CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

MODULE 2: REGRESSION AND CLASSIFICATION

DAT405, 2019-2020, READING PERIOD 1

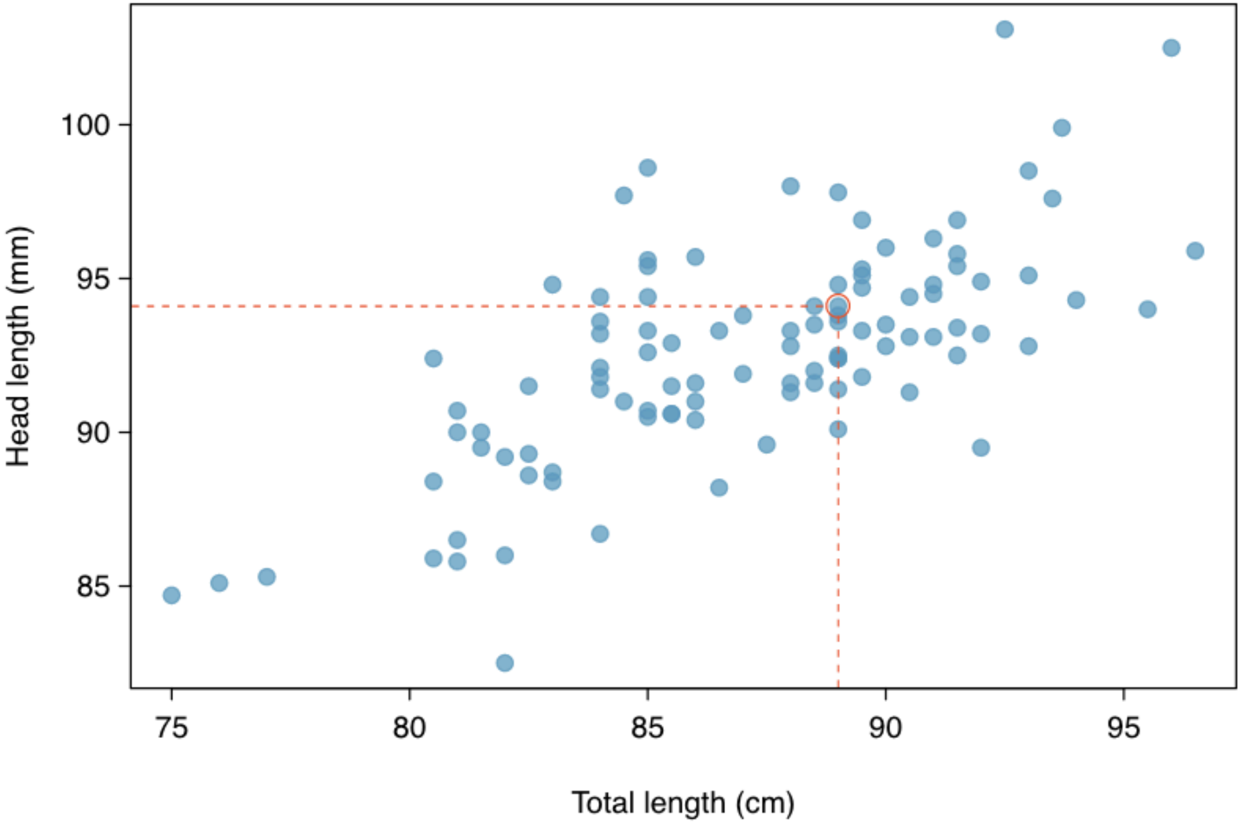
Core data science tasks

- Regression
  - Predicting a numerical quantity
- Classification
  - Assigning a label from a discrete set of possibilities
- Clustering
  - Grouping items by similarity

REGRESSION

- Predicting a numerical quantity

Brushtail possums (n=104)



Goal:
Express one variable as a function of other(s)

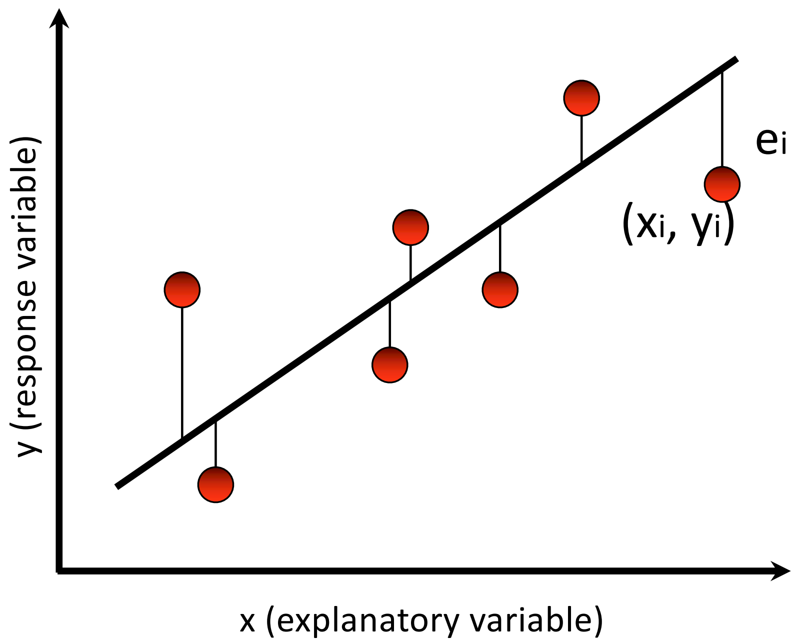
https://courses.lumenlearning.com/odessa-introstats1-1/chapter/line-fitting-residuals-and-correlation/

Linear regression

- “Linear regression is a bread-and-butter modeling technique that should serve as your baseline approach to building data-driven models.”
- “These models are typically easy to build, straightforward to interpret, and often do quite well in practice.”
- “With enough skill and toil, more advanced machine learning techniques might yield better performance, but the possible payoff is often not worth the effort.”
- “Build your linear regression models first, then decide whether it is worth working harder to achieve better results.”

S.S. Skiena, “The Data Science Design Manual”, 2017

Least squares linear regression



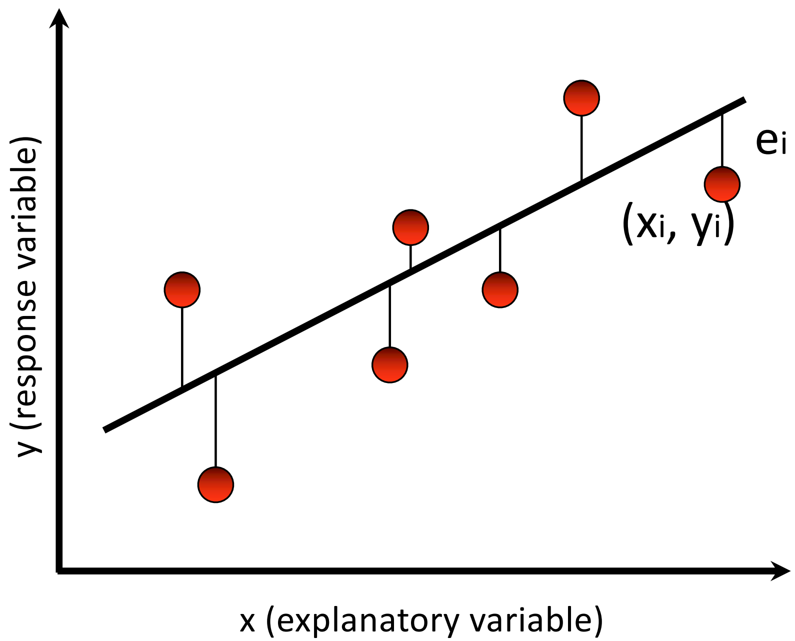
Data (x_i, y_i) $i=1,\ldots,n$

Model (Fit): $y = b_1 x + b_0$

Residuals: $e_i = y_i - (b_1 x_i + b_0)$

Data = Fit + Residual

Least squares linear regression



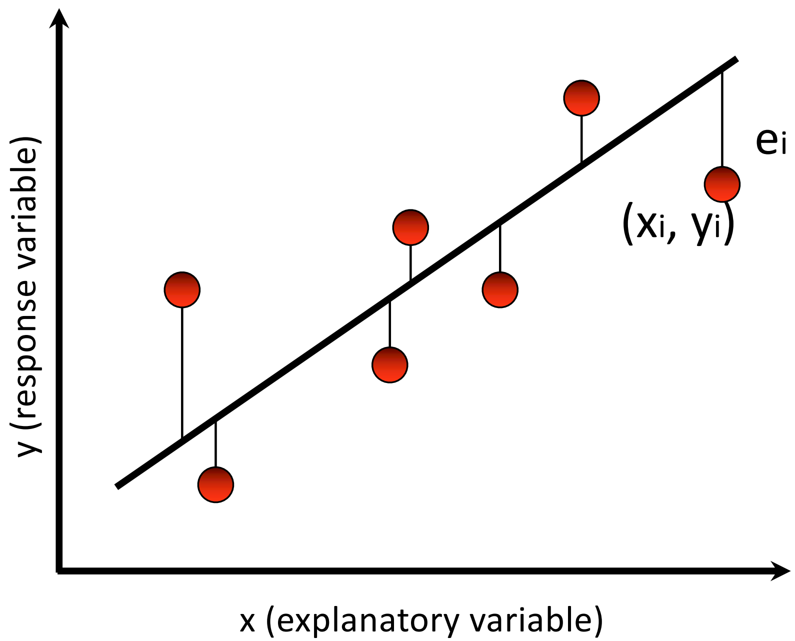
Data (x_i, y_i) i=1,...,n

Model (Fit): $y = b_1 x + b_0$

Residuals: $e_i = y_i - (b_1 x_i + b_0)$

Data = Fit + Residual

Least squares linear regression



Data (x_i, y_i) $i=1,\ldots,n$

Model (Fit): $y = b_1 x + b_0$

Residuals: $e_i = y_i - (b_1 x_i + b_0)$

Data = Fit + Residual

Least squares criterion:

$$\min_{b_0, b_1} \sum_{i=1}^{n} e_i^2 =$$

$$\min_{b_0, b_1} \sum_{i=1}^{n} (y_i - (b_1 x_i + b_0))^2$$

Linear regression

- Regression line is useful for visualisation
- A method for forecasting
- Residual error of a regression line is the difference between the predicted and actual values

- Seeks to find the line $y = f(x)$ which minimises the sum of the squared errors over all training points
- An optimisation problem

Goal: a linear model

Predict head length y (the independent variable) from total length x

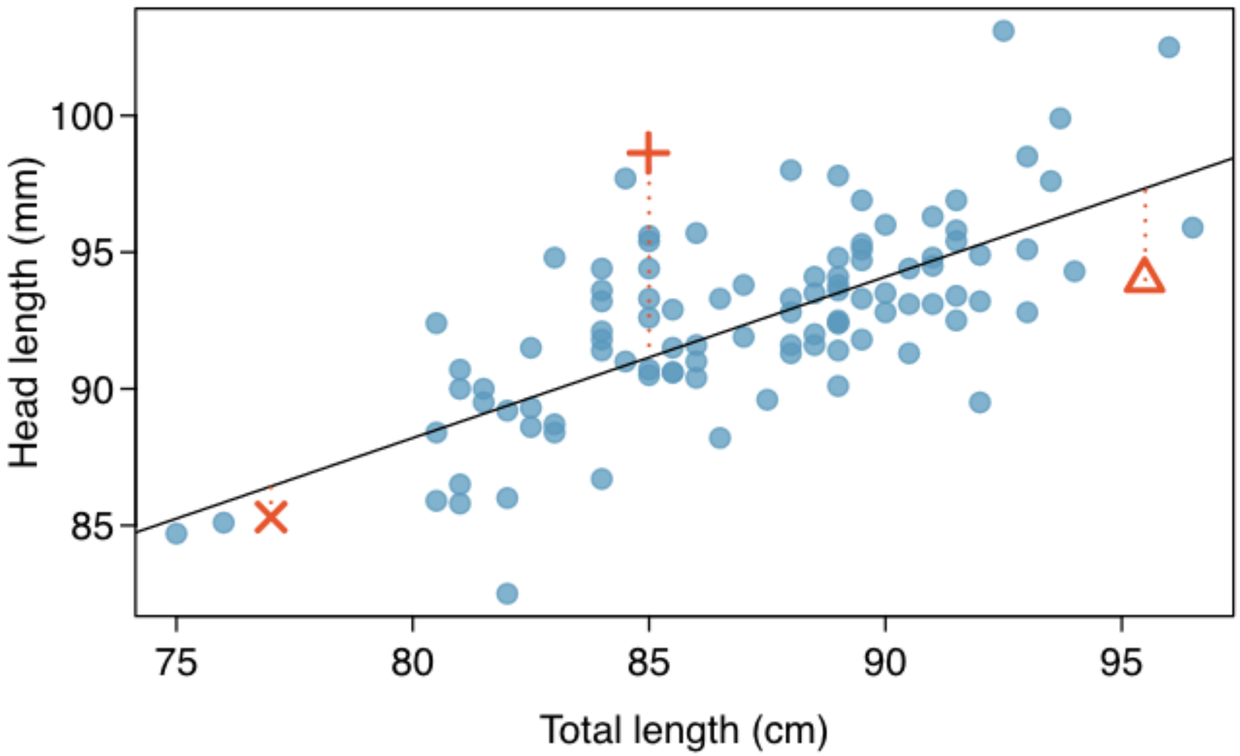
$$y = f(x) + r$$

where f( ) is some function and r the residual.

Simplest case:

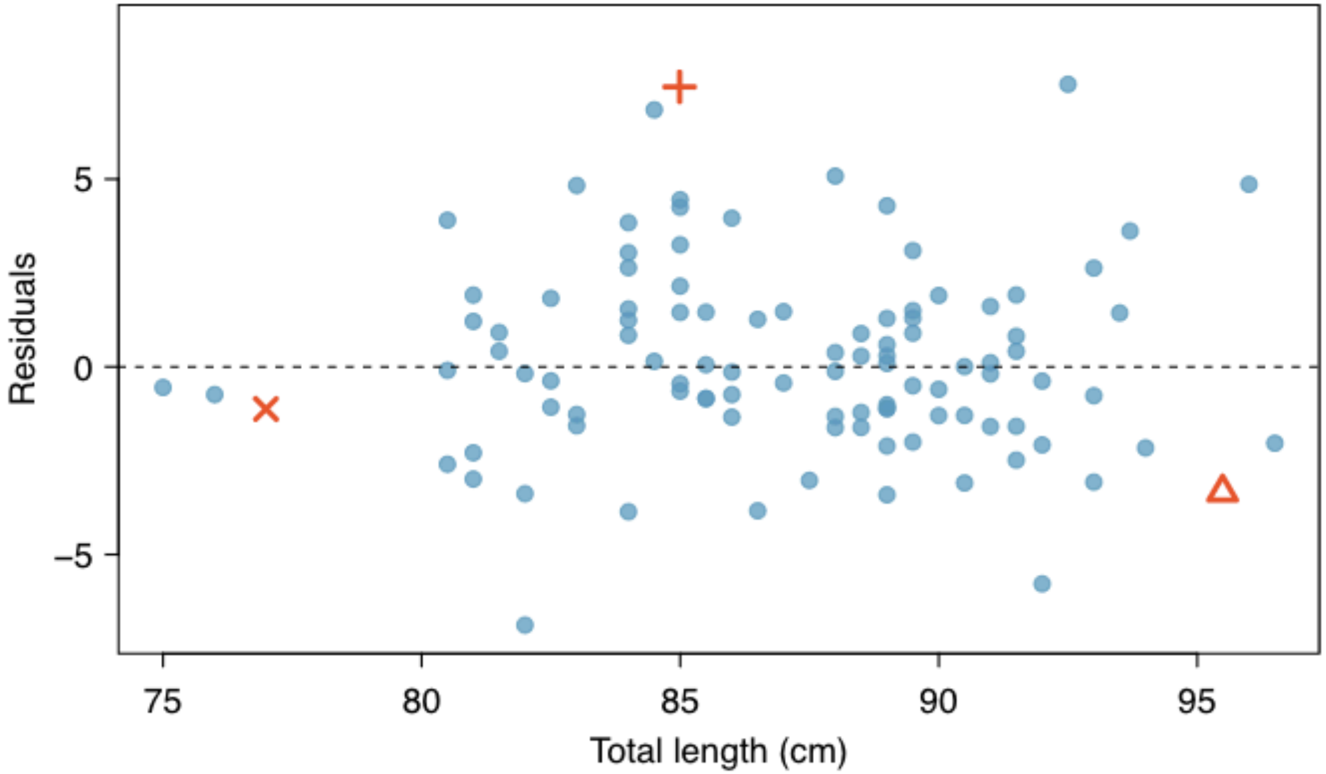
f( ) is linear, that is $f(x) = a\,x + b$

A linear model



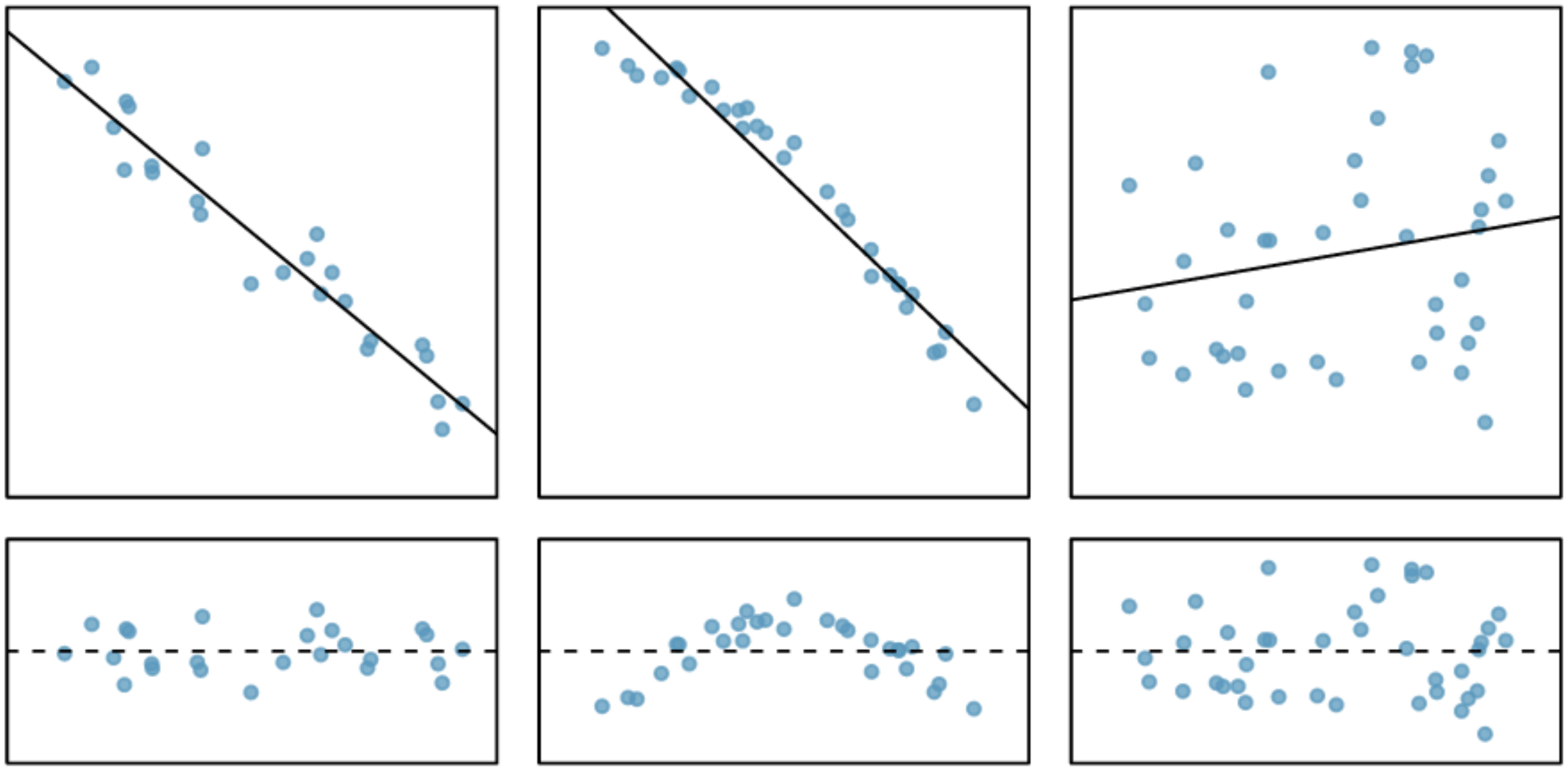
https://courses.lumenlearning.com/odessa-introstats1-1/chapter/line-fitting-residuals-and-correlation/

Residual plot



https://courses.lumenlearning.com/odessa-introstats1-1/chapter/line-fitting-residuals-and-correlation/

Sample data and residual plots



https://courses.lumenlearning.com/odessa-introstats1-1/chapter/line-fitting-residuals-and-correlation/

Correlation

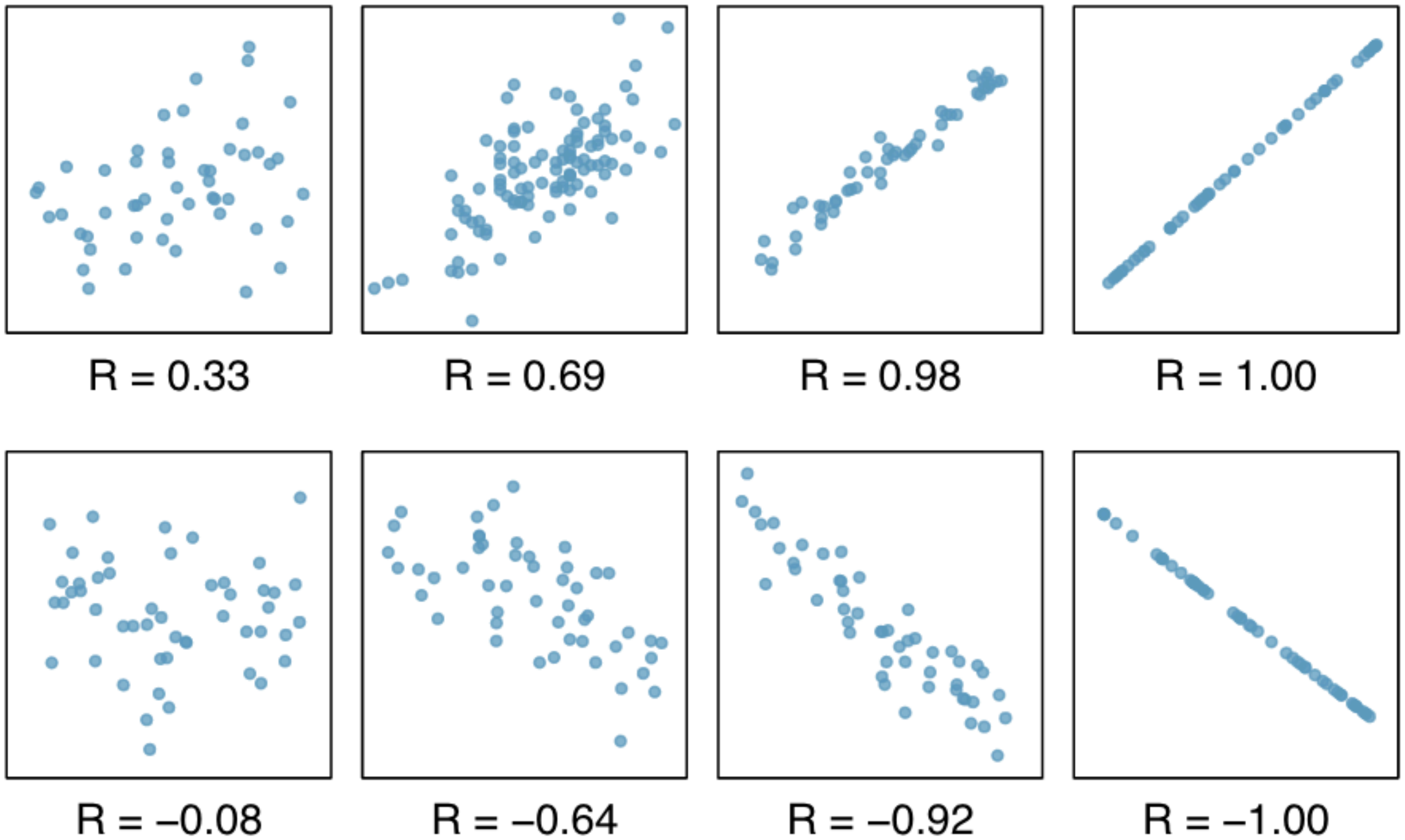
mean x and y

$$R = \frac{1}{n-1}\sum_{i=1}^{n}\frac{x_i - \bar{x}}{s_x}\frac{y_i - \bar{y}}{s_y}$$

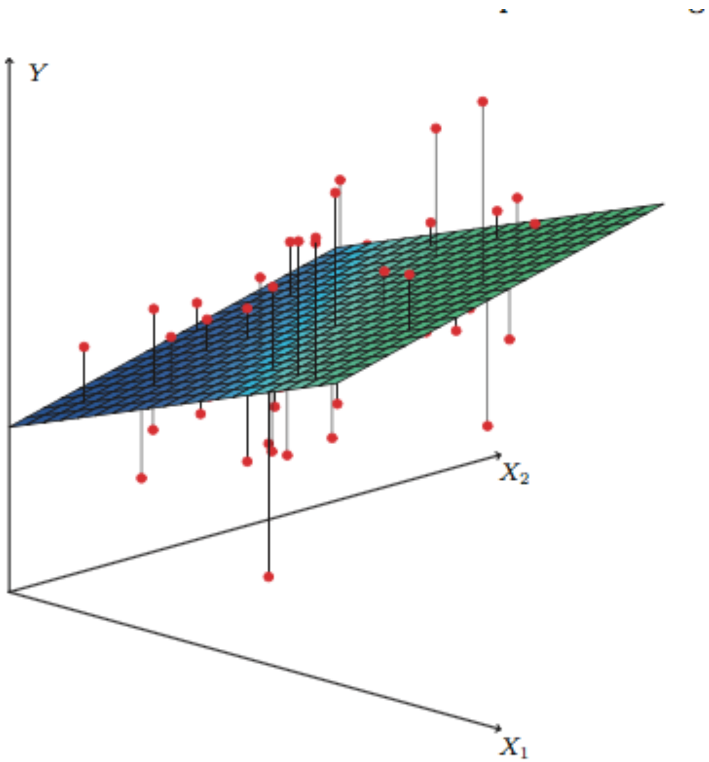
variance of x and y

Quantifies the strength of a linear trend

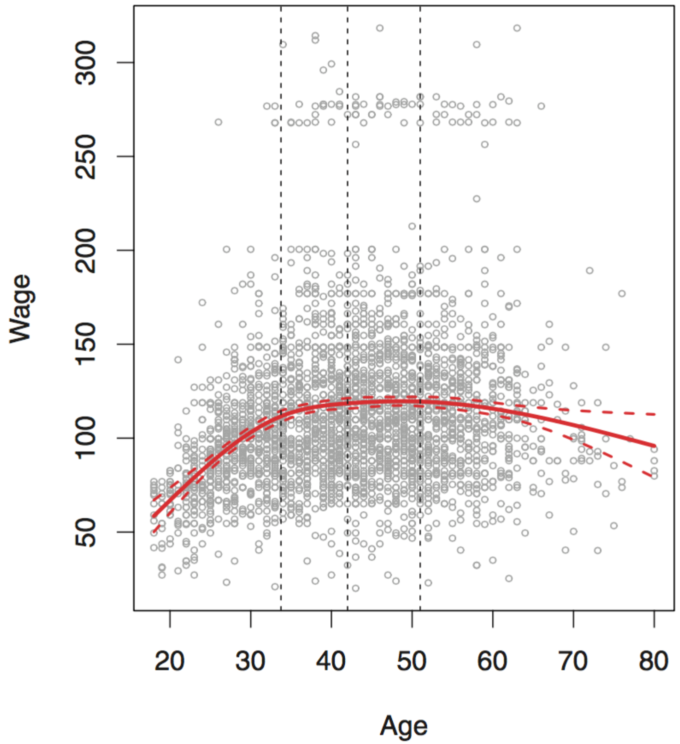
Scatter plots and correlations



Regression problems



Multi-dimensional data



Non-linear

From Intro to Statistical Learning.

Linear regression in python

sklearn.linear_model.LinearRegression()

scikit-learn documentation:

- https://scikit-learn.org/stable/modules/linear_model.html
- https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Python Data Science Handbook

- https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html

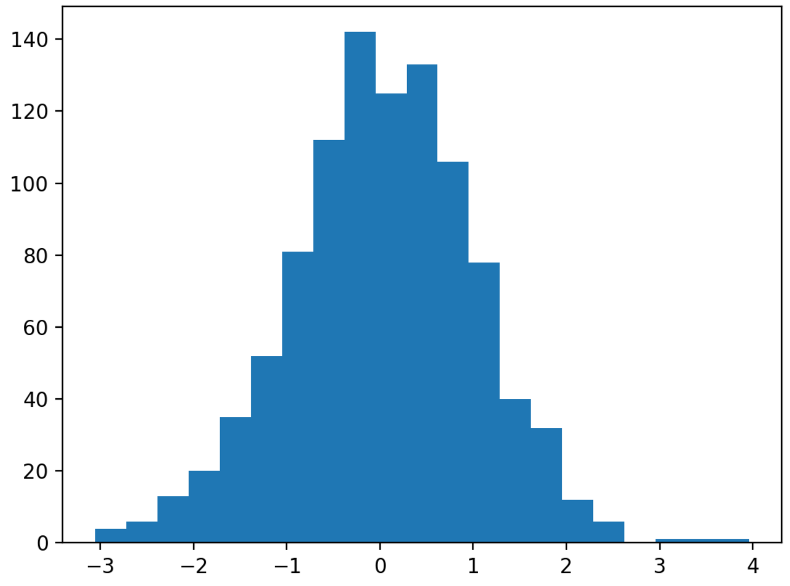
Generating an array of random numbers

- Random values in the range [0,1)
- Construct and initialise a pseudo-random number generator

```
>>> import numpy as np
>>> rng = np.random.RandomState(1)
>>> rng.rand(10)
array([4.17022005e-01, 7.20324493e-01, 1.14374817e-04, 3.02332573e-01,
       1.46755891e-01, 9.23385948e-02, 1.86260211e-01, 3.45560727e-01,
       3.96767474e-01, 5.38816734e-01])
>>>
```

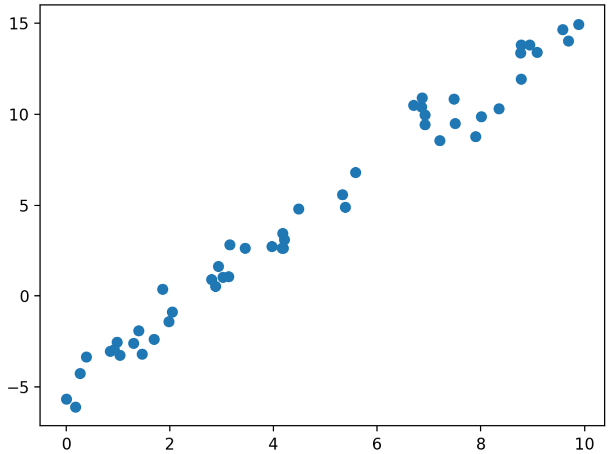
Random numbers from the "standard normal" distribution

```
import matplotlib.pyplot as plt
import numpy as np

rng = np.random.RandomState(1)

plt.hist(rng.randn(1000), bins=21)
plt.show()
```



Scatter data about a line with slope 2 and intercept -5

```
import matplotlib.pyplot as plt
import numpy as np

rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)

print("x is ", x)
print("y is ", y)

plt.scatter(x, y)
plt.show()
```




```
$ python plot50.py
x is  [4.17022005e+00 7.20324493e+00 1.14374817e-03 3.02332573e+00
 1.46755891e+00 9.23385948e-01 1.86260211e+00 3.45560727e+00
 3.96767474e+00 5.38816734e+00 4.19194514e+00 6.85219500e+00
 2.04452250e+00 8.78117436e+00 2.73875932e-01 6.70467510e+00
 4.17304802e+00 5.58689828e+00 1.40386939e+00 1.98101489e+00
 8.00744569e+00 9.68261576e+00 3.13424178e+00 6.92322616e+00
 8.76389152e+00 8.94606664e+00 8.50442114e-01 3.90547832e-01
 1.69830420e+00 8.78142503e+00 9.83468338e-01 4.21107625e+00
 9.57889530e+00 5.33165285e+00 6.91877114e+00 3.15515631e+00
 6.86500928e+00 8.34625672e+00 1.82882773e-01 7.50144315e+00
 9.88861089e+00 7.48165654e+00 2.80443992e+00 7.89279328e+00
 1.03226007e+00 4.47893526e+00 9.08595503e+00 2.93614148e+00
 2.87775339e+00 1.30028572e+00]
y is  [ 2.65326739  8.56128423 -5.66895863  1.03398685 -3.18219253 -
2.91881241
  0.3850064   2.6532587   2.74351393  4.88870572  2.63673199 10.39684461
 -0.86014725 11.92535308 -4.26133265 10.50960534  3.466255    6.79099968
 -1.89209091 -1.39022006  9.87237318 14.01588879  1.05958933  9.4330755
 13.36676646 13.82323535 -3.01352845 -3.33376317 -2.35778955 13.81571822
 -2.5201335   3.12405966 14.64630875  5.58773399  9.96917167  2.83012944
 10.91559396 10.2960171  -6.07834826  9.49842044 14.93725885 10.83948201
  0.92451479  8.76338535 -3.24168388  4.78584517 13.4020048   1.63429415
  0.53317863 -2.60018663]
```

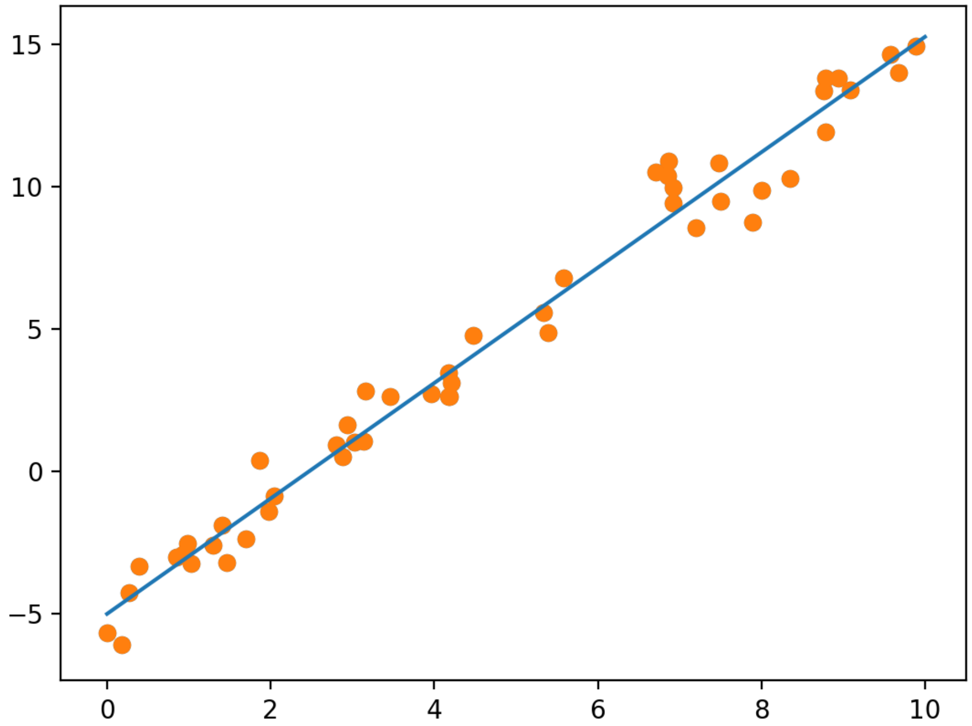
numpy.linspace() and numpy.newaxis

```
>>> import numpy
>>> values = numpy.linspace(0, 1, 5)
>>> values
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
>>> numpy.shape(values)
(5,)
>>> values[:, numpy.newaxis]
array([[0.  ],
       [0.25],
       [0.5 ],
       [0.75],
       [1.  ]])
>>> numpy.shape(values[:, numpy.newaxis])
(5, 1)
>>>
```

- Return evenly spaced numbers over a specified interval.

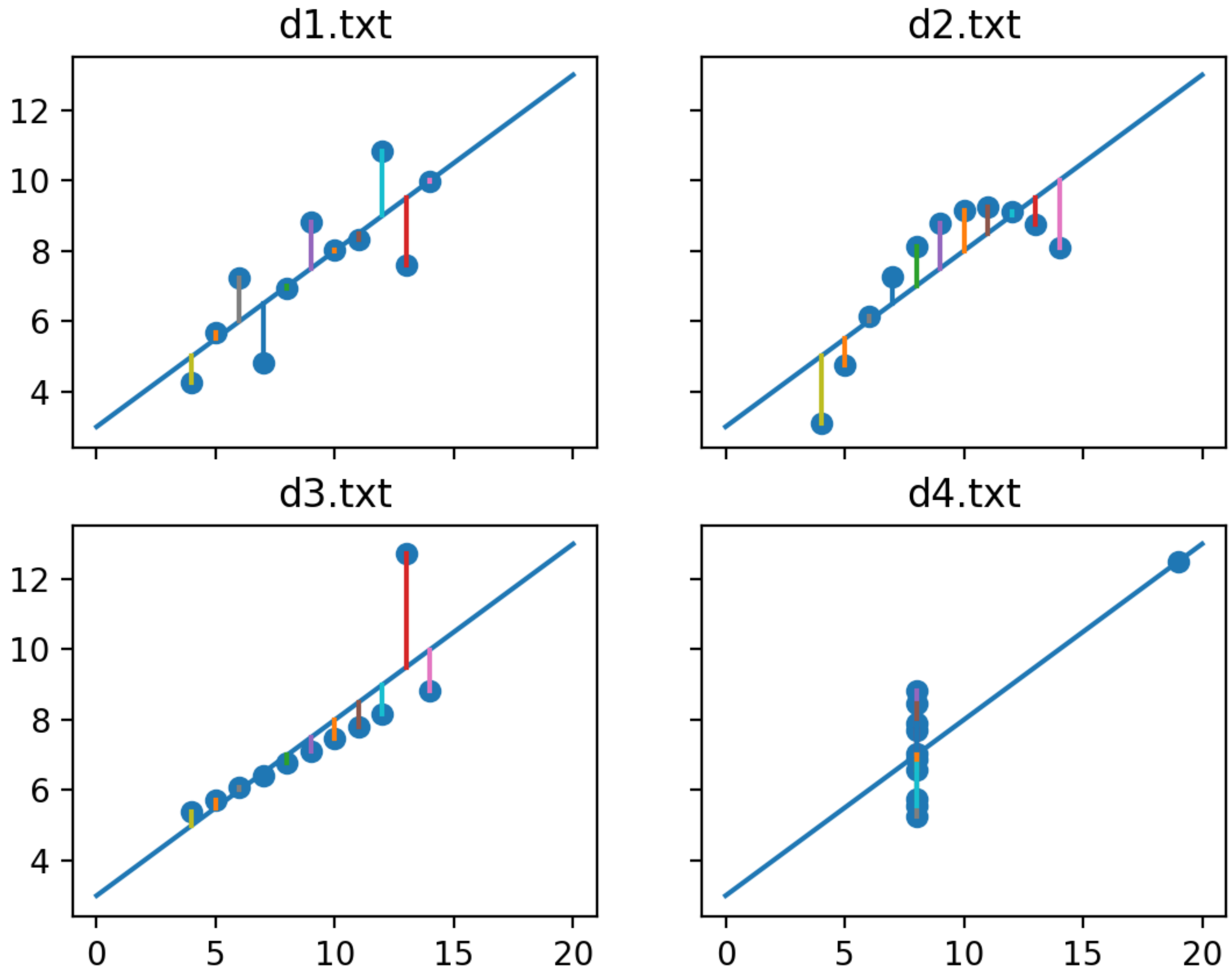
Least squares linear regression

```
import matplotlib.pyplot as plt
import numpy as np

rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)
plt.scatter(x, y)

from sklearn.linear_model import LinearRegression
model = LinearRegression()

model.fit(x[:, np.newaxis], y)

xfit = np.linspace(0, 10, 1000)
yfit = model.predict(xfit[:, np.newaxis])

plt.scatter(x, y)
plt.plot(xfit, yfit);

plt.show()
```



Slope and intercept of the regression line

```
>>> import numpy as np
>>>
>>> rng = np.random.RandomState(1)
>>> x = 10 * rng.rand(50)
>>> y = 2 * x - 5 + rng.randn(50)
>>> from sklearn.linear_model import LinearRegression
>>> model = LinearRegression()
>>>
>>> model.fit(x[:, np.newaxis], y)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
>>> print(model.intercept_)
-4.998577085553202
>>> print(model.coef_)
[2.02720881]
>>>
```


d1.txt
d2.txt
d3.txt
d4.txt
12
10
8
6
4
0
5
10
15
20

```python
import sys
import pandas
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

model = LinearRegression()
fig, axs = plt.subplots(2, 2, sharex = 'all', sharey = 'all')


for i in range(4):
    df = pandas.read_csv(sys.argv[i+1], sep=' ')
    xValues = df['x']
    yValues = df['y']
    model.fit(xValues[:, np.newaxis], yValues)
    xfit = np.linspace(0, 20, 1000)
    yfit = model.predict(xfit[:, np.newaxis])
    axs[ i // 2, i % 2 ].scatter(xValues, yValues)
    axs[ i // 2, i % 2 ].plot(xfit, yfit)
    axs[ i // 2, i % 2 ].set_title(sys.argv[i+1])

    yPredicted = model.predict(xValues[:, np.newaxis])
    for j in range(len(xValues)):
        lineXdata = (xValues[j], xValues[j])
        lineYdata = (yValues[j], yPredicted[j])
        axs[ i // 2, i % 2 ].plot(lineXdata, lineYdata)


# Hide x labels and tick labels for top plots and y ticks for right plots.
for ax in axs.flat:
    ax.label_outer()


plt.show()
```

Fitting non-linear functions

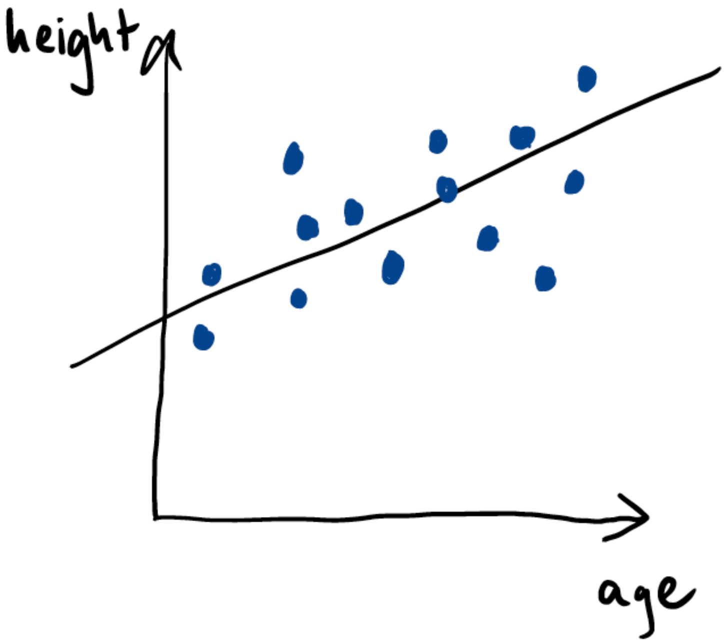
- Could fit quadratics, arbitrary higher order polynomials, exponential and logarithmic curves, etc. instead of straight lines

- Alternatively, we could explicitly include other component variables in our data matrix, e.g. sqrt(x), log(x), x^3, 1/x, sin(x), etc.
- Can then capture a non-linear relationship with a linear model!

- Inconvenient and impractical to explicitly enumerate all possibilities
- Consider using more powerful learning methods, e.g. support vector machines.

CLASSIFICATION

- Assigning a label from a discrete set of possibilities regression



classification

Iris data set

R. A. Fisher (1936). "The use of multiple measurements in taxonomic problems". Annals of Eugenics. ***7*** *(2): 179–188.*

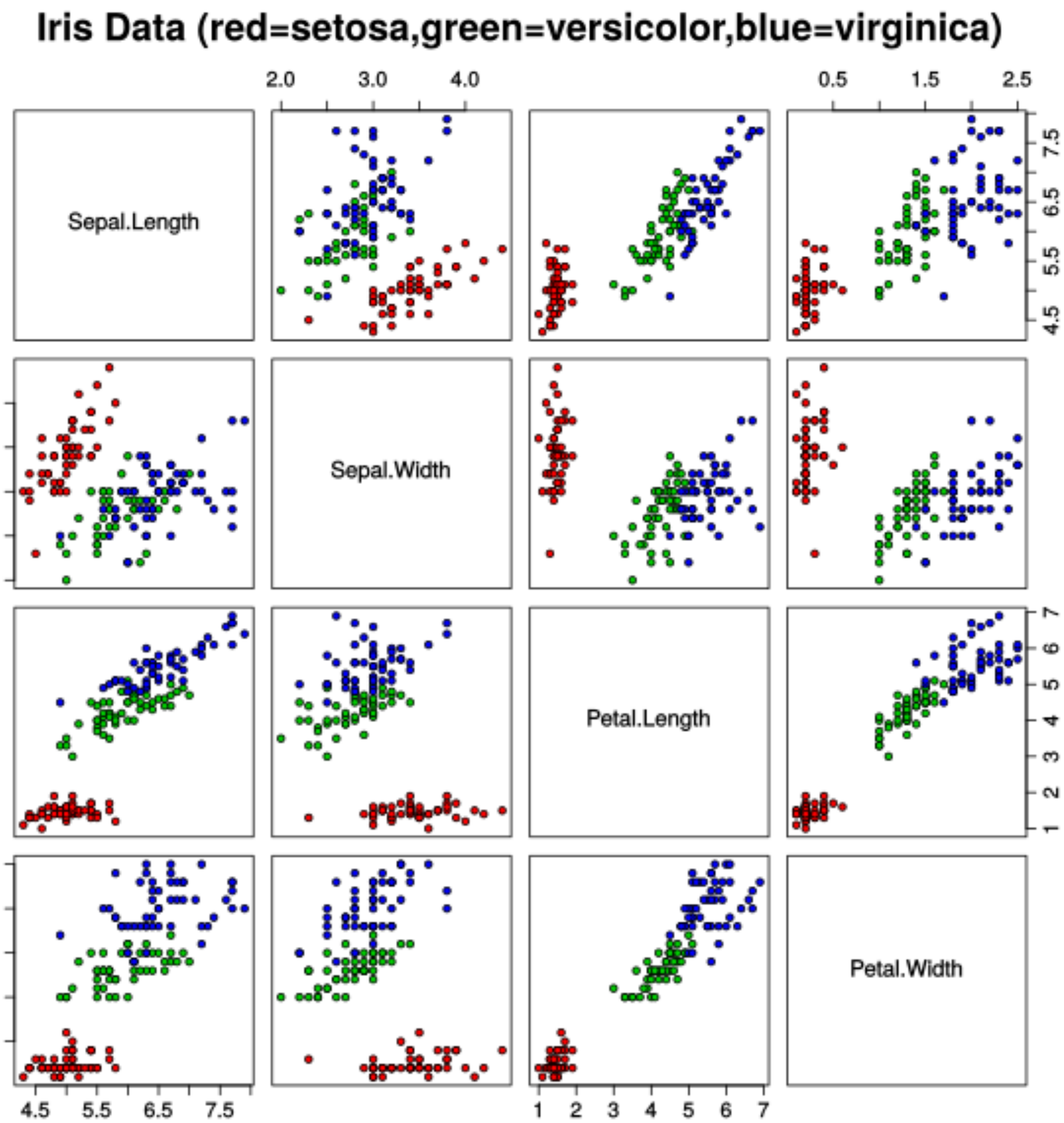
- Petal length
- Petal width
- Sepal length
- Sepal width

50 samples from each of three species

Iris setosa

Iris versicolor

Iris virginica



https://commons.wikimedia.org/wiki/File:Iris_dataset_scatterplot.svg (User:Nicoguaro)