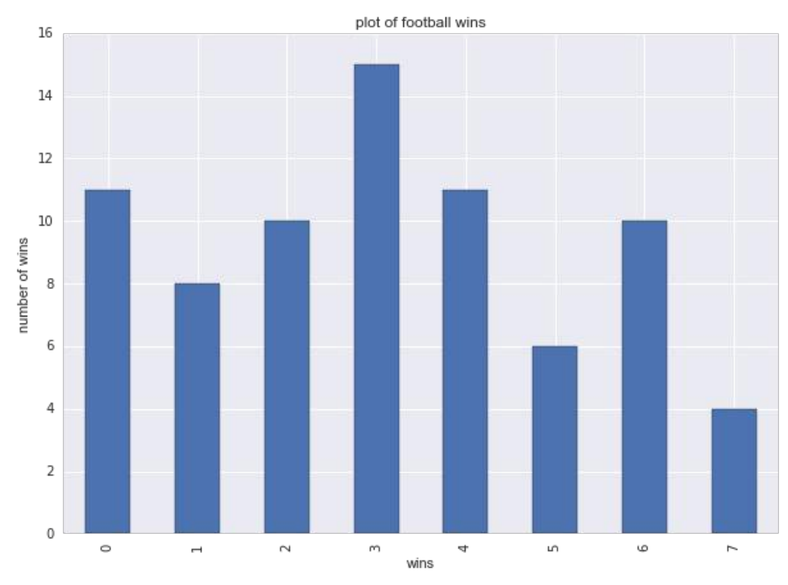
In [25]:

```
# Common imports
%matplotlib inline
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns  #need to pip install
import sys
getsize=sys.getsizeof
#pd.set_option('max_columns', 50)
```

In [26]:

```
# Pandas Series and Dataframes

import pandas as pd
# pandas Series example
s = pd.Series([7, 'Heisenberg', 3.14, -1789710578, 'Happy Eating!'],
              index=['A', 'Z', 'C', 'Y', 'E'])
print('\nSERIES WITH INDEX\n' + str(s))
d = {'Chicago': 1000, 'New York': 1300, 'Portland': 900, 'San Francisco': 1100,
     'Austin': 450, 'Boston': None}
cities = pd.Series(d)
print('\nSERIES FROM DICT\n' + str(cities))

# pandas Dataframe examples
data = {'year': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],
        'team': ['Bears', 'Bears', 'Bears', 'Packers', 'Packers', 'Lions', 'Lions', 'Lions'],
        'wins': [11, 8, 10, 15, 11, 6, 10, 4],
        'losses': [5, 8, 6, 1, 5, 10, 6, 12]}
football = pd.DataFrame(data, columns=['year', 'team', 'wins', 'losses'])
football.head()
football.to_excel('football.xlsx', index=False)
football1 = pd.read_excel('football.xlsx', 'Sheet1')
football1

plt.rcParams["figure.figsize"] = (10,7)
football.wins.plot.bar()  # plot options: area, bar, barh, box, density, hist, kde, line, pie
plt.title("plot of football wins")
plt.ylabel('number of wins')
plt.xlabel('wins')

# graphical filtering of dataframe
import qgrid
qgrid.nbinstall(overwrite=True)
qgrid.set_defaults(remote_js=True)
#need to click export for some reason to get to appear
qgrid.show_grid(football, show_toolbar=True)
```

```
SERIES WITH INDEX
A                7
Z       Heisenberg
C             3.14
Y      -1789710578
E    Happy Eating!
dtype: object

SERIES FROM DICT
Austin            450.0
Boston              NaN
Chicago          1000.0
New York         1300.0
Portland          900.0
San Francisco    1100.0
dtype: float64
```



In [27]:

```
# pandas loading a dataframe

# READING FROM A TEXT FILE
#from_csv = pd.read_csv('pandas_data.csv')
#from_csv.head()

# READING FROM A SQL DATABASE
from sqlalchemy import create_engine
from pandas.io import sql

db = create_engine('sqlite:///sample1.db')
conn = db.connect()
#creation = conn.execute("Create table btable (a varchar(10))")
insertion = conn.execute("Insert into btable values ('hello')" )
query = 'SELECT * from btable'
selection = conn.execute(query)
#for row in selection:
#    print("AAAA:", row['a'])
df1 = sql.read_sql(query, con=conn)
print(df1.head())

# READ FROM THE CLIPBOARD
#df = pd.read_clipboard()
#df.head()

#READ FROM URL
url = 'https://raw.github.com/gjreda/best-sandwiches/master/data/best-sandwiches-geocode.tsv'
from_url = pd.read_table(url, sep='\t')
#from_url.head()

#IFrame('http://www.eia.gov/coal/data.cfm',width=700,height=350)
IFrame('https://www.vocalink.com/media/1518/valacdos.txt',width=700,height=350)
```

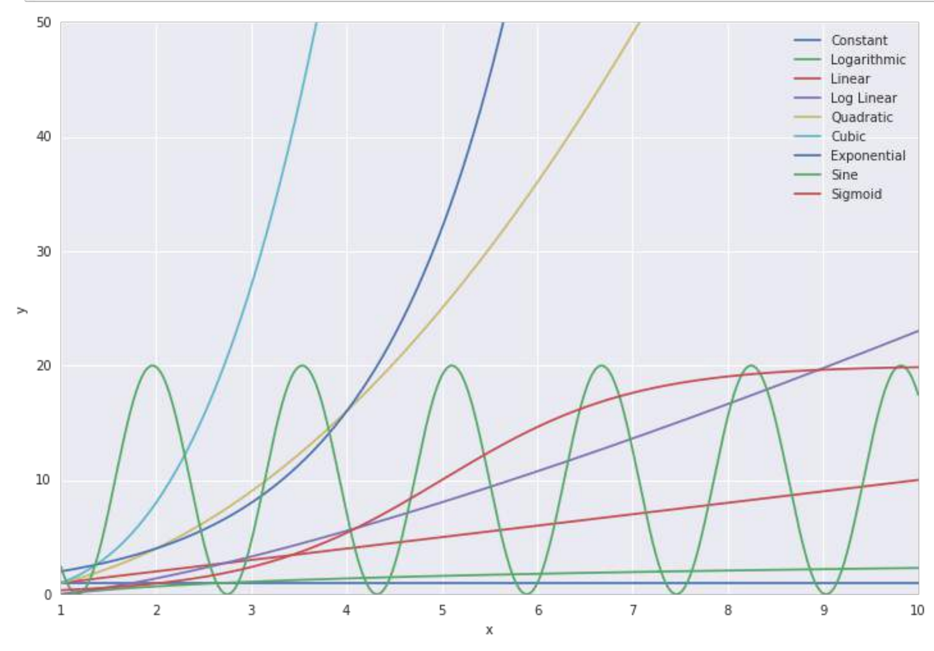
```
       a
0  hello
1  hello
2  hello
3  hello
4  hello

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-47f60ada42c9> in <module>()
     30
     31 #IFrame('http://www.eia.gov/coal/data.cfm',width=700,height=350)
---> 32 IFrame('https://www.vocalink.com/media/1518/valacdos.txt',width=700,height=350)

NameError: name 'IFrame' is not defined
```

In [ ]:

```
# pandas data manipulation

heading = ['classify', 'message']
sms = pd.read_table('sms.txt', header=None, names = heading)
print('info:\n',sms.info(),'\n')
print('dtypes:\n',sms.dtypes,'\n')
#print(sms.head(),'\n')
print('slice:\n',sms[:2],'\n')
print('filter:\n',sms[(sms.classify == 'spam') & (sms.message != 'qwe')].head(),'\n')
print('loc:\n',sms.loc[100],'\n')
#print('describe:\n',sms.describe,'\n')

sms1 = sms.copy()
newsms = pd.merge(sms, sms1, on='classify', how='inner')  # equivalent of SQL inner join,
# outer joins are how='left', how='right', how='outer
newsms1 = pd.concat([sms, sms1])
newsms2 = pd.concat([sms, sms1], axis=1)

#grouping data like with SQL groupby
newsms3 = sms.groupby('classify')
pd.unique(sms.classify)

print('count:\n',sms.count(),'\n')
print('size:\n',sms.size,'\n')
#newsms.sum()
#newsms.mean()
#newsms.median()
print('groupby:\n',sms.groupby('classify').size().sort_values(ascending=False)[:25],'\n')
print('value_counts\n',sms.classify.value_counts()[:25],'\n')
```

In [28]:

```
# matplotlib graphing and Iframe via http

from IPython.display import IFrame
from math import log, sin, pi
from scipy.special import expit
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-notebook')  #plt.style.use('bmh')

# Set up runtime comparisons
n = np.linspace(1,10,1000)
labels = ['Constant','Logarithmic','Linear','Log Linear','Quadratic','Cubic','Exponential','Sine', 'Sigmoid']
big_o = [np.ones(n.shape), np.log(n), n, n*np.log(n), n**2, n**3, 2**n,  np.sin(n*4)*10+10, expit(n-5)*20]
# Plot setup
plt.figure(figsize=(12,8))
plt.ylim(0,50)
for i in range(len(big_o)):
    plt.plot(n,big_o[i],label = labels[i])
plt.legend(loc='best')
plt.ylabel('y')
plt.xlabel('x')
plt.grid(True)
plt.show()
# plt.savefig('C:\\Users\\Graham\Desktop\\qwe.jpg')
```

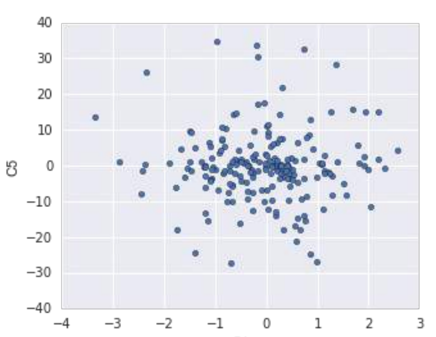


In [ ]:

```
# scatter plotting to find correlations
from pandas.tools.plotting import scatter_matrix
import seaborn as sns
import pandas as pd

df1=pd.DataFrame(np.random.randn(200,4), columns=['C1', 'C2', 'C3','C4'])
df1['C5'] = df1['C4'] * df1['C3'] * 10
df1['C6'] = df1['C3'].div(3).round(decimals=1)
df1.plot.scatter('C1','C5',figsize=(5,4))
sns.pairplot(df1[['C1','C2','C3','C6']], hue='C6') #investigate correlation between pairs
scatter_matrix(df1, alpha=0.2, figsize=(8, 8), diagonal='kde')
df1.describe()
```

Out[ ]:

	C1	C2	C3	C4	C5	C6
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	-0.057384	0.011201	0.064698	-0.103176	0.012646	0.024000
std	1.013912	0.990390	0.984075	1.037638	9.799857	0.332531
min	-3.367947	-2.820465	-2.600070	-2.846129	-27.002184	-0.900000
25%	-0.719461	-0.663084	-0.570535	-0.903423	-4.106004	-0.200000
50%	0.008291	-0.043751	-0.037310	-0.244528	-0.185433	0.000000
75%	0.519131	0.648046	0.697244	0.638693	3.324048	0.200000
max	2.561217	2.353550	2.987037	3.342051	35.015469	1.000000



In [ ]:

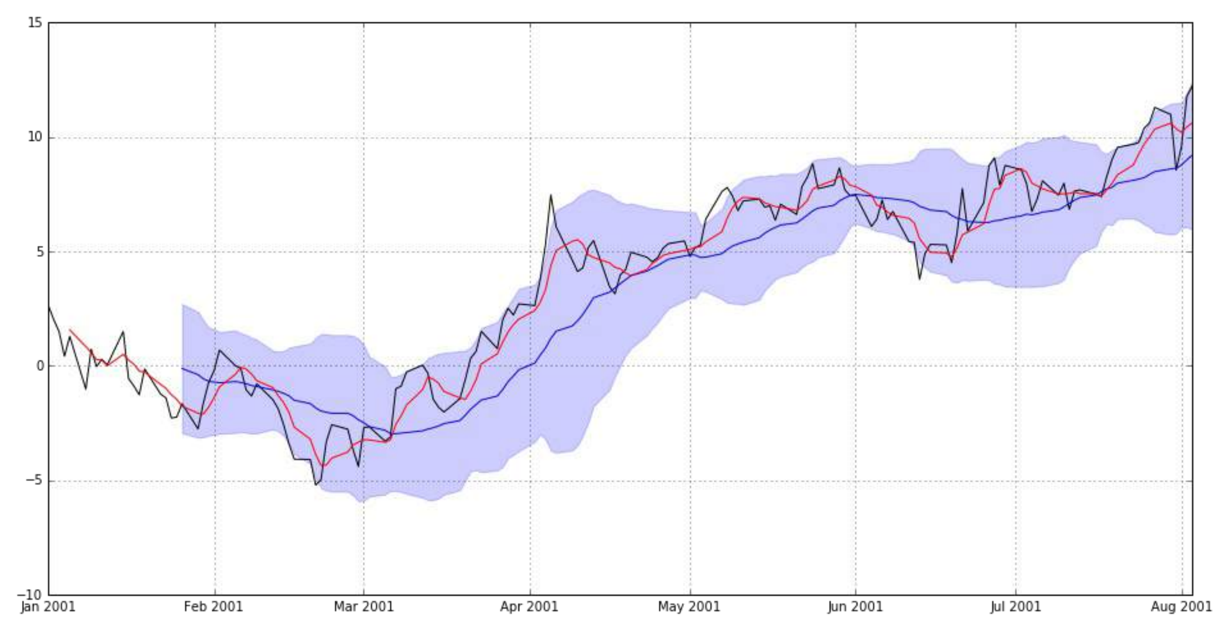
```
# from OS command line do=> jupyter nbconvert "data science tinkering.ipynb" --to slides --post serve

# slider input of parameters
import matplotlib.pyplot as plt
from ipywidgets import interact
%matplotlib inline
#plt.rcParams["figure.figsize"] = (6,6)

@interact
def echo(split=[0,100]):
    plt.pie([split/100, 1-split/100], labels=['this split','that split'])
    plt.show()
```

In [23]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1)
price = pd.Series(np.random.randn(155).cumsum()+1, index=pd.date_range('2001-01-01', periods=155, freq='B'))
mavg20 = price.rolling(20).mean()
mstd20 = price.rolling(20).std()
mavg5 = price.rolling(5).mean()

plt.figure(figsize=(16,8)); plt.grid(True)
plt.plot(price.index, price, c='k')
plt.plot(mavg20.index, mavg20, c='b')
plt.plot(mavg5.index, mavg5, c='r')
plt.fill_between(mstd20.index, mavg20-2*mstd20, mavg20+2*mstd20, color='b', alpha=0.2)
```

Out[23]:

```
<matplotlib.collections.PolyCollection at 0x208b4a972e8>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: