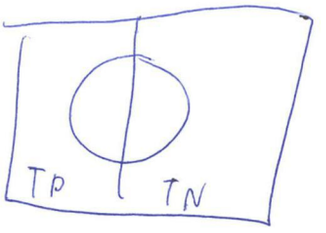
NP. reshape (rows, columns)

NP. shape
??? (rows, columns)

NP. size
??? (size)

NP. std
??? standard dev.



TP / (TP - FP)

pd. df. plot (kind = '...') : bar, barh, hist, box, density, area, scatter, pie

pd. sort_values (ascending = False)

pd. query ('column > 5')
x = 13 'column == @x'

c = Counter (text. split)

pd [index == 'James']

pd. groupby (by = 'column', axis = 0)
example:
df. groupby (['animal']). mean() # takes mean of each animal

find highest number of 1 kind
df. groupby (0)[1]. count(). sort_values(). tail (1) # of .orgnarc()

pd. merge()
pd. join

% time it # returns time he/she took on average
% time # ~~qq~~ quicker
%% time # for whole cell

NP. ~~rod~~ random. randint (0, 10)

%
% ls -lh
% rm # remove
% prun
x?

Data science is : acc = TP + TN + All

precision = TP/TP+FP   F1 = 2 * rec x prec / rec + pres

recall = TP/TP+Fn

Based on patterns in data predicting a value on new unseen instances of data.

Classification + predicting a class

Regression + predicting a value

Different from traditional statistics ~~that~~ in which you try to explain something.

: Interacting with the outside world

2 preparation 3 transformation

4 modeling and computing 5. presentation

Informatie verwerking is:

- Omzetten ene formaat naar andere
- Uitvogelen wat er bedoelt is

impliciete informatie expliciet maken = leesbaar computer

Onderzoek cyclus:

1. Kom met idee/hyp/vraag 2. verzamel data
3. Schoon data op 4. herstructureer data 5. analyse
6. Rapporteer.

waarom geen excel?

1. Schaalt niet
2. Beperkte functionaliteit
3. integratie met andere tools

Serie: one-dimensional labeled arry capable of holding any datatype.

Data frame: two-dimensional labeled data structure with columns of potentially different types.

pd. "TAB"

pd.read_csv?

*?

help(collections)

dir (scipy) x stats ~~...~~

Square x**2

%magic %run

%lsmagic

%past

%cpaste

** power x4

% modulo

// floor

L + np.arange(0, L.shape

transpose = .T

L[1:i, 1:i]

(Grid != 3).sum() - 10

pd.crosstab(year, party) index.values

Kur[~Kur.party.str.contains('vragen')]

Kur.loc[Kur.party == 'pvda', 'ministerie'].val...

index max = idx max   iloc[[0,1]]

new column = Kur['text'] = ...

Pivot[['F','M']].min(axis=1).sum()

Pivot[pivot.ratio < 5]['F'].sum

x1[3:]   x2[:2,1:]   x2.shape, x2.size

x2.reshape(12,1)   np.square(x1 - x1.mean())

np.sqrt(((x1 - x1.mean())**2).mean()) == x1.std()

x1[x1%3 == 0]   x2[[0,1,2],[0,1,2]]

%timeit sum([x**2 for x in big_array])

%timeit np.sum(np.square(big_array))

iris.species.value_counts().plot(Kind = barh)

X, y = iris.drop(columns = ['species']), iris.species

iris.sort_values(['sepal_length', 'sepal_width'], ascending [true, false])

iris.groupby('species')['sepal_length'].max()

iris[(iris.sepal_length > 5) & (iris.sepal_width < np.sqrt(5))]

CS = pd.series(data = Counter(text.split())).sort_values(ascending = True)

CS[CS * CS.index.str.len() == 24].index

CS[CS == 1].sum() / CS.count()   (sum)

CS.value_counts().sort_index().plot(Kind = "bar")

pd.groupby(0)[1].count().sort_values().tail(1)

%ls -lh = dir & size

%rm = remove

time / timeit / prun

x?   %cat = new file

%pwd = print work dir

%cd = change direct

%mkdir = make new dir

%mv = move file ../ text.t ./

cp, env, man, more, rmdir (copie)

Theory

precision = positive predicted value. measures howmany of the samples predicted positive actually are positive. TP / (TP + FP)

recall = howmany of the positive samples are captured by the positive predictions. TP / (TP + FN)

ondenzoeksyclus: idee/hypothese/vraag → data verzamelen / opschonen → hessstructeer / analyseer → Pandas

Data science: based on patterns in data, a prediction a value on new unseen instances of data.
steps: interacting with outside world → preparation → transformation → modeling & computing → presentation.

nadeel excel → schaalt niet, beperkte functionaliteit, integratie andere tools.

Pandas.

argmax
idxmin() /max

→ [x for x in df.iterrows]

Count and plot:

df[column].value_counts().plot.bar()

max value for species:

df.groupby('species')['sepal_length'].max()

Show top 10 sorted

df.sont_values(ascending=False).head(10) of [:10]

lower all in column:

df.column.str.lower()

replaces all non words with "":

df.column.neplace(to_neplace="\W", value="", regex=True)

Can also be value np.nan!

Series.value_counts()
df.sort_index()

get length of index words:

Series.index.str.len()

pd.drop(axis=1)

- .shape
- .info
- .count / .sum / .mean

iloc[[0],[0]] → pos
ix[2] → row
ix[1, "col"]

Numpy

arange
reshape
sort — with axis
zeros
add / subtract / multiply / exp / sqrt / mod
vstack → np.vstack([x, grid])
hstack
any
all

min/max with axis to slice for each row/col
ax=0 is col
concat([grid, grid], axis=1)

fancy index:

array([1,2,3], [1,2,3]) → rij 1 col 1, rij 2 col 2 ...

slicing: a[0:4] → index 0,1,2,3
a[0:4, 3] → row 0,1,2,3 at col 3
a[:2] → index 0,1.
a[:, 1] → col 1 all rows

x[::2]
x[1::2]
x[::-1]
x[5::-2]

np.arange(12).reshape((3,4))
row = np.array[0,1,2]
col = np.array[1,2,3]

x[:2, :3] → 2 rows 3 cols
x[:3, ::2] → 3 rows every other col
x[:, 0] → first col
x[0, :] → first row

x[row, col]

table multiply:
x = np.arange(1,6)
np.multiply.outer(x, x)

reverse array:
x[::-1, ::-1]

x = soup.findAll('li', class_='class')
first comment = x[0]
print(firstcomment.prettify())

np.count_zero(x < 6)
np.sum(x < 6, axis=1)

pd.crosstab(df.col1, df.col2, margins=True)

pd.pivot_table(df, index=, cols=, values=count, aggfunc="[illegible]" np.sum, margins=True)

df[df.col.isin(list of options)]

Counter([df[col] for s in records if 'TZ' in s]).most_common

s[~(s > 1)]
s[(s < -1) | (s > 2)]
set: s['a'] = 6

. - Any Character
\d - (0-9)
\D - Not \d
\w - (a-z, A-Z, 0-9, _)
\W - Not \w
\s - (space tab, newline)
\S - Not \s \t - tab
\b - white space around words
\B - Not \b
^ - Beginning of str
$ - end of str
[^] - Matching characters NOT in []
[abc] - a, b or c. (range)
+ - Match 1 or more
? - Match 0 or ~~more~~ 1
* - Match 0 or more
| - either, or
{x} - expecting "x" amount

p = re.compile('\d+')
p.findall('string here')
re.split('[\W]+', 'string')

sns.distplot(x) #
np.log([1, 5, 8])

DataFrame.plot(kind='bar')

Pandas: kvr[~kvr.Partij.str.contains()], inplace=True
kvr.dropna(subset=['Partij']) *removes all rows with NaN values.
pd.crosstab(kvr.Jaar, kvr.Partij)[[top10value_counts(Partij).index.values]]
pd.read_csv('file', header=None, sep='\t', names=['', ''], skipinitialspace=True,
DataFrame.idxmax(axis=0, skipna=True) #for each column

df.columns
df.index

Numpy:
np.random.randint(low, size=size)
np.arange(start, stop, step) #output is array
L.reshape(int(), int()) #1D → 2D
np.mean(L, axis=0) #gemiddelde, axis=0 is hoeveel'decimal'
L.sum(axis=0) #0=vertical axis, 1=horizontal.

L[:, :]
column, row

df.describe()
~~df.sort~~

df.size #amount of elements
df.shape #output (rows, columns)

df.sort_values('columnname', ascending=False)

df['columnName'] or df.columnName

slice columns: df[['', '']]

kvr['Partij'].str.replace('kloord', 'iets')

deelbaar = L[L%n == 0]

Steps:
1. Interacting with the outside world
2. Preparation
3. Transformation
4. Modeling & computation
→ Connecting your data to statistical models, machine learning algorithms or other computational tools.
5. Presentation.

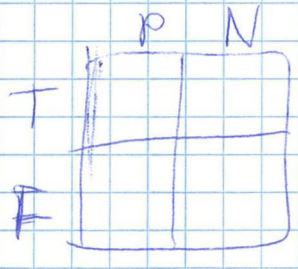
- Data science is the study of the generalizable extraction of knowledge from data.
- A ~~common~~ epistemic requirement is predictive power.
- A data scientist requires skill set: Mathematics, machine learning, AI, statistics, databases, optimization and a deep understanding of the craft of problem formulation to engineer effective solutions.

* Evaluate models: Classification & regression

- Precision = how many of the predictions of class C were correct. (Dus de hoeveelheid false beschuldigingen)
Recall = how many instances of class C predicted to be C.
↳ TP/(TP+FN) = how many of the "true class" (those with disease) are picked by the test.
↳ TP/(TP+FP) = how many of those with a positive test do have the disease
Accuracy = how many of all predictions were correct.

- Pivot table is typisch een 3D Matrix
- Gebruik bij categoriale eigenschappen.
- Groepeer rijen op een eigenschap (bv. geslacht)
- Kies 1 kolom, maak groepjes voor elke waarden. bv. gelijding
- kies een kolom en vatdie samen.

Waarom niet blijven Excelen?
1. Schaalt niet
2. Beperkte functionaliteit (niet makkelijk schoonmaken)
3. Integratie met andere tools. In python wel.



bs4 imp. BeautifulSoup local = '../data/name'
imp re doc = open(local).read()

1 url = 'https:'
2 html_doc = requests.get(url)
3 soup = BeautifulSoup(html_doc.content)
# get li elements
Li = soup.FindAll('li', class_ = re.compile('class 2'))
Li.attrs
Dict = {i:a.attrs['rang']: i.a.attrs['href'] for i in Li}

Pandas
df.loc[[0],[0]]
→ select by row/column
df.loc
→ select by row/column name
df.ix [2] = new
Li['capital'] = column
df.drop
sort_index()
sort_values(by='country')
Sheet[rows,columns]
columns
count = # non NA values
sum
min/max
idxmin/idxmax = index/pos
describe = summary
mean/median

append/insert/delete

web scraping comment Harvest.
1 → 2 → 3 :
comments = soup.FindAll('li', class_='comment')
com_list = [(soup.title.text, c.attrs['data-id'],
c.find('span', class_='user-name').text, c.p.text)
for c in comments if c.p]
comment DF = pd.DataFrame(com_list)
comment DF.columns = ['article', 'comment id', 'reagender', 'rea']

Numpy as np
np.array([1,2,3])
zeros,ones
np.arange(0,15,5)
a = np.array([1,2,3])
a.shape #dimension
ndim #dim
size #elem
type = convert an array to type
astype
len(a)

np.subtract
add
divide
multiply
exp
sqrt

if both arrays:
a == b : [True, False, False]
[False, True, True]

a.sum
min
max (axis=0)
cumsum (axis) = 0
mean/median
convolve
sort
a[::-1] = reversed array
transpose

np.random.randint

Pd.DF.dropna
~~L> missing values replaced by None~~
L> drop ... where at least one element is missing
(axis)
Pd.a.fillna(0)
L> replaces NA with Sub.

DF3 = pd.merge(df1, df2)

0.9 x 0.9
9A x 0.9 + 1 x 0.9

```
import re | os | numpy as np | matplotlib.pyplot as plt | seaborn as sn
%.matplotlib inline

from nosetools import assert_equal, assert_almost_equal
from pandas.testing import assert_frame_equal

col_Names = ["Jaar", "Partij", "Titel...."]
kvr = pd.read_csv( linktofile(), sep = '\t', header = None, skipinitialspace = True,
                   names = col_Names )

1. cito = pd.read_excel ( 'http...xls', index_col = 'naam')

2. print (cito.shape, cito.columns)        2. laatste 5 regels
   cito.tail()

3. dubbele index?
   (len(set(cito.index)) == len (cito.index)

4. dubbele: welke zijn dubbel index?
   dubbel = cito.index.value_counts()
   dubbel[dubbel > 1]

5. orden:
   cito.sort_index

6. mooi schoon
   cito.index.str.lower().str.replace?
   cito.index.str.lower().str.strip().str.replace('...', '').str.replace('..', ').

7. bepaal cito do scholen zonder 'school' in naam        sort_values()'
   cito[~cito.index.str.lower().str.contains('school')].head(10)

8. add column 'RMSE' tussen kolom 'quasicito' en 'verwacht'
                    (kwadraat van verschil tussen)
   cito['RMSE'] = np.sqrt((cito.quasicito - cito.verwacht)**2)

9. elke waarde, gemiddeld verwachte kolom aftrekken, op die manier
   alle waardes normaliseren
   M = cito.mean()
   (cito[M.index] - M).tail()

10. plot inkomens per school
    cito.inkomen.sort_values().plot();

    cito[cito.inkomen>50000].gemeente.
    value_counts().sort_values().plot(
    kind='barh');

11. welke gemeenten inkomen > 50.000, hoeveel per gemeente, horizontale barplot.
```

numpy

create array linear sequence
np.arange(0,20,2)
[0,2,4,6... 18]

---

geef 1 dimensionale array van eerste n getallen getallen die deelbaar zijn door 0.
return (np.arange(n))* 0

---

accepteert 1 dimensionaal array met lengte van $\sqrt{n}$ voor een geheel getal n, verandert in vierkant numpy array
return (L.reshape((int(np.sqrt((len(L)), int(np.sqrt(len(L)))

L.shape = array dimensions
L.size = number of elements in array

---

maak array met shape n,1
return L.reshape(int(L.size), 1)

---

laatste kolom in array
return np.array([column[-1] for column in ~~colum~~ L])

---

geef 1D array met alle getallen deelbaar door n
return L[L % n == 0] ← boolean mask

RMSE = np.sqrt(x-y)**2

---

gemiddelde per kolom
return np.mean(L, axis = 0)

---

add_per_column
voor elk getal + hoeveelheid van index
return [x + [0] for x in enumarate(L)] + L

[10,15], [20,30] → [10,15] [21,31]
return add_per_column(L.transpose()).transpose
((1,0))

shift tab bij

---

change each element in x1 by its squated difference from the mean of x1
x1 - x1.mean())**2 = variance

---

np.sqrt((variance).mean()) == x1.std()

---

np.diagonal() = diagonaal

Pandas

1) Lees de file in een df en maak de naam van de school de index. Df = cito
cito = pd.read_excel('....xls', index_col = 'naam')

2) geef de dimensies en de namen van kolommen
print(cito.shape, cito.columns) | cito.tail()/head

3 Zitten er dubbelen in de index? Laat ze zien
len(set(cito.index)) == len(cito.index)

4 Laat zien welke namen dubbel voorkomen
dubbel = cito.index.value_counts()
dubbel[dubbel > 1]

5 Orden cito op index
cito.sort_index()

6 verwijder spaties aan begin en eind, tekens als " en andere rommel van de namen van index.
cito.index.str.lower().str.strip().str.replace('"',"").str.replace(' ',"").sort_values()

7 Beperk cito tot de scholen zonder de substring school in hun naam
cito[~cito.index.str.lower().str.contains('school')].head(10)

8 Voeg een nieuwe kolom 'RMSE' toe waarin de wortel - tussen de kolommen quasicito en verwacht staat
cito.describe() handige lijst met berekeningen
cito['RMSE'] = np.sqrt((cito.quesico - cito.verwacht)**2)

10 plot de inkomens
cito.inkomen.sort_values().plot();

11 In welke gemeentes staan de scholen met een inkomen van meer dan 50 000. Hoeveel scholen zijn er per gemeente. Maak een horizontale barplot. Gemeente met meeste scholen bovenaan.
cito[cito.inkomen > 50000].gemeente.value_counts().sort_values().plot(kind = barh);

---

1 Python notebook vragen

lijst van magic functies = % lsmagic
? magic voor uitgebreide beschrijving

---

kopieer file naar andere file %cp

---

verzameling ?= set()

Dir(np), Dir(pd) = lijst met functies
create the dataframe ~~iris~~ X containing the numerical data only with species
X, y = iris.drop('species', axis = 1), iris.species
2 normalize the data in X
((X - X.mean())/X.std()).head()

---

sort values
iris.sort_values(['sepal_length', 'sepal_width'], ascending = [False, True]

What are the maximal sepal lengths for each species
iris.groupby('species')['sepal_length'].max()

C = counter(text.split())
CS = pd.Series(C).sort_values(ascending = False)

Tokenize the text in variable text using split() count each token create a Series with token counts and sort with most occuring tokens on top

what percentage of unique words occurs once.
CS[CS == 1].sum()/CS.count()

---

df.groupby(0)[1].count().sort_values(
0 = winnaar, 1 = land, naam land met meeste bondsleden

winnaar.argmax land
winnaar.max → leeftijd

PANDAS

Len(set(...)) = unique | soms () achter functie | shape = dimensies

dubbel = cito.index.value_counts()
dubbel[dubbel > 1] < BM

cito[~cito.index.str.lower().str.contains('school')]

cito[cito.mean().index] - cito.mean()

iets(x) voor iets(y) = Groupby('x')[y] | 10-7-3 = .ascending[False, True] | CS[CS...]
col1 - col2

iris.query('colum > 5') | RMSE = np.sqrt(x-y)**2 | keren dat het voor komt * len woor
CS * CS.index.str.len()

c = Counter(text.split())
cs = pd.Series(c).s...v(asc..= False)

CS[CS==1].sum / CS.count() = just once
CS[CS==1].sum() / CS.sum() = all words | float(...)

kvr[kvr.Partij == 'pvdd'].Ministeie.u...c()

kvr.drop(kvr[kvr >= 5].index)

numpy

Tafel van x = np.arange(n) * x | Last_c = np.array([c[-1] For c in L])

Varray L = int(np.sqrt(len(L))) | ⅄ = L[L % N = 0] | diag = x.diagonal()

verticaal = np.reshape(L, np.size(L), 1))

mean_p_c = np.mean(L, axis = 0)

add 0,1,2 = [x[0] for x in enumerate(L)] + L

std = np.sqrt(((x-x.mean()²).mean

middle square:
r = int((L.shape[0]-4)/2)
c = int((L.shape[1]-4)/2)
return = (L[r:-r, c:-c])

sqrt diff from mean =
(x1 - x1.mean())**2

np.array([1,3,6].dtype = 'Float32')

ndim = n. of dimensions

n.shape = size each dimension | n.size = size total arra

big = np.random.randit(10**3, size=10²)
compr = sum([x**2 for x in big])
np = sum(big**2)
↑ sum of squares

every other element = x[::2] of x[1::2]

x1,x2,x3 = np.split(x[3,5]) | np.mutiply(x*x)

((x >= 7) | (x < 0.5)) | np.argsort = sort indices

np.partition(x,n) | ²log(8) = 3, 2³ = 8, ²log(16) = 4 = 2⁴ = 16

Magic shell

Files in working directory, and now large = %.ls -lh | Remove = %.rm

datatype of x = x? | ! = acces shell

Beautiful Soup

x = requests.get(url) | f.text.split('\n') | html_text

line.split(;) for line in lines | ''.join | BeautifulSoup(url, 'xml') | re.sub(ok, oke, text)
↑ = ↗

spec tekens = [^a-zA-Z0-9]

BFS(source_file, 'Html.Parser').find_all('page')

Recall = TP / (TP + FN)
Precision = TP / (TP + FP)

\w = alle woorden

Extra Pandas

S.value_counts(dropna = False) | count = nr of rows. 3+3 = 2r.

sum = 3+3 = 6

df.apply(pd.series.value_counts)

Partij_10 = [....]
cross = pd.crosstab(kvr.jaar, kvr.p..)[Partij_10]

Rename c = df.columns = ['a','b']

dropna(axis = 1) = for columns | % = vragen.max() / sum(vrage) * 100

pivot_table(df, values = 'count', columns = ['sex'], index = ['name'], fill_value = 0,
margins = True, agsfunc = np.sum

Numpy

Slicing `x[:stop:step]`

10

`x[:5]` → first 5 elements
`x[5:]` → elements after index 5
`x[4:7]` → middle sub-array
`x[::2]` → every other element
`x[1::2]` → " " starting at index 1

When step element is negative → reverse
`x[::-1]` → All elements reversed
`x[5::-2]` → reversed every other element from index 5

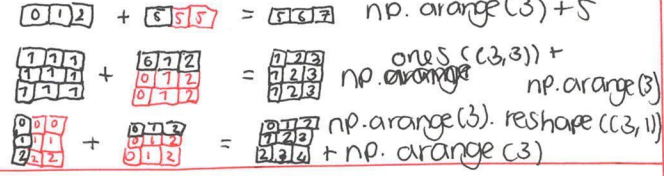
Reshape → andere vorm
concatenate → arrays platten
axis=0 → kolommen
`M.min(axis=0)` → kleinste waarde per kolom
fancy indexing: passing an array of indices to access multiple array elements at once.

Multi dimensional
`X2[:2, :3]` → two rows, three columns
`X2[:3, ::2]` → three rows, every other column
`x2[::-1, ::-1]` → rijen én kolommen omdraaien

Accessing
`X2[:,0]` → first column
`X2[0, :]` → first row
`X2[:2, :2]` → extract 2x2

Broadcasting
```
a= np.array([0,1,2])
b= np.array([5,5,5])
a+b
→ array([5,6,7])
```

2 kolommen selecteren
`df[[kolom1, kolom2]]`
Broadcasting:



np.sum
np.prod
np.mean
np.var
np.std
np.min
np.max
np.argmin
np.argmax
np.median
np.any → any True?
np.all → all True?
np.size
np.log
np.split → 1d
np.vsplit → 2d
np.hsplit → 2d

Pandas

Series als dictionaries
`Pd.Series([lijstje], index=[lijstje], index=(lijstje))`
loc: allows indexing and slicing that always references the explicit index → key value achtig
iloc: allows indexing and slicing that allows references the implicit index → echt indexen!

Pivot table
`Pivot = pd.pivot_table(df, index=..., margins=True)` ↔ All

`df = pd.read_csv(..., index_col=...)`
`df = pd.read_excel(..., ...)`
`cito[cito.index.str.lower().str.contains('achthoek')]`

Pd.len
Pd.lstrip
Pd.startswith
Pd.split
pd.index
pd.replace
Pd.value_counts
Pd.plot(kind=...)
Pd.sort_values()
↓
ascending=[True, False
str.contains
.sort_index()
.describe()
Pd.groupby
index_col=...
Pd.tail
Pd.columns
Pd.rows
Pd.shape

Ipython notebooks

%magic
%time
%timeit
%ls → welke files in dir
%cp kopieer file

DS steps

- interacting with outside world
- Preperation
- Transformation
- Modeling & computation
- Presentation

Precision = how many of the prediction of class C were correct: TP/(TP+FP)
Recall = how many instances of class C predicted to be C: TP/(TP+FN)
Accuracy = how many of all predictions were correct: ~~TP/(TP+FP)~~
Werkt niet met:
- Skewed distribution
- you can get a high reliability by the law of large numbers on the false class.

Waarom geen excel?
- Schaalt niet
- beperkte functionaliteit
- integratie met andere tools is moeilijk.

Series: 1d labeled array capable of holding any datatype
Dataframe: 2d labeled data structures

True positive	False Negative
False positive	True Negative

Predicted

vertical

Accuracy = TP+TN/all

Precision = TP/ TP+FP

Recall = TP/ TP+FN

$F_1 = 2 * (pre * Re) / (pre + Re)$

variance = sum([(x - mean)** 2 for x in list]) / (len(list) - 1)

standard dev = np.sqrt(((x1 - x1.mean())**2).mean()) == x1.std()

Boolean mask = x < 5 voor array of x[x < 5] voor list values

df.fillna(), df.dropna(), df.replace("a", "b")

df.unique(), df.duplicated("Type"), df.drop_duplicates()

% matplotlib inline

import matplotlib.pyplot as plt

s.plot(), plt.show()

Specifieke kolom = df["kolom"] of df[["kolom", "kolom"]]

Specifieke Rij = df.iloc[i]

Kolom van Rij = peiling.loc['D66']['Zetels']

unieke waarde teller = df['kolom'].value_counts()

lowercase = df['PartyName'].str.lower().str.contains(partij in lijst.lower())

pd.read_csv('File')

df.to_csv('File.csv')

df.shape, df.count()

np.zeros, np.ones, np.arange, np.linspace, np.random.random, np.loadtxt, np.info

\d digit

\D non digit

\w word

\W non word

\s space

\S non space

90 100 60 5

5 30

45 5

5 45

90	10
10	90

1000

10 9 1

990 99 891

100

9 1

Ziek 10

niet ziek 90

9 81

810

891

np.ones((3,5), dtype=float) → array([[1., 1. 1. ...
[1., 1., 1.
np.arange(0,20,2) → array([0,2,4,6..]) [1., 1., 1.

X[::2] → every other element

X[:2, :3] → first two rows & three columns
row column
column rows

np.arange(1,10).reshape((3,3))

np.any(x>5) True
np.all(x>5) false

x = rand.randint(100, size=10)

np.sort

np.argsort

df.values
df.index

data.items()

data[(data>3) & (data<5)]

data.loc
data.iloc

100 100

np.concatenate → appends

5 40%
8 60%
? 100%

5·4 + 8·6 = 68/10

200

TP	TN
FP	FN

90	90
10	10

99 99 99·0.9

import numpy as np

np.zeros/ones/full(n, Fill.value)
↳ (x,y)[z] ⇄ waardein array

np.arange(x,y) → als y leeg is, 0 x de range

np.empty(n) → Random array met n verschillende waarden.

X[:,::n] every other column

.Reshape((x,y)) → Reshape into x rows, y columns

np.vsplit(x,n) → Split array x in n arrays

h.split → spreidt over zich

np.hstack(x) → make x an horizontal array

np.vstack(x) → make x a vertical array

X>5 → array vol met booleans

X[X>5] → waarde terug die hieraan voldoen

np.sort(x) → Sorteert de array

X.std() = Standard deviation

masking = df[(df)5) & (df<10)]

fancy indexing = df[['A', 'C']]

np.sqrt(((x1-x1.mean())**2).mean() → standard deviation

Pd.Series.values → an array

pd.DataFrame(x) → maak van de array x een dataframe

X.loc[n] → haal de nde rij eruit (print)

X.iloc[n] → haal de rij index n eruit (print)

X.dropna or x.fillna(0) → vul leeg met 0 of drop dan

X.Reindex([x', y']) → geef x de nieuwe name

X.concat([df1, df2]) → voeg dfs samen

X.join() → merge dfs op index.

↳ X.set_index('key').join(other.set_index('key'))

X.mean() → per column (ax=) = columns
↳ per row

X.groupby(column list) → gesorteerd df

X.pivot_table(value, index=col, colums=col)

X.date_range(start='x', end='y')

X=date-range(start='x', periods='5')

X = pd.read_csv('x.csv')

X.Species.value_counts()

.plot(x,ind='x') → plotte dit maar als bars

X.drop('y', axis=1) → drop de rij 'y'

(X-X.mean())/X.std() → z-normalize de df

X.sort_values(['y']), ascending=[True] → sorteer op y

X.groupby('y')['z'].max() → max y for each z

X.Counter(text.split())

CS = pd.Series(X).sort_values(ascending=False) → tokenize text and create dict with counts

X[X*X.index.str.len()==2y].index → bind tovers waarin 2y herhaaldelijk voorkomt

CS[CS==1].sum()/(S.count() → percentage unieke woorden die twee voorkomen in text

CS[CS==1].sum()/(S.sum() → percentage unieke op file woorden

pd.Series(Counter(CS.values)).plot(kind='bar') → maak een bar plot met hoevaak de woorden voorkomen

X['y'] = (len(str(x)) for x in X['z']) → maak nieuwe Ry in df met de len

with v.v.open(open, encoding='utf8', mode='rt' as source-file

np.zeros
np.ones
np.full((3,5), dtype=float)
np.arange(start, end, step)
np.random.random((3,3))
→ 3x3 array, value 0-1
np.random.normal(mean, SD, (shape))
np.randint(0,10,(3,3))
→ 3x3 array, random integers [0,10]
np.random.randint(10, size(3,4))
np.any(...)
np.all( ) → TRUE/false
np.add.at(x, [ ], 1)
grid = np.arange(1,10).reshape((3,3))

x[:, np.newaxis]
→ array([[1],
[2],
[3]])

recall = TP/(TP+FN)
precision = TP/(TP+FP)
acc = How many of all pred where correct

dtypes: bool_
= int_
= intfy
= float

(:) → slicen

X[start:stop:step]

fancy indexing:
shape index array
not array being
indexed.

np.argsort(x
→ indices sorted
elements

Rand = np.random.RandomState()
x = Rand.randint(...)
np.partition(x, 2, axis=)

np.concatenate([grid, grid], axis=1]
np.vstack(x, y)
np.hstack(x, y) → vert/horz. stack arrays
np.vsplit(x, [2])
np.hsplit(x, [2])

sqrt = **2
np.sqrt

np.abs()
np.sqrt(((x1-x1.mean())**2).mean())
== x1.std()
np.add.reduce(x) → sum all elements

$std = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n}}$

np.multiply.reduce(x)
np.sum(L)
min(array), max(array)
array.min() ...
np.nansum()
np.nanmedian()
np.percentile(x, 25)

array = fancy indexing
x2[[0,1,2], [0,1,2]]
rows ↑ index/kolom ↑

series:
data.query('colom >5 and colom @a')
a = np.sqrt(5)

== np.equal != np.not_equal
< np.less <= np.less_equal
> np.greater >= np.greater_equal
np.count_nonzero(x<6) → telt alles zonder 0 en <6)

voorbeeld = pd.read_csv('....csv')['PRCP'].values

booleanoperators
& = and
| = OR
^ = XOR

display('pd.concat([x,y], ignore_index=True)')
, keys=['x','y']
, join='inner'
, join_axes=[ ]
pd.merge(x,y, left_index=True, right_on='name')
pd.merge(x,y, how='...') → inner/outer/left/right

! pwd → print working directory
! ls → list working directory
! cd → change directory
! mkdir → make new directory
! ls -lh → info bestanden
! wc -l → how many lines
! head -5 → show the top 5 lines
! rm ...text → verwijder tekst
%timeit %time %prun %paste
%cpaste %run

df.dropna(axis=
dropna(axis='columns', how='all')
, thresh=3) minimaal 3 ≠ nan
→ of 'any'

data.fillna(0)
data.fillna(method='ffill') → vorige waarde
'bfill')

① pd.MultiIndex.from_tuples(index)
② pop.reindex(index)
pop.unstack() pop.index.names('', '')
pop.stack() → multi index
idx = pd.IndexSlice → data.loc[idx[:,1], idx[:, 'HR']]
pd.MultiIndex.from_products([ ], [ ])
data.sort_index() data.sortlevel()
pop.unstack(level=)
pop.set_index([ ]) → MultiIndex
data.mean(axis=1, level=' ') → mean kolommen
pd.concat() → zelfde np.concatenate()

pd.Series([0.25, 0.5, 0.75],
index=['a','b','c'])

→ a 0.25
b 0.50
c 0.75

reverse
→ ascending=[False, True]
data.groupby('', [' '].max)

& intersection
| union
^ symmetric difference
slicing by explicit index:
data['a':'c']
implicit index:
data[0:2]
masking:
data[(data>0.3) & (data<0.8)]
fancy: data[['a', 'e']]
explicit: data.loc[1]
implicit: data.iloc[1] → plek 1
data.iloc[:row, :column]
data.ix[:3, :'pop']

pd.Series(dict)
pd.Series(data=index)
pd.Series({2:'a', 1:'b', 3:'c'}, index[3,2])
→ 3 c
2 a

states = pd.DataFrame(
pd.DataFrame(population, columns=['population'])
states.index
states.columns
pd.DataFrame(np.random.rand(3,2),
columns=['foo','bar'],
index=['a','b','c'])

np.zeros(3, dtype=[(), ()])
wanneer nan values
A+B → A.add(B, fill_value=0)

isnull()
notnull()
dropna()
fillna()

count()

+ add()
- sub(), substract()
* mul(), multiply()
/ truediv(), div(), divide()
// floordiv()
% mod()
** pow()
→ kolomen

• value_counts() plot
x,y = data.drop(' ', axis=1), data.colom
(kind='barh')

bij mergezelfde kolom:
pd.merge(x, y, on='name', suffixes=["_L", "_R"])
merged.isnull().any()
merged.loc[merged['state'].isnull(), 'state/region'].unique()

$$\text{precision} = \frac{TP}{FP + TP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{accuracy} = \frac{TP + TN}{\text{Total}}$$

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recal}}$$

predicted \ actual		
	TP	FP (Type I error)
	FN (Type II error)	TN