In [ ]:

```
ccss.index1.hist(by = ccss.time) # 注意此处使用的是hist，不是plot.hist
```

In [ ]:

```
ccss.boxplot(column = 'index1', by = 'time') # 注意参数的引用方法
```

In [ ]:

```
ccss.groupby('time').boxplot(column = 'index1') # 绘制箱图图组
```

3.3.2 用假设检验考察方差齐性

scipy.stats.bartlett(a, b)

Bartlett's方差齐性检验，对数据有正态性要求。

scipy.stats.levene(a, b, center = {'mean', 'median', 'trimmed'})

Levene检验，在数据非正态时精度比Bartlett检验好，可调中间值的度量，默认median。

scipy.stats.fligner(a, b, center='mean')

Fligner-Killeen检验，非参检验，不依赖于分布。

In [ ]:

```
ss.bartlett(ccss.index1[ccss.time == 200704],
            ccss.index1[ccss.time == 200712],
            ccss.index1[ccss.time == 200812],
            ccss.index1[ccss.time == 200912]
         )
```

In [ ]:

```
ss.levene(ccss.index1[ccss.time == 200704],
          ccss.index1[ccss.time == 200712],
          ccss.index1[ccss.time == 200812],
          ccss.index1[ccss.time == 200912]
         )
```

3.4 实战练习

提取北京2007年4月的总指数、现状指数和预期指数，用假设检验方法和图形化方法考察其正态性，思考这两种方式的优缺点和使用场景，并比较不同检验方法的特点。

分别用图形方法和假设检验方法考察北京的总指数、现状指数和预期指数在四个时间点的方差齐性。

4 单因素方差分析

4.1 基本原理与适用条件

例4.1

CCSS案例中提供了2007年4月，以及2007、2008、2009年12月四个时间点的消费者信心监测数据，现希望分析这四个时间点的消费者信心指数平均水平是否存在差异。考虑到信心指数在不同地域间可能存在差异，这里只使用北京消费者的数据进行分析。

4.2 scipy的实现方式

scipy由于使用的数据格式并非标准统计格式，代码实现上非常笨拙。

scipy中没有提供两两比较方法。

In [ ]:

```
ccss.query("s0 == '北京'").groupby('time').index1.describe()
```

In [ ]:

```
ccss.query("s0 == '北京'").groupby('time').index1.mean().plot()
```

In [ ]:

```
ccss[ccss.s0 == '北京'].boxplot('index1', 'time')
```

In [ ]:

```
a = ccss.query("s0 == '北京' & time == '200704'").index1
b = ccss.query("s0 == '北京' & time == '200712'").index1
c = ccss.query("s0 == '北京' & time == '200812'").index1
d = ccss.query("s0 == '北京' & time == '200912'").index1

ss.levene(a, b, c, d) # 默认为使用中位数
```

In [ ]:

```
ss.f_oneway(a, b, c, d)
```

In [ ]:

```
# 没有两两比较功能，半拉工程，到此为止
```

4.3 statsmodels的实现方式

statsmodels中实际上是直接进行了一般线性模型的拟合，因此需要先进行模型设定

In [ ]:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

ccss.time = ccss.time.astype('str') # 必须调整为category或者str，否则模型拟合不正确

model = ols('index1 ~ time', data = ccss.loc[ccss.s0 == '北京',:]).fit()
restable = sm.stats.anova_lm(model, typ = 3) # III型方差分解
restable
```

4.4 均数间的多重比较

4.4.1 多重比较的基本原理

4.4.2 statsmodels的实现方式

statsmodels.sandbox.stats.multicomp包中提供了比较完整的两两比较方法，但使用上比较复杂

直接进行P值校正
直接给出两两比较的结果

statsmodels.sandbox.stats.multicomp.multipletests(

```
pvals : 类数组格式的原始P值
alpha = 0.05 : 希望控制的总Alpha水准，FWER
method = 'hs' : 具体的校正方法，写全称或者可区别的前几个字母均可
   'bonferroni' : one-step correction
   'sidak' : one-step correction

   'holm-sidak' : step down method using Sidak adjustments
   'holm' : step-down method using Bonferroni adjustments

   'simes-hochberg' : step-up method  (independent)
   'hommel' : closed method based on Simes tests (non-negative)
   'fdr_bh' : Benjamini/Hochberg  (non-negative)
   'fdr_by' : Benjamini/Yekutieli (negative)
   'fdr_tsbh' : two stage fdr correction (non-negative)
   'fdr_tsbky' : two stage fdr correction (non-negative)
is_sorted = 'False' : 是否将结果按照P值升序排列
```

) # 返回值：检验结果、检验P值、alphacSidak、alphacBonf

In [ ]:

```
from statsmodels.sandbox.stats import multicomp as mc

mc.multipletests([0.1, 0.2, 0.3], method = 'b')
```

GroupsStats和MultiComparison命令的输出更接近oneway ANOVA的需求，但是目前尚未完善

In [ ]:

```
poshoc = mc.MultiComparison(ccss.index1, ccss.time)
res = poshoc.tukeyhsd()
res
```

In [ ]:

```
res.summary()
```

4.4.3 scikit_posthocs的实现

功能简单完善，短期内可以考虑作为posthoc输出的工具

pip install scikit_posthocs

scikit_posthocs.posthoc_conover(

```
    data
    val_col =
    group_col =
    p_adjust =
        'bonferroni' : one-step correction
        'sidak' : one-step correction
        'holm-sidak' : step-down method using Sidak adjustments
        'holm' : step-down method using Bonferroni adjustments
        'simes-hochberg' : step-up method  (independent)
        'hommel' : closed method based on Simes tests (non-negative)
        'fdr_bh' : Benjamini/Hochberg  (non-negative)
        'fdr_by' : Benjamini/Yekutieli (negative)
        'fdr_tsbh' : two stage fdr correction (non-negative)
        'fdr_tsbky' : two stage fdr correction (non-negative)
```

)

In [ ]:

```
import scikit_posthocs as sp

pc = sp.posthoc_conover(ccss, val_col='index1', group_col='time',
                        p_adjust = 'bonferroni')
pc
```

In [ ]:

```
# 使用热力图显示比较结果
heatmap_args = {'linewidths': 0.25, 'linecolor': '0.5', 'clip_on': False,
                'square': True, 'cbar_ax_bbox': [0.80, 0.35, 0.04, 0.3]}
sp.sign_plot(pc, **heatmap_args)
```

In [ ]:

```
# 自定义热力图参数
# Format: diagonal, non-significant, p<0.001, p<0.01, p<0.05
cmap = ['1', '#fb6a4a',  '#08306b',  '#4292c6', '#c6dbef']
heatmap_args = {'cmap': cmap, 'linewidths': 0.25, 'linecolor': '0.5',
                'clip_on': False, 'square': True,
                'cbar_ax_bbox': [0.80, 0.35, 0.04, 0.3]}
sp.sign_plot(pc, **heatmap_args)
```

4.5 实战练习

针对上海、广州的数据分别完成四个时间点的比较，并进行两两比较。

尝试编写一个自动两两比较的函数，调用后可以直接完成两两比较的任务，并对检验水平进行所需的校正。