# 时间序列分析 3&4

**对平稳序列的参数估计及定阶**

**2016100104028　李科**

**内容简述：利用去趋势和去周期后的移动开户数据，对平稳序列进行模型的选择，模型参数的估计和定阶以及模型的优化和检验。本次实验内容及结论概括如下。**

## 1. 前期工作概述

1.1 观察数据三倍标准差去除异常值

1.2 去趋势去周期得到平稳时间序列并将平稳序列存储为 detrend_deT_data.csv

1.3 根据理解自己定义自相关函数和自偏相关函数

## 2. 处理后数据适用的模型判别

观察平稳序列的自相关函数和偏相关函数，通过是否拖尾/结尾判断使用的模型。

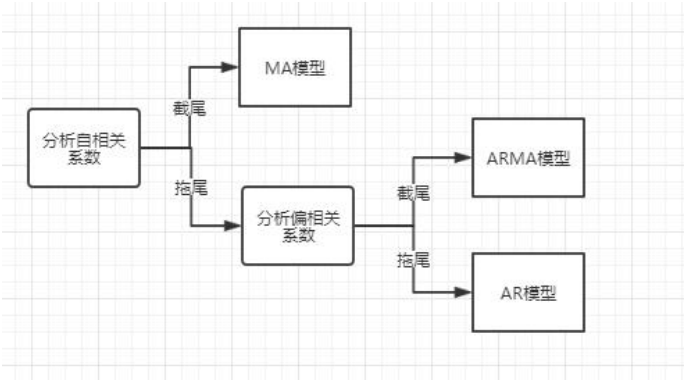


图*1 模型判别流程图

通过分析，自相关系数和自偏相关系数均呈现拖尾性，故使用 ARMA 模型进行进一步分析。

## 3. ARMA 模型的参数估计和定阶

拟采用课件 4.3 介绍的求解 ARMA 方法二：拟函数法进行求解，求解步骤如下：

3.1 初定阶数 p 和 q,运用 Yule-Walker 求解逆函数 I

3.2 求解 MA 滑动平均系数

3.3 求解 AR 自回归系数

3.4 在求出所有参数后，利用极大似然法残差平方和求方差，算出 AIC/BIC 指标

3.5 重复不走 3.1-3.4，在一定大范围内找出极小 BIC 值对应的阶数 p，q

## 4. 参数检验和定阶的进一步修正

通过模型的显著性检验和参数的显著性检验，在参数估计和定阶最小 BIC 情况下对应的结束 p,q 附近进行，最终找到最佳的 ARMA 阶数为 ARMA（1，1），参数求解为

$$\mathrm{X}_t - 0.1686\, X_{t-1} = \varepsilon_t - 0.7095\, \varepsilon_{t-1}$$

经检验残差序列为白噪声，认为模型效果可以接受。

# 1. 适用模型识别

首先，我们需要做的是根据动态数据，从各类模型族中选择与实际过程相吻合的模型；即处理后的平稳序列应该用什么模型来表示最为合理。经过前期的预处理，去趋势和去周期，得到平稳时间序列的自相关系数，偏相关系数如下图 1.1。

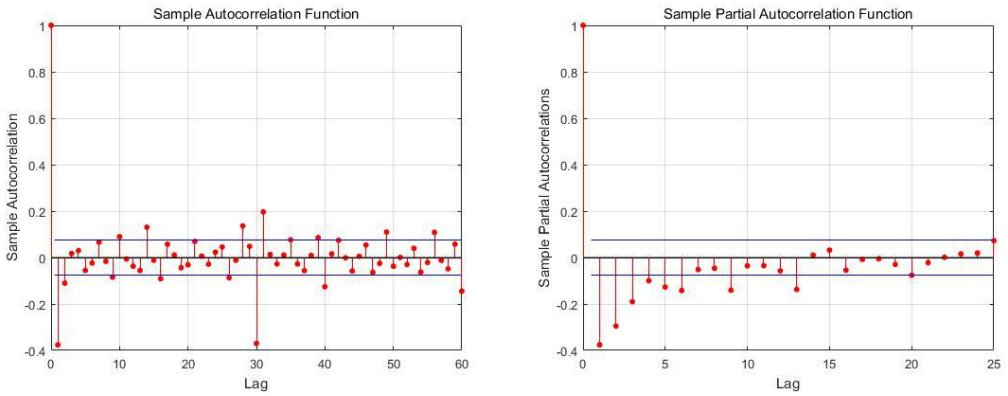


图 1. 1 平稳序列的自相关/自协相关系数

不难发现，该平稳序列的自相关系数和自偏相关系数均有拖尾的性质，且自相关拖尾阶数较高分布在 32-35 阶左右，而自偏相关系数拖尾阶数较低，分布在 12-20 之间。综上，拟采用 ARMA 模型对其进行拟合。

# 2.ARMA 模型的参数估计和定阶

确定好适用的模型之后，下一步我们要进行的是对确定的模型类别，定出模型中的阶数 p 和 q，这里采用先大范围搜索，在搜索的结束中求的模型的参数估计——自回归系数和滑动平均系数，并由此求出拟合残差的方差。最后，计算每个阶数下对应的 AIC/BIC 进行比较，这里适用最小 BIC 值对应的阶数和参数估计做下一步的分析。

## 2.1 方法的选取

课件 4.3 关于 ARMA 模型的参数估计中介绍了两种估计参数的方法，对这两种方法的理解如下：

*ARMA 模型参数的矩估计：通过 Y-W 方程求解出自回归系数，利用 MA(q)模型参数的线性迭代法或牛顿—拉裴森算法求得滑动平均系数，最后合并可得拟合模型。这种方法 矩估计方法精度较低，通常只能作为其他迭代参数估计算法的初值。

*模型参数的初估计法—逆函数法：利用可逆域的逆转形式进行求解。这种方法仅涉及到线性方程的求解，计算简单便于实现。

综合上述两种方法，考虑到逆函数法对于低阶或高阶模型都很有效且求解方便。这里采用逆函数法对模型参数进行估计，逆函数法求解步骤如下

逆函数法求解步骤

1. 令 P(*) = max(p,q)+q,用 y—w 方法估计 AR(p*)的 p* 个自回归参数，作为相应的 ARMA( p, q)模型的前 p*个逆函数值.

2. 求解线性方程组 $(1-\theta_1 B-\theta_2 B^2-\cdots-\theta_q B^q)I_j=0$ 得到滑动平均系数的估计。

3. 将前两步的结果代入课件 4.3.6 公式，求得自回归系数的估计。

**2.2 模型参数和求解和定阶**

**2.2.1 模型的定阶指标**

*最终预报误差准则（PDE)

$$FPE(k)=\frac{N+k+1}{N-k-1}\hat{\sigma}_k^2,\quad k=1,2,\ldots,M$$

其中，方差依赖于 k，其大小反映了模型与数据的拟合程度。第一个因子随 k 增大而增大，放大了残差方差的不确定性影响.

*最小信息准则(AIC)

$$AIC=N\ln\widehat{\sigma_\varepsilon^2}+2m$$

AIC 准则可应用于 ARMA 模型及其他统计模型(如多项式回归定阶).能在模型参数极大似然估计的基础上，对于 ARMA(p, q)的阶数和相应的参数，同时给出一种最佳估计.

*BIC 准则

$$BIC=N\ln\widehat{\sigma_\varepsilon^2}+m\ln N$$

AIC 准则避免了统计检验中由于选取置信度而产生的人为性，为模型定阶带来许多方便.但 AIC 方法未给出相容估计.

**注意：AIC 阶数估计一般偏高，BIC 阶数估计偏低。这里适用 BIC 准则进行初始阶数的定阶，取 BIC 值最小下对应的阶数 p,q 即为初始定阶。

**2.2.2 模型的求解**

观察之前的自相关系数图和自偏相关系数图，初定 ARMA 模型初始阶数的选取 p 和 q 均在 1-20 阶内，利用逆函数求解并算的每个 p,q 下参数对应的 BIC 值，取最小 BIC 对应的 p,q 即为初定的阶数。不同怕 p,q 下对应的 BIC 值如下图。

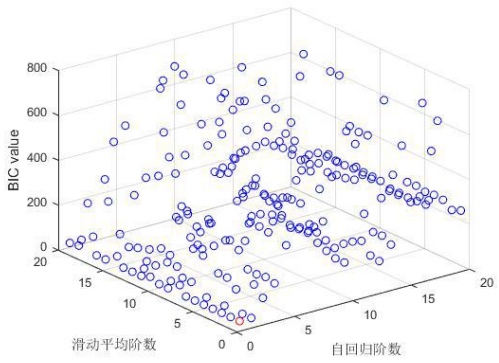


图 2.2.2 BIC 值的变化

由 BIC 值的变化我们可以看出，在取 ARMA(1,1)时 BIC 值最小。此时 ARMA(1,1)模型建立如下：

$$X_t-0.1686\,X_{t-1}=\varepsilon_t-0.7095\,\varepsilon_{t-1}$$

# 3. 定阶的进一步修正-模型的检验

确定好最小 BIC 准则下的参数估计值和模型阶数 p 和 q 后，我们知道——以上各个问题是相互关联的，需整体进行系统化的模型优化. 而根据已有的一组样本数据建立模型，对模型阶数和参数作出判断和估计，是重要的两部分工作.所以最后，我们需要在已经初定的阶数周围小范围内进行搜索和检验，通过模型的有效性检验的参数的显著性检验进一步优化模型，寻找最优的阶数和参数估计值。统计模型只是对生成观测数据真实过程的近似，在模型拟合后，还需要进行模型的有效性检验，及检验拟合模型对序列中信息的提取是否充分。模型的有效性检验即对残差的白噪声检验。

观察残差序列及其自相关系数如下图所示:

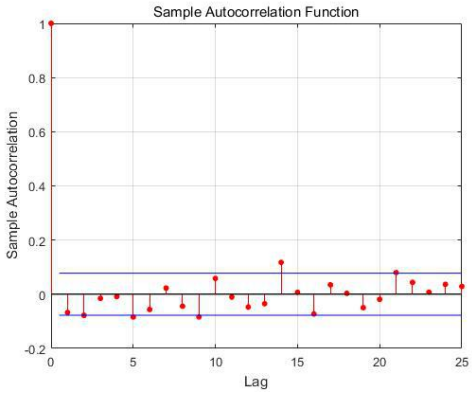


图 3.1 残差序列自相关系数

可以发现序列间没有相关性，不具有拖尾和截尾的性质，可以初步判断改模型是可靠的。为了进一步验证残差序列是否为白噪声序列，下面进行 LB 统计量的检验结果如下:

| 延迟期数 | LB 统计量 | 卡方统计量 |
|---|---|---|
| 6 | 2.64 | 1.63 |
| 12 | 4.67 | 5.23 |
| 18 | 7.67 | 9.93 |

由 LB 统计量小于对应的卡方统计量，认为该序列可以认为是白噪声序列。模型效果较好，模型检验通过。

*相关代码及解释请详见以下附件

# 附录

## 1 主函数　main1.m

实现功能：读取处理后的平稳数据进行建模，通过观察自相关/自偏相关系数确定选用 ARMA 模型，在一定确定的阶数范围内求模型的参数估计，根据最小 BIC 准则初定 p,q 的选择，最后通过模型的有效性检验和参数的显著性检验确定最终的参数估计和阶数。

```
% 读取处理后的平稳时间序列
processed_data = csvread('detrend_deT_data.csv');
figure(1)
autocorr(processed_data,60)
figure(2)
parcorr(processed_data,25)
data_autocorr = autocorr(processed_data,600);
% 自相关系数向量转换为自相关系数矩阵
autocorr_matrix = vec2mat(data_autocorr);
max_p = 15;
max_q = 20;
AICs = ones(max_p,max_q);
BICs = ones(max_p,max_q);
N = 600;
for p=1:20
    for q = 1:20
        % 计算 I
        P = max(p+q)+q;
        I = autocorr_matrix(1:P,1:P)\autocorr_matrix(1,2:P+1)';
        % 计算 MA 部分的滑动平均系数
        seta = calu_MA_parameter(I,p,q);
        % 计算 AR 部分的自回归系数
        fine = calu_AR_parameter(seta,I,p,q);
        % 计算方差
        residual_var = calu_var(p,q,fine,seta,N,processed_data);
        % 计算 AIC 和 BIC 并存储
        [AICs(p,q),BICs(p,q)] = calu_AIC_BIC(p,q,N,residual_var);
    end
end
% 找出最小 BIC 下的阶数 p,q
[min_BIC_p,min_BIC_q] = find(BICs == min(BICs));
```

## 2.MA 滑动平均系数的求解函数

## [seta]=calu_MA_parameter（I,p,q）

函数输入：逆函数{ Ij }的估计 I,给定的自回归和滑动平均阶数 p 和 q

函数输出：滑动平均系数的估计值 seta

```
function [seta]= calu_MA_parameter(I,p,q)
P = max(p,q);
I_matrix = zeros(q,q);
for ii = 1:q
    I_matrix(ii,:) = I(P+ii-1:-1:P+ii-q);
end
seta = I_matrix\I(P+1:P+q);
end
```

# 3.AR 自回归系数的求解函数

# [fine] = calu_AR_parameter(seta,I,p,q)

函数输入：逆函数{ Ij }的估计 I,给定的自回归和滑动平均阶数 p 和 q 以及滑动平均系数 seta
函数输出：自回归系数的估计值 fine

```
function [fine] = calu_AR_parameter(seta,I,p,q)
fine = ones(1,p);
for ii = 1:p
    if ii>q
        seta(ii) = 0;
    end
    fine(ii) = seta(ii)+I(ii);
    for jj = 1:ii-1
        if (ii-jj)>0
            fine(ii) = fine(ii)-seta(jj)*I(ii-jj);
        end
    end
end
```

# 4. 残差方差的计算函数

## [residual_var]= calu_var(p,q,fine,seta,N,processed_data)

函数输入：逆函数{ Ij }的估计 I,给定的自回归和滑动平均阶数 p 和 q 以及滑动平均系数 seta，自回归系数 fine，处理后的平稳序列。
函数输出：残差的方差

```
function [residual_var] = calu_var(p,q,fine,seta,N,processed_data)
error = zeros(1,N);
for ii = max(p,q)+1:N
    error(ii) =
[1,-fine]*processed_data(ii:-1:ii-p)+seta'*error(ii-1:-1:ii-q)';
end
```

```
residual_var = (error*error')/N;
end
```

## 5.AIC/BIC 指标的计算函数

### [AIC,BIC] = calu_AIC_BIC(p,q,N,residual_var)

函数输入：给定的自回归和滑动平均阶数 p 和 q，样本个数 N 以及残差方差。
函数输出：不同阶数下对应的 AIC/BIC 指标。

```
function [AIC,BIC] = calu_AIC_BIC(p,q,N,residual_var)
AIC = log(residual_var)+2*(p+q+1)/N;
BIC = log(residual_var)+log(N)*(p+q+1)/N;
End
```

## 6.计算 LB 统计量函数

## lb = calu_LB(data, m)

函数输入：残差序列，延迟期数
函数输出：LB 卡方统计量

```
function lb = calu_LB(data, m)
N = 25;
power_autocorr = 0;
data_autocorr = autocorr(data,25);
for k=1:m
    power_autocorr = power_autocorr+data_autocorr(k+1)^2/(N-k);
end
lb = N*(N+2)*power_autocorr;
End
```

## 7.自相关系数向量转为矩阵 vec2mat

## [autocorr_matrix] = vec2mat(autocorr_vector)

函数输入：自相关系数函数输出：LB 卡方统计量

```
function [autocorr_matrix] = vec2mat(autocorr_vector)
n = length(autocorr_vector);
autocorr_matrix = zeros(n);
for ii = 1:n
    autocorr_matrix(ii,ii:n) = autocorr_vector(1:n-ii+1);
end
autocorr_matrix = triu(autocorr_matrix)+tril(autocorr_matrix',-1);
end
```