Jupyter ~~[illegible]~~ HackerRank

Jupyter comes with abilities to convert to other formats

jupyter nbconvert -- to FORMAT mynotebook.ipynb

Slideshow

jupyter nbconvert notebook.ipynb -- to slides

Numpy → arrays / work with numerical values

Pandas → manipulate / analyze data

Matpotlib → visualize data

→ /c/Users/abahn/AppData/Roaming/SPB_Data

Process

1) Question 2) Wrangle make sure that data is in good quality

A) Gather B) Assess C) Clean

3) Explore patterns & relationships exploratory data analysis

4) Conclusions Descriptive Statistics 5) Communications

Packages

Questions

CSV Files

df = pd.read_csv('csv file', , separator if another separator other than comma is used

, header pick a line to be header

, index_col specify 1st column

replace column names → header = 0, names = labels

df.info() summary of dataframe

.nunique() number of unique values in each column

.describe() descriptive analysis for each column

Select multiple columns

df = df.iloc[:, np.r_[: , : , single line]]

View index number & label for each column

```
for i, v in enumerate(df.columns):
    print(i, v)
```

if diagnosis refers to an element as object

type(df[' '][0]) for further investigation

Clean Data

1) missing data

.fillna(mean, inplace=True)

2) Duplicates

df.duplicated()

df.drop_duplicates(inplace=True)

for large datasets

sum(df.duplicated())

3) incorrect datatypes

pd.to_datetime()

Plotting %matplotlib inline

quick histogram view

.hist(figsize = (,));

.plot(kind = ' ') bar, scatter, pie, box

pd.plotting.scatter_matrix(____ , figsize = (15, 15));

show all possible relations

.plot(x= , y=)

Conclusions descriptive statistics & visualizations

```
df_M = df[df['diagnosis'] == 'M']
```
boolean indexing

Communication

to sort graphs according to index

```
ind = df_a['education'].value_counts().index
df_a['education'].value_counts()[ind].plot(kind = "")
```

→ Gathering

- **TSV** tab seperated files
- **CSV** comma seperated values

File Types

Flat Files → simple, human readable, software readable

data in plain text format with one data record per line

Web Scraping get readable html using code

span → groups text together

to know which encoding to use → `<meta charset = "__">`

```
from bs4 import BeatifulSoup
with open('rt_html/                ') as file :
    soup = BeatifulSoup(file, 'lxml')
soup.find('    ').contents[0]
```

returns a list of tags' children

- Download Files using codes

```
import requests                    # creates request
url = '____'
response = requests.get(url)
```

```
import os                          # creates folder
folder_name = '____'
if not os.path.exists(  ):
    os.makedirs(folder_name)
```

access content & write to a file

```
with open(os.path.join(folder_name, url.split('/')[-1]), mode = 'wb')
    as file :
        file.write(response.content)
```

glob

```
for ebert_review in glob.glob('      /*.txt'):
    with open(ebert-review, encoding = ' ') as file:
```

file.read

file.readline

after read the file

put the variables in a dictionary

put dictiontionary in a list

add list to dataframe

```
with open (....) as f:
    for line in f:
        # Do something with 'line'
```

API (application programming interface)

→ choose API over scrapping

we use wikipedia API MediaWiki

"access library ease work for programmers

MediaWiki

wptools

- page("last bit of wikipedia code")
- get() gets all data

JSON

great for representing complicated data hierarchies

greater than

```
df.query('__ > __')
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

more visualization controls

```
plt.bar(x-coord , bar heights)
```

→ specify names for x-coordinates

```
plt.xticks([1,2,3], ['', '', ''])
```

- title('__')
- xlabel('__')
- ylabel('__');
- tick_label

normalize produces proportions

```
pd.cut(df['__'] ,          , labels =      )
```

column name — values — strings that represents