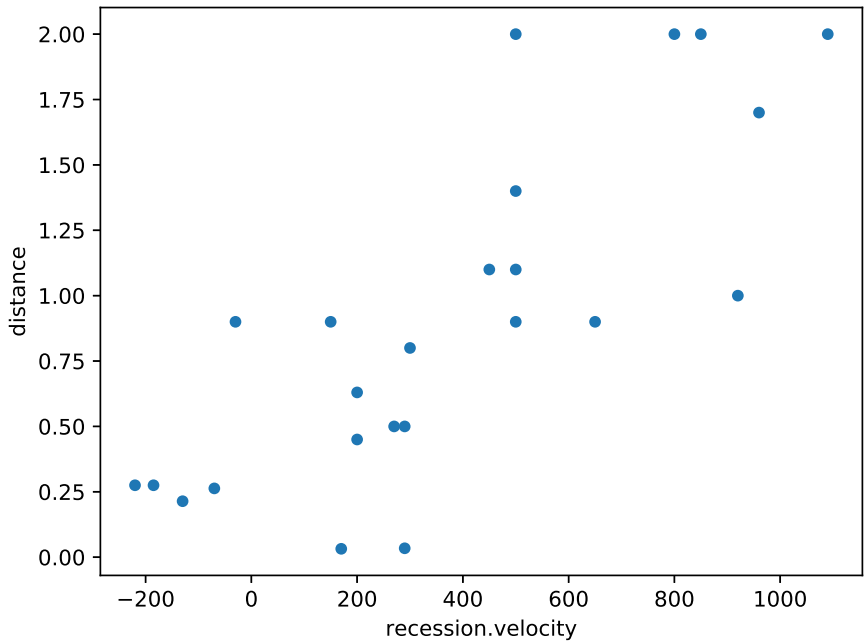
A 3.5: 0.6

Musterlösungen zu Serie 3

Lösung 3.1

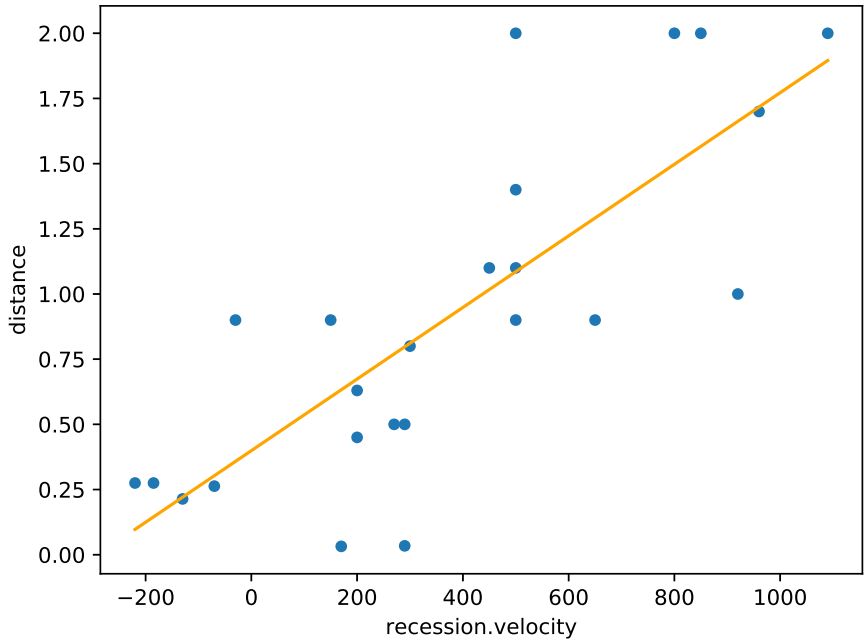
a) (zu **R**) Streudiagramm in **Python**

```
from pandas import Series
import matplotlib.pyplot as plt
hubble.plot(kind="scatter",
            x="recession.velocity",
            y="distance")
```



b) (zu **R**) Die Koeffizienten der Regressionsgerade berechnen sich mit

```
beta1, beta0 = np.polyfit(y=hubble["distance"],
                          x=hubble["recession.velocity"],
                          deg=1)
print(beta0, beta1)

## 0.3990982158435929 0.0013729361049417948
```




```
hubble.plot(kind="scatter",
            x="recession.velocity",
            y="distance")

beta1, beta0 = np.polyfit(y=hubble["distance"],
                          x=hubble["recession.velocity"],
                          deg=1)

x = np.linspace(hubble["recession.velocity"].min(),
                hubble["recession.velocity"].max())

plt.plot(x, beta0 + beta1*x, color="orange")

plt.show()
```

c) (zu R)

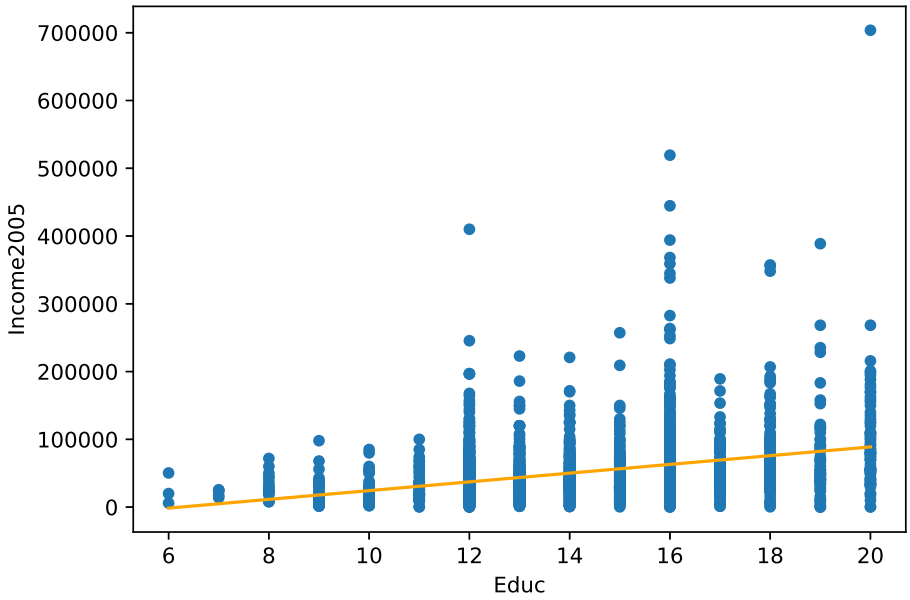
```
hubble.corr()

##                      distance  recession.velocity
## distance             1.000000            0.789639
## recession.velocity   0.789639            1.000000
```

Der Korrelationskoeffizient ist 0.79 und damit recht hoch (nahe bei 1), was auf einen linearen Zusammenhang von Abstand und Fluchtgeschwindigkeit der Galaxien hindeutet.

Lösung 3.2

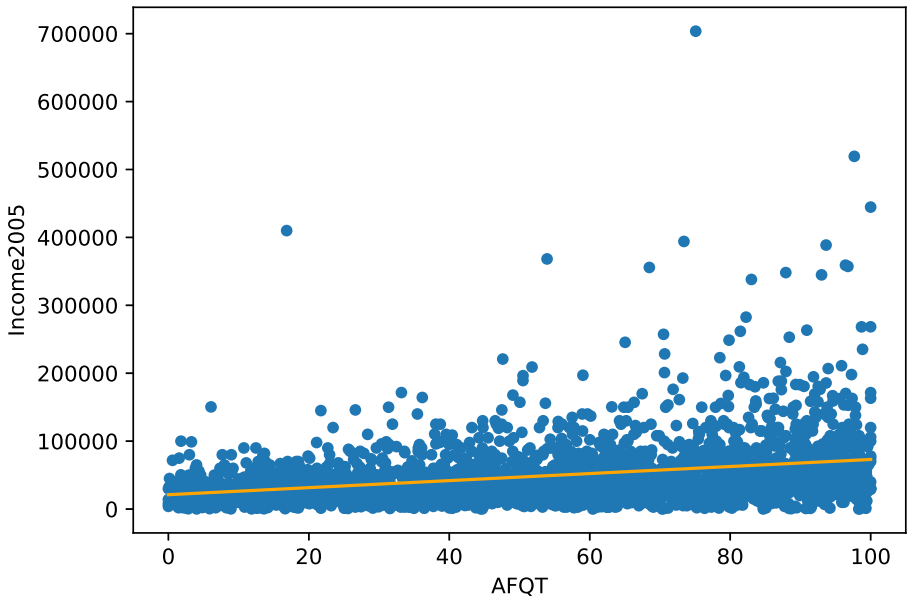
a) (zu R)

Streudiagramme




```
income.plot(kind="scatter", x="Educ", y="Income2005")

x = np.linspace(income["Educ"].min(), income["Educ"].max())

a, b = np.polyfit(income["Educ"], income["Income2005"], deg=1)

plt.plot(x,a*x+b,c="orange")

plt.show()
```

und



b) (zu **R**)

Mit **Python** ermitteln wir für a und b

```
b, a = np.polyfit(income["AFQT"], income["Income2005"], deg=1)
print(a, b)

## 21181.656863527787 518.6820790195897
```

```
b, a = np.polyfit(income["Educ"], income["Income2005"], deg=1)
print(a, b)

## -40199.57535260002 6451.4745559458015
```

Wir finden also die Werte $a = -40'200$ und $b = 6451$ für den Fall von Einkommen gegen Anzahl Jahre Schulbildung (und $a = 21'182$ und $b = 518.68$ für den betrachteten Fall Einkommen gegen AFQT). Mit jedem zusätzlichen Jahr Schulbildung geht also eine jährliche Einkommenszunahme von etwa 6500 USD einher. Jeder zusätzlichen Punkt im AFQT bewirkt laut Modell eine jährliche Einkommenszunahme von rund 500 USD.

Nun ist allerdings Vorsicht geboten: jemand ohne Schulbildung würde ein Einkommen von $-40'200$ USD haben. Dies macht natürlich keinen Sinn. Wann immer man in Bereiche extrapoliert, wo keine Datenpunkte vorhanden waren, ist Vorsicht bei der Interpretation geboten.

c) Für die *empirische Korrelation* erhalten wir dann

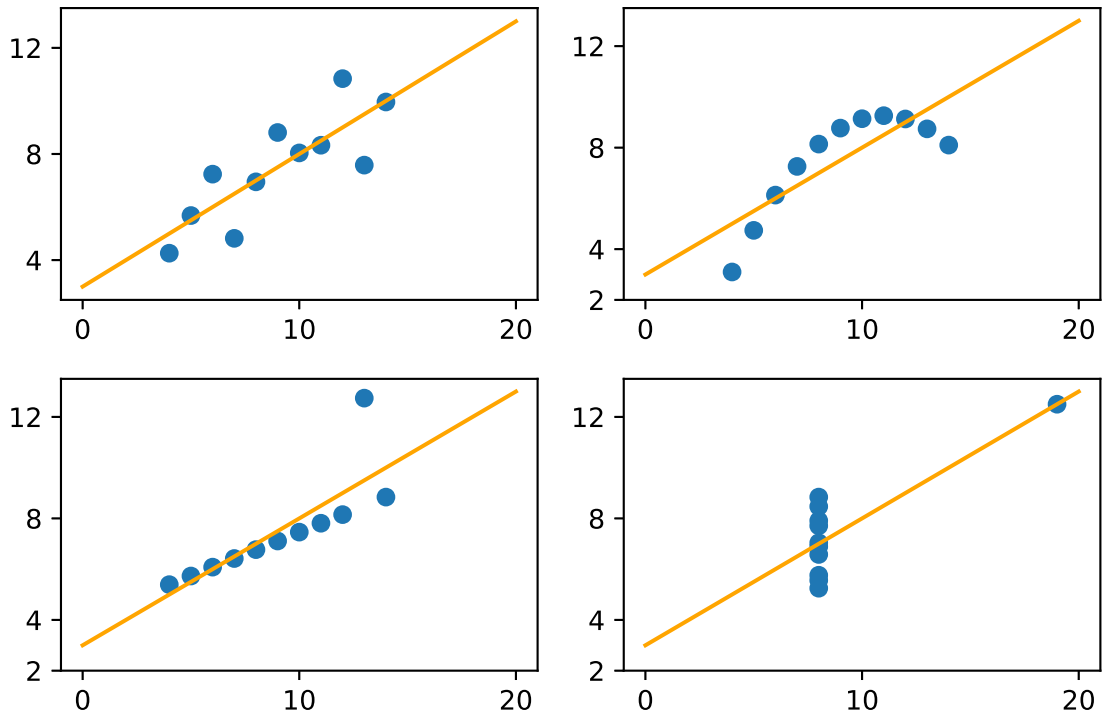
```
income.corr()

##                  AFQT      Educ  Income2005
## AFQT         1.000000  0.595160    0.308153
## Educ         0.595160  1.000000    0.345647
## Income2005   0.308153  0.345647    1.000000
```

Da beide Korrelationskoeffizienten mit 0.346 bzw. 0.308 relativ klein sind, scheint in beiden Fällen ein Modell beruhend auf einem linearen Zusammenhang nicht angebracht zu sein.

Lösung 3.3

a) (zu R)

Betrachtet man die vier Streudiagramme (hier noch skaliert), so sieht man, dass nur im ersten Fall eine lineare Regression korrekt ist. Im zweiten Fall ist die Beziehung zwischen X und Y nicht linear, sondern quadratisch. Im dritten Fall gibt es einen Ausreisser, welcher die geschätzten Parameter stark beeinflusst. Im vierten Fall wird die Regressionsgerade durch einen einzigen Punkt bestimmt.



b) (zu R)

Bei allen vier Modellen sind die Schätzungen des Achsenabschnitts β_0 und der Steigung β_1 fast identisch. Für das erste Beispiel erhalten wir

```
np.polyfit(x, y1, deg=1)

## [0.50009091 3.00009091]
```

	Modell 1	Modell 2	Modell 3	Modell 4
Achsenabschnitt ($\widehat{\beta}_0$)	3.000	3.001	3.002	3.002
Steigung ($\widehat{\beta}_1$)	0.500	0.500	0.500	0.500

Alle Streudiagramme haben praktisch dieselbe Regressionsgerade, obwohl die Streudiagramme sehr unterschiedlich aussehen.

Fazit: Es genügt **nicht**, nur $\widehat{\beta}_0$ und $\widehat{\beta}_1$ anzuschauen. In allen Modellen sind diese Schätzungen fast gleich, aber die Datensätze sehen ganz unterschiedlich aus. Eine (graphische) Überprüfung der Modellannahmen ist also unumgänglich.

c) (zu **R**)

Für das erste Beispiel erhalten wir

```
np.corrcoef(x, y1)

## [[1.         0.81642052]
##  [0.81642052 1.        ]]
```

Lösung 3.4

a) Sei X die Zufallsvariable für die geworfene Augensumme. Dann ist

$$\Omega = \{2, 3, 4, \ldots, 12\}$$

Es ergibt sich dann für die Wahrscheinlichkeitsverteilung folgende Tabelle

x_i	Elementarereignis	abs. Häufigkeit	p_i
2	11	1	$\frac{1}{36}$
3	12,21	2	$\frac{2}{36}$
4	13,22,31	3	$\frac{3}{36}$
5	14,23,32,41	4	$\frac{4}{36}$
6	15,24,33,42,51	5	$\frac{5}{36}$
7	16,25,34,43,52,61	6	$\frac{6}{36}$
8	26,35,44,53,62	5	$\frac{5}{36}$
9	36,45,54,63	4	$\frac{4}{36}$
10	46,55,64	3	$\frac{3}{36}$
11	56,65	2	$\frac{2}{36}$
12	66	1	$\frac{1}{36}$

b) Der Erwartungswert berechnet sich durch

$$\begin{aligned} \mathrm{E}(X) &= x_1 p_1 + x_2 p_2 + \ldots + x_{11} p_{11} \\ &= 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + \ldots + 12 \cdot \frac{1}{36} \\ &= 7 \end{aligned}$$

Wir verwenden zur Berechnung **Python**

```
import numpy as np
x = np.arange(2 , 13)
p = np.array([1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]) / 36

np.sum(x * p)

## 6.999999999999998
```

Wenn wir sehr viele Male die beiden Würfel werfen, so werfen wir durchschnittlich die Augenzahl 7. Das war aber auch so zu erwarten, weil die Tabelle oben symmetrisch ist.

Lösung 3.5

Zuerst müssen wir den Wert für p_2 bestimmen. Da die Summe 1 ergeben muss, gilt

$$p_2 = 1 - 0.3 - 0.1 - 0.2 - 0.3 = 0.1$$

Der Erwartungswert berechnet sich durch

$$\begin{aligned} \mathrm{E}(X) &= x_1 p_1 + x_2 p_2 + \ldots + x_5 p_5 \\ &= -5 \cdot 0.3 + (-4) \cdot 0.1 + \ldots + 6 \cdot 0.3 = 0.6 \end{aligned}$$

Wir verwenden wieder **Python**:

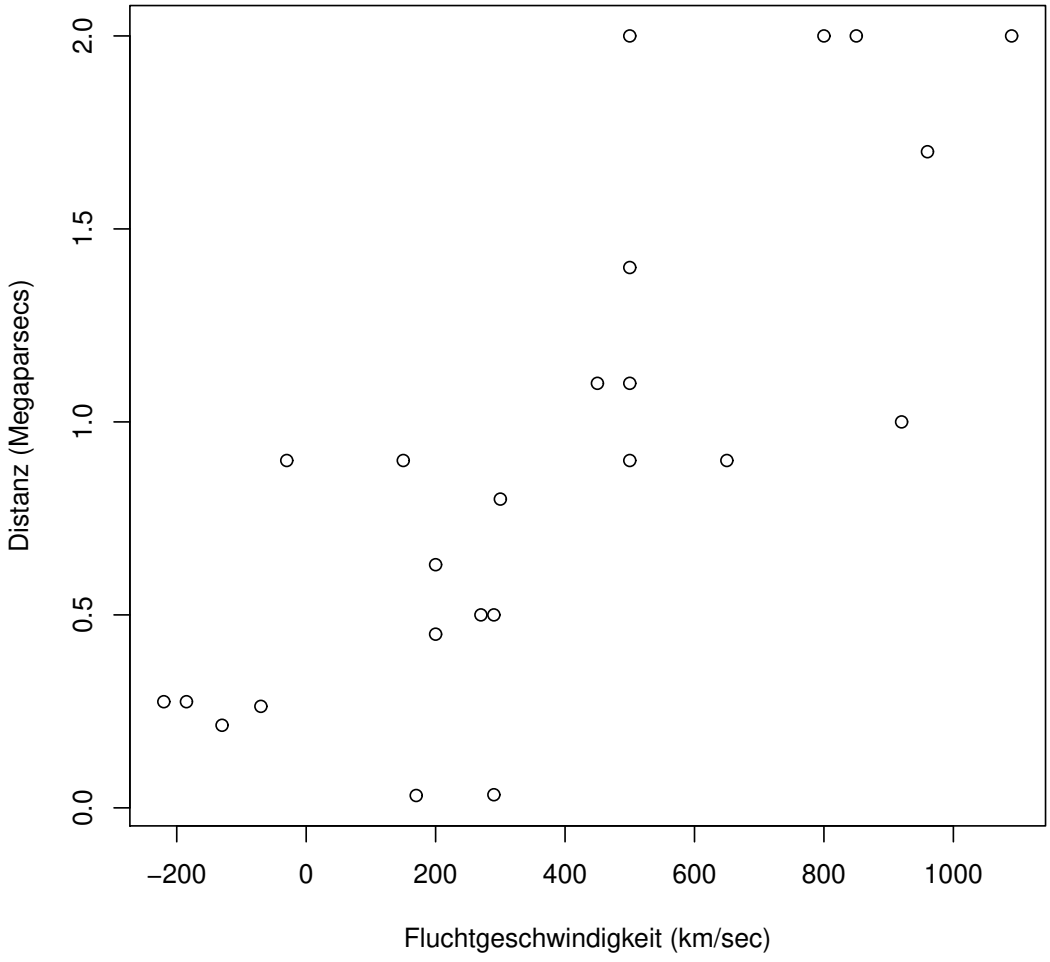
```
x = np.array([-5, -4, 1, 3, 6])
p = np.array([0.3, 0.1, 0.1, 0.2, 0.3])

np.sum(x * p)


## 0.6000000000000001
```

R-Code

3.1

a) (zu **Python**)

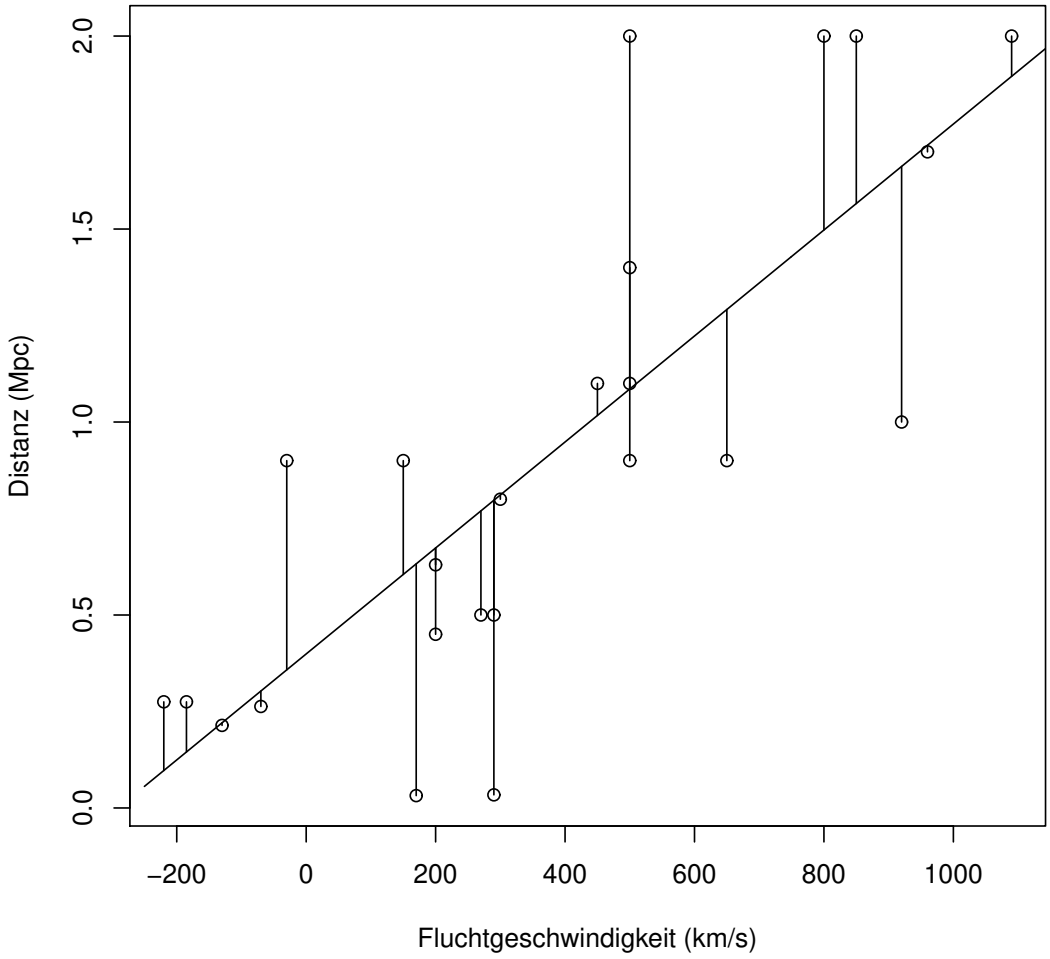
```
recession.velocity <- c(170, 290, -130, -70, -185,
    -220, 200, 290, 270, 200, 300, -30, 650, 150, 500,
    920, 450, 500, 500, 960, 500, 850, 800, 1090)
distance <- c(0.032, 0.034, 0.214, 0.263, 0.275, 0.275,
    0.45, 0.5, 0.5, 0.63, 0.8, 0.9, 0.9, 0.9, 0.9,
    1, 1.1, 1.1, 1.4, 1.7, 2, 2, 2, 2)
plot(recession.velocity, distance, ylab = "Distanz (Megaparsecs)",
    xlab = "Fluchtgeschwindigkeit (km/sec)")
```



b) (zu **Python**)

```
regr_model <- lm(distance ~ recession.velocity)
beta_0 <- regr_model$coefficients[1]
beta_0

## (Intercept)
##   0.3990982

beta_1 <- regr_model$coefficients[2]
beta_1

## recession.velocity
##        0.001372936
```

```
plot(recession.velocity, distance, ylab = "Distanz (Mpc)",
    xlab = "Fluchtgeschwindigkeit (km/s)")
lines(-250:1200, beta_0 + beta_1 * (-250:1200), type = "l",
    new = TRUE)
segments(recession.velocity, beta_0 + beta_1 * (recession.velocity),
    recession.velocity, distance)
```

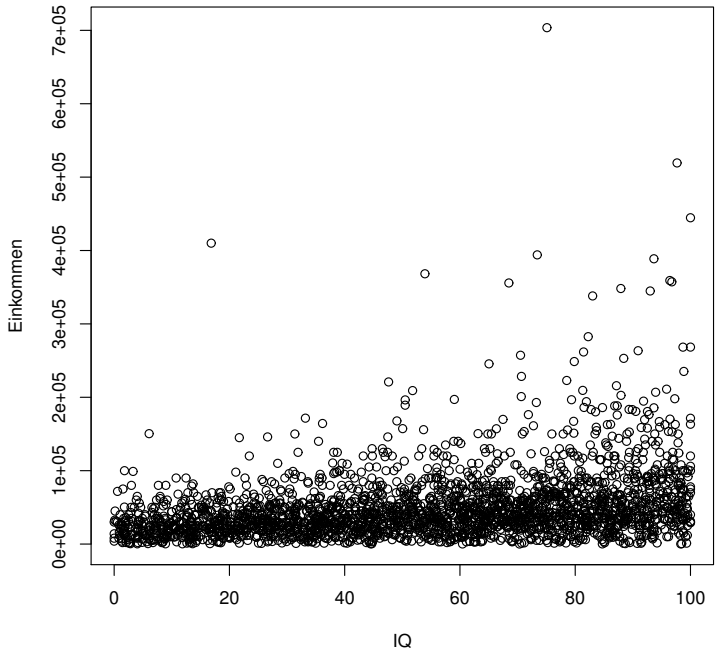


Aufgabe 3.2

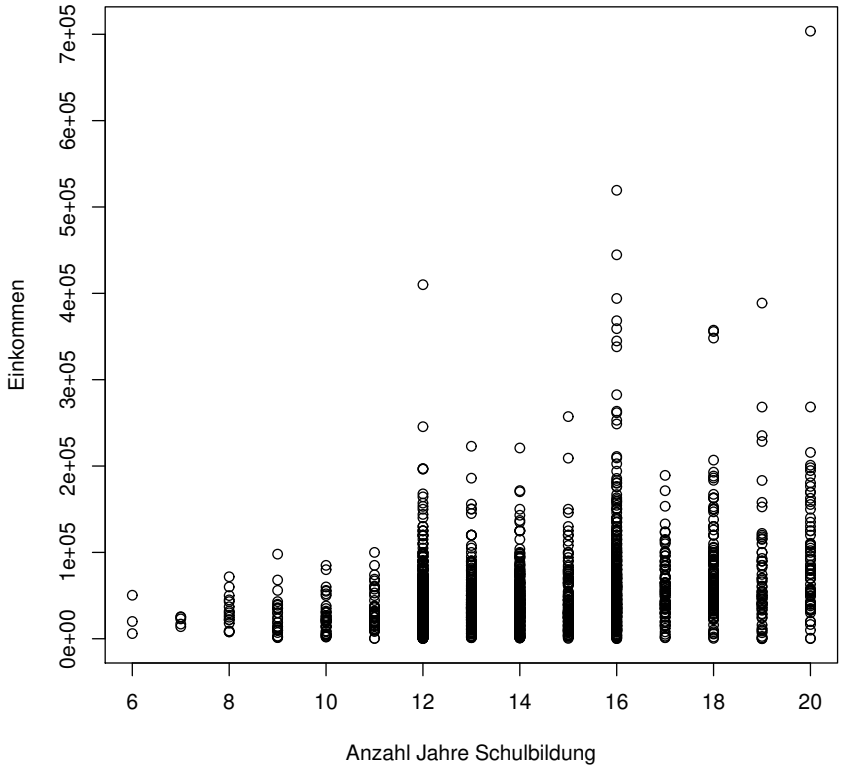
a) (zu **Python**)

```
income <- read.table(file = "./Daten/income.dat", header = TRUE)
head(income)

##      AFQT Educ Income2005
## 1   6.841   12       5500
## 2 99.393   16      65000
## 3 47.412   12      19000
## 4 44.022   14      36000
## 5 59.683   14      65000
```

```
## 6 72.313   16       8000

iq <- income[, 1]
anzahl.jahre.schule <- income[, 2]
einkommen <- income[, 3]
plot(iq, einkommen, type = "p", xlab = "IQ", ylab = "Einkommen")
```




```
plot(anzahl.jahre.schule, einkommen, type = "p",
    xlab = "Anzahl Jahre Schulbildung", ylab = "Einkommen")
```



b) (zu **Python**)

```
lm(einkommen ~ iq)

##
## Call:
## lm(formula = einkommen ~ iq)
##
## Coefficients:
## (Intercept)           iq
##     21181.7        518.7

lm(einkommen ~ anzahl.jahre.schule)

##
## Call:
## lm(formula = einkommen ~ anzahl.jahre.schule)
##
## Coefficients:
##         (Intercept)  anzahl.jahre.schule
##               -40200                 6451
```

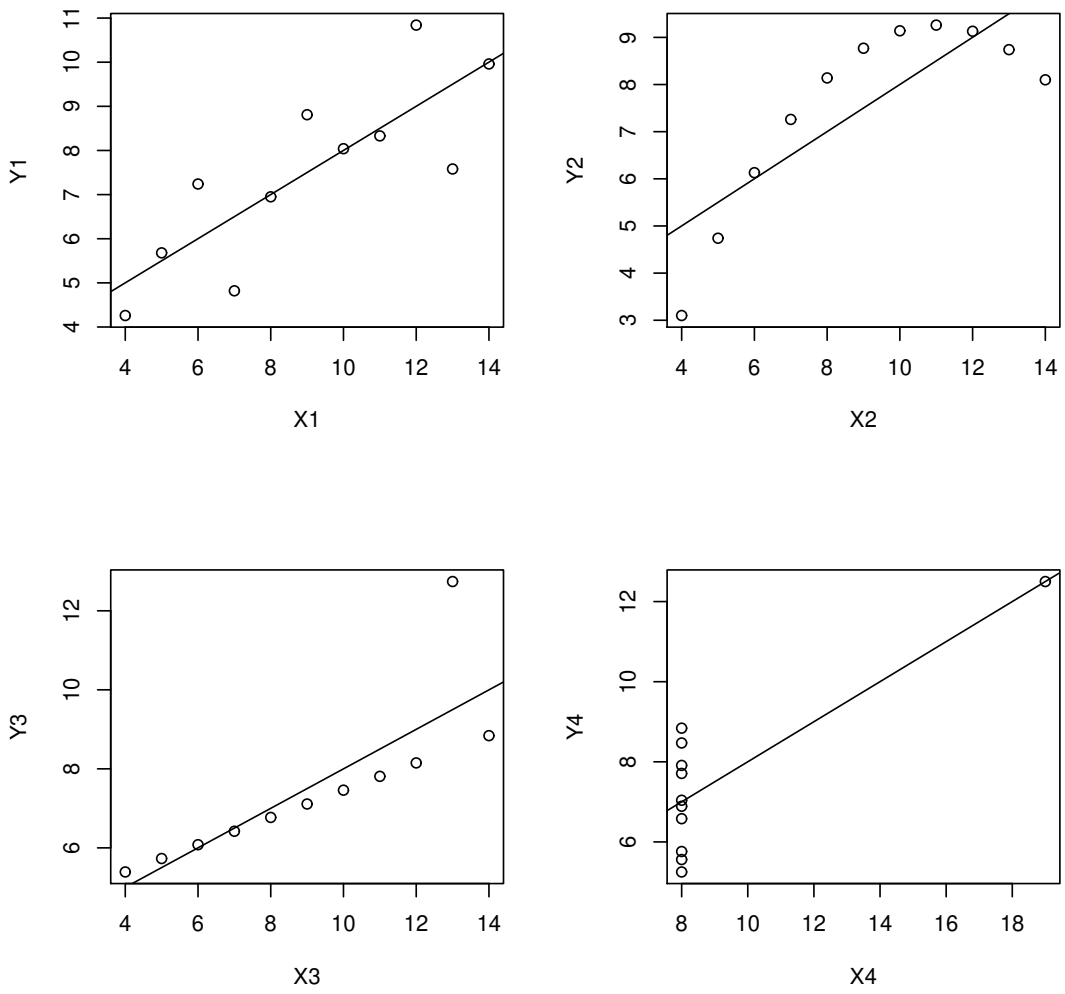
c) (zu **Python**)

```
cor(anzahl.jahre.schule, einkommen)

## [1] 0.3456474
```

Aufgabe 3.3

a) (zu **Python**)

```
data(anscombe)  ## Einlesen des Datensatzes

reg <- lm(anscombe$y1 ~ anscombe$x1)
reg2 <- lm(anscombe$y2 ~ anscombe$x2)
reg3 <- lm(anscombe$y3 ~ anscombe$x3)
reg4 <- lm(anscombe$y4 ~ anscombe$x4)
par(mfrow = c(2, 2))
plot(anscombe$x1, anscombe$y1, ylab = "Y1", xlab = "X1")
abline(reg)
plot(anscombe$x2, anscombe$y2, ylab = "Y2", xlab = "X2")
abline(reg2)
plot(anscombe$x3, anscombe$y3, ylab = "Y3", xlab = "X3")
abline(reg3)
plot(anscombe$x4, anscombe$y4, ylab = "Y4", xlab = "X4")
abline(reg4)
```



b) (zu **Python**)

```
lm(anscombe$y1 ~ anscombe$x1)

##
## Call:
## lm(formula = anscombe$y1 ~ anscombe$x1)
##
## Coefficients:
## (Intercept)  anscombe$x1
##      3.0001       0.5001
```

c) (zu **Python**)

```
cor(anscombe$y1, anscombe$x1)
```

```
## [1] 0.8164205
```