practical_exercise_6, Methods 3, 2021, autumn semester

Solution, Lau Møller Andersen

November 18, 2021

Exercises and objectives

1) Get acquainted with python learn some of the differences between it and R

REMEMBER: In your report, make sure to include code that can reproduce the answers requested in the exercises below (**MAKE A KNITTED VERSION**)
REMEMBER: All exercises should be done in *Python*

EXERCISE 1 Get acquainted with *Python* learn some of the differences between it and *R*

To make sure that *Python* runs within *R Markdown*, make sure you have the *reticulate* package installed `install.packages('reticulate')`
Also create a text file that is called *.Renviron* (remember the dot) placed in the folder where your *RProj* file is. It should have a single line: `RETICULATE_PYTHON=PATH` where `PATH` is the path to your *methods3* conda environment. Use the commands below to find the paths:

```
library(reticulate)
print(conda_list())
```

```
##           name                                        python
## 1   miniconda3                    /home/lau/miniconda3/bin/python
## 2     methods3      /home/lau/miniconda3/envs/methods3/bin/python
## 3          mne           /home/lau/miniconda3/envs/mne/bin/python
## 4     mne_0.17      /home/lau/miniconda3/envs/mne_0.17/bin/python
## 5 mne_func_sig  /home/lau/miniconda3/envs/mne_func_sig/bin/python
## 6       mnedev        /home/lau/miniconda3/envs/mnedev/bin/python
## 7     psychopy      /home/lau/miniconda3/envs/psychopy/bin/python
## 8    fslpython                /usr/local/fsl/fslpython/bin/python
## 9    fslpython /usr/local/fsl/fslpython/envs/fslpython/bin/python
```

Good to know about *Python* (in no particular order)

```
## assignment is done and only done with "=" (no arrows)
a = 2
# a <- 2 # results in a syntax error
## already assigned variables can be reassigned with basic arithmetic operations
a += 2
print(a)
```

```
## 4
```

```
a -= 1
print(a)
```

```
## 3
```

```
a *= 4
print(a)
```

```
## 12
```

```
a //= 2 # integer division
print(a)
```

```
## 6
```

```
a /= 2 # float  (numeric from R) division
print(a)
```

```
## 3.0
```

```
a **= 3 # exponentiation
print(a)

## lists (mutable)
```

```
## 27.0
```

```
a_list = [1, 2] # initiate a list (the square brackets) with the integers 1 and 2
b = a_list ## b now points to a_list, not to a new list with the integers 1 and 2

a_list.append(3) # add a new value to the end of the list
print(a_list)
```

```
## [1, 2, 3]
```

```
print(b) # make sure you understand this
```

```
## [1, 2, 3]
```

```
print(a_list[0]) # zero-indexing
```

```
## 1
```

```
print(a_list[1])
```

```
## 2
```

```
new_list = [0, 1, 2, 3, 4, 5] # slicing
print(new_list[0:3])
```

```
## [0, 1, 2]
```

```
for index in range(0, 5): # indentation (use tabulation) controls scope of control variables (no bracke
    if index == 0: # remember the colon
        value = 0
    else:
        value += index
    print(value)
```

```
## 0
## 1
## 3
## 6
## 10
```

```
this_is_true = True # logical values
this_is_false = False

# define functions using def
def fix_my_p_value(is_it_supposed_to_be_significant):
    if is_it_supposed_to_be_significant:
        p = 0.01
    else:
        p = 0.35
    return(p)

print(fix_my_p_value(True))

# importing packages (similar to library)
```

```
## 0.01
```

```
import numpy # methods of numpy can now be accessed as below
print(numpy.arange(1, 10)) # see the dot
```

```
## [1 2 3 4 5 6 7 8 9]
```

```
print(numpy.abs(-3))
```

```
## 3
```

```
import numpy as np # you can import them with another name than its default
print(np.cos(np.pi))
```

```
## -1.0
```

```
from numpy import pi, arange # or you can import specific methods
print(arange(1, 7))
```

```
## [1 2 3 4 5 6]
```

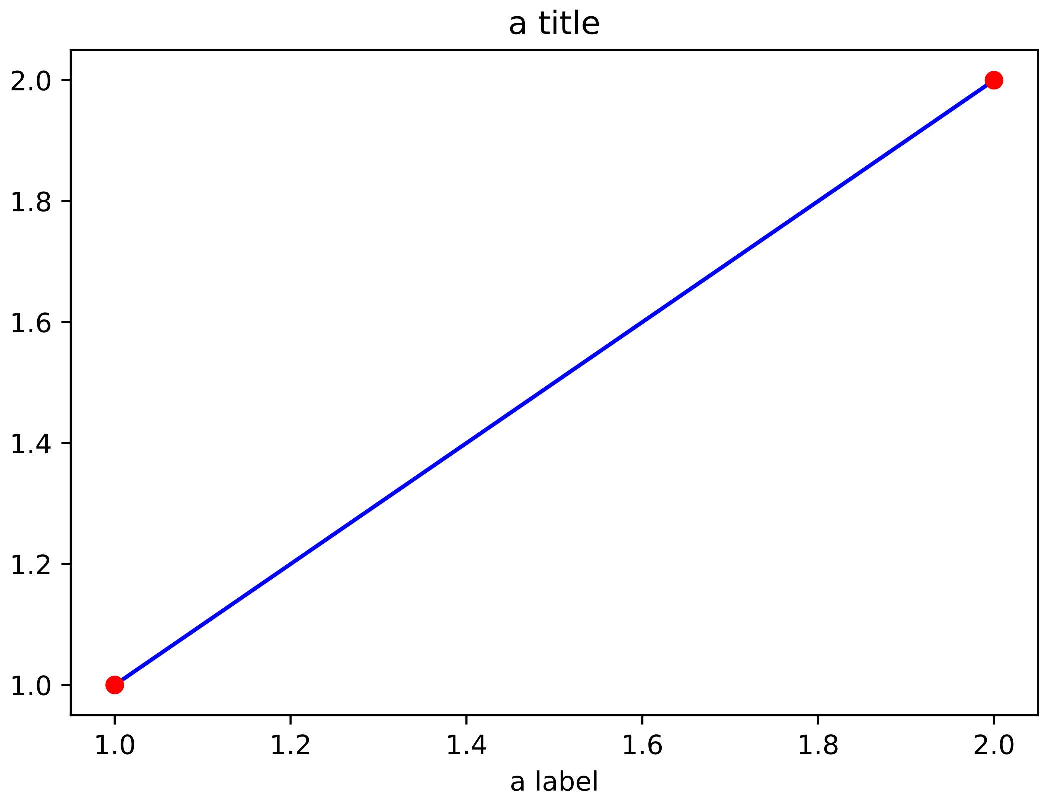
```
print(pi)
```

```
## 3.141592653589793
```

```
matrix = np.ones(shape=(5, 5)) # create a matrix of ones
identity = np.identity(5) # create an identity matrix (5x5)
identity[:, 2] = 5 # exchange everything in the second column with 5's

## no dots in names - dots indicate applying a method

import matplotlib.pyplot as plt
plt.figure() # create new figure
plt.plot([1, 2], [1, 2], 'b-') # plot a blue line
# plt.show() # show figure
plt.plot([2, 1], [2, 1], 'ro') # scatter plot (red)
# plt.show()
plt.xlabel('a label')
plt.title('a title')
plt.show()
```



1) Do a linear regression based on x, X and y below (y as the dependent variable) (Exercise 1.1)
   i. find $\hat{\beta}$ and $\hat{y}$ (@ is matrix multiplication)
   ii. plot a scatter plot of x, y and add a line based on $\hat{y}$ (use `plt.plot` after running `import matplotlib.pyplot as plt`)

2) Create a model matrix, X that estimates, $\hat{\beta}$ the means of the three sets of observations below, y_1, y_2, y_3 (Exercise 1.2)
   i. find $\hat{\beta}$ based on this X

   ii. Then create an X where the resulting $\hat{\beta}$ indicates: 1) the difference between the mean of y_1 and the mean of y_2; 2) the mean of y_2; 3) the difference between the mean of y_3 and the mean of ~~y_1~~y_2

3) Finally, find the F-value for this model (from exercise 1.2.ii) and its degrees of freedom. What is the p-value associated with it? (You can import the inverse of the cumulative probability density function `ppf` for F using `from scipy.stats import f` and then run `1 - f.ppf`)
   i. plot the probability density function `f.pdf` for the correct F-distribution and highlight the F-value that you found

   ii. how great a percentage of the area of the curve is to right of the highlighted point

```
# Exercise 1.1
import numpy as np
np.random.seed(7) # for reproducibility
```

```
x = np.arange(10)
y = 2 * x
y = y.astype(float)
n_samples = len(y)
y += np.random.normal(loc=0, scale=1, size=n_samples)

X = np.zeros(shape=(n_samples, 2))
X[:, 0] = x ** 0
X[:, 1] = x ** 1
```

SOLUTION 1.1

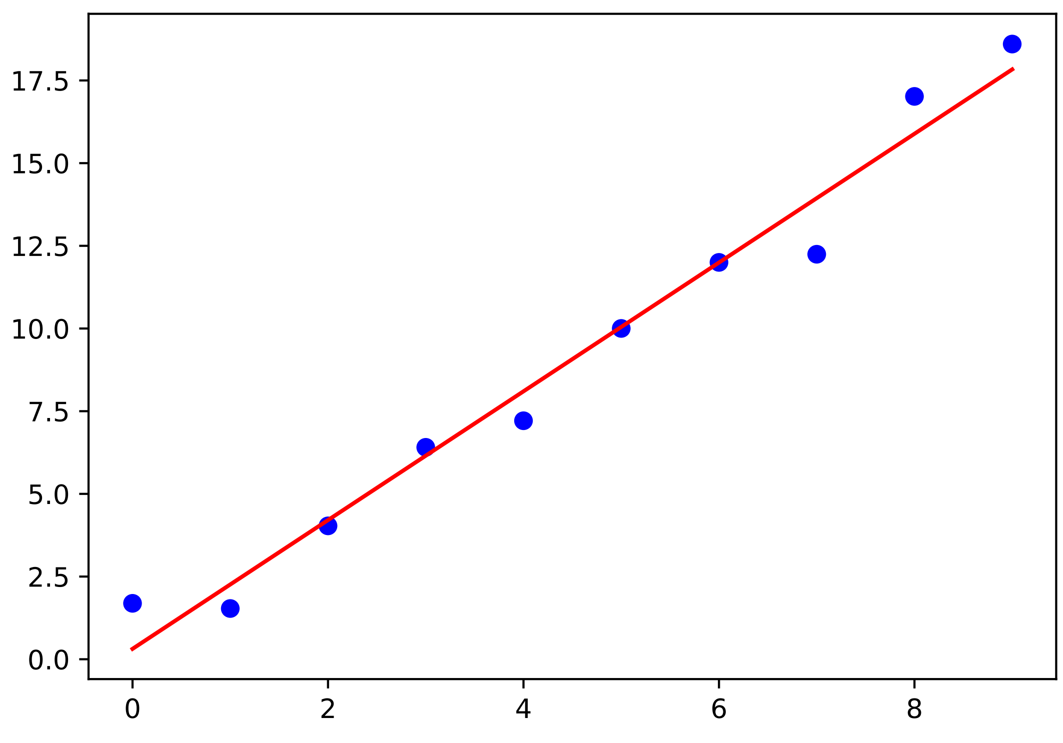
```
import matplotlib.pyplot as plt
beta_hat = np.linalg.inv(X.T @ X) @ X.T @ y
print(beta_hat)
```

```
## [0.31397994 1.94668466]
```

```
y_hat = beta_hat[0] + beta_hat[1] * x
print(y_hat)
```

```
## [ 0.31397994  2.2606646   4.20734926  6.15403392  8.10071858 10.04740325
##  11.99408791 13.94077257 15.88745723 17.83414189]
```

```
plt.figure()
plt.plot(x, y, 'bo')
plt.plot(x, y_hat, 'r')
plt.show()
```



```
# Exercise 1.2
y1 = np.array([3, 2, 7, 6, 9])
y2 = np.array([10, 4, 2, 1, -3])
y3 = np.array([15, -2, 0, 0, 3])
y = np.concatenate((y1, y2, y3))
```

SOLUTION 1.2

```
X = np.zeros(shape=(len(y), 3))
X[0:5,   0] = 1
X[5:10,  1] = 1
X[10:15, 2] = 1

beta_hat = np.linalg.inv(X.T @ X) @ X.T @ y
print(beta_hat)
```

```
## [5.4 2.8 3.2]
```

```
X2 = np.zeros(shape=(len(y), 3))
X2[0:5,   0] = 1
X2[:,     1] = 1
X2[10:15, 2] = 1

beta_hat2 = np.linalg.inv(X2.T @ X2) @ X2.T @ y
print(beta_hat2)
```

```
## [2.6 2.8 0.4]
```

SOLUTION 1.3

```
from scipy.stats import f
X3 = np.ones(shape=len(y))
beta_hat3 = 1 / (X3.T @ X3) * X3.T @ y
len_beta_hat3 = 1

y_hat2 = X2 @ beta_hat2
y_hat3 = X3 * beta_hat3

RSS2 = np.sum((y - y_hat2)**2)
RSS3 = np.sum((y - y_hat3)**2)

F = ((RSS3 - RSS2) / (len(beta_hat2) - len_beta_hat3)) / \
    ((RSS2) / (len(y) - len(beta_hat2)))

df1 = len(beta_hat2) - len_beta_hat3
df2 = len(y) - len(beta_hat2)

p = 1 - f.ppf(F, df1, df2)
print(F)
```

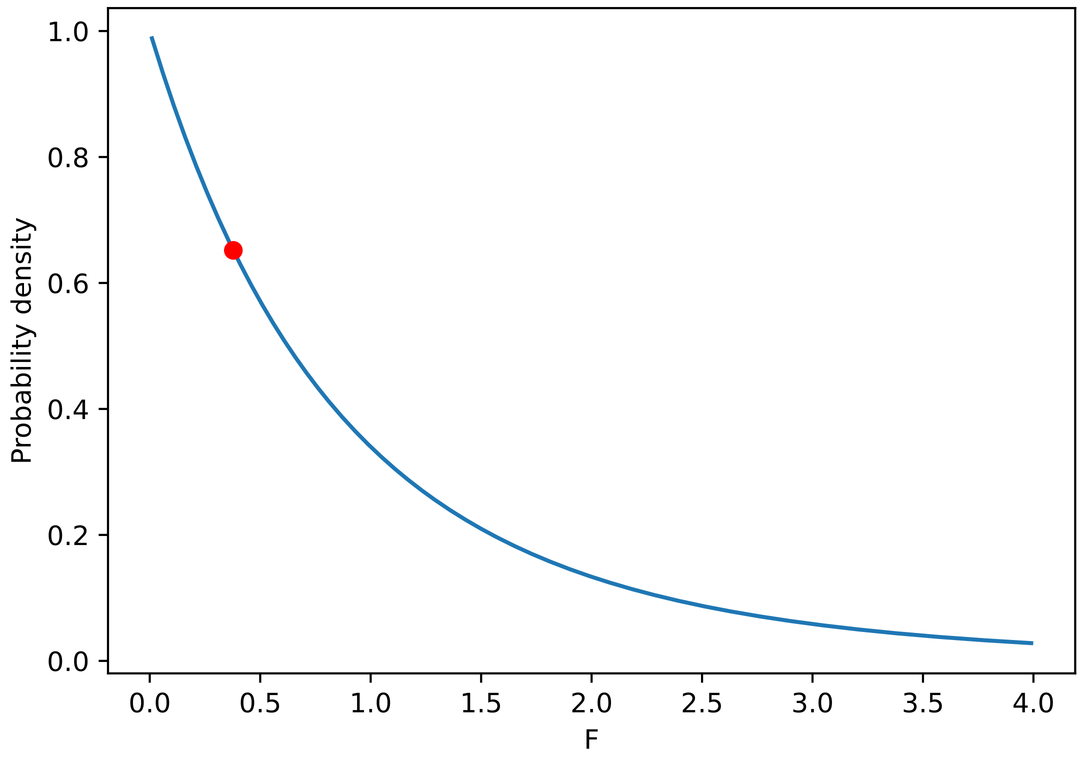
```
## 0.3783783783783765
```

```
print(p)
```

```
## 0.505233154030844
```

```
x_f = np.arange(0.01, 4, 0.01)
y_f = f.pdf(x_f, df1, df2)
```

```
plt.figure()
plt.plot(x_f, y_f)
plt.plot(F, f.pdf(F, df1, df2), 'ro')
plt.xlabel('F')
plt.ylabel('Probability density')
plt.show()
```


Probability density
F