Chapter 9 exercises

The set of exercises given in the document are intended to give a more "hands on" approach to the material in Chapter 9. The execireses are given per lecture, and are of course intended to be solved after that lecture, only lectures that cover material in Chapter 9 is included in this document.

Exercise 1.1 Lecture 1

a) **Summary statistics:** For the data in Example 1.20, write $\bar{y}$, S, $\hat{\sigma}$, and R, when $y_i = [x_i, y_i]$.

Solution

The solution is simply to insert the values calculated in Example 1.20:

$$\bar{y} = \begin{bmatrix} 178 \\ 78.1 \end{bmatrix}; \quad S = \begin{bmatrix} 12.21^2 & 165.9 \\ 165.9 & 14.07^2 \end{bmatrix}; \quad \hat{\sigma} = \begin{bmatrix} 12.21 & 0 \\ 0 & 14.07 \end{bmatrix}; \quad R = \begin{bmatrix} 1 & 0.97 \\ 0.97 & 1 \end{bmatrix}.$$

Exercise 1.2 Lecture 2

a) **Eigenvalue:** From the previous question, use Python to find the eigenvalue decomposition of S, and use that to find a matrix $S^{\frac{1}{2}}$, such that $S = S^{\frac{1}{2}} S^{\frac{T}{2}}$.

Solution

```
S = np.array([[12.21**2,165.9],[165.9,14.07**2]])
Eigvals, Eigvectors = np.linalg.eig(S)
Eigvals


array([  5.83388264, 341.21511736])


Eigvectors


array([[-0.75688406, -0.65354917],
       [ 0.65354917, -0.75688406]])
```

And hence reffering to Lemma 9.3 we have (we multiply $\boldsymbol{V}$, with -1)

$$\boldsymbol{V} = \begin{bmatrix} 0.75 & -0.65 \\ 0.65 & 0.76 \end{bmatrix}; \quad \boldsymbol{\Lambda} = \begin{bmatrix} 5.53 & 0 \\ 0 & 341.2 \end{bmatrix}, \tag{1-1}$$

and we can write $\boldsymbol{S}^{\frac{1}{2}}$

$$\boldsymbol{S}^{\frac{1}{2}} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}} = \begin{bmatrix} 0.75 & -0.65 \\ 0.65 & 0.76 \end{bmatrix} \begin{bmatrix} \sqrt{5.53} & 0 \\ 0 & \sqrt{341.2} \end{bmatrix} = \begin{bmatrix} 0.75\sqrt{5.53} & -0.65\sqrt{341.2} \\ 0.65\sqrt{5.53} & 0.76\sqrt{341.2} \end{bmatrix}. \tag{1-2}$$

You may check that $\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{S}^{\frac{T}{2}} = \boldsymbol{S}$

Exercise 1.3 Lecture 3

a) **Non-linear error propagation:** When estimating multiple probabilities $p_1, ..., p_k$ (with $\sum p_i = 1$), the multivariate logit transformation, with

$$p_i(\boldsymbol{\theta}) = \frac{e^{\theta_i}}{1 + \sum_{i=1}^{k-1} e^{\theta_i}}; \quad i \neq k,$$

and $p_k = 1 - \sum_{i=1}^{k-1} p_i$, is often used. The transformation imply that for any $\boldsymbol{\theta} \in \mathbb{R}^{k-1}$, then $p_i \in (0, 1)$.
In estimation it turns out that, under general condistions, it is resonable to assume that estimates ($\hat{\boldsymbol{\theta}}$) of $\boldsymbol{\theta}$ follow a multivariate normal distribution.

Now assume that $\boldsymbol{\theta} \sim N_2(\hat{\boldsymbol{\theta}}, \sigma^2 \boldsymbol{I})$. With $\hat{p}_i(\boldsymbol{\theta})$ as above, find approximate values of the mean and variance of

$$\boldsymbol{p}(\boldsymbol{\theta}) = \begin{bmatrix} p_1(\theta_1, \theta_2) \\ p_2(\theta_1, \theta_2) \\ p_3(\theta_1, \theta_2) \end{bmatrix}$$

using the approximative non-linear error propagation. It is enough to write up the Jacobian and the solution in terms of the Jacobian, as a function of p_i.

Solution

We need the Jacobian, and for that we need the derivatives

$$\frac{\partial p_i(\theta_1, \theta_2)}{\partial \theta_j}; j = \{1, 2\}, i = \{1, 2, 3\}$$

we get

$$\begin{aligned} \left.\frac{\partial p_1(\theta_1, \theta_2)}{\partial \theta_1}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} &= p_1(\hat{\boldsymbol{\theta}}) - \frac{e^{2\hat{\theta}_1}}{\left(1 + \sum_{i=1}^{2} e^{\hat{\theta}_i}\right)^2} = p_1(\hat{\boldsymbol{\theta}})(1 - p_1(\hat{\boldsymbol{\theta}})) \\ \left.\frac{\partial p_1(\theta_1, \theta_2)}{\partial \theta_2}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} &= -\frac{e^{\hat{\theta}_1} e^{\hat{\theta}_2}}{\left(1 + \sum_{i=1}^{2} e^{\hat{\theta}_i}\right)^2} = -p_1(\hat{\boldsymbol{\theta}}) p_2(\hat{\boldsymbol{\theta}}) \end{aligned}$$

by similar arguments we get $\frac{\partial p_2}{\partial \theta_2} = p_2(1 - p_2)$ and $\frac{\partial p_2}{\partial \theta_1} = -p_1 p_2$. We can write p_3 as

$$p_3(\boldsymbol{\theta}) = \frac{1}{1 + \sum_{i=1}^{2} e^{\hat{\theta}_i}}$$

with the drivative

$$\left.\frac{\partial p_3(\theta_1, \theta_2)}{\partial \theta_i}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = -p_i p_3.$$

and the Jacobian is

$$\boldsymbol{J}_p(\hat{\boldsymbol{\theta}}) = \begin{bmatrix} p_1(1 - p_1) & -p_1 p_2 \\ -p_1 p_2 & p_2(1 - p_2) \\ -p_1 p_3 & -p_2 p_3 \end{bmatrix}$$

and hence

$$V[\boldsymbol{p}(\boldsymbol{\theta})] \approx \sigma^2 \boldsymbol{J}_p(\hat{\boldsymbol{\theta}}) \boldsymbol{J}_p(\hat{\boldsymbol{\theta}})^T.$$

b) **Non-linear error propagation:** With $\hat{\boldsymbol{\theta}} = \mathbf{0}$ expplicitly write the approximative variance.

Solution

In this case we have $p_i = \frac{1}{3}$ and the Jacobian become

$$\boldsymbol{J}_p(\hat{\boldsymbol{\theta}}) = \frac{1}{9}\begin{bmatrix} 2 & -1 \\ -1 & 2 \\ -1 & -1 \end{bmatrix} \tag{1-3}$$

and by direct matrix multiplication we get

$$V[\boldsymbol{p}(\boldsymbol{\theta})] \approx \frac{\sigma^2}{81}\begin{bmatrix} 5 & -4 & -1 \\ -4 & 5 & -1 \\ -1 & -1 & 2 \end{bmatrix} \tag{1-4}$$

c) **Multivariate normal distribution:** Let $\boldsymbol{Z} \sim N_2(\mathbf{0}, \boldsymbol{I})$ be a standard normal random variable. Also let the matrix $\boldsymbol{A}$ be given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

What is the distribution of $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{Z}$, and can you calculate the value of the pdf at $\boldsymbol{Y} = [1, 1, 1]$?

Solution

Since $\boldsymbol{Y}$ is a linear combination of normal random variables it will also be normal, with mean $\mathbf{0}$ and variance $\boldsymbol{A}\boldsymbol{A}^T$, i.e.

$$V[\boldsymbol{Y}] = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \tag{1-5}$$

Since $\boldsymbol{A}$ hane rank 2, there is no inverse of the variance, hence the pdf cannot be calculated. It can be calculated for all pairs $((y_i, y_j), i \neq j)$ in the presented case.

Exercise 1.4 Lecture 5

a) **Multivariate normal distribution:** Assume that $\boldsymbol{Y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu} = \bar{\boldsymbol{y}}$, and $\boldsymbol{\Sigma} = \boldsymbol{S}$, with $\bar{\boldsymbol{y}}$ and $\boldsymbol{S}$ as in Exercise 1, and consider the random variable

$$Q = (\boldsymbol{Y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{Y} - \boldsymbol{\mu}),$$

what is the expected value and the variance of Q?

Solution

We actually do not need to know the exact values of $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$, we can simply use Corollary 9.18, which teels us that $Q \sim \chi^2(2)$ (the 2 degrees of freedom comes from $\boldsymbol{Y}$ being in $\mathbb{R}^2$). Now use Theorem 2.83 to find that $E[Q] = 2$ and V[Q]=4.

b) **Orthogonal projections:** Consider the 3 orthogonal projection matrices

$$\boldsymbol{H}_1 = \frac{1}{6}\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}; \quad \boldsymbol{H}_2 = \frac{1}{6}\begin{bmatrix} 4 & -2 & -2 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}; \quad \boldsymbol{H}_3 = \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & -3 \\ 0 & -3 & 3 \end{bmatrix}.$$

With $\boldsymbol{Z} \sim N_3(\boldsymbol{0}, \boldsymbol{I})$, what is the expectation of the random variables

$$Q_i = \boldsymbol{Z}^T \boldsymbol{H}_i \boldsymbol{Z}?$$

Hint: Check the sum: $\boldsymbol{H}_1 + \boldsymbol{H}_2 + \boldsymbol{H}_3$.

Solution

It is easy to verify that $\boldsymbol{H}_1 + \boldsymbol{H}_2 + \boldsymbol{H}_3 = \boldsymbol{I}$, and hence by Cochran's theorem we have $\boldsymbol{Z}^T \boldsymbol{H}_i \boldsymbol{Z} \sim \chi^2(Trace(\boldsymbol{H}_i))$ and hence $Q_i \sim \chi^2(1)$, and therefore $E[Q_i] = 1$ for all i.

c) Say you have 3 item (item 1, 2, and 3). Now items are put on a scale in the following order (and the weight is denoted as y_i)

1. Item 1 is put on the scale.
2. Item 1 and 2 is put on the scale.

3. Item 1 and 3 is put on the scale.
4. Item 2 and 3 is put on the scale.

Now consider the following model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}; \quad \boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

here $\boldsymbol{\beta} \in \mathbb{R}^3$. Write the design matrix ($\boldsymbol{X}$) for the following two interpretations of $\boldsymbol{\beta}$.

1. The estimate of β_i is the weight of item i, i.e. $\boldsymbol{\beta} = [\mu_1, \quad , \mu_2, \quad \mu_3]$.
2. Set $\mu_i = \mu + \delta_i$, and, using the constraint $\sum \delta_i = 0$, let $\boldsymbol{\beta} = [\mu, \quad \delta_1, \quad \delta_2]$.

Using Python check that the parametrizations are equivalent (i.e. they have the same projection matrices).

Solution

In the first case the desing matrix is simply

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

For the second parametrization we have ($\delta_3 = -\delta_1 - \delta_2$)

$$\begin{aligned} y_1 &= \mu + \delta_1 + \epsilon_1 \\ y_2 &= 2\mu + \delta_1 + \delta_2 + \epsilon_2 \\ y_3 &= 2\mu + \delta_1 + \delta_3 + \epsilon_3 = 2\mu + \delta_1 - \delta_1 - \delta_2 + \epsilon_3 = 2\mu - \delta_2 + \epsilon_3 \\ y_4 &= 2\mu + \delta_2 + \delta_3 + \epsilon_4 = 2\mu + \delta_2 - \delta_1 - \delta_2 + \epsilon_4 = 2\mu - \delta_1 + \epsilon_4 \end{aligned}$$

and hence the design matrix is

$$\boldsymbol{X}_2 = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 1 \\ 2 & 0 & -1 \\ 2 & -1 & 0 \end{bmatrix}$$

The following Pythin code show that the two model are equivalent.

```
X1 = np.array([[1,1,1,0],[0,1,0,1],[0,0,1,1]]).T
X1


array([[1, 0, 0],
       [1, 1, 0],
       [1, 0, 1],
       [0, 1, 1]])


H1 = X1 @ np.linalg.inv(X1.T@X1) @ X1.T
X2 = np.array([[1,2,2,2],[1,1,0,-1],[0,1,-1,0]]).T
X2


array([[ 1,  1,  0],
       [ 2,  1,  1],
       [ 2,  0, -1],
       [ 2, -1,  0]])


H2 = X2 @ np.linalg.inv(X2.T@X2) @ X2.T
Diff = H1-H2
np.abs(Diff).max()


np.float64(2.220446049250313e-16)
```

Hence the maximum difference is very small (implying the projection metrices are the same).

Exercise 1.5 Lecture 6

a) For the sleep medicine data in Example 3.27:

- Find the design matrix ($\boldsymbol{X}$) for the model.
- Use the design matrix to find the projection matrix ($\boldsymbol{H}$)

Further use Python to calculate:

- $\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y}$
- $\boldsymbol{y}^T(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{y}$

- The observed F-test statistics and the $p - value$ for the hypothesis that there is no effect of the sleep medicine
- compare p-value and F-test statistic with the p-value in Example 3.27.

Solution

The design matrix is a vector of ones with 10 elements, and the projection matrix is

$$\boldsymbol{H} = \mathbf{1}(\mathbf{1}^T\mathbf{1})^{-1}\mathbf{1}^T = \frac{1}{10}\boldsymbol{E} \tag{1-6}$$

where $\boldsymbol{E}$ is a matrix of ones. For the concrete calculations in Python these can be done by:

```
y = np.array([1.2, 2.4, 1.3, 1.3, 0.9, 1.0, 1.8, 0.8, 4.6, 1.4])
X = np.repeat(1,10)
H = 1/10 * np.outer(X,X)
y.T @ H @ y
```

```
np.float64(27.888999999999996)
```

```
r = (np.identity(10) -H) @ y
r.T @ r
```

```
np.float64(11.500999999999998)
```

```
Fobs = y.T @ H @ y/ (r.T @ r /9)
Fobs
```

```
np.float64(21.824276149900008)
```

```
p_value = 1 - stats.f.cdf(Fobs,1,9)
p_value
```

```
np.float64(0.0011658764685528178)
```

Comparing with example 3.27 we see that the p-values are (almost) identical. For comparing the test statistics we should square the t-test statistics:

```
4.6716**2
```

```
21.823846559999996
```

which is equal the observed F-test statistics.

b) What is the estimate of the residual variance for the model?

Solution

This is the sum of squared deviation divided by $n - 1 = 9$. i.e.

```
r.T @ r/9
```

```
np.float64(1.2778888888888886)
```

c) The data is not collected as a time series, and hence checking lag-1 autocorrelation does not make a lot of sense, however for the sake of illstration, test if the lag-1 autocorrelation can be assumed to be zero anyway.

Solution

We may use equation (9.129) and compare the result with quantiles of a normal distribution with mean zero and variance $1/\sqrt{n}$, hence we should compare the observed correlation with $2/\sqrt{n}$

```
np.corrcoef(r[np.arange(0,9)],r[np.arange(1,10)])[0,1]
```

```
np.float64(-0.2848957708731395)
```

```
2/np.sqrt(len(r))
```

```
np.float64(0.6324555320336759)
```

hence no evidence of auto correlation. We could also use direct Python calculations to find the correlation

```
r[np.arange(0,9)].T@r[np.arange(1,10)]/(r.T@r)
```

```
np.float64(-0.2796191635509956)
```

which gives an almost identical result.

Exercise 1.6 Lecture 7

a) Consider the data in the nutrient study in Example 3.55, and consider the model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}; \quad \boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I}).$$

Set up the design matrix $\boldsymbol{X}_1$ when the interpretation of the parameters ($\boldsymbol{\beta}$) is the mean/average in each of the two groups.

Solution

In this case the columns of the design matrix is simply the group indicator, i.e.

$$\boldsymbol{X}_1 = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}, \tag{1-7}$$

where $\boldsymbol{1}$ and $\boldsymbol{0}$ are vectors of zeros and ones of appropriate dimensions.

b) Now consider two alternative parametrizations

$$\boldsymbol{X}_2 = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \end{bmatrix}, \quad \boldsymbol{X}_3 = \begin{bmatrix} \mathbf{1} & \frac{1}{2}\mathbf{1} \\ \mathbf{1} & -\frac{1}{2}\mathbf{1} \end{bmatrix}.$$

Using Python show that the three parametrizations are equvivalent, and give an inpterpretaion of the paremeters (you may consider $(\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T$). Hint: For setting up the design matrix you may use (e.g. for $\boldsymbol{X}_2$):

```
X2 = np.array([np.repeat([1,1],n),np.repeat([0,1],n)]).T
```

Solution

The design matrices can be set up by

```
n = 9
X1 = np.array([np.repeat([1,0],n),np.repeat([0,1],n)]).T
X2 = np.array([np.repeat([1,1],n),np.repeat([0,1],n)]).T
X3 = np.array([np.repeat([1,1],n),np.repeat([0.5,-0.5],n)]).T
```

and the projection matrices can be calculated by

```
H1 = X1 @ np.linalg.inv(X1.T@X1) @ X1.T
H2 =X2 @ np.linalg.inv(X2.T@X2) @ X2.T
H3 =X3 @ np.linalg.inv(X3.T@X3) @ X3.T
```

to see that these are identical we calculate the maximum absolute difference

```
np.abs(H1-H2).max()
```

```
np.float64(0.0)
```

```
np.abs(H1-H3).max()
```

```
np.float64(0.0)
```

for the interpretation we consider

```
np.round(np.linalg.inv(X1.T@X1) @ X1.T,4)


array([[0.1111, 0.1111, 0.1111, 0.1111, 0.1111, 0.1111, 0.1111, 0.1111,
        0.1111, 0.    , 0.    , 0.    , 0.    , 0.    , 0.    , 0.    ,
        0.    , 0.    ],
       [0.    , 0.    , 0.    , 0.    , 0.    , 0.    , 0.    , 0.    ,
        0.    , 0.1111, 0.1111, 0.1111, 0.1111, 0.1111, 0.1111, 0.1111,
        0.1111, 0.1111]])
```

```
np.round(np.linalg.inv(X2.T@X2) @ X2.T,4)


array([[ 0.1111,  0.1111,  0.1111,  0.1111,  0.1111,  0.1111,  0.1111,
         0.1111,  0.1111,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
         0.    ,  0.    ,  0.    ,  0.    ],
       [-0.1111, -0.1111, -0.1111, -0.1111, -0.1111, -0.1111, -0.1111,
        -0.1111, -0.1111,  0.1111,  0.1111,  0.1111,  0.1111,  0.1111,
         0.1111,  0.1111,  0.1111,  0.1111]])
```

```
np.round(np.linalg.inv(X3.T@X3) @ X3.T,4)


array([[ 0.0556,  0.0556,  0.0556,  0.0556,  0.0556,  0.0556,  0.0556,
         0.0556,  0.0556,  0.0556,  0.0556,  0.0556,  0.0556,  0.0556,
         0.0556,  0.0556,  0.0556,  0.0556],
       [ 0.1111,  0.1111,  0.1111,  0.1111,  0.1111,  0.1111,  0.1111,
         0.1111,  0.1111, -0.1111, -0.1111, -0.1111, -0.1111, -0.1111,
        -0.1111, -0.1111, -0.1111, -0.1111]])
```

in the two first cases we see that $\hat{\beta}_1$ is the average in group 1, while in the third case $\hat{\beta}_1$ is the average of all observation. In the first case $\hat{\beta}_2$ is the average in group 2, while in the second and third case $\hat{\beta}_2$ is the difference between the averages in the two groups.

c) Use Theorem 9.29 and one of the above parametrizations to test if it can be assumed that the mean value is each group is the same (this should be done in Python). Does it matter which paramtrization you use for the test?

Solution

We need the projection matrix for the null hypothesis (i.e. that the design matrix is a vector of ones). Further the data should be included

```
y = np.array([7.53, 7.48, 8.08, 8.09, 10.15, 8.4, 10.88, 6.13, 7.9,9.21,
              11.51, 12.79, 11.85, 9.97, 8.79, 9.69, 9.68, 9.19]).T
X0 = np.array([np.repeat(1,18)])
H0 = np.outer(X0,X0)/18
F_obs = y.T @ (H1 - H0) @ y/( y.T @ (np.identity(18) - H1) @ y/16)
F_obs


np.float64(9.054884393867736)


1-stats.f.cdf(F_obs,1,16)


np.float64(0.00831943876505914)
```

It will not matter which parametrization is used as we have just shown that they are equvivalent. The p-value is the same as observed in Example 3.55 (there is a misprint in the Python code of the example though).

Exercise 1.7 Lecture 9

This exercise is about modelling of CO_2 concentrations in a room (and adapted from the 2023 June exam), the data for the exercise should be downloaded from the website, and can be read into Python by

```
dat = pd.read_csv("co2_data.csv", sep=';')
```

the data contain hourly measurement of CO_2 concentrations for a full week. The columns in the dataset are

- `Hour`: hours since the start of measurements
- `Day`: Days since the start of the measurements
- `CO2C`: The measured CO_2 concentration [ppm]

Now consider the following model

$$Y_i = \beta_0 + \sum_{j=1}^{q} \sin\left(2j\pi\frac{h_i}{24}\right)\beta_{1,q} + \cos\left(2j\pi\frac{h_i}{24}\right)\beta_{2,q} + \epsilon_i; \quad \epsilon_i \sim N(0,\sigma^2) \quad \text{and iid,} \tag{1-8}$$

where h_i is the hours since the begining of the data, and q is an order to be determined (i.e. how may basis functions should be included),

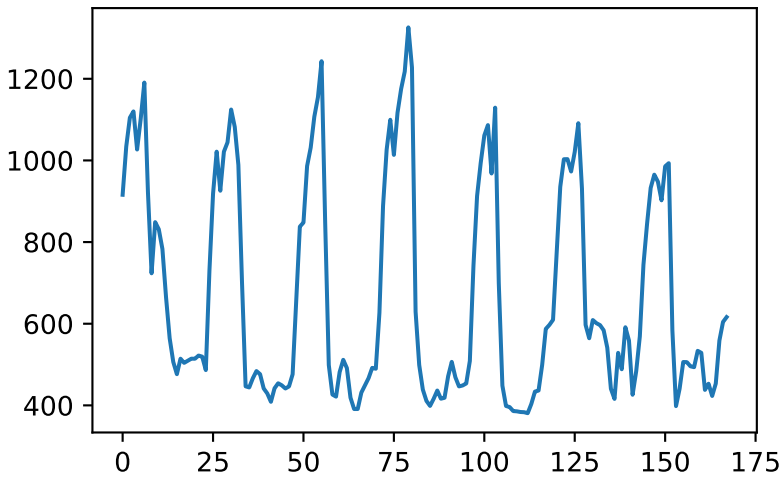
a) Plot the CO_2-concentration as a function of time (i.e. `Hour`), and comment on why the proposed model seems reasonable.

Solution

```
dat.head()
```

```
   Hour  Day         CO2C
0     0    1   915.885417
1     1    1  1034.166667
2     2    1  1104.687500
3     3    1  1120.052083
4     4    1  1026.718750
```

```
plt.plot(dat["Hour"],dat["CO2C"])
```



The 7 days are quite clear and hence it seems reasinable to have a periodic basis functions, starting from a period of 1 day.

b) Starting from $q = 3$, and using significance level $\alpha = 0.05$, test how many sine/cosine pairs should be included in the model using Type I partioning.

Solution

The full model is define by the design matrix (we show row i, and use the notation $\tilde{h}_i = \frac{2h_i\pi}{24}$)

$$\boldsymbol{X}_{3,i} = \begin{bmatrix} 1 & \sin\left(\tilde{h}_i\right) & \cos\left(\tilde{h}_i\right) & \sin\left(2\tilde{h}_i\right) & cos\left(2\tilde{h}_i\right) & \sin\left(3\tilde{h}_i\right) & \cos\left(3\tilde{h}_i\right) \end{bmatrix}, \quad \text{(1-9)}$$

and $\boldsymbol{\beta} = [\beta_0 \quad \beta_{1,1} \quad \beta_{2,1} \quad \beta_{1,2} \quad \beta_{2,2} \quad \beta_{1,3} \quad \beta_{2,3}]$. For two sets we get

$$\boldsymbol{X}_{2,i} = \begin{bmatrix} \beta_0 & \sin\left(\tilde{h}_i\right) & \cos\left(\tilde{h}_i\right) & \sin\left(2\tilde{h}_i\right) & \cos\left(2\tilde{h}_i\right) \end{bmatrix}, \quad \text{(1-10)}$$

and $\boldsymbol{\beta} = [\beta_0 \quad \beta_{1,1} \quad \beta_{2,1} \quad \beta_{1,2} \quad \beta_{2,2}]$, for 1 set it is

$$\boldsymbol{X}_{1,i} = \begin{bmatrix} 1 & \sin\left(\tilde{h}_i\right) & \cos\left(\tilde{h}_i\right) \end{bmatrix} \quad \text{(1-11)}$$

and $\boldsymbol{\beta} = [\beta_0 \quad \beta_{1,1} \quad \beta_{2,1}]$, and finally one could include a model with only the mean value

$$\boldsymbol{X}_{0,i} = 1 \quad \text{(1-12)}$$

and $\boldsymbol{\beta} = \beta_0$ (this will not be included in the analysis below).

The design matrices can be implemented by

```
X3 = np.array([np.repeat(1,168), np.sin(2*np.pi*dat["Hour"]/24),
                                 np.cos(2*np.pi*dat["Hour"]/24),
 np.sin(2*2*np.pi*dat["Hour"]/24),
                                 np.cos(2*2*np.pi*dat["Hour"]/24),
 np.sin(2*3*np.pi*dat["Hour"]/24),
                                 np.cos(2*3*np.pi*dat["Hour"]/24)]).T

X2 = np.array([np.repeat(1,168), np.sin(2*np.pi*dat["Hour"]/24),
                                 np.cos(2*np.pi*dat["Hour"]/24),
 np.sin(2*2*np.pi*dat["Hour"]/24),
                                 np.cos(2*2*np.pi*dat["Hour"]/24)]).T

X1 = np.array([np.repeat(1,168), np.sin(2*np.pi*dat["Hour"]/24),
                                 np.cos(2*np.pi*dat["Hour"]/24)]).T
```

The resulting projection matrices are

```
H3 = X3 @ np.linalg.inv(X3.T@X3)@X3.T
H2 = X2 @ np.linalg.inv(X2.T@X2)@X2.T
H1 = X1 @ np.linalg.inv(X1.T@X1)@X1.T
```

The test statistics can be calculated by Theorem 9.31 (mind the misprint though).

```
y = dat["CO2C"]
I = np.identity(168)
F3 = (y.T@(H3-H2)@y/2)/(y.T@(I-H3)@y/(168-7))
F2 = (y.T@(H2-H1)@y/2)/(y.T@(I-H3)@y/(168-7))
F3
```

```
np.float64(1.6855368888098363)
```

```
F2
```

```
np.float64(47.81161818916731)
```

Hence we see small test statictics when remoiving the last two column, while removing the additional two columns result in a large test statistics, suggesting that we should remove the last two column but include the rest of the columns. This can be quantified by p-values

```
1-stats.f.cdf(F3,2,168-7)
```

```
np.float64(0.18859880822756137)
```

```
1-stats.f.cdf(F2,2,168-7)
```

```
np.float64(1.1102230246251565e-16)
```

Confirming that the last two columns can be omitted.

c) If the parametrization is orthogonal, then the type I and type III test are equivalent (see Theorem 9.44). In the case studied here, is the Type I and

Type III test equivalent?

Solution

We have to check if $\boldsymbol{X}_3^T \boldsymbol{X}_3$ is an digonal matrix. Using Python we get

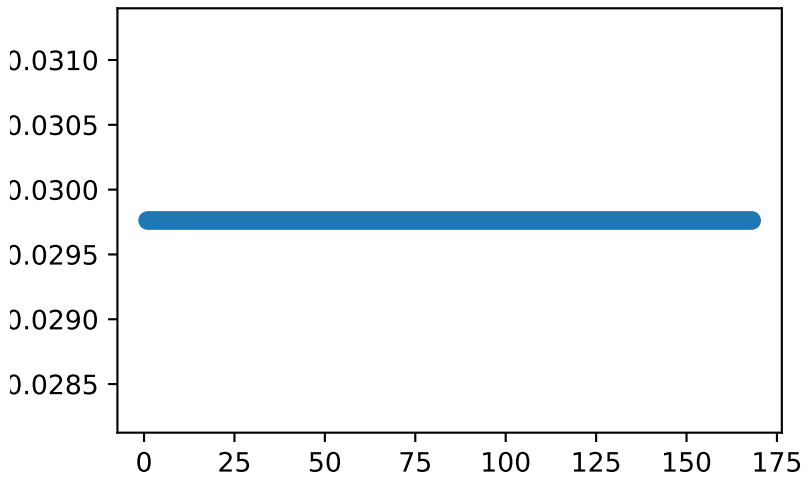
```
np.round(X3.T@X3,6)


array([[168.,   0.,  -0.,   0.,  -0.,   0.,  -0.],
       [  0.,  84.,   0.,  -0.,   0.,  -0.,  -0.],
       [ -0.,   0.,  84.,   0.,  -0.,   0.,   0.],
       [  0.,  -0.,   0.,  84.,   0.,  -0.,  -0.],
       [ -0.,   0.,  -0.,   0.,  84.,   0.,   0.],
       [  0.,  -0.,   0.,  -0.,   0.,  84.,  -0.],
       [ -0.,  -0.,   0.,  -0.,   0.,  -0.,  84.]])
```

hence the design is orthogonal and the Type I and Type III partioning will be identical.

d) Make a leverage plot (similar to Example 9.42) for the chosen model, and comment on the effect it will have on standardized and studentized residuals.
(Hint: you may extract diagonal elements of a matrix (`A`) by `A.diagonal(0)`).

Solution

```
h2 = H2.diagonal(0)
plt.scatter(np.linspace(1,168,168),h2)
```



The leverage is seen to be constatnt, this imply that the raw resuduals will have constant variance also, and that the only (impottant) correction for the studentized residuals are related to difference in magnitude of residuals (i.e. not the levarage).

Exercise 1.8 Lecture 10

This exercise use the data from Example 8.26, and you can read the data into Python by

```
dat = pd.read_csv("cars.csv", sep=';')
dat
```

```
    car  tire  fuel
1     1     1  22.5
2     1     2  21.5
3     1     3  22.2
4     2     1  24.3
5     2     2  21.3
6     2     3  21.9
7     3     1  24.9
8     3     2  23.9
```

```
9     3     3  21.7
10    4     1  22.4
11    4     2  18.4
12    4     3  17.9
```

In this exercise we only consider `car` as the explanatory variable.

a) Explicitly write down the design matrix when the interpretation of the parameters is the mean within each group. Also set up the design matrix when the interpretation is as in Chapter 8 (i.e. $y_{ij} = \mu + \alpha_i + \epsilon_{ij}$, and $\sum_i n_i \alpha_i = 0$).

Solution

In the first case we have

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{1-13}$$

In the second case we have (see eq. (9-241), and note that $n_i = n_j$)

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} \end{bmatrix} \tag{1-14}$$

b) Now assume that car 1 is considered a reference, and we hence want a parametrization where the first parameter is the average for car 1, and the other paramters is the difference between the consumption for car 1 and car i, $i \in \{2,3,4\}$. Again set up the design matrix.

Solution

In the first case we have

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{1-15}$$

c) Using Python find the parameter estimates for the last model and test the three hypotheses ($\beta_2 = 0$, $\beta_3 = 0$, and $\beta_4 = 0$), do this both with and without the Bonferroni correction.
Compare with the result of using

```
dat["car"] = pd.Categorical(dat["car"])
fit = smf.ols(formula = "fuel ~ car", data = dat).fit()
fit.summary(slim=True)
```

Solution

Build the design matrix

```
X = np.column_stack([np.repeat(1,12),np.array([0,0,0,1,1,1,0,0,0,0,0,0]),
np.array([0,0,0,0,0,0,1,1,1,0,0,0]),np.array([0,0,0,0,0,0,0,0,0,1,1,1])])
X
```

```
array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 1, 0, 0],
       [1, 1, 0, 0],
       [1, 1, 0, 0],
       [1, 0, 1, 0],
       [1, 0, 1, 0],
       [1, 0, 1, 0],
       [1, 0, 0, 1],
       [1, 0, 0, 1],
       [1, 0, 0, 1]])
```

Interpretation of parameters

```
np.linalg.inv(X.T@X)@X.T
```

```
array([[ 0.33333333,  0.33333333,  0.33333333,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ],
       [-0.33333333, -0.33333333, -0.33333333,  0.33333333,  0.33333333,
         0.33333333,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ],
       [-0.33333333, -0.33333333, -0.33333333,  0.        ,  0.        ,
         0.        ,  0.33333333,  0.33333333,  0.33333333,  0.        ,
         0.        ,  0.        ],
       [-0.33333333, -0.33333333, -0.33333333,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.33333333,
         0.33333333,  0.33333333]])
```

which is the average difference between car 1 and car i, $i \in \{2,3,4\}$, the parameter estimates and the estimate of the residual variance is calculated by

```
beta = np.linalg.inv(X.T@X)@X.T@dat["fuel"]
H = X@np.linalg.inv(X.T@X)@X.T
I = np.identity(12)
sigma_sq = dat["fuel"].T @(I-H)@dat["fuel"]/(12-4)
print(beta)
```

```
[22.06666667  0.43333333  1.43333333 -2.5       ]
```

```
print(sigma_sq)
```

```
2.886666666666762
```

The standard errors, and the test statistics are

```
se = np.sqrt(np.diag(sigma_sq *np.linalg.inv(X.T@X)))
print(se)
```

```
[0.98092926 1.38724347 1.38724347 1.38724347]
```

```
print(beta/se)
```

```
[22.49567575  0.31237007  1.03322406 -1.80213499]
```

these (i.e. the three last ones) should be compared with the critical value

```
stats.t.ppf(0.975,12-4)
```

```
np.float64(2.306004135204166)
```

hence none of them are significant on the usual level. Using the Bonferroine correction we should use the critical value (we make 3 comparisons)

```
stats.t.ppf(1-0.05/2/3,12-4)
```

```
np.float64(3.015761836887169)
```

and they are ofcourse still not significant.

Comparing with `smf.ols` we get

```
dat["car"] = pd.Categorical(dat["car"])
fit = smf.ols(formula = "fuel ~ car", data = dat).fit()
fit.summary(slim=True)


<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   fuel   R-squared:                       0.522
Model:                            OLS   Adj. R-squared:                  0.342
No. Observations:                  12   F-statistic:                       2.907
Covariance Type:            nonrobust   Prob (F-statistic):               0.101
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     22.0667      0.981     22.496      0.000      19.805      24.329
car[T.2]       0.4333      1.387      0.312      0.763      -2.766       3.632
car[T.3]       1.4333      1.387      1.033      0.332      -1.766       4.632
car[T.4]      -2.5000      1.387     -1.802      0.109      -5.699       0.699
==============================================================================

"""
```

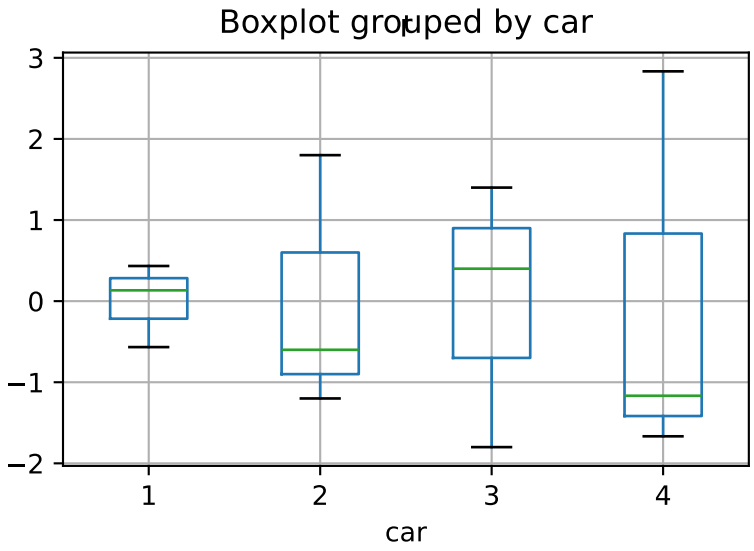
and we see that the parameters, and the test statistics are excactly the same.

d) Make a residual analysis of the model.

Solution

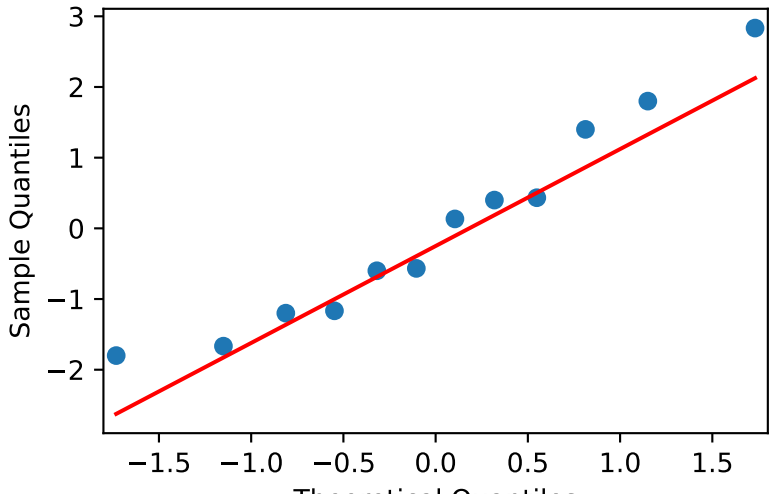
We can produce the residuals and a box-plot by

```
r = (I-H)@dat["fuel"]
dat["r"] = r
dat.boxplot(column="r",by="car")
```



the qq-plot can be done by

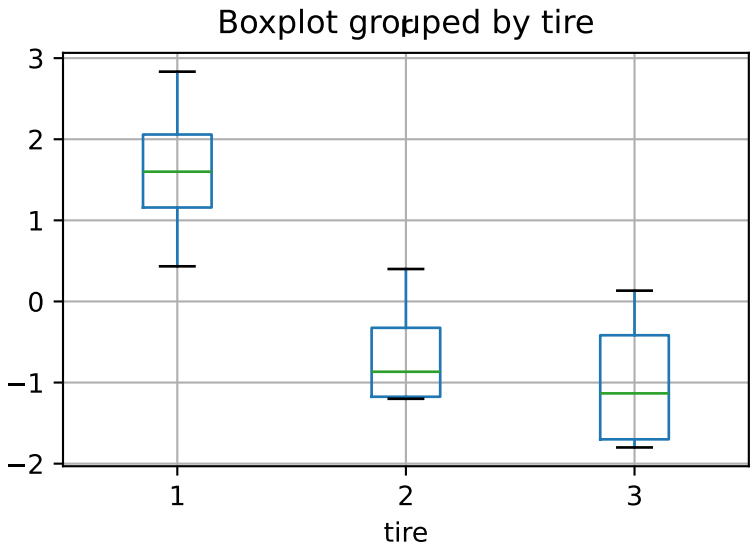
```
sm.qqplot(dat["r"],line="q",a=1/2)
```



The qq-plot seems fine and the box plot also seems fine in articular considering that only 3 points are used for each of the box plot.

Finally we could see the residuals as a function of tire

```
dat.boxplot(column="r",by="tire")
```



and it seems that the residuals are actually a function of tire, and hence that should be included in the model.