Advanced topics (practice)

Programa de Doctorat en Economia, Universitat de Barcelona

David Moriña

Python practice 4

Exercise 1

- Find the solution of the following linear system

$$\begin{cases} 2x + 5y - 2z = 9 \\ -3x + 3y + 2z = 33 \\ 2x + 3y + 4z = 125 \end{cases}$$

Exercise 1

- Find the solution of the following linear system

```
import numpy as np
import numpy.linalg as nplin

A = np.array([[2, 5, -2], [-3, 3, 2], [2, 3, 4]])
b = np.array([9, 33, 125])
x = nplin.solve(A, b)
x
```

```
array([ 9.80327869,  6.47540984, 21.49180328])
```

```
np.allclose(np.dot(A, x), b)
```

```
True
```

Exercise 2

- Define the function of two variables $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2.5)^2$:

```
from scipy.optimize import minimize

def f(x):
  return (x[0] - 1)**2 + (x[1] - 2.5)**2
```

Exercise 2

- Find the minimum of the function $f(x_1, x_2)$

```
from scipy.optimize import minimize

x0 = [0, 0] # the initial guess
result = minimize(f, x0)
result
```

```
  message: Optimization terminated successfully.
  success: True
   status: 0
      fun: 1.968344227868139e-15
        x: [ 1.000e+00  2.500e+00]
      nit: 2
      jac: [-6.956e-08  4.211e-08]
 hess_inv: [[ 9.310e-01 -1.724e-01]
            [-1.724e-01  5.690e-01]]
     nfev: 9
     njev: 3
```

Exercise 3

- Bootstrap is commonly used to assess the variability of a statistic when its standard error has no closed form (or is complicated)
- Roughly, it resamples the original dataset with replacement to create pseudo-datasets that are similar to, but slightly perturbed from, the original dataset
- One example of a statistic for which the bootstrap is useful is the median
- Simulate a random sample ($n = 1000$) based on a normal distribution

```
import numpy as np
import statistics

x = np.random.normal(size=1000)
statistics.median(x)
```

Exercise 3

- Replicate the function 5,000 times, keeping each median in a vector and get the standard deviation of the medians. Get the time it takes for the function to run 5,000 times

```
import time

start_time = time.time()
sample_median = [None]*(5000)
for i in range(5000):
  sample_median[i] = bootst_median(i)

np.asarray(sample_median).std()
print("--- %s seconds ---" % (time.time() -
 ↪  start_time))
```

```
--- 0.6966159343719482 seconds ---
```

Exercise 3

- Run the previous function taking advantage of multiprocessing. Is it saving time?

```
import multiprocessing

start_time = time.time()
pool = multiprocessing.Pool()
results = pool.map(bootst_median, range(5000))
print("--- %s seconds ---" % (time.time() -
 ↪  start_time))
```

```
--- 0.162841796875 seconds ---
```

Exercise 3

- Bootstrap sampling can also be done directly using the *bootstrap* function from *scipy.stats*

```
from scipy.stats import bootstrap

#calculate 95% bootstrapped confidence interval for
 ↪  median (data has to be a sequence)
bootstrap_ci = bootstrap((x, ), np.median,
 ↪  confidence_level=0.95,
                          method='basic')
print(bootstrap_ci.confidence_interval)
```

```
ConfidenceInterval(low=-0.06593598685699506, high=0.082972(
```

Exercise 4

- Load the data file "penguins.csv" into the Python session

```
import pandas as pd

penguins =
 ↪  pd.read_csv("/home/dmorina/Insync/dmorina@ub.edu/OneDri
 ↪  Biz/Docència/UB/2023-2024/PyEcon/4. Econometric
 ↪  analysis in Python/practice/data/penguins.csv")
```

Exercise 4

- Represent the body mass of the penguins (variable "body_mass_g") against the variable "flipper_length_mm" with color depending on the species and two facets (one for males and one for females) and including the linear regression line for flipper_length_mm ~ body_mass_g

```
import seaborn as sns
import matplotlib.pyplot as plt

ax = sns.lmplot(data=penguins, x="flipper_length_mm",
 ↪  y="body_mass_g",
                 hue="species", col="sex", height=4,
 ↪  palette="deep")
ax.set(xlabel='Flipper length (mm)', ylabel='Body
 ↪  mass (g)')
plt.show()
# plt.savefig('penguins.png')
```