Regression

October 14, 2020

[1]:
```
import numpy
from matplotlib import pyplot as plt
%matplotlib inline
```

0.1 Linear regression

Data from Example 6.1 of Seborg, Edgar, Melichamp and Doyle (3rd edition)

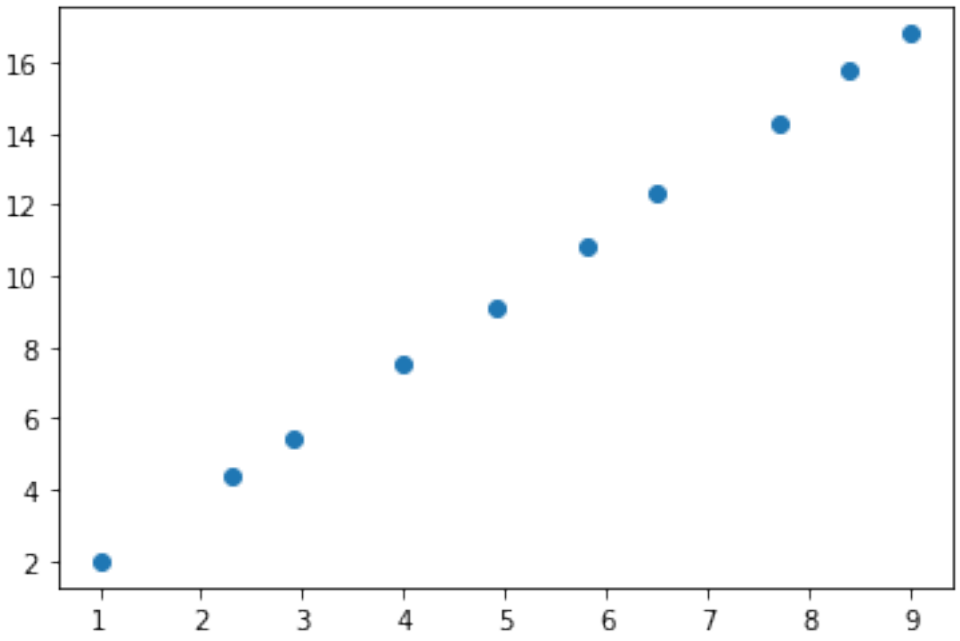
[2]:
```
import pandas
```

[3]:
```
df = pandas.read_excel('example_6_1.xlsx')
```

[4]:
```
x = df['Fuel Flow Rate']
y = df['Power Generated']
```

[5]:
```
plt.scatter(x, y)
```

[5]:
```
<matplotlib.collections.PathCollection at 0x1a4d35b66c8>
```



That resembles a straight line! Let's say the line is described by

$$y = ax + b$$

In regression, we are trying to find a and b given lots of values for y and x, so we have something like

$$y_0 = ax_0 + b \tag{1}$$
$$y_1 = ax_1 + b \tag{2}$$
$$y_2 = ax_2 + b \tag{3}$$
$$\vdots = \vdots \tag{4}$$
$$\tag{5}$$

Which we factor as

$$\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \vdots & \vdots \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\beta}$$

In this case, when there are many points, X is taller than it is wide and as such we know that there is no solution to the exact equations. Instead, we may try to get "close" to the solution. We can write the residuals as

$$\epsilon = Y - X\beta$$

Here, ϵ is a vector of errors, one for each data point. The (euclidian) length of this vector is given by

$$||\epsilon|| = \sqrt{\sum_i \epsilon_i^2}$$

It is common to focus on minimising the part without the square root to make calculations simpler. This leads to the popular way of determining the error of a fit called the "sum of square errors", sometimes expressed as $||\epsilon||^2$.

This minimisation is written in mathematical notation as

$$\min_\beta \sum_i \epsilon_i^2$$

which is read in words as as "minimise (*with respect to* or *by changing* β) the sum of the square error".

0.2 Create the design matrices

The matrices Y and X are sometimes called the design matrices. We can build them using basic numpy functions as follows:

[6]:
```
Y = numpy.asmatrix(y).T
X = numpy.bmat([numpy.c_[x], numpy.ones_like(Y)])
```

Note `numpy.c_` produces a two dimensional array from a one dimensional one in a *c*olumn.

In the case of polynomials, X is a special matrix called a Vandermonde matrix, and numpy has a function which generates them more easily.

[7]:
```
X = numpy.asmatrix(numpy.vander(x, 2))
```

There is also a library called patsy which supplies a simplified syntax to construct these matrices:

[8]:
```
import patsy
```

[9]:
```
Y, X = patsy.dmatrices("Q('Power Generated') ~ Q('Fuel Flow Rate')", df)
```

[10]:
```
Y, X = map(numpy.asmatrix, (Y, X))
```

0.2.1 Pseudoinverse solution

First, let's apply the pseudoinverse method directly (note **you should never do this for production code**, as calculating inverses is computationally expensive)

The solution minimising the sum of the squares of the residual $||\epsilon||^2$ can be shown to be

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

[11]:
```
#Excel: =MMULT((MINVERSE(MMULT(TRANSPOSE(_X); _X))); MMULT(TRANSPOSE(_X); _Y))
betahat = (X.T * X).I * X.T * Y
betahat
```

[11]:
```
matrix([[0.07854918],
        [1.85932397]])
```

Note: The code above is as close as possible to the equation above, as I have made `X` and `Y` matrices. **The numpy developers advise against using the `numpy.matrix` class**.

Matrix properties: * `.T`: Transpose * `.I`: Inverse * `.A`: Array form of matrix

Normal `numpy.arrays` don't have an `.I` property and don't multiply matrix-fasion but rather element-wise. Here is how we would have to write the code if we used arrays:

[12]:
```
Y = Y.A
X = X.A
```

[13]:
```
numpy.linalg.inv(X.T @ X) @ X.T @ Y
```

[13]:
```
array([[0.07854918],
       [1.85932397]])
```

The `@` operator always does matrix multiplication and expects two-dimensional arrays on both sides.

0.2.2 Dedicated solvers

Calculating the inverse is not a numerically well behaved operation, so you should rather use a dedicated routine if you are solving this kind of problem:

[14]:
```
beta, residuals, rank, s = numpy.linalg.lstsq(X, Y, rcond=None)
beta
```

[14]:
```
array([[0.07854918],
       [1.85932397]])
```

However, polynomial fits are such a common operation that there are nicer routines to do this fit. There are a whole range of functions in numpy which start with poly. Press tab to see them.

[15]:
```
numpy.poly
```

[15]:
```
<function numpy.poly(seq_of_zeros)>
```

[16]:
```
poly = numpy.polyfit(x, y, 1)
poly
```

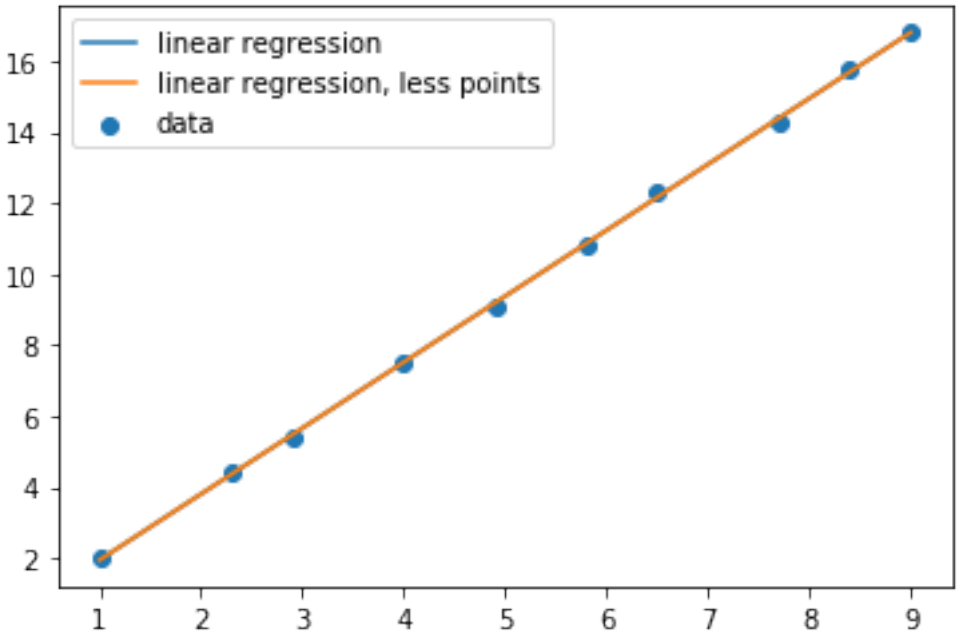
[16]:
```
array([1.85932397, 0.07854918])
```

Notice that we could just pass the data directly, and the routine handled building the various matrices and the fitting itself.

It is useful to plot the regression with the data points, but we should sample on a finer grid.

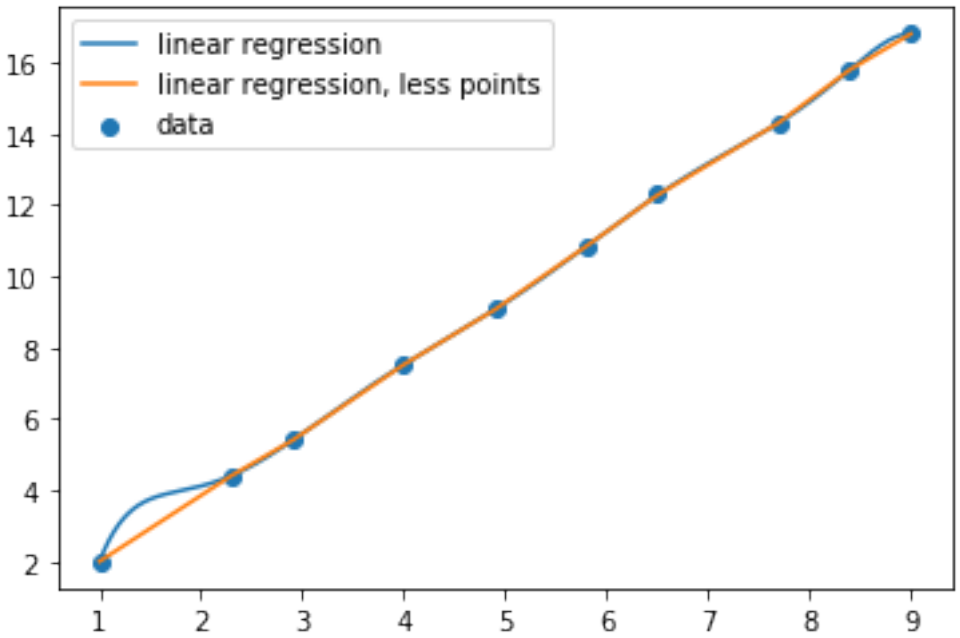
[17]:
```
smoothx = numpy.linspace(min(x), max(x), 1000)
```

[18]:
```
def regplot(poly):
    smoothy = numpy.polyval(poly, smoothx)
    plt.scatter(x, y, label='data')
    plt.plot(smoothx, smoothy, label='linear regression')
    plt.plot(x, numpy.polyval(poly, x), label='linear regression, less points')
    plt.legend(loc='best')
regplot(poly)
```



There's obviously no difference between the two for a linear fit, but what about higher orders?

[19]:
```
poly9 = numpy.polyfit(x, y, 8)
regplot(poly9)
```



If we had just plotted the connecting lines, we would have missed the bit sticking up on the left!

0.3 Nonlinear regression

We can apply the same principles to fit nonlinear functions as well. The `scipy.optimize.curve_fit` function can be used to fit an aribitrary function to data

[20]:
```
import scipy.optimize
```

Let's start by reproducing the results from the linear fit

[21]:
```
def f(x, a, b):
    """fitting function linear in coefficient"""
    return a*x + b
```

[22]:
```
beta, _ = scipy.optimize.curve_fit(f, x, y, [1, 0])
beta
```

[22]:
```
array([1.85932396, 0.07854919])
```

Now let's build some data which is obviously nonlinear

[23]:
```
x = numpy.arange(1, 10)
y = 2*numpy.sin(3*x) + x + 1
```

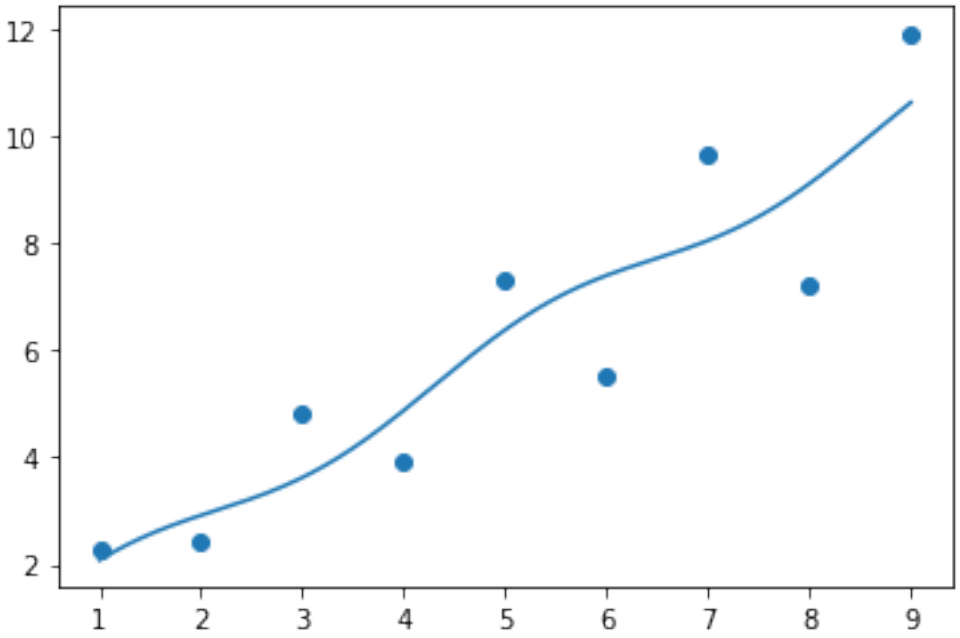
[24]:
```
def f2(x, a, b, c, d):
    """ A nonlinear fitting function"""
    return a*numpy.sin(b*x)+ c*x + d
```

[25]:
```
def fit_and_plot(beta0):
    beta, _ = scipy.optimize.curve_fit(f2, x, y, beta0)

    plt.scatter(x, y)
    plt.plot(smoothx, f2(smoothx, *beta))
    return beta
```

[26]:
```
fit_and_plot([1, 1, 1, 1])
```

[26]:
```
array([0.32658282, 1.45786799, 1.08549627, 0.67636834])
```



What went wrong?

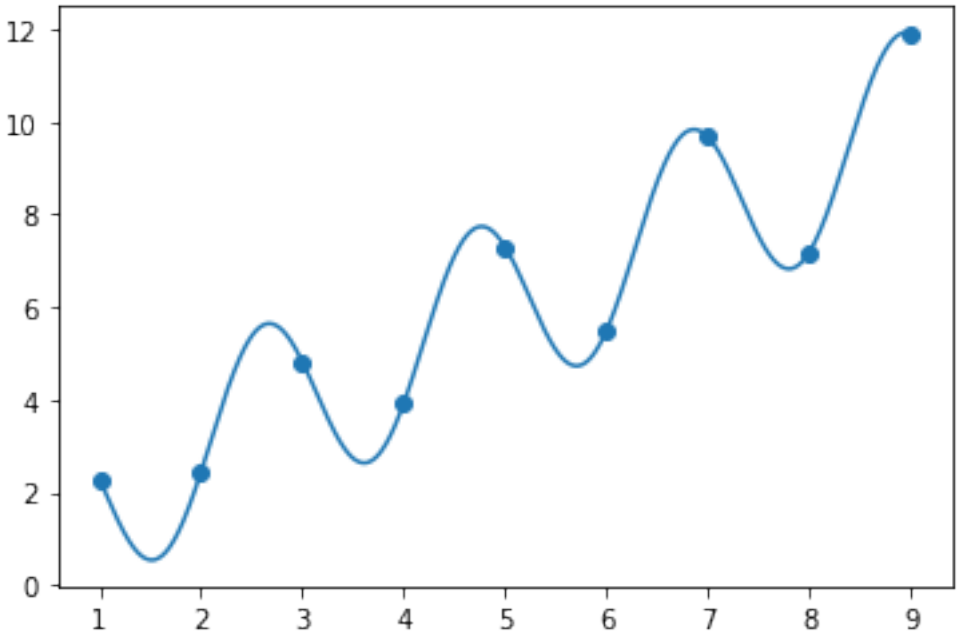
The initial values we chose were not sufficiently close to the "correct" values. This shows the first main problem with nonlinear regression: there may be multiple solutions which are returned based on the initial guess.

Linear regression	Nonlinear regression
single correct solution	multiple solutions possible depending on initial guess
requires no initial guess	requires initial guess
Never returns the "wrong" local minimum solution	Sometimes claims success with "wrong" answer
less flexible in functional form	more flexible functional form

Let's try a different starting point. Remember the initial data were generated with $\beta = [2, 3, 1, 1]$

[27]:
```
beta2 = fit_and_plot([2, 2.8, 1, 1])
beta2
```

[27]:
```
array([2., 3., 1., 1.])
```



OK, now we know we're right, right? The error of this fit is essentially zero.

[28]:
```
def fiterror(beta):
    return sum((y - f2(x, *beta))**2)
```

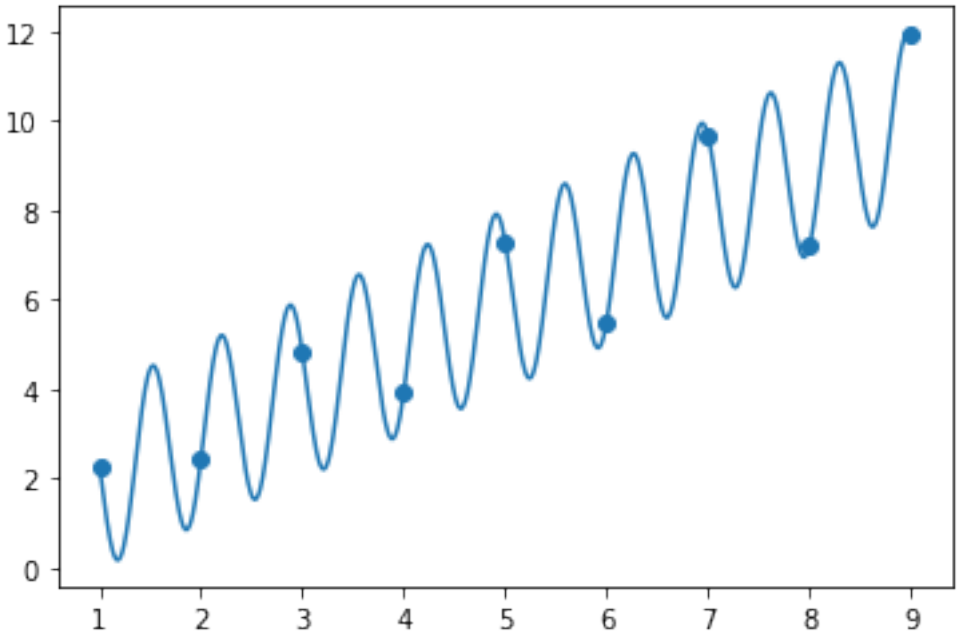
[29]:
```
fiterror(beta2)
```

[29]:
```
4.106021011675366e-28
```

But wait, what about this:

[30]:
```
beta3 = fit_and_plot([2, 9.2, 1, 1])
beta3
```

[30]:
```
array([2.        , 9.28318531, 1.        , 1.        ])
```



[31]:
```
fiterror(beta3)
```

[31]:
```
2.0194839173657902e-28
```

It is clear that there are multiple solutions to this problem which are equally good! This is a property of nonlinear regression. It is often impossible to recover the "right" values of the parameters. You should therefore be careful of interpreting a good fit as evidence of correctness of your model.

[34]:
```
!pip install jupyter_contrib_nbextensions
!jupyter nbconvert notebook_name_here.ipynb --to html
```

```
[NbConvertApp] Converting notebook Regression.ipynb to html
[NbConvertApp] Writing 403468 bytes to Regression.html
```

[35]:
```
!pip install jupyter_contrib_nbextensions
```

```
Requirement already satisfied: jupyter_contrib_nbextensions in
c:\users\jeff\anaconda3\lib\site-packages (0.5.1)
Requirement already satisfied: tornado in c:\users\jeff\anaconda3\lib\site-
packages (from jupyter_contrib_nbextensions) (6.0.3)
Requirement already satisfied: jupyter-latex-envs>=1.3.8 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(1.4.6)
Requirement already satisfied: notebook>=4.0 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(6.0.3)
Requirement already satisfied: jupyter-nbextensions-configurator>=0.4.0 in
```

```
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(0.4.1)
Requirement already satisfied: nbconvert>=4.2 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(5.6.1)
Requirement already satisfied: pyyaml in c:\users\jeff\anaconda3\lib\site-
packages (from jupyter_contrib_nbextensions) (5.3)
Requirement already satisfied: lxml in c:\users\jeff\anaconda3\lib\site-packages
(from jupyter_contrib_nbextensions) (4.5.0)
Requirement already satisfied: jupyter-contrib-core>=0.3.3 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(0.3.3)
Requirement already satisfied: ipython-genutils in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: jupyter-highlight-selected-word>=0.1.1 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(0.2.0)
Requirement already satisfied: jupyter-core in c:\users\jeff\anaconda3\lib\site-
packages (from jupyter_contrib_nbextensions) (4.6.1)
Requirement already satisfied: traitlets>=4.1 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter_contrib_nbextensions)
(4.3.3)
Requirement already satisfied: ipython in c:\users\jeff\anaconda3\lib\site-
packages (from jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions) (7.12.0)
Requirement already satisfied: prometheus-client in
c:\users\jeff\anaconda3\lib\site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (0.7.1)
Requirement already satisfied: Send2Trash in c:\users\jeff\anaconda3\lib\site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (1.5.0)
Requirement already satisfied: ipykernel in c:\users\jeff\anaconda3\lib\site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (5.1.4)
Requirement already satisfied: terminado>=0.8.1 in
c:\users\jeff\anaconda3\lib\site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (0.8.3)
Requirement already satisfied: pyzmq>=17 in c:\users\jeff\anaconda3\lib\site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (18.1.1)
Requirement already satisfied: nbformat in c:\users\jeff\anaconda3\lib\site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (5.0.4)
Requirement already satisfied: jupyter-client>=5.3.4 in
c:\users\jeff\anaconda3\lib\site-packages (from
notebook>=4.0->jupyter_contrib_nbextensions) (5.3.4)
Requirement already satisfied: jinja2 in c:\users\jeff\anaconda3\lib\site-
packages (from notebook>=4.0->jupyter_contrib_nbextensions) (2.11.1)
Requirement already satisfied: bleach in c:\users\jeff\anaconda3\lib\site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (3.1.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
c:\users\jeff\anaconda3\lib\site-packages (from
```

```
nbconvert>=4.2->jupyter_contrib_nbextensions) (1.4.2)
Requirement already satisfied: pygments in c:\users\jeff\anaconda3\lib\site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (2.5.2)
Requirement already satisfied: testpath in c:\users\jeff\anaconda3\lib\site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (0.4.4)
Requirement already satisfied: mistune<2,>=0.8.1 in
c:\users\jeff\anaconda3\lib\site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (0.8.4)
Requirement already satisfied: defusedxml in c:\users\jeff\anaconda3\lib\site-
packages (from nbconvert>=4.2->jupyter_contrib_nbextensions) (0.6.0)
Requirement already satisfied: entrypoints>=0.2.2 in
c:\users\jeff\anaconda3\lib\site-packages (from
nbconvert>=4.2->jupyter_contrib_nbextensions) (0.3)
Requirement already satisfied: setuptools in c:\users\jeff\anaconda3\lib\site-
packages (from jupyter-contrib-core>=0.3.3->jupyter_contrib_nbextensions)
(45.2.0.post20200210)
Requirement already satisfied: pywin32>=1.0; sys_platform == "win32" in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter-
core->jupyter_contrib_nbextensions) (227)
Requirement already satisfied: decorator in c:\users\jeff\anaconda3\lib\site-
packages (from traitlets>=4.1->jupyter_contrib_nbextensions) (4.4.1)
Requirement already satisfied: six in c:\users\jeff\anaconda3\lib\site-packages
(from traitlets>=4.1->jupyter_contrib_nbextensions) (1.14.0)
Requirement already satisfied: backcall in c:\users\jeff\anaconda3\lib\site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.1.0)
Requirement already satisfied: pickleshare in c:\users\jeff\anaconda3\lib\site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
c:\users\jeff\anaconda3\lib\site-packages (from ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (3.0.3)
Requirement already satisfied: jedi>=0.10 in c:\users\jeff\anaconda3\lib\site-
packages (from ipython->jupyter-latex-envs>=1.3.8->jupyter_contrib_nbextensions)
(0.14.1)
Requirement already satisfied: colorama; sys_platform == "win32" in
c:\users\jeff\anaconda3\lib\site-packages (from ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.4.3)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
c:\users\jeff\anaconda3\lib\site-packages (from
nbformat->notebook>=4.0->jupyter_contrib_nbextensions) (3.2.0)
Requirement already satisfied: python-dateutil>=2.1 in
c:\users\jeff\anaconda3\lib\site-packages (from jupyter-
client>=5.3.4->notebook>=4.0->jupyter_contrib_nbextensions) (2.8.1)
Requirement already satisfied: MarkupSafe>=0.23 in
c:\users\jeff\anaconda3\lib\site-packages (from
jinja2->notebook>=4.0->jupyter_contrib_nbextensions) (1.1.1)
Requirement already satisfied: webencodings in c:\users\jeff\anaconda3\lib\site-
```

```
packages (from bleach->nbconvert>=4.2->jupyter_contrib_nbextensions) (0.5.1)
Requirement already satisfied: wcwidth in c:\users\jeff\anaconda3\lib\site-
packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython->jupyter-
latex-envs>=1.3.8->jupyter_contrib_nbextensions) (0.1.8)
Requirement already satisfied: parso>=0.5.0 in c:\users\jeff\anaconda3\lib\site-
packages (from jedi>=0.10->ipython->jupyter-latex-
envs>=1.3.8->jupyter_contrib_nbextensions) (0.5.2)
Requirement already satisfied: attrs>=17.4.0 in
c:\users\jeff\anaconda3\lib\site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat->notebook>=4.0->jupyter_contrib_nbextensions)
(19.3.0)
Requirement already satisfied: pyrsistent>=0.14.0 in
c:\users\jeff\anaconda3\lib\site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat->notebook>=4.0->jupyter_contrib_nbextensions)
(0.15.7)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
c:\users\jeff\anaconda3\lib\site-packages (from
jsonschema!=2.5.0,>=2.4->nbformat->notebook>=4.0->jupyter_contrib_nbextensions)
(1.5.0)
Requirement already satisfied: zipp>=0.5 in c:\users\jeff\anaconda3\lib\site-
packages (from importlib-metadata; python_version < "3.8"->jsonschema!=2.5.0,>=2
.4->nbformat->notebook>=4.0->jupyter_contrib_nbextensions) (2.2.0)
```