EBA35001 Fall 2022

Take home exam

Jonas Moss

Introduction

1. Only show output that supports your argument. If you use Jupyter Notebooks, you may hide the output of a cell using a semi-colon `;`. We will deduct points from shoddily written reports plagued by noisy outputs.
2. Make your plots look nice. Add appropriate axis labels, legends and so on.
3. *"Brevity is the soul of wit."* Strive not to write too much. We prefer pithy to lengthy expositions.
4. The exercises are equally weighted. Each exercise gives $0-2$ points and there are 30 of them. That's a maximum of 60 points.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels as sm
import statsmodels.formula.api as smf
```

1 Binary regression

We're using the Wine quality data set in this exercise. Take a look at the page for more information.

(a)

(i)

Import the data set `winequalityN.csv` as `wine`. We want to use the data to deduce if a wine is acceptable, with acceptable defined as `quality > 5`. Define a column `acceptable` that contains `True`s when `quality > 5` and `False`s otherwise. We won't use `quality` anymore, so drop it from the table. Finally, drop all rows containing `NA`s.

Solution

```
wine = pd.read_csv("winequalityN.csv")
wine["acceptable"] = wine.quality > 5
wine = wine.drop(columns = "quality")
wine = wine.dropna()
```

(ii)

Make a `pairplot` of the data. Take note of any patterns. (*Hint*: If `pairplot` is slow, you can use the `sample` method to reduce the data strain.)

Solution

```
#sns.pairplot(wine.sample(1000), hue = "acceptable", kind = "kde")
#plt.show()
```

There are no bivariate patterns discernible from the data, but at least one univariate pattern, namely that the acceptable wines tend to have lower alcohol content. They also have higher "density".

(iii)

Display the correlation matrix between the numerical values in the data. Then find the columns where the difference between the correlations for the bad wines and good wines is greater than 0.15, along with both correlations. (*Hint:* You may need to iterate over all the column names and use a `set`.)

Solution

```
#wine.corr(numeric_only = True)
```

For the second part, first define

```
good = wine[wine.acceptable].corr(numeric_only = True)
bad = wine[wine.acceptable == False].corr(numeric_only = True)
```

Then find the indices where the difference is greater than 0.2.

```
indices = (good - bad).abs() > 0.15

names = wine.columns[range(1, 12)]
```

```
combs = set()
names = wine.columns[range(1, 12)]
for name1 in names:
  index = 0
  for name2 in names:
    if indices[name1][index]:
      combs.add((name1, name2, good[name1][index], bad[name1][index]))
    index += 1
combs
```

```
{('alcohol', 'fixed acidity', -0.12155755841918862, 0.03801324444613102),
 ('citric acid', 'residual sugar', 0.07119053469057889, 0.24484285448636298),
 ('citric acid',
  'total sulfur dioxide',
  0.1172900450140907,
  0.29736675267490476),
 ('fixed acidity', 'alcohol', -0.12155755841918862, 0.03801324444613102),
 ('residual sugar', 'citric acid', 0.07119053469057889, 0.24484285448636298),
 ('sulphates',
  'total sulfur dioxide',
  -0.3355264553378424,
  -0.18336724047154343),
 ('total sulfur dioxide',
  'citric acid',
  0.1172900450140907,
  0.29736675267490476),
 ('total sulfur dioxide',
  'sulphates',
```

```
  -0.3355264553378424,
  -0.18336724047154343)}
```

(b)

(i)

Run logistic regression model on all covariates with `acceptable` as response variable. Make a `significant` array containing all covariates that are significant at the 0.05 level (along with their p-values) and a `not_significant` array containing the rest. Print those two arrays.

Solution

```
model = smf.logit("I(acceptable*1) ~ Q('fixed acidity') + Q('volatile acidity') + Q('citri
```

```
Optimization terminated successfully.
         Current function value: 0.515532
         Iterations 7
```

```
pvalues = model.pvalues
significant = pvalues[pvalues < 0.05]
not_significant = pvalues[pvalues > 0.05]
significant
not_significant
```

```
C:\Users\A2010578\AppData\Local\Programs\Python\Python310\lib\site-packages\IPython\core\form
  return method()
```

	0
Intercept	0.249261
Q('fixed acidity')	0.105481
chlorides	0.374720
density	0.162111

(ii)

Fit a new regression model with the non-significant covariates removed, but keeping the intercept. Which model do you prefer?

Solution

We fit the model and look at the AIC.

```
model2 = smf.logit("I(acceptable*1) ~ Q('volatile acidity') + Q('citric acid') + Q('residu
model.aic
model2.aic
```

```
Optimization terminated successfully.
         Current function value: 0.515846
         Iterations 6
```

```
6685.818967406001
```

The AIC of the second model is lower, so I prefer it.

(iii)

Using the same two models as in (i) and (ii), change the link function from the logistic link to the Probit link, Cauchit link, and cloglog link. Report the AICs of the models in a table like this:

	Logistic	Probit	Cauchit	Cloglog
Model 1				
Model 2				

(*Hint*: You need to take a good look at the documentation of the `glm` function of statsmodels. Also see the lecture notes.)

Solution

Skipped.

(c)

(i)

The alcohol covariate appears to have a stronger influence than the other covariates. Fit three models using: (a) `log(alcohol + 1)`, (b) `alcohol**2`, (c) `log(alcohol + 1) + alcohol^2`, and report their AICs. Is any of the new models performing better than the others?

Solution

```
model3 = smf.logit("I(acceptable*1) ~ Q('volatile acidity') + Q('citric acid') + Q('residu
model4 = smf.logit("I(acceptable*1) ~ Q('volatile acidity') + Q('citric acid') + Q('residu
model5 = smf.logit("I(acceptable*1) ~ Q('volatile acidity') + Q('citric acid') + Q('residu
model.aic
model2.aic
model3.aic
model4.aic
model5.aic
```

```
Optimization terminated successfully.
         Current function value: 0.515087
         Iterations 7
Optimization terminated successfully.
         Current function value: 0.515190
         Iterations 7
Optimization terminated successfully.
         Current function value: 0.514513
         Iterations 10
```

```
6672.601178179357
```

We see that the model with both the log term and the quadratic term performs better.

(ii)

Fit at least five additional models and report their results in a table containing the formula and the resulting AIC, plus potentially more information. Pick your favorite among these.

Solution

Skipped.

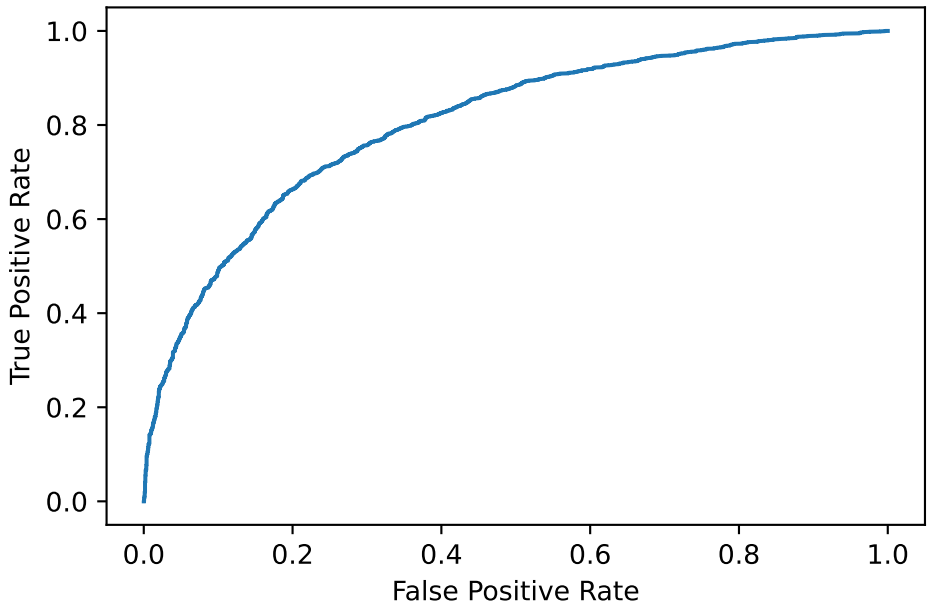
(iii)

Make a receiver operating characterstic curve for your favorite model. Explain what it means.

Solution

```
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(wine.acceptable,  model5.predict(wine))
plt.clf()
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



(iv)

Calculate the AUC for the models you tested in exercise (iii).

Solution

Skipped.

2 Linear regression

We use the student performance prediction, available in the `exams.csv` file. See the page for more information about this data set.

(a)

(i)

Import the exams file into the variable `exams` and make one or more plots displaying the association between the variables.

Solution

```
exams = pd.read_csv("exams.csv")
#sns.pairplot(data = exams) FIXME
#plt.show()
```

(ii)

It is well-known that general intelligence encompasses both math skill and literary skill. Display the correlation matrix between math skill, literary skill, and writing skill.

Solution

```
exams.corr()
```

```
C:\Users\A2010578\AppData\Local\Temp\ipykernel_15548\325883917.py:1: FutureWarning: The defau
  exams.corr()
C:\Users\A2010578\AppData\Local\Programs\Python\Python310\lib\site-packages\IPython\core\form
  return method()
```

	math score	reading score	writing score
math score	1.000000	0.811767	0.790055
reading score	0.811767	1.000000	0.948909
writing score	0.790055	0.948909	1.000000

(iii)

Are the correlations in the previous exercise significant? You can use any valid method to figure this out, but you might want to use linear regression. Don't use the summary method to display the p-values, as it takes too much space. (*Hint:* Remember to use the `Q` function to access columns with spaces inside.)

Solution

Use linear regression as follows:

```
smf.ols("Q('math score') ~ Q('writing score')", data = exams).fit().pvalues[1]
```

```
2.602621225816634e-214
```

```
smf.ols("Q('math score') ~ Q('reading score')", data = exams).fit().pvalues[1]
```

```
2.2844406458918446e-235
```

```
smf.ols("Q('reading score') ~ Q('writing score')", data = exams).fit().pvalues[1]
```

```
0.0
```

(iv)

Find the optimal linear combination of `writing score` and `reading score` to predict `math score`. What is the correlation between `math score` and this optimal linear combination? Recall that a linear combination is on the form `a + b * writing score + c * reading score`.

Solution

Make a fit model.

```
fit = smf.ols("Q('math score') ~ Q('writing score') + Q('reading score')", data = exams).f
```

The parameters of this optimal combination are

```
fit.params
```

```
C:\Users\A2010578\AppData\Local\Programs\Python\Python310\lib\site-packages\IPython\core\form
  return method()
```

	0
Intercept	6.449774
Q('writing score')	0.201431
Q('reading score')	0.673940

And the correlation is the square root of the R^2, i.e.,

```
np.sqrt(fit.rsquared)
```

```
0.8141793022251844
```

The correlation is positive since the coefficients of `writing score` and `reading score` as positive.

(b)

(i)

Display the distinct categories in every column that contains only categorical values.

Solution

```
exams.gender.unique()
exams["race/ethnicity"].unique()
exams["parental level of education"].unique()
exams["test preparation course"].unique()
```

```
array(['completed', 'none'], dtype=object)
```

(ii)

Fit a regression model on `math score` using all covariates except `writing skill` and `reading skill`. Show its summary table. Should you report the adjusted R^2 or the ordinary R^2 for this model?

Solution

You should report the adjusted R^2, as the model contains many covariates.

```
smf.ols("Q('math score') ~ gender + Q('race/ethnicity') + Q('parental level of education')
```

Table 2: OLS Regression Results

Dep. Variable:	Q('math score')	R-squared:	0.157
Model:	OLS	Adj. R-squared:	0.148
Method:	Least Squares	F-statistic:	16.73
Date:	Mon, 17 Apr 2023	Prob (F-statistic):	1.48e-30
Time:	14:39:55	Log-Likelihood:	-4057.6
No. Observations:	1000	AIC:	8139.
Df Residuals:	988	BIC:	8198.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	68.1568	1.915	35.582	0.000	64.398	71.916
gender[T.male]	5.8387	0.895	6.521	0.000	4.082	7.596
Q('race/ethnicity')[T.group B]	-1.3788	1.881	-0.733	0.464	-5.069	2.311
Q('race/ethnicity')[T.group C]	-0.3568	1.775	-0.201	0.841	-3.840	3.126
Q('race/ethnicity')[T.group D]	2.6920	1.816	1.483	0.138	-0.871	6.255
Q('race/ethnicity')[T.group E]	11.6963	1.979	5.909	0.000	7.812	15.581
Q('parental level of education')[T.bachelor's degree]	-0.5428	1.697	-0.320	0.749	-3.872	2.786
Q('parental level of education')[T.high school]	-4.7805	1.380	-3.465	0.001	-7.488	-2.073
Q('parental level of education')[T.master's degree]	0.2036	1.906	0.107	0.915	-3.536	3.943
Q('parental level of education')[T.some college]	-2.1566	1.366	-1.578	0.115	-4.838	0.525
Q('parental level of education')[T.some high school]	-6.6124	1.447	-4.569	0.000	-9.452	-3.773
Q('test preparation course')[T.none]	-3.9116	0.940	-4.159	0.000	-5.757	-2.066

Omnibus:	18.293	Durbin-Watson:	1.950
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18.642
Skew:	-0.317	Prob(JB):	8.95e-05
Kurtosis:	2.787	Cond. No.	12.1

(iii)

From the output above it looks like the results are roughly linear in level of education of the parents. Add a new column (called `education numeric`) to the data where the level of education is numeric, i.e., `some high school` is mapped to 1, `high school` to 2, and so on. (*Hint*: Google `pandas replace`.) Run a linear regression using `education numeric` instead of `parental level of education`. Would you prefer to use this model or the last model?

Solution

Use `replace` to make a new column.

```
exams['education numeric'] = exams['parental level of education'].replace(['some high scho
```

Then run a linear regression.

```
smf.ols("Q('math score') ~ gender + Q('race/ethnicity') + Q('education numeric') + Q('test
```

Table 5: OLS Regression Results

Dep. Variable:	Q('math score')	R-squared:	0.154
Model:	OLS	Adj. R-squared:	0.148
Method:	Least Squares	F-statistic:	25.70
Date:	Mon, 17 Apr 2023	Prob (F-statistic):	1.99e-32
Time:	14:39:55	Log-Likelihood:	-4059.7
No. Observations:	1000	AIC:	8135.
Df Residuals:	992	BIC:	8175.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	60.7418	1.932	31.436	0.000	56.950	64.534
gender[T.male]	5.8293	0.894	6.519	0.000	4.075	7.584
Q('race/ethnicity')[T.group B]	-1.5227	1.877	-0.811	0.417	-5.205	2.160
Q('race/ethnicity')[T.group C]	-0.4872	1.770	-0.275	0.783	-3.961	2.987
Q('race/ethnicity')[T.group D]	2.5974	1.813	1.433	0.152	-0.960	6.155
Q('race/ethnicity')[T.group E]	11.5312	1.975	5.839	0.000	7.656	15.407
Q('test preparation course')[T.none]	-3.9446	0.940	-4.199	0.000	-5.788	-2.101
Q('education numeric')	1.5755	0.299	5.263	0.000	0.988	2.163

Omnibus:	18.674	Durbin-Watson:	1.942
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19.067
Skew:	-0.321	Prob(JB):	7.24e-05
Kurtosis:	2.789	Cond. No.	30.0

I would prefer to use this model, as the adjusted R^2s are equal, and this one is simpler. But any reasonable answer is OK here.

(iv)

Your colleague John wants to use both `education numeric` and `parental level of education`. Is this a good idea? Explain.

Solution

The column `education numeric` contains no information not already present in `parental level of education`, hence there is no reason to use both.

(c)

(i)

Run the model `Q('writing score') ~ gender + Q('race/ethnicity')` and display its parameter estimates. What is the interpretation of `gender[T.male]` and `Q('race/ethnicity')[T.group E]`?

Solution

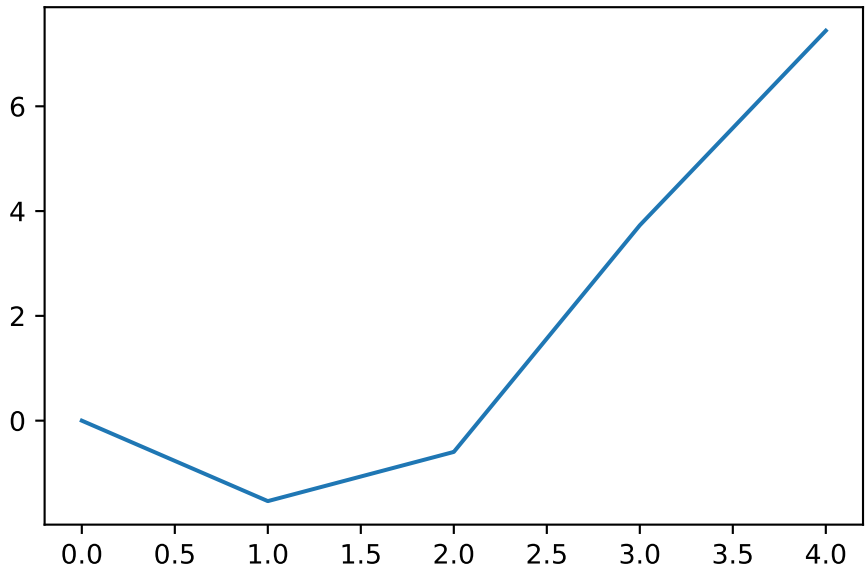
```
fit = smf.ols("Q('writing score') ~ gender + Q('race/ethnicity')", data = exams).fit()
```

(ii)

Referring to the model in (i), we would like to check if it is possible to do a similar trick as we did for level of education, linearizing the categories. Plot the estimated values for `Q('race/ethnicity')` (in the appropriate order) against `[0, 1, 2, 3, 4]` and see if there is a pattern.

Solution

```
x = [0] + fit.params[[2, 3, 4, 5]].tolist()
plt.clf()
plt.plot([0, 1, 2, 3, 4], x)
plt.show()
```



(iii)

Fit a suitable function to the data obtained in the previous exercise. Then replace the categorical values of `race/ethnicity` with the predicted values in a column `race numeric`. Finally, run the regression in (i) again, but with `race numeric` instead of `race/ethnicity`. Which model do you prefer?

Solution

We use a quadratic function.

```
dat = pd.DataFrame({"y": x, "x": [0, 1, 2, 3, 4]})
mod = smf.ols("y ~ x + I(x**2)", data = dat).fit().summary()
```

```
C:\Users\A2010578\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\stats
  warn("omni_normtest is not valid with less than 8 observations; %i "
```

Now we can replace the values:

```
exams["race numeric"] = exams["race/ethnicity"].replace(["group A", "group B", "group C",
exams
```

```
C:\Users\A2010578\AppData\Local\Programs\Python\Python310\lib\site-packages\IPython\core\form
  return method()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math sc
0	female	group D	some college	standard	completed	
1	male	group D	associate's degree	standard	none	
2	female	group D	some college	free/reduced	none	
3	male	group B	some college	free/reduced	none	
4	female	group D	associate's degree	standard	none	
5	male	group C	some high school	standard	none	
6	female	group E	associate's degree	standard	none	
7	female	group B	some high school	standard	none	
8	male	group C	some high school	standard	none	
9	female	group C	bachelor's degree	standard	completed	
10	male	group B	some high school	standard	none	
11	male	group B	master's degree	standard	none	
12	male	group B	bachelor's degree	free/reduced	none	
13	male	group A	some college	standard	none	
14	male	group C	master's degree	free/reduced	none	
15	male	group E	master's degree	free/reduced	none	
16	female	group C	some college	free/reduced	none	
17	female	group C	high school	standard	none	
18	female	group E	associate's degree	free/reduced	none	
19	female	group D	associate's degree	standard	completed	
20	male	group C	high school	free/reduced	none	
21	female	group D	associate's degree	standard	completed	
22	female	group B	some college	standard	none	
23	male	group E	associate's degree	standard	none	
24	male	group D	associate's degree	standard	completed	
25	male	group B	some college	standard	none	
26	male	group D	high school	free/reduced	none	
27	male	group D	some high school	standard	none	
28	female	group E	associate's degree	standard	none	
29	male	group D	associate's degree	standard	none	
30	female	group B	associate's degree	standard	completed	
31	female	group D	associate's degree	free/reduced	none	
32	female	group B	high school	free/reduced	completed	
33	female	group D	master's degree	free/reduced	none	
34	male	group D	high school	standard	completed	
35	female	group A	some high school	standard	none	
36	female	group E	some college	free/reduced	none	
37	male	group D	associate's degree	standard	completed	
38	male	group C	associate's degree	free/reduced	completed	
39	male	group E	master's degree	standard	none	
40	male	group C	associate's degree	free/reduced	completed	
41	male	group C	associate's degree	standard	completed	
42	male	group B	high school	standard	none	
43	male	group D	some college	standard	none	
44	female	group B	high school	standard	none	
45	male	group C	bachelor's degree	standard	none	
46	male	group A	associate's degree	standard	completed	
47	male	group C	some college	standard	completed	
48	female	group E	high school	standard	none	
49	male	group D	some high school	standard	completed	
50	female	group E	high school	free/reduced	completed	
51	female	group E	master's degree	free/reduced	completed	

And now we can run the regressions:

```
fit1 = smf.ols("Q('writing score') ~ gender + Q('race/ethnicity')", data = exams).fit()
fit2 = smf.ols("Q('writing score') ~ gender + Q('race numeric')", data = exams).fit()
fit1.aic
fit2.aic
```

```
8143.63945298202
```

The AIC is much lower for the first model, so I prefer that one. # 3 Simulations

(a)

The central limit theorem states that $\sqrt{n}(\overline{X}_n - \mu)/\sigma \to N(0,1)$ when $X_1, X_2, \ldots, X_n$ are iid with mean μ and standard deviation σ, and the empirical mean is $\overline{X}_n = (\sum_{i=1}^{n} X_i)/n$ and $N(0,1)$ is the standard normal. It's often interesting to see how quick the convergence is; we'll explore that in this problem.

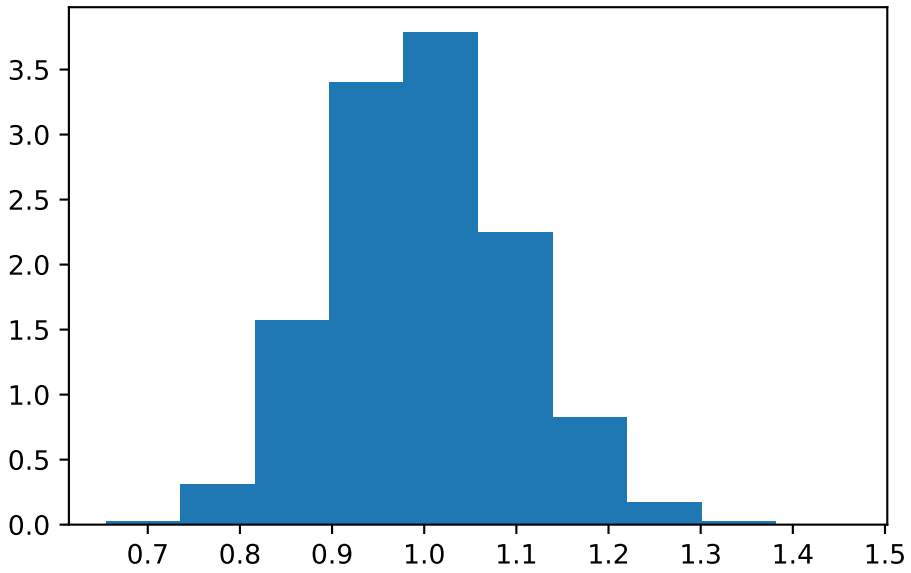
(i)

Make a function `simmean(n, model, n_reps = 10000)`. It simulates `n_reps` times n observations from the random number generator `model` and calculates its empirical mean and empirical standard deviation. Make a histogram (with densities, not counts) of the simulations when the model is the function `np.random.default_rng(seed = 313).exponential(1, dim)`

Solution

```
def simmean(n, model, n_reps = 10000):
  x = model((n, n_reps))
  return x.mean(axis = 0)

plt.clf()
plt.hist(simmean(100, lambda dim: np.random.default_rng(seed = 313).exponential(1, dim)),
plt.show()
```



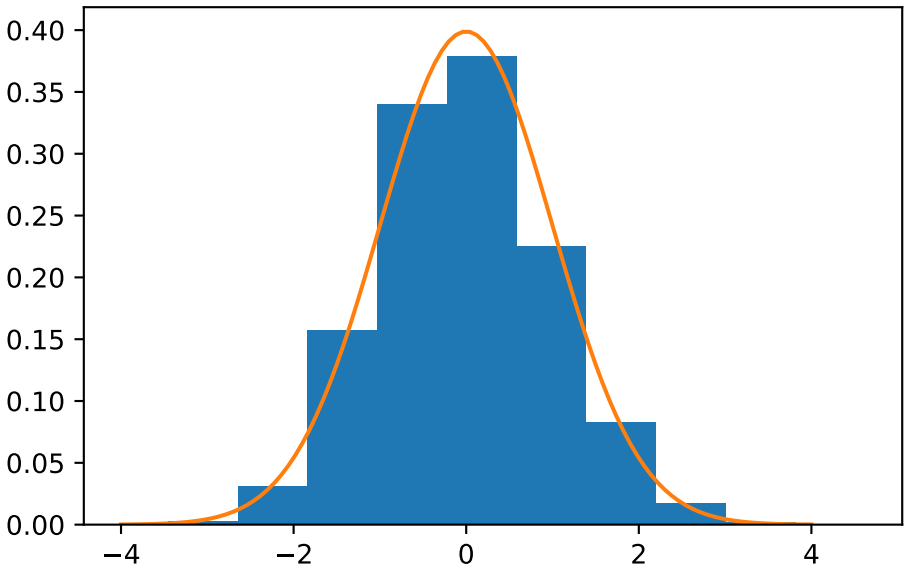
(ii)

Make a function `simclt` that extends to function above with `mu` and `sigma`, and returns samples from $\sqrt{n}(\overline{X}_n - \mu)/\sigma$. Make a (density) histogram for the same input as above, and overlay a standard normal on top of the plot.

Solution

```
from scipy.stats import norm
def simclt(n, model, mu, sigma, n_reps = 10000):
  x = model((n, n_reps))
  return (x.mean(axis = 0) - mu) * np.sqrt(n) / sigma

plt.clf()
plt.hist(simclt(100, lambda dim: np.random.default_rng(seed = 313).exponential(1, dim), mu
x = np.linspace(-4, 4, 100)
plt.plot(x, norm.pdf(x, 0, 1))
plt.show()
```



(iii)

Make similar plots for the Pareto distribution with parameter $b = 3$ (following the Scipy convention) for $n = 10, n = 100, n = 1000$, and extend the exponential analysis to $n = 10$ and $n = 1000$. Comment on the results. You need to figure out the mean and standard deviation for the Pareto yourself. (*Hint:* See the Scipy documentation and wikipedia. Observe that `Numpy` shifts the Pareto distribution towards `0`; see the Numpy docs for details.)

Solution

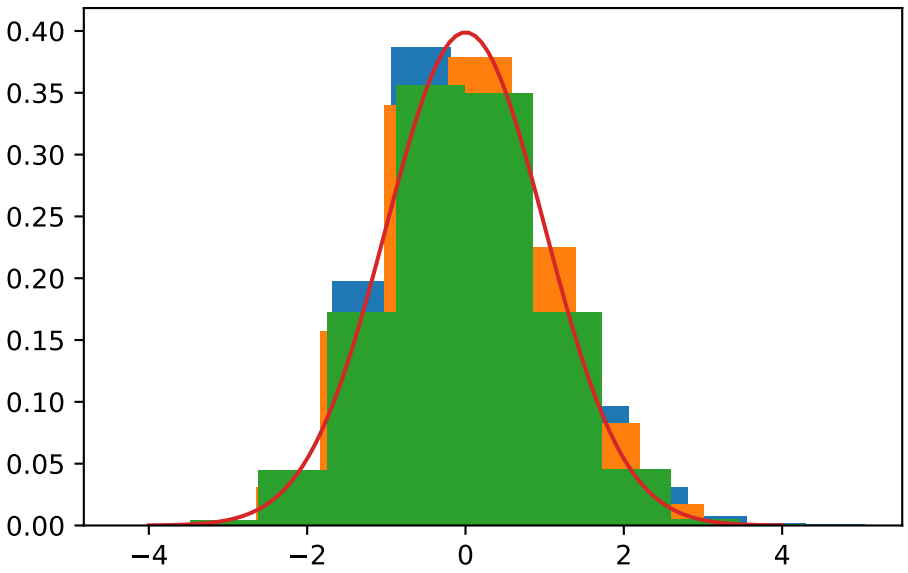
We see that the CLT kick in slower for Pareto than the exponential.

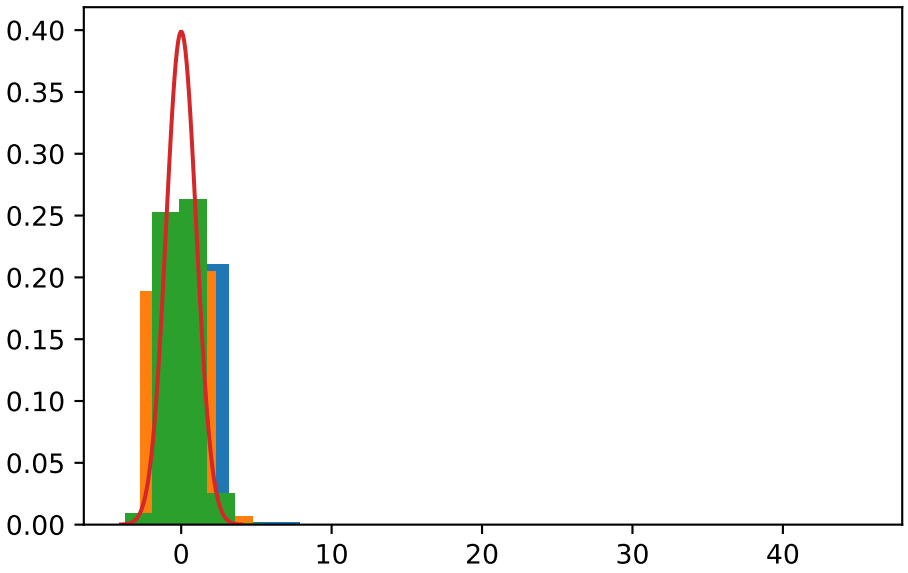
```
plt.clf()
plt.hist(simclt(10, lambda dim: np.random.default_rng(seed = 313).exponential(1, dim), mu
plt.hist(simclt(100, lambda dim: np.random.default_rng(seed = 313).exponential(1, dim), mu
plt.hist(simclt(1000, lambda dim: np.random.default_rng(seed = 313).exponential(1, dim), m
x = np.linspace(-4, 4, 100)
plt.plot(x, norm.pdf(x, 0, 1))
plt.show()

from scipy.stats import pareto
b = 3
mu = b / (b - 1) - 1
```

```
sigma = np.sqrt(b / ((b - 1)**2 * (b - 2)))


plt.clf()
rng = np.random.default_rng(seed = 313)
plt.hist(simclt(10, lambda dim: rng.pareto(b, dim), mu = mu, sigma = sigma), density = Tru
plt.hist(simclt(100, lambda dim: rng.pareto(b, dim), mu = mu, sigma = sigma), density = Tr
plt.hist(simclt(1000, lambda dim: rng.pareto(b, dim), mu = mu, sigma = sigma), density = T
x = np.linspace(-4, 4, 100)
plt.plot(x, norm.pdf(x, 0, 1))
plt.show()
```





(b)

(i)

Make a function `simmax` that simulates n observations from a standard exponential distribution and finds the maximum of the observations. It must take an `rng` as input. Use it to simulate the maximum of $n = 1000$ exponentials when `rng = np.random.default_rng(seed = 313)`.

Solution

```
rng = np.random.default_rng(seed = 313)
def simmax(n, rng):
  return rng.exponential(1, 1000).max()

simmax(1000, rng)
```

```
7.534737707840161
```

(ii)

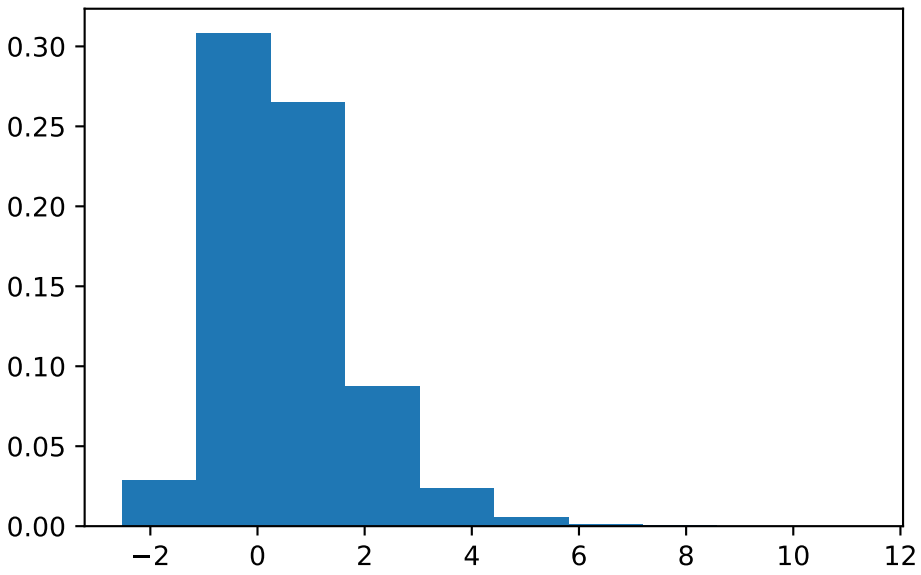
Extend the function above to `simmaxs`, a function that take an `n_reps = 10,000` argument in addition to `n` and `rng`. It should return a Numpy array with `n_reps` independent simulations of

the maximum. Make a histogram of "maxima-log(n)", where $n = 100$ and `n_reps = 10,000`. Make sure the histogram displays the density of the maxima, not the frequency count.

Solution

```
rng = np.random.default_rng(seed = 313)
def simmaxs(n, rng, n_reps = 10000):
  return rng.exponential(1, (n_reps, n)).max(axis = 1)

plt.clf()
plt.hist(simmaxs(100, rng) - np.log(100), density = True)
plt.show()
```



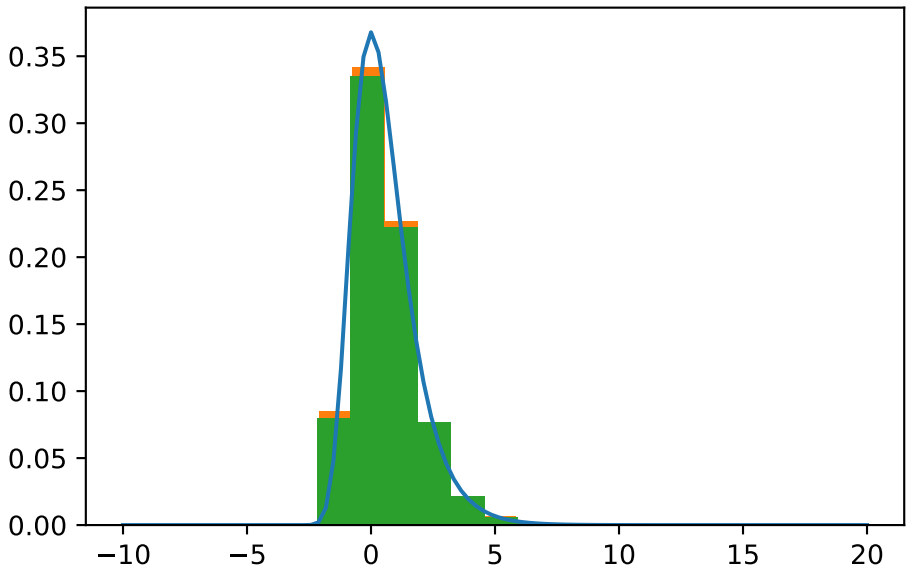
(iii)

Plot the standard Gumbel distribution on top of the histogram. It is part of Scipy, called `gumbel_r`. Make similar plots for $n = 1000$ and $n = 10000$. What do you see?

Solution

```
import scipy.stats as stats
x = np.linspace(-10, 20, 100)
```

```
plt.plot(x, stats.gumbel_r.pdf(x))
plt.hist(simmaxs(1000, rng) - np.log(1000), density = True )
plt.hist(simmaxs(10000, rng) - np.log(10000), density = True )
plt.show()
```



It appears that the maxima converges to a standard Gumbel as $n \to \infty$.

(c)

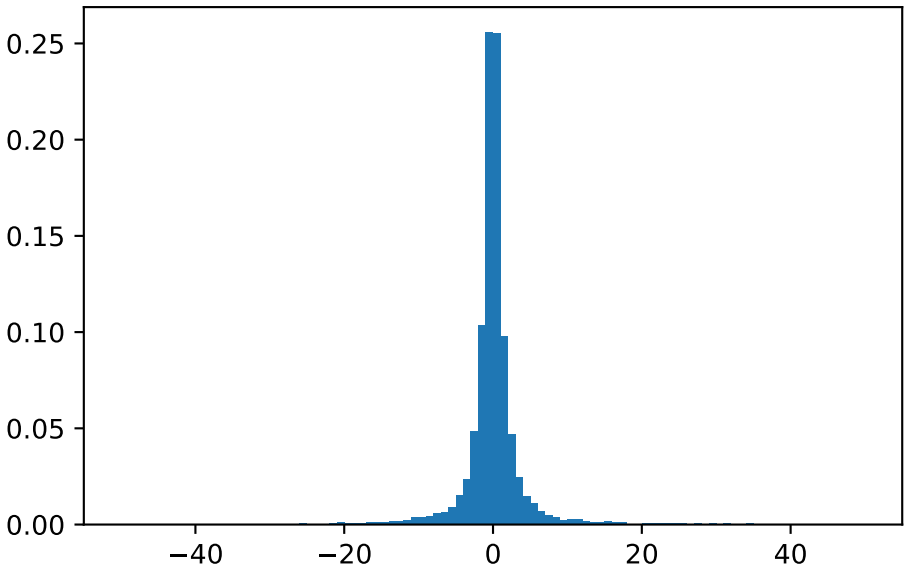
(i)

For a total of `n_reps = 10,000` times, draw $n = 100$ samples from the standard Cauchy distribution (`standard_t` with degrees of freedom equal to 1) and calculate the mean over these 100 values. Then make a histogram of its mean. Make sure the histogram shows a density, not the frequency of counts, give it `100` bins, and restrict it to the range `(-50, 50)`.

Solution

```
n = 100
n_reps = 10000
rng = np.random.default_rng(seed = 313)
```

```
x = rng.standard_t(1, (n_reps, n))
sims = x.mean(axis = 1)
plt.clf()
plt.hist(sims, range = (-50, 50), bins = 100, density = True)
plt.show()
sims
```




```
array([-3.97886746,  2.17656123, -0.44659774, ..., -0.07251852,
       -0.94334788,  3.74926668])
```

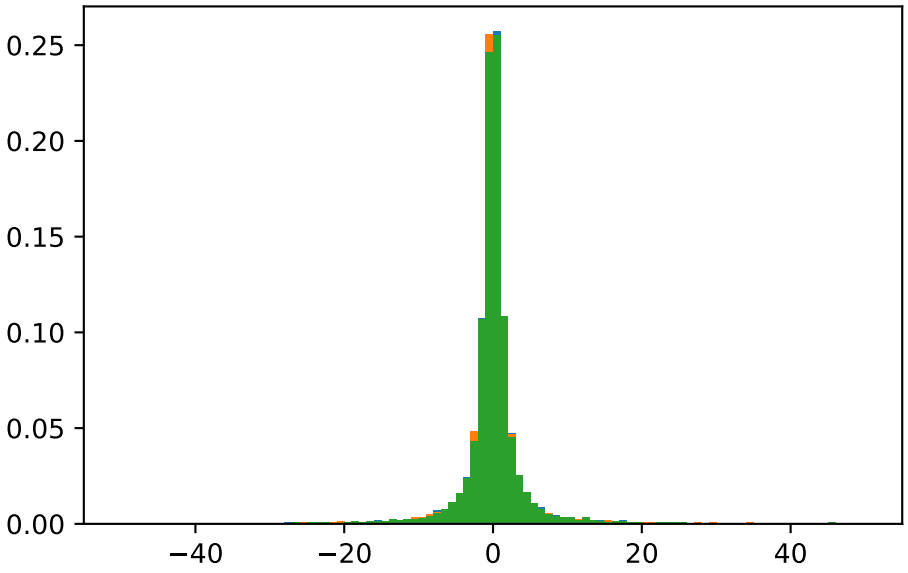
(ii)

Generalize the previous exercise by making a function taking `n` and `n_reps = 10000` as arguments, returning the simulated values. Make three histograms for `n=10`, `n=100` and `n=10,000`. What do you see?

Solution

```
rng = np.random.default_rng(seed = 313)
def simfun(n, rng, n_reps = 10000):
  x = rng.standard_t(1, (n_reps, n))
  return x.mean(axis = 1)
```

```
plt.clf()
plt.hist(simfun(10, rng), range = (-50, 50), bins = 100, density = True)
plt.hist(simfun(100, rng), range = (-50, 50), bins = 100, density = True)
plt.hist(simfun(1000, rng), range = (-50, 50), bins = 100, density = True)
plt.show()
sims
```




```
array([-3.97886746,  2.17656123, -0.44659774, ..., -0.07251852,
       -0.94334788,  3.74926668])
```

All histograms are roughly equal!

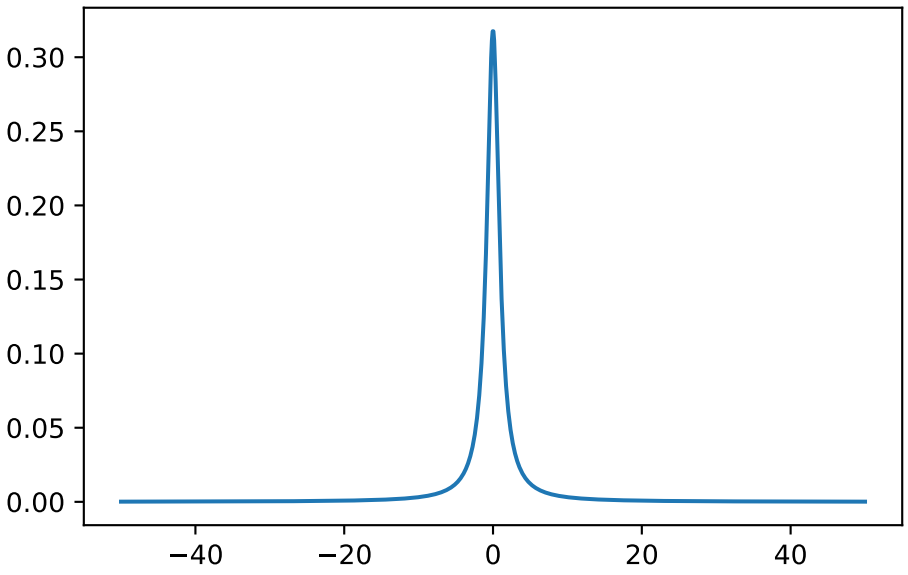
(iii)

Add a curve for the standard Cauchy distribution to the histograms of the last exercise. What do you see? (*Hint:* Use `Scipy`.)

Solution

```
from scipy.stats import cauchy
x = np.linspace(-50, 50, 1000)
plt.plot(x, cauchy.pdf(x))
```

```
plt.show()
```



(iv)

Does the central limit theorem hold for a sequence of iid Cauchy random variables? Why or why not? Demonstrate using a suitable simulation.

Solution

No, it does not hold. You can see that by observing the previous simulation "stabilized" when dividing by `n`. If you divide by $\sqrt{n}$, the histogram will explode, taking on larger and larger values as n increases.