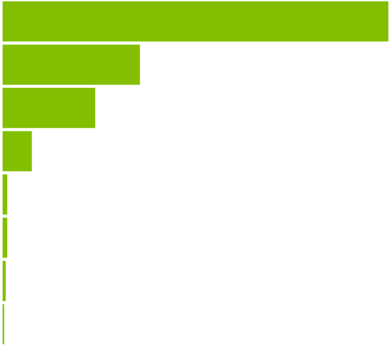
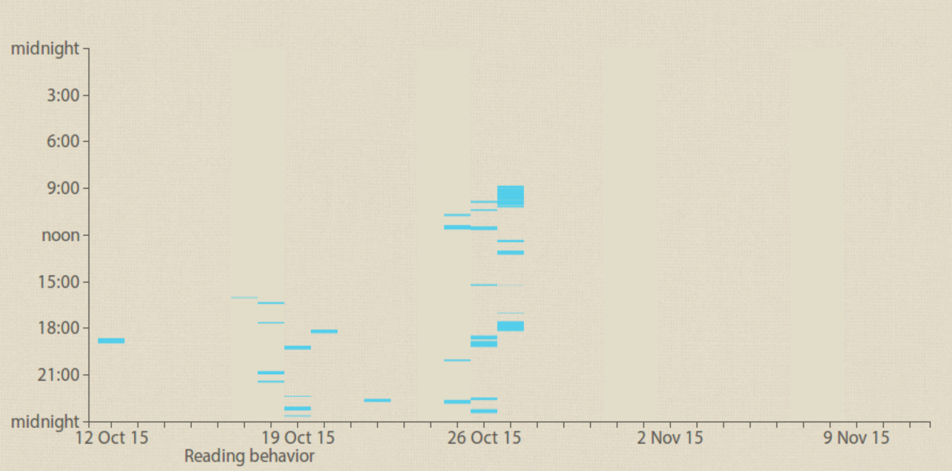
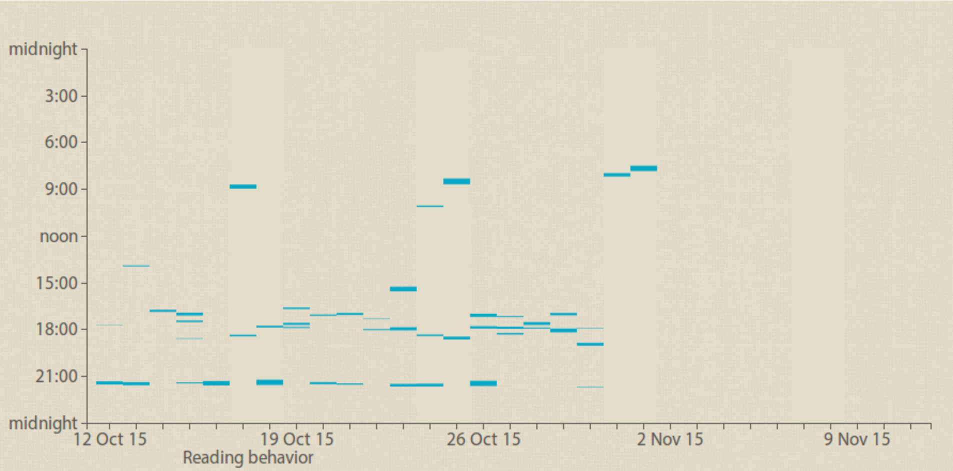
Value	Count	Percent
Mr.	517	58.025%
Miss.	185	20.763%
Mrs.	125	14.029%
Master.	40	4.489%
Dr.	7	0.786%
Rev.	6	0.673%
Sir.	5	0.561%
Col.	2	0.224%
Jonkheer.	1	0.112%
Lady.	1	0.112%
the Countess.	1	0.112%
Ms.	1	0.112%



Kaggle


midnight
3:00
6:00
9:00
noon
15:00
18:00
21:00
midnight
12 Oct 15
19 Oct 15
26 Oct 15
2 Nov 15
9 Nov 15
Reading behavior

Jellybooks


midnight
3:00
6:00
9:00
noon
15:00
18:00
21:00
midnight
12 Oct 15
19 Oct 15
26 Oct 15
2 Nov 15
9 Nov 15
Reading behavior

Jellybooks

pandas

```
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
```

pandas

`Dataframe` has many constructors. For example,

```
In [5]: pd.DataFrame({ 'A' : 1.,
                 'B' : pd.Timestamp('20161209'),
                 'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
                 'D' : np.array([3] * 4, dtype='int32'),
                 'E' : pd.Categorical(["test","train","test","train"]),
                 'F' : 'hello' })
Out[5]:
   A          B  C  D      E      F
0  1 2016-12-09  1  3   test  hello
1  1 2016-12-09  1  3  train  hello
2  1 2016-12-09  1  3   test  hello
3  1 2016-12-09  1  3  train  hello

In [6]:
```

Viewing data

```
In [16]: dates = pd.date_range('20161209', periods=4, freq='1w')

In [17]: df = pd.DataFrame(np.random.randn(4,5), index=dates,
                          columns=list('ABCDE'))

In [18]: df.head()
Out[18]:
                   A         B         C         D         E
2016-12-11 -1.303610 -1.235823  0.621914  0.379340 -0.326934
2016-12-18 -1.218197 -1.113826  0.546314 -0.255001 -0.135573
2016-12-25 -0.124625  0.337268 -0.406295  0.587049 -0.904906
2017-01-01 -0.283182 -0.866213  0.051509  0.693037 -0.661055

In [19]:
```

Basic data exploration

```
In [19]: df.describe()
Out[19]:
              A         B         C         D         E
count  4.000000  4.000000  4.000000  4.000000  4.000000
mean  -0.732403 -0.719648  0.203361  0.351106 -0.507117
std    0.614672  0.721194  0.478728  0.424558  0.342755
min   -1.303610 -1.235823 -0.406295 -0.255001 -0.904906
25%   -1.239550 -1.144325 -0.062942  0.220755 -0.722018
50%   -0.750689 -0.990019  0.298912  0.483195 -0.493995
75%   -0.243543 -0.565343  0.565214  0.613546 -0.279094
max   -0.124625  0.337268  0.621914  0.693037 -0.135573

In [20]:
```

Select a column (series)

```
In [20]: df.loc[dates[1]]
Out[20]:
A   -1.218197
B   -1.113826
C    0.546314
D   -0.255001
E   -0.135573
Name: 2016-12-18 00:00:00, dtype: float64

In [21]:
```

Select a range

```
In [21]: df.loc[:, ['A', 'C']]
Out[21]:
                   A         C
2016-12-11 -1.303610  0.621914
2016-12-18 -1.218197  0.546314
2016-12-25 -0.124625 -0.406295
2017-01-01 -0.283182  0.051509

In [22]:
```

Boolean selection criteria

```
In [23]: df[df.D > 0]
Out[23]:
                   A         B         C         D         E
2016-12-11 -1.303610 -1.235823  0.621914  0.379340 -0.326934
2016-12-25 -0.124625  0.337268 -0.406295  0.587049 -0.904906
2017-01-01 -0.283182 -0.866213  0.051509  0.693037 -0.661055

In [24]:
```

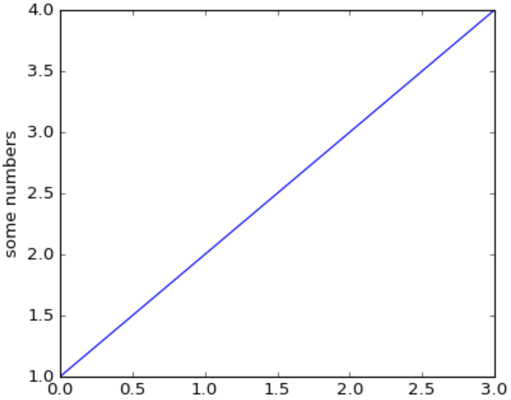
Recommended

`http://www.gregreda.com/2013/10/26/intro-to-pandas-data-structures/`

Plotting

Draw a line

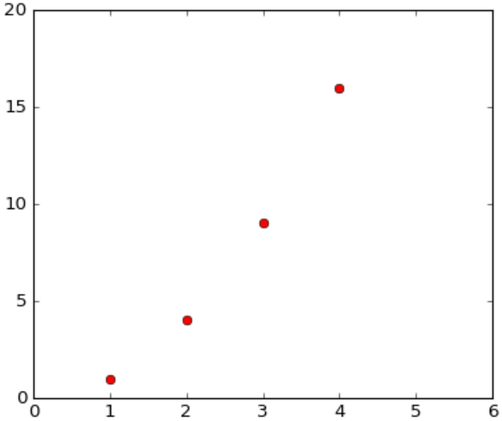
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



Plotting

Draw a line

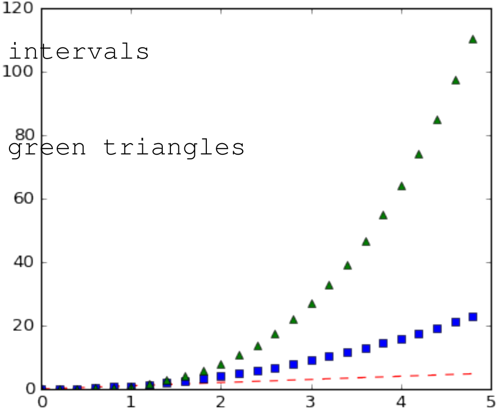
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.ylabel('some numbers')
plt.show()
```



Plotting

Draw a line

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

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t,
         'r--', t,
         t**2, 'bs',
         t, t**3, 'g^')
plt.show()
```

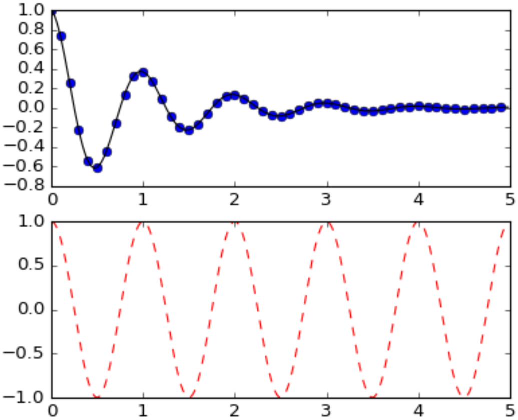


Plotting

Draw two curves

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

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

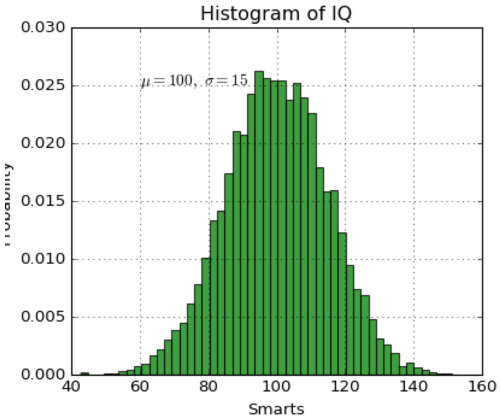
Plotting



Draw two curves

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

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

Plotting



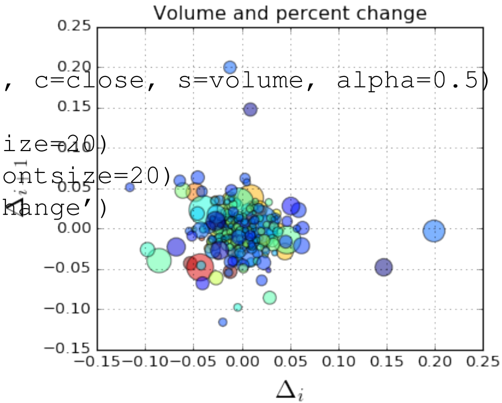
Histogram of IQ
$\mu=100,\ \sigma=15$
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140
160
Smarts

Plotting

Scatter plot

http://matplotlib.org/mpl_examples/pylab_examples/scatter_demo2.py

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

fig, ax = plt.subplots()
ax.scatter(delta1[:-1], delta1[1:], c=close, s=volume, alpha=0.5)

ax.set_xlabel(r'$\Delta_i$', fontsize=20)
ax.set_ylabel(r'$\Delta_{i+1}$', fontsize=20)
ax.set_title('Volume and percent change')

ax.grid(True)
fig.tight_layout()

plt.show()
```



Plotting

http://matplotlib.org/users/pyplot_tutorial.html

http://matplotlib.org/users/beginner.html

Linear models

Problem: $\{(x_i, y_i)\}$.

Given x, predict $\hat{y}$.

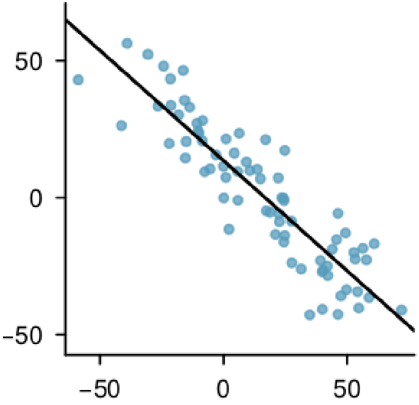
Linear models

x: **explanatory** or **predictor** variable.

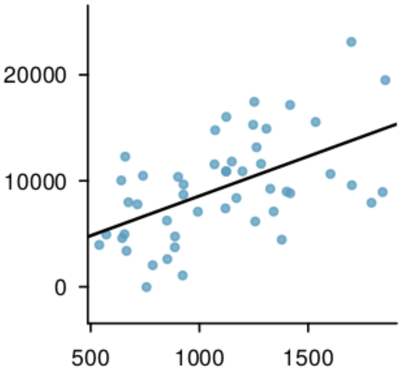
y: **response** variable.

For some reason, we believe a linear model is a good idea.

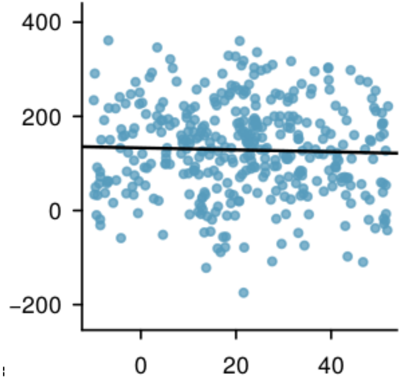
Example:



Example:

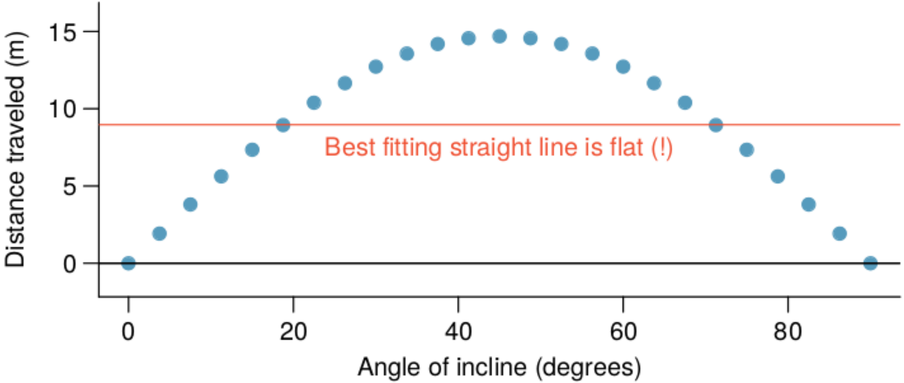


Example:



Linear models

Example:



Residuals

What's left over.

$$\text{data} = \text{fit} + \text{residual}$$

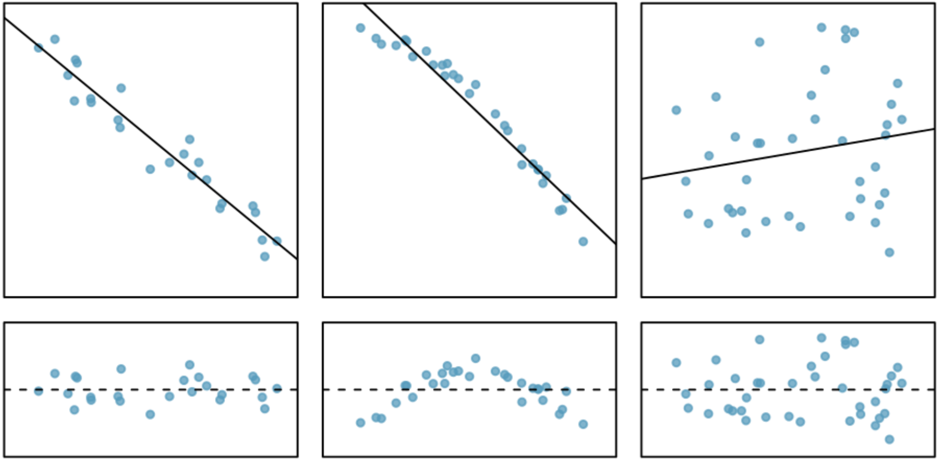
Residuals

What's left over.

$$y_i = \hat{y}_i + e_i$$

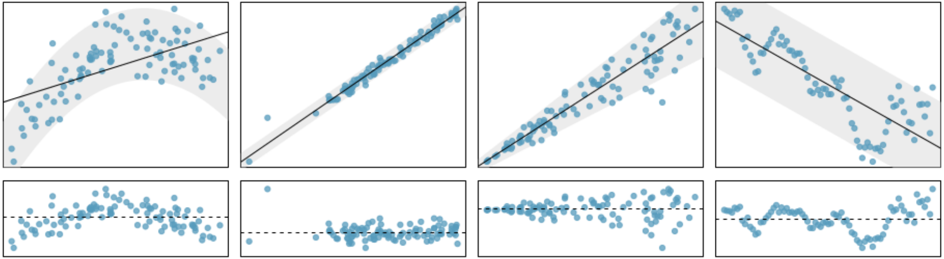
Residuals

What's left over.



Residuals

What's left over.



Residuals

What's left over.

Goal: small residuals.

$$\sum \mid e_i \mid$$

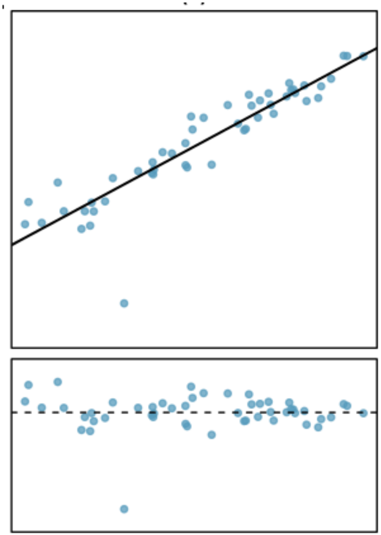
Residuals

What's left over.

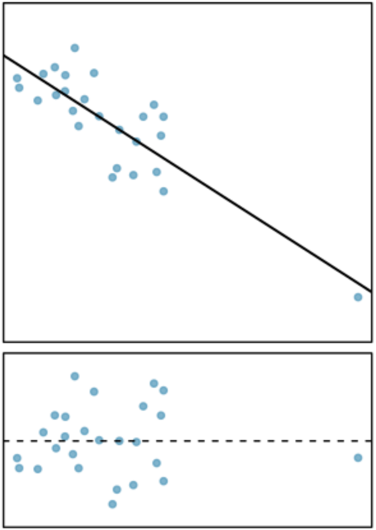
Goal: small residuals.

$$\sum e_i^2$$

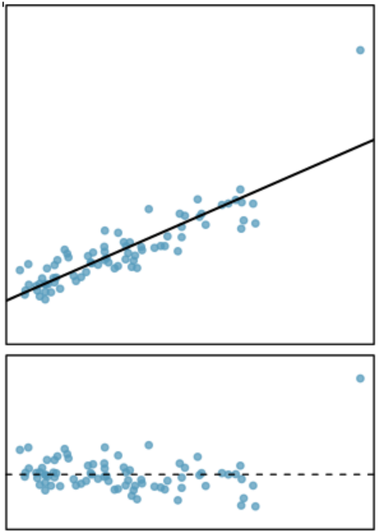
Outliers



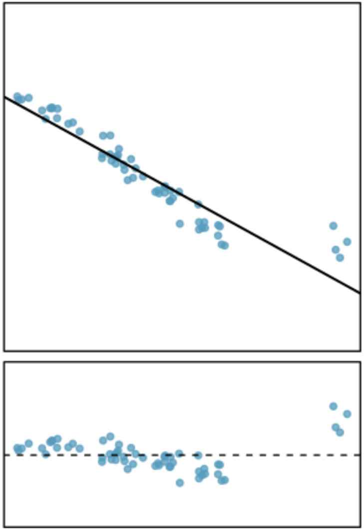
Outliers



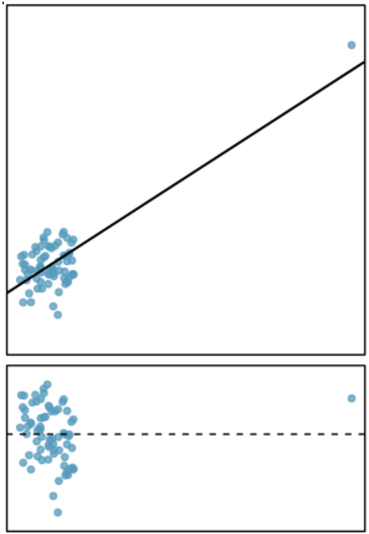
Outliers



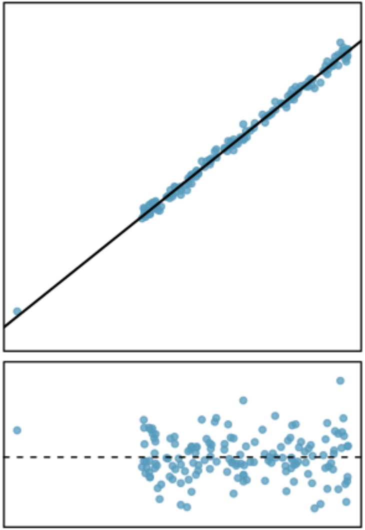
Outliers



Outliers

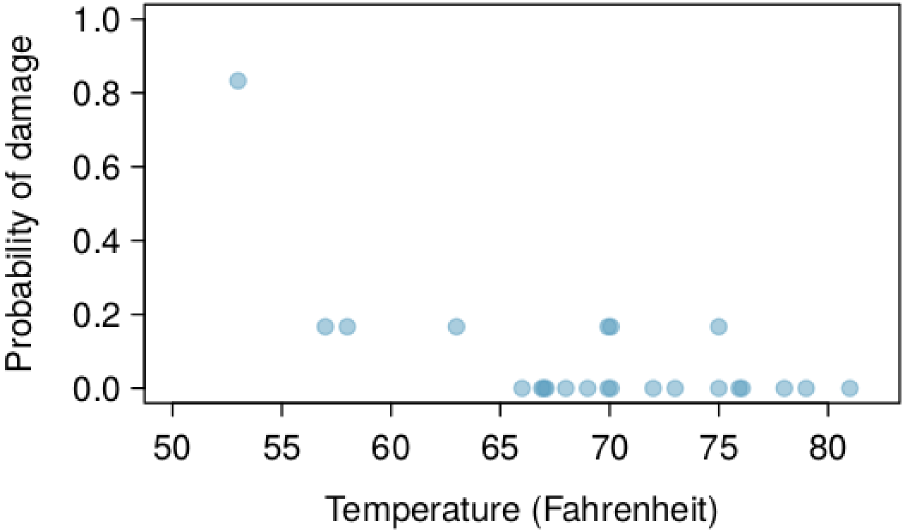


Outliers



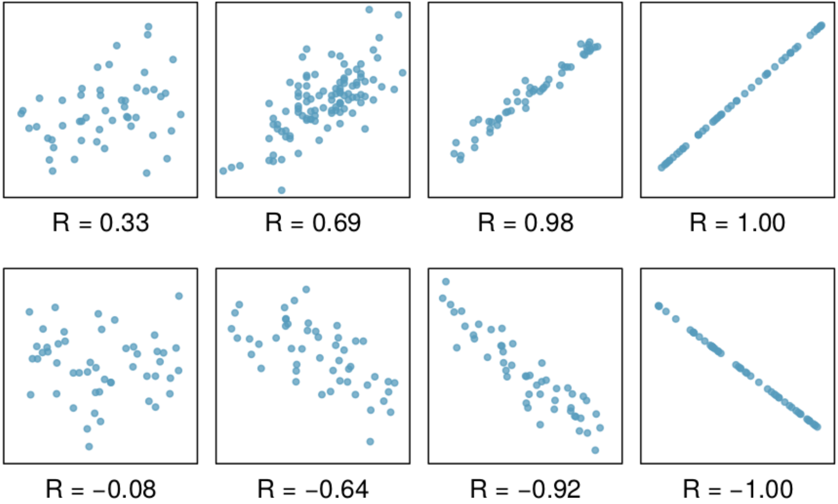
Don't ignore outliers.

Outliers

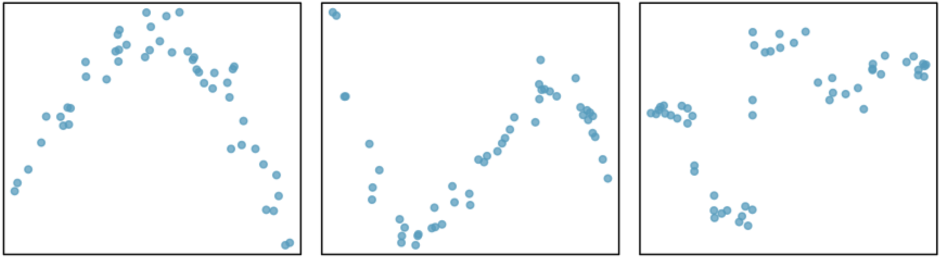


Probability of damage
1.0
0.8
0.6
0.4
0.2
0.0
50
55
60
65
70
75
80
Temperature (Fahrenheit)

Correlation



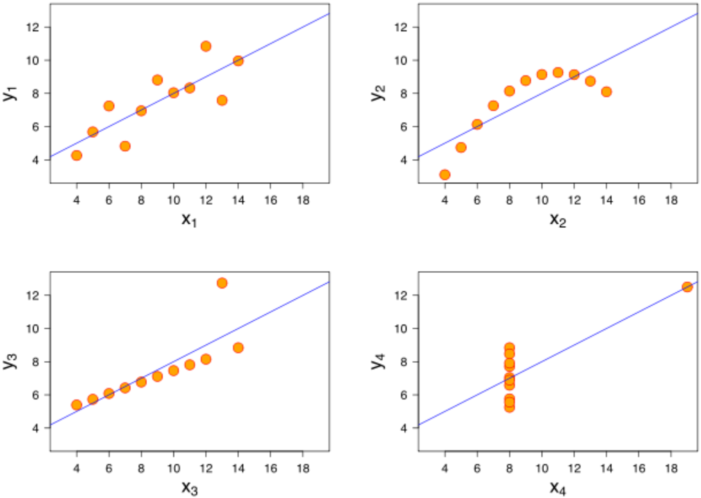
Correlation



R = −0.23 R = 0.31 R = 0.50

Correlation

Anscombe's Quartet



Correlation does not imply causation

Hypothesis (model)

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Cost function

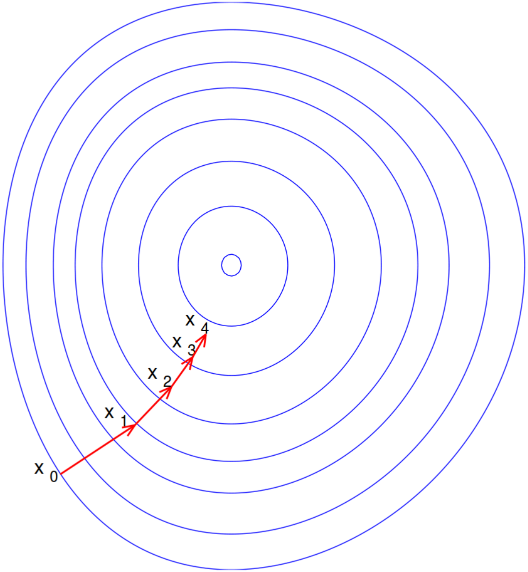
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Gradient descent

$$\begin{cases} \theta_0 & \leftarrow \theta_0 - \alpha \dfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \theta_1 & \leftarrow \theta_1 - \alpha \dfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \end{cases}$$

Gradient descent

$$\begin{cases} \theta_0 & \leftarrow \theta_0 - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) \\ \theta_1 & \leftarrow \theta_1 - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) \end{cases}$$


x 0
x 1
x 2
x 3
x 4

Hypothesis again

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \theta_1 x_1 \\
&= \theta_0 + \sum_{i=1}^{1} \theta_i x_i \\
&= [\theta_0, \theta_1] \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \\
&= \theta^T x
\end{aligned}
$$

Hypothesis (multiple regression)

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \sum_{i=1}^{n} \theta_i x_i \\
&= [\theta_0, \cdots, \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \\
&= \theta^T x
\end{aligned}
$$

Hypothesis (multiple regression)

$$
\begin{aligned}
h_\theta(x) &= \theta^T x \\
&= \theta^T x^{(1)}
\end{aligned}
$$

Hypothesis (multiple regression)

$$X = \begin{bmatrix} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & \cdots & | \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(m)} \end{bmatrix}$$

Hypothesis (multiple regression)

$$
\begin{aligned}
h_\theta(X) &= \theta^T X \\
&= \left[h_0(x^{(1)}), h_0(x^{(2)}), \cdots, h_0(x^{(m)})\right] \\
&= \theta^T X
\end{aligned}
$$

Hypothesis (multiple regression)

or $X\theta$ if row vectors...

Cost function (multiple regression)

$$
\begin{aligned}
J(\theta) &= \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\
&= \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)
\end{aligned}
$$

Gradient descent (multiple regression)

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x_j^{(i)}$$

$$\text{for } j = 1, \cdots, n$$

Gradient descent (multiple regression)

$$\theta \leftarrow \theta - \nabla J(\theta)$$

$$\text{where } \nabla = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \\ \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_n} \end{bmatrix}$$