```
#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
```

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

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

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

```
'''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()
```

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

%ls -lh "File"

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

```
# 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
```

axis=0 =1:

```
# 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')
```

help(pd.DataFrame)
%time

```
# 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})
```

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

① kandidaten = pd.DataFrame(data)

```
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')
```

date time 64

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

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

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

```
#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'})
```

④

```
genders = kandidaten.groupby("Partij")["gender"].sum()
dfg = genders.to_frame()
dfg = dfg.reset_index()
```

⑦

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

②

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

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

③

```
#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
```

⑤

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

vrouwenlaag = []
mannenlaag = []

vrouwenhoog = []
mannenhoog= []

vrouwen = []
mannen = []
```

⑥

```
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()
```

df.c.map({'female':0,'male':

pd.merge(df1,df2,on=c)

c='sex'

df.c.Replace(V,AV)

```
from bs4 import BeautifulSoup
import pandas as pd
import re
from collections import Counter
import numpy as np
from __future__ import division
%matplotlib inline
import seaborn
```

```
hertentamen= pd.read_excel('Cijfers_Peilingwijzer.xlsx')
hertentamen.shape
hertentamen.head()
```

```
soup = BeautifulSoup(open('Kandidatenlijsten_TK2017_Amsterdam.eml.xml').read())
print soup.prettify()
```

```
#Open de XML
soup = BeautifulSoup(open('Kandidatenlijsten_TK2017_Amsterdam.eml.xml').read())
#Zoek alle partij info
parties= soup('affiliation')
#Zoek alle partijnamen
partynames= [p.text   for p in soup.findAll('registeredname')]
#Maak de genderlist per partij
genderlist= [p.find('registeredname').text : [g.text for g in  p.findAll('gender')] for p in parties
```

```
# Vertalen van partijnamen
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'}
```

```
AantalVrouwenPerPartijPerAantalGewonnenZetels={p:{N: Counter(genderlist[p][:N])['female']  for N in range(50)}
                                               for p in genderlist}
df1 = pd.DataFrame.from_dict(AantalVrouwenPerPartijPerAantalGewonnenZetels, orient='index')
df1[49].plot(kind='barh')
```

```
# Voor de eerste N mensen op de lijst zet N =80 (is maximum) voor de hele lijst
N=50
#Partijnaam met aantal mannen en vrouwen
gendercount={p:Counter(genderlist[p][:N]) for p in genderlist}

gdf=pd.DataFrame.from_dict(gendercount, orient='index').fillna(0).astype(int)
#min_lijst_lengte= (gdf.sum(axis=1)).min()
#print min_lijst_lengte
# Voeg de ratio toe
gdf['Percentage']=  ((gdf.female/ gdf.sum(axis=1))*100).round()
gdf.sort_values('Percentage',ascending=False)

ef=pd.DataFrame(translate.values())
ef.index=ef[0]
print ("Aantal vrouwen in  de top (met max 50) van elke lijst")
ef.join(gdf ).sort_values('Percentage' )[['male','female','Percentage']]
```

```
# Dataframe beperken tot een selectie
peiling = pd.read_excel('Cijfers_Peilingwijzer.xlsx')
peiling2 = peiling[peiling.Partij=='VVD']
peiling2
```

```
# Kolom toevoegen aan DF
peiling2['Testen'] = peiling2.ZetelsLaag + peiling2.ZetelsHoog
peiling2
```

```
# Peilingwijzer
laatstepeiling='https://s3.eu-central-1.amazonaws.com/louwerse/Public/Peilingwijzer/Last/Cijfers_Peilingwijzer.xlsx'
allepeilingen= 'https://s3.eu-central-1.amazonaws.com/louwerse/Public/Peilingwijzer/Last/Results_Longitudinal.xlsx'
laatstedf= pd.read_excel(laatstepeiling)
alledf= pd.read_excel(allepeilingen)
print alledf.shape
alledf.tail()
laatstedf.index= laatstedf.Partij
#laatstedf
laatstedf.rename(translate, inplace=True)

# Dit is een handige file met de zetel aantallen
alleZdf=pd.read_csv('https://dlbjgq97if6urz.cloudfront.net/Public/Peilingwijzer/Last/Results_DyGraphs_Seats.csv',
                    index_col='Date')
print alleZdf.shape
print (alleZdf.sum(axis=1) == 150).mean()  # check if all add to 150
alleZdf =alleZdf.applymap(lambda s: int(s.split(';')[1])) # want (ci low; mean; ci high)
# now translate to our names
alleZdf=alleZdf.T
alleZdf.rename(translate, inplace=True)
alleZdf=alleZdf.T
alleZdf.tail()
```

```
alleZdf.rename(columns = {'PP':'Piratenpartij'}, inplace=True)
for p in alleZdf.columns:
    alleZdf[p]= alleZdf[p].apply(lambda z: AantalVrouwenPerPartijPerAantalGewonnenZetels[p][z])
fem= alleZdf.plot(figsize=(18,7),
                  title='Aantal vrouwen in de Tweede Kamer door de tijd, per partij.\n Ge
```

```
som= alleZdf.T.sum()
print  som.describe()
fem= alleZdf.T.sum().plot(figsize=(18,7),
                          title='Aantal  vr
fig=fem.get_figure()
fig.savefig('FemThroughTime.png')
```

```
df=laatstedf.join(gdf)
df
```

```
# Percentage naar aantal zetels
def Perc2Zetels(p):
    return round(p * (150/100))

def Ratio2Zetels(p):
    return round(p * 150)

ef= gdf.join(alledf.T.rename(translate)).dropna().drop([u'male', u'female', u'Percentage'], axis =1)

ZetelsDoorDeTijd= ef.applymap(Ratio2Zetels )
ZetelsDoorDeTijd.head()
```

```
ef= ZetelsDoorDeTijd.T
for p in ef.columns:
    ef[p]= ef[p].apply(lambda z: AantalVrouwenPerPartijPerAantalGewonnenZetels[p][z])

fem= (ef.T.sum()/ ZetelsDoorDeTijd.sum()*100).plot(figsize=(18,7),
                                                   title='Percentage vrouwen in de Tweede
```

```
#Hoeveel vrouwen bij ZetelsLaag CDA
len([1 for mp in  genderlist['CDA'][:df.ZetelsLaag['CDA']]  if mp=='female' ])
```

```
df.rename(index = {'PP':'Piratenpartij'}, inplace=True)
def aantalvrouwen(p,column,df):
    zetels = df[column][p]
    aantalvrouwen= len([1 for mp in  genderlist[p][:zetels ]  if mp=='female' ])
    return aantalvrouwen

for c in  [c for c in df.columns if c.startswith('Z')]:
    Fc= 'Fem'+c
    # compute aantal vrouwen
    Fem=pd.DataFrame.from_dict({p:aantalvrouwen(p,c,df) for p in df.index},orient='index')
    Fem.columns=[Fc]
    # voeg toe aan df
    df=df.join(Fem)
    # bereken de ratio
    df['FratioPeiling'+c]= (df[Fc] /df[c]).fillna(0)
```

```
# Aantal vrouwen in de kamer
VrouwenPeilingwijzer=df.sort_values('Zetels',ascending=True)[[ u'Zetels',u'FemZetels',u ',u'FratioPeilingZetels']]

print "Op basis van deze peiling komen er %s vrouwen in de kamer. Dat is %s procent" %  % (VrouwenPeilingwijzer.FemZetels.sum(),
                                                                                           round(VrouwenPeilingwijzer.FemZetels.sum()/
                                                                                                 VrouwenPeilingwijzer.Zetels.sum()*100))

print VrouwenPeilingwijzer
```

pd.DataFrame.From_dict(aantal_vrouwen['50 PLUS'], orient='index')
.plot(kind='barh', figsize=(10,8))

Filter. [[...]]

```
# Open xml bestand
bestand = open('Kandidatenlijsten_TK2017_Amsterdam.eml.xml').read()
soup = bs.BeautifulSoup(bestand, 'xml')
```

```
# Namen van de partijen
partij_namen = []
for partij in soup.findAll('RegisteredName'):
    partij_namen.append(partij.text)
print partij_namen
```

```
# Zoek voor elke partij de vrouwen en de mannen
partijen = soup('Affiliation')
genderdict = {}
for partij in partijen:
    genderdict[partij.find('RegisteredName').text]
    = [gender.text for gender in partij.findAll('Gender')]
```

```
# Zoek namen
namendict = {}
for partij in partijen:
    namendict[partij.find('RegisteredName').text]
    = [name.text for name in partij.findAll('FirstName')]
namendict['50PLUS'].count('Henk')
```

```
# Maak dataframe mannen en vrouwen
vrouwen = {}
mannen = {}
for partij in genderdict:
    male_female = genderdict[partij]
    vrouwen[partij] = male_female.count('female')
    mannen[partij] = male_female.count('male')
vrouwen_df = pd.DataFrame.from_dict(vrouwen, orient='index')
mannen_df = pd.DataFrame.from_dict(mannen, orient='index')
```

pd.concat([vrouwen_df, mannen_df], axis=1)

```
# Per aantal zetels komen er x aantal vrouwen in de kamer
aantal_vrouwen = {partij:{N: Counter(genderdict[partij][:N])['female']
                  for N in range(50)} for partij in genderdict}
Counter(genderdict['Partij voor de Dieren'][:25])['female']
```

gendercount = {p: Counter(genderdict[p][:N] for p in genderdict}
gdf = pd.DataFrame.From_dict(gendercount, orient='index').fillna(0).astype(int)
(gdf.sum(axis=1)) .min()

```
translate = {... 'D66': 'Democraten 66' (D66)'}
laatstedf.index = laatstedf['Partij']
laatstedf.rename(translate, inplace=True)
```

zetels = pd.read_csv
zetels = zetels.set_index('Da...
zetels = applymap
zetels = zetels.T
zetels.rename
join.
zetels = zetels.T

```
# Voor alle waarden in kolom apply
zetels = zetels.applymap(lambda s: int(s.split(';')[1]))
nieuwedf = alledf.apply(lambda x: (x*150))
```

zetels2 = zetels.sum
zetels2.clesc...

```
# Aantal vrouwen in de Tweede Kamer
for p in zetels.columns:
    zetels[p]= zetels[p].apply(lambda z: aantal_vrouwen[p][z])
```

zetels = dan nieuwe met aantal vrouwen | zetels.sort(as.= F...
.plot

```
# Skippen met loc
alledf = alledf.T
alledf = alledf.iloc[::3]
```

```
# String contains
barry_voor = hoofdelijke_stemming_df['pro'].str.contains('Barry Madlener')
len(adam_GL[adam_GL['voornaam'].str.startswith('T')])
```

str.startswith('..')

```
# Crosstab
kruistabel = pd.crosstab(rutte2['proposaltype'], rutte2['result'], margins= True
```

```
# Hoeveel procent stemmingen heeft voor gestemd
partijnamen = rutte2.columns[11:50]
stemmen = rutte2[partijnamen].apply(pd.value_counts)

partij_dict = {}
for partij in partijnamen:
    partij_dict[partij] = float(stemmen[partij].loc[1])/float(stemmen[partij].sum(

dataframe = pd.DataFrame.from_dict(partij_dict, orient='index')
dataframe = dataframe.dropna()
dataframe = dataframe.sort(0, ascending=False)

plotje_partijen = dataframe.plot(kind='barh', title='Partijen die voor hebben gest
```

```
# Sorten eerst dataframe maken
pd.DataFrame(populair).sort(0,ascending=False).plot(kind='barh', title='mo
```

oplopend

IMPORTING DATA

- `pd.read_csv(filename)` - From a CSV file , sep=';'
- `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

value_counts
↳ count freq that value occurs
bv woonplaats hoe vaak

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 Array
- `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.

names[names.Count == names.Count.median()]

```
# Filteren in een dataframe
peilingen.sort('Zetels', ascendi
.plot('Partij', 'Zetels', kind =
```

```
%matplotlib inline
import pandas as pd
import re
from lxml import etree
from bz2file import BZ2File
import codecs
import urllib2
import urllib
import os
import seaborn as sn
import zipfile
import bs4 as bs
from collections import Coun
```

```
import sys
stdout = sys.stdout
reload(sys)
sys.setdefaultencoding('utf-8')
sys.stdout = stdout
```

names.Count.idxmax()
↳ geeft index

# Filteren df om plotten
peilingen.sort('Zetels', ascending=False).plot('Party', 'Zetels', kind='bar')

```
tree = ET.parse('/Users/Joel/Downloads/Kandidatenlijsten_EML_TK2017-20170213/
Kandidatenlijsten_TK2017_Amsterdam.eml.xml')
root = tree.getroot()
L = []
for hoi in root:
    for doei in hoi:
        for hee in doei:
            for nee in hee.findall('{urn:oasis:names:tc:evs:schema:eml}Affiliation'):
                for iden in nee.findall('{urn:oasis:names:tc:evs:schema:eml}AffiliationIdentifier'):
                    partij = iden.find('{urn:oasis:names:tc:evs:schema:eml}RegisteredName').text
                for iden in nee.findall('{urn:oasis:names:tc:evs:schema:eml}Candidate'):
                    for i in iden.findall('{urn:oasis:names:tc:evs:schema:eml}CandidateIdentifier'):
                        Id = i.get('Id')
                    for can in iden:
                        for person in can.findall('{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}PersonName'):
                            initial = person.find('{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}NameLine').text
                            name = person.find('{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}FirstName').text
                            surname = person.find('{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}LastName').text
                            gender = iden.find('{urn:oasis:names:tc:evs:schema:eml}Gender').text
                        for adres in can.findall('{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}Locality'):
                            stad = adres.find('{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}LocalityName').text
                            L.append({'partij': partij, 'ID': int(Id), 'initial': initial, 'name':name, 'surname': surname, 'gender':gender,
'stad':stad})
Readen    pd.read_excel('path') (of read_csv)
Values van allen    rutte2.proposaltype.value_counts()
alle votetype met roll call    hoofdelijke_stemming_df = rutte2[(rutte2['votetype'] == 'Roll call') (je kan em &'en)]
hoeveel adopted gemiddeld voorkomt    (hoofdelijke_stemming_df['result'] == 'adopted').mean() * 100
Vind alles van Barry (handig als t een lijst heeft binnen in)    hoofdelijke_stemming_df.pro.str.contains('Barry Madlener').sum()
Tabel    pd.crosstab(rutte2.proposaltype,rutte2.result,margins=True)
2 manieren    rutte2.pivot_table(index='proposaltype', columns='result', values='pro', aggfunc=len, margins=True)
Blijkbaar belangrijk  im = rutte2[(rutte2.authorparty==rutte2.supporterparties) & (rutte2.authorparty != '[]')]
im.shape  Checken hoe tabel eruit ziet
staafdiagram machen    im.authorparty.value_counts().sort_values().plot(kind='barh', title='aantal')
Checken of er 568 boys zijn    im.authorparty.value_counts().sum()==568    df.groupby(col1).agg(np.
Column names    print list(rutte2)    mean)
Columnen pakken, gemiddelde pakken en procent, NaN weg, sorten op values en histogram maken    - find the
(rutte2[[partijen]].mean()*100).dropna().sort_values(ascending=False).plot(kind='bar')
Partijen stemmen pakken van normal en aantal stemmen    aanwezig = rutte2[rutte2.votetype=='Normal']
[['partijnamen']].count()
Niet-stemmers weghalen    aanwezig[(aanwezig > 0)]
Denk datum pakken denk = rutte2.dropna(subset=[u'Kuzu/Öztürk'])
denk.date.min()
columnen pakken met 1 of 0 zonder NaN    stems = rutte2[rutte2.votetype=='Normal'][[partijen]].dropna(axis=1)
pakt eerste alle 1 PVV, dan de mean, dan sort je em, dan 2 decimaal en procent
stems[stems.PVV==1].mean().sort_values(ascending=False).round(2) *100
Zelfde value vinden hey.loc[hey['partij'].isin(['VVD', 'CDA', '50PLUS'])]
hey = hey.loc[hey['partij'].isin(['VVD', 'CDA', '50PLUS'])]
samen = peil.join(hey, how = 'outer')
df = pd.DataFrame(L)
```

```
import pandas as pd
import numpy as np
import seaborn as sn
%matplotlib inline
import matplotlib.pyplot as plt

len(set(df['partij']))
len(peil['Partij'])

counter = PVV['gender'].value_counts()
counter.plot(kind = 'pie', title = 'Verhouding mannen tot
vrouwen binnen de PVV')
plt.show()
Groen = df[(df['partij'] == 'GROENLINKS') & (df['gender']
== 'male')]
len(Groen['name'].str.startswith('T'))
steden = df['stad'].value_counts()[:20]
steden.sort_values(ascending = False).plot(kind='bar',
title='hoi ik ben joel')
plt.show()

cross = pd.crosstab(df.stad, df.gender, margins=True)
#print cross.sort_values(by='All', ascending=False)[:20]
cross[(cross['female'] > cross['male'])]

peil = pd.read_excel('/Users/Joel/Documents/
Informatiekunde/Cijfers_Peilingwijzer.xlsx')
peil.sort_values(by='Zetels', ascending =
False).plot('Partij', 'Zetels',kind='bar', title='hoi ik ben
joel')

len(set(df['partij']))
len(peil['Partij'])

partij_kandidatenlijst = ['partijen']
partij_peilingwijzer = list(set(peil.Partij))
partij_peilingwijzer = sorted(partij_peilingwijzer)
partij_kandidatenlijst = sorted(partij_kandidatenlijst)

dic = {}
for partij in partij_kandidatenlijst:
    index = partij_kandidatenlijst.index(partij)
    dic[partij] = partij_peilingwijzer[index]
df.replace(dic,inplace=True)
dfj = df.join(peil.set_index('Partij'), on = 'partij')
dfj.update(dfj[['columnen]].fillna(0))
```

```
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 KOLON
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 ACHTEN 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 eenen 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')]

#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())
```

```
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'
```

XML Inlezen

```
#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')
```

DF

```
#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
```

JPG_CSV

```
: #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

Namen

```
#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')]
```

```
#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())
```

```
#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
```

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

```
#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()
```

```
# 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
```

```
# 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)
```

```
#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'))]
```

```
%ls -lh "File"
import pandas as pd
import numpy as np
import seaborn as sn
import lxml
from lxml import etree
%matplotlib inline
```

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

help(pd.DataFrame)
%tim

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})
```

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

① kandidaten = pd.DataFrame(data)

```
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'})
```

④

```
genders = kandidaten.groupby("Partij")["gender"].sum()
dfg = genders.to_frame()
dfg = dfg.reset_index()
```

②

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

```
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
```

③

⑤

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

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
```

⑥

⑦

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

RATIO + TOTAAL

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

Groupby

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

```
#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
```

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

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

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

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

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

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

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

%ls -lh "File"

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

```
# 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
```

```
# 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')
```

help(pd.DataFram
%tim

Meerdere files

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

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

```
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})
```

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
pd.excel(" ")

(1) kandidaten = pd.DataFrame(data)

```
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()
```

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

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

```
#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'})
```

(4)

```
genders = kandidaten.groupby("Partij")["gender"].sum()
dfg = genders.to_frame()
dfg = dfg.reset_index()
```

(7) →

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

(2)

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

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

(3)

```
#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
```

print. df.columns
df[[..., ..,]].mean() *100).

(5)

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

vrouwenlaag = []
mannenlaag = []

vrouwenhoog = []
mannenhoog= []

vrouwen = []
mannen = []
```

(6)

```
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()
```

test["Fem"].sort_values(ascending=True).dropna
.plot(kind="bar")

[column].apply(lambda x : x.count(" ")).sum()

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import os
from lxml import etree
```

Open Bestand csv & xlsx:

Open kandidatenlijstentk2017.csv en

Cijfers_Peilingwijzer.xlsx in aparte Dataframes.

```
df_kandidaten = pd.read_csv('bestandnaam')
df_peiling = pd.read_excel('bestandnaam')
```

Open bestand excel:

```
cijfers_peilingen = pd.read_excel('Bestandnaam')
cijfers_peilingen.head()
```

Laat alle variabele zien binnen het dataframe:

```
cijfers_peilingen.describe()
```

Laat het aantal rijen en kolommen zien :

```
cijfers_peilingen.shape
```

Tel alle waardes op van een specifieke kolom (zetels, zetelslaag, zetelshoog, percentagelaag, percentagehoog) :

```
cijfers_peilingen.sum()
cijfers_peilingen['kolom'].sum()
```

Verschil tussen twee kolommen in nieuwe kolom (tussen percentagelaag/percentage hoog – zetelslaag/zetelshoog)

```
cijfers_peilingen['Verschil'] = (cijfers_peilingen['kolom']) - cijfers_peilingen['kolom'])
cijfers_peilingen.head(10)
```

Haal een variabele uit het dataframe uit een kolom – voorbeeld "Verschil" "Partij" "PVV"

```
getal =
cijfers_peilingen['Verschil'].loc[(cijfers_peilingen['Partij'] == 'PVV')].values[0]
getal
```

Haal verschillende kolommen eruit en haal de indexcijfers weg en zet daar een andere kolom voor terug.

```
cijfers_peilingen[['Zetels, 'Datum', 'Percentage']].set_index('Zetels').head()
```

Welke partijen hebben boven de 25 zetels of onder bijvoorbeeld (<)?

```
zetels = cijfers_peilingen.loc[cijfers_peilingen['Zetels'] > 25]
zetels
```

Sorteer dataframe op bepaalde column – ascending = false (boven naar onder)

```
cijfers_peilingen_new = cijfers_peiling.sort_values(by='Zetels', ascending=False)
cijfers_peiling_new.head()
```

Hoeveel partijleden zitten er in de dataset van de PVV?

```
len(cijfers_peilingen[cijfers_peilingen.Partij=='PVV'])
```

Hoe lang is de kolom partij?

```
len(cijfers_peilingen.Partij)
```

Hoe veel unieke/dubbele partijen zitten er in deze kolom?

```
len(set(cijfers_peilingen.Partij))
```

Hoogste aantal zetels

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

Sorteer de partijen met het hoogste percentage ?

```
cijfers_peilingen = cijfers_peilingen.set_index ('partij')
percentage_new = cijfers_peilingen.sort_values(by='Percentage', ascending=False)
percentage_new.head()
```

Plot de partij en het percentage – bar

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

Plot de partij en het percentage - pie

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

Plot de partij en het percentage - pie

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

Plot de partij en het percentage - grapic

```
percentage_new['Percentage'].plot(title = 'Hoogste Percentage plot')
```

Open een excel bestand en laat de eerste 5 zien?

```
results_longitudinal = pd.read_excel('Results_Longitudinal.xlsx')
results_longitudinal.head()
```

XML Files openen.

```
import xml.etree.ElementTree as ET
tree_new = ET.parse('bestandnaam')
root = tree_new.getroot()
```

XML Files openen.

```
root.tag
```

XML Files openen.

```
root.attrib
```

XML Files openen.

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

```
#Hoeveel mannen en vrouwen zijn in kandidatenlijst van Amsterdam?
mannen = 0
vrouwen = 0
for event, elem in ET.iterparse('Kandidatenlijsten_TK2017_Amsterdam.eml.xml'):
    tag = elem.tag
    value = elem.text
    if tag == "{urn:oasis:names:tc:evs:schema:eml}Gender" :
        if value == 'female':
            vrouwen += 1
        else:
            mannen += 1
    elem.clear()
print ('aantal mannen: = ', mannen)
print ('aantal vrouwen: = ', vrouwen)
print ('totaal: = ', (mannen + vrouwen))
```

```
#Maak een dataframe met voornaam, initialen, achternaam, man/vrouw, woonplaats
partij = []
voornaam = []
initialen = []
achternaam = []
gender = []
woonplaats = []
for event, elem in ET.iterparse('Kandidatenlijsten_TK2017_Amsterdam.eml.xml'):
    tag = elem.tag
    value = elem.text
    if tag == "{urn:oasis:names:tc:evs:schema:eml}RegisteredName": (partij)
        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':
            gender_val = 'man'
        else:
            gender_val = 'vrouw'
        gender.append(gender_val)
    if tag == '{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}LocalityName':
        woonplaats.append(value)

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

```
Print het aantal mannen en het aantal vrouwen?
print (len(df[df.geslacht=='man']))
print (len(df[df.geslacht=='vrouw']))
```

```
Print een lijst van alle unieke partijen op de kandidatenlijst?
print (df['partij']).unique()
len(set(df['partij']))
```

```
Hoeveel kandidaten per partij en maak een plot?
kandidaten_per_partij= df['partij'].value_counts()
kandidaten_per_partij.plot(kind='bar', title= 'kandidaten per partij')
```

```
Bepaalde partijen met een bepaald geslacht
df.loc[(df['partij'] == 'VVD') & (df['geslacht']== 'man')].head()
```

```
Hoeveel procent is man van de VVD?
totaal aantal mannen
mannen_VVD = len(df.loc[(df['partij'] == 'VVD') & (df['geslacht']== 'man')])
totaal aantal vrouwen
vrouwen_VVD = len(df.loc[(df['partij'] == 'VVD') & (df['geslacht']== 'vrouw')])

totaal_VVD = len(df[df.partij == 'VVD'])

aantal_procenten_man = (mannen_VVD * 100) / totaal_VVD
print (aantal_procenten_man)
```

```
Plot de verhouding man vrouw van de VVD?
verhouding = df['geslacht'].loc[(df['partij'] == 'VVD')].value_counts()
verhouding.plot(title='Verhouding man vrouw VVD)
```

```
Plot de verhouding man vrouw van de VVD?
verhouding = df['geslacht'].loc[(df['partij'] == 'VVD')].value_counts()
verhouding.plot(kind='bar', title='Verhouding man vrouw VVD)
```

```
Plot de verhouding man vrouw van de VVD?
verhouding = df['geslacht'].loc[(df['partij'] == 'VVD')].value_counts()
verhouding.plot(kind='pie', title='Verhouding man vrouw VVD)
```

```
Zoek een Kandidaat op uit een bepaalde woonplaats
df.loc[(df['woonplaats'] == 'Amsterdam')]
```

```
Kandidaat uit Amsterdam die in DENK zit
df.loc[(df.woonplaats == 'Amsterdam') & (df.partij == 'DENK')]
```

```
Welke van de kandidaten wonen in Rotterdam, Amsterdam en Wassenaar?
df.loc[df['woonplaats'].isin(['Rotterdam', 'Amsterdam', 'Wassenaar'])].head()
je kan dit ook doen met 'partij' en dan bijvoorbeeld 'DENK', 'VVD'.
```

```
Geef een lijst met partijen die een 'naam' (Peter) als kandidaat hebben?
df['partij'].loc[(df['voornaam'] == 'naam')].unique()
```

```
Zoek naar kandidaten waar achternaam met een K begint
df[['voornaam', 'achternaam', 'partij']].loc[(df['achternaam'].str.startswith('K'))].head()
```

```
Stel dat de VVD 20 zetels haalt, hoeveel mannelijke VVD'ers komen er dan in de Kamer?
Maak een dataframe met X aantal namen van de kandidatenlijst van een speciefieke partij
df_vvd = df.loc[(df['partij'] == 'VVD ')].head(peiling['Zetels'].loc[(peiling['Partij'] == 'VVD')].values.item())
df_vvd

Tel vervolgens de gegeven conditie binnen dit nieuwe Dataframe
len(df_vvd.loc[(df_vvd['geslacht'] == 'man')])

Plotten van het dataframe
df_vvd['geslacht'].value_counts().plot(kind='pie' title='aantal mannen en vrouwen in de kamer VVD')
```

#Hoeveel mannen en vrouwen zijn in kandidatenlijst van Amsterdam?

STATISTICS

These can all be applied to a series as well.

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

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.

Waarom is het zo dat er alsnog NaN waarden tussen zitten in de kolommen van de Peilingwijzer Dataframe? Vervang de NaN waarden met 0 voor alle kolommen behal de kolom 'Datum' in een oneliner en laat de eerste 5 regels zien.

Er zullen NaN waarden bestaan de kandidaten van partijen die niet genoemd zijn in de peilingwijzer, zullen nog steeds in het Dataframe staan.

```
df_join.update(df_join[['Percentage', u'PercentageLaag', u'PercentageHoog', u'Zetels',

df_join.head(5)
```

Hoeveel partijen heb je op de kandidatenlijst van Amsterdam? En hoeveel partijen worden genoemd door de peilingwijzer?

```
print len(set(df_peiling)), 'genoemd door de peilingwijzer en', len(set(df_kandidaten)), 'op de kandidatenlijst 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')
```

Maak een kruistabel waarin je het gender tegen de woonplaats plaatst. Laat de totalen zien zet en laat de eerste 5 regels van deze kruistabel zien. Het resultaat moet er als volgt uitzien:

```
cross_gender_woonplaats = pd.crosstab(df_kandidatenlijst_adam.woonplaats, df_kandidatenlijst_adam.gender, margins = True)
print cross_gender_woonplaats.sort_values(by = 'All', ascending = False).head(5)
```

Maak een kruistabel waarin je het gender tegen de woonplaats plaatst. Laat de totalen zien zet en laat de eerste 5 regels van deze kruistabel zien. Het resultaat moet er als volgt uitzien:

```
cross_gender_woonplaats = pd.crosstab(df_kandidatenlijst_adam.woonplaats, 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 passend titel.

```
counter = df_kandidatenlijst_adam.woonplaats.value_counts()
Value_counts sorteert
counter = counter[:20]
```

Maak van deze verhouding mannen op vrouwen ook een pie chart met een passende titel.

```
counter = df_kandidatenlijst_adam.gender.value_counts()
counter.plot(kind = 'pie', title = 'Verhouding mannen tot vrouwen binnen de PVV')
```

En hoeveel mannen? Wat is de relatieve verhouding van mannen tot vrouwen Rond dit af op 2 decimalen achter de komma.

```
vrouwen = len(df_kandidatenlijst_adam[(df_kandidatenlijst_adam.partij == 'PVV (Partij voor de Vrijheid)')&(df_kandidatenlijst_adam.gender == 'v')])

mannen = len(df_kandidatenlijst_adam[(df_kandidatenlijst_adam.partij == 'PVV (Partij voor de Vrijheid)')&(df_kandidatenlijst_adam.gender == 'm')])

print vrouwen, ' vrouwen en', mannen, 'mannen op de kandidatenlijst van Amsterdam.
```

Beperk je bij de kandidatenlijst tot de mensen die verkiesbaar zijn voor de kiesring Amsterdam. Als de PVV 20 zetels zouden behalen, hoeveel vrouwen zouden er dan in de kamer komen volgens de kandidatenlijst van Amsterdam?

```
df_kandidatenlijst_adam = df_kandidatenlijst[df_kandidatenlijst.kieskrin == 'Amsterdam']

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.ge
```