预测：方法与实践 3 预测者的工具集

## 第三章 预测者的工具集

本章我们会讨论一些在不同预测场景中应用的常用工具。我们将介绍一些基准预测方法，并探讨如何通过数据变换与调整简化预测，如何判断预测是否充分利用了现有信息以及如何计算预测区间。

在未来的章节中，随着我们对一系列预测方法的开发和探索，本章中的每种方法都会被反复使用。

library(fpp2)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## ── Attaching packages ────────────────────────────────────────────── fpp2 2.4 ──

## ✔ ggplot2 3.3.6 ✔ fma 2.4   
## ✔ forecast 8.16 ✔ expsmooth 2.3

##

### 3.1 一些简单的预测方法

一些预测方法虽然简单但非常有效。本书将以下面的四种预测方法作为基准方法。

#### 均值法

此方法中，所有未来值的预测值等于历史数据的平均值。我们把历史数据记作y1,…,yT，预测值就可以表示为：^yT+h|T=¯y=(y1+⋯+yT)/T.

^yT+h|T代表基于y1,…,yT对yT+h的估计值。

meanf(y, h)  
# y 历史数据时间序列  
# h 预测期数

#### Naïve 方法

在 naïve 预测方法中，我们简单地将所有预测值设为最后一次的观测值，即：^yT+h|T=yT.这种方法在很多经济和金融时间序列预测中表现得非常好。

naive(y, h)  
rwf(y, h) # 二者等价

当数据服从随机游动过程时 Naïve 预测是最优的，（见 @ref(stationarity 节)），因此Naïve 方法也被称为“随机游动预测”。

#### 季节性 Naïve 方法

季节性 Naïve 方法与 Naïve 方法类似，它适用于季节性变化剧烈的数据。我们将每个预测值设为同一季节的前一期观测值（例如：去年的同一个月）。那么，T+h时刻的预测值可以记作：^yT+h|T=yT+h−km,等式中m为周期长度，k是(h−1)/m的整数部分（也就是在T+h时刻前预测期所包含的整年数）。这个等式并没有看起来那么复杂，例如，对于月度数据，未来所有二月的预测值都等于前一年二月的观测值。对于季度数据，未来所有第二季度的预测值都等于前一年第二季度的观测值。相似的规则适用于其他的月份、季度和其他周期长度。

snaive(y, h)

#### 漂移法

相对于 Naïve 方法，趋势法（漂移法）允许预测值随着时间的推移增大或减小，并且我们假定单位时间改变量（称作 “漂移”）等于历史数据的平均改变量，因此T+h时刻的预测值可以表示为:^yT+h|T=yT+hT−1T∑t=2(yt−yt−1)=yT+h(yT−y1T−1).这相当于把第一个观测点和最后一个观测点连成一条直线并延伸到未来预测点。

rwf(y, h, drift=TRUE)

#### 例子

图[3.1](https://otexts.com/fppcn/simple-methods.html#fig:beerf)展示了将前三种方法应用于啤酒季度销售数据的预测结果。

# 设定1992年至2007年为训练数据  
beer2 <- window(ausbeer,start=1992,end=c(2007,4))  
# 画出预测图  
autoplot(beer2) +  
 autolayer(meanf(beer2, h=11)$mean, series="均值") +  
 autolayer(naive(beer2, h=11)$mean, series="Naïve") +  
 autolayer(snaive(beer2, h=11)$mean, series="季节性 naïve") +  
 ggtitle("啤酒季度销量的预测") +  
 xlab("年份") + ylab("百万升") +  
 guides(colour=guide\_legend(title="预测")) +  
 theme(text=element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

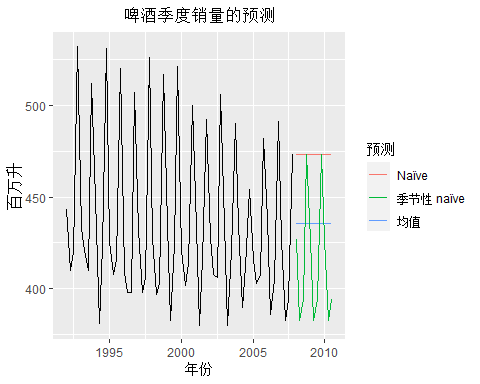


图 3.1: 澳大利亚啤酒季度销量预测。

图[3.2](https://otexts.com/fppcn/simple-methods.html#fig:googf)是基于谷歌公司200天的每日股票收盘价运用非周期方法得出的预测结果。

# 画出预测图  
autoplot(goog200) +  
 autolayer(meanf(goog200, h=40), PI=FALSE, series="均值") +  
 autolayer(rwf(goog200, h=40), PI=FALSE, series="Naïve") +  
 autolayer(rwf(goog200, drift=TRUE, h=40), PI=FALSE, series="漂移") +  
 ggtitle("每日谷歌股票收盘价（截止到2013年12月6日）的预测") +  
 xlab("天") + ylab("收盘价 （美元）") +  
 guides(colour=guide\_legend(title="预测")) +  
 theme(text=element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

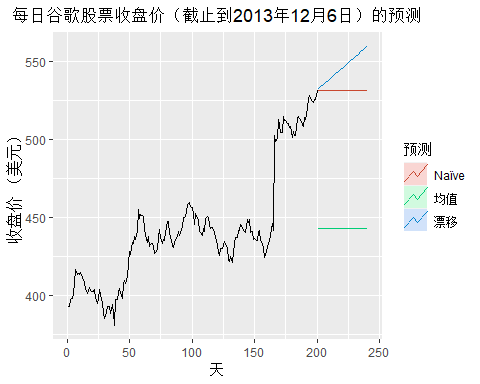


图 3.2: 基于谷歌公司200天的每日股票收盘价预测。

有时候这些最简单的方法可能会是最好的预测方法，但是在很多情况下，这些方法只是作为基准方法而不是被直接运用。也就是说，我们提出的任何预测方法都会和这些简单方法比较，以保证其预测结果优于这些简单方法，否则，新的方法就不值得考虑。

### 3.2 变换和调整

调整历史数据通常会使预测更加简单。 本节我们将会介绍四种调整方法：日历调整、人口调整、通胀调整和数学变换。这些调整和变换的目的是通过去除数据变化的已知来源或增加整个数据集的一致性来简化历史数据的模式。更简单的数据模式往往可以让我们得到更精确的预测结果。

#### 日历调整

季节性数据中的某些变化可能是来自简单的日历效应。这种情况下，在拟合预测模型之前消除这些变化因素往往会使预测更简单。monthdays()函数可以计算每个月或每个季度所包含的天数。

例如，假如你正在研究某农场每月的牛奶产量，一年之中，除了季节性变化因素以外，每月牛奶产量还会由于每月的天数不同而发生变化。

dframe <- cbind(月均 = milk, 日均=milk/monthdays(milk))  
 autoplot(dframe, facet=TRUE) +  
 xlab("年") + ylab("英镑") +  
 ggtitle("每头奶牛牛奶产量的预测") +  
 theme(text=element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

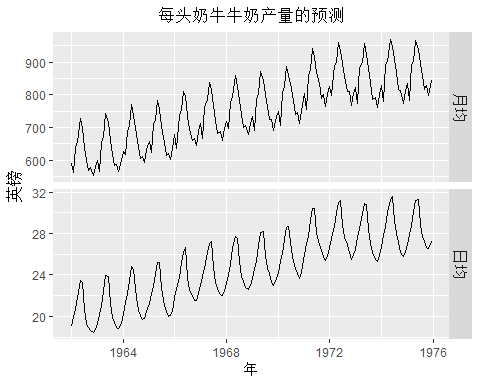


图 3.3: 每头奶牛的月度牛奶产量预测。

可以看到，日均产量图的季节性模式比月均产量图更简洁明了。日均产量图有效消除了不同月份天数不同造成的影响。简单的模式往往更易于建立模型并得出更精确的预测结果。 对于交易天数随着月份改变的销售数据，我们可以做相似的调整。在这种情况下，可以用每个交易日的销量而非每月的总销量来建立模型。

#### 人口调整

任何受人口数量影响的数据都可以调整为人均数据，换句话说，考虑每人（或每千人，或每百万人）而非所有人。 假如你正在研究某区域内医院床位数量随着时间变化的情况，那么通过考虑每千人的床位数量来去除人口数量变化的影响，结果就更容易解释。这样你就能看出床位数量是否有实际的增长，或者床位数量的增长是否完全由人口数量增长所导致。很有可能出现床位总数量增长但每千人的床位数量却在下降的情况，这是由于人口增长速度快于医院床位数量增长速度造成的。对于大多数受人口数量变化影响的数据，最好的选择是采用人均数据而非总量数据。

#### 通胀调整

受货币价值影响的数据在建立模型之前经过最好经过通胀调整。例如，由于通货膨胀的影响，一座新房的平均价格在过去的几十年里提高了很多，一座标价20万美元的房子并不等同于二十年前标价20万美元的房子。因此金融时间序列数据经常进行调整以便所有价格都以某一固定年份的美元价值标价。例如，房价数据可以用2000年的美元标价。

进行通货膨胀调整需要用到一种价格指数，我们把它记作zt，t年的原始房价记作yt，以2000年美元价值调整后的房价记作xt，则xt=yt/zt∗z2000。价格指数通常由政府部门制定，对于消费品，常用的价格指数是消费物价指数（Consumer Price Index，CPI）。

### 数学变换

如果数据数值范围发生变化，而数据的波动变化也会随之变大变小，那么变换会很有用。例如，对数变换经常是有用的。我们把原始观测值记作y1,…,yT，变换后的观测值记作w1,…,wT，则wt=log(yt)。对数变换之所以有用，是因为它们易于解释：对数值的变化是原始值的相对（或百分比）变化。假设进行以10为底的对数变换，那么对数值加1相当于原始值乘以10。对数变换的另一个特点在于能够限制在原来测度下的预测值为正数。

有时候其他变换也会被用到（虽然它们不是那么容易解释），例如平方根和立方根变换，它们也被称作**幂变换**，因为可以以wt=ypt的形式表示。

“Box-cox变换”是一个既包含对数变换，又包含幂变换的依赖于参数λ的变换族，其定义如下：wt={log(yt)ifλ=0;(yλt−1)/λotherwise.

Box-cox变换中的对数变换通常以自然对数e为底，因此如果λ=0，进行自然对数变换，否则，进行幂变换，然后简单缩放。

如果λ=1, 那么wt=yt−1, 变换后的数据向下移动，但是时间序列的形状没有改变。但是对于所有其他的λ值，时间序列的形状会改变。

使用下方的滑动按钮查看λ的变化对澳大利亚月度用电需求数据变换的影响：

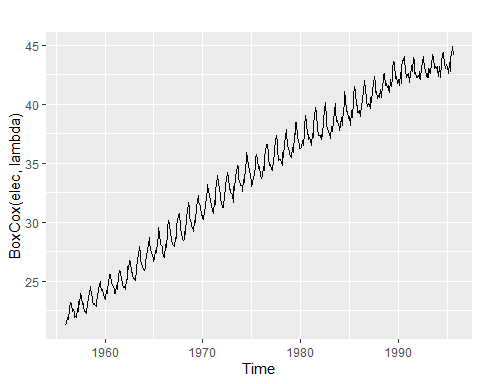
合适的λ值使得在整个时间序列范围内，周期性变化大小差不多相同，因此让预测更加简单。在此例中，λ=0.30的变换效果相当好，虽然在0-0.5之间的所有λ值得出了相似的变换结果。

BoxCox.lambda()函数会给你选择一个合适的λ值。

(lambda <- BoxCox.lambda(elec))

## [1] 0.2654076

#> [1] 0.2654  
autoplot(BoxCox(elec,lambda))



选择好变换以后，我们需要利用变换后的数据进行预测，得出预测值后，我们需要进行逆变换得到原始测度上的预测值。逆Box-Cox变换如下表示:yt={exp(wt)ifλ=0;(λwt+1)1/λotherwise.(3.1)

#### 幂变换的特点

* 如果出现部分yt≤0，除非给所有观测值加上一个常数，否则不能进行幂变换，。
* 选择一个简单的λ值会让解释更容易。
* 预测结果对λ值相对不敏感
* 很多情况下不需要进行变换
* 有时候变换对于预测结果影响不大，但对预测区间有很大影响。

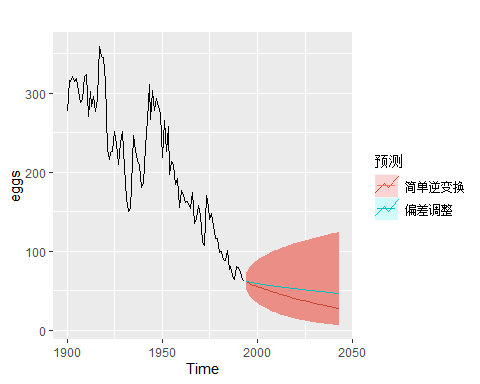
#### 偏差调整（Bias Adjustments）

进行数学变换例如Box-Cox变换的一个问题在于逆变换后得到的预测值不是预测分布的平均值，事实上，它通常是预测分布的中位数。（假定变换空间上的分布是对称的）很多情况下，这可以接受，但有时需要预测均值。例如，你可能希望汇总各地区的销售预测值来形成全国预测值，这是中位数做不到的，但是平均数可以做到。

对于Box-Cox变换，逆变换的均值如下给出：yt=⎧⎪ ⎪ ⎪⎨⎪ ⎪ ⎪⎩exp(wt)[1+σ2h2]如果λ=0；(λwt+1)1/λ[1+σ2h(1−λ)2(λwt+1)2]否则；(3.2)σ2h是h步预测方差，预测方差越大，均值和中位数之间的差异越大。

公式[(3.1)](https://otexts.com/fppcn/transformations.html#eq:backtransform)给出的简单逆变换预测与公式[(3.2)](https://otexts.com/fppcn/transformations.html#eq:backtransformmean)给出的均值变换之间的差异叫做偏差（Bias），当我们用均值而非中位数时，称预测点是经过“偏差调整”（Bias-adjusted）的。下面的例子给出了偏差调整起到的作用。在这个例子中我们利用趋势法（Drift method）进行对数变换(λ=0)来预测每年的鸡蛋平均价格，对数变换可以保证预测值和预测区间为正。

fc <- rwf(eggs, drift=TRUE, lambda=0, h=50, level=80)  
fc2 <- rwf(eggs, drift=TRUE, lambda=0, h=50, level=80, biasadj=TRUE)  
autoplot(eggs) +  
 autolayer(fc, series="简单逆变换") +  
 autolayer(fc2$mean, series="偏差调整") +  
 guides(colour=guide\_legend(title="预测")) +  
 theme(text=element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))



蓝线表示预测值中位数而红色线表示预测值均值，请注意当我们使用偏差调整时，倾斜的预测分布如何拉升预测点。 偏差调整并不会默认进行，如果你想让预测结果为均值而非中位数，在选择Box-Cox变换参数时令biasadj=TRUE。

### ## 3.3残差诊断

#### 拟合值

时间序列中的每个观测值都可以用其之前的观测值预测，我们称其为“拟合值”并记作^yt|t−1，代表基于观测值y1,…,yt−1得出的预测值yt，由于经常用到，有时我们会去掉部分下标并简写成yt，拟合值总是包含到一步预测。

事实上，拟合值通常不是真正的预测值因为包含于预测方法中的任何参数都是利用时间序列中所有可得到的观测值估计出来的，甚至包括未来观测值。例如，假如我们采用平均值方法，拟合值如下给出：

yt=c该式中，c是所有可得观测值的平均值，包括那些在t时间*以后*的观测值。相似地，对于趋势法，趋势参数是用所有可得观测值估计出来的，在这种情况下。拟合值如下给出：yt=yt−1+c其中c=(yT−y1)/(T−1).两种方法中都有参数是用数据估计出来的，c上的“帽子”说明它是个估计值，当c的估计值由包含时间t以后的观测值估计出来时，拟合值不是真正的预测值。另一方面，naïve方法和季节性naïve方法不包括任何参数，因此得到的拟合值是真正的预测值。

#### 残差

时间序列模型中的残差可以理解为是在拟合模型后剩余的值。对于大多数时间序列模型（不是全部），其残差等于观测值和相对应拟合值的差。et=yt−^yt.

在检查模型是否充分地捕获数据中的信息方面，残差有着很大的用处。一个好的预测方法产生的残差具有以下特性：

1. 残差是不相关的。如果残差之间存在相关性，那么残差中包含的数据信息应该被用于计算预测值。
2. 残差的均值为零，如果残差的均值不为零，那么说明预测是有偏差的。

不满足上述性质的预测方法都应进行相应的改进，当然，这并不意味着满足上述性质的预测方法就不能改进了。对于相同的数据集，也可能存在不同的预测方法，且这些方法都满足上述性质。检查预测方法是否满足这些性质对于检查模型是否使用了所有可用的数据信息是非常重要的，但不能通过这个来选择预测方法。 如果在上述性质中有任意一条不满足，则需要改进预测模型以进行更好的预测。调整有偏差的预测较为容易：如果残差的均值为m，那么只需给所有预测值的加上m，有偏的问题就解决了。但如果是残差之间具有相关性，解决该问题会有些困难，我们将在第9章讨论这个问题。 除了上述的基本性质外，残差最好也满足下列两个性质（但不是必须满足）：

1. 残差的方差是常数
2. 残差满足正态分布

这两条性质能够让预测区间的计算变得容易（详见[3.5](https://otexts.com/fppcn/prediction-intervals.html#prediction-intervals)章节中的示例）。然而，不满足这两条性质的预测方法也不一定需要进行改进。有时，我们可以使用Box-Cox变换让模型满足这两条性质。但除此之外，几乎没有什么方法能够确保模型的残差既有常数方差，又满足正态分布。因此，我们可以使用另一种方法计算得到预测区间，我们将在后面的章节讨论该方法。

#### 示例：预测谷歌公司每日收盘价

对于股票市场的价格和指数，最好的预测方法往往是naïve方法，也就相当于，每个预测值就等于最后一个观测值，或者可以用公式表达yt=yt−1，因此，残差就等于连续观测值之间的差值：et=yt−yt=yt−yt−1.

下图显示了谷歌每日股票收盘价。由于2013年10月18日谷歌公司第三季度出人意料的优秀业绩，谷歌的股价上涨了12%，也因此在图中股价有一个大跳跃。

autoplot(goog200) + xlab("天") + ylab("收盘价（美元）") +  
 ggtitle("谷歌公司每日股价（截止至2013年12月6日）")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

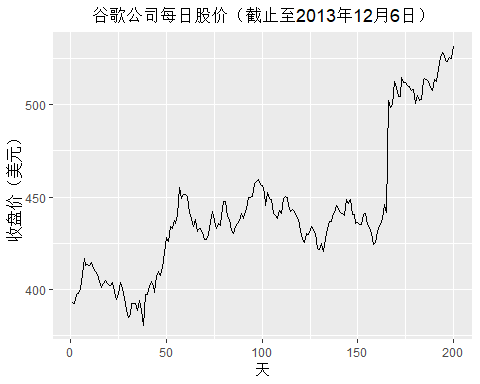


图 3.4: 谷歌公司每日股价（截止至2013年12月6日）。

使用naïve方法预测这一序列计算得到的残差图如图[3.5](https://otexts.com/fppcn/residuals.html#fig:GSPresid)所示，前一个图中出现的意外价格跳跃导致了下图中出现那个很大的正残差。

res <- residuals(naive(goog200))  
autoplot(res) + xlab("天") + ylab("") +  
 ggtitle("Naïve方法的残差")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

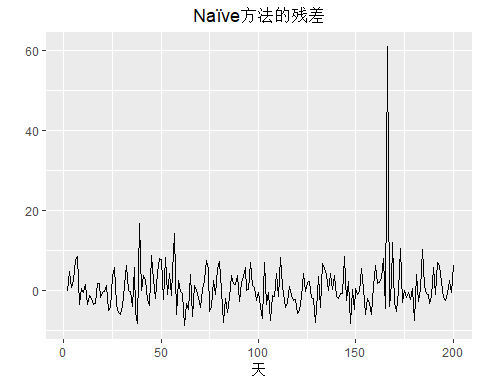


图 3.5: 使用naïve方法预测谷歌公司股价计算得到的残差直方图，对于一个正态分布来说，图中显示右尾有些长。

gghistogram(res) + ggtitle("残差直方图")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

## Warning: Removed 1 rows containing non-finite values (stat\_bin).

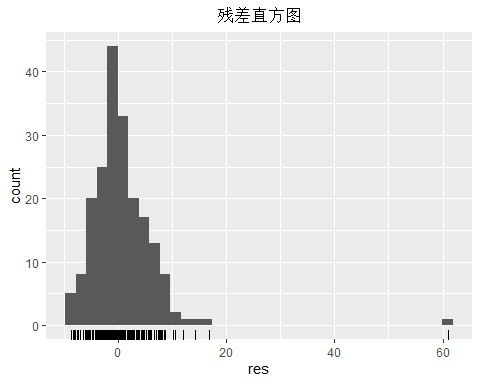


图 3.6: 使用naïve方法预测谷歌公司股价计算得到的残差直方图，对于一个正态分布来说，图中显示右尾有些长。

ggAcf(res) + ggtitle("残差ACF图")+  
 theme(text = element\_text(family = "MSYH"))+  
 theme(plot.title = element\_text(hjust = 0.5))

## Warning in grid.Call(C\_stringMetric, as.graphicsAnnot(x$label)): Windows字体数据  
## 库里没有这样的字体系列  
  
## Warning in grid.Call(C\_stringMetric, as.graphicsAnnot(x$label)): Windows字体数据  
## 库里没有这样的字体系列  
  
## Warning in grid.Call(C\_stringMetric, as.graphicsAnnot(x$label)): Windows字体数据  
## 库里没有这样的字体系列

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列  
  
## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列

## Warning in grid.Call.graphics(C\_text, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :  
## Windows字体数据库里没有这样的字体系列

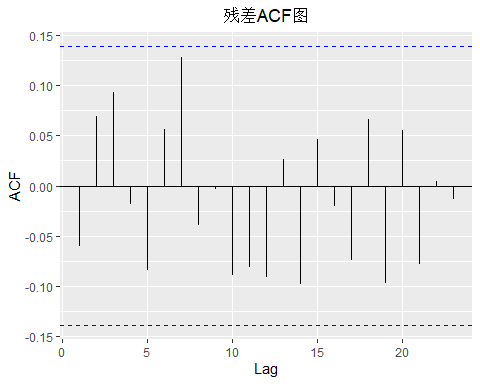


图 3.7: 使用naïve方法预测谷歌公司股价计算得到的残差ACF图，无相关性表明预测效果很好。

这些图表明使用naïve方法进行预测似乎可以解释所有可用的数据信息。残差的均值十分接近零，且残差序列之间没有显著相关性。残差的时间图表明，除去一个异常值外，残差的变化在历史数据上大致保持不变，因此可以将残差方差视为一个常数。这一结论在残差直方图中也可以得出，同时残差直方图说明残差的分布可能不符合正态分布，因为即使忽略掉异常值，残差分布的右尾也有些太长了。因此，使用这种方法进行预测可能已经相当好了，但如果假定残差符合正态分布，进而计算得到的预测区间可能有些不够准确。

#### 复合自相关检验

除了观察ACF图之外，我们也可以将一整列rk值作为一组进行更为正式的自相关检验，而不是把这些rk值单独地看待。

注意,rk是滞后于k的自相关函数。当我们在ACF图中观测是否每个尖峰都在规定的范围内时，我们已经暗含地进行了多次假设检验，每个假设被拒绝的概率很小。但实际上并不存在自相关的情况下，当进行了足够多的检验时，至少会出现一次原假设被拒绝的情况的可能性很高，我们也可能会得出结论：残差存在一些剩余自相关。

为了克服这个问题，我们检验h的自相关函数是否与预期的白噪声过程存在明显的差别，这种对于一组自相关函数的检验被称为*复合检验*，这个词来源于法语，原意是指含有很多东西的手提箱。 基于下式，类似这样的检验叫做*Box-Pierce检验*：

Q=Th∑k=1r2k,其中h是能够考虑到的最大滞后项个数，T是观测值的数量。如果每个rk的值都接近于0，那么Q值就较小；如果存在一些非常大的rk值（无论正负），那么Q指就较大。对于非周期性的数据集，我们建议取h=10；对于周期性的数据集，则取h=2m(其中m是周期）。然而，如果h的值过大，检验的效果就不够理想了。因此，如果h的值大于T/5，则取h=T/5。 一个相关的（更为精确的）检验叫做*Ljung-Box检验*，基于下式：Q∗=T(T+2)h∑k=1(T−k)−1r2k.

若Q∗的值较大，则表明自相关函数并不来自于白噪声序列 。 那么多大的Q∗值算大呢？如果一个自相关函数确实来自于白噪声序列，那么Q和Q∗均服从自由度为（h-K）的χ2分布，其中K是模型中参数的个数，如果Q和Q∗是从原始数据中计算出来的（而不是模型中的残差），则取K=0.

以谷歌公司股票价格为例，navie模型中没有参数，因此也取K=0：

# lag=h and fitdf=K  
Box.test(res, lag=10, fitdf=0)

##   
## Box-Pierce test  
##   
## data: res  
## X-squared = 10.611, df = 10, p-value = 0.3886

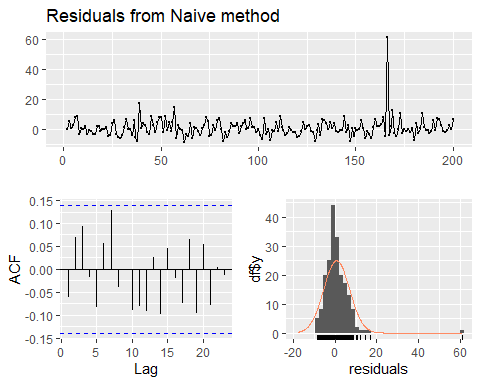
#>   
#> Box-Pierce test  
#>   
#> data: res  
#> X-squared = 11, df = 10, p-value = 0.4  
  
Box.test(res,lag=10, fitdf=0, type="Lj")

##   
## Box-Ljung test  
##   
## data: res  
## X-squared = 11.031, df = 10, p-value = 0.3551

#>   
#> Box-Ljung test  
#>   
#> data: res  
#> X-squared = 11, df = 10, p-value = 0.4

对于Q和Q∗，结果都是无意义的（即p-values非常大）。因此，我们可以得出结论：残差与白噪声序列不可区分。 所有这些检验残差的方法，在R中都被打包成一个函数，这个函数能够产生时间图、ACF图和残差直方图（与叠加正态分布进行对比），还能够以正确的自由度进行Ljung-Box检验：

checkresiduals(naive(goog200))



##   
## Ljung-Box test  
##   
## data: Residuals from Naive method  
## Q\* = 11.031, df = 10, p-value = 0.3551  
##   
## Model df: 0. Total lags used: 10

#>   
#> Ljung-Box test  
#>   
#> data: Residuals from Naive method  
#> Q\* = 11, df = 10, p-value = 0.4  
#>   
#> Model df: 0. Total lags used: 10

### 3.3残差诊断

#### 拟合值

时间序列中的每个观测值都可以用其之前的观测值预测，我们称其为“拟合值”并记作^yt|t−1，代表基于观测值y1,…,yt−1得出的预测值yt，由于经常用到，有时我们会去掉部分下标并简写成yt，拟合值总是包含到一步预测。

事实上，拟合值通常不是真正的预测值因为包含于预测方法中的任何参数都是利用时间序列中所有可得到的观测值估计出来的，甚至包括未来观测值。例如，假如我们采用平均值方法，拟合值如下给出：

yt=c该式中，c是所有可得观测值的平均值，包括那些在t时间*以后*的观测值。相似地，对于趋势法，趋势参数是用所有可得观测值估计出来的，在这种情况下。拟合值如下给出：yt=yt−1+c其中c=(yT−y1)/(T−1).两种方法中都有参数是用数据估计出来的，c上的“帽子”说明它是个估计值，当c的估计值由包含时间t以后的观测值估计出来时，拟合值不是真正的预测值。另一方面，naïve方法和季节性naïve方法不包括任何参数，因此得到的拟合值是真正的预测值。

#### 残差

时间序列模型中的残差可以理解为是在拟合模型后剩余的值。对于大多数时间序列模型（不是全部），其残差等于观测值和相对应拟合值的差。et=yt−^yt.

在检查模型是否充分地捕获数据中的信息方面，残差有着很大的用处。一个好的预测方法产生的残差具有以下特性：

1. 残差是不相关的。如果残差之间存在相关性，那么残差中包含的数据信息应该被用于计算预测值。
2. 残差的均值为零，如果残差的均值不为零，那么说明预测是有偏差的。

不满足上述性质的预测方法都应进行相应的改进，当然，这并不意味着满足上述性质的预测方法就不能改进了。对于相同的数据集，也可能存在不同的预测方法，且这些方法都满足上述性质。检查预测方法是否满足这些性质对于检查模型是否使用了所有可用的数据信息是非常重要的，但不能通过这个来选择预测方法。 如果在上述性质中有任意一条不满足，则需要改进预测模型以进行更好的预测。调整有偏差的预测较为容易：如果残差的均值为m，那么只需给所有预测值的加上m，有偏的问题就解决了。但如果是残差之间具有相关性，解决该问题会有些困难，我们将在第9章讨论这个问题。 除了上述的基本性质外，残差最好也满足下列两个性质（但不是必须满足）：

1. 残差的方差是常数
2. 残差满足正态分布

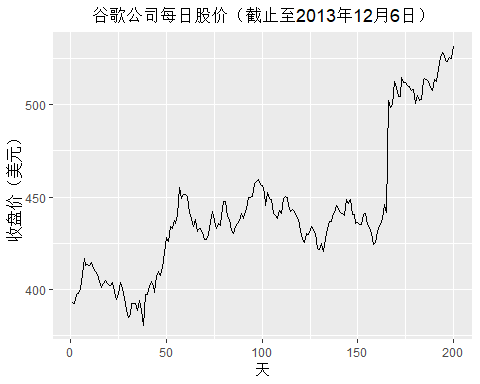
这两条性质能够让预测区间的计算变得容易（详见[3.5](https://otexts.com/fppcn/prediction-intervals.html#prediction-intervals)章节中的示例）。然而，不满足这两条性质的预测方法也不一定需要进行改进。有时，我们可以使用Box-Cox变换让模型满足这两条性质。但除此之外，几乎没有什么方法能够确保模型的残差既有常数方差，又满足正态分布。因此，我们可以使用另一种方法计算得到预测区间，我们将在后面的章节讨论该方法。

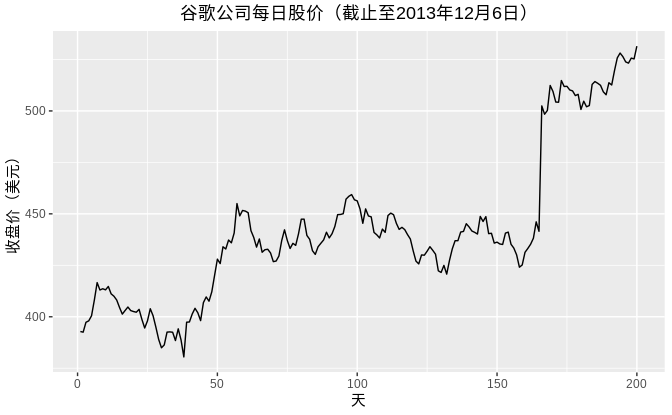
#### 示例：预测谷歌公司每日收盘价

对于股票市场的价格和指数，最好的预测方法往往是naïve方法，也就相当于，每个预测值就等于最后一个观测值，或者可以用公式表达yt=yt−1，因此，残差就等于连续观测值之间的差值：et=yt−yt=yt−yt−1.

下图显示了谷歌每日股票收盘价。由于2013年10月18日谷歌公司第三季度出人意料的优秀业绩，谷歌的股价上涨了12%，也因此在图中股价有一个大跳跃。

autoplot(goog200) + xlab("天") + ylab("收盘价（美元）") +  
 ggtitle("谷歌公司每日股价（截止至2013年12月6日）")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))



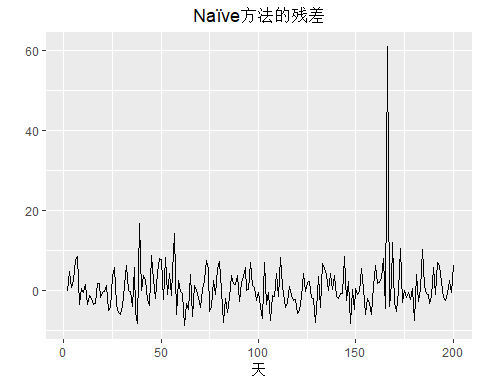


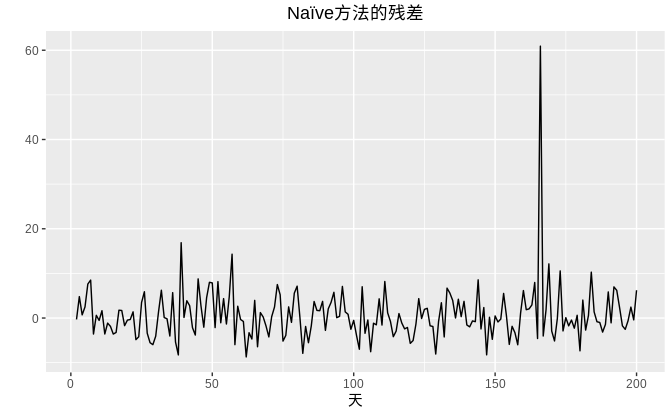
谷歌公司每日股价（截止至2013年12月6日）。

图 3.4: 谷歌公司每日股价（截止至2013年12月6日）。

使用naïve方法预测这一序列计算得到的残差图如图[3.5](https://otexts.com/fppcn/residuals.html#fig:GSPresid)所示，前一个图中出现的意外价格跳跃导致了下图中出现那个很大的正残差。

res <- residuals(naive(goog200))  
autoplot(res) + xlab("天") + ylab("") +  
 ggtitle("Naïve方法的残差")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))



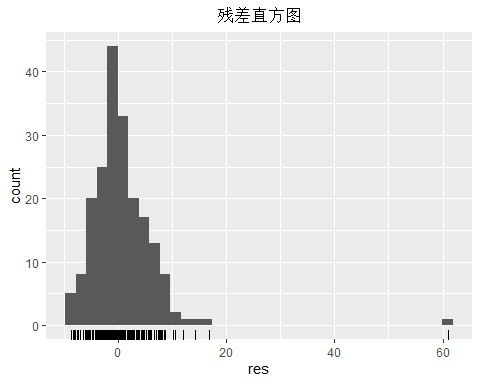


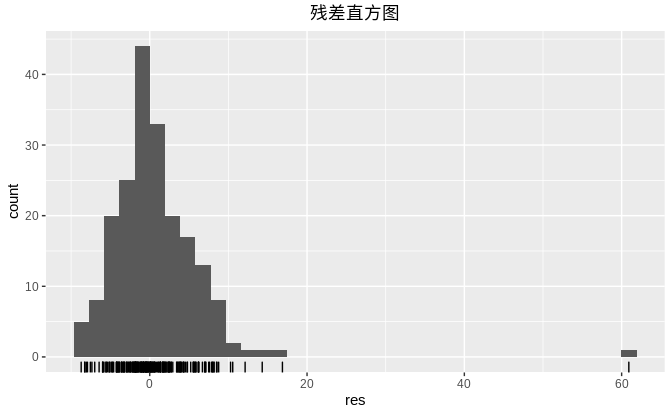
使用naïve方法预测谷歌公司股价计算得到的残差直方图，对于一个正态分布来说，图中显示右尾有些长。

图 3.5: 使用naïve方法预测谷歌公司股价计算得到的残差直方图，对于一个正态分布来说，图中显示右尾有些长。

gghistogram(res) + ggtitle("残差直方图")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

## Warning: Removed 1 rows containing non-finite values (stat\_bin).

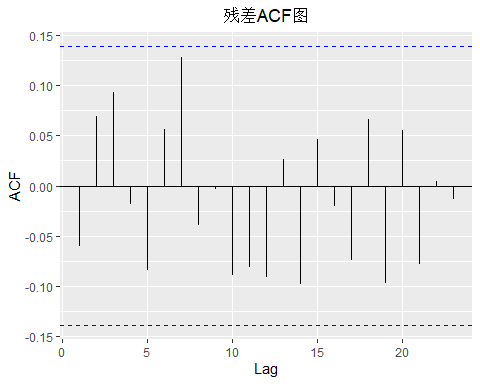


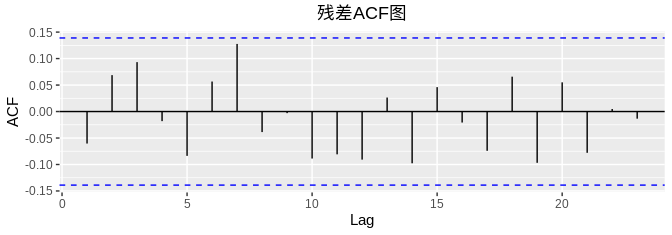


使用naïve方法预测谷歌公司股价计算得到的残差直方图，对于一个正态分布来说，图中显示右尾有些长。

图 3.6: 使用naïve方法预测谷歌公司股价计算得到的残差直方图，对于一个正态分布来说，图中显示右尾有些长。

ggAcf(res) + ggtitle("残差ACF图")+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))





使用naïve方法预测谷歌公司股价计算得到的残差ACF图，无相关性表明预测效果很好。

图 3.7: 使用naïve方法预测谷歌公司股价计算得到的残差ACF图，无相关性表明预测效果很好。

这些图表明使用naïve方法进行预测似乎可以解释所有可用的数据信息。残差的均值十分接近零，且残差序列之间没有显著相关性。残差的时间图表明，除去一个异常值外，残差的变化在历史数据上大致保持不变，因此可以将残差方差视为一个常数。这一结论在残差直方图中也可以得出，同时残差直方图说明残差的分布可能不符合正态分布，因为即使忽略掉异常值，残差分布的右尾也有些太长了。因此，使用这种方法进行预测可能已经相当好了，但如果假定残差符合正态分布，进而计算得到的预测区间可能有些不够准确。

#### 复合自相关检验

除了观察ACF图之外，我们也可以将一整列rk值作为一组进行更为正式的自相关检验，而不是把这些rk值单独地看待。

注意,rk是滞后于k的自相关函数。当我们在ACF图中观测是否每个尖峰都在规定的范围内时，我们已经暗含地进行了多次假设检验，每个假设被拒绝的概率很小。但实际上并不存在自相关的情况下，当进行了足够多的检验时，至少会出现一次原假设被拒绝的情况的可能性很高，我们也可能会得出结论：残差存在一些剩余自相关。

为了克服这个问题，我们检验h的自相关函数是否与预期的白噪声过程存在明显的差别，这种对于一组自相关函数的检验被称为*复合检验*，这个词来源于法语，原意是指含有很多东西的手提箱。 基于下式，类似这样的检验叫做*Box-Pierce检验*：

Q=Th∑k=1r2k,其中h是能够考虑到的最大滞后项个数，T是观测值的数量。如果每个rk的值都接近于0，那么Q值就较小；如果存在一些非常大的rk值（无论正负），那么Q指就较大。对于非周期性的数据集，我们建议取h=10；对于周期性的数据集，则取h=2m(其中m是周期）。然而，如果h的值过大，检验的效果就不够理想了。因此，如果h的值大于T/5，则取h=T/5。 一个相关的（更为精确的）检验叫做*Ljung-Box检验*，基于下式：Q∗=T(T+2)h∑k=1(T−k)−1r2k.

若Q∗的值较大，则表明自相关函数并不来自于白噪声序列 。 那么多大的Q∗值算大呢？如果一个自相关函数确实来自于白噪声序列，那么Q和Q∗均服从自由度为（h-K）的χ2分布，其中K是模型中参数的个数，如果Q和Q∗是从原始数据中计算出来的（而不是模型中的残差），则取K=0.

以谷歌公司股票价格为例，navie模型中没有参数，因此也取K=0：

# lag=h and fitdf=K  
Box.test(res, lag=10, fitdf=0)

##   
## Box-Pierce test  
##   
## data: res  
## X-squared = 10.611, df = 10, p-value = 0.3886

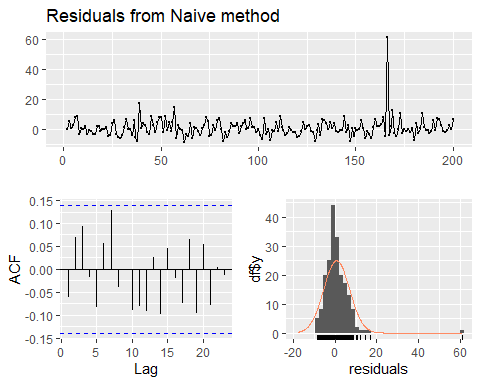
#>   
#> Box-Pierce test  
#>   
#> data: res  
#> X-squared = 11, df = 10, p-value = 0.4  
  
Box.test(res,lag=10, fitdf=0, type="Lj")

##   
## Box-Ljung test  
##   
## data: res  
## X-squared = 11.031, df = 10, p-value = 0.3551

#>   
#> Box-Ljung test  
#>   
#> data: res  
#> X-squared = 11, df = 10, p-value = 0.4

对于Q和Q∗，结果都是无意义的（即p-values非常大）。因此，我们可以得出结论：残差与白噪声序列不可区分。 所有这些检验残差的方法，在R中都被打包成一个函数，这个函数能够产生时间图、ACF图和残差直方图（与叠加正态分布进行对比），还能够以正确的自由度进行Ljung-Box检验：

checkresiduals(naive(goog200))



##   
## Ljung-Box test  
##   
## data: Residuals from Naive method  
## Q\* = 11.031, df = 10, p-value = 0.3551  
##   
## Model df: 0. Total lags used: 10

### 3.4评估预测精度

#### 训练集和数据集

使用真实的预测来评估预测精度是很重要的。因此，残差的大小并不能很好地表示真实预测误差的大小。而评估模型预测精度只能通过模型在新数据中的预测效果决定。

在选择模型时，通常我们会将可用数据分成训练数据集和测试数据集两部分，其中训练数据集用于估计预测方法中的任意参数，而测试数据集用于评估预测精度。由于训练数据集并没有用于确定预测模型，因此它能可靠地检验模型对于新数据的预测准确性。



测试数据集的大小取决于总样本集的大小和所希望的预测的距离，通常为总样本集的20%。理想状况下，测试数据集至少和最大预测区间一样长。读者应注意以下几点：

* 一个适合训练数据集的预测模型不一定能完美预测其他数据集。
* 使用足够多参数的模型总能得到完美的拟合。
* 对数据的过度拟合与未能识别数据的系统模式都是不可取的。

一些参考资料会把测试数据集叫做“保留集”，因为这部分数据是为了更好地拟合模型而被保留下来用于评估预测精度的。也有的参考书把测试数据集叫做“样本内数据”，把训练数据集叫做“样本外数据”，本书则更习惯使用测试数据集和训练数据集。

#### 时间序列子集函数

在我们需要提取一个时间序列中的一部分时，第2章中所介绍的window()函数将非常有用，例如我们在产生训练数据集和测试数据集时，我们就需要使用window()函数。在这个函数中，我们需要使用时间值来指定所需时间序列部分的开始和结束。下面是代码示例：

window(ausbeer, start=1995)

## Qtr1 Qtr2 Qtr3 Qtr4  
## 1995 426 408 416 520  
## 1996 409 398 398 507  
## 1997 432 398 406 526  
## 1998 428 397 403 517  
## 1999 435 383 424 521  
## 2000 421 402 414 500  
## 2001 451 380 416 492  
## 2002 428 408 406 506  
## 2003 435 380 421 490  
## 2004 435 390 412 454  
## 2005 416 403 408 482  
## 2006 438 386 405 491  
## 2007 427 383 394 473  
## 2008 420 390 410 488  
## 2009 415 398 419 488  
## 2010 414 374

上述代码表示抓取从1995年开始的全部数据。

另一个非常有用的函数是subset()函数，它可以允许构造更多类型的子集。这个函数最大的优点是它允许使用索引来选择子集。下面是代码示例：

subset(ausbeer, start=length(ausbeer)-4\*5)

## Qtr1 Qtr2 Qtr3 Qtr4  
## 2005 403 408 482  
## 2006 438 386 405 491  
## 2007 427 383 394 473  
## 2008 420 390 410 488  
## 2009 415 398 419 488  
## 2010 414 374

上述代码表示从ausbeer这个数据集中提取过去5年的观测值。除此之外，subset()函数还允许提取特定时间的所有观测，下面是代码示例：

subset(ausbeer, quarter = 1)

## Time Series:  
## Start = 1956   
## End = 2010   
## Frequency = 1   
## [1] 284 262 272 261 286 295 279 294 313 331 335 353 393 383 387 410 419 458 465  
## [20] 500 510 486 515 503 513 548 493 475 453 464 459 481 474 467 485 464 443 433  
## [39] 449 426 409 432 428 435 421 451 428 435 435 416 438 427 420 415 414

上述代码表示取出所有年份第一季度的数据

最后，head()函数和tail()函数主要用于取出前几个或后几个观测。例如，上文中的从ausbeer这个数据集中取出过去5年的观测也可以使用以下代码实现：

tail(ausbeer, 4\*5)

## Qtr1 Qtr2 Qtr3 Qtr4  
## 2005 408 482  
## 2006 438 386 405 491  
## 2007 427 383 394 473  
## 2008 420 390 410 488  
## 2009 415 398 419 488  
## 2010 414 374

#### 预测误差

预测误差是指实际值与预测值的差，这里的误差并非意味着错误，它指的是观测中不可预知的部分，可以如下表示：eT+h=yT+h−^yT+h|T,用{y1,…,yT}表示训练集，用{yT+1,yT+2,…}表示测试集。

注意，预测误差与残差的区别有两点：第一，残差是在训练数据集上计算得到的，而预测误差是在测试数据集上计算得到的；第二，残差仅仅是基于一步预测得到的，而预测误差可能是涉及多步预测。

我们可以使用不同的方式去总结预测误差，进而衡量预测的准确性。

#### 尺度效应误差

预测误差与数据的标度相同，因此，仅基于et的预测精度测量是依赖标度的，不能用于比较包含不同单位的时间序列。

测量尺度效应（依赖于标度）最常使用的两种方法都是基于绝对误差和平方误差的。平均绝对误差(Mean absolute error: MAE)=mean(|et|),均方根误差(Root mean squared error: RMSE)=√mean(e2t).当我们将这两种预测方法应用于单个时间序列或者多个具有相同单位的时间序列时，通过对比，我们可以发现，MAE方法更流行，因为这种方法不仅容易被理解，而且便于计算。使MAE最小化的预测方法的预测结果是预测分布的中位数，而使RMSE最小化的预测方法的预测结果是预测分布的平均值。因此，RMSE方法尽管在解释方面有些困难，但使用范围非常广。

#### 百分比误差

百分比误差的计算公式是pt=100et/yt。百分误差最大的优点是无单位，因此常常被用来比较不同数据集之间的预测性能。最常用的预测方法是：均方绝对百分比误差(Mean absolute percentage error: MAPE)=mean(|pt|).基于百分误差的计算方法有两个缺点：第一，如果在我们感兴趣的时间段内某个t使得yt=0，则计算结果是无穷或者是无意义的；第二，如果任何yt值趋近于0，则计算结果有极值。百分误差还有一个常常被忽视的缺点是其假定度量单位是一个有意义的零。[2](https://otexts.com/fppcn/accuracy.html#fn2)例如，在测量华氏温度或者摄氏温度的温度预测准确性时，百分误差是没有意义的，因为温度具有任意的零点。

这种测量方法还有一个缺点，就是其对负的误差的惩罚比对正的误差要重，因此Armstrong ([1978, p. 348](https://otexts.com/fppcn/accuracy.html#ref-Armstrong85))提出了对称MAPE法（简称sMAPE法）对这个缺点进行了改进，还曾在M3的预测竞赛中使用了这种方法，其计算公式是：sMAPE=mean(200|yt−yt|/(yt+yt)).然而，如果yt值趋近于0，则^yt的预测值也可能趋近于0.因此，这种计算方法还是有可能除以一个趋近于0的数，让计算结果不够稳定。除此之外，sMAPE的值也可能是负数，因此，这种衡量方法并不是一个真正意义上能够衡量“绝对百分误差”的方法。

Hyndman & Koehler ([2006](https://otexts.com/fppcn/accuracy.html#ref-HK06))在2006年建议不要使用sMAPE方法，本书之所以讲解这种方法是因为这种方法被广泛使用着，但在本书中，我们不会使用这种方法。

#### 比例误差（scaled errors）

Hyndman & Koehler ([2006](https://otexts.com/fppcn/accuracy.html#ref-HK06))在2006年提出了比例误差的概念，用于替代百分误差去衡量具有不同单位的时间序列的预测准确性。他们的方法思路是通过训练来自简单预测方法的MAE去修正误差。

对于非周期性时间序列，可以使用下面的naïve方法来定义比例误差：qj=ej1T−1T∑t=2|yt−yt−1|.因为上式中的分子和分母都包含了在原始数据标度上的值，而qj是一个与数据标度无关的值。如果选用一个比平均naïve预测方法更好的预测方法，那么比例误差会小于1；相对地，如果选用了比平均naïve预测方法更差的预测方法，那么比例误差会大于1。

对于季节性时间序列，可以使用下面的周期性naïve方法来定义比例误差：qj=ej1T−mT∑t=m+1|yt−yt−m|.

*平均绝对比例误差*如下表示：MASE=mean(|qj|).

#### 例子

beer2 <- window(ausbeer,start=1992,end=c(2007,4))  
beerfit1 <- meanf(beer2,h=10)  
beerfit2 <- rwf(beer2,h=10)  
beerfit3 <- snaive(beer2,h=10)  
autoplot(window(ausbeer, start=1992)) +  
 autolayer(beerfit1$mean, series="均值") +  
 autolayer(beerfit2$mean, series="Naïve") +  
 autolayer(beerfit3$mean, series="季节性naïve") +  
 xlab("年份") + ylab("百万升") +  
 ggtitle("啤酒季度产量预测") +  
 guides(colour=guide\_legend(title="预测")) +  
 theme(text = element\_text()) +  
 theme(plot.title = element\_text(hjust = 0.5))

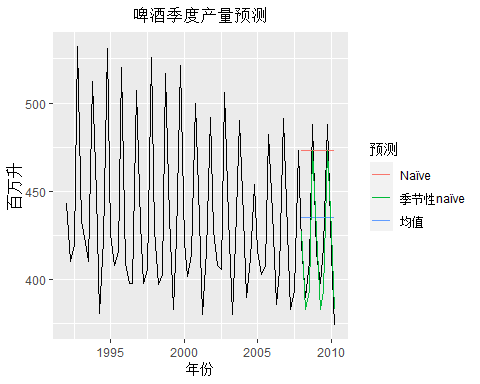


图 3.8: 利用2007年年底相关数据预测澳大利亚啤酒季度产量。

图[3.8](https://otexts.com/fppcn/accuracy.html#fig:beeraccuracy)显示的是使用2007年年底相关数据预测澳大利亚啤酒季度产量的三种方法。还显示了2008至2010期间的实际产量值。可以看出，我们计算出了这一时期的预测准确度。

beer3 <- window(ausbeer, start=2008)  
accuracy(beerfit1, beer3)

## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 0.000 43.62858 35.23438 -0.9365102 7.886776 2.463942 -0.10915105  
## Test set -13.775 38.44724 34.82500 -3.9698659 8.283390 2.435315 -0.06905715  
## Theil's U  
## Training set NA  
## Test set 0.801254

accuracy(beerfit2, beer3)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.4761905 65.31511 54.73016 -0.9162496 12.16415 3.827284  
## Test set -51.4000000 62.69290 57.40000 -12.9549160 14.18442 4.013986  
## ACF1 Theil's U  
## Training set -0.24098292 NA  
## Test set -0.06905715 1.254009

accuracy(beerfit3, beer3)

## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -2.133333 16.78193 14.3 -0.5537713 3.313685 1.0000000 -0.2876333  
## Test set 5.200000 14.31084 13.4 1.1475536 3.168503 0.9370629 0.1318407  
## Theil's U  
## Training set NA  
## Test set 0.298728

|  | RMSE | MAE | MAPE | MASE |
| --- | --- | --- | --- | --- |
| 平均值法 | 38.45 | 34.83 | 8.28 | 2.44 |
| Naïve方法 | 62.69 | 57.40 | 14.18 | 4.01 |
| 季节性naïve方法 | 14.31 | 13.40 | 3.17 | 0.94 |

由图可知，对于这些数据而言，周期性naïve方法是最好的；但之后我们就会发现，这种方法仍然能够被改进。有时，当我们需要去选择其中最佳的预测方法时，发现不同的精确度测量方法会产生不同的结果.但在这个例子中，所有的结果显示季节性naïve方法是三种方法中最好的。

下面举一个非季节性数据的例子，考虑谷歌的股票价格。下图是截止2013年12月6日之前的观测数据，并用三种方法对之后40天股市收盘价进行预测的结果。

googfc1 <- meanf(goog200, h=40)  
googfc2 <- rwf(goog200, h=40)  
googfc3 <- rwf(goog200, drift=TRUE, h=40)  
autoplot(subset(goog, end = 240)) +  
 autolayer(googfc1, PI=FALSE, series="均值") +  
 autolayer(googfc2, PI=FALSE, series="Naïve") +  
 autolayer(googfc3, PI=FALSE, series="漂移") +  
 xlab("天") + ylab("收盘价（美元）") +  
 ggtitle("谷歌公司每日股价（截止至2013年12月6日）") +  
 guides(colour=guide\_legend(title="预测"))+  
 theme(text = element\_text())+  
 theme(plot.title = element\_text(hjust = 0.5))

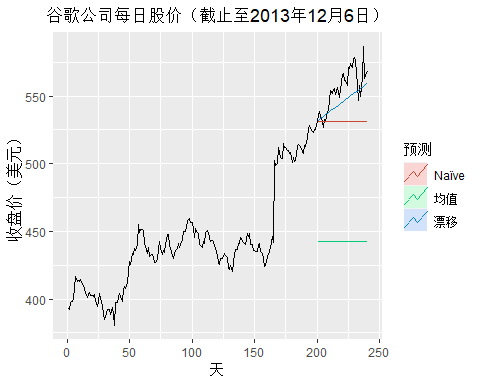


图 3.9: 从2013年12月7日开始对于谷歌股价的预测。

googtest <- window(goog, start=201, end=240)  
accuracy(googfc1, googtest)

## ME RMSE MAE MPE MAPE MASE  
## Training set -4.296286e-15 36.91961 26.86941 -0.6596884 5.95376 7.182995  
## Test set 1.132697e+02 114.21375 113.26971 20.3222979 20.32230 30.280376  
## ACF1 Theil's U  
## Training set 0.9668981 NA  
## Test set 0.8104340 13.92142

accuracy(googfc2, googtest)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.6967249 6.208148 3.740697 0.1426616 0.8437137 1.000000  
## Test set 24.3677328 28.434837 24.593517 4.3171356 4.3599811 6.574582  
## ACF1 Theil's U  
## Training set -0.06038617 NA  
## Test set 0.81043397 3.451903

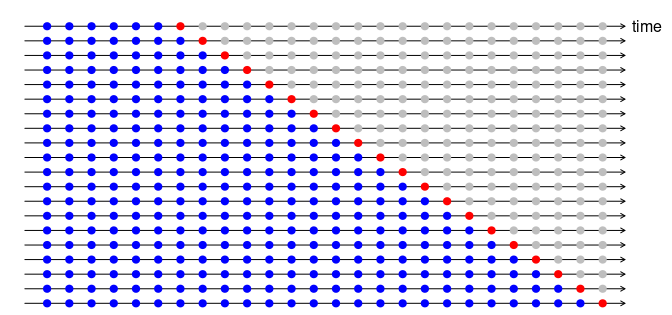
accuracy(googfc3, googtest)

## ME RMSE MAE MPE MAPE MASE  
## Training set -5.998536e-15 6.168928 3.824406 -0.01570676 0.8630093 1.022378  
## Test set 1.008487e+01 14.077291 11.667241 1.77566103 2.0700918 3.119002  
## ACF1 Theil's U  
## Training set -0.06038617 NA  
## Test set 0.64732736 1.709275

|  | RMSE | MAE | MAPE | MASE |
| --- | --- | --- | --- | --- |
| 均值法 | 114.21 | 113.27 | 20.32 | 30.28 |
| Naïve方法 | 28.43 | 24.59 | 4.36 | 6.57 |
| 漂移法 | 14.08 | 11.67 | 2.07 | 3.12 |

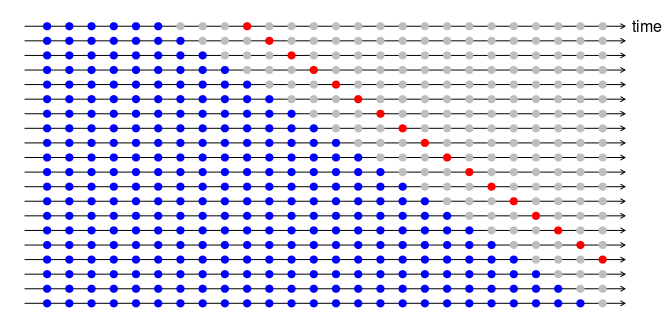
在此例中，最好的方法是趋势法。（不管采用哪种精度度量）

#### 时间序列交叉验证

更加复杂的训练数据集和测试数据集叫做时间序列交叉验证。在这个过程中，有一系列的测试数据集，每个测试数据集都由一个单独的观测值组成。相对应的训练测试集仅包含组成测试数据集的观测值之前发生的观测值。因此，没有未来的观测值能够被用来构建预测。由于在一个较小的训练数据集中得到可靠的预测是不可能的，所以最早的观测值不视为测试数据集。 下面的图演示了一系列训练数据集和测试数据集，其中蓝色的观测点组成训练数据集，红色的观测点组成测试数据集。

预测精度是通过测试数据集中观测值的平均值得到的，这个过程有时被称为“评估滚动原点预测”，因为预测所依据的“原点”是及时向前推进的。

在时间序列预测中，一步预测可能不如多步预测相关性那么高，在这种情况下，基于滚动预测原点的交叉验证过程可以通过使用多步误差进行改进。假设我们对于能够尝试良好四步预测的模型感兴趣，那么相应的图如下所示：



时间序列的交叉验证可以使用tsCV()函数来实现，在下面的代码示例中，我们比较了通过时间序列的交叉验证过程得到的RMSE和通过残差得到的RMSE.

e <- tsCV(goog200, rwf, drift=TRUE, h=1)  
sqrt(mean(e^2, na.rm=TRUE))

## [1] 6.233245

#> [1] 6.233  
sqrt(mean(residuals(rwf(goog200, drift=TRUE))^2, na.rm=TRUE))

## [1] 6.168928

#> [1] 6.169

正如所预期的那样，通过残差计算得到的RMSE较小，说明相应的“预测”是基于拟合到整个数据集的模型的拟合值，并非真正的预测值。 选择最佳预测模型的一个好方法是选择用时间序列交叉过程计算得出RMSE最小的模型。

#### 管道（Pipe）操作

上述的R代码有些难看，所以这里我们介绍一些把R函数串在一起的其他方法。在上述代码中，我们在函数中的函数嵌套函数，所以你必须从内到外的读取代码，这使得我们很难理解正在计算的到底是什么。因此我们可以使用管道操作符%>%进行替代，代码如下：

goog200 %>% tsCV(forecastfunction=rwf, drift=TRUE, h=1) -> e  
e^2 %>% mean(na.rm=TRUE) %>% sqrt()

## [1] 6.233245

#> [1] 6.233  
goog200 %>% rwf(drift=TRUE) %>% residuals() -> res  
res^2 %>% mean(na.rm=TRUE) %>% sqrt()

## [1] 6.168928

#> [1] 6.169

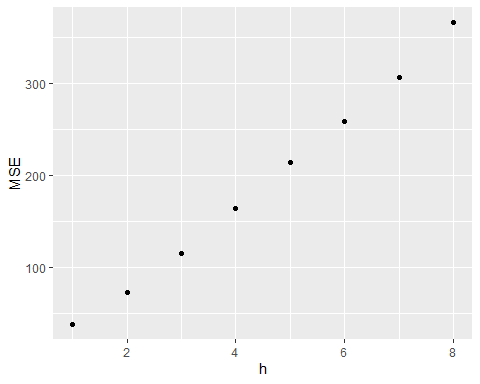
每个管道的左边作为第一个参数传递给右边的函数。这与我们读英语从左到右读的习惯相一致。使用管道的一个重要前提是必须指定所有其他参数，同时这也有助于提高可读性。当使用管道时，使用右箭头赋值->而不是使用左箭头赋值是很自然的。例如，上述代码的第三行可以被读作把goog200这个参数，以drift=TRUE这个条件传递给rwf()函数，计算得到的残差结果，存储为res。

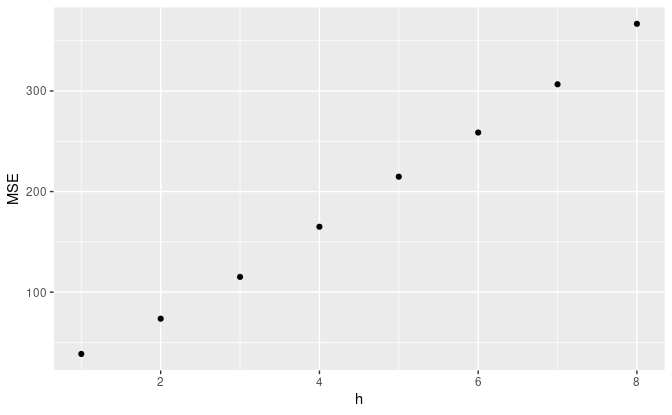
在本书的其余部分，我们将使用管道操作符让代码变得易于阅读。为了保持整体一致，即使函数没有参数，我们也将始终使用括号来区分函数与其他对象。正如上面我们所看到的，我们对sqrt()函数和residuals()函数的表示。

#### 示例：使用tsCV()函数

绘制在图[3.4](https://otexts.com/fppcn/residuals.html#fig:GSPautoplot)中的goog200数据包括了在纳斯达克交易所从2013年2月25日开始连续200个交易日的谷歌公司每日股市收盘价。 下面的代码将使用tsCV()函数来评估用naïve方法进行1步到8步预测的预测性能，使用MSE来衡量预测误差。根据我们的预期，这个图表明预测误差会随着预测区间的增大而增大。

e <- tsCV(goog200, forecastfunction=naive, h=8)  
# Compute the MSE values and remove missing values  
mse <- colMeans(e^2, na.rm = T)  
# Plot the MSE values against the forecast horizon  
data.frame(h = 1:8, MSE = mse) %>%  
 ggplot(aes(x = h, y = MSE)) + geom\_point()





#### 参考文献

Armstrong, J Scott. 1985. Long-Range Forecasting: From Crystal Ball to Computer. Wiley.

Hyndman, Rob J, and Anne B Koehler. 2006. “Another Look at Measures of Forecast Accuracy.” International Journal of Forecasting 22:679–88.

#### 参考文献

Armstrong, J. S. (1978).*Long-range forecasting: From crystal ball to computer*. John Wiley & Sons.[[Amazon]](http://buy.geni.us/Proxy.ashx?TSID=140570&GR_URL=http%3A%2F%2Fwww.amazon.com%2Fdp%2F0471030023)

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy.*International Journal of Forecasting*,*22*(4), 679–688.[[DOI]](https://doi.org/10.1016/j.ijforecast.2006.03.001)

1. 也就是说，百分比在定比型数据上是有意义的，但在定距型数据上是不成立的。因为只有定比型数据具有有意义的零。[↩︎](https://otexts.com/fppcn/accuracy.html#fnref2)

#>   
#> Ljung-Box test  
#>   
#> data: Residuals from Naive method  
#> Q\* = 11, df = 10, p-value = 0.4  
#>   
#> Model df: 0. Total lags used: 10

### 3.5预测区间

正如在[1.7](https://otexts.com/fppcn/perspective.html#perspective)中所讨论的，预测区间给出了一定置信度下的置信区。例如，假设预测误差为正态分布，则置信度为95%的h-step预测预测区间yT+h|T±1.96σh,其中，σh是h步预测标准差的预测分布的估计值，更一般地，可以将预测区间写为yT+h|T±k^σh其中，乘子k取决于置信概率。在这本书中，我们通常计算置信度为80%和95%的置信区间，尽管任意百分比都有可能出现。下表给出了在不同正态分布预测误差范围（置信度）内对应乘子k值。

| |  
— | — |表 3.1:预测区间中所用的乘子取值  
百分比 | 乘子 |  
— | — |  
50 | 0.67 |  
55 | 0.76 |  
60 | 0.84 |  
65 | 0.93 |  
70 | 1.04 |  
75 | 1.15 |  
80 | 1.28 |  
85 | 1.44 |  
90 | 1.64 |  
95 | 1.96 |  
96 | 2.05 |  
97 | 2.17 |  
98 | 2.33 |  
99 | 2.58 |

预测区间的值表示的是预测中的不确定性。如果我们只做点预测，就无法得到这些预测的准确程度。然而，如果我们同时得到预测间隔，那么就会很清楚每个预测有多大的不确定性。由于这个原因，如果没有后续的预测区间，点预测几乎没有价值。

#### 一步预测区间

在进行向前一步预测时，预测分布的标准差与残差的标准差几乎相同。（事实上，如果没有需要被估计的参数，这两个标准差是相同的，比如 naïve 方法。对于那些有需要估计参数的预测方法，预测分布的标准差略大于残差标准差，尽管这种差异经常被忽略。）

例如，我们参照一个用naïve预测方法的例子，谷歌的股票价格数据（GSP）goog200（图[3.4](https://otexts.com/fppcn/residuals.html#fig:GSPautoplot)）。所观察到的序列的最后一个值是531.48，因此对GSP的下一个值的预测是531.48。通过naïve方法计算的残差标准差为6.21，因此，GSP下一个值的置信度为95%的预测区间是531.48±1.96(6.21)=[519.3,543.6].类似地，置信度80%的预测区间是：531.48±1.28(6.21)=[523.5,539.4].

对应乘数的值 (1.96 or 1.28) 可以取自表[3.1](https://otexts.com/fppcn/prediction-intervals.html#tab:pcmultipliers).

#### 多步预测区间

预测间隔的一个共同特点是随着预测时间的增加它们的长度也在增加。预测得越远，预测的不确定性越大，预测区间也越宽。也就是说,σh通常随步数h的增加而增加（虽然有一些非线性预测方法不是这样）。

得到一个预测区间之前，必须先得到σh的估计值。正如已经指出的，一步预测(h=1)对应的残差标准差可以很好地估计预测标准差σ1。对于多步预测，则需要一种更复杂的计算方法。注意，这些计算都是假定以上残差不相关。

#### 基准方法

对于四种基准方法，在残差不相关的前提下，通过数学推导得出预测标准差是可能的。如果令σh表示h步预测的标准差，并令σ为残差标准差，那么我们可以用下面的表达式：

**均值预测**a\_h = $

**Naïve 预测**:σh=σ√h

**季节性naïve预测**σh=σ√⌊(h−1)/m⌋+1.

**漂移预测**:σh=σ√h(1+h/T).

注意，当h=1且T很大时，这些方法都可以计算出相同的近似值^σ。

不论使用何种基准预测方法，都能计算出预测区间。例如，这是通过naïve方法预测谷歌股票价格（GSP）的结果：

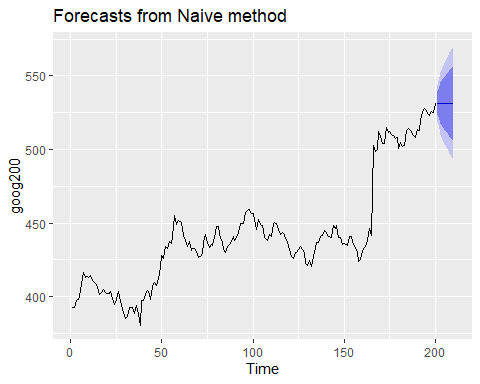
naive(goog200)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 201 531.4783 523.5222 539.4343 519.3105 543.6460  
## 202 531.4783 520.2267 542.7298 514.2705 548.6861  
## 203 531.4783 517.6980 545.2586 510.4031 552.5534  
## 204 531.4783 515.5661 547.3904 507.1428 555.8138  
## 205 531.4783 513.6880 549.2686 504.2704 558.6862  
## 206 531.4783 511.9900 550.9666 501.6735 561.2830  
## 207 531.4783 510.4285 552.5280 499.2854 563.6711  
## 208 531.4783 508.9751 553.9814 497.0627 565.8939  
## 209 531.4783 507.6101 555.3465 494.9750 567.9815  
## 210 531.4783 506.3190 556.6375 493.0005 569.9561

#> Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
#> 201 531.5 523.5 539.4 519.3 543.6  
#> 202 531.5 520.2 542.7 514.3 548.7  
#> 203 531.5 517.7 545.3 510.4 552.6  
#> 204 531.5 515.6 547.4 507.1 555.8  
#> 205 531.5 513.7 549.3 504.3 558.7  
#> 206 531.5 512.0 551.0 501.7 561.3  
#> 207 531.5 510.4 552.5 499.3 563.7  
#> 208 531.5 509.0 554.0 497.1 565.9  
#> 209 531.5 507.6 555.3 495.0 568.0  
#> 210 531.5 506.3 556.6 493.0 570.0

当绘制图像时，用阴影区域表示预测区间，颜色的深度表示与该区间相关的概率。

autoplot(naive(goog200))



#### 从自助（bootstrap）残差得到的预测区间

当预测误差服从正态分布是一个不合理的假定时，可以使用自举法（它只假定预测误差都是不相关的）。

预测误差被定义为et=yt−yt|t−1。我们可以重新写成：yt=yt|t−1+et.所以我们可以用如下公式模拟时间序列的下一个观测值：yT+1=yT+1|T+eT+1其中，yT+1|T是一步预测值，eT+1是未来的未知误差。假设将来的错误与过去的错误相似，我们可以从过去看到的错误集合（即残差）中抽样来代替eT+1。将新的模拟观察添加到我们的数据集中，重复此过程即可得到：yT+2=^yT+2|T+1+eT+2其中eT+2是从残差集合中提取的另一个值。以这种方式继续操作，我们可以为时间序列模拟一整套未来的值。

重复操作，我们获得了许多可能的未来预测值。然后我们可以通过计算每个预测区间的百分比来计算整个的预测区间，其结果我们称为“自助”的预测区间。“自助”是利用自己的力量，因为在这种方法中我们通过利用历史数据来测量未来的不确定性。

为了生成这样的区间，我们可以简单地将bootstrap参数添加到预测函数中。例如:

naive(goog200, bootstrap=TRUE)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 201 531.4783 525.7393 537.4780 522.8631 541.3329  
## 202 531.4783 523.2064 539.7370 519.8481 546.5242  
## 203 531.4783 521.3577 541.4112 516.6776 551.4153  
## 204 531.4783 519.3323 543.0781 514.4028 557.7721  
## 205 531.4783 517.7811 544.2263 511.8253 568.5871  
## 206 531.4783 516.3744 545.9069 509.9582 582.6038  
## 207 531.4783 514.8215 547.8828 507.6365 584.7688  
## 208 531.4783 513.5417 549.6699 505.6479 586.5182  
## 209 531.4783 512.0949 551.3792 503.9676 588.0916  
## 210 531.4783 511.4880 553.2640 502.3303 589.5307

#> Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
#> 201 531.5 525.7 537.8 522.9 542.9  
#> 202 531.5 523.2 539.5 519.4 547.0  
#> 203 531.5 520.9 541.2 516.7 552.3  
#> 204 531.5 519.0 543.0 514.0 560.3  
#> 205 531.5 517.5 544.6 511.8 582.1  
#> 206 531.5 516.1 545.9 509.5 582.4  
#> 207 531.5 514.8 547.3 508.0 583.5  
#> 208 531.5 513.5 548.9 505.8 584.9  
#> 209 531.5 512.3 549.8 503.9 586.6  
#> 210 531.5 510.7 551.4 502.1 587.5

在这种情况下，它们与基于正态分布的预测区间非常相似（但不完全相同）。

#### 包含变换过程的预测区间

如果使用了变换，则预测区间应该在转换后的测度里计算，再将最后一个点点逆转换到原来的测度后，给出预测区间。这种方法保留了预测区间的置信度，尽管它在点预测中不再是对称的。

如果在预测时使用lambda参数，则用Rforecast包中的函数可以自动完成预测区间的逆变换。

### 3.6R中的预测包

本书所用forecast包中的函数，都会在加载fpp2包时自动加载。本节附录简要概述fpp2包的一些特性，请参阅个人功能的帮助文件，以了解更多信息，并查看它们使用的一些示例。

#### 输出预测对象的函数：

包括meanf()，naive()，snaive()和rwf()等很多的函数，都对应以forecast类对象的形式输出（例如，一个forecast类的对象），这就允许其他的函数（如autoplot()）可以在一系列的预测模型中被一致的使用。

forecast类对象包括预测方法、所用数据、所得到的预测点、预测区间、残差和拟合值等信息。有很多函数，例如autoplot()，summary()和print()都可以处理这些对象。

下面的列表显示了所有能生成forecast对象的函数：

* meanf()
* naive(),snaive()
* rwf()
* croston()
* stlf()
* ses()
* holt(),hw()
* splinef()
* thetaf()
* forecast()

#### forecast()函数

到目前为止，我们已经使用了一些可以生成forecast对象的函数。但更常见的做法是，选择一个适合给定数据的模型，从而用forecast()函数生成预测值，这将在本书的后半部分进行介绍。

forecast()函数的输入有多种方式，但通常是以时间序列或时间序列模型作为主要参数，从而进行预测。

预测结果总是以forecast类对象的形式进行返回。但如果第一个参数属于ts类，则返回自动ETS算法的预测值，该算法将在第[7](https://otexts.com/fppcn/expsmooth-cn.html#expsmooth-cn)章中讨论。

下面是一个简单的例子，将forecast()函数应用于ausbeer数据：

forecast(ausbeer)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2010 Q3 404.6025 385.8695 423.3355 375.9529 433.2521  
## 2010 Q4 480.3982 457.5283 503.2682 445.4216 515.3748  
## 2011 Q1 417.0367 396.5112 437.5622 385.6456 448.4277  
## 2011 Q2 383.0996 363.5063 402.6930 353.1341 413.0651  
## 2011 Q3 403.0006 380.1438 425.8574 368.0442 437.9570  
## 2011 Q4 478.4943 450.1973 506.7913 435.2177 521.7708  
## 2012 Q1 415.3821 389.6837 441.0806 376.0797 454.6845  
## 2012 Q2 381.5781 356.8145 406.3418 343.7053 419.4509

#> Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
#> 2010 Q3 404.6 385.9 423.3 376.0 433.3  
#> 2010 Q4 480.4 457.5 503.3 445.4 515.4  
#> 2011 Q1 417.0 396.5 437.6 385.6 448.4  
#> 2011 Q2 383.1 363.5 402.7 353.1 413.1  
#> 2011 Q3 403.0 380.1 425.9 368.0 438.0  
#> 2011 Q4 478.5 450.2 506.8 435.2 521.8  
#> 2012 Q1 415.4 389.7 441.1 376.1 454.7  
#> 2012 Q2 381.6 356.8 406.3 343.7 419.5

其实通过调包，就算你不知道使用什么样的模型，依然可以很好的解决问题。但到了本书的结尾，你应该不需要再在“黑箱”里使用forecast()。相反，你可以选择适当的数据模型，然后用forecast()函数生成预测。