Video

Bookmark this page

Histograms

different

types of wheat.

as a histogram which demonstrates that to you.

So even as they are, I think you probably still would need to do

a little bit of feature scaling

3:23 / 3:23 1.25x HD

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Histograms are one of the The Seven Basic Tools of Quality, graphical techniques which have been identified as being most helpful for troubleshooting issues. Histograms help you understand the distribution of a feature in your dataset. They accomplish this by

simultaneously answering the questions *where* in your feature's domain your records are located at, and *how many* records exist there. Coincidentally, these two questions are also answered by the `.unique()` and `.value_counts()` methods discussed in the feature wrangling section, but in a graphical way. Be sure to take note of this in the exploring section of your course map!

NOTE: To follow the instruction below, download and extract the following zip file.

Since the wheat dataset was used to explore histograms in the video lecture, and since you're going to get intimately familiar with that dataset with the labs, in this reading we'll keep it fresh by using a different dataset about student grades, conveniently downloaded to your /DAT210x/Datasets/**students.data**.

Recall from the Features Primer section that there are two types of features: continuous and categorical. Histograms are only really meaningful with categorical data. If you have a continuous feature, it must first be *binned* or discretized by transforming the continuous feature into a categorical one by grouping similar values together. To accomplish this, the entire range values is divided into a series of intervals that are usually consecutive, equal in length, and non-overlapping. These intervals will become the *categories*. Then, a count of how many values fall into each interval serves as the categorical bin count.

To render a histogram with MatPlotLib through Pandas, call the `.plot.hist()` method on either a dataframe or series. The `.plot.hist()` method fully handles the discretization of your continuous features for you behind the scenes as needed!

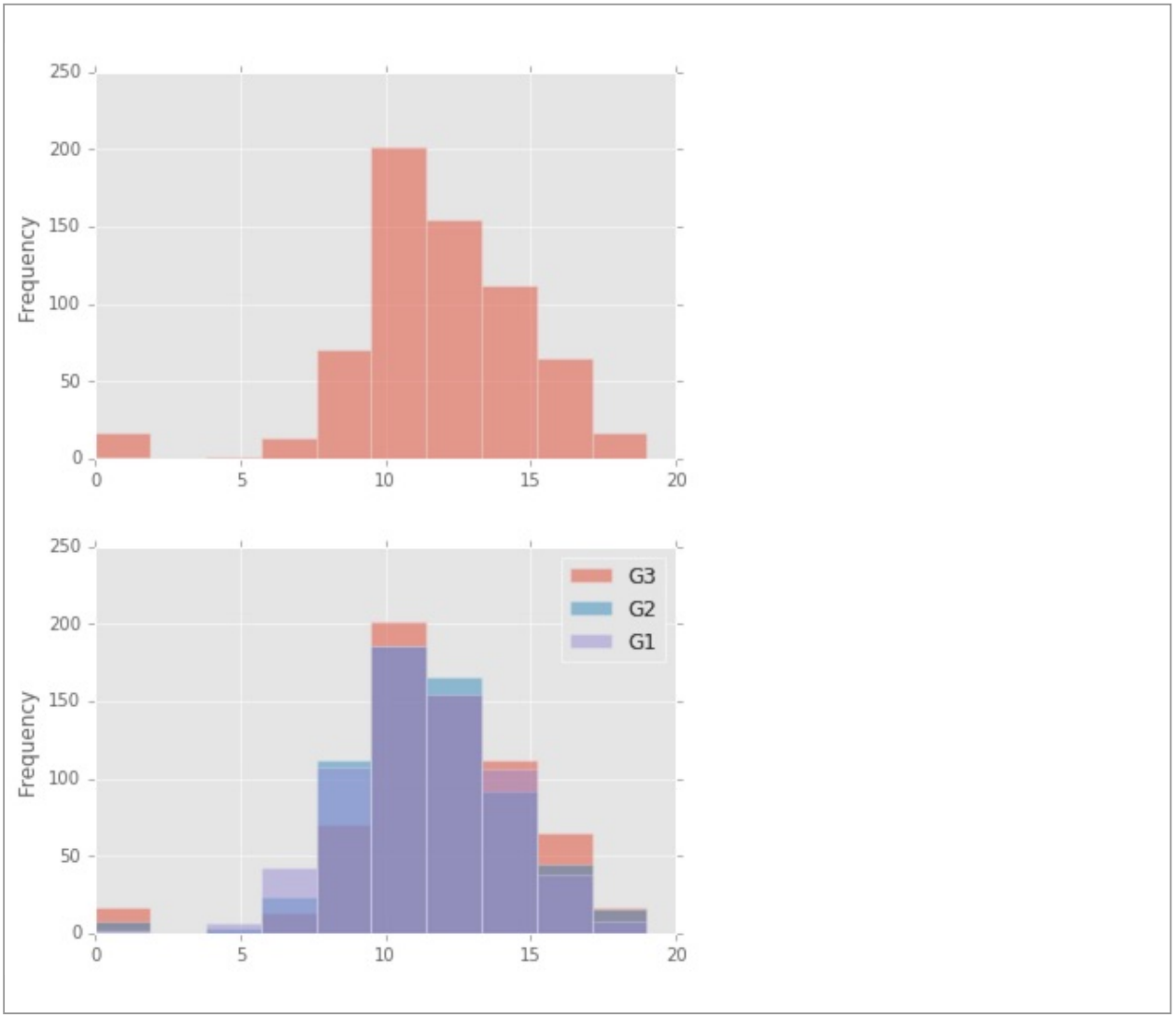
```
import pandas as pd
import matplotlib
```

```
matplotlib.style.use('ggplot') # Look Pretty
# If the above line throws an error, use plt.style.use('ggplot') instead

student_dataset = pd.read_csv("/Datasets/students.data", index_col=0)

my_series = student_dataset.G3
my_dataframe = student_dataset[['G3', 'G2', 'G1']]

my_series.plot.hist(alpha=0.5)
my_dataframe.plot.hist(alpha=0.5)
```



If your interest lies in probabilities per bin rather than frequency counts, set the named parameter `normed=True`, which will normalize your results as percentages. MatPlotLib's online API documentation exposes many other features and optional parameters that can be used with histograms, such as `cumulative` and `histtype`. Be sure to experiment with them in the exercises so that you can use histograms effectively as needed.

Knowing how a feature is distributed throughout your dataset is useful, as some machine learning models expect that, and only work when, the provided data is normally (Gaussian) distributed! For such models, if exploring your data with a histogram proves your data is skewed, all hope isn't lost. There are transformation techniques that will correct for this, and you'll learn about them in the Transforming and Munging module.

English