causality

April 22, 2022

[6]:
```
import numpy as np
import pandas as pd
from patsy import dmatrices
import matplotlib.pyplot as plt
import seaborn as sb

import statsmodels.api as sm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LassoCV
```

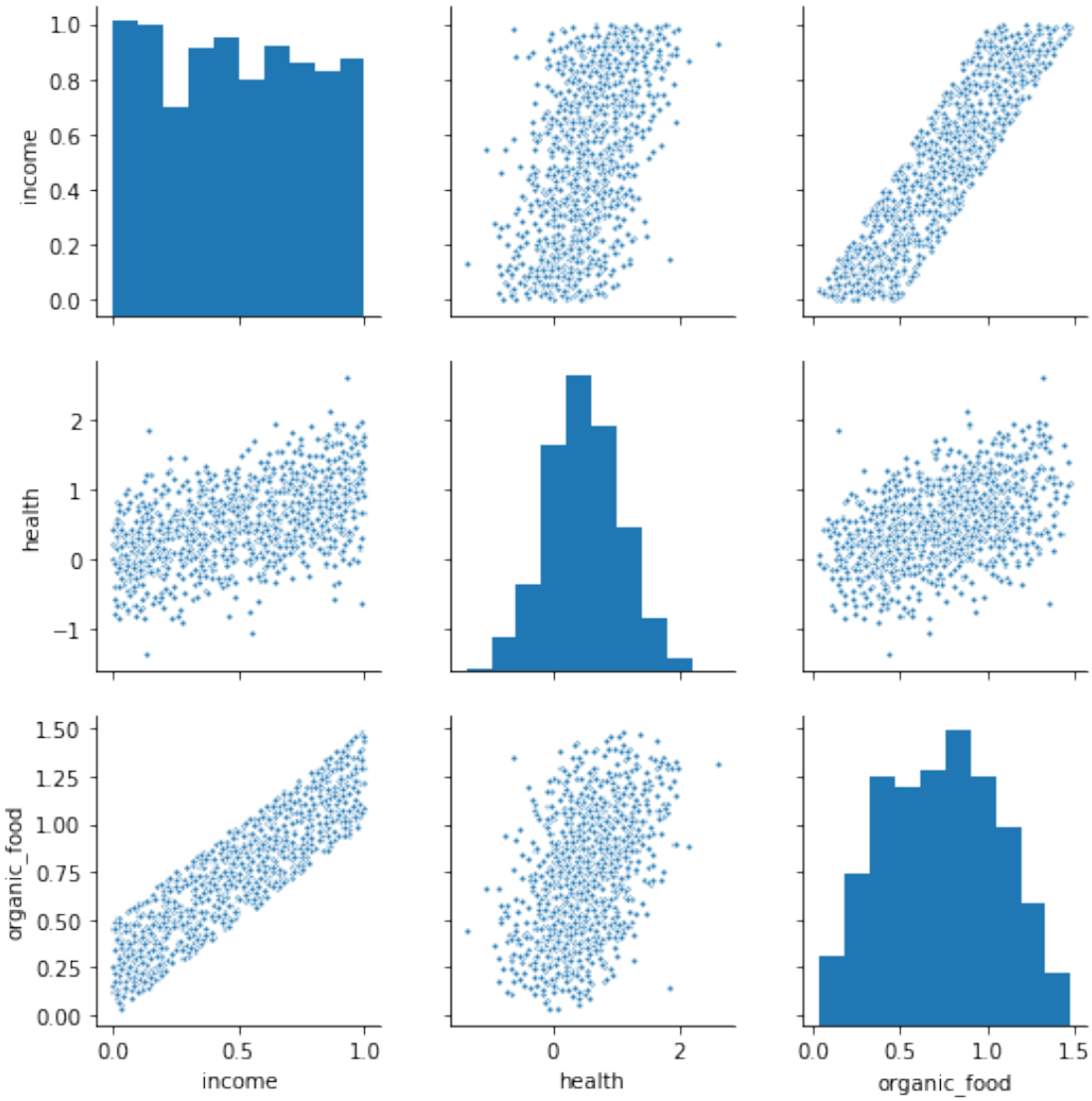
1 Hidden Confounders

1.0.1 Example 1: No relationship between health and organic food

[26]:
```
n = 1000
df1 = pd.DataFrame()

df1['income']=np.random.uniform(size=n)

# Health depends on income, but not on organic food consumption
alpha_0 = 0
alpha_1 = 1
df1['health'] = alpha_0 + alpha_1 * df1['income'] + 0.5*np.random.normal(size=n)

# Organic food consumptions also depends on income
df1['organic_food'] = df1['income'] + 0.5*np.random.uniform(size=n)
```

[24]:
```
# Observe that health is correlated with both income and organic food
sb.pairplot(df1,markers='.')
```

[24]:
```
<seaborn.axisgrid.PairGrid at 0x7fab42b997d0>
```



[27]:
```
# When we regress health on organic food consumption,
# we see that there is a positive coefficient (.7742)
# estimated for the dependence of health on organic food consumption, and the␣
 ↪p-value is significant
model = sm.OLS(df1['health'],sm.add_constant(df1['organic_food']))
model = model.fit()
model.summary()
```

[27]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.214
```

```
Model:                            OLS   Adj. R-squared:                  0.213
Method:                 Least Squares   F-statistic:                     271.9
Date:                Thu, 14 Apr 2022   Prob (F-statistic):           3.36e-54
Time:                        23:50:00   Log-Likelihood:                -755.19
No. Observations:                1000   AIC:                             1514.
Df Residuals:                     998   BIC:                             1524.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const           -0.1045      0.042     -2.476      0.013      -0.187      -0.022
organic_food     0.8406      0.051     16.488      0.000       0.741       0.941
==============================================================================
Omnibus:                        1.781   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.410   Jarque-Bera (JB):                1.655
Skew:                           0.092   Prob(JB):                        0.437
Kurtosis:                       3.078   Cond. No.                         5.07
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Q1: What is the change in health caused by increasing organic food intake by 1 unit?

1.0.2 Example 2: Some relationship between health and organic food

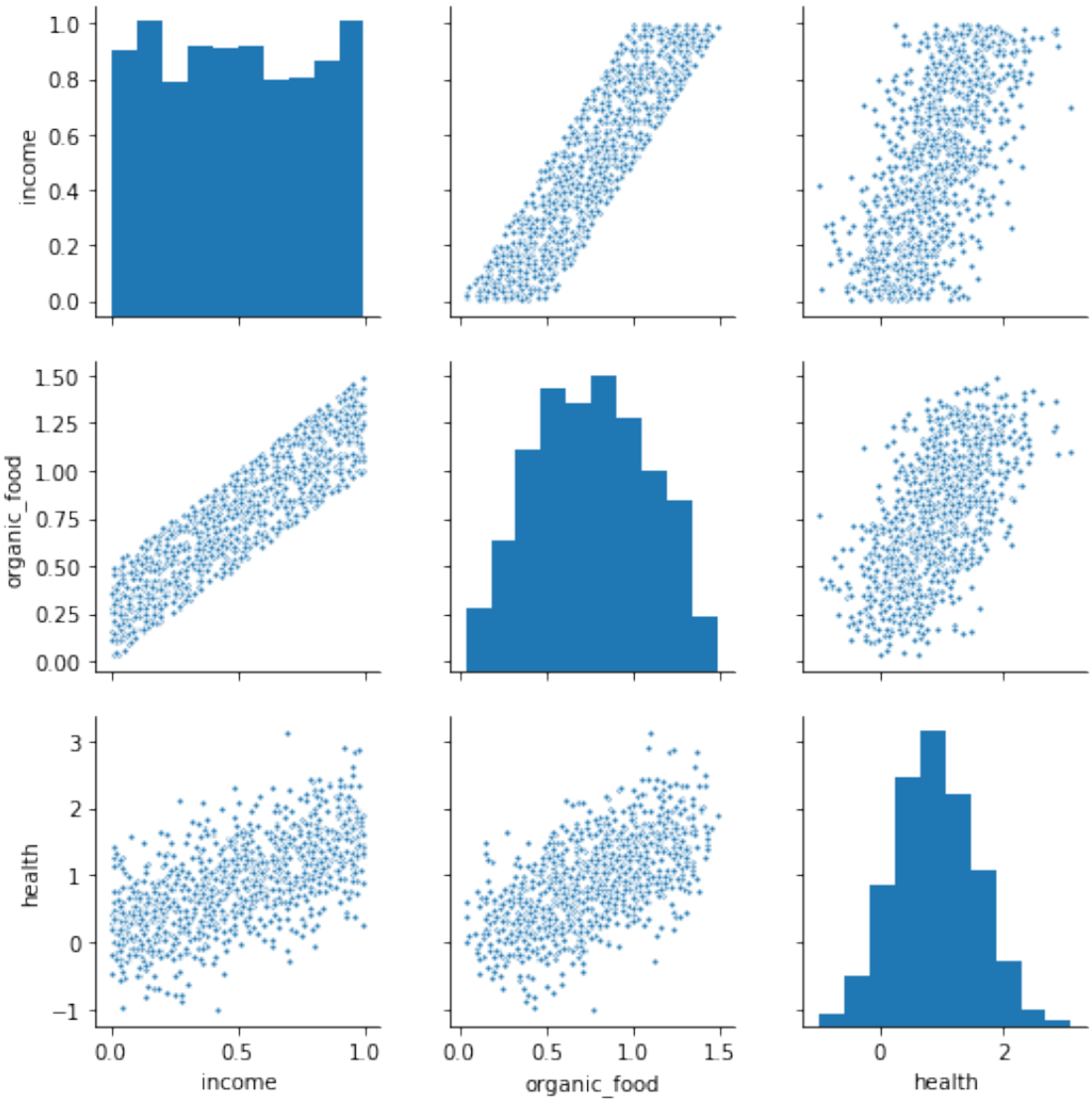
[29]:
```
n = 1000
df2 = pd.DataFrame()

df2['income']=np.random.uniform(size=n)

# Organic food consumptions also depends on income
df2['organic_food'] = df2['income'] + 0.5*np.random.uniform(size=n)

# Health depends on income AND on organic food consumption
alpha_0 = 0
alpha_1 = 1
alpha_2 = 0.5 # The causal impact of organic food on health is 0.5
df2['health'] = alpha_0 + alpha_1 * df2['income'] + alpha_2 *␣
 ↪df2['organic_food'] + 0.5*np.random.normal(size=n)
```

[32]:
```
model.summary()
```

[32]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.381
Model:                            OLS   Adj. R-squared:                  0.380
Method:                 Least Squares   F-statistic:                     614.4
Date:                Thu, 14 Apr 2022   Prob (F-statistic):          4.48e-106
Time:                        23:52:05   Log-Likelihood:                -773.35
No. Observations:                1000   AIC:                             1551.
Df Residuals:                     998   BIC:                             1561.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.0618      0.042     -1.476      0.140      -0.144       0.020
organic_food    1.2546      0.051     24.787      0.000       1.155       1.354
==============================================================================
Omnibus:                        0.745   Durbin-Watson:                   2.056
Prob(Omnibus):                  0.689   Jarque-Bera (JB):                0.655
Skew:                           0.058   Prob(JB):                        0.721
Kurtosis:                       3.050   Cond. No.                         4.93
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[30]:
```
# Health is correlated with both health consicousness and organic food
sb.pairplot(df2,markers='.')
```

[30]:
```
<seaborn.axisgrid.PairGrid at 0x7fab30f38a90>
```



[31]:
```
# When we regress health on organic food consumption,
# we see that there is a coefficient estimated for the dependence of
# health on organic food consumption of 1.3.  This is much bigger than the
# actual impact of organic food consumption on health of 0.5.
model = sm.OLS(df2['health'],sm.add_constant(df2['organic_food']))
model = model.fit()
model.summary()
```

[31]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.381
```

```
Model:                            OLS   Adj. R-squared:                  0.380
Method:                 Least Squares   F-statistic:                     614.4
Date:                Thu, 14 Apr 2022   Prob (F-statistic):          4.48e-106
Time:                        23:51:43   Log-Likelihood:                -773.35
No. Observations:                1000   AIC:                             1551.
Df Residuals:                     998   BIC:                             1561.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.0618      0.042     -1.476      0.140      -0.144       0.020
organic_food    1.2546      0.051     24.787      0.000       1.155       1.354
==============================================================================
Omnibus:                        0.745   Durbin-Watson:                   2.056
Prob(Omnibus):                  0.689   Jarque-Bera (JB):                0.655
Skew:                           0.058   Prob(JB):                        0.721
Kurtosis:                       3.050   Cond. No.                         4.93
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Q2: What is the change in health caused by increasing organic food intake by 1 unit?

1.0.3 Solution: include hidden confounders into regression

[35]:
```
# Example 1 (no relationship between organic food and health)
# With this regression including income,
# organic_food's p-value is no longer statistically significant
model = sm.OLS(df1['health'],sm.add_constant(df1[['organic_food','income']]))
model = model.fit()
model.summary()
```

[35]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.274
Model:                            OLS   Adj. R-squared:                  0.272
Method:                 Least Squares   F-statistic:                     187.9
Date:                Thu, 14 Apr 2022   Prob (F-statistic):           5.66e-70
Time:                        23:55:06   Log-Likelihood:                -715.72
No. Observations:                1000   AIC:                             1437.
Df Residuals:                     997   BIC:                             1452.
```

```
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0005      0.042      0.012      0.990      -0.082       0.083
organic_food  -0.0245      0.107     -0.228      0.820      -0.235       0.186
income         1.0764      0.119      9.050      0.000       0.843       1.310
==============================================================================
Omnibus:                        0.649   Durbin-Watson:                   2.034
Prob(Omnibus):                  0.723   Jarque-Bera (JB):                0.649
Skew:                           0.062   Prob(JB):                        0.723
Kurtosis:                       2.986   Cond. No.                         13.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[36]:
```
# Example 2: some relationship between organic food and health
# With this regression, the coefficient for organic food is much closer to the␣
 ↪correct causal impact of 0.5.
# The confidence interval contains 0.5.
model = sm.OLS(df2['health'],sm.add_constant(df2[['organic_food','income']]))
model = model.fit()
model.summary()
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.408
Model:                            OLS   Adj. R-squared:                  0.407
Method:                 Least Squares   F-statistic:                     344.1
Date:                Thu, 14 Apr 2022   Prob (F-statistic):          2.33e-114
Time:                        23:55:16   Log-Likelihood:                -750.77
No. Observations:                1000   AIC:                             1508.
Df Residuals:                     997   BIC:                             1522.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0334      0.043      0.771      0.441      -0.052       0.118
organic_food   0.5514      0.115      4.802      0.000       0.326       0.777
income         0.8776      0.129      6.786      0.000       0.624       1.131
```

```
==============================================================================
Omnibus:                        1.802   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.406   Jarque-Bera (JB):                1.671
Skew:                           0.071   Prob(JB):                        0.434
Kurtosis:                       3.142   Cond. No.                         14.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

1.0.4 Why trying to include hidden confounders doesn't always fix the problem

[37]:
```
n = 1000
df3 = pd.DataFrame()

df3['income']=np.random.uniform(size=n)
df3['health_consciousness']=np.random.uniform(size=n)

# Health depends on income and health consciousness, but not on organic food␣
 ↪consumption
alpha_0 = 0
alpha_1 = 1
df3['health'] = df3['income'] + df3['health_consciousness'] + 0.5*np.random.
 ↪normal(size=n)

# Organic food consumptions also depends on income and health consciousness
df3['organic_food'] = df3['income'] + df3['health_consciousness'] + 0.5*np.
 ↪random.uniform(size=n)
```

[38]:
```
# We include income into our regression but don't include health consciousness
# because we don't have a good way to measure it.
# As a result, our regression has a stat sig p-value for the impact of organic␣
 ↪food on health,
# even though the causal impact is 0.
model = sm.OLS(df3['health'],sm.add_constant(df3[['organic_food','income']]))
model = model.fit()
model.summary()
```

[38]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.391
Model:                            OLS   Adj. R-squared:                  0.390
```

```
Method:                 Least Squares   F-statistic:                     319.8
Date:                Thu, 14 Apr 2022   Prob (F-statistic):          4.87e-108
Time:                        23:55:42   Log-Likelihood:                -750.91
No. Observations:                1000   AIC:                             1508.
Df Residuals:                     997   BIC:                             1523.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.1504      0.051     -2.957      0.003      -0.250      -0.051
organic_food    0.8478      0.052     16.301      0.000       0.746       0.950
income          0.2179      0.078      2.802      0.005       0.065       0.370
==============================================================================
Omnibus:                        1.231   Durbin-Watson:                   2.099
Prob(Omnibus):                  0.540   Jarque-Bera (JB):                1.243
Skew:                           0.085   Prob(JB):                        0.537
Kurtosis:                       2.974   Cond. No.                         9.57
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

1.0.5 Solution 2: Randomized Controlled Trial (also called an A/B test)

- Some people will be randomly "treated", influencing them to eat organic food.
- We refer to the "treated" people as the treatment group, and everyone else as the control group.
- To influence the treatment group to eat more organic food, we could:
  - deliver free groceries to both the treatment group and the control group
  - supply organic fruit to the treatment group
  - supply non-organic fruit to the control group
- If we ran this experiment in the U.S. as part of a research study to create generalizable knowledge, we would need to get approval from an Institutional Review Board (IRB), which would assess whether the experiment was ethical. They would ask questions like:
  - could anyone be harmed by the experiment?
  - how would we protect the health information of the research subjects?
- Many companies use A/B testing to improve their products. See, e.g., this ad from Google https://www.indeed.com/jobs?q=Data%20Scientist%20A%2FB&vjk=833cb89e47ae09cf

[40]:
```
n = 10000
df4 = pd.DataFrame()

# Randomly generate income
df4['income']=np.random.uniform(size=n)
```

```
# Half of people are randomly treated
df4['is_treated']=np.random.choice([0,1],size=n)

# Generate organic food consumption, which depends on income
df4['organic_food'] = df4['income'] + 0.5*np.random.uniform(size=n)

# People in the treatment group consume extra organic food
extra_organic = 0.1
df4['organic_food'] = df4['organic_food'] + extra_organic*df4['is_treated']

# Health depends on income AND on organic food consumption
alpha_0 = 0
alpha_1 = 1
alpha_2 = 0.5
df4['health'] = alpha_0 + alpha_1 * df4['income'] + alpha_2 *␣
↪df4['organic_food'] + 0.5*np.random.normal(size=n)
```

The impact of treatment on health (the "treatment effect") is the difference in expected health between the treatment group and the control group.

The health of a person is: `alpha_0 + alpha_1 * income + alpha_2 * organic_food + noise` where being in treatment increases `organic_food` by `extra_organic`

So the treatment effect is `alpha_2 * extra_organic = 0.5 * 0.1 = 0.05`

The causal effect of organic food on health is the derivative of health with respect to organic food. This is `alpha_2`.

So if we can estimate the treatment effect (`alpha_2 * extra_organic`), then we can estimate the causal effect of organic food on health by dividing by extra_organic

1.0.6 Estimating the treatment effect: Approach 1

There are two ways to estimate the treatment effect. The first is conceptually simpler: - The treatment effect is the difference in expected health between the treatment and control groups - The average health of a group estimates its expected health - To estimate the treatment effect, use the difference in averageg health between treatment and control

[41]:
```
# Average health of treatment
df4[df4['is_treated']==1]['health'].mean()
```

[41]:
```
0.9201384029641919
```

[42]:
```
# Average health of control
df4[df4['is_treated']==0]['health'].mean()
```

[42]:
```
0.8602455537850413
```

[43]:
```
# The difference in average health between the two groups estimates the true␣
↪treatment effect (0.05)
df4[df4['is_treated']==1]['health'].mean() -␣
↪df4[df4['is_treated']==0]['health'].mean()
```

[43]:
```
0.05989284917915061
```

1.0.7 Estimating the treatment effect: Approach 2

Another way to get the same estimate is via regression.

Observe: - `treatment_effect = E[health(treatment)] - E[health(control)]` - Recall that `is_treated` is 0 (control) or 1 (treatment) - So `E[health(is_treated)] = E[health(control)] + treatment_effect * is_treated`

Therefore, after fitting the regression - `E[health(is_treated)] = beta_0 + beta_1 * is_treated`

$\hat{\beta}_1$ estimates the treatment effect

1.0.8 Lecture Question Q3

What does $\hat{\beta}_0$ estimate? - A. The expected health of the treatment group - B. The expected health of the control group - C. The treatment effect - D. Negative 1 times the treatment effect - E. None of the above

[44]:
```
model = sm.OLS(df4['health'],sm.add_constant(df4['is_treated']))
model = model.fit()
model.summary()
```

[44]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 health   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     20.50
Date:                Fri, 15 Apr 2022   Prob (F-statistic):           6.03e-06
Time:                        00:09:59   Log-Likelihood:                -10053.
No. Observations:               10000   AIC:                         2.011e+04
Df Residuals:                    9998   BIC:                         2.013e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8602      0.009     91.388      0.000       0.842       0.879
is_treated     0.0599      0.013      4.528      0.000       0.034       0.086
==============================================================================
```

```
Omnibus:                       13.909   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               11.922
Skew:                           0.015   Prob(JB):                      0.00258
Kurtosis:                       2.833   Cond. No.                         2.63
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

The regression coefficient on is_treated is close to the treatment effect, extra_organic * alpha_2 = 0.05.

The confidence interval on this coefficient contains the correct value.

[45]:
```
print(f'The actual causal impact of organic food on health is {alpha_2}')
print(f'The estimated causal impact is {model.params["is_treated"] /␣
↪extra_organic}')
```

```
The actual causal impact of organic food on health is 0.5
The estimated causal impact is 0.5989284917915086
```

1.1 Power analysis for A/B test

1.1.1 Lec 16 Q4

Based on intuition, which characteristics would increase the chance that an experiment would say that organic food improves health? (Assume that it does actually improve health)

- A. Bigger sample size or a bigger causal effect
- B. Smaller sample size or a bigger causal effect
- C. Bigger sample size or a smaller causal effect
- D. Smaller sample size or a smaller causal effect

[24]:
```
def simulate_experiment(n = 10000, treatment_effect = 0.5*.1):
    # n is the number of people in the trial, in total between control and␣
↪treatment
    # treatment effect is the difference in expected outcome (health) between␣
↪treatment and control,
    # i.e., E[health | is_treated = 1] - E[health | is_treated = 0]
    # The default values are taken from our example A/B experiment above

    # This code is copied from above, where we simulated data for our A/B␣
↪experiment
    df4 = pd.DataFrame()
    df4['is_treated']=np.random.choice([0,1],size=n) # Randomly assign to␣
↪treatment and control
```

```
    df4['income']=np.random.uniform(size=n)

    # Organic food consumptions also depends on income
    df4['organic_food'] = df4['income'] + 0.5*np.random.uniform(size=n)

    # Those that are in the treatment group have extra organic food consumption
    extra_organic = 0.1
    df4['organic_food'] = df4['organic_food'] + extra_organic*df4['is_treated']

    # Health depends on income AND on organic food consumption
    alpha_0 = 0
    alpha_1 = 1

    # The treatment effect is extra_organic * alpha_2.  Set alpha_2 so that␣
↪this is equal to the passed value
    alpha_2 = treatment_effect / extra_organic

    # Simulate our outcome
    df4['health'] = alpha_0 + alpha_1 * df4['income'] + alpha_2 *␣
↪df4['organic_food'] + 0.5*np.random.normal(size=n)

    # Fit our model
    model = sm.OLS(df4['health'],sm.add_constant(df4['is_treated']))
    model = model.fit()

    # is_stat_sig is true if the fit found the coefficient associated with␣
↪is_treated to be statistically significant
    is_stat_sig = (model.pvalues['is_treated'] <= 0.05)

    return is_stat_sig, model.params['is_treated'], model.pvalues['is_treated']
```

[9]:
```
# Simulate the experiment 100 times
# Run it more to get a more accurate answer
experiment_reps = 100

n = 100
treatment_effect = 1
sim_results = []
for i in range(experiment_reps):
    sim_results.append(simulate_experiment(n,treatment_effect)[0])
```

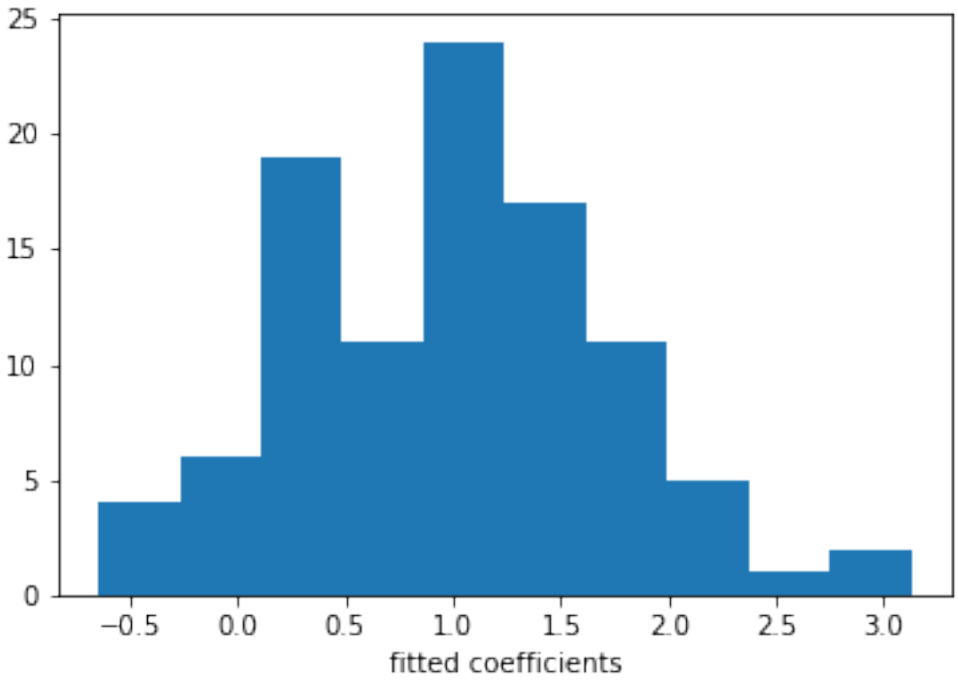
[10]:
```
# Here's a shorter way to do the same thing.  This is called a "list␣
↪comprehension"
sim_results = [simulate_experiment(n,treatment_effect)[0] for i in␣
↪range(experiment_reps)]
```

[11]:
```
power = np.mean(sim_results)
print(f'The probability that we will reject the null hypothesis and report a␣
 ↪stat sig treatment effect is {power}')
```

```
The probability that we will reject the null hypothesis and report a stat sig
treatment effect is 0.23
```

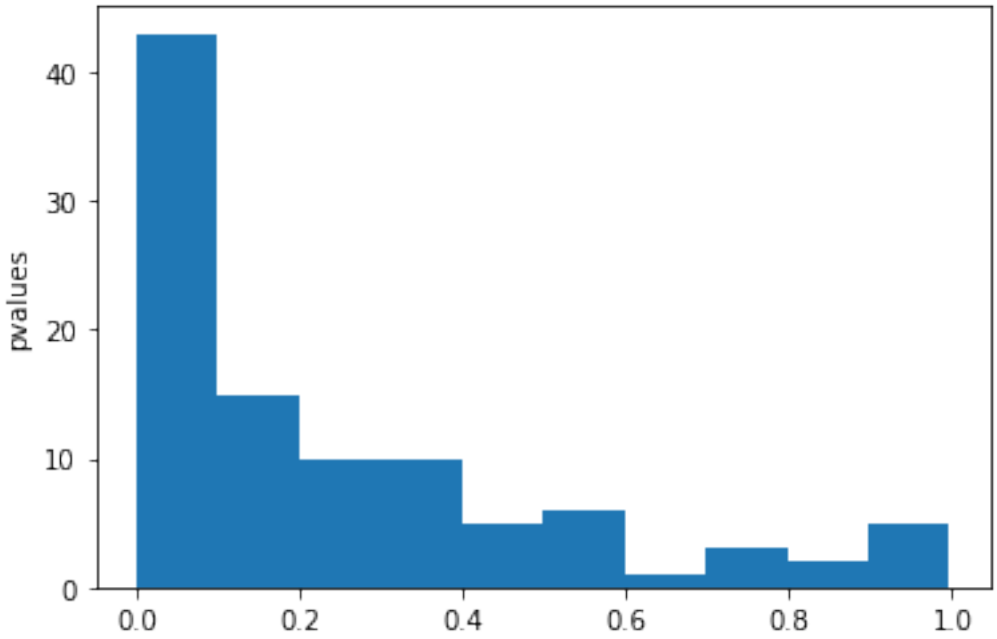
[12]:
```
plt.hist([simulate_experiment(n,treatment_effect)[1] for i in␣
 ↪range(experiment_reps)])
plt.xlabel('fitted coefficients')
```

[12]:
```
Text(0.5, 0, 'fitted coefficients')
```



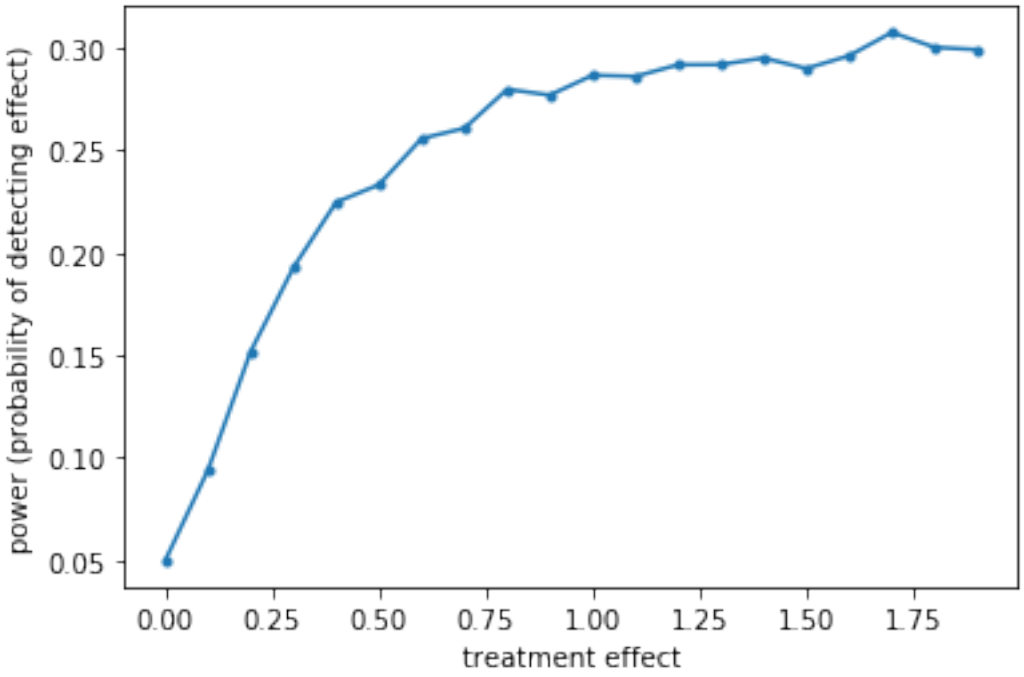
[13]:
```
plt.hist([simulate_experiment(n,treatment_effect)[2] for i in␣
 ↪range(experiment_reps)])
plt.ylabel('pvalues')
```

[13]:
```
Text(0, 0.5, 'pvalues')
```



[17]:
```
# Let's see how our power varies with the treatment effect
# Takes a long time to run with this many experiment_reps
experiment_reps = 10000
n = 100
treatment_effect = np.arange(0,2,.1)
power = []
for t in treatment_effect:
    sim_results = [simulate_experiment(int(n),t)[0] for i in␣
 ↪range(experiment_reps)]
    power.append(np.mean(sim_results))

plt.plot(treatment_effect, power, '.-')
plt.xlabel('treatment effect')
plt.ylabel('power (probability of detecting effect)')
```

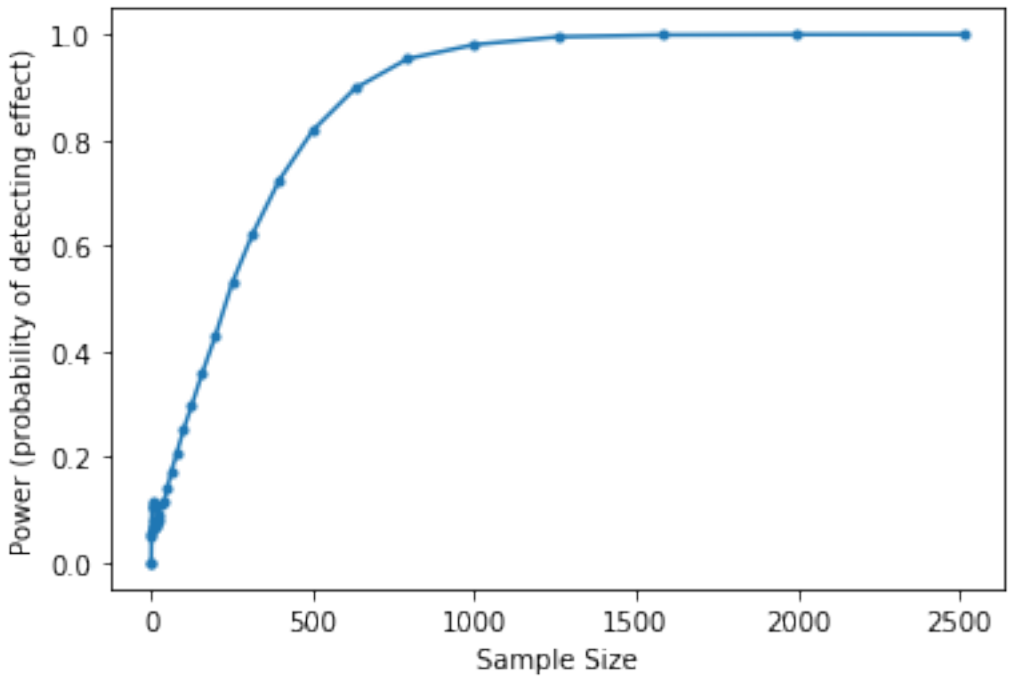
[17]:
```
Text(0, 0.5, 'power (probability of detecting effect)')
```



- Power goes up with the treatment effect
- When we design the experiment, we choose a "minimum detectable effect" (MDE)
- The MDE is the smallest treatment effect that we want to be able to detect reliably
- Then we increase sample size until we have the desired power (usually 80%) when the treatment effect is equal to the MDE
- If the treatment effect is bigger than the MDE, then the power will be bigger than 80%

[83]:
```
# Let's see that power increases with the sample size
# Takes a long time to run with this many experiment_reps
experiment_reps = 10000
t = 0.5
sample_size = np.round(10**np.arange(0,3.5,.1))
power = []
for n in sample_size:
    sim_results = [simulate_experiment(int(n),t)[0] for i in␣
 ↪range(experiment_reps)]
    power.append(np.mean(sim_results))

plt.plot(sample_size, power, '.-')
plt.xlabel('Sample Size')
plt.ylabel('Power (probability of detecting effect)')

# Power goes up with sample size, for any fixed treatment effect
```

[83]:
```
Text(0, 0.5, 'Power (probability of detecting effect)')
```



This graph tells us that, at an MDE of 0.5, a sample size of 500 is sufficient to give us a power of 80% or higher.

1.1.2 Lecture question Q5:

Suppose I ran the simulation above using t = 0.6 and plotted it in orange.

Where would the orange line sit?

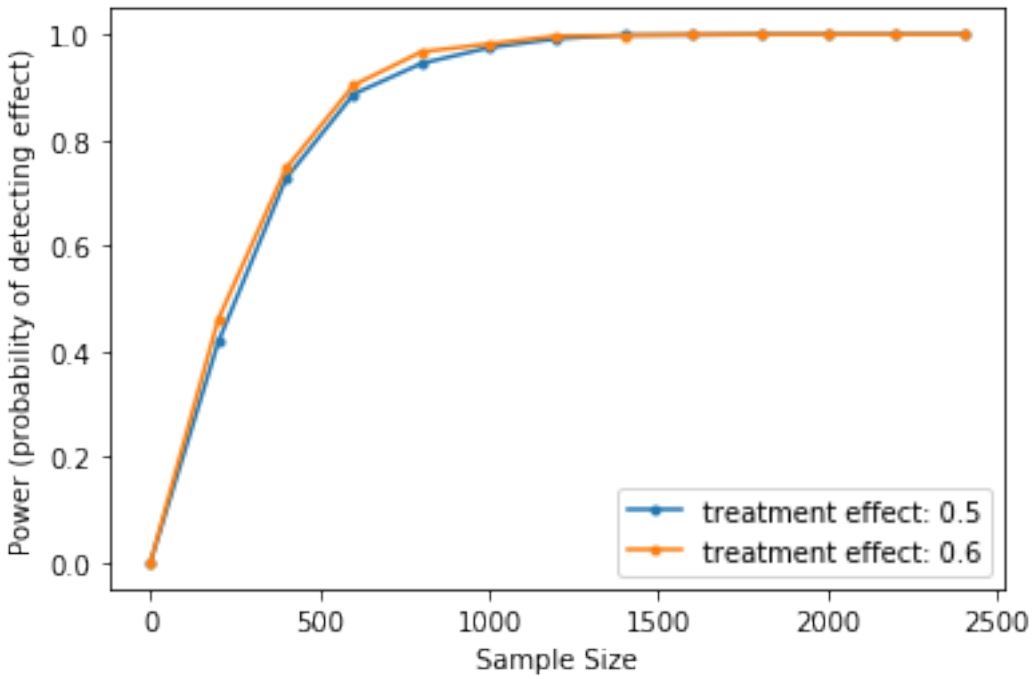
- A. Entirely below the blue line
- B. Entirely above the blue line
- C. Starting below and then crossing and ending above
- D. Starting above and then crossing and ending below

[34]:
```
# Let's see that power increases with the sample size
# Takes a long time to run with this many experiment_reps

experiment_reps = 1000
sample_size = np.arange(1,2500,200)
power = {}

for t in (0.5, 0.6):

    power[t] = np.zeros(len(sample_size))
    for i in range(len(sample_size)):
        n = sample_size[i]
```

```
        sim_results = [simulate_experiment(int(n),t)[0] for i in␣
↪range(experiment_reps)]
        power[t][i] = np.mean(sim_results)
```

[35]:
```
for t in power.keys():
    plt.plot(sample_size, power[t], '.-', label='treatment effect: {}'.
↪format(t))

plt.legend()
plt.xlabel('Sample Size')
plt.ylabel('Power (probability of detecting effect)')
```

[35]: Text(0, 0.5, 'Power (probability of detecting effect)')

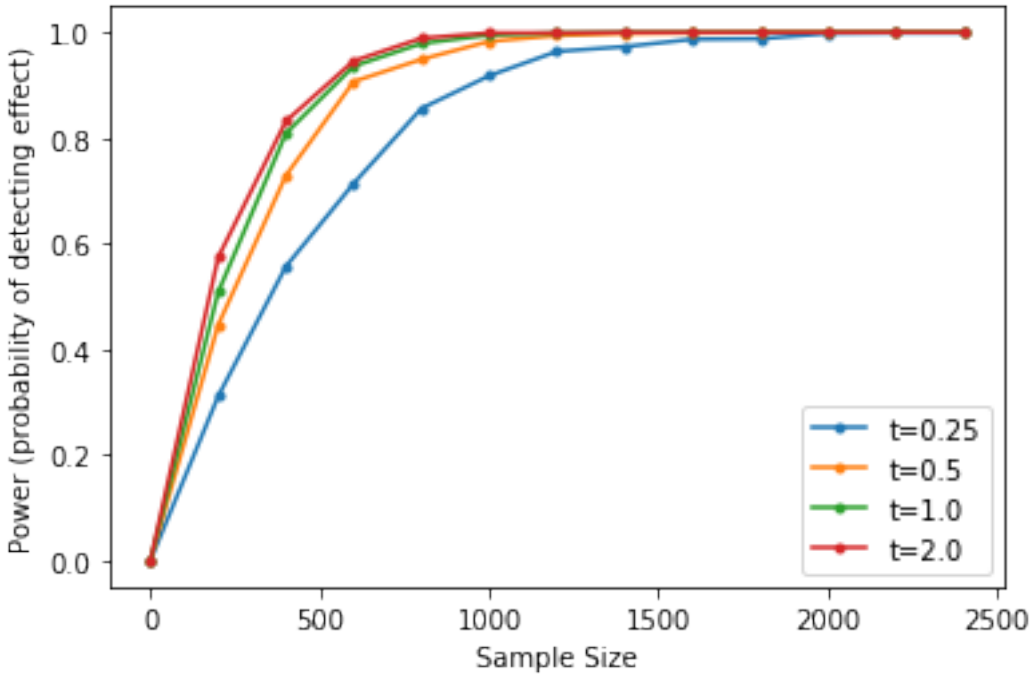


[36]:
```
# Let's see that power increases with the sample size
# Takes a long time to run with this many experiment_reps

experiment_reps = 1000
sample_size = np.arange(1,2500,200)
power = {}

for t in (0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2):

    power[t] = np.zeros(len(sample_size))
```

```
    for i in range(len(sample_size)):
        n = sample_size[i]
        sim_results = [simulate_experiment(int(n),t)[0] for i in␣
↪range(experiment_reps)]
        power[t][i] = np.mean(sim_results)
```

[38]:
```
for t in [.25,.5, 1., 2.]:
    plt.plot(sample_size, power[t], '.-', label='t={}'.format(t))

plt.legend()
plt.xlabel('Sample Size')
plt.ylabel('Power (probability of detecting effect)')
```

[38]:
```
Text(0, 0.5, 'Power (probability of detecting effect)')
```



2 Overfitting leads to correlation that is not causation

[8]:
```
# Generate n data points, each with p predictors
df = pd.DataFrame()
n = 150
p = 100
for j in range(p):
    xp = 'x'+str(j)
```

```
    df[xp] = np.random.randn(n) # the predictors are independent standard normal
df = sm.add_constant(df)
y = np.random.randn(n) # y is independent standard normal
df.head()
```

[8]:
```
   const        x0        x1        x2        x3        x4        x5  \
0    1.0  0.383591 -0.537063 -0.717021  0.610033  0.137464  0.309159
1    1.0  0.771901  0.756850  0.704892 -0.012215  0.131729 -1.477986
2    1.0 -0.225037  1.092338 -0.311512 -0.301447 -0.132149  0.514359
3    1.0 -0.516535  2.101092  0.425625  0.456291  0.559512 -0.383371
4    1.0  0.142738 -1.331120 -0.027631  1.393236 -2.259672 -0.413737

         x6        x7        x8  …       x90       x91       x92       x93  \
0  0.728812 -1.383054 -0.503435  … -0.143675 -0.827860 -0.050531 -0.400625
1  1.523221  0.457930  1.102413  …  0.510880 -0.185643 -2.166296 -1.336478
2 -1.726471 -0.063883 -0.408150  … -0.028491 -0.022345  1.203843 -0.208912
3  1.497923 -1.320909  0.980372  … -1.048141 -2.585921 -1.465636  0.571273
4 -0.627031 -0.530243 -1.576605  … -0.650646 -0.632516 -0.077412  0.204340

        x94       x95       x96       x97       x98       x99
0 -0.294959 -1.055640 -0.696595 -0.041200 -0.917636  1.325042
1  0.215508 -1.698482  0.244187  0.734012  1.210038 -1.019763
2 -1.590956 -0.113082 -1.404412  0.204309 -2.103601 -1.245443
3 -0.062290 -1.636470 -0.133300  0.273599 -0.920997 -0.481804
4  1.135676 -0.192141 -0.651783  0.630529 -1.856983 -0.064061

[5 rows x 101 columns]
```

[9]:
```
# Fit linear regression
model = sm.OLS(y,df).fit()
model.summary()
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.631
Model:                            OLS   Adj. R-squared:                 -0.122
Method:                 Least Squares   F-statistic:                    0.8380
Date:                Thu, 14 Apr 2022   Prob (F-statistic):              0.773
Time:                        23:48:09   Log-Likelihood:                -145.38
No. Observations:                 150   AIC:                             492.8
Df Residuals:                      49   BIC:                             796.8
Df Model:                         100
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
```

const	-0.0857	0.163	-0.527	0.600	-0.412	0.241
x0	-0.0303	0.150	-0.202	0.840	-0.331	0.270
x1	0.1706	0.146	1.168	0.248	-0.123	0.464
x2	-0.1438	0.151	-0.953	0.345	-0.447	0.160
x3	-0.0107	0.172	-0.062	0.951	-0.356	0.335
x4	0.0179	0.148	0.121	0.904	-0.280	0.316
x5	0.1730	0.163	1.063	0.293	-0.154	0.500
x6	-0.0023	0.150	-0.015	0.988	-0.305	0.300
x7	-0.1770	0.188	-0.944	0.350	-0.554	0.200
x8	0.4127	0.166	2.486	0.016	0.079	0.746
x9	0.1817	0.147	1.234	0.223	-0.114	0.478
x10	0.0602	0.166	0.364	0.718	-0.272	0.393
x11	0.2569	0.147	1.742	0.088	-0.039	0.553
x12	0.2368	0.145	1.628	0.110	-0.055	0.529
x13	-0.0570	0.147	-0.388	0.700	-0.352	0.238
x14	0.0165	0.168	0.098	0.922	-0.321	0.354
x15	0.0888	0.164	0.543	0.590	-0.240	0.418
x16	0.1777	0.165	1.078	0.286	-0.154	0.509
x17	0.2473	0.152	1.624	0.111	-0.059	0.553
x18	0.1382	0.167	0.826	0.413	-0.198	0.474
x19	-0.1421	0.161	-0.881	0.383	-0.466	0.182
x20	-0.0064	0.146	-0.044	0.965	-0.299	0.286
x21	0.1850	0.153	1.212	0.231	-0.122	0.492
x22	-0.0398	0.162	-0.246	0.807	-0.365	0.286
x23	0.0003	0.128	0.003	0.998	-0.256	0.257
x24	-0.0243	0.165	-0.147	0.883	-0.355	0.307
x25	-0.0712	0.152	-0.469	0.641	-0.376	0.234
x26	0.1842	0.143	1.284	0.205	-0.104	0.472
x27	0.0583	0.149	0.391	0.698	-0.242	0.359
x28	0.0629	0.190	0.331	0.742	-0.320	0.445
x29	0.1170	0.142	0.826	0.413	-0.168	0.401
x30	-0.0598	0.168	-0.357	0.723	-0.397	0.277
x31	-0.2116	0.143	-1.477	0.146	-0.500	0.076
x32	-0.0754	0.174	-0.433	0.667	-0.425	0.275
x33	0.1837	0.150	1.226	0.226	-0.117	0.485
x34	-0.0943	0.168	-0.563	0.576	-0.431	0.242
x35	0.0666	0.143	0.465	0.644	-0.222	0.355
x36	-0.0510	0.139	-0.367	0.715	-0.330	0.228
x37	0.1855	0.156	1.186	0.241	-0.129	0.500
x38	-0.0071	0.175	-0.041	0.968	-0.358	0.344
x39	0.1184	0.173	0.684	0.497	-0.230	0.466
x40	0.1857	0.149	1.245	0.219	-0.114	0.485
x41	0.1978	0.137	1.439	0.157	-0.079	0.474
x42	-0.1407	0.166	-0.846	0.402	-0.475	0.194
x43	-0.0333	0.144	-0.232	0.818	-0.322	0.256
x44	-0.1076	0.137	-0.785	0.436	-0.383	0.168

x45	-0.0659	0.152	-0.432	0.667	-0.372	0.240
x46	-0.3318	0.146	-2.271	0.028	-0.625	-0.038
x47	-0.3066	0.160	-1.921	0.060	-0.627	0.014
x48	-0.0465	0.137	-0.339	0.736	-0.322	0.229
x49	-0.0179	0.140	-0.128	0.899	-0.299	0.263
x50	0.1153	0.160	0.720	0.475	-0.206	0.437
x51	-0.0121	0.154	-0.079	0.938	-0.321	0.297
x52	0.0194	0.154	0.126	0.900	-0.291	0.329
x53	-0.0354	0.135	-0.262	0.795	-0.307	0.236
x54	-0.1067	0.153	-0.698	0.488	-0.413	0.200
x55	-0.2340	0.132	-1.773	0.082	-0.499	0.031
x56	-0.2137	0.160	-1.334	0.188	-0.536	0.108
x57	0.1222	0.154	0.795	0.431	-0.187	0.431
x58	0.3492	0.149	2.339	0.023	0.049	0.649
x59	-0.0711	0.135	-0.528	0.600	-0.342	0.199
x60	-0.0950	0.172	-0.553	0.583	-0.441	0.251
x61	0.0966	0.168	0.574	0.569	-0.242	0.435
x62	-0.1816	0.181	-1.002	0.321	-0.546	0.183
x63	-0.0057	0.141	-0.040	0.968	-0.290	0.278
x64	-0.0745	0.144	-0.518	0.607	-0.364	0.215
x65	-0.1484	0.150	-0.992	0.326	-0.449	0.152
x66	0.1675	0.167	1.000	0.322	-0.169	0.504
x67	0.0122	0.155	0.079	0.938	-0.299	0.323
x68	-0.1382	0.152	-0.910	0.367	-0.443	0.167
x69	0.3445	0.186	1.848	0.071	-0.030	0.719
x70	0.4156	0.151	2.752	0.008	0.112	0.719
x71	-0.0054	0.170	-0.032	0.975	-0.347	0.336
x72	-0.0681	0.150	-0.454	0.652	-0.369	0.233
x73	-0.1882	0.147	-1.276	0.208	-0.485	0.108
x74	0.0530	0.151	0.351	0.727	-0.250	0.356
x75	-0.0271	0.168	-0.161	0.872	-0.365	0.311
x76	0.0336	0.166	0.203	0.840	-0.299	0.367
x77	-0.1172	0.142	-0.827	0.412	-0.402	0.167
x78	-0.3093	0.145	-2.128	0.038	-0.601	-0.017
x79	0.1131	0.166	0.683	0.498	-0.220	0.446
x80	-0.1876	0.167	-1.125	0.266	-0.523	0.148
x81	0.1054	0.160	0.659	0.513	-0.216	0.427
x82	0.0621	0.149	0.417	0.678	-0.237	0.361
x83	-0.1455	0.163	-0.893	0.376	-0.473	0.182
x84	-0.0307	0.158	-0.194	0.847	-0.348	0.286
x85	-0.2824	0.173	-1.637	0.108	-0.629	0.064
x86	-0.1637	0.186	-0.882	0.382	-0.537	0.209
x87	0.1152	0.134	0.860	0.394	-0.154	0.385
x88	0.0121	0.172	0.070	0.944	-0.334	0.358
x89	-0.1152	0.157	-0.735	0.466	-0.430	0.200
x90	0.0071	0.169	0.042	0.967	-0.332	0.346
x91	-0.0061	0.168	-0.036	0.971	-0.345	0.332

```
x92           -0.0762      0.147     -0.518      0.607      -0.372       0.219
x93            0.1452      0.144      1.006      0.319      -0.145       0.435
x94            0.0675      0.149      0.454      0.652      -0.231       0.366
x95            0.0895      0.145      0.617      0.540      -0.202       0.381
x96           -0.0042      0.171     -0.024      0.981      -0.347       0.339
x97            0.3317      0.157      2.114      0.040       0.016       0.647
x98            0.0072      0.166      0.043      0.966      -0.327       0.342
x99            0.0638      0.174      0.368      0.715      -0.285       0.413
==============================================================================
Omnibus:                        5.319   Durbin-Watson:                   1.490
Prob(Omnibus):                  0.070   Jarque-Bera (JB):                5.014
Skew:                           0.346   Prob(JB):                       0.0815
Kurtosis:                       3.568   Cond. No.                         7.68
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[10]:
```
# R-squared is high
model.rsquared
```

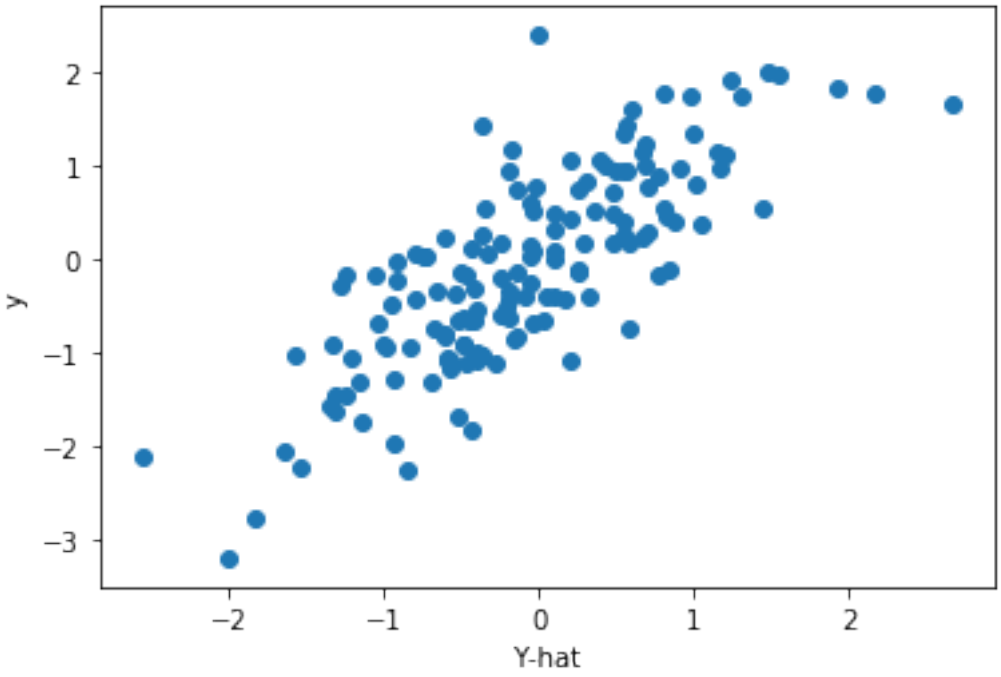
[10]:
```
0.6310223932876349
```

[11]:
```
# There are a lot of stat sig covariates
model.pvalues[model.pvalues<0.05]
```

[11]:
```
x8     0.016389
x46    0.027545
x58    0.023460
x70    0.008289
x78    0.038386
x97    0.039614
dtype: float64
```

[12]:
```
# The scatterplot shows correlation
plt.scatter(model.predict(df),y)
plt.ylabel('y')
plt.xlabel('Y-hat')
```

[12]:
```
Text(0.5, 0, 'Y-hat')
```



2.0.1 Solution: held out test set

[14]:
```
X = df
```

[15]:
```
X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.5, random_state=1)
print("Training set: ", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
```

```
Training set:  (75, 101) (75,)
Test set: (75, 101) (75,)
```

2.0.2 Solution: fit model using the training set, produce plots on test set

[17]:
```
model = sm.OLS(y_train,X_train).fit()
model.summary()
# The p-values are nan because the number of datapoints is smaller than the␣
 ↪number of predictors
```

[17]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                    nan
```

Method:	Least Squares	F-statistic:	0.000
Date:	Thu, 14 Apr 2022	Prob (F-statistic):	nan
Time:	23:48:35	Log-Likelihood:	2427.2
No. Observations:	75	AIC:	-4704.
Df Residuals:	0	BIC:	-4531.
Df Model:	74		
Covariance Type:	nonrobust		

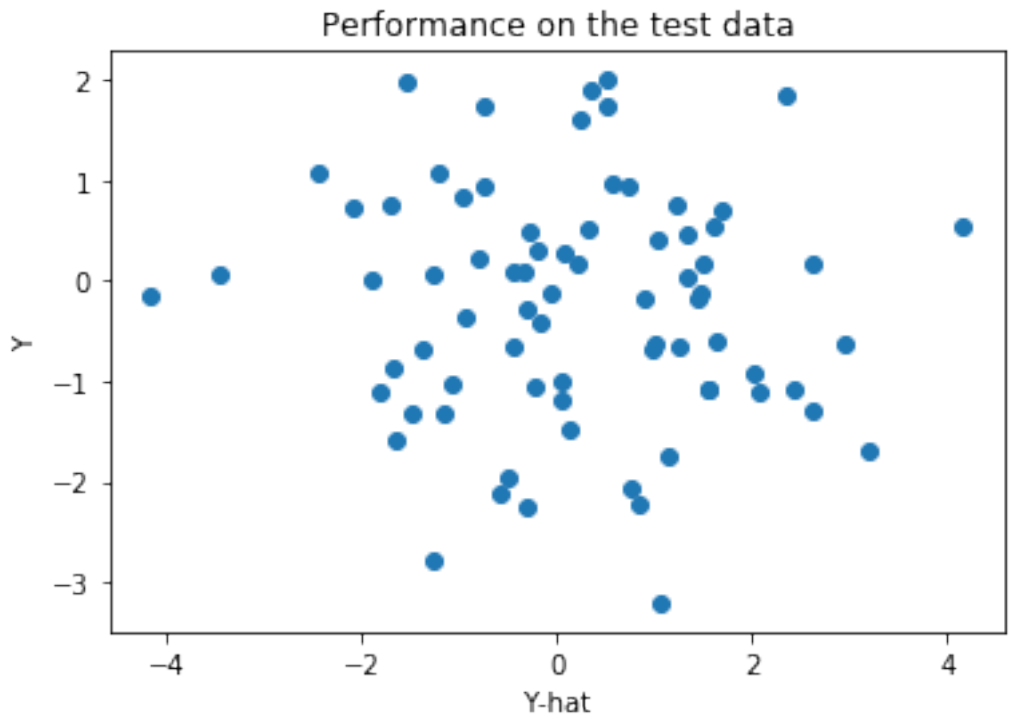
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1937	inf	0	nan	nan	nan
x0	0.0862	inf	0	nan	nan	nan
x1	0.0069	inf	0	nan	nan	nan
x2	0.0313	inf	0	nan	nan	nan
x3	0.0282	inf	0	nan	nan	nan
x4	0.0963	inf	0	nan	nan	nan
x5	0.0422	inf	0	nan	nan	nan
x6	-0.0658	inf	-0	nan	nan	nan
x7	0.0433	inf	0	nan	nan	nan
x8	0.1779	inf	0	nan	nan	nan
x9	-0.0757	inf	-0	nan	nan	nan
x10	0.0320	inf	0	nan	nan	nan
x11	0.0023	inf	0	nan	nan	nan
x12	-0.1675	inf	-0	nan	nan	nan
x13	0.0213	inf	0	nan	nan	nan
x14	0.1364	inf	0	nan	nan	nan
x15	0.1265	inf	0	nan	nan	nan
x16	0.2727	inf	0	nan	nan	nan
x17	-0.0713	inf	-0	nan	nan	nan
x18	0.0768	inf	0	nan	nan	nan
x19	-0.0221	inf	-0	nan	nan	nan
x20	-0.2043	inf	-0	nan	nan	nan
x21	-0.0247	inf	-0	nan	nan	nan
x22	-0.0657	inf	-0	nan	nan	nan
x23	-0.2989	inf	-0	nan	nan	nan
x24	-0.1376	inf	-0	nan	nan	nan
x25	-0.0563	inf	-0	nan	nan	nan
x26	0.2096	inf	0	nan	nan	nan
x27	0.2526	inf	0	nan	nan	nan
x28	-0.0587	inf	-0	nan	nan	nan
x29	0.1058	inf	0	nan	nan	nan
x30	-0.0939	inf	-0	nan	nan	nan
x31	-0.1173	inf	-0	nan	nan	nan
x32	-0.0656	inf	-0	nan	nan	nan
x33	0.0330	inf	0	nan	nan	nan
x34	0.0288	inf	0	nan	nan	nan
x35	0.0426	inf	0	nan	nan	nan

x36	0.0136	inf	0	nan	nan	nan
x37	0.0398	inf	0	nan	nan	nan
x38	0.0336	inf	0	nan	nan	nan
x39	0.0920	inf	0	nan	nan	nan
x40	-0.1476	inf	-0	nan	nan	nan
x41	0.2662	inf	0	nan	nan	nan
x42	-0.4084	inf	-0	nan	nan	nan
x43	-0.0104	inf	-0	nan	nan	nan
x44	-0.0468	inf	-0	nan	nan	nan
x45	0.1276	inf	0	nan	nan	nan
x46	0.0225	inf	0	nan	nan	nan
x47	-0.1516	inf	-0	nan	nan	nan
x48	-0.0761	inf	-0	nan	nan	nan
x49	-0.0088	inf	-0	nan	nan	nan
x50	-0.2068	inf	-0	nan	nan	nan
x51	-0.0031	inf	-0	nan	nan	nan
x52	-0.1885	inf	-0	nan	nan	nan
x53	-0.1501	inf	-0	nan	nan	nan
x54	-0.0557	inf	-0	nan	nan	nan
x55	-0.0704	inf	-0	nan	nan	nan
x56	0.0923	inf	0	nan	nan	nan
x57	0.1927	inf	0	nan	nan	nan
x58	0.0409	inf	0	nan	nan	nan
x59	-0.0454	inf	-0	nan	nan	nan
x60	-0.0930	inf	-0	nan	nan	nan
x61	0.0059	inf	0	nan	nan	nan
x62	-0.0791	inf	-0	nan	nan	nan
x63	0.1501	inf	0	nan	nan	nan
x64	0.0784	inf	0	nan	nan	nan
x65	-0.2142	inf	-0	nan	nan	nan
x66	0.2749	inf	0	nan	nan	nan
x67	0.1251	inf	0	nan	nan	nan
x68	0.0039	inf	0	nan	nan	nan
x69	-0.0762	inf	-0	nan	nan	nan
x70	-0.0107	inf	-0	nan	nan	nan
x71	-0.1068	inf	-0	nan	nan	nan
x72	0.1573	inf	0	nan	nan	nan
x73	0.1257	inf	0	nan	nan	nan
x74	-0.0066	inf	-0	nan	nan	nan
x75	0.2636	inf	0	nan	nan	nan
x76	0.0655	inf	0	nan	nan	nan
x77	-0.3448	inf	-0	nan	nan	nan
x78	-0.0298	inf	-0	nan	nan	nan
x79	-0.0221	inf	-0	nan	nan	nan
x80	-0.1618	inf	-0	nan	nan	nan
x81	0.1274	inf	0	nan	nan	nan
x82	-0.0389	inf	-0	nan	nan	nan

```
x83            0.2513        inf           0        nan         nan         nan
x84            0.0550        inf           0        nan         nan         nan
x85           -0.2670        inf          -0        nan         nan         nan
x86           -0.1975        inf          -0        nan         nan         nan
x87           -0.0104        inf          -0        nan         nan         nan
x88           -0.0401        inf          -0        nan         nan         nan
x89            0.0700        inf           0        nan         nan         nan
x90           -0.0205        inf          -0        nan         nan         nan
x91            0.1657        inf           0        nan         nan         nan
x92           -0.0936        inf          -0        nan         nan         nan
x93            0.0685        inf           0        nan         nan         nan
x94            0.1142        inf           0        nan         nan         nan
x95            0.2447        inf           0        nan         nan         nan
x96            0.1389        inf           0        nan         nan         nan
x97            0.1927        inf           0        nan         nan         nan
x98           -0.1564        inf          -0        nan         nan         nan
x99            0.0251        inf           0        nan         nan         nan
==============================================================================
Omnibus:                        0.124   Durbin-Watson:                   1.943
Prob(Omnibus):                  0.940   Jarque-Bera (JB):                0.230
Skew:                           0.090   Prob(JB):                        0.891
Kurtosis:                       2.798   Cond. No.                         9.42
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The input rank is higher than the number of observations.
"""
```

[18]:
```
plt.scatter(model.predict(X_test),y_test)
plt.xlabel('Y-hat')
plt.ylabel('Y')
plt.title('Performance on the test data')
```

[18]:
```
Text(0.5, 1.0, 'Performance on the test data')
```



[21]:
```
# Sum of squares for the trained model is poor
# compared to simply predicting 0, below
np.sum((model.predict(X_test)-y_test)**2)
```

[21]: 307.34493796477454

[20]:
```
# Sum of squares for simply predicting 0
np.sum(y_test**2)
```

[20]: 105.3234363907932