

```
In [1]: # Importing necessary libraries and modules
import os
import datetime
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline
import matplotlib.dates as dates
```

```
In [2]: # Load data
countries = pd.read_csv("../data/village_savings/Countries.csv")
locations = pd.read_csv("../data/village_savings/Locations.csv")
regions = pd.read_csv("../data/village_savings/Regions.csv")
members = pd.read_csv("../data/village_savings/Members.csv") #tab seperated csv
group_members = pd.read_csv("../data/village_savings/GroupMembers.csv")
groups = pd.read_csv("../data/village_savings/SavingsGroups.csv")
```

```
In [3]: # Count active groups.
groups.groupby([groups.IsActive]).GroupID.count()
```

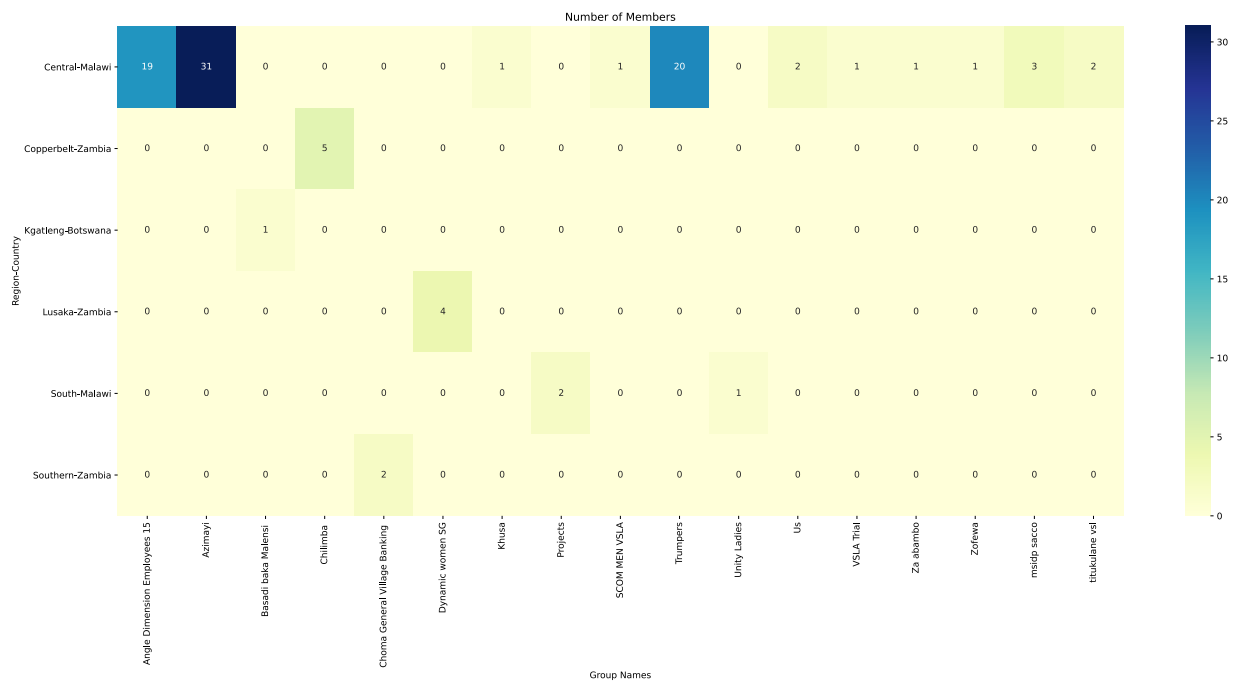
```
Out[3]: IsActive
1      132
Name: GroupID, dtype: int64
```

```
In [4]: # Join data: Location Name, Region Name, and Country Name
groups = groups.merge(locations, on="LocationId").merge(regions, on="RegionId")
```

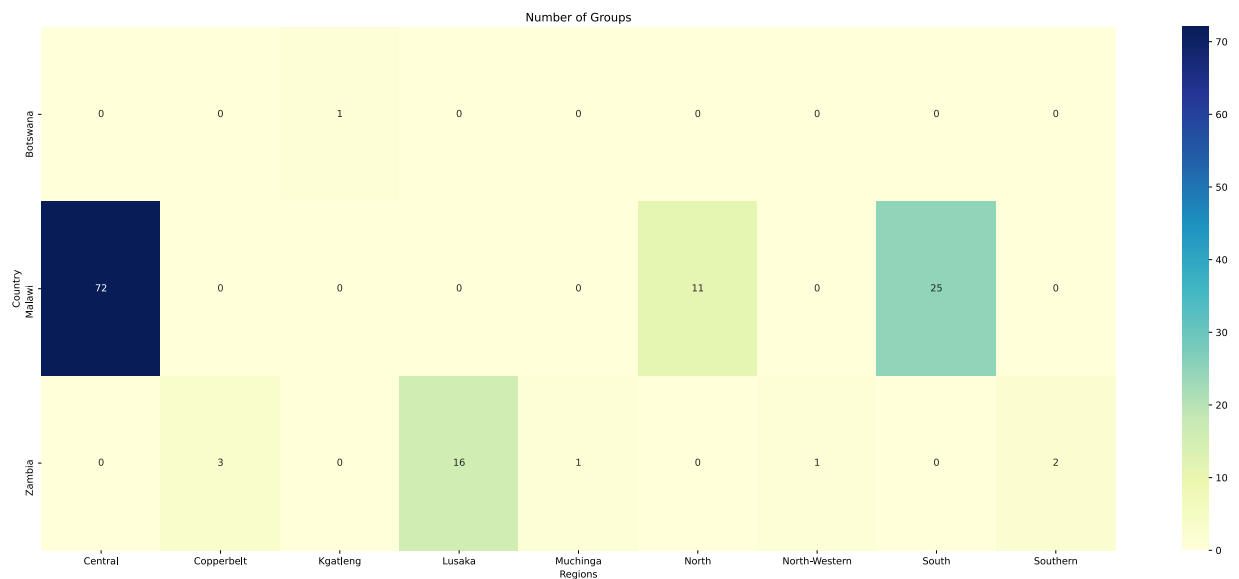
```
In [5]: # Get Member of groups
# groups[groups.GroupStatusCode != 'E']
groups['NumberOfMembers'] = 0
for idx, row in groups.iterrows():
    if row.IsKhusaServiceOffered == 1:
        groups.at[idx, 'NumberOfMembers'] = group_members[lambda x: x.GroupID
```

```
In [6]: # Create pivot table using RegionName and CountryName columns.
group_pivot_table = pd.pivot_table(groups[groups.NumberOfMembers > 0], values
```

```
In [7]: #Create graph of number of memebers in region and country
fig,ax = plt.subplots(figsize=(25, 10))
sns.heatmap(group_pivot_table, annot=True,ax=ax,cmap="YlGnBu", )
plt.ylabel('Region-Country')
plt.xlabel('Group Names')
plt.title("Number of Members")
plt.show()
```

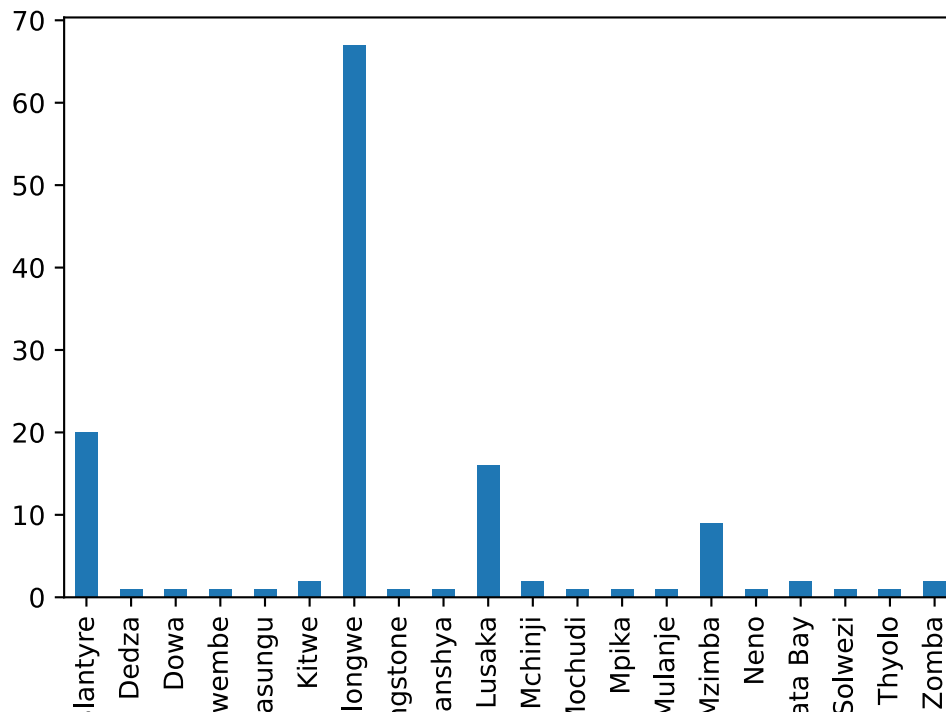


```
In [8]: # Create pivot table using RegionName and CountryName columns.
group_counts = groups.groupby([groups.CountryName, groups.RegionName]).GroupID
group_count_pvt_table = pd.pivot_table(group_counts, values='GroupID', index=
fig, ax = plt.subplots(figsize=(25, 10))
sns.heatmap(group_count_pvt_table, annot=True, ax=ax, cmap="YlGnBu", )
plt.ylabel('Country')
plt.xlabel('Regions')
plt.title("Number of Groups")
plt.show()
```



```
In [9]: # Number of Groups based Location
groups.groupby([groups.LocationName]).GroupID.count().plot.bar()
```

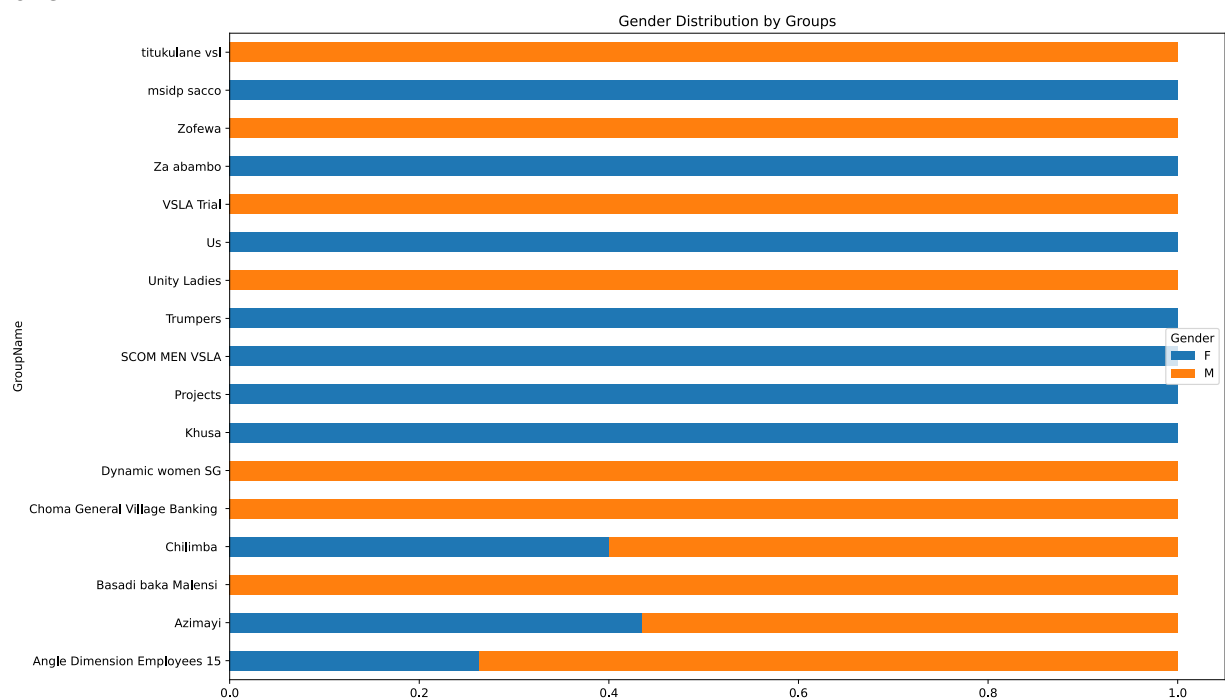
```
Out[9]: <AxesSubplot:xlabel='LocationName'>
```



```
In [10]: group_members = group_members.merge(members[['MemberID', 'Gender', 'DateOfBirth']])
```

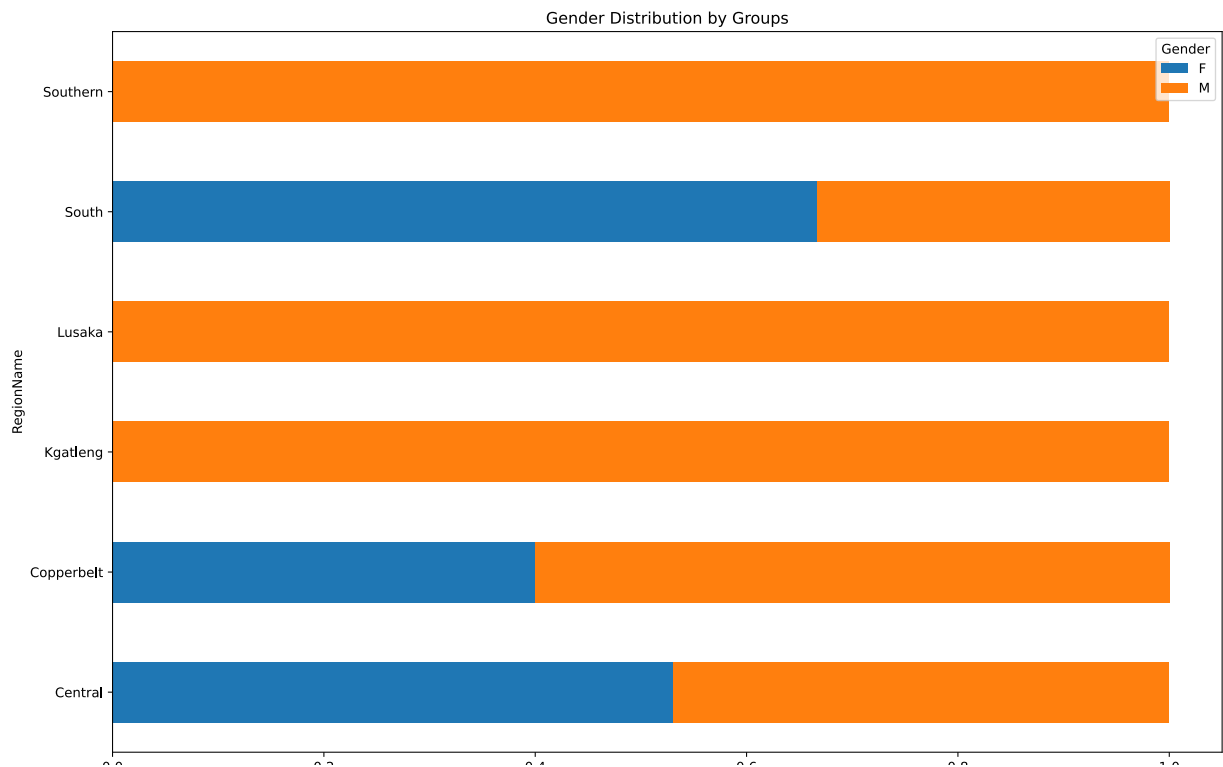
```
In [11]: # Gender distribution based on Groups
# pd.crosstab([group_members.GroupName], group_members.Gender).plot.bar()
group_members.dropna(subset=['Gender']).groupby(group_members.GroupName).Gender
```

```
Out[11]: <AxesSubplot:title={'center': 'Gender Distribution by Groups'}, ylabel='Group Name'>
```



```
In [12]: group_members.dropna(subset=['Gender']).groupby([group_members.RegionName]).Gender
```

```
Out[12]: <AxesSubplot:title={'center': 'Gender Distribution by Groups'}, ylabel='Region Name'>
```



```
In [13]: threshold = 0.4
gender_by_locations = group_members.groupby(group_members.LocationName).Gender
location_names = gender_by_locations.index.get_level_values(0).to_numpy()
biased_locs=[]
for location in location_names:
    try:
        bias = abs(gender_by_locations.loc[location, 'F'] - gender_by_locations.loc[location, 'M'])
        if bias > threshold:
            biased_locs.append([location, round(bias, 2)])
    except KeyError:
        biased_locs.append([location, 1.0])
biased_locs = np.array(biased_locs)
fig, ax = plt.subplots(figsize=(10,10))
plt.scatter(biased_locs[:, 0], biased_locs[:, 1], s=np.pi*9, vmin=0.0, vmax=1)
plt.title('Locations that have Gender Bias')
plt.xlabel('Location')
plt.ylabel('Bias')
```

Out[13]: Text(0, 0.5, 'Bias')



```
In [14]: group_members.DateOfBirth = pd.to_datetime(group_members.DateOfBirth, format=
group_members['Age'] = datetime.datetime.now().year - group_members.DateOfBir
# group_members.fillna({'Age': 0}, inplace=True)
group_members
```

```
Out[14]:
```

	MemberGroupID	MemberID	GroupID	IsMainMember	MemberStatusCode	DateCreated	Gender
0	1.0	1.0	1	1.0	A	24-09-18 15:20:10	M
1	2.0	2.0	1	1.0	A	24-09-18 16:04:23	M
2	3.0	3.0	1	1.0	A	24-09-18 16:06:17	M
3	4.0	4.0	1	1.0	A	24-09-18 16:08:07	M
4	5.0	5.0	1	1.0	A	24-09-18 16:10:24	M
...	...	...	...	...	...	...	...
237	NaN	NaN	124	NaN	NaN	NaN	NaN
238	NaN	NaN	126	NaN	NaN	NaN	NaN
239	NaN	NaN	56	NaN	NaN	NaN	NaN

	MemberGroupID	MemberID	GroupID	IsMainMember	MemberStatusCode	DateCreated	Gender
240	NaN	NaN	88	NaN	NaN	NaN	NaN
241	18.0	3.0	12	1.0	A	02-10-18 10:07:12	M

```
In [15]: group_members[group_members.Age > 0].groupby([group_members.GroupName]).Age.agg
```

```
Out[15]:
```

	GroupName	Lowest Age	Highest Age	Average
0	Angle Dimension Employees 15	4.0	61.0	34.894737
1	Azimayi	2.0	31.0	4.080645
2	Basadi baka Malensi	34.0	34.0	34.000000
3	Chilimba	26.0	61.0	36.600000
4	Choma General Village Banking	37.0	39.0	38.000000
5	Dynamic women SG	38.0	50.0	46.250000
6	Khusa	34.0	34.0	34.000000
7	Projects	30.0	30.0	30.000000
8	SCOM MEN VSLA	35.0	35.0	35.000000
9	Trumpers	23.0	40.0	32.500000
10	Unity Ladies	30.0	30.0	30.000000
11	Us	4.0	31.0	17.500000
12	VSLA Trial	15.0	15.0	15.000000
13	Za abambo	26.0	26.0	26.000000
14	Zofewa	42.0	42.0	42.000000
15	msidp sacco	3.0	42.0	22.666667
16	titukulane vsl	21.0	21.0	21.000000

```
In [16]: group_members[group_members.Age > 0].groupby([group_members.RegionName]).Age.agg
```

```
Out[16]:
```

	RegionName	Lowest Age	max	mean
0	Central	2.0	61.0	16.486726
1	Copperbelt	26.0	61.0	36.600000
2	Kgatleng	34.0	34.0	34.000000
3	Lusaka	38.0	50.0	46.250000
4	South	30.0	30.0	30.000000
5	Southern	37.0	39.0	38.000000

```
In [17]: age_groups = pd.qcut(group_members[group_members.Age > 0].Age, 4, duplicates=
group_members[group_members.Age > 0].groupby([group_members.RegionName, age_g
```

```
Out[17]:
```

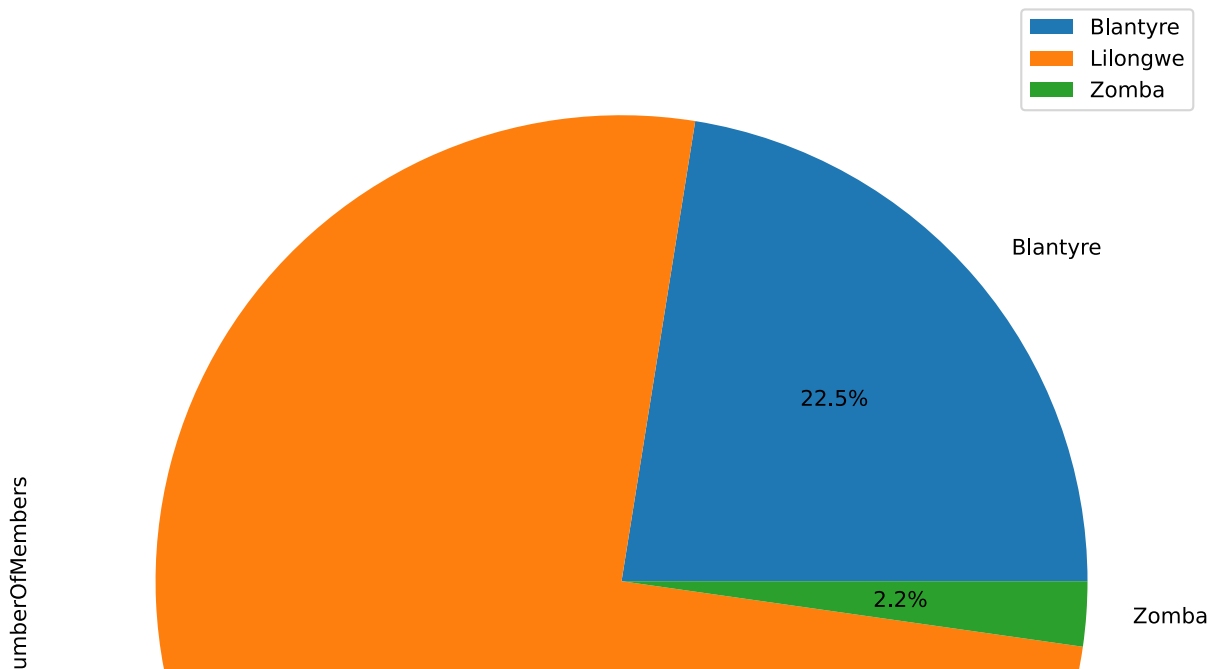
RegionName	Age	Gender	
Central	Below 21	F	29
		M	39
	Between 21 and 32	F	20
		M	4
Copperbelt	Above 32	F	11
		M	10
	Below 21	F	0
		M	0
Kgatlang	Between 21 and 32	F	2
		M	1
	Above 32	F	0
		M	2
Lusaka	Below 21	F	0
		M	0
	Between 21 and 32	F	0
		M	0
South	Above 32	F	0
		M	4
	Below 21	F	0
		M	0
Southern	Between 21 and 32	F	2
		M	1
	Above 32	F	0
		M	0
	Below 21	F	0
		M	0
	Between 21 and 32	F	0
		M	0
	Above 32	F	0
		M	2

Name: Age, dtype: int64

```
In [18]: major_cities = ['Lilongwe', 'Zomba', 'Mzuzu', 'Blantyre'] #major cities in Ma
groups[groups.CountryName == 'Malawi'][groups.LocationName.isin(major_cities)]
```

```
Out[18]: array([<AxesSubplot:ylabel='NumberOfMembers'>], dtype=object)
```

## Coverage of Members in Major Cities



```
In [19]: group_members.DateCreated = pd.to_datetime(group_members.DateCreated, format=
groups.DateCreated = pd.to_datetime(groups.DateCreated, format="%d-%m-%y %H:%
```

```
In [20]: groups['Period'] = groups.DateCreated.dt.strftime("%b-%Y")
group_members['Period'] = group_members.DateCreated.dt.strftime("%b-%Y")
member_annual_data = group_members.groupby([group_members.Period]).MemberID.c
annaul_group_data = groups.groupby([groups.Period]).GroupID.count().reset_ind
data = [annaul_group_data, member_annual_data]
member_annual_data.sort_values(by="Period", key=lambda x: dates.datestr2num(x
annaul_group_data.sort_values(by="Period", key=lambda x: dates.datestr2num(x)
labels = ['Groups', 'Members']
fig, ax = plt.subplots(figsize=(20, 15))
for i in range(len(data)):
    period = dates.datestr2num(data[i].Period.to_numpy())
    smooth_x = np.linspace(period.min(), period.max(), 300)
    spline = make_interp_spline(np.sort(period), data[i].Count.to_numpy(), k=3
    interp_values = spline(smooth_x).clip(0, max(spline(smooth_x)))
    max_index = data[i].Count.idxmax()
    max_period = dates.datestr2num(data[i].loc[max_index].Period)
    max_count = data[i].loc[max_index].Count

    ax.plot(dates.num2date(max_period), max_count, 'ro')
    ax.annotate('{0}'.format(max_count), xy=(dates.num2date(max_period), max_
    ax.plot(dates.num2date(smooth_x), interp_values, label=labels[i])
    ax.legend(loc="upper right")

plt.title("Annual Group and Member growth")
```

```
Out[20]: Text(0.5, 1.0, 'Annual Group and Member growth')
```



