IPO analysis

June 30, 2019

In [1]:
```
import seaborn as sns
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.stats.api as sms
import matplotlib.pyplot as plt
import scipy
from scipy import stats
from statsmodels.compat import lzip

%matplotlib inline

ipo = pd.read_excel('IPO_Scandinavia.xlsx')
ipo.set_index('Year', inplace = True, )
ipo.head()
```

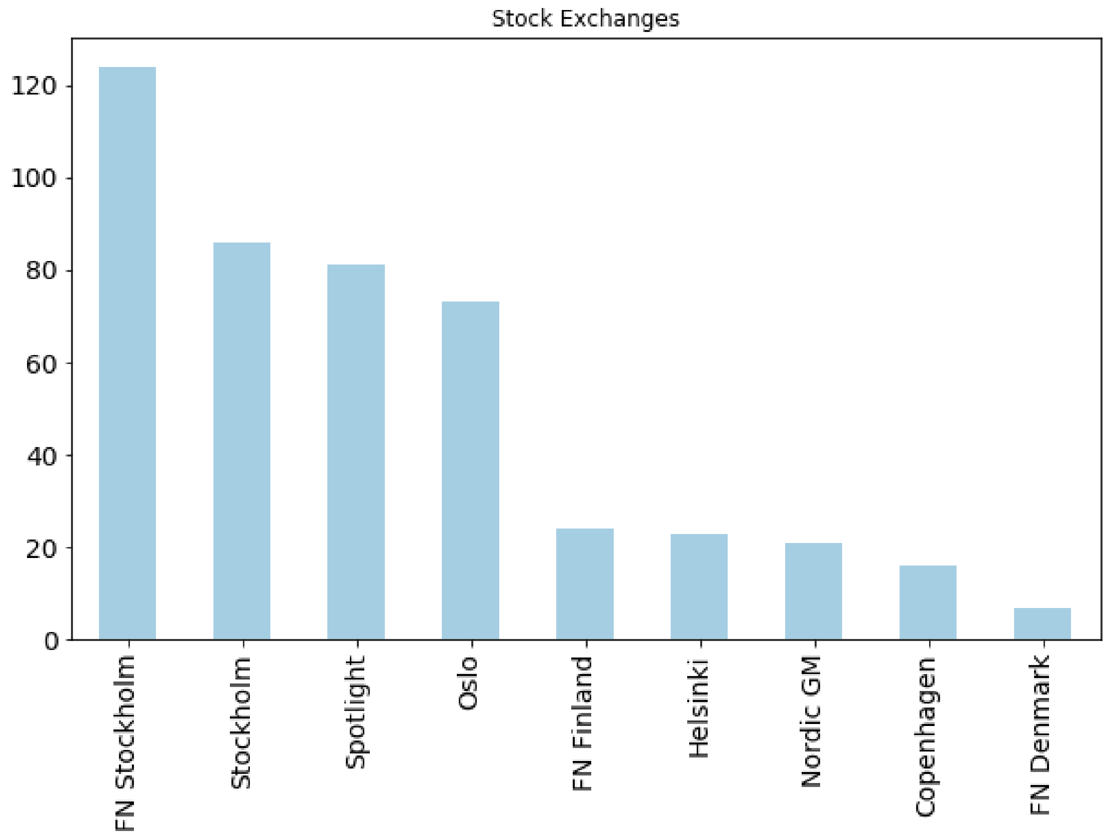
Out[1]:
```
      Month  Day ListingDate  Registration  Age IssuerTicker  \
Year
2010      2    5  2010-02-05          2007    3     NORTH NO
2010      3    2  2010-03-02          1963   47     SJOHB SS
2010      3   24  2010-03-24          1986   24     ARISE SS
2010      3   26  2010-03-26          1968   42     BAKKA NO
2010      3   29  2010-03-29          2004    6      BRAW SS

                  IssuerName  Revenuest-1      EBITt-1  CurrencyRevenues  \
Year
2010       North Energy ASA          0.0 -207630000.0               0.0
2010          Sportjohan AB    4479000.0     -728000.0         4132773.3
2010               Arise AB   29700000.0  -10800000.0        27404190.0
2010         Bakkafrost P/F  596565000.0  192394000.0       768674002.5
2010  Brandworld Sverige AB   83187000.0   18468000.0        76756644.9

        ...         InitialPubOffer(Lead Mgr) InitialPubOffer(SharesOffered)  \
Year    ...
2010    ...              PARETO,PLATOU,SEBENS                       12087000
2010    ...       Sedermera Fondkommission AB                        1150000
2010    ...            ABG Sundal Collier Asa                       10730000
```

```
2010    ...                   NORDIK,NORDEA                   2608000
2010    ...                         Unknown                    750000

     OfferSizeAdj ClosingPrice1stDay AdjustedRevenues  AdjustedEbit  \
Year
2010  320306000.0              26.80         0.000000     -0.648224
2010    1652900.0               2.00         2.709783     -0.440438
2010  486636000.0              53.75         0.061031     -0.022193
2010   80848000.0              34.40         7.378847      2.379700
2010    2444660.0               4.90        34.028045      7.554425

     CurrencyAdjustedRevenues CurrencyAdjustedEbit  Unnamed: 33  YearDummy
Year
2010                 0.000000            -0.648224    1000000.0       2010
2010                 2.500317            -0.406392          NaN       2010
2010                 0.056314            -0.020478          NaN       2010
2010                 9.507644             3.066244          NaN       2010
2010                31.397677             6.970468          NaN       2010

[5 rows x 34 columns]
```

In [2]:
```
# Extracting and creating a table with IPO return and sectors only
df = pd.DataFrame()
df['ipo'] = ipo['OfferTo1stClose']
df['sector'] = ipo['IndustrySector']
df['sector'].value_counts()
df['ipo'].describe()
```

Out[2]:
```
count    455.000000
mean       6.316785
std       32.334439
min      -95.555557
25%       -6.000000
50%        1.052632
75%       14.712171
max      310.714294
Name: ipo, dtype: float64
```

In [3]:
```
# Exchange count table
table1 = pd.DataFrame()
table1['Primary Exchange'] = ipo['PrimaryExchange'].value_counts()

Table1 = table1.plot(kind='bar',stacked=False,figsize=(10,6),
                     colormap='Paired', title='Stock Exchanges',
                    fontsize=14, legend= False)
plt.savefig("Figure X")
```

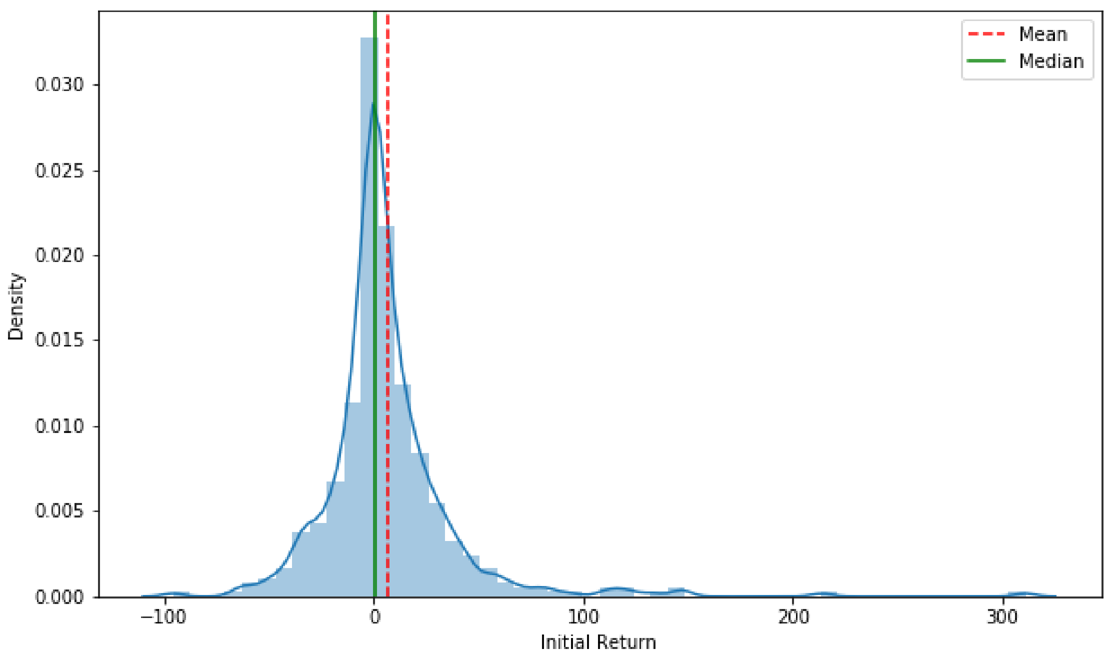


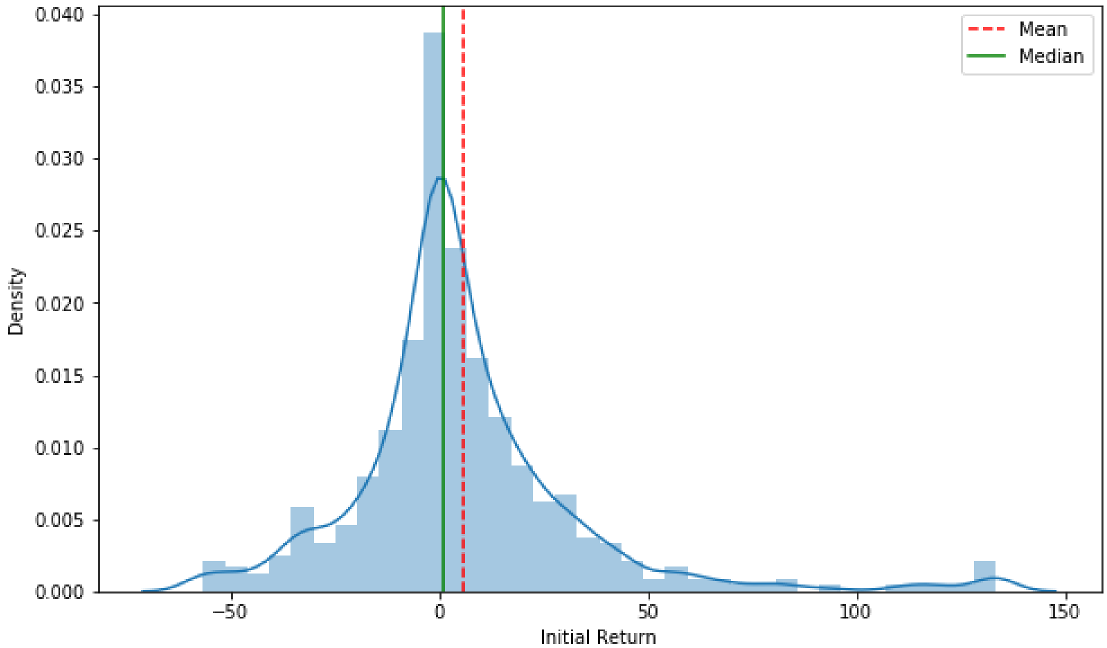
In [4]:
```
## Winsorizing data - Handling Extreme Values - Removing Outliers ##
# 1% winsorizing fraction
df['ipo'].mean() # 6.31 NO WINSORIZATION
winzipo = pd.DataFrame(index=ipo.index)
winzipo['0.5% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.005)
winzipo['1% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.01)
winzipo['2% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.02)
winzipo['3% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.03)
winzipo['4% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.04)
winzipo['10% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.1)
#winzipo['5% winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits = 0.05)
winzipo['1% winz'].mean() # 5.89%

# Distribution plot before winsorizing !
distfig, ax1 = plt.subplots()
distfig.set_size_inches(10, 6)
sns.distplot(df['ipo'])
plt.axvline(df['ipo'].mean(), color='r', linestyle='--',label='Mean')
plt.axvline(df['ipo'].median(), color='g', linestyle='-',label='Median')
plt.xlabel('Initial Return')
plt.ylabel('Density')
```

```
plt.legend()

#Distribution plot after 1% winsorizing!
distfig2, ax2 = plt.subplots()
distfig2.set_size_inches(10, 6)
sns.distplot(winzipo['1% winz'])
#sns.distplot(winzipo['1% winz'])
plt.axvline(winzipo['1% winz'].mean(), color='r', linestyle='--',label='Mean')
plt.axvline(winzipo['1% winz'].median(), color='g', linestyle='-',label='Median')
plt.xlabel('Initial Return')
plt.ylabel('Density')
plt.legend()
```

```
/Applications/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: U
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
Out[4]: <matplotlib.legend.Legend at 0x1c1bf2a7f0>
```





In [5]:
```
# Dividing into years
# 2010
From2010 = '2010'
To2010   = '2010'
ipo2010 = df.loc[From2010:To2010,:]
#2011
From2011 = '2011'
To2011   = '2011'
ipo2011 = df.loc[From2011:To2011,:]
#2012
From2012 = '2012'
To2012   = '2012'
ipo2012 = df.loc[From2012:To2012,:]
#2013
From2013 = '2013'
To2013   = '2013'
ipo2013 = df.loc[From2013:To2013,:]
#2014
From2014 = '2014'
To2014   = '2014'
ipo2014 = df.loc[From2014:To2014,:]
#2015
From2015 = '2015'
To2015   = '2015'
ipo2015 = df.loc[From2015:To2015,:]
```

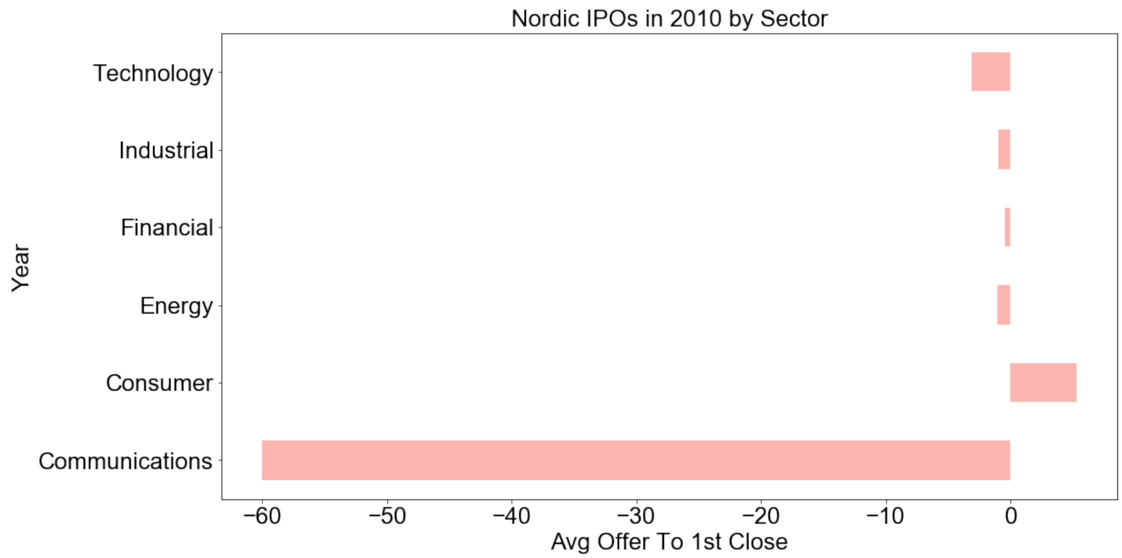
```
#2016
From2016 = '2016'
To2016   = '2016'
ipo2016 = df.loc[From2016:To2016,:]
#2017
From2017 = '2017'
To2017   = '2017'
ipo2017 = df.loc[From2017:To2017,:]
#2018
From2018 = '2018'
To2018   = '2018'
ipo2018 = df.loc[From2018:To2018,:]
```

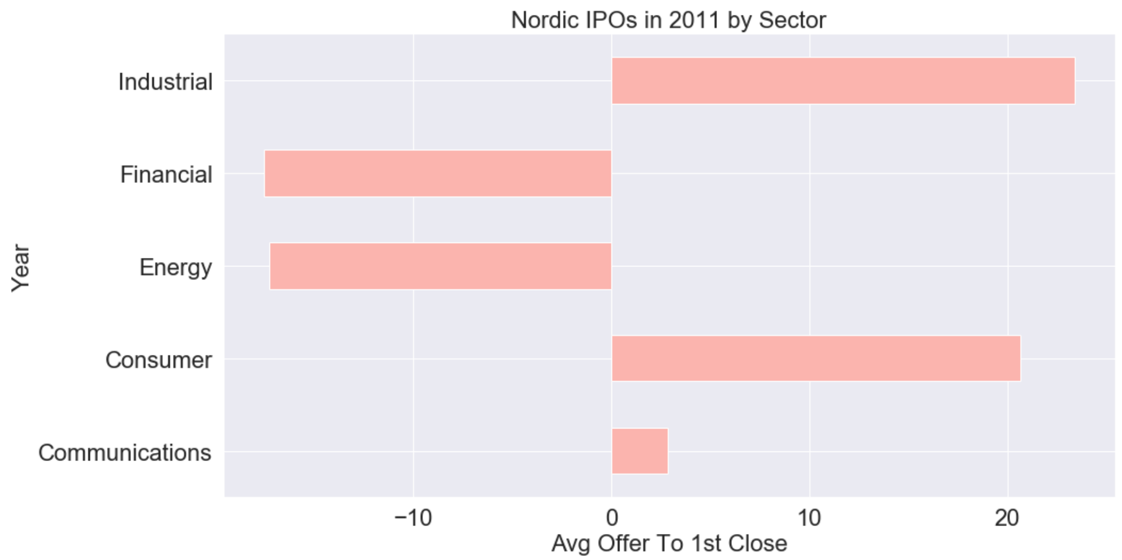
In [6]:
```
#grouping by sectors
ipo2010_by_sector = ipo2010.groupby('sector')
ipo2011_by_sector = ipo2011.groupby('sector')
ipo2012_by_sector = ipo2012.groupby('sector')
ipo2013_by_sector = ipo2013.groupby('sector')
ipo2014_by_sector = ipo2014.groupby('sector')
ipo2015_by_sector = ipo2015.groupby('sector')
ipo2016_by_sector = ipo2016.groupby('sector')
ipo2017_by_sector = ipo2017.groupby('sector')
ipo2018_by_sector = ipo2018.groupby('sector')

ipo2010_by_sectormean = pd.DataFrame(ipo2010_by_sector.ipo.mean())
ipo2011_by_sectormean =pd.DataFrame(ipo2011_by_sector.ipo.mean())
ipo2012_by_sectormean =pd.DataFrame(ipo2012_by_sector.ipo.mean())
ipo2013_by_sectormean =pd.DataFrame(ipo2013_by_sector.ipo.mean())
ipo2014_by_sectormean =pd.DataFrame(ipo2014_by_sector.ipo.mean())
ipo2015_by_sectormean =pd.DataFrame(ipo2015_by_sector.ipo.mean())
ipo2016_by_sectormean =pd.DataFrame(ipo2016_by_sector.ipo.mean())
ipo2017_by_sectormean =pd.DataFrame(ipo2017_by_sector.ipo.mean())
ipo2018_by_sectormean =pd.DataFrame(ipo2018_by_sector.ipo.mean())
```

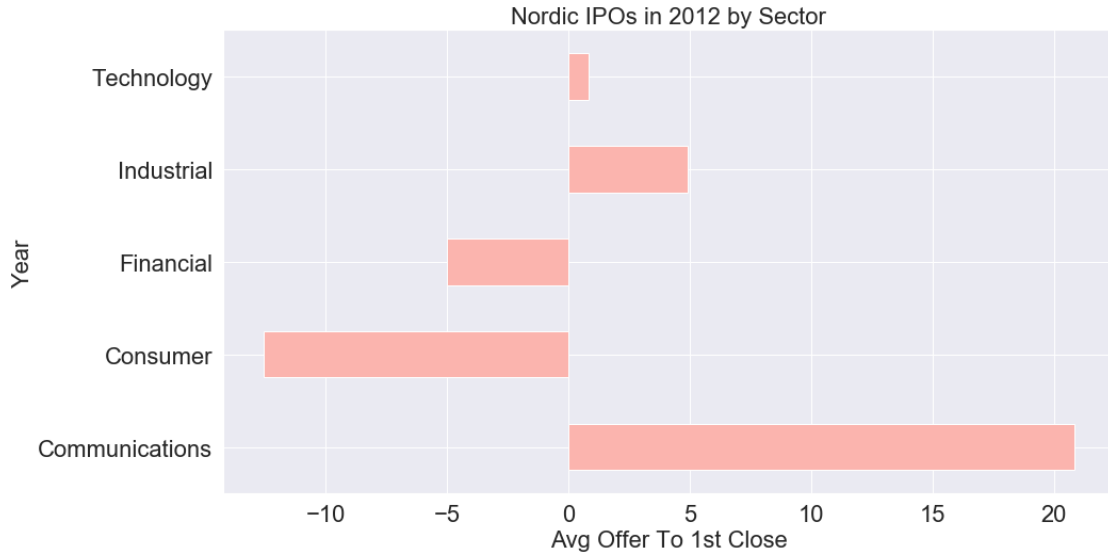
In [7]:
```
## Plotting ##
# Figure 1#
plt.Figure()
fig1 = ipo2010_by_sectormean.plot(kind='barh', figsize=(15,8), fontsize=22, colormap='
sns.set(style="darkgrid")
fig1.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig1.set_ylabel('Year', fontsize=22)
fig1.set_title('Nordic IPOs in 2010 by Sector', fontsize=22)
plt.savefig('Figure 1')

#Figure 2
plt.Figure()
fig2 = ipo2011_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig2.set_title('Nordic IPOs in 2011 by Sector', fontsize=22)
```

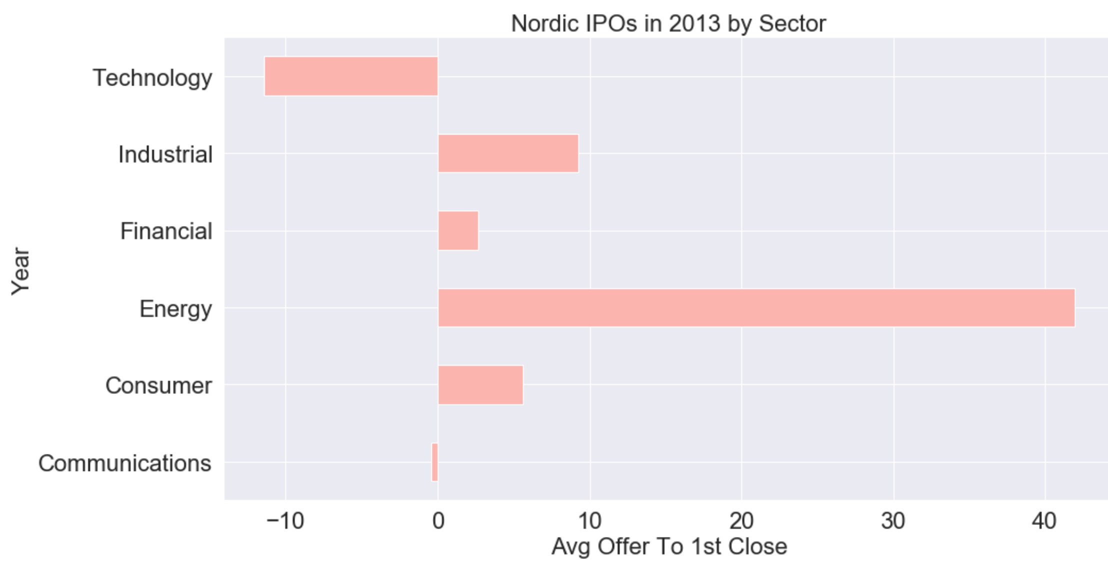
```
fig2.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig2.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 2")

#Figure 3#
plt.Figure()
fig3 = ipo2012_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig3.set_title('Nordic IPOs in 2012 by Sector', fontsize=22)
fig3.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig3.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 3")

#Figure 4#
plt.Figure()
fig4 = ipo2013_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig4.set_title('Nordic IPOs in 2013 by Sector', fontsize=22)
fig4.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig4.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 4")

plt.Figure()
fig5 = ipo2014_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig5.set_title('Nordic IPOs in 2014 by Sector', fontsize=22)
fig5.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig5.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 5")

plt.Figure()
fig6 = ipo2015_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig6.set_title('Nordic IPOs in 2015 by Sector', fontsize=22)
fig6.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig6.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 6")

plt.Figure()
fig7 = ipo2016_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig7.set_title('Nordic IPOs in 2016 by Sector', fontsize=22)
fig7.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig7.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 7")

plt.Figure()
fig8 = ipo2017_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig8.set_title('Nordic IPOs in 2017 by Sector', fontsize=22)
fig8.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig8.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 8")
```

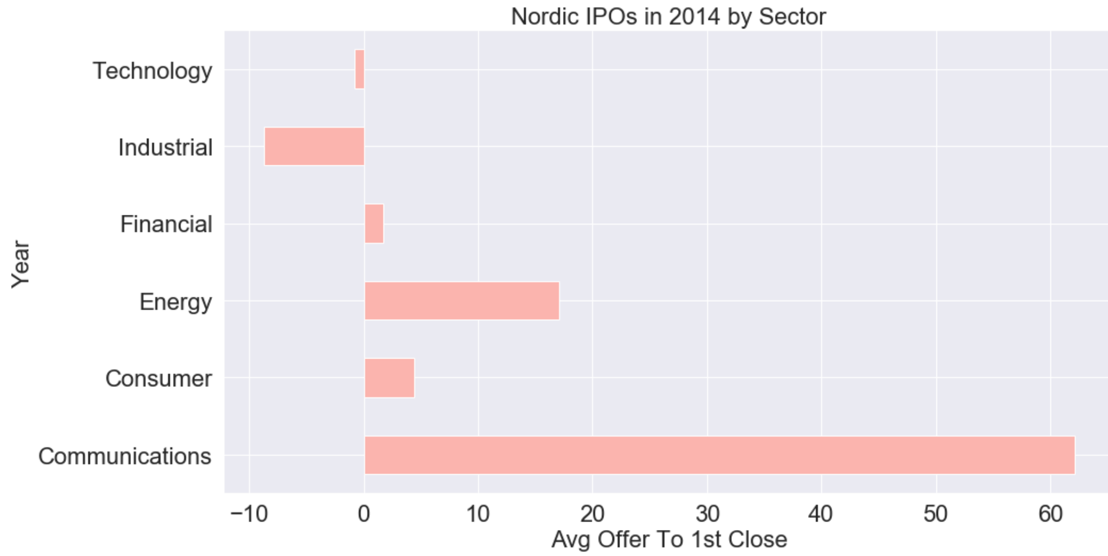
```
plt.Figure()
fig4 = ipo2018_by_sectormean.plot(kind='barh', figsize=(15,8),fontsize=22, colormap='Pa
fig4.set_title('Nordic IPOs in 2018 by Sector', fontsize=22)
fig4.set_xlabel('Avg Offer To 1st Close', fontsize=22)
fig4.set_ylabel('Year', fontsize=22)
plt.savefig("Figure 9")
```

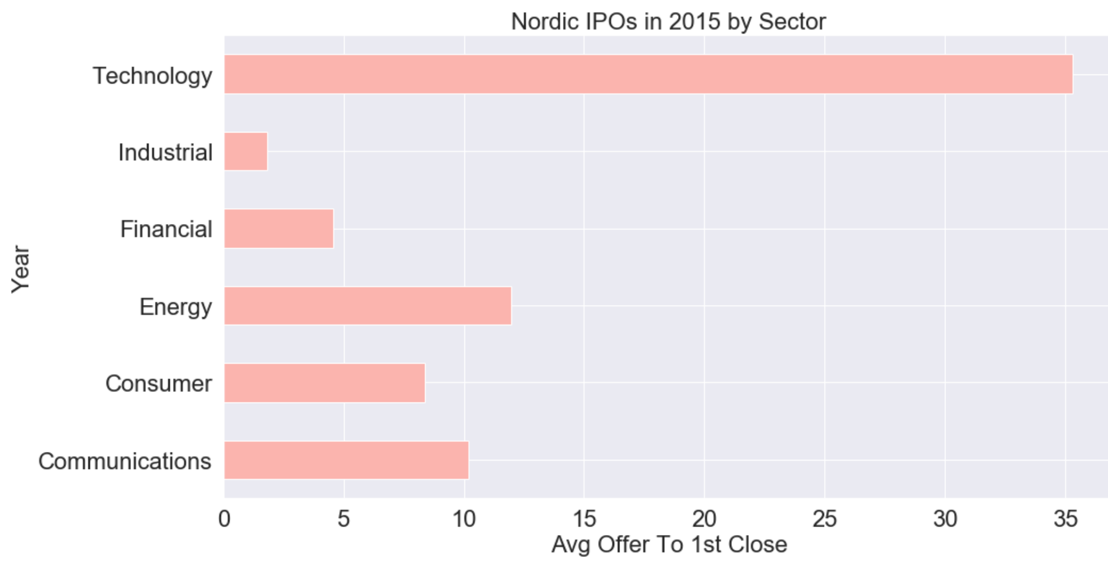


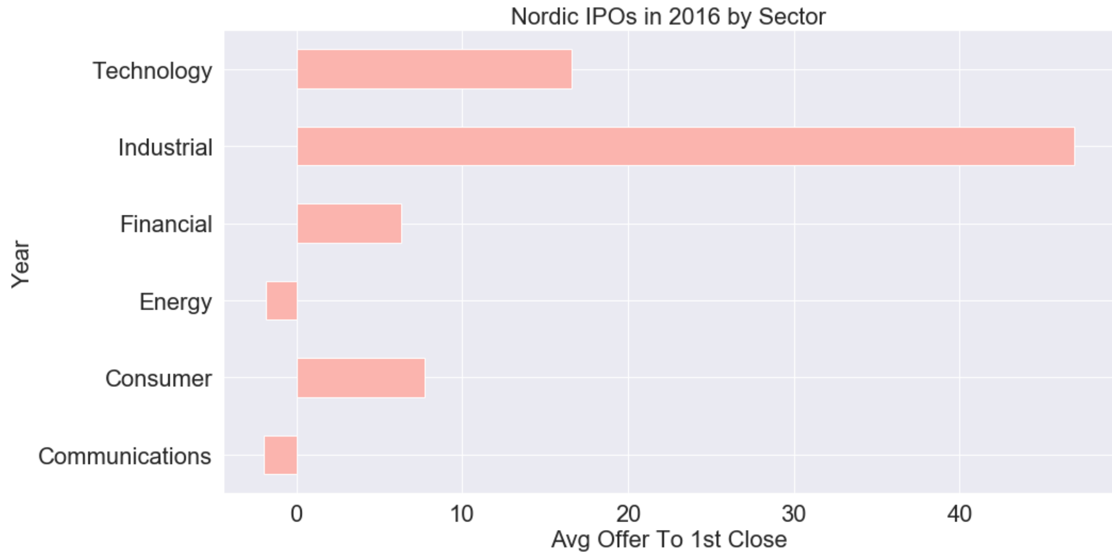



Nordic IPOs in 2012 by Sector
Technology
Industrial
Financial
Consumer
Communications
Year
−10
−5
0
5
10
15
20
Avg Offer To 1st Close

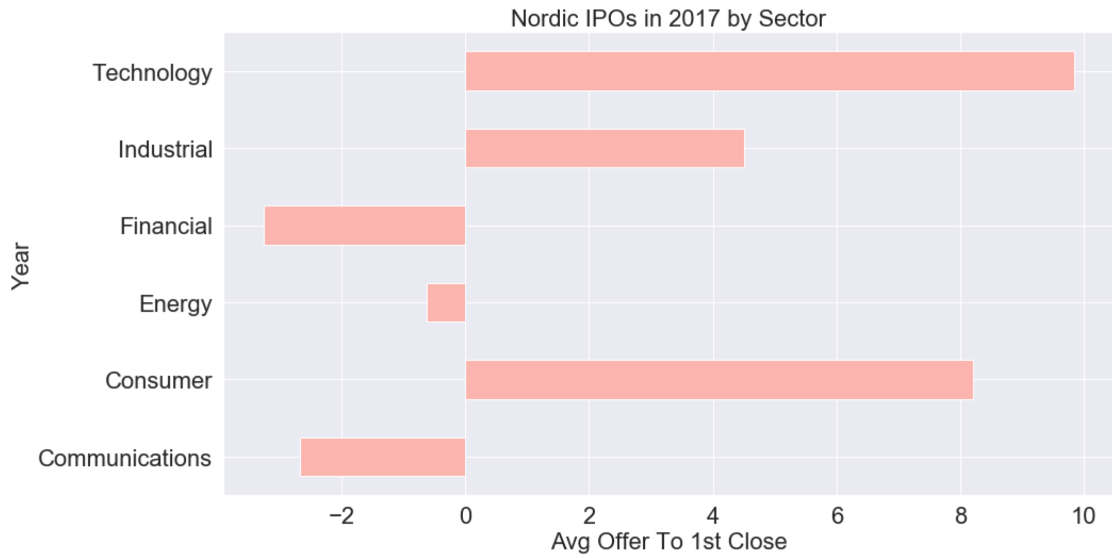

Nordic IPOs in 2013 by Sector
Technology
Industrial
Financial
Energy
Consumer
Communications
Year
−10
0
10
20
30
40
Avg Offer To 1st Close


Nordic IPOs in 2014 by Sector
Technology
Industrial
Financial
Energy
Consumer
Communications
Year
−10
0
10
20
30
40
50
60
Avg Offer To 1st Close

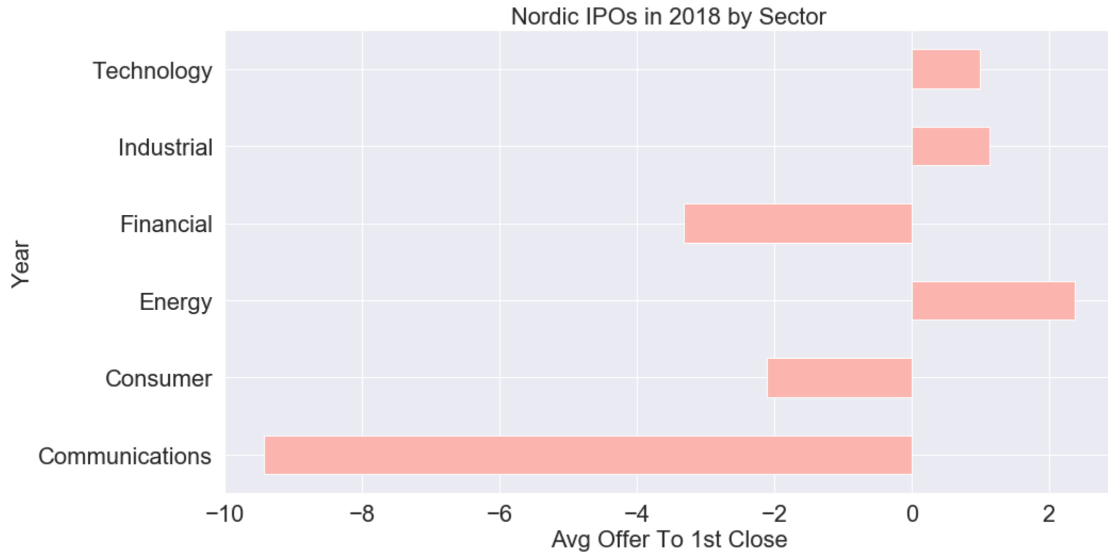

Nordic IPOs in 2015 by Sector
Technology
Industrial
Financial
Energy
Consumer
Communications
Year
0
5
10
15
20
25
30
35
Avg Offer To 1st Close



Nordic IPOs in 2016 by Sector
Technology
Industrial
Financial
Energy
Consumer
Communications
Year
0
10
20
30
40
Avg Offer To 1st Close



Nordic IPOs in 2017 by Sector
Technology
Industrial
Financial
Energy
Consumer
Communications
Year
−2
0
2
4
6
8
10
Avg Offer To 1st Close




```
In [8]: ## !! Grouping winsorized returns after sectors !! ##
        secdes = pd.DataFrame()
        secdes['Init Ret 1% Winz'] = winzipo['1% winz']
        secdes['Sector'] = ipo['IndustrySector']
        secdes_grouped = secdes.groupby('Sector')
        secdes_grouped.median()
        secdes_grouped.describe()
```

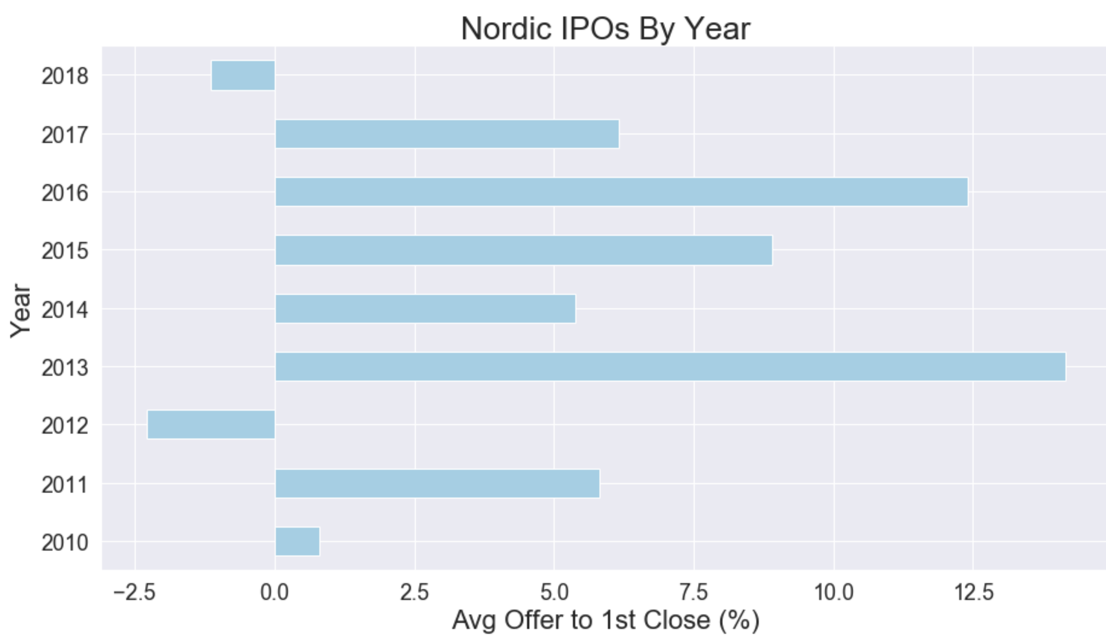
Out[8]:

	Init Ret 1% Winz							
	count	mean	std	min	25%	50%	75%	max
Sector								
Communications	33.0	1.713010	30.433982	-56.571430	-11.500000	1.428571	10.857142	114.666664
Consumer	184.0	6.101483	28.173722	-56.571430	-7.019005	2.905121	14.312500	133.333328
Energy	30.0	5.046451	32.326770	-50.000000	-2.781250	-0.780872	13.161096	133.333328
Financial	59.0	0.603797	15.764849	-56.571430	-0.620446	2.142857	5.427273	53.333332
Industrial	79.0	7.397199	30.164572	-50.000000	-5.463730	0.270270	16.325281	133.333328
Technology	70.0	9.937950	29.736525	-34.782608	-9.412956	5.185524	26.838984	133.333328

```
In [9]: # By year Plotting
        df2 = pd.DataFrame()
```

```
df2['ipo'] = df['ipo']
df2['sector'] = ipo['IndustrySector']
ipo_by_year = df2.groupby('Year')
ipo_by_year_mean = (ipo_by_year.mean())
figure9 = ipo_by_year_mean.plot(kind='barh', legend=False, colormap='Paired',
                                figsize=(15,8), fontsize=18, style= 'seaborn')
figure9.set_xlabel("Avg Offer to 1st Close (%)", fontsize=22)
figure9.set_ylabel("Year", fontsize=22)
figure9.set_title('Nordic IPOs By Year', fontsize=26)
#figure9(style="darkgrid")
```

Out[9]: Text(0.5, 1.0, 'Nordic IPOs By Year')



In [10]:
```
# Different statistics for returns - Descriptive

# !! Returns before Winsorizing !!
initret = df['ipo']
print(initret.mean())
print(initret.median())
print(initret.max())
print(initret.min())
print(initret.describe())

# !! Returns AFTER Winsorizing with 1% !!
print(winzipo['1% winz'].mean())
print(winzipo['1% winz'].median())
print(winzipo['1% winz'].max())
```

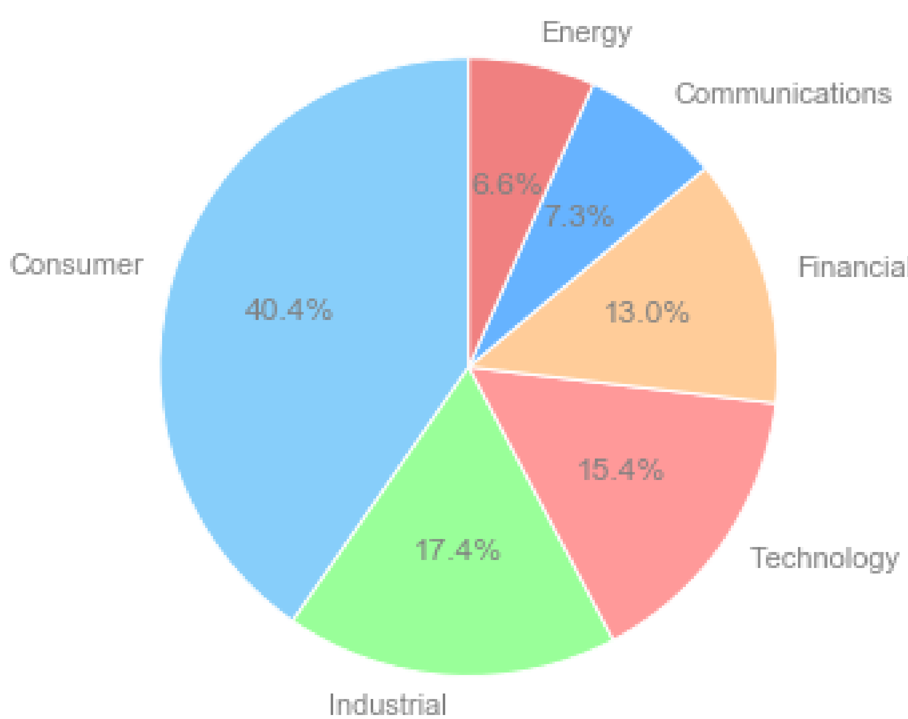
```
        print(winzipo['1% winz'].min())
        print(winzipo['1% winz'].describe())
```

```
6.3167849321350635
1.052631617
310.71429439999997
-95.55555725
count    455.000000
mean       6.316785
std       32.334439
min      -95.555557
25%       -6.000000
50%        1.052632
75%       14.712171
max      310.714294
Name: ipo, dtype: float64
5.8159449354537465
1.052631617
133.3333282
-56.57143021
count    455.000000
mean       5.815945
std       27.970318
min      -56.571430
25%       -6.000000
50%        1.052632
75%       14.712171
max      133.333328
Name: 1% winz, dtype: float64
```

```
In [11]: # Underpricing between industries
         df_by_sector = df.groupby('sector')
         secmean = df_by_sector.mean()
         secmed = df_by_sector.median()
         secstd = df_by_sector.std()
         print(df_by_sector.describe())


         # Pie Chart of Sectors division
         labels = 'Consumer', 'Industrial', 'Technology', 'Financial','Communications','Energy
         sizes = [40.4, 17.4, 15.4, 13, 7.3 ,6.6]
         colors = ['lightskyblue','#99ff99','#ff9999','#ffcc99','#66b3ff','lightcoral']
         fig1, ax1 = plt.subplots()
         patches, texts, autotexts = ax1.pie(sizes, colors = colors, labels=labels, autopct='%
         for text in texts:
             text.set_color('grey')
         for autotext in autotexts:
```

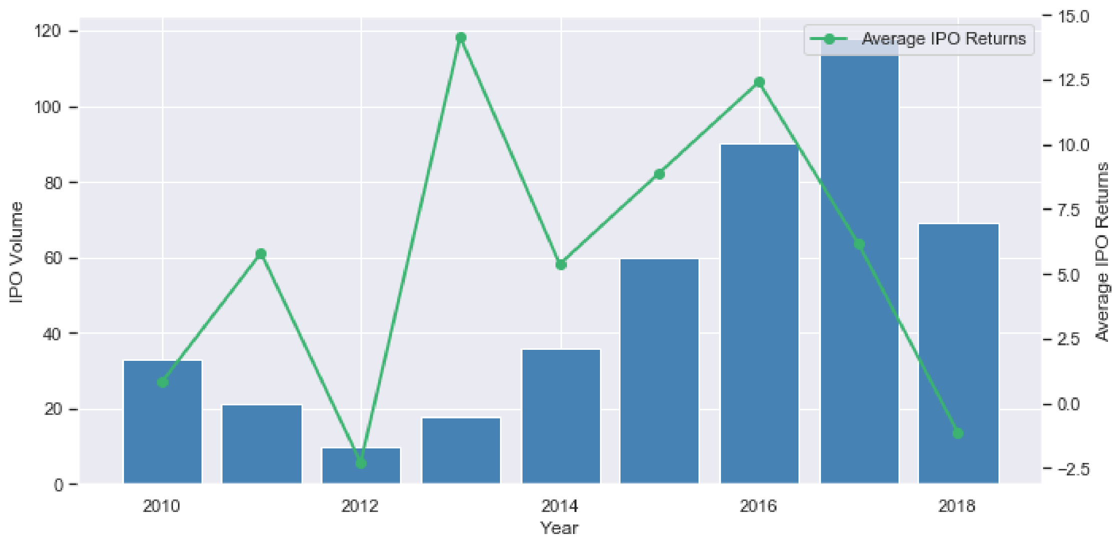
```
        autotext.set_color('grey')
    ax1.axis('equal')
    plt.tight_layout()
    plt.show()
    df_by_sector.median()
```

```
                 ipo                                                              \
               count       mean        std        min        25%        50%
sector
Communications  33.0   1.437878  30.993362 -62.222225 -11.500000   1.428571
Consumer       184.0   6.124026  28.615286 -65.243904  -7.019005   2.905121
Energy          30.0   7.727007  44.407855 -50.000000  -2.781250  -0.780872
Financial       59.0  -0.056950  18.738961 -95.555557  -0.620446   2.142857
Industrial      79.0   9.642528  43.370110 -50.000000  -5.463730   0.270270
Technology      70.0  10.137950  30.612651 -34.782608  -9.412956   5.185524


                      75%         max
sector
Communications  10.857142  114.666664
Consumer        14.312500  146.153854
Energy          13.161096  213.750000
Financial        5.427273   53.333332
Industrial      16.325281  310.714294
Technology      26.838984  147.333328
```



Out[11]:
```
                     ipo
sector
Communications  1.428571
Consumer        2.905121
Energy         -0.780872
Financial       2.142857
Industrial      0.270270
Technology      5.185524
```

In [12]:
```
# Number of IPOs during the timeperiod with respective avg returns

mydf = pd.DataFrame()
mydf['IPOs per year'] = df.index.value_counts()
t = mydf.index
data1 = mydf['IPOs per year']
data2 = ipo_by_year_mean
fig, ax1 = plt.subplots(figsize=(10,5))

color1 = 'steelblue'
ax1.set_xlabel('Year')
ax1.set_ylabel('IPO Volume')
ax1.bar(t, data1, color=color1)
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
ax2.set_ylabel('Average IPO Returns')
ax2.tick_params(axis='y')
ax2.plot(t.sort_values(),data2,color='mediumseagreen',label='Average IPO Returns',lin
        marker='o')
plt.legend()
fig.tight_layout()
plt.grid(False)
plt.show()
```

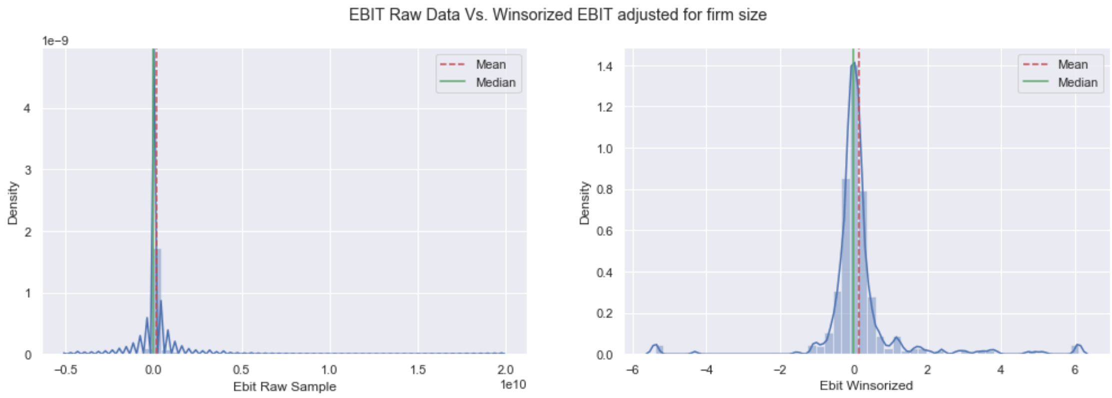



```
In [13]: # Avg offer to 1st close by sector
         df['sector'] = ipo['IndustrySector']
         ipo_by_sector = df.groupby('sector')
         ipo_by_sector.mean()
         #sectnum = ipo_by_sector['sector'].astype('category').cat.codes
         sectnum = df['sector'].astype('category').cat.codes # 5 is tech

In [14]: # Log returns and log age
         logret = pd.DataFrame()
         logret['logret winz'] = np.log(1 + winzipo['1% winz']/100)
         logretuw = pd.DataFrame()
         logretuw['logret uw'] = np.log(1 + df['ipo']/100)
         logage = pd.DataFrame()
         logage['logage'] = np.log(1 + ipo['Age'])

         # Winsorize de uavhengige variablene (?)
         winzipotest = pd.DataFrame(index=ipo.index)
         revt1 = pd.DataFrame()
         ebitt1 = pd.DataFrame()
         osadj = pd.DataFrame(index=ipo.index)
         revt1['log rev'] = np.log(1 + ipo['CurrencyRevenues']/10000000)
         ebitt1['log ebit'] = np.log(1 + ipo['CurrencyEbit'])
         ebitt1['1% winz ebit'] = scipy.stats.mstats.winsorize(ebitt1['log ebit'], limits=0.01
         osadj['offer size adjusted'] = np.log(1 + ipo['Offer Size(M)'])
         winzipotest['1% winz Ebit'] = scipy.stats.mstats.winsorize(ipo['CurrencyAdjustedEbit'

         #Distribution plot of independent variable: Ebit winsorized NO LOG
         distfig3, (ax3,ax4) = plt.subplots(1,2)
         distfig3.suptitle('EBIT Raw Data Vs. Winsorized EBIT adjusted for firm size')
```

```
distfig3.set_size_inches(17, 5)
sns.distplot(winzipotest['1% winz Ebit'], ax=ax4)
ax4.axvline(winzipotest['1% winz Ebit'].mean(), color='r', linestyle='--',label='Mean
ax4.axvline(winzipotest['1% winz Ebit'].median(), color='g', linestyle='-',label='Medi
plt.xlabel('Ebit Winsorized')
plt.ylabel('Density')
ax4.legend()
sns.distplot(ipo['CurrencyEbit'],ax=ax3, axlabel='Ebit Raw Sample')
ax3.axvline(ipo['CurrencyEbit'].mean(), color='r', linestyle='--',label='Mean')
ax3.axvline(ipo['CurrencyEbit'].median(), color='g', linestyle='-',label='Median')
ax3.set_ylabel('Density')
ax3.legend()
```

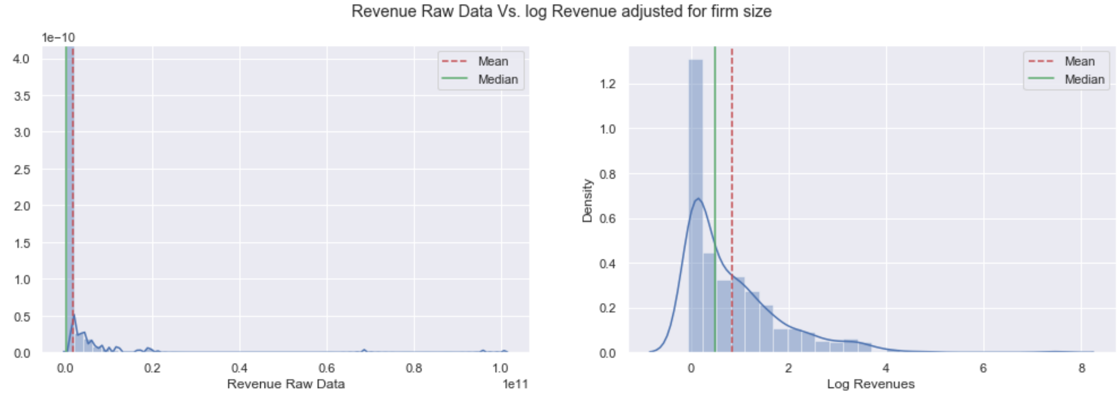
```
/Applications/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15: RuntimeWarning:
  from ipykernel import kernelapp as app
```

Out[14]: <matplotlib.legend.Legend at 0x1c1e34bf28>



In [15]:
```
# Distribution plot of independent variable: Revenue Raw data and LOG revenue
distfig4, (ax5,ax6) = plt.subplots(1,2)
distfig4.suptitle('Revenue Raw Data Vs. log Revenue adjusted for firm size')
distfig4.set_size_inches(17, 5)
sns.distplot(ipo['CurrencyRevenues'], ax=ax5, axlabel= 'Revenue Raw Data')
ax5.axvline(ipo['CurrencyRevenues'].mean(), color='r', linestyle='--',label='Mean')
ax5.axvline(ipo['CurrencyRevenues'].median(), color='g', linestyle='-',label='Median')
plt.ylabel('Density')
ax5.legend()
sns.distplot(np.log(1 + ipo['CurrencyAdjustedRevenues']),ax=ax6, axlabel='Log Revenue
ax6.axvline(np.log(1 + ipo['CurrencyAdjustedRevenues']).mean(), color='r', linestyle=
ax6.axvline(np.log(1 + ipo['CurrencyAdjustedRevenues']).median(), color='g', linestyl
ax6.set_ylabel('Density')
ax6.legend()
```

Out[15]: <matplotlib.legend.Legend at 0x1c1e6420b8>



In [16]:
```
## Regression of raw data !! Nothing is winsorized
technum = df['sector'].astype('category').cat.codes # 5 is tech
delistednum = ipo['Delisted'].astype('category').cat.codes # 1 means delisted
ydata = pd.DataFrame()
ydata = logretuw['logret uw']
xdata = pd.DataFrame()
xdata['Logage'] = logage['logage']
xdata['Adj Revenues'] = ipo['CurrencyAdjustedRevenues']
xdata['Adj Ebit'] = ipo['CurrencyAdjustedEbit']
#xdata['Adj OS'] = osadj['offer size adjusted']
xdata['Techdummy'] = np.where(technum==5,1,0)
xdata['Non-Tech Dummy'] = np.where(technum==5,0,1)
xdata['Young dummy'] = np.where(ipo['Age']<7, 1,0)
xdata['Old dummy'] = np.where(ipo['Age']>=7, 1,0)
xdata['Delisted dummy'] = delistednum

modelA = sm.OLS(ydata,sm.add_constant(xdata.astype(float)),missing='drop').fit(cov_ty
modelA.summary()
#stats.kstest(modelA.resid, 'norm')
#residuals = modelA.resid
#sns.distplot(residuals)
```

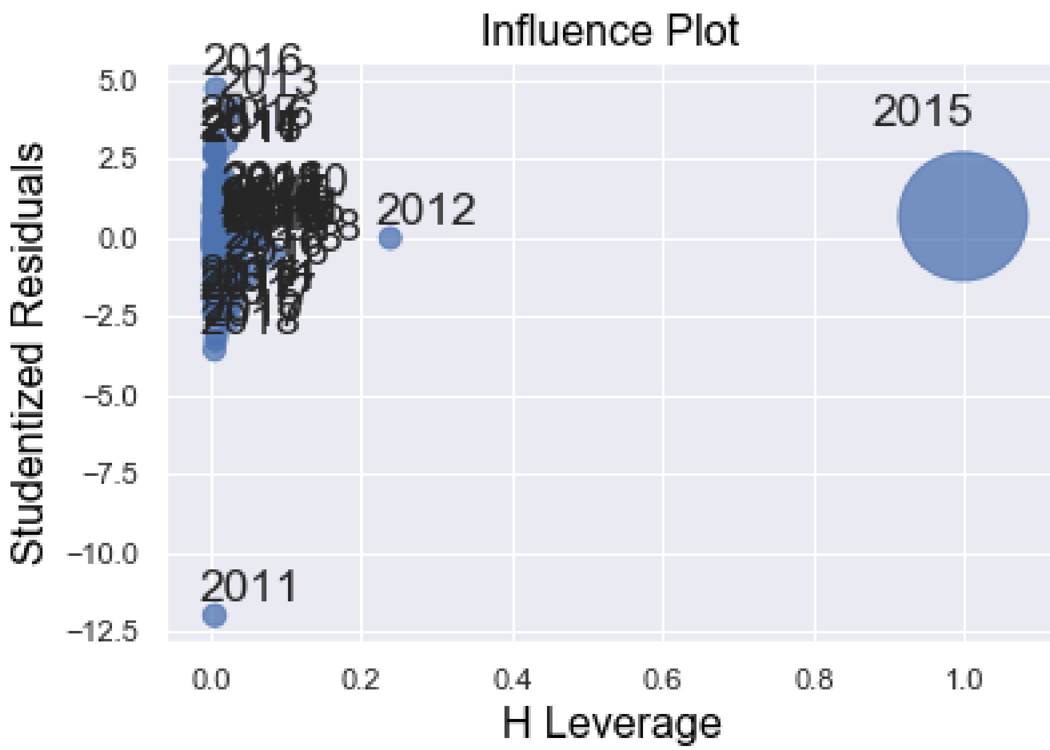
```
/Applications/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:1532: ValueWarni
  'rank is %d' % (J, J_), ValueWarning)
```

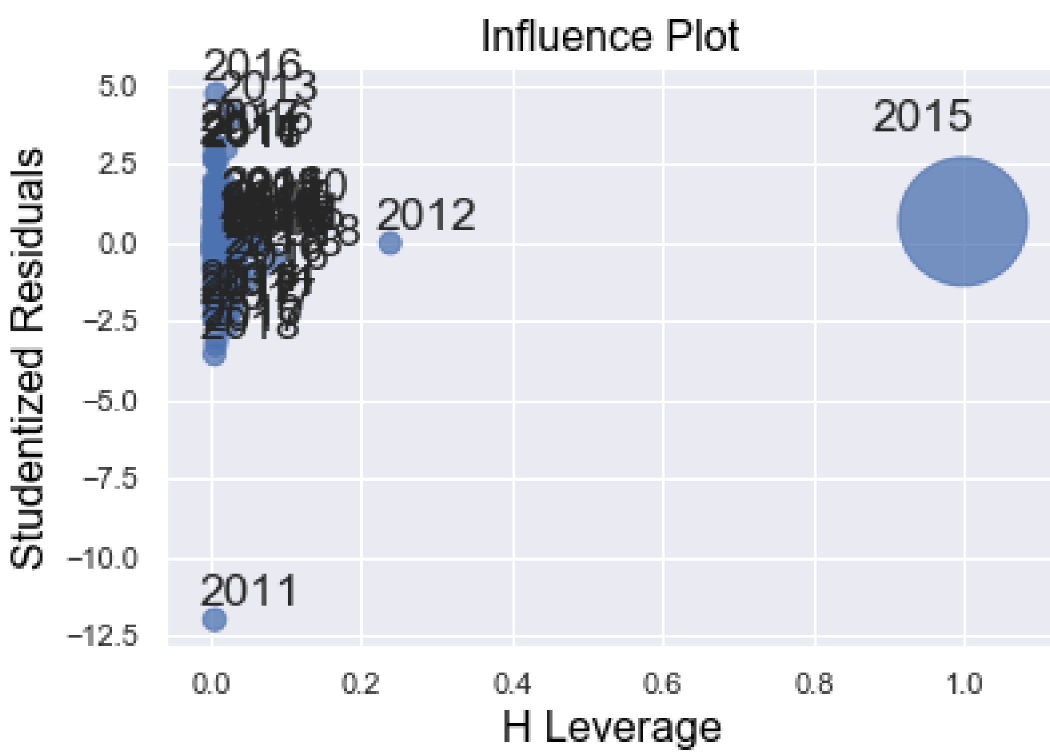
Out[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:              logret uw   R-squared:                       0.014
```

```
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.836
Date:                Thu, 27 Jun 2019   Prob (F-statistic):             0.0787
Time:                        11:17:02   Log-Likelihood:                -91.740
No. Observations:                 455   AIC:                             197.5
Df Residuals:                     448   BIC:                             226.3
Df Model:                           6
Covariance Type:                  HC3
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const              0.0250      0.019      1.312      0.190      -0.012       0.062
Logage            -0.0050      0.017     -0.303      0.762      -0.038       0.028
Adj Revenues       0.0001      0.001      0.116      0.908      -0.002       0.002
Adj Ebit           0.0008      0.007      0.120      0.905      -0.012       0.014
Techdummy          0.0350      0.021      1.691      0.091      -0.006       0.076
Non-Tech Dummy    -0.0100      0.019     -0.541      0.589      -0.046       0.026
Young dummy       -0.0146      0.018     -0.831      0.406      -0.049       0.020
Old dummy          0.0396      0.026      1.504      0.133      -0.012       0.091
Delisted dummy    -0.0555      0.047     -1.189      0.234      -0.147       0.036
==============================================================================
Omnibus:                      311.005   Durbin-Watson:                   2.018
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            15304.597
Skew:                          -2.293   Prob(JB):                         0.00
Kurtosis:                      31.040   Cond. No.                     3.07e+18
==============================================================================

Warnings:
[1] Standard Errors are heteroscedasticity robust (HC3)
[2] The smallest eigenvalue is 3.57e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```

In [17]:
```
# Outlier test of raw data regression
sm.graphics.influence_plot(modelA,criterion='cooks')
```

Out[17]:


Influence Plot
Studentized Residuals
H Leverage
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
0.0
0.2
0.4
0.6
0.8
1.0
2016
2013
2015
2012
2011


Influence Plot
Studentized Residuals
H Leverage
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
0.0
0.2
0.4
0.6
0.8
1.0
2016
2013
2015
2012
2011

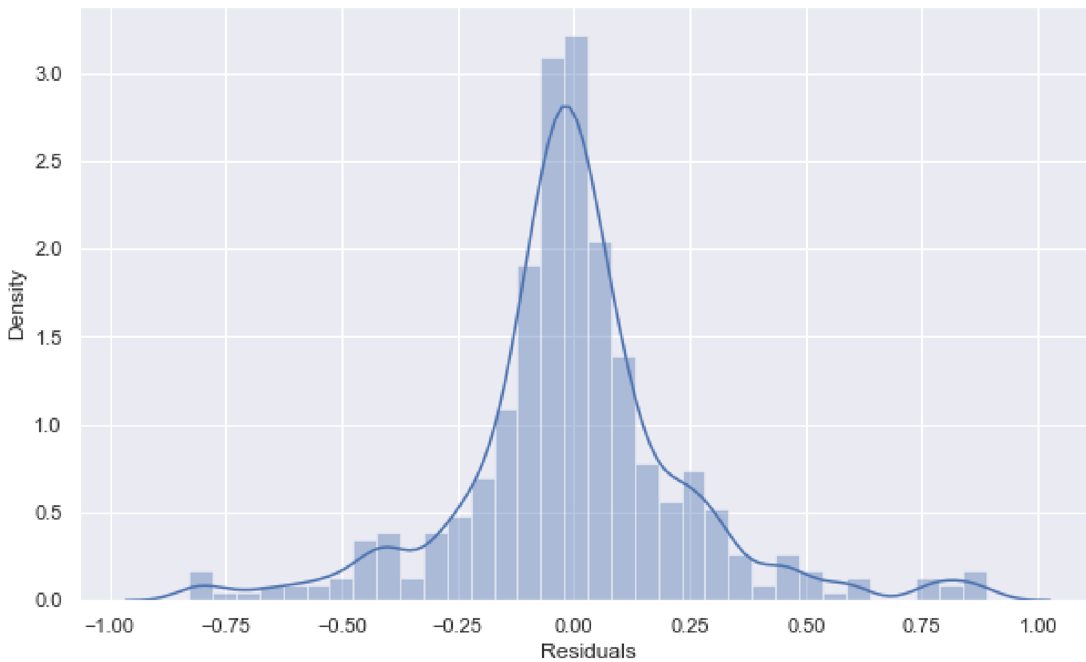
In [18]:
```
## !! Regression with winsorized data !! Men ikke revenue, 1 % ebit !!
ydata1 = pd.DataFrame()
ydata1 = logret['logret winz']
xdata1 = pd.DataFrame()
xdata1['LN Age'] = logage['logage']
xdata1['LN Revenue'] = np.log(1 + ipo['CurrencyAdjustedRevenues'])
xdata1['Ebit W'] = winzipotest['1% winz Ebit']
xdata1['LN Offer Size'] = osadj['offer size adjusted']
xdata1['Techdummy'] = np.where(technum==5,1,0)
xdata1['Young dummy'] = np.where(ipo['Age']<7, 1,0)
xdata1['Delisted dummy'] = delistednum

modelA1 = sm.OLS(ydata1,sm.add_constant(xdata1.astype(float)),missing='drop').fit(cov_
modelA1.summary()
```

Out[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            logret winz   R-squared:                       0.030
Model:                            OLS   Adj. R-squared:                  0.015
Method:                 Least Squares   F-statistic:                     1.617
Date:                Thu, 27 Jun 2019   Prob (F-statistic):              0.128
Time:                        11:17:03   Log-Likelihood:                -8.2437
No. Observations:                 455   AIC:                             32.49
Df Residuals:                     447   BIC:                             65.45
Df Model:                           7
Covariance Type:                  HC3
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const             -0.0004      0.057     -0.007      0.995      -0.111       0.111
LN Age            -0.0104      0.016     -0.655      0.512      -0.042       0.021
LN Revenue        -0.0066      0.012     -0.533      0.594      -0.031       0.018
Ebit W             0.0064      0.015      0.424      0.671      -0.023       0.036
LN Offer Size      0.0148      0.006      2.454      0.014       0.003       0.027
Techdummy          0.0527      0.034      1.563      0.118      -0.013       0.119
Young dummy       -0.0454      0.037     -1.215      0.224      -0.119       0.028
Delisted dummy    -0.0646      0.042     -1.545      0.122      -0.146       0.017
==============================================================================
Omnibus:                       37.283   Durbin-Watson:                   1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              137.569
Skew:                           0.237   Prob(JB):                     1.34e-30
Kurtosis:                       5.652   Cond. No.                         34.0
==============================================================================
```

```
Warnings:
[1] Standard Errors are heteroscedasticity robust (HC3)
"""
```

In [19]:
```
# Residual distribution plot
distfig3, ax3 = plt.subplots()
distfig3.set_size_inches(10, 6)
residuals = modelA1.resid
sns.distplot(residuals)
plt.xlabel('Residuals')
plt.ylabel('Density')
```

Out[19]:
```
Text(0, 0.5, 'Density')
```



In [20]:
```
## Checking for heteroscedasticity
distfig3, ax3 = plt.subplots()
distfig3.set_size_inches(10, 6)
residuals = modelA1.resid
FittedVal = modelA1.fittedvalues
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
sns.scatterplot(FittedVal,residuals)


# Breusch pagan test
```

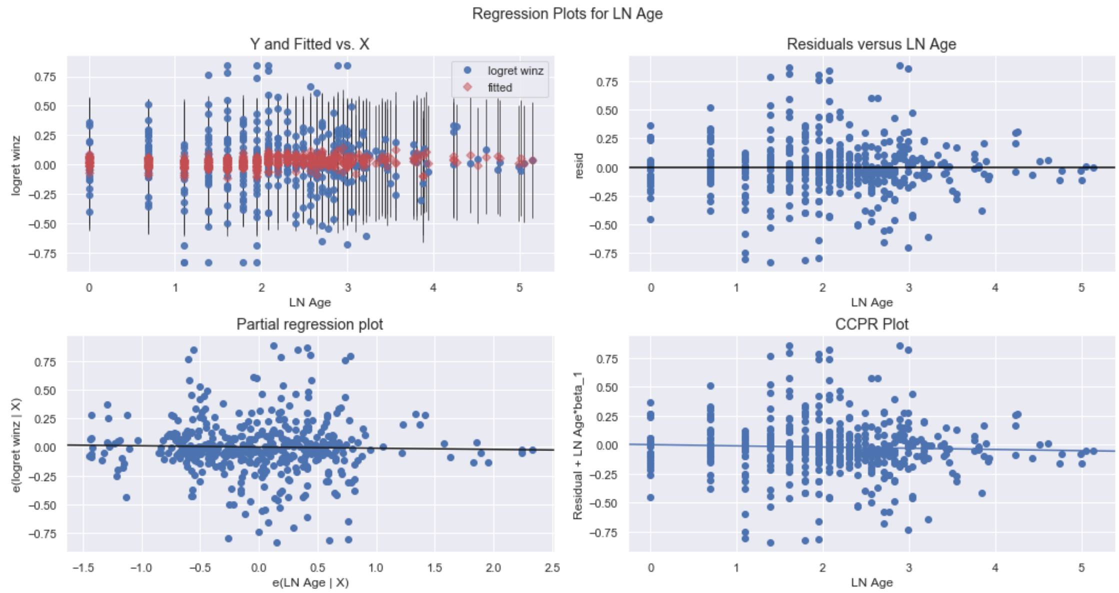
```
name = ['Lagrange multiplier statistic', 'p-value',
        'f-value', 'f p-value']
test = sms.het_breuschpagan(modelA1.resid, modelA1.model.exog)
lzip(name, test)
```

Out[20]:

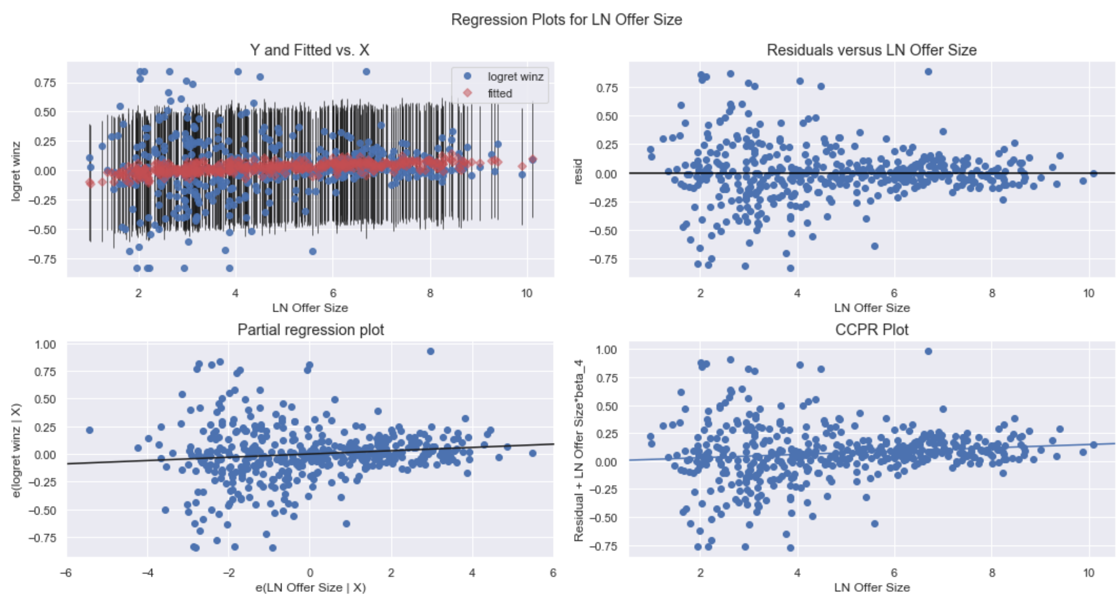
```
[('Lagrange multiplier statistic', 57.55490972389226),
 ('p-value', 4.636394028373317e-10),
 ('f-value', 9.247295242256726),
 ('f p-value', 1.0514042386573649e-10)]
```

In [21]:

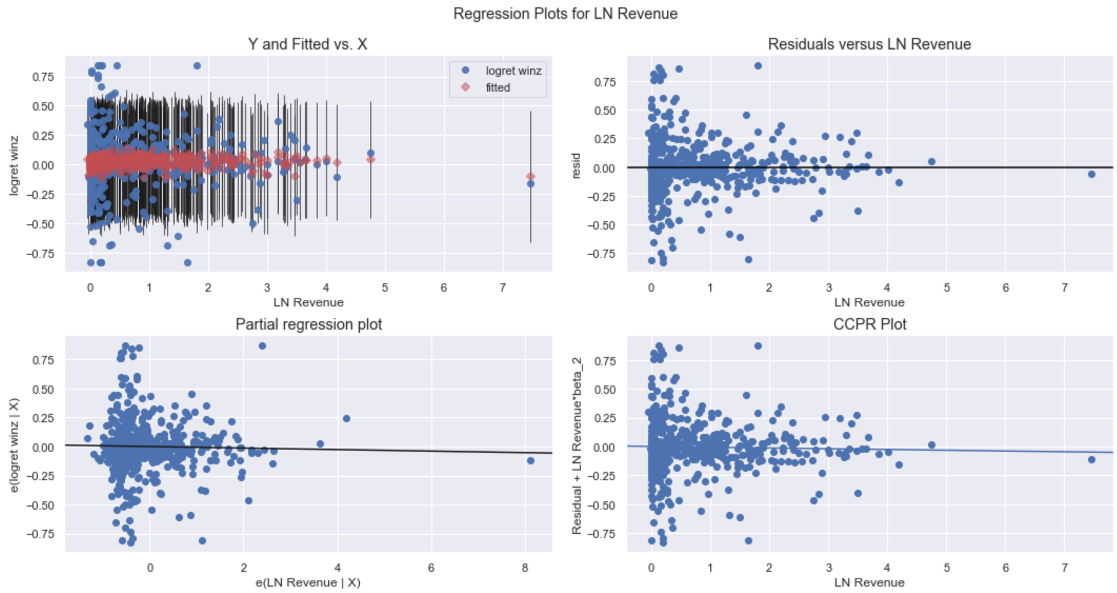
```
# This produces our four regression plots for Log Age
fig0 = plt.figure(figsize=(15,8))
fig0 = sm.graphics.plot_regress_exog(modelA1, "LN Age", fig=fig0)
```



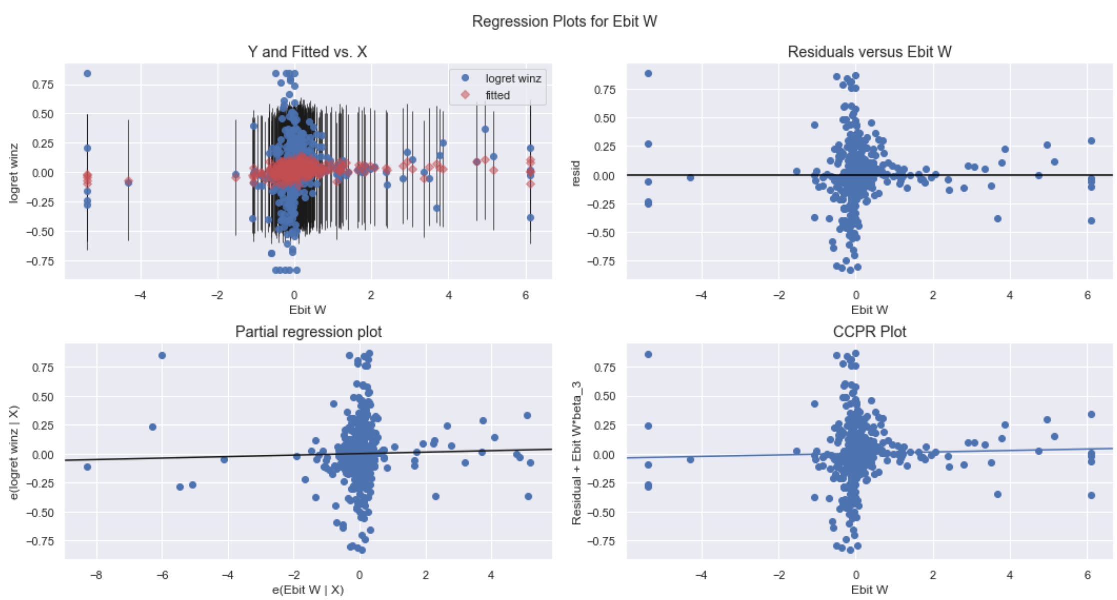
In [22]:
```
# This produces our four regression plots for Adjusted Offer Size
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(modelA1, "LN Offer Size", fig=fig)
```



In [23]:
```
# This produces our four regression plots for Earnings

fig1 = plt.figure(figsize=(15,8))
fig1 = sm.graphics.plot_regress_exog(modelA1, "LN Revenue", fig=fig1)
```

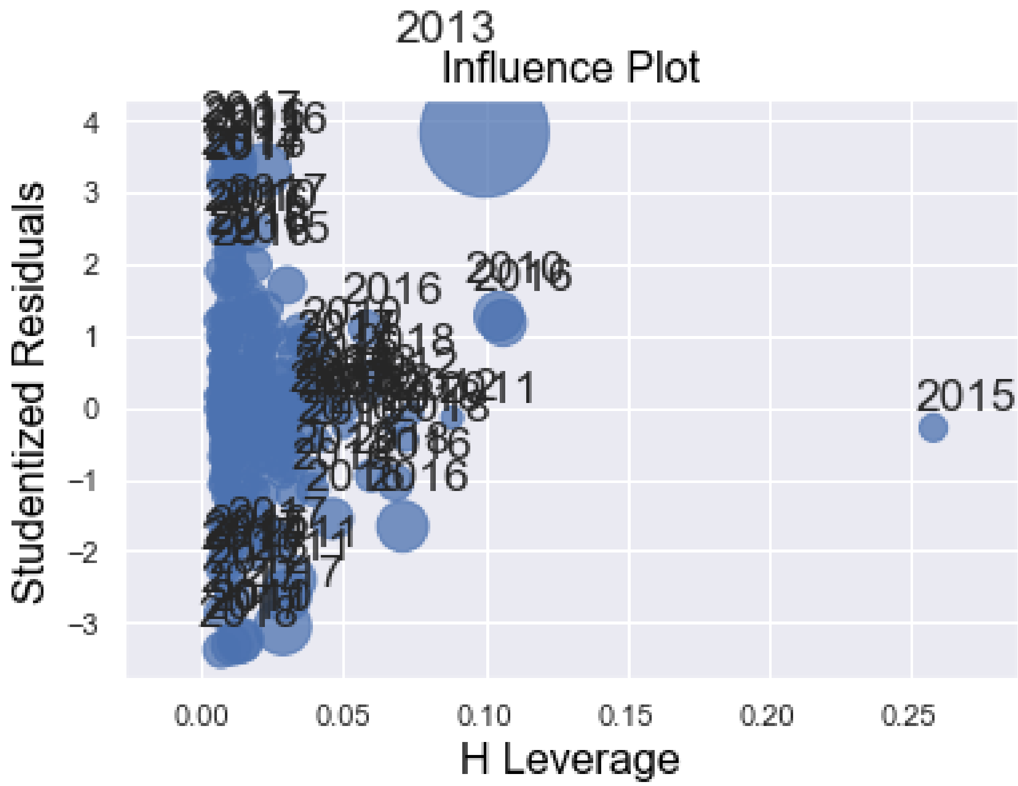


In [24]:
```
# This produces our four regression plots for Ebit

fig2 = plt.figure(figsize=(15,8))
fig2 = sm.graphics.plot_regress_exog(modelA1, "Ebit W", fig=fig2)
```

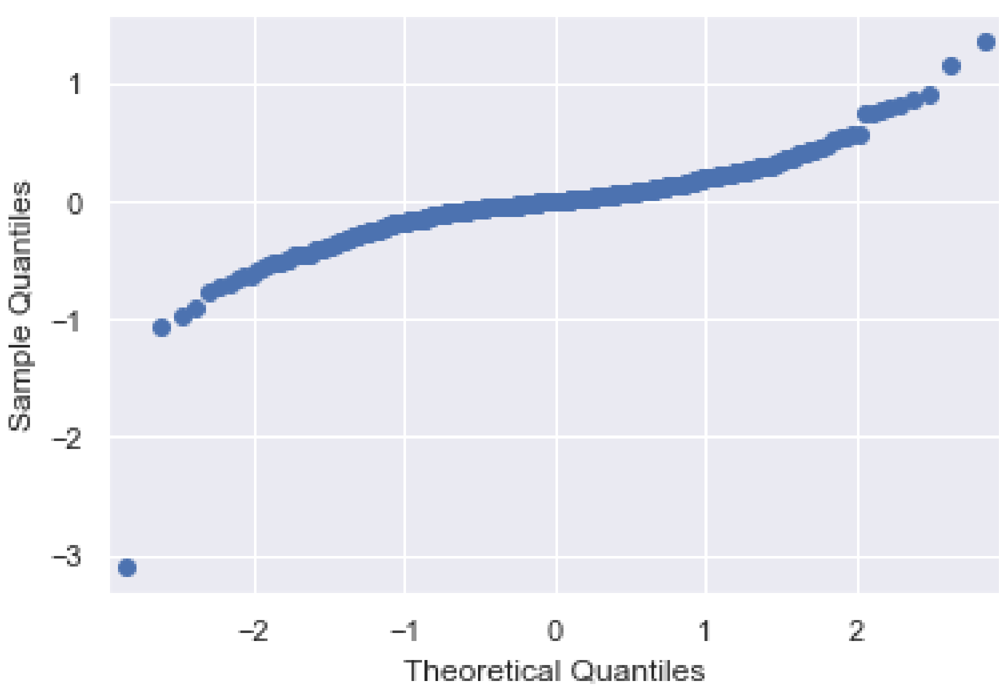


In [25]:
```
# Sensitivity Analysis #
sm.qqplot(modelA.resid) ## Normal probability plot OF non winsorized dataset
sm.qqplot(modelA1.resid) ## Normal probability plot OF winsorized dataset ##
sm.graphics.influence_plot(modelA1,criterion='cooks') # Outlier test of winsorized da
```

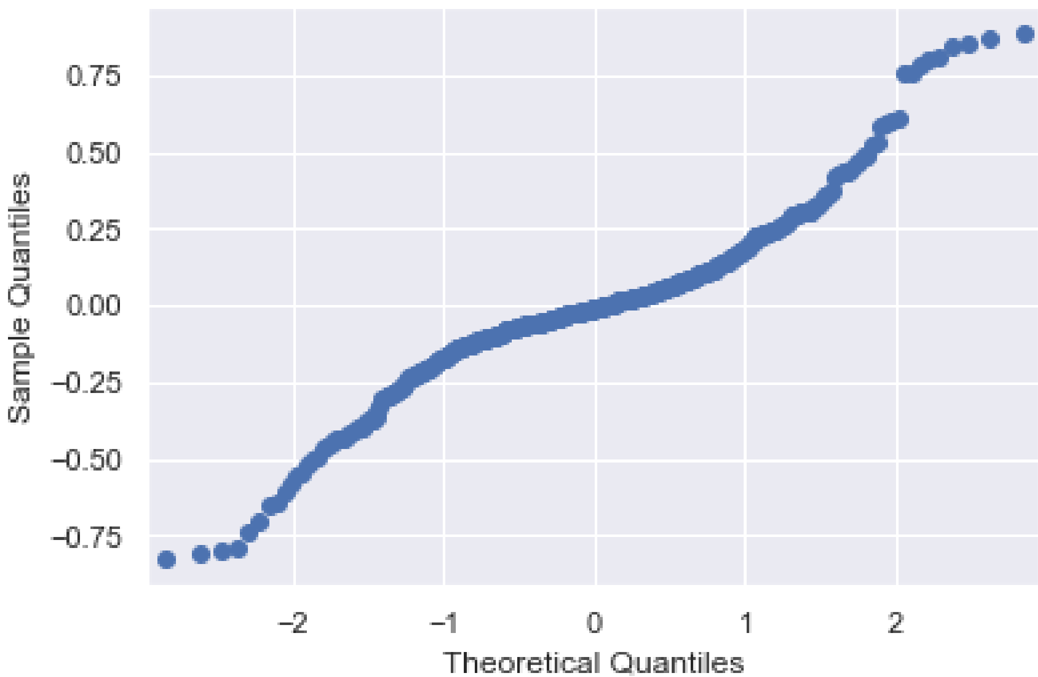
Out[25]:

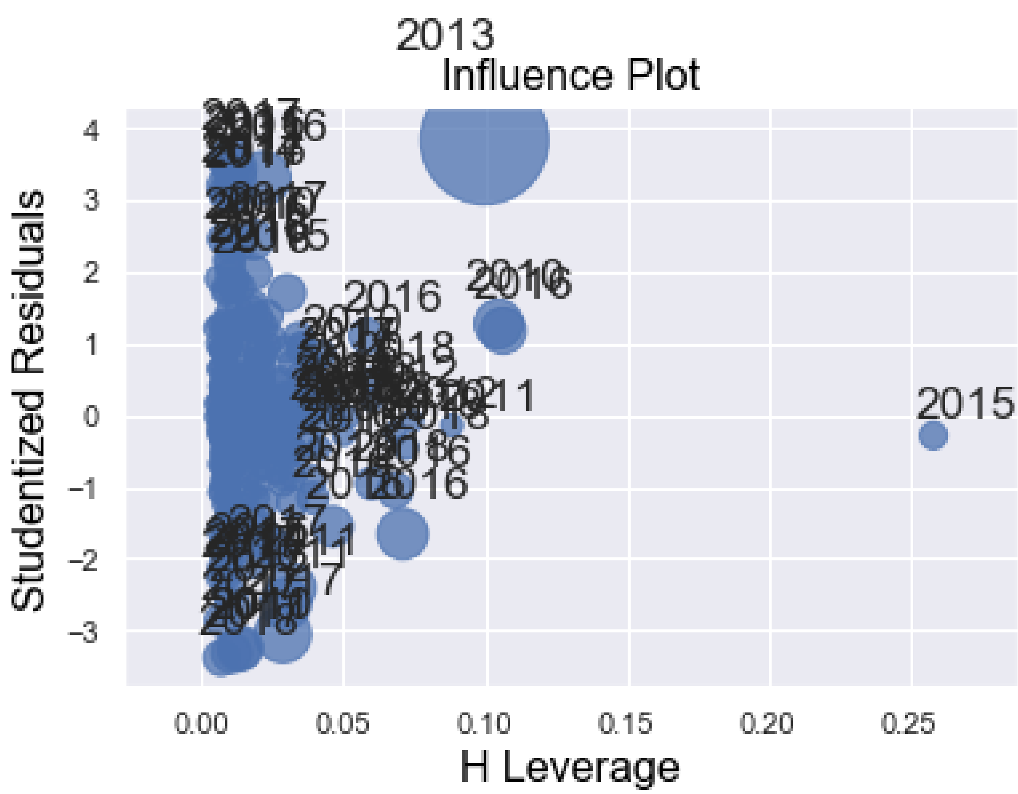


2013
Influence Plot
Studentized Residuals
H Leverage
4
3
2
1
0
−1
−2
−3
0.00
0.05
0.10
0.15
0.20
0.25
2016
2015



Sample Quantiles
Theoretical Quantiles
1
0
−1
−2
−3
−2
−1
0
1
2


Sample Quantiles
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−2
−1
0
1
2
Theoretical Quantiles


2013
Influence Plot
Studentized Residuals
4
3
2
1
0
−1
−2
−3
2016
2015
0.00
0.05
0.10
0.15
0.20
0.25
H Leverage

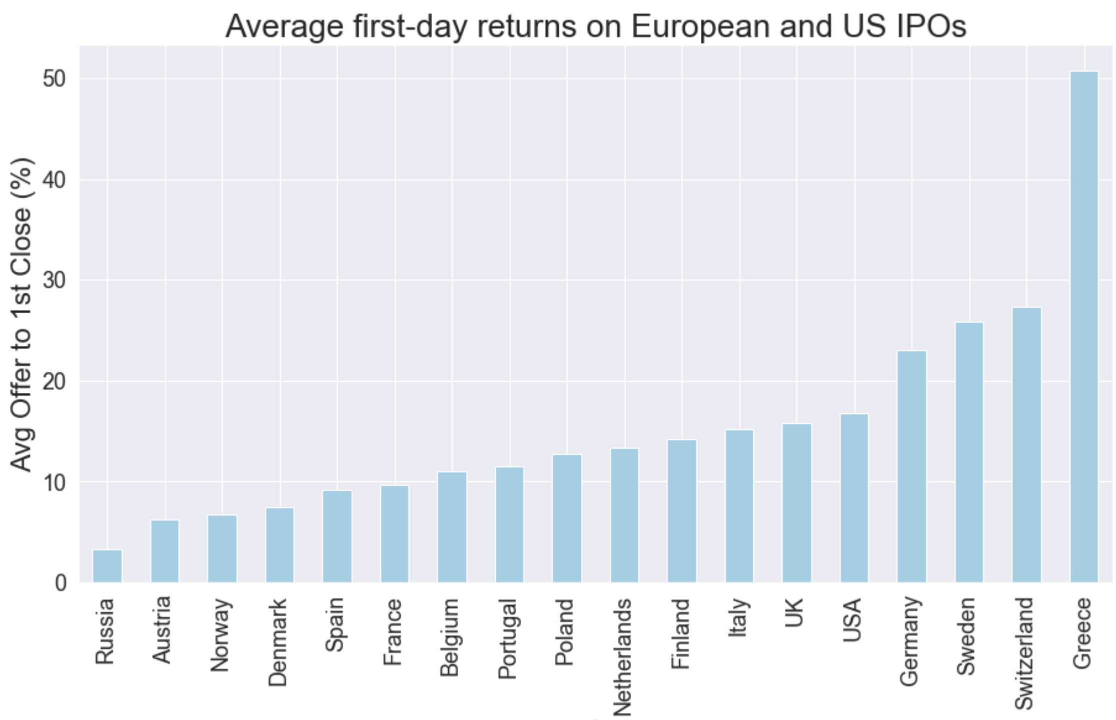
In [26]:
```python
## TECH SPECIFIC REGRESSION ##
## Running regression for only technology firms. i.e. extracting data from the previo
# otherwise dont include in regression.

Ydata = pd.DataFrame()
Xdata = pd.DataFrame()
Ydata = logret['logret winz']
Xdata['LN Revenue'] = xdata1['LN Revenue']
Xdata['Ebit W'] = xdata1['Ebit W']
Xdata['LN Offer Size'] = osadj['offer size adjusted']
#Xdata['Young dummy'] = np.where(ipo['Age']<=7, 1,0)
#Xdata['Delisted dummy'] = delistednum
Ydata = logret['logret winz'][xdata['Techdummy']>0]
Xdata = Xdata[xdata['Techdummy']>0]

Techmodel = sm.OLS(Ydata,sm.add_constant(Xdata),missing='drop').fit(cov_type='HC3')
Techmodel.summary()
```

Out[26]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            logret winz   R-squared:                       0.055
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     2.484
Date:                Thu, 27 Jun 2019   Prob (F-statistic):             0.0684
Time:                        11:17:11   Log-Likelihood:               -0.16911
No. Observations:                  70   AIC:                             8.338
Df Residuals:                      66   BIC:                             17.33
Df Model:                           3
Covariance Type:                  HC3
=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const            -0.0184      0.083     -0.223      0.824      -0.180       0.144
LN Revenue       -0.0417      0.033     -1.277      0.202      -0.106       0.022
Ebit W            0.0360      0.023      1.542      0.123      -0.010       0.082
LN Offer Size     0.0275      0.015      1.855      0.064      -0.002       0.056
==============================================================================
Omnibus:                        4.286   Durbin-Watson:                   1.740
Prob(Omnibus):                  0.117   Jarque-Bera (JB):                3.427
Skew:                           0.438   Prob(JB):                        0.180
Kurtosis:                       3.640   Cond. No.                         13.2
==============================================================================
```

```
Warnings:
[1] Standard Errors are heteroscedasticity robust (HC3)
"""
```

In [27]:
```python
# Figure showing avg first day returns on european and us IPOs
country = pd.read_excel('IPO_Country.xlsx')
country.set_index('Country', inplace = True, )
countrytable = pd.DataFrame()
countrytable['Avg Initial return'] = country['Avg Initial return']*100

Countryfigure = countrytable.plot(kind='bar', legend=False, colormap='Paired',
                                figsize=(15,8), fontsize=18, style= 'seaborn',)
Countryfigure.set_xlabel("Country", fontsize=0)
Countryfigure.set_ylabel("Avg Offer to 1st Close (%)", fontsize=22)
Countryfigure.set_title('Average first-day returns on European and US IPOs', fontsize
```

Out[27]:
```
Text(0.5, 1.0, 'Average first-day returns on European and US IPOs')
```



In [28]:
```python
## 1 Sided T-test to check if init return is stat sign. diff from zero!
ipow = pd.DataFrame(index=ipo.index)
ipow['IR winz'] = scipy.stats.mstats.winsorize(df['ipo'], limits=0.01)
stats.ttest_1samp(ipow['IR winz'],0) # Yes, stat. sign. diff from zero because we rej
#ipow.describe()
```

Out[28]:
```
Ttest_1sampResult(statistic=4.4353569646445195, pvalue=1.154551184419486e-05)
```

In [35]:
```python
# Kruskal wallis and levenes test
# Hypothesis 2: No sign. difference between tech variance and non-tech variance

# Split the sample: Tech and rest of sample
Techframe = pd.DataFrame()
Nontechframe = pd.DataFrame()
Techframe = ipow['IR winz'][xdata['Techdummy']>0]
Nontechframe = ipow['IR winz'][xdata['Techdummy']==0]

#Levenes test of variances
stats.levene(Techframe, Nontechframe) # Low pval, no homo, hence
# We need to conduct a Welchs t-test.

#Normality tests
#diff = Techframe -Nontechframe
#stats.shapiro(Nontechframe) # Significant pval
#stats.shapiro(Techframe) # Significant pval
#stats.shapiro(diff) # Significant pval
#sns.distplot(Nontechframe)

# Hypothesis 3: No sign. difference between tech and non-tech ipos
#Kruskal wallis H-test
stats.kruskal(Techframe, Nontechframe)
# There is not a difference between tech returns and non tech ret.
#Techframe.describe()
```

Out[35]:
```
KruskalResult(statistic=0.8879017252652394, pvalue=0.34604655451388877)
```

In [38]:
```python
# Kruskal wallis test of hypothesis 4: Young vs Old companies

# Splitting young and old frames
Youngframe = pd.DataFrame()
Oldframe = pd.DataFrame()
Youngframe = ipow['IR winz'][xdata['Young dummy']>0]
Oldframe = ipow['IR winz'][xdata['Old dummy']>0]

#Kruskal wallis
stats.kruskal(Youngframe,Oldframe)
#Oldframe.describe()
```

Out[38]:
```
KruskalResult(statistic=1.3690480611783002, pvalue=0.24197550729844092)
```

In [31]:
```python
## Exchanges characteristics table
Exchanges2 = pd.DataFrame()
Exchanges2['Primary Exchange'] = ipo['PrimaryExchange']
Exchanges2['EBIT'] = ipo['CurrencyEbit']
Exchanges2['Revenue'] = ipo['CurrencyRevenues']
Exchanges2['Age'] = ipo['Age']
Exchanges2['Init Ret'] = ipo['OfferTo1stClose']
```

```python
Exchanges3 = Exchanges2.groupby('Primary Exchange')
Exchanges2['Primary Exchange'].value_counts()
Exchanges3.mean()
```

Out[31]:

Primary Exchange	EBIT	Revenue	Age	Init Ret
Copenhagen	4.604688e+08	1.502190e+10	15.875000	8.940336
FN Denmark	-3.683822e+06	5.768799e+06	10.142857	9.663138
FN Finland	1.605689e+07	2.733977e+08	15.541667	0.543334
FN Stockholm	3.468654e+05	8.341040e+07	8.709677	8.290332
Helsinki	4.094416e+08	2.787897e+09	12.869565	5.333792
Nordic GM	-4.013835e+06	2.217212e+07	14.333333	-16.423745
Oslo	5.048941e+08	3.652262e+09	18.438356	2.711652
Spotlight	-1.556451e+06	1.003447e+07	8.197531	9.278488
Stockholm	1.903284e+08	2.397588e+09	15.372093	10.408394