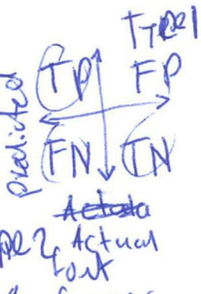
randint(range order, range boven) for x in range(lengte)
random.choice(list)
plt.plot(list1, list2)
re.sub = substitute
re.findall = match

precision = Tp / Tp+Fp Recall ~~Accuracy~~ = TP / TP+FN

~~Accuracy~~ Accuracy = Tp+TN / Total F1 = 2 * precision*Recall / precision+Recall



text = re.sub(r'\w = word of a-g, . = 0 of meer, ? = 0 of meer, + = 1 of meer,
\d = digit, \ = escape, () = capture inhoud, {3} = exact 3 instances,
[1-4] = cijfers 1 t/m 4, \w{3}+$ = de laatste 3 letters van string)

np.arange(begin range, eind range, grootte van stappen) = array van cijfers)

L.reshape(int(math.sqrt(len(L))), int(math.sqrt(len(L)))) = hervorm dimensies array, ~~vierkant~~ vierkant is dit geval.

L.reshape(L.size, 1) = 1 dimensionele array maken.

L[:,-1] geeft laatste kolom terug L[-1,:] geeft de laatste rij terug. L.mean(axis=0) column

Tel de 0e ~~kolom~~ column 0 op, etc: for row in range(0, len(L)): for column in range(0, len[row]):
L[row, column] += column return L

np.transpose(L) = spiegelen L

Middelste vierkant bestaande uit 4 cellen vierkant array: def middle_square(L):
midden = np.empty((4,4), int), middenlijst = []; for row in range(int((len(L)-4)*0,5), int((len(L)+4)*0,5):

~~float~~ temp = [], for column in range(... ↵); temp.append(~~column~~) L[row, column])
middenlijst.append(temp), midden = np.array(middenlijst) return midden

np.ones(m*n, dtype = int) = maak grid met 1'en

Buren vinden: boven = grid[(i-1) % grid.shape[0], j % grid.shape[1]]
rechtsb = grid[(i-1 % grid.shape[0], (j+1) % grid.shape[1]]

testgrid = buren(buren(grid,:,:) == grid[i,j]] = schrijft 1 voor 1 de waarden in buren die matchen op basis van conditie (vergelijkbaar met DF)

kvr = pd.read_csv('KVR.csv.gz', compression = 'gzip', sep = '\t', names = ['Bestand', 'Jaa
skipinitialspace = True)

counterdict.most_common(10) = 10 meest voorkomende

Kvr.loc[kvr['Partij'].isin(poppartjes)] of kvr.loc['kvr['Partij'] == 'd66'

pd.crosstab(kvr.Jaar, Kvr.Partij, margins = True]) = tel aantal combinaties voor beide kolommen

kvr['Partij'].str.lower()

kvr[pd.notnull(kvr['Partij'])] = drop alle NaN in Partij

$\log_2(8) = 2^{**}3 = 8$

kvr[~kvr['Partij'].str.contains('vragen')] = verwijder alle rijen met vragen in de string

plt.barh(list1, list2) = horizontale bar graph plt.hist = histogram

pivot = df.pivot_table(values = 'Count', index = 'Name', columns = 'Set', aggfunc=sum, margins = True)
pivot = pivot.sort_values(['All', 'Name'], ascending = [False, True])

cvr.Party.sum = totaal van waarde Party. kvr.Party.index.values = waardes van index.

np.info(), np.arange? df.loc[df['a'] > 10, ['a', 'c']]

Url opener

```
url = "      "
html_doc = request.get(url)
s = beautifulsoup(html_doc.content)
comments = s.findAll("li", class = " ")
```

Numpy

numpy.arange() → maak array

- reshape()
- sort()
- zeros()
- np.vstack()
- np.hstack()
- np.concatenate()

DATAFRAMES

```
df.drop("col", axis=0/1)
df.sort_index() → sort by labels
df.sort_values() → by = "country"

f = lambda x: x*2
df.apply(f)

df["col"].plot()
df["col"].value_counts()
df.loc["row/col"]
   → cf iloc voor integers.
df.dropna(thresh, how)
df.iterrows()
for index, row in ...
```

```
with gzip.open(" ".gz) as fin:
    for line in fin:
        ...
```

Numpy slicing 1D

```
x = np.arange(10)
x[:5] → eerste 5
x[5:] → elemente na 5
x[::2] → every other element
x[1::2] → ↑ starting at 1
x[::-1] → all elements reversed
x[5::-2] → reversed every other from index 5
```

2D

```
x2[:2, :3] → 2 rijen, 3 cols
x2[:2, ::3] → alle rijen, ontbeerten col
x2[:, 0] → eerste kol
```

Oefententamen

$x_2 = \begin{matrix} 35 & 24 \\ 26 & 00 \\ 16 & 77 \end{matrix}$

$x_1 = [5\ 0\ 3\ 37\ 9]$

> Laatste 3 kolomen of first 2 rows:

```
x2[0:2, -3:]
```

> change each element of x_1 by its squared differe from the mean of x_1

```
np.square(x1 - np.mean(x1))
```

> apply a bool mask to x_1 and reduce x_1 to its deelwaarde by 3

```
x1[x1 % 3 == 0]
```

> use fancy indexing to get the diagonal of x_2 (3,6,7)

```
x2[[0,1,2], [0,1,2]]
```

Slides

precision = TP/(TP+FN)

Data science = based on patterns, predicting a value on new, unseen instances of data. steps - interact with outside world, prep - transform - modeling & computation, presentation

Implicieten info explicit maken = leesbaar maken voor pc, formaat dat computationeel snel is & aansluit by beeld vd ~~mens~~ over die data

Nadeel excel - schaalt niet (geen heavy berekeningen), beperkte functionaliteit integratie andere tools

Onderzoekscyclus: idee, hypothese, vraag, data verzamelen, opschonen, herstructureren, analyse: relaties, doe statistieken, toets H_0, rapporteer.

outer products

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

> count # of True values in bool array

```
np.count_nonzero(x < 6)
```

> how many <6 in each row?

```
np.count_zeros(x<6, axis=1)
```

> any values < 6 ?

```
np.any(x<6)
```
→ same with np.all()

> sort each col of x

```
np.sort(x, axis=0)
```

#Plot how many samples there are in the iris dataset of each species → iris['species'].value_counts()
*Create the df X containing all the numerical data in iris and the series y which contains species information
⇒ X, y = iris.drop('species', axis=1), iris.species ⇒ X, y = iris[iris.columns[0:4]], iris.species
*Sort iris first by 'sepal_length' and then by 'sepal_width', ⇒ iris.sort(by=['s_length', 's_width'], ascending=[True,
*What are the maximal s_lengths for each species? ⇒ iris.groupby('species').s_length.max()
#Groupby('species') zorgt dat de kolom species alle zelfde values achter elkaar komen en alfabetisch geord
*Restrict iris to those samples with a s_length > 5 and a s_width < than √5 ⇒ iris[(iris.s_length > 5) & (iris.s_width < np.sqrt(5))]
*text = 'large string' ⇒ C = Counter(text.split()) cs = pd.series(C).sort_values(ascending = False)

---

* pd.read_excel(... .xls$, index_col='naam') ⇒ cito
* pd.read_csv(...csv, sep='\t', compression='gzip', header=0, skipinitialspace=True, name=['id', ..], im
* len(cito.index) == len(cito.index.unique()) of len(cito.index) == len(set(cito.index))
* dubbel = cito.index.value_counts() dubbel[dubbel >= 2]
* cito.sort_index()
* cito.str.lower(), str.strip().str.replace('"',''), str.replace("'", "").sort_values()
* cito[~cito.index.str.lower().str.contains('school')]
* cito['RSME'] = np.sqrt((cito.quasicito - cito.verwacht)**2)
* Je wilt van elke waarde het gemiddelde van die waardes in die kolom aftrekken. ⇒ M = cito.mea
(cito[M.index] - M)
*Plot de inkomens per school, zorg netjes geordend. ⇒ cito.inkomen.sort_values().plot();
*In welke gemeente staan de scholen waar de inkomen meer dan 50000 zijn? Hoeveel scholen heb je dan per gemeente? ⇒ cito[cito['inkomen']>50000]['gemeente'].value_counts().sort_values(
ascending = False).plot(kind='barh')

---

*Welke files zitten in deze directory (!ls) en hoe groot zijn ze? ⇒ !ls -lh | %lsmagic
*Remove the file no_good.txt ⇒ !rm no_good.txt
*Hoeveel methodes die beginnen met een i heeft een verzameling in Python? ⇒ Een set aanmaken (a = set()) en daarna naam.iTab en dan tellen

---

Fancy indexing: x = np.random.randit(100, size=10) [x[3], x[7], x[2]] vs. x[[3,7,2]]
X2 = np.arange(12).reshape((3,4)) X2[[0,1,2], [2,1,3]] ⇒ output: [2,5,11]
x[2, [2,0,1]] ⇒ [x[2,2], x[2,0], x[2,1]]

Boolean arrays as Mask: x > 3 ⇒ x[x > 3]
Sorting arrays: x = np.array([2,1,3,4,5]) np.sort(x) ↳ array([1,2,3,4,5]) np.argsort # returns indices ↳ array([1,0,3,2,4])

---

*Regex: r'[^a-z\d]'
^ ⇒ Match any character that is not in the set
a-z ⇒ Range. Match any char. in the range
\d ⇒ Digit. Match any digit char. (0-9)
.str.replace(r'[^a-z\d]', '')
↳ dit zorgt dat alle specifieke tekens wegvallen

---

*np.where()
*df.groupby('columns')
*df.sort_values(by=[column])
* Operating on Null values:
- isnull(): genarates a boolean mask indicating missing values.
- isnotnull(): opposite of isnull()
- dropna(): return a filtered version of the data #data.dropna(axis=1)
- fillna(): return a copy of the data with missing values filled or imputed

---

* pd.Series(data, index=[index])
* pd.DataFrame(np.array, columns=[...], index=[...])

* A = pd.index([1,3,5,7,9])
B = pd.index([2,3,5,7,11])

intersection (A & B ↳ Index([3,5,7])

union (A | B ↳ Index([1,2,3,5,7,9,11])

symmetric difference (A ^ B ↳ Index([1,2,9,11])

---

* data.loc['a'] ⇒ explicit
* data.iloc[0] ⇒ implicit
* ix index = hybrid of two

np.array ([range(1, 1+6) for i in n[2,4,6]])
np.zeros
np.ones
np.full
np.arange
np.random.random
randint

np.eye
np.empty
np.dtype
x.ndim
x.shape
x.size
x.itemsize
x.nbytes
x[-1]
x[-2]
x[2,2]
x[5:]
x[2,2]
x[::2] alles behalve
x[1::2]
x[:3, ::2]
all rows, except
x[:, 0] first column
x[0, :] first row
x[0,0] = 99 between
x_subcopy = x2[::2, ::2].copy()
grid.reshape
x[np.newaxis, :]
concatenate [grid, grid], axis=1)
np.vstack
np.hstack
np.sort(x)
np.argsort(x)
np.sort(x, axis=1)
np.partition
data[-1][name]

np.add(x,2)
np.absolute(x)
np.abs()
np.sin cos tan
np.arcsin
np.exp exp2 power
np.log log2
special.gamma
np.multiply
np.multiply.outer
np.random.random(100)
x.size
min(x) max(x)
M.sum()
M.sum(axis=0)
np.all
np.any
np.sort_values(col)
data[' '].sort
np.std
mean
median
percentile
plt.plot
array + 5
x - x.mean

x<3
>
>=
!=
==

np.count_nonzero(x<6
np.sum(x<6)
np.any()
np.sum(~((in<0.3) | in>0.7))
~ = not

x[x<5]
ind = [3,7,7]
x[ind]
row = [1,2,3]
col = [1,2,3]
x[row, col]
x[row[:, np.newaxis], mask
x[0] = 10

df.head
tail
shape
value_counts
dropna
replace
rename

groupby
str.len
count

index.str.contains
split
describe

Recall TP / TP+FN
Precision
TP / TP+FP

	ziek	nie
+	95	495
-	5	9405

prec 95 / 95+495 = 0,16
recall

- interactor
- preparation
- transformation
- modeling and comp
- presentation

study of th generalizable extraction of knowledge from data
common epistemic requirement in assessing whether new knowledge is actionable for decision making is its predictive power, not just its ability to explain the past

drop
requires skills in math, ML, AI stat, DB, communi + understand to Prob

---

d.values
df.iloc
df[1]
df[1:3]
df['b']
['c':'e']
{'2':4, '1':6}
df.keys
df.index
df.items
df[(data>0.3)&(data<0.8)]
df[['a','e']]
df.loc[1]
df.iloc
df.iloc[:5, :2]

df.loc[(data.dens > 100, ['pop', 'dens'])
df = pd.DataFrame(np.random.rand(3,4)), columns = [ ])
np.exp, sin, pi
A.add(data, fill_value=0)
rank
stack
df.subtract(df['R'], axis=0)
df.iloc[0::2]

isnull, notnull, dropna, fillna
pop[[i for i in pop.index if i[1]=2010]]
health.loc[:, ('Bob', 'HR')]
data.sort_index()
reset_index

pd.concat → append
df.append(df2)
pd.merge pd.merge(df, df2, on='hours')
left_on / right_on
how = inner / outer / right / left

help(function)
DF = Dataframe
NA = Numpy Array

%timeit

Recall = True P / (True P + False N)

precision = tp / (Tp + Fp)

Numpy → ?np → documentatie
Datatypes: Int[8/16/32/64], bool, float[32/64]

np.array([list], dtyp="DT")

np.[ones/zeros]([length/(length, width)], dtyp="dt")
np.random.[random/randint/normal](low, high, (height, width))
np.linspace(low, high, N_steps)
np.array(low, high, step_size)
NA.[shape/size/ndim]
NA[row_slice, col_slice]
np.[concatenate/vstack/hstack/dstack/split/[v/h]split] => adding/splitting
NA.reshape((height, width))
np.[log2/log/log/sqrt]
np.[sum/[idx/arg][min/max],()
NA.[sum/mean/std]()
mask => NA[NA % 3 == 0]

---

pandas
pd.read_[csv/html/excel]('file', args...)
Kolom => df.colname of df['colname']
aantallen => df.value_counts()[.plot(kind='')] => plotten
sorteren => df.sort_[index/values](by=['col'], ascending=[false/true]
nooi maken => df.[col/index].str.[strip/replace/lower]()
kolom delete => df.drop('col', axis=1)
groeperen => df.groupby('col')
crosstabel => pd.crosstab(df1, df2)
entry in df => df.isin([df/list])
Functie over => df.apply(func)
pivot => pd.pivot_table(df, index='colname', columns=['col'], values=
aggfunc=np.sum, margins=True)
Query = pd df.query('query') => vb query 'col1 > 5 and col2 < @var' var = 2

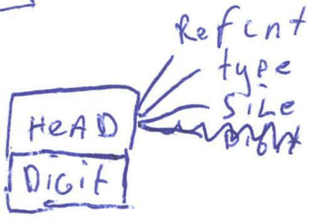
Python

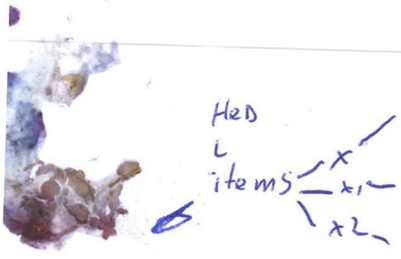
* keyword ? <tab>

%% paste Bij plakken code
%% Run external code
%% ls magic !
%% history = prev output

print(out[8])
print(_) = prev output
; = Suppres out
%% +mode = Debug mode (verbose, plain)
%% pfun = where most time
%% memit = memory

Numpy





R1 -> Left +1
R2 -> 1 set to match
R3 -> any dis and no 1 =err

x[[1,2,3]] = 5 !

np.zeros(4, dtype=[('x','type')])

Step = [::2]
Reverse = [::-1]
ReverseCol = [:,::-1]
Fcol = [:,0]
Slice = view
(out=) = where to save
Reduce = func on all values
(Axis=0) colums
(axis=1) Rows

x[np.newaxis,:] = x.reshape(1,5)

x[[3,2,4]] = x[3] ...

Shape of index array
argsort = indeces of sort items

Pandas

Precision - Positive predicted value
Measures how many of the samples predicted positive are actually positive. TP / TP + FP

Recall - How many of the positive samples are captured by the positive predictions
TP / TP + FN

Data Science: based on patterns in data prediction a value on new unseen instances of data
steps: interacting with outside world - preparation - transformation - modeling & computing - presentation
Def: The study of the generalizable extraction of knowledge
Impliciete informatie expliciet maken voor een computer & in een formaat dat computationeel snel is en aansluit bij beeld dat de mens heeft van die informatie
Nadeel excel schaalt niet, beperkte functionaliteit, integratie andere tools
Series one dimensional, DF two

onderzoekcyclus: idee/hypothese/vraag
data verzamelen en opschonen. herstructureer en analyseer
Pandas goed voor analyseren en modelleren

which files, how big
%ls -lh
remove
%rm filename
inspect algorithm
%time, %timeit, %prun
x? -> datatype

boolean mask
X1[X1 % 3 == 0]
Fancy indexing
X2[[0,1,2],[0,1,2]]
plt.plot
mean per colum
L.mean(axis=0)
telop bij colum 0,1,2 enz
L + np.where(L==L)[1].reshape(L.shape[0], L.shape[1])

wissel colum & rows
transpose
np.log2()

x for x in df.iterrows

Count and plot:
df.colum.value_counts().plot.barh()
maximum value for class species
df.groupby('species')['sepal-length'].max()
restrict df met meerdere argumenten
iris = iris[(arg1) & (arg2)]
sort values and show top 10
df.sort_values(ascending=False)[:10] or .head(10)
lower all in colum
df.colum.str.lower()
replace all not word with ""
df.colum.replace(to_replace='\W', value="", regex=True)
can also be value = np.nan

series.value_counts()
DF.sort_index()
Get len of index words
usefull in series
CS.index.str.len()
pd.drop(colum, axis=1)
drinks.groupby('continent').beer_servings.mean()
.corr
df.pivot_table.sort_values

crosstab().nlargest(11, "name")
change order
df.columns.tolist()
columns = col[1:] + col[0]
cross = cross[columns]

Text parsen

" ".join(lijst) → string van elementen van lijst

string.split() → lijst met woorden, split op spatie

BeautifulSoup(string) → string zonder Html, .text alleen tekst

soup.findAll("li", class_=re.compile("/interlanguage-link"))

df.kolom.str.lower()

df.kolom.str.replace(item, '')

pandas

pandas

df.fillna(0) Wissen

np.log2( )

log2(16) = 4 ; 2^4 = 16

np.absolute(g) Kolo

X[[0,1,2],[0,

df[[kolom1, kolom2]].min(axis=1

df.drop(kolom, axis=1)

df = df.mean()

df.std()

Numpy

%ls -lh groottefile directory %rm remove het %time var? je lijn dataset	Broadc. shapes equal no eq dem, shape 1

(Array1 + Array2)/2 gemiddelde tussen elementen in kolomm

list(range(100))[1::2] oneven getallen in range

" "[2::2] even " "

df.groupby(k)[k2].ma

max van gegroep op k2

df.groupby(kolom).count()

Counter( ) → unieke items lijst teller

Array[rij][kolom]

int(Float) → geheel getal

Accuracy: Trues / totaal 1
precision: Tp/(Tp+Fp) 2
recall: Tp/(Tp+Fn) 3
F→2*2*3/(2+3)
willekeurig

Aarray + np.Arrange(0, Array.shape[1]) → index + het item

item

df[df.index.str.l
* df == 24].ind

df[df==1].size/
df.size

Array.mean(axis=1) → gemiddeldes kolommen

Array[Array % 10 == 0] → alleen waardes deelbaar door 10

Array.size → aantal elementen, ~~of~~ len(array) → aantal rijen, Array.shape[0] aantal rijen

Array.reshape(rijen, kolommen) reshape(12,1) id [12]

random.randint(0, 5), random.choice([ ]) → kan je ook excluden

Pandas

linux topic

pd.read_csv( ) * sep= , skipinitialspace= , names= , index_col

df=[[df.columns[0]]] → eerste kolom

df[['string','string2']] → selecteer twee kolomm

pd.pivot_table(values=kolom
vullen en functie, index=unieke
columns=sex(F/m), margin=True,

df.kolom.value_counts() → unieke waardes count • values → count • index → unieke waardes • sort_index() → sorteren (by=[kolom, kolom2], ascending=[T,F]

alfabetisch

df[pd.notnull(df.kolom)] → bepaalde kolom nans uit halen

df[~df.kolom.str.contains(string)] → ~~kolom~~ waardes in kolomm die bepaalde string bevatten vervallen

df.kolom.loc[df.kolom >= 5] → rijen waar count >= 5, • idxmax() → max index • max() → value

df.kolom.str.count("\?")

M.shape (2,3) -> (2,3) - (2,3)
a.shape (3,) -> (1,3) -> (2,3)
np.sqrt(((x1 - x1.mean())**2).mean())
(x1 - x1.mean()) ** 2
x1.std()

x1[x1 % 3 == 0]
x2[[0,1,2],[0,1,2]]
[x**2 for x in lgar]

x,y = df.drop('spe', axis=1), df.spe
((X - X.mean())/X.std()).head()
iris.sort_values(['sw','sl'], asc=[F,T])
'sw'.groupby('sw')['sw'].max()
iris[(iris.sleng>5) & (iris.sw<(np.sqrt(5)))]
a = np.sqrt(5)
iris.query('sepal >5 and sew <@a')

Counter(text.split()) = C
pd.Series(C).sort_values(asc=False) = CS
CS[CS.index.str.len() == 2].index

t = [cs.index.str.len() * CS.values == 2]
np.where(t)[1]
CS.index[np.where(t)[1]]

CS[CS==1].sum() / CS.count()
(CS==1).mean()
pd.Series(Counter(CS.values)).plot()
CS.value_counts().sort_index().plot

bd = pd.read_html
bd.groupby(0)[1].count().sort_values().tail(1)
wm = bd.groupby(0)[1].count()
wm.argmax(), wm.max
bd.groupby(0)[2].value_counts().sort_values().tail(1)

np.multiply.outer(x,x)
x = np.random.random((10,3))
xmean = x.mean(0)
xcenter = x - xmean

np.sort(x)[x, axis=0,1]
np.argsort for index

Regex: r'[^a-z\d]'
^ = Match any char. not inset.
a-z = range match '' '' in range
\d = digit Match any digit (0-9)
.str.replace(r'[^a-z\d]', '')

df = pd.read_excel('', index_col="~")
c.to.shape, c.to.columns
dwo = c.to.index.value_counts()
dwo[dwo>1]
c.to.sort_index()
''.index.str.lower().str.strip().str.replace(" ", '').sort_values()
''[~c.to.index.str.lower().str.contains('sauce')]
c.to.describe()
c.to['rew'] = np.sqrt((c.to.qu - c.t.v)**2)
M = c.to.mean(), (c.to[M.index] - M].tail()
c.to.inkomen.sort_values().plot();
c.to[c.to.inkomen > 50000].gemeente.value_counts().s_val

url = 'wiki'
html_doc = requests.get(url)
soup = BeautifulSoup(html_doc.content)
soup.prettify()
soup.findAll('li', class_ = re.compile("^interlink"))[:5]
a = soup.[0].a
a.attrs

li =
dict = {i.a.attrs['lang']: i.a.attrs['href'] for i in li}
comments = s.jA('li', class_ = 'comment')
c_list = [(soup.title.text, c.attrs['data-id'], c.find['span', class_ = 'uname'])
comm_df = pd.DataFrame(c_list)
''.columns = ['', '', '']

comment_df.head()

x[::2] = [0,2,4,6]
x[1::2] = [1,3,5,7]
x[::-1] = [9,8,7,6]
x[5::-2] = [5,3,1]

x2[::-1, ::-1] -> reverse
x[np.newaxis, :]

x1,x2,x3 = np.split(x, [1,2

np.linspace(0,1,5)
np.random.random((3,3))
np.random.normal(0,1,(3,3)), np.ra.randint(0,10,(3,3))
np.eye(3), np.empty, np.zeros
np.vstack([x,grid])
hstack

sum = np.add.reduce(x)
np.multiply.reduce(x)
'' ''.accumulate(x)

np.sum(x<6, axis=1)
np.any(x>8)
np.all(x==6)

[x[3], x[4], x[2]]
x[2, [2,0,1]]
x[1:, [2,0,1]]

data.isnull()
''.notnull()
.fillna(0)

pd.merge(df1, df2)
left_on
right_on
on

data.keys()
list(data.items())
data[(data>0.3) & (data<0.8)]

Numpy

```
x1 = array([5,0,3,3,7,9])   x2 = array([[3,5,2,4],
                                        [7,6,8,8],
                                        [1,6,7,7]])
last 3: x1[-3:] = [3,7,9]
last 3 col, first 2 rows: x2[:2,-3:]
x2.ndim, x2.size: (3,4), 12   Reshape 1 dim: x2.reshape(x2.size,)
squared diff each element: Q = (x1 - x1.mean())**2
Std deviation: x1.std = np.sqrt(Q.mean())
adding different shapes: "Broadcast regels worden bekeken"
ints div by 3: x1[x1 % 3 == 0] = [0,3,3,9]
fancy index: x2[[0,1,2],[0,1,2]] = [3,6,7]
Sum of squares big array: ba = np.random.randint(10**3, size=10**6)
%timeit sum([x**2 for x in ba])
%timeit sum(ba**2)
```

```
tafel van 7 in 4 x: np.arange(0,(4*7)+7,7)
Reshape to square: length = int(math.sqrt(array.size))
                   array.reshape(length, length)
Reshape to 1 dim: array.reshape(array.size,)
last row: array[-1:] | last column: array[:,-1]
mean per column: array.mean(axis=0)
middle square: Afstek = int(((array.shape[1]-4)/2))
               & array[Afstek:Afstek+4, Afstek:Afstek+4]
```

```
working file directory: %ls -lh | Rem: %rm no_good.txt
inspect algoritm: %time %timeit %prun | find data type x: x?
```

DataScience: Based on patterns in data Prediction:
- a value on new unseen instances of data (classification (class) - regression (grade))
- traditional statistics you explain

Steps: Interacting with outside world - Preparation - transformation - modeling & computation - Presentation

def: the study of the generalizable extraction of knowl.

Recall: TP/(TP+FN) | Precision: TP/(TP+FP)

```
A column: mozilla/5.0
frame.a.dropna().str.split('/').str.get(0).value_counts.head(1)
For mean, max, min, std: dataFrame.describe()
Number of unique names: dataFrame.names.unique().size
Pivot table with year as index, gender as column sum all births as value:
names.pivot_table(values='birth', index='year', columns='sex', aggfunc=sum)
john = dataFrame[(dataFrame.name=='john')
FemJohn = dataFrame[(dataFrame.name=='john') & (names.sex=='F')]
```

W2

Impliciete informatie expliciet maken:
- leesbaar maken voor een computer
- in een formaat dat computationeel snel is en aansluit bij het beeld dat de mens heeft van die informatie

W3

Nadelen excel: 1 schaalt niet (geen grote datasets, kan geen heavy berekeningen aan) 2. Beperkte functionaliteit (opschonen van data is lastig) 3. Integratie met andere tools (kan natuurlijk wel in Python)

Ondezoeks Cyclus: kom met idee/hypot/vraag - verzamel data - schoon je data op - herstructureer je data - analyse: ga rekenen, doe statistics, toets hypot - Rapporteer

Pandas datastructures: Series & DataFrame

Series: one dimensional labeled array (dict)

```
s = pd.Series(Randn(5), index=['a','b','c','d','e'])
a  0,448597
b -1,385417
c -0,992218
d -0,656515
e  0,124775
```

```
S['a'] = 0,448597
S[2] = -0,992218
S.mean() = -0,337068
S.max() = 0,448597
S.idxmax = 'a'
```

the dot-Product/cosine symilarity:

```
[1,2,3] . [5,6,7] = (1*5)+(2*6)+(3*7)
S.dot(s)
s*s =
a  0,201239
b  1,919375
c  0,984496
d  0,431012
e  0,015569
```

```
DataFrame = df = pd.read_csv('x.csv', index_col='date')
eerste kolom: df.date | df.index
kolom a&B: df[['A','B']]
df.(>=3 (True false)
df.B.median = 3 | df.loc['2009-01-02'] | row
df[df['B']>=3] = rows that give true
df * 5 = mult everything by 5
```

date	A	B	C
2009-01-01	a	1	2
2009-01-02	b	3	4
2009-01-03	c	4	5

Pandas

```
iris = pd.read_csv('bestand.csv')
iris:
```

	sepal_length	sepal_width	petal_l	petal_w	species
0	5,1	3,5	1,4	0,2	setosa
1	6,9	3,2	5,7	2,3	virginica

```
aantal samples each species + Plot: iris.species.value_counts().Plot(kind='bo
Create df x with all numerical: X = iris.drop('species', axis=1)
Y with species: iris.species
2 normalize data: ((X-X.mean()) / X.std()).head()
Std of normal data: ((X-X.mean())/x.std()).std() (should 1)
Sort by sepal l,w: iris.sort_values(['sepal_l','sepal_w'], ascending=[fa, Tru]
max sepal length per species: iris.groupby('species')['sepal_l'].max()
sepal_l > 5 & sepal_w < sqrt(5): x = np.sqrt(5)
                                 iris.query('sepal_l > 5 and sepal_w < @x'
Tokenize text: C = Counter(text.split())
               CS = pd.Series(C).sort_values(ascending=False)
Find all tokens which occupy 24 characters:
CS[CS.index.str.len() == 24].index
Percentage unique words: (CS==1).mean()
percentage all words: (CS==1).sum()/CS.sum
Plot # of words how many times:
pd.Series(Counter(cs.values)).plot(kind='bar'); CS.value_counts().sort_index().plot(kind=bar);
```

```
kvr = pd.Read_csv(linktofile(), sep='\t', header=None, skipinitialspace=True
kvr.columns = ['ID', 'jaar', 'Partij', 'Onderwerp', 'vraag', 'Reactie', 'minis']
Crosstable jaren x partijen top 10 Partijen:
meest actief = kvr.Partij.value_counts()[:10]
CROSS = pd.crosstab(kvr.Partij, kvr.Jaar)[meestactief]
Vraagcanke = kvr.Vraag[kvr.Vraag.str.count(r'\?') <= 50]
Vraagcounter = Vraag counter.str.count(r'\?')
```

```
Pivot = df.pivot_table(values='count', index='name', column
= 'sex', margins='All', aggfunc=sum).sort_values(['All','name']
ascending = [False, True])
Pivot['Ratio'] = pd.Series(np.log2(Pivot['M']/Pivot['F'],
index = Pivot.index)
Pivot = Pivot[((Pivot['Ratio'] <= -4) | (Pivot['Ratio'] >= 4))
& (Pivot['All'] >= 10000)]
Pivot['error'] = Pivot[['M','F']].min(axis=1)
act_error = (Pivot['error'].sum()) / Pivot['All'].sum()) * 100
Pivot['kleinste'] = Pivot[['M','F']].min(axis=1)
most_errors = Pivot.kleinste.idxmax()
kvr.Partij.replace(to_replace="Groen|fe", value='', regex=True)
```

```
aantal bomen elke soort: bomen_df.Boomsoort.str.lower()
                         .value_counts()
aantal bomen per jaar: df.Plantjaar.dropna().astype(int)
                       .value_counts().sort_index().plot(kind='bar'
laat 2 kolommen zien: df[df['Plantjaar']>=2000]
                      [['Plantjaar','Boomsoort']]
veranderen: stammen['Stamdiameter'].str.replace('cm','')
```

```
magic: %lsmagic %%cellmagic
Pasting code blocks: %paste %cpaste
Run external code: %run
Timing: %time %timeit %prun
magic func: %magic -> list: %lsmagic
            %history
working dir: %pwd
content size: %ls
change ddir: %cd
make dir: %mkdir
move file: %mv file loc
Remove: %rm
all data vars: %who?
mem usage:
```

a science = processing values on [illegible]
1. interact w/ outside world (read/write variety of file formats/databases)
2. prep data
3. transformation (apply math/stats)
4. modelling + computation
5. presentation (visualization)

series = 1D labeled array any data type
df = 2D labeled data structure.

precision = how many of pred correct (TP/TP+FP)
recall = how many instances of true cc pred. to be cc?
accuracy = how many actually correct. (TP/TP+FN)

```
np.array([1,2,3], dtype='float64')
np.zeros(4, dtype=int)
np.ones((3,5), dtype=float)
np.full((3,5), 7)
np.arange(0,20,2)
np.linspace(0,1,5)
np.random.random((3,3))
np.random.randint(0,10(3,3))
```

```
df.ndim
df.shape
df.size
df.dtype
```

```
data.values
data.columns
data.index
```

```
df[2,0] = 12 #index en modify
df[::2] # every other element
x[1::2] # ↗ from index 1
x[::-1] # all elem. reversed.
x[5::-2] # reverse from 5
x[:2, :3] # two rows, three columns
x[:3, ::2] # all rows, every other column
x[:, 0] first column
x[0, :] first row
```

```
grid = np.arange(16).reshape((4,4))
np.concatenate([x,y,m])
np.vstack([x,grid]) vertically stack array
np.hstack([grid,x])
np.vsplit(grid,[2]) split hor/vert.
np.hsplit(grid,[2])
np.absolute / abs(x)
np.add(x,2) of subtract, multiply, mod
ufuncs: array+5
np.sum(array)
np.min(a)
np.max(a)
df.min(axis=0)
df.min(axis=1)
np.argmin/max ↗ find index of minimum.
np.any() any/all elements true?
np.all()
np.sort(array)
np.argsort() #indexes of sorted el.
np.sort(x, axis=0) – sort column
          axis=1) – sort row
np.partition(x, axis=1) smallest values left from index 3.
ind = [3,4,7]
x[ind] # returns these values
x[2,[2,0,1]] # from 3rd row get those indexes.
```

```
#name from last row:
data[-1]['name']
#name where age < 30
data[data['age']<30]['name']
```

```
pd.DataFrame(data)
# transform dict to df
```

```
# Index.
ind.size
ind.shape
ind.ndim
ind.dtype.
```

```
#intersection
A & B
# union
A | B
# sym. difference
A ^ B
```

```
data['a':'c']
data[['a','c']]
Fancy
#masking
data[(data>0.3) & (data<4)]
# locate:
data.loc[1:3]
data.iloc[1] for index ⇒
data.ix[:3,'pop']
data.loc[data.dens>100, ['pop','dens']]
#.transpose.
data.T
```

```
np.nan
isnull()
notnull()
df.dropna()
  axis='columns'
  how='all'
  tresh = 3
df.fillna(0) #fill missing values with 0's
```

```
pd.concat([x,y])
df1.append(df2)
pd.concat(x,y,
      ignore_index=True)
df.merge(d1,d2)
df1.join(df2)
```

```
#index
.index[0]
```

```
drop: kvr = kvr[~kvr.index[kvr[party] == row]]
df.pivot_table(values='count', index=name, columns=sex, aggfunc=sum
kvr.columns = ['str.']
kruistabel: pd.crosstab(x, y)[kvr.party_value_cou
Top10                                [:10].in
```

```
cities = [r['cy'] for r in records if 'cy' in r]
from collections import counter
cities = Counter(cities)
cities.most_common(10)
```

Amejet Robgn
1126911

```
cit = pd.DataFrame.from_dict(cities, orient='index')
cit.columns = ['aantal'] # geef naam
cit.sort_values('aantal', ascending=False, inplace=
                                          True)
short_records = [{'country_code': r[c], 'TimeZone': r['tz']}
  for r in records if 'tz' in r and
  'c' in r]
#how often time zone occurs
tz_counts = frame.tz.dropna().str.split('/')
frame.tz.value_counts()
C = frame.a.dropna().str.split('/').str.get(0)
      .value_counts().head(6)
#unique names
df.name.unique().size
```

```
# births per year per gender:
total_births = names.pivot_table(
    values='births', index='year',
    columns='sex', aggfunc=sum)
#find col with name:
john = names[names.name==]ohn]
subset = ↗ | 'mary']
#percentage of all kids
totals = names.groupby('year')['births'].su
(johnandmary['John']/totals*100.round(2)
#percentage of all boys
totalsboys = names[names.sex=='M'].groupby('year')[birt
johnmary['John']/totalsboys*100.round(2).head(5)
```

```
#n getallen deelbaar door 0
np.arange(0,n*0,0)
# viernant array
n = int(math.sqrt(len(L)))
np.reshape(L,(n,n))
# reshape vertical
np.reshape(L(L.size,1))
# laatste kolom vs row
L[-1,:] vs L[:][-1]
# array deelbaar door n
L[np.where(L.n==0)]
#gem per kolom
(L.mean(axis=0))
# else waarde + 1
L+np.arange(len(L[0]))
```

```
#square:
x = int(L.shape[0]/2)
L[x-2:x+2, x-2:x+2]
```

```
# replace special char
.replace('\W', '')
# remove words
kvr[party] = [(re.sub(r'a
|beident', '', str(p))) for p
                    in kvr[party
pf
for row in kvr.party:
  if row == '':
    i = kvr.index[kvr[party
                  == row]
    kvr.drop(i)
# voor elke pvdd welke
ministerie en hoevaak.
kvr2 = (kvr.loc[kvr[
'party'] == party]).Ministerie.
  value_counts().to_frame()
```

```
# max value
Male.loc[Male['f'].idxmax()
```

NUMPY

arange:	generate array
.reshape:	change shape
sort:	sorted copy
zeros:	generate filled 0
add:	add two arr
substract:	- two arr
multiply:	* two arr
divide:	/ two arr
exp:	$2^{\wedge}$ two arr
sqrt:	$\sqrt{*}$ one arr
l.sum:	sum arr
l.min:	min of arr
l.max(a=0):	max of arr
l.mean:	mean of arr
.mode:	mode of arr
l.median:	median of arr
vstack:	2 arr to matrix
concatenate:	add 2 arr together
count_nonzero:	counts True
l.any:	bool over any
l.all:	bool over all
array:	turn list into np
log2:	log2 over arr
transpose:	flip arr
.where:	find all to bool

NUMPY SLICING

a[0:4, 3] on col 3
a[:, 1] whole col 1
a[::2] every other el
a[::-1] all, reversed
a[::-2] every other reversed
a[:2, :3] two rows, three cols
a[:3, ::2] every other col
a[:, 0] first col
a[0, :] first row
a[::-1, ::-1] mirror

PANDAS

DataFrame:	create df	~ → not
.iat/.iloc:	element at index	| → or
.at/.loc:	element at name	& → and
.ix:	select row	for index, row in df.iterrows()
l.drop:	drop values	Str.split(del)
l.sort_index:	sort along axis	del.join(list)
l.sort_values:	sort by values.	Str.replace(x, y).
l.rank:	assign ranks	
l.shape:	(x, y) shape	
l.index:	arr of index col	
l.columns:	arr of column names	
l.info:	datatype info	
l.count:	number of non NA values	
l.sum:	sum of df	
l.cumsum:	accum. sum of df	
l.apply:	apply func to df	
pivot.table	pivot df	
crosstab	crosstable df	
l.isin	bool check	
l.hist	histogram of df	

FORMULAS

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

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

SNIPPETS

X[X % 3 == 0]
X[[0,1,2],[0,1,2]] → fancy index
df.col.value_counts().plot.barh
plt.plot

REGEX

\w \d \s
\W \D \S
[abc] ↔ [^abc]
[a-z]
\t \n
(?=abc)
(?!abc)
* 0+
\+ 1+
\? 0/1
? lazy.

re.sub
re.findall
re.match
re.search

RANDOM

random.randin
random.choice

STEPS OF DATA SCIENCE

1. interacting with ~~real~~ outside world
2. preparation
3. transformation
4. modelling & computing
5. presentation

DEFINITION

Study of the generalizable extraction of knowledge

CON EXCEL

Schaalt niet, beperkte function.

ONDERZOEKSCYCLUS

hypothese/vraag → verzamel data → schoon data op → herstructureer data → analyse (statistiek, hypothesetests) → rapporteer

pandas

- Series: one dimen. labeled arr.
- dataframe: two dimen. labeled