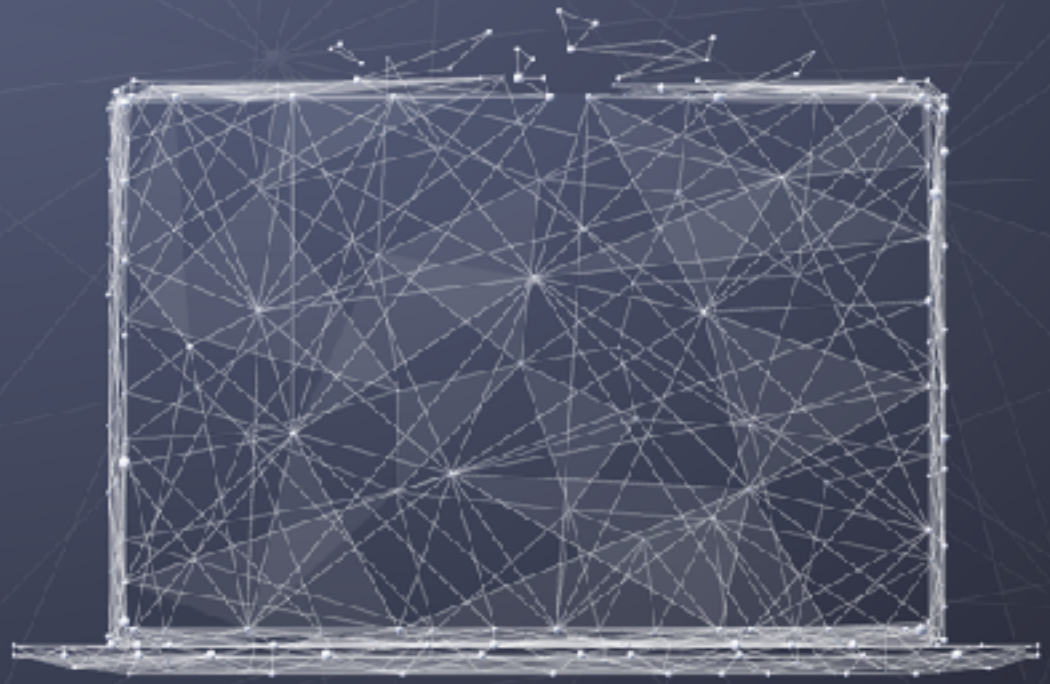
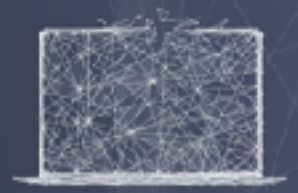
Data Science
Data Engineering I

Reading in data

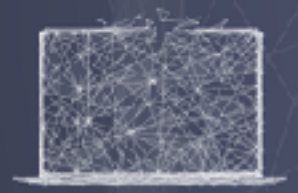


PURDUE UNIVERSITY | College of Science



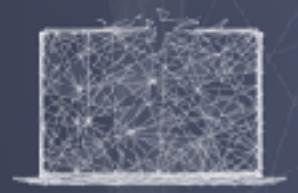
Initial data tasks

- Read in data from file
- Parse and store in internal data structure
- Select rows and/or columns
- Sort data
- Find values that match a target pattern
- Aggregate values
- Transform values, construct features



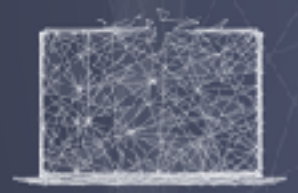
Introduction to PANDAS

- PANDAS is a Python data analysis library
- Uses data frames like R
- Can read and parse CSV and JSON easily, XML is a little more difficult



Data frames

- Data frames are a data structure used for storing tabular data
- Used in Pandas and R (statistical package)
- 2-dimensional labeled data structure, organized into rows and columns
- Each row corresponds to an example
- Each column contains data for a specific variable
- Can access a row, column, cell, or subset



PANDAS: Read in data

```
import pandas as pd
# read in data in csv format (sep=',' is default)
data = pd.read_csv("oscar_age_female.csv")

# print data
data
```

	Index	Year	Age	Name	Movie
0	1	1928	22	Janet Gaynor	Seventh Heaven,Street Angel and Sunrise: A Son...
1	2	1929	37	Mary Pickford	Coquette
2	3	1930	28	Norma Shearer	The Divorcee
3	4	1931	63	Marie Dressler	Min and Bill
4	5	1932	32	Helen Hayes	The Sin of Madelon Claudet
5	6	1933	26	Katharine Hepburn	Morning Glory

pandas.read_csv

pandas.**read_csv**(*filepath_or_buffer, sep=', ', delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, skipfooter=0, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar='"', quoting=0, doublequote=True, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=None, error_bad_lines=True, warn_bad_lines=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None*) ¶ [source]

Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file into chunks.

Additional help can be found in the online docs for IO Tools.

filepath_or_buffer : *str, path object, or file-like object*

Any valid string path is acceptable. The string could be a URL. Valid URL schemes include http, ftp, s3, and file. For file URLs, a host is expected. A local file could be: file://localhost/path/to/table.csv.

If you want to pass in a path object, pandas accepts either `pathlib.Path` or `py._path.local.LocalPath`.

By file-like object, we refer to objects with a `read()` method, such as a file handler (e.g. via builtin `open` function) or `StringIO`.

sep : *str, default ','*

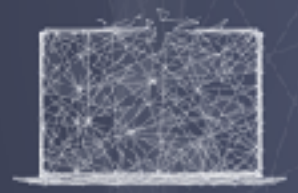
Delimiter to use. If sep is None, the C engine cannot automatically detect the separator, but the Python parsing engine can, meaning the latter will be used and automatically detect the separator by Python's builtin sniffer tool, `csv.Sniffer`. In addition, separators longer than 1 character and different from `'\s+'` will be interpreted as regular expressions and will also force the use of the Python parsing engine. Note that regex delimiters are prone to ignoring quoted data. Regex example: `'\r\t'`.

delimiter : *str, default None*

Alias for sep.

header : *int, list of int, default 'infer'*

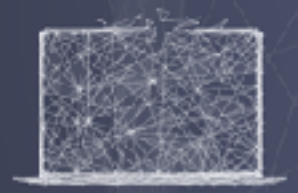
Row number(s) to use as the column names, and the start of the data. Default behav-



Reading in CSV files

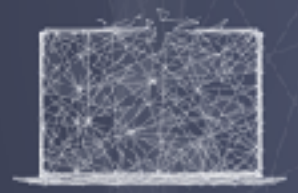
- PANDAS.read_csv function takes a separator (sep) as an argument
  - Default is sep=“,” because comma separated files are most common
  - Other choices include tab (“\t”), space (“ “) and semi-colon (“;”)

- Delimiters in text fields
  - Sometimes separator characters will appear in longer text fields
  - To treat those characters as part of the text value rather than a separator, the fields should be delimited with a quote character (e.g., quotechar=““)
  - E.g., 1,1928,22,"Janet Gaynor","Seventh Heaven,Street Angel and Sunrise: A Song of Two Humans"



File locations

- Pandas.read_csv looks for the file argument that is given to the function (e.g., “oscar_age_female.csv”) in the current working directory, unless a path is explicitly specified
  - The working directory is typically the directory that you started your Jupyter notebook from
- When specifying file names with paths, you can use either absolute or relative paths
  - A relative path specifies the path to the file starting from your current working directory, e.g., “data/movies/oscar_age_female.csv”
  - An absolute path specifies the complete path from the base of your file system to the file, e.g., “/Users/neville/data/movies/oscar_age_female.csv”

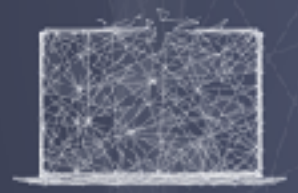


PANDAS: Accessing data

```
# get numbers of rows
len(data)
89

# get names of columns
data.columns
Index(['Index', 'Year', 'Age', 'Name', 'Movie'], dtype='object')

# columns can be accessed by name
data[‘Age']
data.Age
```



PANDAS: Accessing data

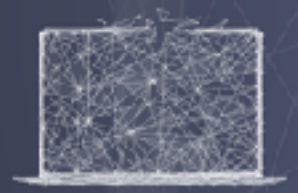
```
# head and tail return first/last k rows of column
data.Name.head(5)
0      Janet Gaynor
1      Mary Pickford
2      Norma Shearer
3      Marie Dressler
4      Helen Hayes
Name: Name, dtype: object
```

```
# use [] to select via slices
data[:3]   #first three rows
```

	Index	Year	Age	Name	Movie
0	1	1928	22	Janet Gaynor	Seventh Heaven,Street Angel and Sunrise: A Son...
1	2	1929	37	Mary Pickford	Coquette
2	3	1930	28	Norma Shearer	The Divorcee

```
# select by position with iloc
data.iloc[3:5,1:4]   #row slice, column slice
```

	Year	Age	Name
3	1931	63	Marie Dressler
4	1932	32	Helen Hayes



PANDAS: Accessing data

```
data.Age.head(5)
0    22
1    37
2    28
3    63
4    32

# sort results
data.Age.sort_values().head(5)
59    21
0     22
85    22
25    24
13    24
```

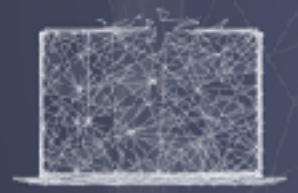
PURDUE UNIVERSITY | College of Science

PANDAS: Accessing data

```
# summarize the data
data.describe()
```

	Index	Year	Age
count	89.000000	89.000000	89.000000
mean	45.000000	1972.000000	36.123596
std	25.836021	25.836021	11.745231
min	1.000000	1928.000000	21.000000
25%	23.000000	1950.000000	28.000000
50%	45.000000	1972.000000	33.000000
75%	67.000000	1994.000000	41.000000
max	89.000000	2016.000000	80.000000

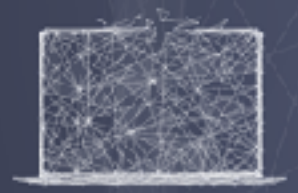
```
data.Age.mean()
data.Year.sum()
36.12359550561798
175508
```



PANDAS: Accessing data in a loop

```
for i in range(10):  # loop over first ten rows of data frame
    tmpyr = data.iloc[i,1]
    tmpage = data.iloc[i,2]
    tmpname = data.iloc[i,3]
    print(tmpname + ": current age (if still alive) =" + str(2019-tmpyr+tmpage))
```

```
Janet Gaynor: current age (if still alive) = 113
Mary Pickford: current age (if still alive) = 127
Norma Shearer: current age (if still alive) = 117
Marie Dressler: current age (if still alive) = 151
Helen Hayes: current age (if still alive) = 119
Katharine Hepburn: current age (if still alive) = 112
Claudette Colbert: current age (if still alive) = 116
Bette Davis: current age (if still alive) = 111
Luise Rainer: current age (if still alive) = 110
Luise Rainer: current age (if still alive) = 110
```



PANDAS: Write data out to file

```
# getting data out of pandas, output to csv file
data.to_csv('oscar_age_female_mod.csv', sep=',')
```