Import Python Libraries

Libraries Used

Pandas

Data manipulation and analysis

MatPlotLib Pyplot ¶

2D plotting

Numpy

Supports large, multi-dimensional arrays and matrix manipulation and high level mathematical functions on these arrays

Scipy Stats

Hypothesis testing

In [1]:
```
# Perform library imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
```

Import Data Dictionaries to Convert Codes to Descriptions

Sources

Non-Profit Data - NCSS Data Archive: https://nccs-data.urban.org/dd2.php?close=1&form=BMF+08/2016 (https://nccs-data.urban.org/dd2.php?close=1&form=BMF+08/2016)

Region Data - https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf (https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf)

Non-Profit Method

- Created text files in json format within Visual Studio for each Data Dictionary
- Saved files to Resources folder
- Used pandas library to read the files into dataframes

In [2]:
```
# Import Level1 Data Dictionary and display : NOTE - This is only a break out
 of Public Charities
file_Level1 = "./Resources/NCSSDataDictLevel1.txt"
dict_Level1 = pd.read_json(file_Level1)
dict_Level1
```

Out[2]:

	Description1
O	Other Nonprofits
PC	Public Charity
PF	Private Foundation
U	Unknown

In [3]:
```
# Import Region csv file
region_csv = './Resources/State_region.csv'
df_region = pd.read_csv(region_csv)
df_region
```

Out[3]:

	STATE	REGION	STATE_POP	REGION_POP
0	AK	WEST	737438	77993663
1	AL	SOUTH	4887871	124753948
2	AR	SOUTH	3013825	124753948
3	AZ	WEST	7171646	77993663
4	CA	WEST	3955704	77993663
5	CO	WEST	5695564	77993663
6	CT	NORTHEAST	3572665	5611079
7	DE	SOUTH	967171	124753948
8	FL	SOUTH	21299325	124753948
9	GA	SOUTH	10519475	124753948
10	HI	WEST	1420491	77993663
11	IA	MIDWEST	3156145	68308744
12	ID	WEST	1754208	77993663
13	IL	MIDWEST	12741080	68308744
14	IN	MIDWEST	66918178	68308744
15	KS	MIDWEST	2911505	68308744
16	KY	SOUTH	4468402	124753948
17	LA	SOUTH	4659978	124753948
18	MA	NORTHEAST	6902149	5611079
19	MD	SOUTH	6042718	124753948
20	ME	NORTHEAST	1338404	5611079
21	MI	MIDWEST	9995915	68308744
22	MN	MIDWEST	5611179	68308744
23	MO	MIDWEST	6126452	68308744
24	MS	SOUTH	2986530	124753948
25	MT	WEST	1062305	77993663
26	NC	SOUTH	10383620	124753948
27	ND	MIDWEST	760077	68308744
28	NE	MIDWEST	1929268	68308744
29	NH	NORTHEAST	1356458	5611079
30	NJ	NORTHEAST	8908520	5611079
31	NM	WEST	2095428	77993663
32	NV	WEST	3034392	77993663
33	NY	NORTHEAST	19542209	5611079
34	OH	MIDWEST	11689442	68308744

	STATE	REGION	STATE_POP	REGION_POP
35	OK	SOUTH	3943079	124753948
36	OR	WEST	4190713	77993663
37	PA	NORTHEAST	12807060	5611079
38	RI	NORTHEAST	1057315	5611079
39	SC	SOUTH	5084127	124753948
40	SD	MIDWEST	882235	68308744
41	TN	SOUTH	6770010	124753948
42	TX	SOUTH	28701845	124753948
43	UT	WEST	3161105	77993663
44	VA	SOUTH	8517685	124753948
45	VT	NORTHEAST	626299	5611079
46	WA	WEST	7535591	77993663
47	WI	MIDWEST	5813568	68308744
48	WV	SOUTH	1805832	124753948
49	WY	WEST	577737	77993663

In [4]:
```
# Import Category name data
gov_data_ntmaj10_csv = "./Resources/ntmaj10_values.csv"
gov_data_ntmaj10_df = pd.read_csv(gov_data_ntmaj10_csv)
gov_data_ntmaj10_df
```

Out[4]:

	NTMAJ10	Long_Name
0	AR	Arts
1	ED	Education
2	EN	Environment
3	HE	Health
4	HU	Human Services
5	IN	International
6	PU	Public Benefit
7	RE	Religion
8	MU	Mutual Benefit
9	UN	Unknown

Import IRS Business Master Files

Source

NCSS Data Archive https://nccs-data.urban.org/data.php?ds=bmf (https://nccs-data.urban.org/data.php?ds=bmf)

File

bmf.bm1812.csv

In [5]:
```
# specify file name
gov_data_file = "./Resources/2018_BMF.csv"

# import file
gov_data = pd.read_csv(gov_data_file, low_memory=False)

# Display resulting dataframe header
gov_data.head()
```

Out[5]:

	EIN	SEC_NAME	FRCD	SUBSECCD	TAXPER	ASSETS	INCOME	NAME	ADDRESS
0	19818	3514	60	3	NaN	NaN	NaN	PALMER SECOND BAPTIST CHURCH	10 THORNDI
1	29215	NaN	60	3	NaN	NaN	NaN	ST GEORGE CATHEDRAL	523 BROADW
2	260049	NaN	60	3	NaN	NaN	NaN	CORINTH BAPTIST CHURCH	PO BOX
3	490336	NaN	60	3	NaN	NaN	NaN	EASTSIDE BAPTIST CHURCH	PO BOX 2
4	587764	NaN	60	3	NaN	NaN	NaN	IGLESIA BETHESDA INC	1 ANDOV

5 rows × 40 columns

Data Cleaning Phase

In [6]:
```
gov_data.count()
```

Out[6]:
```
EIN          1499450
SEC_NAME      404909
FRCD         1499450
SUBSECCD     1499450
TAXPER       1240366
ASSETS       1223112
INCOME       1223112
NAME         1499450
ADDRESS      1499450
CITY         1499450
STATE        1498426
NTEECONF        4518
NTEEFINAL    1499450
NAICS        1495191
ZIP5         1499353
OUTNCCS      1499450
OUTREAS         3021
RULEDATE     1499450
FIPS         1496643
FNDNCD       1499450
PMSA          506362
MSA_NECH     1175676
CASSETS       542418
CFINSRC       542418
CTAXPER       542418
CTOTREV       542418
ACCPER       1240366
RANDNUM      1499450
NTEECC       1499450
NTEE1        1499450
LEVEL4       1499450
LEVEL1       1499450
NTMAJ10      1499450
MAJGRPB      1499450
LEVEL3       1499450
LEVEL2       1499450
NTMAJ12      1499450
NTMAJ5       1499450
FILER        1499450
ZFILER       1499450
dtype: int64
```

In [7]:
```
# Select only the columns of data we need for analysis
gov_data = gov_data[["EIN", "CTOTREV", "CASSETS", "NAME", "CITY", "STATE", "NT
MAJ10", "LEVEL1", "OUTNCCS"]]
```

In [8]:
```
# check the number of data rows per column
gov_data.count()
```

Out[8]:
```
EIN         1499450
CTOTREV      542418
CASSETS      542418
NAME        1499450
CITY        1499450
STATE       1498426
NTMAJ10     1499450
LEVEL1      1499450
OUTNCCS     1499450
dtype: int64
```

In [9]:
```
# display the data read in
gov_data
```

Out[9]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10	LEV
0	19818	NaN	NaN	PALMER SECOND BAPTIST CHURCH	PALMER	MA	RE	
1	29215	NaN	NaN	ST GEORGE CATHEDRAL	SOUTH BOSTON	MA	RE	
2	260049	NaN	NaN	CORINTH BAPTIST CHURCH	HOSFORD	FL	RE	
3	490336	NaN	NaN	EASTSIDE BAPTIST CHURCH	LABELLE	FL	RE	
4	587764	NaN	NaN	IGLESIA BETHESDA INC	LOWELL	MA	RE	
...	...	...	...	...	...	...	...	
1499445	996089401	NaN	NaN	TOYO SAKUMOTO CHARITABLE TR	HONOLULU	HI	PU	
1499446	996165005	NaN	NaN	INDEPENDENT ORDER OF ODD FELLOWS	CUPERTINO	CA	MU	
1499447	998010224	NaN	NaN	HAWAII FOUNDATION FOR THE BLIND	HONOLULU	HI	HU	
1499448	998997790	NaN	NaN	CHAMPAIGN COUNTY EXTENSION EDUCATION FOUNDATION	TAYLORVILLE	IL	ED	
1499449	999009356	25283.0	44346.0	NATIONAL ASSOCIATION OF LETTER CARRIERS	HONOLULU	HI	HU	

1499450 rows × 9 columns

In [10]:
```
indexNames = gov_data[gov_data["OUTNCCS"]!="IN"].index
len(indexNames)
```

Out[10]: 3021

In [11]:
```
# drop the rows identified and show the resulting dataframe
gov_data.drop(indexNames, inplace=True)
gov_data.head()
```

Out[11]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10	LEVEL1	OUTNCCS
0	19818	NaN	NaN	PALMER SECOND BAPTIST CHURCH	PALMER	MA	RE	PC	IN
1	29215	NaN	NaN	ST GEORGE CATHEDRAL	SOUTH BOSTON	MA	RE	PC	IN
2	260049	NaN	NaN	CORINTH BAPTIST CHURCH	HOSFORD	FL	RE	PC	IN
3	490336	NaN	NaN	EASTSIDE BAPTIST CHURCH	LABELLE	FL	RE	PC	IN
4	587764	NaN	NaN	IGLESIA BETHESDA INC	LOWELL	MA	RE	PC	IN

In [12]:
```
gov_data.count()
```

Out[12]:
```
EIN         1496429
CTOTREV      541062
CASSETS      541062
NAME        1496429
CITY        1496429
STATE       1496305
NTMAJ10     1496429
LEVEL1      1496429
OUTNCCS     1496429
dtype: int64
```

In [13]:
```
# identify the indices of rows we want to eliminate and display how many rows
 are found
indexNames = gov_data[gov_data["LEVEL1"]=="O"].index
len(indexNames)
```

Out[13]:
```
348758
```

In [14]:
```
# drop the rows identified and show the resulting dataframe
gov_data.drop(indexNames, inplace=True)
gov_data.head()
```

Out[14]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10	LEVEL1	OUTNCCS
0	19818	NaN	NaN	PALMER SECOND BAPTIST CHURCH	PALMER	MA	RE	PC	IN
1	29215	NaN	NaN	ST GEORGE CATHEDRAL	SOUTH BOSTON	MA	RE	PC	IN
2	260049	NaN	NaN	CORINTH BAPTIST CHURCH	HOSFORD	FL	RE	PC	IN
3	490336	NaN	NaN	EASTSIDE BAPTIST CHURCH	LABELLE	FL	RE	PC	IN
4	587764	NaN	NaN	IGLESIA BETHESDA INC	LOWELL	MA	RE	PC	IN

In [15]:
```
# identify the indices of rows we want to eliminate and display how many rows
indexNames = gov_data[gov_data["LEVEL1"]=="U"].index
len(indexNames)
```

Out[15]: 30

In [16]:
```
# drop the rows identified and show the resulting dataframe
gov_data.drop(indexNames, inplace=True)
gov_data.head()
```

Out[16]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10	LEVEL1	OUTNCCS
0	19818	NaN	NaN	PALMER SECOND BAPTIST CHURCH	PALMER	MA	RE	PC	IN
1	29215	NaN	NaN	ST GEORGE CATHEDRAL	SOUTH BOSTON	MA	RE	PC	IN
2	260049	NaN	NaN	CORINTH BAPTIST CHURCH	HOSFORD	FL	RE	PC	IN
3	490336	NaN	NaN	EASTSIDE BAPTIST CHURCH	LABELLE	FL	RE	PC	IN
4	587764	NaN	NaN	IGLESIA BETHESDA INC	LOWELL	MA	RE	PC	IN

In [17]:
```
# Determine if rows are even yet
gov_data.count()
```

Out[17]:
```
EIN        1147641
CTOTREV     401062
CASSETS     401062
NAME       1147641
CITY       1147641
STATE      1147542
NTMAJ10    1147641
LEVEL1     1147641
OUTNCCS    1147641
dtype: int64
```

In [18]:
```
# drop invalid rows and display
gov_data.dropna(axis=0, how='any', inplace=True)
gov_data
```

Out[18]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10	L
20	10015091	109998.0	57145.0	HANOVER SOCCER CLUB INC	CEDAR KNOLLS	NJ	HU	
21	10017496	110522.0	235622.0	AGAMENTICUS YACHT CLUB OF YORK	YORK HARBOR	ME	HU	
36	10024645	1032510.0	1947235.0	BANGOR SYMPHONY ORCHESTRA	BANGOR	ME	AR	
102	10130427	52152619.0	57577945.0	BRIDGTON HOSPITAL	LEWISTON	ME	HE	
113	10133442	393145.0	824589.0	OXFORD COUNTY AGRICULTURAL SOCIETY	NORWAY	ME	HU	
...	...	...	...	...	...	...	...	
1499429	996064620	3893395.0	259000997.0	PARKER RANCH FOUNDATION TR 091092	KAMUELA	HI	PU	
1499434	996074970	16088.0	280096.0	HONOLULU FIRE DEPARTMENT FIREMANS FUND FOUNDAT...	HONOLULU	HI	ED	
1499436	996078202	0.0	1.0	JOSEPH CAMPBELL FOUNDATION ENDOWMENT TRUST	NEW YORK	NY	PU	
1499437	996078252	568256.0	3843893.0	DAVID C AI CRAT 05051996	HONOLULU	HI	PU	
1499442	996086871	2123617.0	36960599.0	WATERHOUSE CHARITABLE TR	HONOLULU	HI	PU	

401031 rows × 9 columns

In [19]:
```
# Determine the remaining count of rows and ensure our data set is full (no un
even row counts)
gov_data.count()
```

Out[19]:
```
EIN         401031
CTOTREV     401031
CASSETS     401031
NAME        401031
CITY        401031
STATE       401031
NTMAJ10     401031
LEVEL1      401031
OUTNCCS     401031
dtype: int64
```

Merge Region Data

In [20]:
```
# Merge Region and gov_data
gov_data = pd.merge(gov_data, df_region, on= "STATE", how="inner")
gov_data
```

Out[20]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10
0	10015091	109998.0	57145.0	HANOVER SOCCER CLUB INC	CEDAR KNOLLS	NJ	HU
1	10494871	57370.0	30265.0	NORTHEAST AGRICULTURAL-AVIATION ASSOCIATION	WILLIAMSTOWN	NJ	HU
2	10553431	1598747.0	1476724.0	AMERICAN FRIENDS OF YESHIVAT HESDDER SDEROT INC	PASSAIC	NJ	RE
3	10553478	4578318.0	2157398.0	EAST MOUNTAIN HOSPITAL INC	BELLE MEAD	NJ	HE
4	10554061	562209.0	5238304.0	MOUNT EPHRAIM SENIOR HOUSING INITIATIVE INC	MOUNT EPHRAIM	NJ	HU
...	...	...	...	...	...	...	...
396700	911806767	298889.0	2039505.0	MOUNTAIN VISTA RETIREMENT RESIDENCE	LANDER	WY	HU
396701	912052088	48941.0	35108.0	WHIT PRESS	JACKSON	WY	AR
396702	916413193	17148.0	3853059.0	SPENCE FOUNDATION	JACKSON	WY	PU
396703	916470560	3883816.0	16184459.0	NEWTON FOUNDATION	JACKSON	WY	PU
396704	930793885	362609.0	367048.0	SEXUAL ASSAULT AND FAMILY VIOLENCE TASK FORCE ...	EVANSTON	WY	HU

396705 rows × 12 columns

Save file

In [21]:
```
gov_data.to_csv(index=False, path_or_buf="./Resources/gov_data.csv")
```

Create the data sets for Asset testing amongst regions

In [22]:
```
# Create a new datafram for testing assets
assets = pd.DataFrame(gov_data[["REGION", "CASSETS", "LEVEL1"]]).set_index("RE
GION")

# Now split the data by Public Charity vs Private Foundation (LEVEL1)
assets_pc = assets.loc[(assets["LEVEL1"] == "PC"),"CASSETS"].reset_index()
assets_pf = assets.loc[(assets["LEVEL1"] == "PF"),"CASSETS"].reset_index()

# Finally split the data by region
assets_pf_ne = assets_pf.loc[(assets_pf["REGION"] == "NORTHEAST"),"CASSETS"]
assets_pf_s = assets_pf.loc[(assets_pf["REGION"] == "SOUTH"),"CASSETS"]
assets_pf_mw = assets_pf.loc[(assets_pf["REGION"] == "MIDWEST"),"CASSETS"]
assets_pf_w = assets_pf.loc[(assets_pf["REGION"] == "WEST"),"CASSETS"]

assets_pc_ne = assets_pc.loc[(assets_pc["REGION"] == "NORTHEAST"),"CASSETS"]
assets_pc_s = assets_pc.loc[(assets_pc["REGION"] == "SOUTH"),"CASSETS"]
assets_pc_mw = assets_pc.loc[(assets_pc["REGION"] == "MIDWEST"),"CASSETS"]
assets_pc_w = assets_pc.loc[(assets_pc["REGION"] == "WEST"),"CASSETS"]
```

Create the data sets for Revenue testing amongst regions

In [23]:
```
# Create a new datafram for testing revenue
revenue = pd.DataFrame(gov_data[["REGION", "CTOTREV", "LEVEL1"]]).set_index("R
EGION")

# Now split the data by Public Charity vs Private Foundation (LEVEL1)
rev_pc = revenue.loc[(revenue["LEVEL1"] == "PC"),"CTOTREV"].reset_index()
rev_pf = revenue.loc[(revenue["LEVEL1"] == "PF"),"CTOTREV"].reset_index()

# Finally split the data by region
rev_pf_ne = rev_pf.loc[(rev_pf["REGION"] == "NORTHEAST"),"CTOTREV"]
rev_pf_s = rev_pf.loc[(rev_pf["REGION"] == "SOUTH"),"CTOTREV"]
rev_pf_mw = rev_pf.loc[(rev_pf["REGION"] == "MIDWEST"),"CTOTREV"]
rev_pf_w = rev_pf.loc[(rev_pf["REGION"] == "WEST"),"CTOTREV"]

rev_pc_ne = rev_pc.loc[(rev_pc["REGION"] == "NORTHEAST"),"CTOTREV"]
rev_pc_s = rev_pc.loc[(rev_pc["REGION"] == "SOUTH"),"CTOTREV"]
rev_pc_mw = rev_pc.loc[(rev_pc["REGION"] == "MIDWEST"),"CTOTREV"]
rev_pc_w = rev_pc.loc[(rev_pc["REGION"] == "WEST"),"CTOTREV"]
```

Review Basic Statistics of the Remaining Data

- Run basic statistics on the numeric columns

In [24]:
```
gov_data.describe()
```

Out[24]:

	EIN	CTOTREV	CASSETS	STATE_POP	REGION_POP
count	3.967050e+05	3.967050e+05	3.967050e+05	3.967050e+05	3.967050e+05
mean	4.444691e+08	4.822967e+06	9.086742e+06	1.097046e+07	7.431519e+07
std	2.475034e+08	1.132784e+08	2.236824e+08	1.083852e+07	4.343714e+07
min	1.001509e+07	-7.906811e+07	-3.057337e+07	5.777370e+05	5.611079e+06
25%	2.600430e+08	4.017400e+04	2.553300e+04	3.955704e+06	6.830874e+07
50%	4.162965e+08	1.233240e+05	1.330810e+05	7.171646e+06	7.799366e+07
75%	5.925408e+08	5.319560e+05	8.420830e+05	1.280706e+07	1.247539e+08
max	9.960869e+08	4.846638e+10	7.351824e+10	6.691818e+07	1.247539e+08

Beth's section - Visualization 1-3

In [25]:
```
bethdf=pd.read_csv("./Resources/gov_data.csv")
medians=bethdf.groupby(["REGION","LEVEL1"]).agg({'median'}).reset_index()
medians.values.tolist()
medians
```

Out[25]:

	REGION	LEVEL1	EIN	CTOTREV	CASSETS	STATE_POP	REGION_POP
			median	median	median	median	median
0	MIDWEST	PC	381303843.0	130995.0	174064.0	9995915	68308744
1	MIDWEST	PF	384807179.0	31793.5	227734.0	11689442	68308744
2	NORTHEAST	PC	223236840.0	128860.0	162787.0	12807060	5611079
3	NORTHEAST	PF	256788940.0	35650.5	103147.5	12807060	5611079
4	SOUTH	PC	541459968.0	117983.0	116135.0	10383620	124753948
5	SOUTH	PF	465294746.0	21794.0	20009.0	10383620	124753948
6	WEST	PC	680564956.5	121792.5	104885.0	3955704	77993663
7	WEST	PF	464271920.5	27515.5	30479.5	3955704	77993663

In [26]:
```
df_medians = pd.DataFrame(medians)
df_medians
```

Out[26]:

	REGION	LEVEL1	EIN	CTOTREV	CASSETS	STATE_POP	REGION_POP
			median	median	median	median	median
0	MIDWEST	PC	381303843.0	130995.0	174064.0	9995915	68308744
1	MIDWEST	PF	384807179.0	31793.5	227734.0	11689442	68308744
2	NORTHEAST	PC	223236840.0	128860.0	162787.0	12807060	5611079
3	NORTHEAST	PF	256788940.0	35650.5	103147.5	12807060	5611079
4	SOUTH	PC	541459968.0	117983.0	116135.0	10383620	124753948
5	SOUTH	PF	465294746.0	21794.0	20009.0	10383620	124753948
6	WEST	PC	680564956.5	121792.5	104885.0	3955704	77993663
7	WEST	PF	464271920.5	27515.5	30479.5	3955704	77993663

In [27]:
```
df = pd.DataFrame({"REGION": ["MIDWEST", "MIDWEST", "NORTHEAST", "NORTHEAST",
"SOUTH", "SOUTH", "WEST", "WEST"],
                   "LEVEL1": ["PC", "PF", "PC", "PF","PC", "PF","PC", "PF"],
                   "EIN":    [370964193.5, 382597740.0, 223236840.0, 25678894
0.0, 541459968.0, 465294746.0, 680362025.0, 464309410.5],
                   "CTOTREV": [130582.0, 32055.0, 128860.0, 35650.5, 117983.0,
21784.0, 120796.0, 28673.0],
                   "CASSETS": [171237.5, 233340.5, 184787.0, 103147.5, 116135.
0, 20009.0, 103491.0, 29364.5]})
```

In [28]:
```
ein_df = df[["REGION", "LEVEL1", "EIN"]].pivot(index="REGION", columns="LEVEL
1", values="EIN")
ctotrev_df = df[["REGION", "LEVEL1", "CTOTREV"]].pivot(index="REGION", columns
="LEVEL1", values="CTOTREV")
cassets_df = df[["REGION", "LEVEL1", "CASSETS"]].pivot(index="REGION", columns
="LEVEL1", values="CASSETS")
```

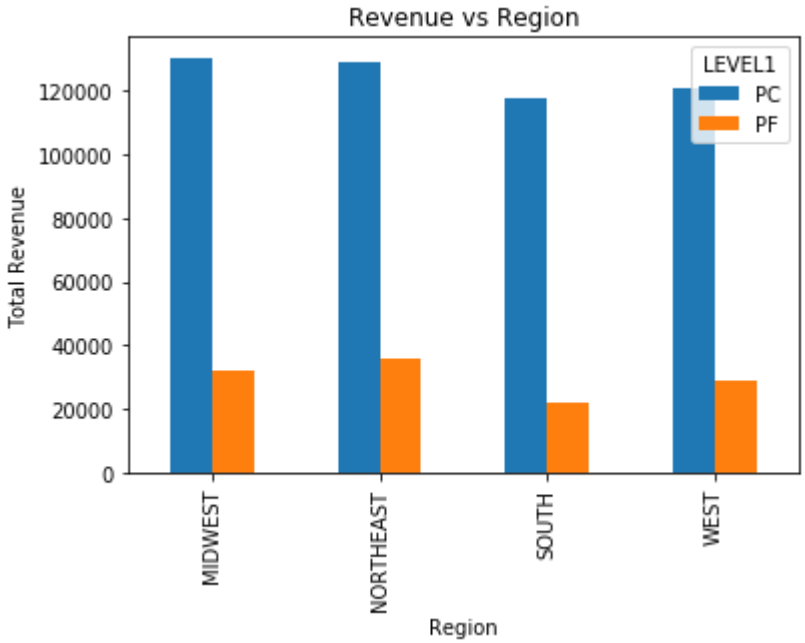
In [29]:
```
ctotrev_df
```

Out[29]:

LEVEL1	PC	PF
REGION		
MIDWEST	130582.0	32055.0
NORTHEAST	128860.0	35650.5
SOUTH	117983.0	21784.0
WEST	120796.0	28673.0

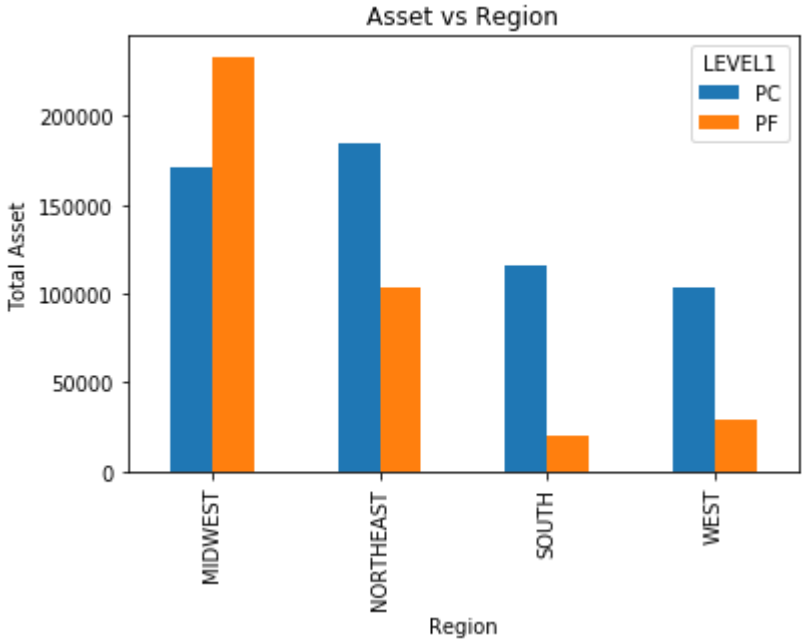
In [30]:
```
rev_chart = ctotrev_df.plot(kind='bar')
rev_chart.set_xlabel("Region")
rev_chart.set_ylabel("Total Revenue")
plt.title("Revenue vs Region")
```

Out[30]: Text(0.5, 1.0, 'Revenue vs Region')



In [31]:
```
asset_chart = cassets_df.plot(kind='bar')
asset_chart.set_xlabel("Region")
asset_chart.set_ylabel("Total Asset")
plt.title("Asset vs Region")
```

Out[31]: Text(0.5, 1.0, 'Asset vs Region')



In [32]:
```
# df_medians["CASSETS"],df_medians["CTOTREV"]
```

In [33]:
```
# df_medians.dtypes,df_medians.info()
```

In [34]:
```
# df_medians.rename(columns={'EIN, median':'EIN','CTOTREV, median':'Revenu
e','CASSETS, median':'Assets'})
```

In [35]:
```
# df_medians.pivot_table(index=['REGION','LEVEL1'],values=['']
```

In [36]:
```
# assets = [171237.5, 233340.5, 162787.0, 103147.5, 116135.0, 20009.0, 103491.
0, 29364.5]
# revenues = [130582.0, 32055.0, 128860.0, 35650.5, 117983.0, 21794.0, 120796.
0, 28673.0]
# # types = ['PC','PF','PC','PF','PC','PF','PC','PF']
# index = ["MidwestPC","MidwestPF","NortheastPC","NortheastPF","SouthPC","Sout
hPF","WestPC","WestPF"]

# assets_df = pd.DataFrame(assets, index=index)
# ax = df_medians.plot.bar(rot=0)
```

In [37]:
```
# df_medians["REGION"] = df_medians["REGION"].str.replace('/','')
# df_medians["LEVEL1"] = df_medians["LEVEL1"].str.replace('/','')
# df_medians.dtypes
```

In [38]:
```
# d = {"REGION": ["MIDWEST","NORTHEAST","SOUTH","WEST"],
#      'colors': ['red','black','blue','dog']}

# keys = [k for k in d.keys() for v in d[k]]
# values = [v for k in d.keys() for v in d[k]]
# pd.DataFrame.from_dict({'index': keys, 'values': values})
```

In [39]:
```
# assets_bar=(bethdf.groupby(["REGION","LEVEL1"]).agg({"CASSETS":['media
n']})).reset_index()
# assets_bar
```

In [40]:
```
# print(df_medians.index)
```

In [41]:
```
# print(bethdf.groupby(["REGION","LEVEL1"]).agg({"CTOTREV":['median']}))
# print(bethdf.groupby(["REGION","LEVEL1"]).agg({"EIN":['count']}))
```

Emile's section - Visualization 4

In [ ]:
```

```

Scott's section - Visualization 5

In [ ]:
```

```

Deanna section - Visualization 6-7

In [42]:
```
# Create pie charts, for comparison purposes for the number of NTMAJ10 organiz
ation categories for
# public charities and for the private foundations

#List unique values in the gov_data['NTMAJ10'] column

gov_data.NTMAJ10.unique()

#Create dataframe to use for pie chart creation from original datasource
gov_data_ntmaj10 = gov_data
```

In [43]:
```
# Merge NTMAJ10 to include the long name of the NTMAJ10 column
gov_data_ntmaj10_merged = pd.merge(gov_data, gov_data_ntmaj10_df, on= "NTMAJ10", how="inner")
gov_data_ntmaj10_merged
```

Out[43]:

	EIN	CTOTREV	CASSETS	NAME	CITY	STATE	NTMAJ10
0	10015091	109998.0	57145.0	HANOVER SOCCER CLUB INC	CEDAR KNOLLS	NJ	HU
1	10494871	57370.0	30265.0	NORTHEAST AGRICULTURAL-AVIATION ASSOCIATION	WILLIAMSTOWN	NJ	HU
2	10554061	562209.0	5238304.0	MOUNT EPHRAIM SENIOR HOUSING INITIATIVE INC	MOUNT EPHRAIM	NJ	HU
3	10562775	189485.0	162076.0	NEW JERSEY EMERGENCY PREPAREDNESS ASSOCIATION	MAYS LANDING	NJ	HU
4	10562891	863976.0	131478.0	ATLANTIC CAPE FAMILY SERVICE ORGANIZATION	NORTHFIELD	NJ	HU
...	...	...	...	...	...	...	...
396700	320312957	159790.0	86919.0	RAILYARD PARK CONSERVANCY	SANTA FE	NM	UN
396701	464417223	54975.0	4291.0	ALEXAS HUGS INC	BOSQUE FARMS	NM	UN
396702	452759984	109082.0	48007.0	DISCERNING HEARTS	OMAHA	NE	UN
396703	470842819	445.0	14625.0	TUSKEEGEE AIRMEN INC ALFONZA W DAVIS CHAPTER	OFFUTT AFB	NE	UN
396704	205674859	378947.0	163952.0	YELLOWSTONE QUAKE	CODY	WY	UN

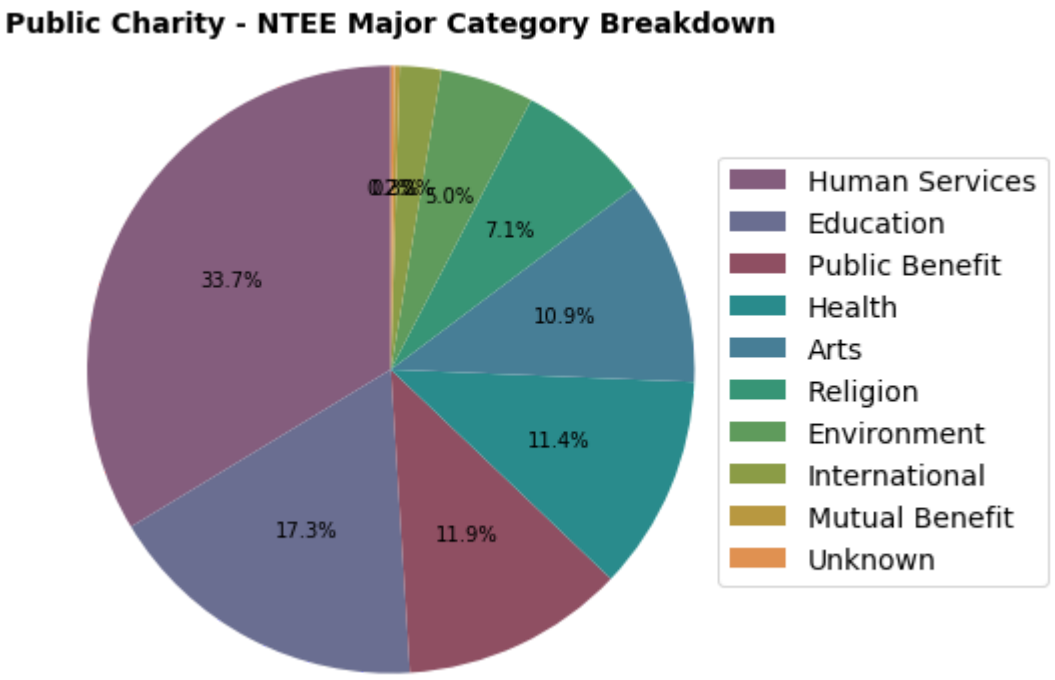
396705 rows × 13 columns

Public Charity - NTEE Major Category Breakdown

In [44]:
```
# Apply filter where Level 1 is PC
gov_data_level1_pc = gov_data_ntmaj10_merged[gov_data_ntmaj10_merged.LEVEL1 ==
"PC"]

# count NTMAJ10 values for PC group level 1
pc_ntmaj10_count = gov_data_ntmaj10_merged["Long_Name"].value_counts()
pc_ntmaj10_count
```

Out[44]:
```
Human Services    133531
Education          68793
Public Benefit     47232
Health             45386
Arts               43129
Religion           28315
Environment        19749
International       8705
Mutual Benefit      1099
Unknown              766
Name: Long_Name, dtype: int64
```

In [45]:
```
#create pie chart
# set colors
colors = ["#845D7D","#6A6E91","#8F4F62","#298B8C","#477E96","#379576","#5F9B5
C","#8B9C46","#B89840","#E09151"]

# set labels
ntmaj10_labels_pc = ["Human Services", "Education", "Public Benefit", "Health"
, "Arts", "Religion", "Environment", "International", "Mutual Benefit", "Unkno
wn"]

# create pie chart
plt.pie(pc_ntmaj10_count, labels = ntmaj10_labels_pc, labeldistance = None, sh
adow = False, colors = colors, explode=(0,0,0,0,0,0,0,0,0,0), startangle = 90,
autopct='%1.1f%%')
plt.axis("equal")

# add legend, title, size
plt.legend(labels=ntmaj10_labels_pc, fontsize=14, loc="center right", bbox_to_
anchor=(1, 0, .5, 1))
plt.title("Public Charity - NTEE Major Category Breakdown", y=1, weight='bold'
, size=14)
fig = plt.gcf()
fig.set_size_inches(6,6)
plt.show()
```

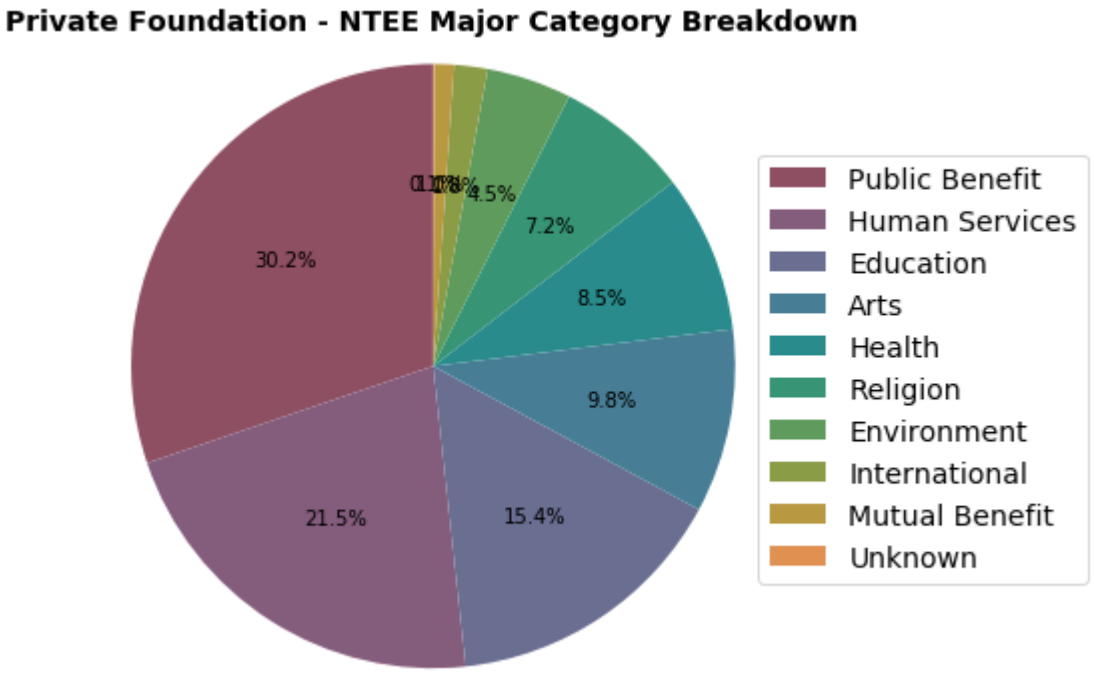


Private Foundation - NTEE Major Category Breakdown

In [46]:
```
# Apply filter whereLevel 1 is PF
gov_data_level1_pf = gov_data_ntmaj10_merged[gov_data_ntmaj10_merged.LEVEL1 ==
"PF"]

# count NTMAJ10 values for PF group level 1
pf_ntmaj10_count = gov_data_level1_pf["Long_Name"].value_counts()
pf_ntmaj10_count
```

Out[46]:
```
Public Benefit    439
Human Services    313
Education         224
Arts              143
Health            123
Religion          104
Environment        66
International      26
Mutual Benefit     15
Unknown             1
Name: Long_Name, dtype: int64
```

In [47]:
```
# set colors
colors = ["#8F4F62","#845D7D","#6A6E91","#477E96","#298B8C","#379576","#5F9B5
C","#8B9C46","#B89840","#E09151"]

# set labels
ntmaj10_labels_pf = ["Public Benefit", "Human Services", "Education", "Arts",
"Health", "Religion", "Environment", "International", "Mutual Benefit", "Unkno
wn"]

# create pie chart
plt.pie(pf_ntmaj10_count, labels = ntmaj10_labels_pf, labeldistance = None, sh
adow = False, colors = colors, explode=(0,0,0,0,0,0,0,0,0,0), startangle = 90,
autopct='%1.1f%%')
plt.axis("equal")

# add legend, title, size
plt.legend(labels=ntmaj10_labels_pf, fontsize=14, loc="center right", bbox_to_
anchor=(1, 0, .5, 1))
plt.title("Private Foundation - NTEE Major Category Breakdown", y=1, weight='b
old', size=14)
fig = plt.gcf()
fig.set_size_inches(6,6)
plt.show()
```



Katherine's section - Hypothesis Testing

Create a function to compare four populations (Northeast, South, Midwest, West)

Inputs

- Four series (populations)
- Title
- yAxis label

Displays

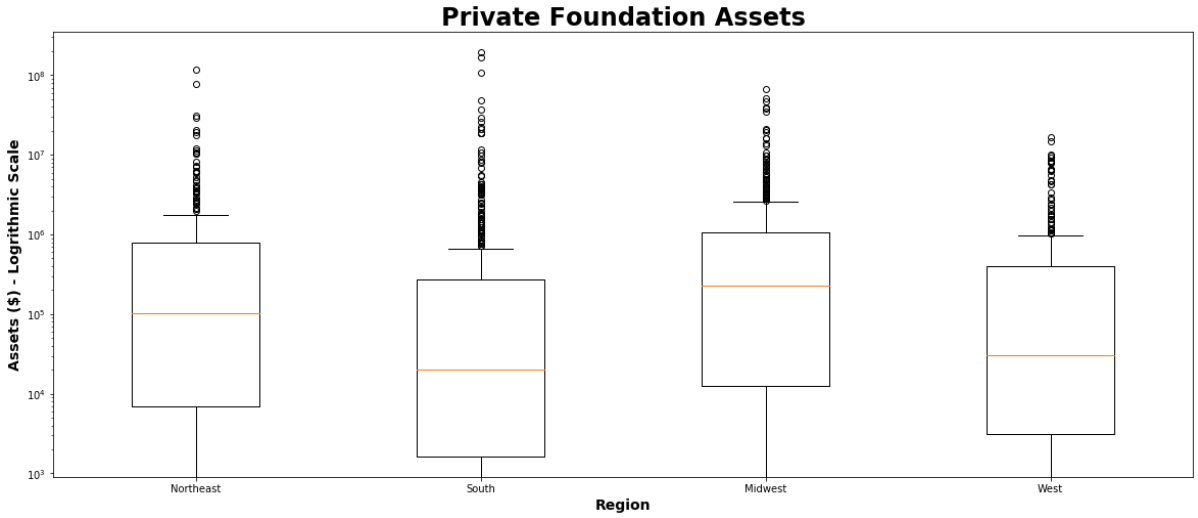
- Boxplot
- ANOVA
- Kruskal Wallis tests

Create a function to plot four regional boxplots, perform ANOVA and Kruskal hypothesis tests

In [48]:
```
# Function comparing four populations by Boxplots
def boxPlotCompare(northeast, south, midwest, west, title, y_label):

    # Set the figure size
    fig = plt.figure(figsize=(20,8))
    axBox = fig.add_subplot()

    # Show box plots of the data
    box_plot_data=[northeast, south, midwest, west]
    plt.boxplot(box_plot_data)
    plt.title(title, color='k', size=24, weight='bold')
    plt.xticks([1, 2, 3, 4], ['Northeast', 'South', 'Midwest', 'West'])

    # KEY!  Set the scale to logrithmic
    axBox.set_yscale('log')
    plt.xlabel("Region", size=14, weight='bold')
    plt.ylabel(y_label, size=14, weight='bold')
    plt.show()

    # Perform an ANOVA test assuming populations are not equal
    print('\033[1m' + '\nANOVA test' + '\033[0m')
    print(stats.f_oneway(northeast, south, midwest, west))

    # Perform a Kruskal test assuming population medians are not equal
    print('\033[1m' + '\nKruskal test' + '\033[0m')
    print(stats.kruskal(northeast, south, midwest, west))

    # Because we found many differences, compare 1 to 1
    print('\033[1m' + '\nKruskal test - NE to S' + '\033[0m')
    print(stats.kruskal(northeast, south))

    print('\033[1m' + '\nKruskal test - NE to MW' + '\033[0m')
    print(stats.kruskal(northeast, midwest))

    print('\033[1m' + '\nKruskal test - NE to W' + '\033[0m')
    print(stats.kruskal(northeast, west))

    print('\033[1m' + '\nKruskal test - S to MW' + '\033[0m')
    print(stats.kruskal(south, midwest))

    print('\033[1m' + '\nKruskal test - S to W' + '\033[0m')
    print(stats.kruskal(south, west))

    print('\033[1m' + '\nKruskal test - MW to W' + '\033[0m')
    print(stats.kruskal(midwest, west))

    return
```

Null Hypothesis: Private foundation assets are equal across regions.

In [49]:
```
# Compare the four regions assets for Private Foundations - boxplot
boxPlotCompare(assets_pf_ne, assets_pf_s, assets_pf_mw, assets_pf_w,
        "Private Foundation Assets", "Assets ($) - Logrithmic Scale")
```



ANOVA test
F_onewayResult(statistic=1.5991297876338924, pvalue=0.18774699235546738)

Kruskal test
KruskalResult(statistic=65.64098672556152, pvalue=3.657966313635066e-14)

Kruskal test - NE to S
KruskalResult(statistic=21.19949192533357, pvalue=4.138742128024859e-06)

Kruskal test - NE to MW
KruskalResult(statistic=4.89198782908695, pvalue=0.02698160306364065)

Kruskal test - NE to W
KruskalResult(statistic=10.257981013090541, pvalue=0.0013609454059984938)

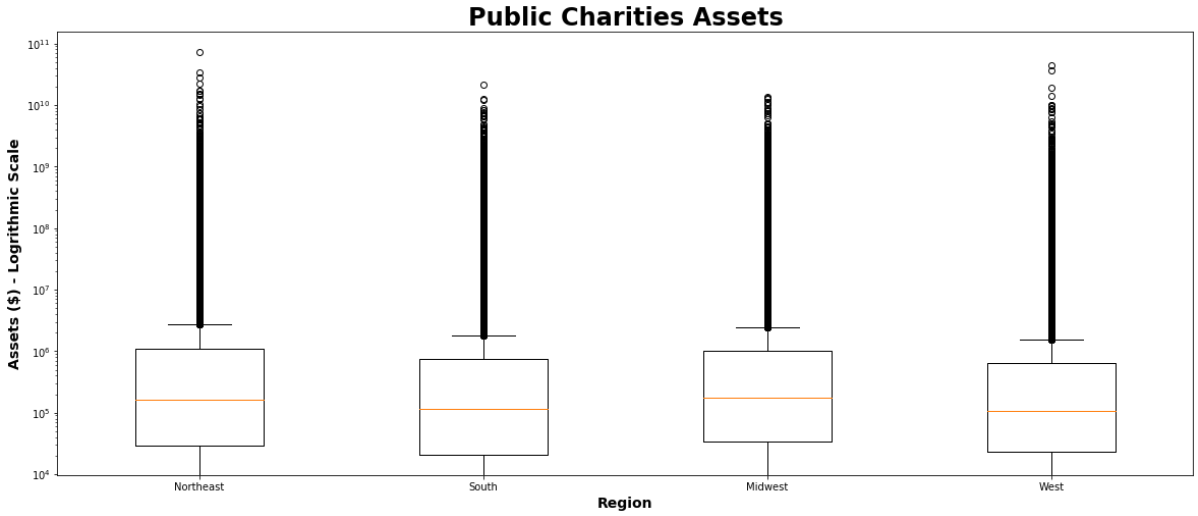
Kruskal test - S to MW
KruskalResult(statistic=54.034001360720005, pvalue=1.9705004004839034e-13)

Kruskal test - S to W
KruskalResult(statistic=2.0482479275156438, pvalue=0.15238146336062583)

Kruskal test - MW to W
KruskalResult(statistic=32.36860784507243, pvalue=1.2753012446444715e-08)

Null Hypothesis: Public Charity assets are equal across regions.

In [50]:
```
# Compare the four regions assets for Public Charities
boxPlotCompare(assets_pc_ne, assets_pc_s, assets_pc_mw, assets_pc_w,
        "Public Charities Assets", "Assets ($) - Logrithmic Scale")
```



ANOVA test
F_onewayResult(statistic=9.977841999540356, pvalue=1.4259678934716717e-06)

Kruskal test

C:\Users\katro\Anaconda3\envs\PythonData\lib\site-packages\scipy\stats\stats.py:5879: RuntimeWarning: overflow encountered in long_scalars
  h = 12.0 / (totaln * (totaln + 1)) * ssbn - 3 * (totaln + 1)

KruskalResult(statistic=114460436.28216954, pvalue=0.0)

Kruskal test - NE to S
KruskalResult(statistic=-28972976.542478856, pvalue=1.0)

Kruskal test - NE to MW
KruskalResult(statistic=14806747.063532323, pvalue=0.0)

Kruskal test - NE to W
KruskalResult(statistic=-20512143.057604466, pvalue=1.0)

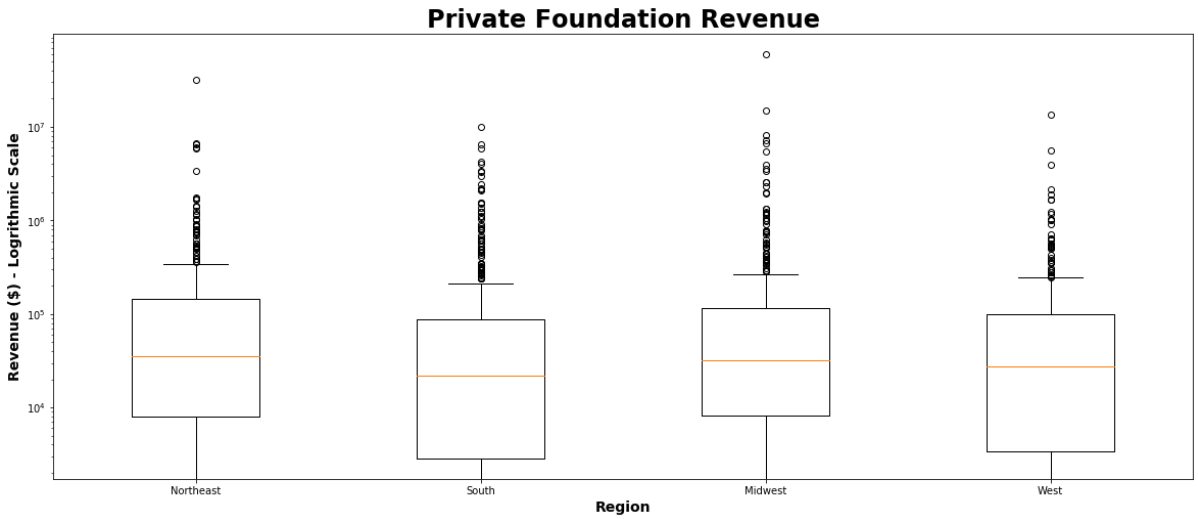
Kruskal test - S to MW
KruskalResult(statistic=-14120448.973866025, pvalue=1.0)

Kruskal test - S to W
KruskalResult(statistic=49750023.46017419, pvalue=0.0)

Kruskal test - MW to W
KruskalResult(statistic=-10120503.307078624, pvalue=1.0)

Null Hypothesis: Private foundation revenue is equal across regions.

In [51]:
```
# Compare the four regions revenue for Private Foundations
boxPlotCompare(rev_pf_ne, rev_pf_s, rev_pf_mw, rev_pf_w,
        "Private Foundation Revenue", "Revenue ($) - Logrithmic Scale")
```



ANOVA test
F_onewayResult(statistic=1.5507443368076232, pvalue=0.1995987426944119)

Kruskal test
KruskalResult(statistic=16.3435183471287, pvalue=0.0009641398152239498)

Kruskal test - NE to S
KruskalResult(statistic=9.771525006709846, pvalue=0.0017723538365773932)

Kruskal test - NE to MW
KruskalResult(statistic=0.06461297087045435, pvalue=0.7993481455202777)

Kruskal test - NE to W
KruskalResult(statistic=5.4236502854815125, pvalue=0.01986578517512666)

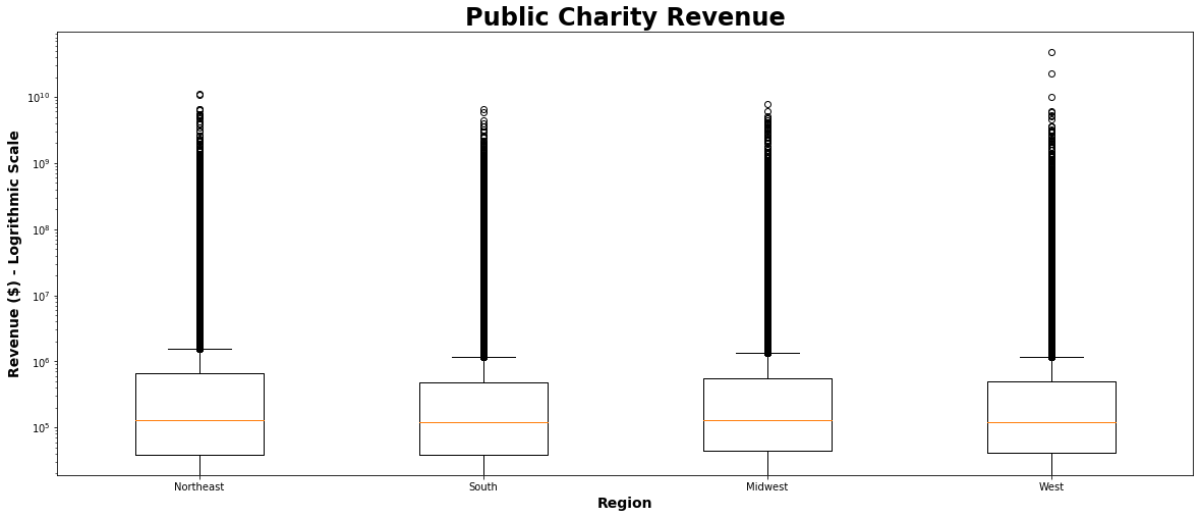
Kruskal test - S to MW
KruskalResult(statistic=11.048889346975777, pvalue=0.000887403434323604)

Kruskal test - S to W
KruskalResult(statistic=0.4832721144020404, pvalue=0.4869439097118484)

Kruskal test - MW to W
KruskalResult(statistic=4.850456641237852, pvalue=0.027638814287329096)

Null Hypothesis: Public Charity revenue is equal across regions.

In [52]:
```
# Compare the four regions revenue for Public Charities
boxPlotCompare(rev_pc_ne, rev_pc_s, rev_pc_mw, rev_pc_w,
        "Public Charity Revenue", "Revenue ($) - Logrithmic Scale")
```



ANOVA test
F_onewayResult(statistic=6.2903816751459045, pvalue=0.00029075250150393985)

Kruskal test
KruskalResult(statistic=114267593.1077235, pvalue=0.0)

Kruskal test - NE to S

C:\Users\katro\Anaconda3\envs\PythonData\lib\site-packages\scipy\stats\stats.py:5879: RuntimeWarning: overflow encountered in long_scalars
  h = 12.0 / (totaln * (totaln + 1)) * ssbn - 3 * (totaln + 1)

KruskalResult(statistic=-28938166.555687387, pvalue=1.0)

Kruskal test - NE to MW
KruskalResult(statistic=14806634.997257186, pvalue=0.0)

Kruskal test - NE to W
KruskalResult(statistic=-20476338.68960117, pvalue=1.0)

Kruskal test - S to MW
KruskalResult(statistic=-14099540.75095499, pvalue=1.0)

Kruskal test - S to W
KruskalResult(statistic=49751272.29590349, pvalue=0.0)

Kruskal test - MW to W
KruskalResult(statistic=-10098602.69915487, pvalue=1.0)

Based on results, look at means and medians for populations

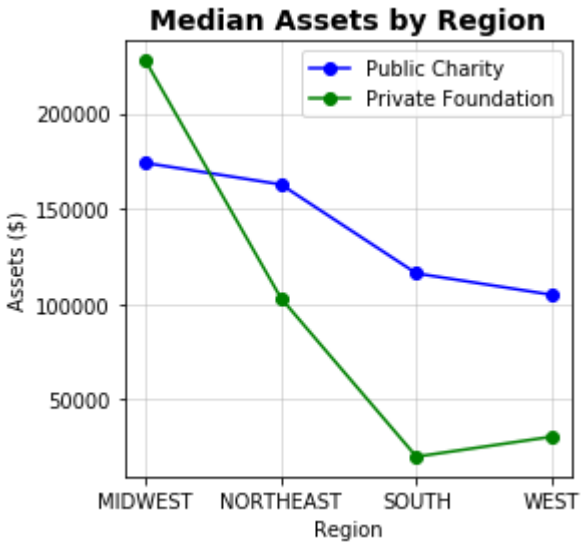
In [53]:
```
# Create means
asset_pc_means = assets_pc.groupby(["REGION"]).mean()
asset_pf_means = assets_pf.groupby(["REGION"]).mean()
rev_pc_means = rev_pc.groupby(["REGION"]).mean()
rev_pf_means = rev_pf.groupby(["REGION"]).mean()

# Create medians
asset_pc_medians = assets_pc.groupby(["REGION"]).median()
asset_pf_medians = assets_pf.groupby(["REGION"]).median()
rev_pc_medians = rev_pc.groupby(["REGION"]).median()
rev_pf_medians = rev_pf.groupby(["REGION"]).median()
```

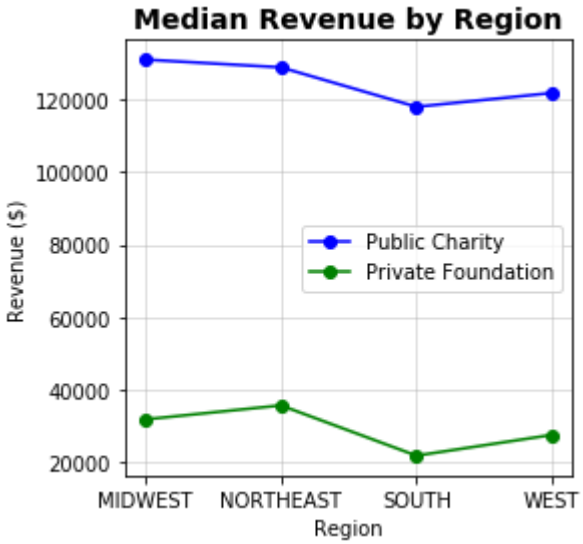
Create a function to plot means and medians

In [54]:
```
def plotbyRegion(series1, series2, ylabel, title, series_label1, series_label2
):

    # Set the figure size
    plt.figure(figsize=(4,4))

    # plot the values
    plt.plot(series1, label=series_label1, color='b', marker='o')
    plt.plot(series2, label=series_label2, color='g', marker='o')

    # add labels, etc.
    plt.title(title, weight='bold', size=14)
    plt.xlabel("Region")
    plt.ylabel(ylabel)
    plt.grid(alpha=0.5)
    plt.legend(loc='best')

    # save the graph
    plt.savefig('./Output/' + title.replace(" ", "") + '.png')

    return
```

Compare medians (Kruskal Wallis results)

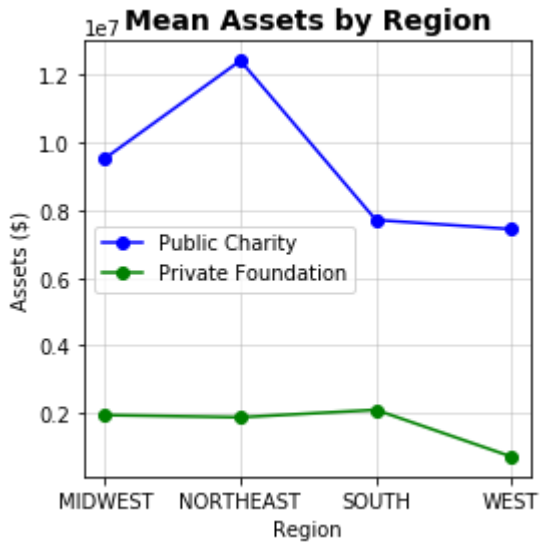
In [55]:
```
# Plot by region and type the median assets and revenue
plotbyRegion(asset_pc_medians, asset_pf_medians, "Assets ($)", "Median Assets
 by Region",
             "Public Charity", "Private Foundation")

plotbyRegion(rev_pc_medians, rev_pf_medians, "Revenue ($)", "Median Revenue by
Region",
             "Public Charity", "Private Foundation")
```

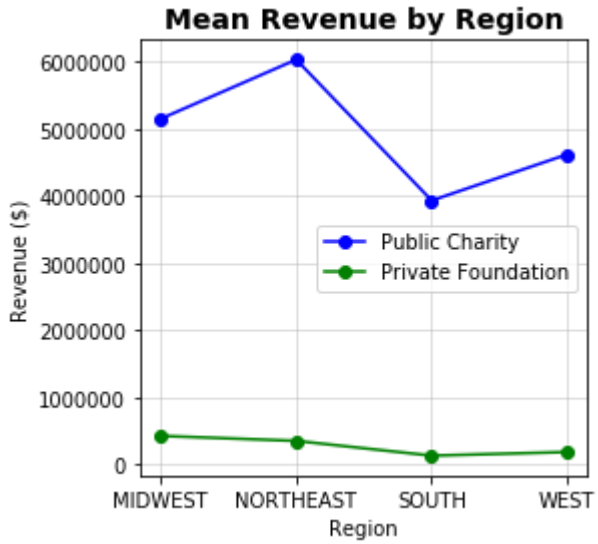




Compare means (ANOVA results)

In [56]:
```
# Plot by region and type the mean assets and revenue
plotbyRegion(asset_pc_means, asset_pf_means, "Assets ($)", "Mean Assets by Reg
ion",
             "Public Charity", "Private Foundation")

plotbyRegion(rev_pc_means, rev_pf_means, "Revenue ($)", "Mean Revenue by Regio
n",
             "Public Charity", "Private Foundation")
```





In [ ]: