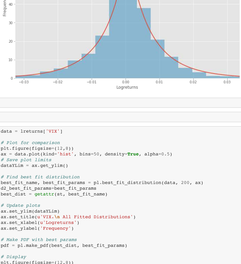
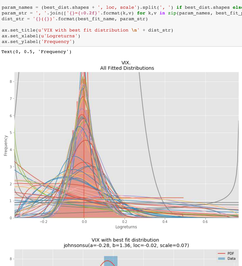
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime
import statsmodels.api as sm
import pfinance as pf
import scipy.stats as st
import os
import pickle as pl
import seaborn as sns
from scipy import stats
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
from statsmodels import regression
from pandas_datareader import data as pdr

yf.pdr_override()

df01 = pdr.get_data_yahoo("^GSPC", start="2000-07-03", end="2019-06-03").Close.rename('GSPC')
df12 = pdr.get_data_yahoo("^VIX", start="2000-07-03", end="2019-06-03").Close.rename('VIX')
index_prime=pd.concat([df01, df12], axis=1).dropna()
lreturns=np.log(index_prices/index_prices.shift(1))
lreturns.dropna(inplace=True)
```

```
[**********************100%**********************]  1 of 1 downloaded
[**********************100%**********************]  1 of 1 downloaded
```

In [2]:
```
data = lreturns['GSPC']

# Plot for comparison
plt.figure(figsize=(12,5))
ax = data.plot(kind='hist', bins=50, density=True, alpha=0.5)
# Save plot limits
dataYLim = ax.get_ylim()

# Find best fit distribution
best_fit_name, best_fit_params = pl.best_fit_distribution(data, 200, ax)
dl_best_fit_params=best_fit_params
best_dist = getattr(st, best_fit_name)

# Update plots
ax.set_ylim(dataYLim)
ax.set_title('GSPC\n All Fitted Distributions')
ax.set_xlabel('Logreturns')
ax.set_ylabel('Frequency')

# Make PDF with best params
pdf = pl.make_pdf(best_dist, best_fit_params)

# Display
plt.figure(figsize=(12,5))
ax = pdf.plot(lw=2, label='PDF', legend=True)
data.plot(kind='hist', bins=50, density=True, alpha=0.5, label='Data', legend=True, ax=ax)

param_names = (best_dist.shapes + ', loc, scale').split(', ') if best_dist.shapes else ['loc', 'scale']
param_str = ', '.join(['{}={:0.2f}'.format(k,v) for k,v in zip(param_names, best_fit_params)])
dist_str = '{}({})'.format(best_fit_name, param_str)

ax.set_title('GSPC with best fit distribution \n' + dist_str)
ax.set_xlabel('Logreturns')
ax.set_ylabel('Frequency')
```

Out[2]: Text(0, 0.5, 'Frequency')



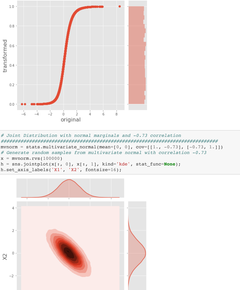


In [ ]:

In [ ]:

In [3]:
```
data = lreturns['VIX']

# Plot for comparison
plt.figure(figsize=(12,5))
ax = data.plot(kind='hist', bins=50, density=True, alpha=0.5)
# Save plot limits
dataYLim = ax.get_ylim()

# Find best fit distribution
best_fit_name, best_fit_params = pl.best_fit_distribution(data, 200, ax)
dl_best_fit_params=best_fit_params
best_dist = getattr(st, best_fit_name)

# Update plots
ax.set_ylim(dataYLim)
ax.set_title('VIX\n All Fitted Distributions')
ax.set_xlabel('Logreturns')
ax.set_ylabel('Frequency')

# Make PDF with best params
pdf = pl.make_pdf(best_dist, best_fit_params)

# Display
plt.figure(figsize=(12,5))
ax = pdf.plot(lw=2, label='PDF', legend=True)
data.plot(kind='hist', bins=50, density=True, alpha=0.5, label='Data', legend=True, ax=ax)

param_names = (best_dist.shapes + ', loc, scale').split(', ') if best_dist.shapes else ['loc', 'scale']
param_str = ', '.join(['{}={:0.2f}'.format(k,v) for k,v in zip(param_names, best_fit_params)])
dist_str = '{}({})'.format(best_fit_name, param_str)

ax.set_title('VIX with best fit distribution \n' + dist_str)
ax.set_xlabel('Logreturns')
ax.set_ylabel('Frequency')
```

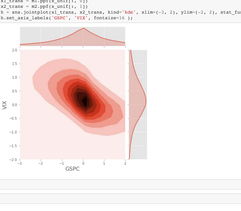
Out[3]: Text(0, 0.5, 'Frequency')



In [ ]:

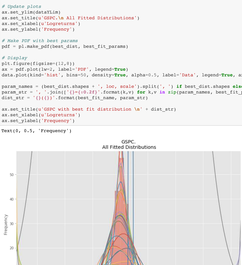
In [ ]:

In [4]:
```
x = stats.uniform(0, 1).rvs(10000)
gennormt = stats.distributions.gennormt(dl_best_fit_params[0])
x_trans = gennormt.ppf(x)
h = sns.jointplot(x, x_trans, stat_func=None)
h.set_axis_labels('original', 'transformed', fontsize=16);
```

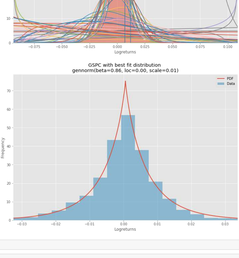


In [5]:
```
x_trans_trans = gennormt.cdf(x_trans)
h = sns.jointplot(x_trans, x_trans_trans, stat_func=None)
h.set_axis_labels('original', 'transformed', fontsize=16);
```

In [6]:
```
x = stats.uniform(0, 1).rvs(10000)
johnsonsu2 = stats.distributions.johnsonsu(dl_best_fit_params[0],dl_best_fit_params[1])
x_trans = johnsonsu2.ppf(x)
h = sns.jointplot(x, x_trans, stat_func=None)
h.set_axis_labels('original', 'transformed', fontsize=16);
```



In [7]:
```
x_trans_trans = johnsonsu2.cdf(x_trans)
h = sns.jointplot(x_trans, x_trans_trans, stat_func=None)
h.set_axis_labels('original', 'transformed', fontsize=16);
```



In [8]:
```
# Joint Distribution with normal marginals and +0.7 correlation
##############################################################################
mvnorm = stats.multivariate_normal(mean=[0, 0], cov=[[1., -0.7], [-0.7, 1.]])
# Generate random samples from multivariate normal with correlation -0.7
x = mvnorm.rvs(10000)
h = sns.jointplot(x[:, 0], x[:, 1], kind='hex', stat_func=None);
h.set_axis_labels('X1', 'X2', fontsize=16);
```


In [ ]:

In [ ]:

In [10]:
```
# Joint Distribution assuming gennormt and johnsonsu marginals
##############################################################################
norm = stats.norm()
x_unif = norm.cdf(x)
h1 = stats.distributions.gennormt(dl_best_fit_params[0])
x1 = h1.ppf(x_unif)
h2 = stats.distributions.johnsonsu(dl_best_fit_params[0],dl_best_fit_params[1])
x2 = h2.ppf(x_unif)
h = sns.jointplot(x1, x2, kind='hex', xlim=(-3, 3), ylim=(-3, 3), stat_func=None);
h.set_axis_labels('GSPC', 'VIX', fontsize=16 );
```


In [ ]:

In [ ]: