CS 38003 PYTHON PROGRAMMING

Ruby Tahboub

PANDAS

INTRO TO PANDAS

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

material obtained from CS242

DATA FRAMES

- Data frames are a data structure used for storing tabular data.
- Used in R and Pandas.
- 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.

material obtained from CS242

GETTING STARTED WITH PANDAS

```
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

material obtained from CS242

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
```

material obtained from CS242

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

# 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

material obtained from CS242

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
```

material obtained from CS242

PANDAS: ADDING COLUMNS

```
# add a column to a data frame with a constant value
data['Tmp'] = 'testing'
```

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

```
# add a column with a value calculated from other column
data['Tmp2']=data.Year+1
data.head()
```

	Index	Year	Age	Name	Movie	Tmp	Tmp2
0	1	1928	22	Janet Gaynor	Seventh Heaven,Street Angel and Sunrise: A Son...	testing	1929
1	2	1929	37	Mary Pickford	Coquette	testing	1930
2	3	1930	28	Norma Shearer	The Divorcee	testing	1931
3	4	1931	63	Marie Dressler	Min and Bill	testing	1932
4	5	1932	32	Helen Hayes	The Sin of Madelon Claudet	testing	1933

material obtained from CS242

PANDAS: ADDING ROWS

```
# construct new example as a Series object
newData = [89, 2017, 29, ‘Emma Stone', ‘La La Land']
idx = list(data.columns)
newExample = pd.Series(newData, index=idx)
print(newExample)

# add rows by appending Data Frame or Data Series, returns new object
data2 = data.append(newExample, ignore_index=True)
data2.tail()
```

	Index	Year	Age	Name	Movie
85	86	2013	22	Jennifer Lawrence	Silver Linings Playbook
86	87	2014	44	Cate Blanchett	Blue Jasmine
87	88	2015	54	Julianne Moore	Still Alice
88	89	2016	26	Brie Larson	Room
89	89	2017	29	Emma Stone	La La Land

material obtained from CS242

SUBSETTING DATA

PANDAS: FILTERING

```
# select only rows of interest
data[data.Year > 2000]

data[(data.Year > 2000) & (data.Age <30)]
```

	Index	Year	Age	Name	Movie
73	74	2001	33	Julia Roberts	Erin Brockovich
74	75	2002	35	Halle Berry	Monster's Ball
75	76	2003	35	Nicole Kidman	The Hours
76	77	2004	28	Charlize Theron	Monster
77	78	2005	30	Hilary Swank	Million Dollar Baby
78	79	2006	29	Reese Witherspoon	Walk the Line
79	80	2007	61	Helen Mirren	The Queen
80	81	2008	32	Marion Cotillard	La Vie en rose
81	82	2009	33	Kate Winslet	The Reader
82	83	2010	45	Sandra Bullock	The Blind Side
83	84	2011	29	Natalie Portman	Black Swan
84	85	2012	62	Meryl Streep	The Iron Lady
85	86	2013	22	Jennifer Lawrence	Silver Linings Playbook
86	87	2014	44	Cate Blanchett	Blue Jasmine
87	88	2015	54	Julianne Moore	Still Alice
88	89	2016	26	Brie Larson	Room

material obtained from CS242

PANDAS: SELECTING COLUMNS

```
# select a subset of columns by name
data[[‘Age’,’Movie']]

# another way to select columns by name
data.filter(items=['Year', 'Name'])

# make new attribute that is True for all movies in even years
data['IsOddYr'] = data.Year%2==1
data.head()
```

	Age	Movie
0	22	Seventh Heaven,Street Angel and Sunrise: A Son...
1	37	Coquette
2	28	The Divorcee
3	63	Min and Bill
4	32	The Sin of Madelon Claudet

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

material obtained from CS242

PANDAS: SELECTING A SUBSET OF ROWS

```
# select based on Boolean conditions
data[(data.IsOddYr) & ((data.Age==29)|(data.Age==42))]
```

	Index	Year	Age	Name	Movie	IsOddYr
47	48	1975	42	Ellen Burstyn	Alice Doesn't Live Here Anymore	True
63	64	1991	42	Kathy Bates	Misery	True
83	84	2011	29	Natalie Portman	Black Swan	True

```
# select rows uses regular expression
data[data.Name.str.contains('^Kath')]
```

	Index	Year	Age	Name	Movie	IsOddYr
5	6	1933	26	Katharine Hepburn	Morning Glory	True
39	40	1967	60	Katharine Hepburn	Guess Who's Coming to Dinner	True
40	41	1968	61	Katharine Hepburn	The Lion in Winter	False
54	55	1982	74	Katharine Hepburn	On Golden Pond	False
63	64	1991	42	Kathy Bates	Misery	True

material obtained from CS242

PANDAS: SELECTING A SUBSET OF ROWS

```
# get a random sample of ten rows
data.sample(10)
```

	Index	Year	Age	Name	Movie	IsOddYr
80	81	2008	32	Marion Cotillard	La Vie en rose	False
20	21	1948	32	Jane Wyman	Johnny Belinda	False
5	6	1933	26	Katharine Hepburn	Morning Glory	True
56	57	1984	49	Shirley MacLaine	Terms of Endearment	False
52	53	1980	33	Sally Field	Norma Rae	False
66	67	1994	36	Holly Hunter	The Piano	False
40	41	1968	61	Katharine Hepburn	The Lion in Winter	False
86	87	2014	44	Cate Blanchett	Blue Jasmine	False
84	85	2012	62	Meryl Streep	The Iron Lady	False
54	55	1982	74	Katharine Hepburn	On Golden Pond	False

material obtained from CS242

CLEANING DATA

PANDAS: FIND MISSING VALUES

```
data2 = pd.read_csv("grades.csv")
#find missing values by looking at non-null counts
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16 entries, 0 to 15
Data columns (total 9 columns):
Lastname     16 non-null object
Firstname    16 non-null object
SSN          16 non-null object
Test1        16 non-null float64
Test2        13 non-null float64
Test3        15 non-null float64
Test4        15 non-null float64
Final        16 non-null float64
Grade        16 non-null object
dtypes: float64(5), object(4)
memory usage: 1.2+ KB
```

```
# drop all rows with missing values
data2.dropna()
```

	Lastname	Firstname	SSN	Test1	Test2	Test3	Test4	Final	Grade
0	Alfalfa	Aloysius	123-45-6789	40.0	90.0	100.0	83.0	49.0	D-
1	Alfred	University	123-12-1234	41.0	97.0	96.0	97.0	48.0	D+
2	Gerty	Gramma	567-89-0123	41.0	80.0	60.0	40.0	44.0	C
3	Android	Electric	087-65-4321	42.0	23.0	36.0	45.0	47.0	B-
4	Bumpkin	Fred	456-78-9012	43.0	78.0	88.0	77.0	45.0	A-
5	Rubble	Betty	234-56-7890	44.0	90.0	80.0	90.0	46.0	C-
7	Buff	Bif	632-79-9939	46.0	20.0	30.0	40.0	50.0	B+
11	Dandy	Jim	087-75-4321	47.0	25.0	23.0	36.0	45.0	C+
12	Elephant	Ima	456-71-9012	45.0	29.0	78.0	88.0	77.0	B-
13	Franklin	Benny	234-56-2890	50.0	10.0	90.0	80.0	90.0	B-
15	Heffalump	Harvey	632-79-9439	30.0	16.0	20.0	30.0	40.0	C

material obtained from CS242

PANDAS: FILL IN MISSING VALUES

```
# fill missing values in with a particular value
data2.fillna(0)
```

```
# fill in missing values with average score
test2avg = data2.Test2.mean()
data2.Test2 = data2.Test2.fillna(test2avg)
```

	Lastname	Firstname	SSN	Test1	Test2	Test3	Test4	Final	Grade
0	Alfalfa	Aloysius	123-45-6789	40.0	90.000000	100.0	83.0	49.0	D-
1	Alfred	University	123-12-1234	41.0	97.000000	96.0	97.0	48.0	D+
2	Gerty	Gramma	567-89-0123	41.0	80.000000	60.0	40.0	44.0	C
3	Android	Electric	087-65-4321	42.0	23.000000	36.0	45.0	47.0	B-
4	Bumpkin	Fred	456-78-9012	43.0	78.000000	88.0	77.0	45.0	A-
5	Rubble	Betty	234-56-7890	44.0	90.000000	80.0	90.0	46.0	C-
6	Noshow	Cecil	345-67-8901	45.0	11.000000	NaN	4.0	43.0	F
7	Buff	Bif	632-79-9939	46.0	20.000000	30.0	40.0	50.0	B+
8	Airpump	Andrew	223-45-6789	49.0	46.384615	90.0	100.0	83.0	A
9	Backus	Jim	143-12-1234	48.0	46.384615	97.0	96.0	97.0	A+
10	Carnivore	Art	565-89-0123	44.0	46.384615	80.0	60.0	40.0	D+
11	Dandy	Jim	087-75-4321	47.0	25.000000	23.0	36.0	45.0	C+

material obtained from CS242

AGGREGATING DATA

PANDAS: GROUPBY

```
# drop +/- from grades
tmp = data2.Grade
tmp2 = tmp.str.replace('-','')
tmp3 = tmp2.str.replace('+','')
data2.Grade = tmp3

# find unique values
data2.Grade.unique()

# group by final grade and calculate avg Test1 score
for grade, grade_data in data2.groupby("Grade"):
    print(grade, grade_data.Test1.mean())
```

```
A 46.666666666666664
B 44.6
C 40.5
D 41.666666666666664
F 45.0
```

When you iterate over the results of a groupBy, each result is a tuple: *first element is a unique value, second element is a DataFrame filtered by that value*

material obtained from CS242

PANDAS: GROUPBY

```
# can also aggregate directly over group object
groups = data2.groupby("Grade")
groups.agg('sum')
```

Grade	Test1	Test2	Test3	Test4	Final
A	140.0	170.769231	275.0	273.0	225.0
B	223.0	116.000000	245.0	253.0	268.0
C	162.0	211.000000	183.0	196.0	175.0
D	125.0	233.384615	276.0	240.0	137.0
F	45.0	11.000000	NaN	4.0	43.0

```
# applying multiple aggregators
groups.Final.agg(['min', 'max'])
```

Grade	min	max
A	45.0	97.0
B	4.0	90.0
C	40.0	46.0
D	40.0	49.0
F	43.0	43.0

material obtained from CS242

LAMBDA FUNCTIONS

- Lambda functions are functions without names, for use in situations where the function will be discarded and not used again
- Example:
  `lambda x: x > 0`
- The term lambda makes the temporary function, x is the parameter name, and the code after : denotes what to do
- Often used in **`apply( )`** when transforming data

material obtained from CS242

PANDAS: MAP & APPLY

```
# create a rounding function, apply to each value in Test1
rndGrade = lambda x: int(x / 10) * 10
data2.Test1.map(rndGrade)
```

```
# apply function to more than one column of data frame
data2[['Test1','Final']].apply('sum')
```

```
# standardize Final grade by subtracting mean and divding by stdev
avgFinal = data2.Final.mean()
stdFinal = data2.Final.std()
print(avgFinal,stdFinal)
data2.Final.map(lambda x: (x - avgFinal)/stdFinal)
```

```
0    -0.173596
1    -0.216995
2    -0.390591
3    -0.260394
4    -0.347192
5    -0.303793
6    -0.433990
7    -0.130197
8     1.301971
9     1.909557
10   -0.564187
11   -0.347192
12    1.041577
13    1.605764
14   -2.126552
15   -0.564187
Name: Final, dtype: float64
```


material obtained from CS242

WRITING DATA TO A FILE

```
# getting data out of pandas, output to csv
data2.to_csv('grades_mod.csv', sep='\t')
```

THANK YOU!