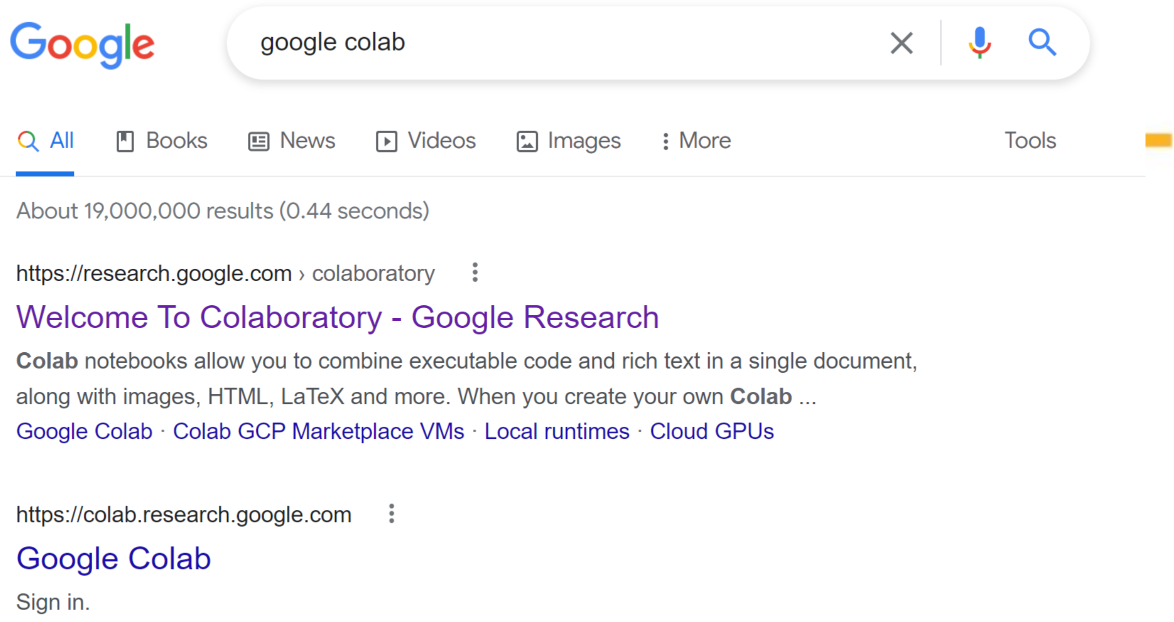
SJSU SAN JOSÉ STATE UNIVERSITY

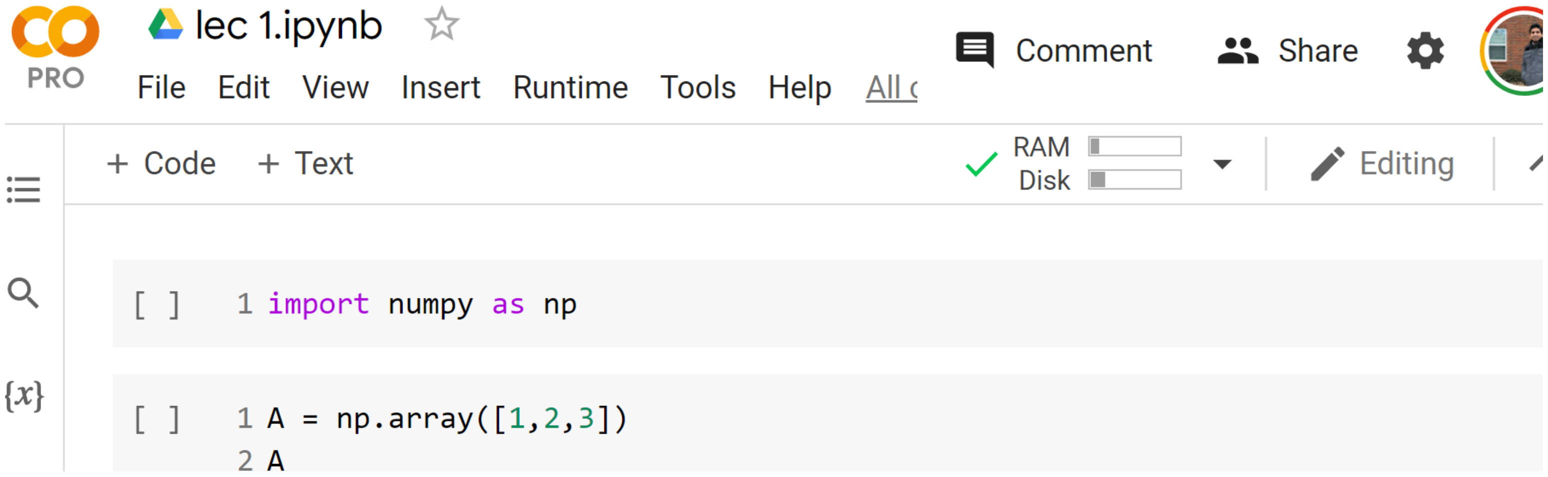
DATA 220
Mathematical Methods for Data Analytics

Dr. Mohammad Masum

Google CoLab



- Open CoLab
- File → New Notebook
- Sign in may require
  - Start coding !!!!



Numpy

- Fundamental package for scientific computing in python
- N-dimensional array object
- Linear algebra, random number capabilities
- Building block for other important packages – Scipy
- Open source

How to import NumPy

To access NumPy and its functions import it in your Python code like this:

```
import numpy as np
```

https://numpy.org/doc/stable/user/absolute_beginners.html

Numpy

- Numpy array – N-dimensional array - ndarray

```
1 import numpy as np
```

```
[4] 1 A = np.array([1,2,3])
    2 A
```

```
array([1, 2, 3])
```

1-D array (Vector)

```
[7] 1 list1= [1,2,3]
    2 A_from_list = np.array(list1)
    3 A_from_list
```

```
array([1, 2, 3])
```

Python list to array

```
[11] 1 A = np.array([[1,2], [3,5]])
     2 A
```

```
array([[1, 2],
       [3, 5]])
```

2-D array (Matrix)

```
[13] 1 A.shape
```

```
(2, 2)
```

```
[14] 1 A.dtype
```

```
dtype('int64')
```

Numpy

- How to create Basic arrays
  - np.zeros
  - np.ones

```
1 np.zeros(4)
```

```
array([0., 0., 0., 0.])
```

```
1 np.zeros((4,2))
```

```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

```
[19]  1 np.ones((4,2))
```

```
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

Numpy

- Basic- array creation
  - np.eye
  - np.arange

```
[20]  1 np.eye(4)

array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

```
[21]  1 np.arange(1,10,2)

array([1, 3, 5, 7, 9])
```

Numpy

- Basic- array creation
  - np.rand.random()
  - np.random.normal()

```
[30]  1 np.random.random((3,2))

array([[0.17573659, 0.27342487],
       [0.63744027, 0.8139542 ],
       [0.30307022, 0.5041594 ]])
```

```
[36]  1 np.random.normal(10,2,5)

array([11.42377993, 11.91339579,  6.72506633, 12.04767299, 11.32460001])
```

```
[38]  1 np.random.normal(10,2,(3,2))

array([[10.77772563,  8.09826527],
       [10.45554279, 10.1527667 ],
       [12.05869924,  8.90020535]])
```

Numpy

- Basic- Slicing Arrays

  - Use square brackets ([]) to access the array
  - remember that indexing in NumPy starts at 0.

```
[51]  1 A = np.array([[1,2,3], [4,5,6], [7,8,9]])
      2 A

array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

[52]  1 A[0,0]

1

[53]  1 A[2,1]

8

[54]  1 A[0:2, 1:3]

array([[2, 3],
       [5, 6]])
```

Numpy

- Basic- Arrays are mutable

```
[43]  1 A = np.ones((3,2))
      2 A

array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
[45]  1 A[0,0] = 10
      2 A

array([[10.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])
```

Numpy

- Basic- basic operation

```
[61]  1 A = np.array([[1,2], [3,4]])
      2 A
```

```
array([[1, 2],
       [3, 4]])
```

```
[60]  1 B = np.array([[5,6], [7,8]])
      2 B
```

```
array([[5, 6],
       [7, 8]])
```

```
[59]  1 A+B
```

```
array([[ 6,  8],
       [10, 12]])
```

```
[62]  1 A-B
```

```
array([[-4, -4],
       [-4, -4]])
```

```
[63]  1 A*B
```

```
array([[ 5, 12],
       [21, 32]])
```

Numpy

- Basic- vector operation

```
[73]  1 A = np.array([1,2])
      2 A
```

```
array([1, 2])
```

```
[74]  1 B = np.array([0,1])
      2 B
```

```
array([0, 1])
```

```
[75]  1 np.inner(A,B)
```

```
2
```

```
[76]  1 np.outer(A, B)
```

```
array([[0, 1],
       [0, 2]])
```

```
[77]  1 np.dot(A,B )
```

```
2
```

Numpy

- Basic- matrix operation

```
[80]  A = np.ones((3,2))
      A

array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
[82]  B = np.ones((2,3))
      B

array([[1., 1., 1.],
       [1., 1., 1.]])
```

```
[83]  A * B

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-83-a4cedde81ed0> in <module>
----> 1 A * B

ValueError: operands could not be broadcast together with shapes (3,2) (2,3)

SEARCH STACK OVERFLOW
```

Numpy

- Basic- matrix operation

```
[80]  A = np.ones((3,2))
      A
```

```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
[82]  B = np.ones((2,3))
      B
```

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

```
[84]  np.dot(A,B)
```

```
array([[2., 2., 2.],
       [2., 2., 2.],
       [2., 2., 2.]])
```

```
[85]  A@B
```

```
array([[2., 2., 2.],
       [2., 2., 2.],
       [2., 2., 2.]])
```

Numpy

- Basic- matrix operation

```
[80]  A = np.ones((3,2))
      A

array([[1., 1.],
       [1., 1.],
       [1., 1.]])

[87]  A.T

array([[1., 1., 1.],
       [1., 1., 1.]])
```

Numpy

- Basic- matrix operation

```
[80]  1 A = np.ones((3,2))
      2 A

array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
[82]  1 B = np.ones((2,3))
      2 B

array([[1., 1., 1.],
       [1., 1., 1.]])
```

```
[89]  1 A @ B.T

---------------------------------------------------------------------------
ValueError                                Traceback (most recent c
<ipython-input-89-e36b85885131> in <module>
----> 1 A @ B.T

ValueError: matmul: Input operand 1 has a mismatch in its core dir
3 is different from 2)
```

SEARCH STACK OVERFLOW

Numpy

- Basic- operating over axes

```
[93]  A = np.arange(5)
      A
```

```
array([0, 1, 2, 3, 4])
```

```
[94]  A.sum()
```

```
10
```

```
[95]  A.min()
```

```
0
```

```
[97]  A.cumsum()
```

```
array([ 0,  1,  3,  6, 10])
```

```
[121]  np.mean(A)
```

```
2.0
```

```
[120]  np.median(A)
```

```
2.0
```

```
[122]  np.std(A)
```

```
1.4142135623730951
```

```
[123]  np.var(A)
```

```
2.0
```

Numpy

- Basic-reshape

```
[101]  1 A  = np.ones((4,4))
       2 A

array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

```
[102]  1 A.reshape((2,8))

array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.]])
```

```
[107]  1 B = np.arange(10)
       2 B

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
[109]  1 B.reshape((5,2))

array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

Numpy

- Basic-reshape

```
[125] countries = np.array(['USA', 'France', "Germany","USA","India", "France"])
```

```
[126] countries
```

```
array(['USA', 'France', 'Germany', 'USA', 'India', 'France'], dtype='<U7')
```

```
[127] np.unique(countries)
```

```
array(['France', 'Germany', 'India', 'USA'], dtype='<U7')
```

Numpy

- Basic-Input/Output of Array

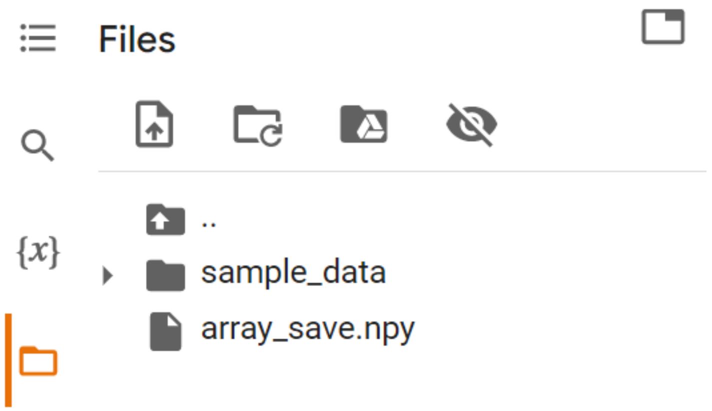
```
[130]  1 A = np.arange(1,20,5)
       2 A

array([ 1,  6, 11, 16])
```

```
[131]  1 np.save("array_save", A)
```

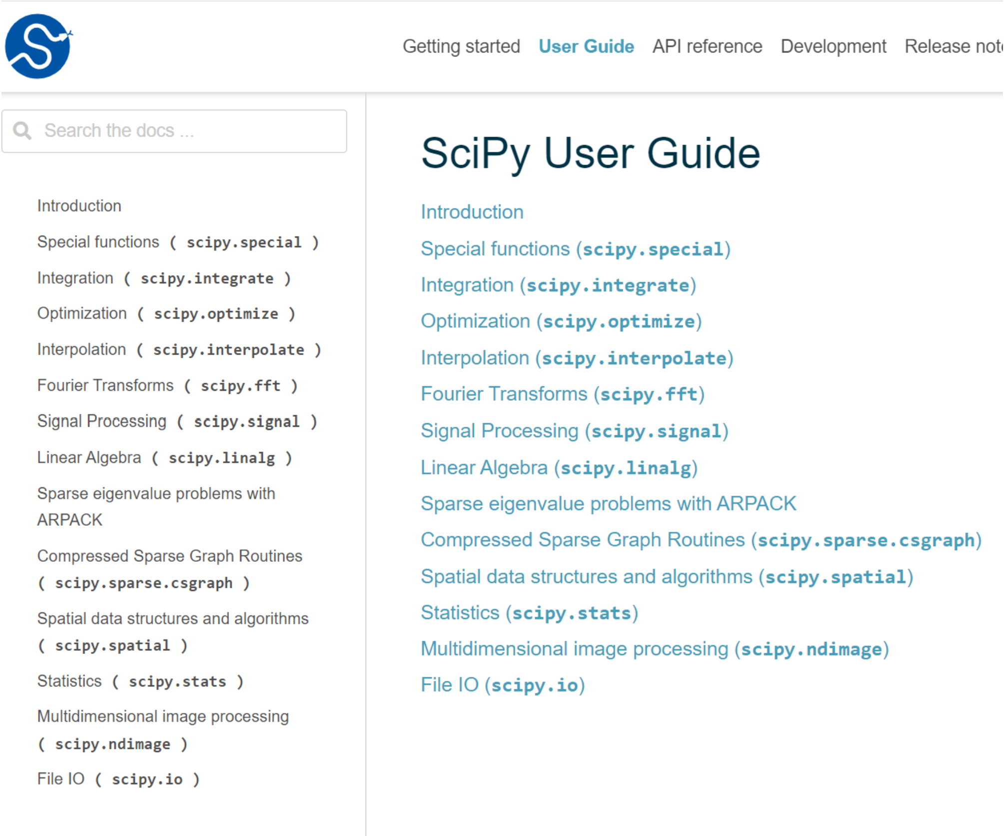
```
[133]  1 np.load('array_save.npy')
0s

array([ 1,  6, 11, 16])
```



SciPy

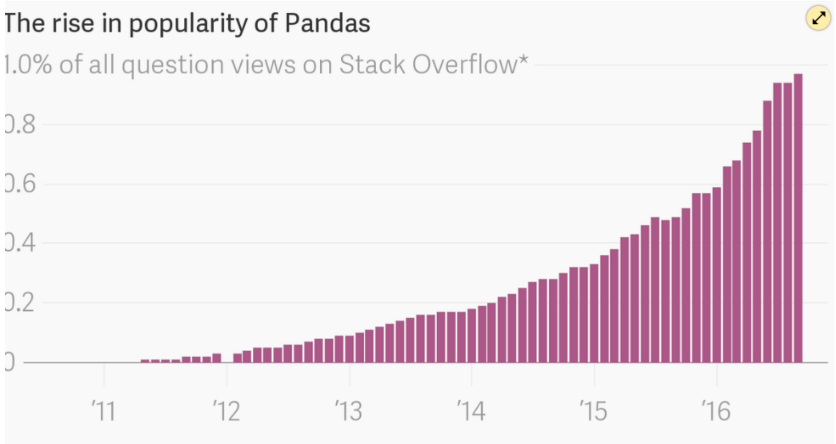
- SciPy
  - Library of algorithms and mathematical tools built to work with NumPy
  - Linear algebra- scipy.linalg
  - Statistics – scipy.stats
  - Optimization – scipy.optimize



https://docs.scipy.org/doc/scipy/tutorial/index.html

Pandas

- Pandas
  - A library used for data analysis/manipulation/visualization/cleaning/transformation
  - Can easily deal with different data format- csv/txt/excel/json…
  - Built on top of NumPy
  - Data in Pandas can be directly feed into many libraires – SciPy, Matplotlib, Scikit-learn



https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/

Pandas

- Pandas- Data structure
  - Series: one dimensional array like object containing data and labels (index)
  - DataFrame: two dimensional – spreadsheet-like- data structure containing an ordered collection of columns
    - Has both row and column labels (index)
    - Can perform arithmetic operation on both rows and columns
    - Columns can be different data types

```
[2]  1 import numpy as np
     2 import pandas as pd
```

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

```
0    1
1    2
2    3
dtype: int64
```

```
[4]  1 pd.DataFrame([[1,2,3], [4,5,6]])
```

	0	1	2
0	1	2	3
1	4	5	6

Pandas

- Pandas-DataFrame Creation

```
[2]  import numpy as np
     import pandas as pd
```

```
[5]  pd.DataFrame([[1,2,3], [4,5,6]], columns= ["A", "B", "C"])
```

	A	B	C
0	1	2	3
1	4	5	6

```
[7]  pd.DataFrame([[1,2,3], [4,5,6]],
                  index= ["index0", "index1"],
                  columns= ["A", "B", "C"])
```

	A	B	C
index0	1	2	3
index1	4	5	6

Pandas

- Pandas-DataFrame

Dictionary to DataFrame

```
[10] data = {"k1": [1,2,3],
             "k2": [4,5,6]}
```

```
[11] pd.DataFrame(data)
```

	k1	k2
0	1	4
1	2	5
2	3	6

Pandas

- Pandas-DataFrame Creation
  - Columns and rows slicing

```
[15]  A = np.arange(25).reshape(5,5)
      A
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
[18]  df = pd.DataFrame(A)
      df
```

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[26]  df.iloc[0:2,2]
```

```
0    2
1    7
Name: 2, dtype: int64
```

```
[25]  df.iloc[0:2,2:4]
```

	2	3
0	2	3
1	7	8

Pandas

- Pandas-DataFrame Creation
  - Columns selection

```
[15]  A = np.arange(25).reshape(5,5)
      A
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
[28]  df = pd.DataFrame(A)
      df.columns = ["A", "B", "C", "D","E"]
      df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[29]  df["D"]
```

```
0     3
1     8
2    13
3    18
4    23
Name: D, dtype: int64
```

```
[30]  df[["D","E"]]
```

	D	E
0	3	4
1	8	9
2	13	14
3	18	19
4	23	24

Pandas

- Pandas-DataFrame Creation
  - Conditions on columns

```
[28]  df = pd.DataFrame(A)
      df.columns = ["A", "B", "C", "D","E"]
      df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
df['D']>10
```

```
0    False
1    False
2     True
3     True
4     True
Name: D, dtype: bool
```

```
[33]  df[df['D']>10]
```

	A	B	C	D	E
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
df[(df['D']>10) & (df['D']<20)]
```

	A	B	C	D	E
2	10	11	12	13	14
3	15	16	17	18	19

Pandas

- Pandas-DataFrame Creation
  - Conditions on rows

```
[28]  df = pd.DataFrame(A)
      df.columns = ["A", "B", "C", "D","E"]
      df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[47]  df.loc[3]

A    15
B    16
C    17
D    18
E    19
Name: 3, dtype: int64
```

```
[36]  df.loc[3] < 17

A     True
B     True
C    False
D    False
E    False
Name: 3, dtype: bool
```

```
[46]  df.loc[3][df.loc[3] < 17]

A    15
B    16
Name: 3, dtype: int64
```

Pandas

- Pandas-DataFrame Creation
  - Conditions on rows

```
[28]  df = pd.DataFrame(A)
      df.columns = ["A", "B", "C", "D","E"]
      df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[51]  df[df < 12]
```

	A	B	C	D	E
0	0.0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0
2	10.0	11.0	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

Pandas

- Pandas-DataFrame Creation
  - Dropping columns

```
[28]  df = pd.DataFrame(A)
      df.columns = ["A", "B", "C", "D","E"]
      df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[86]  df.drop("A", axis = 1)
```

	B	C	D	E
0	1	2	3	4
1	6	7	8	9
2	11	12	13	14
3	16	17	18	19
4	21	22	23	24

```
[87]  df.drop(["A", "C"], axis = 1)
```

	B	D	E
0	1	3	4
1	6	8	9
2	11	13	14
3	16	18	19
4	21	23	24

Pandas

- Pandas-DataFrame Creation
  - Dropping rows

```
[28]  1 df = pd.DataFrame(A)
      2 df.columns = ["A", "B", "C", "D","E"]
      3 df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[88]  1 df.drop(0, axis = 0)
```

	A	B	C	D	E
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[90]  1 df.drop([0,3])
```

	A	B	C	D	E
1	5	6	7	8	9
2	10	11	12	13	14
4	20	21	22	23	24

Pandas

- Pandas-DataFrame Creation
  - Dropping missing values

```
[28]  1 df = pd.DataFrame(A)
      2 df.columns = ["A", "B", "C", "D","E"]
      3 df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[51]  1 df[df < 12]
```

	A	B	C	D	E
0	0.0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0
2	10.0	11.0	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

```
[91]  1 df[df < 12].dropna()
```

	A	B	C	D	E
0	0.0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0

Pandas

- Pandas-DataFrame
  - Input/output

[28]
```
df = pd.DataFrame(A)
df.columns = ["A", "B", "C", "D","E"]
df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

{x}

..
sample_data
test_df_save.csv

[54] 0s
```
df.to_csv("test_df_save.csv")
```

```
pd.read_csv("test_df_save.csv")
```

	Unnamed: 0	A	B	C	D	E
0	0	0	1	2	3	4
1	1	5	6	7	8	9
2	2	10	11	12	13	14
3	3	15	16	17	18	19
4	4	20	21	22	23	24

Pandas

- Pandas-DataFrame
  - Input/output

```
[28] df = pd.DataFrame(A)
     df.columns = ["A", "B", "C", "D","E"]
     df
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

```
[56] df.to_csv("test_df_save.csv", index = False)
```

```
[57] pd.read_csv("test_df_save.csv" )
```

	A	B	C	D	E
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

Pandas

- Pandas-DataFrame
  - Statistical summaries

```
[70]  A = np.random.normal(0,1,40).reshape((10,4))
      A
```

```
array([[ 0.43546946,  0.92326554, -0.39510305,  0.07560304],
       [-1.87615205,  0.2205231 ,  0.15980144, -1.41829336],
       [-0.57707374,  0.08911378,  0.02437065, -1.06805692],
       [-1.93323326,  1.22253341, -0.19077731, -1.16430164],
       [ 0.57012146, -1.3333652 , -0.53219669, -0.74079257],
       [-0.57295688,  1.06974951,  1.59335091, -0.70266645],
       [-0.58752005,  0.73742432, -0.8459136 , -0.55501911],
       [ 1.0681923 , -0.70303115, -0.95793017,  0.73023133],
       [ 0.91589076, -0.95263239, -0.62574103, -1.4900199 ],
       [-0.13082636,  1.25465475,  0.83511018, -0.59126495]])
```

```
[71]  df = pd.DataFrame(A)
      df
```

	0	1	2	3
0	0.435469	0.923266	-0.395103	0.075603
1	-1.876152	0.220523	0.159801	-1.418293
2	-0.577074	0.089114	0.024371	-1.068057
3	-1.933233	1.222533	-0.190777	-1.164302
4	0.570121	-1.333365	-0.532197	-0.740793
5	-0.572957	1.069750	1.593351	-0.702666
6	-0.587520	0.737424	-0.845914	-0.555019
7	1.068192	-0.703031	-0.957930	0.730231
8	0.915891	-0.952632	-0.625741	-1.490020
9	-0.130826	1.254655	0.835110	-0.591265

```
[72]  df.describe()
```

	0	1	2	3
count	10.000000	10.000000	10.000000	10.000000
mean	-0.268809	0.252824	-0.093503	-0.692458
std	1.056877	0.954889	0.794101	0.680864
min	-1.933233	-1.333365	-0.957930	-1.490020
25%	-0.584908	-0.504995	-0.602355	-1.140240
50%	-0.351892	0.478974	-0.292940	-0.721730
75%	0.536458	1.033129	0.125944	-0.564081
max	1.068192	1.254655	1.593351	0.730231

Pandas

- Pandas-DataFrame
  - Statistical properties- correlation

```
[98]  df.corr()
```

	0	1	2	3
0	1.000000	-0.372631	-0.061025	-0.260623
1	-0.372631	1.000000	0.245247	0.023555
2	-0.061025	0.245247	1.000000	-0.719317
3	-0.260623	0.023555	-0.719317	1.000000

Pandas

- Pandas-DataFrame
  - Statistical properties- covariance matrix

```
[99]  df.cov()
```

	0	1	2	3
0	0.763601	-0.302801	-0.043131	-0.258705
1	-0.302801	0.864750	0.184458	0.024883
2	-0.043131	0.184458	0.654179	-0.660887
3	-0.258705	0.024883	-0.660887	1.290382

Pandas

- Pandas-DataFrame
  - Basic Plotting

```
[80]  1 df.iloc[:, 2]
```

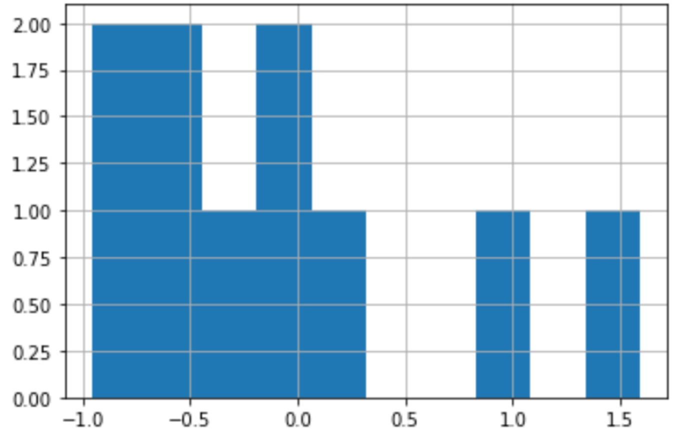
```
0   -0.395103
1    0.159801
2    0.024371
3   -0.190777
4   -0.532197
5    1.593351
6   -0.845914
7   -0.957930
8   -0.625741
9    0.835110
Name: 2, dtype: float64
```

```
[70]  1 A = np.random.normal(0,1,40).reshape((10,4))
      2 A
```

```
array([[ 0.43546946,  0.92326554, -0.39510305,  0.07560304],
       [-1.87615205,  0.2205231 ,  0.15980144, -1.41829336],
       [-0.57707374,  0.08911378,  0.02437065, -1.06805692],
       [-1.93323326,  1.22253341, -0.19077731, -1.16430164],
       [ 0.57012146, -1.3333652 , -0.53219669, -0.74079257],
       [-0.57295688,  1.06974951,  1.59335091, -0.70266645],
       [-0.58752005,  0.73742432, -0.8459136 , -0.55501911],
       [ 1.0681923 , -0.70303115, -0.95793017,  0.73023133],
       [ 0.91589076, -0.95263239, -0.62574103, -1.4900199 ],
       [-0.13082636,  1.25465475,  0.83511018, -0.59126495]])
```

```
[79]  1 df.iloc[:, 2].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0dee7b9d0>
```



Pandas

- Pandas-DataFrame
  - Basic Plotting

```
[80]  1 df.iloc[:, 2]
```

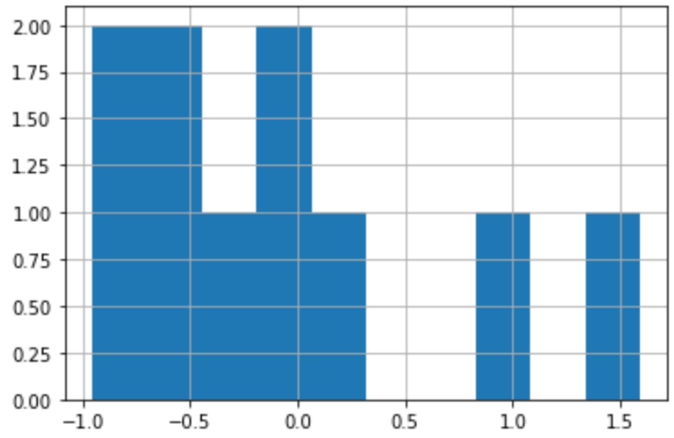
```
0   -0.395103
1    0.159801
2    0.024371
3   -0.190777
4   -0.532197
5    1.593351
6   -0.845914
7   -0.957930
8   -0.625741
9    0.835110
Name: 2, dtype: float64
```

```
[70]  1 A = np.random.normal(0,1,40).reshape((10,4))
      2 A
```

```
array([[ 0.43546946,  0.92326554, -0.39510305,  0.07560304],
       [-1.87615205,  0.2205231 ,  0.15980144, -1.41829336],
       [-0.57707374,  0.08911378,  0.02437065, -1.06805692],
       [-1.93323326,  1.22253341, -0.19077731, -1.16430164],
       [ 0.57012146, -1.3333652 , -0.53219669, -0.74079257],
       [-0.57295688,  1.06974951,  1.59335091, -0.70266645],
       [-0.58752005,  0.73742432, -0.8459136 , -0.55501911],
       [ 1.0681923 , -0.70303115, -0.95793017,  0.73023133],
       [ 0.91589076, -0.95263239, -0.62574103, -1.4900199 ],
       [-0.13082636,  1.25465475,  0.83511018, -0.59126495]])
```

```
[79]  1 df.iloc[:, 2].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0dee7b9d0>
```



Scikit-learn

scikit learn | Install | User Guide | API | Examples | Community | More

scikit-learn

Machine Learning in Python

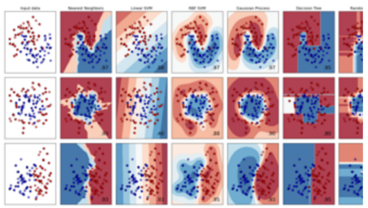
Getting Started | Release Highlights for 1.1 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

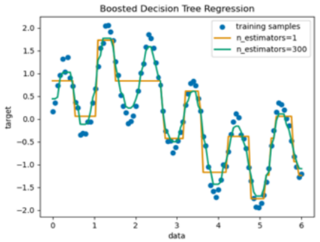


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

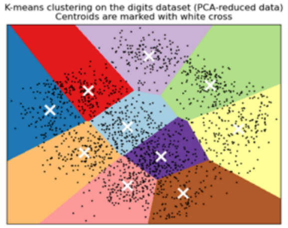


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

https://scikit-learn.org/stable/

Scikit-learn

scikit learn Install User Guide API Examples Community More

Prev Up Next

scikit-learn 1.1.2
Other versions

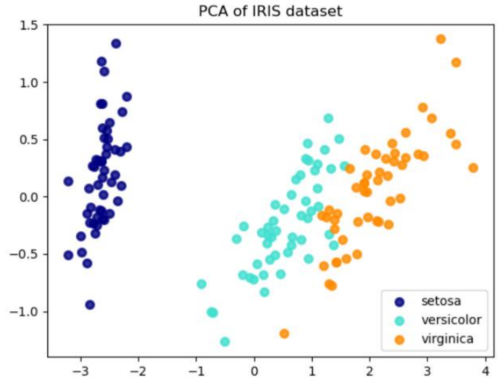
Please **cite us** if you use the software.

2.5. Decomposing signals in components (matrix

2.5. Decomposing signals in components (matrix factorization problems)

2.5.1. Principal component analysis (PCA)

2.5.1.1. Exact PCA and probabilistic interpretation



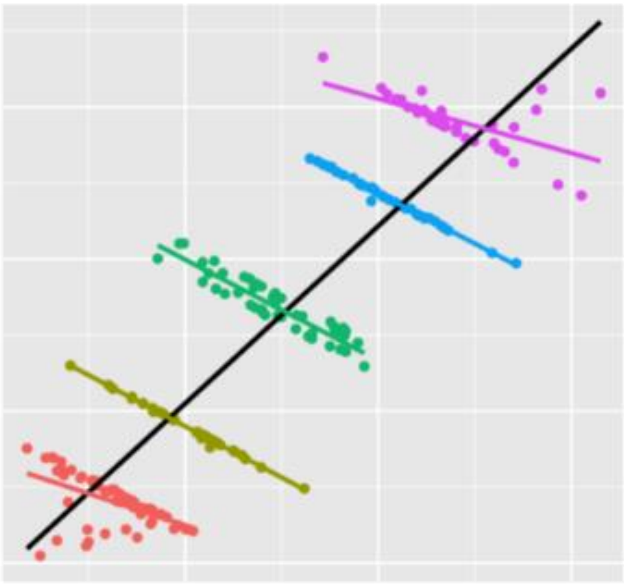

```
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()
```

https://scikit-learn.org/stable/

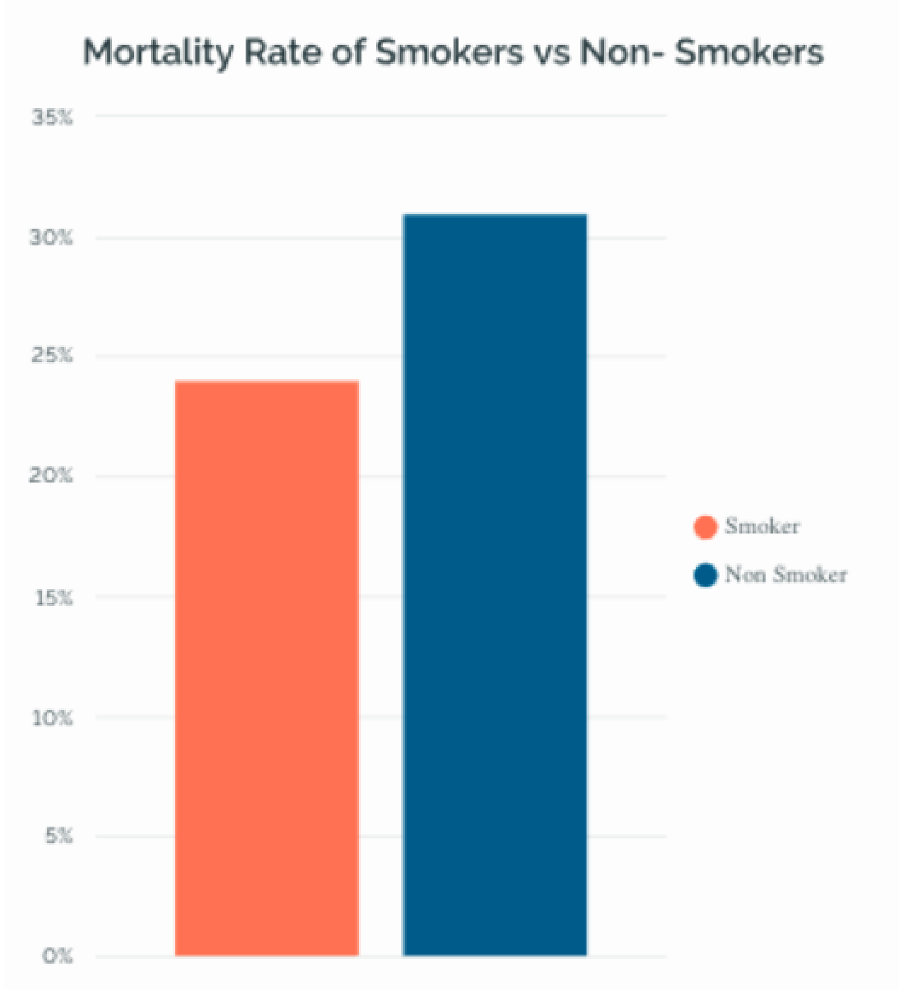
Simpson’s Paradox



https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/

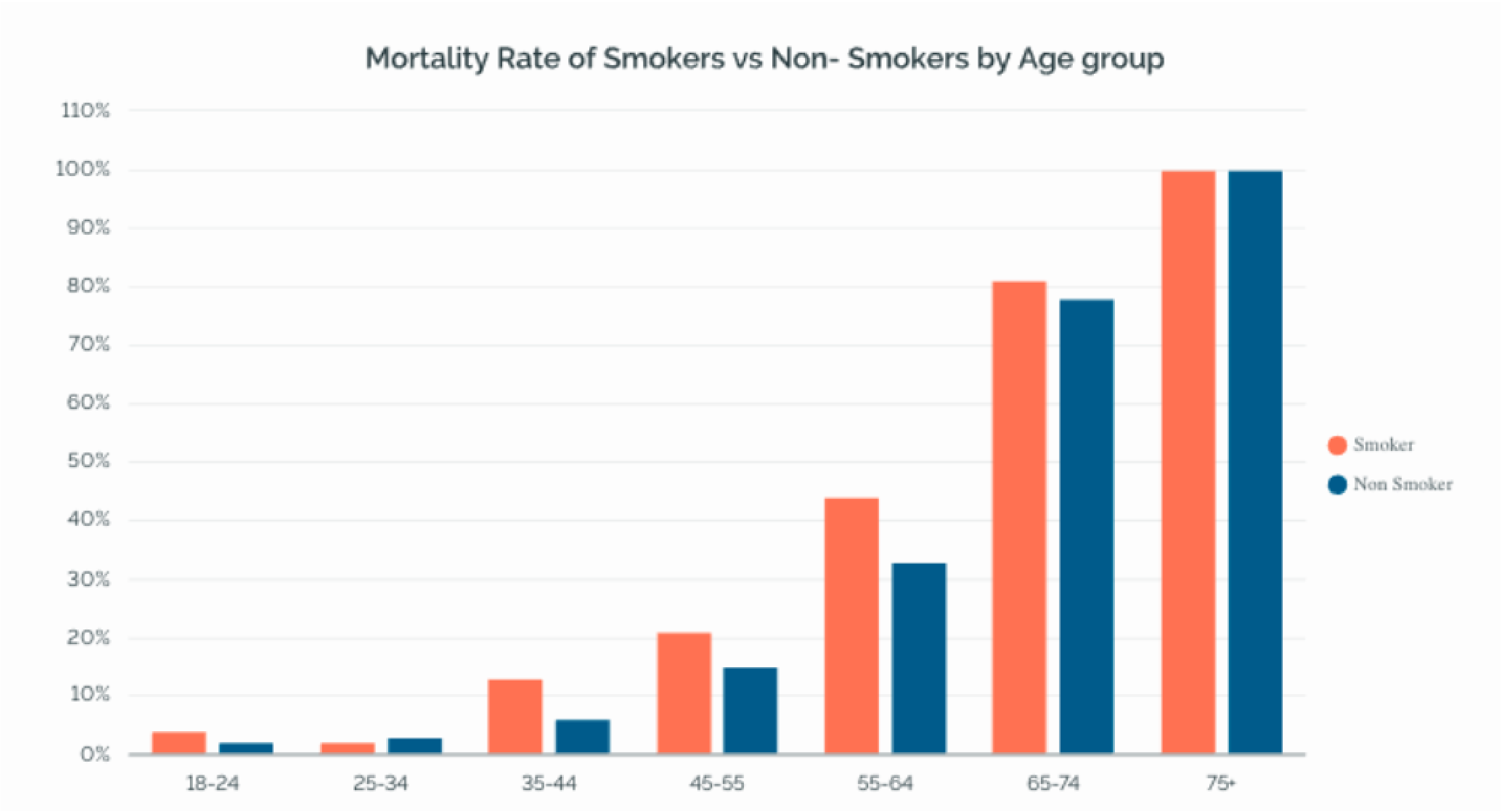
An Interesting Experiment

- Conducted in 1996 - to study the effect of smoking on a sample of population
    - Over twenty years and included 1314 English women
    - The study showed that **Smokers tend to live longer than** non-smokers
    - Smokers had a mortality rate of 23%, while for non-smokers, it was around 31%



An Interesting Experiment

- On breaking the same data by age group, we get an entirely different picture
  - in most age groups, smokers have a high mortality rate compared to non-smokers



Results of the study broken down by age group | Image by Author

What is Simpson's Paradox

- This phenomenon that we just saw above is a classic case of Simpson's paradox
- Simpson's Paradox
  - a trend appears in several different groups of data but disappears or reverses when these groups are combined
  - the same dataset can appear to show opposite trends depending on how it's grouped
- When grouped age-wise, the data shows that non-smokers tend to live longer, but for overall picture, smokers tend to live longer

What is Simpson's Paradox

- What is exactly happening here?
- Why are there different interpretations of the same data, and what is evading our eye in the first case?
- This happens when data from the two variables are tested in a group that has a **confounding variable**
- **Confounding (or lurking) variable** — a conditional variable that can affect our conclusions about the relationship between two variables — smoking and mortality in our case.

Does eating ice-cream leading lead to sunburn?

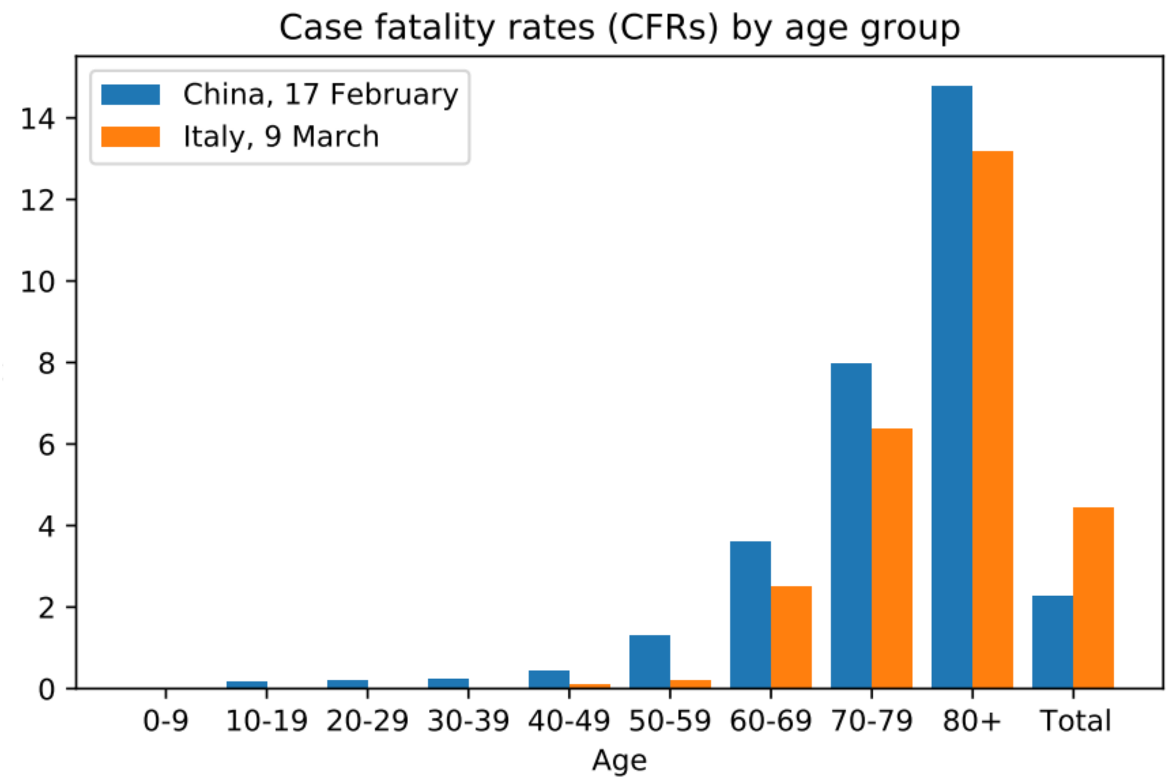
https://growthlab.center/essays/confounding-variables

What is Simpson's Paradox

- This means that during analysis, a confusing variable was present, the variable alters the facts of the data, and because the variable was not supposed to be present in the data, the researcher did not consider it before conducting the test
- When this happens, it can lead the researcher to conclude falsely, and the result of the test will be inaccurate.

Another Recent Phenomenon

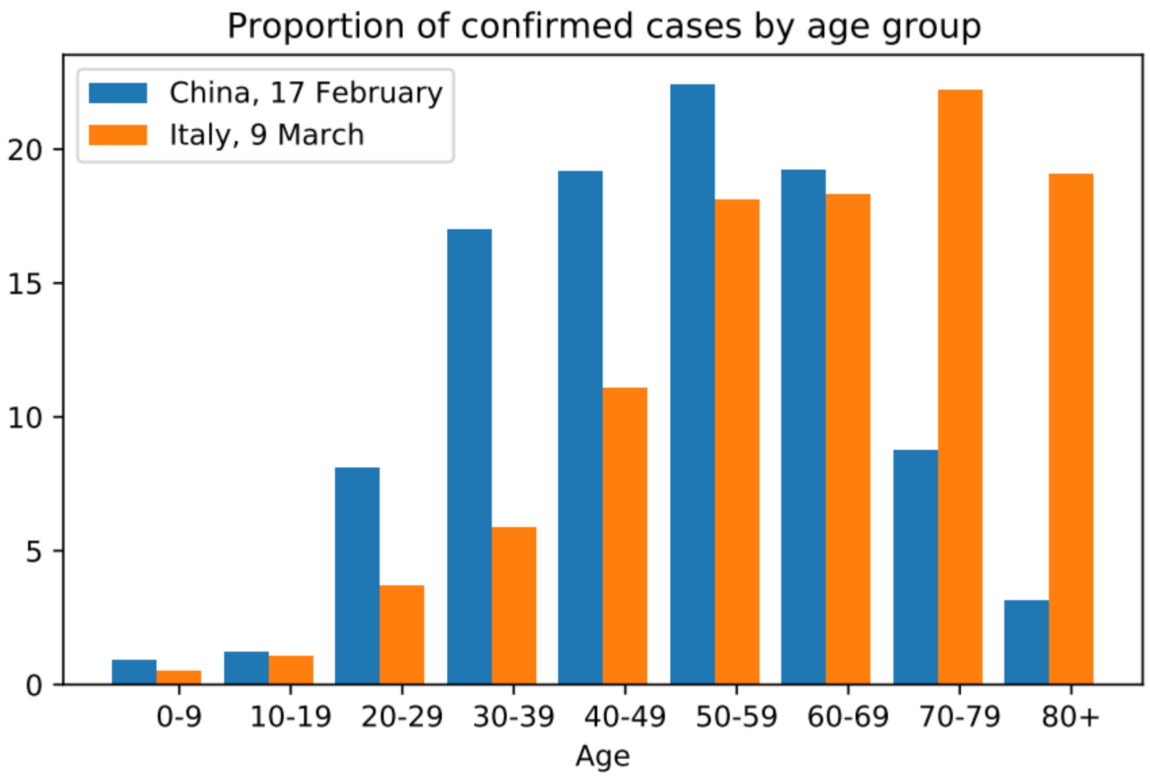
- Comparative analysis of **Case Fatality Rate** of COVID-19 confirmed cases between China & Italy
  - Case Fatality Rate (CFR) indicates the proportion of confirmed cases which end fatally



von Kügelgen, J., Gresele, L., & Schölkopf, B. (2021). Simpson's paradox in Covid-19 case fatality rates: a mediation analysis of age-related causal effects. *IEEE Transactions on Artificial Intelligence*, *2*(1), 18-27.

Another Recent Phenomenon

- In Italy - higher proportion of older patients
- Key feature of COVID-19: younger patients have higher chance for surviving

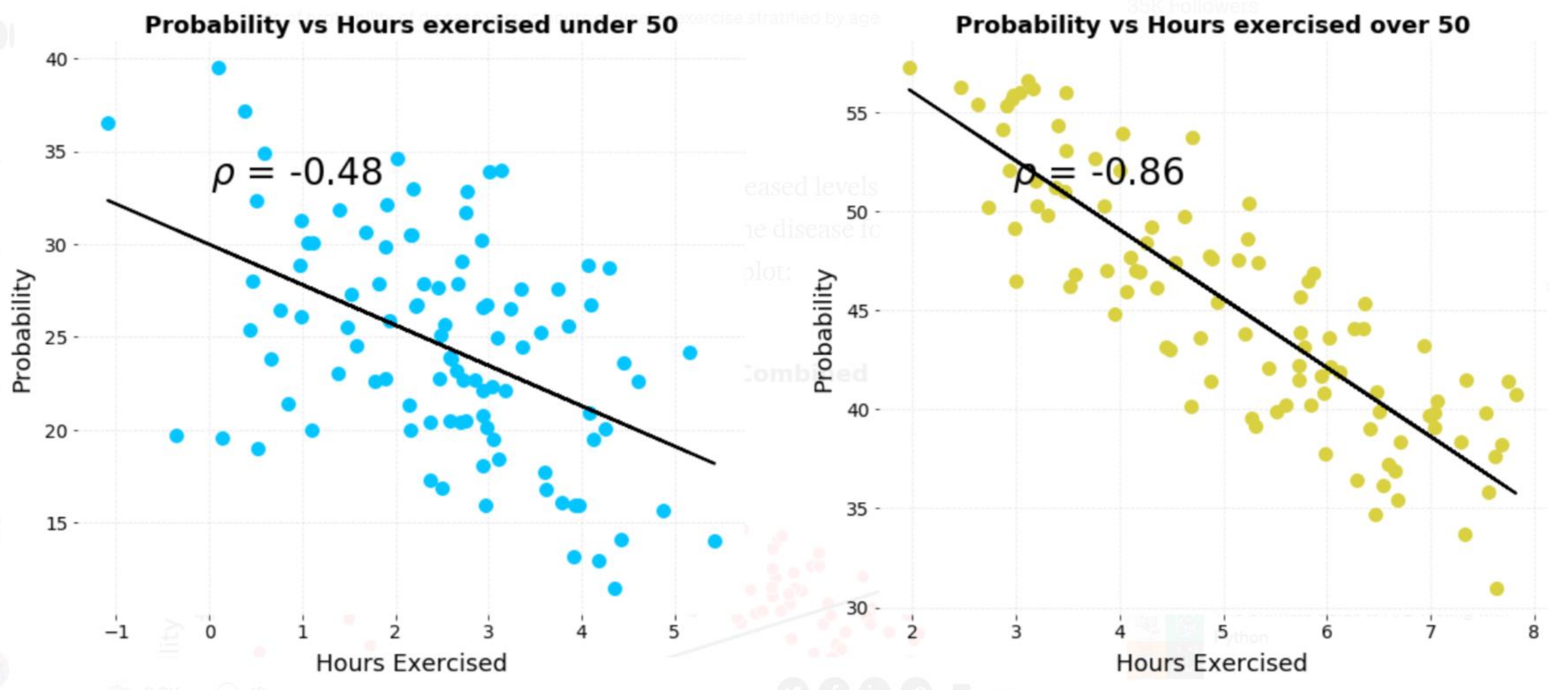


von Kügelgen, J., Gresele, L., & Schölkopf, B. (2021). Simpson's paradox in Covid-19 case fatality rates: a mediation analysis of age-related causal effects. *IEEE Transactions on Artificial Intelligence*, *2*(1), 18-27.

SJSU SAN JOSÉ STATE UNIVERSITY

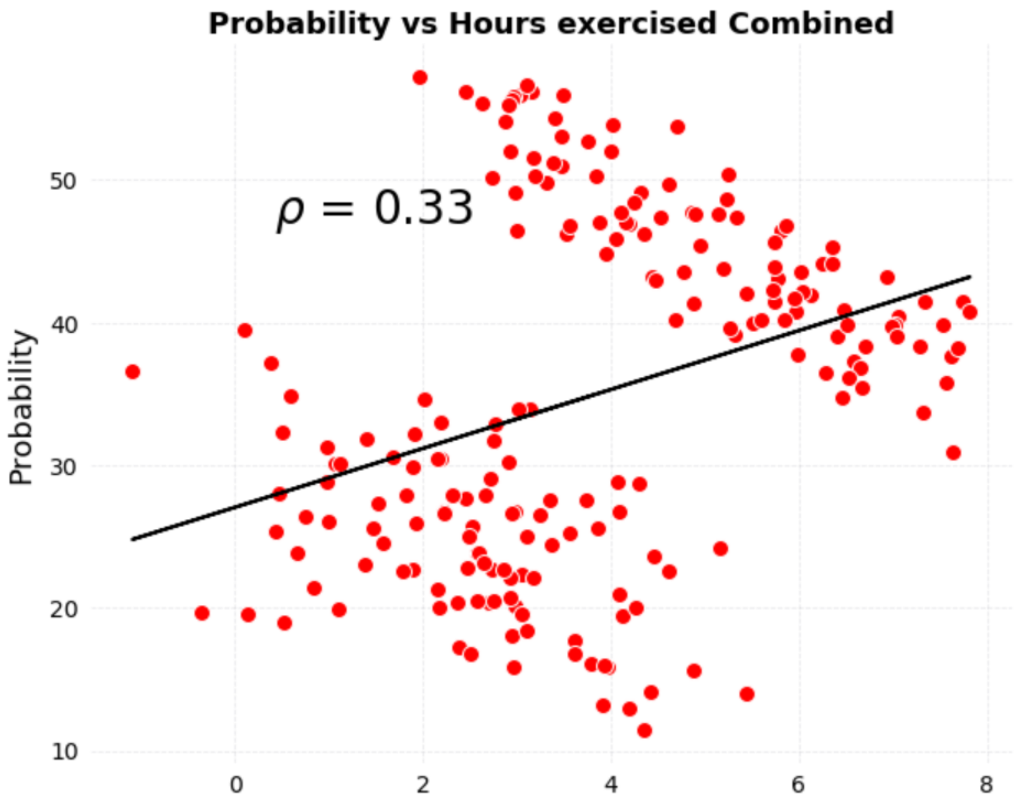
Correlation Reversal

- Another intriguing version of Simpson's Paradox - Correlation flips upon aggregation
  - Exercise hours vs. disease risk for <50 and >50 age groups
- Initial correlations differ within groups while combined data shows opposite correlation



Correlation Reversal

- The correlation has completely **reversed**!
- Combine figure - exercise increases the risk of disease
- How can exercise both decrease and increase the risk of disease?
- The answer is that it doesn't and to figure out how to resolve the paradox, we need to look beyond the data we are shown and reason through the data generation process — what caused the results



Why Simpson’s Paradox Matters?

- Simpson’s Paradox is important because it reminds us that **the data we are shown is not all the data there is**
- We can’t be satisfied only with the numbers or a figure, we must consider the data generation process — the causal model — responsible for the data.
- Once we understand the mechanism producing the data, we can look for other factors influencing a result that are not on the plot
- Thinking causally is not a skill most data scientists are taught, but it’s critical to prevent us from drawing faulty conclusions from numbers
- We can use our experience and domain knowledge — or those of experts in the field — in addition to data to make better decisions.

Why Simpson’s Paradox Matters?

- Moreover, while our intuitions usually serve us well. They can fail in cases where not all the information is immediately available
- We tend to fixate on what’s in front of us — all we see is all there is — instead of digging deeper and using our rational, slow mode of thinking
- Particularly when someone has a product to sell or an agenda to implement
- we should be extremely skeptical of the numbers by themselves
- Data is a powerful weapon, but it can be used by both those who want to help us and nefarious actors

What can we do?

- Without enough domain knowledge, it's hard to know which view of the relationship between two variables makes more sense
- But before we think about how to deal with Simpson's Paradox, we need to find a way to efficiently detect it in a dataset
- It is possible to find an instance of Simpson's Paradox (a "Simpson's Pair") simply by disaggregating a contingency table or a plot of data points and studying the results
- However, there are other ways we can find Simpson's Pairs using models, e.g.:
  - By building decision trees and comparing the distributions, or
  - By building regression models and comparing the signs of the coefficients

What can we do?

- There are benefits to both, however, this can get difficult very quickly, especially when working with big datasets
- It's hard to know which variables in the dataset may reverse the relationship between two other variables
- It can be hard to check all possible pairs of variables manually
  - Imagine we have a dataset with only 20 variables: we'd need to check almost 400 pairs to be sure to find all cases of Simpson's Paradox

What can we do?

- Unequal distribution of data into groups and undetected confounding variables may lead to Simson's paradox
- In **Experimental** Studies:
  - It can be avoided with a correct setting of experimental design
  - In the planning stage- confounding variables should be included in the analysis to correctly answer the research question
  - To ensure the balanced groups in experimental design:
    - Sample Randomization
    - Randomized Block Design
    - Minimization

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2880329/

Conclusion

- Simpson's paradox is no doubt tricky, but a researcher equipped with the right tools and sound knowledge can manage it well.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2880329/

A case where model implementation went wrong

- An evaluation of 650 Zillow-owned homes revealed that two-thirds of them were for sale for less than the company paid for them, resulting in an average loss of more than $80,000 per house.

GeekWire NEWS JOBS EVENTS RESOURCES ABOUT Search

Trending: OceanGate's explorers update their view of a tattered Titanic and the life around it

Why the iBuying algorithms failed Zillow, and what it says about the business world's love affair with AI

BY JOHN COOK on November 3, 2021 at 10:34 am

Home » Topics » AI Deep Learning » The $500mm+ Debacle at Zillow Offers – What Went Wrong with the AI Models?

The $500mm+ Debacle at Zillow Offers – What Went Wrong with the AI Models?

December 13, 2021 by Editorial Team Leave a Comment

tweet share share share email

Zillow, an online real estate marketplace, recently shuttered its Zillow Offers business

https://medium.com/@michael.proksch/learning-from-zillows-ai-inflicted-loss-of-10-billion-79d9b1625ac7
https://insidebigdata.com/2021/12/13/the-500mm-debacle-at-zillow-offers-what-went-wrong-with-the-ai-models/
, https://www.geekwire.com/2021/ibuying-algorithms-failed-zillow-says-business-worlds-love-afir-ai/

A case where model implementation went wrong

- An evaluation of 650 Zillow-owned homes revealed that two-thirds of them were for sale for less than the company paid for them, resulting in an average loss of more than $80,000 per house.

> “All the AI and machine learning in the world isn’t yet up to the task of the complexity of valuing a home in a rapidly changing market, and this move by Zillow is proof.”

GeekWire NEWS JOBS EVENTS RESOURCES ABOUT

Trending: OceanGate's explorers update their view of a tattered Titanic and the life around it

Why the iBuying algorithms failed Zillow, and what it says about the business world’s love affair with AI

BY JOHN COOK on November 3, 2021 at 10:34 am

Home » Topics » AI Deep Learning » The $500mm+ Debacle at Zillow Offers – What Went Wrong with the AI Models?

The $500mm+ Debacle at Zillow Offers – What Went Wrong with the AI Models?

December 13, 2021 by Editorial Team Leave a Comment

tweet share share share email

Zillow, an online real estate marketplace, recently shuttered its Zillow Offers business

https://medium.com/@michael.proksch/learning-from-zillows-ai-inflicted-loss-of-10-billion-79d9b1625ac7
https://insidebigdata.com/2021/12/13/the-500mm-debacle-at-zillow-offers-what-went-wrong-with-the-ai-models/
, https://www.geekwire.com/2021/ibuying-algorithms-failed-zillow-says-business-worlds-love-afir-ai/