Contents

Introduction

Lab materials adapted from the UC Berkely Data 6

The objectives for this lab are to, learn more about:

- Categorical Distributions
- Bar Charts
- Grouped Bar Charts
- NumPy Array Ranges
- Numerical Distributions, Histograms

- Choosing Bins
- Overlaid and Side-by-Side Histograms

Categorical Distributions

working with lab11.blank.ipynb

Variable Types

Question: How do we know what kind of visualization to create?

Answer: It depends on the type(s) of informaiton we want to visualize.

- Bar charts work for some types of variables, but not all.

Where we are visualizing different variables (features) of a dataset, a lot depends on the type(s) of variables we're working with:

- Variable types are different from data types list `string`, **`int`**, `Table`, etc.

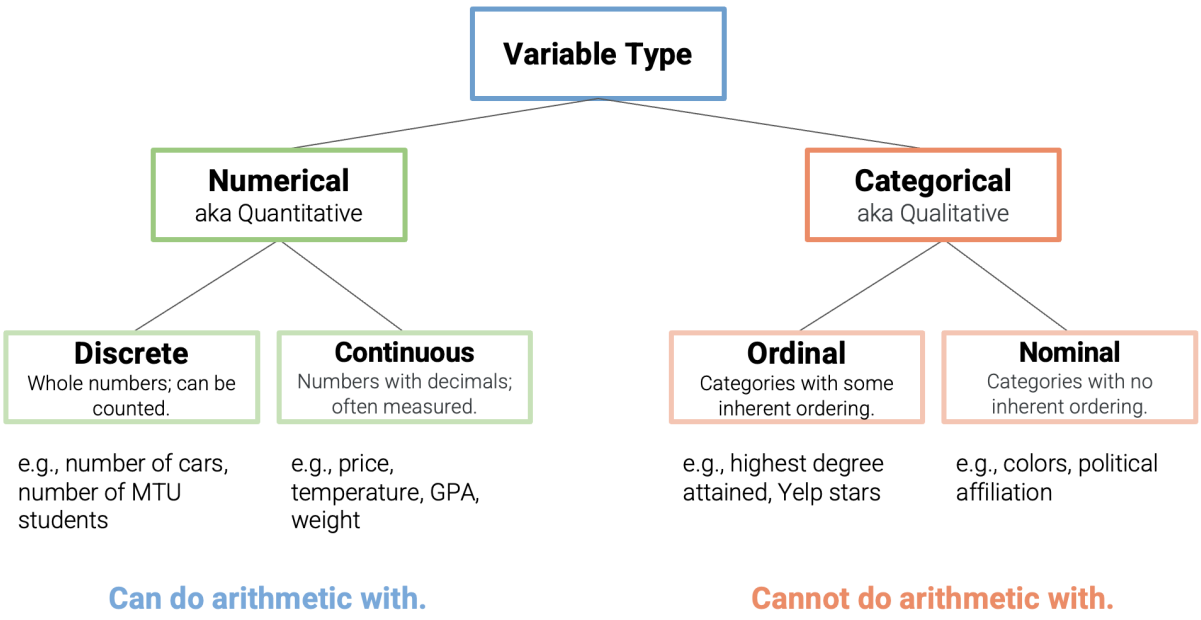


Figure 1: Variable Types

Distribution of Categorical Variables

In a distribution, each individual belongs to exactly one category and thus has exactly one value.

A distribution table (or frequency table) shows all the values of a categorical variable along with the frequency of each one (i.e., number of occurrences).

The table below is a distribution of data on 55 cookies. The values of the categorical variable `Cookie` are: sugar, chocolate chip, red velvet, oatmeal raisin, and peanut butter

`Cookie`	Count
chocolate chip	15
red velvet	15
oatmeal raisin	10
sugar	10
peanut butter	5

Bar Charts

Bar Charts

Bar charts are often used to display the **relationship** between a categorical variable and a numerical variable:

- Average GPAs of Data Science, History, and Biology majors.
- The number of streams by the top 10 songs on Spotify yesterday.

Bar Charts and Categorical Distributions

Bar charts are often used to visualize a categorical distribution:

- The proportion of adults in upper, middle, and lower class.
- The number of public vs private institutions in the top 100 universities
- The numerical variable here is Count.
- Lengths encode values. Widths encode nothing.
- Colors can encode sub-categories! (more soon)

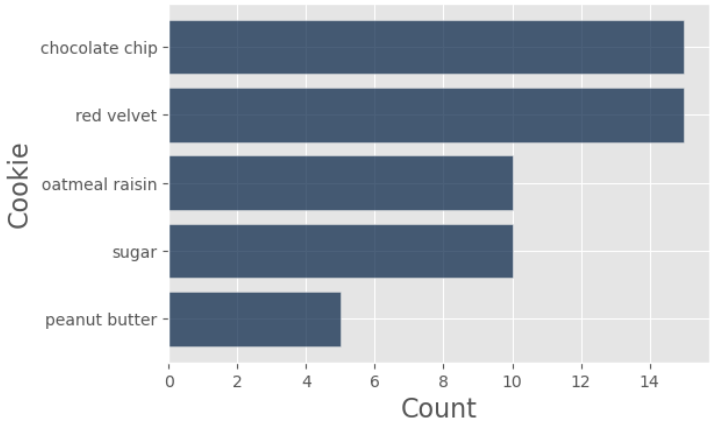


Figure 2: Categorical Distribution of Cookies in a Bar Chart

`.barh()`

```
cookies.barh('Cookie')
```

The method: `t.barh(column_for_categories)`

creates a bar chart with:

- The values of the column **column_for_categories** as the categories on the y-axis. This column should contain a categorical variable, and values should be unique.
- Bars for **every other column** in `t`. These columns should contain numerical variables.

Example: Spotify Daily Top 10 Songs

Let's look at the Global Daily Top 10 songs on Spotify.

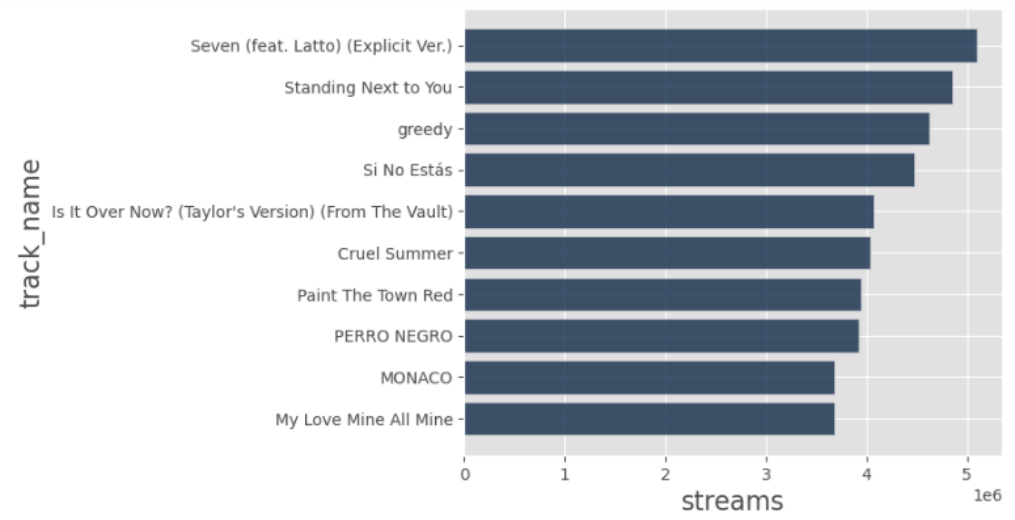
You can download an up-to-date copy of this data here using a Spotify account.

```
top_10_simple
```

track_name	streams
Seven (feat. Latto) (Explicit Ver.)	5098014
Standing Next to You	4851071
greedy	4627366
Si No Estás	4476616
Is It Over Now? (Taylor's Version) (From The Vault)	4072746
Cruel Summer	4037989
Paint The Town Red	3949768
PERRO NEGRO	3918959
MONACO	3682558
My Love Mine All Mine	3675868

We can create a bar chart from this data showing for each track, the number of streams.

```
top_10_simple.barh('track_name')
```



Example 2: Spotify Daily Top 10 Songs

In the prior example, the Table had only the single categorical and numeric variables (columns) to use in the bar chart.

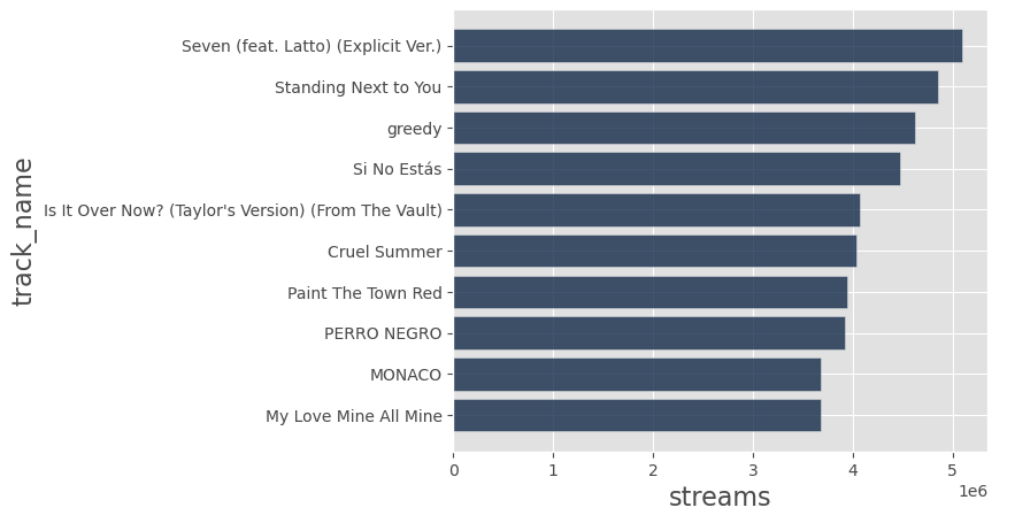
Now, we have many more variables (columns) in the Table.

```
top_10
```

rank	uri	artist_names	track_name	source	peak_rank	previous_rank	days_on_chart	streams
1	spotify:track:2HRgqmZQC0MC7GeNuDIXHN	Jung Kook, Latto	Seven (feat. Latto) (Explicit Ver.)	BIGHIT MUSIC	1	1	115	5098014
2	spotify:track:2KslE17cAJNHTsl2MI0jb2	Jung Kook	Standing Next to You	BIGHIT MUSIC	2	2	3	4851071
3	spotify:track:3rUGC1vUpkDG9CZFHMur1t	Tate McRae	greedy	RCA Records Label	1	3	52	4627366
4	spotify:track:2HafqoJbgXdtjwCOvNEF14	iñigo quintero	Si No Estás	Acqustic	1	4	42	4476616
5	spotify:track:1lq8oo9XkmmvCQiGOfORiz	Taylor Swift	Is It Over Now? (Taylor's Version) (From The Vault)	Taylor Swift	1	5	10	4072746
6	spotify:track:1BxfuPKGuaTgP7aM0Bbdwr	Taylor Swift	Cruel Summer	Taylor Swift	2	8	273	4037989
7	spotify:track:56y1jOTK0XSvJzVv9vHQBK	Doja Cat	Paint The Town Red	Kemosabe Records/RCA Records	1	7	94	3949768
8	spotify:track:7iQXYTyuG13aoeHxGG28Nh	Bad Bunny, Feid	PERRO NEGRO	Rimas Entertainment LLC	3	6	24	3918959
9	spotify:track:4MjDJD8cW7iVeWInc2Bdyj	Bad Bunny	MONACO	Rimas Entertainment LLC	1	9	24	3682558
10	spotify:track:3vkCueOmm7xQDoJ17W1Pm3	Mitski	My Love Mine All Mine	Dead Oceans	9	12	48	3675868

Therefore, you must specify in `barh(categorical_column, numeric_column)`

```
top_10.barh('track_name', 'streams')
```



Pre-Process the Table Before Plotting

It's important to first create a table with only the information you want, sorted in the order you want, before trying to visualize.

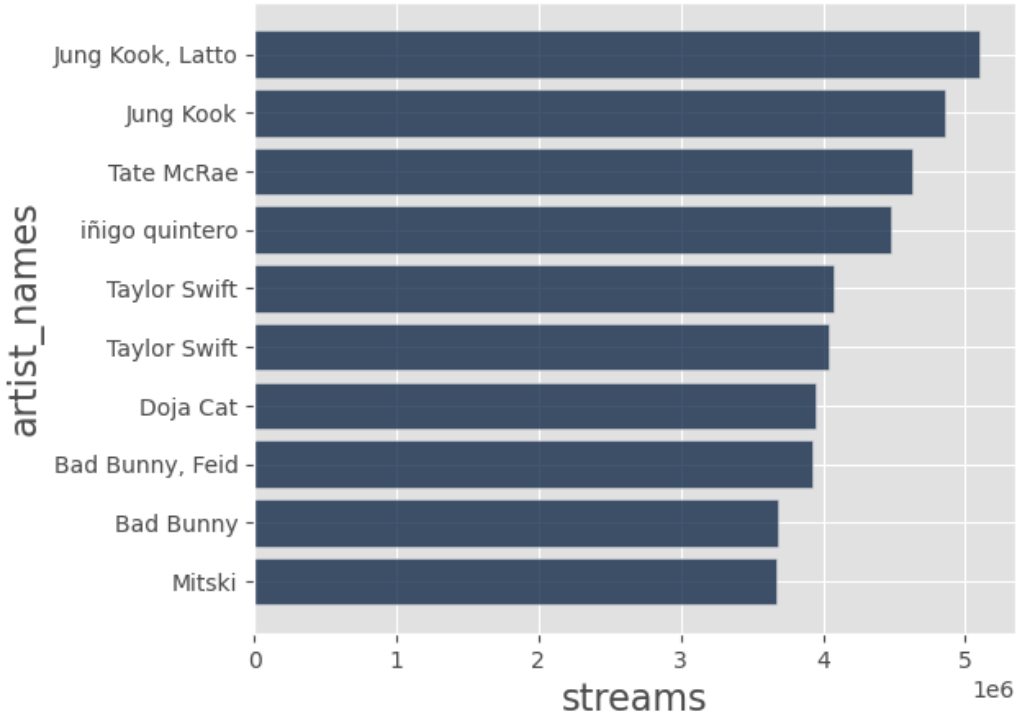
If you call `barh` on your original table without any additional selection, the results you get back might not make any sense.

> In a categorical distribution, each category should only appear once.

Here is an example, where we see categorical values appearing more than once.

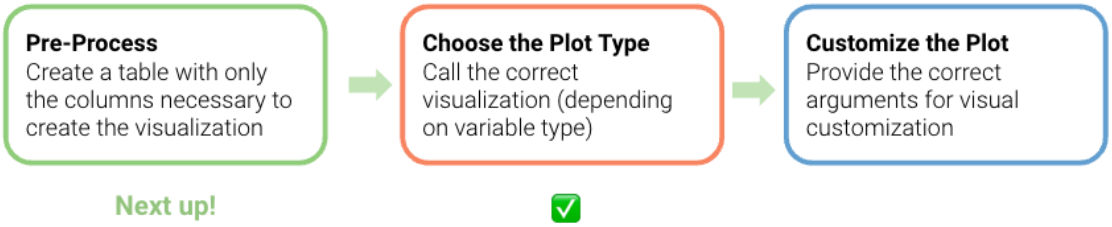
We will learn how to adjust and change our code for situations like this.

```
top_10.barh('artist_names', 'streams')
```



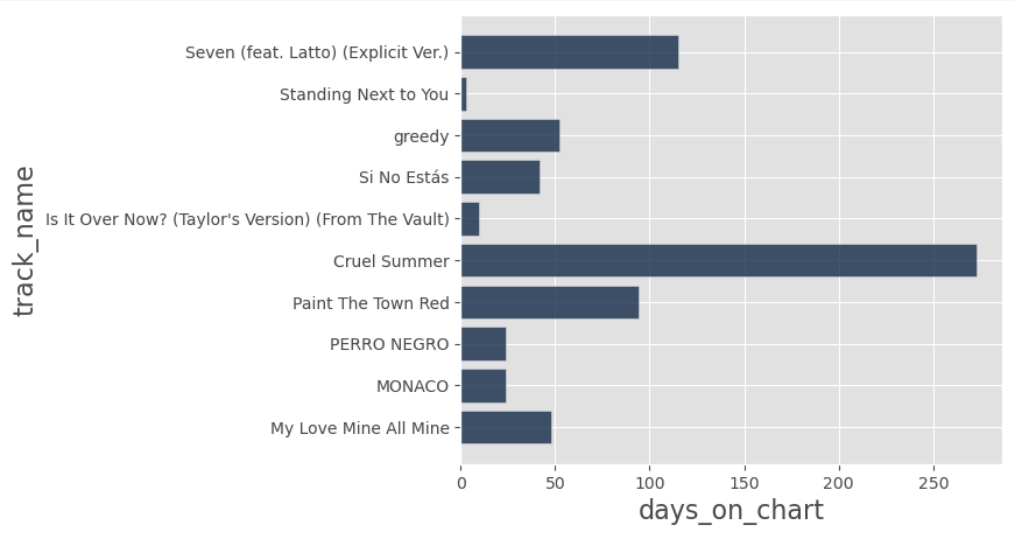
Three-Step Process for Visualization

This process will apply to **all visualizations** for this class, not just today's bar charts.



`.sort`

```
top_10.barh('track_name', 'days_on_chart')
```



`sort()`

The method **`t.sort(...)`** returns a new table with the rows sorted according to the values in some column. There are two ways we can call it:

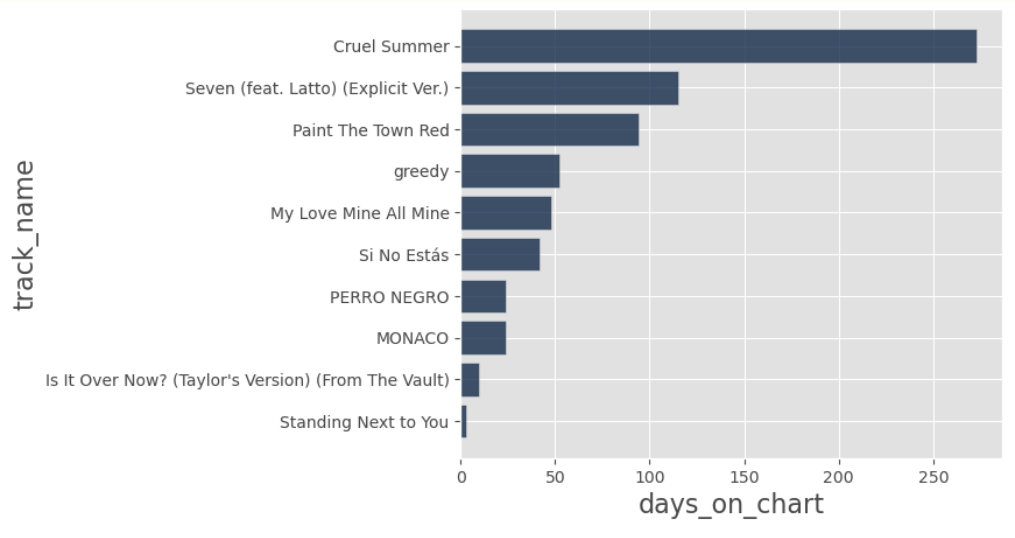
1. `t.sort(column_or_label)`
   - Sorts rows according to the specified column, in ascending order.
2. `t.sort(column_or_label, descending = True)`

- Sorts rows according to the specified column, in descending order.
- descending is an optional parameter. If it is not provided, then Python supplies a default. In this case, the default is ascending, i.e., Approach 1.
- 'True' is a boolean value. We will learn more about booleans next week; just remember the sort() syntax for now.

We'll most often specify columns by their label (i.e., variable name), though we could also specify by index (0, 1, 2, etc.)

Sorting by Number of Days on Chart

```
top_10.sort('days_on_chart', descending=True).barh('track_name', '
    days_on_chart')
```



With **method chaining**, Python evaluates method calls **left-to-right**.

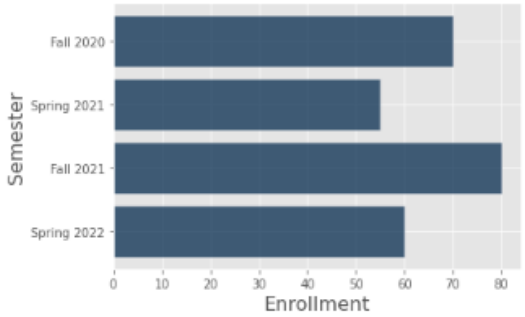
The new table returned by **sort()** is the one that is then plotted by `barh()`.

Visualization Note: Bar Order

Depending on the type of categorical variable we're displaying, we may want to sort the bars of our bar charts differently.

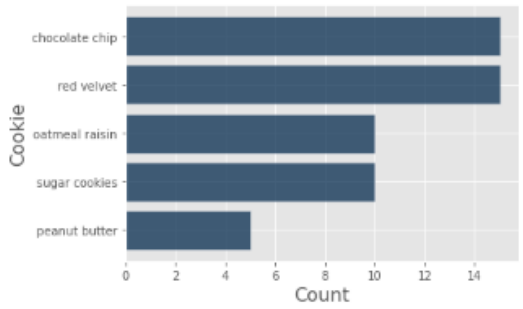
Sort by category: e.g., if categorical variable has an inherent ordering like alphabetical, numerical, etc.

Semester	Enrollment
Fall 2020	70
Spring 2021	55
Fall 2021	80
Spring 2022	60



Sort by bar length: e.g., if categorical variable has no natural order to the categories.

Cookie	Count
chocolate chip	15
red velvet	15
oatmeal raisin	10
sugar cookies	10
peanut butter	5

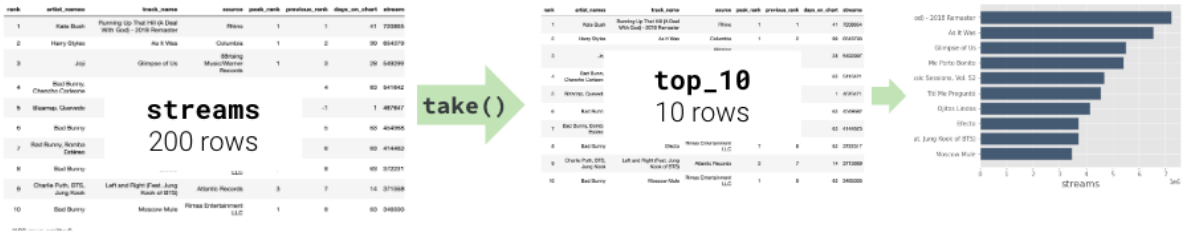


The bar order depends on what you want to express through your visualization.

`.take`

The Spotify Daily Chart dataset actually contains the Top 200 songs from each day. This would be pretty tedious to plot.

Instead, we make a new table that *limits the number of rows to 10*, then we plot that table:



`take(np.aranage(n))`

The method call `t.take(np.arange(n))` returns a new table with only the first `n` rows. We will cover all components of this method call in detail over the next few days.

Quick Check 1

Given `streams_top_15`, a table of the top 15 most streamed songs, generate a bar chart showing how many streams each artist has.

```
streams_top_15 = streams.sort('streams', descending=True).take(np.
    arange(15)) \
                        .select('artist_names', 'track_name', '
                            streams')
streams_top_15
```

artist_names	track_name	streams
Tate McRae	greedy	5483445
iñigo quintero	Si No Estás	5296773
Taylor Swift	Is It Over Now? (Taylor's Version) (From The Vault)	5175170
Jung Kook, Latto	Seven (feat. Latto) (Explicit Ver.)	4833604
Doja Cat	Paint The Town Red	4562022
Taylor Swift	Cruel Summer	4472245
Bad Bunny, Feid	PERRO NEGRO	4403434
Bad Bunny	MONACO	4342383
Taylor Swift	Now That We Don't Talk (Taylor's Version) (From The Vault)	4209179
Kenya Grace	Strangers	3799489

... (5 rows omitted)

Confirm with your neighbors

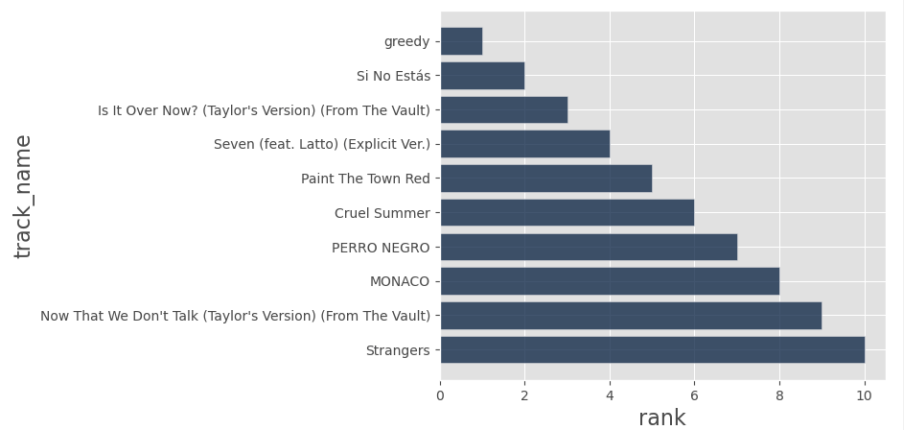
Then, check in the notebook

Grouped Bar Charts

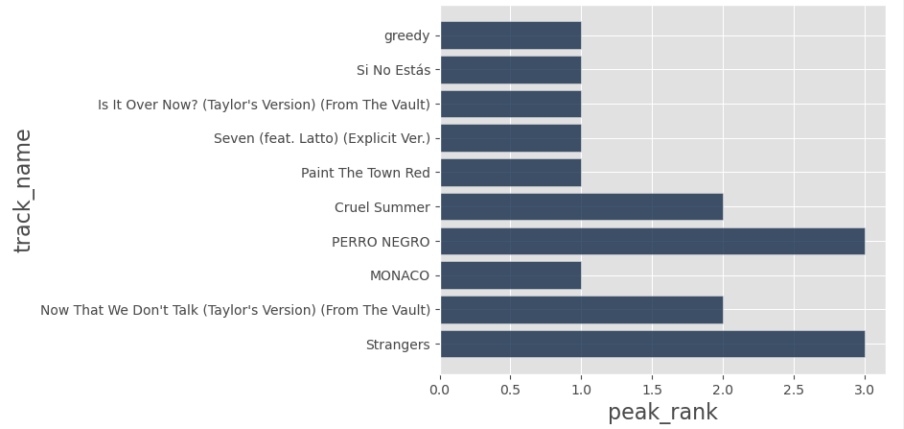
Spotify keeps track of each song's "peak rank," which is a song's lowest rank since its release on the platform.

With our current tools, we'd likely make two different plots, which are difficult to compare:

```
top_10.barh('track_name', 'rank')
```




```
top_10.barh('track_name', 'peak_rank')
```



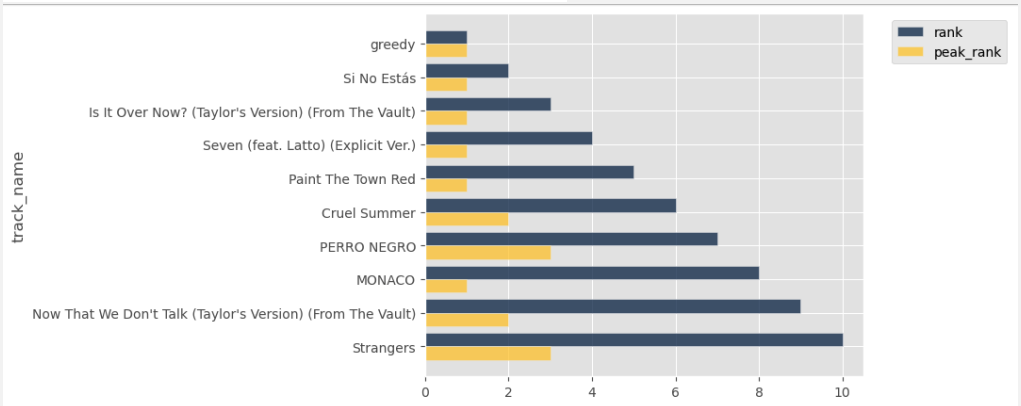
Grouped Bar Chart

We can use `t.select()` to build a “narrower” table with one categorical variable + several variables.

If the table we call `barh` on has multiple numerical columns, it will draw bars for each of them, and each column will get its own color!

```
top_10.select('rank', 'track_name', 'peak_rank').barh('track_name')
```

rank	track_name	peak_rank
1	greedy	1
2	Si No Estás	1
3	Is It Over Now? (Taylor's Version) (From The Vault)	1
4	Seven (feat. Latto) (Explicit Ver.)	1
5	Paint The Town Red	1
6	Cruel Summer	2
7	PERRO NEGRO	3
8	MONACO	1
9	Now That We Don't Talk (Taylor's Version) (From The Vault)	2
10	Strangers	3



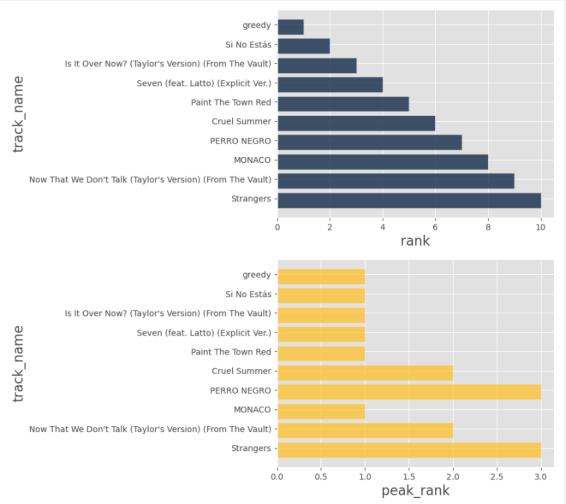
`barh` Optional Parameter: Overlay

`barh()` also has an optional parameter: **overlay**.

If not supplied, then by default `barh` will plot all numerical variables as bars on the same plot.

If supplied (`overlay=False`), then `barh` will plot each numerical variable on separate plots.

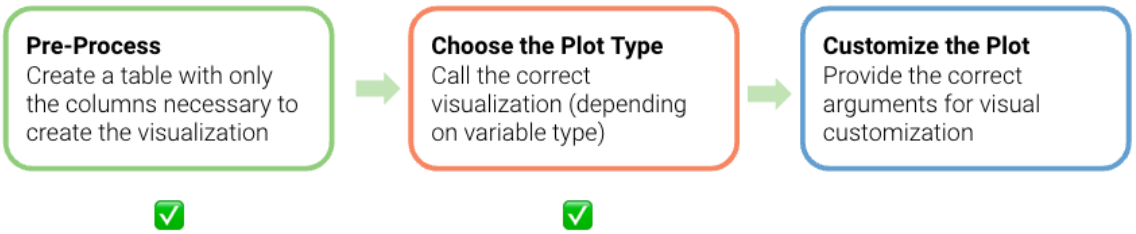
```
top_10.select('rank', 'track_name', 'peak_rank').barh('track_name',
    overlay=False)
```



Summary

Three-Step Process for Visualization

This process will apply to **all visualizations** we create in this class, not just today's bar charts:



Bar charts are used to display the distribution of a categorical variable, or the relationship between a categorical variable and one or more numerical variables.

The method `t.barh(column_for_categories)` creates a bar chart with:

- The values of the column `column_for_categories` as the categories on the y-axis. This column should contain a categorical variable, and values should be unique.
- Bars for every other column in `t`. These columns should contain numerical variables.
  - Multiple other columns -> grouped bar chart.
- Bars should be sorted depending on the type of the categorical variable.

NumPy Array Ranges

working with lab11.part2.blank.ipynb

`np.arange`

The NumPy function `np.arange()` creates a sequence of numbers.

Format	Returns
`np.arange(n)`	An array of all integers from **0** to **n-1**
`np.arange(start, stop)`	An array of all integers from **start** to **stop-1**
`np.arange(start, stop, step)`	An array of all integers from **start** to **stop-1**, counting by **step**

Array ranges work like indexing: **inclusive** of the starting position, and **exclusive** of the ending position.

Array Range Examples

```
arr = np.arange(1, 11)
arr
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
np.sum(arr)
```

```
55
```

```
np.arange(3, 11, 2)
```

```
array([3, 5, 7, 9])
```

```
np.arange(10, 1, -3)
```

```
array([10, 7, 4])
```

Quick Check 1

What do each of these cells output when run? Try to figure out the answers before running the code.

```
np.arange(5)
```

```
np.arange(3, 13, 3)
```

```
2 ** np.arange(8)
```

```
np.sum(np.arange(4) ** 2)
```

Confirm with your neighbors

Then, check in the notebook

Histograms

Bar charts visualize the distribution of a categorical variable, or the relationship between a categorical variable and one or more numerical variables.

Bar charts cannot display the distribution of a numerical variable.

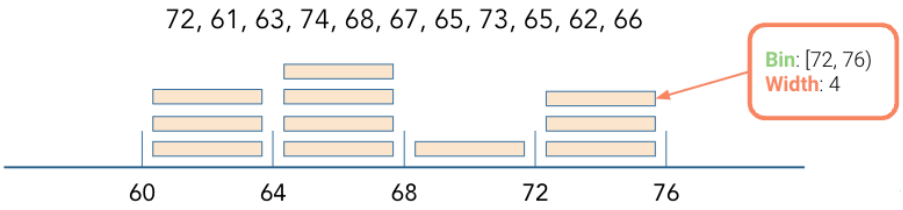
Question: How might we display the distribution of a numerical variable?

Answer: One solution is to combine ranges of values into bins.

Binning

Binning is counting the number of numerical values that fall within ranges, called "bins".

- A bin is defined by a left endpoint (lower bound) and right endpoint (upper bound).
- A value falls in a bin if it is greater than or equal to the left endpoint and less than the right endpoint.
  - $[a, b)$: a is included, b is not.
- The width of a bin is its right endpoint minus its left endpoint.



Histograms

A histogram visualizes the distribution of a numerical variable by binning. The method

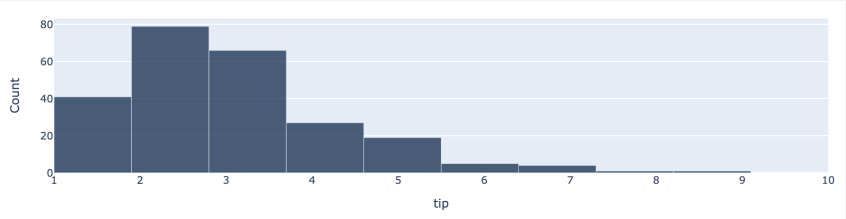
```
t.hist(column, density = False)
```

creates a histogram of the column column of `t`. This column must contain numerical values.

- It automatically chooses bins for us. We can change them.
- We will always set `density = False`.

Example

```
tips.hist('tip', density = False)
```

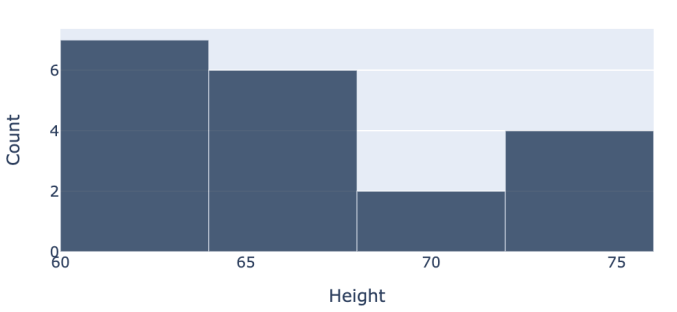


Why `density = False`?

- By setting `density = False`, the resulting histogram shows counts on the y-axis (i.e. number of observations per bin).
- The default, `density = True`, creates a histogram with "percent per unit".
  - The details of this are out of scope. DATA 1202 will teach you about these types of histograms.

Quick Check 2

Answer the following questions about this histogram of (fake) heights. If you don't believe it's possible to answer the question, write "can't tell".



1. How many heights are between 60 inches (inclusive) and 64 inches (exclusive)?
2. How many heights are between 62 inches (inclusive) and 68 inches (exclusive)?

Confirm with your neighbors

Then, check in the notebook

Choosing Bins

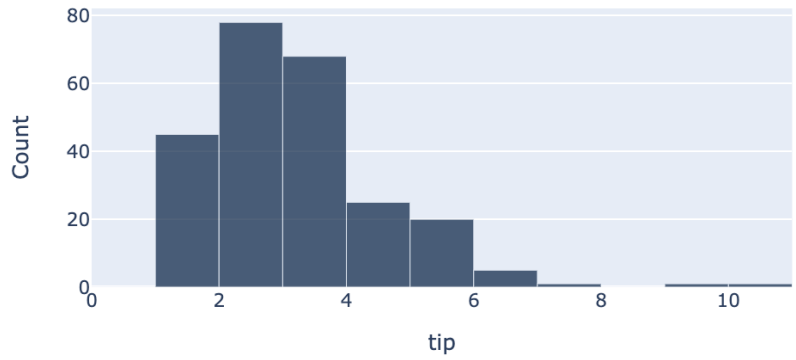
`.hist` chooses bins by default for us.

- The resulting histogram often looks nice, but has non-integer bins which are harder for us to interpret.
- We can choose our own bins.
- We will only consider histograms where all bins have equal width, though it is possible to draw histograms where bins have unequal width.

The optional `bins` argument in `hist` requires an array, and uses the values in the array as bins instead.

::: box

```
tips.hist('tip',
          density = False,
          bins = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]))
```

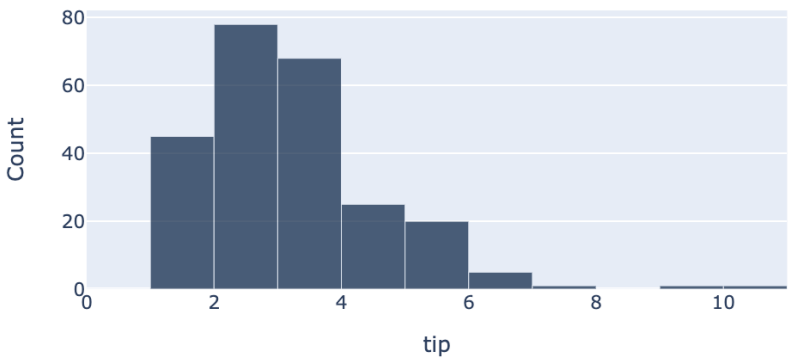


`np.arange()`

If we only need a few bins, we can make an array of bins by hand, like in the previous example.

But if we want more than a handful of bins, it will be tiring to create them by hand. Instead, we can use `np.arange` to create an array of equally spaced values.

```
tips.hist('tip',
          density = False,
          bins = np.arange(12))
```



Example

It's a good idea to determine the min and max values in a column before choosing bins, to make sure your bins encompass the entire range.

```
tips.column('total_bill').min()
```

```
3.06999999999
```

```
tips.column('total_bill').max()
```

```
50.810000000000002
```

```
bins_3 = np.arange(3, 54, 3)
bins_3
```

```
array([ 3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45,
    48, 51])
```

```
bins_7 = np.arange(3, 53, 7)
bins_7
```

```
array([ 3, 10, 17, 24, 31, 38, 45, 52])
```

Tradeoff

The width of each bin used dramatically changes the resulting histogram.

- Narrow Bins -> many bins, each with few values. Resulting histogram is choppy.
- Wide Bins -> few bins, each with many values. Resulting histogram is smooth.

In practice, it's up to you to choose the bin size of your histogram.

Examine the histograms in the notebook with bin widths of 3, 7, and 10.

Overlaid and Side-by-Side Histograms

group and overlay

What if we want to show the distribution of a numerical variable, **grouped** by some categorical variable? For example, tips on weekends vs. tips on weekdays?

We can create grouped histograms by using the **group** and **overlay** arguments in `hist`.

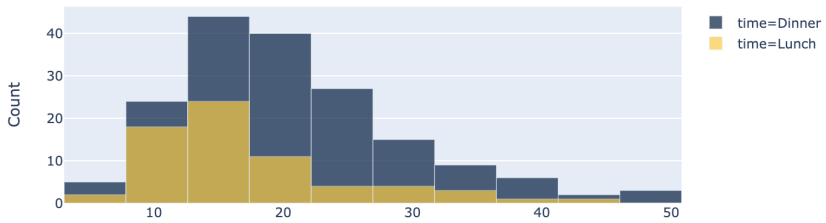
- To create one histogram for every unique category in some categorical column, assign the group argument to the name of the categorical column.

`t.hist(column, density = False, group = categorical_column)` * By default, the above creates multiple histograms on the same set of axes. If you would rather have multiple individual histograms, set the `overlay` argument to `False`.

```
t.hist(column, density = False, group = categorical_column, overlay =
 False)
```

Example

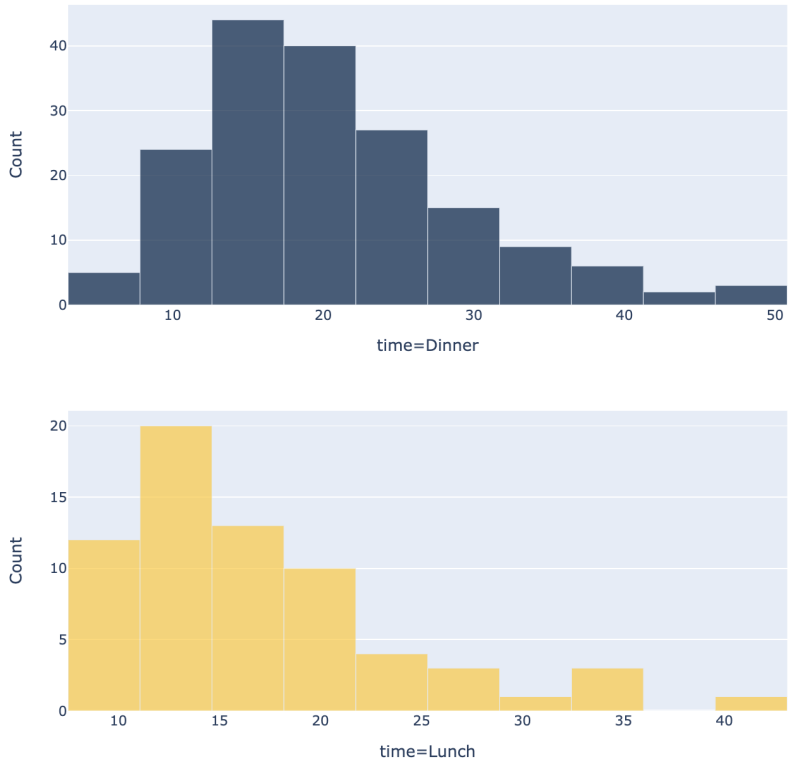
```
tips.hist('total_bill', density = False, group = 'time')
```



It can be hard to compare two distributions if one has significantly more observations than the other.

When comparing distributions, compare the **shape**, not the absolute heights.

```
tips.hist('total_bill', density = False, group = 'time', overlay = False)
```



Bar Charts vs. Histograms

Bar charts visualize the distribution of a categorical variable, or the relationship between a categorical variable and a numerical variable.

- Length of bar corresponds to value.
- Width of bar means nothing.

Histograms visualize the distribution of a numerical variable.

- Length of bar corresponds to number of values within bin.
- Width of bar corresponds to the width of the bin.
  - Wider bin -> more values within bin -> smoother histogram.

Summary

A histogram visualizes the distribution of a numerical variable by binning. The method

```
t.hist(column, density = False)
```

creates a histogram of the column column of `t`. This column must contain numerical values.

Optional arguments: * `bins` (array): allows us to manually select bins. Frequently used with `np.arange`. * `group` (string): allows us to draw separate histograms, one for each unique value in the specified categorical column. * `overlay` (boolean): only used in conjunction with group. If `False`, draws histograms on separate axes rather than on top of one another.