Handout 10

Intro to Pandas – processing tables

Pandas library provides efficient operations on multidimensional arrays of heterogeneous data with attached row and column labels. Built on top of NumPy.

Described in https://pandas.pydata.org/pandas-docs/stable/index.html

```
import pandas
import pandas as pd
```

Main data structures are `Series` (an indexed column of values) and `DataFrame` (represents a table with indexed rows and named columns).

Content of popgdp.csv

```
country,year,population,gdp
us,2017,325.1,19.485
us,2018,327.1,20.612
us,2019,329.1,21.433
ca,2017,36.7,1.65
ca,2018,37.1,1.716
ca,2019,37.4,1.736
mx,2017,124.8,1.159
mx,2018,126.2,1.222
mx,2019,127.6,1.269
```

as a DataFrame

	country	year	population	gdp
0	us	2017	325.1	19.485
1	us	2018	327.1	20.612
2	us	2019	329.1	21.433
3	ca	2017	36.7	1.650
4	ca	2018	37.1	1.716
5	ca	2019	37.4	1.736
6	mx	2017	124.8	1.159
7	mx	2018	126.2	1.222
8	mx	2019	127.6	1.269

```
'''Simple demo of pandas, uses file popgdp.csv'''
import pandas as pd

def main():
    df = pd.read_csv("popgdp.csv") #try also popgdp1.csv
    print(type(df))
    print(df)
    print(df.columns)
    print(df.index)
    print(df.info())


    #select columns
    print (df.country)
    print (df["country"])
    print(df[["year", "country"]])

    #select subset of columns and rows
    print(df.loc[ 2:5 ,["year", "country"] ])

    # filter: select rows based on condition
    df1 = df[df.year >= 2018]
    df2 = df[(df.year >= 2018) & (df.country == 'us')]
    df3 = df[(df.year >= 2018)| (df.country == 'us')]

    print(df1,df2,df3, sep = '\n')
```

```
    # add a new column
    df["perCapitaGDP"] = df.gdp / df.population
    print (df)

    #sort
    df.sort_values(by = "gdp", ascending = False)

    #statistical values
    print("Min value in the whole gdp column", df.gdp.min())
    print("Min gdp by country:", df.groupby(by = 'country').gdp.min())

    #join two data frames with matching column names by that col
    langdf = pd.read_csv("lang.csv")
    dfWithLang  = pd.merge(df, langdf)
    print(dfWithLang)
    #save data frame into a file
    dfWithLang.to_csv("popgdpwithlang.csv", index = False)


main()
```

DATAFRAME

A two-dimensional array with **labeled rows (index)** and **columns**. Each column is of type **Series**.

ACCESSING COLUMNS, SLICING AND SELECTION, LOC AND ILOC

```
#select columns
print (df.country)
print (df["country"])
print(df[["year", "country"]])

#select subset of columns and rows
print(df.loc[ 2:5 ,["year", "country"] ])
print(df.iloc[ 2:5 , 0:2 ])
```

`loc [rowLabels, colLabels]`- *label* based indexing
`iloc [rowNum, Colnum]` - *integer index* based indexing

FILTERING BY CONDITION, SORTING

- like the SQL *Where* clause; and is &. or is |, not is ~

```
~ not
& and
| or
```

```
# filter: select rows based on condition
df1 = df[df.year >= 2018]
df2 = df[(df.year >= 2018) & (df.country == 'us')]
df3 = df[(df.year >= 2018)| (df.country == 'us')]
```

```
#sort
df.sort_values(by = "gdp", ascending = False)
```

ADDING COLUMNS, VECTOR-BASED OPERATION

**** Note - function descriptions below are not complete**

`df[newcolname] = val`

```
df["perCapitaGDP"] = df.gdp / df.population
```

AGGREGATION, GROUPBY

```
#statistical values
print("Min value in the whole gdp column", df.gdp.min())
print("Min gdp by country:", df.groupby(by = 'country').gdp.min())
```

Function	Description
count	Number of non-NA observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Bessel-corrected sample standard deviation
var	Unbiased variance
quantile	Sample quantile (value at %)

COMBINING DATA: MERGE AND JOIN

Merge function provides SQL-style 'join' capabilities, based on equality of column or index values.

https://pandas.pydata.org/pandas-docs/stable/merging.html

```
dfWithLang  = pd.merge(df, langdf)
```

`pd.merge(leftDF, rightDF, how='inner')`
Similar to SQL join; also **`leftDF.join(leftDF)`**

PRACTICE

- Explore files **`djia.csv, djia-prices.csv`** and practice to perform a variety of tasks: filtering, grouping to compute aggregate values, sorting, joining tasks.

WORKING WITH MISSING DATA

Real world data is rarely complete and noiseless/homogenous (i.e. error-free).
https://pandas.pydata.org/pandas-docs/stable/missing_data.html

Missing data can be referred to as:
`None` – 'Pythonic no-value' constant
`NaN`- Not a Number, a **floating point value**, **`numpy.nan`**
NA, null – are **not** within Python/numpy/pandas vocab

- **Numerical** functions and operations, e.g. sum, max, etc. will perform fine on **NaN,** but generate an error with **None.** Non-numerical functions will fail, generating an error.

`pd.isnull(v)`	Check if value is None or np.nan, return a boolean
`pd.isna(v)`	
	The opposite of the above functions.
`pd.notnull(v)`	
`pd.notna(v)`	

- Removing or filling-in missing values in a data frame

`dataframe.dropna(inplace=True)`	Returns a DataFrame with rows containing missing values as specified by parameters, dropped.
`dataframe.fillna(inplace=True)`	Returns a DataFrame with rows containing missing values filled with value as specified by parameters.

STRING FUNCTIONS IN PANDAS

- **They are accessed with `str` prefix** and generally have names matching the analogous built-in string methods, i.e**. str.upper()** instead of upper.
- Slicing using [::] syntax is done with method **`str.slice`** () – works on strings AND lists
- Indexing - **`str.get()`** – works on strings AND lists
- These methods exclude missing/NA values automatically
- To convert a string into a number (e.g. "25"→ 25.0), use function **astype(float)**

http://pandas.pydata.org/pandas-docs/stable/text.html#text-string-methods

Method name	Description
astype(float)	Convert a string into a number (e.g. "25"→ 25.0)
cat	Concatenate strings element-wise with optional delimiter
contains	
count	Count occurrences of pattern

---

endswith, startswith Equivalent to x.endswith(pattern) or x.startswith(pattern) for each element.
get Index into each element (retrieve i-th element)
len
lower, upper
replace Replace occurrences of pattern/regex with some other string
slice Slice each string in the Series.
split Split strings on delimiter or regular expression
strip, rstrip, lstrip Trim whitespace, including newlines; equivalent to x.strip()