

*7 TIPS FOR
SUCCESS:*

*INTRODUCTION TO
DATA ANALYSIS W/
PYTHON*

James Wilson

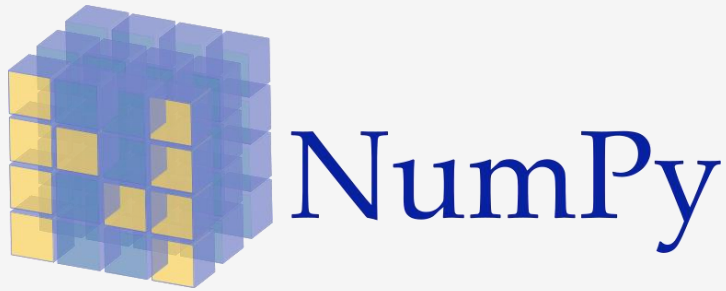


Download Anaconda

- Download version 3.7 from <https://www.anaconda.com/distribution/>
- Once downloaded, please open Jupyter Notebooks



Libraries



- **Pandas**
 - A fast, powerful, flexible and easy to use open source data analysis and manipulation tool.
- **Numpy**
 - Tool to support working with large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
- **Plotly**
 - Provides online graphing, analytics, and statistics tools for individuals and collaboration

The Data: Jeopardy Player Details



Jeopardy Archive

The fan-created archive of Jeopardy! games and players—386,007 clues and counting!

J! ARCHIVE

[All] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36]

RECENT FINAL JEOPARDY!

SPORTS TERMS

A member of the British Amateur Athletic Club wrote this 1867 set of 12 regulations whose name honors nobleman John Sholto Douglas

from show #6156, aired 2020-02-10

RANDOM FINAL JEOPARDY!

BUSINESS & INDUSTRY

Headquartered near St. Paul, this company churns out the top-selling brand of butter in the U.S.

from show #1064, aired 1989-03-30

@CoolJeopardyStories Twitter Account

A screenshot of the Twitter profile for @CoolJepStories. The profile picture is a circular logo with the text 'COOL JEOPARDY! STORIES' and a background of the Jeopardy! logo. The header shows the account name 'Cool Jeopardy! Stories' and the handle '@CoolJepStories'. The bio reads: 'Simply yet accurately summarizing the contestant interviews part on Jeopardy! Read right to left. Not affiliated with Jeopardy Productions, Inc. By @ChadMosher.' It also shows 'Joined February 2014', '3 Following', and '10.2K Followers'. At the bottom, there are tabs for 'Tweets', 'Tweets & replies', 'Media', and 'Likes'.

Creating the Data: Archive

Show #8032 - Tuesday, July 9, 2019

Contestants

Kevin Paquette, a math teacher from Charlottesville, Virginia
Hannah Safford, an engineering Ph.D. student from Davis, California
Ryan Bilger, a student from Macungie, Pennsylvania (whose 3-day cash winnings total \$96,650)

- Can extract the show date, player names, occupations, hometowns, final scores, and other details.

Final scores:

Ryan	Hannah	Kevin
\$10,399	\$8,800	\$7,199
4-day champion: \$107,049	2nd place: \$2,000	3rd place: \$1,000

Creating the Data: Twitter



- Can extract fun facts and the show date the players shared them.
- We can also deduce who said which fact!

Python Tips!

- Tip #1 – Evaluate column names and types
- Tip #2 – Detect NA's & duplicates; why are they there?
- Tip #3 - String cleaning & variable creation
- Tip #4 - Join and reshape data
- Tip #5 - Numeric summaries and outlier detection
- Tip #6 - Evaluating categorical variables with frequency tables
- Tip #7 - Build graphics to explore ideas

Tip #1:

Evaluate column names and types

- Consistent names are easier to read
- Lower case is the best case
- Remove special characters and spaces
- Make your life simpler!

Code:

- `df.shape`
- `df.info()`
- `df.dtypes()`
- `df.columns`

Tip #1:

Evaluate column names and types

- Consistent names are easier to read
- Lower case is the best case
- Remove special characters and spaces
- Make your life simpler!

Code:

- `df.shape`
- `df.info()`
- `df.dtypes()`
- `df.columns`

Returns dimensions of
the DataFrame (df)

Tip #1:

Evaluate column names and types

- Consistent names are easier to read
- Lower case is the best case
- Remove special characters and spaces
- Make your life simpler!

Code:

- `df.shape`
- `df.info()`
- `df.dtypes()`
- `df.columns`



Returns summary of the
DataFrame (df)


Tip #1:

Evaluate column names and types

- Consistent names are easier to read
- Lower case is the best case
- Remove special characters and spaces
- Make your life simpler!

Code:

- `df.shape`
- `df.info()`
- `df.dtypes()`
- `df.columns`



Returns the data type (dtype) of each column in the given dataframe

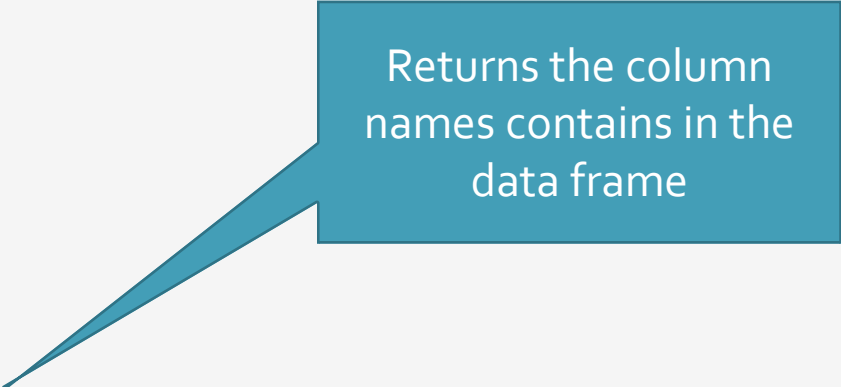
Tip #1:

Evaluate column names and types

- Consistent names are easier to read
- Lower case is the best case
- Remove special characters and spaces
- Make your life simpler!

Code:

- `df.shape`
- `df.info()`
- `df.dtypes()`
- `df.columns`



Returns the column
names contains in the
data frame

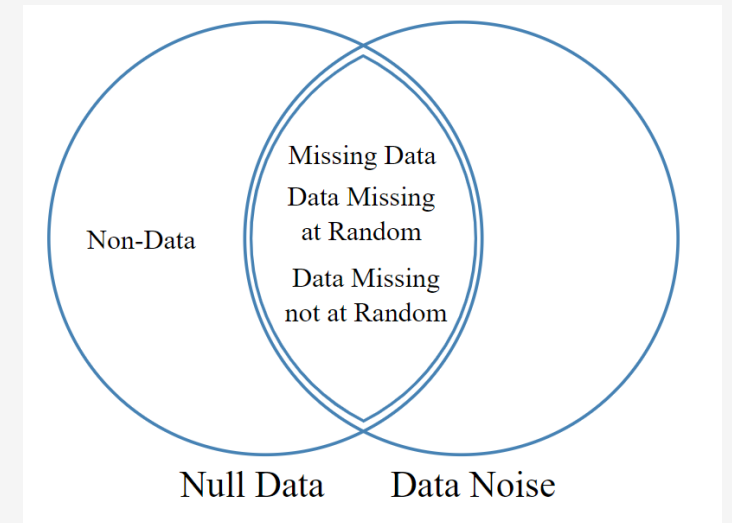
Tip #2:

Detect NA's & Duplicates; why are they there?

- Need to understand what data is present
- Are you missing important information?
- What will you be able to work with?

Code:

- `df.isnull() / df.isna()`
- `df.fillna()`
- `df.duplicated()`
- `df.drop_duplicates()`



Tip #2:

*Detect NA's
&
Duplicates;
why are
they there?*

- Need to understand what data is present
- Are you missing important information ?
- What will you be able to work with?

Code:

- `df.isnull() / df.isna()`
- `df.fillna()`
- `df.duplicated()`
- `df.drop_duplicates()`



Detect Null values
in the DF

Tip #2:

*Detect NA's
&
Duplicates;
why are
they there?*

- Need to understand what data is present
- Are you missing important information ?
- What will you be able to work with?

Code:

- `df.isnull() / df.isna()`
- `df.fillna()`
- `df.duplicated()`
- `df.drop_duplicates()`



Fill Missing Values in
Date Frame

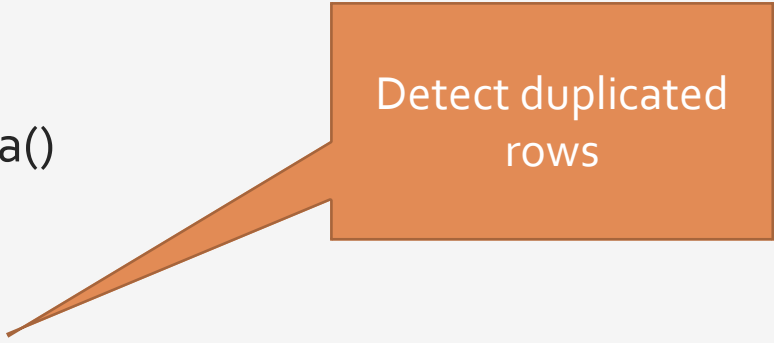
Tip #2:

*Detect NA's
&
Duplicates;
why are
they there?*

- Need to understand what data is present
- Are you missing important information ?
- What will you be able to work with?

Code:

- `df.isnull() / df.isna()`
- `df.fillna()`
- `df.duplicated()`
- `df.drop_duplicates()`



Detect duplicated
rows

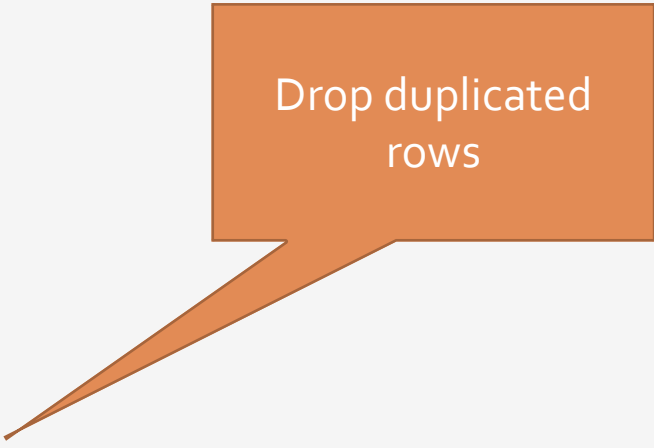
Tip #2:

*Detect NA's
&
Duplicates;
why are
they there?*

- Need to understand what data is present
- Are you missing important information ?
- What will you be able to work with?

Code:

- `df.isnull() / df.isna()`
- `df.fillna()`
- `df.duplicated()`
- `df.drop_duplicates()`



Drop duplicated
rows

Tip #3:

String cleaning & variable creation

- Cleaning variables early is essential for later success
- Allows you to quality control the data as you go
- Creating features and more granular variables will make for more detailed analytics

Code:

- `df['x'].str.replace()`
- `df['x'].astype()`
- `df['x'].split()`
- `df['x'].strip()`
- And more!


Tip #3:

String cleaning & variable creation

- Cleaning variables early is essential for later success
- Allows you to quality control the data as you go
- Creating features and more granular variables will make for more detailed analytics

Code:

- `df['x'].str.replace()`
- `df['x'].astype()`
- `df['x'].split()`
- `df['x'].strip()`
- And more!



Replace specified
characters in string


Tip #3:

String cleaning & variable creation

- Cleaning variables early is essential for later success
- Allows you to quality control the data as you go
- Creating features and more granular variables will make for more detailed analytics

Code:

- `df['x'].str.replace()`
- `df['x'].astype()`
- `df['x'].split()`
- `df['x'].strip()`
- And more!



Change the data type
of a variable

Tip #3:

String cleaning & variable creation

- Cleaning variables early is essential for later success
- Allows you to quality control the data as you go
- Creating features and more granular variables will make for more detailed analytics

Code:

- `df['x'].str.replace()`
- `df['x'].astype()`
- `df['x'].split()`
- `df['x'].strip()`
- And more!



Split a variable at a specified delimiter


Tip #3:

String cleaning & variable creation

- Cleaning variables early is essential for later success
- Allows you to quality control the data as you go
- Creating features and more granular variables will make for more detailed analytics

Code:

- `df['x'].str.replace()`
- `df['x'].astype()`
- `df['x'].split()`
- `df['x'].strip()`
- And more!



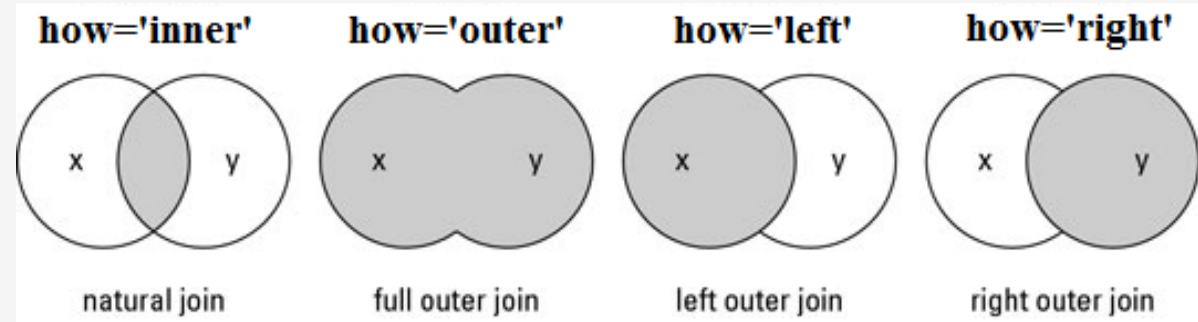
String a variable of
white space

Tip #4:

Join and reshape data

Merge

- Combine datasets to have a full picture of the data
- Make sure not to lose any data in the process



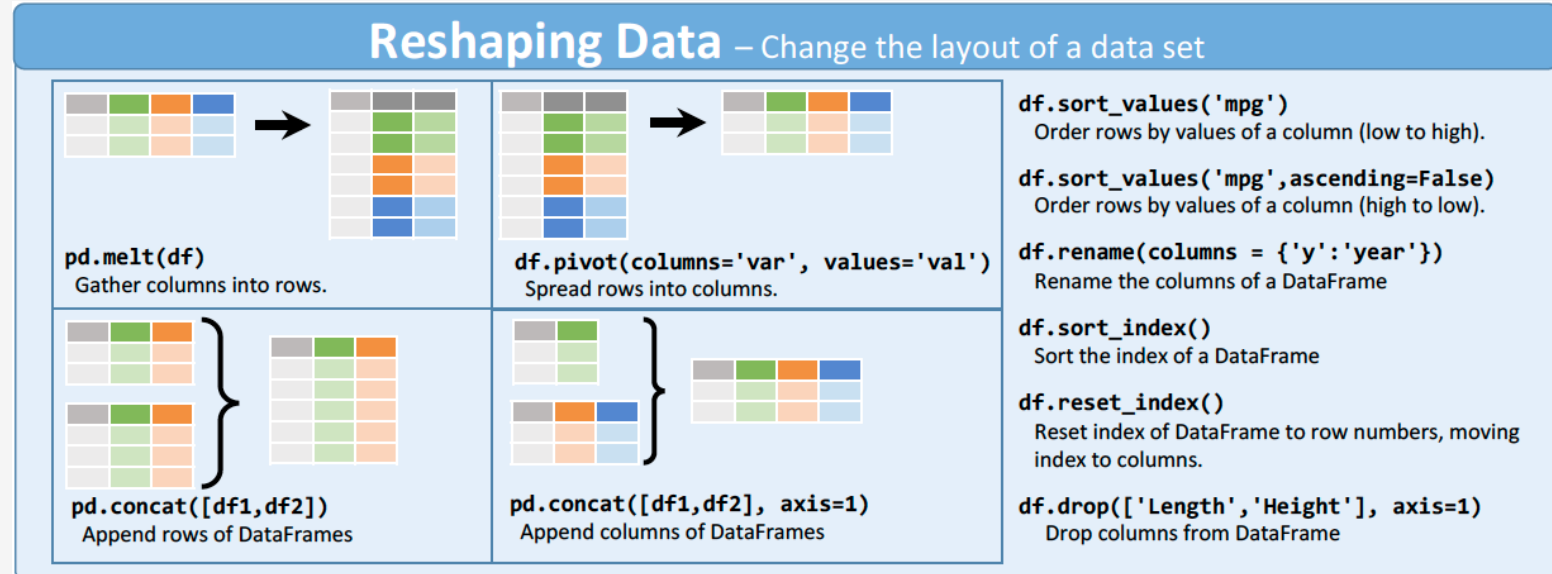
Code:

- `pd.merge()`

Combine data sets
on specified keys

Tip #4: Join and reshape data

Melt



Code:

- `pd.melt()`

Massage dataframe into a format where one or more columns are identifiers, where all other columns are considered unpivoted, measured variables

Tip #5:

Numeric summaries and outlier detection

- Evaluate the spread and shape of the numerical fields
- Search for outliers that may indicate problems with data collection – will be an issue for model building

Code:

- `df['x'] > y`
- `df['x'].sum()`
- `df['x'].describe()`
- `df['x'].transform('function')`
- `np.where(x, a, b)`

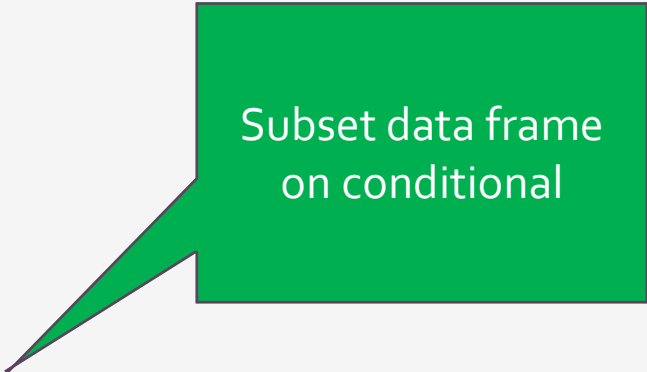
Tip #5:

Numeric summaries and outlier detection

- Evaluate the spread and shape of the numerical fields
- Search for outliers that may indicate problems with data collection – will be an issue for model building

Code:

- `df['x'] > y`
- `df['x'].sum()`
- `df['x'].describe()`
- `df['x'].transform('function')`
- `np.where(x, a, b)`



Subset data frame
on conditional


Tip #5:

Numeric summaries and outlier detection

- Evaluate the spread and shape of the numerical fields
- Search for outliers that may indicate problems with data collection – will be an issue for model building

Code:

- `df['x'] > y`
- `df['x'].sum()`
- `df['x'].describe()`
- `df['x'].transform('function')`
- `np.where(x, a, b)`



Create summary stats on specified variable (e.g. sum)


Tip #5:

Numeric summaries and outlier detection

- Evaluate the spread and shape of the numerical fields
- Search for outliers that may indicate problems with data collection – will be an issue for model building

Code:

- `df['x'] > y`
- `df['x'].sum()`
- `df['x'].describe()`
- `df['x'].transform('function')`
- `np.where(x, a, b)`



Generate ALL
summary statistics
for specified variable


Tip #5:

Numeric summaries and outlier detection

- Evaluate the spread and shape of the numerical fields
- Search for outliers that may indicate problems with data collection – will be an issue for model building

Code:

- `df['x'] > y`
- `df['x'].sum()`
- `df['x'].describe()`
- `df['x'].transform('function')`
- `np.where(x, a, b)`



Create a new variable based on transformation of a variable

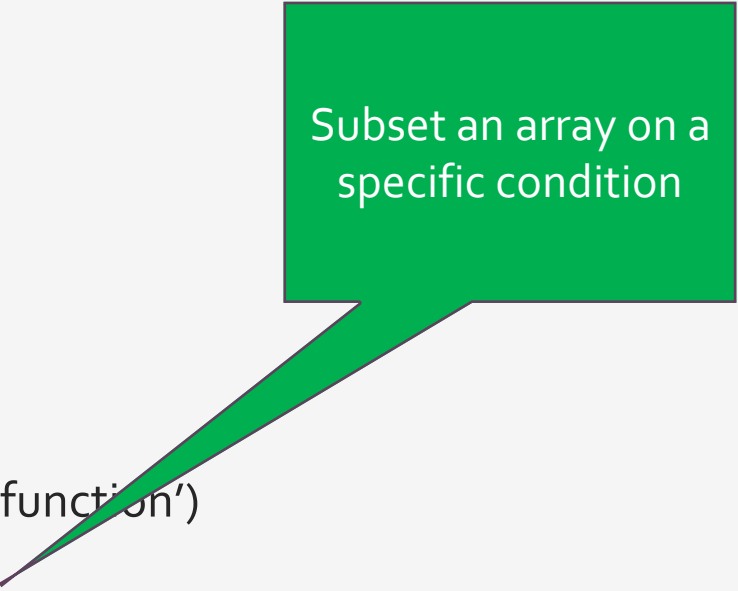
Tip #5:

Numeric summaries and outlier detection

- Evaluate the spread and shape of the numerical fields
- Search for outliers that may indicate problems with data collection – will be an issue for model building

Code:

- `df['x'] > y`
- `df['x'].sum()`
- `df['x'].describe()`
- `df['x'].transform('function')`
- `np.where(x, a, b)`



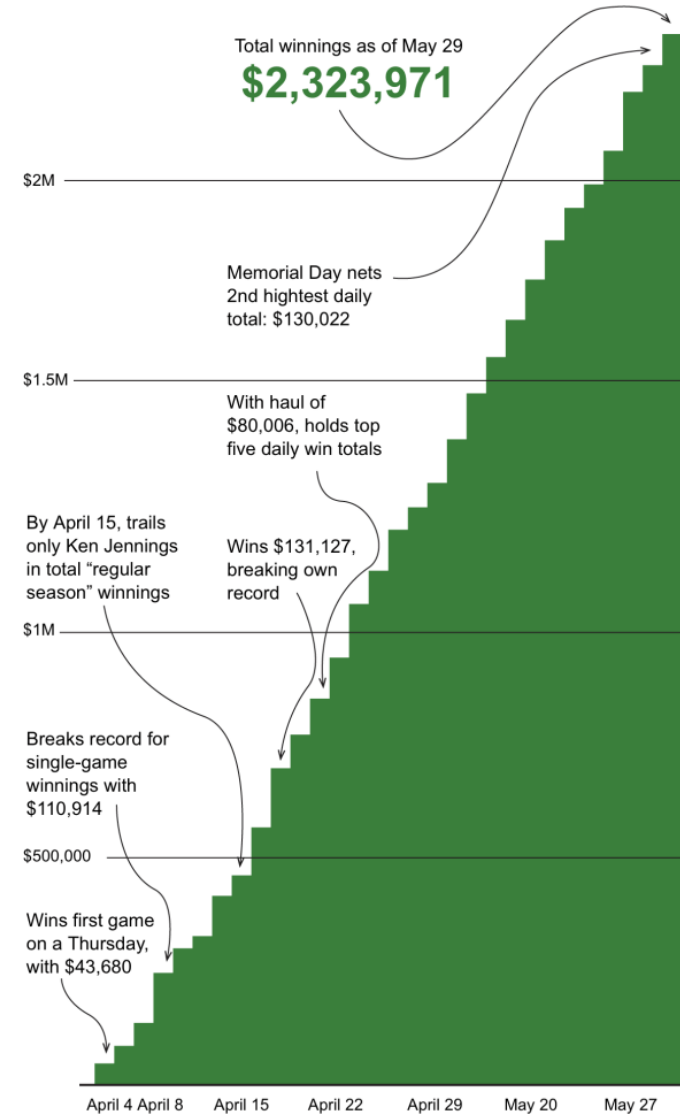
Subset an array on a specific condition

Outlier!

James Holzhauser

Holzhauser's winning streak

Each bar rises by the amount won per show.



Tip 6:

Evaluating categorical variables

- Determine the scope of variables at your disposal
- What unique groupings are there?
- Any relevant frequency counts?
- What story can be told ?

Code:

- `df['x'].nunique()`
- `df['x'].apply(function)`
- `df['x'].groupby()`
- `df['x'].agg(['count'])`
- `df['x'].sort_values()`

Tip 6:

Evaluating categorical variables

- Determine the scope of variables at your disposal
- What unique groupings are there?
- Any relevant frequency counts?
- What story can be told ?

Code:

- `df['x'].nunique()`
- `df['x'].apply(function)`
- `df['x'].groupby()`
- `df['x'].agg(['count'])`
- `df['x'].sort_values()`

Calculate number of unique groups in a variable


Tip 6:

Evaluating categorical variables

- Determine the scope of variables at your disposal
- What unique groupings are there?
- Any relevant frequency counts?
- What story can be told ?

Code:

- `df['x'].nunique()`
- `df['x'].apply(function)`
- `df['x'].groupby()`
- `df['x'].agg(['count'])`
- `df['x'].sort_values()`



Apply a function to a specified variable


Tip 6:

Evaluating categorical variables

- Determine the scope of variables at your disposal
- What unique groupings are there?
- Any relevant frequency counts?
- What story can be told ?

Code:

- `df['x'].nunique()`
- `df['x'].apply(function)`
- `df['x'].groupby()`
- `df['x'].agg(['count'])`
- `df['x'].sort_values()`



Group by a specified variable


Tip 6:

Evaluating categorical variables

- Determine the scope of variables at your disposal
- What unique groupings are there?
- Any relevant frequency counts?
- What story can be told ?

Code:

- `df['x'].nunique()`
- `df['x'].apply(function)`
- `df['x'].groupby()`
- `df['x'].agg(['count'])`
- `df['x'].sort_values()`



Create a summary table breakout

Tip 6:

Evaluating categorical variables

- Determine the scope of variables at your disposal
- What unique groupings are there?
- Any relevant frequency counts?
- What story can be told ?

Code:

- `df['x'].nunique()`
- `df['x'].apply(function)`
- `df['x'].groupby()`
- `df['x'].agg(['count'])`
- `df['x'].sort_values()`



Sort a table

Tip #7:

*Build
graphics to
explore
ideas*

- Visualize and communicate your results
- Simple and to the point
- Clearly label you axes and titles!

Code:

- `px.bar()`
- `px.line()`
- `plt.box()`
- `plt.scatter()`
- `chloropleth()`

Tip #7:

*Build
graphics to
explore
ideas*

- Visualize and communicate your results
- Simple and to the point
- Clearly label you axes and titles!

Code:

- `px.bar()`
- `px.line()`
- `plt.box()`
- `plt.scatter()`
- `chloropleth()`



Create a barplot

Tip #7:

*Build
graphics to
explore
ideas*

- Visualize and communicate your results
- Simple and to the point
- Clearly label you axes and titles!

Code:

- `px.bar()`
- `px.line()`
- `plt.box()`
- `plt.scatter()`
- `chloropleth()`



Create a line plot

Tip #7:

*Build
graphics to
explore
ideas*

- Visualize and communicate your results
- Simple and to the point
- Clearly label you axes and titles!

Code:

- `px.bar()`
- `px.line()`
- `plt.box()`
- `plt.scatter()`
- `chloropleth()`



Create a boxplot

Tip #7:

*Build
graphics to
explore
ideas*

- Visualize and communicate your results
- Simple and to the point
- Clearly label you axes and titles!

Code:

- `px.bar()`
- `px.line()`
- `plt.box()`
- `plt.scatter()`
- `chloropleth()`



Create a scatter plot

Tip #7:

*Build
graphics to
explore
ideas*

- Visualize and communicate your results
- Simple and to the point
- Clearly label you axes and titles!

Code:

- `px.bar()`
- `px.line()`
- `plt.box()`
- `plt.scatter()`
- `choropleth()`



Create a map

Additional Resources

- Reticulate Package (use Python in R!)
 - <https://rstudio.github.io/reticulate/index.html>
- Install Packages in Anaconda
 - <https://www.geeksforgeeks.org/python-add-packages-to-anaconda-environment/>
 - When in doubt, try using *conda forge* command
- Attributes vs methods in Python
 - <https://stackoverflow.com/questions/28798781/differences-between-data-attributes-and-method-attributes>



*THANK
YOU*

