

# Lecture 2

# Simple Linear Regression

---

EE-UY 4563/EL-GY 9123: INTRODUCTION TO MACHINE LEARNING

PROF. SUNDEEP RANGAN

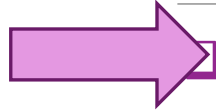
# Learning Objectives

---

- ❑ How to load data from a text file
- ❑ How to visualize data via a **scatter plot**
- ❑ Describe a **linear model** for data
  - Identify the **target variable** and **predictor**
- ❑ Compute optimal parameters for the model using the regression formula
- ❑ Fit parameters for related models by minimizing the residual sum of squares
- ❑ Compute the  $R^2$  measure of fit
- ❑ Visually determine goodness of fit and identify different causes for poor fit

# Outline

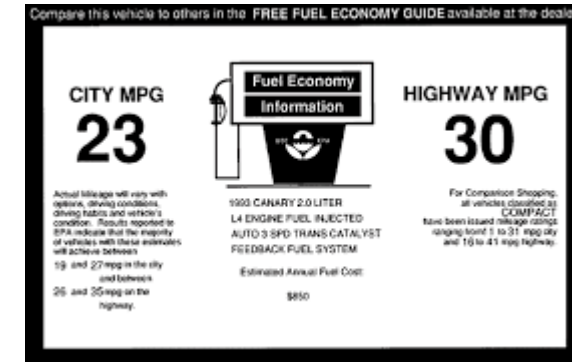
---

 Motivating Example: Predicting the mpg of a car

- ☐ Linear Model
- ☐ Least Squares Fit Problem
- ☐ Sample Mean and Variance; LS Fit Solution
- ☐ Assessing Goodness of Fit

# Example: What Determines mpg in a Car?

- ❑ What engine characteristics determine fuel efficiency?
- ❑ Why would a data scientist be hired to answer this question?
- ❑ Not to help purchasing a specific car.
  - The mpg for a currently available car is already known.
  - (If the car company isn't lying?)
- ❑ To guide building new cars.
  - Understand what is reasonably achievable before full design
- ❑ To find cars that are outside the trend.
  - Example: What cars give great mpg for the cost or size?



# Demo in Github

---

📄 [https://github.com/sdrangan/introml/blob/master/simp lin reg/auto mpg.ipynb](https://github.com/sdrangan/introml/blob/master/simp%20lin%20reg/auto%20mpg.ipynb)

## Simple Linear Regression for Automobile mpg Data

In this demo, you will see how to:

- Load data from a text file using the pandas package
- Create a scatter plot of data
- Handle missing data
- Fit a simple linear model
- Plot the linear fit with the test data
- Use a nonlinear transformation for an improved fit

## Loading the Data

The python [pandas](#) library is a powerful package for data analysis. In this course, we will use a small portion of its features -- just reading and writing data from files. After reading the data, we will convert it to numpy for all numerical processing including running machine learning algorithms.

We begin by loading the packages.

```
In [86]: import pandas as pd
import numpy as np
```







The data for this demo comes from a survey of cars to determine the relation of mpg to engine characteristics. The data can be found in the UCI library: <https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg>

You can directly read the data in the file, <https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data> We will load the data into ipython notebook, using the pandas library. Unfortunately, the file header does not include the names of the fields,

# Getting Data

□ Data from UCI dataset library: <https://archive.ics.uci.edu/ml/datasets.html>

Browser Through: 61 Data Sets Table View [List View](#)

Default Task - <a href="#">Undo</a>	Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Classification (257) <b>Regression (61)</b> Clustering (52) Other (51)	 <a href="#">3D Road Network (North Jutland, Denmark)</a>	Sequential, Text	Regression, Clustering	Real	434874	4	2013
Attribute Type Categorical (1) Numerical (52) Mixed (6)	 <a href="#">Air Quality</a>	Multivariate, Time-Series	Regression	Real	9358	15	2016
Data Type Multivariate (56) Univariate (4) Sequential (5) Time-Series (16) Text (4) Domain-Theory (3) Other (0)	 <a href="#">Airfoil Self-Noise</a>	Multivariate	Regression	Real	1503	6	2014
Area Life Sciences (9) Physical Sciences (8) CS / Engineering (25) Social Sciences (7)	 <a href="#">Amazon Access Samples</a>	Time-Series, Domain-Theory	Regression, Clustering, Causal-Discovery		30000	20000	2011
	 <a href="#">Auto MPG</a>	Multivariate	Regression	Categorical, Real	398	8	1993
	 <a href="#">Auto MPG</a>			Categorical			

← Today's lecture

# Loading the Data in Jupyter Notebook

## Try 1: The Wrong Way!

```
import pandas as pd
import numpy as np
```

```
In [67]: names = ['mpg', 'cylinders', 'displacement', 'horsepower',
                  'weight', 'acceleration', 'model year', 'origin', 'car name']
```

```
In [122]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data')
```

```
In [123]: df.head(6)
```

```
Out[123]:
```

	18.0	8	307.0	130.0	3504.	12.0	70	1	"chevrolet chevelle malibu"
0	15.0	8	350.0	165.0	3693.	11...			
1	18.0	8	318.0	150.0	3436.	11...			
2	16.0	8	304.0	150.0	3433.	12...			
3	17.0	8	302.0	140.0	3449.	10...			
4	15.0	8	429.0	198.0	4341.	10...			
5	14.0	8	454.0	220.0	4354.	9...			

❑ Python pandas library

- Read\_csv command.
- Read URL or file location.

❑ Creates a **dataframe** object

- <http://pandas.pydata.org/pandas-docs/stable/dsintro.html#dataframe>

❑ Problems

❑ Does not parse columns

- All data in a single column
- Read\_csv assumes columns are delimited by commas

❑ Mistakes first line as header

# Loading the Data in Jupyter

## Try 2: Fixing the Errors

```
In [125]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/' +  
                        'auto-mpg/auto-mpg.data',  
                        header=None, delim_whitespace=True, names=names, na_values='?')
```

You can display a first few lines of the dataframe by using head command:

```
In [126]: df.head(6)
```

Out[126]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18	8	307	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15	8	350	165	3693	11.5	70	1	buick skylark 320
2	18	8	318	150	3436	11.0	70	1	plymouth satellite
3	16	8	304	150	3433	12.0	70	1	amc rebel sst
4	17	8	302	140	3449	10.5	70	1	ford torino
5	15	8	429	198	4341	10.0	70	1	ford galaxie 500

- ❑ Fix the arguments in read\_csv
- ❑ Pandas routines have many options
- ❑ When you get a problem:
  - Google is your friend!
  - You are not the first to have these problems.
- ❑ Dataframe has three components
  - df.columns, df.index, df.values
- ❑ More in recitation



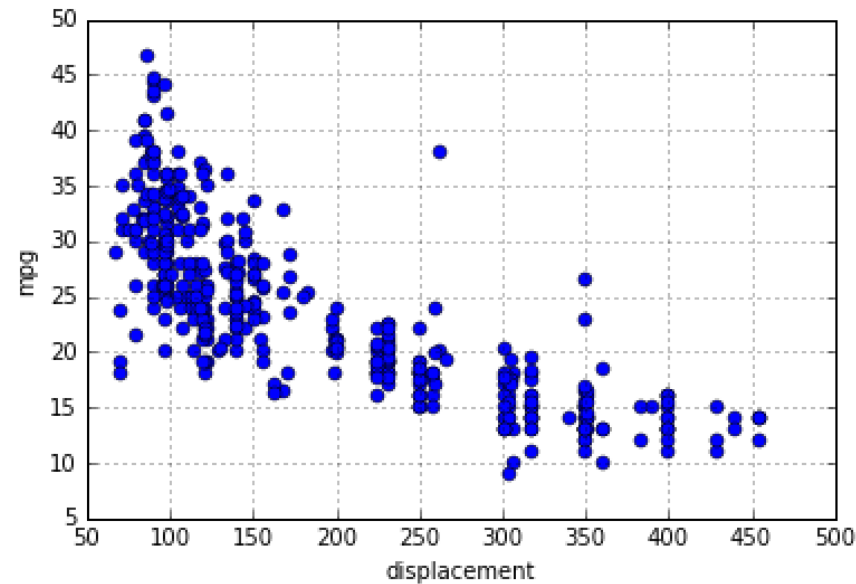
# Visualizing the Data

```
In [150]: xstr = 'displacement'  
x = np.array(df[xstr])  
y = np.array(df['mpg'])
```

```
In [146]: import matplotlib  
import matplotlib.pyplot as plt  
%matplotlib inline
```

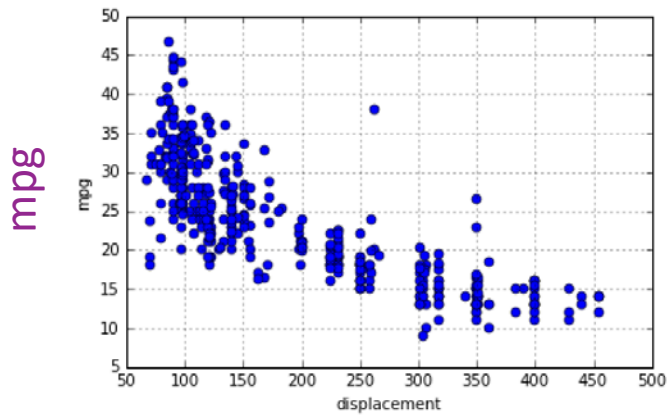
```
In [151]: plt.plot(x,y,'o')  
plt.xlabel(xstr)  
plt.ylabel('mpg')  
plt.grid(True)
```

- ❑ When possible, look at data before doing anything
- ❑ Python has MATLAB-like plotting
  - Matplotlib module

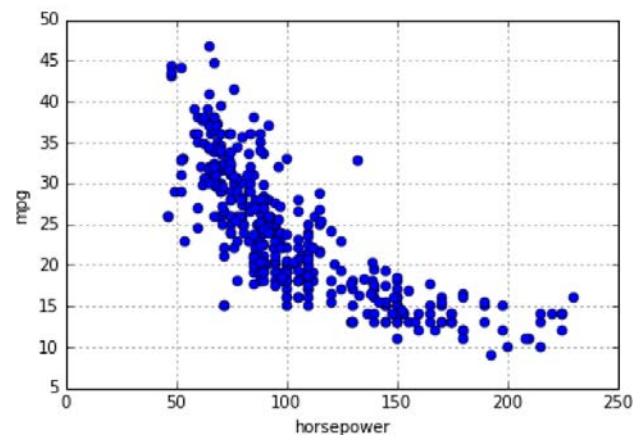


# Postulating a Model

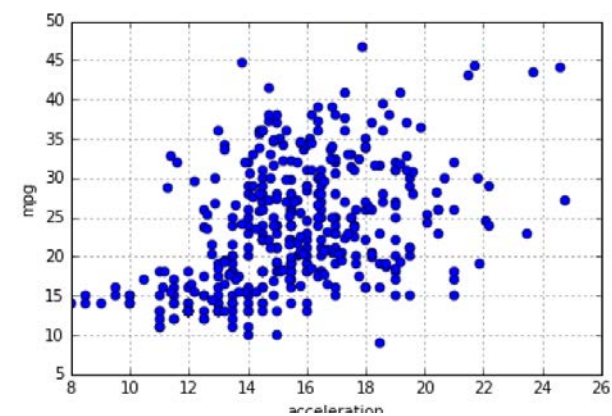
- ❑ What relationships do you see?
- ❑ Is there a mathematical model relating the variables?
- ❑ How well can you predict mpg from these variables?



Displacement



Horsepower



Acceleration

# Outline

---

- ❑ Motivating Example: Predicting the mpg of a car

- ❑ Linear Model

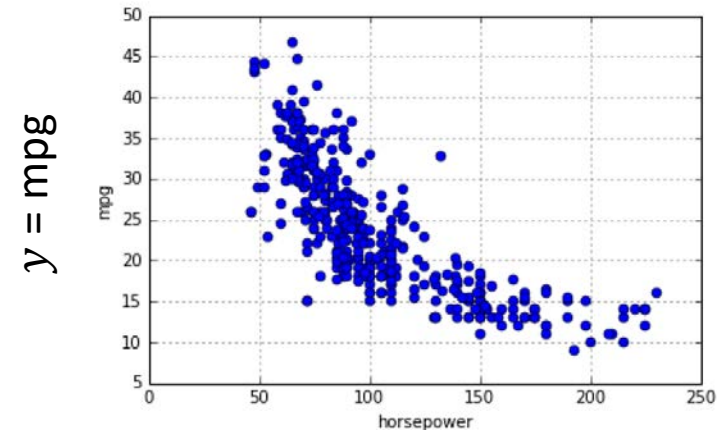
- ❑ Least Squares Fit Problem

- ❑ Sample Mean and Variance; LS Fit Solution

- ❑ Assessing Goodness of Fit

# Data

- $y$  = variable you are trying to predict.
  - Called many names: Dependent variable, response variable, target, regressand, ...
- $x$  = what you are using to predict:
  - Predictor, attribute, covariate, regressor, ...
- Data: Set of points,  $(x_i, y_i), i = 1, \dots, n$ 
  - Each data point is called a sample.
- Scatter plot



$x$  = horsepower

# Linear Model

- Assume a linear relation

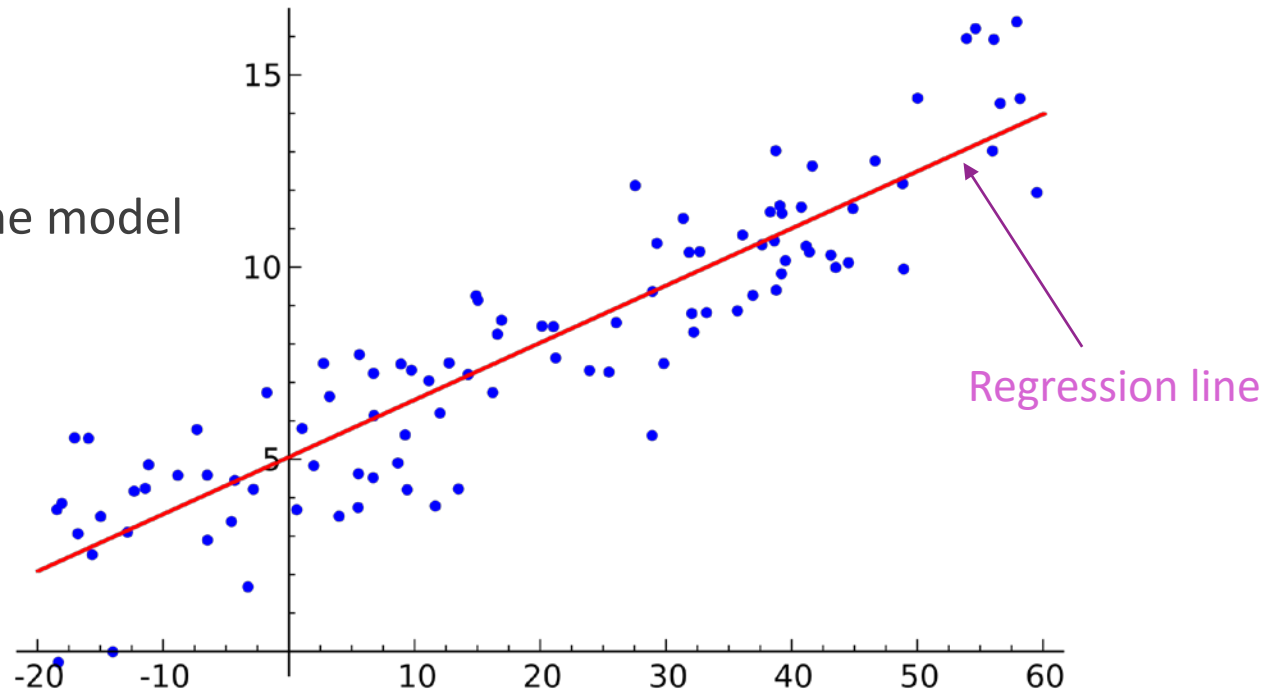
$$y \approx \beta_0 + \beta_1 x$$

- $\beta_0$  = intercept
- $\beta_1$  = slope

- $\beta = (\beta_0, \beta_1)$  are the **parameters** of the model

- What are the units of  $\beta_0, \beta_1$ ?

- When is this model good?



# Why Use a Linear Model?

---

- ❑ Many natural phenomena have linear relationship
- ❑ Predictor has small variation
  - Suppose  $y = f(x)$
  - If variation of  $x$  is small around some value  $x_0$ , then

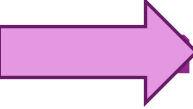
$$y \approx f(x_0) + f'(x_0)(x - x_0) = \beta_0 + \beta_1 x,$$

$$\beta_0 = f(x_0) - f'(x_0)x_0, \quad \beta_1 = f'(x_0)$$

- ❑ Gaussian random variables
- ❑ Simple to compute
- ❑ Easy to interpret relation

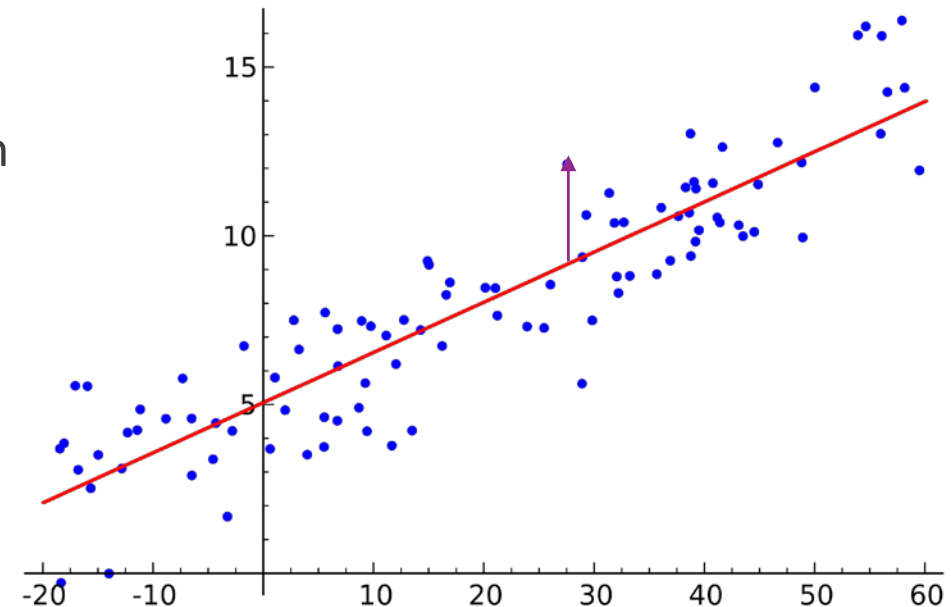
# Outline

---

- ❑ Motivating Example: Predicting the mpg of a car
- ❑ Linear Model
- ❑ Least Squares Fit Problem
- ❑ Sample Mean and Variance; LS Fit Solution
- ❑ Assessing Goodness of Fit

# Linear Model Residual

- Knowing  $x$  does not exactly predict  $y$
- Add a **residual** term
$$y = \beta_0 + \beta_1 x + \epsilon$$
- Residual = component the model does not explain
  - Predicted value:  $\hat{y}_i = \beta_1 x_i + \beta_0$
  - Residual:  $\epsilon_i = y_i - \hat{y}_i$
- Vertical deviation from the regression line





# Least Squares Model Fitting

---

❑ How do we select parameters  $\beta = (\beta_0, \beta_1)$ ?

❑ Define  $\hat{y}_i = \beta_1 x_i + \beta_0$

- Predicted value on sample  $i$  for parameters  $\beta = (\beta_0, \beta_1)$

❑ Define average **residual sum of squares**:

$$\text{RSS}(\beta_0, \beta_1) := \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- Note that  $\hat{y}_i$  is implicitly a function of  $\beta = (\beta_0, \beta_1)$
- Also called the sum of **squared residuals** (SSR) and **sum of squared errors** (SSE)

❑ **Least squares solution**: Find  $(\beta_0, \beta_1)$  to minimize RSS.

- Geometrically, minimizes squared distances of samples to regression line

# Finding Parameters via Optimization





## A general ML recipe

---

### General ML problem

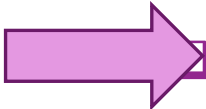
- Find a **model** with **parameters**
- Get data
- Pick a **loss function**
  - Measures goodness of fit model to data
  - Function of the parameters

### Simple linear regression

- Find a **model** with **parameters**  Linear model:  $\hat{y} = \beta_0 + \beta_1 x$
- Get data  Data:  $(x_i, y_i), i = 1, 2, \dots, N$
- Pick a **loss function**  Loss function:  
$$RSS(\beta_0, \beta_1) := \sum (y_i - \beta_0 + \beta_1 x_i)^2$$
- Find parameters that **minimizes** loss  Select  $\beta_0, \beta_1$  to minimize  $RSS(\beta_0, \beta_1)$

# Outline

---

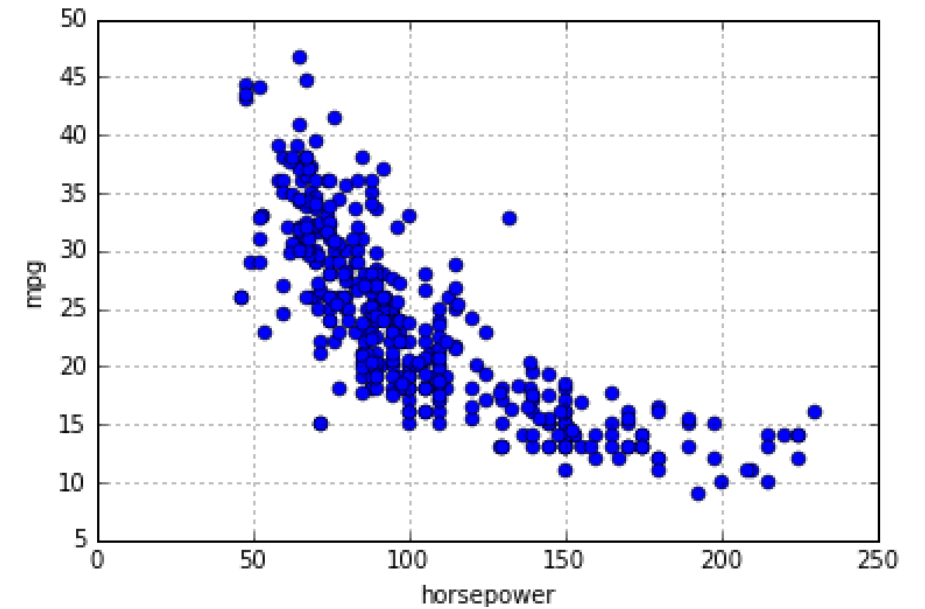
- ❑ Motivating Example: Predicting the mpg of a car
- ❑ Linear Model
- ❑ Least Squares Fit Problem
-  ❑ Sample Mean and Variance; LS Fit Solution
- ❑ Assessing Goodness of Fit

# Visualizing Mean and SD on Scatter Plot

## Question

Using the picture only (no calculators), estimate the following (roughly):

- ☐ The sample mean mpg and horsepower:  $\bar{x}$ ,  $\bar{y}$
- ☐ The sample std deviations:  $s_x$ ,  $s_y$



# Sample Mean and Standard Deviations

## □ Sample mean

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$$

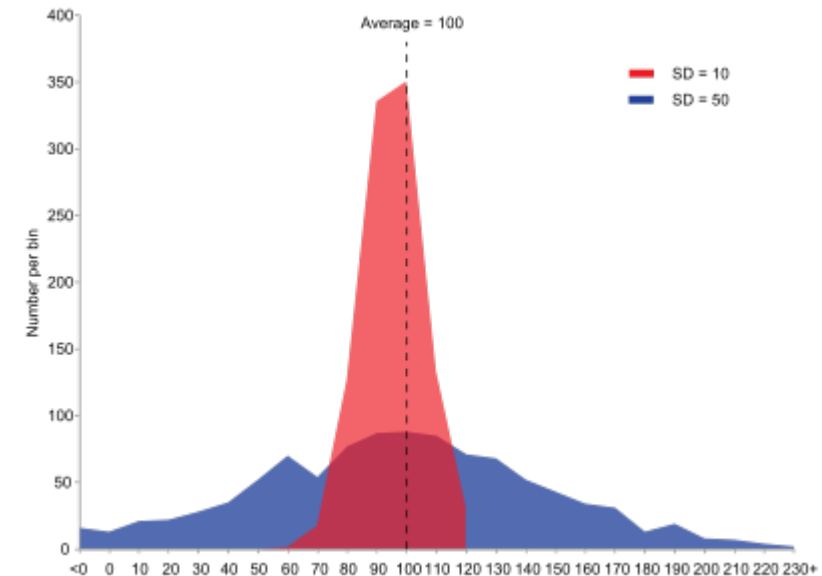
## □ Sample variances

$$s_x^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2, \quad s_y^2 = \frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2$$

- Some formulae have a  $N - 1$  on denominator
- Creates an unbiased estimate. More on this later.

## □ Sample standard deviation

- $s_x, s_y$
- Square root of variances



Visualizing standard deviation

[https://en.wikipedia.org/wiki/Standard\\_deviation](https://en.wikipedia.org/wiki/Standard_deviation)

# Visualizing Mean and SD on Scatter Plot

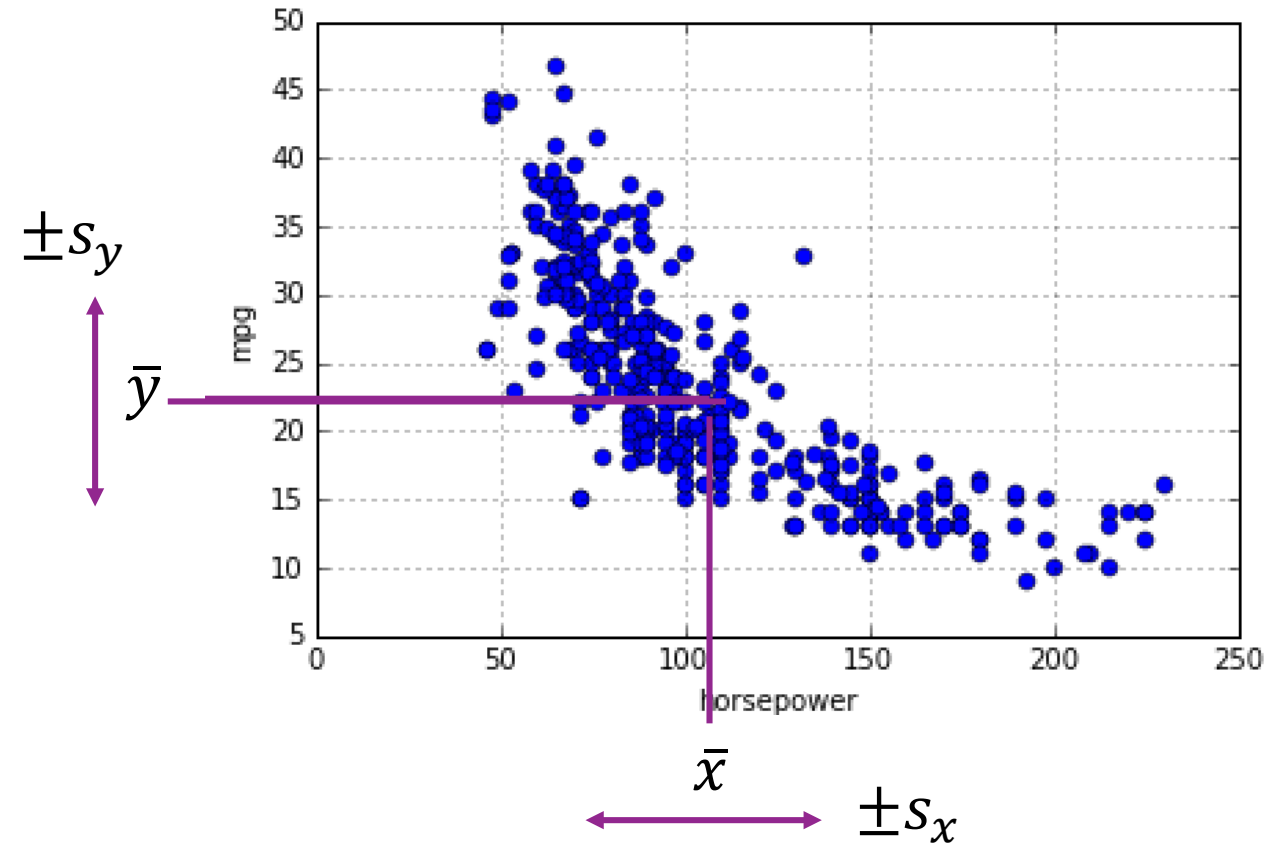
## Approximate answer

### Means: $\bar{x}$ and $\bar{y}$

- Weighted center of the points in each axis

### Standard deviations: $s_x$ and $s_y$

- Represents “variation” in each axis from mean
- With Gaussian distributions:  
1% of points are 3 SDs from mean

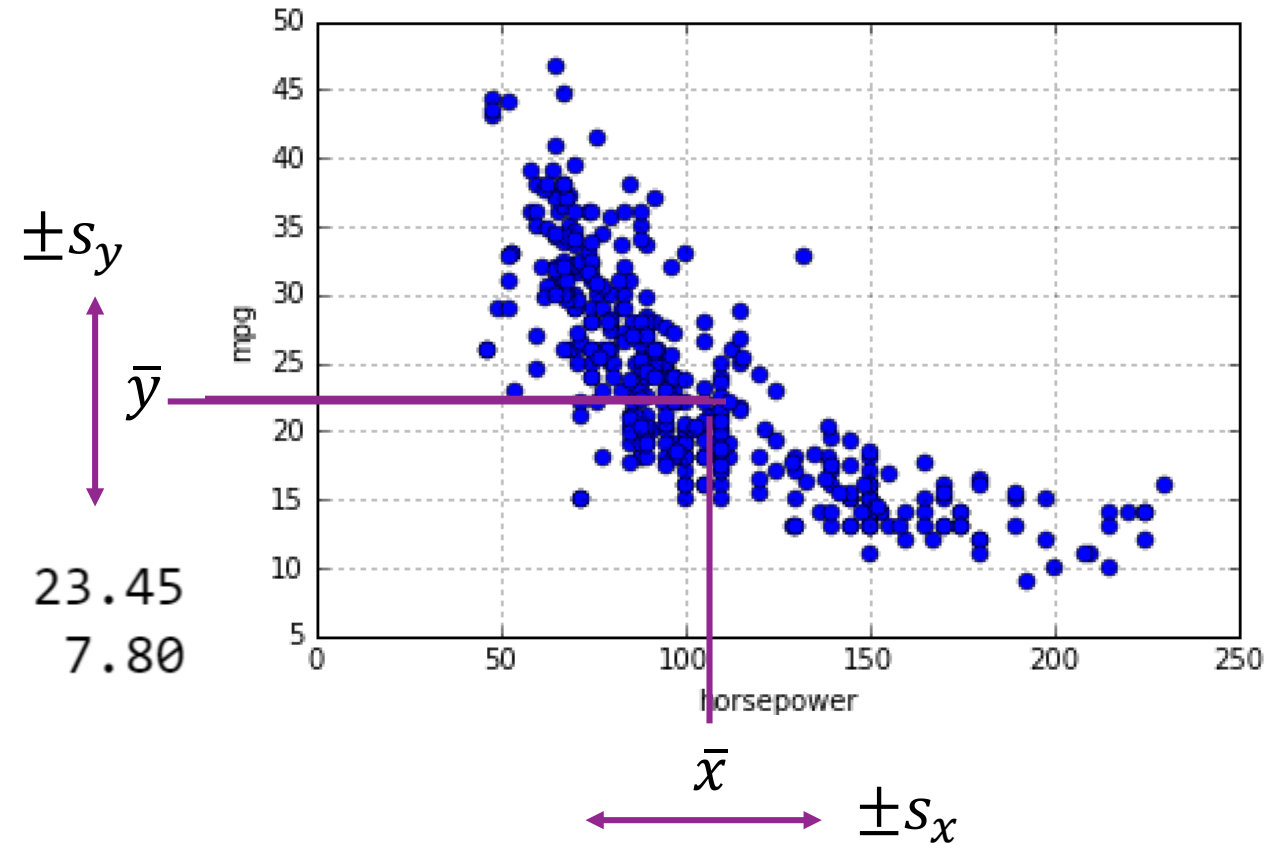


# Computing Means and SD in Python

□ Exact answer can be computed in python

```
xm = np.mean(x)
ym = np.mean(y)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
sxx = np.mean((x-xm)**2)
beta1 = syx/sxx
beta0 = ym - beta1*xm
```

xbar = 104.47, ybar = 23.45  
sqrt(sxx) = 38.44, sqrt(syy) = 7.80



# Sample Covariance

---

□ Sample covariance:

$$s_{xy} = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

□ Will interpret this momentarily

□ Cauchy-Schwarz Law:  $|s_{xy}| < s_x s_y$

□ Sample correlation coefficient

$$r_{xy} = \frac{s_{xy}}{s_x s_y} \in [-1, 1]$$



# Alternate Equation for Variance

---

□ Recall sample variance:  $s_x^2 = \frac{1}{N} \sum (x_i - \bar{x})^2$

◦ Alternate formula:

$$s_x^2 = \frac{1}{N} \sum x_i^2 - \left( \frac{1}{N} \sum x_i \right)^2 = \frac{1}{N} \sum x_i^2 - \bar{x}^2$$

□ Similarly, for covariance:

$$s_{xy} = \frac{1}{N} \sum x_i y_i - \bar{x} \bar{y}$$

# Notation

- ❑ This class will use the following notation
- ❑ We will try to be consistent
- ❑ Note: Other texts use different notations

Statistic	Notation	Formula	Python
Sample mean	$\bar{x}$	$\frac{1}{n} \sum_{i=1}^n x_i$	xm
Sample variance	$s_x^2 = s_{xx}$	$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$	sxx
Sample standard deviation	$s_x = \sqrt{s_{xx}}$	$s_x = \sqrt{s_{xx}}$	sx
Sample covariance	$s_{xy}$	$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$	sxy

# Minimizing RSS

---

- To minimize  $RSS(\beta_0, \beta_1)$  take partial derivatives:

$$\frac{\partial RSS}{\partial \beta_0} = 0, \quad \frac{\partial RSS}{\partial \beta_1} = 0$$

- Taking derivatives we get two conditions (proof on board):

$$\sum_{i=1}^N \epsilon_i = 0, \quad \sum_{i=1}^N x_i \epsilon_i = 0 \quad \text{where } \epsilon_i = y_i - \beta_0 - \beta_1 x_i$$

- Regression equation:

- After some manipulation, (proof on board), solution to optimal slope and intercept:

$$\beta_1 = \frac{s_{xy}}{s_x^2} = \frac{r_{xy}s_y}{s_x}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

# Simple Example

From:

<http://stattrek.com/regression/regression-example.aspx?Tutorial=AP>

- Very nice simple problems

Predict aptitude on one test from an earlier test

Draw a scatter plot and regression line

## How to Find the Regression Equation

In the table below, the  $x_i$  column shows scores on the aptitude test. Similarly, the  $y_i$  column shows statistics grades. The last two rows show sums and mean scores that we will use to conduct the regression analysis.

	Student	$x_i$	$y_i$	$(x_i - \bar{x})$	$(y_i - \bar{y})$	$(x_i - \bar{x})^2$	$(y_i - \bar{y})^2$	$(x_i - \bar{x})(y_i - \bar{y})$
	1	95	85	17	8	289	64	136
	2	85	95	7	18	49	324	126
	3	80	70	2	-7	4	49	-14
	4	70	65	-8	-12	64	144	96
	5	60	70	-18	-7	324	49	126
<b>Sum</b>		390	385			730	630	470
<b>Mean</b>		78	77					

The regression equation is a linear equation of the form:  $\hat{y} = b_0 + b_1x$ . To conduct a regression analysis, we need to solve for  $b_0$  and  $b_1$ . Computations are shown below.

$$b_1 = \Sigma [(x_i - \bar{x})(y_i - \bar{y})] / \Sigma [(x_i - \bar{x})^2]$$

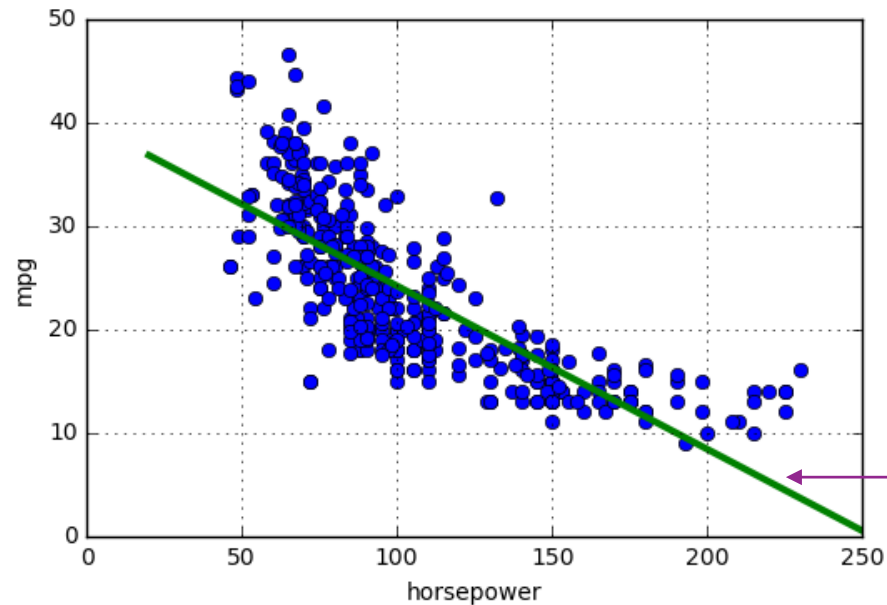
$$b_1 = 470/730 = 0.644$$

$$b_0 = \bar{y} - b_1 * \bar{x}$$

$$b_0 = 77 - (0.644)(78) = 26.768$$

# Auto Example

Python code



```
xm = np.mean(x)
ym = np.mean(y)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
sxx = np.mean((x-xm)**2)
beta1 = syx/sxx
beta0 = ym - beta1*xm
```

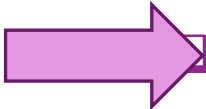
beta0= 39.94, beta1= -0.16

Regression line:

$$\text{mpg} = \beta_0 + \beta_1 \text{ horsepower}$$

# Outline

---

- ❑ Motivating Example: Predicting the mpg of a car
- ❑ Linear Model
- ❑ Least Squares Fit Problem
- ❑ Sample Mean and Variance; LS Fit Solution
-  Assessing Goodness of Fit

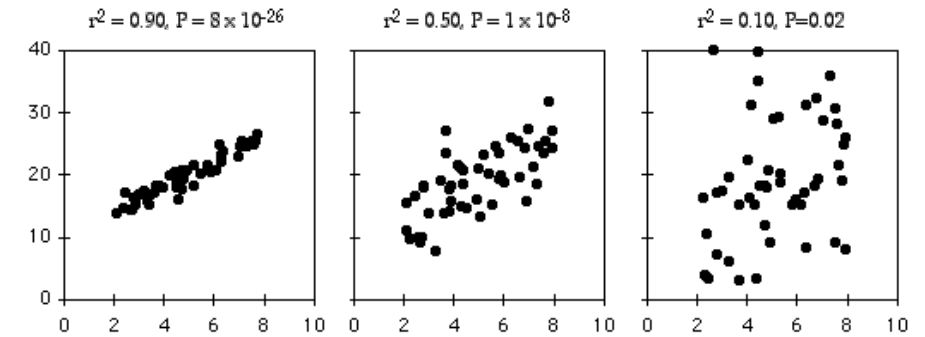
# Mimimum RSS

## □ Minimum RSS (Proof on board)

$$\min_{\beta_0, \beta_1} \text{RSS}(\beta_0, \beta_1) = N(1 - r_{xy}^2)s_y^2$$

## □ Coefficient of Determination: $R^2 = r_{xy}^2$

- Explains portion of variance in  $y$  explained by  $x$
- $s_y^2$ =variance in target  $y$
- $(1 - R^2)s_y^2$ =residual sum of squares after accounting for  $x$

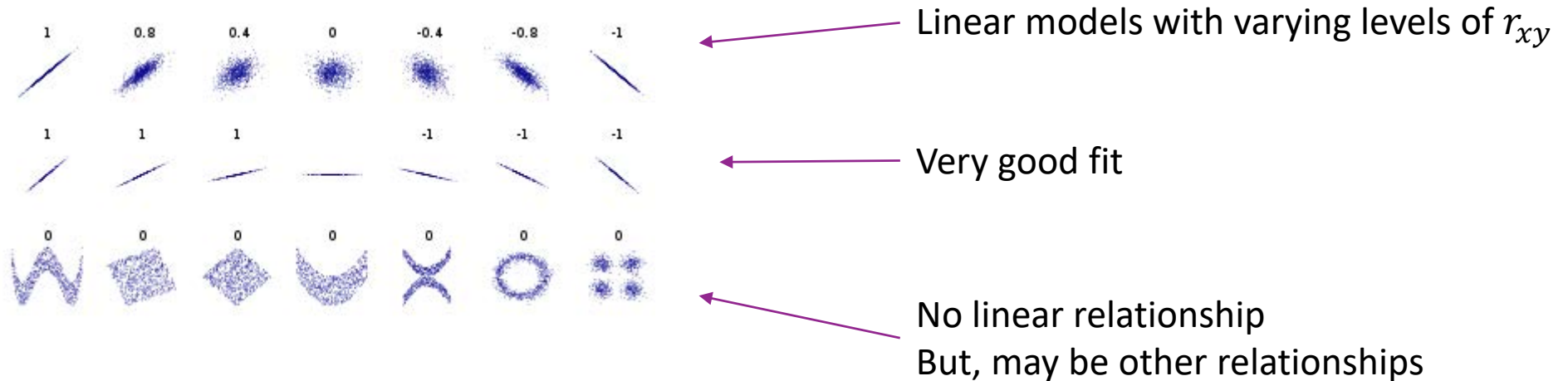


# Visually seeing correlation

□  $R^2 = r_{xy}^2 \approx 1$ : Linear model is a very good fit

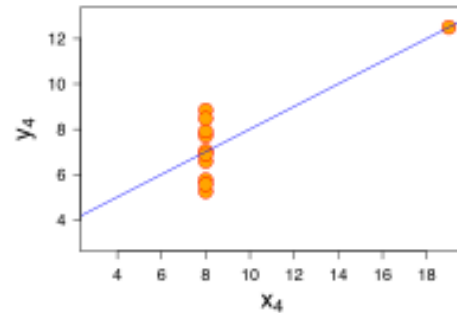
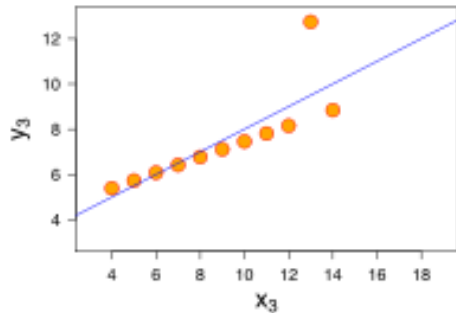
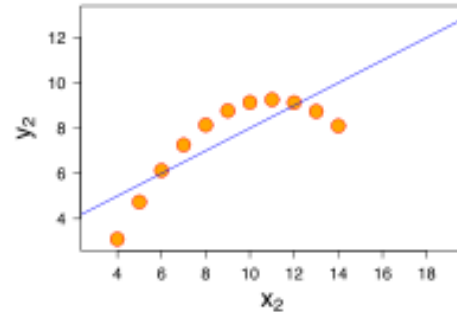
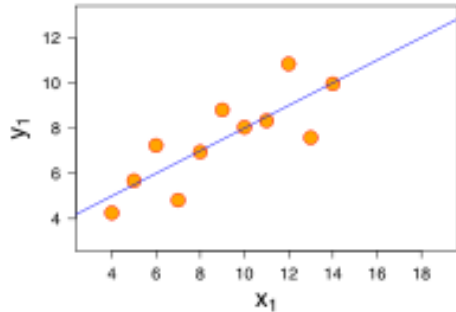
□  $R^2 = r_{xy}^2 \approx 0$ : Linear model is a poor fit.

□  $\beta_1 = \frac{r_{xy}s_y}{s_x} \Rightarrow \text{Sign}(\beta_1) = \text{Sign}(r_{xy})$





# When the Error is Large...



- ❑ Many sources of error for a linear model
- ❑ Always good to visual inspect the scatter plot
  - Look for trends
- ❑ Example to the left
  - All four data sets have same regression line
  - But, errors and their reasons are different
- ❑ How would you describe these errors?

# A Better Model for the Auto Example

- Fit the inverse:  $\frac{1}{\text{mpg}} = \beta_0 + \beta_1 \text{horsepower}$
- Uses a nonlinear transformation
- Will cover this idea later

