150 mensen in TK

```
%matplotlib inline

from __future__ import division
```

Element letterlijk printen
```
print etree.tostring(xml OF element,
pretty_print=True)
```

Boxplotje maken
```
df.boxplot(by = 'Party_name', column =
'Fraction', figsize = (16,10), rot =
30)
```

Bepaalde kolommen (met bepaalde inhoud) niet krijgen:
```
party_columns =
df.columns[~(df.columns.str.contains('
low') |
df.columns.str.contains('high'))]
```

Selecteren op inhoud van index:
```
index.str.contains ('…..')
```

Specifieke kolom pakken
```
df['kolomnaam']
```

Meerdere kolommen pakken
```
df[['kolom1', 'kolom2']]
```

Specifieke rij pakken met index
```
df.loc['index-waarde']
```

De i-de rij pakken
```
df.iloc[i]
df.iloc[3]
df.iloc[-1]
```

Kolom als index instellen
```
df = df.set_index('kolom')
```

Specifieke kolom van specifieke rij pakken
```
peiling.loc['D66']['Zetels']
peiling['Zetels'].loc['D66']
```

Unieke waarden in kolom
```
df['kolom'].unique()
```

Unieke waarden tellen
```
df['kolom'].value_counts()
```

Groupby element unstacken
```
groupby element.size().unstack()
```

dataframe unstacken + index resetten
```
df.unstack().reset_index()
```

Capslock van kandidaten vermijden
```
candidates_df['Party_name'].str.lower(
).str.contains(party_in_lijst.lower())
```

Sorteren op kolom
```
df.sort_values('kolom',
ascending=True)
```

List van DataFrames aan elkaar plakken
```
pd.concat
```

XML bestand importen
```
from lxml import etree

with open
('Kandidatenlijsten_TK2017_Amsterdam.e
ml.xml')as f:

      parser =
      etree.XMLParser(ns_clean=True)
      xml = etree.parse(f, parser)

ns = { ns: '{'+ url +'}' for ns, url
in xml.getroot().nsmap.items() }

ns

contest = xml.find('//'+ ns[None]
+'Contest')

all_candidates = []

affiliations =
contest.findall(ns[None] +
'Affiliation')

for party in affiliations:
 > party.find(affiliation_identifier)
 > affiliation_identifier.find
  (registered_name).text
 > candidates: party.findall

for candidate in candidates:

      candidate_name,
candidate_adress, candidate_country,
candidate_gender

      if candidate_country is not
      None:
candidate_country_initials,
candidate_locality_notNL,
candidate_adress
      else: candidate_country_initials
= 'NL', candidate_locality,
candidate_adress
```

Attribuut van element
```
Candidate_identifier =
candidate.find(ns[None] +
'CandidateIdentifier')

candidate_position =
candidate_identifier.attrib['Id']


if candidate_prefix is not None: …
```

Alle variabelen toevoegen aan een lijstje
```
.append(( ))
```

Kruistabel: `pd.crosstab(df[kolom], df[kolom], margins = True (voor total))`
apply voor elk element in het hele dataframe: `applymap`

```
partijen = peilingen.columns[1:]

for partij in partijen:

 zetels_gesplit =
 peilingen[partij].str.split(';').str

 peilingen[partij +'.low'],
 peilingen[partij], peilingen[partij
 +'.high'] = zetels_gesplit

def alleen_de_middelste(x):
      return x.split(';')[1]

peilingen[partijen] =
peilingen[partijen].applymap(alleen_de
_middelste)
```

```
import pandas as pd
import os
from lxml import etree
import xml.etree.ElementTree as ET
%matplotlib inline
from math import sqrt
import matplotlib.pyplot as plt
```

```
#DF van excel
pwijzer = pd.read_excel('naam.xlsx')

#informatie van je DF (eerst aantal tabellen)
print (pwijzer.shape, pwijzer.describe)

pwijzer.head()

# geeft kolom
pwijzer['Partij']

pwijzer['Partij'].sum() #telt alle waardes op
pwijzer['Partij'].count() #telt aantal rijen

#kolom toevoegen met nieuwe waarde,bijv.
gem.
pwijzer['GemZetels'] = (pwijzer['ZetelsLaag'] +
pwijzer['ZetelsHoog']) / 2
pwijzer

#getal uit DF (als variabele voor functie)
inter = pwijzer['Percentage'].loc[(pwijzer['Partij']
== 'PVV')].values[0] #kan ook met 'Partij'

#kolom weg en index op partij
pwijzer[['Partij',
'Percentage']].set_index('Partij').head()

#partij meer dan 25 zetels
zetels = pwijzer.loc[pwijzer['Zetels'] > 25]

#Sorteer op kolom
pwijzer2 = pwijzer2.sort_values(by='Zetels',
ascending=False)
```

```
#XML
#TAGS
import xml.etree.ElementTree as ET
tree = ET.parse('.xml')
root = tree.getroot()
root.tag
root.attrib

# tags:
for tag in tree.iter():
  print (tag)

#OF
tags=[ ]
for event, elem in ET.iterparse('naam.xml'):
  tag=elem.tag
  tags.append(tag)
  elem.clear()

tags

#aantal V/M per kandidaatlijst (loop door alle
.xml)
totalman = 0
totalvrouw = 0
path = 'naam-map'
#Kandidatenlijsten_EML_TK2017-20170213/
#print os.listdir(path) (test)
for _file in os.listdir(path):
  if not _file.endswith('xml'):continue
  #print _file(test)
  countman = 0
  countvrouw = 0
  with open(path+_file):
    for event, elem in ET.iterparse(path +_file):
      tag = elem.tag
      value = elem.text
      if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Gender" :
        if value == 'male':
          countman += 1
          totalman += 1
        else:
          countvrouw += 1

    elem.clear()
  print (_file.strip('.eml.xml'), countman,
countvrouw,   (countman + countvrouw))

#hoeveel m/v+tot. Per kandidaatlijst
countman2 = 0
countvrouw2 = 0
for event, elem in
ET.iterparse('amsterdam.xml'):
  tag = elem.tag
  value = elem.text
  if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Gender" :
    if value == 'male':
      countman2 += 1
    else:
      countvrouw2 += 1
  elem.clear()
print ('aantal man: = ', countman2)
print ('aantal vrouw: = ', countvrouw2)
print ('total: = ', (countvrouw2 + countman2))
```

```
#list partijen op kandidaatlijst
partijen = [ ]
for event, elem in
ET.iterparse('amsterdam.xml'):
  tag = elem.tag
  value = elem.text
  if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Registere
dName":
    partijen.append(value)
  elem.clear()
partijen

#MAAK DF
partij = [ ]
voornaam = [ ]
achternaam = [ ]
gender = [ ]

for event, elem in
ET.iterparse('Kandidatenlijsten_TK2017_Amster
dam.eml.xml'):
  tag = elem.tag
  value = elem.text
  if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Registere
dName":
    partij_val = value
  if tag ==
"{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}Firs
tName":
    partij.append(partij_val)
    voornaam.append(value)
  if tag ==
"{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}Las
tName":
    achternaam.append(value)
  if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Gender":
    if value == 'male':
      gender_val = 'man'
    else:
      gender_val = 'vrouw'
    gender.append(gender_val)

df = pd.DataFrame({'partij':partij,
'voornaam':voornaam,
'achternaam':achternaam, 'geslacht':gender})
df.head()

#aantal unique partijen
len(df['partij'].unique())

#values kolom weergeven
df['partij'].head()

#alle M van partij
df.loc[(df['partij'] == 'X') & (df['geslacht'] ==
'man')]

len(df.loc[(df['partij'] == 'X') & (df['geslacht'] ==
'man')])

#wie zit in partij x?
df.loc[df['partij'].isin(['X','Y'])]

#Vrouw in partij?
df['partij'].loc[(df['geslacht'] ==
'vrouw')].unique()

len(df['partij'].loc[(df['geslacht'] ==
'vrouw')].unique())

#alle voornamen met Mo
df[['voornaam',
'partij']].loc[(df['voornaam'].str.startswith('Mo'))
]
```

```
PLOTTEN

#M/V van een partij
verhouding = df['geslacht'].loc[(df['partij'] ==
'X')].value_counts()

verhouding.plot(kind='pie')

#woonplaats binnen een partij
verhouding = df['woonplaats'].loc[(df['partij'] ==
'X')].value_counts()

verhouding.plot(kind='pie')

#partij met aantal kandidaten
partij_aantal = df['partij'].value_counts()
partij_aantal.plot(kind='bar')

#M-V van partij in BAR
vrouwen = [ ]
mannen = [ ]
for a in df['partij'].unique():
  try:
    V = df['geslacht'].loc[(df['partij'] == a) &
(df['geslacht'] == 'vrouw')].value_counts()[0]
  except:
    V = 0

  M = df['geslacht'].loc[(df['partij'] == a) &
(df['geslacht'] == 'man')].value_counts()[0]
  vrouwen.append(V)
  mannen.append(M)
```

```
#DF
mvdf =
pd.DataFrame({'partijen':df['partij'].unique(),
'mannen':mannen, 'vrouwen':vrouwen})
mvdf

#BARPLOT
ax = manvrouwdf.plot(x='partijen', y='mannen',
kind = 'bar')
manvrouwdf.plot(x='partijen', y='vrouwen', kind
= 'bar', color='red', ax=ax)

#HOOGSTE PERCENTAGE MANNEN EN
VROUWEN PER PARTIJ OP DE
KANDIDATENLIJST?
manvrouwdf['percentagevrouw'] =
((manvrouwdf['vrouwen'] /
(manvrouwdf['mannen'] +
manvrouwdf['vrouwen'])) * 100).round(2)

manvrouwdf['percentageman'] =
((manvrouwdf['mannen'] /
(manvrouwdf['mannen'] +
manvrouwdf['vrouwen'])) * 100).round(2)

manvrouwdf

#PLOT DATAFRAME
ax = manvrouwdf.plot(x='partijen',
y='percentageman', kind = 'bar')
manvrouwdf.plot(x='partijen',
y='percentagevrouw', kind = 'bar', color='red',
ax=ax)

manvrouwdf

#PVV 21 zetels, Hoeveel vrouwelijk PVV in de
kamer?

1. DF met X aantal namen van kandidatenlijst
in partij
df_pvv = df.loc[(df['partij'] == 'PVV (Partij voor
de
Vrijheid)')].head(pwijzer['Zetels'].loc[(pwijzer['P
artij'] == 'PVV')].values.item())

#OF
df_ adam_pvv_20 =
df_kandidatenlijst_adam[df_kandidatenlijst_ada
m.partij == 'PVV (Partij voor de
Vrijheid)'].head(20)
print
(len(df_kandidatenlijst_adam_pvv_20[df_kandid
atenlijst_adam_pvv_20.gender == 'v']),
'vrouwen.')

2. Tel conditie in DF
len(df_pvv.loc[(df_pvv['geslacht'] == 'vrouw')])

3. plot
df_pvv['geslacht'].value_counts().plot(kind='bar'
)

#kruistabel: woonplaats x gender
cross_gender_woonplaats =
pd.crosstab(df_kandidatenlijst_adam.woonplaat
s, df_kandidatenlijst_adam.gender, margins =
True)
print cross_gender_woonplaats.sort_values(by
= 'All', ascending = False).head(5)

#Maak een horizontale staafdiagram met de
verdeling van de 20 meest populaire
woonplaatsen van alle kandidaten van de
kandidatenlijst van Amsterdam. Doe dit ook
netjes op volgorde en geef de grafiek een
passende titel.

counter =
df_kandidatenlijst_adam.woonplaats.value_cou
nts()
counter = counter[:20]
counter.plot(kind = 'barh', title = '20 meest
populaire woonplaatsen voor de kandidaten van
Amsterdam')

#Wat is de verdeling van de zetels volgens de
peilingwijzer? Neem hierbij de kolom 'zetels'
en plot dit in een staafdiagram tegen de
verschillende partijen op volgorde van de
hoeveelheid zetels met een passende titel.

df_peilingwijzer.sort_values(by='Zetels',
ascending = False).plot('Partij', 'Zetels', kind =
'bar', title = 'Verdeling Zetels volgens
Peilingwijzer')

#Vervang de NaN waarden met 0 voor alle
kolommen behalve de kolom 'Datum' in een
oneliner en laat de eerste 5 regels zien.
df_join.update(df_join[['Percentage',
u'PercentageLaag', u'PercentageHoog', u'Zetels',
u'ZetelsLaag',  u'ZetelsHoog']].fillna(0))

df_join.head(5)
```

IMPORTING DATA

pd.read_csv(filename) - From a CSV file
pd.read_table(filename) - From a delimited text file (like TSV) pd.read_excel(filename) - From an Excel file
pd.read_sql(query, connection_object) - Read from a SQL table/database pd.read_json(json_string) - Read from a JSON formatted string, URL or file. pd.read_html(url) - Parses an html URL, string or file and extracts tables to a list of dataframes pd.read_clipboard() - Takes the contents of your clipboard and passes it to read_table()
pd.DataFrame(dict) - From a dict, keys for col- umns names, values for data as lists

EXPORTING DATA

df.to_csv(filename) - Write to a CSV file
df.to_excel(filename) - Write to an Excel file
df.to_sql(table_name, connection_object) - Write to a SQL table
df.to_json(filename) - Write to a file in JSON format df.to_html(filename) - Save as an HTML table
df.to_clipboard() - Write to the clipboard

CREATE TEST OBJECTS

pd.DataFrame(np.random.rand(20,5)) - 5 col- umns and 20 rows of random floats pd.Series(my_list) - Create a series from an iterable my_list
df.index = pd.date_range('1900/1/30', periods=df.shape[0]) - Add a date index

VIEWING/INSPECTING DATA

df.head(n) - First n rows of the DataFrame df.tail(n) - Last n rows of the DataFrame df.shape() - Number of rows and columns df.info() - Index, Datatype and Memory informa- tion
df.describe() - Summary statistics for numerical columns s.value_counts(dropna=False) - View unique values and counts df.apply(pd.Series.value_counts) - Unique values and counts for all columns

SELECTION

df[col] - Return column with label col as Series df[[col1, col2]] - Return Columns as a new DataFrame s.iloc[0] - selection by position
s.loc[0] - selection by index df.iloc[0,:] - first row df.iloc[0,0] - first element of first column

DATA CLEANING

df.columns = ['a','b','c'] - Rename columns pd.isnull() - Checks for null Values, Returns Boolean Arrray pd.notnull() - Opposite of s.isnull() df.dropna() - Drop all rows that contain null values
df.dropna(axis=1) - Drop all columns that con- tain null values df.dropna(axis=1,thresh=n) - Drop all rows have have less than n non null values df.fillna(x) - Replace all null values with x s.fillna(s.mean()) - Replace all null values with the mean (mean can be replaced with almost any function from the statistics section) s.astype(float) - Convert the datatype of the series to float
s.replace(1,'one') - Replace all values equal to 1 with 'one' s.replace([1,3],['one','three']) - Replace all 1 with 'one' and 3 with 'three' df.rename(columns=lambda x: x + 1) - mass renaming of columns df.rename(columns={'old_name': 'new_ name'}) - selective renaming
df.set_index('column_one') - change the index
df.rename(index=lambda x: x + 1) - mass renaming of index

FILTER, SORT, & GROUPBY

df[df[col] > 0.5] - Rows where the col column is greater than 0.5 df[(df[col] > 0.5) & (df[col] < 0.7)] - Rows where 0.7 > col > 0.5 df.sort_values(col1) - Sort values by col1 in ascending order df.sort_values(col2,ascending=False) - Sort values by col2 in descending order
df.sort_values([col1,col2],

ascending=[True,False]) - Sort values by col1 in ascending order then col2 in descending order df.groupby(col) - Return a groupby object for values from one column
df.groupby([col1,col2]) - Return a groupby object values from multiple columns df.groupby(col1)[col2].mean() - Return the mean of the values in col2, grouped by the values in col1 (mean can be replaced with almost any function from the statistics section)
df.pivot_table(index=col1,values= [col2,col3],aggfunc=max) - Create a pivot table that groups by col1 and calculates the mean of col2 and col3

df.groupby(col1).agg(np.mean) - find the average across all columns for every unique column 1
group data.apply(np.mean) - apply a function across each column data.apply(np.max, axis=1) - apply a function across each row

JOIN/COMBINE

df1.append(df2) - Add the rows in df1 to the end of df2 (columns should be identical) df.concat([df1, df2],axis=1) - Add the columns in df1 to the end of df2 (rows should be identical) df1.join(df2,on=col1,how='inner') - SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values. how can be one of 'left', 'right', 'outer', 'inner'

STATISTICS

df.describe() - Summary statistics for numerical columns df.mean() - Return the mean of all columns
df.corr() - finds the correlation between columns in a DataFrame.
df.count() - counts the number of non-null values in each DataFrame column. df.max() - finds the highest value in each column. df.min() - finds the lowest value in each column. df.median() - finds the median of each column.
df.std() - finds the standard deviation of each column.

```
import pandas as pd
import os
from lxml import etree
import xml.etree.ElementTree as ET
%matplotlib inline
import pandas as pd
from lxml import etree
from math import sqrt
import matplotlib.pyplot as plt

peiling = pd.read_excel('Cijfers_Peilingwijzer.xlsx')
peiling.head()
results = pd.read_excel('Results_Longitudinal.xlsx')
results.head(2)
print peiling.describe()
print peiling.shape
```

#ALLE VALUES OPTELLEN

```
print peiling.sum()
```

#OPTELLEN SPECIFIEKE KOLOM

```
peiling['PercentageLaag'].sum()
```

#VERSCHIL ZETELSHOOG, ZETELSLAAG IN NIEUWE KOLOM

```
peiling['Verschil'] = (peiling['ZetelsHoog']) -
(peiling['ZetelsLaag'])
peiling.head(5)
```

#KOLOMEN UIT DATAFRAME EN INDEXCIJFERS WEG

```
peiling[['Partij', 'Datum',
'Percentage']].set_index('Partij').head(2)
```

#WELKE PARTIJ BOVEN 20 ZETELS?

```
zetels = peiling.loc[peiling['Zetels'] > 20]
zetels
```

#PLOT HOEVEEL ZETELS PER PARTIJ

```
df_peiling.sort_values(by='Zetels', ascending =
False).plot('Partij', 'Zetels', kind = 'bar', title =
'Hoeveelheid zetels per partij')
```

#GETAL UIT DATAFRAME

```
getal = peiling['Verschil'].loc[(peiling['Partij'] ==
'CDA')].values[0]
getal
```

#SORTEER DATAFRAME OP BEPAALDE KOLOM

```
peiling1 = peiling.sort_values(by='Zetels',
ascending=False)
peiling1.head(2)

len(peiling[peiling.Partij=='VVD'])
```

#HOELANG IS KOLOM 'PARTIJ'?

```
print len(peiling.Partij)
```

#HOEVEEL UNIEKE PARTIJEN ZITTEN IN PEILINGWIJZER

```
print len(set(peiling.Partij))
```

#HOEVEEL UNIEKE PARTIJEN KANDIDATENLIJST?

```
len(set(df.partij))
```

#WIE HEEFT HOOGSTE AANTAL ZETELS?

```
peiling[peiling['ZetelsHoog']== peiling.ZetelsHoog.max()]
```

#PARTIJEN SORTEREN IN DATAFRAME MET HOOGSTE PERCENTAGE

```
peiling = peiling.set_index('Partij')
percentage1 = peiling.sort_values(by='Percentage',
ascending=False)
percentage1.head(2)
```

#PLOTTEN VERSCH MANIEREN

```
percentage1['Percentage'].plot(kind='bar', title =
'Hoogste Percentage')
percentage1['Percentage'].plot(kind='pie', title =
'Hoogste Percentage')
percentage1['Percentage'].plot(title = 'Hoogste
Percentage')
```

#XML FILES

```
import xml.etree.ElementTree as ET
tree =
ET.parse('Kandidatenlijsten_TK2017_Amsterdam.eml.xml
')
root = tree.getroot()
root.tag
root.attrib
```

#ALLE TAGS UIT XML FILE HALEN

```
for tag in tree.iter():
  print tag
```

#HOEVEEL M-V ZITTEN IN KANDIDATENLIJST VAN AMSTERDAM

```
mannen = 0
vrouwen = 0
for event, elem in
ET.iterparse('Kandidatenlijsten_TK2017_Amsterdam.eml.
xml'):
  value = elem.text
  tag = elem.tag
   if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Gender" :
    if value == 'female':
      vrouwen += 1
    else:
      mannen += 1
  elem.clear()
print 'TEKST' mannen
print 'TEKST' vrouwen
print 'TEKST TOTAAL' (mannen + vrouwen)
```

#EEN DATAFRAME UIT XML FILE HALEN

```
partij = [], voornaam = [], initialen = [], achternaam = [],
geslacht = [], woonplaats = []
for event, elem in
ET.iterparse('Kandidatenlijsten_TK2017_Amsterdam.eml.
xml'):
  tag = elem.tag
  value = elem.text          (ZOEKEN IN ALLE TAGS)
  if tag ==
"{urn:oasis:names:tc:evs:schema:eml}RegisteredName":
    partijvalue = value
  if tag ==
"{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}FirstName":
    partij.append(partijvalue)
    voornaam.append(value)
  if tag ==
'{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}NameLine':
    initialen.append(value)
  if tag ==
"{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}LastName":
    achternaam.append(value)
  if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Gender":
    if value == 'male':
      geslachtvalue = 'man'
    else:
      geslachtvalue = 'vrouw'
    geslacht.append(geslachtvalue)
  if tag ==
'{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}LocalityNam
e':
    woonplaats.append(value)

df = pd.DataFrame({'partij':partij, 'voornaam':voornaam,
'initialen':initialen, 'achternaam':achternaam,
'geslacht':geslacht, 'woonplaats':woonplaats})
df.head(5)

print len(df[df.geslacht=='man'])
print len(df[df.geslacht=='vrouw'])
```

#WAT IS PERCENTAGE KANDIDATEN VAN DENK VAN TOTAAL KANDIDATEN

```
(df.partij == 'DENK').mean()*100
```

#WAT IS LIJST VAN ALLE UNIEKE PARTIJEN

```
print (df['partij']).unique()
len(set(df['partij']))
```

#HOEVEEL KANDIDATEN ZIJN ER PER PARTIJ + PLOT?

```
totaalaantalkandidaten = df['partij'].value_counts()
totaalaantalkandidaten.plot(kind='bar', title=
'Hoeveelheid kandidaten per partij')
```

#VIND ALLE MANNEN OF VROUWEN UIT DENK

```
df.loc[(df['partij'] == 'DENK') & (df['geslacht']==
'man')].head()
df.loc[(df['partij'] == 'DENK') & (df['geslacht']==
'vrouw')].head()
```

#HOEVEEL PROCENT IS MAN VAN DENK?
#TOTAAL MANNEN UIT DENK

```
mannen_denk = len(df.loc[(df['partij'] == 'DENK') &
(df['geslacht']== 'man')])
```

#TOTAAL VROUWEN UIT DENK

```
vrouwen_denk = len(df.loc[(df['partij'] == 'DENK') &
(df['geslacht']== 'vrouw')])
totaal_denk = len(df[df.partij == 'DENK'])
procentman = (mannen_denk * 100) / totaal_denk
print procentman
```

#WAT IS RELATIEVE VERHOUDING M-V? ROND AF 2 DECIMALEN.

```
np.round(vrouwen_denk/float(mannen_denk), 2)
```

#VERHOUDING M-V DENK + PLOT

```
manvrouw = df['geslacht'].loc[(df['partij'] ==
'DENK')].value_counts()
manvrouw.plot(kind='bar', title='Verhouding man vrouw
in partij Denk')
```

#VERHOUDING WOONPLAATSEN KANDIDATEN UIT DENK + PLOT

```
verhouding = df['woonplaats'].loc[(df['partij'] ==
'DENK')].value_counts()
verhouding.plot(kind='pie', title='Verdeling
woonplaatsen kandidaten DENK')
```

#KANDIDAAT UIT WOONPLAATS?

```
df.loc[(df['woonplaats'] == 'Wassenaar')]
```

#KANDIDAAT UIT WOONPLAATS EN PARTIJ?

```
df.loc[(df.woonplaats == 'Wassenaar') & (df.partij ==
'VVD')]
# of optie df.loc[(df['woonplaats'] == 'Wassenaar') &
(df['partij] == 'VVD)]
```

#LIJST KANDIDATEN MEERDERE WOONPLAATSEN?

```
df.loc[df['woonplaats'].isin(['Wassenaar', 'Assen',
'Utrecht'])].head()
df.loc[df['partij'].isin(['VVD', 'SP'])].head()
```

#GEEF EEN LIJST VAN DE PARTIJEN DIE EEN BAS ALS KANDIDAAT HEBBEN

```
df['partij'].loc[(df['voornaam'] == 'Bas')].unique()
```

#WEERGEGEVEN IN DATAFRAME

```
df.loc[(df['voornaam'] == 'Bas')]
```

#KANDIDATEN ACHTERNAAM MET W BEGINT

```
df[['voornaam', 'achternaam',
'partij']].loc[(df['achternaam'].str.startswith('W'))].head()
```

#PVV 21 ZETELS, HOEVEEL VROUWELIJKE PVVERS IN DE KAMER?
1.DATAFRAME ALLE KANDIDATEN PVV

```
df_vanpvv = df.loc[(df['partij'] == 'PVV (Partij voor de
Vrijheid)')].head(peiling['Zetels'].loc[(peiling['Partij'] ==
'PVV')].values.item())
```

2. HOEVEEL VROUWEN ZITTEN HIER IN?

```
len(df_vanpvv.loc[(df_vanpvv['geslacht'] == 'vrouw')])
```

3.PLOTTEN

```
df_vanpvv['geslacht'].value_counts().plot(kind='bar')
```

OFFFF OPTIE:

```
df_kandidatenlijst_adam_pvv_20 =
df_kandidatenlijst_adam[df_kandidatenlijst_adam.partij
== 'PVV (Partij voor de Vrijheid)'].head(20)
print
len(df_kandidatenlijst_adam_pvv_20[df_kandidatenlijst_
adam_pvv_20.gender == 'v']), 'vrouwen.'
```

#VERHOUDING M/V + PLOT

```
vrouwen_partij = []
mannen_partij = []
for a in df['partij'].unique():
  try:
    vrouw1 = df['geslacht'].loc[(df['partij'] == a) &
(df['geslacht'] == 'vrouw')].value_counts()[0]
  except:
    vrouw1 = 0

  man1 = df['geslacht'].loc[(df['partij'] == a) &
(df['geslacht'] == 'man')].value_counts()[0]
  vrouwen_partij.append(vrouw1)
  mannen_partij.append(man1)
```

#DATAFRAME MAKEN

```
mannenenvrouwen_df =
pd.DataFrame({'partijen':df['partij'].unique(),
'mannen':mannen_partij, 'vrouwen':vrouwen_partij})
mannenenvrouwen_df
```

#PLOTTEN ZELFDE GRAFIEK, TWEE KLEUREN .3

```
plotverhouding =
mannenenvrouwen_df.plot(x='partijen', y='mannen',
kind = 'bar')
mannenenvrouwen_df.plot(x='partijen', y='vrouwen',
kind = 'bar', color='red', ax=ax)
```

#HOOGSTE PERCENTAGE M-V PARTIJ?

```
mannenenvrouwen_df['percentagevanvrouw'] =
((mannenenvrouwen_df['vrouwen'] /
(mannenenvrouwen_df['mannen'] +
mannenenvrouwen_df['vrouwen'])) * 100).round(2)
mannenenvrouwen_df['percentagevanman'] =
((mannenenvrouwen_df['mannen'] /
(mannenenvrouwen_df['mannen'] +
mannenenvrouwen_df['vrouwen'])) * 100).round(2)
mannenenvrouwen_df
```

#PLOT DEZE PERCENTAGES IN EEN DATAFRAME

Zelfde als .3 hierboven alleen dan met y = 'percentagevanman' en y = 'percentagevanvrouw'

 # DOOR ALLE LIJSTEN LOOPEN NAAR ALLE MANNEN EN VROUWEN (DUS DUBBELE)

```
totaalmannen = 0
totaalvrouwen = 0

path = 'mappie/'
#print os.listdir(path)
for _file in os.listdir(path):
  if not _file.endswith('xml'):continue
#   print _file
  aantalmannen = 0
  aantalvrouwen = 0
  with open(path+_file):
    for event, elem in ET.iterparse(path +_file):
      tag = elem.tag
      value = elem.text
      if tag ==
"{urn:oasis:names:tc:evs:schema:eml}Gender" :
        if value == 'male':
          aantalmannen += 1
          totaalmannen  += 1
        else:
          aantalvrouwen += 1

      elem.clear() # discard the element
  print _file.strip('.eml.xml'), aantalmannen,
aantalvrouwen, (aantalmannen + aantalvrouwen)
```

#20 POPULAIRE WOONPLAATSEN

```
woonplaats = df.woonplaats.value_counts()
woonplaats = woonplaats[:20]
woonplaats.plot(kind='barh', title = 'blablalb')
(barh = horizontaal)
```

#MAAK KRUISTABEL WAARIN JE GENDER TEGEN DE WOONPLAATS PLAATST

```
cross = pd.crosstab(df.woonplaats, df.gender, margins =
True)
print cross.sort_values(by= 'All', ascending =
False).head()
```

#NaN WAARDEN OPLOSSEN

```
df_join.update(df_join[['Percentage', u'PercentageLaag',
u'PercentageHoog', u'Zetels',  u'ZetelsLaag',
u'ZetelsHoog']].fillna(0))
df_join.head(5)
```

VIEWING/INSPECTING DATA

df.info() - Index, Datatype and Memory information
df.describe() - Summary statistics for numerical columns
s.value_counts(dropna=False) - View unique values and counts **df.apply(pd.Series.value_counts)** - Unique values and counts for all columns

SELECTION

df[col] - Return column with label col as Series
df[[col1, col2]] - Return Columns as a new DataFrame
s.iloc[0] - selection by position
s.loc[0] - selection by index df.iloc[0,:] - first row
df.iloc[0,0] - first element of first column

DATA CLEANING

df.columns = ['a','b','c'] - Rename columns
pd.isnull() - Checks for null Values, Returns Boolean Arrray
pd.notnull() - Opposite of s.isnull() df.dropna() - Drop all rows that contain null values
df.dropna(axis=1) - Drop all columns that contain null values
df.dropna(axis=1,thresh=n) - Drop all rows have have less than n non null values **df.fillna(x)** - Replace all null values with x
s.fillna(s.mean()) - Replace all null values with the mean (mean can be replaced with almost any function from the statistics section)
s.astype(float) - Convert the datatype of the series to float
s.replace(1,'one') - Replace all values equal to 1 with 'one'
s.replace([1,3],['one','three']) - Replace all 1 with 'one' and 3 with 'three' **df.rename(columns=lambda x: x + 1)** - mass renaming of columns **df.rename(columns={'old_name': 'new_name'})** - selective renaming **df.set_index('column_one')** - change the index df.rename(index=lambda x: x + 1) - mass renaming of index

FILTER, SORT, & GROUPBY

df[df[col] > 0.5] - Rows where the col column is greater than 0.5
df[(df[col] > 0.5) & (df[col] < 0.7)] - Rows where 0.7 > col > 0.5
df.sort_values(col1) - Sort values by col1 in ascending order
df.sort_values(col2,ascending=False) - Sort values by col2 in descending order **df.sort_values([col1,col2], ascending=[True,False])** - Sort values by col1 in ascending order then col2 in descending order
df.groupby(col) - Return a groupby object for values from one column **df.groupby([col1,col2])** - Return a groupby object values from multiple columns **df.groupby(col1)[col2].mean()** - Return the mean of the values in col2, grouped by the values in col1 (mean can be replaced with almost any function from the statistics section)
df.pivot_table(index=col1,values= [col2,col3],aggfunc=max) - Create a pivot table that groups by col1 and calculates the mean of col2 and col3
df.groupby(col1).agg(np.mean) - find the average across all columns for every unique column 1 group
data.apply(np.mean) - apply a function across each column
data.apply(np.max, axis=1) - apply a function across each row

JOIN/COMBINE

df1.append(df2) - Add the rows in df1 to the end of df2 (columns should be identical)
df.concat([df1, df2],axis=1) - Add the columns in df1 to the end of df2 (rows should be identical)
df1.join(df2,on=col1,how='inner') - SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values. how can be one of 'left', 'right', 'outer', 'inner'

STATISTICS

df.describe() - Summary statistics for numerical columns
df.mean() - Return the mean of all columns
df.corr() - finds the correlation between columns in a DataFrame.
df.count() - counts the number of non-null values in each DataFrame column.
df.max() - finds the highest value in each column.
df.min() - finds the lowest value in each column.
df.median() - finds the median of each column.
df.std() - finds the standard deviation of each column.

```
# LIJST AAN KANDIDATEN INLEZEN XML
import bs4 as bs
bestand = open('Kandidatenlijsten_TK2017_Amsterdam.eml2.xml').read()
soup = bs.BeautifulSoup(bestand, 'xml')

# NAMEN VAN DE PARTIJEN
partynames = []
for x in soup.findAll('RegisteredName'):
    partynames.append(x.text)
print partynames

# PLOT HOEVEEL MENSEN ANNE HETEN PER PARTIJ
namendict = {}
for partij in soup('Affiliation'):
    namendict[partij.find('RegisteredName').text] = \
    [name.text for name in partij.findAll('FirstName')].count('Anne')
y = pd.DataFrame.from_dict(namendict, orient='index')
y.plot(kind='barh')

# DICT PARTIJNAAM: [male, female, male]
manvrouwdict = {}
for p in soup('Affiliation'):
    manvrouwdict[p.find('RegisteredName').text] = [g.text for g in p.findAll('Gender')]

# Stel dat de PVV 25 zetels haalt, hoeveel vrouwelijke PVVers komen er dan in de Kamer?
from collections import Counter
print Counter(manvrouwdict['PVV (Partij voor de Vrijheid)'][:25])['female']

# PLOT VROUWEN VAN PVV IN DE KAMER MET N GEWONNEN ZETELS
vrouwenperzetel = {p: {N: Counter(manvrouwdict[p][:N])['female'] \
                  for N in range(50)} for p in manvrouwdict}
pd.DataFrame.from_dict(vrouwenperzetel['PVV (Partij voor de Vrijheid)'], orient='index')\
    .plot(kind='bar',title='vrouwen van PVV in de kamer met N aantal zetels');

# DATAFRAME AANTAL MENSEN OP DE LIJST, PERCENTAGE MAN/VROUW
vrouwen = {}
mannen = {}
for x in manvrouwdict:
    manofvrouw = manvrouwdict[x]
    vrouwen[x] = manofvrouw.count('female')
    mannen[x] = manofvrouw.count('male')
df = pd.DataFrame.from_dict(vrouwen, orient='index')
df2 = pd.DataFrame.from_dict(mannen, orient='index')
df.columns = ['vrouwen']
df2.columns = ['mannen']
new = pd.concat([df, df2],axis=1)
new['vrouwenperc'] = new.vrouwen/(new.vrouwen+new.mannen)*100
new['mannenperc'] = new.mannen/(new.vrouwen+new.mannen)*100
new.sort_values('vrouwenperc',ascending=False)

# PLOT VROUWENPERCENTAGES PER PARTIJ
new['vrouwenperc'].sort_values().plot(kind='barh')

# MINIMAAL AANTAL MENSEN VAN PARTIJ
min_lijst_lengte = (new.filter(['vrouwen','mannen']).sum(axis=1)).min()
print min_lijst_lengte
# LEES PEILINGWIJZER IN, VERANDER NAMEN OM SAMEN TE VOEGEN
peilingen = pd.read_excel('Cijfers_Peilingwijzer.xlsx')
translate={u'50PLUS':u'50PLUS',
           u'CDA':u'CDA',
           u'CU':u'ChristenUnie',
           u'D66':u'Democraten 66 (D66)',
           u'Denk': u'DENK',
           u'FvD':u'Forum voor Democratie',
           u'GL': u'GROENLINKS',
           u'PVV':u'PVV (Partij voor de Vrijheid)',
           u'PvdA':u'Partij van de Arbeid (P.v.d.A.)',
           u'PvdD':u'Partij voor de Dieren',
           u'SGP':u'Staatkundig Gereformeerde Partij (SGP)',
           u'SP':u'SP (Socialistische Partij)',
           u'VNL':u'VNL (VoorNederland)',
           u'VVD':u'VVD',
           u'PP':u'Piratenpartij'}
peilingen.index = peilingen.Partij
peilingen.rename(translate, inplace=True)
del peilingen['Partij']

# JOIN DE TWEE DATAFRAMES
peilingen.join(new)

# PEILINGWIJZER DOOR DE TIJD OPGESCHOOND
allepeilingen = pd.read_excel('Results_Longitudinal.xlsx')
x = allepeilingen.T.iloc[::3]
x.rename(translate,inplace= True)
x.T.tail(3)

# ANDERE OPTIE
zetelstijd = pd.read_csv('https://dlbjgq97if6urz.cloudfront.net/Public/Peilingwijzer/Last/Results_Dy
                         ,index_col='Date')
# check if all add to 150
print (zetelstijd.sum(axis=1) == 150).mean()
# pak alleen de middelste
zetelstijd = zetelstijd.applymap(lambda s: int(s.split(';')[1]))

# WEER NAMEN AANPASSEN
zetelstijd = zetelstijd.T
zetelstijd.rename(translate, inplace=True)
zetelstijd = zetelstijd.T
zetelstijd.tail()

# ALLEEN VROUWEN IN DE KAMER DOOR DE TIJD
for p in zetelstijd.columns:
    zetelstijd[p] = zetelstijd[p].apply(lambda z: vrouwenperzetel[p][z])

# PLOT VROUWEN IN DE KAMER PER PARTIJ DOOR DE TIJD
zetelstijd.plot(figsize=(20,10),title='Aantal vrouwen in de Tweede Kamer door de tijd, per partij');

# TOTAAL AANTAL VROUWEN IN DE KAMER DOOR DE TIJD
zetelstijd.T.sum().plot(figsize=(18,7),title='Aantal vrouwen in de Tweede Kamer door de tijd');

# VERDELING ZETELS VOLGENS PEILINGWIJZER
df_peilingwijzer.sort_values(by='Zetels', ascending = False)\
            .plot('Partij', 'Zetels', kind = 'bar', title = 'Verdeling Zetels volgens
```

```
from __future__ import division
# HOEVEEL PROCENT VAN DE STEMMINGEN ZIJN AANGENOMEN?
hoofdelijke_stemming_df = rutte2[rutte2['votetype'] == 'Roll call']
print len(hoofdelijke_stemming_df[hoofdelijke_stemming_df['result'] == 'adopted'])\
                                    /len(hoofdelijke_stemming_df) * 100
# HOE VAAK STEMDE BARRY MEE?
barry_stemt_voor = hoofdelijke_stemming_df['pro'].str.contains('Barry Madlener')
print 'Barry stemde', len(hoofdelijke_stemming_df[barry_stemt_voor]), 'keer voor'

# KRUISTABEL
pd.crosstab(rutte2['proposaltype'], rutte2['result'], margins=True)

# HOEVEEL STUKKEN ELKE PARTIJ INGEDIEND
im = rutte2[(rutte2['authorparty'] != '[]') \
          & (rutte2['authorparty'] == rutte2['supporterparties'])]
per_party_counts = im['authorparty'].str[1:-1].value_counts()
per_party_counts.sort_values().plot.barh()

print per_party_counts.idxmax()

# PLOT PER PARTIJ HOEVEEL PROCENT VD STEMMINGEN PARTIJ VOOR HEEFT GESTEMD
parties = rutte2.columns[11:50]
vote_distribution = rutte2[parties].apply(pd.value_counts)
vote_distribution = vote_distribution.T
vote_distribution.dropna(inplace=True)
vote_distribution.columns = ['contra', 'pro']
vote_distribution['pro_percentage'] = (vote_distribution['pro'] \
                        / vote_distribution.sum(axis='columns')) * 100
percentage_sorted = vote_distribution\
                        .sort_values(by='pro_percentage', ascending=False)
percentage_sorted.plot.bar(y='pro_percentage', ylim=(0, 100), \
                        title='Percentage voor-stemmingen per partij')

# PARTIJEN DIE MINSTENS 1 KEER GESTEMD HEBBEN, HOE VAAK?
im2 = rutte2[rutte2['votetype'] == 'Normal']
party_vote_counts = im2[parties].count()
sorted_vote_counts = party_vote_counts[party_vote_counts > 0]\
                        .sort_values(ascending=False)

# WANNEER ONTSTOND DENK?
denk_index = parties[18]
denk_is_nan = np.isnan(rutte2[denk_index])
# All items where DENK voted
denk_votes = rutte2[~denk_is_nan]
denk_votes['date'].min()

# HOEVAAK STEMMEN PARTIJEN MEE MET PVV?
pvv_voted_pro = rutte2[rutte2['PVV'] == 1]
other_parties = pvv_voted_pro[parties].dropna(axis='columns')
other_party_votes = other_parties.sum()
relative_votes = other_party_votes / len(pvv_voted_pro)
relative_votes.sort_values(ascending=False)
```

FILTER, SORT, & GROUPBY

`df[df[col] > 0.5]` - Rows where the `col` column is greater than `0.5`

`df[(df[col] > 0.5) & (df[col] < 0.7)]` - Rows where `0.7 > col > 0.5`

`df.sort_values(col1)` - Sort values by `col1` in ascending order

`df.sort_values(col2,ascending=False)` - Sort values by `col2` in descending order

`df.sort_values([col1,col2],`

`ascending=[True,False])` - Sort values by `col1` in ascending order then `col2` in descending order

`df.groupby(col)` - Return a groupby object for values from one column

`df.groupby([col1,col2])` - Return a groupby object values from multiple columns

`df.groupby(col1)[col2].mean()` - Return the mean of the values in `col2`, grouped by the values in `col1` (mean can be replaced with almost any function from the statistics section)

`df.pivot_table(index=col1,values=[col2,col3],aggfunc=max)` - Create a pivot table that groups by `col1` and calculates the mean of `col2` and `col3`

`df.groupby(col1).agg(np.mean)` - find the average across all columns for every unique column 1 group

`data.apply(np.mean)` - apply a function across each column

`data.apply(np.max, axis=1)` - apply a function across each row

JOIN/COMBINE

`df1.append(df2)` - Add the rows in `df1` to the end of `df2` (columns should be identical)

`df.concat([df1, df2],axis=1)` - Add the columns in `df1` to the end of `df2` (rows should be identical)

`df1.join(df2,on=col1,how='inner')` - SQL-style join the columns in `df1` with the columns on `df2` where the rows for `col` have identical values. `how` can be one of `'left'`, `'right'`, `'outer'`, `'inner'`

STATISTICS

These can all be applied to a series as well.

`df.describe()` - Summary statistics for numerical columns

`df.mean()` - Return the mean of all columns

`df.corr()` - finds the correlation between columns in a DataFrame.

`df.count()` - counts the number of non-null values in each DataFrame column.

`df.max()` - finds the highest value in each column

VIEWING/INSPECTING DATA

`df.head(n)` - First `n` rows of the DataFrame

`df.tail(n)` - Last `n` rows of the DataFrame

`df.shape()` - Number of rows and columns

`df.info()` - Index, Datatype and Memory information

`df.describe()` - Summary statistics for numerical columns

`s.value_counts(dropna=False)` - View unique values and counts

`df.apply(pd.Series.value_counts)` - Unique values and counts for all columns

SELECTION

`df[col]` - Return column with label `col` as Series

`df[[col1, col2]]` - Return Columns as a new DataFrame

`s.iloc[0]` - selection by position

`s.loc[0]` - selection by index

`df.iloc[0,:]` - first row

`df.iloc[0,0]` - first element of first column

DATA CLEANING

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

`pd.isnull()` - Checks for null Values, Returns Boolean Arrray

`pd.notnull()` - Opposite of `s.isnull()`

`df.dropna()` - Drop all rows that contain null values

`df.dropna(axis=1)` - Drop all columns that contain null values

`df.dropna(axis=1,thresh=n)` - Drop all rows have have less than `n` non null values

`df.fillna(x)` - Replace all null values with `x`

`s.fillna(s.mean())` - Replace all null values with the mean (mean can be replaced with almost any function from the statistics section)

`s.astype(float)` - Convert the datatype of the series to float

`s.replace(1,'one')` - Replace all values equal to `1` with `'one'`

`s.replace([1,3],['one','three'])` - Replace all `1` with `'one'` and `3` with `'three'`

`df.rename(columns=lambda x: x + 1)` - mass renaming of columns

`df.rename(columns={'old_name': 'new_name'})` - selective renaming

`df.set_index('column_one')` - change the index

`df.rename(index=lambda x: x + 1)` - mass renaming of index

bestand = open(filename).read()
soup = bs.BeautifulSoup(bestand,'lxml')

Data Science Cheat Sheet

Pandas

x = {}
for item in soup.findAll('aff.'):
x[item.find('regn.').text =
[t.text for t in item.findAll('gend')]
print Counter(x['VVD'][:25])

KEY

We'll use shorthand in this cheat sheet

`df` - A pandas DataFrame object

`s` - A pandas Series object

IMPORTS

Import these to start

```
import pandas as pd
import numpy as np
```

IMPORTING DATA

`pd.read_csv(filename)` - From a CSV file

`pd.read_table(filename)` - From a delimited text file (like TSV)

`pd.read_excel(filename)` - From an Excel file

`pd.read_sql(query, connection_object)` - Read from a SQL table/database

`pd.read_json(json_string)` - Read from a JSON formatted string, URL or file.

`pd.read_html(url)` - Parses an html URL, string or file and extracts tables to a list of dataframes

`pd.read_clipboard()` - Takes the contents of your clipboard and passes it to `read_table()`

`pd.DataFrame(dict)` - From a dict, keys for columns names, values for data as lists

EXPORTING DATA

`df.to_csv(filename)` - Write to a CSV file

`df.to_excel(filename)` - Write to an Excel file

`df.to_sql(table_name, connection_object)` - Write to a SQL table

`df.to_json(filename)` - Write to a file in JSON format

`df.to_html(filename)` - Save as an HTML table

`df.to_clipboard()` - Write to the clipboard

CREATE TEST OBJECTS

Useful for testing

`pd.DataFrame(np.random.rand(20,5))` - 5 columns and 20 rows of random floats

`pd.Series(my_list)` - Create a series from an iterable `my_list`

`df.index = pd.date_range('1900/1/30', periods=df.shape[0])` - Add a date index

VIEWING/INSPECTING DATA

`df.head(n)` - First `n` rows of the DataFrame

`df.tail(n)` - Last `n` rows of the DataFrame

`df.shape()` - Number of rows and columns

`df.info()` - Index, Datatype and Memory information

`df.describe()` - Summary statistics for numerical columns

`s.value_counts(dropna=False)` - View unique values and counts

`df.apply(pd.Series.value_counts)` - Unique values and counts for all columns

SELECTION

`df[col]` - Return column with label `col` as Series

`df[[col1, col2]]` - Return Columns as a new DataFrame

`s.iloc[0]` - selection by position

`s.loc[0]` - selection by index

`df.iloc[0,:]` - first row

`df.iloc[0,0]` - first element of first column

DATA CLEANING

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

`pd.isnull()` - Checks for null Values, Returns Boolean Arrray

`pd.notnull()` - Opposite of `s.isnull()`

`df.dropna()` - Drop all rows that contain null values

`df.dropna(axis=1)` - Drop all columns that contain null values

`df.dropna(axis=1,thresh=n)` - Drop all rows have have less than `n` non null values

`df.fillna(x)` - Replace all null values with `x`

`s.fillna(s.mean())` - Replace all null values with the mean (mean can be replaced with almost any function from the statistics section)

`s.astype(float)` - Convert the datatype of the series to float

`s.replace(1,'one')` - Replace all values equal to `1` with `'one'`

`s.replace([1,3],['one','three'])` - Replace all `1` with `'one'` and `3` with `'three'`

`df.rename(columns=lambda x: x + 1)` - mass renaming of columns

`df.rename(columns={'old_name': 'new_name'})` - selective renaming

`df.set_index('column_one')` - change the index

`df.rename(index=lambda x: x + 1)` - mass renaming of index

FILTER, SORT, & GROUPBY

`df[df[col] > 0.5]` - Rows where the `col` column is greater than `0.5`

`df[(df[col] > 0.5) & (df[col] < 0.7)]` - Rows where `0.7 > col > 0.5`

`df.sort_values(col1)` - Sort values by `col1` in ascending order

`df.sort_values(col2,ascending=False)` - Sort values by `col2` in descending order

`df.sort_values([col1,col2], ascending=[True,False])` - Sort values by `col1` in ascending order then `col2` in descending order

`df.groupby(col)` - Return a groupby object for values from one column

`df.groupby([col1,col2])` - Return a groupby object values from multiple columns

`df.groupby(col1)[col2].mean()` - Return the mean of the values in `col2`, grouped by the values in `col1` (mean can be replaced with almost any function from the statistics section)

`df.pivot_table(index=col1,values=[col2,col3],aggfunc=max)` - Create a pivot table that groups by `col1` and calculates the mean of `col2` and `col3`

`df.groupby(col1).agg(np.mean)` - find the average across all columns for every unique column 1 group

`data.apply(np.mean)` - apply a function across each column

`data.apply(np.max, axis=1)` - apply a function across each row

JOIN/COMBINE

`df1.append(df2)` - Add the rows in `df1` to the end of `df2` (columns should be identical)

`df.concat([df1, df2],axis=1)` - Add the columns in `df1` to the end of `df2` (rows should be identical)

`df1.join(df2,on=col1,how='inner')` - SQL-style join the columns in `df1` with the columns on `df2` where the rows for `col` have identical values. `how` can be one of `'left'`, `'right'`, `'outer'`, `'inner'`

STATISTICS

These can all be applied to a series as well.

`df.describe()` - Summary statistics for numerical columns

`df.mean()` - Return the mean of all columns

`df.corr()` - finds the correlation between columns in a DataFrame.

`df.count()` - counts the number of non-null values in each DataFrame column.

`df.max()` - finds the highest value in each column.

`df.min()` - finds the lowest value in each column.

`df.median()` - finds the median of each column.

`df.std()` - finds the standard deviation of each column.

LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io

__future__ division
collections Counter
bs4 as bs

.value_counts()
.plot.barh()
pd.crosstab(col1,col2,margins=True)

import seaborn as sn
%matplotlib inline
From '' import pyplot as plt
plt.figure(figsize=(x,y))
plot.bar(y=legend, ylim=(0,100), title=

```
#parse xml
tree = lxml.etree.parse("Kandidatenlijsten_TK2017_Amsterdam.eml.xml")

#haal alle elementen uit xml
lst = []
candidates = tree.findall('//{urn:oasis:names:tc:evs:schema:eml}Candidate')
affil = tree.findall('//{urn:oasis:names:tc:evs:schema:eml}Affiliation')
for a in affil:
    candidates = a.findall("{urn:oasis:names:tc:evs:schema:eml}Candidate")
    for c in candidates:
        dct = {}
        dct["partij"] = a.find('{urn:oasis:names:tc:evs:schema:eml}AffiliationIdentifier//{urn:oasis:names:tc:evs:schema:eml}RegisteredName').text
        dct["initials"] = c.find("{urn:oasis:names:tc:evs:schema:eml}CandidateFullName//{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}NameLine").text
        dct["voornaam"] = c.find("{urn:oasis:names:tc:evs:schema:eml}CandidateFullName//{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}FirstName").text
        dct["gender"] = c.find('{urn:oasis:names:tc:evs:schema:eml}Gender').text
        dct["achternaam"] = c.find("{urn:oasis:names:tc:evs:schema:eml}CandidateFullName//{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}LastName").text
        dct["id"] = c.find('{urn:oasis:names:tc:evs:schema:eml}CandidateIdentifier').values()
        dct["woonplaats"] = c.find('{urn:oasis:names:tc:evs:schema:eml}QualifyingAddress//{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}Locality//{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}LocalityName').text
        lst.append(dct)

#iter on tags from tree
[x.tag for x in tree.getiterator()]

#maak dataframe van lijst met dictionary, maak van id een nummer
import numpy as np
df = pd.DataFrame.from_dict(lst)
df['id'] = df['id'].str.get(0)
df.id = pd.to_numeric(df.id, errors='coerce').fillna(0).astype(np.int64)
print df.shape
df.head()
#df.partij.value_counts() #aantal kandidaten lijst per partij

#neem specifieke partij
cda = df[df.partij == 'CDA']

#aantal unieke partijen
len(df.partij.unique())

#kruistabel van alle partijen en hun gender verdeling
geslacht = pd.crosstab(df.partij, df.gender, margins= True)
geslacht.head()

'''Pak het aantal zetels uit de peilingen
dataset voor d66 en kijk hoeveel mannen en
vrouwen er uiteindelijk in de kamer komen voor PVV'''

pvv = df.loc[(df.partij == 'PVV (Partij voor de Vrijheid)')].head(peil.Zetels.loc[(peil.Partij == 'PVV')].values.item())
pvv.gender.value_counts()

vr = df2[df2['partij'] == "PVV (Partij voor de Vrijheid)"]
vr = vr[vr['nummer'] <26]
vr.gender.value_counts()

#read peilingwijzer
peil = pd.read_excel('peilingwijzer.xlsx')
print peil.shape

# man/vrouw verhouding met ratio
geslacht["ratio"] = geslacht["male"] / geslacht["female"]
#ratio = geslacht.apply(np.log2)
#ratio.loc[(ratio["ratio"] > -0.5) & (ratio["ratio"] < 0.5)].sort_values(ascending=False, by="All").head()
geslacht.head().sort_values(ascending=False, by="ratio")
#ratio

import pandas as pd
import numpy as np
import seaborn as sn
import lxml
from lxml import etree
%matplotlib inline

# man en vrouw verhouding, stacked graph kieslijsten
new = pd.crosstab(df.partij, df.gender).sort_values(['female'],
                                                    ascending=[True]).plot(kind='bar',
                                                                           figsize=(5, 5), stacked=True)

kind : str
'line' : line plot (default)
'bar' : vertical bar plot
'barh' : horizontal bar plot
'hist' : histogram
'box' : boxplot
'kde' : Kernel Density Estimation plot
'density' : same as 'kde'
'area' : area plot
'pie' : pie plot
'scatter' : scatter plot
'hexbin' : hexbin plot

#percentage man en vrouw
geslacht["avg_F"] = (geslacht["female"] / geslacht["All"])*100
geslacht["avg_M"] = (geslacht["male"] / geslacht["All"])*100
geslacht.sort_values(['avg_M'], ascending=[False]).head(10).round(2)

#aantal procent man en aantal procent vrouw gehele kieslijst
geslacht2 = pd.crosstab(df.partij, df.gender, margins=True)
total = np.sum(geslacht2.ix[['PVV (Partij voor de Vrijheid)'],['male', 'female']].values)
(geslacht2.ix[['PVV (Partij voor de Vrijheid)'],['male', 'female']].sum(axis=0)/total * 100).plot(kind='pie')

# partijleden van de vvd die in rotterdam of amsterdam wonen
vvd_ams_rot = df.loc[(df.partij == 'VVD') & ((df.woonplaats == 'Rotterdam') | (df.woonplaats == 'Amsterdam'))]

Meerdere files
for _file in os.listdir(path):
    if not _file.endswith('xml'):continue
    with open(path+ file):
        tree = ET.parse(path+_file)

ams = "Kandidatenlijsten_TK2017_Amsterdam.eml.xml"

tree = ET.parse(ams)
root = tree.getroot()
data = []

for item in tree.iter():
    if item.tag == '{urn:oasis:names:tc:evs:schema:eml}Affiliation':
        partij = item.getchildren()[0][0].text
        for candidate in item.findall("{urn:oasis:names:tc:evs:schema:eml}Candidate"):
            id = candidate.getchildren()[0].attrib["Id"]
            firstname = candidate.getchildren()[1][0][1].text
            lastname = candidate.getchildren()[1][0][2].text
            initialen = candidate.getchildren()[1][0][0].text
            gender = candidate.getchildren()[2].text
            place = candidate.getchildren()[3][0][0].text

            data.append({"initialen":initialen, "Partij":partij, "id":id, "naam": firstname, "achternaam": lastname, "geslacht": gender, "woonplaats":place})

(2)

Min/Max/Aantal vrouwen/mannen
for item in compleet.iterrows():
vrouwenlaag.append(item[1]["geslacht"][:item[1]["ZetelsLaag"]].count("v"))
compleet["vrouwenlaag"] = vrouwenlaag etc…

import xml.etree.ElementTree as ET

counter = df.woonplaats.value_counts() #Value_counts sorteert ook meteen voor je
counter = counter[:20]
counter.plot(kind = 'barh', title = '20 meest populaire woonplaatsen voor de kandidaten van Amsterdam')

#TEL ALLE VALUES IN EEN KOLOM OP
print peil['Zetels'].sum()

#BEREKEN HET VERSCHIL VAN TWEE KOLOMMEN EN VOEG DIT TOE IN EEN EXTRA KOL
peil['VerschilHoogLaag'] = (peil['ZetelsHoog'] - peil['ZetelsLaag'])
peil.head()

#PLOT DE ZETELVERDELING VOLGENS DE PEILINGWIJZER
peil[['Partij', 'Zetels']].plot(x ='Partij', y= 'Zetels', kind='barh', title = 'Zetel verdeling')

#Crosstab Partij en Vrouwen hele kieslijst
vrouwen = df[df['gender'] == 'female']
vrouwen = pd.crosstab(vrouwen.partij, vrouwen.gender).sort_values(['female'], ascending=True).plot(kind='barh', figsize=(10, 10), stacked=True, title='Man/Vrouw')

#HOEVEEL % VAN TOTAAL KANDIDATEN STAAT OP DE LIJST VOOR SPECIFIEKE PARTIJ?
(df.partij == 'VVD').mean()*100

#VIND ALLE KANDIDATEN DIE IN EEN VAN DE VOLGENDE STEDEN WONEN: BREDA, TILBURG, S-HERTOGENBOSCH, EINDHOVEN EN PL
df_woonplaats = df['woonplaats'].loc[df['woonplaats'].isin(['Breda', 'Tilburg', 'Eindhoven'])]
df_woonplaats.value_counts().plot(kind = 'bar')

#s-HERTOGENBOSCH WORDT NIET GEVONDEN DOOR HET STREEPJE (HYPHEN) DUS MOET JE ZO ZOEKEN ALS JE DEZE WILT VINDEN
df['woonplaats'].loc[df['woonplaats'].str.contains('Hertogenbosch')]

#GEEF EEN LIJST VAN DE PARTIJEN DIE VROUWEN IN HUN KANDIDATEN LIJST HEBBEN
df['partij'].loc[(df['geslacht'] == 'vrouw')].unique()

#VINDEN ALLE KANDIDATEN WAARVAN HUN VOORNAAM MET EEN B BEGINT WONEN IN ADAM OF DIEMEN EN GEEF VOORNAAM EN PARTIJ WOONPLAATS
df[['voornaam', 'partij', 'woonplaats']].loc[(df['voornaam'].str.startswith('B')) & df['woonplaats'].isin(['Amsterdam', 'Diemen'])]

#pivot table op basis van gender
piv = pd.pivot_table(df, index=["partij","gender"],aggfunc='count').dropna()
piv =piv.drop(['achternaam','initials','voornaam','woonplaats'],axis=1).rename(columns={'id':'aantal'})

(3)
genders = kandidaten.groupby('Partij')['gender'].sum()
dfg = genders.to_frame()
dfg = dfg.reset_index()

compleet = peilingwijzer.merge(dfg, on="Partij")

(5)
%matplotlib inline
ding = compleet[["Partij", "mannen", "vrouwen"]]
ding[(ding.mannen > 0) | (ding.vrouwen > 0)]

(1)
kandidaten.gender.replace({"male": "m", "female": "v"}, inplace=True)

(2)
peilingwijzer = pd.read_excel("peilingwijzer.xlsx")

#partijnamen goed zetten
peilingwijzer.Partij[2] = 'Partij van de Arbeid (P.v.d.A.)'
peilingwijzer.Partij[3] = 'PVV (Partij voor de Vrijheid)'
peilingwijzer.Partij[4] = 'SP (Socialistische Partij)'
peilingwijzer.Partij[6] = 'Democraten 66 (D66)'
peilingwijzer.Partij[7] = 'ChristenUnie'
peilingwijzer.Partij[8] = 'GROENLINKS'
peilingwijzer.Partij[9] = 'Staatkundig Gereformeerde Partij (SGP)'
peilingwijzer.Partij[10] = 'Partij voor de Dieren'
peilingwijzer.Partij[12] = 'VNL (VoorNederland)'
peilingwijzer.Partij[13] = 'DENK'
peilingwijzer.Partij[14] = 'Forum voor Democratie'
peilingwijzer.Partij[15] = 'Piratenpartij'

peilingwijzer

(4)
vrouwenlaag = []
mannenlaag = []

vrouwenhoog = []
mannenhoog= []

vrouwen = []
mannen = []

for item in compleet.iterrows():
    vrouwenlaag.append(item[1]["gender"][:item[1]["ZetelsLaag"]].count("v"))
    vrouwenhoog.append(item[1]["gender"][:item[1]["ZetelsHoog"]].count("v"))
    mannenlaag.append(item[1]["gender"][:item[1]["ZetelsLaag"]].count("m"))
    mannenhoog.append(item[1]["gender"][:item[1]["ZetelsHoog"]].count("m"))
    vrouwen.append(item[1]["gender"][:item[1]["Zetels"]].count("v"))
    mannen.append(item[1]["gender"][:item[1]["Zetels"]].count("m"))

compleet["vrouwenlaag"] = vrouwenlaag
compleet["vrouwenhoog"] = vrouwenhoog
compleet["mannenlaag"] = mannenlaag
compleet["mannenhoog"] = mannenhoog
compleet["mannen"] = mannen
compleet["vrouwen"] = vrouwen

RATIO + TOTAAL
df1["total"] =(df1["m"]+df1["v"])
df1["ratio"] =df1["m"]/(df1["m"]+df1["v"])

Groupby
g = amsterdam.groupby(["partij", "gender"])
g.size()
```

```
import pandas as pd
import os
from lxml import etree
import xml.etree.ElementTree as ET

XML naar DataFrame
ams = "Kandidatenlijsten_TK2017_Amsterdam.eml.xml"
tree = ET.parse(ams)
root = tree.getroot()
data = []
for item in tree.iter():
    if item.tag == '{urn:oasis:names:tc:evs:schema:eml}Affiliation':
        partij = item.getchildren()[0][0].text
        for candidate in item.findall("{urn:oasis:names:tc:evs:schema:eml}Candidate"):
            nummer = int(candidate.getchildren()[0].attrib["Id"])
            firstname = candidate.getchildren()[1][0][1].text
            lastname = candidate.getchildren()[1][0][2].text
            initialen = candidate.getchildren()[1][0][0].text
            gender = candidate.getchildren()[2].text
            place = candidate.getchildren()[3][0][0].text

            data.append({"initialen":initialen, "partij":partij, "nummer":nummer, "naam": firstname, "achternaam": lastname,
"geslacht": gender, "woonplaats":place})

kandidaten = pd.DataFrame(data)
kandidaten.geslacht.replace({"male": "m", "female": "v"}, inplace=True)

Aantal vrouwen/mannen per partij kandidatenlijst
%matplotlib inline
hi= kandidaten.pivot_table(index=["partij","geslacht"], aggfunc='count', values='woonplaats').dropna()
#hi=hi.drop(['naam','initialen','achternaam','woonplaats'],axis=1).rename(columns={'nummer':'aantal'})
hi.unstack("geslacht").sort_values('v').plot(kind="barh", figsize=(20,10))

Ratio man/vrouw
df = hi.unstack("geslacht")
df["ratio"] = df["aantal","v"]/(df["aantal","m"]+df["aantal","v"])*100
df.fillna(0).sort_values("ratio")["aantal"].plot(kind="barh", figsize=(20,10))

Aantal zetels per partij peilingenwijzer 2017
peilingenwijzer=peilingwijzer[["Partij","Zetels"]].set_index("Partij").sort_values('Zetels').plot(kind='barh')

Geslacht op volgorde per partij mergen met pijlingenwijzer en plotten
genders = kandidaten.groupby("Partij")["geslacht"].sum()
dfg = genders.to_frame()
dfg = dfg.reset_index()
compleet = peilingwijzer.merge(dfg, on="Partij")
vrouwenlaag = []
vrouwenhoog = []
vrouwen = []

for item in compleet.iterrows():
    vrouwenlaag.append(item[1]["geslacht"][:item[1]["ZetelsLaag"]].count("v"))
    vrouwenhoog.append(item[1]["geslacht"][:item[1]["ZetelsHoog"]].count("v"))
    vrouwen.append(item[1]["geslacht"][:item[1]["Zetels"]].count("v"))

compleet["vrouwenlaag"] = vrouwenlaag
compleet["vrouwenhoog"] = vrouwenhoog
compleet["vrouwen"] = vrouwen
%matplotlib inline
hoi = compleet[['Partij','vrouwen','mannen']]
hoi = hoi[(hoi.vrouwen > 0) | (hoi.mannen > 0)]
hoi.set_index('Partij').sort_values('vrouwen').plot(kind='barh') .set(xlabel='aantal m/v', ylabel='partijen')
Partijen over de jaren plotten
peilingwijzer = pd.read_excel("Results_Longitudinal-2.xlsx")
[col for col in peilingwijzer if not col.endswith("low") and not col.endswith("high")]
%matplotlib inline
(peilingwijzer[[u'VVD',
 u'PvdA', u'PVV', u'SP', u'CDA', u'D66', u'CU', u'GL', u'SGP', u'PvdD', u'50PLUS',
 u'VNL', u'Denk', u'FvD',
 u'PP']]*150).plot(figsize=(20,10)).set(xlabel='datum', ylabel='aantal zetels')

Stel VVD 26 zetels hoeveel vrouwen in kamer?
df_vvd = kandidaten[(kandidaten.Partij == 'VVD') & (kandidaten.geslacht == 'v') & (kandidaten.nummer < 26)]

Top 20 mannen namen
mannen.voornaam.value_counts()[:20].sort_values().plot('barh')

Pijlingenwijzer namen gelijk zetten
peilingwijzer = pd.read_excel("peilingwijzer.xlsx")
peilingwijzer.Partij[2] = 'Partij van de Arbeid (P.v.d.A.)'
peilingwijzer.Partij[3] = 'PVV (Partij voor de Vrijheid)'

peilingwijzer = peilingwijzer.rename(columns={'Partij':'partij'})
```

kolom.apply.(lambda x:)

XML

```
import zipfile
import pandas as pd
import xml.etree.ElementTree as ET
import numpy as np
import seaborn as sns
import urllib2 #voor als je geen !wget gebruikt
%matplotlib inline
```

```
#downloaden bestand
#!wget https://www.kiesraad.nl/binaries/kiesraad/documenten/rapporten/2017/2/definitieve-ka
url = urllib2.urlopen('https://www.kiesraad.nl/binaries/kiesraad/documenten/rapporten/2017/
with open('map.zip','wb') as code:
    code.write(url.read())
```

```
zfile = zipfile.ZipFile('map.zip') #wanneer je wget gebruikt, naam van url ipv map.zip
zfile.extractall()
```

```
_file = 'Kandidatenlijsten EML_TK2017-20170213/Kandidatenlijsten_TK2017_Amsterdam.eml.xml'
```

```
#MAAK EEN LIJST VOOR ELKE COLUMN DIE JE WILT HEBBEN
partij = []
positie_lijst = []
initialen = []
voornaam = []
prefix = []
achternaam = []
woonplaats = []
gender = []
prefix_val = ''
tags = []

#PARSE DE FILE MET DE ET.ITERPARSE
for elem,event in ET.iterparse(_file):
    tag = event.tag
    value = event.text
    tags.append(tag)

    if tag == '{urn:oasis:names:tc:evs:schema:eml}RegisteredName':
        partij_val = value

    if tag == '{urn:oasis:names:tc:evs:schema:eml}CandidateIdentifier':
        positie_lijst.append(int(event.attrib.values()[0]))

    if tag == '{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}NameLine':
        initialen.append(value)

    if tag == '{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}FirstName':
        voornaam.append(value)
        partij.append(partij_val)

    if tag == '{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}LastName':
        achternaam.append(value)
        prefix.append(prefix_val)

    if tag == '{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}NamePrefix':
        prefix_val = value
    else:
        prefix_val = ''

    if tag == '{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}LocalityName':
        woonplaats.append(value)

    if tag == '{urn:oasis:names:tc:evs:schema:eml}Gender':
        if value == 'male':
            gender.append('man')
        elif value == 'female':
            gender.append('vrouw')
```

```
#COMBINEER ALLE LIJSTEN TOT EEN DATAFRAME
df = pd.DataFrame({'partij':partij, 'positie_lijst':positie_lijst, 'prefix':prefix, 'initialen':initialen, 'voornaam':v
                   'achternaam':achternaam, 'woonplaats':woonplaats, 'gender':gender})

df = df[['partij', 'positie_lijst', 'initialen', 'voornaam', 'prefix', 'achternaam', 'gender', 'woonplaats']]

df.head()
# vind alle tags: set(tags) voor kopieren hierboven
```

```
#MEEST RECENTE PEILINGWIJZER INLADEN

#MAAK EEN DATAFRAME
peilingwijzer = pd.read_excel('Cijfers_Peilingwijzer.xlsx')

#GEEF DE MATEN  EN DE DESCRIPTION VAN HET DATAFRAME
peilingwijzer.shape, peilingwijzer.describe()

peilingwijzer.head()
```

	Partij	Datum	Percentage	PercentageLaag	PercentageHoog	Zetels	ZetelsLaag	ZetelsHoog

```
#Stel de partij namen van de twee dataframes aan elkaar gelijk
pw_df_namen = ['VVD', 'Partij van de Arbeid (P.v.d.A.)', 'PVV (Partij voor de Vrijheid)',
               'SP (Socialistische Partij)', 'CDA', 'Democraten 66 (D66)', 'ChristenUnie',
               'GROENLINKS', 'Staatkundig Gereformeerde Partij (SGP)',
               'Partij voor de Dieren', '50PLUS', 'VNL (VoorNederland)',
               'DENK', 'Forum voor Democratie', 'Piratenpartij']

pw_df['Partij'] = pw_df_namen
pw_df.head(10)
```

```
#VIND ALLE KANDIDATEN DIE OP DE LIJST STAAN VOOR DE PVDA
df.loc[df['partij'] == 'Partij van de Arbeid (P.v.d.A.)'].head()
```

```
#GEEF EEN SPECIFIEK GETAL UIT HET DATAFRAME ALS INTEGER (ALS VARIABELE GEBRUIKEN VOOR FUNCTIES)
integer = peilingwijzer['PercentageLaag'][(peilingwijzer.Partij == 'PVV (Partij voor de Vrijheid)')].values[0]
integer
```

```
13.0
```

```
#TEL ALLE VALUES IN EEN KOLOM OP
peilingwijzer['Zetels'].sum()

#BEREKEN HET VERSCHIL VAN TWEE KOLOMMEN EN VOEG DIT TOE IN EEN EXTRA KOLOM
peilingwijzer['VerschilHoogLaag'] = (peilingwijzer['ZetelsHoog'] - peilingwijzer['ZetelsLaag'])
peilingwijzer.head()
```

	Partij	Datum	Percentage	PercentageLaag	PercentageHoog	Zetels	ZetelsLaag	ZetelsHoog	VerschilHoogLaag
1	VVD	2017-03-14	17.2	16.5	17.9	26	24	28	4

```
#VIND ALLE MANNAN VAN PARTIJ EN ZET ZE ONDERELKAAR
df.loc[(df['partij'] == 'VVD') & (df['geslacht'] == 'man')]

#TEL HET AANTAL MANNEN VAN EEN SPCIFIEKE PARTIJ
len(df.loc[(df['partij'] == 'VVD') & (df['geslacht'] == 'man')])
```

```
#VIND ALLE KANDIDATEN DIE IN EEN BEPAALDE WOONPLAATS WONEN
df.loc[(df['woonplaats'] == 'Amsterdam')]

#VIND ALLE KANDIDATEN VAN EEN SPECIFIEKE PARTIJ DIE IN EEN BEPAALDE WOONPLAATS WONEN
df.loc[(df['partij'] == 'CDA') & (df['woonplaats'] == 'Amsterdam')]
```

```
#VIND AANTAL KANDIDATEN DIE PER STAD WONEN EN PLOT DIT IN EEN pie PLOT
df['woonplaats'].value_counts().plot(kind='pie')
```

```
#PLOT DE VERHOUDING VAN WOONPLAATS VAN EEN SPCIEFIEKE PARTIJ
df['woonplaats'].loc[(df['partij'] == 'PVV (Partij voor de Vrijheid)')].value_counts().plot(kind='pie', figsize=(10,10)
```

```
#PLOT DE VERHOUDING VAN DE VERHOUDING MAN VROUW VAN EEN SPECIFIEKE PARTIJ IN EEN PIE-GRAPH
df['geslacht'].loc[(df['partij'] == 'PVV (Partij voor de Vrijheid)')].value_counts().plot(kind='pie')
```

```
HERTOGENBOSCH WORDT NIET GEVONDEN DOOR HET STREEPJE (HYPHEN) DUS MOET JE ZO ZOEKEN ALS JE DEZE WILT VINDEN
'woonplaats'].loc[df['woonplaats'].str.contains('Hertogenbosch')]
```

```
#VIND ALLE KANDIDATEN DIE IN EEN VAN DE VOLGENDE STEDEN WONEN: BREDA, TILBURG, S-HERTOGENBOSCH, EINDHOVEN EN PLOT
df_woonplaats = df['woonplaats'].loc[df['woonplaats'].isin(['Breda', 'Tilburg', 'Eindhoven'])]
df_woonplaats.value_counts().plot(kind = 'bar')
```

```
#PARTIJ MET HET AANTAL KANDIDATEN OP DE LIJST PLOT DIT IN EEN BAR GRAPH
df['partij'].value_counts().plot(kind='barh')
```

```
#Hoeveel kandidaten staan er per partij op de lijst plot dit
partijen = []
aantal = []
vrouwen = []
mannen = []

#FOR LOOP OM DE DATA TE GENEREN VOOR DE LIJSTEN
for a in df['partij'].unique():
    partijen.append(a)
    aantal.append(len(df.loc[df['partij'] == a]))
    vrouwen.append(len(df.loc[(df['partij'] == a) & (df['gender'] == 'vrouw')]))
    mannen.append(len(df.loc[(df['partij'] == a) & (df['gender'] == 'man')]))

#MAAK EEN DATAFRAME VAN DE LIJSTEN
kandidaten_per_partij = pd.DataFrame({'partijen':partijen,  'aantal':aantal, 'vrouwen':vrouwen, 'mannen':mannen})
kandidaten_per_partij = kandidaten_per_partij[['partijen', 'aantal', 'vrouwen', 'mannen']]

#GEEF HET PERCENTAGE MANNEN EN VROUWEN EN VOEG DIT TOE AAN HET NIEUWE DATAFRAME
kandidaten_per_partij['perc_vrouw'] = ((kandidaten_per_partij['vrouwen'] /
                                        kandidaten_per_partij['aantal']) * 100).round(2)
kandidaten_per_partij['perc_man'] = ((kandidaten_per_partij['mannen'] /
                                      kandidaten_per_partij['aantal']) * 100).round(2)

#PLOT DE VERHOUDING MAN VROUW PER PARTIJ
kandidaten_per_partij[['partijen', 'mannen', 'vrouwen']].plot(x = 'partijen', kind = 'barh', figsize = (10,10))
```

```
#Vind het aantal vrouwen en mannen die in de tweede kamer komen volgens de laatste peiling
partijen = []
vrouwen = []
mannen = []

for a in pw_df['Partij']:
    partijen.append(a)
    vrouwen.append(len(df.loc[(df['partij'] == a) &
                              (df['positie_lijst'] <= (pw_df['Zetels'].loc[pw_df['Partij'] == a]).values[0]) &
                              (df['gender'] == 'vrouw')]))
    mannen.append(len(df.loc[(df['partij'] == a) &
                             (df['positie_lijst'] <= (pw_df['Zetels'].loc[pw_df['Partij'] == a]).values[0]) &
                             (df['gender'] == 'man')]))
man_vrouw_in_kamer = pd.DataFrame({'partijen':partijen, 'vrouwen':vrouwen, 'mannen':mannen})
man_vrouw_in_kamer[['partijen', 'mannen', 'vrouwen']]
man_vrouw_in_kamer.plot(kind='barh', x='partijen', figsize=(10,10))
```

```
#VINDEN ALLE KANDIDATEN WAARVAN HUN VOORNAAM MET EEN B BEGINT WONEN IN ADAM OF DIEMEN EN GEEF VOORNAAM EN PARTIJ WOONPL
df[['voornaam', 'partij', 'woonplaats']].loc[(df['voornaam'].str.startswith('B')) & df['woonplaats'].isin(['Amsterdam',

#PLOT DE ZETELVERDELING VOLGENS DE PEILINGWIJZER
peilingwijzer[['Partij', 'Zetels']].plot(x ='Partij', y= 'Zetels', kind='barh', title = 'Zetel verdeling')
```

```
#VIND ALLE UNIEKEN PARTIJEN VAN DE TWEE VERSCHILLENDE DATAFRAMES
df['partij'].unique(), peilingwijzer['Partij'].unique()
```

```
#MAAK EEN CROSSTAB VAN DE WOONPLAATS EN HIER DE MAN VROUW VERHOUDING, PLOT DIT IN EEN STACKED PLOT
#ALLES ACHTER ELKAAR
pd.crosstab(df.woonplaats,
            df.gender).sort_values(by='man',
                                   ascending=False).head(5).plot(stacked=True,
                                                                 colormap='Greens',
                                                                 kind='barh',
                                                                 title='Man vrouw verhouding top 5 grootste steden')
```

```
#Maak een formule die de genders naar nullen en éénen maakt, vervolgens maak je hier een nieuwe colomn voor. Hierna deel je
#Alle vrouwen door het totaal aantal leden van de partij zodat je erachter komt hoeveel procent vrouw is.
def gender_to_num(x):
    if x == 'man':
        return 0
    if x == 'vrouw':
        return 1
```

```
df['gender_nieuw'] = df.gender.apply(gender_to_num)

a =df.groupby('partij').gender_nieuw.sum() / df.partij.value_counts() * 100
a.round(2)
```

```
#VIND DE SPECIFIEKE NUMMER VAN EEN PERSOON AAN DE HAND VAN VOORNAAM EN ACHTERNAAM
df['nummer'].loc[(df['voornaam'] == 'Mark') & (df['achternaam'] == 'Rutte')]
```

Positie lijst

```
#TOTAAL AANTAL KANDIDATEN VOOR SPECIFIEKE LIJST
len(df.loc[(df['partij'] == 'VVD')])

#HOEVEEL % VAN TOTAAL KANDIDATEN STAAT OP DE LIJST VOOR SPECIFIEKE PARTIJ?
(df.partij == 'VVD').mean()*100
```

```
#vind alle vrouwen van de vvd en het cda, met het of statement
df.loc[((df['partij'] == 'VVD') | (df['partij'] == 'CDA')) & (df['gender'] == 'vrouw')]
```

```
#VIND ALLE MENSEN OP DE LIJST DIE ALS VOORNAAM GEERT HEBBEN
df.loc[df['voornaam'] == 'Geert']

#VIND ALLE MENSEN OP DE LIJST VAN DE VVD MET HET PREFIX: VAN
df.loc[(df['prefix'] == 'van') & (df['partij'] == 'VVD')]
```

```
#VIND ALLE VROUWEN DIE ALS DE VVD 26 ZETELS KRIJGEN IN DE KAMER KOMEN
df.loc[(df['partij'] == 'VVD') & (df['nummer'] <= 26) & (df['geslacht'] == 'vrouw')]
```

```
#MAAK EEN LIJST VAN ALLE UNIEKE PARTIJEN OP DE LIJST
df['partij'].unique()

#AANTAL PARTIJEN
len(df['partij'].unique())
```

```
from bs4 import BeautifulSoup
import pandas as pd
#import re
from collections import Counter
import numpy as np
from __future__ import division
import pprint
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')
plt.rcParams['figure.figsize'] = (12, 5)
```

```
# hoeveel % van VVD is vrouw
len(KandidatenlijstenFrame[KandidatenlijstenFrame['Partij'] ==
'VVD'].values)/len(KandidatenlijstenFrame)*100
```

```
#MAAK EEN DATAFRAME VAN ALLE ITEMS
PartijZoeker = Kandidatenlijsten('affiliation')
PartijenLijst= [x.text for x in
Kandidatenlijsten.findAll('registeredname')] #lijst met alle
partijen

# maakt dict met partij: [geslacht]
PartijGeslachtDict = {}
PartijenLijstIndex = 0
for partij in PartijZoeker:
  GeslachtLijst = []
  for geslacht in partij.findAll('gender'):
    GeslachtLijst.append(geslacht.text)
  PartijGeslachtDict[PartijenLijst[PartijenLijstIndex]] =
GeslachtLijst
  PartijenLijstIndex+=1
pprint.pprint(PartijGeslachtDict)
```

```
#OPEN FILE
Kandidatenlijsten = BeautifulSoup(open('Kandidatenlijsten_EML_TK2017-
20170213/Kandidatenlijsten_TK2017_Amsterdam.eml.xml').read(), 'lxml')
```

```
#Merge de dataframes samen

PartijGeslachtDF =
pd.DataFrame.from_dict(PartijGeslachtDict,
orient='index')
PartijGeslachtDF =
PartijGeslachtDF.stack().reset_index()
PartijGeslachtDF.columns = ['Partij','Notering',
'Geslacht']

KandidatenlijstenFrame =
PartijKandidaatDF.merge(PartijGeslachtDF)
.merge(PartijWoonplaatsDF)
KandidatenlijstenFrame['Notering'] =
KandidatenlijstenFrame[['Notering']]+1
KandidatenlijstenFrame
```

```
# bar chart met aantal kandidaten op lijs per
partij
AantalKandidatenDict = {}
for partij in PartijenLijst:
  AantalKandidaten =
len(KandidatenlijstenFrame[KandidatenlijstenFrame['Partij'] == partij].values)
  AantalKandidatenDict[partij] =
AantalKandidaten

AantalKandidatenDictDF =
pd.DataFrame.from_dict(AantalKandidatenDict,
orient='index')
AantalKandidatenDictDF.columns = ['Partij']
AantalKandidatenDictDF =
AantalKandidatenDictDF.sort_values('Partij',
ascending=False)
AantalKandidatenDictDF.plot(kind='bar')
```

```
# dit is een test voor de loop in volgende code block
AantalVrouwenVanEenPartijTest = KandidatenlijstenFrame[(KandidatenlijstenFrame['Partij']) ==
'Partij voor de Dieren']
AantalVrouwenVanEenPartijTestRangeTest =
AantalVrouwenVanEenPartijTest[(AantalVrouwenVanEenPartijTest['Notering'] < 20)]
AantalVrouwenVanEenPartijTestRangeTestAantalVrouwen =
AantalVrouwenVanEenPartijTestRangeTest[(AantalVrouwenVanEenPartijTestRangeTest['Geslacht'
]) == 'female'].values
len(AantalVrouwenVanEenPartijTestRangeTestAantalVrouwen)
# Dit is dus de loop waar in vorige code block wordt gerefereerd
PartijDict = {}
for p in PartijenLijst:
  AantalVrouwenVanEenPartijTest = KandidatenlijstenFrame
[(KandidatenlijstenFrame['Partij']) == p ]
  AantalVrouwenDict = {}
  for i in range(50):
    AantalVrouwenVanEenPartijTestRangeTest =
AantalVrouwenVanEenPartijTest[(AantalVrouwenVanEenPartijTest['Notering'] < i)]
    AantalVrouwenVanEenPartijTestRangeTestAantalVrouwen =
    len(AantalVrouwenVanEenPartijTestRangeTest[(AantalVrouwenVanEenPartijTestRangeTest['Geslacht']) ==
'female'].values)
    AantalVrouwenDict[i] = AantalVrouwenVanEenPartijTestRangeTestAantalVrouwen
  PartijDict[p] =  AantalVrouwenDict
PartijDict
#MAAK DATAFRAME VAN DEZE DICT
PartijDictFrame = pd.DataFrame.from_dict(PartijDict, orient='columns')
    #plot aantal vrouwen per partij voor bepaalde partij
    for p in PartijenLijst:
      AantalVrouwenNavAantalZetelsFrame =
    pd.DataFrame.from_dict(PartijDict[p], orient='index')
      AantalVrouwenNavAantalZetelsFrame.plot(kind='bar', title='Aantal Vrouwen
    N.a.v. Aantal Zetels Voor '+p).set_ylim(0, 40)
```

```
#PEILINGWIJZER METHODES OP GOED DATAFRAME TE KRIJGEN
#LAATSTE PEILINGEN
LaatstePeilingWijzer = pd.read_excel('https://s3.eu-central-
1.amazonaws.com/louwerse/Public/Peilingwijzer/Last/Cijfers_Peilingwijzer.xlsx')
PartijNamenXML = {u'PvdA':u'Partij van de Arbeid (P.v.d.A.)', u'PVV':u'PVV
(Partij voor de Vrijheid)', u'SP':u'SP (Socialistische Partij)',
          u'D66':u'Democraten 66 (D66)', u'CU':u'ChristenUnie', u'GL':
u'GROENLINKS', u'SGP':u'Staatkundig Gereformeerde Partij (SGP)',
          u'PvdD':u'Partij voor de Dieren', u'VNL':u'VNL (VoorNederland)',
u'Denk': u'DENK', u'FvD':u'Forum voor Democratie',
          u'PP':u'Piratenpartij'}
LaatstePeilingWijzer = LaatstePeilingWijzer.set_index('Partij')
LaatstePeilingWijzer.rename(PartijNamenXML, inplace=True)
LaatstePeilingWijzer = LaatstePeilingWijzer.reset_index()
LaatstePeilingWijzer
```

```
#ALLE PEILINGEN
AllePeilingWijzers = pd.read_excel('https://s3.eu-central-
1.amazonaws.com/louwerse/Public/Peilingwijzer/Last/Results_Longitudinal.xlsx')
AllePeilingWijzers = AllePeilingWijzers[['VVD', 'PvdA', 'PVV', 'SP','CDA','D66',
'CU', 'GL', 'SGP', 'PvdD', '50PLUS', 'VNL', 'Denk', 'FvD', 'PP']] * 150
AllePeilingWijzers = AllePeilingWijzers.round()
#AllePeilingWijzers.plot(figsize=(18,10)).set_ylim(0,45)
AllePeilingWijzers = AllePeilingWijzers.rename(columns={u'PvdA':u'Partij van de
 Arbeid (P.v.d.A.)', u'PVV':u'PVV (Partij voor de Vrijheid)', u'SP':u'SP
(Socialistische Partij)',
          u'D66':u'Democraten 66 (D66)', u'CU':u'ChristenUnie', u'GL':
u'GROENLINKS', u'SGP':u'Staatkundig Gereformeerde Partij (SGP)',
          u'PvdD':u'Partij voor de Dieren', u'VNL':u'VNL (VoorNederland)',
u'Denk': u'DENK', u'FvD':u'Forum voor Democratie',
          u'PP':u'Piratenpartij'})
AllePeilingWijzers = AllePeilingWijzers.reset_index()
AllePeilingWijzers = AllePeilingWijzers.rename(columns={'index': 'date'})
AllePeilingWijzers = AllePeilingWijzers.set_index('date')
AllePeilingWijzers.head()
```

```
#ALLEPEILINGEN SECUURDER
alle=pd.read_csv('https://d1bjgq97if6urz.cloudfront.net/Public/Peilingwijze
r/Last/Results_DyGraphs_Seats.csv',index_col='Date')
#print (alle.shape)
alle =alle.applymap(lambda s: int(s.split(';')[1])) #if you only want to keep
the middle number
#now translate to our names
alle=alle.T
alle.rename(PartijNamenXML, inplace=True)
alle = alle.T
alle
```

```
#Opmaken tweede kamer op basis van laatste peiling
AllePeilingWijzersLaatstePeiling = AllePeilingWijzers.tail(1)
VerdelingLijst = []

for partij in AllePeilingWijzersLaatstePeiling.columns:
  if len(AllePeilingWijzersLaatstePeiling[AllePeilingWijzersLaatstePeiling[partij]>0]):
    Kandidaten = int(AllePeilingWijzersLaatstePeiling[partij].item())

  PartijFrame = AllePeilingWijzersLaatstePeiling[partij]
  KandidatenPerPartij = KandidatenlijstenFrame[KandidatenlijstenFrame['Partij'] == partij ][0:Kandidaten]
  VerdelingLijst.append(KandidatenPerPartij)
tweede_kamer = pd.concat(VerdelingLijst)
tweede_kamer= tweede_kamer.reset_index()
tweede  kamer
```

```
#OM DATAFRAME TE MAKEN EN PLOTTEN HOEVEEL VROUWEN
OVER TIJD IN DE KAMER MAAK COPY
AllePeilingWijzerscopy = AllePeilingWijzers.copy()
AllePeilingWijzerscopy2 = AllePeilingWijzerscopy.copy()
AllePeilingWijzerscopy2
```

```
#AANTAL VROUWEN IN DE TWEEDE KAMER OVER DE TIJD (PEILINGWIJZER)
for partij in AllePeilingWijzerscopy.columns:
  for i in AllePeilingWijzerscopy[partij]:
    vrouwen = PartijDict[partij][i]
    AllePeilingWijzerscopy2.loc[AllePeilingWijzerscopy[partij] == i, partij] = vrouwen
AllePeilingWijzerscopy2
#PLOT DIT DATAFRAME
AllePeilingWijzerscopy2.plot(figsize=(25,10)).set_ylim(0,20)
```

```
#EXTRA KOLUM MET TOTAAL EN TOTAAL PERCENTAGE
AllePeilingWijzersTotalen = AllePeilingWijzerscopy2.copy ()
AllePeilingWijzersTotalen['Totaal'] = AllePeilingWijzersTotalen.sum(axis=1)
AllePeilingWijzersTotalen['Totaal percentage'] = AllePeilingWijzersTotalen
['Totaal']/150
AllePeilingWijzersTotalen
```

```
# hoeveel vrouwen in 2e kamer op basis van laatste peiling?
Aantal_vrouwen = AllePeilingWijzersTotalen.tail(1)
print ('Het aantal vrouwen volgens de peiling van 14 Maart =',
int(Aantal_vrouwen['Totaal'].item()))
```

```
#PLOT TOTALITEIT AAN VROUWEN OVER DE TIJD IN DE KAMER
AllePeilingWijzersTotalen['Totaal'].plot(figsize=(25,10))
#PLOT PERCENTAGE TOTALITEIT OVER DE TIJD IN DE KAMER
AllePeilingWijzersTotalen['Totaal percentage'].plot(figsize=(18,7))
```

```
#selecteren van twee partijen
KandidatenlijstFrame[KandidatenlijstFram
e['Partij'].isin(['PVV (Partij voor de
Vrijheid)','VVD'])]
```

```
#selecteren van twee items in twee
kolommen
KandidatenlijstFrame[(KandidatenlijstFra
me['Woonplaats'] == 'Amsterdam') &
(KandidatenlijstFrame['Geslacht'] ==
'male')]
```

```
#selecteren van een kolom als kolom
begint met een letter
KandidatenlijstFrame[['Naam',
'Partij']].loc[(KandidatenlijstFrame['Naam'
].str.startswith('B'))]
```

```
#IAATSTE AANTAL ZETELS VAN PARTIJ
AllePeilingWijzers14Maart = AllePeilingWijzers.tail(1)
AllePeilingWijzers14MaartT = AllePeilingWijzers14Maart.T
AllePeilingWijzers14MaartT
```

```
#AANTAL VROUWEN LAATSTE PEILING 14/03
AllePeilingWijzerscopy14Maart =
AllePeilingWijzerscopy2.tail(1)
AllePeilingWijzerscopy14MaartT =
AllePeilingWijzerscopy14Maart.T
AllePeilingWijzerscopy14MaartT
```

```
#PLOT MAN/VROUW VERDELING PER PARTIJ
t = pd.merge(AllePeilingWijzerscopy14MaartT,
AllePeilingWijzers14MaartT, left_index=True,
right_index=True, how='inner')
t.columns = ['Vrouwen', 'Zetels']
t['Manner'] = t['Zetels'] - t['Vrouwen']
t[['Manner', 'Zetels']].plot(kind='bar')
```

```
#PERCENTAGE VROUWEN
t['Percentage Vrouwen'] = (t['Vrouwen']/t['Zetels'])*100
t[['Percentage Vrouwen']].plot(kind='bar')
t
```

```
#GEEF SIMPELE BEREKENINGEN
#TEL TOTAAL AANTAL VROUWEN
sm =
KandidatenlijstenFrame[KandidatenlijstenFrame.Ge
slacht == 'female']
len(sm)
#GEEF DE MATEN  EN DE DESCRIPTION VAN HET
DATAFRAME
KandidatenlijstenFrame.shape,
KandidatenlijstenFrame.describe()
#SELECTEER WAARDE UIT DATAFRAME
integer =
LaatstePeilingWijzer['Zetels'].loc[(LaatstePeilingWijz
er['Partij'] == 'VVD')].values[0]
integer
```

```
#MAAK EEN PIVOT TABLE MET ALS INDEX
'partij,geslacht' EN TEL DE KOLUMMEN BIJ ELKAAR
OP
df=KandidatenlijstFrame.pivot_table(index=["Partij"
,"Geslacht"],aggfunc='count').dropna()
```

```
#DROP DE KOLUM NAAM EN HERNOEM DE KOLUM
WOONPLAATS NAAR CRIB
df=df.drop(['Naam','Woonplaats'],axis=1).rename(c
olumns={'Woonplaats':'woonplaats'})
```

```
#PLOT HET GESLACHT IN EEN BAR HORIZONTAAL IN
EEN BALK (STACKED = TRUE)
df.unstack('Geslacht').fillna(0).plot(kind='bar',title=
'verhouding man/vrouw',stacked = True,
figsize=(20,10))
```

```
ascending=[True,False]) - Sort values by col1 in
ascending order then col2 in descending order
df.groupby(col) - Return a groupby object for
values from one column
df.groupby([col1,col2]) - Return a groupby
object values from multiple columns
df.groupby(col1)[col2].mean() - Return the
mean of the values in col2, grouped by the values
in col1 (mean can be replaced with almost any
function from the statistics section)
df.pivot_table(index=col1,values=
[col2,col3],aggfunc=max) - Create a pivot table
that groups by col1 and calculates the mean of
```

pd.read_html