Libreria Pandas

Rachele Sprugnoli

rachele.sprugnoli@unipr.it

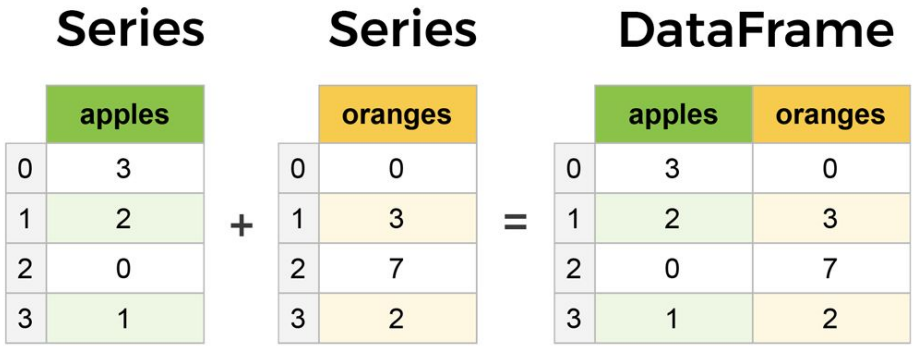
UNIVERSITÀ
DI PARMA

Pandas

- Libreria per la manipolazione di dati in formato sequenziale (*Series*) o tabellare (*Dataframe*): legge e salva file CSV, TSV, Excel…
  - contiene funzioni utili per gestire i dati mancanti, eseguire operazioni su colonne e righe e trasformare i dati
- Importazione libreria:
  - `import pandas as pd`
- Documentazione ufficiale: https://pandas.pydata.org/

Serie e dataframe

- Series: sequenza mono-dimensionale, ogni valore ha un indice
  - funziona un po' come una lista (si può accedere agli elementi in sequenza) e un po' come un dizionario (si può accedere a un elemento tramite il suo indice)
- Dataframe: tabella di oggetti eterogenei, l'equivalente bi-dimensionale di una Series.



Series e dataframe

SERIES	DATAFRAME
Una dimensione	Due dimensioni
Omogenea: gli elementi che la compongono devono essere dello stesso tipo	Eterogenea: gli elementi che la compongono possono essere di tipo diverso
Dimensione immutabile	Dimensione modificabile

Series

- `pd.Series()` → crea una Series, valori e/o indici sono specificati tra parentesi come segue:
  - solo valori, gli indici partono da 0

    ```
    pd.Series([2,3,5])
    ```

  - valori e indici separatamente

    ```
    pd.Series([2,3,5],index=[1,2,3])
    ```

  - valori e indici insieme, passando un dizionario

    ```
    pd.Series({"a": 2, "b": 3, "c" : 5})
    ```

Dataframe

- `pd.read_csv('file', delimiter='\t')` → crea un Dataframe da un file csv o tsv, il separatore di default è “,”
- `pd.read_csv('file', header=None)` → crea un Dataframe da un file csv o tsv senza riga delle intestazioni
- `pd.read_csv('file', usecols=["colonna"])` → crea un Dataframe da un file csv o tsv considerando solo una colonna
- `pd.read_excel('file.xlsx')` → crea un Dataframe da un file Excel
- `pd.read_excel('file.xlsx', sheet_name = 1)` → crea un Dataframe da un file Excel e considerando solo un foglio

Ottenere informazioni

- `df.columns` → intestazione delle colonne
- `df.info()` → informazioni sul Dataframe
- `s.count(), df.count()` → conta valori non vuoti
- `df.head(numero)` → mostra le prime n righe del Dataframe
- `df.tail(numero)` → mostra le ultime n righe del Dataframe

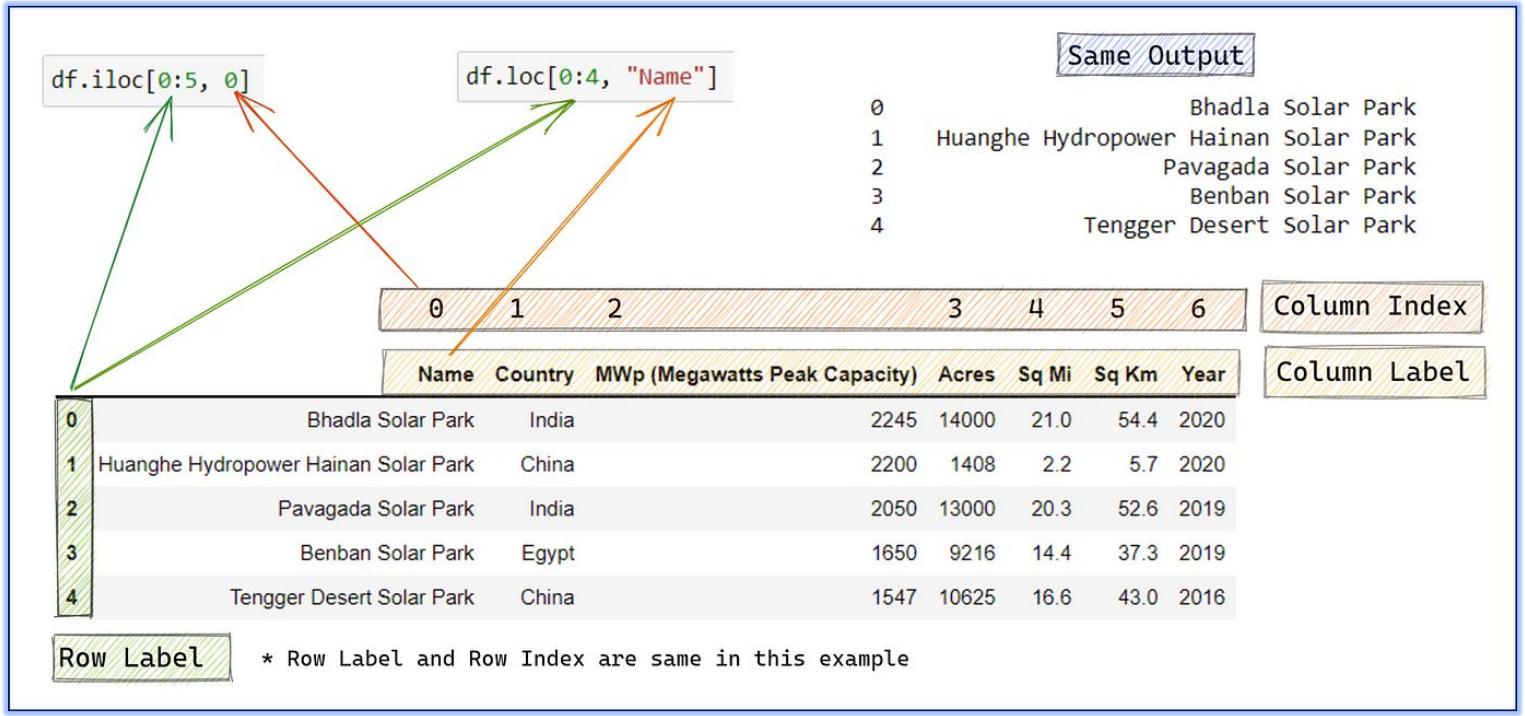
Statistiche

- `s.sum(), df.sum()` → somma di valori
- `s.min(), df.min()` → valore minimo
- `s.max(), df.max()` → valore massimo
- `s.mean(), df.mean()` → media dei valori
- `s.describe(), df.describe()` → riassunto di statistiche
  - percentile: quanti valori sono inferiori al percentile indicato, percentuale di valori che scendono al di sotto di un determinato valore (25, 50, 75)

Selezionare dati in un Dataframe

- `df["label"]` → seleziona la colonna con etichetta *label*
- `df.iloc[indice_row, indice_column]` → seleziona in base alla posizione
  - usa gli indici di posizione di righe e colonne
  - iloc = integer locate
- `df.loc[label_row, label_column]` → seleziona in base all'intestazione
  - usa le etichette di righe e colonneù
  - loc = locate

iloc versus loc



Da: https://towardsdatascience.com/select-data-in-python-pandas-easily-with-loc-iloc-c6e8bc979e3e

`iloc`: esempi

- `df.iloc[0,:]` → trova i dati della riga con indice 0 per tutte le colonne
- `df.iloc[:,0]` → trova i dati di tutte le righe per la colonna con indice 0
- `df.iloc[2,0]` → trova il contenuto della cella con indice di riga 2 e indice di colonna 0
- `df.iloc[0:3]` → trova le righe dalla posizione 0 alla posizione 2 (la posizione finale non è inclusa) per tutte le colonne
- `df.iloc[:,0:2]` → trova le colonne 0 e 1 per tutte le righe

`loc`: esempi

- `df.loc[:,"label"]` → trova tutte le righe della colonna *label*
- `df.loc[:,"label1":"label3"]`→ trova tutte le righe per le colonne
- `df.loc[0,"label"]`→ trova il contenuto della cella alla riga 0 e della colonna *label*
- `df.loc[0:5,"label"]`→ trova le righe da 0 a 5 della colonna *label*

Rimozione

- `s.drop(), df.drop()`
  - `df.drop(columns=["label"])` → rimuove una colonna
  - `df.drop(columns=["label1",”label2”])` → rimuove due colonne
  - `df.drop([0])` → rimuove una riga in base alla posizione
- `dropna` → rimuove le righe con valori mancanti (nan = not a number)
- `fillna(new_value)` → riempie le celle vuote con un nuovo valore

Raggruppare

- `df.groupby("label")` → raggruppa righe in base a una sola colonna
  - `df.groupby(["label1", "label2"])` → raggruppa righe in base a due colonne
- Si aggiungono dopo i metodi da applicare al raggruppamento, es.:
  - `df.groupby("label").sum()` → somma
  - `df.groupby("label").first()` → prima occorrenza
  - `df.groupby("label").get_group("string")` → trova un sottogruppo in base al valore del `get_group()`

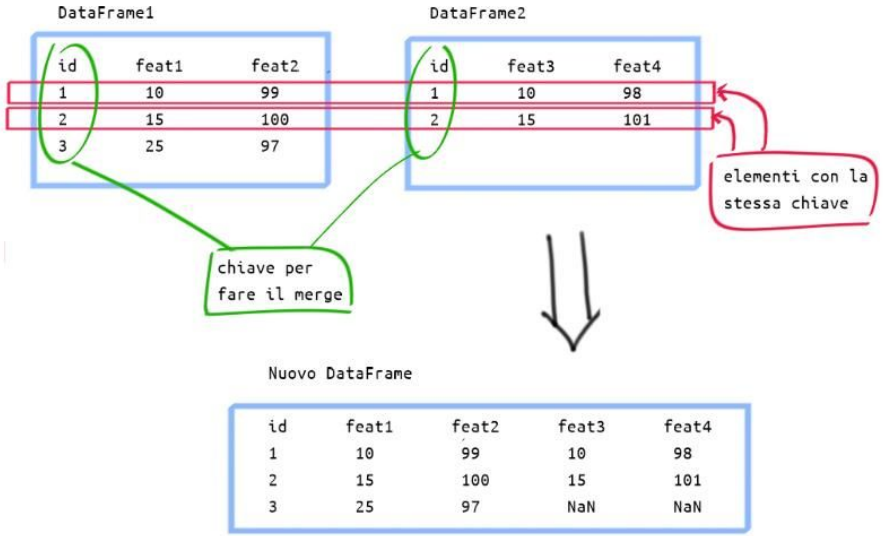
N.B. Da usare se si vogliono applicare dei metodi su una o più colonne

Filtrare

- `df.filter[("label")]` → restituisce solo le righe o le colonne specificate nel filtro creando una versione ridotta del Dataframe originale
  - `df.filter(["label1","label2"])` → filtra le colonne in base a due colonne
  - `df.filter(regex='o$')` → filtra le colonne in base a una regex

Merge

- Unire due dataframe in base a una chiave (in base all'etichetta della colonna), gli elementi dei due DataFrame con la stessa chiave vengono combinati in un'unica riga nel nuovo DataFrame



Da: https://www.1week4.com/it/python/pandas-dataframe-merge/

Merge

- `df1.merge(df2,on=["label","label"],how='left')`
  - `on` → colonne che fanno da chiave nei due Dataframe
  - `how` → il tipo di unione da effettuare
    - `inner`: usa le chiavi comuni ai due Dataframe
    - `outer`: usa le chiavi di entrambi di Dataframe
    - `left`: usa solo le chiavi del Dataframe a sinistra
    - `right`: usa solo le chiavi del Dataframe a destra

Approfondimento: https://datacomy.com/data_analysis/pandas/merge/

Grafici

- `s.plot(), df.plot()` → metodo per la creazione di grafici, di default crea un grafico a linea
  - `plot(kind="bar")`
  - `plot(kind="pie")`
  - `plot(kind="scatter")`
  - `plot(kind="hist")`
- `df.plot(kind="bar",title="Titolo grafico")` → l'argomento title specifica il titolo del grafico che verrà visualizzato sopra il grafico
- `df.plot(kind="bar",figsize=(20,5))` → l'argomento figsize specifica la grandezza del grafico, larghezza x altezza

Salvataggio

- `df.to_csv('nome-file.csv', sep="\t", encoding='utf-8')` → salvataggio del Dataframe specificando nome del file, separatore e codifica
- `plt.savefig("nome-file.png")` → salvataggio di un grafico in un file, è necessario importare il modulo Pyplot libreria Matplotlib (https://matplotlib.org/)


```
import matplotlib.pyplot as plt
…
plt.savefig("nome-file.png")
```

Un po’ di pratica

- Lezione8.ipynb

  https://colab.research.google.com/drive/1u9uPLjqz9PvATfdQBXMbXEliT39koupn?usp=sharing

Esercizio

- A partire dal file LatinAffectus4.tsv:
  - contare quanti nomi e quanti aggettivi sono presenti
  - cercare la riga con il lemma “purus”
  - cercare tutti le righe dei lemmi che iniziano con “in-”
  - salvare in un file un dataframe che contiene tutte le colonne tranne la colonna “provenance”
  - creare un grafico a barre basato sulla colonna “has_polarity”
  - salvare il grafico in un file png

Soluzione esercizio

- https://colab.research.google.com/drive/1cYsekw6eDluz10AWniMe2kLrFgDBQQNE?usp=sharing