# linear-regression2-2022

March 16, 2022

## 1 Linear Regression Demo 2 (Lecture 12)

Load packages we will use

[2]:
```
# math
import numpy as np

# dataframes
import pandas as pd

# plotting
import matplotlib.pyplot as plt
import seaborn

import statsmodels.api as sm
import statsmodels.formula.api as smf

# for choosing covariates to include in model
from patsy import dmatrices
```

- What is the distribution of beta-hat_j?
- What do confidence intervals mean?
- What do p-values mean?

[16]:
```
# This code fits linear regression
n = 20
x = np.random.randn(n) # generates a series of n Normal(0,1) random variables
eps = np.random.randn(n) # generates another series of n Normal(0,1) random␣
 ↪variables
y = 1 + 2 * x + eps
x = sm.add_constant(x)
model = sm.OLS(y, x).fit()
```

[21]:
```
# Estimate betahat_1 for beta_1
model.params[1]
```

[21]: 2.1740831907960456

[22]:
```
# Get the confidence interval for beta_1
model.conf_int()[1]
```

[22]:
```
array([1.68488886, 2.66327753])
```

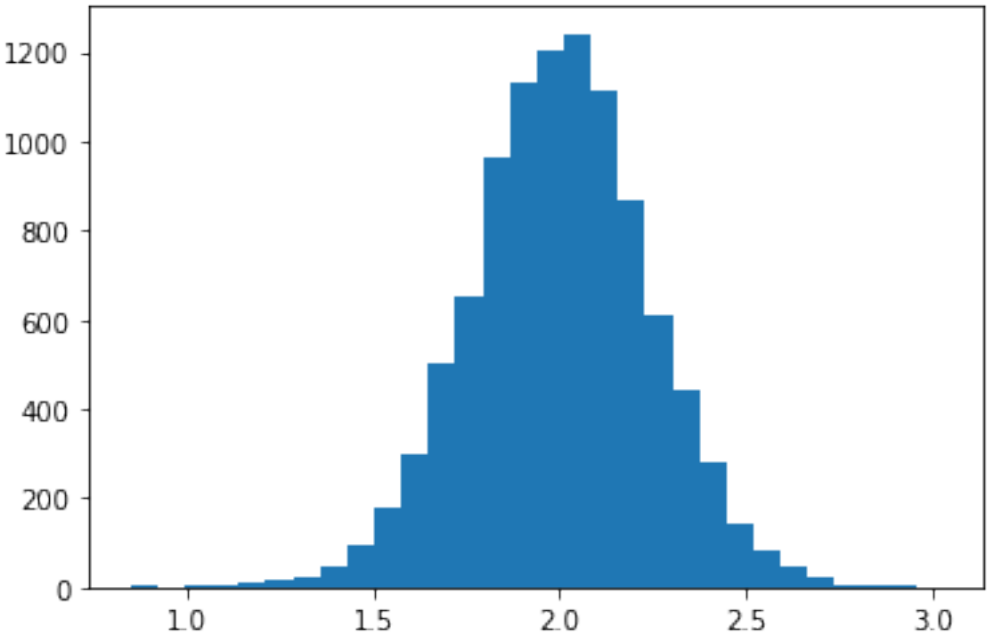
[23]:
```
# p-value for beta_1
model.pvalues[1]
```

[23]:
```
2.5395656695538832e-08
```

[54]:
```
pvalues = []
betahat1 = []
lcb = []
ucb = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random␣
 ↪variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random␣
 ↪variables
    y = 1 + 2 * x + eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    betahat1.append(model.params[1]) # extracts betahat1 from the model and␣
 ↪adds it to a list
    pvalues.append(model.pvalues[1])
    lcb.append(model.conf_int()[1][0])
    ucb.append(model.conf_int()[1][1])
```

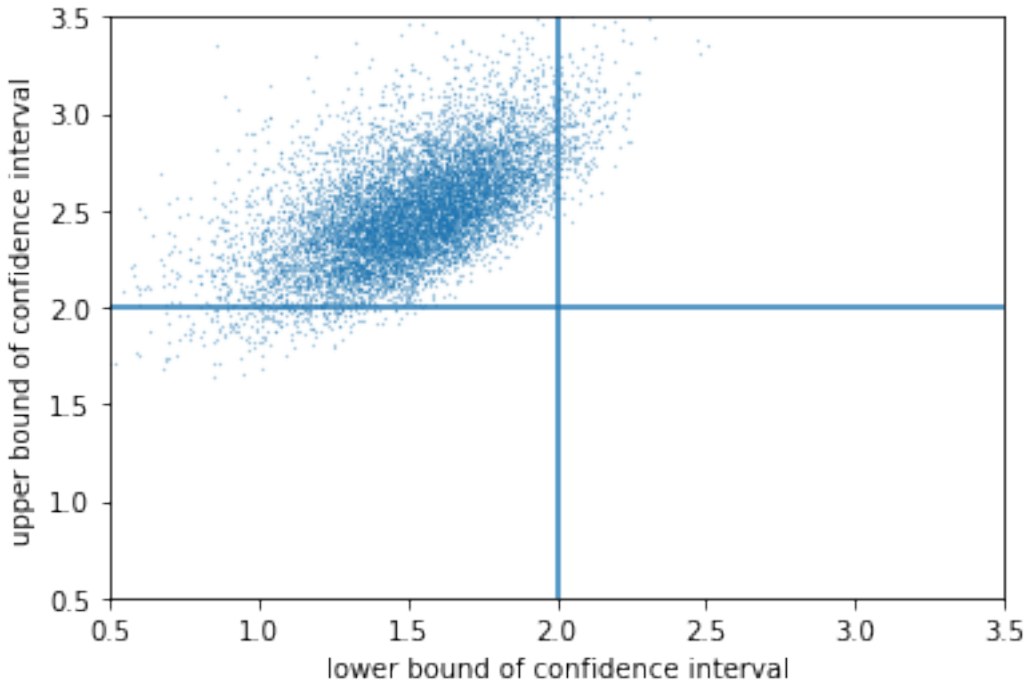
[55]:
```
plt.hist(betahat1,bins=30)
```

[55]:
```
(array([2.000e+00, 0.000e+00, 2.000e+00, 5.000e+00, 1.000e+01, 1.900e+01,
        2.300e+01, 4.900e+01, 9.700e+01, 1.760e+02, 2.980e+02, 5.030e+02,
        6.530e+02, 9.610e+02, 1.130e+03, 1.206e+03, 1.241e+03, 1.112e+03,
        8.700e+02, 6.100e+02, 4.430e+02, 2.800e+02, 1.420e+02, 8.200e+01,
        4.700e+01, 2.200e+01, 6.000e+00, 7.000e+00, 3.000e+00, 1.000e+00]),
 array([0.84883798, 0.9215122 , 0.99418642, 1.06686064, 1.13953486,
        1.21220908, 1.2848833 , 1.35755752, 1.43023174, 1.50290596,
        1.57558018, 1.64825439, 1.72092861, 1.79360283, 1.86627705,
        1.93895127, 2.01162549, 2.08429971, 2.15697393, 2.22964815,
        2.30232237, 2.37499659, 2.44767081, 2.52034503, 2.59301924,
        2.66569346, 2.73836768, 2.8110419 , 2.88371612, 2.95639034,
        3.02906456]),
 <a list of 30 Patch objects>)
```




[44]:
```
plt.plot(lcb,ucb,'.',markersize=0.3)
plt.axhline(2)
plt.axvline(2)
plt.xlim(0.5,3.5)
plt.ylim(0.5,3.5)
plt.xlabel('lower bound of confidence interval')
plt.ylabel('upper bound of confidence interval')
```

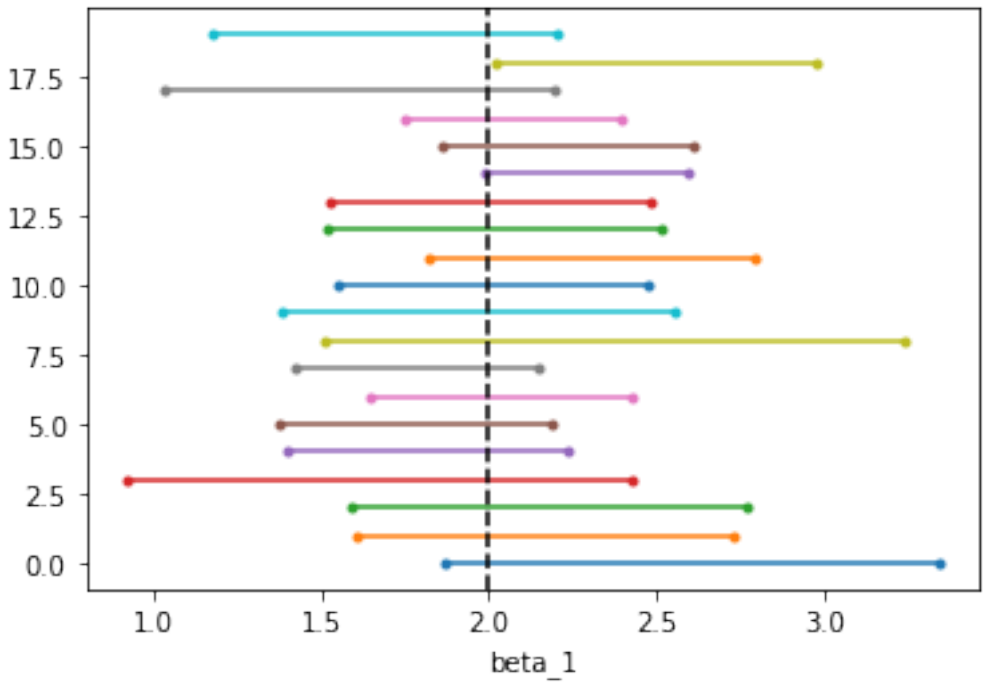
[44]: Text(0, 0.5, 'upper bound of confidence interval')




[51]:
```
for i in range(20):
    plt.plot([lcb[i],ucb[i]],[i,i],'.-')

plt.xlabel('beta_1')
plt.axvline(2,color='black',ls='--')
```

[51]: <matplotlib.lines.Line2D at 0x7fe38066db90>




[ ]:
```
lcb
```

[58]:
```
model.tvalues
```

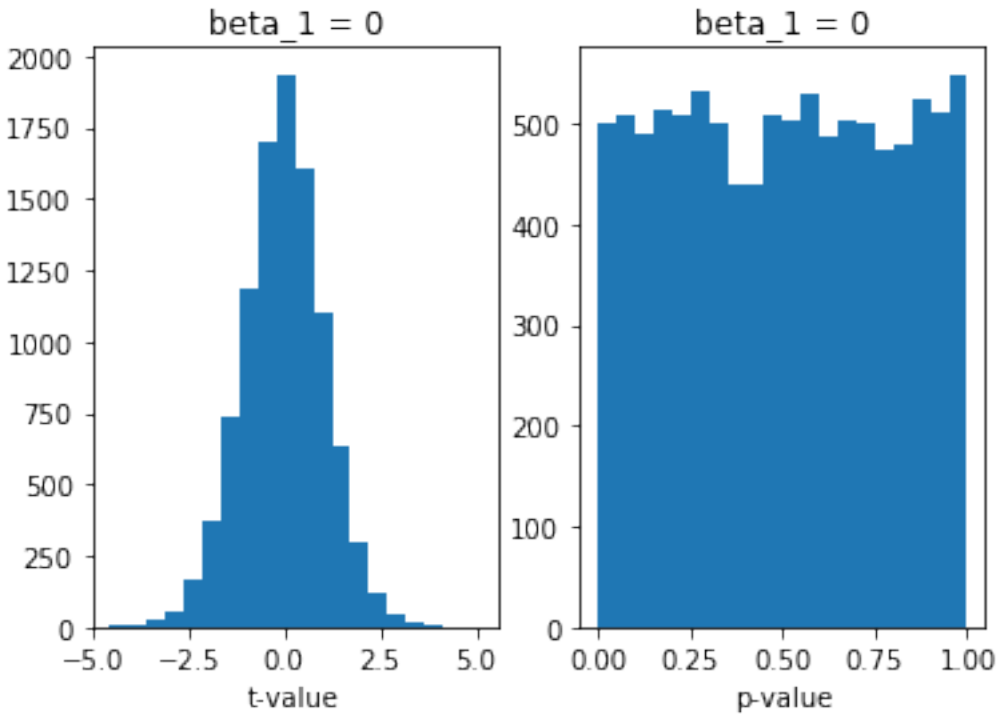
[58]:
```
array([4.4794539 , 8.75388319])
```

## 2 What do p values mean?

[69]:
```
pvalues = []
tvalues = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random␣
 ↪variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random␣
 ↪variables
    y = eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    tvalues.append(model.tvalues[1])
    pvalues.append(model.pvalues[1])
```

[70]:
```
plt.subplot(1,2,1)
plt.hist(tvalues,bins=20)
plt.xlabel('t-value')
plt.title('beta_1 = 0')
plt.subplot(1,2,2)
plt.hist(pvalues, bins=20)
plt.xlabel('p-value')
plt.title('beta_1 = 0')
```

[70]:
```
Text(0.5, 1.0, 'beta_1 = 0')
```

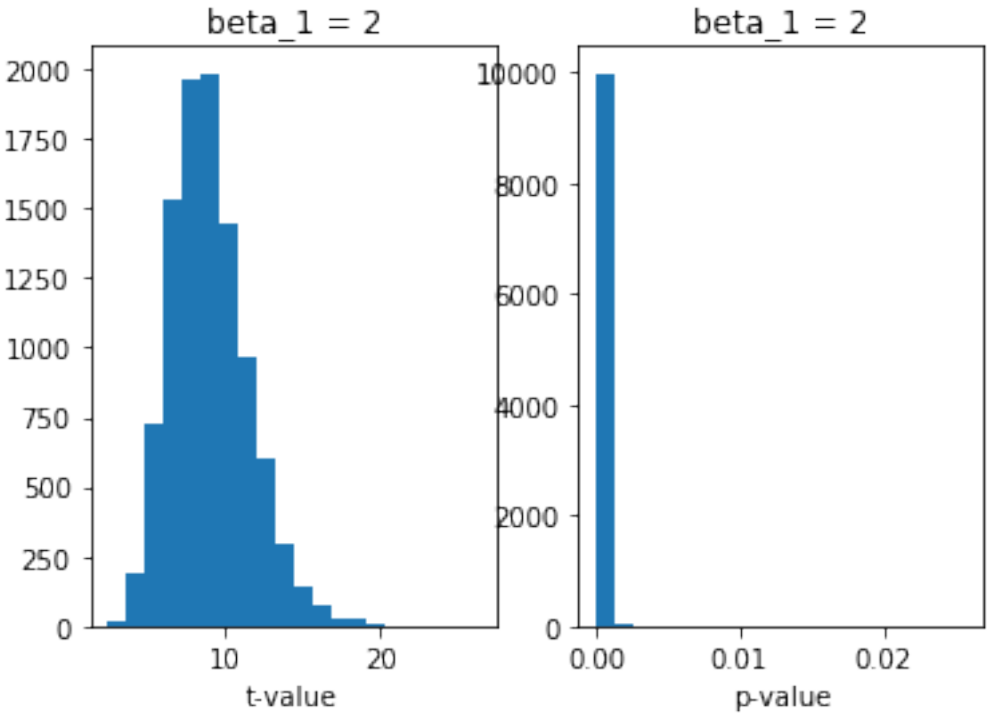



[71]:
```
pvalues = []
tvalues = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random␣
↪variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random␣
↪variables
    y = 2*x + eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    tvalues.append(model.tvalues[1])
```

```
    pvalues.append(model.pvalues[1])
```

[72]:
```
plt.subplot(1,2,1)
plt.hist(tvalues,bins=20)
plt.xlabel('t-value')
plt.title('beta_1 = 2')
plt.subplot(1,2,2)
plt.hist(pvalues, bins=20)
plt.xlabel('p-value')
plt.title('beta_1 = 2')
```

[72]:
```
Text(0.5, 1.0, 'beta_1 = 2')
```




[44]:
```
# around 5% of the pvalues are <.05, just by random chance
sum(np.array(pvalues) <= .05) / len(pvalues)
```

[44]:
```
0.047
```

## 3 Model selection and p values: be careful!

[39]:
```
df = pd.DataFrame()
n = 500
p = 100
for j in range(p):
    xp = 'x'+str(j)
    df[xp] = np.random.randn(n)
y = np.random.randn(n)
df.head()
```

[39]:
```
         x0        x1        x2        x3        x4        x5        x6  \
0  0.374605 -1.471182  0.272202  0.760021 -1.222600 -0.122771 -0.412346
1  0.047311 -0.457857  1.967839 -0.274292  2.526630 -0.601652  0.213363
2  0.231405 -1.435710 -0.345093 -1.780339 -0.091606  1.010120  0.312508
3 -0.657278 -0.902671 -0.217043  0.467411 -1.233392 -1.511210 -0.846402
4 -2.233219 -0.421696 -0.279707  1.137544 -0.380108 -0.312297 -0.533366

         x7        x8        x9  …       x90       x91       x92       x93  \
0  0.356779  1.325712 -0.333066  …  0.687681  2.820019  0.632996 -0.831833
1 -1.114084  0.844745 -0.618595  …  0.383744 -0.646661  1.505726  0.231351
2  0.628716  0.899247  1.185614  …  1.383114 -2.111851  1.184291 -0.243784
3  0.226872  0.047084  0.125291  … -0.425932  0.974371 -0.429321  0.979431
4 -1.390396  0.374429  0.649990  …  1.057409 -0.367487 -1.722719 -0.135082

        x94       x95       x96       x97       x98       x99
0 -0.213222  0.042854  1.308103 -0.730357 -1.113284 -0.267020
1 -0.415279  0.322218 -0.579459 -1.007643  0.254060 -1.227380
2  0.851140  0.068269  0.459190  0.553544 -1.100938 -0.655753
3  0.925717  1.129974  0.280579  1.306097 -0.195457  0.923107
4  0.827299  1.586168 -0.349127  1.093985  0.530184 -0.216399

[5 rows x 100 columns]
```

[40]:
```
variables = df.columns
model = sm.OLS(y,df[variables]).fit()
model.summary()
```

[40]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                        y   R-squared (uncentered):
0.191
Model:                              OLS   Adj. R-squared (uncentered):
-0.011
```

Method: Least Squares F-statistic:
0.9453
Date: Sun, 28 Mar 2021 Prob (F-statistic):
0.626
Time: 19:45:12 Log-Likelihood:
-655.35
No. Observations: 500 AIC:
1511.
Df Residuals: 400 BIC:
1932.
Df Model: 100
Covariance Type: nonrobust

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| x0 | -0.0236 | 0.049 | -0.481 | 0.631 | -0.120 | 0.073 |
| x1 | -0.0118 | 0.049 | -0.239 | 0.811 | -0.109 | 0.085 |
| x2 | 0.0385 | 0.050 | 0.773 | 0.440 | -0.059 | 0.136 |
| x3 | 0.0630 | 0.050 | 1.255 | 0.210 | -0.036 | 0.162 |
| x4 | 0.0109 | 0.049 | 0.224 | 0.823 | -0.085 | 0.107 |
| x5 | 0.1144 | 0.052 | 2.208 | 0.028 | 0.013 | 0.216 |
| x6 | 0.0461 | 0.050 | 0.929 | 0.353 | -0.051 | 0.144 |
| x7 | 0.0500 | 0.051 | 0.984 | 0.326 | -0.050 | 0.150 |
| x8 | 0.0146 | 0.048 | 0.301 | 0.763 | -0.081 | 0.110 |
| x9 | 0.0721 | 0.049 | 1.473 | 0.141 | -0.024 | 0.168 |
| x10 | 0.0491 | 0.050 | 0.991 | 0.322 | -0.048 | 0.147 |
| x11 | 0.0092 | 0.052 | 0.175 | 0.862 | -0.094 | 0.112 |
| x12 | -0.0792 | 0.048 | -1.634 | 0.103 | -0.175 | 0.016 |
| x13 | 0.0771 | 0.050 | 1.527 | 0.128 | -0.022 | 0.176 |
| x14 | 0.0669 | 0.048 | 1.382 | 0.168 | -0.028 | 0.162 |
| x15 | -0.0308 | 0.050 | -0.621 | 0.535 | -0.128 | 0.067 |
| x16 | -0.0540 | 0.049 | -1.102 | 0.271 | -0.150 | 0.042 |
| x17 | 0.0601 | 0.052 | 1.158 | 0.247 | -0.042 | 0.162 |
| x18 | 0.0446 | 0.054 | 0.824 | 0.410 | -0.062 | 0.151 |
| x19 | 0.0354 | 0.050 | 0.706 | 0.481 | -0.063 | 0.134 |
| x20 | -0.0883 | 0.050 | -1.768 | 0.078 | -0.186 | 0.010 |
| x21 | -0.0099 | 0.053 | -0.187 | 0.852 | -0.114 | 0.094 |
| x22 | -0.0504 | 0.051 | -0.996 | 0.320 | -0.150 | 0.049 |
| x23 | -0.0060 | 0.050 | -0.121 | 0.903 | -0.104 | 0.092 |
| x24 | 0.0444 | 0.053 | 0.842 | 0.400 | -0.059 | 0.148 |
| x25 | 0.0414 | 0.050 | 0.828 | 0.408 | -0.057 | 0.140 |
| x26 | -0.0685 | 0.051 | -1.335 | 0.183 | -0.169 | 0.032 |
| x27 | 0.1020 | 0.053 | 1.933 | 0.054 | -0.002 | 0.206 |
| x28 | -0.0886 | 0.046 | -1.923 | 0.055 | -0.179 | 0.002 |
| x29 | -0.0480 | 0.054 | -0.897 | 0.370 | -0.153 | 0.057 |
| x30 | -0.0610 | 0.048 | -1.261 | 0.208 | -0.156 | 0.034 |
| x31 | 0.0035 | 0.051 | 0.070 | 0.944 | -0.096 | 0.103 |

| | | | | | | |
|---|---|---|---|---|---|---|
| x32 | -0.0455 | 0.050 | -0.910 | 0.363 | -0.144 | 0.053 |
| x33 | -0.0129 | 0.050 | -0.256 | 0.798 | -0.112 | 0.086 |
| x34 | -0.0679 | 0.048 | -1.413 | 0.158 | -0.162 | 0.027 |
| x35 | 0.0502 | 0.051 | 0.983 | 0.326 | -0.050 | 0.150 |
| x36 | 0.0263 | 0.049 | 0.534 | 0.594 | -0.071 | 0.123 |
| x37 | -0.0862 | 0.050 | -1.734 | 0.084 | -0.184 | 0.012 |
| x38 | 0.0018 | 0.050 | 0.036 | 0.971 | -0.096 | 0.100 |
| x39 | -0.0069 | 0.051 | -0.137 | 0.891 | -0.107 | 0.093 |
| x40 | -0.0082 | 0.050 | -0.165 | 0.869 | -0.106 | 0.089 |
| x41 | 0.0456 | 0.052 | 0.871 | 0.384 | -0.057 | 0.149 |
| x42 | -0.0337 | 0.052 | -0.646 | 0.519 | -0.136 | 0.069 |
| x43 | -0.0701 | 0.048 | -1.455 | 0.146 | -0.165 | 0.025 |
| x44 | 0.0229 | 0.053 | 0.433 | 0.665 | -0.081 | 0.127 |
| x45 | -0.1215 | 0.051 | -2.380 | 0.018 | -0.222 | -0.021 |
| x46 | 0.0419 | 0.049 | 0.849 | 0.397 | -0.055 | 0.139 |
| x47 | -0.0270 | 0.051 | -0.535 | 0.593 | -0.126 | 0.072 |
| x48 | -0.0257 | 0.050 | -0.512 | 0.609 | -0.124 | 0.073 |
| x49 | 0.0927 | 0.049 | 1.878 | 0.061 | -0.004 | 0.190 |
| x50 | -0.0343 | 0.048 | -0.711 | 0.478 | -0.129 | 0.061 |
| x51 | 0.0658 | 0.050 | 1.311 | 0.190 | -0.033 | 0.164 |
| x52 | 0.0420 | 0.051 | 0.817 | 0.414 | -0.059 | 0.143 |
| x53 | -0.0072 | 0.048 | -0.150 | 0.881 | -0.102 | 0.087 |
| x54 | -0.0225 | 0.050 | -0.448 | 0.654 | -0.121 | 0.076 |
| x55 | 0.0145 | 0.054 | 0.270 | 0.787 | -0.091 | 0.120 |
| x56 | -0.0471 | 0.052 | -0.908 | 0.364 | -0.149 | 0.055 |
| x57 | 0.0023 | 0.050 | 0.045 | 0.964 | -0.097 | 0.101 |
| x58 | -0.0101 | 0.049 | -0.205 | 0.838 | -0.107 | 0.087 |
| x59 | -0.0141 | 0.052 | -0.270 | 0.787 | -0.117 | 0.089 |
| x60 | -0.0185 | 0.052 | -0.357 | 0.721 | -0.121 | 0.084 |
| x61 | 0.0219 | 0.047 | 0.462 | 0.644 | -0.071 | 0.115 |
| x62 | -0.0415 | 0.050 | -0.830 | 0.407 | -0.140 | 0.057 |
| x63 | 0.0627 | 0.052 | 1.197 | 0.232 | -0.040 | 0.166 |
| x64 | -0.0983 | 0.049 | -2.019 | 0.044 | -0.194 | -0.003 |
| x65 | -0.0681 | 0.053 | -1.282 | 0.201 | -0.173 | 0.036 |
| x66 | -0.0370 | 0.050 | -0.741 | 0.459 | -0.135 | 0.061 |
| x67 | 0.0384 | 0.049 | 0.781 | 0.435 | -0.058 | 0.135 |
| x68 | -0.0188 | 0.054 | -0.349 | 0.728 | -0.125 | 0.087 |
| x69 | -0.0512 | 0.049 | -1.046 | 0.296 | -0.147 | 0.045 |
| x70 | -0.0413 | 0.052 | -0.794 | 0.428 | -0.144 | 0.061 |
| x71 | 0.0596 | 0.049 | 1.207 | 0.228 | -0.037 | 0.157 |
| x72 | 0.0222 | 0.052 | 0.429 | 0.668 | -0.080 | 0.124 |
| x73 | 0.0486 | 0.047 | 1.023 | 0.307 | -0.045 | 0.142 |
| x74 | 0.0747 | 0.049 | 1.511 | 0.132 | -0.022 | 0.172 |
| x75 | -0.0324 | 0.049 | -0.666 | 0.506 | -0.128 | 0.063 |
| x76 | 0.0316 | 0.049 | 0.650 | 0.516 | -0.064 | 0.127 |
| x77 | 0.0110 | 0.047 | 0.233 | 0.816 | -0.081 | 0.103 |
| x78 | -0.0287 | 0.050 | -0.572 | 0.568 | -0.127 | 0.070 |

```
x79            0.0474      0.052      0.909      0.364      -0.055       0.150
x80            0.0090      0.051      0.178      0.859      -0.090       0.108
x81            0.0338      0.046      0.732      0.465      -0.057       0.125
x82           -0.0218      0.052     -0.423      0.673      -0.123       0.080
x83            0.0411      0.050      0.816      0.415      -0.058       0.140
x84            0.0606      0.050      1.224      0.222      -0.037       0.158
x85           -0.0133      0.051     -0.262      0.793      -0.113       0.087
x86            0.0658      0.051      1.290      0.198      -0.034       0.166
x87            0.0774      0.050      1.537      0.125      -0.022       0.176
x88            0.0510      0.047      1.090      0.276      -0.041       0.143
x89           -0.0282      0.051     -0.551      0.582      -0.129       0.072
x90            0.0402      0.051      0.789      0.431      -0.060       0.140
x91            0.0214      0.048      0.450      0.653      -0.072       0.115
x92           -0.0427      0.050     -0.853      0.394      -0.141       0.056
x93            0.0023      0.049      0.047      0.962      -0.095       0.099
x94            0.0277      0.050      0.550      0.582      -0.071       0.127
x95            0.0459      0.051      0.907      0.365      -0.054       0.145
x96           -0.0118      0.051     -0.233      0.816      -0.111       0.088
x97           -0.0207      0.052     -0.401      0.688      -0.122       0.081
x98           -0.0873      0.051     -1.715      0.087      -0.187       0.013
x99           -0.0745      0.052     -1.427      0.154      -0.177       0.028
==============================================================================
Omnibus:                        0.735   Durbin-Watson:                   1.978
Prob(Omnibus):                  0.692   Jarque-Bera (JB):                0.681
Skew:                           0.090   Prob(JB):                        0.711
Kurtosis:                       3.008   Cond. No.                         2.56
==============================================================================

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

[41]:
```
# uncommment this line the first time you run the cell; comment it out for
 ↪subsequent runs
stat_sig_vars = variables
stat_sig_vars = stat_sig_vars[model.pvalues <= .05]
model = sm.OLS(y,df[stat_sig_vars]).fit()
model.summary()
```

[41]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                        y   R-squared (uncentered):
0.028
```

```
Model:                            OLS   Adj. R-squared (uncentered):
0.022
Method:                 Least Squares   F-statistic:
4.802
Date:                Sun, 28 Mar 2021   Prob (F-statistic):
0.00263
Time:                        19:45:13   Log-Likelihood:
-701.24
No. Observations:                 500   AIC:
1408.
Df Residuals:                     497   BIC:
1421.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x5             0.0657      0.046      1.432      0.153      -0.024       0.156
x45           -0.1287      0.045     -2.869      0.004      -0.217      -0.041
x64           -0.0997      0.043     -2.314      0.021      -0.184      -0.015
==============================================================================
Omnibus:                        0.197   Durbin-Watson:                   1.990
Prob(Omnibus):                  0.906   Jarque-Bera (JB):                0.115
Skew:                           0.032   Prob(JB):                        0.944
Kurtosis:                       3.037   Cond. No.                         1.11
==============================================================================

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

## 4 3. Now let's fit the same model, using a different package: sklearn

[52]:
```
lm = LinearRegression()
X = usage[['temperature','tempsqr']] # sklearn also includes an intercept␣
 ↪automatically
Y = usage['usage']
model = lm.fit(X,Y)
model
```

[52]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[68]:
```
Yhat = lm.predict(X)
Yhat[:5] # This plots the first 5 entries in Yhat.  Yhat is a numpy array, so␣
 ↪there is no head() function
```

[68]:
```
array([21.73113332, 23.75922713, 32.05646535, 53.99847078, 96.83797293])
```

[54]:
```
# how well does the model fit?
lm.score(X,Y) # R^2 score
```

[54]:
```
0.8468764841785454
```

[55]:
```
# what about the coefficients?  The first one is for temperature, the second␣
 ↪for tempsqr
lm.coef_
```

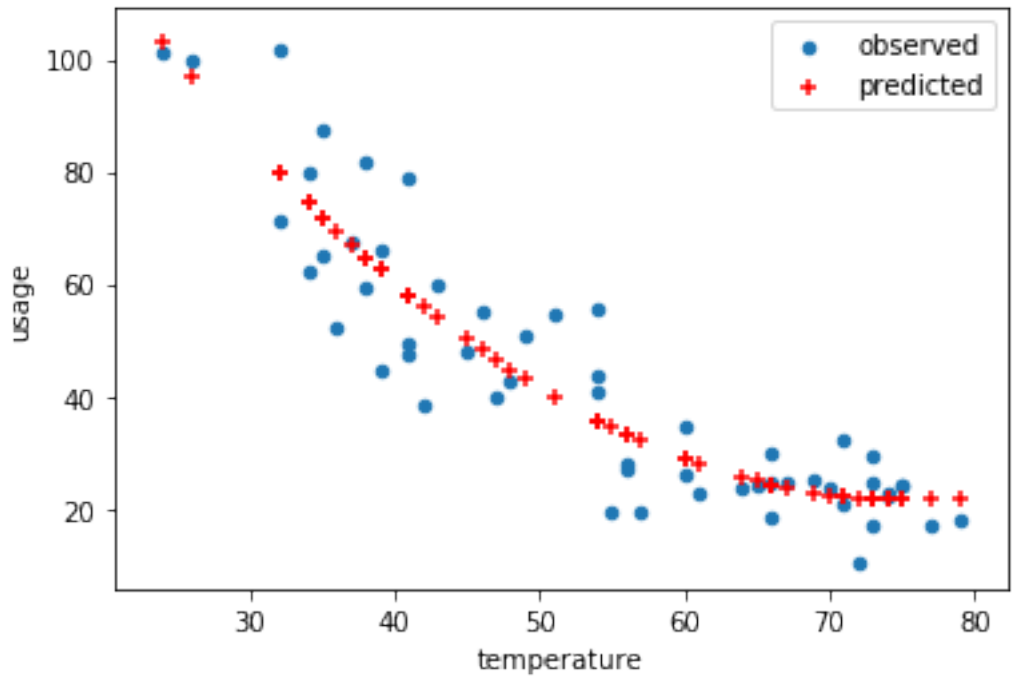
[55]:
```
array([-4.64046227,  0.03073176])
```

[56]:
```
# and the intercept _0
lm.intercept_
```

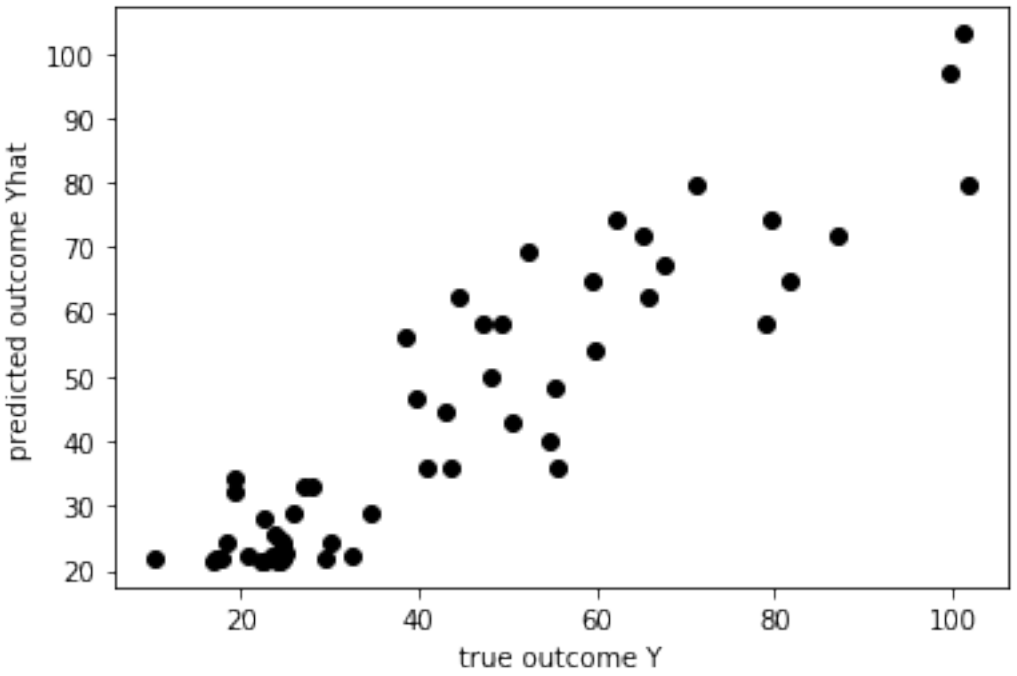
[56]:
```
196.71532106952645
```

[63]:
```
# Plot the data and our predictions vs. temperature
p = usage.plot.scatter(x='temperature', y='usage', label='observed')
p.scatter(x=usage['temperature'], y=Yhat, color='r', marker="+",␣
 ↪label='predicted')
plt.legend()
```

[63]:
```
<matplotlib.legend.Legend at 0x7fe78092cdd0>
```




[69]:
```
# and visualize the fit
plot_fit(Y, Yhat)
```



[ ]: