In [1]:

```
! pip install PyAthena

import io
import boto3
import pandas as pd
import os
from pyathena import connect
from pyathena.pandas.util import as_pandas
from pyathena.pandas.cursor import PandasCursor

## connect Athena with s3 & test
cursor = connect(s3_staging_dir='s3://query-results-bucket-athena-2021/',region_name='ap-northeast-2',cursor_class=PandasCursor).cursor()
```

```
Requirement already satisfied: PyAthena in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (2.3.0)
Requirement already satisfied: boto3>=1.4.4 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from PyAthena) (1.17.83)
Requirement already satisfied: botocore>=1.5.52 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from PyAthena) (1.20.83)
Requirement already satisfied: tenacity>=4.1.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from PyAthena) (7.0.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from boto3>=1.4.4->PyAthena) (0.10.0)
Requirement already satisfied: s3transfer<0.5.0,>=0.4.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from boto3>=1.4.4->PyAthena) (0.4.2)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (1.26.5)
Requirement already satisfied: six>=1.5 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from python-dateutil<3.0.0,>=2.1->botocore>=1.5.52->PyAthena) (1.15.0)
```

In [2]:

```
################################## prameters   :        ###################################

start_yr = '2005'
end_yr = '2018'

start_date = start_yr + '-01-01'
end_date = end_yr + '-12-31'

#ipc_n = '500'
#ranking = int(ipc_n)
# 2nd Filter (confusion) parameter
#top_rank_m = 100
# 3rd Filter (assignment) paramenter
#top_rank_n = 100

top_rank = 500
material_ipc_data = pd.read_csv('data/material_ipc.csv')

sub_class_ipc = set(list(material_ipc_data['Sub_class IPC']))
```

In [3]:

```
###   :
import time
start = time.time()
```

In [4]:

```
############################################## step 01.    IPC   data gathering ##########################################\n",

def raw_data_ipc(start_yr, end_yr, sub_class_ipc) :
    query1_1 = "SELECT dataset1.new_ipc_code AS IPC, dataset1.app_year AS application_year, dataset1.app_no AS application_num, \
            dataset1.applicant1 AS applicant1, dataset1.applicant2 AS applicant2, dataset1.neo_company_univ1 AS neo_company_univ1, dataset1.neo_company_univ2 AS neo_company_univ2 \
            FROM tm_database.df_us_datamart_assignee_parquet AS dataset1 \
            WHERE dataset1.app_year >= '"+start_yr+"' and dataset1.app_year <= '"+end_yr+"' and dataset1.new_ipc_code LIKE '%"+sub_class_ipc+"%' "

    return cursor.execute(query1_1).as_pandas()

## pivoting

saved_file_name1="data/" + start_yr + "_to_" + end_yr + "_sub_class_ipc_no_"+ str(len(sub_class_ipc)) + "_raw_data_of_new_dataset_20210608.csv"
```

```
agg_raw_data = pd.DataFrame([], columns =['IPC', 'application_year', 'application_num', 'applicant1', 'applicant2', 'neo_company_univ1', 'neo_company_univ2'])

if os.path.isfile(saved_file_name1) == False:

    i = 0

    for sci in sub_class_ipc:

        if len(sci) >= 4:
            raw_data = raw_data_ipc(start_yr, end_yr, sci)
            agg_raw_data = pd.concat([agg_raw_data, raw_data])

        print("%d th sub_class_ipc data gathering completed" %(i+1))

        i = i + 1

    agg_raw_data.to_csv(saved_file_name1)

else:

    agg_raw_data = pd.read_csv(saved_file_name1,index_col=0)
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/numpy/lib/arraysetops.py:580: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  mask |= (ar1 == a)
```

In [5]:

```
top_rank_ipc = agg_raw_data['IPC'].value_counts()
top_ipc = list(top_rank_ipc.index)[0:top_rank]
year=list(range(int(start_yr), int(end_yr)+1))
```

In [6]:

```
agg_raw_data.head()
```

Out[6]:

	IPC	application_year	application_num	applicant1	applicant2	neo_company_univ1	neo_company_univ2
0	H01B7/00	2018	16611284	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.
1	H01B7/18	2018	16611284	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.
2	H01B3/30	2018	16611284	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.
3	H01B13/02	2018	16611284	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.
4	H01B11/04	2018	16611284	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.	AUTONETWORKS TECHNOLOGIES, LTD.	SUMITOMO WIRING SYSTEMS, LTD.

In [7]:

```
######################################################## yearly patents per ipc ########################################
yearly_patents_per_ipc = pd.DataFrame([],index=top_ipc, columns=year)

for ipca in top_ipc:
    temp_data = agg_raw_data.loc[agg_raw_data['IPC']==ipca, ['application_num', 'application_year']].drop_duplicates(subset=['application_num'])
    temp_data = temp_data ['application_year'].value_counts()

    year_list = list(temp_data.index)

    for yeara in year:
        if yeara in year_list:
            yearly_patents_per_ipc.loc[ipca, yeara] = temp_data[yeara]
        else:
            yearly_patents_per_ipc.loc[ipca, yeara] = 0
```

In [8]:

```
yearly_patents_per_ipc.head()
```

Out[8]:

	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
A61K9/00	193	254	439	457	591	645	704	724	1062	1759	2962	4334	4557	4189
A61K39/395	1348	1828	1910	1996	1942	1979	1746	1379	904	774	949	1009	1064	1021
A61K45/06	3	0	4	4	6	63	621	1287	2689	2714	2679	2579	2691	2777
C12N15/82	694	727	731	495	792	1018	740	963	1274	1345	1342	1372	1314	701
C07H21/04	2444	2696	1875	1378	1484	1446	836	421	42	37	53	64	110	123

In [9]:

```
#        (\"   \"                    )
#                        (yearly applicants per ipc)

modi_ipc_yr_applicant_filename = "data/" + start_yr + "_to_" + end_yr + "_sub_class_ipc_no_"+ str(len(sub_class_ipc)) + "_modi_ipc_yr_applicant_
name.csv"

garbage_company_name=["incorporation", "incorporated", "inc.", "corporation", "corp.", "company", "co.", "limited", "ltd", "ltd.", "plc", "plc.", \
                      "llc", "llc.", "unlimited partnership", "ul", "ul.", "u.l.", "u. l.", "limited partnership", "lp.", "l.p.", "l. p.", "n. v.", \
                      "n.v.", "l.l.c.", "societe", "s.a.", "sa.", "s. a.", "s a.", "responsabilite", "collectif limite", \
                      "commandite", "corporacion", "sociedad", "sdrl.", "s.d.r.l.", "s.l.", "s.l", "s. l.", "sl.", "compania en comandita", \
                      "societa", "s.p.a.", "s. p. a.", "s.r.l.", "dionicko drustvo", "sdn bhd",\
                      "s. r. l.", "accomandita","osakeyhtio", "o.y.", "o. y.", "o y.", "o. y", "aktiebolag", "a.b.", "ab.", \
                      "kommanditbolag", "aktiengesellschaft", "ag.", "a.g.", "gmbh", "egmbh", "partnerschaft", "kommanditgesellschaft", \
                      "k.g.", "kg.", "kabushiki kaisha", "kabushikikaisha", "k.k.", "k. k.", "kk.", "k. k", "kabushiki", "kaisha", "l. p.", "a/s"]

acronym_company_name=["ab", "oy", "o y", "ag", "ohg", "kg", "kk", "snc", "spa", "n v", "sa", "inc", "corp", "lp", "sdrl", "co"]

garbage_univ_institute_name=["university", "college", "school", "foundation", "academy", "academies", "center", "laboratory", "laboratories", "lab." "as
sociation",\
                             "universite", "faculte", "institute", "centre", "universidad", "facultad", "instituto", "centro", "stazione", "istituto", "istituti",\
                             "yliopisto", "opisto",  "korkeakoulu", "instituutti", "station", "seminar", "anstalt", 'e.v.', 'e.v']


if os.path.isfile(modi_ipc_yr_applicant_filename) == False:

    ipc_yr_applicant = agg_raw_data.fillna(-1)


    ind_ipc = list(set(list(ipc_yr_applicant['IPC'])))

    com_univ1 = list(ipc_yr_applicant['neo_company_univ1'])
    com_univ2 = list(ipc_yr_applicant['neo_company_univ2'])

    new_com_univ1_list=[]
    new_com_univ2_list=[]

    new_splt_com1=[]
    new_splt_com2=[]

    com1_type=[]
    com2_type=[]

    i=0

    company_count = 0
    university_count = 0
    individual_count = 0

    while i < len(com_univ1):
        k = 0
        j = 0
        std_com =[]

        if com_univ1[i] != -1:
            new_com1 = com_univ1[i].lower()

            for garbage in garbage_company_name:
                garbage_word = garbage.lower()

                if garbage_word in new_com1:
                    new_com1 = new_com1.replace(garbage_word,'').strip()
                    new_com1 = new_com1.replace(',','').strip()
                    k = k + 1

            for garbage in acronym_company_name:
                garbage_word = " " + garbage.lower()

                if garbage_word in new_com1 and k == 0:
```

```
            new_com1 = new_com1.replace(garbage_word,'').strip()
            new_com1 = new_com1.replace(',','').strip()
            k = k + 1

    if k == 0 :
        for garbage in garbage_univ_institute_name:
            garbage_word = " " + garbage.lower()

            if garbage_word in new_com1:
            #  don't delete the words "university", and etc.
                new_com1 = new_com1.strip()
                new_com1 = new_com1.replace(',','').strip()
                j = j + 1

    new_com1 = new_com1.replace('.','').strip()
    new_split_com1 = new_com1.strip().split()

    for split_com in new_split_com1:
        split_com = split_com.strip()
        std_com.append(split_com)

    normalized_company =" ".join(std_com)
    new_com_univ1_list.append(normalized_company.upper().strip())

else:
    new_com_univ1_list.append("")

if k > 0:
    com1_type.append('company')
    company_count = company_count + 1
elif j > 0:
    com1_type.append('university')
    university_count = university_count + 1
else:
    com1_type.append('individual')

std_com =[]
k = 0
j = 0

if com_univ2[i] != -1:
    new_com2 = com_univ2[i].lower()

    for garbage in garbage_company_name:
        garbage_word = garbage.lower()

        if garbage_word in new_com2:
            new_com2 = new_com2.replace(garbage_word,'').strip()
            new_com2 = new_com2.replace(',','').strip()
            k = k + 1

    for garbage in acronym_company_name:
        garbage_word = " " + garbage.lower()

        # "co" is included in acronym, but it could be skipped when garbage data first deleted
        if garbage_word in new_com1 and k == 0:
            new_com2 = new_com2.replace(garbage_word,'').strip()
            new_com2 = new_com2.replace(',','').strip()
            k = k + 1

    if k == 0:
        for garbage in garbage_univ_institute_name:
            garbage_word = " " + garbage.lower()

            if garbage_word in new_com2:
            #  don't delete the words "university", and etc.
                new_com2 = new_com2.strip()
                new_com2 = new_com2.replace(',','').strip()
                j = j + 1

    new_com2 = new_com2.replace('.','').strip()
    new_split_com2 = new_com2.strip().split()

    for split_com in new_split_com2:
        split_com = split_com.strip()
        std_com.append(split_com)

    normalized_company =" ".join(std_com)
    new_com_univ2_list.append(normalized_company.upper().strip())

else:
    new_com_univ2_list.append("")
```

```
        if k > 0:
            com2_type.append('company')
            company_count = company_count + 1
        elif j > 0:
            com2_type.append('university')
            university_count = university_count + 1
        else:
            com2_type.append('individual')

        i = i + 1

    ipc_yr_applicant['new_company_univ1'] = new_com_univ1_list
    ipc_yr_applicant['type_of_new_company_univ1'] = com1_type
    ipc_yr_applicant['new_company_univ2'] = new_com_univ2_list
    ipc_yr_applicant['type_of_new_company_univ2'] = com2_type

    # calculating yearly applicants per ipc

    modi_ipc_yr_applicant = ipc_yr_applicant

    pat_data = modi_ipc_yr_applicant.fillna(-1)

    pat_data.to_csv(modi_ipc_yr_applicant_filename)

else:

    pat_data = pd.read_csv(modi_ipc_yr_applicant_filename)


#IPC = list(set(list(pat_data['IPC'])))
IPC = top_ipc

saved_file_name2="data/" + start_yr + "_to_" + end_yr + "_sub_class_ipc_no_"+ str(len(sub_class_ipc)) + "_top_"+str(top_rank)+"_yearly_applicants_per_ipc.csv"

temp_modi = pd.DataFrame([])

for ipca in IPC:
    modi_top_ipc_data = pat_data.loc[pat_data['IPC']==ipca]
    temp_modi = pd.concat([temp_modi,modi_top_ipc_data])

pat_data = temp_modi

if os.path.isfile(saved_file_name2) == False:

    yearly_applicants_per_ipc=pd.DataFrame([],index=IPC)

    temp_data = pd.DataFrame([])
    target_data = pd.DataFrame([])
    pat_data_temp= pd.DataFrame([])

    pat_data_temp = pat_data_temp.append(pat_data.loc[pat_data['type_of_new_company_univ1'] == 'company']\
                                .rename( columns={"new_company_univ1":"new_company_univ"}))
    pat_data_temp = pat_data_temp.append(pat_data.loc[pat_data['type_of_new_company_univ1'] == 'university']\
                                .rename( columns={"new_company_univ1":"new_company_univ"}))

    pat_data_temp = pat_data_temp.append(pat_data.loc[pat_data['type_of_new_company_univ2'] == 'company']\
                                .rename( columns={"new_company_univ2":"new_company_univ"}))

    pat_data_temp = pat_data_temp.append(pat_data.loc[pat_data['type_of_new_company_univ2'] == 'university']\
                                .rename( columns={"new_company_univ2":"new_company_univ"}))

    temp_data = pat_data_temp[['IPC','application_year', 'new_company_univ']].reset_index(drop=True)
    temp_data = temp_data.drop_duplicates()

    yearly_applicants_list = temp_data.groupby(['IPC', 'application_year'])['new_company_univ'].apply(list) \
                        .reset_index(name='company_univ_list')

    for idx, ipca in yearly_applicants_list.iterrows():
        yearly_applicants_per_ipc.loc[ipca[0],ipca[1]]= len(ipca[2])


    yearly_applicants_per_ipc.to_csv(saved_file_name2)

else:

    yearly_applicants_per_ipc=pd.read_csv(saved_file_name2, index_col=0)
```

In [10]:

```
####
```

```
yearly_patents_per_ipc = yearly_patents_per_ipc.fillna(0)
yearly_applicants_per_ipc = yearly_applicants_per_ipc.fillna(0)

yearly_patents = yearly_patents_per_ipc
yearly_applicants = yearly_applicants_per_ipc

sorted_yearly_patents = yearly_patents.sum(axis=1).sort_values(ascending=False)
sorted_ipc = list (sorted_yearly_patents.index)[0:top_rank]
```

In [11]:

```
# yearly_applicants  yearly_patents  ipc          (Query   order by                    IPC          )
set(yearly_applicants.index).difference(set(yearly_patents.index))
```

Out[11]:

```
set()
```

In [12]:

```
# yearly_applicants  yearly_patents  ipc
set(yearly_patents.index).difference(set(yearly_applicants.index))
```

Out[12]:

```
set()
```

In [13]:

```
#### Calculating accumulated Pantets and Applicants for Growth index

period = int(end_yr) - int(start_yr) +1
averaged_patents = sorted_yearly_patents.iloc[0:top_rank].sum() / len(sorted_ipc) / period
displayed_applicants=pd.DataFrame([], index=sorted_ipc, columns=yearly_applicants.columns)
displayed_patents=pd.DataFrame([], index=sorted_ipc, columns=yearly_patents.columns)

for ipc in sorted_ipc:
    displayed_applicants.loc[ipc,:]=yearly_applicants.loc[ipc]
    displayed_patents.loc[ipc,:]=yearly_patents.loc[ipc]

ipc = sorted_ipc


ipc_statistics = ['accumulated patents', 'accumulated applicants', 'averaged patents', \
                  'averaged applicants', 'averaged weighted CAGR of patents', \
                  'averaged weighted CAGR of applicants']

ipc_analysis_data=pd.DataFrame([],index=sorted_ipc, columns=ipc_statistics)

ipc_analysis_data['accumulated patents']=list(round(displayed_patents.sum(axis=1),2))
ipc_analysis_data['accumulated applicants']=list(round(displayed_applicants.sum(axis=1),2))
ipc_analysis_data['averaged patents']=list(round(displayed_patents.mean(axis=1),2))
ipc_analysis_data['averaged applicants']=list(round(displayed_applicants.mean(axis=1),2))

weight = list(range(1, len(year)))
```

In [14]:

```
#### CAGR function

def CAGR (initial, final, period):
    cagr = pow((final/initial),(1/period)) - 1
    return round(cagr,3)
```

In [15]:

```
len(sorted_ipc)
```

Out[15]:

```
500
```

In [16]:

```
#### Calculating weighted CAGR of pantets and applicants for Growth index
```

```
i = 0
while i < len(ipc):
    ipc_patents = list(yearly_patents.loc[ipc[i]])
    ipc_applicants = list(yearly_applicants.loc[ipc[i]])

    j = 1

    weight_accum_cagr1 = 0
    weight_accum_cagr2 = 0

    while j < len (ipc_patents):
        if ipc_patents[0]==0: ipc_patents[0]=1       ###       0  ,
        if ipc_applicants[0]==0: ipc_applicants[0]=1  ###       0  ,

        cagr1 = CAGR(ipc_patents[0], ipc_patents[j], j)
        cagr2 = CAGR(ipc_applicants[0], ipc_applicants[j], j)

        weight_accum_cagr1 = weight[j-1]*cagr1 + weight_accum_cagr1
        weight_accum_cagr2 = weight[j-1]*cagr2 + weight_accum_cagr2

        j = j + 1

    weight_accum_cagr1 = weight_accum_cagr1/sum(weight)
    weight_accum_cagr2 = weight_accum_cagr2/sum(weight)
    ipc_analysis_data.loc[ipc[i],'averaged weighted CAGR of patents'] = round(weight_accum_cagr1,2)
    ipc_analysis_data.loc[ipc[i],'averaged weighted CAGR of applicants'] = round(weight_accum_cagr2,2)

    i = i + 1
```

In [17]:

```
saved_file_name_3 ="data/" + start_yr + "_to_" + end_yr + "_sub_class_ipc_no_"+ str(len(sub_class_ipc)) + "_top_" + str(top_rank) + "_ipc_analysis
.csv"
ipc_analysis_data.to_csv(saved_file_name_3)
```

In [18]:

```
ipc_analysis_data.head()
```

Out[18]:

	accumulated patents	accumulated applicants	averaged patents	averaged applicants	averaged weighted CAGR of patents	averaged weighted CAGR of applicants
A61K9/00	22870.0	12187.0	1633.57	870.50	0.29	0.25
A61K39/395	19849.0	7791.0	1417.79	556.50	0	-0
A61K45/06	18117.0	8468.0	1294.07	604.86	0.89	0.83
C12N15/82	13508.0	2898.0	964.86	207.00	0.04	-0.01
C07H21/04	13009.0	5170.0	929.21	369.29	-0.25	-0.2

In [19]:

```
################################## ipc picking from material top 100 ipc ##################################

x_minus_range = 500
x_plus_range = 4000
y_minus_range = 100
y_plus_range = 500

min_patent_cagr = 0.3
min_applicant_cagr = 0.3

max_patent_cagr = 0.5
max_applicant_cagr = 0.5

mask1 = (ipc_analysis_data['averaged applicants'] >= x_minus_range) & (ipc_analysis_data['averaged applicants'] <= x_plus_range)

mask2 = (ipc_analysis_data['averaged patents'] >= y_minus_range) & (ipc_analysis_data['averaged patents'] <= y_plus_range)

mask3 = (ipc_analysis_data['averaged weighted CAGR of patents'] >= min_patent_cagr) & (ipc_analysis_data['averaged weighted CAGR of patents']
< max_patent_cagr)

mask4 = (ipc_analysis_data['averaged weighted CAGR of applicants'] >= min_applicant_cagr) & (ipc_analysis_data['averaged weighted CAGR of ap
plicants'] < max_applicant_cagr)

ipc_set1 = set(list(ipc_analysis_data.loc[mask1].index))
ipc_set2 = set(list(ipc_analysis_data.loc[mask2].index))
```

```
ipc_set3 = set(list(ipc_analysis_data.loc[mask3].index))
ipc_set4 = set(list(ipc_analysis_data.loc[mask4].index))

###  patent cagr >= 20% and applicant cagr >= 20%

target_ipc = ipc_set3.intersection(ipc_set4)
target_ipc
```

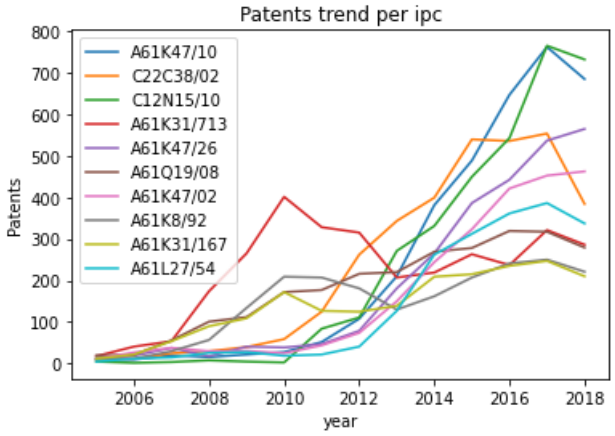
Out[19]:

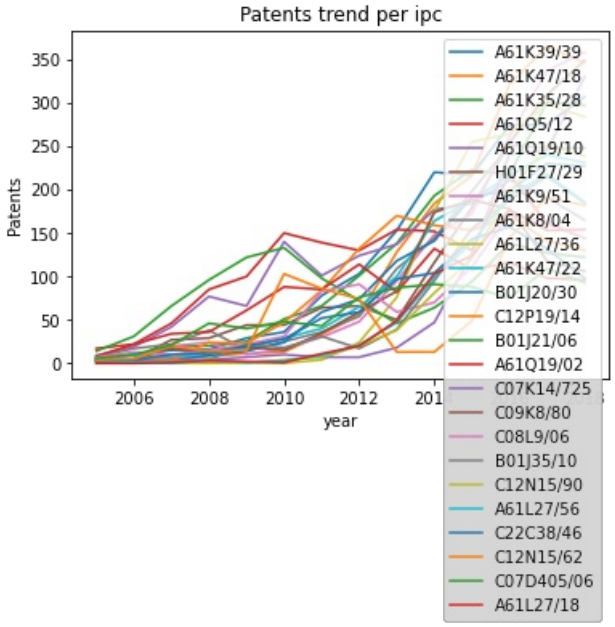
```
{'A61K31/167',
 'A61K31/713',
 'A61K35/28',
 'A61K39/39',
 'A61K47/02',
 'A61K47/10',
 'A61K47/18',
 'A61K47/22',
 'A61K47/26',
 'A61K8/04',
 'A61K8/92',
 'A61K9/51',
 'A61L27/18',
 'A61L27/36',
 'A61L27/54',
 'A61L27/56',
 'A61Q19/02',
 'A61Q19/08',
 'A61Q19/10',
 'A61Q5/12',
 'B01J20/30',
 'B01J21/06',
 'B01J35/10',
 'C07D405/06',
 'C07K14/725',
 'C08L9/06',
 'C09K8/80',
 'C12N15/10',
 'C12N15/62',
 'C12N15/90',
 'C12P19/14',
 'C22C38/02',
 'C22C38/46',
 'H01F27/29'}
```

In [20]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.patches as patches

i = 0

plt.figure()
plt.title('Patents trend per ipc')

sorted_ipc1 = list(target_ipc)

target_ipc_analysis_data = pd.DataFrame([], index = sorted_ipc1, columns = ipc_analysis_data.columns)

for ipc in sorted_ipc1:
    target_ipc_analysis_data.loc[ipc,:] =ipc_analysis_data.loc[ipc]

Sorted_Common_Trd_picked_ipc = list(target_ipc_analysis_data['accumulated patents'].sort_values(ascending=False).index)

target_ipc_analysis_data = pd.DataFrame([], index = Sorted_Common_Trd_picked_ipc, columns = ipc_analysis_data.columns)

for ipc in Sorted_Common_Trd_picked_ipc:
    target_ipc_analysis_data.loc[ipc,:] =ipc_analysis_data.loc[ipc]


for common_ipc in Sorted_Common_Trd_picked_ipc[0:10]:
    yearly_trend = list (yearly_patents.loc[common_ipc])
    plt.plot(year,yearly_trend, label=common_ipc)

plt.legend()
#plt.xticks(['2005', '2007', '2009', '2011', '2013', '2015', '2017'])
plt.xlabel('year')
plt.ylabel('Patents')
plt.show()
```

```
plt.figure()
plt.title('Patents trend per ipc')
for common_ipc in Sorted_Common_Trd_picked_ipc[10:len(Sorted_Common_Trd_picked_ipc)]:
    yearly_trend = list (yearly_patents.loc[common_ipc])
    plt.plot(year,yearly_trend, label=common_ipc)

plt.legend()
#plt.xticks(['2005', '2007', '2009', '2011', '2013', '2015', '2017'])
plt.xlabel('year')
plt.ylabel('Patents')
plt.show()
```

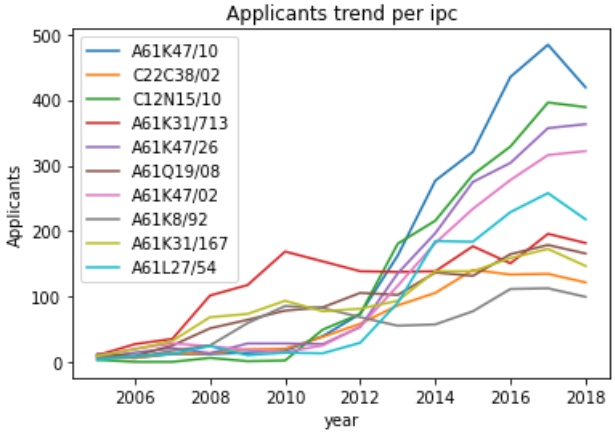


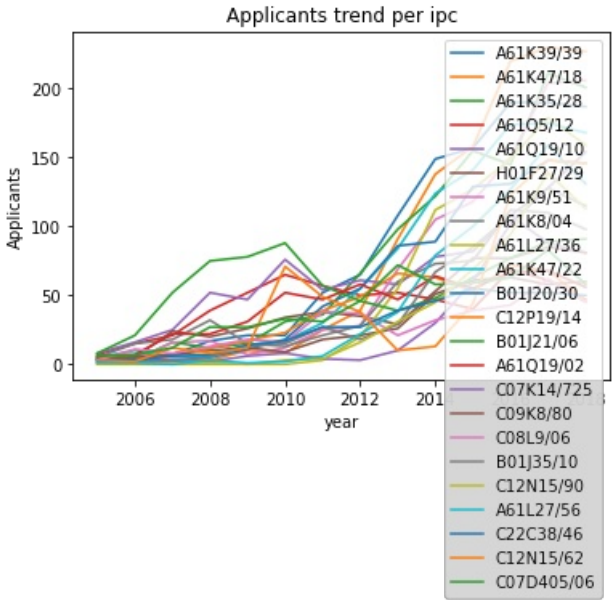


In [21]:

```
plt.figure()
plt.title('Applicants trend per ipc')

for common_ipc in Sorted_Common_Trd_picked_ipc[0:10]:
    yearly_trend = list (yearly_applicants.loc[common_ipc])
    plt.plot(year,yearly_trend, label=common_ipc)

plt.legend()
#plt.xticks(['2005', '2007', '2009', '2011', '2013', '2015', '2017'])
plt.xlabel('year')
plt.ylabel('Applicants')
plt.show()


plt.figure()
plt.title('Applicants trend per ipc')
for common_ipc in Sorted_Common_Trd_picked_ipc[10:-1]:
    yearly_trend = list (yearly_applicants.loc[common_ipc])
    plt.plot(year,yearly_trend, label=common_ipc)

plt.legend()
#plt.xticks(['2005', '2007', '2009', '2011', '2013', '2015', '2017'])
plt.xlabel('year')
plt.ylabel('Applicants')
```

```
plt.show()
```





In [22]:

```
saved_file_name_4 ="data/" + start_yr + "_to_" + end_yr + "_sub_class_ipc_no_"+ str(len(sub_class_ipc)) + "_top_" + str(top_rank) + "_pat_cagr_min" + str(min_patent_cagr)+ \
                   "_max" + str(max_patent_cagr) + "_applicant_cagr_min" + str(min_applicant_cagr) + "_max" + str(max_applicant_cagr) + "_analysis_data.csv"
target_ipc_analysis_data.to_csv(saved_file_name_4)
```

In [23]:

```
original_pat_data = pd.read_csv(modi_ipc_yr_applicant_filename)

original_pat_data = original_pat_data.fillna(-1)

target_ipc1 = list(target_ipc_analysis_data['accumulated patents'].sort_values(ascending=False).index)

top_applicants_per_ipc = pd.DataFrame([],columns=target_ipc1)

top_app_rank = 40

top_applicant_list = []



for tg_ipc in target_ipc1:
    target_ipc_applicants = original_pat_data.loc[(original_pat_data['IPC']==tg_ipc) & (original_pat_data["new_company_univ1"]!=-1) ,["new_company_univ1", "new_company_univ2"]]
    top_applicants = target_ipc_applicants["new_company_univ1"].value_counts()
    top_applicants_list = top_applicants.index[0:top_app_rank]
    top_applicants_per_ipc[tg_ipc]=top_applicants_list
```

In [24]:

```
target_ipc_analysis_data
```

Out[24]:

	accumulated patents	accumulated applicants	averaged patents	averaged applicants	averaged weighted CAGR of patents	averaged weighted CAGR of applicants
A61K47/10	3437	2290	245.5	163.57	0.38	0.31
C22C38/02	3316	905	236.86	64.64	0.46	0.34
C12N15/10	3310	1939	236.43	138.5	0.45	0.4
A61K31/713	3129	1740	223.5	124.29	0.44	0.42
A61K47/26	2666	1831	190.43	130.79	0.38	0.37
A61Q19/08	2557	1317	182.64	94.07	0.35	0.3
A61K47/02	2333	1650	166.64	117.86	0.36	0.33
A61K8/92	2054	875	146.71	62.5	0.39	0.32
A61K31/167	1966	1311	140.43	93.64	0.41	0.37
A61L27/54	1953	1287	139.5	91.93	0.47	0.44
A61K39/39	1738	1175	124.14	83.93	0.48	0.4
A61K47/18	1697	1162	121.21	83	0.39	0.37
A61K35/28	1661	1115	118.64	79.64	0.47	0.43
A61Q5/12	1648	622	117.71	44.43	0.31	0.31
A61Q19/10	1643	834	117.36	59.57	0.44	0.33
H01F27/29	1541	588	110.07	42	0.47	0.38
A61K9/51	1530	964	109.29	68.86	0.41	0.38
A61K8/04	1412	675	100.86	48.21	0.39	0.35
A61L27/36	1384	806	98.86	57.57	0.42	0.3
A61K47/22	1323	1008	94.5	72	0.37	0.35
B01J20/30	1298	870	92.71	62.14	0.38	0.36
C12P19/14	1245	520	88.93	37.14	0.47	0.38
B01J21/06	1223	832	87.36	59.43	0.31	0.36
A61Q19/02	1080	602	77.14	43	0.39	0.32
C07K14/725	1049	513	74.93	36.64	0.41	0.45
C09K8/80	1013	404	72.36	28.86	0.48	0.33
C08L9/06	991	386	70.79	27.57	0.33	0.37
B01J35/10	975	616	69.64	44	0.48	0.42
C12N15/90	962	515	68.71	36.79	0.46	0.34
A61L27/56	922	670	65.86	47.86	0.37	0.44
C22C38/46	919	398	65.64	28.43	0.47	0.33
C12N15/62	913	679	65.21	48.5	0.4	0.39
C07D405/06	833	582	59.5	41.57	0.38	0.3
A61L27/18	798	579	57	41.36	0.48	0.37

In [25]:

```
######   :
r_time = round((time.time()-start)/60, 2)
print('Running time :', r_time, 'minutes')
```

Running time : 5.75 minutes

In [26]:

```
top_applicants_per_ipc
saved_file_name_5 ="data/" + start_yr + "_to_" + end_yr + "_sub_class_ipc_no_"+ str(len(sub_class_ipc)) + "_top_" + str(top_rank) + "_pat_cagr_mi
n" + str(min_patent_cagr)+ \
                   "_max" + str(max_patent_cagr) + "_applicant_cagr_min" + str(min_applicant_cagr) + "_max" + str(max_applicant_cagr) + "_top_applica
nts_data.csv"
top_applicants_per_ipc.to_csv(saved_file_name_5)
```

In [27]:

```
len(target_ipc_analysis_data.index)
```

Out[27]:

34

In [ ]: