Chapter 2

Type conversions (aka type casting or type coercion): changing an expression from one data type to another

**Methods** or **attributes** are a special type of function that operate only on a specific data structure When using a method in python, you can use a period . to apply the function to an object.

List: one dimensional column of data: things in the list can be stored as different types

List Indexing: []

Vector: one dimensional column of data that can only be the same data type

Boolean masking: (logical vectors)

* Tells us which elements of a corresponding vector we want to keep Logical index must be the same length as the vector we’re indexing.

isin: numpy function for arrays

matrix: numpy function for matrices

.info() from numpy gives metadata

2.5: editing elements in a dataframe and Boolean masking

Chapter 3: Data Viz (with plotnine)

**Aesthetic**: Which *variables* are dictating which *plot elements*.

* X , y, fill, color

**Geometry**: What *shape* of plot you are making.

* Geom\_boxplot, geom\_bar, geom\_point, geom\_histogram()
* Geom\_bar(position = “fill”) for stacked bar plot (percents)
* Geom\_bar(position = “dodge”) for side by side

**Scales**: changes the scale of the x and y axes

* Scale\_y\_continuous(), scale\_y\_log10(), scale\_y\_reverse(), etc

**Themes:** added after geom, changes the general look of the plot

**Facet\_wrap:** includes multiple plots side by side

Chapter 4

Regular Expressions: A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

*# Group by 'City' and calculate total sales for each city*  
city\_sales = df.groupby('City')['Sales'].sum()

Practice Qs

1.1:

What is the length in miles of the 4th longest trail in california?

Start with a Boolean mask of the State column. Filter for only California trails:

Ca = np\_trails[np\_trails[‘state’] == “CA”]

Then convert to miles, possibly through new column

Ca[“length\_in\_miles”] = Ca[“length”] \* (conversion math)

Finally sort by the new column in order from largest to smallest, using the pandas sort\_values method. Then identify the 4th trail.

Ca.sort\_values(“length\_in\_miles”, ascending = False)

Ca.iloc[3]

1.2:

Consider an “easy” trail to be one that has an elevation gain of less than 500 meters, a distance of less than 5 kilometers, and a difficulty rating below 4.

Which California National Park has the highest average rating of its “easy” trails?

Boolean mask again,

Easy\_trails = np\_trails[np\_trails[‘elevation\_gain’] < 500]

Easy\_trails = Easy\_trails[Easy\_trails[‘length’] \* (conversion math to meters) < 5]

Easy\_trails = Easy[Easy\_trails[‘difficulty\_rating’] < 4]

Easy\_trails.sort\_values(‘avg\_rating’, ascending = False)

Easy\_trails.iloc[1]

1.3

How many National Parks contain the word “Land” somewhere in their name?

Np\_trails[‘area\_name’].str.contains(r‘\bland\b’, case = False).sum()

Or

matches = data['column\_name'].apply(lambda x: re.findall(r'\bland\b', str(x), flags=re.IGNORECASE))

flattened\_matches = [match for sublist in matches for match in sublist]

count = len(flattened\_matches)

1.4

Consider the following plot, made from a dataset called np\_trails\_new. Although you can see the code that made the plot, you do not see the code that made the np\_trails\_new object.

Your job in this question is to create the np\_trails\_new object. You’ll know when you have succeeded, because the code provided will produce the plot.

measure: column that includes whether the row is avg\_rating, difficulty\_rating, popularity, and visitor\_usage

score: column that includes the rating based on the measure

1.5:

Recreate the plot below, by filling in the blanks in the code on Canvas.

from plotnine import ggplot, aes, geom\_boxplot, labs, theme, scale\_y\_log10, element\_text

from plotnine.scales import scale\_fill\_manual

(ggplot(ca\_parks, aes(x="**area\_name**", y="**num\_reviews**", **fill** = "**area\_name**"))

+ geom\_boxplot()

+ scale**\_y\_log10**()

+ scale\_**fill\_manual**(guide=False)

+ **labs(**

x = "",

y = "",

**title** = "Number of reviews per trail for CA Parks"

)

+ theme()

+ theme(**axis\_text\_x** =element\_text(rotation=**45**, hjust=1))

)

Part 2:

2.1:

Write a functions to help recommend trails to a user. The function will have as its input the dataset and the name of a National Park, and the following optional arguments:

* A minimum trail elevation gain (in meters)
* A maximum trail elevation gain (in meters)
* A minimum trail rating
* The type of trail

See positron

2.2:

It’s reasonable to think we might also want to search for hikes according to climate. Therefore, we will scrape the average temperatures for each U.S. state from the following website:

* <https://www.extremeweatherwatch.com/us-state-averages>

Write a function that takes one optional argument, month.

The function should scrape the temperature website above for the desired month. If no month is specified then it should scrape the overall average temperatures. The function should return a data frame of the temperatures for the specified month in the names of the temperature columns.