时间序列的平稳性及使用差分法处理非平稳时间序列

python万
发布时间：01-18 17:16

时间序列的平稳性简介

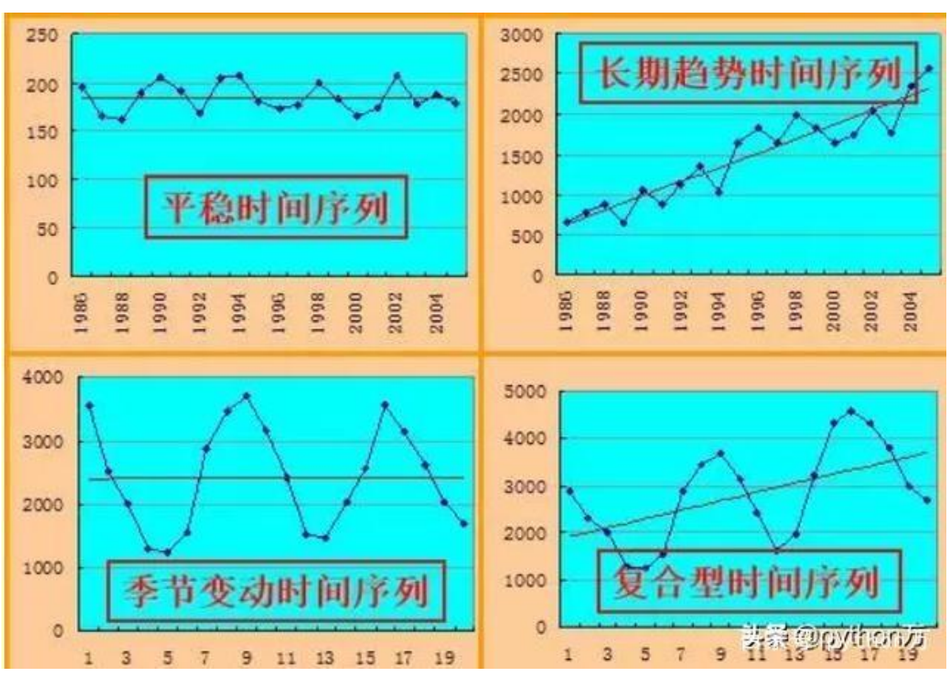
由于系统编辑器限制，无序列表代表代码行!

前面几篇文章中，我们分别讨论了pandas.date_range、pandas.DataFrame.rasample及pandas.DataFrame.rolling函数，其实这些函数可以把它归类为时间序列建模过程中的预处理函数，最终的目的都是建立时间序列模型，但是建立时间序列模型的前提是序列是平稳的，这是因为我们用时间序列做预测时，我们的随机变量的基本特性必须能在包括未来阶段的一个长时期里维持不变，否则，基于历史和现状来预测未来的思路便是错误的。所以在时间序列建模时第一步就是要将不平稳的序列平稳化。

所以，这里首先对时间序列的平稳性做简单介绍，如果一个时间序列均值没有系统的变化（无趋势）、方差没有变化，且严格消除了周期性变化，那么我们就称这种时间序列是平稳的。其中，平稳性分为严平稳和弱平稳，严平稳是一种条件比较苛刻的平稳性定义,它认为只有当序列所有的统计性质都不会随着时间的推移而发生变化时,该序列才能被认为平稳.而弱平稳是使用序列的特征统计量来定义的一种平稳性.它认为序列的统计性质主要由它的低阶矩决定,所以只要保证序列低阶矩平稳（二阶）,就能保证序列的主要性质近似稳定.

非平稳序列与差分法

上面我们已经强调，进行时间序列建模的前提条件就是要求时间序列是平稳的，那么如果遇到非平稳的序列，我们首要的任务就是将非平稳的序列转化为平稳序列，一般来说非平稳时间序列可以分为四种：只包含长期趋势、季节变动和循环变动的时间序列和包含三种变动中两种或三种的时间序列，也就是复合型时间序列。如下图（图片来源：东山草堂君）：



平稳序列及非平稳序列

而非平稳序列的平稳化也有很多方法，主要有去除趋势、差分和变换。而本篇文章，我们主要讨论差分法，所谓的差分法就是将t时刻与t-1时刻的数据做差，得到新值的过

作者最新文章

时间序列的平稳性及使用差分法处理非平稳时间序列

python时间序列分析之_用pandas中的rolling函数计算时间窗口数据

Python时间序列分析之_时间重采样（降采样和升采样）

相关文章

懂Excel就能轻松入门Python数据分析包pandas(五)：重复值…

中移动营收净利双下滑或为必然，5G能否成为新的解药？

七旬老妪进山找草药走失，警犬引导警民成功救出

华泰联合证券董事长刘晓丹：纯粹大吃小的同质并购没有意义…

pandas.DataFrame.diff实现差分

pandas中提供了差分的函数，函数原型为：DataFrame.diff(periods=1, axis=0)，这个函数的参数只有2个，第一个表示差分阶数，第二个表示按行还是按列计算。下面直接上代码：

- import pandas as pd
- index=pd.date_range('20190116','20190130')
- data=[4,8,6,5,9,1,4,5,2,4,6,7,9,13,6]
- ser_data=pd.Series(data,index=index)
- ser_data.diff(2)

差分过程

这里默认是按行做差分，如果我们想按列做差分，只需要指定axis=1即可，如下：

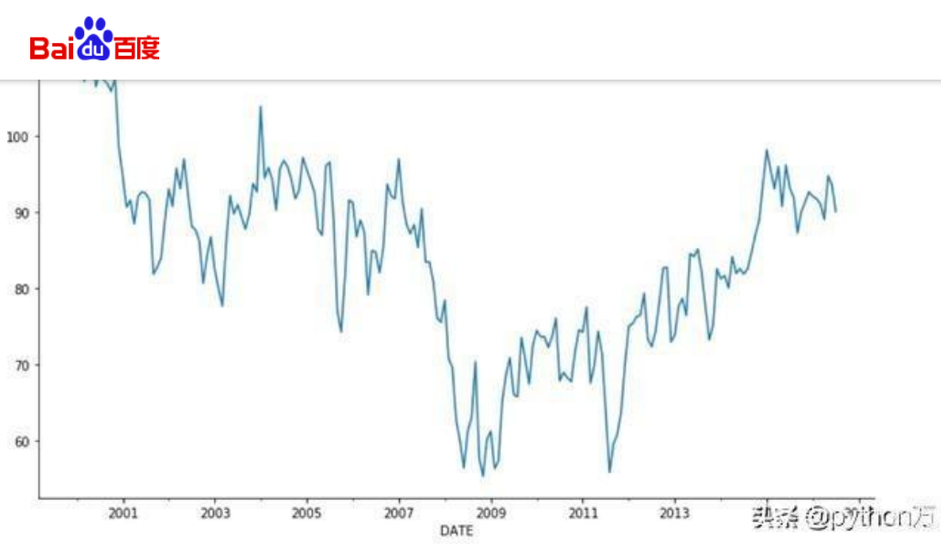
- df = pd.DataFrame({'a': [1, 2, 3, 4, 5, 6],... 'b': [1, 1, 2, 3, 5, 8],... 'c': [1, 4, 9, 16, 25, 36]})
- df.diff(axis=0,periods=2)
- df.diff(axis=1,periods=2)

按行和按列做差分

将非平稳时间序列经过差分平稳化案例

通过上面的介绍，我们已经清楚了时间序列平稳性的意义及pandas差分法函数的用法，下面我们通过演示一个例子来加深理解。

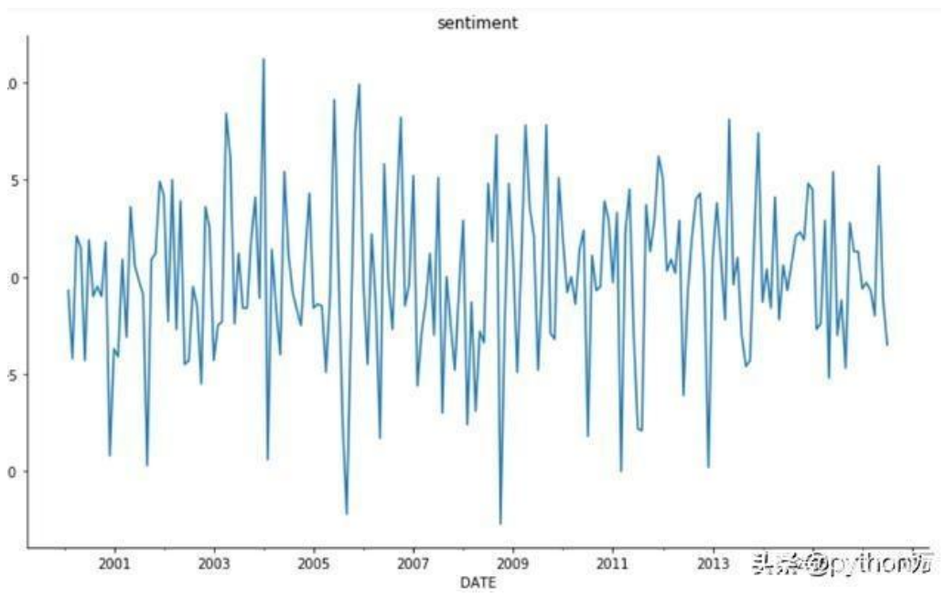
- st=pd.read_csv('sentiment.csv',index_col=0,parse_dates=[0])
- st.plot(figsize=(12,7))
- plt.legend(bbox_to_anchor=(1.25,0.5))
- plt.title('sentiment')sns.despine()



原始数据

上面的图表示的是原始数据的变化频率，现在我们通过一阶差分将原始数据进行处理，然后观察变化情况，如下：

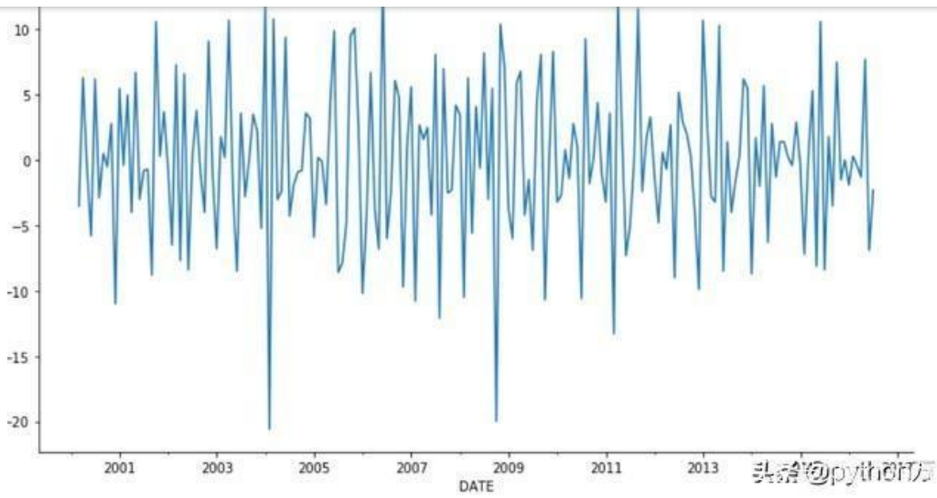
- st_1=st.diff(1)st_1.plot(figsize=(12,7))
- plt.legend(bbox_to_anchor=(1.25,0.5))
- plt.title('sentiment')
- sns.despine()



一阶差分图

在一阶差分的基础上在进行差分，变为二阶差分，如下：

- st_2=st_1.diff(1)
- st_2.plot(figsize=(12,7))
- plt.legend(bbox_to_anchor=(1.25,0.5))
- plt.title('sentiment')sns.despine()



二阶差分

可以看到，相比于原始数据，经过差分处理的数据已经变得相对平缓很多，这就是差分的作用。今天的内容到此为止，下篇开始讲ARIMA 差分自回归移动平均模型，喜欢的小伙伴点击关注。