Data Science Assignment 2

Nike Marie Pulow – Henri Paul Heyden

stu239549 – stu240825

1 Contigency tables, Chi-squared test

Calculating the marginal total frequencies:

	good	medium	strong delay	total
without alcohol	120	60	20	200
with alcohol	60	100	40	200
total	180	160	60	400

Looking at the results for **good reaction time** we can see a strong decrease in the test group that has consumed alcohol, compared to the test group that has not consumed alcohol. For the **medium reaction time** we can observe an increase in frequency for the with-alcohol group. Alongside the doubling in test subjects that had a **strong delay** in their reaction time after consumption of alcohol, this might indicate that the consumption of alcohol might have negative effects on the reaction time of humans.

To analyze this further, we'll now calculate the conditional relative frequency distribution.

$$\begin{aligned}
f(\text{without alcohol} \mid \text{good}) &= \frac{120}{120+60} = \frac{2}{3} \\
f(\text{without alcohol} \mid \text{medium}) &= \frac{60}{60+100} = \frac{3}{8} \\
f(\text{without alcohol} \mid \text{strong delay}) &= \frac{20}{20+40} = \frac{1}{3} \\
f(\text{with alcohol} \mid \text{good}) &= \frac{60}{180} = \frac{1}{3} \\
f(\text{with alcohol} \mid \text{medium}) &= \frac{100}{160} = \frac{5}{8} \\
f(\text{with alcohol} \mid \text{strong delay}) &= \frac{40}{60} = \frac{2}{3}
\end{aligned}$$

These results can be displayed as a table again:

	good	medium	strong delay
without alcohol	66.6%	37.5%	33.3%
with alcohol	33.3%	62.5%	66.6%

These results show that the above stated indication might be true. Less subjects under the influence of alcohol had a good reaction time, while more subjects under the influence of alcohol showed a strong delay in their reactions. To further explore these findings and find out, whether the consumption of alcohol is associated with the worsening of reaction times in humans, we will now conduct a Chi-squared test with significance level $\alpha = 0.005$.

Null Hypothesis: There is no association between the consumption of alcohol and reaction times.
Alternative Hypothesis: There is an association between the consumption of alcohol and reaction times.

$$e_{11} = \frac{200 \cdot 180}{400} = 90 \qquad \text{(good | without alcohol)}$$
$$e_{21} = \frac{200 \cdot 180}{400} = 90 \qquad \text{(good | with alcohol)}$$
$$e_{12} = \frac{200 \cdot 160}{400} = 80 \qquad \text{(medium | without alcohol)}$$
$$e_{22} = \frac{200 \cdot 160}{400} = 80 \qquad \text{(medium | with alcohol)}$$
$$e_{13} = \frac{200 \cdot 60}{400} = 30 \qquad \text{(strong delay | without alcohol)}$$
$$e_{23} = \frac{200 \cdot 60}{400} = 30 \qquad \text{(strong delay | with alcohol)}$$

With these expected values e_{ij} we can update the contingency table as follows:

	good	medium	strong delay	total
without alcohol	120 (90)	60 (80)	20 (30)	200
with alcohol	60 (90)	100 (80)	40 (30)	200
total	180	160	60	400

We'll now compute the x^2-value for this data.

$$x^2{}_{11} = \frac{(120-90)^2}{90} = 10 \qquad \text{(good | without alcohol)}$$
$$x^2{}_{12} = \frac{(60-80)^2}{80} = 5 \qquad \text{(medium | without alcohol)}$$
$$x^2{}_{13} = \frac{(20-30)^2}{30} = 3.3 \qquad \text{(strong delay | without alcohol)}$$
$$x^2{}_{21} = \frac{(60-90)^2}{90} = 10 \qquad \text{(good | with alcohol)}$$
$$x^2{}_{12} = \frac{(60-80)^2}{80} = 5 \qquad \text{(medium | with alcohol)}$$
$$x^2{}_{13} = \frac{(20-30)^2}{30} = 3.3 \qquad \text{(strong delay | with alcohol)}$$

$x^2 = \sum_{i,j=1}^{i,j} x_{ij} = 36.6$

Next, calculate the Degrees of Freedom: $df = (2-1)(3-1) = 1 \cdot 2 = 2$. For a significance level of $\alpha = 0.005$, the chi-squared value to compare x^2 to is 10.597. It holds $36.6 > 10.597$, so we can safely reject the Null-Hypothesis. In conclusion, the data and the chi-squared test show, that there is in fact an association between the consumption of alcohol and reaction times in humans.

2 Feature Types

1.
   - Interval-scale
   - Ordinal
   - Ratio-scale
   - Ratio-scale
   - Ordinal
   - Nominal
   - Interval-scale

2.

Interval	Ratio
blood pressure in $mmHg$	water temperature in °C
air pressure in atm	elementary charge in C
Lorentz-factor	voltage in V
refractive index	electric current in A

3 Time Series

To avoid any calculation errors, we have written a small python script using pandas to do any calculations for this task. The code is provided in section 4. First, the L_1 and L_∞ distances are calculated for the raw data:

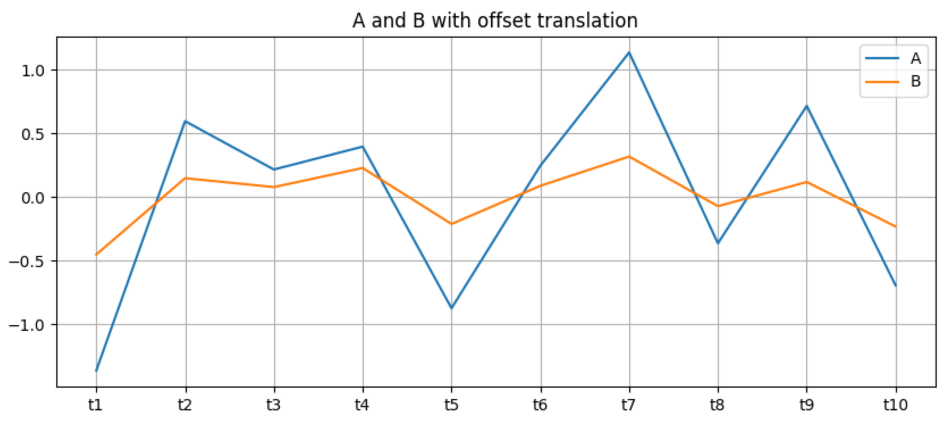
$$\begin{aligned}\text{dist}_1 =& |-1.66-0.29| + |0.30-0.89| + |-0.08-0.82| + |0.10-0.97| + |-1,17-0.53| \\ &+ |-0.05-0.83| + |0.84-1.06| + |-0.66-0.67| + |0.42-0.86| + |-0.99-0.51| \\ =& 8.88\end{aligned}$$

$$\begin{aligned}\text{dist}_\infty =& \max(|-1.66-0.29| + |0.30-0.89| + |-0.08-0.82| + |0.10-0.97| + |-1,17-0.53| \\ &+ |-0.05-0.83| + |0.84-1.06| + |-0.66-0.67| + |0.42-0.86| + |-0.99-0.51|) \\ =& 1.95\end{aligned}$$

Next, we carry out offset translation, i.e. we calculate the mean for each time series and substract the respective mean from each value in the series. This is the translated data for both time series. We have also included a plot that shows both time series after offset translation. The distances L_1 and L_∞ based on this updated data are, using the same calculation schema as shown above:

$$\text{dist}_1 = 4.194$$
$$\text{dist}_\infty = 0.912$$

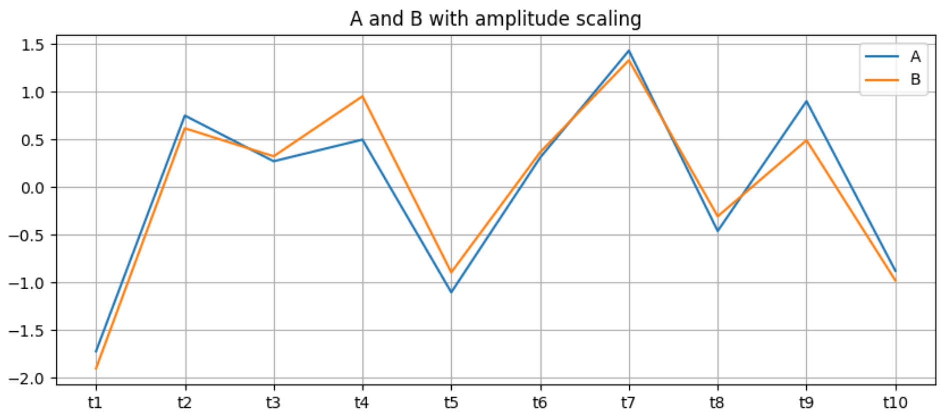
Series	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
A	−1.365	0.595	0.215	0.395	−0.875	0.245	1.135	−0.365	0.715	−0.695
B	−0.453	0.147	0.077	0.227	−0.213	0.087	0.317	−0.073	0.117	−0.233



Now, we take the results from offset translation and carry out amplitude scaling for both time series. Note that, in the python code in section 4 the function amp_scaling() takes the raw data, not the translated data from the previous part to avoid any rounding errors during the calculation. We again have summarized the results in a table and plot. The distances based on the data after amplitude scaling are again calculated in the above demonstrated scheme:

$$\text{dist}_1 = 1.7506937958804625$$
$$\text{dist}_\infty = 0.45459264054347487$$

Series	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
A	-1.721128	0.750235	0.271093	0.498055	-1.103287	0.308920	1.431121	-0.460228	0.901543	-0.876325
B	-1.901099	0.616913	0.323145	0.952648	-0.893894	0.365112	1.330350	-0.306358	0.491012	-0.977828



4 Appendix A

```python
import pandas as pd

a = {'t1': -1.66, 't2': 0.30, 't3': -0.08, 't4': 0.10, 't5': -1.17, 't6': -0.05, 't7':
    0.84, 't8': -0.66, 't9': 0.42, 't10': -0.99}
b = {'t1': 0.29, 't2': 0.89, 't3': 0.82, 't4': 0.97, 't5': 0.53, 't6': 0.83, 't7': 1.06,
     't8': 0.67, 't9': 0.86, 't10': 0.51}

sera = pd.Series(data=a, index=['t1','t2','t3','t4','t5','t6','t7','t8','t9','t10'])
serb = pd.Series(data=b, index=['t1','t2','t3','t4','t5','t6','t7','t8','t9','t10'])

indexlist=['t1','t2','t3','t4','t5','t6','t7','t8','t9','t10']

# function to calculate manhatten distances
def man_dist(series1, series2):
    if series1.size == series2.size:
        result = []
        series1list = series1.to_list()
        series2list = series2.to_list()
        for i in range(0,series1.size-1):
            result = result + [abs(series1list[i] - series2list[i])]
        return sum(result)
    else:
        print("series not the same size, aborting")

# function to calculate max distance
def inf_dist(series1, series2):
    if series1.size == series2.size:
        result = []
        series1list = series1.to_list()
        series2list = series2.to_list()
        for i in range(0,series1.size -1):
            result = result + [abs(series1list[i] - series2list[i])]
        return max(result)
    else:
        print('series not the same size, aborting')

# task 1:
print('Results for task 1, calculating the distances for raw data:')

print('Manhatten Distance:')
print(man_dist(sera, serb))

print('Inf Distance:')
print(inf_dist(sera, serb))

# offset translation

def offset_translation(series, mean):
    result = []
    serieslist = series.to_list()
    for i in range(0,series.size):
        result = result + [(serieslist[i] - mean)]
    return pd.Series(result, index = indexlist)

sera_offset = offset_translation(sera, sera.mean())
serb_offset = offset_translation(serb, serb.mean())

print('Series A with offset translation:')
print(sera_offset)
print('Series B with offset translation:')
print(serb_offset)

sera_offset.name = 'A'
serb_offset.name = 'B'

amp_df = pd.merge(sera_offset,serb_offset,left_index=True,right_index=True)
plot = amp_df.plot(grid=True,figsize=(10,4),title='A and B with offset translation')
plot.set_xticks(range(len(amp_df)))
plot.set_xticklabels(["%s" % item for item in indexlist])
```

```python
fig = plot.get_figure()
fig.savefig('plots/offset_translation.png')

print('Recalculating distances:')
print('Manhatten Distance:')
print(man_dist(sera_offset, serb_offset))
print('Inf Distance:')
print(inf_dist(sera_offset,serb_offset))

# amplitude scaling:
def amp_scaling(series):
    mean = series.mean()    # mean
    s = series.std()        # standard deviation
    result = []
    serieslist = series.to_list()
    for i in range(0,series.size):
        result = result + [(serieslist[i] - mean)/s]
    return pd.Series(result, index=indexlist)

sera_amp = amp_scaling(sera)
serb_amp = amp_scaling(serb)

print('Series A with amplitude scaling:')
print(sera_amp)
print('Series B with amplitude scaling:')
print(serb_amp)
print('Recalculate distances:')
print('Manhatten Distance:')
print(man_dist(sera_amp,serb_amp))
print('Inf Distance:')
print(inf_dist(sera_amp,serb_amp))

sera_amp.name = 'A'
serb_amp.name = 'B'

amp_df = pd.merge(sera_amp,serb_amp,left_index=True,right_index=True)
plot = amp_df.plot(grid=True,figsize=(10,4),title='A and B with amplitude scaling')
plot.set_xticks(range(len(amp_df)))
plot.set_xticklabels(["%s" % item for item in indexlist])
fig = plot.get_figure()
fig.savefig('plots/amplitude_scaling.png')

```

4

November 23, 2024

0.1 4 Linear Regression

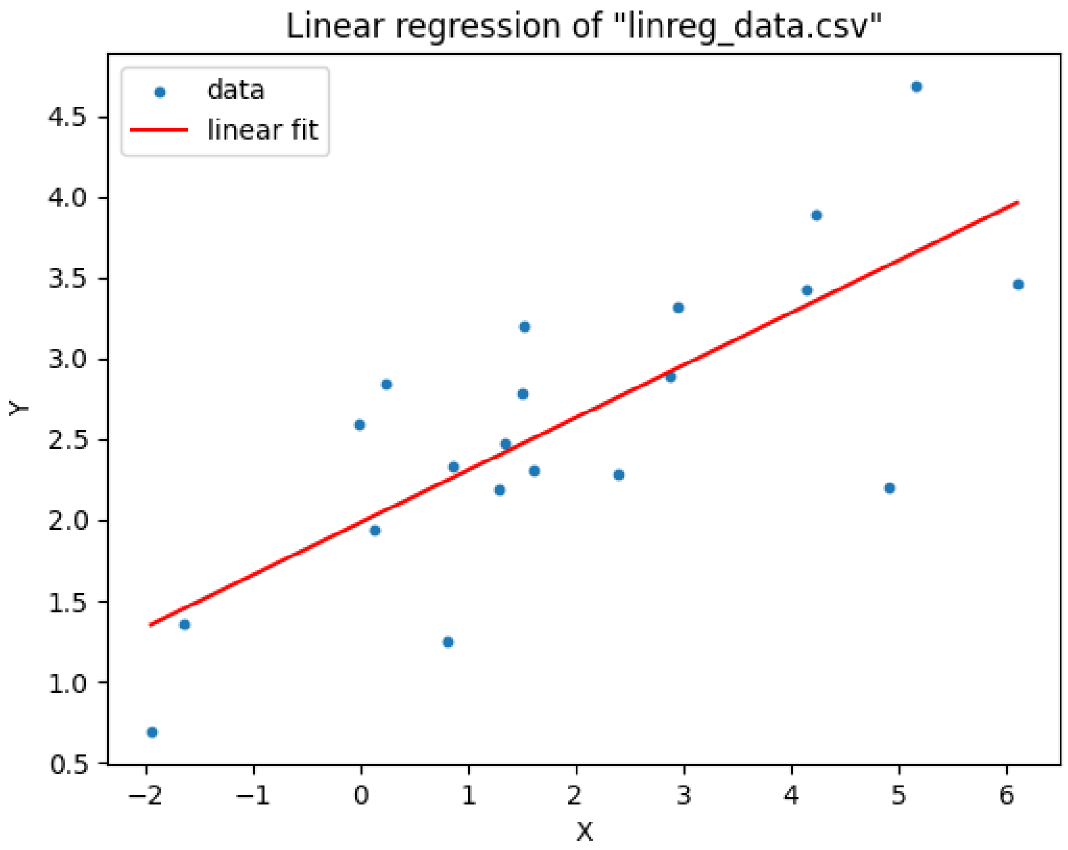
[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("./linreg_data.csv")
data.head()
```

[2]:
```
   Unnamed: 0         X         Y
0           0  4.910131  2.196544
1           1  1.500393  2.776927
2           2  2.946845  3.316272
3           3  6.102233  3.459587
4           4  5.168895  4.685546
```

1. For the given example, implement a linear regression minimizing the sum of squared error from scratch.

[5]:
```
y = data["Y"].to_list()
X = data["X"].to_list()

minX = min(X)
maxX = max(X)

assert len(y) == len(X)

# very stupid, but it works
for i in range(len(y)):
    y[i] = [y[i]]
    X[i] = [1, X[i]]

B = np.matmul(
    (
        np.linalg.inv(
            np.matmul(
                np.transpose(X), X
            )
        )
    ), np.matmul(
        np.transpose(X),
        y
    )
)

B = B.tolist()
B = [B[0][0], B[1][0]]
print(B)

plt.title("Linear regression of \"linreg_data.csv\"")
plt.scatter(data["X"].to_list(), data["Y"].to_list(), s=10, label="data")
x = np.linspace(minX, maxX, 1000)
plt.plot(x, B[0] + B[1] * x, label="linear fit", color="r")


plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.show()
```

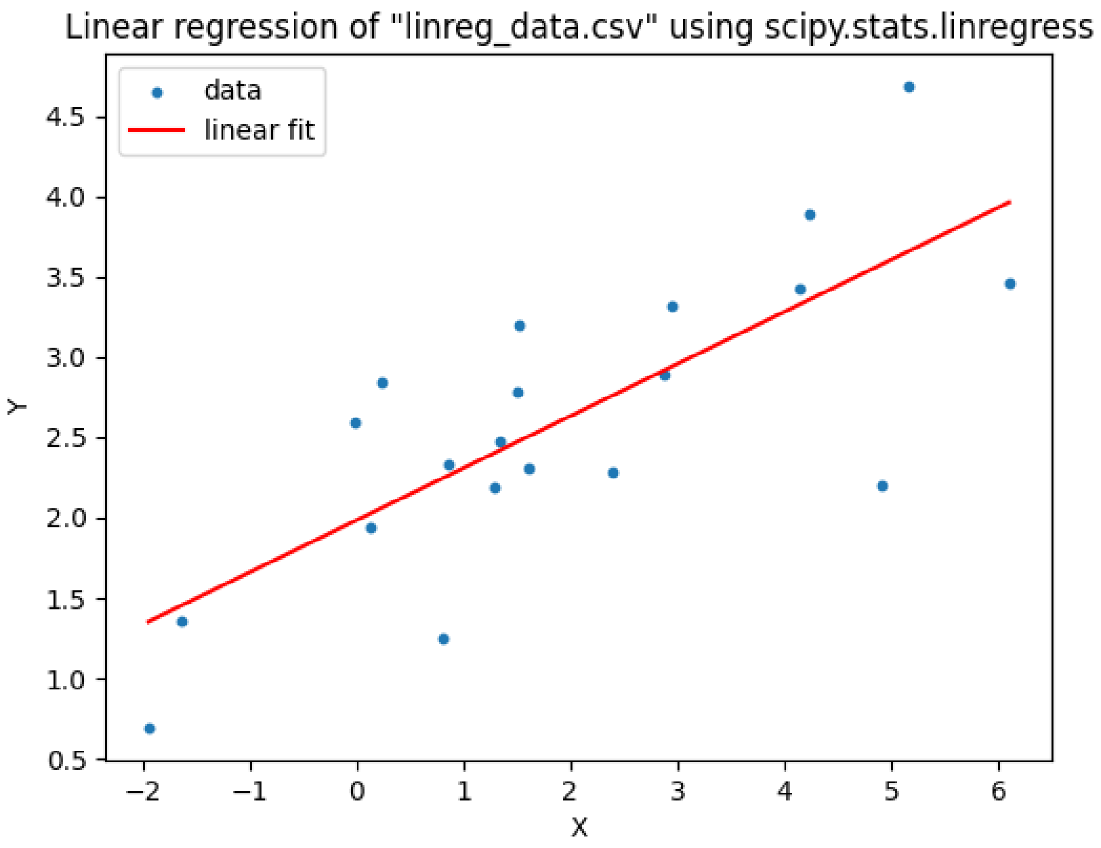
```
[1.9809107076502455, 0.32441822817991023]
```



2. Do the same task again, but this time using "scipy.stats.linregress".

```
[8]: import scipy.stats as sf

     X = data["X"].to_list()
     Y = data["Y"].to_list()

     res = sf.linregress(X, Y)
     print([res.intercept, res.slope])

     plt.title("Linear regression of \"linreg_data.csv\" using scipy.stats.
       ↪linregress")

     plt.scatter(X, Y, s=10, label="data")
     x = np.linspace(minX, maxX, 1000)
     plt.plot(x, res.intercept + res.slope * x, "r", label="linear fit")

     plt.xlabel("X")
     plt.ylabel("Y")
     plt.legend()
     plt.show()
```

```
[np.float64(1.9809107076502441), np.float64(0.32441822817991056)]
```



3. Use this regression to fill in a missing value at X = 2. What is the corresponding Y value?

```
[9]: print(f"Value at x = 2 is: {res.intercept + res.slope * 2}")
```

```
Value at x = 2 is: 2.629747164010065
```

4. How can we use this regression to smooth the original observations? Now that we have a basic line we can fit through the dataset, we can reduce each datapoint by the linear regression value, so that the average is at zero and the standard derivation is the average of this adjusted dataset. With this modification of the data, we can easily see which points “stand out” of a linear model.