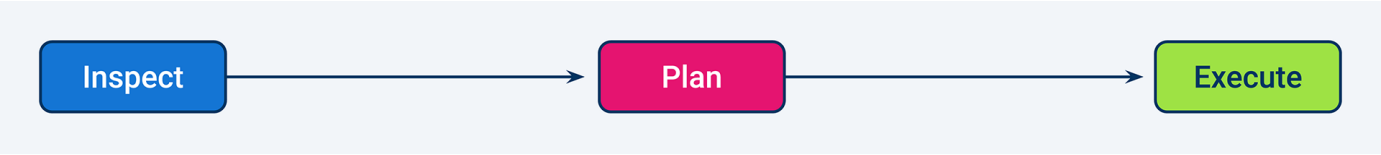
8.3.2

Using the Iterative Process for Cleaning Data

You and Britta feel confident in your data cleaning strategy. You're going to follow an iterative process that's based on three key steps: inspect, plan, and execute.



Specifically, we break down the iterative process for cleaning data as follows:

1. **Inspect:** We inspect the data and identify problem areas.
2. **Plan:** We make a plan to fix the problems. We then decide if executing the plan is worth the needed time and effort.
3. **Execute:** We execute the plan.

What makes this process iterative is that we might bounce among the steps. For example, we might inspect the data, make a plan, realize that we need to further inspect the data, execute the plan, realize that a step was missed, and then need to quickly rework the plan. That's why we offer these steps as a descriptive and not a prescriptive strategy. Cleaning data can be a long, laborious, and sometimes messy process. That's why a best practice is to document in detail every step of both our thought process and our actions.

IMPORTANT

As mentioned earlier, you should document your data cleaning assumptions, decisions, and motivations for those decisions. Transforming a messy dataset into a clean one is an iterative process. So as you clean one part of the data, you might reveal something messy in another part. Sometimes, that means unwinding lots of work that you've already done and having to redo it with a slight change. Documenting why a particular step is necessary will show you how to redo it without introducing more errors.

Early iterations focus on making the data easier to investigate. This includes deleting obviously bad data, removing superfluous columns (like a column that has only one value or that's missing an overwhelming amount of data), removing duplicate rows, consolidating columns, and reshaping the data if necessary.

As the data becomes easier to investigate, the iterations focus on fixing the most obvious problems. Then, any more-subtle problems become noticeable.

As the iterations shift toward fixing the more-subtle problems, we might discover that an earlier step needs to change—and that all the subsequent iterations thus need to change, as well. Needing to undo our work is frustrating. But, we gain a better understanding of the data.

NOTE

In general, the earlier iterations try to handle big chunks of data at once. An example is removing columns and rows. Later iterations focus on smaller chunks of data. An example is parsing values.

Reaching a point where no more problems exist in the data is rare. Reaching a point where the work to fix any remaining problems isn't worth the amount of potentially recovered data is more likely. So after we document the remaining problems, we consider the transform phase finished.

Now that we know how to use the iterative process, let's review each of the three steps in detail.

Inspect

Before we can do anything to clean the data, we have to inspect it. The first thing that we want to know is whether the data was correctly imported. The simplest way to check this is to print the first few data points and examine the first few rows for irregularities. Examples include data that exists in the wrong columns, values that are all missing, column headers that don't make sense, and garbled characters.

If the data doesn't seem correct, we know that it wasn't correctly imported. But even if the beginning of the data seems correct, the import might have gone wrong in the middle of the process. In that case, the rest of the data might be affected.

So, it's a good practice to check the last few rows and a random sample of rows. We can also answer two simple questions about the data:

- Does it have a consistent structure (like a CSV table does), or is it unstructured (like a collection of email messages is)?
- How is each data point identified? That is, does an explicit, unique ID exist for each data point, or will we need to build one?

Most usable data contains too many data points for us to review every one, so we need to use techniques that tell us about the whole dataset.

First, we can count the number of data points or rows that exist. For structured data, we count the number of columns and the number of missing values in each column. If possible, we count the number of unique values in each column and how frequently each of those values appears. To determine if that's possible, we need to investigate the data types for each column.

When investigating the data type for a column, we want to know both what the data type is and what it should be. For example, if a column has "True" and "False" values, we'll expect the data type to be Boolean. If the data type is instead a string, we'll need to further investigate.

If the data type of a column is numeric, we'll be able to summarize its data with some basic statistics. These include measures of central tendency (like the mean and median) and measures of spread (like the standard deviation, interquartile range, minimum, and maximum). We can also investigate columns by using statistical plots, like scatter plots and histograms.

Plan

After investigating the data and identifying problem areas, we can make a plan about how to fix the problems. This requires clearly articulating the problems—even if that means just expressing the problems to ourselves. We then make decisions about how to modify the data and fix the problems. In this phase, we answer several questions:

- If a column doesn't have the correct data type, does that apply to the whole column or to just a few rows?
- If rows have outliers, are the data points real, data entry errors, or natural variation?
- If values are missing, will we need to remove, replace, or interpolate them?

The answers tell us how to modify our data. Keep in mind that we have two main ways to do so: modifying the data values and modifying the data structure.

Modifying the data values includes removing rows or columns, replacing values, or generating new columns from old ones. For example, we might remove rows that have missing or corrupted data, columns that have only one value, or columns that are mostly missing data.

We might replace the data in many ways. For example, instead of dropping missing values, we might replace them with zeros or empty strings. If we have a column that contains nonstandard values, such as percentages that are stored as both whole numbers from 0 to 100 and fractions from 0 to 1, we might replace those values with ones that are all in a standard form. Converting a column to a new data type is also a way of replacing values. We might also bin data (like rounding to the nearest hundred) or replace numeric data (like income) with categorical data (like income brackets).

We might generate new columns by splitting an existing column into several new columns. An example is splitting an address column into street, city, state, and zip code columns. Or, we might calculate a new column from multiple existing columns. An example is calculating total prices by multiplying item prices by quantities.

Modifying the data structure includes pivoting the values of one column into multiple columns, aggregating rows, or merging multiple datasets. It can also include aggregating large amounts of data into either summary data or summary statistics.

With a plan of clearly stated steps to fix the problems, we can make an informed decision about whether implementing the plan is worth the effort. Sometimes, we have multiple viable options to choose from. In that case, we weigh the trade-offs and choose the best option.

Execute

Once we have a plan, it's time to execute it. So, we start writing code to fix the problems that we're focusing on.

As we do so, we might discover that a problem is more difficult than expected. This is a normal part of the process. So when writing the code, we try to take into account any unintended consequences that we might introduce.

After executing the changes, the next step is to inspect the data in a new iteration. This step is important, especially when modifying the data structure. That’s because doing so can introduce missing data points or inadvertently create more bad data.

Now that we understand the iterative process in detail, we’ll use it to clean the data from the crowdfunding DataFrames to create the new, transformed DataFrames. We’ll start with the category and subcategory DataFrames and then move on to the campaign and contacts DataFrames.