GH(??)_0820

September 12, 2018

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

import seaborn
from sklearn import *
import glob
import os

#from jupyterthemes import jtplot
#jtplot.style()

import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:44: DeprecationWarning: This
  "This module will be removed in 0.20.", DeprecationWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/grid_search.py:43: DeprecationWarning: This modul
  DeprecationWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/lda.py:6: DeprecationWarning: lda.LDA has been mo
  "in 0.17 and will be removed in 0.19", DeprecationWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/learning_curve.py:23: DeprecationWarning: This mo
  DeprecationWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/qda.py:6: DeprecationWarning: qda.QDA has been mo
  "in 0.17 and will be removed in 0.19.", DeprecationWarning)
```

In [2]:
```
from matplotlib import font_manager, rc

font_fname = '/Users/minjunggim/Library/Fonts/-330.ttf'
#font_name = font_manager.FontProperties(fname=font_fname).get_name()


#rc('font', family=font_name)
```

In [3]:
```
path = "Data"
filenames = glob.glob(os.path.join(path,"*.xlsx"))
filenames.sort()
filenames
```

Out[3]:
```
['Data/()__1.xlsx',
 'Data/()__2.xlsx',
 'Data/()__3.xlsx',
 'Data/()__4.xlsx',
 'Data/()__5.xlsx',
 'Data/()__6.xlsx',
 'Data/()__7.xlsx']
```

In [4]:
```
data_frame = pd.read_excel(filenames[0]).fillna(0)

for i in filenames[1:7]:
    data_frame = pd.concat([data_frame, pd.read_excel(i)]).fillna(0)

del data_frame[""]

data_frame.describe()
```

Out[4]:

	()		()	()	
count	5.533200e+04	55332.000000	55332.000000	55332.000000	55332.000000
mean	4.081383e+04	1985.750777	201803.812477	266.671914	160.631934
std	5.240766e+04	62.088644	1.842472	609.413174	150.398171
min	2.000000e+02	0.000000	201801.000000	4.000000	5.900000
25%	1.200000e+04	1978.000000	201802.000000	123.000000	69.315000
50%	2.530000e+04	1989.000000	201804.000000	185.000000	112.100000
75%	5.200000e+04	2001.000000	201805.000000	317.000000	192.565000
max	1.500000e+06	2018.000000	201807.000000	119119.000000	5225.220000

In [5]:
```
cut_value = 2.5

data_frame = data_frame[data_frame[""] > np.percentile(data_frame[""], 5)]
data_frame = data_frame[(data_frame["()"] < np.percentile(data_frame["()"], 100-cut_valu
    (data_frame["()"] > np.percentile(data_frame["()"], cut_value))]
data_frame = data_frame[(data_frame["()"] < np.percentile(data_frame["()"], 100-cut_valu
    (data_frame["()"] > np.percentile(data_frame["()"], cut_value))]
data_frame = data_frame[(data_frame[""] < np.percentile(data_frame[""], 100-cut_value))
    (data_frame[""] > np.percentile(data_frame[""], cut_value))]
```

In [6]:
```
print(data_frame.columns.values)
data_frame.describe()
```

```
['()' '' '' '()' '' '' '' '' '()' '']
```

Out[6]:

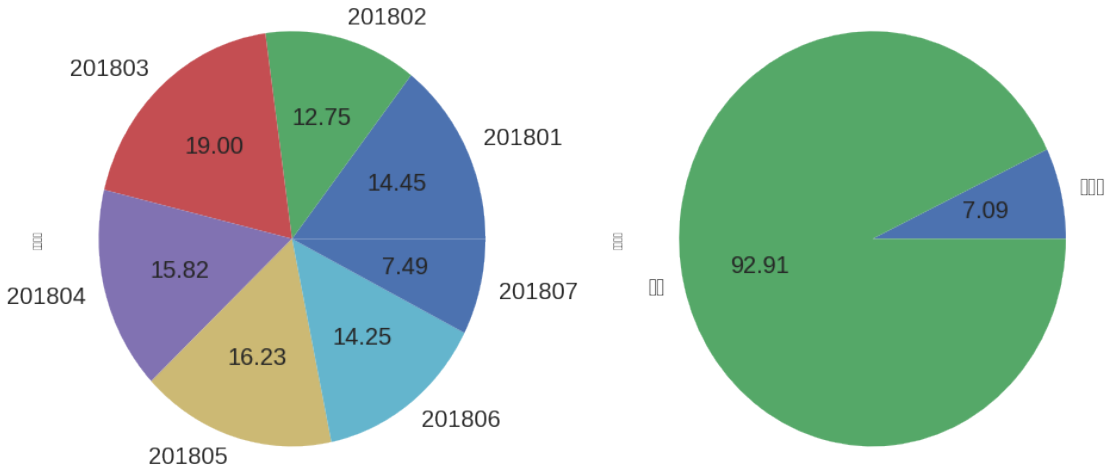
	()		()	()	
count	4.362900e+04	43629.000000	43629.000000	43629.000000	43629.000000

```
mean    4.013783e+04   1990.090949  201803.793348    228.994908    156.917971
std     4.343620e+04     13.137546       1.834272    158.915420    112.249496
min     2.000000e+02   1962.000000  201801.000000     46.040000     32.300000
25%     1.370000e+04   1980.000000  201802.000000    125.000000     78.250000
50%     2.700000e+04   1989.000000  201804.000000    175.200000    121.490000
75%     5.247400e+04   1999.000000  201805.000000    267.100000    194.520000
max     1.408400e+06   2016.000000  201807.000000    853.000000    565.680000
```

In [7]:
```
def split_cat(text):
    try: return text.split(" ")
    except: return ("", "")

medianprops = dict(linestyle='-', linewidth=3, color="w")
boxprops = dict(linestyle='-', linewidth=5, color="k")


data_frame["Region1"], data_frame["Region2"], data_frame["Region3"] = zip(*data_frame[""
    lambda x: split_cat(x)))

data_frame["/"] = data_frame["()"]/data_frame["()"]
```

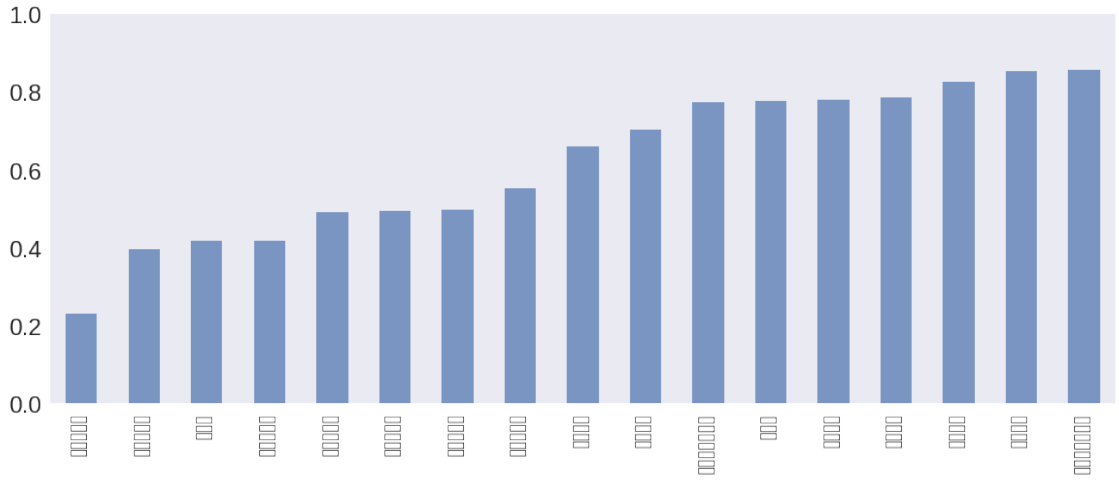
In [8]:
```
purpose = data_frame[""].value_counts().sort_index()
transaction = data_frame[""].value_counts().sort_index()

fig, axes = plt.subplots(nrows=1, ncols=2)
transaction.plot.pie(autopct='%.2f', fontsize=20, figsize=(16, 8),ax=axes[0])
purpose.plot.pie(autopct='%.2f', fontsize=20, figsize=(16, 8),ax=axes[1])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6a8352160>
```

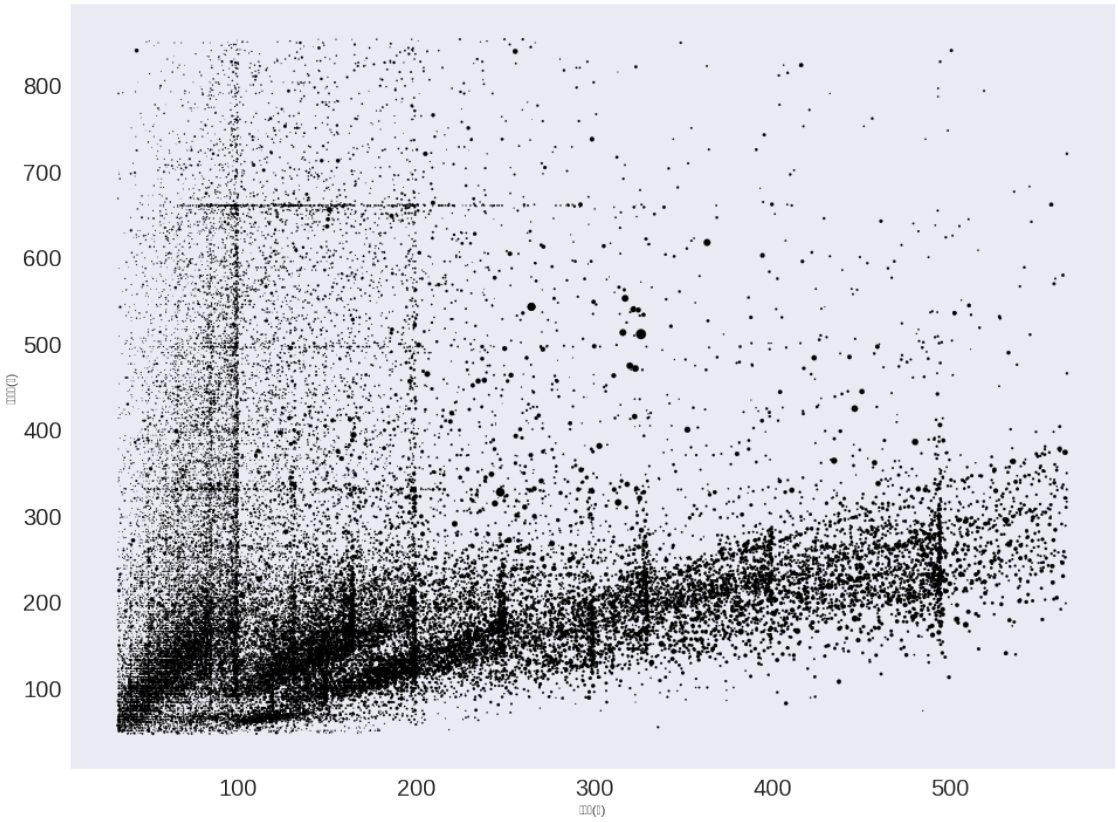


In [9]:
```
#
# data_frame.to_excel("house_data.xlsx")
```

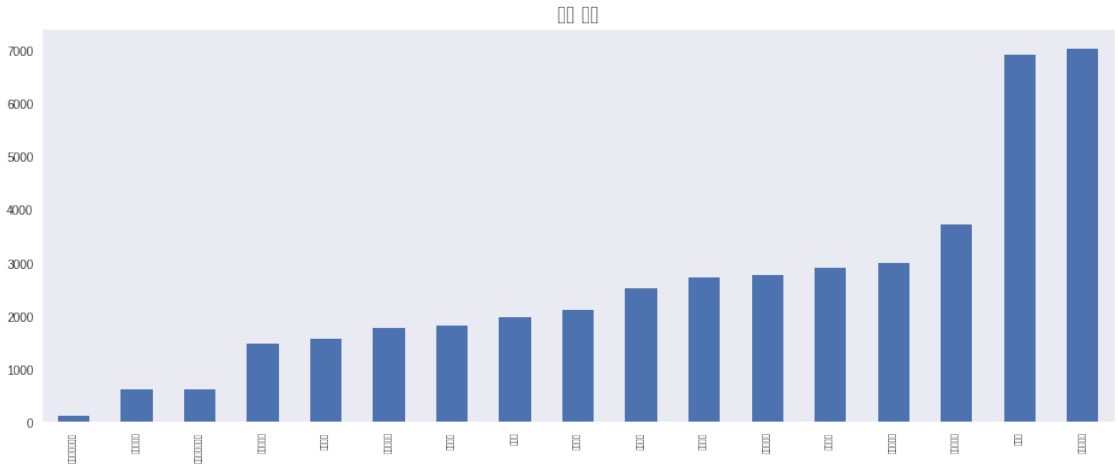
In [10]:
```
z = data_frame[(data_frame["()"] < 0.8 * data_frame["()"])]
zz = z["Region1"].value_counts().sort_index() / data_frame["Region1"].value_counts().so
zz.sort_values().plot.bar(figsize=(16,6),alpha=0.7,fontsize=20)
plt.ylim(0,1)
plt.grid(False)
```



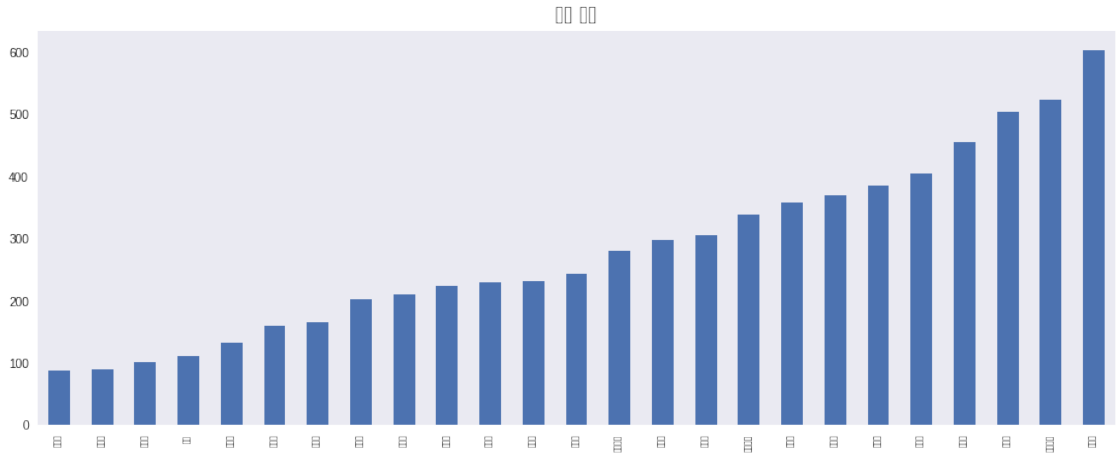
In [11]:
```
data_frame.plot.scatter(x="()", y="()", s= data_frame["()"]/20000, color="k", figsize=(
plt.grid(False)
```



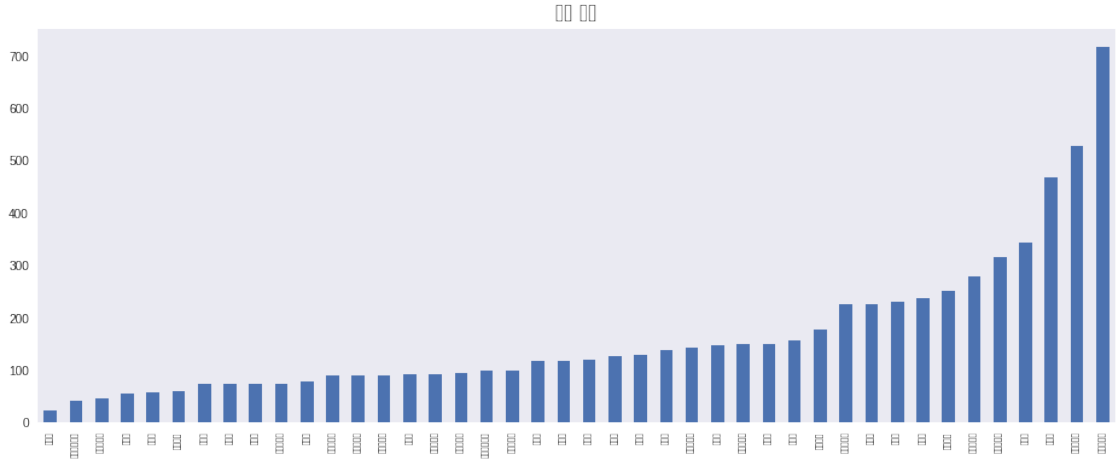

```
In [12]: data_frame["Region1"].value_counts().sort_values().plot.bar(figsize=(16,6))
         plt.title(" ", fontsize=20)
         plt.grid(False)
```



In [13]:
```
data_frame["Region2"][data_frame["Region1"]==""].value_counts().sort_values().plot.bar(
plt.title(" ", fontsize=20)
plt.grid(False)
```



In [14]:
```
data_frame["Region2"][data_frame["Region1"]==""].value_counts().sort_values().plot.bar(
plt.title(" ", fontsize=20)
plt.grid(False)
```



In [15]:
```
#

# data_frame = data_frame[(data_frame["Region1"]=="") | (data_frame["Region1"]=="")]
```

In [16]:
```
z = data_frame["Region3"].value_counts().sort_index()
z = z[z >= 50]
list_region3 = z.index.values
```

In [17]:
```
# data_frame["mean_ratio"] = data_frame.groupby('Region3')['/'].transform('mean')
# data_frame.groupby(["Region3"]).agg("mean")["/"]
```

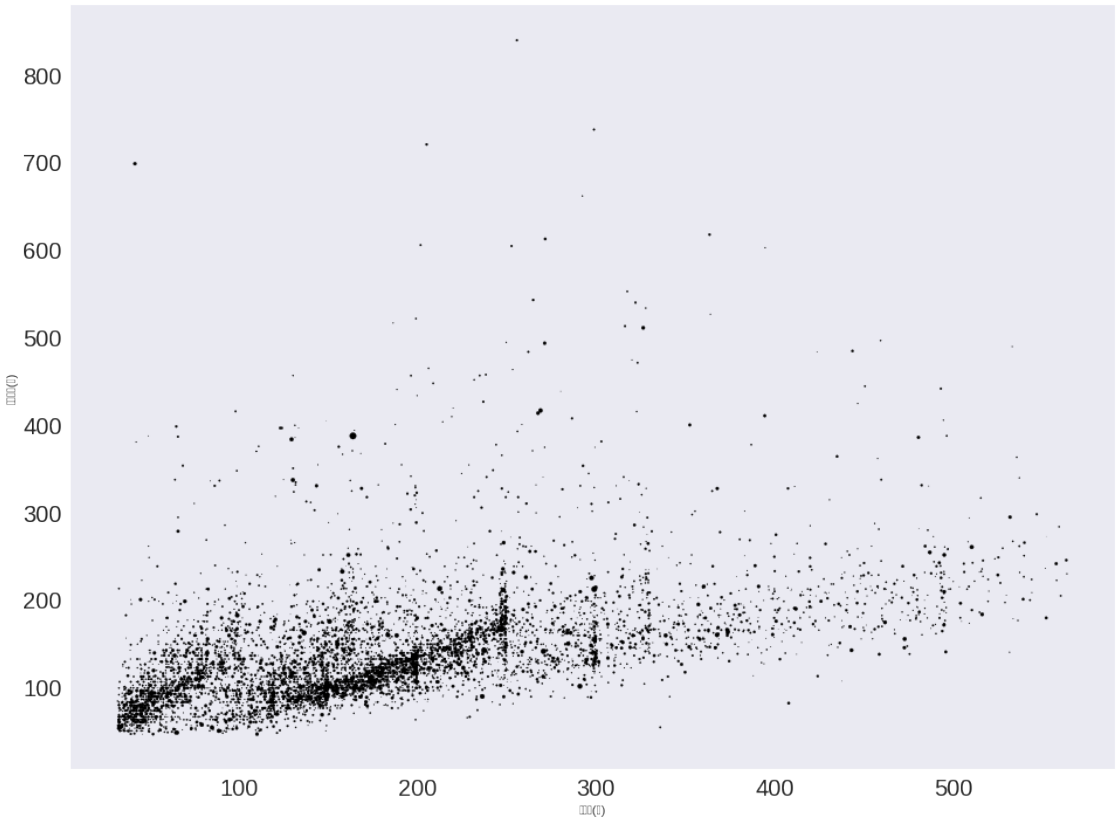
In [18]:
```
data_frame.head()
```

Out[18]:

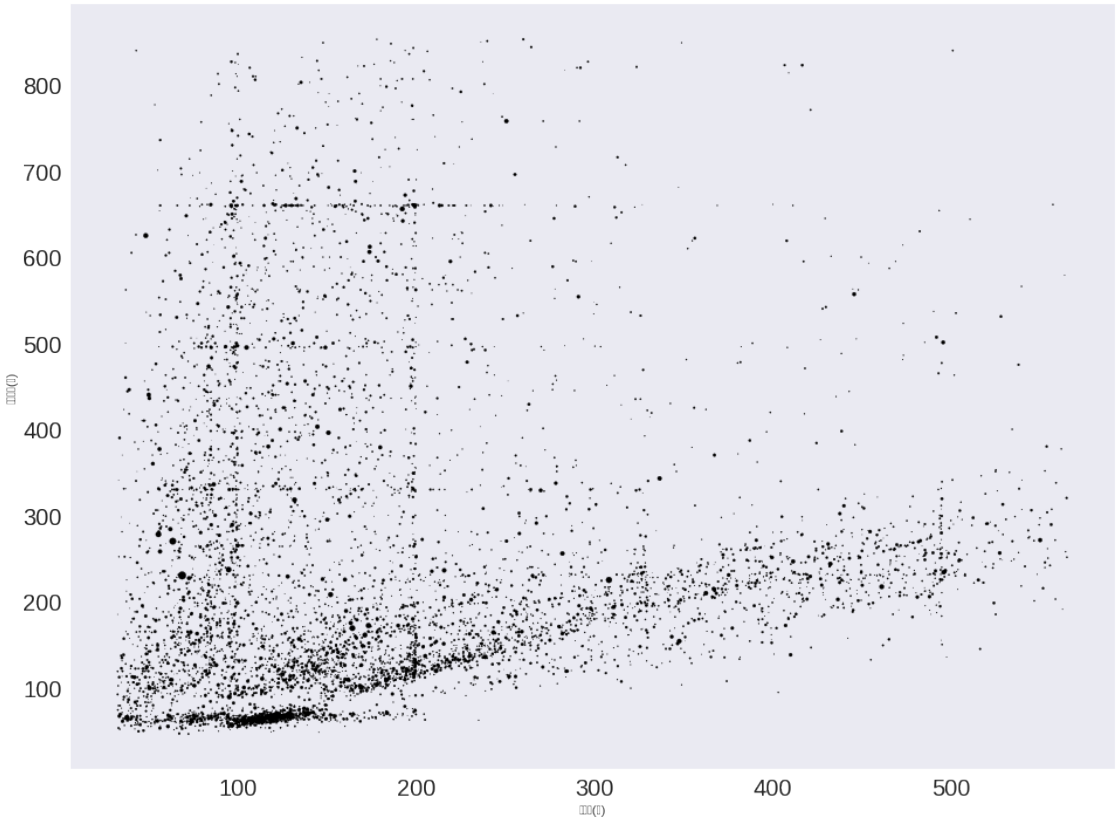
	()	()			\			
1	4000	1990.0	201801	81.0		12m	2**	
2	16500	1982.0	201801	160.0	8	12m	9**	
3	64300	2005.0	201801	211.6	47	12m	1***	
4	19000	1980.0	201801	240.2	20	8m	9**	
6	6450	1990.0	201801	407.0		8m	3**	

	()	Region1	Region2	Region3	/
1	35.60				0.439506
2	99.90				0.624375
3	326.88				1.544802
4	154.99				0.645254
6	69.03				0.169607

In [19]:
```
data_frame[data_frame["Region1"]==""].plot.scatter(x="()", y="()",
                                                   s= data_frame["()"]/20000, color="k
plt.grid(False)
```



In [20]:
```
data_frame[data_frame["Region1"]==""].plot.scatter(x="()", y="()",
                                                   s= data_frame["()"]/20000, color="k
plt.grid(False)
```



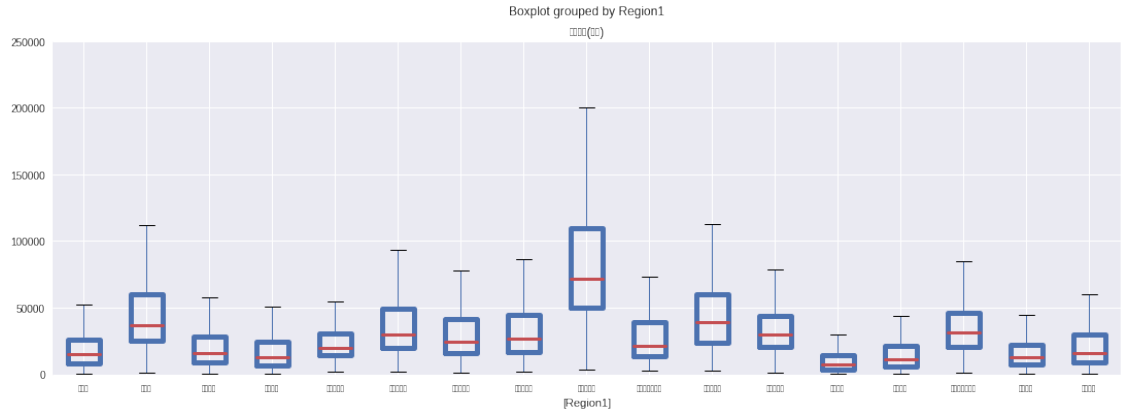
In [21]:
```
data_frame.head()
```

Out[21]:

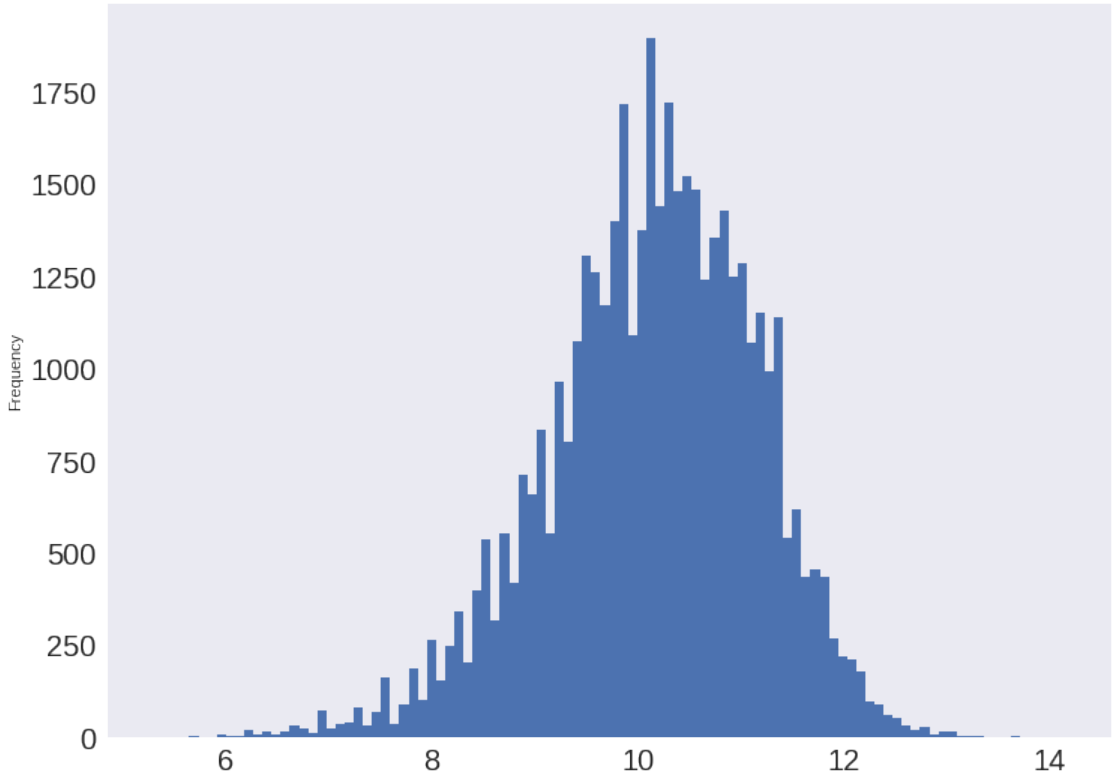
	()	()				\
1	4000	1990.0	201801	81.0	12m	2**
2	16500	1982.0	201801	160.0	8 12m	9**
3	64300	2005.0	201801	211.6	47 12m	1***
4	19000	1980.0	201801	240.2	20 8m	9**
6	6450	1990.0	201801	407.0	8m	3**

	()	Region1	Region2	Region3	/
1	35.60			0.439506	
2	99.90			0.624375	
3	326.88			1.544802	
4	154.99			0.645254	
6	69.03			0.169607	

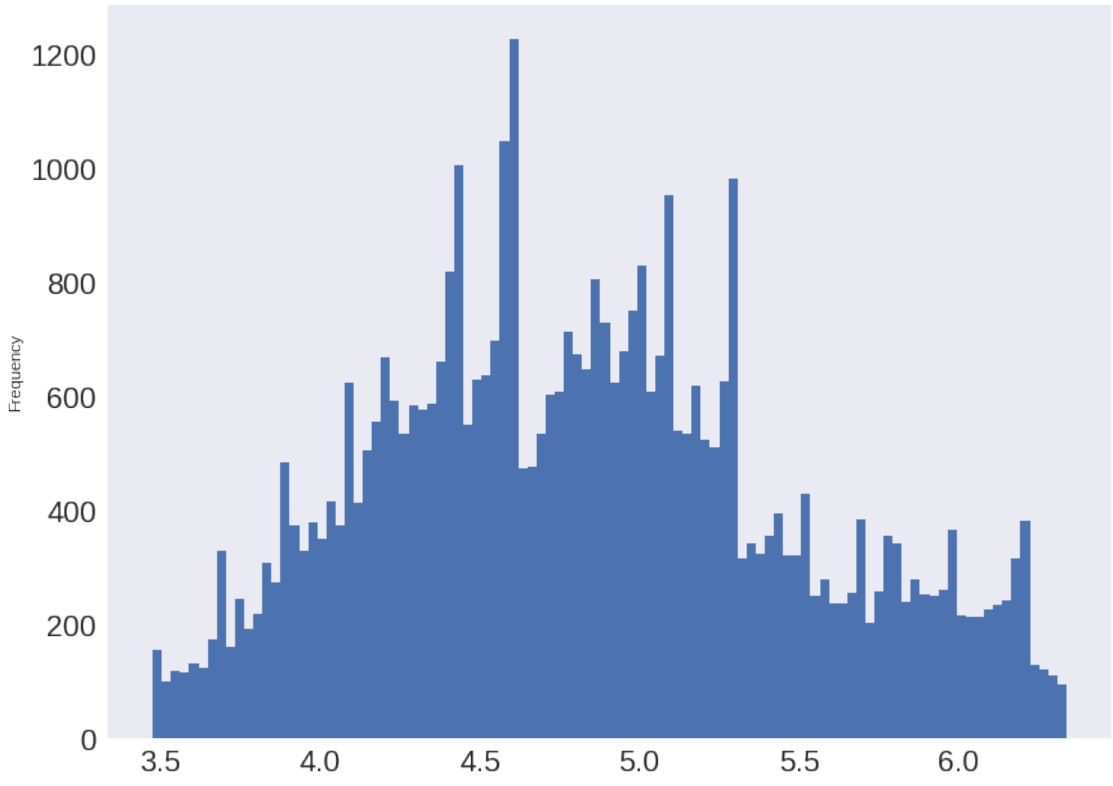
In [22]:
```
data_frame[["Region1", "()"]].boxplot(by="Region1", figsize=(18,6),boxprops=boxprops,me
_ = plt.ylim(-1000, 250000)
```



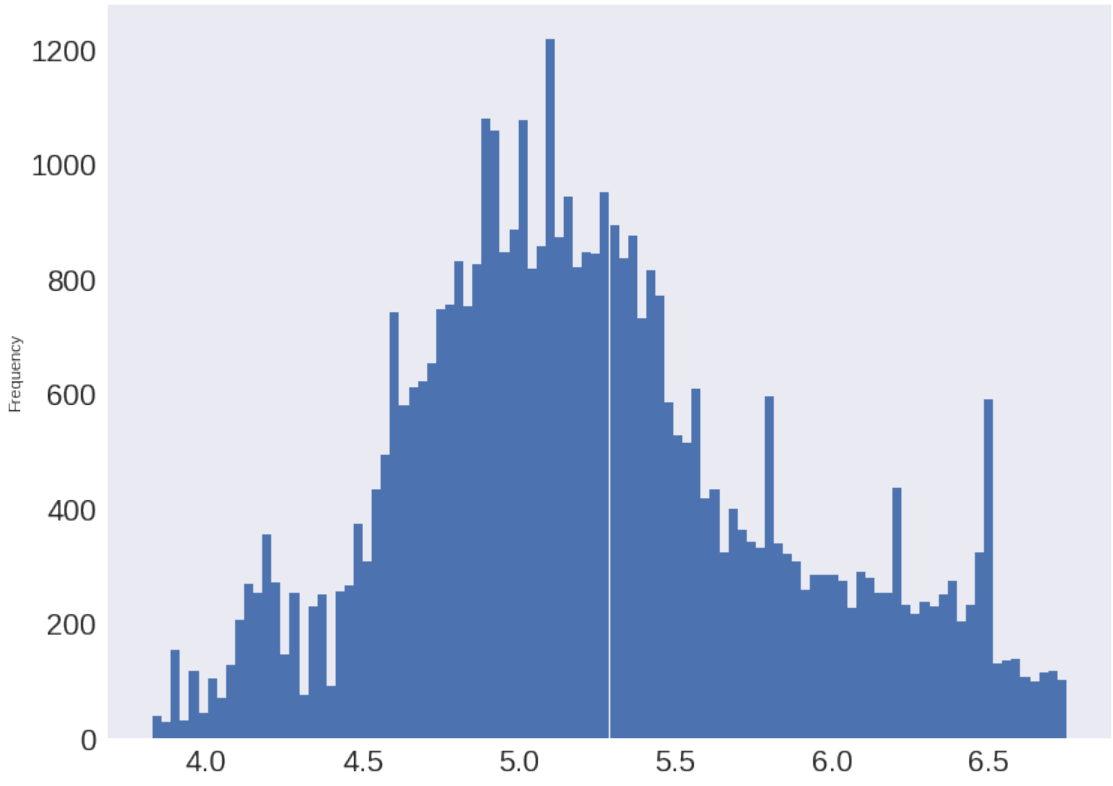
In [23]:
```
np.log(data_frame["()"]).plot.hist(figsize=(12,9),fontsize=20, bins=100)
plt.grid(False)
```



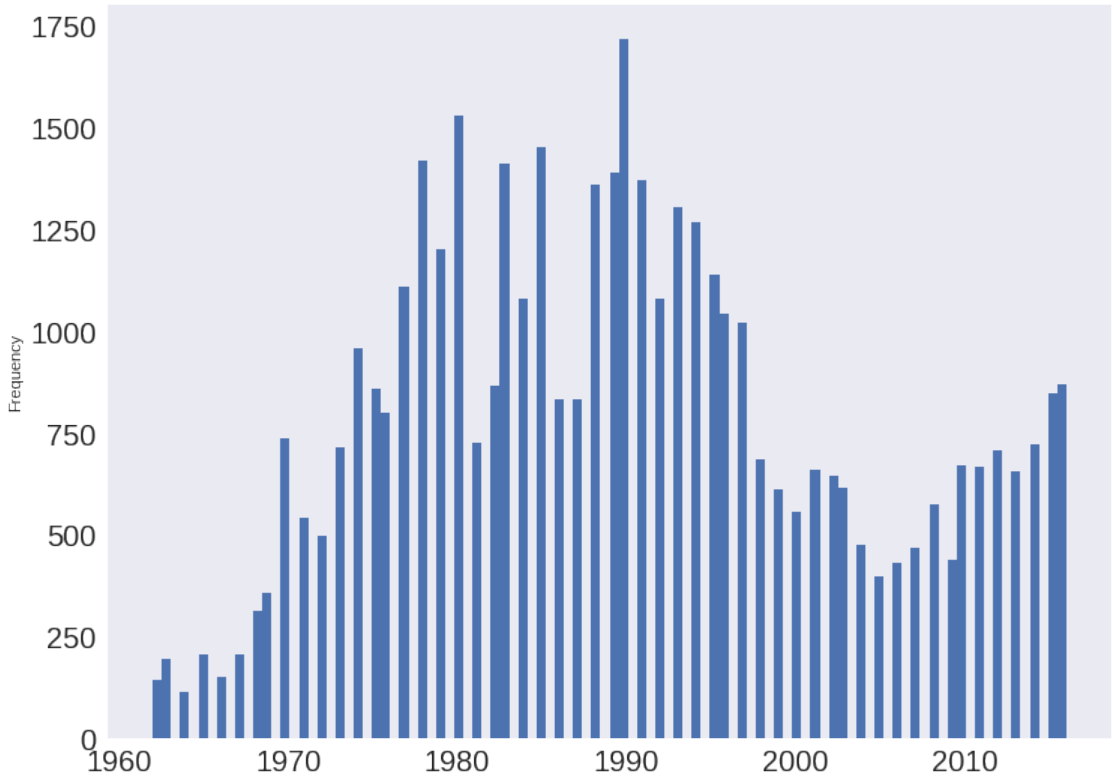
In [24]:
```
np.log(data_frame["()"]).plot.hist(figsize=(12,9),fontsize=20, bins=100)
plt.grid(False)
```



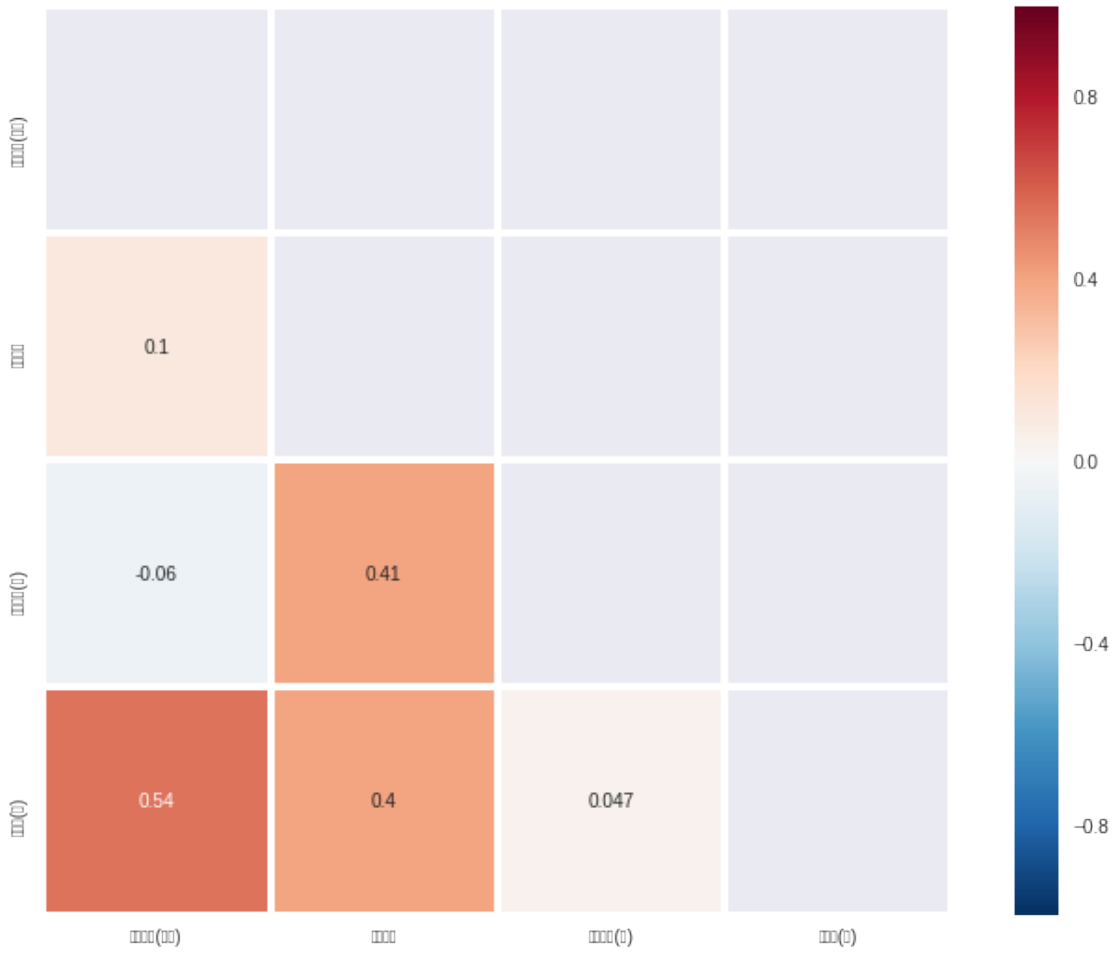

```
In [25]: np.log(data_frame["()"]).plot.hist(figsize=(12,9),fontsize=20, bins=100)
         plt.grid(False)
```




```
In [26]: data_frame[""].plot.hist(figsize=(12,9),fontsize=20, bins=100)
         plt.grid(False)
```

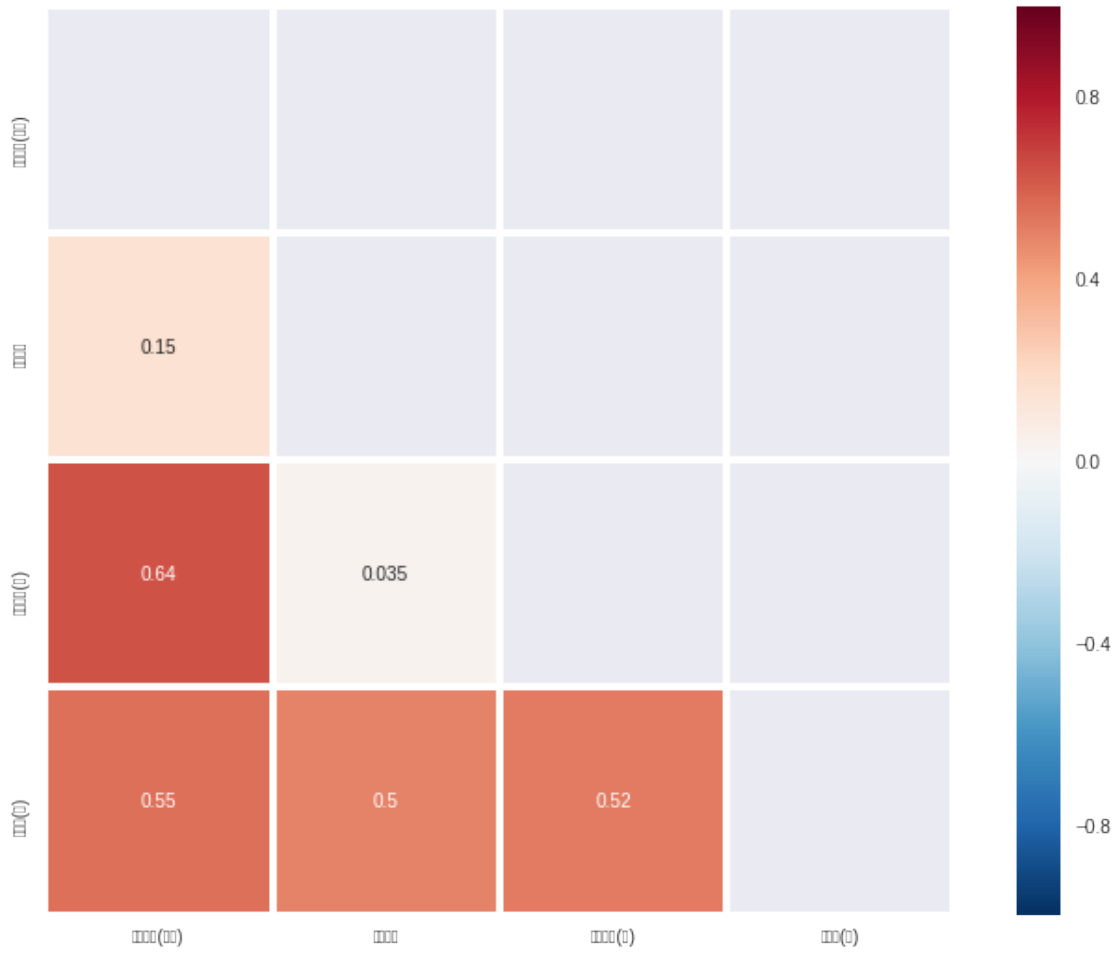



```
In [27]: num_cols = list(data_frame.select_dtypes(exclude=['object']).columns)
         num_cols.remove("")
         num_cols.remove("/")
         num_cols

Out[27]: ['()', '', '()', '()']

In [28]: corr = data_frame[num_cols].corr()
         mask = np.zeros_like(corr, dtype=np.bool)
         mask[np.triu_indices_from(mask)] = True
         plt.figure(figsize=(12,9))
         seaborn.heatmap(corr, mask=mask, center=0, annot=True,
                         square=True, linewidths=3)

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb6a7c55c50>
```



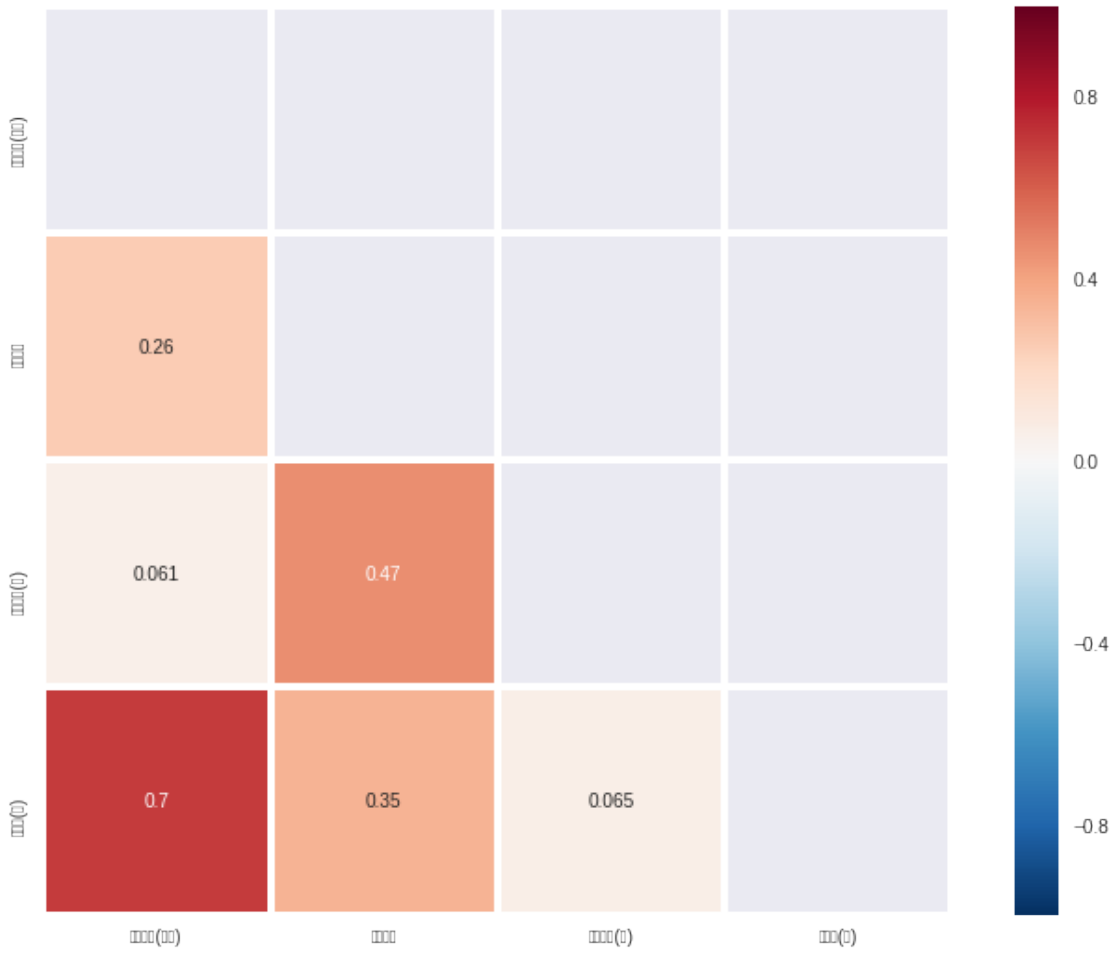

```
In [29]: corr = data_frame[data_frame["Region1"]==""][num_cols].corr()
         mask = np.zeros_like(corr, dtype=np.bool)
         mask[np.triu_indices_from(mask)] = True
         plt.figure(figsize=(12,9))
         seaborn.heatmap(corr, mask=mask, center=0, annot=True,
                         square=True, linewidths=3)
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb6a89e7cc0>
```




```
In [30]: corr = data_frame[data_frame["Region1"]==""][num_cols].corr()
         mask = np.zeros_like(corr, dtype=np.bool)
         mask[np.triu_indices_from(mask)] = True
         plt.figure(figsize=(12,9))
         seaborn.heatmap(corr, mask=mask, center=0, annot=True,
                         square=True, linewidths=3)
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb6a7bb2c50>
```

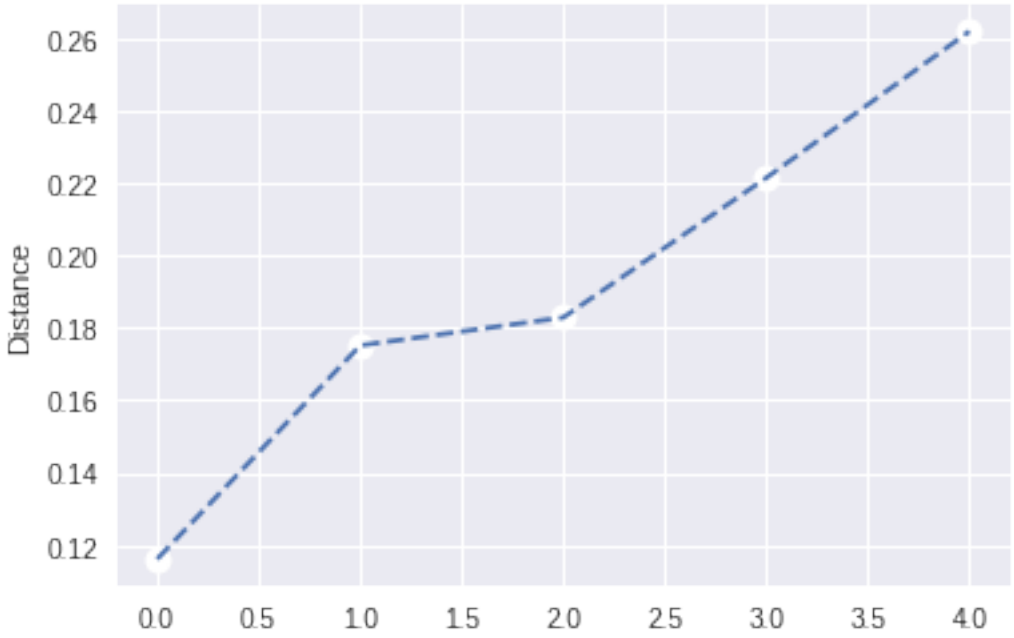


In [31]: print(list_region3)

```
['' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '1' '2' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '1' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '' '' '' ''
 '' '' '' '' '' '' '' '' '']
```

```
In [53]: list_col = ['Distance','()', '',  '()','()','', '', '', '', '']

         def find_house(inp_val, weight = [1,1,1,1],n_output = 5):
             df = data_frame
             if inp_region in list_region3:
                 df = df[df["Region3"]==inp_region]
             else:
                 region2 = (df[df["Region3"] == inp_region]["Region2"]).values[0]
                 df = df[df["Region2"]==region2]
             d = (df[num_cols] - inp_val)/ df[num_cols].std().values
             d = np.abs(d)
             #d *= np.array([0.552, 0.289, 0.549, 0.557])
             d *= np.array(weight / (np.sum(weight)+0.01 ))

             df["Distance"] = d.sum(axis=1)

             print(inp_region, "  sample :", len(df))
             print("  [(): %d, : %d, (): %.1f, (): %.1f] "
                   %(inp_val[0],inp_val[1],inp_val[2],inp_val[3]))
             output = df[list_col].sort_values("Distance").head(n_output)
             plt.plot(range(n_output), output["Distance"], "wo", markersize=10)
             plt.plot(range(n_output), output["Distance"], "--")
             plt.ylabel("Distance")
             return (output)

In [64]: print(num_cols)
         inp_val = np.array([80000, 2010, 200, 400])
         weight = [1,1,1,1]
         inp_region =""

['()', '', '()', '()']


In [65]: find_house(inp_val, weight, n_output=5)

   sample : 105
  [(): 80000, : 2010, (): 200.0, (): 400.0]


Out[65]:       Distance  ()      ()  ()         \
         8508  0.116172     73000  2010.0    224.4  390.24  201805  12m
         6669  0.175160     80500  2014.0    203.1  349.74  201802  12m
         7586  0.182874     84500  2010.0    245.9  442.85  201801  12m
         9879  0.221449     76000  2010.0    238.7  323.74  201803  12m
         4106  0.261898     70000  2011.0    177.7  327.73  201807  12m


         8508       8**
         6669      1***
```

```
7586       1***
9879       1***
4106       1***
```




```
In [35]: data_frame[num_cols].std()

Out[35]: ()    43436.196379
                    13.137546
         ()       158.915420
         ()        112.249496
         dtype: float64
```