CS 696 Intro to Big Data: Tools and Methods
Spring Semester, 2019
Doc 10 Regression
Feb 26, 2019

Machine Learning

Supervised

Unsupervised

Reinforcement learning

Classification

Regression

Clustering

Density Estimation

Dimensionality Reduction

Supervised learning

Artificial neural network
Bayesian statistics
Bayesian network
Gaussian process regression
Inductive logic programming
Learning Vector Quantization
Logistic Model Tree
Nearest Neighbor Algorithm
Random Forests
Ordinal classification
ANOVA
Linear classifiers
Fisher's linear discriminant
Linear regression
Logistic regression
Multinomial logistic regression
Naive Bayes classifier

Quadratic classifiers
k-nearest neighbor
Boosting
Decision trees
Random forests
Bayesian networks
Naive Bayes
Hidden Markov models

Unsupervised learning

Expectation-maximization algorithm
Vector Quantization
Generative topographic map
Information bottleneck method
Artificial neural networks

Hierarchical clustering
- Single-linkage clustering
- Conceptual clustering
- Cluster analysis[edit]
- K-means algorithm
- Fuzzy clustering
- DBSCAN
- OPTICS algorithm

Outlier Detection
- Local Outlier Factor

Other

Reinforcement learning
- Temporal difference learning
- Q-learning
- Learning Automata
- SARSA

Deep learning
- Deep belief networks
- Deep Boltzmann machines
- Deep Convolutional neural networks
- Deep Recurrent neural networks
- Hierarchical temporal memory

Machine Learning & Patterns

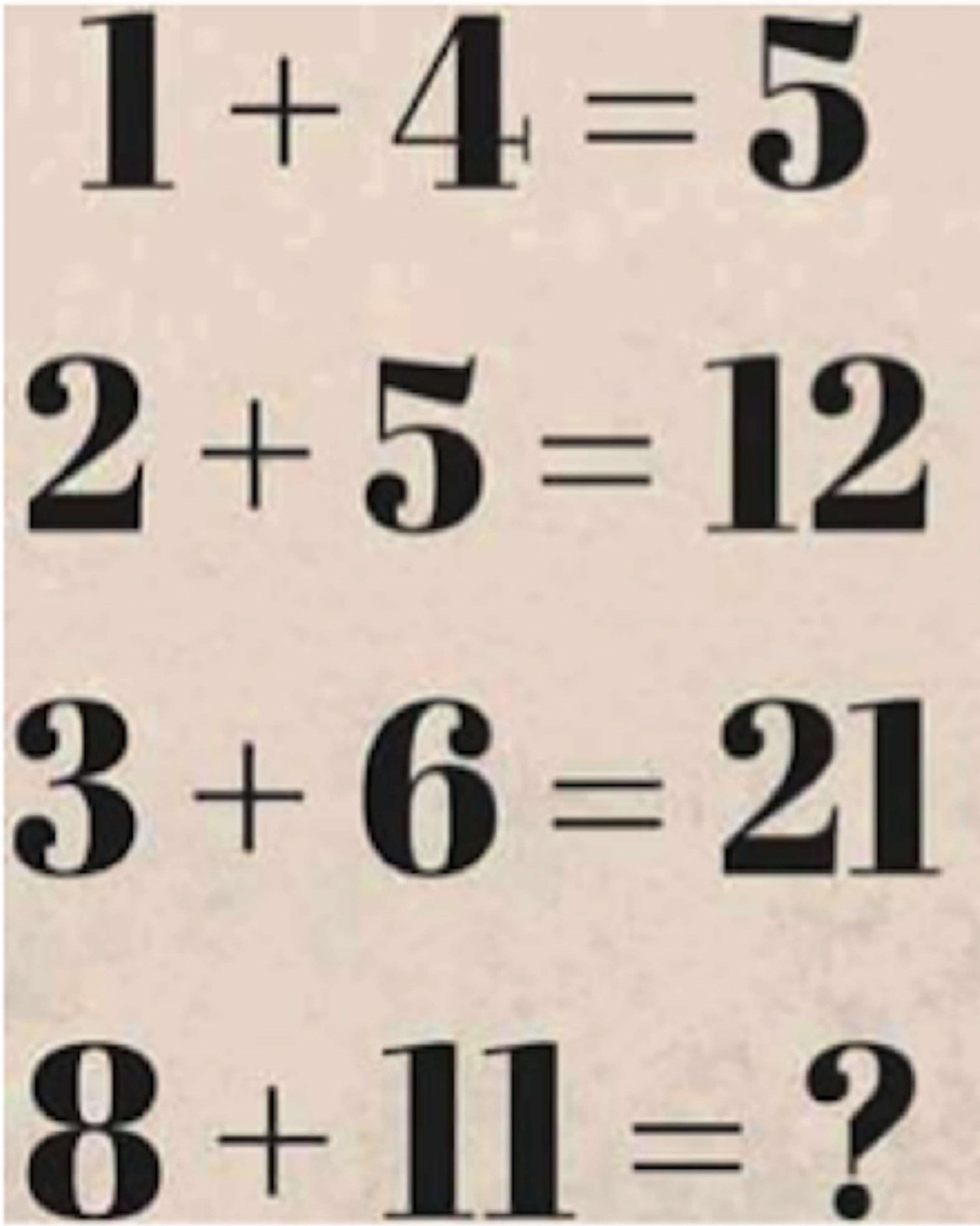
Machine learning algorithms
  Detect patterns
  Generate models based on those patterns

Feed a neural network pictures of cats
  Neural net can identify cats
  Can automate finding cat photo on internet

Drive a car with neural network “watching”
  You actions
  Videos of surroundings

  Neural net can identify patterns & start to drive

Limits of Pattern Matching



1 * (4 + 1) = 5

2 * (5 + 1) = 12

3 * (6 + 1) = 21

8 * (11 + 1) = 96

0 + 1 + 4 = 5

5 + 2 + 5 = 12

12 + 3 + 6 = 21

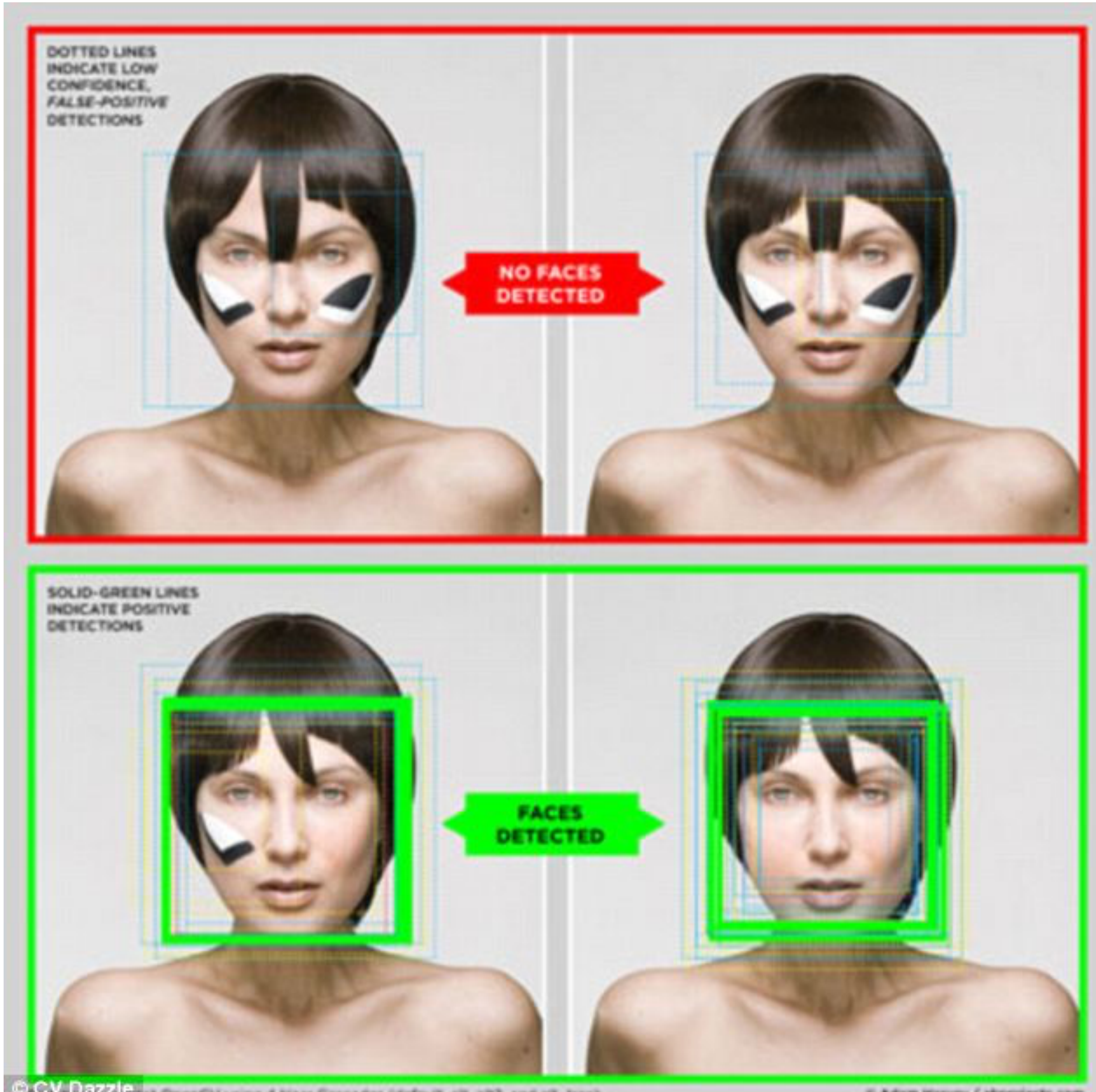
21 + 8 + 11 = 40

No Free Lunch Theorems

David Wolpert

For every pattern a machine learning algorithm is good at learning, there's another pattern that same learner would be terrible at picking up

No Free Lunch



7 Deadly Sins of AI Predictions

Rodney Brooks, October 6, 2017

https://goo.gl/oK6z5Z

1. Amara's law

We tend to overestimate the effect of a technology in the short run
and underestimate the effect in the long run.

Example U.S. Global Positioning System (GPS)

- Started 1978
- Precise delivery of bombs
- First real use 1991, but not fully embraced by US military for several more years
- Now
  - On mobile phones
  - Tracks planes, trucks
  - Sync US electrical grid
  - Determines which seed to plant in a field

7 Deadly Sins of AI Predictions

Rodney Brooks, October 6, 2017

https://goo.gl/oK6z5Z

3. Performance versus competence

7 Deadly Sins of AI Predictions

Rodney Brooks, October 6, 2017

https://goo.gl/oK6z5Z

3. Exponentials

Exponential growth not sustainable

iPod memory

year	gigabytes
2002	10
2003	20
2004	40
2006	80
2007	160

Deep Learning Breakthrough

Breakthrough paper on deep learning - back propagation

1986

Idea was abandoned for ~20 years because it was not producing results

Models

Machine Learning algorithms produce models

Models allow predictions or offer insights

Examples

Decreasing latency by X increases Amazon's daily revenue by Y

White males without college degrees favor Trump by X%
Females favor Clinton by Y%
...

Models Approximate Reality

World is flat

World is a sphere

World is an oblate ellipsoid

Does the model provide useful predictions/insights

Under what condidtions is the model useful

What are the estimates of the model’s error

Multiple Factors in Model

Amazon's daily revenue depends on

- Latency
- Price
- Steps needed to order
- Page layout
- Relevant suggestions
- Search results
- Font sizes
- Color
- Shipping costs

Some factors will be more important

Stochastic in nature

Independent variables

Regression

Regression

Measure of relation between mean of one variable (dependent) on

one or more other variables (independent)

In chapter 11 of Julia for Data Science

Download the Jupyter notebook before reading

https://technicspub.com/analytics/
https://app.box.com/v/codefiles

Overview

Linear regression

Multiple linear regression

Generalized linear regression (model)

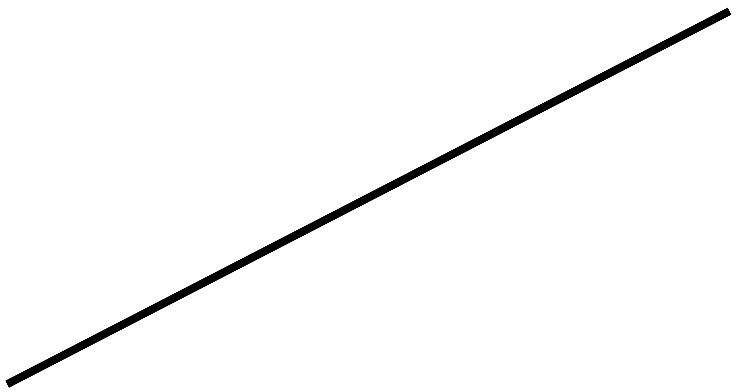
Is the dependent variable related to the independent variable

Generating the model

Error in the model

Effect of independent variables

Linear Regression



Model

f(x) = 2x + 3

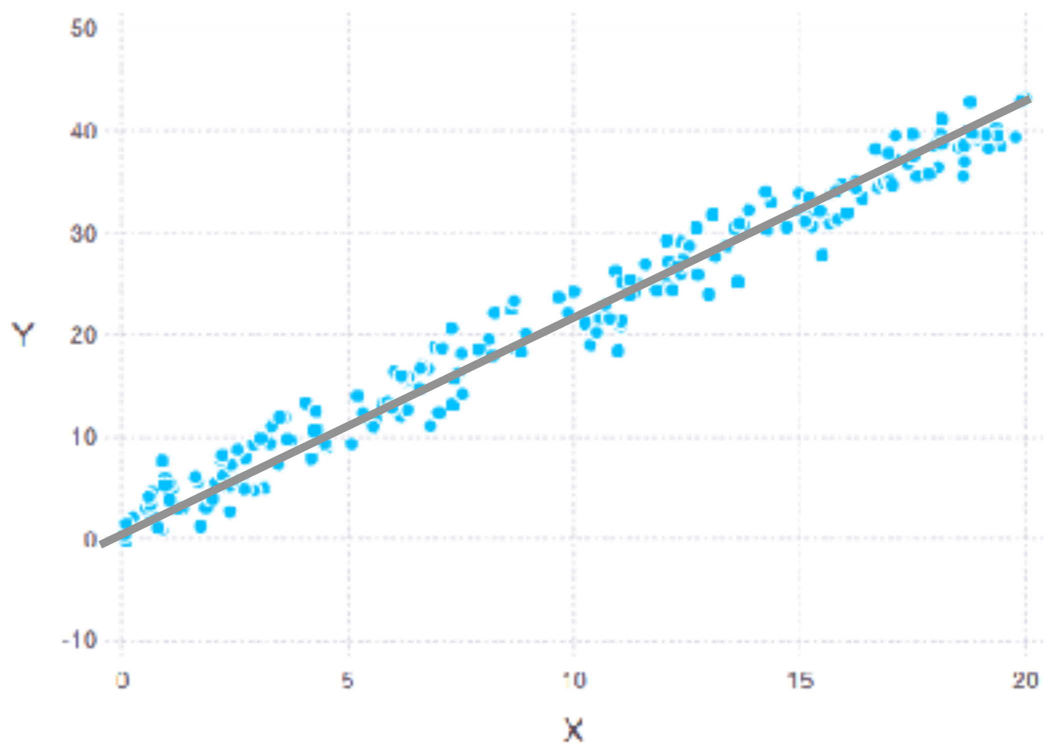
y = 2x + 3

y = 2x + 3

Dependent Variable

Independent Variable

Linear Regression



Actual relation (assumed)

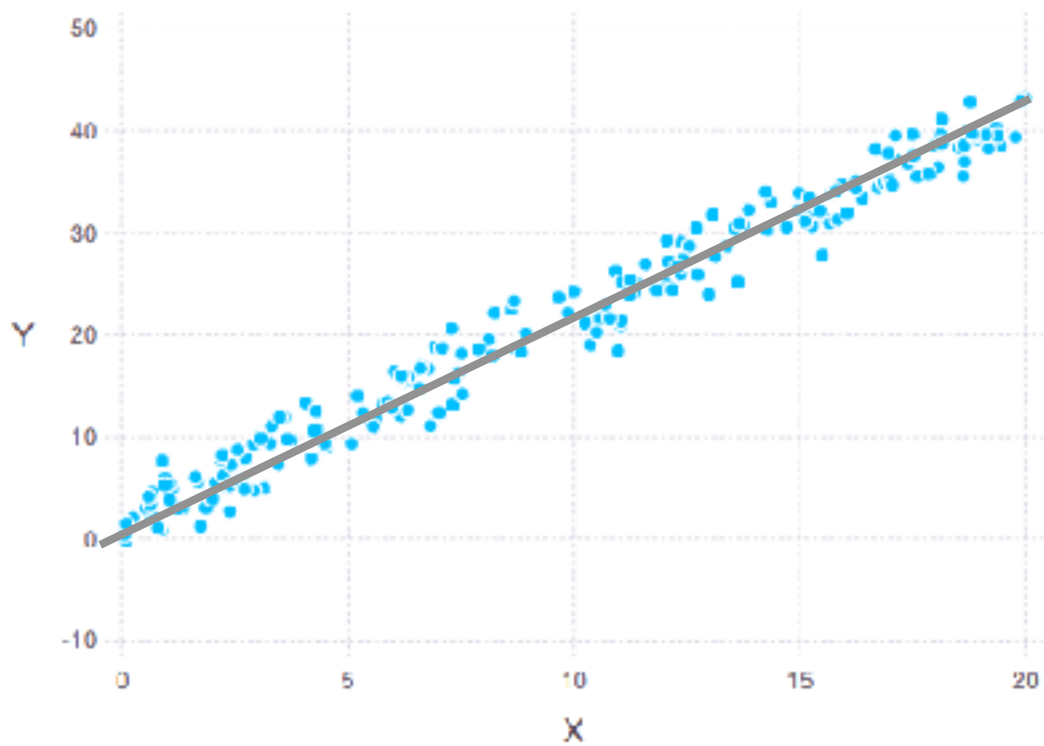
$y = a + bx$

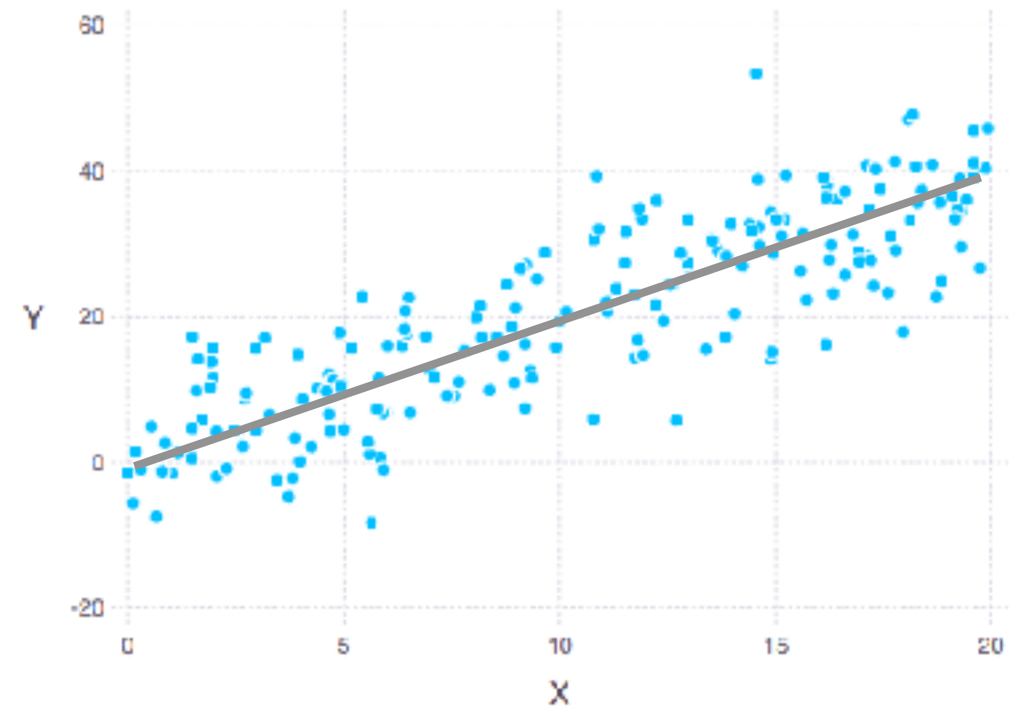
Compute linear line that fits the data best

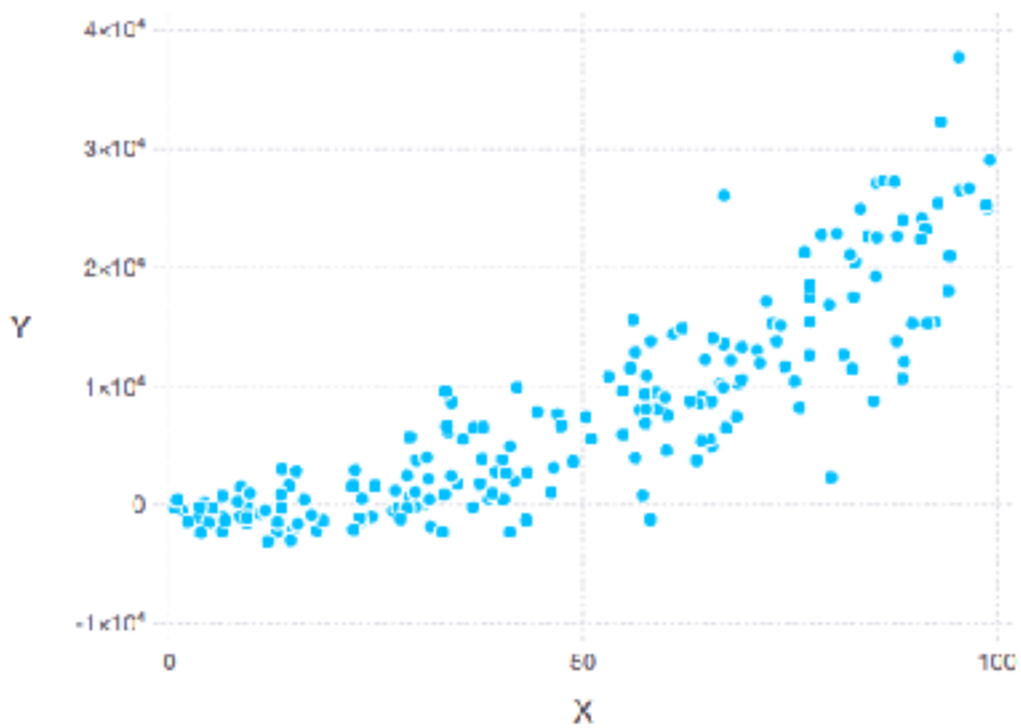
$\hat{y} = a + bx + e$

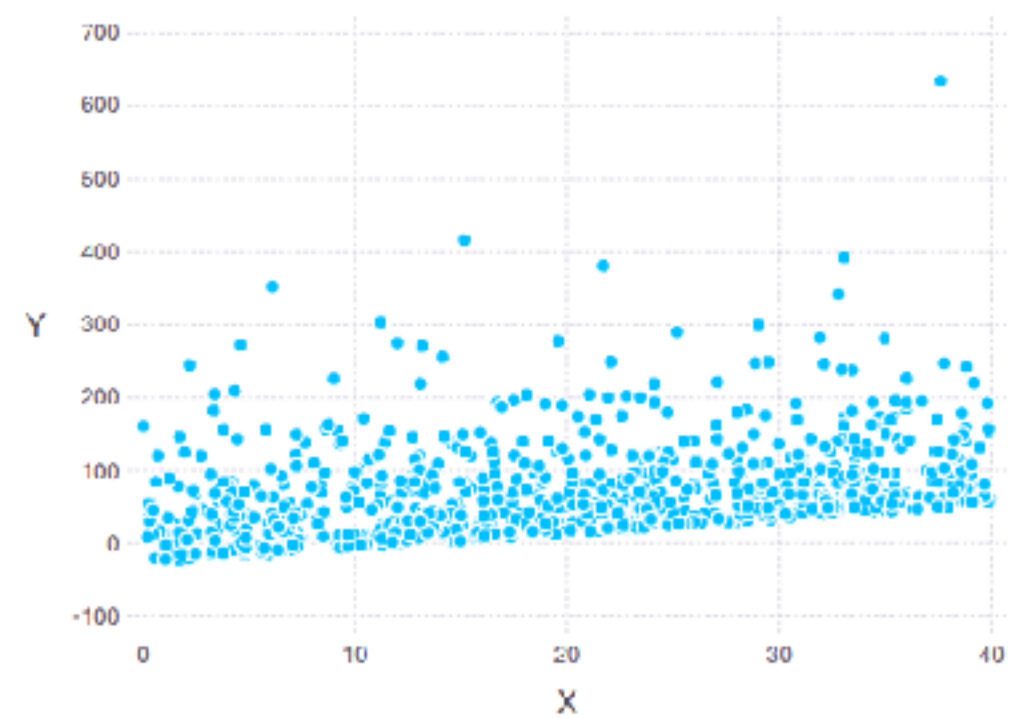
e - error or residual

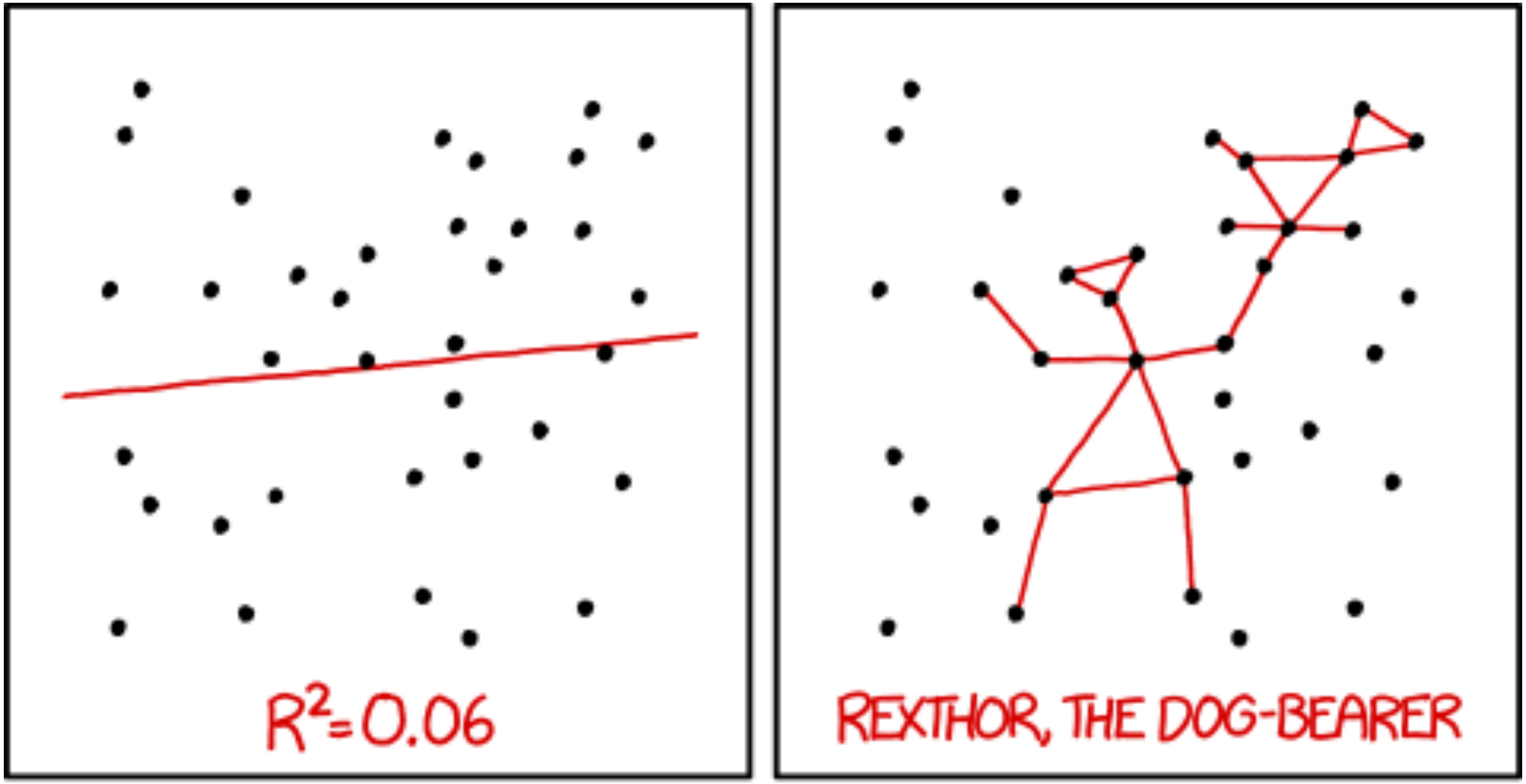
Goal is to minimize residual overall


50
40
30
20
10
0
-10
Y
0
5
10
15
20
X


60
40
20
0
-20
Y
0
5
10
15
20
X

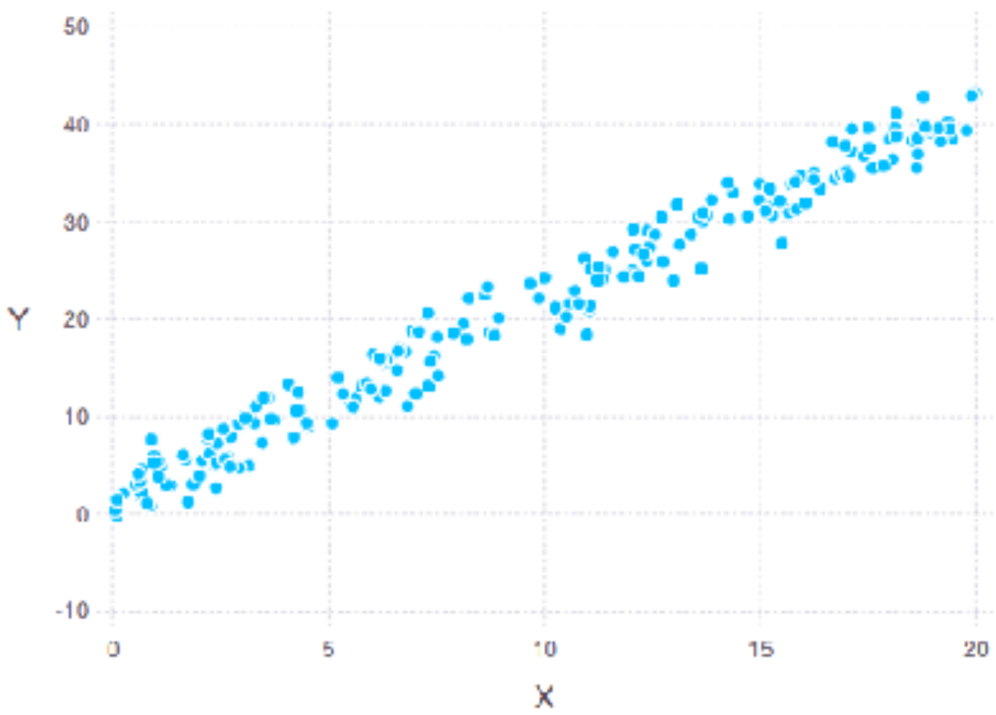

4×10⁴
3×10⁴
2×10⁴
1×10⁴
0
-1×10⁴
Y
0
50
100
X

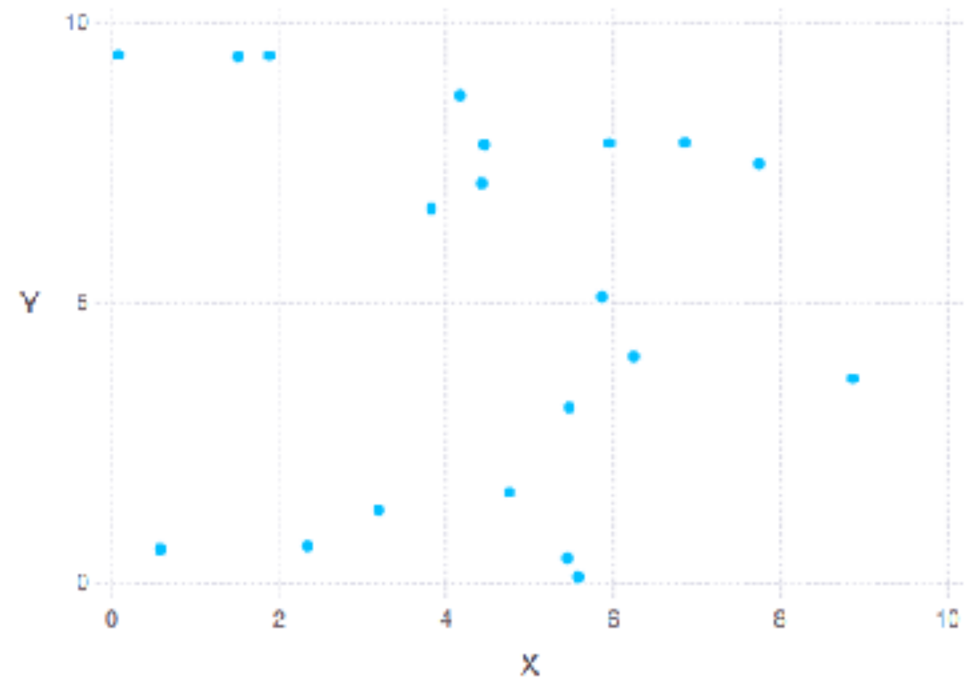

700
600
500
400
300
200
100
0
-100
Y
0
10
20
30
40
X



I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

Are They Related?





Covariance

If x & y are related then they should vary from their means in a similar way

$$dx_i = x_i - \overline{x}$$

Values near zero indicate no relation

positive values - positive relation

$$dy_i = y_i - \overline{y}$$

$$\operatorname{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n} dx_i dy_i$$

negative values - negative relation

DataFrame.cov

Effects of Scale

1 Pound = 452.6 grams

Cost USD	Pounds	Grams
9	3	1357.8
24	7	3168.2
38	10	4526

Changing the scale of units
- Does not change the relationship
- Does change magnitude of Covariance

Makes covariance hard to evaluate

```
cost_usd = pd.Series([9, 24, 38])
pounds = pd.Series([3, 7, 10])
grams = pd.Series([1357.8, 3168.2, 4526])
cost_usd.cov(pounds)                          50.8

cost_usd.cov(grams)                           23007.2

cost_inr = cost_usd * 71.04                   1_634_429.1
cost_inr.cov(grams)
```

```
pounds.cov(cost_usd)
                                              50.8
```

```
data = pd.DataFrame({"cost": [9, 24, 38],
                     "pounds":[3, 7, 10],
                     "grams": [1357.8, 3168.2, 4526]})
```

	cost	pounds	grams
0	9	3	1357.8
1	24	7	3168.2
2	38	10	4526

```
data.cov()
```

	cost	pounds	grams
cost	210.333333	50.833333	2.300717E+04
pounds	50.833333	12.333333	5.582067E+03
grams	23007.166667	5582.066667	2.526443E+06

Units

$$dx_i = x_i - \overline{x}$$ Lbs

$$dy_i = y_i - \overline{y}$$ USD

$$\text{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n} dx_i dy_i$$

cost_usd.cov(pounds) == 50.8 lbs*USD

cost_usd.cov(grams) == 23007 grams*USD

Cost USD	Pounds	Grams
9	3	1357.8
24	7	3168.2
38	10	4526

Normalizing Data

Convert data to a common scale

Example - divide by maximum value

Cost USD	Pounds	Grams
9	3	1357.8
24	7	3168.2
38	10	4526

Cost	Amount
0.237	0.3
0.632	0.7
1	1

```
normalized_amount = pounds/pounds.max()
normalized_cost = cost_usd/cost_usd.max()

normalized_amount.cov(normalized_cost)
```

0.134

Pearson's Correlation - r

$$r = \frac{\mathrm{cov}(X,Y)}{\sigma_x \sigma_y}$$

corr

Normalized Covariance

Unit less

Range -1 to 1

1 = maximumly related

-1 - maximumly inversely related

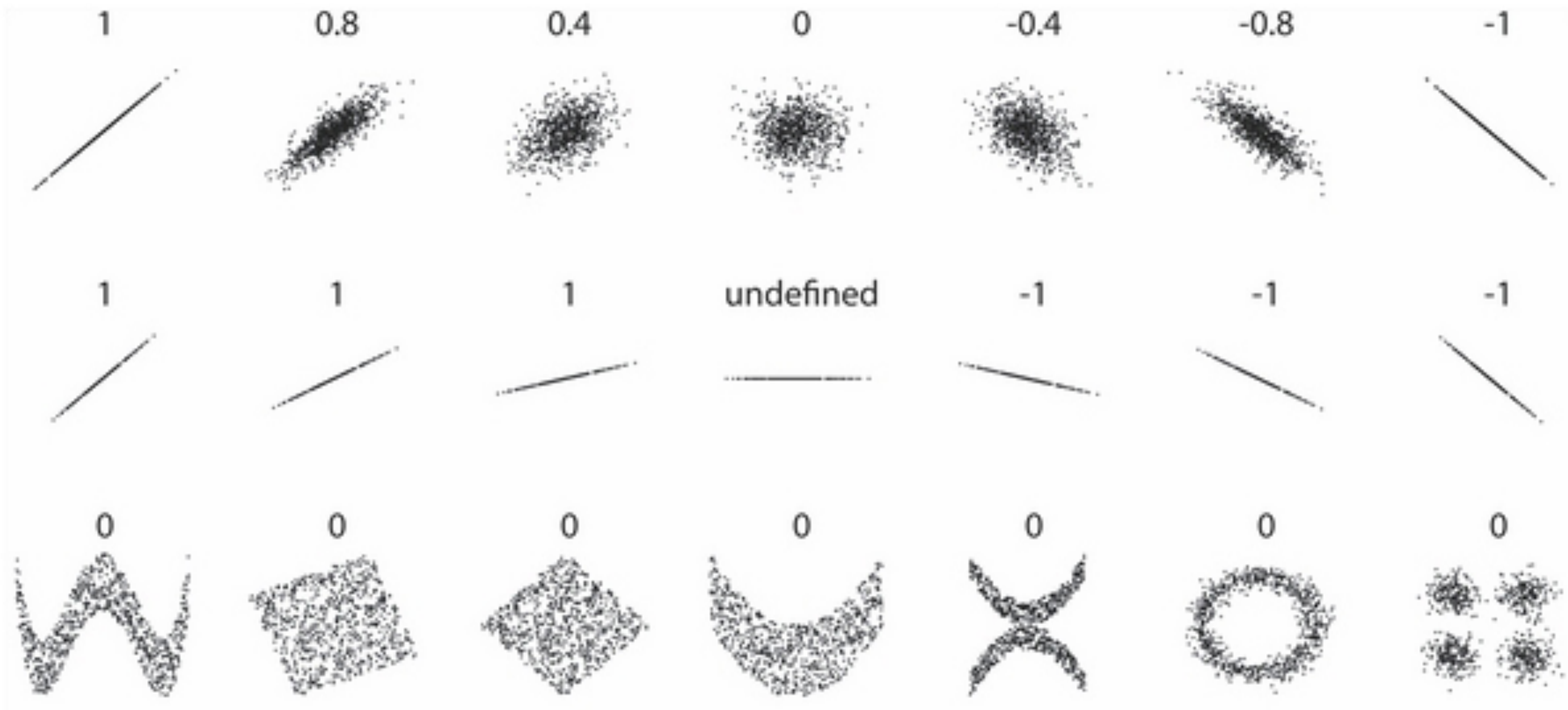
0 - not related

Pearson’s Correlation - r

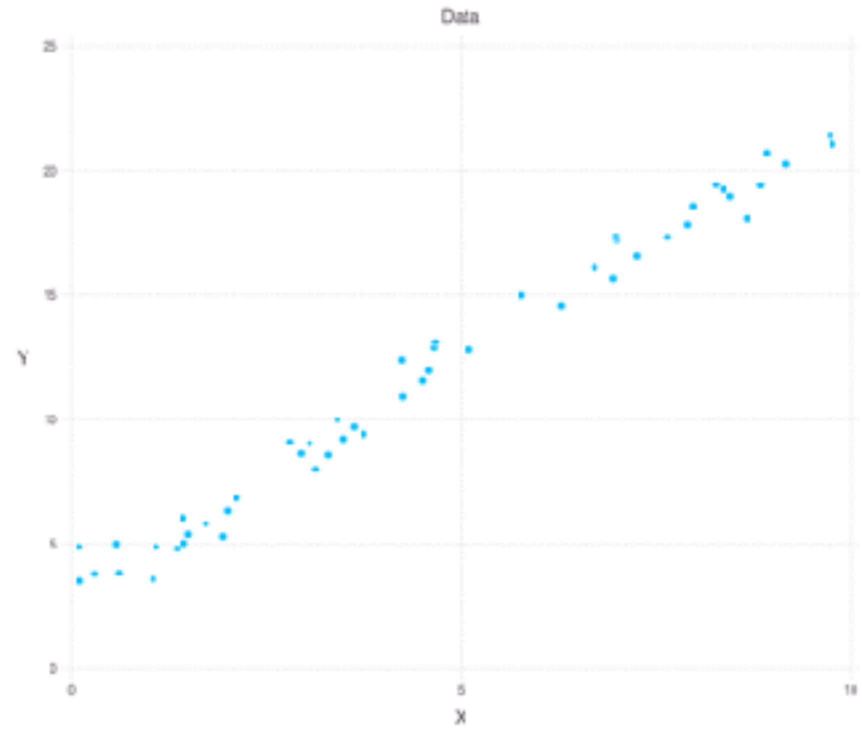
Cost USD	Pounds	Grams
9	3	1357.8
24	7	3168.2
38	10	4526

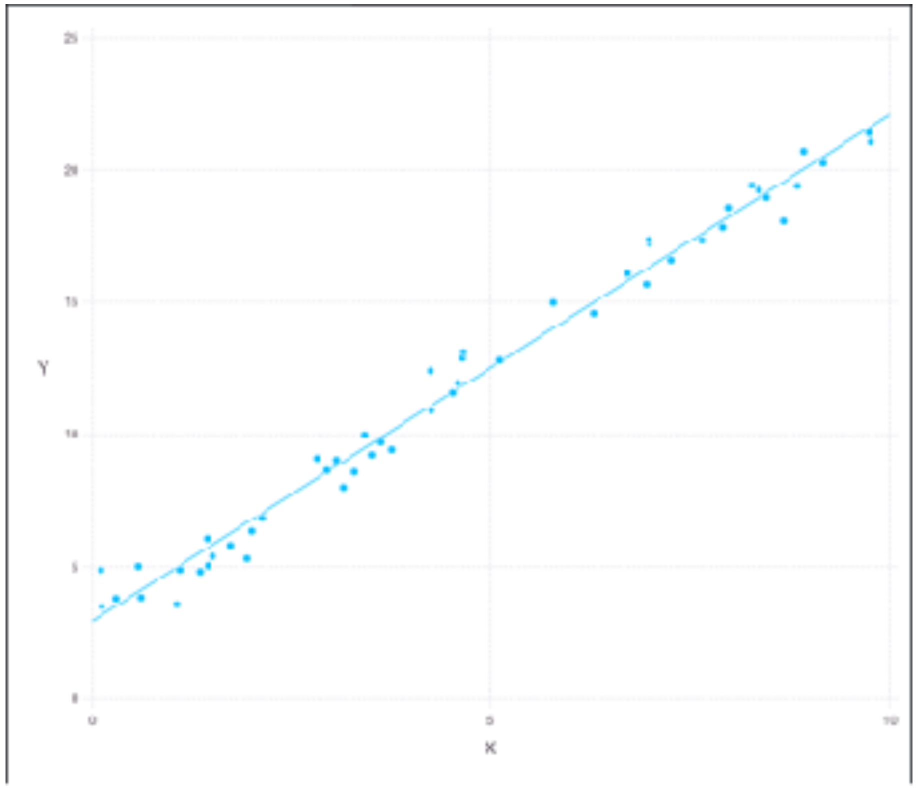
cost_usd.corr(pounds)
cost_usd.corr(grams) 0.998

Pearson’s Correlation r Value Examples



Regression Line





Pearson's Co
x.corr(y) == 0.992

What the line that minimizes the amount of residuals

Ordinary least squares

$$b = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Standard way to fit line to data

$$\mathrm{b} = \frac{\mathrm{cov}(\mathrm{X},\mathrm{Y})}{\mathrm{var}(\mathrm{X})}$$

$$a = \overline{y} - b\overline{x}$$

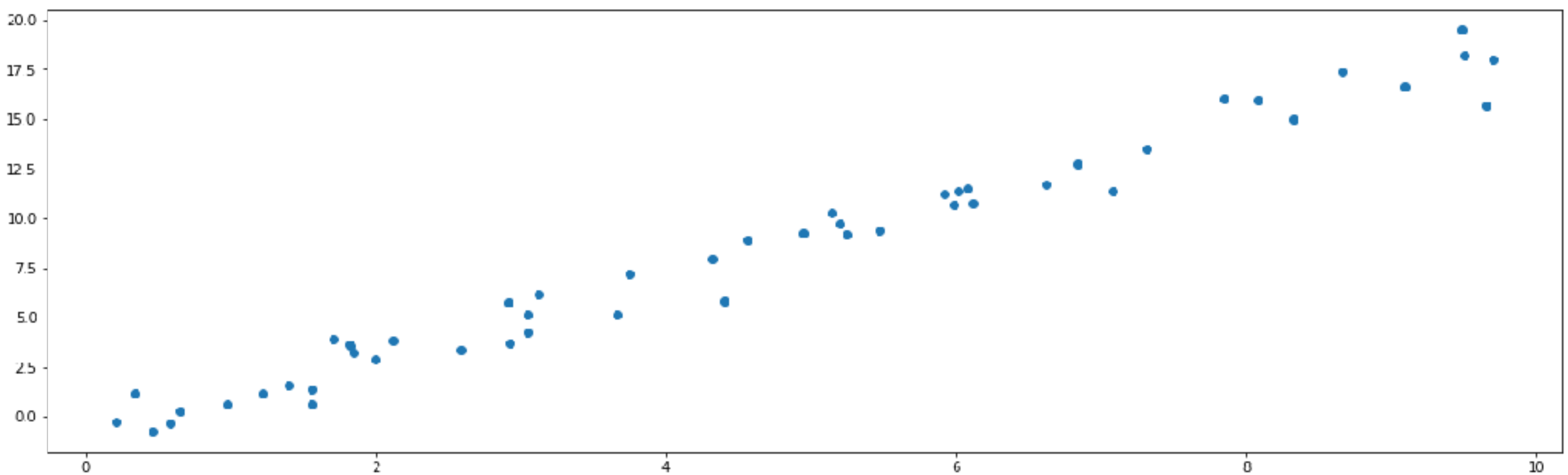
Computing Linear Regression - Some Data

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

rng = np.random.RandomState(42)
x = 10 * rng.rand(50)
y = 2 * x - 1 + rng.randn(50)
plt.figure(figsize=(20,6))
plt.scatter(x, y);
```

y is our fake observed values

```
pd.Series(x).corr(pd.Series(y))
```

0.99



Training the Model

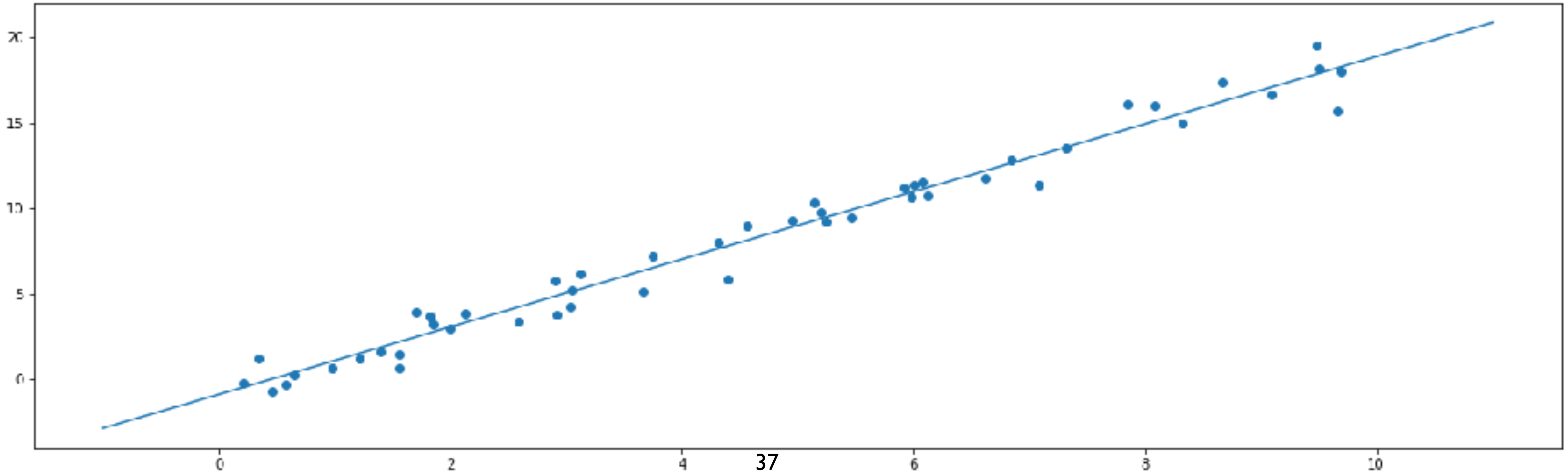
```
from sklearn.linear_model import LinearRegression

X = x[:,np.newaxis]                                         //sklearn requires 2D for independent var
model = LinearRegression(fit_intercept=True)
model.fit(X, y)


model.coef_[0]                      1.977656600385311


model.intercept_                   -0.9033107255311164
```

Regression line

y = 1.977656600385311 *x - 0.9033107255311164

Regression Line

```
def regression_function(model):
    return lambda x: x * model.coef_[0] + model.intercept_

regression = regression_function(model)
regression(1)

plt.figure(figsize=(20,6))
plt.scatter(x, y);

xfit = np.linspace(-1,11, num=50)
plt.plot(xfit, regression(xfit))
```



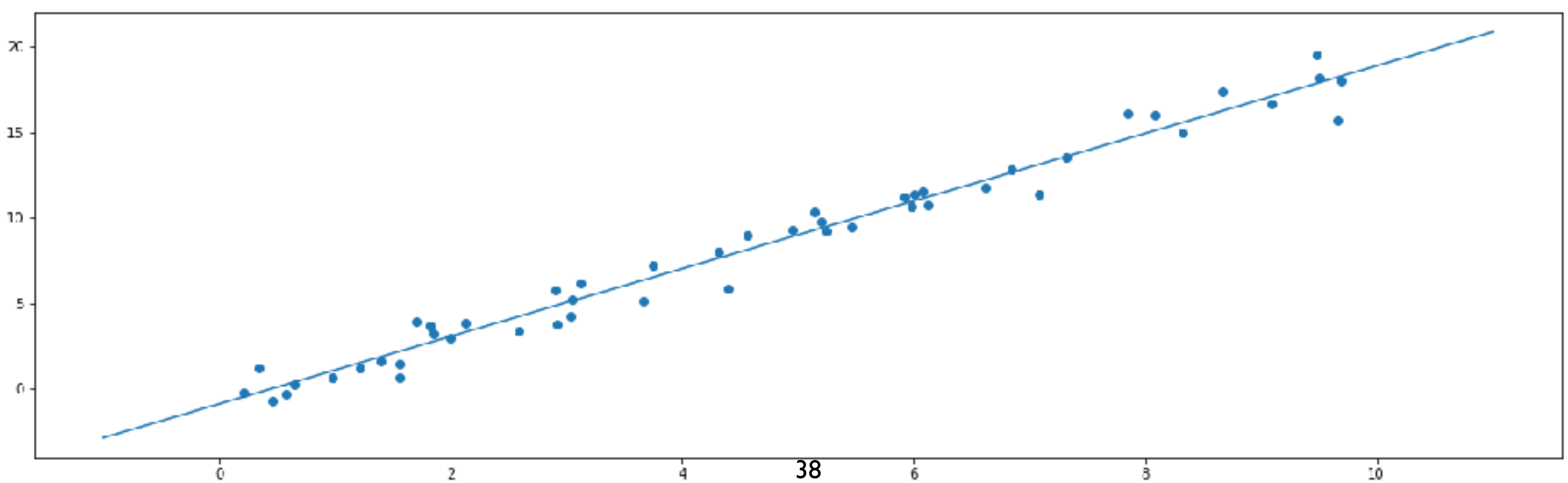
Regression Line

Linear regression is simple

Other models are not so simple

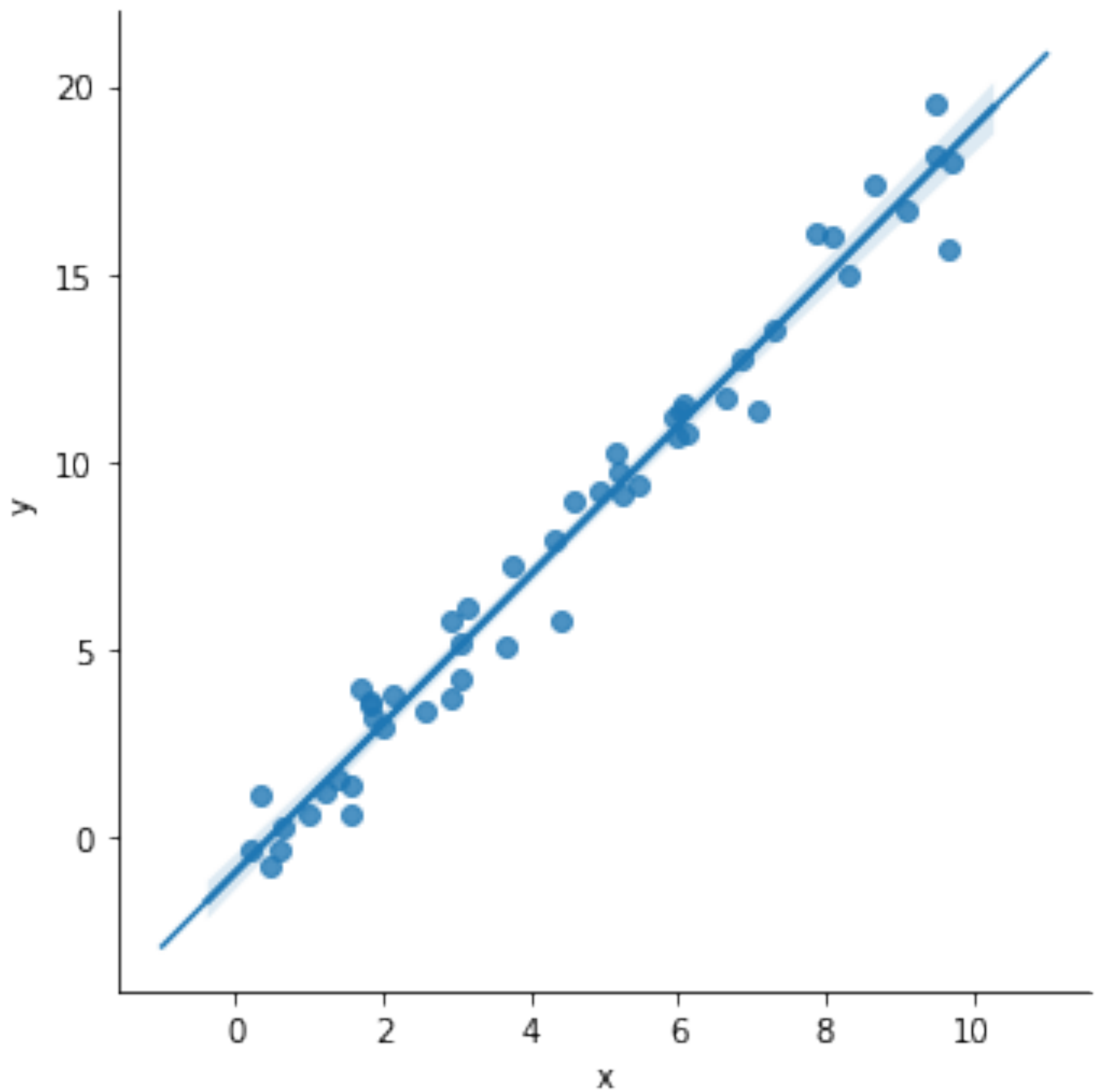
Let the model compute the regression line

```
xfit = np.linspace(-1,11, num=50)  # 50 evenly spaced points from -1 to 11
Xfit = xfit[:, np.newaxis]                # need 2D

yfit = model.predict(Xfit)

plt.figure(figsize=(20,6))
plt.scatter(x, y)
plt.plot(xfit, yfit);
```



Who does it compare to SNS Regression Line?

```
import seaborn as sns

sns.lmplot(x='x', y='y', data=pd.DataFrame({'x': x, 'y':y}))
plt.plot(xfit, yfit)
```



Is Linear Regression the Correct Model?

Scikit Learn Generalized Linear Models

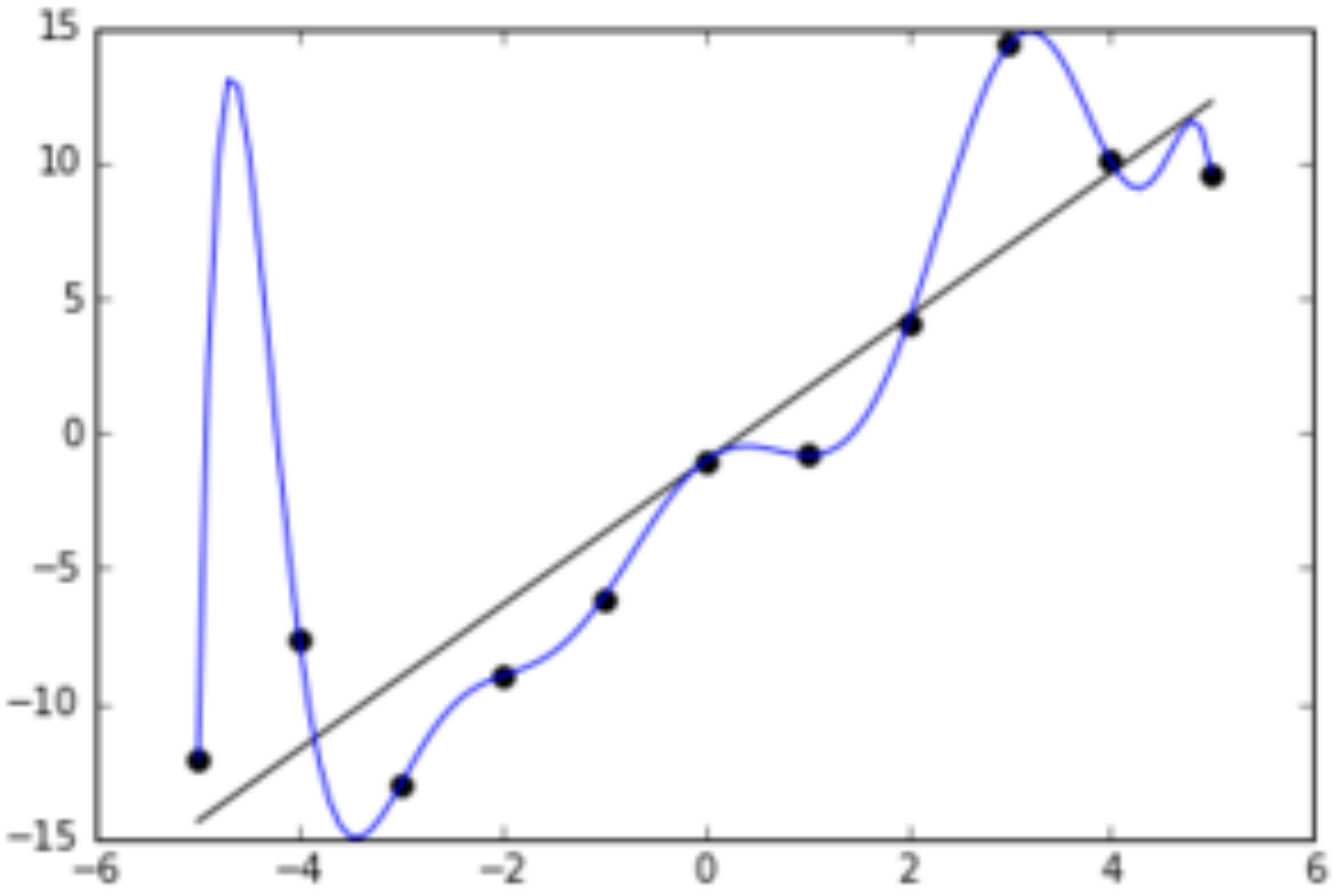
Ordinary Least Squares
Ridge Regression
Lasso
Multi-task Lasso
Elastic Net
Multi-task Elastic Net
Least Angle Regression
LARS Lasso
Orthogonal Matching Pursuit (OMP)
Bayesian Regression
Logistic regression
Stochastic Gradient Descent - SGD
Huber Regression
Polynomial regression

Overfitting

Model describes random error or noise instead of the underlying relationship

Overfitting occurs when a model is excessively complex,

Too many parameters relative to the number of observations



Is Linear Regression the Correct Model?

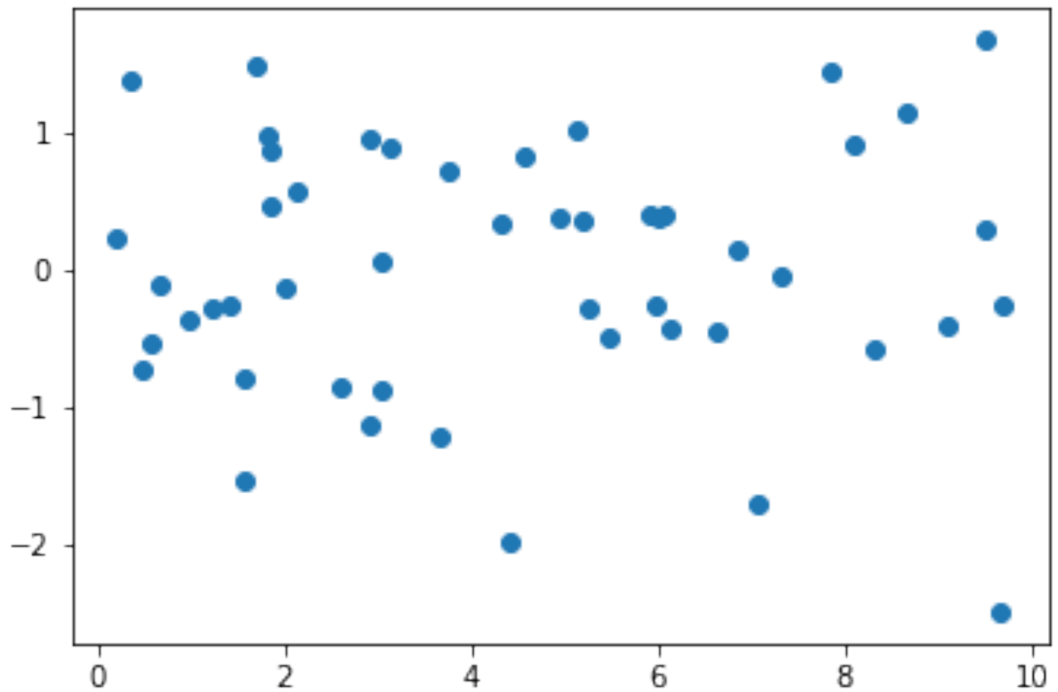
Residuals

R^2

Residuals

Residual = Observed value - Predicted value

```
rng = np.random.RandomState(42)
x = 10 * rng.rand(50)
y = 2 * x - 1 + rng.randn(50)  # Observed values
X = x[:,np.newaxis]

residual = y - model.predict(X)
plt.scatter(X,residual)
```



Residuals
- Should be randomly distributed
- Sum = 0
- mean = 0

Coefficient of Determination R^2

$$R^2 = 1 - \frac{\mathrm{var}(\varepsilon)}{\mathrm{var}(Y)}$$

e = residuals

Y = observed data

Measure of how much the independent variable explains the variance of the data

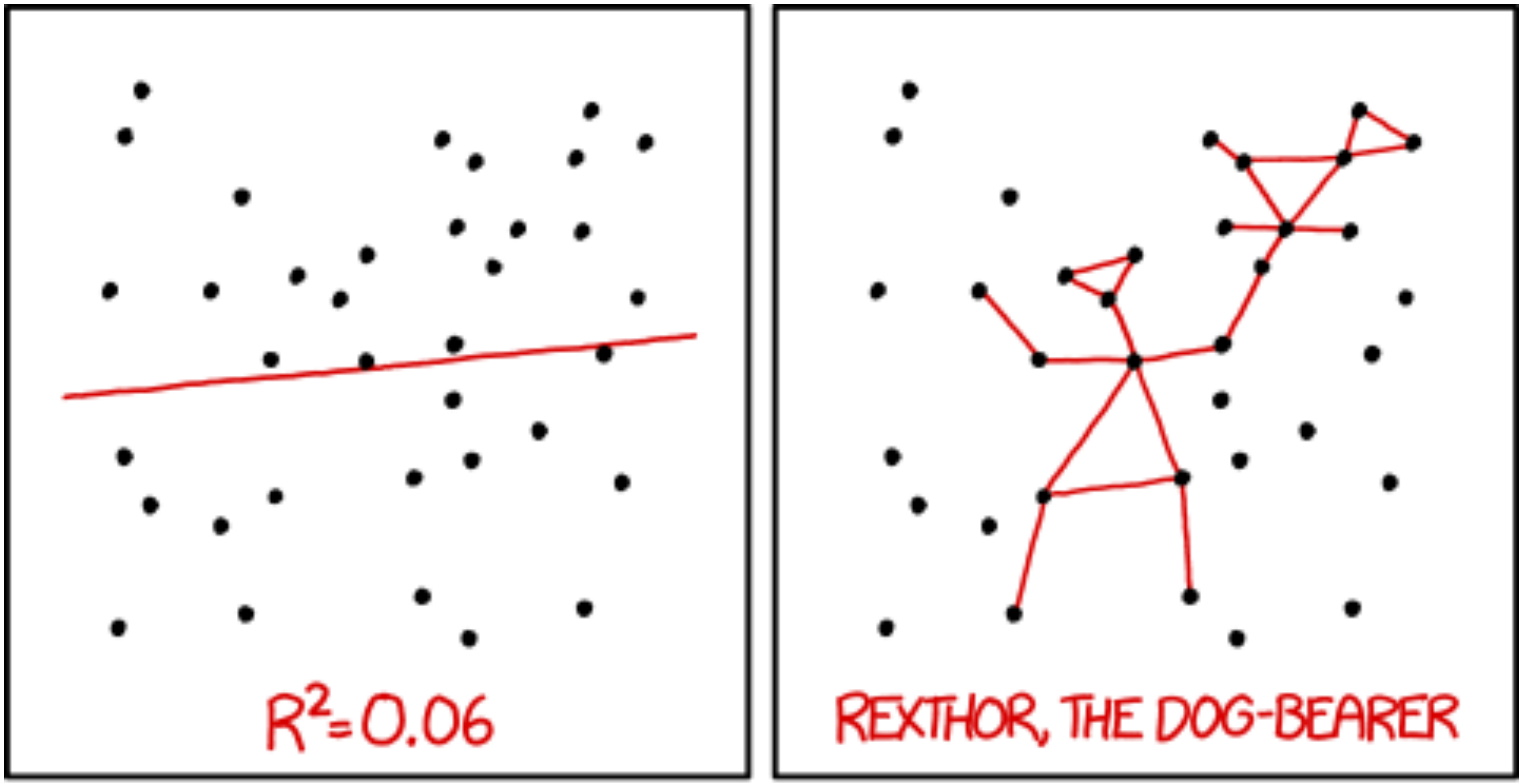
```
from sklearn.metrics import r2_score
r2_score(y, model.predict(X))
```

0.9749140085676858

Simple Regression and R^2

If only one independent variable

$R^2 = r^2$ (Pearson's Correlation squared)



I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

Multiple Linear Regression

Using multiple independent variables

Amazon’s daily revenue depends on

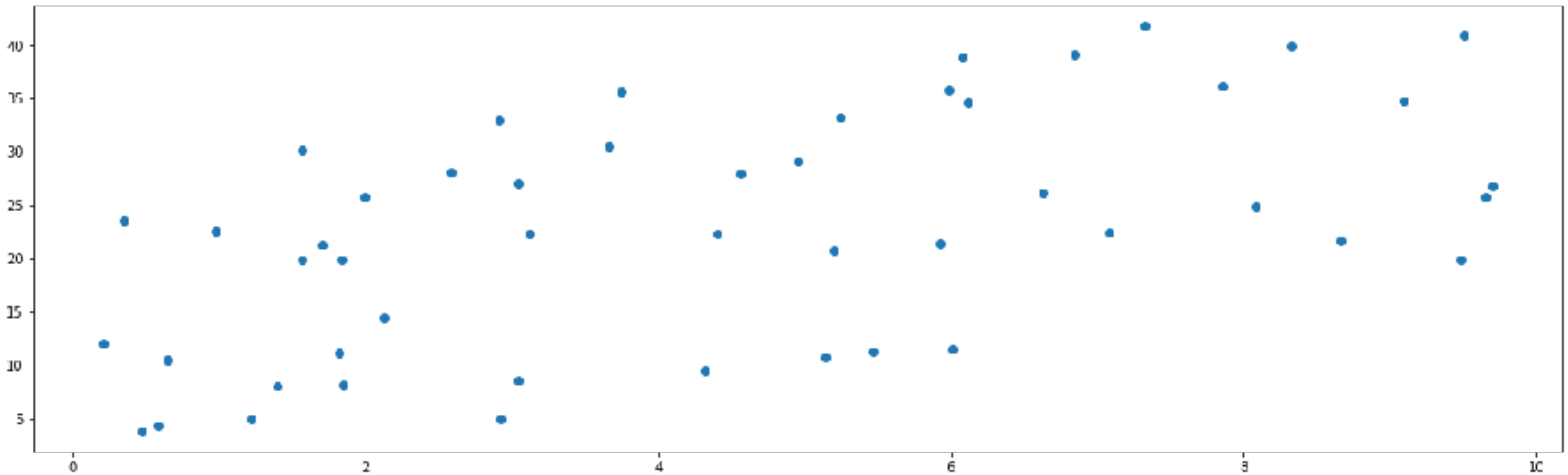
- Latency
- Price
- Steps needed to order
- Page layout
- Relevant suggestions
- Search results
- Font sizes
- Color
- Shipping costs

Two Independent Variable Example

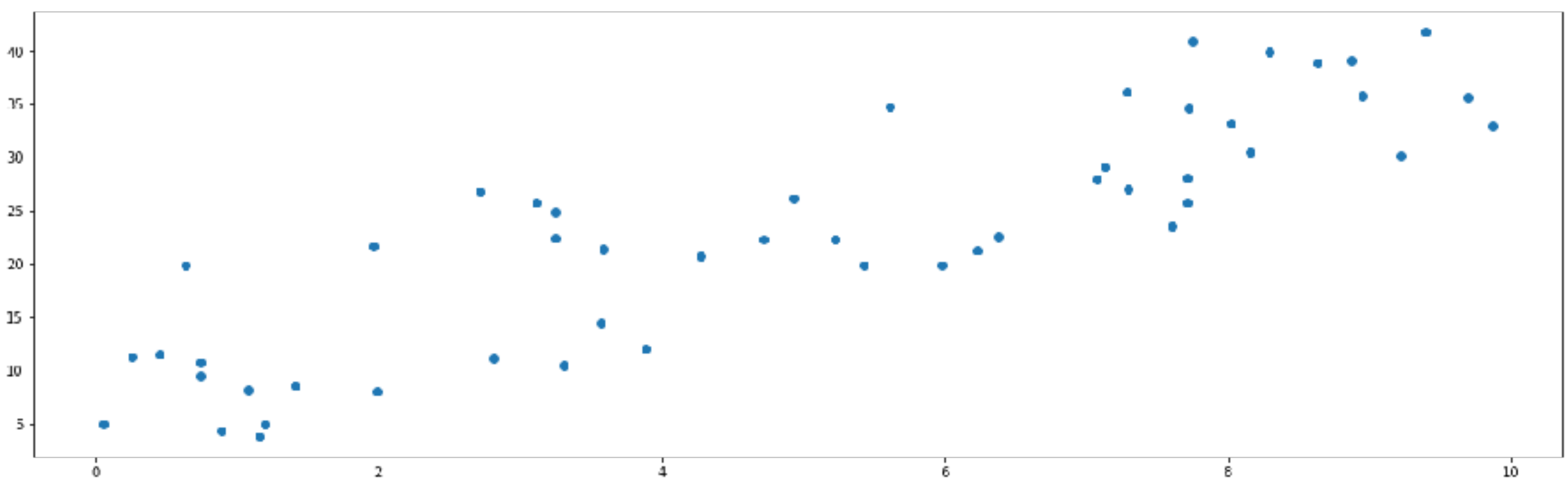
y = f(x, z) = 2*x + 3*z - 1

```
rng = np.random.RandomState(42)
x = 10 * rng.rand(50)
z = 10 * rng.rand(50)
y = 2 * x + 3 * z - 1 + rng.randn(50)   #faking data
```

plt.scatter(x, y)



plt.scatter(z, y)



Fitting the Model

```
from sklearn.linear_model import LinearRegression
model2 = LinearRegression(fit_intercept=True)
model2.fit(pd.DataFrame({'x':x,'z':z}), y)
```

model2.coef_	array([1.86706076, 2.96638451])
model.intercept_	-0.3049071881469345

Model

y = 1.86706076 * x + 2.96638451 * z - 0.3049071881469345

Actual

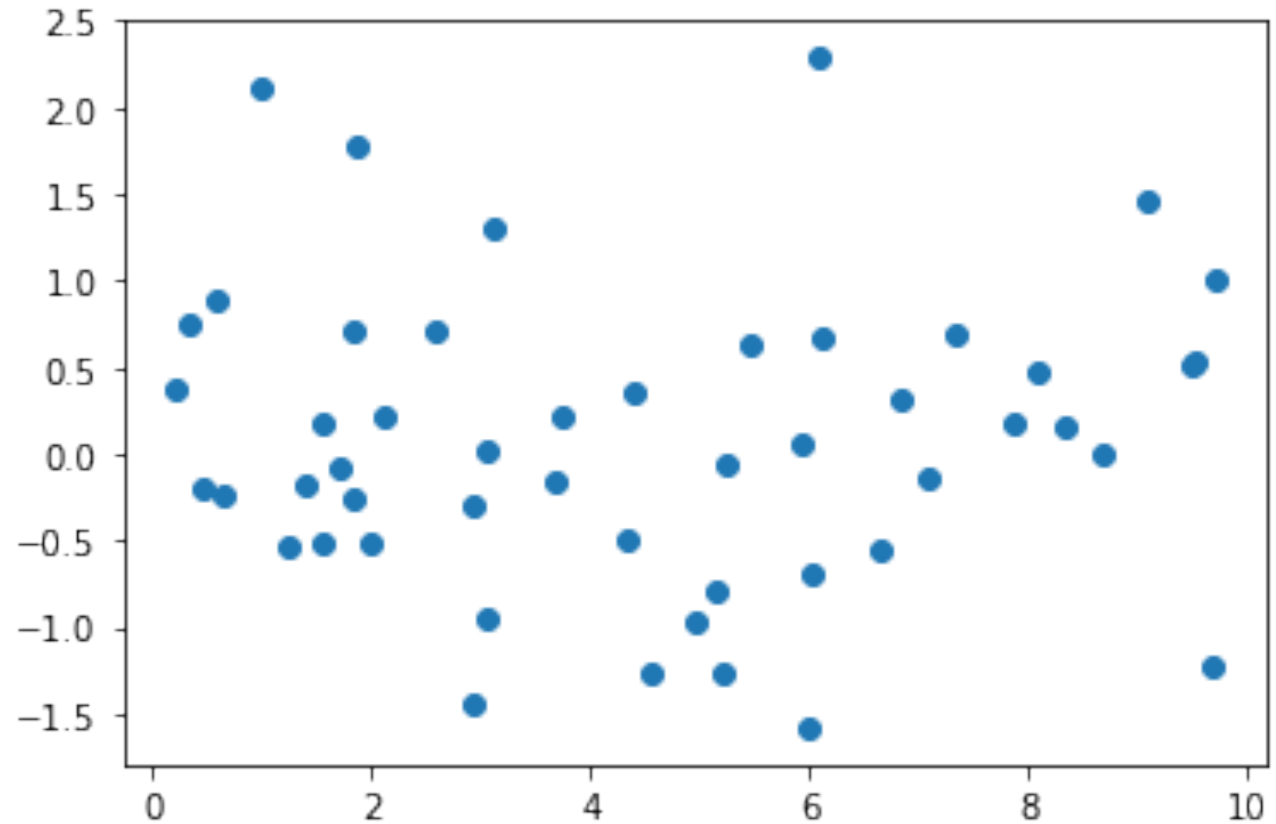
f(x, z) = 2*x + 3*z - 1

Residuals

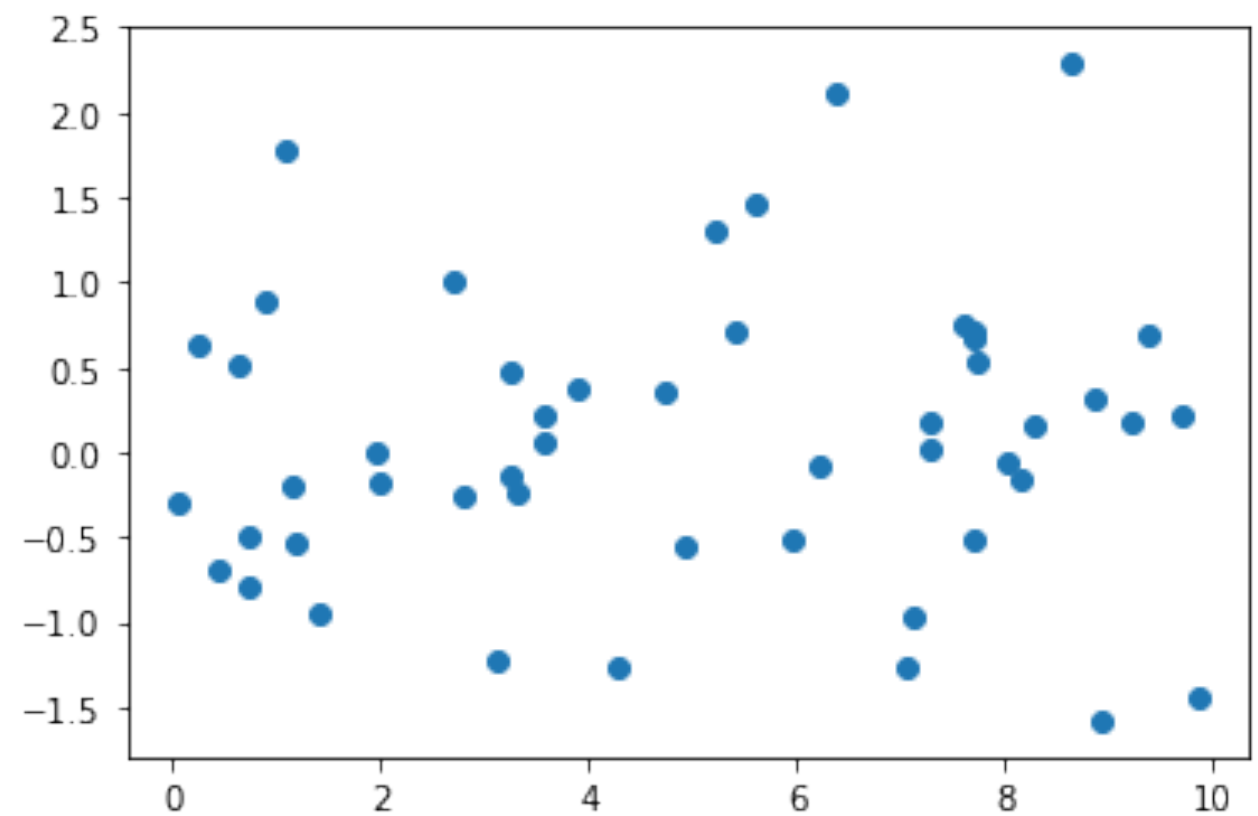
```
predicted = model2.predict(pd.DataFrame({'x':x,'z':z}))
residuals2 = y - predicted
residuals2.mean()                    # 0.0831037982077566
residuals2.sum()                     # 4.1551899103878
```

Residuals

plt.scatter(x,residuals2)



plt.scatter(z,residuals2)



Two Independent Variable Example

Model With 50 Data points

y = 1.86706076 * x + 2.96638451 * z - 0.3049071881469345

Actual

f(x, z) = 2*x + 3*z - 1

Model With 100 Data points

y = 1.96582747 * x + 3.07193114 * z - 1.0893899635499302

Model With 1000 Data points

y = 1.99662862 * x + 2.99969211 * z - 0.9689826659919589

R^2 - Coefficient of Multiple Determination

When have multiple independent variables R^2 has new name

Adding an other independent variable

Contributes to explain dependent variable

R^2 increases

Has nothing to do with dependent variable

R^2 increases