KyotoUx-009x (/github/ryo0921/KyotoUx-009x/tree/master)
/ 06 (/github/ryo0921/KyotoUx-009x/tree/master/06)

Stochastic Processes: Data Analysis and Computer Simulation

Stochastic processes in the real world

1. Time variations and distributions of real world processes

1.1. Test score distribution

TOEIC Listening & Reading Test

- TOEIC (Test of English for International Communication) is the most popular test for English communication skills in Japan. Below we show the official data from the most recent test.

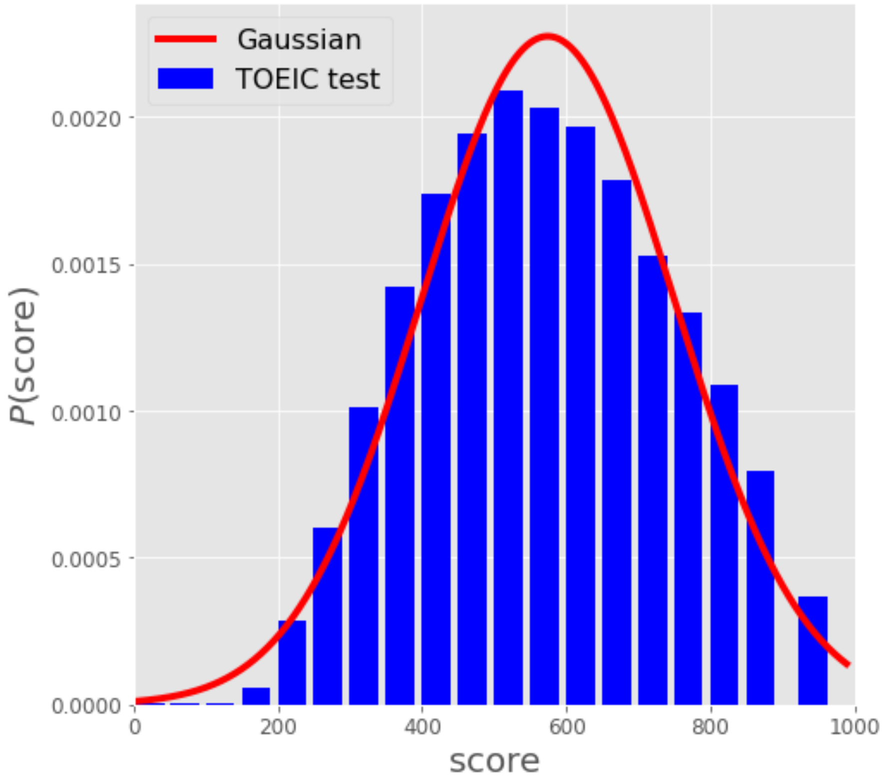
Test date	Jan. 2017
Number of examinees	134,269
Maximum Score	990
Minimum Score	5
Mean Score	574.3
Standard Deviation	175.4

score interval	count		score interval	count		score interval	count
895-990	4,697		595-644	12,934		295-344	6,675
845-894	5,215		545-594	13,371		245-294	3,951
795-844	7,151		495-544	13,742		195-244	1,899
745-794	8,790		445-494	12,802		145-194	376
695-744	10,056		395-444	11,444		95-144	30
645-694	11,762		345-394	9,340		45-94	18
						0-44	16

</pre>

In [1]:
```
% matplotlib inline
import numpy as np  # import numpy library as np
import math         # use mathematical functions defined by the C standa
import matplotlib.pyplot as plt # import pyplot library as plt
plt.style.use('ggplot') # use "ggplot" style for graphs
pltparams = {'legend.fontsize': 16, 'axes.labelsize': 20, 'axes.titlesiz
             'xtick.labelsize': 12, 'ytick.labelsize':12, 'figure.figsiz
plt.rcParams.update(pltparams)
```

In [2]:
```
data=np.array([[895,990,4697],[845,894,5215],[795,844,7151],
               [745,794,8790],[695,744,10056],[645,694,11762],[595,644,1
               [545,594,13371],[495,544,13742],[445,494,12802],[395,444,
               [345,394,9340],[295,344,6675],[245,294,3951],[195,244,189
               [145,194,376],[95,144,30],[45,94,18],[0,44,16],])
tot = 134269 # set total number of examinees
ave = 574.3  # set average
std = 175.4  # set standard deviation
x = np.arange(0,990,1) # create array of x from 0 to 1 with increment 0.
y = np.exp(-(x-ave)**2/(2*std**2))/np.sqrt(2*np.pi*std**2) # create arra
fig, ax = plt.subplots(subplot_kw={'xlabel':r'$\mathrm{score}$', 'ylabel
ax.bar((data[:,0]+data[:,1])/2,data[:,2]/tot/(data[:,1]-data[:,0]),width
ax.plot(x,y,linewidth=4,color='r',label=r'Gaussian') # plot y vs. x with
ax.set_xlim(0,1000)     # set x-range
ax.legend(loc=2)        # set legends
plt.show()              # display plots
```



1.2. Size distribution of earthquake magnitudes

Gutenberg–Richter law

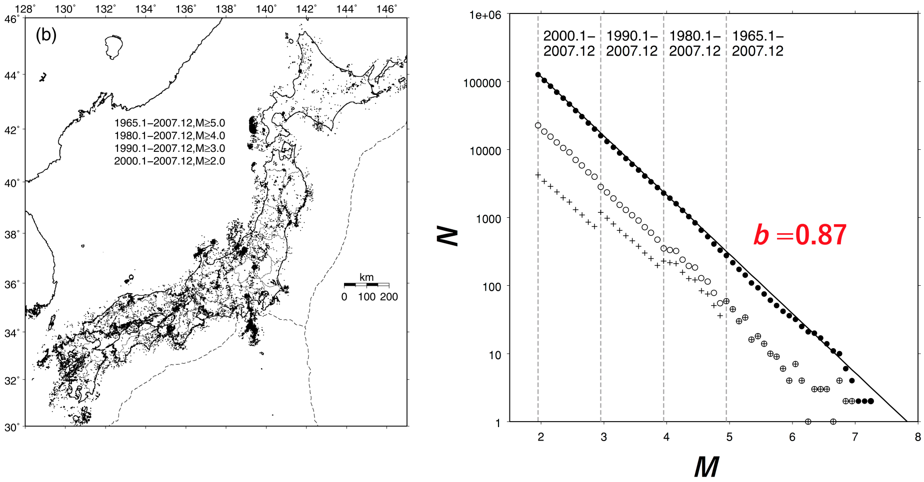
- B. Gutenberg and C.F. Richter, Annali di Geofisica, 9, 1–15 (1956).
- Eq.(J1) expresses the relationship between the magnitude M and the total number of earthquakes, in any given region and time period, of magnitude greater than or equal to M.

$$N = 10^{(a-bM)} \tag{J1}$$

- N is the number of events having a magnitude $\geq M$
- a is a normalization constant
- b is a parameter referred to as the "b-value" which is usually close to 1, with some regional variations depending on the local subsurface structures.

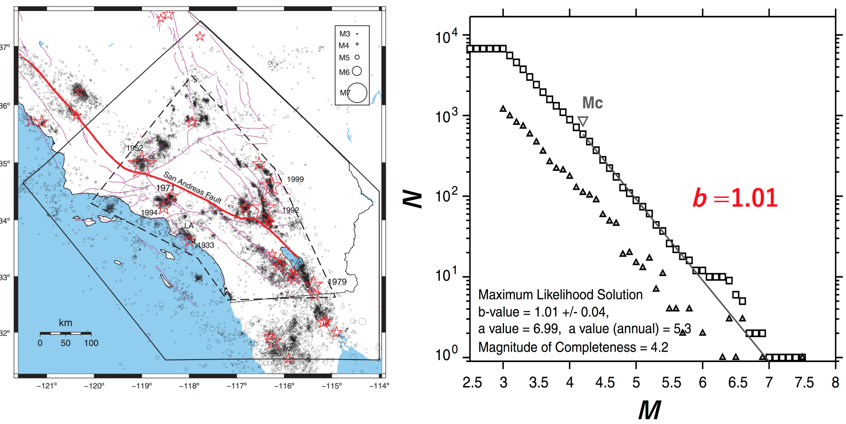
Japan

- F. Hirose and K. Maeda, Earth Planets Space, 63, 239–260 (2011).



California

- K. Hutton, J. Woessner, and E. Hauksson, Bull. Seismological Soc. America, 100, 423–446 (2010)



1.3. Return distribution of stock prices

Logarithmic change in price

- Consider the time series data for some stock price or market index $P(t)$.
- The relevant quantity is not the abolute value of the stock at any given time, but the return: the change in price after some time τ.
- For convenience one usually considers the logarithmic change $G_\tau(t)$, defined as

$$G_\tau(t) \equiv \log P(t+\tau) - \log P(t) \tag{J2}$$

- For small relative changes $\Delta_\tau P = P(t+\tau) - P(t) \ll P(t)$, this is equivalent to the relative change in price

$$G_\tau(t) = \ln\left[\frac{P(t+\tau)}{P(t)}\right] = \ln\left[\frac{P(t)+\Delta_\tau P(t)}{P(t)}\right] = \ln\left[1 + \frac{\Delta_\tau P(t)}{P(t)}\right]$$

using the Taylor expansion $\ln(1+x) = x - \frac{x^2}{2} + \cdots$ (for $|x| < 1$), we have

$$G_\tau(t) \simeq \frac{\Delta_\tau P(t)}{P(t)} = \frac{P(t+\tau) - P(t)}{P(t)},$$

In [3]:
```
## You may install pandas-datareader by typing the following command in
## conda install pandas-datareader
import pandas as pd # import pandas library as pd
from datetime import datetime
from pandas_datareader import data as pdr
from pandas_datareader import wb   as pwb
```

In [4]:
```
# Logarithmic return of price time series
def logreturn(Pt,tau=1):
    return np.log(Pt[tau:]) - np.log(Pt[0:-tau]) # Eq.(J2) : G_tau(t) =
# normalize data to have unit variance (<(x - <x>)^2> = 1)
def normalized(data):
    return (data/np.sqrt(np.var(data)))
# compute normalized probability distribution function
def pdf(data,bins=50):
    hist, edges = np.histogram(data[~np.isnan(data)], bins=bins, density
    edges   = (edges[:-1] + edges[1:])/2.0 # get bar centers
    nonzero = hist > 0.0                    # only keep non-zero points
    return edges[nonzero], hist[nonzero]
```

In [5]:

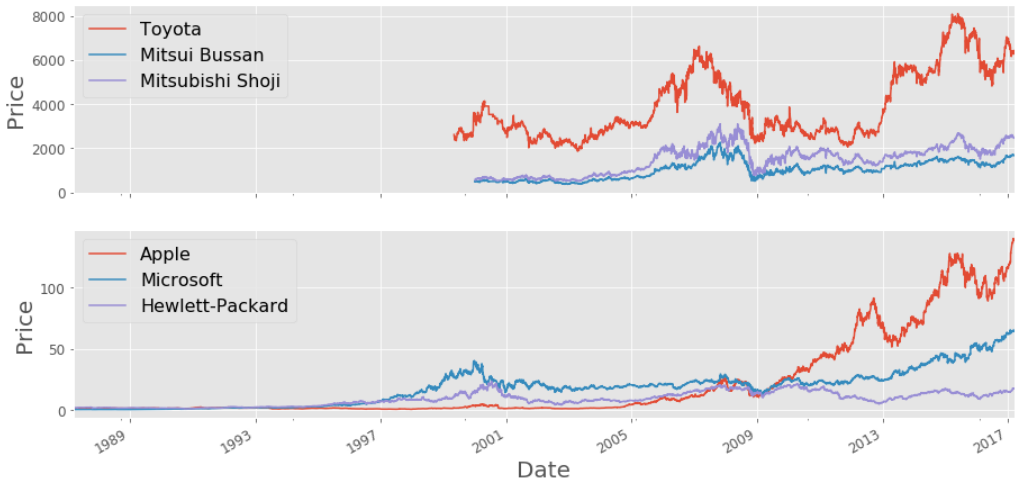
```
# define time interval
end_time   = datetime.now()
start_time = datetime(end_time.year - 30, end_time.month, end_time.day)
# get pandas data for Japanese and American stocks
toyota     = pdr.DataReader('7203','yahoo',start_time,end_time)
mitsui     = pdr.DataReader('8031','yahoo',start_time,end_time)
mitsubishi = pdr.DataReader('8058','yahoo',start_time,end_time)
apple      = pdr.DataReader('AAPL','yahoo',start_time,end_time)
msft       = pdr.DataReader('MSFT','yahoo',start_time,end_time)
hpq        = pdr.DataReader('HPQ', 'yahoo',start_time,end_time)
nikkei     = pdr.DataReader('^N225','yahoo',start_time,end_time)
sp500      = pdr.DataReader('^GSPC','yahoo',start_time,end_time)
toyota.tail()
```

Out[5]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-08	6399.0	6402.0	6360.0	6370.0	5242600	6271.30
2017-03-09	6411.0	6439.0	6397.0	6434.0	5010200	6334.31
2017-03-10	6498.0	6525.0	6480.0	6520.0	8240000	6418.98
2017-03-13	6490.0	6530.0	6467.0	6530.0	4737800	6428.82
2017-03-14	6500.0	6508.0	6451.0	6454.0	5582300	6354.00

In [6]:

```
fig,[ax,bx]=plt.subplots(figsize=(15,7.5),nrows=2,sharex=True,subplot_kw
for stock,lbl in zip([toyota,mitsui,mitsubishi],['Toyota','Mitsui Bussan
    stock['Adj Close'].plot(ax=ax,legend=True,label=lbl)
for stock,lbl in zip([apple,msft,hpq],['Apple','Microsoft','Hewlett-Pack
    stock['Adj Close'].plot(ax=bx,legend=True,label=lbl)
plt.show()
```



In [7]:

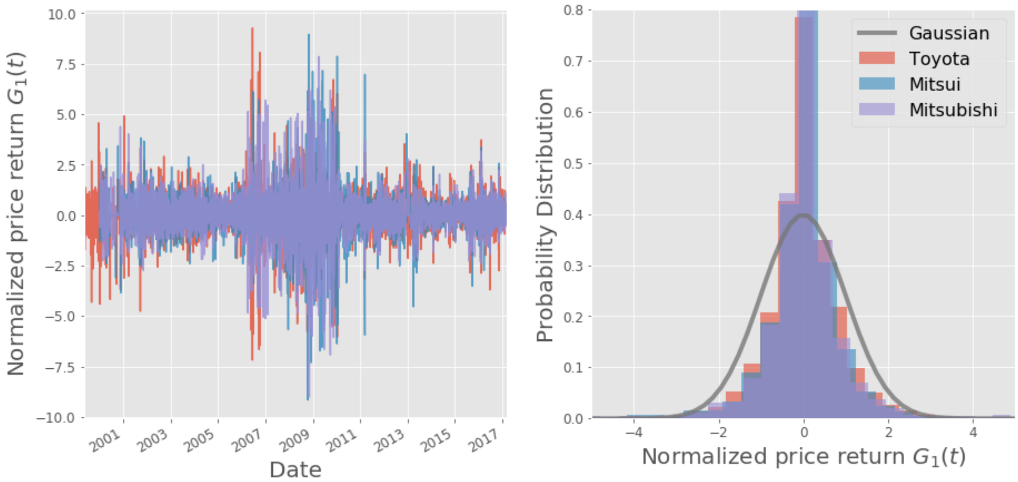
```
# add logarithmic return data to pandas DataFrame data using the 'Adjust
def computeReturn(data, name, tau):
    data[name]=pd.Series(normalized(logreturn(data['Adj Close'].values,
for stock in [toyota,mitsui,mitsubishi,nikkei,apple,msft,hpq,sp500]:
    computeReturn(stock,'Return d1',1)
toyota.tail()
```

Out[7]:

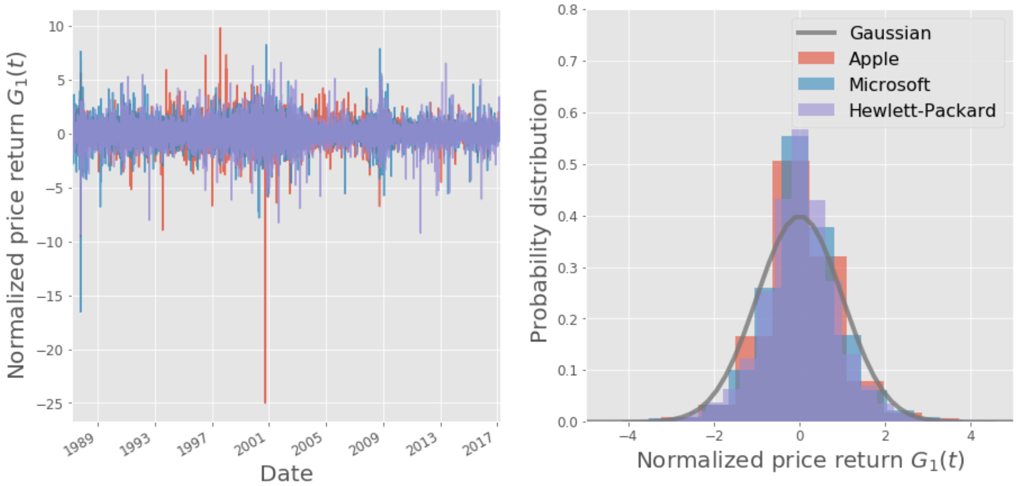
	Open	High	Low	Close	Volume	Adj Close	Return d1
Date							
2017-03-08	6399.0	6402.0	6360.0	6370.0	5242600	6271.30	0.407992
2017-03-09	6411.0	6439.0	6397.0	6434.0	5010200	6334.31	0.541896
2017-03-10	6498.0	6525.0	6480.0	6520.0	8240000	6418.98	0.062513
2017-03-13	6490.0	6530.0	6467.0	6530.0	4737800	6428.82	-0.477747
2017-03-14	6500.0	6508.0	6451.0	6454.0	5582300	6354.00	NaN

In [8]:

```
fig, [ax,bx] = plt.subplots(figsize=(15.0, 7.5),ncols=2)
for data,lbl in zip([toyota,mitsui,mitsubishi],['Toyota','Mitsui','Mitsu
    data['Return d1'].plot(ax=ax,alpha=0.8)
    data['Return d1'].hist(ax=bx,alpha=0.6,normed=True,bins=40,lw=0,labe
x = np.linspace(-5,5)
bx.plot(x,np.exp(-x**2/2)/np.sqrt(2*np.pi),lw=4,alpha=0.8,label='Gaussia
bx.legend()
ax.set_ylabel('Normalized price return $G_1(t)$')
bx.set_xlabel('Normalized price return $G_1(t)$')
bx.set_ylabel('Probability Distribution')
bx.set_xlim([-5,5])
bx.set_ylim([0,0.8])
plt.show()
```



In [9]:
```
fig,[ax,bx]=plt.subplots(figsize=(15.0,7.5),ncols=2)
for data,lbl in zip([apple,msft,hpq],['Apple','Microsoft','Hewlett-Packa
    data['Return d1'].plot(ax=ax,alpha=0.8,label=lbl)
    data['Return d1'].hist(ax=bx,alpha=0.6,normed=True,bins=40,lw=0,labe
x = np.linspace(-5,5)
bx.plot(x,np.exp(-x**2/2)/np.sqrt(2*np.pi),lw=4,alpha=0.8,label='Gaussia
bx.legend()
ax.set_xlabel('Date')
ax.set_ylabel('Normalized price return $G_1(t)$')
bx.set_xlabel('Normalized price return $G_1(t)$')
bx.set_ylabel('Probability distribution')
bx.set_xlim([-5,5])
bx.set_ylim([0,0.8])
plt.show()
```



In [10]:
```
fig,ax=plt.subplots(subplot_kw={'xlabel':r'Absolute normalized price ret
# probability distribution for stocks
for stock,lbl in zip([toyota,mitsui,mitsubishi,apple,msft,hpq],
        ['Toyota','Mitsui','Mitsubishi','Apple','Microsoft','Hewlett-Pac
    edges,hist=pdf(np.abs(stock['Return d1']),bins=30)
    ax.plot(edges,hist,label=lbl,lw=3)
# probability distribution for stock indices
for stock,lbl in zip([sp500,nikkei],['S&P 500','Nikkei 225']):
    edges,hist=pdf(np.abs(stock['Return d1']),bins=30)
    ax.plot(edges,hist,label=lbl,lw=6,alpha=0.5)
# power law x^-3
x = np.logspace(-1, 1.2)
ax.plot(x,0.4*x**(-3),lw=6,ls='--',color='k',alpha=0.8,label=r'$\propto
ax.plot(x,np.abs(np.exp(-x**2/2)/np.sqrt(2*np.pi)),lw=6,ls='--',color='g
ax.semilogy()
ax.semilogx()
ax.set_ylim(1e-4, 2e0)
ax.set_xlim(1e-1, 2e1)
ax.legend(loc=3, fontsize=16, framealpha=0.8)
plt.show()
```