Time Series Analysis(11)- Smoothing(下)

如无特殊说明，本系列文章中的数据将使用2012~2017年，分别代表国内股票、香港股票、国内债卷和国内货币的四个指数数据。

> 在本篇文章中的前两部分，我们已经介绍了多种平滑技术，重点讲解了在时序分析和预测中占有重要地位的指数平滑技术。在最后一篇中，我们需要进入实践部分。我们将比较多种平滑技术对四指数数据的平滑效果。

1. 导入python包

In [1]:

```
import warnings
warnings.simplefilter('ignore')
```

In [2]:

```
import pandas as pd
import numpy as np
%matplotlib inline

from fintechtools.backtest import *
from fintechtools.datasource import *
from fintechtools.SimuMultiTest import *
#from lib.portfolio import DailySimulator
#from lib.experiment import Experiment


import pandas as pd
#import pandas_datareader.data as web
import numpy as np

import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from sklearn.metrics import mean_squared_error,mean_absolute_error
from arch import arch_model


#sns.set_context("talk")
import matplotlib as mpl
from matplotlib.ticker import FuncFormatter
#mpl.style.use('classic')

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['font.serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
import seaborn as sns
sns.set_style("whitegrid",{"font.sans-serif":['simhei', 'Arial']})
sns.set_context("talk")

#zhfont1 = matplotlib.font_manager.FontProperties(fname='C:\Users\ktwc37\Documents\ZNTG\notebooks\S7


%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

2. 读入数据

In [3]:

```
start = '2012-01-01'
end = '2017-02-05'
```

In [4]:

```
indexs = pd.read_excel('./data/华夏指数.xlsx')
indexs_pv = indexs.pivot_table(index='日期', columns= '简称', values= '收盘价(元)')
indexs_pv.index = pd.to_datetime(indexs_pv.index, unit='d')
```

In [5]:

```
indexs_pv.columns = ['国内债券', '国内股票', '香港股票', '国内货币']
indexs_pv = indexs_pv[['国内债券', '国内股票', '国内货币', '香港股票']]
indexs_pv.fillna(axis=0,method='bfill',inplace=True)
indexs_sub = indexs_pv.loc[start:end,]
```

In [6]:

```
indexs_sub = indexs_pv.loc[start:]
indexs_sub_logret = indexs_sub.apply(log_return)
```

我们只对国内股票和香港股票的净值数据进行平滑

- SMA
  窗口为30天

In [7]:

```
data = indexs_sub[['国内股票','香港股票']].copy()
```

In [8]:

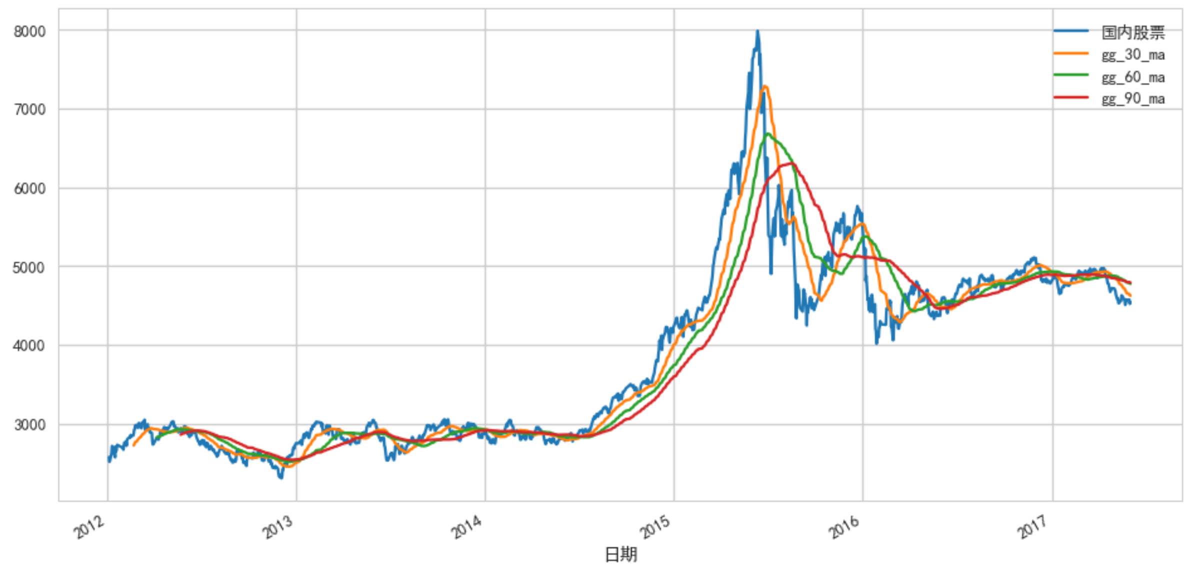
```
data['gg_30_ma'] = pd.rolling_mean(data['国内股票'],30)
data['xg_30_ma'] = pd.rolling_mean(data['香港股票'],30)
data['gg_60_ma'] = pd.rolling_mean(data['国内股票'],60)
data['xg_60_ma'] = pd.rolling_mean(data['香港股票'],60)
data['gg_90_ma'] = pd.rolling_mean(data['国内股票'],90)
data['xg_90_ma'] = pd.rolling_mean(data['香港股票'],90)
```

In [9]:

```
data[['国内股票','gg_30_ma','gg_60_ma','gg_90_ma']].plot(figsize=(16,8))
```

Out[9]:

<matplotlib.axes._subplots.AxesSubplot at 0x17a622454a8>

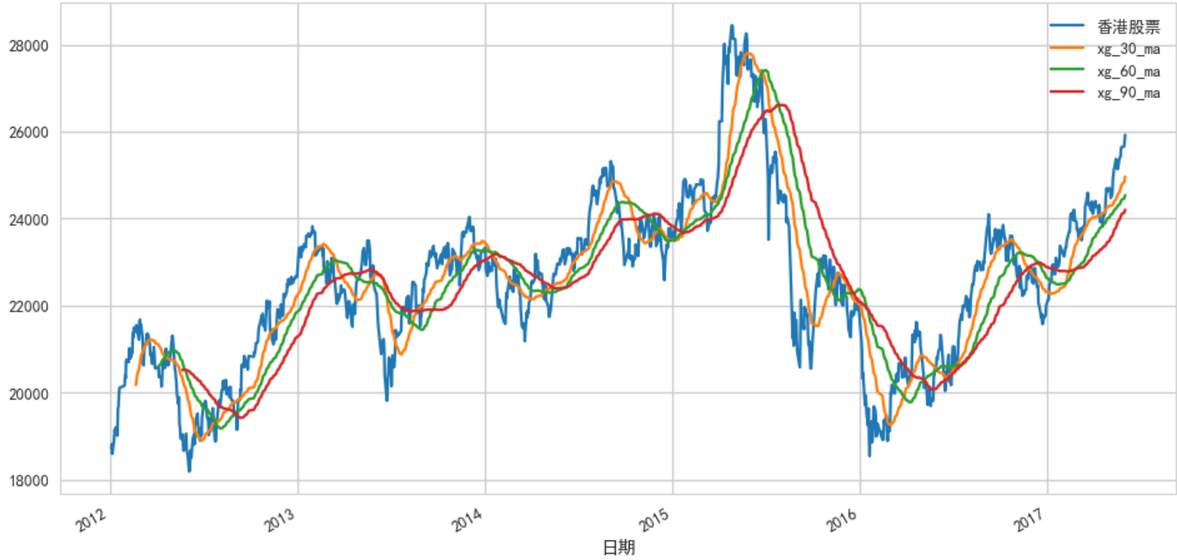


In [10]:

```
data[['香港股票','xg_30_ma','xg_60_ma','xg_90_ma']].plot(figsize=(16,8))
```

Out[10]:

<matplotlib.axes._subplots.AxesSubplot at 0x17a6200ff98>



看上去，效果还算不错。很明显的是，滑动窗口越小，平滑结果越接近原时序。

下面我们计算一下窗口为30天的SMA与原时序数据的RMSE，MAE。

In [11]:

```
mse_gg = mean_squared_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_30_ma'])
rmse_gg = np.sqrt(mse_gg)

mae_gg = mean_absolute_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_30_ma'])

mse_xg = mean_squared_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_30_ma'])
rmse_xg = np.sqrt(mse_xg)

mae_xg = mean_absolute_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_30_ma'])
```

In [12]:

```
res = pd.DataFrame([[mse_gg,rmse_gg,mae_gg],[mse_xg,rmse_xg,mae_xg]],
             index=['国内股票','香港股票'],
             columns=['SMA30_MSE','SMA30_RMSE','SMA30_MAE']).T
```

In [13]:

```
res
```

Out[13]:

	国内股票	香港股票
SMA30_MSE	88497.567727	576898.744557
SMA30_RMSE	297.485408	759.538508
SMA30_MAE	178.808682	593.674498

- EWMA
  $\alpha = 2/(1 + span)$
  在需要更加精确的情景下，一般α值都需要进行回归估算，我们这里采用通用的方法来简单估算α

In [14]:

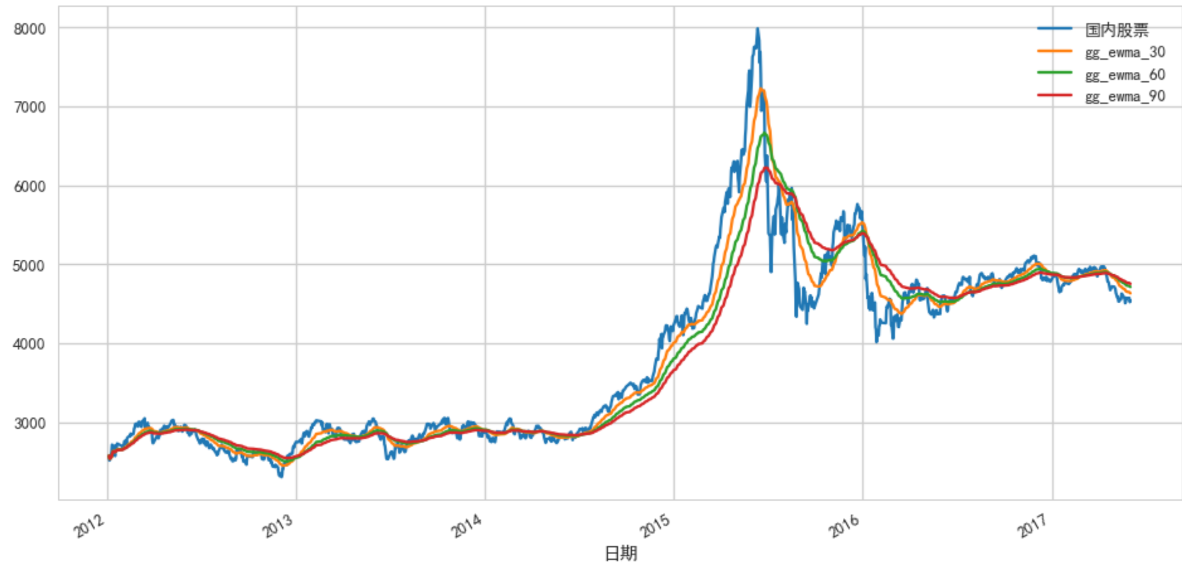
```
data['gg_ewma_30'] = data[['国内股票']].ewm(span=30).mean()
data['gg_ewma_60'] = data[['国内股票']].ewm(span=60).mean()
data['gg_ewma_90'] = data[['国内股票']].ewm(span=90).mean()
data['xg_ewma_30'] = data[['香港股票']].ewm(span=30).mean()
data['xg_ewma_60'] = data[['香港股票']].ewm(span=60).mean()
data['xg_ewma_90'] = data[['香港股票']].ewm(span=90).mean()
```

In [15]:

```
data[['国内股票','gg_ewma_30','gg_ewma_60','gg_ewma_90']].plot(figsize=(16,8))
```

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x17a62765ba8>

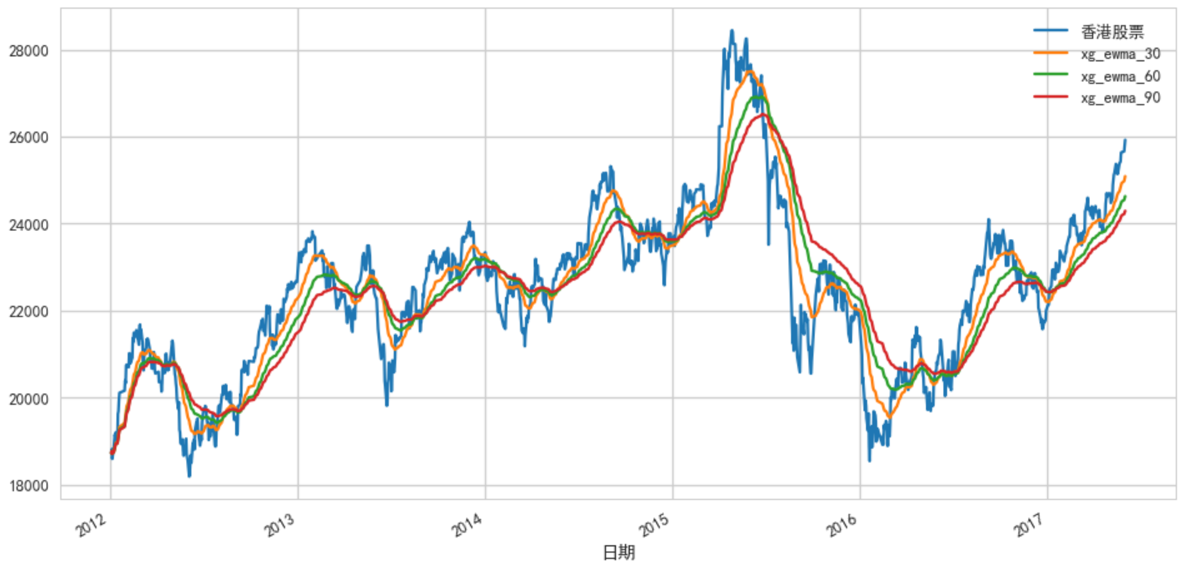


In [16]:

```
data[['香港股票','xg_ewma_30','xg_ewma_60','xg_ewma_90']].plot(figsize=(16,8))
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17a61fce048>
```



计算EWMA30天的RMSE,MAE

In [17]:

```
mse_gg_ewma = mean_squared_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_ewma_30'])
rmse_gg_ewma = np.sqrt(mse_gg_ewma)

mae_gg_ewma = mean_absolute_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_ewma_30'])

mse_xg_ewma = mean_squared_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_ewma_30'])
rmse_xg_ewma = np.sqrt(mse_xg_ewma)

mae_xg_ewma = mean_absolute_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_ewma_30'])
```

In [18]:

```
res_ewma = pd.DataFrame([[mse_gg_ewma,rmse_gg_ewma,mae_gg_ewma],[mse_xg_ewma,rmse_xg_ewma,mae_xg_ewm
             index=['国内股票','香港股票'],
             columns=['EWMA30_MSE','EWMA30_RMSE','EWMA30_MAE']).T
```

In [19]:

```
res_ewma
```

Out[19]:

	国内股票	香港股票
EWMA30_MSE	62278.522944	416017.758776
EWMA30_RMSE	249.556653	644.994387
EWMA30_MAE	151.842400	503.395953

In [20]:

```
res = pd.concat([res,res_ewma])
```

In [21]:

```
res
```

Out[21]:

	国内股票	香港股票
SMA30_MSE	88497.567727	576898.744557
SMA30_RMSE	297.485408	759.538508
SMA30_MAE	178.808682	593.674498
EWMA30_MSE	62278.522944	416017.758776
EWMA30_RMSE	249.556653	644.994387
EWMA30_MAE	151.842400	503.395953

相比于SMA，EWMA的RMSE和MAE都有所下降。

- Holt’s Linear Trend method

In [22]:

```
data.to_csv('data/index4_data.csv',encoding='utf-8')
```

In [23]:

```
def plotseasonal(res, axes, title,color='steelblue' ):
    res.observed.plot(ax=axes[0], legend=False, title=title, color=color)
    axes[0].set_ylabel('Observed')
    res.trend.plot(ax=axes[1], legend=False, color=color)
    axes[1].set_ylabel('Trend')
    res.seasonal.plot(ax=axes[2], color=color, legend=False)
    axes[2].set_ylabel('Seasonal')
    res.resid.plot(ax=axes[3], color=color, legend=False)
    axes[3].set_ylabel('Residual')
```

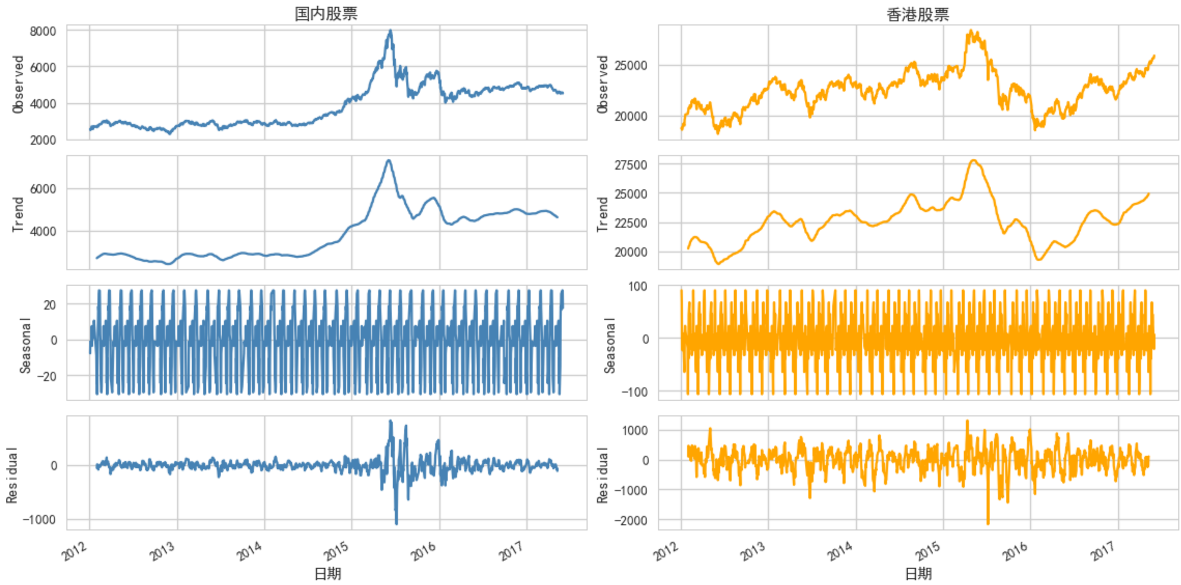
In [24]:

```
gg_hl = sm.tsa.seasonal_decompose(data['国内股票'],freq=30)
xg_hl = sm.tsa.seasonal_decompose(data['香港股票'],freq=30)
```

In [25]:

```
fig, axes = plt.subplots(ncols=2, nrows=4, sharex=True, figsize=(16,8))
plotseasonal(gg_hl, axes[:,0],'国内股票')
plotseasonal(xg_hl, axes[:,1],'香港股票', color='orange')

plt.tight_layout()
```



In [27]:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing,Holt,SimpleExpSmoothing,HoltWintersResu
```

In [36]:

```
fit_HL_gg = Holt(np.asarray(data['国内股票'])).fit(smoothing_level = 0.3,smoothing_slope = 0.1)
data['gg_HL']=fit_HL_gg.predict(start=0,end=data.shape[0]-1)
```

In [39]:

```
fit_HL_xg = Holt(np.asarray(data['香港股票'])).fit(smoothing_level = 0.3,smoothing_slope = 0.1)
data['xg_HL']=fit_HL_xg.predict(start=0,end=data.shape[0]-1)
```

- Holt Exponential

In [42]:

```
fit_HE_gg = Holt(np.asarray(data['国内股票']),exponential=True).fit(smoothing_level = 0.3,smoothing_
data['gg_HE']=fit_HE_gg.predict(start=0,end=data.shape[0]-1)
```

In [43]:

```
fit_HE_xg = Holt(np.asarray(data['香港股票']),exponential=True).fit(smoothing_level = 0.3,smoothing_
data['xg_HE']=fit_HE_xg.predict(start=0,end=data.shape[0]-1)
```

- Damped Trend

In [44]:

```
fit_DT_gg = Holt(np.asarray(data['国内股票']),damped=True).fit(smoothing_level = 0.3,smoothing_slope
data['gg_DT']=fit_DT_gg.predict(start=0,end=data.shape[0]-1)
```

In [45]:

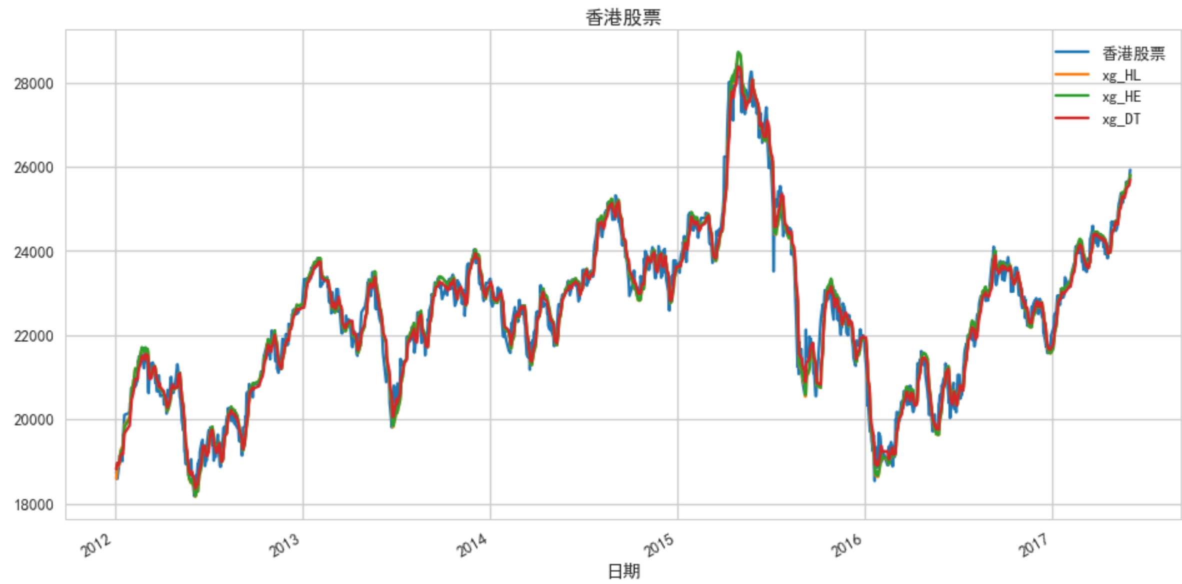
```
fit_DT_xg = Holt(np.asarray(data['香港股票']),damped=True).fit(smoothing_level = 0.3,smoothing_slope
data['xg_DT']=fit_DT_xg.predict(start=0,end=data.shape[0]-1)
```

In [54]:

```
data[['香港股票','xg_HL','xg_HE','xg_DT']].plot(figsize=(16,8),title="香港股票")
```

Out[54]:

<matplotlib.axes._subplots.AxesSubplot at 0x17a62953f28>



图中显示的是样本内(in sample)拟合结果，很明显是过拟合了。

下面计算MSE,RMSE,MAE

In [65]:

```
mse_gg_HL = mean_squared_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_HL'])
rmse_gg_HL = np.sqrt(mse_gg_HL)
mae_gg_HL = mean_absolute_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_HL'])

mse_gg_HE = mean_squared_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_HE'])
rmse_gg_HE = np.sqrt(mse_gg_HE)
mae_gg_HE = mean_absolute_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_HE'])

mse_gg_DT = mean_squared_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_DT'])
rmse_gg_DT = np.sqrt(mse_gg_DT)
mae_gg_DT = mean_absolute_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_DT'])
```

In [66]:

```
mse_xg_HL = mean_squared_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_HL'])
rmse_xg_HL = np.sqrt(mse_xg_HL)
mae_xg_HL = mean_absolute_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_HL'])

mse_xg_HE = mean_squared_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_HE'])
rmse_xg_HE = np.sqrt(mse_xg_HE)
mae_xg_HE = mean_absolute_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_HE'])

mse_xg_DT = mean_squared_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_DT'])
rmse_xg_DT = np.sqrt(mse_xg_DT)
mae_xg_DT = mean_absolute_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_DT'])
```

In [67]:

```
res_HL = pd.DataFrame([[mse_gg_HL,rmse_gg_HL,mae_gg_HL],[mse_xg_HL,rmse_xg_HL,mae_xg_HL]],
             index=['国内股票','香港股票'],
             columns=['HL_MSE','HL_RMSE','HL_MAE']).T
res_HE = pd.DataFrame([[mse_gg_HE,rmse_gg_HE,mae_gg_HE],[mse_xg_HE,rmse_xg_HE,mae_xg_HE]],
             index=['国内股票','香港股票'],
             columns=['HE_MSE','HE_RMSE','HE_MAE']).T
res_DT = pd.DataFrame([[mse_gg_DT,rmse_gg_DT,mae_gg_DT],[mse_xg_DT,rmse_xg_DT,mae_xg_DT]],
             index=['国内股票','香港股票'],
             columns=['DT_MSE','DT_RMSE','DT_MAE']).T
```

In [69]:

```
res = pd.concat([res,pd.concat([res_HL,res_HE,res_DT])])
```

In [70]:

```
res
```

Out[70]:

	国内股票	香港股票
SMA30_MSE	88497.567727	576898.744557
SMA30_RMSE	297.485408	759.538508
SMA30_MAE	178.808682	593.674498
EWMA30_MSE	62278.522944	416017.758776
EWMA30_RMSE	249.556653	644.994387
EWMA30_MAE	151.842400	503.395953
HL_MSE	13906.946906	114677.589241
HL_RMSE	117.927719	338.640797
HL_MAE	69.306345	263.018980
HE_MSE	13631.418063	114346.239659
HE_RMSE	116.753664	338.151208
HE_MAE	68.752372	262.931316
DT_MSE	11986.401221	100890.649489
DT_RMSE	109.482424	317.632885
DT_MAE	65.470113	244.927531

- Holt Winters

In [71]:

```
fit_hw_gg = ExponentialSmoothing(np.asarray(data['国内股票']) ,seasonal_periods=30 ,trend='add', sea
fit_hw_xg = ExponentialSmoothing(np.asarray(data['香港股票']) ,seasonal_periods=30 ,trend='add', sea
```

In [72]:

```
data['gg_HW']=fit_hw_gg.predict(start=0,end=data.shape[0]-1)
data['xg_HW']=fit_hw_xg.predict(start=0,end=data.shape[0]-1)
```

In [76]:

```
data[['国内股票','gg_HW']].plot(figsize=(16,8),title="国内股票 Holt Winter")
```

Out[76]:

<matplotlib.axes._subplots.AxesSubplot at 0x17a65422f60>

In [75]:

```
data[['香港股票','xg_HW']].plot(figsize=(16,8),title="香港股票 Holt Winter")
```

Out[75]:

<matplotlib.axes._subplots.AxesSubplot at 0x17a64c38780>

In [77]:

```
mse_gg_HW = mean_squared_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_HW'])
rmse_gg_HW = np.sqrt(mse_gg_HW)
mae_gg_HW = mean_absolute_error(data.iloc[30:]['国内股票'],data.iloc[30:]['gg_HW'])
```

In [78]:

```
mse_xg_HW = mean_squared_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_HW'])
rmse_xg_HW = np.sqrt(mse_xg_HW)
mae_xg_HW = mean_absolute_error(data.iloc[30:]['香港股票'],data.iloc[30:]['xg_HW'])
```

In [79]:

```
res_HW = pd.DataFrame([[mse_gg_HW,rmse_gg_HW,mae_gg_HW],[mse_xg_HW,rmse_xg_HW,mae_xg_HW]],
             index=['国内股票','香港股票'],
             columns=['HW_MSE','HW_RMSE','HW_MAE']).T
```

In [80]:

```
res_HW
```

Out[80]:

	国内股票	香港股票
HW_MSE	6601.105841	75728.360340
HW_RMSE	81.247190	275.187864
HW_MAE	50.688125	210.148808

In [81]:

```
res = pd.concat([res,res_HW])
```

In [82]:

```
res
```

Out[82]:

	国内股票	香港股票
SMA30_MSE	88497.567727	576898.744557
SMA30_RMSE	297.485408	759.538508
SMA30_MAE	178.808682	593.674498
EWMA30_MSE	62278.522944	416017.758776
EWMA30_RMSE	249.556653	644.994387
EWMA30_MAE	151.842400	503.395953
HL_MSE	13906.946906	114677.589241
HL_RMSE	117.927719	338.640797
HL_MAE	69.306345	263.018980
HE_MSE	13631.418063	114346.239659
HE_RMSE	116.753664	338.151208
HE_MAE	68.752372	262.931316
DT_MSE	11986.401221	100890.649489
DT_RMSE	109.482424	317.632885
DT_MAE	65.470113	244.927531
HW_MSE	6601.105841	75728.360340
HW_RMSE	81.247190	275.187864
HW_MAE	50.688125	210.148808

Holt Winter 获得了最好的RMSE和MAE。

总结：

这一次我们使用Python实现了多种指数平滑模型，其中有些实现方法并不是非常精细，主要是参数还缺乏严格的回归测算和验证。这仅仅是一个演示而已。

附录：采用R代码来实现

In [1]:

```
library(forecast)
DataR <- read.csv(file="data/index4_data.csv", header=TRUE, sep=",",encoding="UTF-8")
```

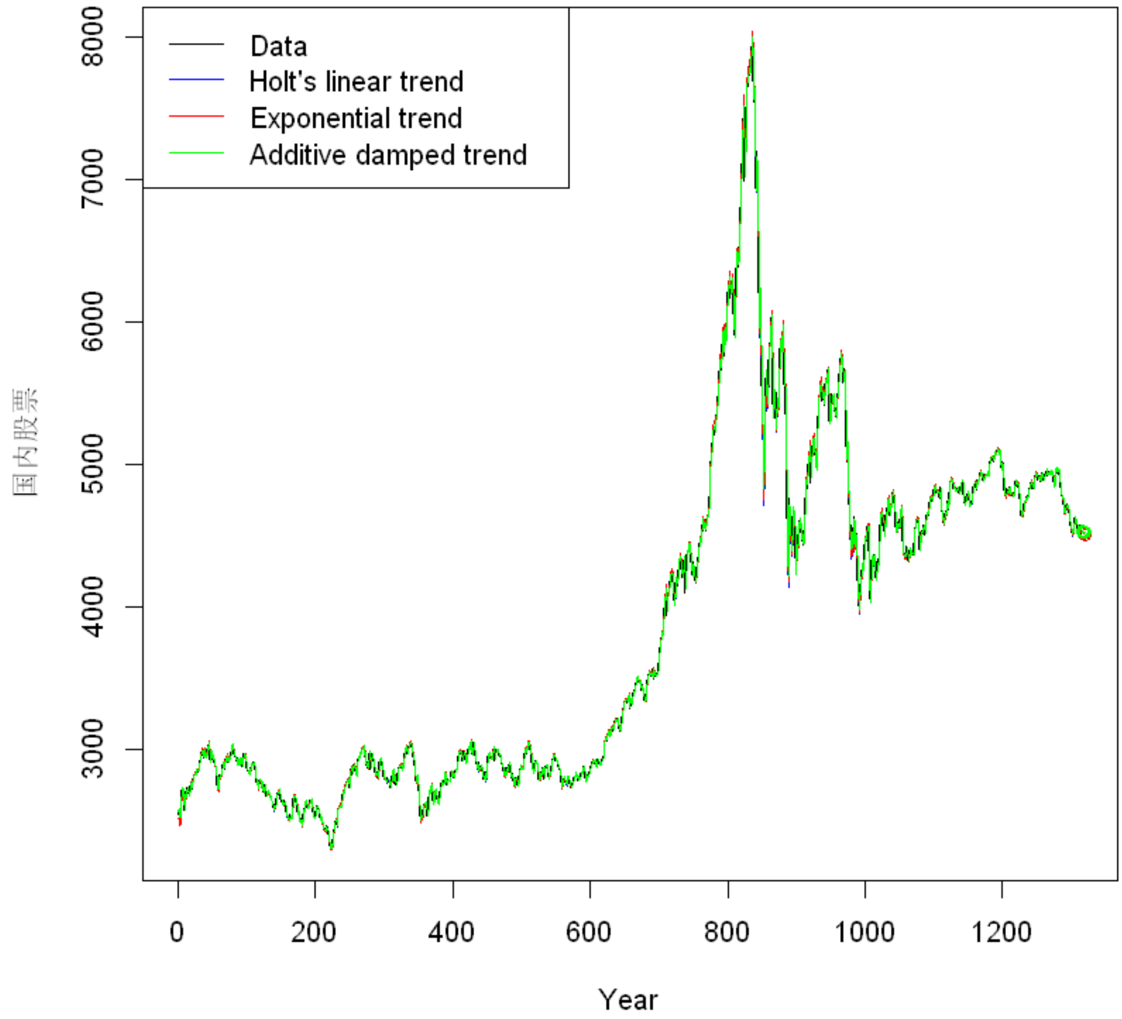
In [2]:

```
fit_HL=holt(as.ts(DataR$国内股票),alpha=0.8,beta=0.2,initial="simple",h=5)
fit_ET=holt(as.ts(DataR$国内股票),alpha=0.8,beta=0.2,initial="simple",exponential=TRUE,h=5)
fit_DT=holt(as.ts(DataR$国内股票), alpha=0.8, beta=0.2, damped=TRUE, initial="optimal", h=5)
```

In [3]:

```
plot(as.ts(DataR$国内股票), ylab="国内股票", xlab="Year")
lines(fitted(fit_HL), col="blue")
lines(fitted(fit_ET), col="red")
lines(fitted(fit_DT), col="green")

lines(fit_HL$mean, col="blue", type="o")
lines(fit_ET$mean, col="red", type="o")
lines(fit_DT$mean, col="green", type="o")
legend("topleft", lty=1, col=c("black","blue","red","green"),
   c("Data","Holt's linear trend","Exponential trend","Additive damped trend"))
```



In [4]:

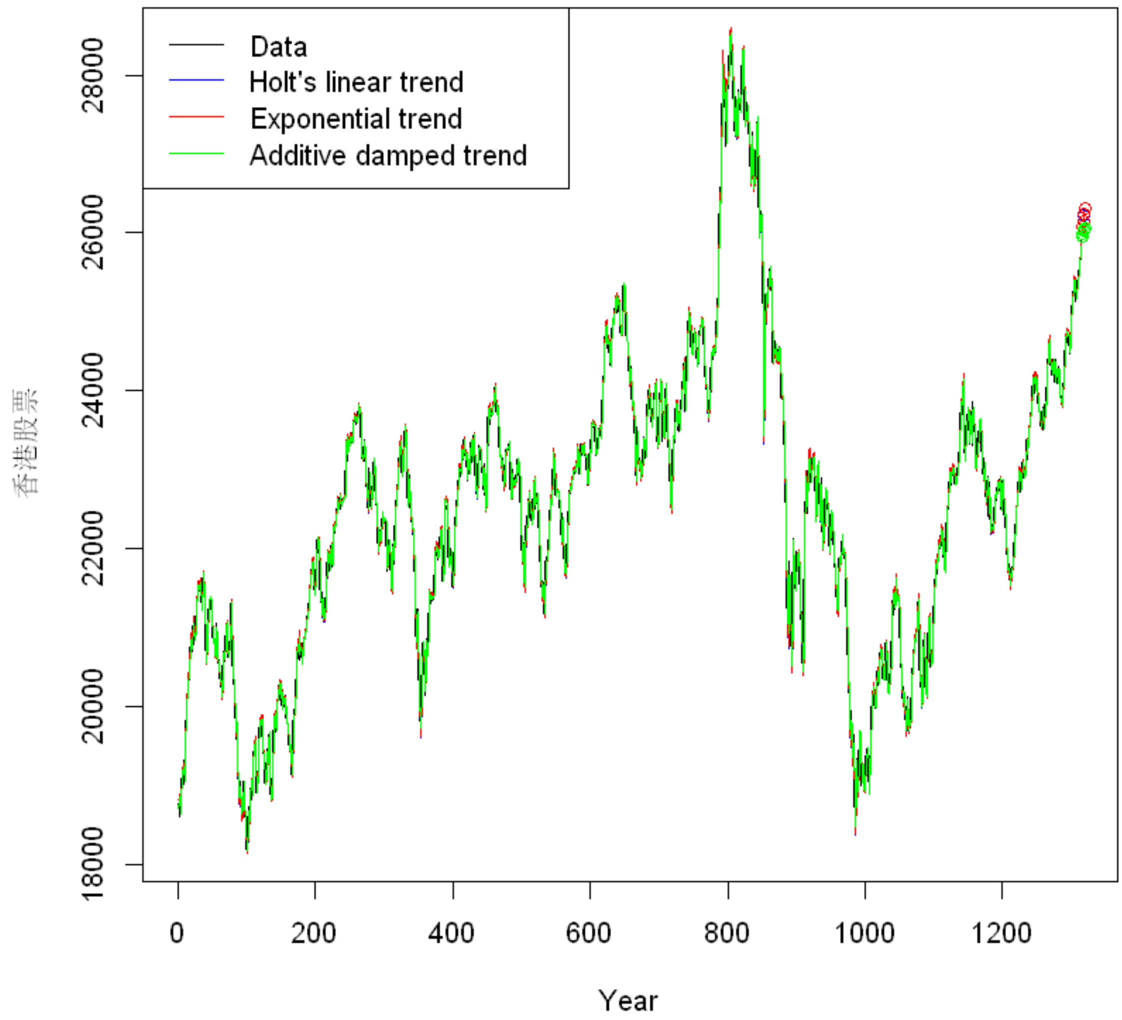
```
df = ts.union(fit_HL$fitted,fit_ET$fitted,fit_DT$fitted,dframe=TRUE)
```

In [5]:

```
fit_HL=holt(as.ts(DataR$香港股票),alpha=0.8,beta=0.2,initial="simple",h=5)
fit_ET=holt(as.ts(DataR$香港股票),alpha=0.8,beta=0.2,initial="simple",exponential=TRUE,h=5)
fit_DT=holt(as.ts(DataR$香港股票), alpha=0.8, beta=0.2, damped=TRUE, initial="optimal", h=5)
```

In [6]:

```
plot(as.ts(DataR$香港股票), ylab="香港股票", xlab="Year")
lines(fitted(fit_HL), col="blue")
lines(fitted(fit_ET), col="red")
lines(fitted(fit_DT), col="green")
lines(fit_HL$mean, col="blue", type="o")
lines(fit_ET$mean, col="red", type="o")
lines(fit_DT$mean, col="green", type="o")
legend("topleft", lty=1, col=c("black","blue","red","green"),
   c("Data","Holt's linear trend","Exponential trend","Additive damped trend"))
```



In [7]:

```
df1 = ts.union(fit_HL$fitted,fit_ET$fitted,fit_DT$fitted,dframe=TRUE)
```

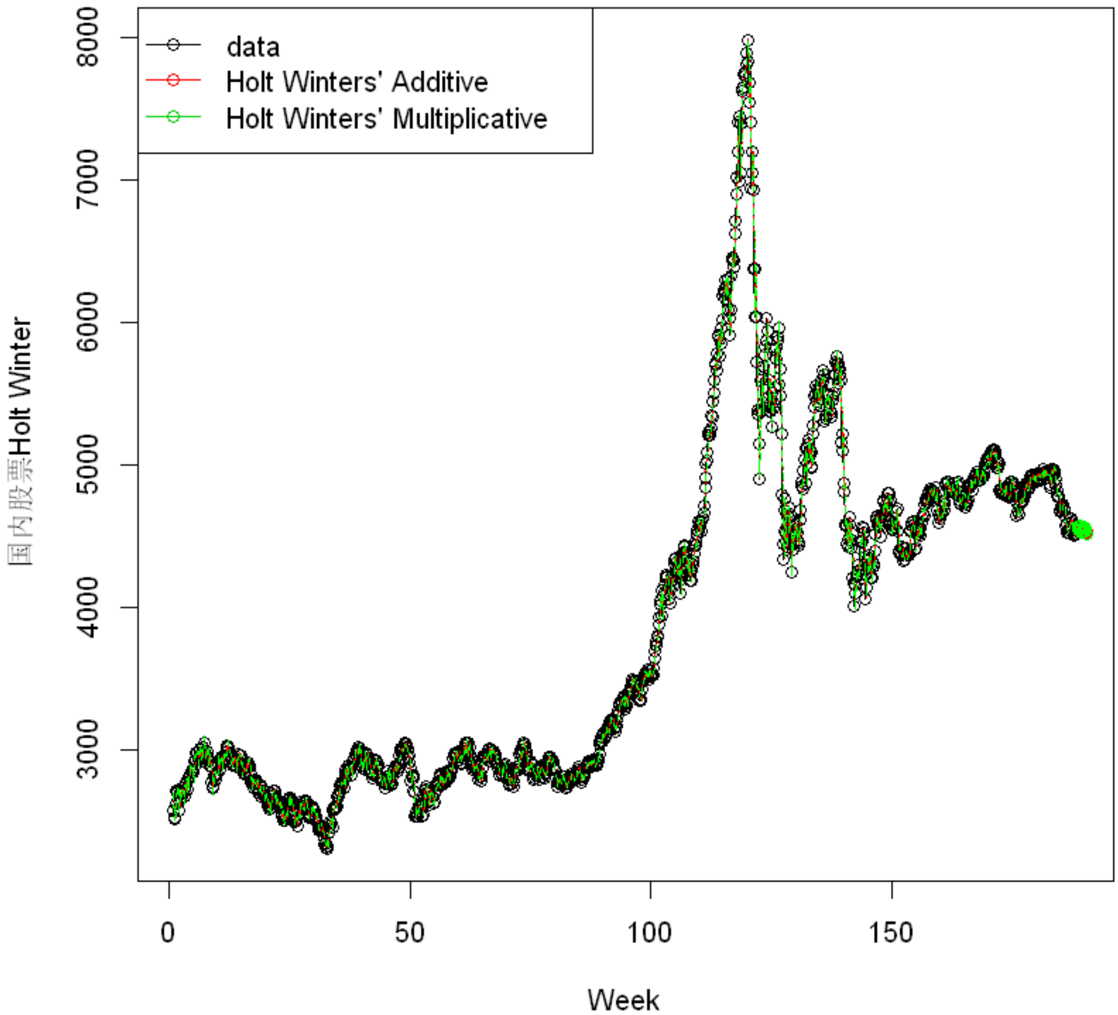
- Holt Winter Smoothing

In [13]:

```
fit1 <- hw(ts(DataR$国内股票,frequency=7),seasonal="additive")
fit2 <- hw(ts(DataR$国内股票,frequency=7),seasonal="multiplicative")
```

In [17]:

```
plot(ts(DataR$国内股票,frequency=7),ylab="国内股票Holt Winter",
     type="o", xlab="Week")
lines(fitted(fit1), col="red", lty=2)
lines(fitted(fit2), col="green", lty=2)
lines(fit1$mean, type="o", col="red")
lines(fit2$mean, type="o", col="green")
legend("topleft",lty=1, pch=1, col=1:3,
  c("data","Holt Winters' Additive","Holt Winters' Multiplicative"))
```



In [52]:

```
colnames(df) <- c('gg_HL','gg_ET','gg_DT')
colnames(df1) <- c('xg_HL','xg_ET','xg_DT')
```

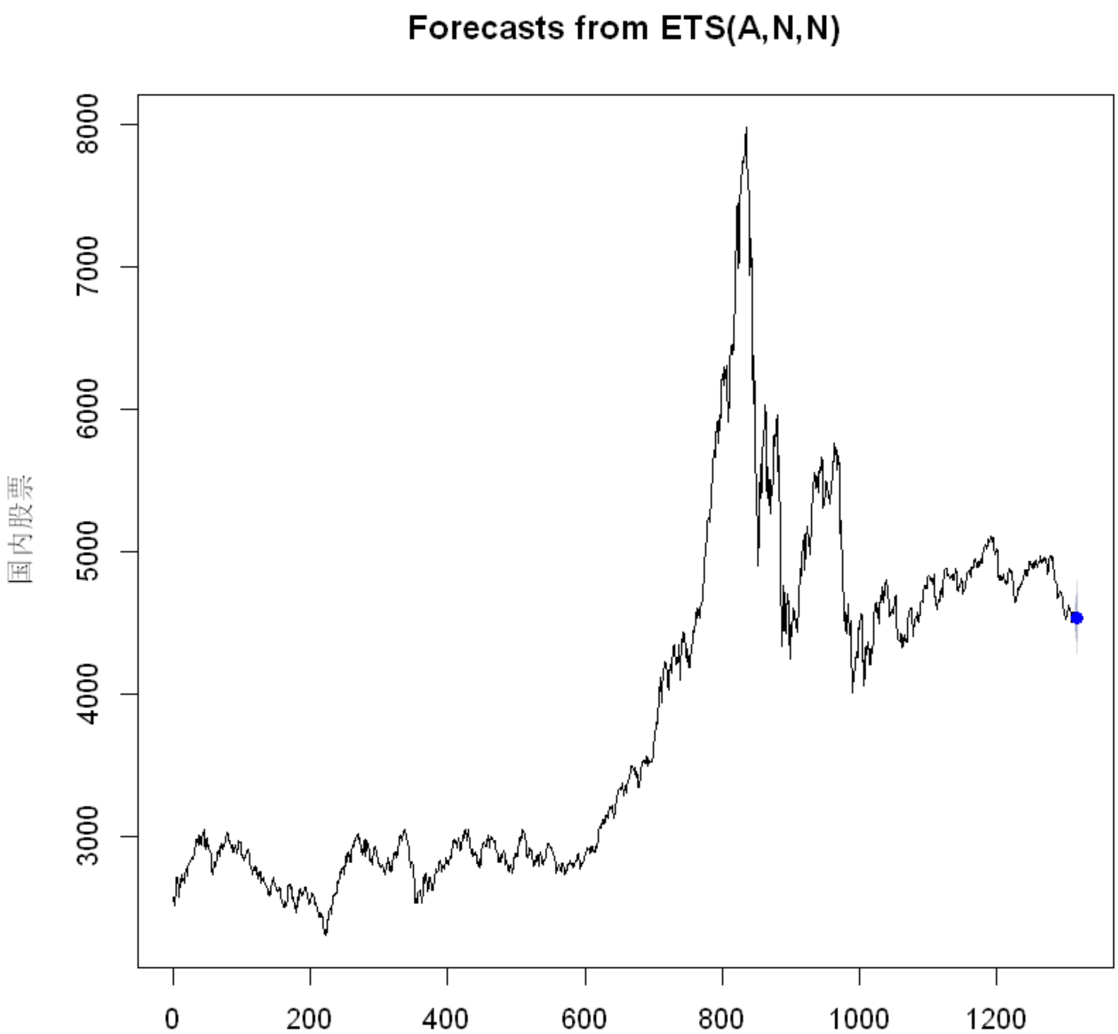
- Innovation State Space Model

指定ETS(ANN)模型，就是加性错误、无趋势和无周期

In [24]:

```
fit_ets_gg <- ets(as.ts(DataR$国内股票), model="ANN")
plot(forecast(fit_ets_gg, h=3), ylab="国内股票")
fit_ets_gg$par
```

alpha
0.999899642674678
l
2625.83228814494



In [23]:

```
fit_model_ets <- ets(as.ts(DataR$国内股票))
summary(fit_model_ets)
```

```
ETS(M,N,N)

Call:
 ets(y = as.ts(DataR$国内股票))

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 2625.8323

  sigma:  0.0163

     AIC     AICc      BIC
20220.05 20220.07 20235.60

Training set error measures:
                   ME     RMSE      MAE        MPE     MAPE     MASE      ACF1
Training set 1.453165 76.60358 45.72725 0.02802061 1.122095 1.000143 0.1212894
```

In [67]:

```
write.csv(cbind(DataR,df,df1),file="data/index_new.csv",fileEncoding="UTF-8",quote=FALSE)
```