Exploratory Data Analysis

Contents

The third edition of *Think Stats* is available now from Bookshop.org and Amazon (those are affiliate links). If you are enjoying the free, online version, consider buying me a coffee.

The thesis of this book is we can use data to answer questions, resolve debates, and make better decisions.

This chapter introduces the steps we'll use to do that: loading and validating data, exploring, and choosing statistics that measure what we are interested in. As an example, we'll use data from the National Survey of Family Growth (NSFG) to answer a question I heard when my wife and I were expecting our first child: do first babies tend to arrive late?

Click here to run this notebook on Colab.

▶ Show code cell content

▶ Show code cell content

▶ Show code cell content

1.1. Evidence

You might have heard that first babies are more likely to be late. If you search the web with this question, you will find plenty of discussion. Some people claim it's true, others say it's a myth, and some people say it's the other way around: first babies come early.

In many of these discussions, people provide data to support their claims. I found many examples like these:

> "My two friends that have given birth recently to their first babies, BOTH went almost 2 weeks overdue before going into labour or being induced."

> "My first one came 2 weeks late and now I think the second one is going to come out two weeks early!!"

> "I don't think that can be true because my sister was my mother's first and she was early, as with many of my cousins."

Reports like these are called **anecdotal evidence** because they are based on data that is unpublished and usually personal. In casual conversation, there is nothing wrong with anecdotes, so I don't mean to pick on the people I quoted.

But we might want evidence that is more persuasive and an answer that is more reliable. By those standards, anecdotal evidence usually fails, because:

- Small number of observations: If pregnancy length is longer for first babies, the difference is probably small compared to natural variation. In that case, we might have to compare a large number of pregnancies to know whether there is a difference.
- Selection bias: People who join a discussion of this question might be interested because their first babies were late. In that case the process of selecting data would bias the results.
- Confirmation bias: People who believe the claim might be more likely to contribute examples that confirm it. People who doubt the claim are more likely to cite counterexamples.
- Inaccuracy: Anecdotes are often personal stories, and might be misremembered, misrepresented, repeated inaccurately, etc.

To address the limitations of anecdotes, we will use the tools of statistics, which include:

- Data collection: We will use data from a large national survey that was designed explicitly with the goal of generating statistically valid inferences about the U.S. population.
- Descriptive statistics: We will generate statistics that summarize the data concisely, and evaluate different ways to visualize data.
- Exploratory data analysis: We will look for patterns, differences, and other features that address the questions we are interested in. At the same time we will check for inconsistencies and identify limitations.
- Estimation: We will use data from a sample to estimate characteristics of the general population.
- Hypothesis testing: Where we see apparent effects, like a difference between two groups, we will evaluate whether the effect might have happened by chance.

By performing these steps with care to avoid pitfalls, we can reach conclusions that are more justified and more likely to be correct.

1.2. The National Survey of Family Growth

Since 1973 the U.S. Centers for Disease Control and Prevention (CDC) have conducted the National Survey of Family Growth (NSFG), which is intended to gather "information on family life, marriage and divorce, pregnancy, infertility, use of contraception, and men's and women's health. The survey results are used...to plan health services and health education programs, and to do statistical studies of families, fertility, and health."

You can read more about the NSFG at http://cdc.gov/nchs/nsfg.htm.

We will use data collected by this survey to investigate whether first babies tend to be born late, and other questions. In order to use this data effectively, we have to understand the design of the study.

In general, the goal of a statistical study is to draw conclusions about a **population**. In the NSFG, the target population is people in the United States aged 15-44.

Ideally surveys would collect data from every member of the population, but that's seldom possible. Instead we collect data from a subset of the population called a **sample**. The people who participate in a survey are called **respondents**.

The NSFG is a **cross-sectional** study, which means that it captures a snapshot of a population at a point in time. The NSFG has been conducted several times now; each deployment is called a **cycle**. We will use data from Cycle 6, which was conducted from January 2002 to March 2003.

In general, cross-sectional studies are meant to be **representative**, which means that the sample is similar to the target population in all ways that are important for the purposes of the study. That ideal is hard to achieve in practice, but people who conduct surveys come as close as they can.

The NSFG is not representative; instead it is **stratified**, which means that it deliberately **oversamples** some groups. The designers of the study recruited three groups – Hispanics, African-Americans and teenagers – at rates higher than their representation in the U.S. population, in order to make sure that the number of respondents in each group is large enough to draw valid conclusions. The drawback of oversampling is that it is not as easy to draw conclusions about the population based on statistics from the sample. We will come back to this point later.

When working with this kind of data, it is important to be familiar with the **codebook**, which documents the design of the study, the survey questions, and the encoding of the responses.

The codebook and user's guide for the NSFG data are available from http://www.cdc.gov/nchs/nsfg/nsfg_cycle6.htm

1.3. Reading the Data

Before downloading NSFG data, you have to agree to the terms of use:

> Any intentional identification or disclosure of an individual or establishment violates the assurances of confidentiality given to the providers of the information. Therefore, users will:
>
> - Use the data in this dataset for statistical reporting and analysis only.
> - Make no attempt to learn the identity of any person or establishment included in these data.
> - Not link this dataset with individually identifiable data from other NCHS or non-NCHS datasets.
> - Not engage in any efforts to assess disclosure methodologies applied to protect individuals and establishments or any research on methods of re-identification of individuals and establishments.

If you agree to comply with these terms, instructions for downloading the data are in the notebook for this chapter.

The data files are available directly from the NSFG web site at https://www.cdc.gov/nchs/data_access/ftp_dua.htm?url_redirect=ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Datasets/NSFG, but we will download them from the repository for this book, which provides a compressed version of the data file.

The following cells download the data files and install `statadict`, which we need to read the data.

```
download("https://github.com/AllenDowney/ThinkStats/raw/v3/data/2002FemPreg.dct")
download("https://github.com/AllenDowney/ThinkStats/raw/v3/data/2002FemPreg.dat.gz"
```

```
try:
    import statadict
except ImportError:
    %pip install statadict
```

The data is stored in two files, a "dictionary" that describes the format of the data, and a data file.

```
dct_file = "2002FemPreg.dct"
dat_file = "2002FemPreg.dat.gz"
```

The notebook for this chapter defines a function that reads these files. It is called `read_stata` because this data format is compatible with a statistical software package called Stata.

The following function takes these file names as arguments, reads the dictionary, and uses the results to read the data file.

```
from statadict import parse_stata_dict


def read_stata(dct_file, dat_file):
    stata_dict = parse_stata_dict(dct_file)
    resp = pd.read_fwf(
        dat_file,
        names=stata_dict.names,
        colspecs=stata_dict.colspecs,
        compression="gzip",
    )
    return resp
```

Here's how we use it.

```
preg = read_stata(dct_file, dat_file)
```

The result is a `DataFrame`, which is a Pandas data structure that represents tabular data in rows and columns. This `DataFrame` contains a row for each pregnancy reported by a respondent and a column for each **variable**. A variable can contain responses to a survey question or values that are calculated based on responses to one or more questions.

In addition to the data, a `DataFrame` also contains the variable names and their types, and it provides methods for accessing and modifying the data. The `DataFrame` has an attribute called `shape` that contains the number of rows and columns.

```
preg.shape
```

```
(13593, 243)
```

This dataset has 243 variables with information about 13,593 pregnancies. The `DataFrame` provides a method called `head` that displays the first few rows.

```
preg.head()
```

	caseid	pregordr	howpreg_n	howpreg_p	moscurrp	nowprgdk	pregend1	p
0	1	1	NaN	NaN	NaN	NaN	6.0	
1	1	2	NaN	NaN	NaN	NaN	6.0	
2	2	1	NaN	NaN	NaN	NaN	5.0	
3	2	2	NaN	NaN	NaN	NaN	6.0	
4	2	3	NaN	NaN	NaN	NaN	6.0	

5 rows × 243 columns

The left column is the index of the `DataFrame`, which contains a label for each row. In this case, the labels are integers starting from 0, but they can also be strings and other types.

The `DataFrame` has an attribute called `columns` that contains the names of the variables.

```
preg.columns
```

```
Index(['caseid', 'pregordr', 'howpreg_n', 'howpreg_p', 'moscurrp', 'nowprgdk',
       'pregend1', 'pregend2', 'nbrnaliv', 'multbrth',
       ...
       'poverty_i', 'laborfor_i', 'religion_i', 'metro_i', 'basewgt',
       'adj_mod_basewgt', 'finalwgt', 'secu_p', 'sest', 'cmintvw'],
      dtype='object', length=243)
```

The column names are contained in an `Index` object, which is another Pandas data structure. To access a column from a `DataFrame`, you can use the column name as a key.

```
pregordr = preg["pregordr"]
type(pregordr)
```

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

The result is a Pandas `Series`, which represents a sequence of values. `Series` also provides `head`, which displays the first few values and their labels.

```
pregordr.head()
```

Print to PDF

```
0    1
1    2
2    1
3    2
4    3
Name: pregordr, dtype: int64
```

The last line includes the name of the `Series` and `dtype`, which is the type of the values. In this example, `int64` indicates that the values are 64-bit integers.

The NSFG dataset contains 243 variables in total. Here are some of the ones we'll use for the explorations in this book.

- `caseid` is the integer ID of the respondent.
- `pregordr` is a pregnancy serial number: the code for a respondent's first pregnancy is 1, for the second pregnancy is 2, and so on.
- `prglngth` is the integer duration of the pregnancy in weeks.
- `outcome` is an integer code for the outcome of the pregnancy. The code 1 indicates a live birth.
- `birthord` is a serial number for live births: the code for a respondent's first child is 1, and so on. For outcomes other than live birth, this field is blank.

- `birthwgt_lb` and `birthwgt_oz` contain the pounds and ounces parts of the birth weight of the baby.
- `agepreg` is the mother's age at the end of the pregnancy.
- `finalwgt` is the statistical weight associated with the respondent. It is a floating-point value that indicates the number of people in the U.S. population this respondent represents.

If you read the codebook carefully, you will see that many of the variables are **recodes**, which means that they are not part of the **raw data** collected by the survey – they are calculated using the raw data.

For example, `prglngth` for live births is equal to the raw variable `wksgest` (weeks of gestation) if it is available; otherwise it is estimated using `mosgest * 4.33` (months of gestation times the average number of weeks in a month).

Recodes are often based on logic that checks the consistency and accuracy of the data. In general it is a good idea to use recodes when they are available, unless there is a compelling reason to process the raw data yourself.

1.4. Validation

When data is exported from one software environment and imported into another, errors might be introduced. And when you are getting familiar with a new dataset, you might decode data incorrectly or misunderstandings its meaning. If you invest time to validate the data, you can save time later and avoid errors.

One way to validate data is to compute basic statistics and compare them with published results. For example, the NSFG codebook includes tables that summarize each variable. Here is the table for `outcome`, which encodes the outcome of each pregnancy.

Value	Label	Total
1	LIVE BIRTH	9148
2	INDUCED ABORTION	1862
3	STILLBIRTH	120
4	MISCARRIAGE	1921
5	ECTOPIC PREGNANCY	190
6	CURRENT PREGNANCY	352
Total		13593

The “Total” column indicates the number of pregnancies with each outcome. To check these totals, we’ll use the `value_counts` method, which counts the number of times each value appears, and `sort_index`, which sorts the results according to the values in the `Index` (the left column).

```
preg["outcome"].value_counts().sort_index()
```

```
outcome
1    9148
2    1862
3     120
4    1921
5     190
6     352
Name: count, dtype: int64
```

Comparing the results with the published table, we can confirm that the values in `outcome` are correct. Similarly, here is the published table for `birthwgt_lb`.

Value	Label	Total
.	inapplicable	4449
0-5	UNDER 6 POUNDS	1125
6	6 POUNDS	2223
7	7 POUNDS	3049
8	8 POUNDS	1889
9-95	9 POUNDS OR MORE	799
97	Not ascertained	1
98	REFUSED	1
99	DON'T KNOW	57
Total		13593

Birth weight is only recorded for pregnancies that ended in a live birth. The table indicates that there are 4449 cases where this variable is inapplicable. In addition, there is one case where the question was not asked, one where the respondent did not answer, and 57 cases where they did not know.

Again, we can use `value_counts` to compare the counts in the dataset to the counts in the codebook.

```
counts = preg["birthwgt_lb"].value_counts(dropna=False).sort_index()
counts
```

```
birthwgt_lb
0.0        8
1.0       40
2.0       53
3.0       98
4.0      229
5.0      697
6.0     2223
7.0     3049
8.0     1889
9.0      623
10.0     132
11.0      26
12.0      10
13.0       3
14.0       3
15.0       1
51.0       1
97.0       1
98.0       1
99.0      57
NaN     4449
Name: count, dtype: int64
```

The argument `dropna=False` means that `value_counts` does not ignore values that are "NA" or "Not applicable". These values appear in the results as `NaN`, which stands for "Not a number" – and the count of these values is consistent with the count of inapplicable cases in the codebook.

The counts for 6, 7, and 8 pounds are consistent with the codebook. To check the counts for the weight range from 0 to 5 pounds, we can use an attribute called `loc` – which is short for "location" – and a slice index to select a subset of the counts.

```
counts.loc[0:5]
```

```
birthwgt_lb
0.0      8
1.0     40
2.0     53
3.0     98
4.0    229
5.0    697
Name: count, dtype: int64
```

And we can use the `sum` method to add them up.

```
counts.loc[0:5].sum()
```

```
np.int64(1125)
```

The total is consistent with the codebook.

The values 97, 98, and 99 represent cases where the birth weight is unknown. There are several ways we might handle missing data. A simple option is to replace these values with `NaN`. At the same time, we will also replace a value that is clearly wrong, 51 pounds.

We can use the replace method like this:

```
preg["birthwgt_lb"] = preg["birthwgt_lb"].replace([51, 97, 98, 99], np.nan)
```

The first argument is a list of values to be replaced. The second argument, `np.nan`, gets the `NaN` value from NumPy.

When you read data like this, you often have to check for errors and deal with special values. Operations like this are called **data cleaning**.

1.5. Transformation

As another kind of data cleaning, sometimes we have to convert data into different formats, and perform other calculations.

For example, `agepreg` contains the mother's age at the end of the pregnancy. According to the codebook, it is an integer number of centiyears (hundredths of a year), as we can tell if we use the `mean` method to compute its average.

```
preg["agepreg"].mean()
```

```
np.float64(2468.8151197039497)
```

To convert it to years, we can divide through by 100.

```
preg["agepreg"] /= 100.0
preg["agepreg"].mean()
```

```
np.float64(24.6881511970395)
```

Now the average is more credible.

As another example, `birthwgt_lb` and `birthwgt_oz` contain birth weights with the pounds and ounces in separate columns. It will be more convenient to combine them into as single column that contains weights in pounds and fractions of a pound.

First we'll clean `birthwgt_oz` as we did with `birthwgt_lb`.

```
preg["birthwgt_oz"].value_counts(dropna=False).sort_index()
```

```
birthwgt_oz
0.0     1037
1.0      408
2.0      603
3.0      533
4.0      525
5.0      535
6.0      709
7.0      501
8.0      756
9.0      505
10.0     475
11.0     557
12.0     555
13.0     487
14.0     475
15.0     378
97.0       1
98.0       1
99.0      46
NaN     4506
Name: count, dtype: int64
```

```
preg["birthwgt_oz"] = preg["birthwgt_oz"].replace([97, 98, 99], np.nan)
```

Now we can use the cleaned values to create a new column that combines pounds and ounces into a single quantity.

```
preg["totalwgt_lb"] = preg["birthwgt_lb"] + preg["birthwgt_oz"] / 16.0
preg["totalwgt_lb"].mean()
```

```
np.float64(7.265628457623368)
```

The average of the result seems plausible.

1.6. Summary Statistics

A **statistic** is a number derived from a dataset, usually intended to quantify some aspect of the data. Examples include the count, mean, variance, and standard deviation.

A `Series` object has a `count` method that returns the number of values that are not `nan`.

```
weights = preg["totalwgt_lb"]
n = weights.count()
n
```

```
np.int64(9038)
```

It also provides a `sum` method that returns the sum of the values – we can use it to compute the mean like this.

```
mean = weights.sum() / n
mean
```

```
np.float64(7.265628457623368)
```

But as we've already seen, there's also a `mean` method that does the same thing.

```
weights.mean()
```

```
np.float64(7.265628457623368)
```

In this dataset, the average birth weight is about 7.3 pounds.

Variance is a statistic that quantifies the spread of a set of values. It is the mean of the squared deviations, which are the distances of each point from the mean.

```
squared_deviations = (weights - mean) ** 2
```

We can compute the mean of the squared deviations like this.

```
var = squared_deviations.sum() / n
var
```

```
np.float64(1.983070989750022)
```

As you might expect, `Series` provides a `var` method that does *almost* the same thing.

```
weights.var()
```

```
np.float64(1.9832904288326545)
```

The result is slightly different because when the `var` method computes the mean of the squared deviations, it divides by `n-1` rather than `n`. That's because there are two ways to compute the variance of a sample, depending on what you are trying to do. I'll explain the difference in Chapter 8 – but in practice it usually doesn't matter. If you prefer the version with `n` in the denominator, you can get it by passing `ddof=0` as a keyword argument to the `var` method.

```
weights.var(ddof=0)
```

```
np.float64(1.983070989750022)
```

In this dataset, the variance of the birth weights is about 1.98, but that value is hard to interpret – for one thing, it is in units of pounds squared. Variance is useful in some computations, but not a good way to describe a dataset. A better option is the **standard deviation**, which is the square root of variance. We can compute it like this.

```
std = np.sqrt(var)
std
```

```
np.float64(1.40821553384062)
```

Or, we can use the `std` method.

```
weights.std(ddof=0)
```

```
np.float64(1.40821553384062)
```

In this dataset, the standard deviation of birth weights is about 1.4 pounds. Informally, values that are one or two standard deviations from the mean are common – values farther from the mean are rare.

1.7. Interpretation

To work with data effectively, you have to think on two levels at the same time: the level of statistics and the level of context. As an example, let's select the rows in the pregnancy file with `caseid` 10229. The `query` method takes a string that can contain column names, comparison operators, and numbers, among other things.

```
subset = preg.query("caseid == 10229")
subset.shape
```

```
(7, 244)
```

The result is a `DataFrame` that contains only the rows where the query is `True`. This respondent reported seven pregnancies – here are their outcomes, which are recorded in chronological order.

```
subset["outcome"].values
```

```
array([4, 4, 4, 4, 4, 4, 1])
```

The outcome code `1` indicates a live birth. Code `4` indicates a miscarriage – that is, a pregnancy loss, usually with no known medical cause.

Statistically this respondent is not unusual. Pregnancy loss is common and there are other respondents who reported as many instances. But remembering the context, this data tells the story of a woman who was pregnant six times, each time ending in miscarriage. Her seventh and most recent pregnancy ended in a live birth. If we consider this data with empathy, it is natural to be moved by the story it tells.

Each row in the NSFG dataset represents a person who provided honest answers to many personal and difficult questions. We can use this data to answer statistical questions about family life, reproduction, and health. At the same time, we have an obligation to consider the people represented by the data, and to afford them respect and gratitude.

1.8. Glossary

The end of each chapter provides a glossary of words that are defined in the chapter.

- **anecdotal evidence:** Data collected informally from a small number of individual cases, often without systematic sampling.
- **cross-sectional study:** A study that collects data from a representative sample of a population at a single point or interval in time.
- **cycle:** One data-collection interval in a study that collects data at multiple intervals in time.
- **population:** The entire group of individuals or items that is the subject of a study.
- **sample:** A subset of a population, often chosen at random.
- **respondents:** People who participate in a survey and respond to questions.
- **representative:** A sample is representative if it is similar to the population in ways that are important for the purposes of the study.
- **stratified:** A sample is stratified if it deliberately oversamples some groups, usually to make sure that enough members are included to support valid conclusions.
- **oversampled:** A group is oversampled if its members have a higher chance of appearing in a sample.
- **variable:** In survey data, a variable is a collection of responses to questions or values computed from responses.
- **codebook:** A document that describes the variables in a dataset, and provides other information about the data.
- **recode:** A variable that is computed based on other variables in a dataset.
- **raw data:** Data that has not been processed after collection.
- **data cleaning:** A process for identifying and correcting errors in a dataset, dealing with missing values, and computing recodes.
- **statistic:** A value that describes or summarizes a property of a sample.
- **standard deviation:** A statistic that quantifies the spread of data around the mean.

1.9. Exercises

The exercises for this chapter are based on the NSFG pregnancy file.

1.9.1. Exercise 1.1

Select the `birthord` column from `preg`, print the value counts, and compare to results published in the codebook at https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NSFG/Cycle6Codebook-Pregnancy.pdf.

1.9.2. Exercise 1.2

Create a new column named `totalwgt_kg` that contains birth weight in kilograms (there are approximately 2.2 pounds per kilogram). Compute the mean and standard deviation of the new column.

1.9.3. Exercise 1.3

What are the pregnancy lengths for the respondent with `caseid` 2298?

What was the birth weight of the first baby born to the respondent with `caseid` 5013? Hint: You can use `and` to check more than one condition in a query.

Think Stats: Exploratory Data Analysis in Python, 3rd Edition