Basic data analysis with Python (practice)

Programa de Doctorat en Economia, Universitat de Barcelona

David Moriña

Python practice 2

Exercise 1

- Download the *csv* files in the *practice/data* folder and define the proper working directory
- Import all the files to the JupyterLab session

```
import pandas as pd
import os

os.chdir('/home/dmorina/Insync/dmorina@ub.edu/OneDrive
 ↪  Biz/Docència/UB/2023-2024/PyEcon/2. Intermediate
 ↪  Python/practice/data/')

df1 = pd.read_csv('Total.csv')
```

Exercise 2

- A company drills 9 wild-cat oil exploration wells, each with an estimated probability of success of 0.1. All nine wells fail. What is the probability of that happening? Simulate 50,000 instances of the experiment to answer.

```
import numpy as np

np.random.seed(1234)
sum(np.random.binomial(9, 0.1, 50000) == 0)/50000
```

```
0.38648
```

Exercise 3

- With a confidence of 95%, is the average number of acts per week in “Total.csv” higher than 100?

```
from scipy.stats import ttest_1samp

df1['Unidades Acto'].mean()
```

```
125.05660377358491
```

```
t_stat, p_value = ttest_1samp(df1['Unidades Acto'],
 ↪  popmean=100, alternative='greater')
print("t-statistic value: ", t_stat)
print("p-Value: ", p_value)
```

```
t-statistic value:  4.667727967082299
p-Value:  1.0882425980312585e-05
```

Exercise 3

- With a confidence of 95%, is the average number of acts per week in “Total.csv” higher than 100?

```
ttest_1samp(df1['Unidades Acto'], popmean=100,
 ↪  alternative='greater').confidence_interval(confidence_l
```

```
ConfidenceInterval(low=116.0667862752866, high=inf)
```

Exercise 4

- Generate two random samples with 100 observations from two normal distributions with different means, and conduct a test to check whether the means are different, at a 95% confidence level.

```python
from scipy.stats import ttest_ind

np.random.seed(1234)
sample1 = np.random.normal(size=100)
sample2 = np.random.normal(loc=10, size=100)
t_stat, p_value = ttest_ind(sample1, sample2)
print("t-statistic value: ", t_stat)
print("p-Value: ", p_value)
```

```
t-statistic value:  -70.66373790651313
p-Value:  2.081639823957165e-142
```

Exercise 5

- Load the file *musk.dta*. Generate a new column with name *hadSick* stating if a subject had a sickness leave in the follow-up period.

```
df2 = pd.read_stata('musk.dta')
df2['hadSick'] = np.where(df2['nbajas'] == 0, 0, 1)
df2.head()
```

	nid	nbajas	tseg	ed	form	cfis	rep
0	587.0	12.0	1536.817278	27.565973	No	Alta	Adecuad
1	253.0	10.0	1236.408353	42.279304	No	Alta	Adecuad
2	821.0	9.0	1117.078039	42.476569	Si	Alta	Adecuad
3	116.0	8.0	1215.961234	37.593875	No	Alta	Adecuad
4	243.0	8.0	1167.940300	35.026569	No	Media	Adecuad

Exercise 5

- Fit an appropriate model for the variable *hadSick*, using age (variable *ed*) as independent variable.

```python
import statsmodels.formula.api as sm

model = sm.logit("hadSick ~ ed", data=df2).fit()
print(model.summary())
print("AIC = ", model.aic)
```

```
Optimization terminated successfully.
         Current function value: 0.641705
         Iterations 4
                           Logit Regression Results
==========================================================
Dep. Variable:                hadSick   No. Observations:
Model:                          Logit   Df Residuals:
Method:                           MLE   Df Model:
Date:                Tue, 19 Mar 2024   Pseudo R-squ.:
Time:                        15:57:56   Log-Likelihood:
```

Exercise 5

- Fit an appropriate model for the variable *hadSick*, using age (variable *ed*) as independent variable.

```python
import statsmodels.formula.api as sm

model = sm.probit("hadSick ~ ed", data=df2).fit()
print(model.summary())
print("AIC = ", model.aic)
```

```
Optimization terminated successfully.
         Current function value: 0.641705
         Iterations 4
                          Probit Regression Results
==========================================================
Dep. Variable:                hadSick   No. Observations:
Model:                         Probit   Df Residuals:
Method:                           MLE   Df Model:
Date:                Tue, 19 Mar 2024   Pseudo R-squ.:
Time:                        15:57:56   Log-Likelihood:
```

Exercise 6

- Find the best model for the variable *nbajas*, using all other features as independent variables and *tseg* as offset.

```
import statsmodels.formula.api as sm

model = sm.poisson("nbajas ~
 ↪  ed+form+cfis+rep+rt+est",
 ↪  offset=np.log(df2.tseg), data=df2).fit()
print(model.summary())
print("AIC = ", model.aic)
```

```
Optimization terminated successfully.
         Current function value: 1.446228
         Iterations 6
                          Poisson Regression Results
==========================================================
Dep. Variable:                 nbajas   No. Observations:
Model:                        Poisson   Df Residuals:
Method:                           MLE   Df Model:
```

Exercise 6

- Find the best model for the variable *nbajas*, using all other features as independent variables and *tseg* as offset.

```
import statsmodels.formula.api as sm

model = sm.negativebinomial("nbajas ~
 ↪  ed+form+cfis+rep+rt+est",
 ↪  offset=np.log(df2.tseg), data=df2).fit()
print(model.summary())
print("AIC = ", model.aic)
```

```
Optimization terminated successfully.
         Current function value: 1.145746
         Iterations: 32
         Function evaluations: 38
         Gradient evaluations: 38
                     NegativeBinomial Regression Results
==============================================================
Dep. Variable:                  nbajas   No. Observations:
```

Exercise 6

- Find the best model for the variable *nbajas*, using all other features as independent variables and *tseg* as offset.

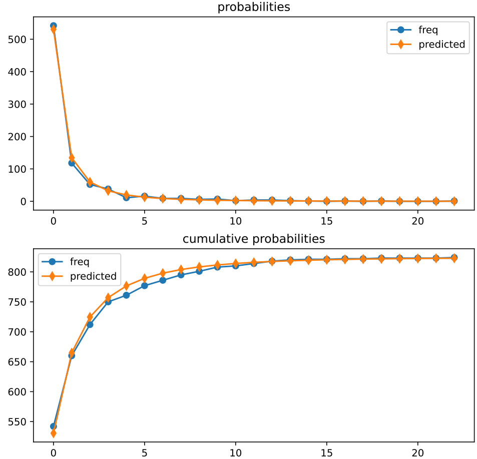
```
import statsmodels.formula.api as sm

model = sm.negativebinomial("nbajas ~
↪  ed+form+cfis+rep+rt+est", exposure=df2.tseg,
↪  data=df2).fit()
print(model.summary())
print("AIC = ", model.aic)
```

```
Optimization terminated successfully.
         Current function value: 1.145746
         Iterations: 32
         Function evaluations: 38
         Gradient evaluations: 38
                     NegativeBinomial Regression Results
==============================================================
Dep. Variable:                  nbajas   No. Observations:
```

Exercise 6

- Find the best model for the variable *nbajas*, using all other features as independent variables and *tseg* as offset.

```
dia_model = model.get_diagnostic()
dia_model.plot_probs()
```


probabilities
freq
predicted
cumulative probabilities
freq
predicted