Chanin Nantasenamat Follow

Jun 30, 2021 · 12 min read · · Listen

Created (with license) using the image by mir_design from envato elements.

GETTING STARTED

How to Master Pandas for Data Science

Here's the Essential Pandas you Need for Data Science

Pandas is an open source Python library that allows the handling of tabular data (*i.e.* explore, clean and process). The term originated from the econometrics term *panel data* and thus **PAN(**el)-**DA(**ta)-**S**.

At a high-level, Pandas works very much like a spreadsheet (i.e. think Microsoft Excel or Google Sheets) as you work with rows and columns. Pandas serves as one of the

In this article, we will be exploring the essential bare minimal knowledge that you need in order to master Pandas for getting started in data science. As you will see Pandas is a feature-rich library and even the entire *Pandas documentation* may be a challenge to go through where it sits at a whopping 3,411 pages (at the time of this writing)!

1. The Need for Data Wrangling

A popular phrase commonly used in computer science and software engineering by George Fuechsel stated:

> "Garbage in, Garbage out."
>
> — George Fuechsel, IBM programmer

which explicitly says that no matter how good a program is, the generated output will be irrelevant if the input is of poor quality. This generalizes well in data science as the model will only be as good as the input data from which it is given. Therefore, data cleaning helps to process the data in such a way that it is free of missing values, have consistency in the data structures and values, etc.

Simply put, if you're working with tabular data then Pandas is the go-to library that you need for data wrangling.

2. Why Do We Need Pandas?

A typical workflow for the data science lifecycle can be summarized by the following illustration.

Once data has been collected, they may be stored in separate databases and the compilation of these data from heterogeneous data sources will have to be retrieved for usage in a data science project. Most of the effort in a data science project may be spent in the data cleaning phase as well as exploratory data analysis as both helps to provide high quality dataset (i.e. after intensive data cleaning) to work with as well as provide a high-level understanding of the data, which serves as great starting point for the formulation of hypothesis that can be subsequently validated through exploratory data analysis as well as from machine learning model building.

The Pandas library has a large set of features that will allow you to perform tasks from the first intake of raw data, its cleaning and transformation to the final curated form in order to produce high quality data (i.e. hopefully free of missing values and errors) for further hypothesis testing and validation via exploratory data analysis and machine learning.

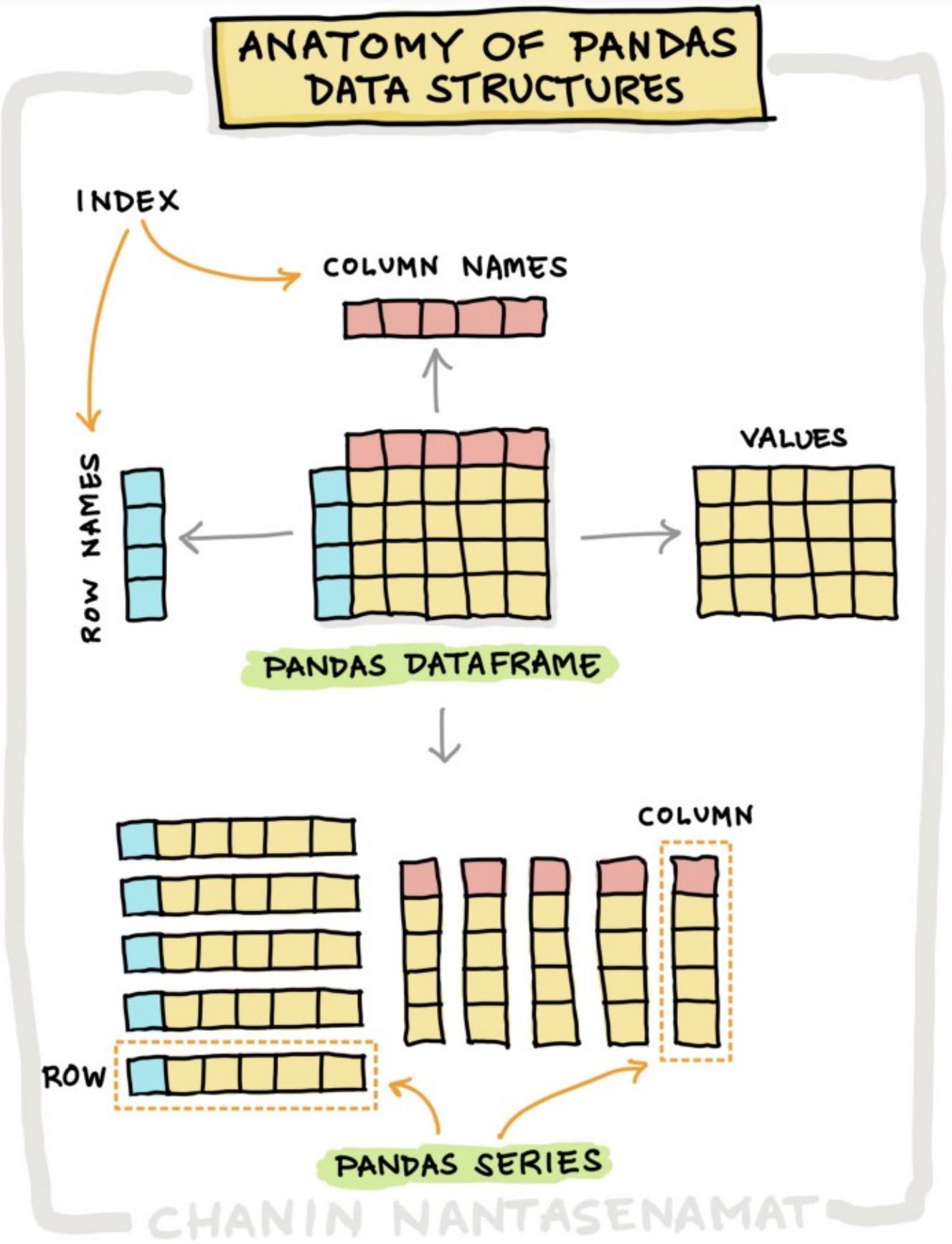
3. Basics of Pandas

3.1. Pandas Objects

Pandas allow us to work with tabular datasets. Let us take a look at the basic data structures of Pandas that consists of 3 types as follows (*i.e.* the first two are the data structures while the latter serves as a point of reference):

1. Series
2. DataFrame
3. Index

In the following illustration, I've summarized the basic anatomy of Pandas data structures. In a nutshell, you will see that a Pandas DataFrame and Pandas Series are labeled data (*i.e.* it has row names and column names). You'll also see that a Pandas DataFrame is a collection of Pandas Series (*i.e.* the individual columns and rows).



Anatomy of Pandas Data Structures. Drawn by Author.

Series can hold an integer, float, string, python object, etc. At a high-level, a Series can be thought of as a column in Microsoft Excel.

3.3. DataFrame

A Pandas ***DataFrame*** is a two-dimensional array. At a high-level, a DataFrame can be thought of as the spreadsheet in Microsoft Excel (*i.e.* a M × N matrix where M denotes the rows and N the columns).

3.4. Index

The ***index*** in Pandas is an inherent property of Series and DataFrame objects that serves as a point of reference as to which rows and/or columns to perform operations on (i.e. for a DataFrame) or the specific element in a Series to perform operations on. By default, Pandas automatically assigns index numbers starting from 0 to denote the row numbers or column numbers (*i.e.* if none are explicitly defined).

Index has two main features: (1) is an immutable array and (2) is an ordered set.

3.4.1. Immutatable

Simply put, by being immutable it means that we cannot easily modify the index value by simple assignment (*i.e.* as we would do for any other row of a DataFrame) as shown below

```
index_value = pd.Index([0,1,2,3,4,5])
index_value[0] = 10
```

which will produce an error:

```
TypeError: Index does not support mutable operations
```

The advantage of this is that we can be rest assured that the index would be kept intact during the coding process. However, if we do want to name the index, it can be done via the `pd.Index.rename()` function

As an array, we can retrieve the specific values in the index by performing a simple slicing as in:

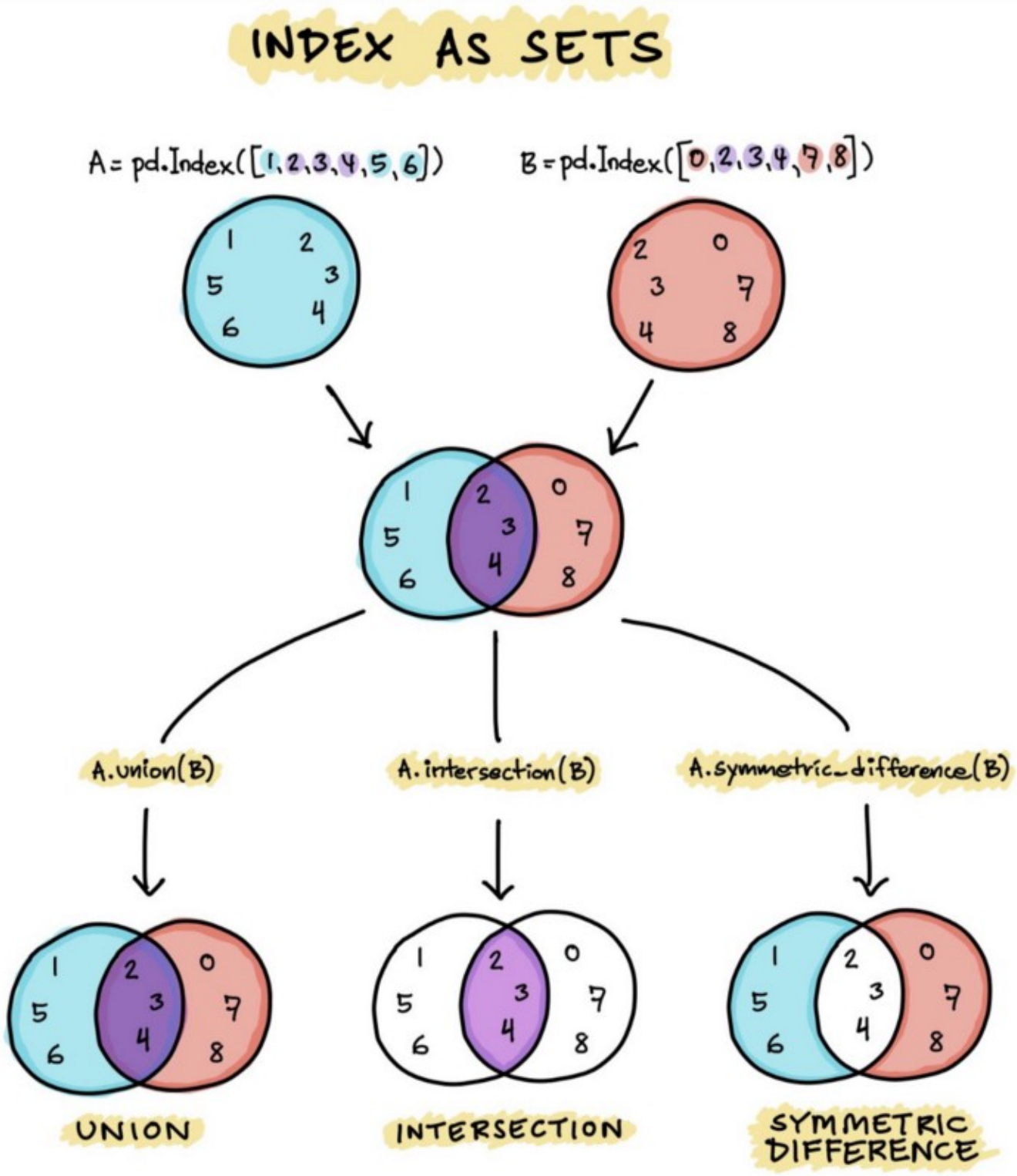
```
index_value = pd.Index([0,1,2,3,4,5])
index_value[0]
```

which will give:

```
0
```

3.4.3. Sets

The Pandas index works very much like the `set` data structure that allows you to apply set operations on it. Remember, how I mentioned earlier on that index can be used as a point of reference for your Series and DataFrame?

Index as Sets. Drawn by Author.

As such, you can compare and join (*i.e.* intersection, union, difference, symmetric difference, join and append) Pandas Series and DataFrame on the basis of their index.

For example, you can leverage Pandas index to figure out which elements are common

4. Use Cases for Pandas

In this section, we will be taking a look at a high-level on some of the functionalities that Pandas has to offer that you can use for your data science projects.

In a nutshell, here's a list of common tasks that you can perform using Pandas:

- Create data objects
- Load and save data
- Data inspection
- Missing data
- Index, select, add, modify and delete data
- Data filtering
- Merge and join data
- Reshape and pivot data
- Group data
- Sort data
- Summary statistics
- Data visualization
- Time series data
- Strings and categorical data

4.1. Pandas for Exploratory Data Analysis

Let's now take a look at Pandas in action for a typical exploratory data analysis (EDA).

In essence, EDA gives us a general understanding at the high-level of the dataset being


- Load the data from a CSV file (`pd.read_csv()`). Assign it to a variable such as `df` .
- Look at the dimension size of the data via `df.shape` .
- Explore the first or last few rows of the data to get the gist of the data (*i.e.* what variables we have and their data type, e.g. categorical or numerical) via `df.head()` or `df.tail()` .
- Look for missing values (*i.e.* via `pd.isna(df)`). Decide on whether to delete missing values or impute missing values (*i.e.* replacing it with an arbitrary value such as 0 or replacing it with the column's mean or median).
- Calculate summary statistics (*e.g.* min, max, mean, median, SD, etc.) for the entire data via `df.describe()` or on stratified data (*i.e.* grouping the data by a specific criteria followed by calculating the summary statistics; *e.g.* group the data by their class labels and for each class calculate the summary statistics) via the groupby function as shown here:
  `df.groupby('name_of_variable_used_as_criteria').describe()` .
- Create some basic data visualizations (*e.g.* bar plot, box plot, heat map, histograms, etc.) for the entire data or on stratified data via the `df.plot.bar()` function.
- Write a report to summarize some of the preliminary findings and insights that we get out of this EDA analysis. This will also help us figure out if the data being analyzed is sufficient or if we will need to collect additional data.

4.2. Pandas as precursor for Statistical Analysis and Machine Learning

As mentioned in the previous section, some preliminary insights gained from the EDA analysis serves as a great starting point that can help point us in the right direction for more in-depth statistical analysis and machine learning model building.

Statistical analysis could entail the comparison of different subsets of the data that could be previously obtained by performing ***groupby*** operations via Pandas. A popular statistical package in Python is the *statsmodels* library and there's an emerging and quite robust library called *pingouin*, which I've reviewed on my YouTube channel in the video *How to easily perform statistical analysis in Python with the Pengouin library*


- Load the data from a CSV file (`pd.read_csv()`).
- Look at the dimension size of the data via `df.shape` .
- Inspect the data by examining the first or last few rows via `df.head()` or `df.tail()` functions.
- Look for missing values (*i.e.* via `pd.isna(df)`). Decide on whether to delete missing values or impute missing values (*i.e.* replacing it with an arbitrary value such as 0 or replacing it with the column's mean or median).
- Assign contents from `df` to `X` and `Y` variables in order to prepare for model building via the scikit-learn library. This can be performed as shown below:

```
X = df.drop(['Y_variable'], axis=1)
Y = df['Y_variable]
```

- The `X` and `Y` variables are then used as input data to the scikit-learn library for data splitting as follows:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)
```

- A machine learning model is then trained on the generated training and test data via random forest followed by evaluation of their model performance metrics as follows:

```
from sklearn.ensemble import RandomForestClassifier

# 1. Define classifier
rf = RandomForestClassifier(n_estimators=10)

# 2. Train model
rf.fit(X_train, y_train)
```


```
# 4.1. Calculate Accuracy
rf_train_accuracy = accuracy_score(y_train, y_train_pred)

# 4.2. Calculate MCC
rf_train_mcc = matthews_corrcoef(y_train, y_train_pred)

# 4.3. Calculate F1-score
rf_train_f1 = f1_score(y_train, y_train_pred, average='weighted')


# 5. Test set performance

# 5.1. Calculate Accuracy
rf_test_accuracy = accuracy_score(y_test, y_test_pred)

# 5.2. Calculate MCC
rf_test_mcc = matthews_corrcoef(y_test, y_test_pred)

# Calculate F1-score
rf_test_f1 = f1_score(y_test, y_test_pred, average='weighted')
```

How to Build your First Machine Learning Model in Python

towardsdatascience.com

- Hyperparameter tuning could also be performed in order to boost the model performance by searching for the optimal set of hyperparameters.

How to Tune Hyperparameters of Machine Learning Models

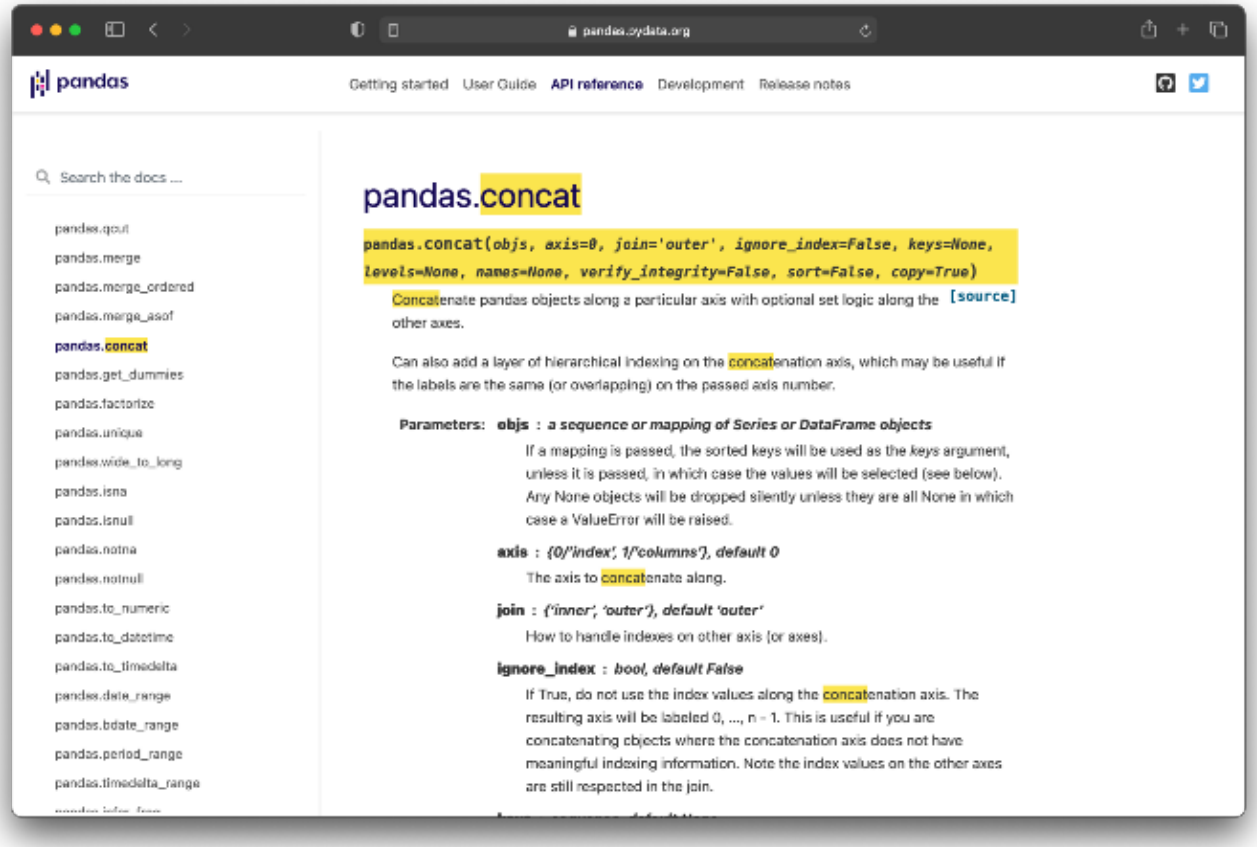
A Step-by-Step Tutorial using Scikit-learn

towardsdatascience.com

As briefly mentioned in the introductory paragraphs, the entire Pandas documentation is comprised of 3,411 pages. Particularly, the documentation is comprised of 4 parts: (1) *Getting started*, (2) *User guide*, (3) *API reference* and (4) *Developer guide*.

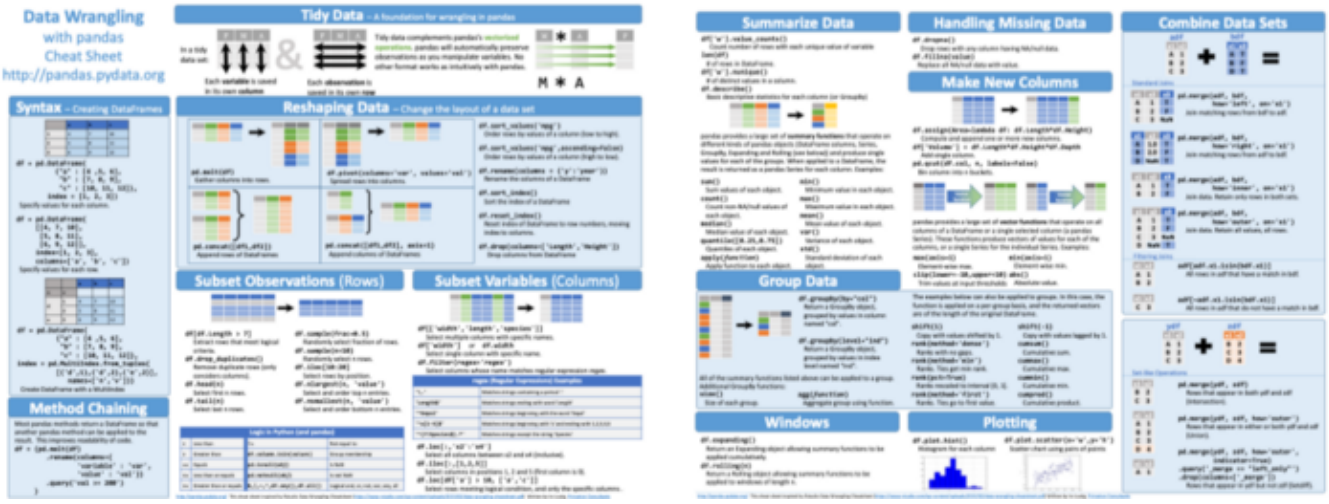
The official *Pandas User Guide* provides a comprehensive documentation with extensive examples to explain how each functions are used. These user guides are conveniently grouped together to common tasks spanning a wide range of topics. This is particularly useful if you somewhat know what you want to do. For example, if you know that you want more information about how to select data in Pandas then you can read the guide on *Indexing and selecting data*.

If you're super clear about a specific Pandas function that you wanted more information about you can take a dive into the API documentation. As the following screenshot shows you can see all of the input arguments that the `pd.concat()` function has and what each one means along with examples on how to use it.



Screenshot showing the API documentation for a specific function of interest.

Wrangling Cheatsheet.



Pandas cheat sheet created by Irv Lustig at Princeton Consultant.

A Pandas cheat sheet created by DataQuest is available as a webpage version as well as a downloadable version. The functions are conveniently grouped by tasks and will allow quick implementation if you have a general understanding of what each task group or task allows you to do.

Pandas cheat sheet created by DataQuest.

DataCamp has also created a Pandas cheat sheet summarizing all the functions that is also grouped according to tasks.

Pandas cheat sheet created by DataCamp.

Enthought not only group Pandas functions in this Pandas cheat sheet but they also provided schematic illustrations to help understand the various Pandas operations in this lengthiest cheat sheet at 8 pages.

Pandas cheat sheet created by Enthought.

Recommended books on Pandas.

- ***Python for Data Analysis:*** *Data Wrangling with Pandas, NumPy, and IPython (2nd Edition)*
  By Wes McKinney
  Wes McKinney is the creator of Pandas and the previous edition of this book is one of the first books to provide coverage on Pandas. Aside from Pandas, other libraries important for data science (e.g. NumPy, Matplotlib and scikit-learn) are also covered.

- ***Python Data Science Handbook:*** *Essential Tools for Working with Data*
  By Jake VanderPlas
  Jake VanderPlas' book has a chapter dedicated to Pandas which provides a great and concise coverage of the essentials of Pandas as applied to data science.

- ***Hands-On Data Analysis with Pandas:*** *A Python data science handbook for data collection, wrangling, analysis, and visualization (2nd Edition)*
  By Stefanie Molin
  The book covers the entire span of the data science process from data wrangling with Pandas to data visualization with Matplotlib and Seaborn as well as model building with Scikit-learn.

- ***Pandas 1.x Cookbook:*** *Practical recipes for scientific computing, time series analysis, and exploratory data analysis using Python (2nd Edition)*
  By Matt Harrison and Theodore Petrou
  The book is the first to cover Pandas 1.x and nearly all contents of the book is focused only on Pandas with a pinch of data visualization libraries (e.g. Matplotlib and Seaborn) as applied to the visualization of data prepared from Pandas.

- ***Pandas for Everyone:*** *Python Data Analysis*
  By Daniel Chen

Conclusion

Pandas is a core component in any data science workflow. To a beginner starting out, learning to use Pandas may seem like a formidable task owing to large collection of Pandas functions that are available. It is hoped that this article may serve as a blueprint that readers can refer to in using Pandas for their data projects.

In summary, we have taken a high-level overview of the Pandas library, considered the importance of data wrangling, learned about the basic data structures in Pandas, considered the use cases of Pandas for exploratory data analysis and machine learning model building as well as explored the available learning resources out there for learning more about Pandas.

Disclosure

- As an Amazon Associate, I may earn from qualifying purchases, which goes into helping the creation of future contents.

Read These Next

- **How to Master Python for Data Science**
  Here's the Essential Python you Need for Data Science
- **How to Build an AutoML App in Python**
  Step-by-Step Tutorial using the Streamlit Library
- **Strategies for Learning Data Science**
  Practical Advice for Breaking into Data Science
- **How to Build a Simple Portfolio Website for FREE**
  Step-by-step tutorial from scratch in less than 10 minutes


About Me

I work full-time as an Associate Professor of Bioinformatics and Head of Data Mining and Biomedical Informatics at a Research University in Thailand. In my after work hours, I'm a YouTuber (AKA the Data Professor) making online videos about data science. In all tutorial videos that I make, I also share Jupyter notebooks on GitHub (Data Professor GitHub page).

Data Professor

Data Science, Machine Learning, Bioinformatics, Research and Teaching are my passion. The Data Professor YouTube...

www.youtube.com

Connect with Me on Social Network

✅ YouTube: http://youtube.com/dataprofessor/
✅ Website: http://dataprofessor.org/ (Under construction)
✅ LinkedIn: https://www.linkedin.com/company/dataprofessor/
✅ Twitter: https://twitter.com/thedataprof/
✅ FaceBook: http://facebook.com/dataprofessor/
✅ GitHub: https://github.com/dataprofessor/
✅ Instagram: https://www.instagram.com/data.professor/