Exploratory Data Analysis and Regression

Introduction to Data Science

Kigali, Rwanda

July 8th, 2019

What is this course?

This is a 5 day short workshop on the ***practical*** fundamentals of data science using python.

This workshop will cover a variety of models solving a range of different data science tasks; it will familiarize participants with useful `python` libraries for data science.

What is the structure of each day?

Each session of the workshop is 4 hours. Roughly half of which will be lecture and half will be hands-on exercises.

You will work in groups during class to complete coding exercises solving problems involving ***real*** data.

What is expected of the participants?

This workshop involves a great deal of programming and requires a intuitive but sound understanding of the mathematical or statistical theory behind machine learning models.

To be successful, participants need to have some previous experience with programming as well as basic statistics; participants need to be committed to gaining a working knowledge of `python` during the workshop.

How is the workshop graded?

Participants are recommended to complete all in-class exercises as homework.

There will be an in-class, 4-hour, open-book exam on Day 5. The exam will require participants to solve a data science task using a real-data set and will be based on the skills demonstrated in the in-class exercises.

More information about the course

All materials and information can be found at the course website:

```
https://github.com/onefishy/rwanda_workshop
```

Introduction to Data Science

Outline

1. What Is Data?
2. What is Data Science?
3. Exploring Data
    - Descriptive Statistics
    - Data Visualization
4. Data Science Tools

What is Data?

What is Data?

*"A **datum** is a single measurement of something on a scale that is understandable to both the recorder and the reader. **Data** is multiple such measurements."*

Claim: everything is (can be) data!

What is Data Science

What Does it Mean to Do Data Science?

The "data science" process.

1. Find a problem you want to solve
2. Collection data you think will help solve it
3. Exploration Data Analysis:
    - examine your data for patterns and trends.
4. Build mathematical models to describe trends and patterns
5. Analyze your model
    - What does it say about your data?
    - How do your finding help solve your problem?
6. Visualization and Presentation of Results

Note: This process is not linear!

Where Does Data Come From?

Internal sources: already collected by or is part of the overall data collection of you organization.

Existing External Sources: available in ready to read format from an outside source for free or for a fee.

External Sources Requiring Collection Efforts: available from external source but acquisition requires special processing.

What Does Data Look Like?

Simple or atomic data:

- **Numeric:** integers, floats
- **Boolean:** binary or true false values
- **Strings:** sequence of symbols

What Does Data Look Like?

Compound, composed of a bunch of atomic types:

- **Date and time:** compound value with a specific structure
- **Lists:** a list is a sequence of values
- **Dictionaries:** collections of key-value pairs, a pair of values `x : y` , where x is usually a string called key representing the "name" of the value, and y is a value of any type.

  Example: Student record
  - First: Weiwei
  - Last: Pan
  - Nationality: USA

How is your data represented and stored?

Data formats:

- **Tabular Data:** a dataset that is a two-dimensional table, where each row typically represents a single data record, and each column represents one type of measurement (csv, xlsx etc.).

- **Structured Data:** each data record is presented in a form of a, possibly complex and multi-tiered, dictionary (json, xml etc.)
- **Semistructured Data:** not all records are represented by the same set of keys or some data records are not represented using the key-value pair structure.

Tabular Data

In tabular data, each row or ***observation*** represents a set of measurements of a single object or event.

	Hight	Radius	Do I Like It?
Cylinder # 1	10	5	Yes
Cylinder # 2	3	7.5	No

Each type of measurement is an ***attribute*** of the data. The number of attributes is the ***dimension*** of the data.

Types of Attributes in Tabular Data

It's vital to distinguish between classes of attributes based on the type of values they take on.

- **Quantitative attribute:** a real valued number whose values can be ordered.

  Example: Height is a quantitative attribute

- **Categorical attribute:** a real valued or string with no inherent order among the values.

  Example: "What kind of pet you have" is a categorical attribute

Is the data any good?

- **Missing values:** how do we fill in?
- **Wrong values:** how can we detect and correct?
- **Messy format:** how can we convert structured or semistructured data into tabular data?
- **Not usable:** what if the data cannot answer the question posed?

Exploring Data

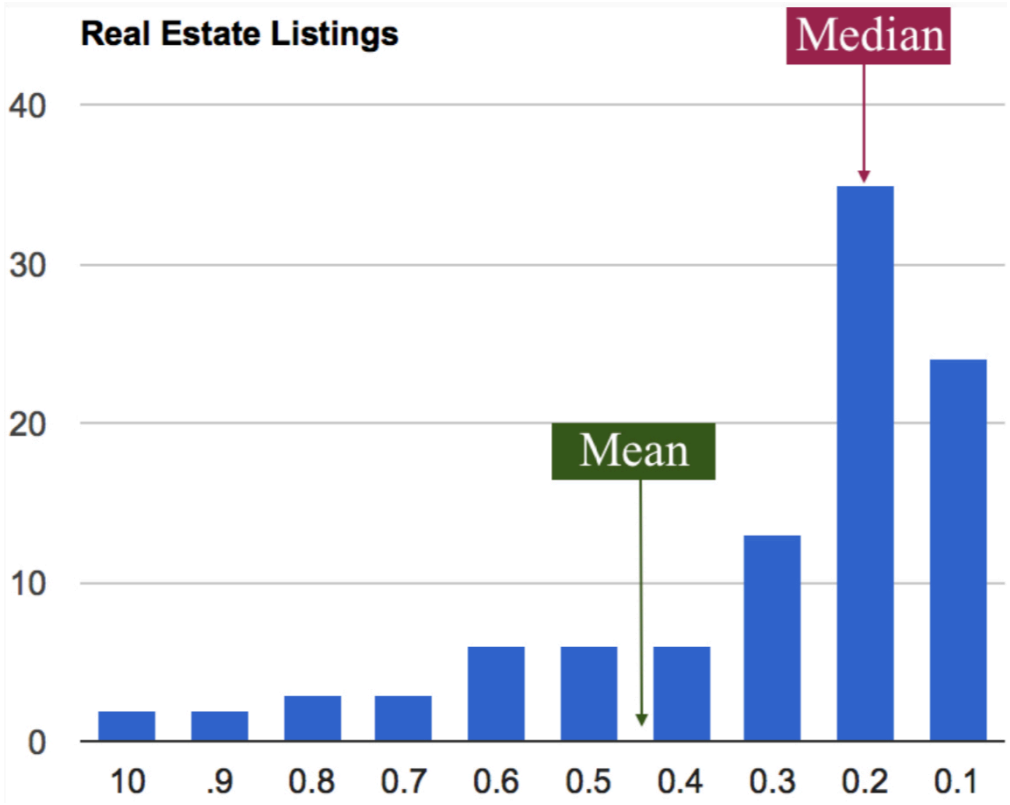
Describing Individual Attributes

Given a large dataset, we want to compute a few quantities that intuitively summarizes the data. We'd like to know:

1. what are "typical" values for our attributes?
2. how "representative" are these typical values?

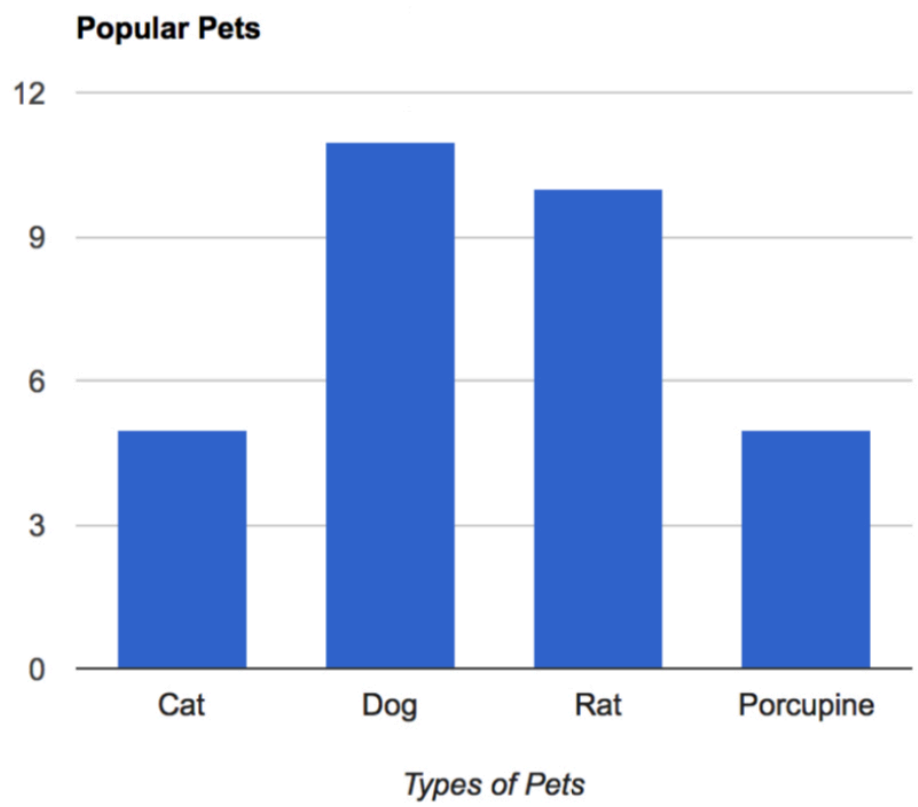
Describing A Typical Value

We can describe a typical value for nn samples of a **quantitative** attribute xx by the ***mean*** or the ***median***.



Describing A Typical Value

We can describe a typical value for nn samples of a **categorical** attribute xx by the ***mode***.



Describing the Spread of a Value

The spread of samples measures how well the mean or median describes the sample set:

1. ***Range***
2. ***Standard variance***
3. ***Standard deviation***

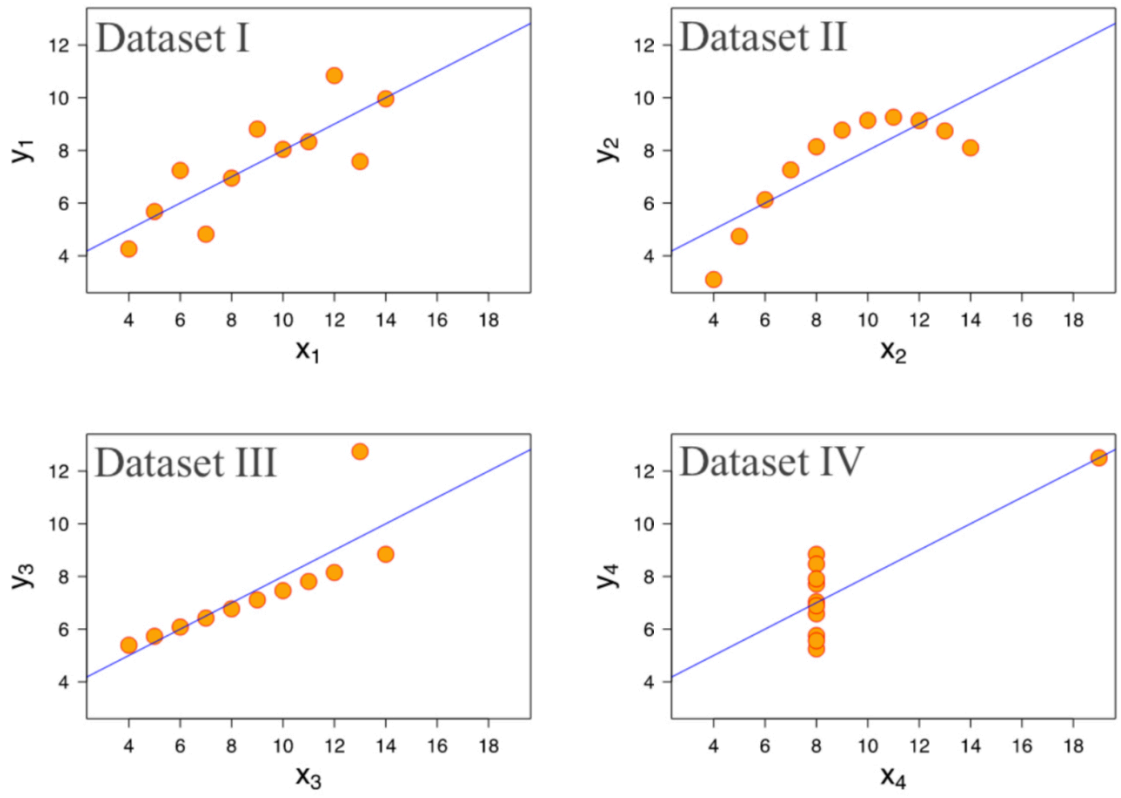
Why Data Visualization?

The following is called Anscombe's Quartet; all four sets of data have identical simple summary statistics.

	Dataset I		Dataset II		Dataset III		Dataset IV	
	x	y	x	y	x	y	x	y
	10	8.04	10	9.14	10	7.46	8	6.58
	8	6.95	8	8.14	8	6.77	8	5.76
	13	7.58	13	8.74	13	12.74	8	7.71
	9	8.81	9	8.77	9	7.11	8	8.84
	11	8.33	11	9.26	11	7.81	8	8.47
	14	9.96	14	8.1	14	8.84	8	7.04
	6	7.24	6	6.13	6	6.08	8	5.25
	4	4.26	4	3.1	4	5.39	19	12.5
	12	10.84	12	9.13	12	8.15	8	5.56
	7	4.82	7	7.26	7	6.42	8	7.91
	5	5.68	5	4.74	5	5.73	8	6.89
Sum:	99.00	82.51	99.00	82.51	99.00	82.51	99.00	82.51
Avg:	9.00	7.50	9.00	7.50	9.00	7.50	9.00	7.50
Std:	3.32	2.03	3.32	2.03	3.32	2.03	3.32	2.03

Anscombe's Quartet

The following is called Anscombe's Quartet; all four sets of data have identical simple summary statistics.



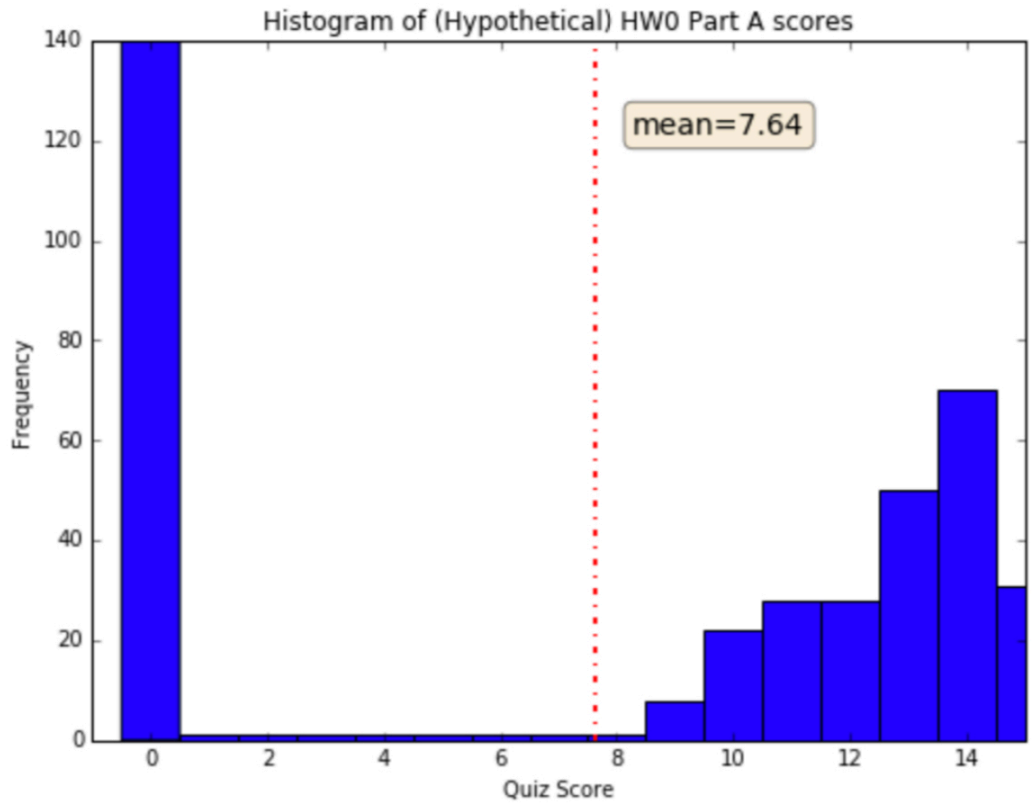
Why Data Visualization?

If I tell you that the average score for the final examis: 7.64/15.

What does that suggest?

Why Data Visualization?

If I then show you the following graph, what does it suggest?



What is Data Visualization Good For?

Analyze:

- Identify hidden patterns and trends
- Help formulate/test hypothese
- Help determine the next step in analysis/modeling

What is Data Visualization Good For?

Communicate:

- Present information and ideas succinctly
- Provide evidence and support
- Influence and persuade

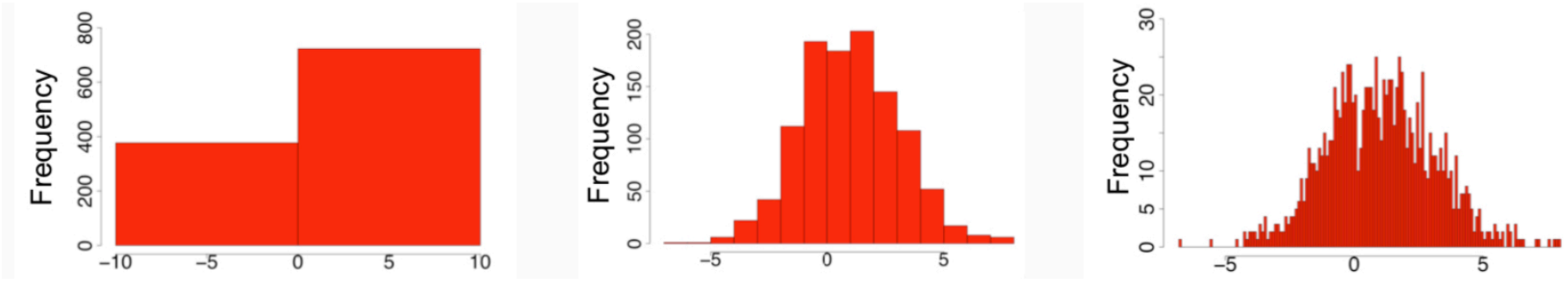
Types of Data Visualization

What do you want your visualization to show about your data?

- **Distribution:** how a variable or variables in the dataset distribute over a range of possible values.
- **Relationship:** how the values of multiple variables in the dataset relate
- **Composition:** how the dataset breaks down into subgroups
- **Comparison:** how trends in multiple variable or datasets compare

Visualizing Distributions

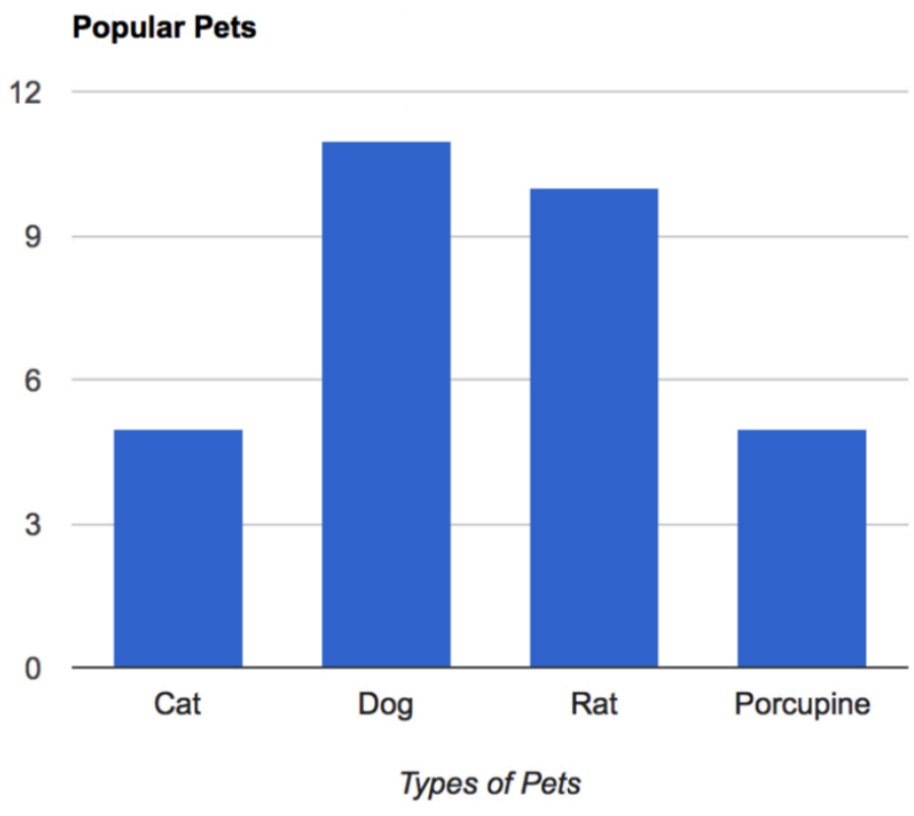
A ***histogram*** is a way to visualize how a **quantitative** attribute is distributed across certain values.



Note: Trends in histograms are sensitive to number of bins.

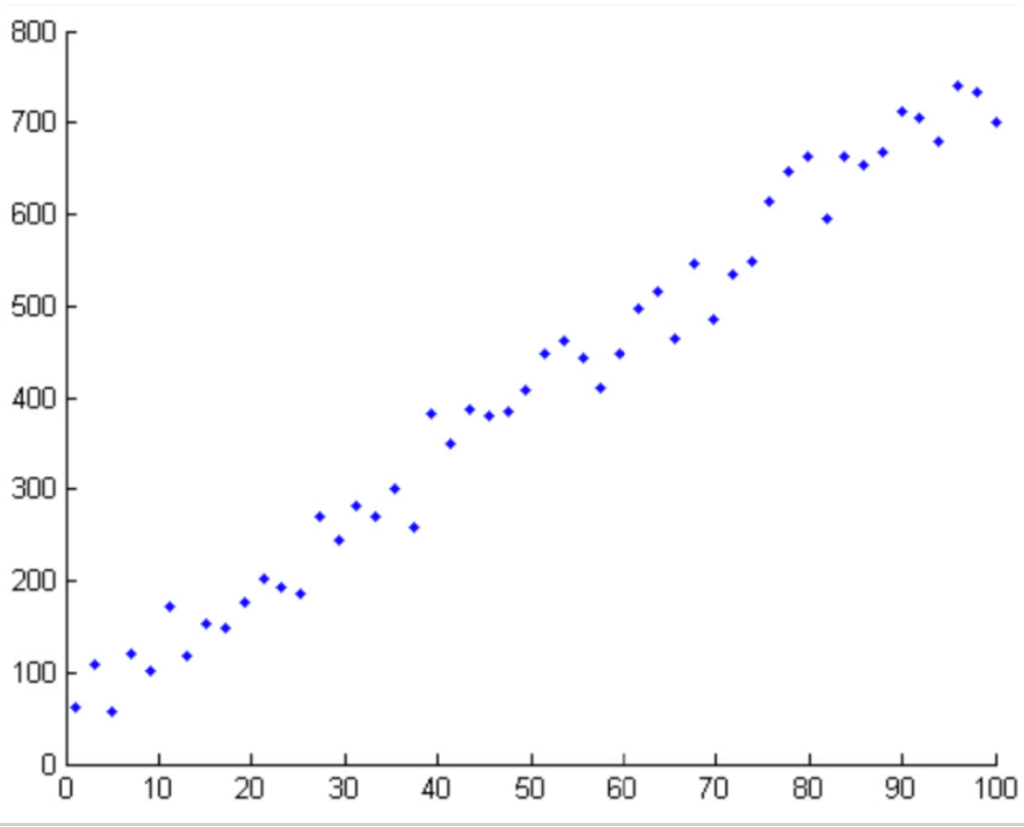
Visualizing Distributions

A ***bar chart*** is a way to visualize how a **categorical** attribute is distributed across certain values.



Visualizing Relationships

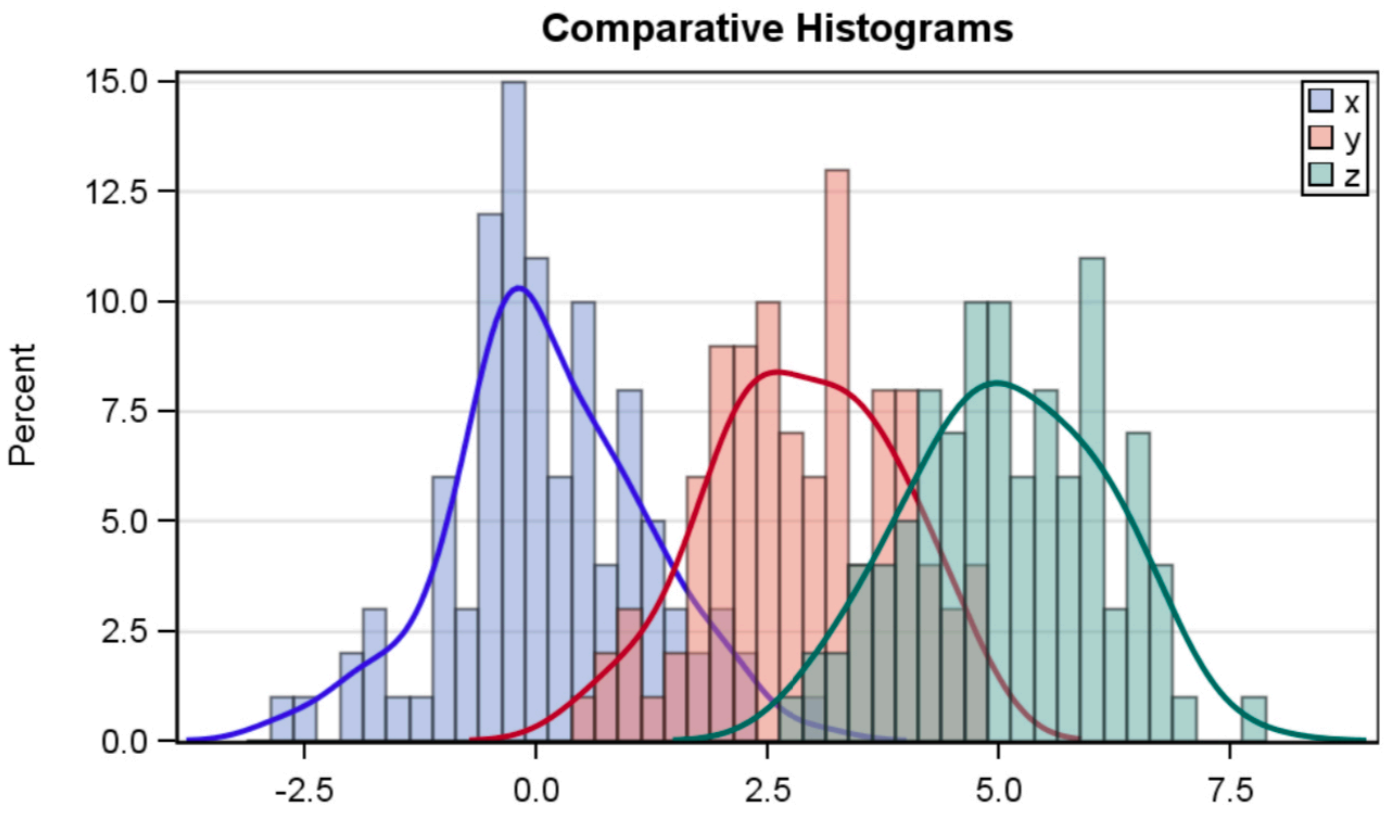
A ***scatter plot*** is a way to visualize the relationship between two different attributes.



Visualizing Comparisons of Subgroups

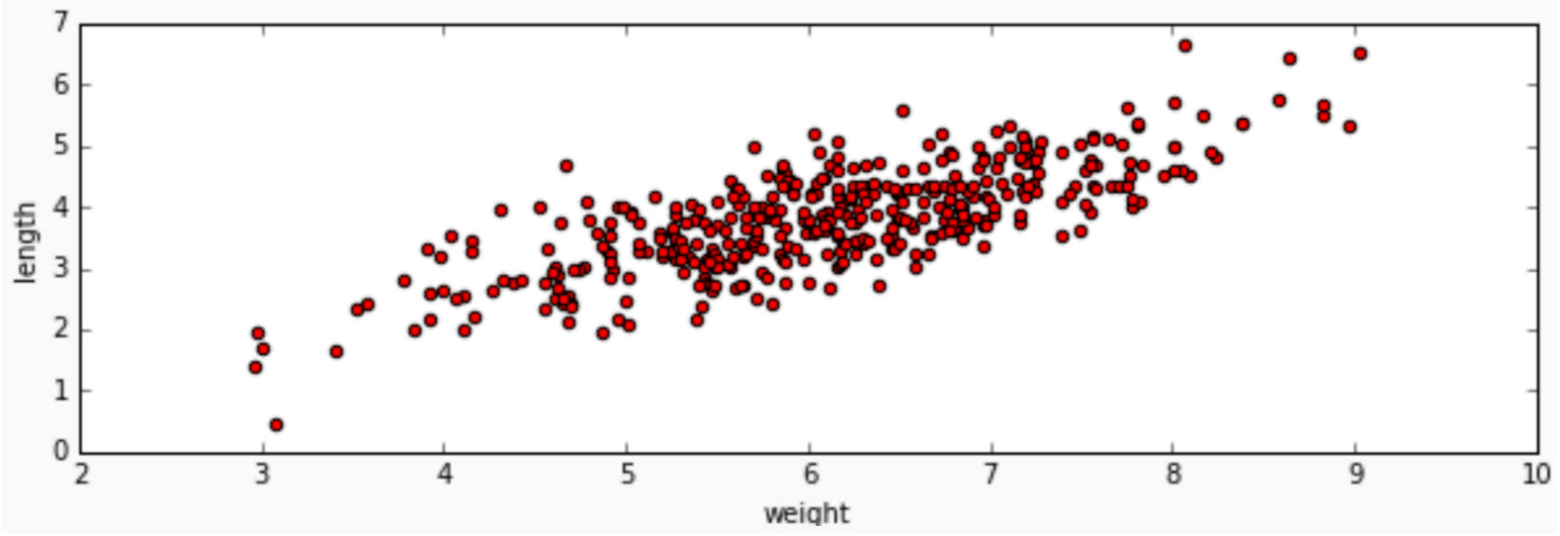
Plotting multiple histograms or curves on the same axes is a way to visualize how different subgroups of data compare.

For example: the following are the blood-glucose distributions of three subgroups within the dataset.



Generating Hypothese

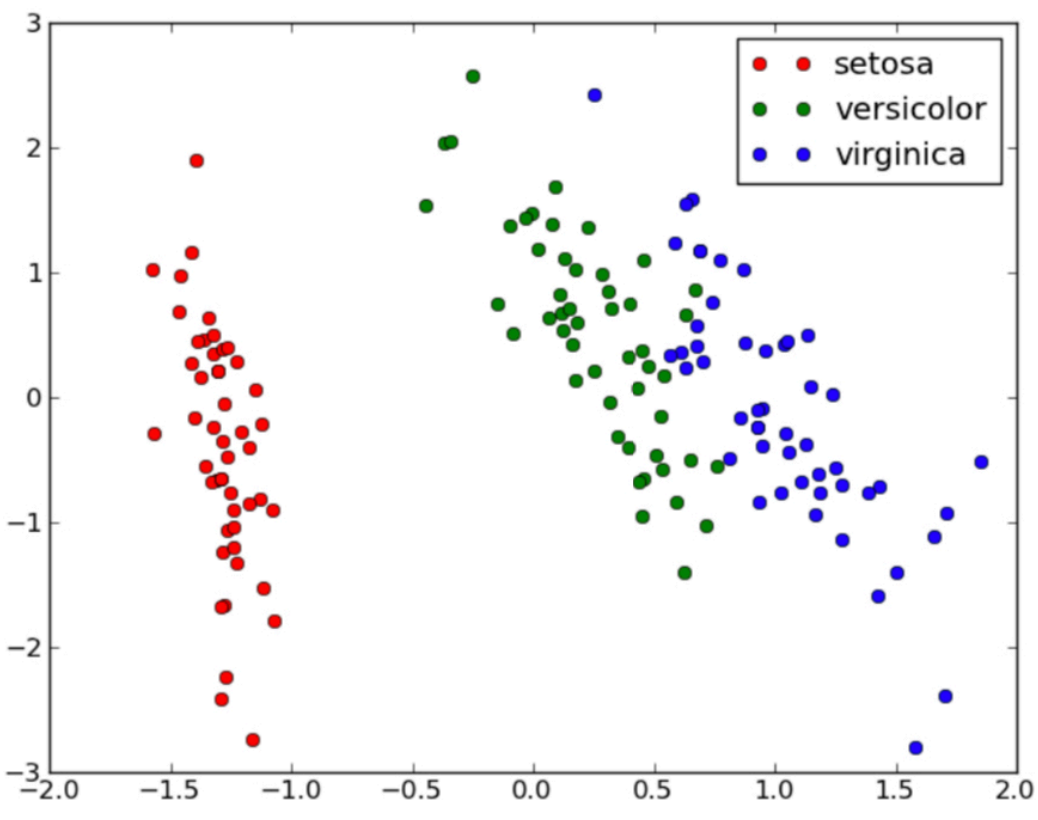
We can see that birth weight is positively correlated with femur length.



Can we describe exactly how they are correlated?

Generating Hypothese

We can see that types of iris seem to be distinguished by petal and sepal lengths.



Can we predict the type of iris given petal and sepal lengths?

Data Science Tools

Tools for the Workshop

In this workshop, we will be using `python`, for which there are many robust and well documented libraries for machine learning and data science.

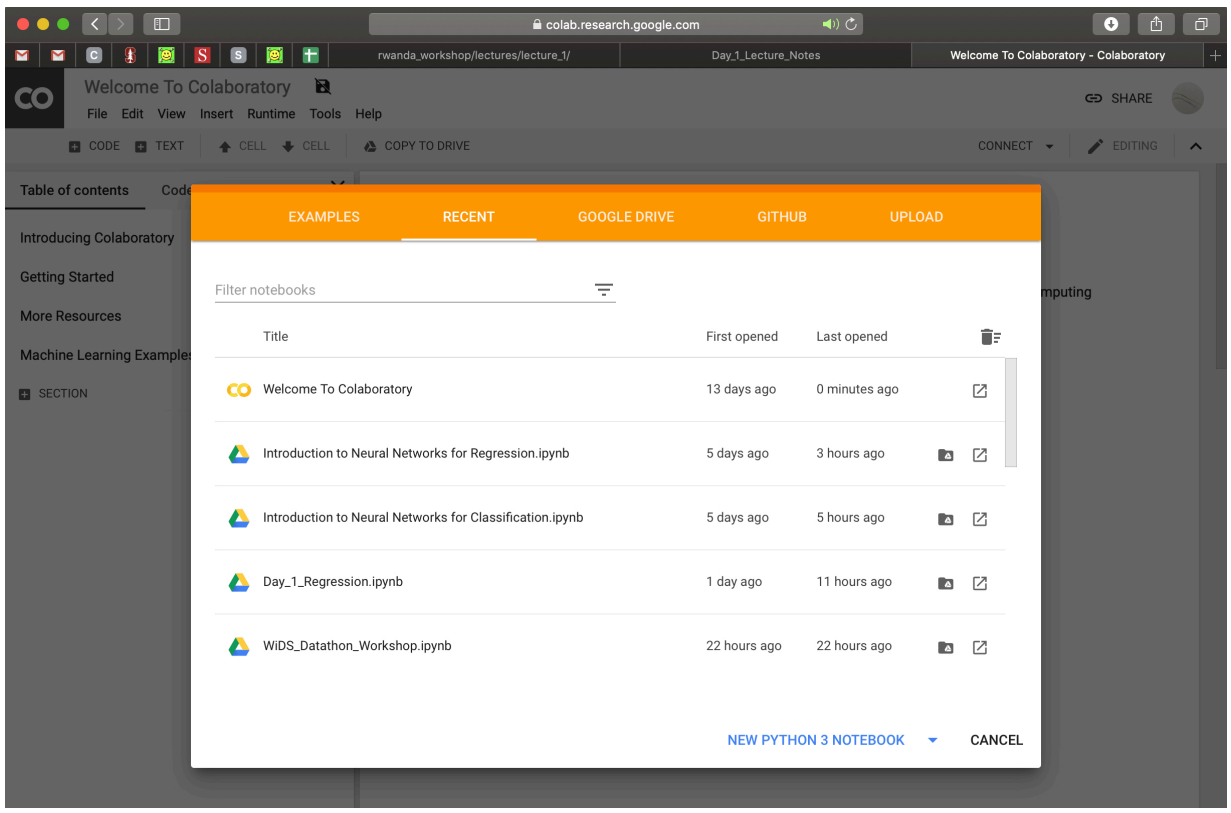
Specifically, we will use

1. `colab notebook` - a Google web app for creating interactive document containing text, equations, live code and outputs.
2. `pandas` - a `python` library for reading and manipulating data
3. `matplotlib` - a `python` library for data visualization
4. `numpy` - a `python` library for manipulating numeric data
5. `scikit-learn` - a `python` library with many machine learning models
6. `keras` - a `python` library for deep learning

`colab` Notebooks

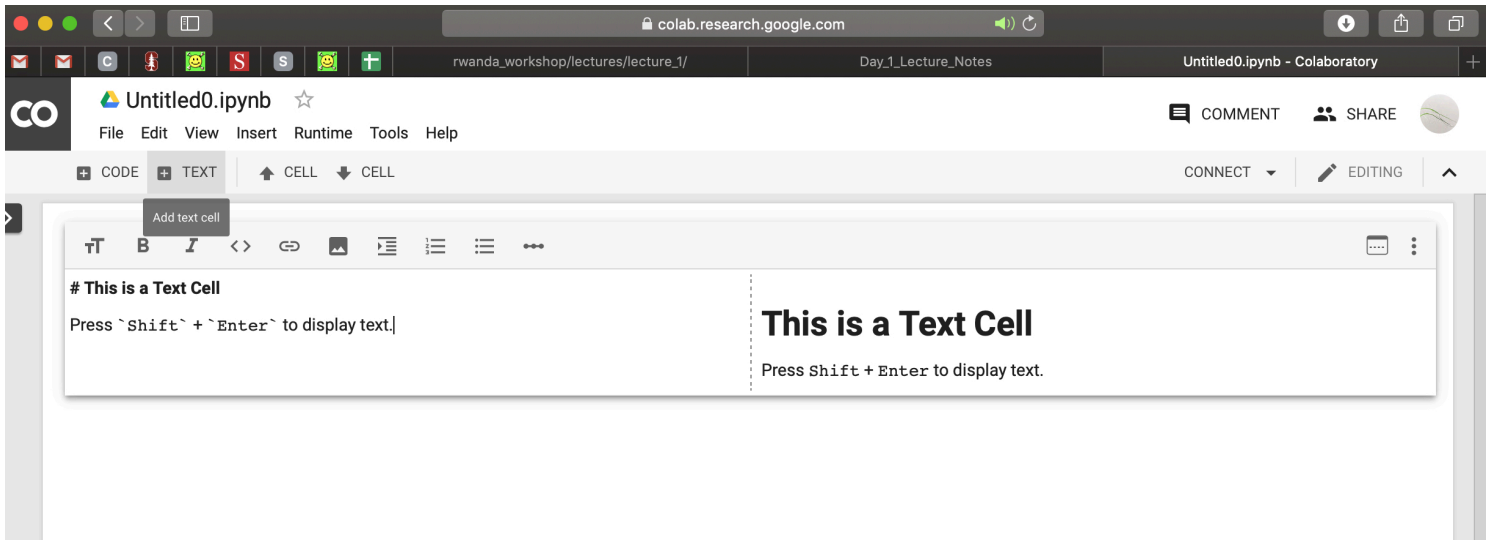
What Does Colab Notebook Look Like?

Go to `https://colab.research.google.com/`



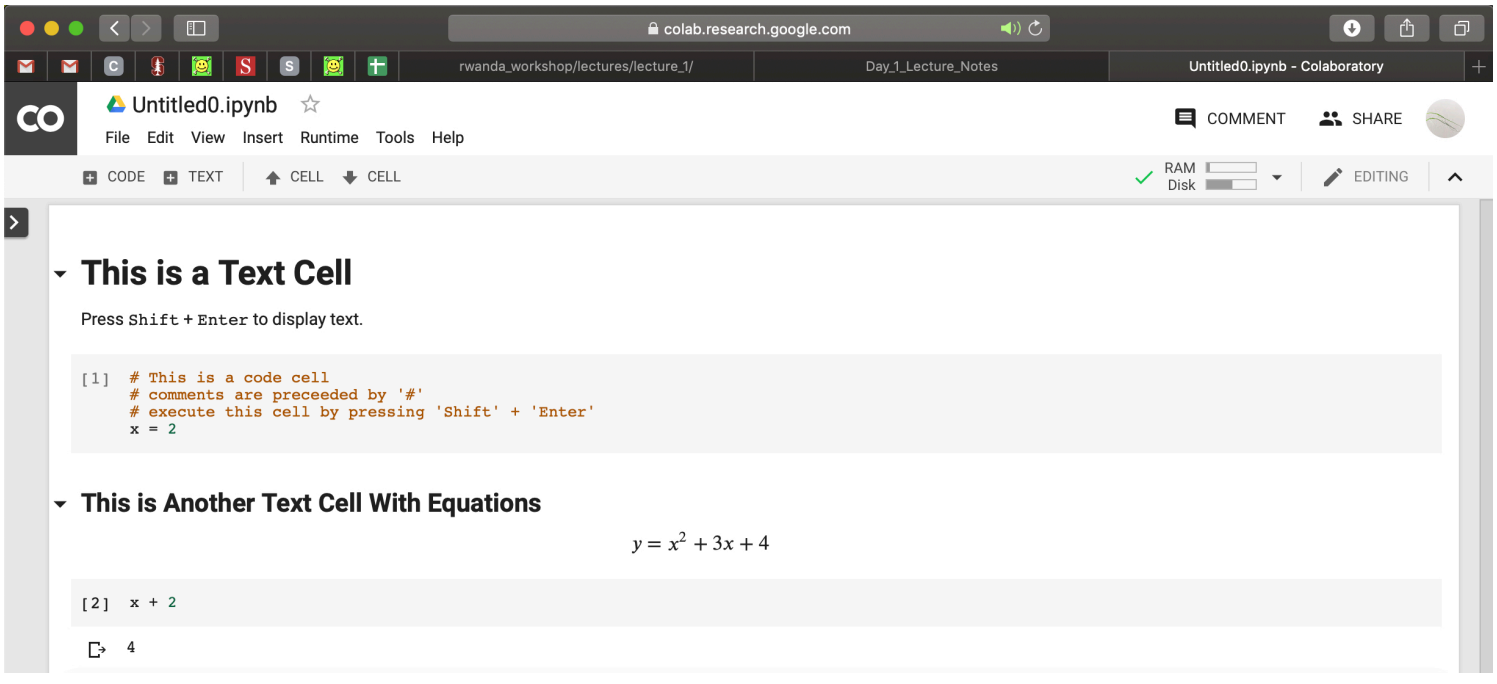
What Does Colab Notebook Look Like?

Each notebook consists of blocks of cells. Each cell can be of two types: (1) rich text (Markdown) cells or (2) code cells.



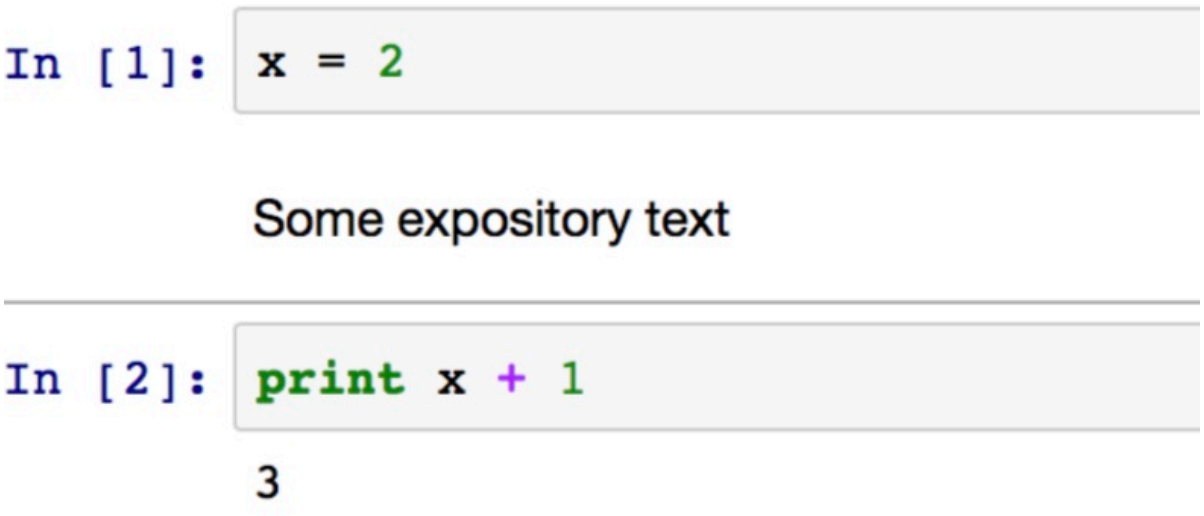
What Does Colab Notebook Look Like?

Code is executed by an "computational engine" called the ***kernel*** (IPython). The output of the code is displayed directly below..



What Does Colab Notebook Look Like?

Each cell can be executed independently, but once a block of code is executed, it lives in the memory of the kernel.



General Introduction to `python`

What Does Python Look Like?

Code readability is key, Python syntax itself is close to plain english.

Your variables should be given descriptive identifiers!

BAD	GOOD
`var6 = 25`	`age_of_mother = 25`
`AG3ofMoTh3R = 25`	`age_of_mother = 25`

Identifiers for variable should be descriptive words separated by underscore (not spaces) and in all lower case.

What Does Python Look Like?

You should use white space to increase readability.

BAD	GOOD
`x=[2,3,4]`	`num_list = [2, 3, 4]`
`v=1/3*(pi*r**2*h)`	`v = 1/3 * (pi * r**2 * h)`

What Does Python Look Like?

You should liberally intersperse your code with comments!

BAD

```
v = 1/3 * (pi * r**2 * h)
```

GOOD

```
#volume of cone
v = 1/3 * (pi * r**2 * h)
```

What Does Python Look Like?

Proper indentation is non-negotiable!

BAD

```
for i in range(5):
print i
```

GOOD

```
for i in range(5):
    print i
```

Code blocks are not indicated by delimiters (e.g. `{ }`) only by indentation!

Python Data Types

The basic built-in Python data types we'll be using today are:

1. **integers, floating points:** `7`, `7.0`
2. **booleans:** `True`, `False`
3. **strings:** `'hi'`, `"7.0"`
4. **lists:** `['hi', False, 7]`

Variables and Assignment

In `python` the type of variables is inferred based on the valued assigned to the variable. For example: The assignment

```
my_var = 7
```

types `my_var` as an integer. Later, the assignment

```
my_var = 'hello'
```

will cause `my_var` to be typed as a string.

Learning `python`

The course website contains readings, cheatsheets and tutorials for mastering general `python` programming as well as using `python` libraries for data manipulation/visualization.

Tips for learning a programmming language:

1. Don't read code line by line (at first)
2. Copy as much as possible (learn patterns and templates)
3. Don't be afriad to play (trial and error)
4. Read documentation!

`pandas`: Reading and Manipulating Data

`pandas`

`pandas` provides ways of storing tabular data along with labels on the rows and columns.

To use any `python` library, we must include the line

```
import pandas as pd
```

After which we may use any function or object in this library using for example, `pd.Series()`.

`pandas` Series

The `pandas` Series object stores a 1D array of data with an index object (labels).

In [4]:
```
import pandas as pd

column = pd.Series([0.25, 0.5, 0.75, 1.0],
                   index=['one', 'two', 'three', 'four'])
column
```

Out[4]:
```
one      0.25
two      0.50
three    0.75
four     1.00
dtype: float64
```

In [6]:
```
column['four']
```

Out[6]:
```
1.0
```

`pandas` Series

Each `pandas` series has a `.values` and an `.index` attribute.

In [4]:

Slide Type -

```
column = pd.Series([0.25, 0.5, 0.75, 2.0],
                   index=['one', 'two', 'three', 'four'])
column.values
```

Out[4]:
```
array([0.25, 0.5 , 0.75, 2.  ])
```

In [5]:

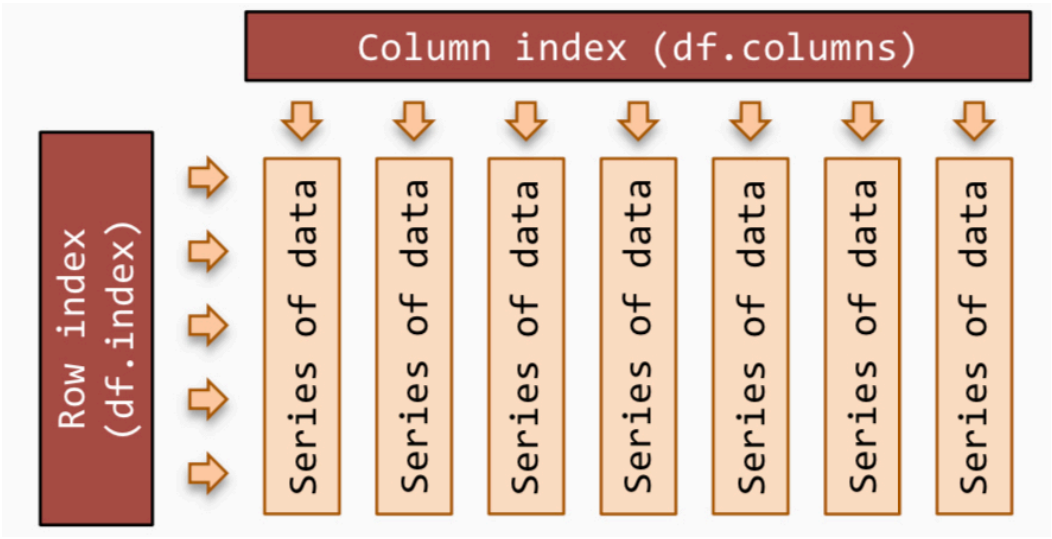
Slide Type -

```
column.index
```

Out[5]:
```
Index(['one', 'two', 'three', 'four'], dtype='object')
```

`pandas` DataFrames

The `pandas` DataFrame object stores a 2D table of data with column and row index object (labels).



Each column in the data frame is a series.

Reading Data with `pandas`

We can import tabular data in a csv file into a data frame:

In [16]:
```
df = pd.read_csv('dataset_HW0.txt')
df
```

Out[16]:

	birth_weight	femur_length	mother_age
0	2.969489	1.979156	16
1	4.038963	3.555681	16
2	5.302643	3.385633	15
3	6.086107	4.495427	17

Basic Data Exploration with `pandas`

We should start by checking how large is the data.

We do this by checking the shape of the DataFrame.

```
In [23]: df.shape
Out[23]: (400, 3)

In [10]: len(df.index)
Out[10]: 400
```

Basic Data Exploration with `pandas`

We should start by getting a rough sense of what's in the data.

```
In [32]: df.columns
Out[32]: Index([u'birth_weight', u'femur_length', u'mother_age'], dtype='object')

In [5]: df.columns.values
Out[5]: array(['birth_weight', 'femur_length', 'mother_age'], dtype=object)

In [6]: df.index
Out[6]: Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
                    ...
                    390, 391, 392, 393, 394, 395, 396, 397, 398, 399],
                   dtype='int64', length=400)

In [7]: df.index.values
Out[7]: array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
                13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
                26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
                39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
```

Basic Data Exploration with `pandas`

The `.head()` function returns the first `N` rows of your data frame!

```
In [25]: df.head(n=5)
```

Out[25]:

	birth_weight	femur_length	mother_age
0	2.969489	1.979156	16
1	4.038963	3.555681	16
2	5.302643	3.385633	15
3	6.086107	4.495427	17
4	5.749260	4.017437	16

Basic Data Exploration with `pandas`

The `.describe()` function returns the descriptive stats for each numeric column as a data frame object!

```
In [29]: df.describe()
```

Out[29]:

	birth_weight	femur_length	mother_age
count	400.000000	400.000000	400.00000
mean	6.104070	3.827591	27.06000
std	1.097011	0.853577	10.34984
min	2.967426	0.479154	15.00000
25%	5.429120	3.281786	17.75000
50%	6.110025	3.817888	25.00000
75%	6.839935	4.351204	34.25000
max	9.021942	6.648730	49.00000

Basic Data Manipulation with `pandas`

Accessing a column by label:

```
In [34]: type(df['birth_weight'])
Out[34]: pandas.core.series.Series

In [35]: df['birth_weight']
Out[35]: 0      2.969489
         1      4.038963
         2      5.302643
         3      6.086107
         4      5.749260
         5      6.049903
```

Basic Data Manipulation with `pandas`

Accessing columns by label:

```
In [34]: type(df['birth_weight'])
Out[34]: pandas.core.series.Series

In [35]: df['birth_weight']
Out[35]: 0      2.969489
         1      4.038963
         2      5.302643
         3      6.086107
         4      5.749260
         5      6.049903
```

Basic Data Manipulation with `pandas`

Accessing columns by criteria (***filtering***):

```
In [57]: df[(df['mother_age'] > 18) & (df['mother_age'] < 35)]
```

Out[57]:

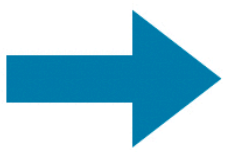
	birth_weight	femur_length	mother_age
100	6.904530	4.164637	34
101	8.096642	4.536759	22
102	8.165373	5.507030	20
104	6.255286	3.769024	19
105	6.515220	5.568954	23
106	6.464462	3.310628	25
107	6.579616	3.670224	20
108	7.171024	5.159946	24

Basic Data Transformation with `pandas`

Since machine learning models are mathematical functions (i.e. require numerical input), what do we do with categorical attributes?

We encode them as numerical vectors, whose position encodes for the possible values of the attribute. This is called ***one-hot encoding***. Why can't we encode them as integers?

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

Basic Data Transformation with `pandas`

One-hot encoding in `pandas`.

```
# convert categorical columns to numerical columns by one-hot encoding
one_hot_categorical = pd.get_dummies(df_categorical_columns, prefix=df_cate
gorical_columns.columns)
```

Exercise: Perform EDA on the Berlin Airbnb Dataset

Today we will be working with a real data set: the Berlin Airbnb Dataset.

Find the link to this notebook on the workshop website:

```
https://github.com/onefishy/rwanda_workshop
```

Exploratory Data Analysis and Regression

Introduction to Data Science

Kigali, Rwanda

July 8th, 2019

What is this course?

This is a 5 day short workshop on the ***practical*** fundamentals of data science using python.

This workshop will cover a variety of models solving a range of different data science tasks; it will familiarize participants with useful `python` libraries for data science.

What is the structure of each day?

Each session of the workshop is 4 hours. Roughly half of which will be lecture and half will be hands-on exercises.

You will work in groups during class to complete coding exercises solving problems involving ***real*** data.

What is expected of the participants?

This workshop involves a great deal of programming and requires a intuitive but sound understanding of the mathematical or statistical theory behind machine learning models.

To be successful, participants need to have some previous experience with programming as well as basic statistics; participants need to be committed to gaining a working knowledge of `python` during the workshop.

How is the workshop graded?

Participants are recommended to complete all in-class exercises as homework.

There will be an in-class, 4-hour, open-book exam on Day 5. The exam will require participants to solve a data science task using a real-data set and will be based on the skills demonstrated in the in-class exercises.

More information about the course

All materials and information can be found at the course website:

```
https://github.com/onefishy/rwanda_workshop
```

Introduction to Data Science

Outline

1. What Is Data?
2. What is Data Science?
3. Exploring Data
    - Descriptive Statistics
    - Data Visualization
4. Data Science Tools

What is Data?

What is Data?

*“A **datum** is a single measurement of something on a scale that is understandable to both the recorder and the reader. **Data** is multiple such measurements.”*

Claim: everything is (can be) data!

What is Data Science

What Does it Mean to Do Data Science?

The "data science" process.

1. Find a problem you want to solve
2. Collection data you think will help solve it
3. Exploration Data Analysis:
    - examine your data for patterns and trends.
4. Build mathematical models to describe trends and patterns
5. Analyze your model
    - What does it say about your data?
    - How do your finding help solve your problem?
6. Visualization and Presentation of Results

Note: This process is not linear!

Where Does Data Come From?

Internal sources: already collected by or is part of the overall data collection of you organization.

Existing External Sources: available in ready to read format from an outside source for free or for a fee.

External Sources Requiring Collection Efforts: available from external source but acquisition requires special processing.

What Does Data Look Like?

Simple or atomic data:

- **Numeric:** integers, floats
- **Boolean:** binary or true false values
- **Strings:** sequence of symbols

What Does Data Look Like?

Compound, composed of a bunch of atomic types:

- **Date and time:** compound value with a specific structure
- **Lists:** a list is a sequence of values
- **Dictionaries:** collections of key-value pairs, a pair of values `x : y`, where x is usually a string called key representing the "name" of the value, and y is a value of any type.

  Example: Student record
  - First: Weiwei
  - Last: Pan
  - Nationality: USA

How is your data represented and stored?

Data formats:

- **Tabular Data:** a dataset that is a two-dimensional table, where each row typically represents a single data record, and each column represents one type of measurement (csv, xlsx etc.).
- **Structured Data:** each data record is presented in a form of a, possibly complex and multi-tiered, dictionary (json, xml etc.)
- **Semistructured Data:** not all records are represented by the same set of keys or some data records are not represented using the key-value pair structure.

Tabular Data

In tabular data, each row or ***observation*** represents a set of measurements of a single object or event.

	Hight	Radius	Do I Like It?
Cylinder # 1	10	5	Yes
Cylinder # 2	3	7.5	No

Each type of measurement is an ***attribute*** of the data. The number of attributes is the ***dimension*** of the data.

Types of Attributes in Tabular Data

It's vital to distinguish between classes of attributes based on the type of values they take on.

- **Quantitative attribute:** a real valued number whose values can be ordered.

  Example: Height is a quantitative attribute

- **Categorical attribute:** a real valued or string with no inherent order among the values.

  Example: "What kind of pet you have" is a categorical attribute

Is the data any good?

- **Missing values:** how do we fill in?
- **Wrong values:** how can we detect and correct?
- **Messy format:** how can we convert structured or semistructured data into tabular data?
- **Not usable:** what if the data cannot answer the question posed?

Exploring Data

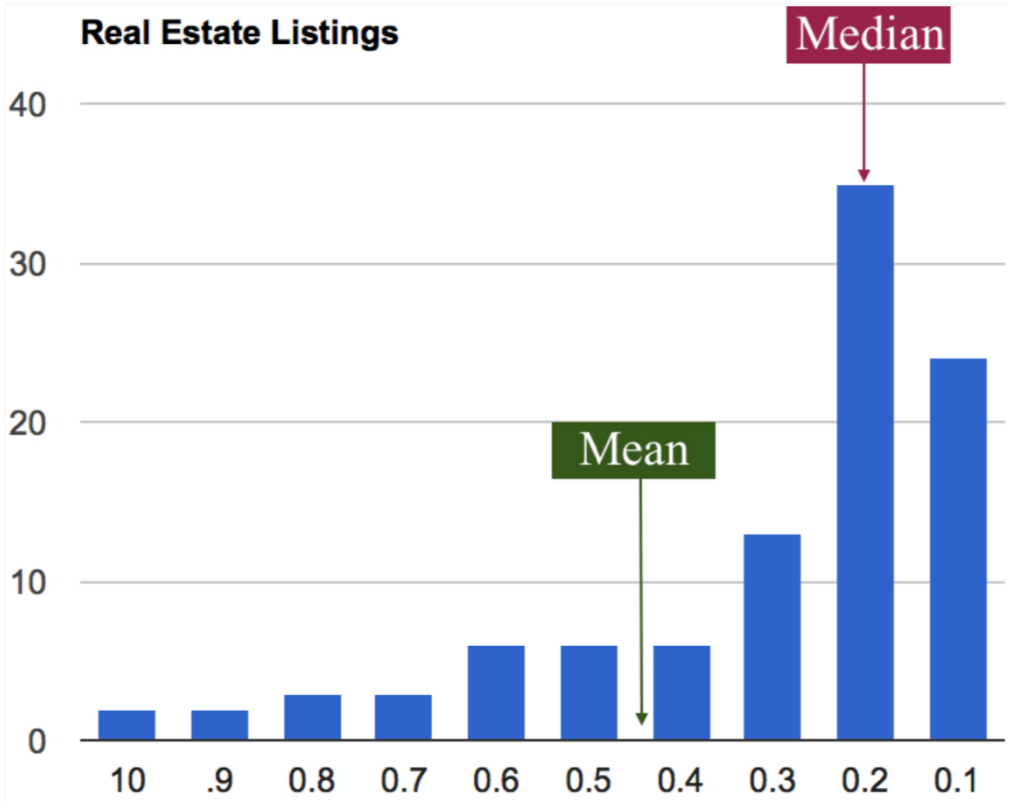
Describing Individual Attributes

Given a large dataset, we want to compute a few quantities that intuitively summarizes the data. We'd like to know:

1. what are "typical" values for our attributes?
2. how "representative" are these typical values?

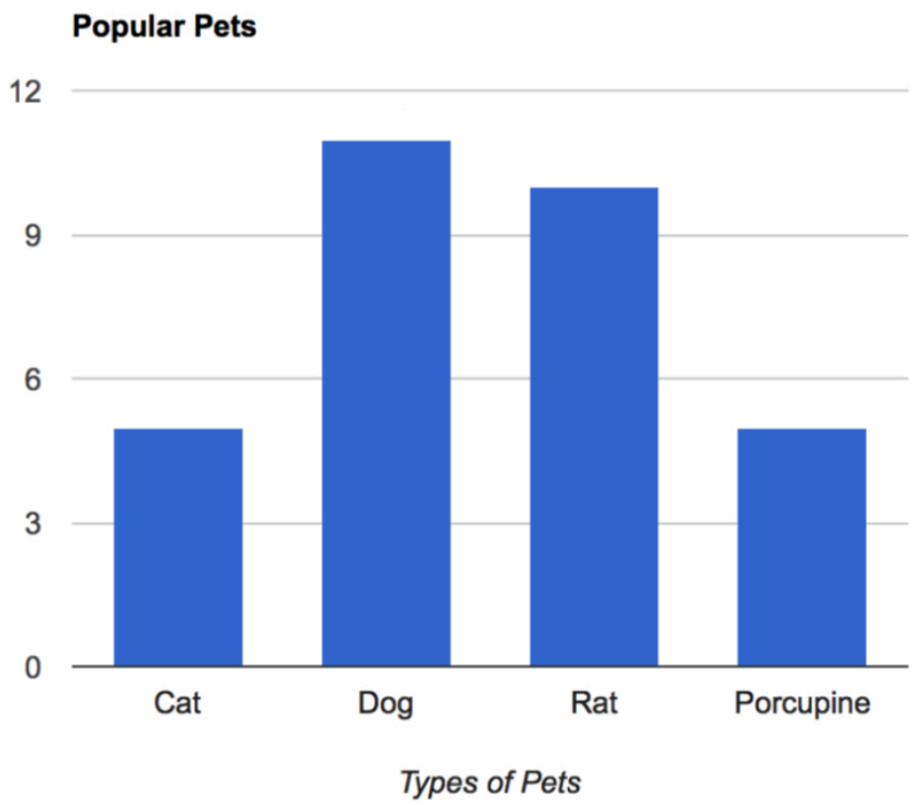
Describing A Typical Value

We can describe a typical value for n samples of a **quantitative** attribute x by the ***mean*** or the ***median***.



Describing A Typical Value

We can describe a typical value for n samples of a **categorical** attribute x by the ***mode***.



Describing the Spread of a Value

The spread of samples measures how well the mean or median describes the sample set:

1. ***Range***
2. ***Standard variance***
3. ***Standard deviation***

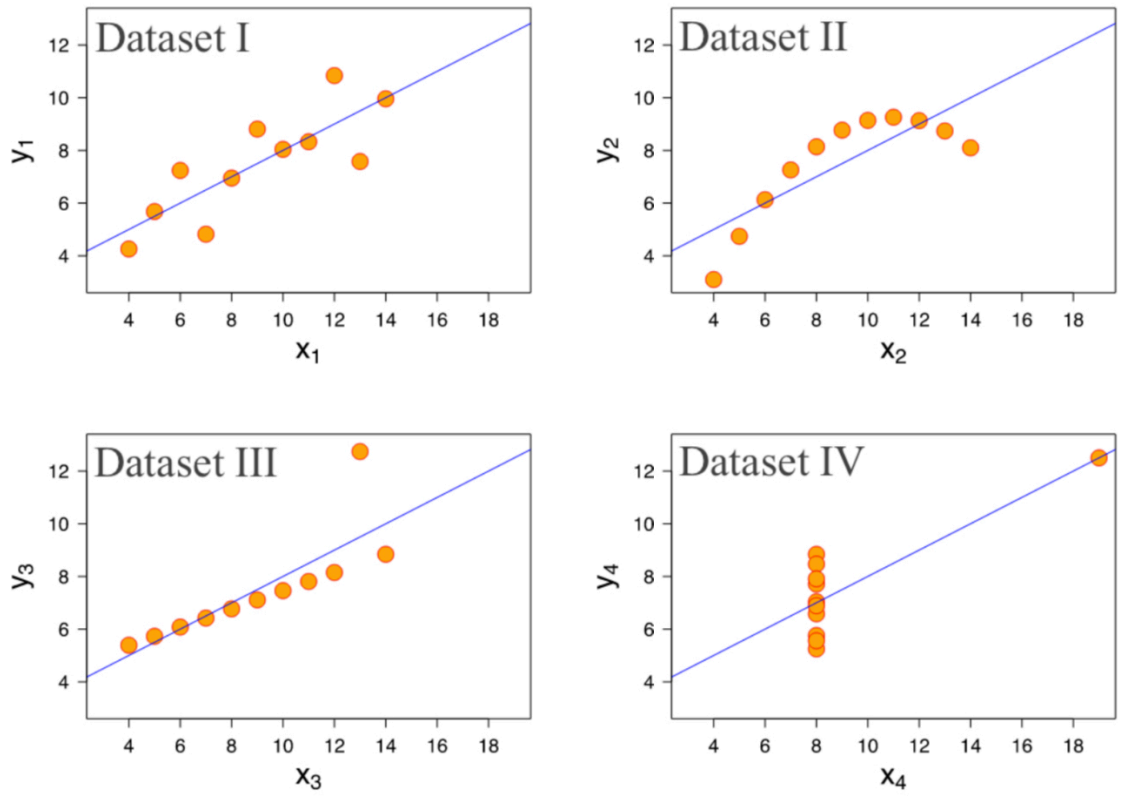
Why Data Visualization?

The following is called Anscombe's Quartet; all four sets of data have identical simple summary statistics.

	Dataset I		Dataset II		Dataset III		Dataset IV	
	x	y	x	y	x	y	x	y
	10	8.04	10	9.14	10	7.46	8	6.58
	8	6.95	8	8.14	8	6.77	8	5.76
	13	7.58	13	8.74	13	12.74	8	7.71
	9	8.81	9	8.77	9	7.11	8	8.84
	11	8.33	11	9.26	11	7.81	8	8.47
	14	9.96	14	8.1	14	8.84	8	7.04
	6	7.24	6	6.13	6	6.08	8	5.25
	4	4.26	4	3.1	4	5.39	19	12.5
	12	10.84	12	9.13	12	8.15	8	5.56
	7	4.82	7	7.26	7	6.42	8	7.91
	5	5.68	5	4.74	5	5.73	8	6.89
Sum:	99.00	82.51	99.00	82.51	99.00	82.51	99.00	82.51
Avg:	9.00	7.50	9.00	7.50	9.00	7.50	9.00	7.50
Std:	3.32	2.03	3.32	2.03	3.32	2.03	3.32	2.03

Anscombe's Quartet

The following is called Anscombe's Quartet; all four sets of data have identical simple summary statistics.



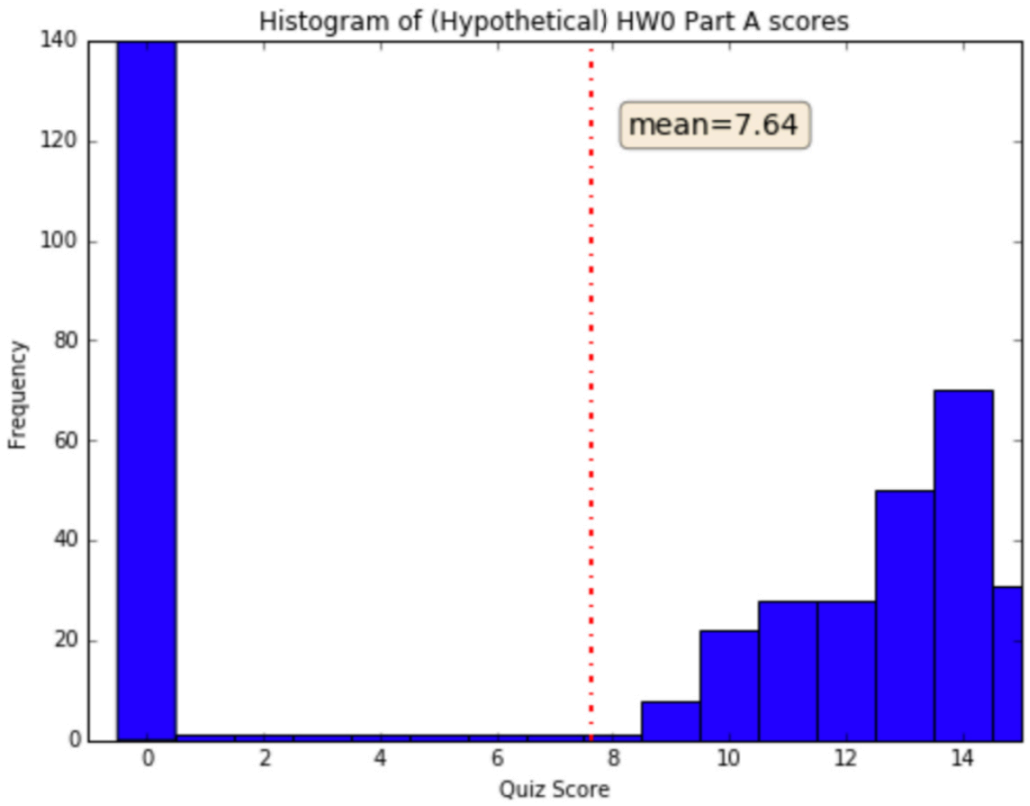
Why Data Visualization?

If I tell you that the average score for the final examis: 7.64/15.

What does that suggest?

Why Data Visualization?

If I then show you the following graph, what does it suggest?



What is Data Visualization Good For?

Analyze:

- Identify hidden patterns and trends
- Help formulate/test hypothese
- Help determine the next step in analysis/modeling

What is Data Visualization Good For?

Communicate:

- Present information and ideas succinctly
- Provide evidence and support
- Influence and persuade

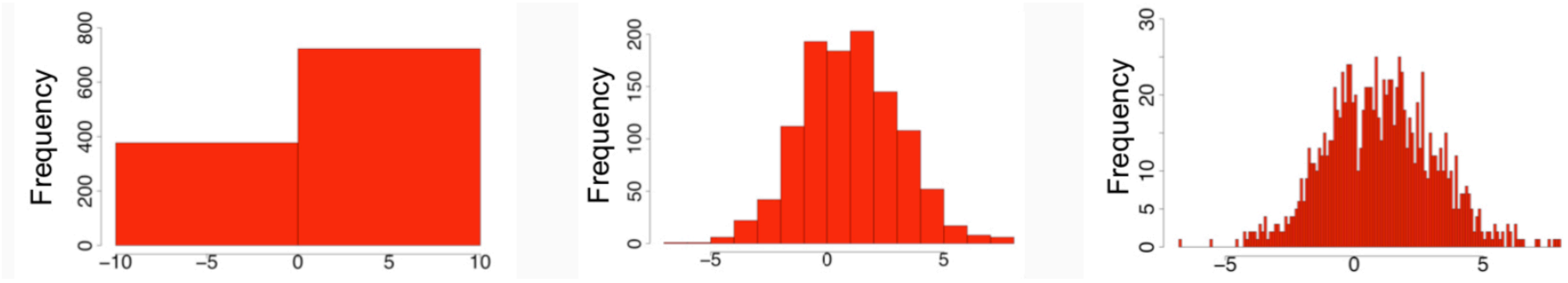
Types of Data Visualization

What do you want your visualization to show about your data?

- **Distribution:** how a variable or variables in the dataset distribute over a range of possible values.
- **Relationship:** how the values of multiple variables in the dataset relate
- **Composition:** how the dataset breaks down into subgroups
- **Comparison:** how trends in multiple variable or datasets compare

Visualizing Distributions

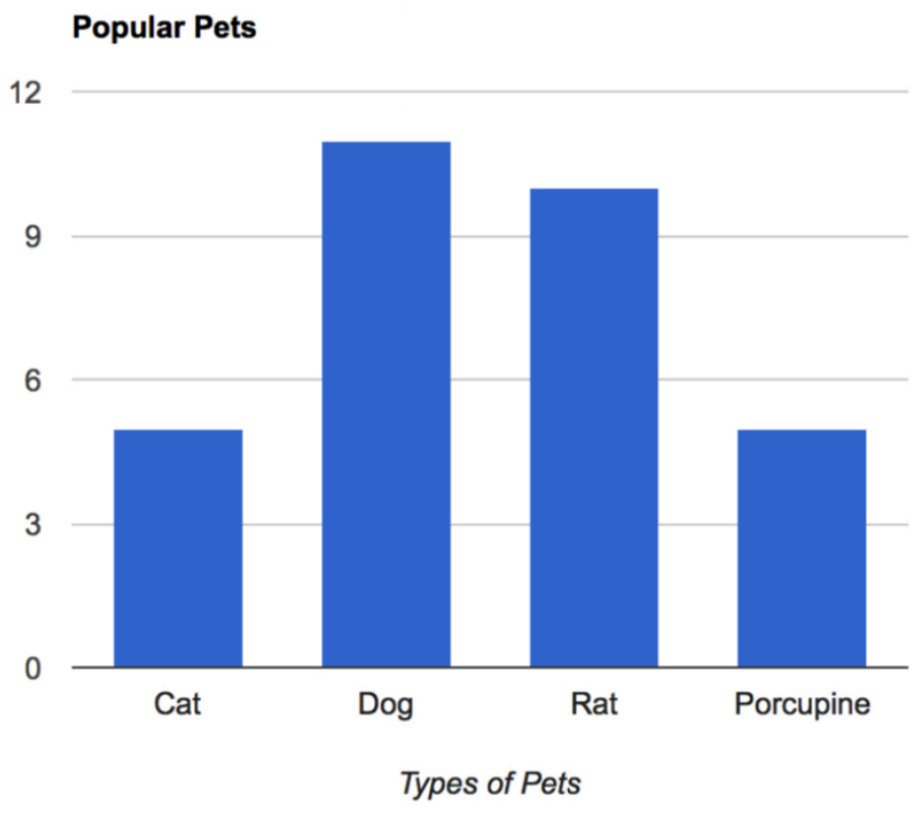
A ***histogram*** is a way to visualize how a **quantitative** attribute is distributed across certain values.



Note: Trends in histograms are sensitive to number of bins.

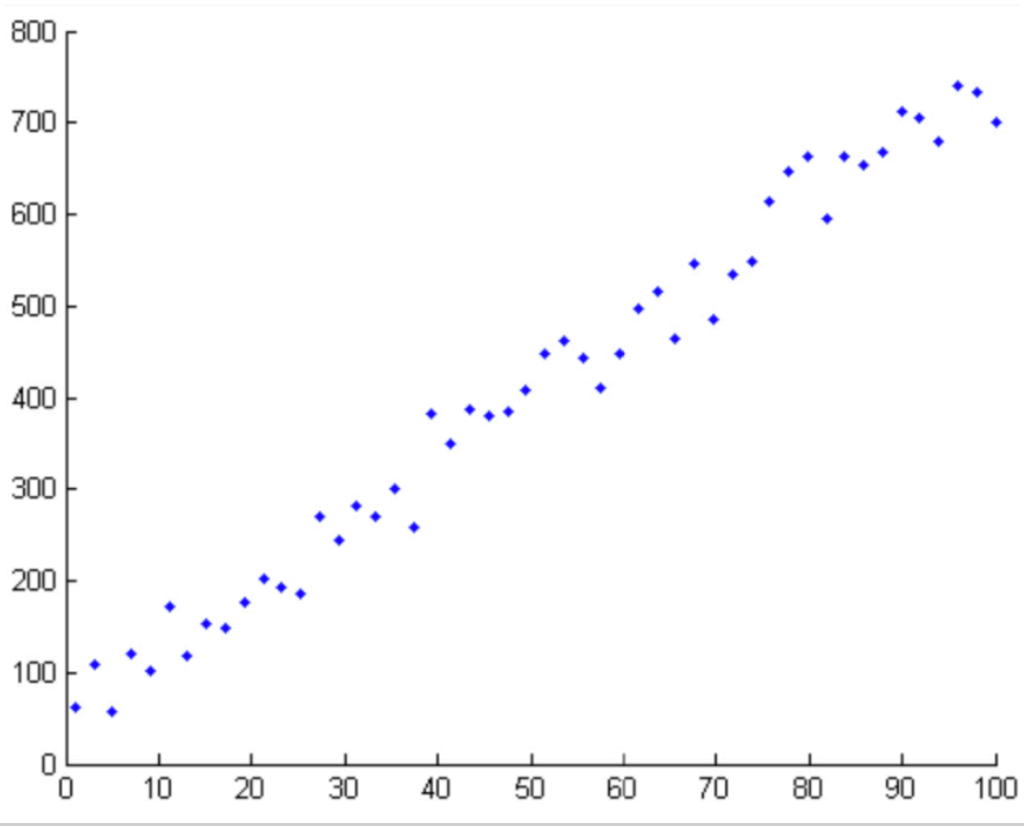
Visualizing Distributions

A ***bar chart*** is a way to visualize how a **categorical** attribute is distributed across certain values.



Visualizing Relationships

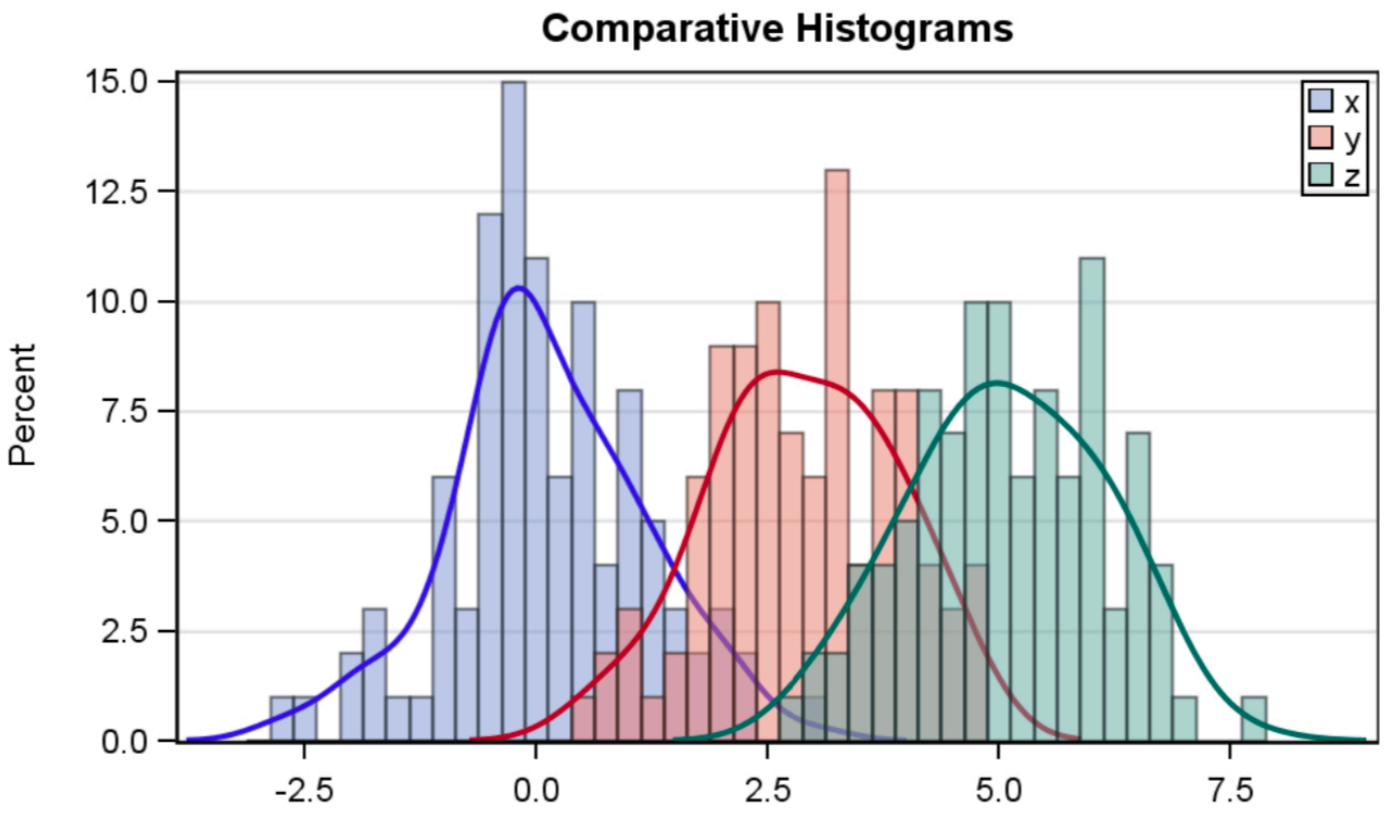
A ***scatter plot*** is a way to visualize the relationship between two different attributes.



Visualizing Comparisons of Subgroups

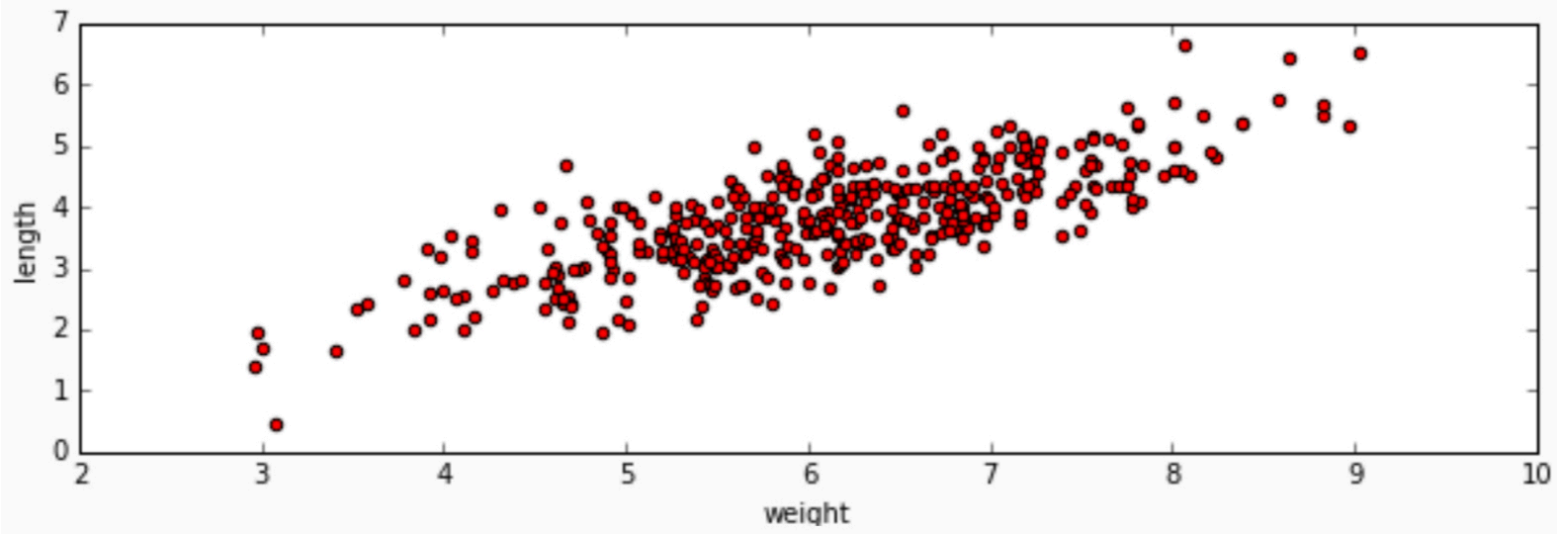
Plotting multiple histograms or curves on the same axes is a way to visualize how different subgroups of data compare.

For example: the following are the blood-glucose distributions of three subgroups within the dataset.



Generating Hypothese

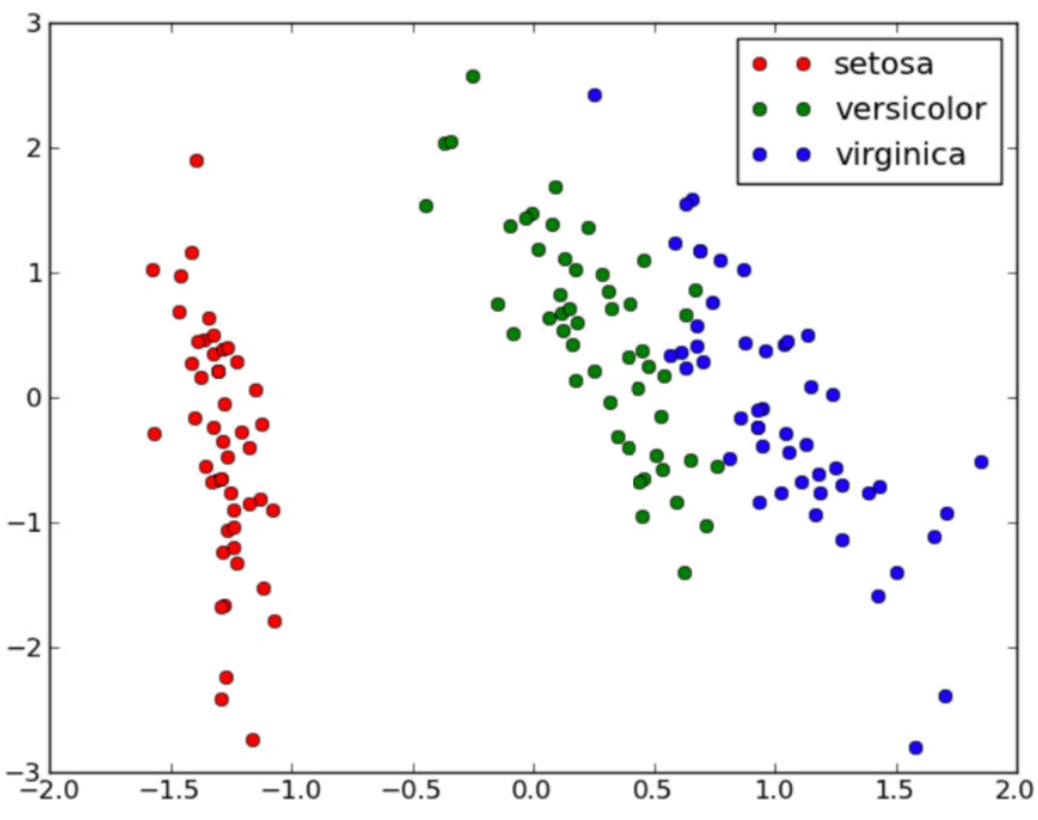
We can see that birth weight is positively correlated with femur length.



Can we describe exactly how they are correlated?

Generating Hypothese

We can see that types of iris seem to be distinguished by petal and sepal lengths.



Can we predict the type of iris given petal and sepal lengths?

Data Science Tools

Tools for the Workshop

In this workshop, we will be using `python`, for which there are many robust and well documented libraries for machine learning and data science.

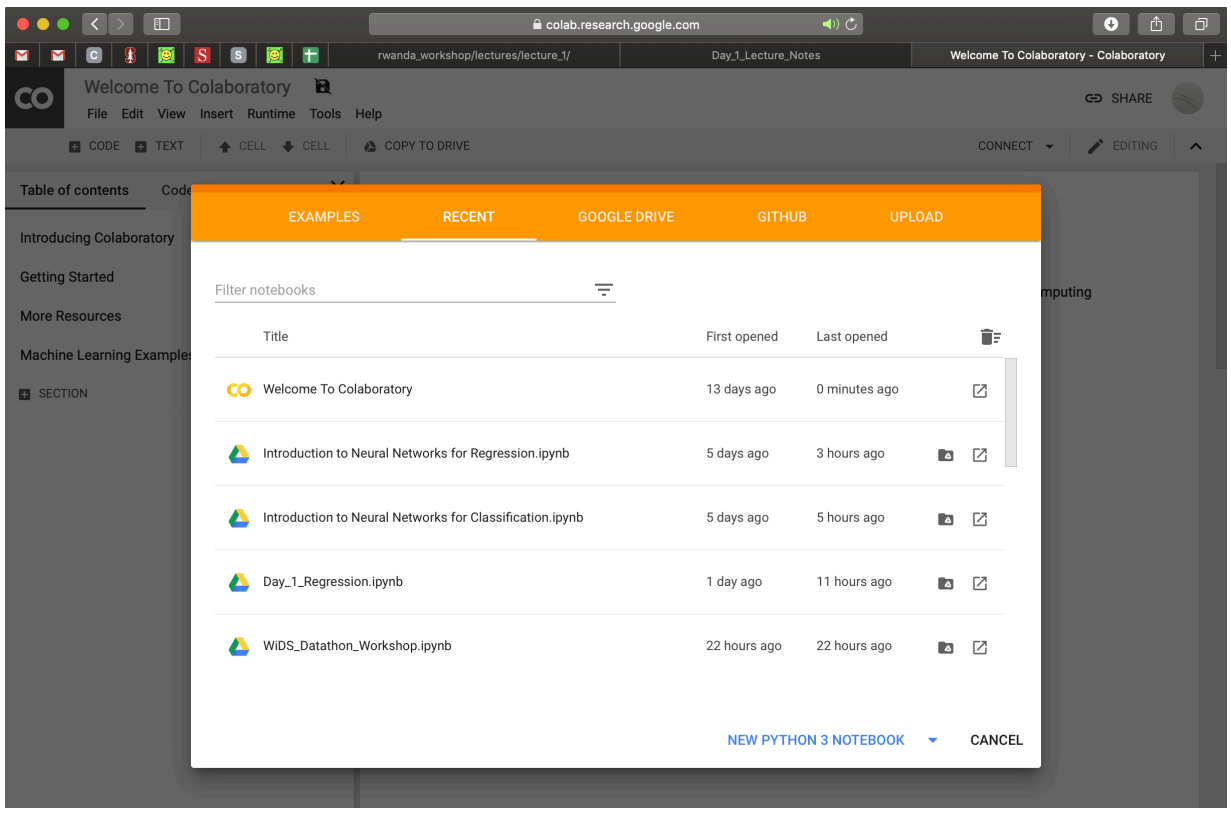
Specifically, we will use

1. `colab notebook` - a Google web app for creating interactive document containing text, equations, live code and outputs.
2. `pandas` - a `python` library for reading and manipulating data
3. `matplotlib` - a `python` library for data visualization
4. `numpy` - a `python` library for manipulating numeric data
5. `scikit-learn` - a `python` library with many machine learning models
6. `keras` - a `python` library for deep learning

`colab` Notebooks

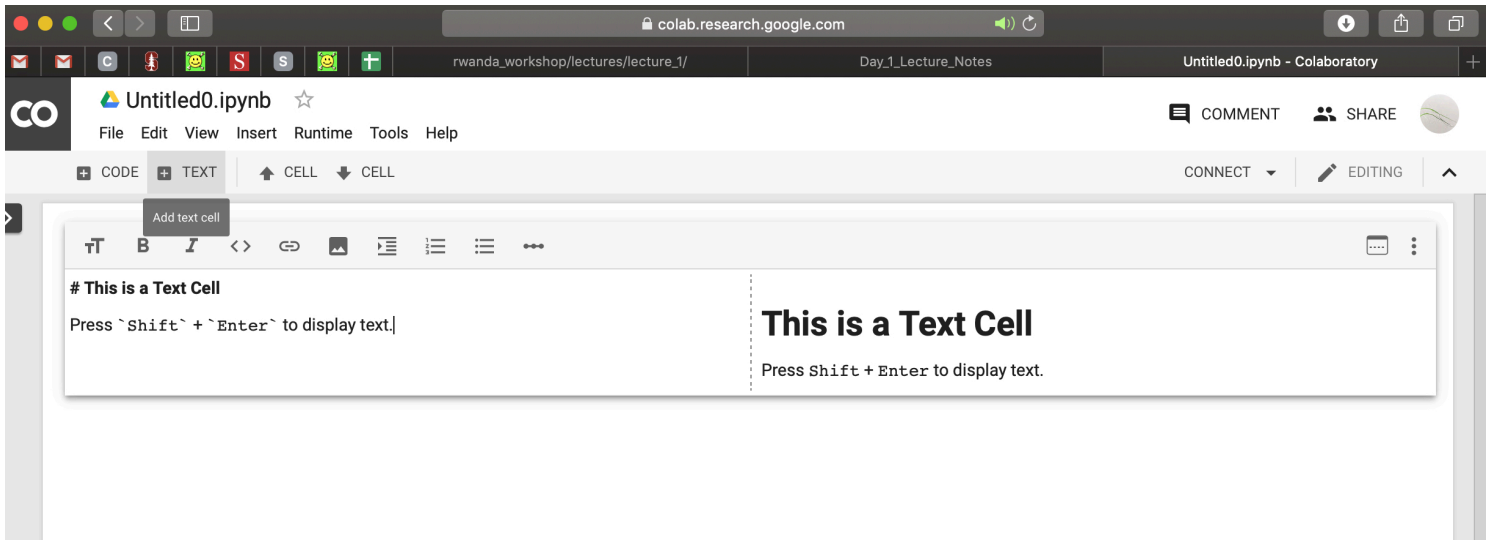
What Does Colab Notebook Look Like?

Go to `https://colab.research.google.com/`



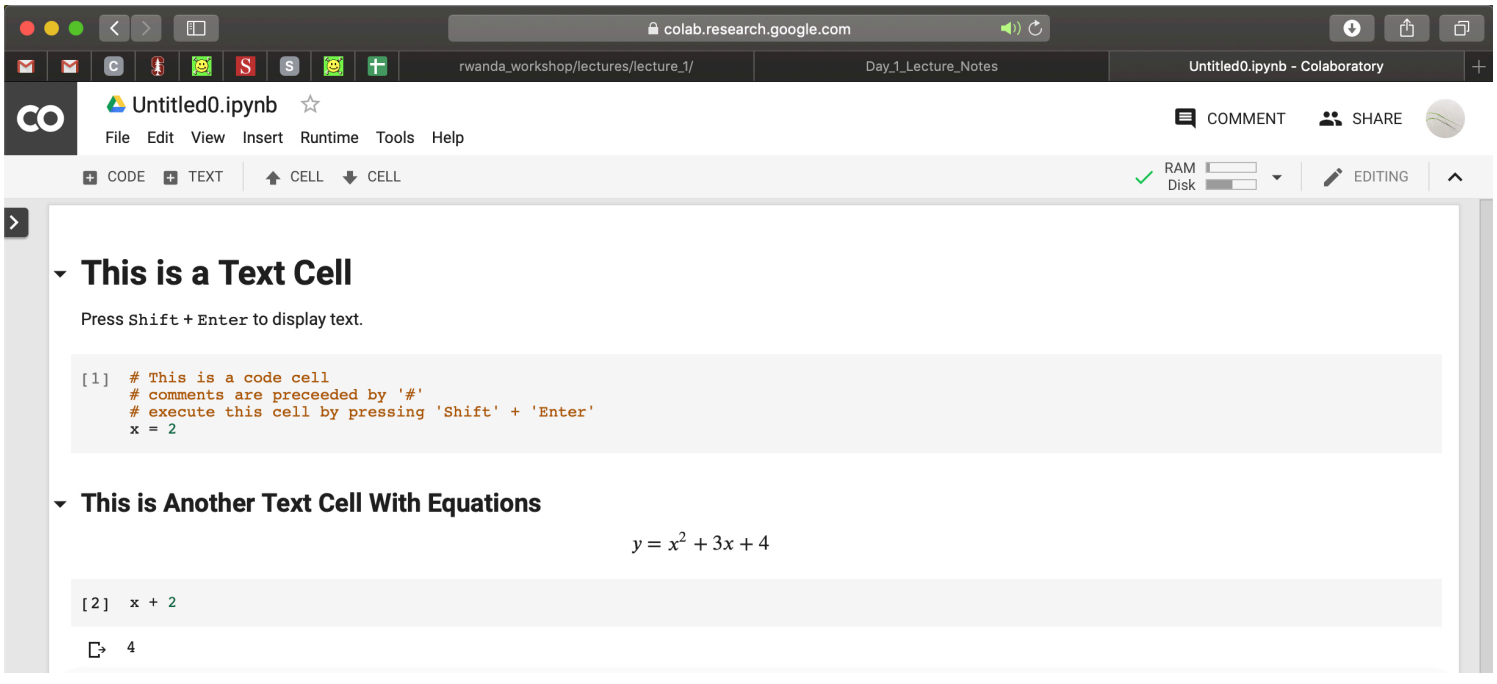
What Does Colab Notebook Look Like?

Each notebook consists of blocks of cells. Each cell can be of two types: (1) rich text (Markdown) cells or (2) code cells.



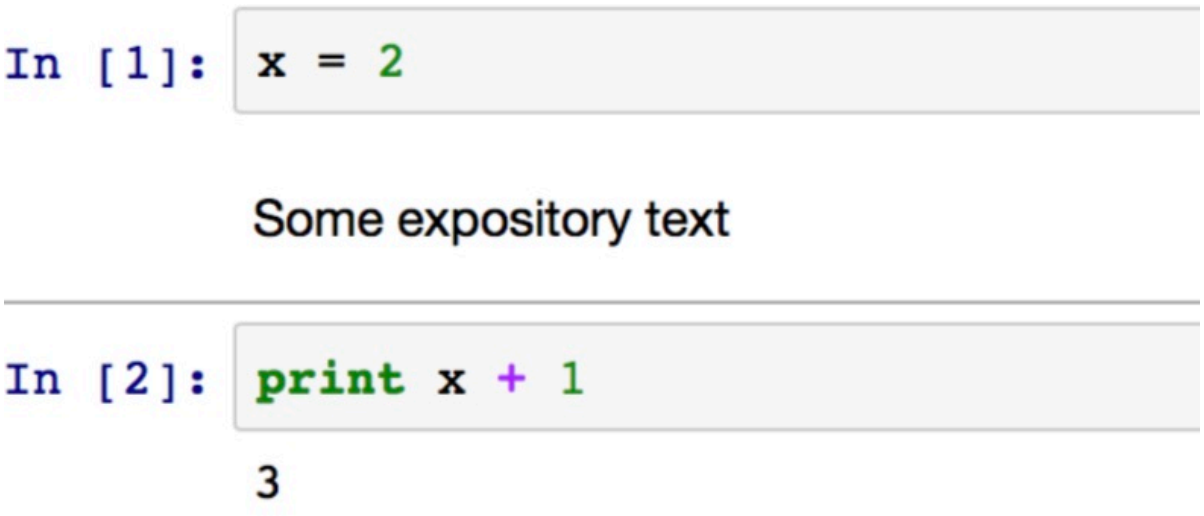
What Does Colab Notebook Look Like?

Code is executed by an "computational engine" called the ***kernel*** (IPython). The output of the code is displayed directly below..



What Does Colab Notebook Look Like?

Each cell can be executed independently, but once a block of code is executed, it lives in the memory of the kernel.



General Introduction to `python`

What Does Python Look Like?

Code readability is key, Python syntax itself is close to plain english.

Your variables should be given descriptive identifiers!

BAD	GOOD
`var6 = 25`	`age_of_mother = 25`
`AG3ofMoTh3R = 25`	`age_of_mother = 25`

Identifiers for variable should be descriptive words separated by underscore (not spaces) and in all lower case.

What Does Python Look Like?

You should use white space to increase readability.

BAD	GOOD
`x=[2,3,4]`	`num_list = [2, 3, 4]`
`v=1/3*(pi*r**2*h)`	`v = 1/3 * (pi * r**2 * h)`

What Does Python Look Like?

You should liberally intersperse your code with comments!

BAD

```
v = 1/3 * (pi * r**2 * h)
```

GOOD

```
#volume of cone
v = 1/3 * (pi * r**2 * h)
```

What Does Python Look Like?

Proper indentation is non-negotiable!

BAD

```
for i in range(5):
print i
```

GOOD

```
for i in range(5):
    print i
```

Code blocks are not indicated by delimiters (e.g. `{ }`) only by indentation!

Python Data Types

The basic built-in Python data types we'll be using today are:

1. **integers, floating points:** `7`, `7.0`
2. **booleans:** `True`, `False`
3. **strings:** `'hi'`, `"7.0"`
4. **lists:** `['hi', False, 7]`

Variables and Assignment

In `python` the type of variables is inferred based on the valued assigned to the variable. For example: The assignment

```
my_var = 7
```

types `my_var` as an integer. Later, the assignment

```
my_var = 'hello'
```

will cause `my_var` to be typed as a string.

Learning `python`

The course website contains readings, cheatsheets and tutorials for mastering general `python` programming as well as using `python` libraries for data manipulation/visualization.

Tips for learning a programmming language:

1. Don't read code line by line (at first)
2. Copy as much as possible (learn patterns and templates)
3. Don't be afriad to play (trial and error)
4. Read documentation!

`pandas`: Reading and Manipulating Data

`pandas`

`pandas` provides ways of storing tabular data along with labels on the rows and columns.

To use any `python` library, we must include the line

```
import pandas as pd
```

After which we may use any function or object in this library using for example, `pd.Series()`.

`pandas` Series

The `pandas` Series object stores a 1D array of data with an index object (labels).

In [4]:
```
import pandas as pd

column = pd.Series([0.25, 0.5, 0.75, 1.0],
                   index=['one', 'two', 'three', 'four'])
column
```

Out[4]:
```
one      0.25
two      0.50
three    0.75
four     1.00
dtype: float64
```

In [6]:
```
column['four']
```

Out[6]:
```
1.0
```

`pandas` Series

Each `pandas` series has a `.values` and an `.index` attribute.

In [4]:

Slide Type -

```
column = pd.Series([0.25, 0.5, 0.75, 2.0],
                   index=['one', 'two', 'three', 'four'])
column.values
```

Out[4]:
```
array([0.25, 0.5 , 0.75, 2.  ])
```

In [5]:

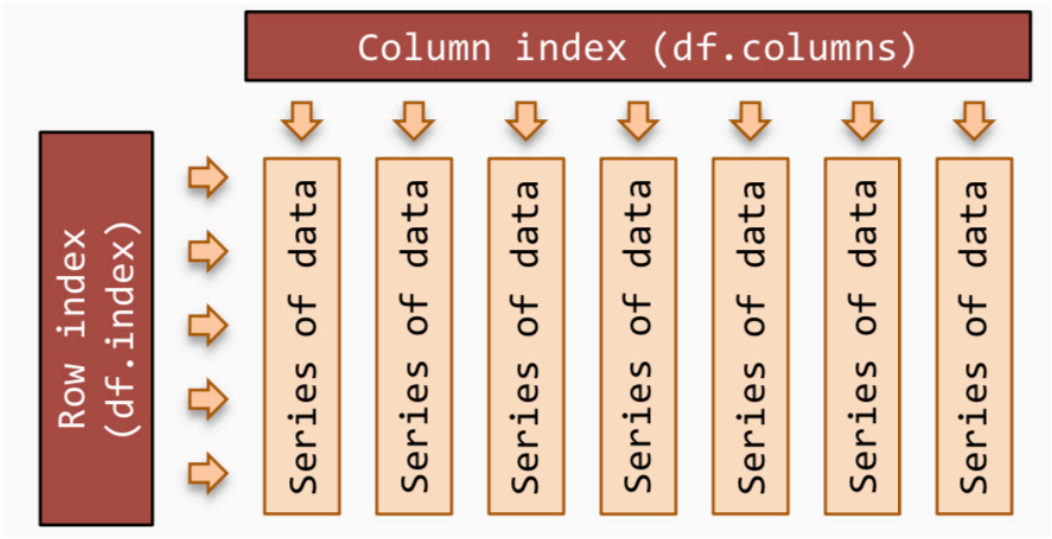
Slide Type -

```
column.index
```

Out[5]:
```
Index(['one', 'two', 'three', 'four'], dtype='object')
```

`pandas` DataFrames

The `pandas` DataFrame object stores a 2D table of data with column and row index object (labels).



Each column in the data frame is a series.

Reading Data with `pandas`

We can import tabular data in a csv file into a data frame:

In [16]:
```
df = pd.read_csv('dataset_HW0.txt')
df
```

Out[16]:

	birth_weight	femur_length	mother_age
0	2.969489	1.979156	16
1	4.038963	3.555681	16
2	5.302643	3.385633	15
3	6.086107	4.495427	17

Basic Data Exploration with `pandas`

We should start by checking how large is the data.

We do this by checking the shape of the DataFrame.

```
In [23]: df.shape
Out[23]: (400, 3)

In [10]: len(df.index)
Out[10]: 400
```

Basic Data Exploration with `pandas`

We should start by getting a rough sense of what's in the data.

```
In [32]: df.columns
Out[32]: Index([u'birth_weight', u'femur_length', u'mother_age'], dtype='object')

In [5]: df.columns.values
Out[5]: array(['birth_weight', 'femur_length', 'mother_age'], dtype=object)

In [6]: df.index
Out[6]: Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
                    ...
                    390, 391, 392, 393, 394, 395, 396, 397, 398, 399],
                   dtype='int64', length=400)

In [7]: df.index.values
Out[7]: array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
                13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
                26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
                39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
```

Basic Data Exploration with `pandas`

The `.head()` function returns the first `N` rows of your data frame!

```
In [25]: df.head(n=5)
```

Out[25]:

	birth_weight	femur_length	mother_age
0	2.969489	1.979156	16
1	4.038963	3.555681	16
2	5.302643	3.385633	15
3	6.086107	4.495427	17
4	5.749260	4.017437	16

Basic Data Exploration with `pandas`

The `.describe()` function returns the descriptive stats for each numeric column as a data frame object!

```
In [29]: df.describe()
```

Out[29]:

	birth_weight	femur_length	mother_age
count	400.000000	400.000000	400.00000
mean	6.104070	3.827591	27.06000
std	1.097011	0.853577	10.34984
min	2.967426	0.479154	15.00000
25%	5.429120	3.281786	17.75000
50%	6.110025	3.817888	25.00000
75%	6.839935	4.351204	34.25000
max	9.021942	6.648730	49.00000

Basic Data Manipulation with `pandas`

Accessing a column by label:

```
In [34]: type(df['birth_weight'])
Out[34]: pandas.core.series.Series

In [35]: df['birth_weight']
Out[35]: 0      2.969489
         1      4.038963
         2      5.302643
         3      6.086107
         4      5.749260
         5      6.049903
```

Basic Data Manipulation with `pandas`

Accessing columns by label:

```
In [34]: type(df['birth_weight'])
Out[34]: pandas.core.series.Series

In [35]: df['birth_weight']
Out[35]: 0      2.969489
         1      4.038963
         2      5.302643
         3      6.086107
         4      5.749260
         5      6.049903
```

Basic Data Manipulation with `pandas`

Accessing columns by criteria (***filtering***):

```
In [57]: df[(df['mother_age'] > 18) & (df['mother_age'] < 35)]
```

Out[57]:

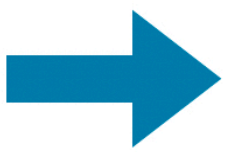
	birth_weight	femur_length	mother_age
100	6.904530	4.164637	34
101	8.096642	4.536759	22
102	8.165373	5.507030	20
104	6.255286	3.769024	19
105	6.515220	5.568954	23
106	6.464462	3.310628	25
107	6.579616	3.670224	20
108	7.171024	5.159946	24

Basic Data Transformation with `pandas`

Since machine learning models are mathematical functions (i.e. require numerical input), what do we do with categorical attributes?

We encode them as numerical vectors, whose position encodes for the possible values of the attribute. This is called ***one-hot encoding***. Why can't we encode them as integers?

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

Basic Data Transformation with `pandas`

One-hot encoding in `pandas`.

```
# convert categorical columns to numerical columns by one-hot encoding
one_hot_categorical = pd.get_dummies(df_categorical_columns, prefix=df_cate
gorical_columns.columns)
```

Exercise: Perform EDA on the Berlin Airbnb Dataset

Today we will be working with a real data set: the Berlin Airbnb Dataset.

Find the link to this notebook on the workshop website:

```
https://github.com/onefishy/rwanda_workshop
```