

The internet's best courses on:

Data Science

Machine Learning

Python



Daily Learning ▾ Courses ▾ Team

549

2.6K

You are reading Tutorials



Learn Machine Learning by Doing

Learn Now



A Complete . . .

Enter your email

Subscribe



Author: [George McIntire](#)
Data Scientist



Author: [Brendan Mart...](#)
Founder of LearnDataSci



Author: [Lauren Washington](#),
Lead Data Scientist & ML Developer

Python Pandas Tutorial: A Complete Introduction for Beginners

Learn some of the most important pandas features for exploring, cleaning, transforming, visualizing, and learning from data.

Contents Index

+

You should already know:

- Python fundamentals – you should have beginner to intermediate-level knowledge, which can be learned from most entry-level [Python courses](#)



Learn Machine Learning by Doing

Learn Now



[pandas] is derived from the term "pane" which refers to a window containing data sets that include observations over multiple time periods for the same individuals. — Wikipedia

If you're thinking about data science as a career, then it is imperative that one of the first things you do is learn pandas. In this post, we will go over the essential bits of information about pandas, including how to install it, its uses, and how it works with other common Python data analysis packages such as **matplotlib** and **scikit-learn**.

Article Resources

- iPython notebook and data available on GitHub

Other articles in this series

- Applied Introduction to NumPy





data by cleaning, transforming, and analyzing

For example, say you want to explore a dataset. pandas will extract the data from that CSV into a DataFrame — a table, basically — then let you do things like:

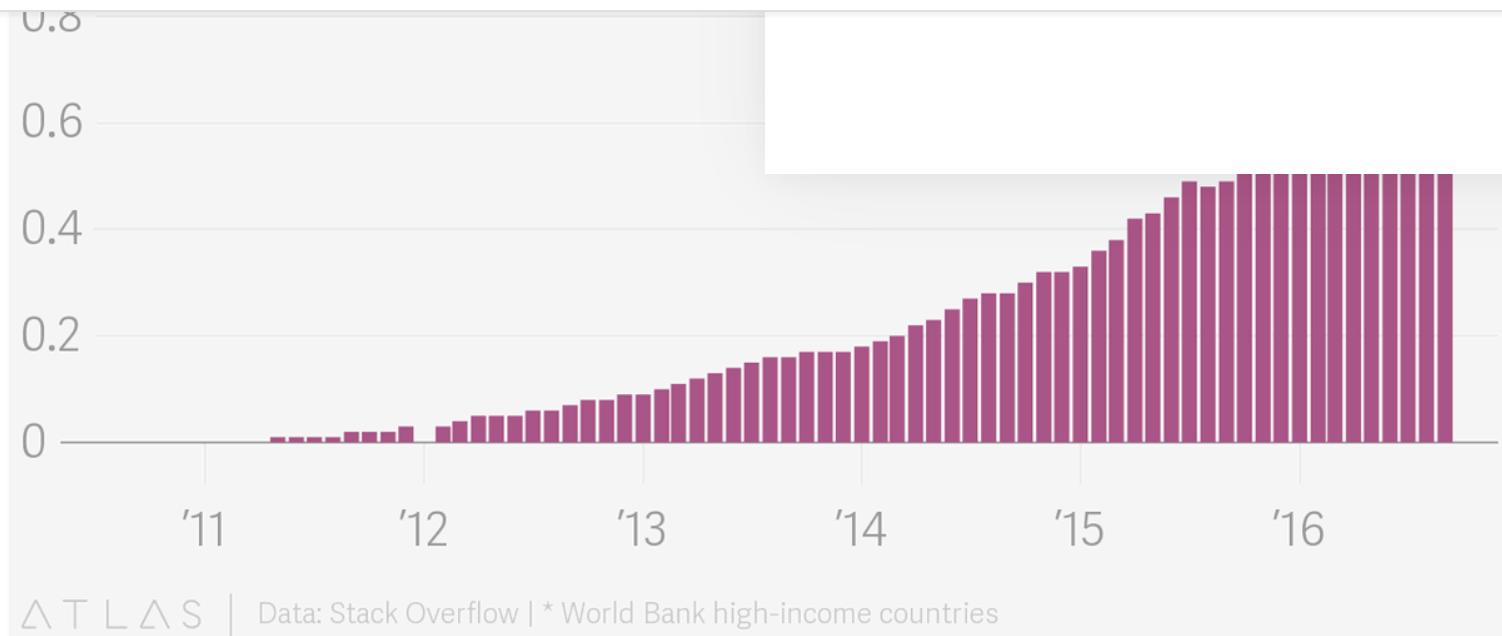
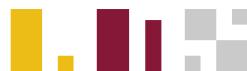
- Calculate statistics and answer questions about the data, like
 - What's the average, median, max, or min of each column?
 - Does column A correlate with column B?
 - What does the distribution of data in column C look like?
- Clean the data by doing things like removing missing values and filtering rows or columns by some criteria
- Visualize the data with help from Matplotlib. Plot bars, lines, histograms, bubbles, and more.
- Store the cleaned, transformed data back into a CSV, other file or database

Before you jump into the modeling or the complex visualizations you need to have a good understanding of the nature of your dataset and pandas is the best avenue through which



Learn Machine Learning by Doing

Learn Now



How does pandas fit into the data science toolkit?

Not only is the pandas library a central component of the data science toolkit but it is used in conjunction with other libraries in that collection.

Pandas is built on top of the **NumPy** package, meaning a lot of the structure of NumPy is





running the entire file. This saves a lot of time complex transformations. Notebooks also provide DataFrames and plots. As a matter of fact, this Notebook.

When should you start using pandas?

If you do not have any experience coding in Python, then you should stay away from learning pandas until you do. You don't have to be at the level of the software engineer, but you should be adept at the basics, such as lists, tuples, dictionaries, functions, and iterations. Also, I'd also recommend familiarizing yourself with **NumPy** due to the similarities mentioned above.

If you're looking for a good place to learn Python, [Python for Everybody](#) on Coursera is great (and Free).

Test your Python knowledge 



Learn Machine Learning by Doing

Learn Now



```
fruit = 'apple'

if fruit == 'Apple':
    print('The fruit is an apple')
elif fruit == 'Orange':
    print('The fruit is an orange')
else:
    print('The fruit is unidentified')
```

'apple'

'The fruit is an orange'

'The fruit is an apple'

'The fruit is unidentified'

I don't know yet

Prev

Next

SCORE



Learn Machine Learning by Doing

Learn Now



Even though accelerated programs teach you
you'll be able to maximize time for learning an

Pandas First Steps

Install and import

Pandas is an easy package to install. Open up your terminal program (for Mac users) or command line (for PC users) and install it using either of the following commands:

```
conda install pandas
```

OR

```
pip install pandas
```

Alternatively, if you're currently viewing this article in a Jupyter notebook you can run this cell:



Learn Machine Learning by Doing

Learn Now



The `!` at the beginning runs cells as if they w

To import pandas we usually import it with a shorter name since it's used so much:

```
import pandas as pd
```

Learn Data Science with  DATAQUEST

Now to the basic components of pandas.

Core components of pandas: Series and DataFrames

The primary two components of pandas are the `Series` and `DataFrame`.

A `Series` is essentially a column, and a `DataFrame` is a multi-dimensional table made up of a collection of Series.



Learn Machine Learning by Doing

Learn Now



The diagram illustrates the addition of two 4x2 DataFrames. On the left, there is a 4x2 DataFrame with the following values:

0	3
1	2
2	0
3	1

On the right, there is another 4x2 DataFrame with the following values:

0	0
1	3
2	7
3	2

A plus sign (+) is placed between the two DataFrames, indicating their sum. To the right of the plus sign is the resulting 4x3 DataFrame:

2	0	7
3	1	2

DataFrames and Series are quite similar in that many operations that you can do with one you can do with the other, such as filling in null values and calculating the mean.

You'll see how these components work when we start working with data below.

Creating DataFrames from scratch

Creating DataFrames right in Python is good to know and quite useful when testing new methods and functions you find in the pandas docs.

There are *many* ways to create a DataFrame from scratch, but a great option is to just use a simple `dict`.





```
data = {  
    'apples': [3, 2, 0, 1],  
    'oranges': [0, 3, 7, 2]  
}
```

Learn Data Science with  DATAQUEST

And then pass it to the pandas DataFrame constructor:

```
purchases = pd.DataFrame(data)  
  
purchases
```

OUT:

	apples	oranges
0	3	0
1	2	3
2	0	7



Learn Machine Learning by Doing

Learn Now



The **Index** of this DataFrame was given to us and we can also create our own when we initialize the DataFrame:

Let's have customer names as our index:

```
purchases = pd.DataFrame(data, index=['June', 'Robert', 'Lily', 'David'])

purchases
```

OUT:

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

So now we could **locate** a customer's order by using their name:



Learn Machine Learning by Doing

Learn Now



```
apples      3  
oranges     0  
Name: June, dtype: int64
```

Learn Data Science with  DATAQUEST

There's more on locating and extracting data from the DataFrame later, but now you should be able to create a DataFrame with any random data to learn on.

Let's move on to some quick methods for creating DataFrames from various other sources.

Want to learn more?

See Best Data Science Courses

[View](#)



Learn Machine Learning by Doing

[Learn Now](#)



Reading data from CSVs

With CSV files all you need is a single line to load in the data:

```
df = pd.read_csv('purchases.csv')

df
```

OUT:

	Unnamed: 0	apples	oranges
0	June	3	0
1	Robert	2	3
2	Lily	0	7
3	David	1	2

CSVs don't have indexes like our DataFrames, so all we need to do is just designate the



Learn Machine Learning by Doing

Learn Now



OUT:

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Here we're setting the index to be column zero.

You'll find that most CSVs won't ever have an index column and so usually you don't have to worry about this step.

Reading data from JSON

If you have a JSON file — which is essentially a stored Python `dict` — pandas can read this just as easily:



Learn Machine Learning by Doing

Learn Now



OUT:

	apples	oranges
David	1	2
June	3	0
Lily	0	7
Robert	2	3

Notice this time our index came with us correctly since using JSON allowed indexes to work through nesting. Feel free to open `data_file.json` in a notepad so you can see how it works.

Pandas will try to figure out how to create a DataFrame by analyzing structure of your JSON, and sometimes it doesn't get it right. Often you'll need to set the `orient` keyword argument depending on the structure, so check out `read_json` docs about that argument to see which orientation you're using.

Reading data from a SQL database



Learn Machine Learning by Doing

Learn Now



First, we need `pysqlite3` installed, so run this

```
pip install pysqlite3
```

Or run this cell if you're in a notebook:

```
!pip install pysqlite3
```

Learn Data Science with  DATAQUEST

`sqlite3` is used to create a connection to a database which we can then use to generate a DataFrame through a `SELECT` query.

So first we'll make a connection to a SQLite database file:

```
import sqlite3

con = sqlite3.connect("database.db")
```



Learn Machine Learning by Doing

Learn Now



If you have data in PostgreSQL, MySQL or SQLite, you'll need to obtain the right Python library. For example, `psycopg2` (link) is a common, well-known, well-maintained library for making connections to PostgreSQL. Furthermore, you would make a connection to a database URI instead of a file like we did here with SQLite.

For a great course on SQL check out [The Complete SQL Bootcamp](#) on Udemy

In this SQLite database we have a table called *purchases*, and our index is in a column called "index".

By passing a SELECT query and our `con`, we can read from the *purchases* table:

```
df = pd.read_sql_query("SELECT * FROM purchases", con)
df
```





1	Robert	2	3
2	Lily	0	7
3	David	1	2

Just like with CSVs, we could pass `index_col='index'`, but we can also set an index after-the-fact:

```
df = df.set_index('index')

df
```





June	3	0
Robert	2	3
Lily	0	7
David	1	2

In fact, we could use `set_index()` on *any* DataFrame using *any* column at *any* time.

Indexing Series and DataFrames is a very common task, and the different ways of doing it is worth remembering.

Converting back to a CSV, JSON, or SQL

So after extensive work on cleaning your data, you're now ready to save it as a file of your choice. Similar to the ways we read in data, pandas provides intuitive commands to save it:





```
df.to_sql('new_purchases', con)
```

When we save JSON and CSV files, all we have to input into those functions is our desired filename with the appropriate file extension. With SQL, we're not creating a new file but instead inserting a new table into the database using our `con` variable from before.

Let's move on to importing some real-world data and detailing a few of the operations you'll be using a lot.

Most important DataFrame operations

DataFrames possess hundreds of methods and other operations that are crucial to any analysis. As a beginner, you should know the operations that perform simple transformations of your data and those that provide fundamental statistical analysis.

Let's load in the IMDB movies dataset to begin:





We're loading this dataset from a CSV and des

Viewing your data

The first thing to do when opening a new dataset is print out a few rows to keep as a visual reference. We accomplish this with `.head()`:

```
movies_df.head()
```





Guardians of the Galaxy	1	Action, Adventure, Sci-Fi	Ai c are forced ...	Cooper, Zoe S...	
Prometheus	2	Adventure, Mystery, Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...
Split	3	Horror, Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...
Sing	4	Animation, Comedy, Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey, Reese Witherspoon, Seth Ma...
Suicide Squad	5	Action, Adventure, Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...





printing the bottom two rows.:

```
movies_df.tail(2)
```

OUT:

	Rank	Genre	Description	Director	Actors	
Title						
Search Party	999	Adventure,Comedy	A pair of friends embark on a mission to reuni...	Scot Armstrong	Adam Pally, T.J. Miller, Thomas Middleditch,Sh...	:
Nine Lives	1000	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	:

Typically when we load in a dataset, we like to view the first five or so rows to see what's under the hood. Here we can see the names of each column, the index, and examples of



Learn Machine Learning by Doing

Learn Now



Getting into about your data

.info() should be one of the very first commands.

```
movies_df.info()
```





```

Rank           1000 non-null int64
Genre          1000 non-null object
Description    1000 non-null object
Director       1000 non-null object
Actors         1000 non-null object
Year           1000 non-null int64
Runtime (Minutes) 1000 non-null int64
Rating          1000 non-null float64
Votes           1000 non-null int64
Revenue (Millions) 872 non-null float64
Metascore       936 non-null float64
dtypes: float64(3), int64(4), object(4)
memory usage: 93.8+ KB

```

Learn Data Science with DATAQUEST

`.info()` provides the essential details about your dataset, such as the number of rows and columns, the number of non-null values, what type of data is in each column, and how much memory your DataFrame is using.

Notice in our movies dataset we have some obvious missing values in the `Revenue` and `Metascore` columns. We'll look at how to handle those in a bit.



Learn Machine Learning by Doing

Learn Now



objects.

Another fast and useful attribute is `.shape`, wh

```
movies_df.shape
```

```
(1000, 11)
```

Learn Data Science with  DATAQUEST

Note that `.shape` has no parentheses and is a simple tuple of format (rows, columns). So we have **1000 rows** and **11 columns** in our movies DataFrame.

You'll be going to `.shape` a lot when cleaning and transforming data. For example, you might filter some rows based on some criteria and then want to know quickly how many rows were removed.

...



Learn Machine Learning by Doing

Learn Now



```
temp_df = movies_df.append(movies_df)
```

```
temp_df.shape
```

(2000, 11)

Learn Data Science with  DATAQUEST

Using `append()` will return a copy without affecting the original DataFrame. We are capturing this copy in `temp` so we aren't working with the real data.

Notice call `.shape` quickly proves our DataFrame rows have doubled.

Now we can try dropping duplicates:



Learn Machine Learning by Doing

Learn Now



(1000, 11)

Learn Data Science with  DATAQUEST

Just like `append()`, the `drop_duplicates()` method will also return a copy of your DataFrame, but this time with duplicates removed. Calling `.shape` confirms we're back to the 1000 rows of our original dataset.

It's a little verbose to keep assigning DataFrames to the same variable like in this example. For this reason, pandas has the `inplace` keyword argument on many of its methods. Using `inplace=True` will modify the DataFrame object in place:

```
temp_df.drop_duplicates(inplace=True)
```

Learn Data Science with  DATAQUEST

Now our `temp_df` will have the transformed data automatically.



Learn Machine Learning by Doing

Learn Now



- `last`: Drop duplicates except for the last c
- `False`: Drop all duplicates.

Since we didn't define the `keep` argument in the previous example it was defaulted to `first`. This means that if two rows are the same pandas will drop the second row and keep the first row. Using `last` has the opposite effect: the first row is dropped.

`keep`, on the other hand, will drop all duplicates. If two rows are the same then both will be dropped. Watch what happens to `temp_df`:

```
temp_df = movies_df.append(movies_df) # make a new copy  
  
temp_df.drop_duplicates(inplace=True, keep=False)  
  
temp_df.shape
```

(0, 11)



Learn Machine Learning by Doing

Learn Now



Column cleanup

Many times datasets will have verbose column names with symbols, upper and lowercase words, spaces, and typos. To make selecting data by column name easier we can spend a little time cleaning up their names.

Here's how to print the column names of our dataset:

```
movies_df.columns
```

```
Index(['Rank', 'Genre', 'Description', 'Director', 'Actors', 'Year',
       'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
       'Metascore'],
      dtype='object')
```

Learn Data Science with  DATAQUEST



Learn Machine Learning by Doing

Learn Now



We can use the `.rename()` method to rename columns. We don't want parentheses, so let's rename those:

```
movies_df.rename(columns={  
    'Runtime (Minutes)': 'Runtime',  
    'Revenue (Millions)': 'Revenue_millions'  
}, inplace=True)
```

```
movies_df.columns
```

```
Index(['Rank', 'Genre', 'Description', 'Director', 'Actors', 'Year', 'Runtime',  
       'Rating', 'Votes', 'Revenue_millions', 'Metascore'],  
      dtype='object')
```

Learn Data Science with  DATAQUEST

Excellent. But what if we want to lowercase all names? Instead of using `.rename()` we could also set a list of names to the columns like so:



Learn Machine Learning by Doing

Learn Now



```
movies_df.columns
```

```
Index(['rank', 'genre', 'description', 'director', 'actors', 'year', 'runtime',
       'rating', 'votes', 'revenue_millions', 'metascore'],
       dtype='object')
```

Learn Data Science with  DATAQUEST

But that's too much work. Instead of just renaming each column manually we can do a list comprehension:

```
movies_df.columns = [col.lower() for col in movies_df]
```

```
movies_df.columns
```



Learn Machine Learning by Doing

Learn Now



`list` (and `dict`) comprehensions come in handy a lot when working with pandas and data in general.

It's a good idea to lowercase, remove special characters, and replace spaces with underscores if you'll be working with a dataset for some time.

How to work with missing values

When exploring data, you'll most likely encounter missing or null values, which are essentially placeholders for non-existent values. Most commonly you'll see Python's `None` or NumPy's `np.nan`, each of which are handled differently in some situations.

There are two options in dealing with nulls:

1. Get rid of rows or columns with nulls
2. Replace nulls with non-null values, a technique known as **imputation**





movies_df.isnull()

OUT:

	rank	genre	description	director	actors	year	runtime	rating	
Title									
Guardians of the Galaxy	False	False	False	False	False	False	False	False	F
Prometheus	False	False	False	False	False	False	False	False	F
Split	False	False	False	False	False	False	False	False	F
Sing	False	False	False	False	False	False	False	False	F
Suicide Squad	False	False	False	False	False	False	False	False	F

Notice `isnull()` returns a DataFrame where each cell is either True or False depending on that cell's null status.

To count the number of nulls in each column we use an aggregate function for summing:



Learn Machine Learning by Doing

Learn Now



```
rank          0
genre         0
description   0
director      0
actors        0
year          0
runtime       0
rating        0
votes         0
revenue_millions 128
metascore     64
dtype: int64
```

Learn Data Science with  DATAQUEST

`.isnull()` just by itself isn't very useful, and is usually used in conjunction with other methods, like `sum()`.

We can see now that our data has **128** missing values for `revenue_millions` and **64** missing values for `metascore`.



Learn Machine Learning by Doing

Learn Now



Remove nulls is pretty simple:

```
movies_df.dropna()
```

Learn Data Science with  DATAQUEST

This operation will delete any **row** with at least a single null value, but it will return a new DataFrame without altering the original one. You could specify `inplace=True` in this method as well.

So in the case of our dataset, this operation would remove 128 rows where `revenue_millions` is null and 64 rows where `metascore` is null. This obviously seems like a waste since there's perfectly good data in the other columns of those dropped rows. That's why we'll look at imputation next.

Other than just dropping rows, you can also drop columns with null values by setting `axis=1`:

```
movies_df.dropna(axis=1)
```



Learn Machine Learning by Doing

Learn Now



Intuition

What's with this `axis=1` parameter?

It's not immediately obvious where `axis` comes from and why you need it to be 1 for it to affect columns. To see why, just look at the `.shape` output:

```
movies_df.shape
```

```
Out: (1000, 11)
```

As we learned above, this is a tuple that represents the shape of the DataFrame, i.e. 1000 rows and 11 columns. Note that the *rows* are at index zero of this tuple and *columns* are at index one of this tuple. This is why `axis=1` affects columns. This comes from NumPy, and is a great example of why learning NumPy is worth your time.





chunk from your dataset, so instead we can im
mean or the **median** of that column.

Let's look at imputing the missing values in the `revenue_millions` column. First we'll extract that column into its own variable:

```
revenue = movies_df['revenue_millions']
```

Learn Data Science with  DATAQUEST

Using square brackets is the general way we select columns in a DataFrame.

If you remember back to when we created DataFrames from scratch, the keys of the `dict` ended up as column names. Now when we select columns of a DataFrame, we use brackets just like if we were accessing a Python dictionary.

`revenue` now contains a Series:

```
revenue.head()
```



Learn Machine Learning by Doing

Learn Now



```
Split           138.12
Sing            270.32
Suicide Squad  325.02
Name: revenue_millions, dtype: float64
```

Learn Data Science with  DATAQUEST

Slightly different formatting than a DataFrame, but we still have our [Title](#) index.

We'll impute the missing values of revenue using the mean. Here's the mean value:

```
revenue_mean = revenue.mean()

revenue_mean
```

```
82.95637614678897
```

Learn Data Science with  DATAQUEST



Learn Machine Learning by Doing

[Learn Now](#)



We have now replaced all nulls in `revenue` with 0. But what about other columns? If we run the same code using `inplace=True` we have actually affected the original DataFrame.

```
movies_df.isnull().sum()
```

```
rank          0
genre         0
description   0
director      0
actors        0
year          0
runtime       0
rating        0
votes          0
revenue_millions  0
metascore     64
dtype: int64
```

Learn Data Science with  DATAQUEST



Learn Machine Learning by Doing

Learn Now



and impute the nulls in each genre with that genre's mean.

Let's now look at more ways to examine and understand our variables.

Understanding your variables

Using `describe()` on an entire DataFrame we can get a summary of the distribution of continuous variables:

```
movies_df.describe()
```





mean	500.500000	2012.783000	113.17			
std	288.819436	3.205962	18.81			
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	

Understanding which numbers are continuous also comes in handy when thinking about the type of plot to use to represent your data visually.

`.describe()` can also be used on a categorical variable to get the count of rows, unique count of categories, top category, and freq of top category:

```
movies_df['genre'].describe()
```



Learn Machine Learning by Doing

Learn Now



```
freq          50  
Name: genre, dtype: object
```

Learn Data Science with  DATAQUEST

This tells us that the genre column has 207 unique values, the top value is Action/Adventure/Sci-Fi, which shows up 50 times (freq).

`.value_counts()` can tell us the frequency of all values in a column:

```
movies_df['genre'].value_counts().head(10)
```





```
Comedy           32
Drama, Romance   31
Action, Adventure, Fantasy 27
Comedy, Drama     27
Animation, Adventure, Comedy 27
Comedy, Romance    26
Crime, Drama, Thriller 24
Name: genre, dtype: int64
```

Learn Data Science with  DATAQUEST

Relationships between continuous variables

By using the correlation method `.corr()` we can generate the relationship between each continuous variable:

```
movies_df.corr()
```

Learn Data Science with  DATAQUEST

Learn Machine Learning by Doing

Learn Now



year	0.201000	0.000000				
runtime	-0.221739	-0.164900	.			
rating	-0.219555	-0.211219	.			
votes	-0.283876	-0.411904	0.407062	0.511537	1.000000	0.60794
revenue_millions	-0.252996	-0.117562	0.247834	0.189527	0.607941	1.00000
metascore	-0.191869	-0.079305	0.211978	0.631897	0.325684	0.13332

Correlation tables are a numerical representation of the bivariate relationships in the dataset.

Positive numbers indicate a positive correlation — one goes up the other goes up — and negative numbers represent an inverse correlation — one goes up the other goes down. 1.0 indicates a perfect correlation.

So looking in the first row, first column we see `rank` has a perfect correlation with itself, which is obvious. On the other hand, the correlation between `votes` and `revenue_millions` is 0.6. A little more interesting.





Let's now look more at manipulating DataFrame

DataFrame slicing, selecting, extracting

Up until now we've focused on some basic summaries of our data. We've learned about simple column extraction using single brackets, and we imputed null values in a column using `fillna()`. Below are the other methods of slicing, selecting, and extracting you'll need to use constantly.

It's important to note that, although many methods are the same, DataFrames and Series have different attributes, so you'll need be sure to know which type you are working with or else you will receive attribute errors.

Let's look at working with columns first.

By column

You already saw how to extract a column using square brackets like this:





```
pandas.core.series.Series
```

Learn Data Science with  DATAQUEST

This will return a *Series*. To extract a column as a *DataFrame*, you need to pass a list of column names. In our case that's just a single column:

```
genre_col = movies_df[['genre']]  
  
type(genre_col)
```

Learn Data Science with  DATAQUEST

```
pandas.core.frame.DataFrame
```

Learn Data Science with  DATAQUEST



Learn Machine Learning by Doing

Learn Now



OUT:

genr		
Title		
Guardians of the Galaxy	Action,Adventure,Sci-Fi	8.1
Prometheus	Adventure,Mystery,Sci-Fi	7.0
Split	Horror,Thriller	7.3
Sing	Animation,Comedy,Family	7.2
Suicide Squad	Action,Adventure,Fantasy	6.2

Now we'll look at getting data by rows.

By rows

For rows, we have two options:

- `loc` - locates by name



Learn Machine Learning by Doing

Learn Now



```
prom = movies_df.loc["Prometheus"]
```

```
prom
```

```
rank                                2
genre                               Adventure, Mystery, Sci-Fi
description             Following clues to the origin of mankind, a te...
director                            Ridley Scott
actors                             Noomi Rapace, Logan Marshall-Green, Michael Fa...
year                                2012
runtime                             124
rating                               7
votes                                485820
revenue_millions                  126.46
metascore                            65
Name: Prometheus, dtype: object
```

Learn Data Science with  DATAQUEST

On the other hand, with `iloc` we give it the numerical index of Prometheus:



Learn Machine Learning by Doing

Learn Now



`loc` and `iloc` can be thought of as similar to further, let's select multiple rows.

How would you do it with a list? In Python, just slice with brackets like `example_list[1:4]`. It's works the same way in pandas:

```
movie_subset = movies_df.loc['Prometheus':'Sing']

movie_subset = movies_df.iloc[1:4]

movie_subset
```





Prometheus	2	Adventure,Mystery,Sci-Fi	F C t C r....., a te...		
Split	3	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...
Sing	4	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...

One important distinction between using `.loc` and `.iloc` to select multiple rows is that `.loc` includes the movie *Sing* in the result, but when using `.iloc` we're getting rows 1:4 but the movie at index 4 (*Suicide Squad*) is not included.

Slicing with `.iloc` follows the same rules as slicing with lists, the object at the index at the end is not included.





Ridley Scott or films with a rating greater than

To do that, we take a column from the DataFra

Here's an example of a Boolean condition:

```
condition = (movies_df['director'] == "Ridley Scott")  
  
condition.head()
```

```
Title  
Guardians of the Galaxy    False  
Prometheus                  True  
Split                        False  
Sing                          False  
Suicide Squad                False  
  
Name: director, dtype: bool
```

Learn Data Science with  DATAQUEST



Learn Machine Learning by Doing

Learn Now

The internet's best courses on:

Data Science

Machine Learning

Python



Daily Learning ▾ Courses ▾ Team

549

2.6K

```
movies_df[movies_df['director'] == "Ridley
```



Learn Machine Learning by Doing

Learn Now



Prometheus	2	Adventure,Mystery,Sci-Fi	mankind, a te...		Michael F...	
The Martian	103	Adventure,Drama,Sci-Fi	An astronaut becomes stranded on Mars after hi...	Ridley Scott	Matt Damon, Jessica Chastain, Kristen Wiig, Ka...	2015
Robin Hood	388	Action,Adventure,Drama	In 12th century England, Robin and his band of...	Ridley Scott	Russell Crowe, Cate Blanchett, Matthew Macfady...	2010
American Gangster	471	Biography,Crime,Drama	In 1970s America, a detective works to bring d...	Ridley Scott	Denzel Washington, Russell Crowe, Chiwetel Ej...	2007
Exodus: Gods and	517	Action,Adventure,Drama	The defiant leader Moses	Ridley Scott	Christian Bale, Joel Edgerton, Ren	2014





Select `movies_df` where `movies_df['director'] == 'Quentin Tarantino'`

Let's look at conditional selections using numerical ratings:

```
movies_df[movies_df['rating'] >= 8.6].head(3)
```





Interstellar	37	Adventure,Drama,Sci-Fi	A e t r t v ...	Jessica Ch...	
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havoc...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...

We can make some richer conditionals by using logical operators `|` for "or" and `&` for "and".

Let's filter the DataFrame to show only movies by Christopher Nolan OR Ridley Scott:



The internet's best courses on:

Data Science

Machine Learning

Python



Daily Learning ▾ Courses ▾ Team

549

2.6K



Learn Machine Learning by Doing

Learn Now



Prometheus	2	Adventure,Mystery,Sci-Fi
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...
The Prestige	65	Drama,Mystery,Sci-Fi	Two stage magicians engage in competitive one-...	Christopher Nolan	Christian Bale, Hugh Jackman, Scarlett Johanss...
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...



The internet's best courses on:

Data Science

Machine Learning

Python



Daily Learning ▾ Courses ▾ Team

549

2.6K

```
movies_df[movies_df['director'].isin(['Chris', 'Judd', 'Peter'])]
```



Learn Machine Learning by Doing

Learn Now



Prometheus	2	Adventure,Mystery,Sci-Fi
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...
The Prestige	65	Drama,Mystery,Sci-Fi	Two stage magicians engage in competitive one-...	Christopher Nolan	Christian Bale, Hugh Jackman, Scarlett Johanss...
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...





```
movies_df[  
    ((movies_df['year'] >= 2005) & (movies_df['year'] <= 2010))  
    & (movies_df['rating'] > 8.0)  
    & (movies_df['revenue_millions'] < movies_df['revenue_millions'].quantile(0.25))  
]
```





3 Idiots	431	Comedy,Drama	Two friends are searching for their long lost ...	Joshi	
The Lives of Others	477	Drama,Thriller	In 1984 East Berlin, an agent of the secret po...	Florian Henckel von Donnersmarck	Ulrich Mühe, Martina Gedeck,Sebastian Koch, Ul...
Incendies	714	Drama,Mystery,War	Twins journey to the Middle East to discover t...	Denis Villeneuve	Lubna Azabal, Mélissa Désormeaux-Poulin, Maxim...
Taare Zameen Par	992	Drama,Family,Music	An eight-year-old boy is thought to be a lazy ...	Aamir Khan	Darsheel Safary, Aamir Khan, Tanay Chheda, Sac...





It is possible to iterate over a DataFrame or Series, especially on large datasets — is very slow.

An efficient alternative is to `apply()` a function to the dataset. For example, we could use a function to convert movies with an 8.0 or greater to a string value of "good" and the rest to "bad" and use this transformed values to create a new column.

First we would create a function that, when given a rating, determines if it's good or bad:

```
def rating_function(x):
    if x >= 8.0:
        return "good"
    else:
        return "bad"
```

Learn Data Science with  DATAQUEST

Now we want to send the entire rating column through this function, which is what `apply()` does:



Learn Machine Learning by Doing

Learn Now



OUT:

Title	rank	genre	desc	dir	cast	year	duration
Guardians of the Galaxy	1	Action, Adventure, Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121 min
Prometheus	2	Adventure, Mystery, Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fassbender, Charlize Theron,羿	2012	124 min

The `.apply()` method passes every value in the `rating` column through the `rating_function` and then returns a new Series. This Series is then assigned to a new column called `rating_category`.

You can also use anonymous functions as well. This lambda function achieves the same



Learn Machine Learning by Doing

Learn Now



OUT:

Title	rank	genre	desc. snippet	dir. notes	actrs.	year	rate
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	12.1
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael F...	2012	12.4

Overall, using `apply()` will be much faster than iterating manually over rows because pandas is utilizing vectorization.



Learn Machine Learning by Doing

Learn Now



Brief Plotting

Another great thing about pandas is that it integrates with Matplotlib, so you get the ability to plot directly off DataFrames and Series. To get started we need to import Matplotlib ([pip install matplotlib](#)):

```
import matplotlib.pyplot as plt  
plt.rcParams.update({'font.size': 20, 'figure.figsize': (10, 8)}) # set font and plot
```

Learn Data Science with  DATAQUEST

Now we can begin. There won't be a lot of coverage on plotting, but it should be enough to explore your data easily.

Plotting Tip



Learn Machine Learning by Doing

[Learn Now](#)

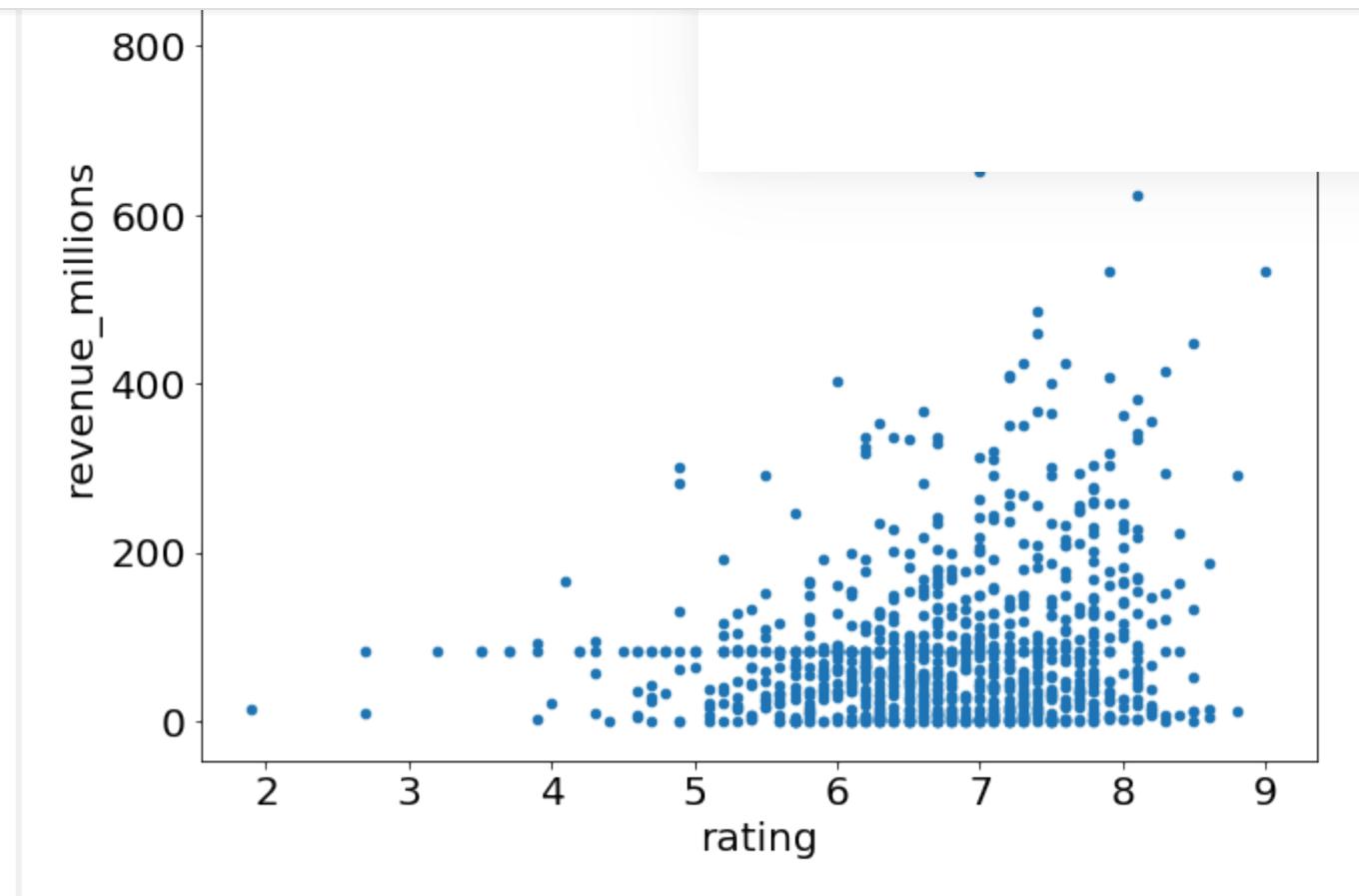


Let's plot the relationship between ratings and
on `movies_df` with some info about how to construct the plot.

```
movies_df.plot(kind='scatter', x='rating', y='revenue_millions', title='Revenue (millions) vs Rating')
```

Learn Data Science with  DATAQUEST





What's with the semicolon? It's not a syntax error, just a way to hide the
<matplotlib.axes._subplots.AxesSubplot at 0x26613b5cc18> output when plotting in Jupyter notebooks



Learn Machine Learning by Doing

Learn Now

The internet's best courses on:

Data Science

Machine Learning

Python



Daily Learning ▾ Courses ▾ Team

549

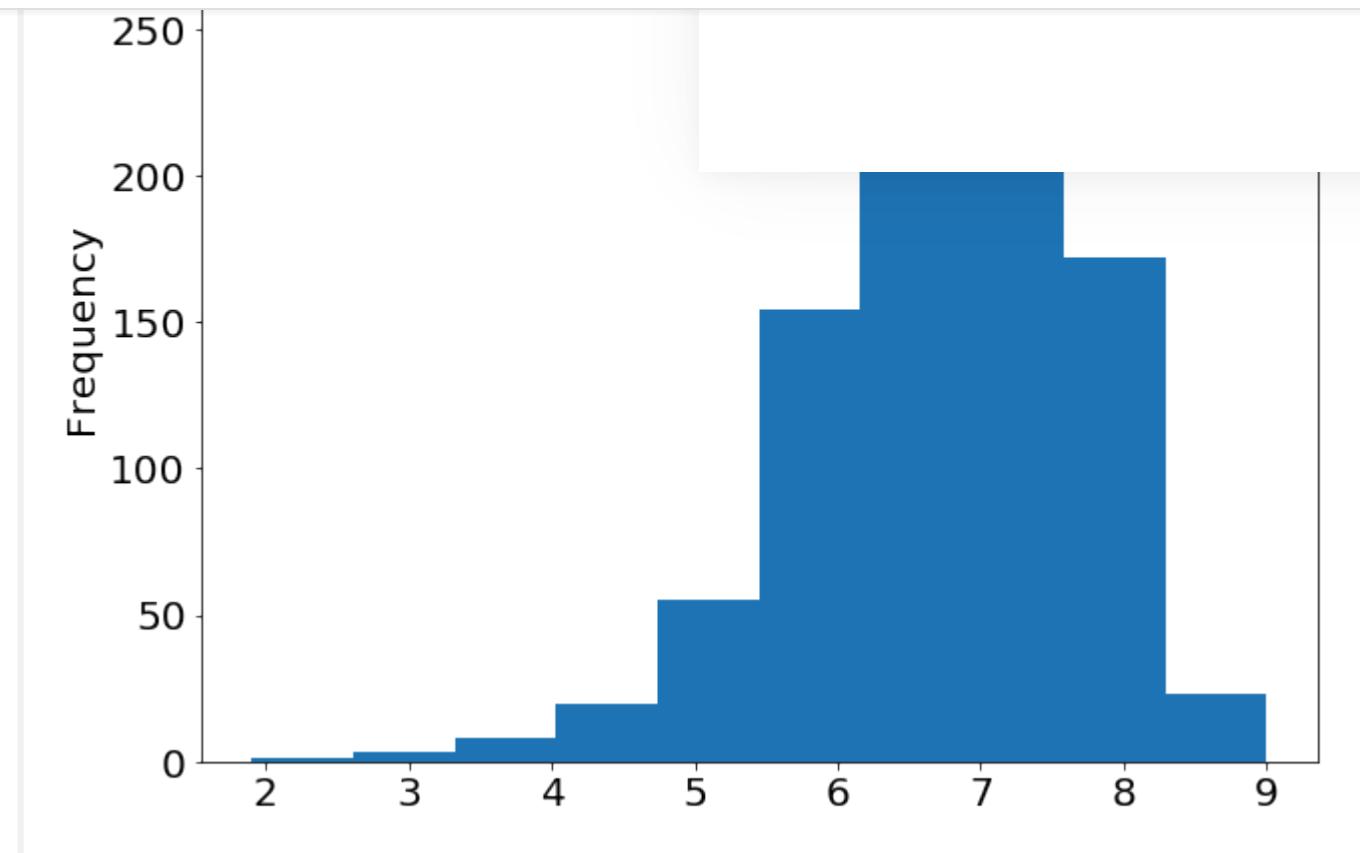
2.6K

```
movies_df['rating'].plot(kind='hist', tit]
```



Learn Machine Learning by Doing

Learn Now



Do you remember the `.describe()` example at the beginning of this tutorial? Well, there's a graphical representation of the interquartile range, called the Boxplot. Let's recall what `describe()` gives us on the ratings column:





```
count    1000.000000
mean     6.723200
std      0.945429
min     1.900000
25%    6.200000
50%    6.800000
75%    7.400000
max     9.000000
Name: rating, dtype: float64
```

Learn Data Science with  DATAQUEST

Using a Boxplot we can visualize this data:

```
movies_df['rating'].plot(kind="box");
```

Learn Data Science with  DATAQUEST

Learn Machine Learning by Doing

Learn Now

The internet's best courses on:

Data Science

Machine Learning

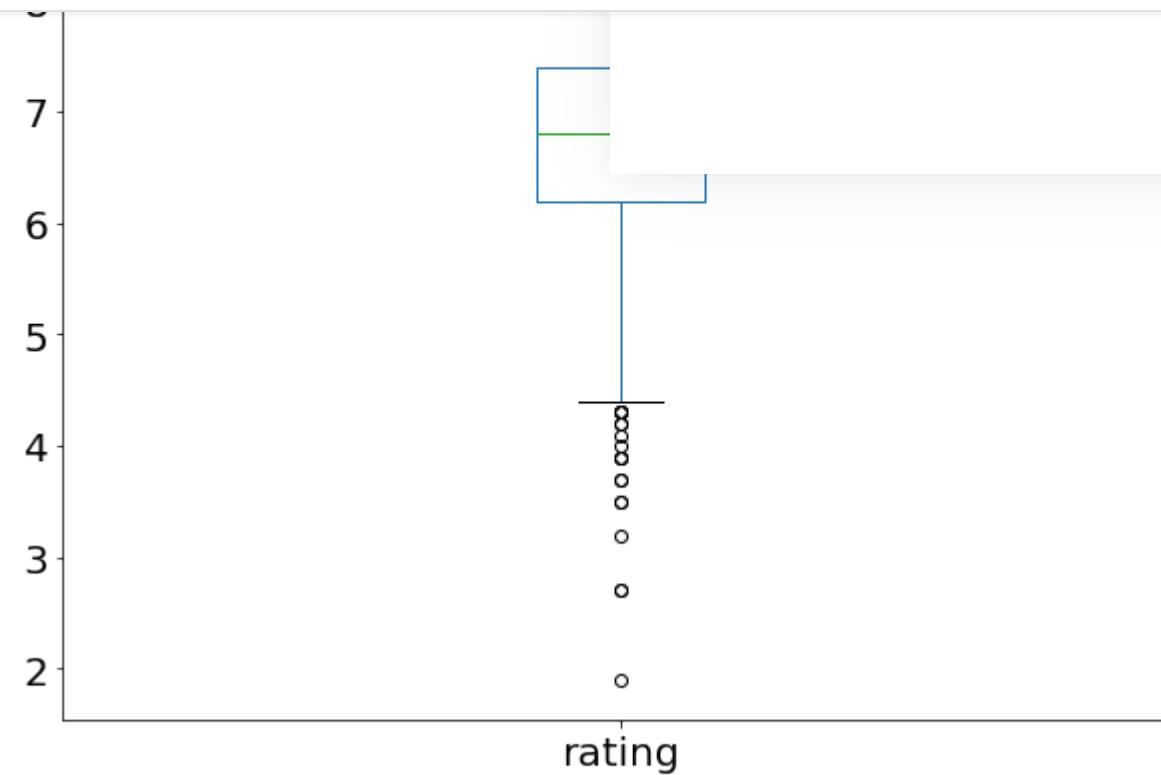
Python



Daily Learning ▾ Courses ▾ Team

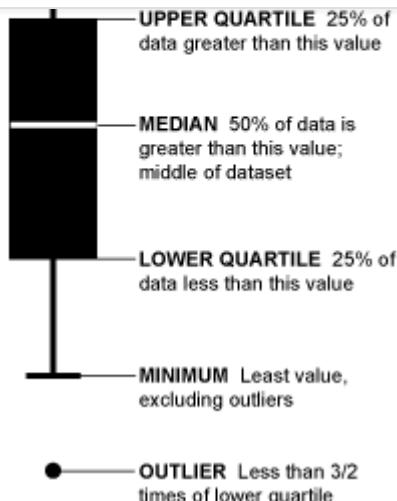
549

2.6K



Learn Machine Learning by Doing

Learn Now



Source: *Flowing Data*

By combining categorical and continuous data, we can create a Boxplot of revenue that is grouped by the Rating Category we created above:

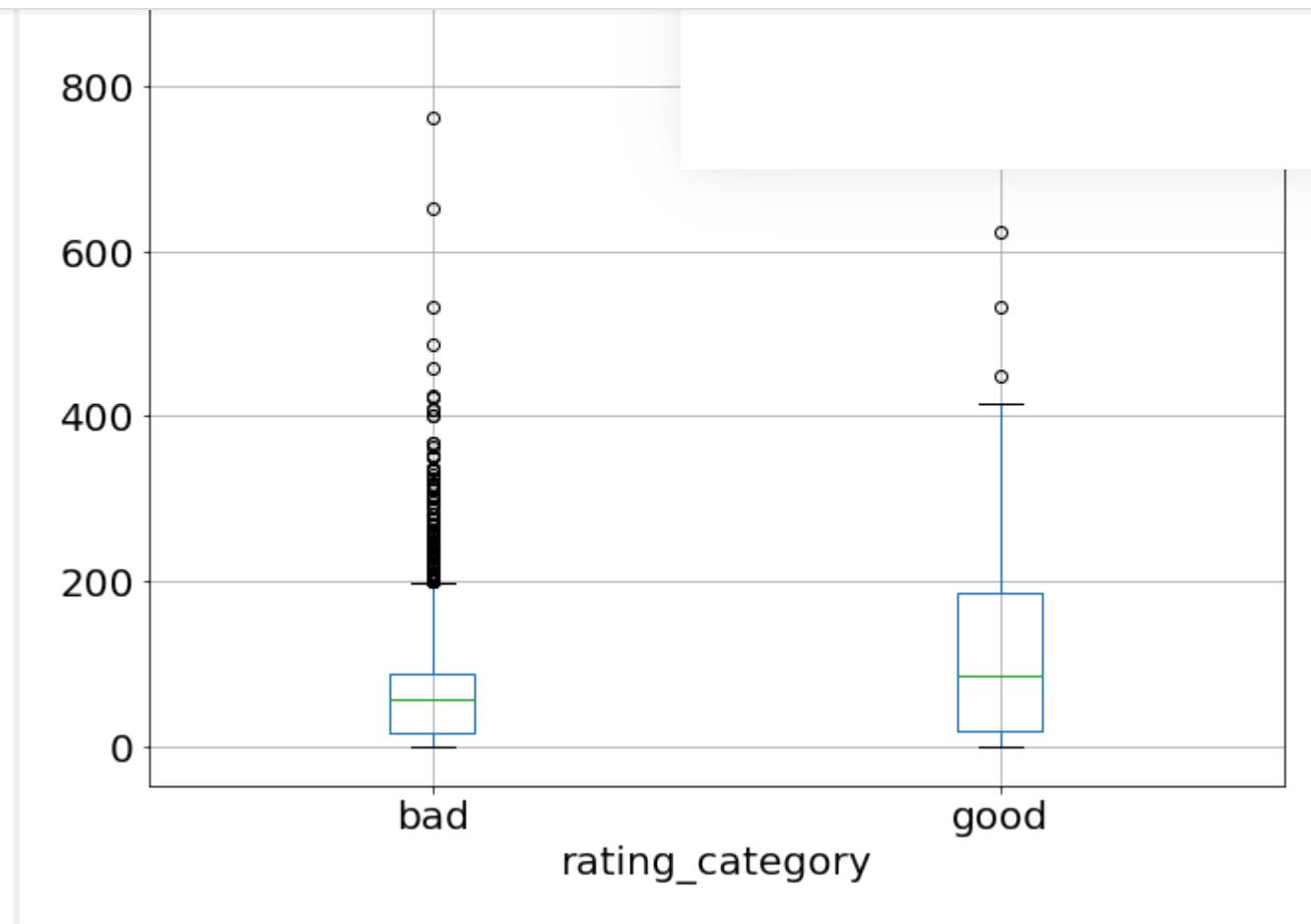
```
movies_df.boxplot(column='revenue_millions', by='rating_category');
```

Learn Data Science with DATAQUEST



Learn Machine Learning by Doing

Learn Now



That's the general idea of plotting with pandas. There's too many plots to mention, so definitely take a look at the `plot()` docs here for more information on what it can do.



Learn Machine Learning by Doing

Learn Now



essential skill in data science. Just cleaning wr Scientist. After a few projects and some practi of the basics.

To keep improving, view the extensive tutorials offered by the official pandas docs, follow along with a few Kaggle kernels, and keep working on your own projects!

Resources

[Applied Data Science with Python – Coursera](#)

Covers an intro to Python, Visualization, Machine Learning, Text Mining, and Social Network Analysis in Python. Also provides many challenging quizzes and assignments to further enhance your learning.

[Complete SQL Bootcamp – Udemy](#)

An excellent course for learning SQL. The instructor explains everything from beginner to advanced SQL queries and techniques, and



Learn Machine Learning by Doing

Learn Now



Get updates in your inbox

Join over 7,500 data science learners.

Enter your email

Subscribe

Meet the Authors



George McIntire

Data Scientist

Data Scientist and writer, currently working as a Data Visualization Analyst at Callisto Media



Learn Machine Learning by Doing

Learn Now

The internet's best courses on:

Data Science

Machine Learning

Python



Daily Learning ▾ Courses ▾ Team



Lauren Washington

Lead Data Scientist & ML Developer

Lead data scientist and machine learning developer at smartQED, and mentor at the Thinkful Data Science program.

[Back to blog index](#)

[Load Comments](#)



Learn Machine Learning by Doing

[Learn Now](#)



[Best Data Science Courses](#) [Best Machine Learning Courses](#) [Best Udemy Courses](#)

[Data Science & Machine Learning Glossary](#) [Free Data Science Books](#)

[Privacy Policy](#)

© 2024 LearnDataSci. All rights reserved.

Use of and/or registration on any portion of this site constitutes acceptance of our [Privacy Policy](#). The material on this site may not be reproduced, distributed, transmitted, cached or otherwise used, except with the prior written permission of LearnDataSci.com.



Learn Machine Learning by Doing

[Learn Now](#)