基于 PA 分布的 INAR(1) 模型分析 (代码)

李天恺　　林瑞洋　　董宇坤

目录

1　写在前面

该部分内容为正文的附加内容，主要作用为展示正文中的代码函数，但展示内容不包括作图代码，具体循环结构 (例如第 4 节中模拟实验当中的循环部分) 等复杂的部分。文中会附带对每部分代码的简略解释，所有代码都已经过测试。

2　代码展示

```
## 返回 PA 分布的 cdf 值
pad_cdf <- function(x,lambda){
  return(1-(4*lambda+2*lambda*x+1)/((1+2*lambda)^(x+2)))
}
```

```r
## 生成服从 PA 分布的随机数，该函数主要用于第 4 节中的模拟实验
pad_sim <- function(n,lambda){
  u <- runif(n)
  x <- rep(0,n)
  for (i in 1:n) {
    k <- 0
    repeat{
      if(u[i] < pad_cdf(k,lambda)) break
      k <- k + 1
    }
    x[i] <- k
  }
  return(x)
}

## 细化算子
multi <- function(alpha,x){
  x_new <- sum(sample(0:1,x,prob = c((1-alpha),alpha),replace = T))
  return(x_new)
}

## 生成算法 (可以生成 PA-INAR(1) 和 P-INAR(1))
inar_sim <- function(n = 100, alpha = 0.5, lambda1 = 1
                     , type = "PAD",initial = 0){
  x <- rep(0,n)
  x[1] <- initial
  if(type == "Poisson"){
    eps <- rpois(n,lambda1)
  }else if (type=="PAD") {
    eps <- pad_sim(n,lambda1)
  }
  for (t in 2:n) {
      x[t] <- multi(alpha,x[t-1]) + eps[t]
  }
  return(x)
}

##Yule-Walker 方法
pad_yw <- function(x){
  x_bar <- mean(x)
  n <- length(x)
  gam0 <- sum((x-x_bar)^2)
  gam1 <- 0
  for (t in 2:n) {
    gam1 <- gam1 + (x[t] - x_bar) * (x[t-1] - x_bar)
  }
  alpha <- gam1/gam0
  lambda <- 1/((1-alpha)*x_bar)
  return(c(alpha,lambda))
}

##Conditional Least Squares Estimation
pad_cls <- function(x){
  n <- length(x)
  alpha <- (sum(x*c(0,x[-n]))-
              sum(x[-1])*sum(x[-n])/(n-1))/(sum((x[-n])^2)-(sum(x[-n]))^2/(n-1))
  mu <- 1/(n-1)*(sum(x[-1])-alpha*sum(x[-n]))
  lambda <- 1/(mu)
  return(c(alpha,lambda))
}

##Conditional Maximum Likelihood Estimation

## 对数条件似然函数
pad_loglh <- function(parameters,data){
  x <- data
  n <- length(x)
  loglh <- 0
  alpha <- pmin(pmax(parameters[1], .Machine$double.eps),
               1-(.Machine$double.eps))
  lambda <- pmax(parameters[2], .Machine$double.eps)
  for (t in 2:n) {
    u <- 0
    for (j in 0:(min(x[t],x[t-1]))) {
      u <- u + choose(x[t-1],j) * (alpha^j) * ((1-alpha)^(x[t-1]-j))
      * 4 * lambda^2 * (1+x[t]-j)/((1+2*lambda)^(x[t]-j+2))
    }
    loglh <- loglh + log(u)
  }
  return(-loglh)
}

## 条件极大似然估计
pad_ml <- function(x){
  theta_initial <- pad_yw(x)
  result <- optim(par = theta_initial,fn = pad_loglh,data = x,method = "BFGS")
  theta <- result$par
  return(c(pmin(pmax(theta[1], 0), 1), theta[2]))
}

# 模拟实验一次循环展示
#thm_a 和 thm_t 用于控制初值
set.seed(123)
times <- c(100,250,500,1000)
n <- 200
alpha_yw <- rep(0,4)
theta_yw <- rep(0,4)
alpha_cls <- rep(0,4)
theta_cls <- rep(0,4)
```

```
alpha_ml <- rep(0,4)
theta_ml <- rep(0,4)
err <- matrix(0, nrow = 4, ncol = 6)
a_yw <- rep(0,n)
t_yw <- rep(0,n)
a_cls <- rep(0,n)
t_cls <- rep(0,n)
a_ml <- rep(0,n)
t_ml <- rep(0,n)
thm_a <- 0.75
thm_t <- 1.5
for (i in 1:4) {
  for (j in 1:n) {
    x <- inar_sim(n = times[i],alpha = thm_a,lambda1 = thm_t,
                  type = "PAD",initial = 0)
    yw <- pad_yw(x)
    cls <- pad_cls(x)
    ml <- pad_ml(x)
    a_yw[j] <- yw[1]
    t_yw[j] <- yw[2]
    a_cls[j] <- cls[1]
    t_cls[j] <- cls[2]
    a_ml[j] <- ml[1]
    t_ml[j] <- ml[2]
  }
  alpha_yw[i] <- mean(a_yw)
  err[i,1] <- mean((a_yw-thm_a)^2)
  theta_yw[i] <- mean(t_yw)
  err[i,2] <- mean((t_yw-thm_t)^2)
  alpha_cls[i] <- mean(a_cls)
  err[i,3] <- mean((a_cls-thm_a)^2)
  theta_cls[i] <- mean(t_cls)
  err[i,4] <- mean((t_cls-thm_t)^2)
  alpha_ml[i] <- mean(a_ml)
  err[i,5] <- mean((a_ml-thm_a)^2)
  theta_ml[i] <- mean(t_ml)
  err[i,6] <- mean((t_ml-thm_t)^2)
}


## 真实数据案例代码
#poisson 对数似然函数
poi_loglh <- function(parameters,data){
  x <- data
  n <- length(x)
  loglh <- 0
  alpha <- pmin(pmax(parameters[1], .Machine$double.eps),
                1-(.Machine$double.eps))
  lambda <- pmax(parameters[2], .Machine$double.eps)
  for (t in 2:n) {
    u <- 0
    for (j in 0:(min(x[t],x[t-1]))) {
      u <- u + choose(x[t-1],j) * (alpha^j) * ((1-alpha)^(x[t-1]-j)) *
        exp(-lambda) * (lambda^(x[t]-j))/factorial((x[t]-j))
    }
    loglh <- loglh + log(u)
  }
  return(-loglh)
}

##poisson lindley 对数似然函数
pl_loglh <- function(parameters,data){
  x <- data
  n <- length(x)
  loglh <- 0
  alpha <- pmin(pmax(parameters[1], .Machine$double.eps),
                1-(.Machine$double.eps))
  theta <- pmax(parameters[2], .Machine$double.eps)
  for (t in 2:n) {
    u <- 0
    for (j in 0:(min(x[t],x[t-1]))) {
      u <- u + choose(x[t-1],j) * (alpha^j) * ((1-alpha)^(x[t-1]-j)) *
        theta^2 * (theta + 2 + x[t]-j)/(theta + 1)^(x[t]-j+3)
    }
    loglh <- loglh + log(u)
  }
  return(-loglh)
}

##poisson Yule-Walker 估计
poi_yw <- function(x){
  x_bar <- mean(x)
  n <- length(x)
  gam0 <- sum((x-x_bar)^2)
  gam1 <- 0
  for (t in 2:n) {
    gam1 <- gam1 + (x[t] - x_bar) * (x[t-1] - x_bar)
  }
  alpha <- gam1/gam0
  lambda <- (1-alpha)*x_bar
  return(c(alpha,lambda))
}

##poisson lindley Yule-Walker 估计
pl_yw <- function(x){
  x_bar <- mean(x)
  n <- length(x)
  gam0 <- sum((x-x_bar)^2)
  gam1 <- 0
  for (t in 2:n) {
    gam1 <- gam1 + (x[t] - x_bar) * (x[t-1] - x_bar)
  }
  alpha <- gam1/gam0
  theta <- 2/((1-alpha)*x_bar)
  return(c(alpha,theta))
}



##aic 和 bic 计算
ic_cal <- function(x,type = "UPA"){
  if(type=="PAD"){
    theta_initial <- pad_yw(x)
    result <- optim(par = theta_initial,fn = pad_loglh,data = x,method = "BFGS")
  }else if(type=="Poisson"){
    theta_initial <- poi_yw(x)
    result <- optim(par = theta_initial,fn = poi_loglh,data = x,method = "BFGS")
  }else if(type=="PL"){
    theta_initial <- pl_yw(x)
    result <- optim(par = theta_initial,fn = pl_loglh,data = x,method = "BFGS")
  }

  n <- length(x)
  log_value <- - result$value
  par <- result$par
  aic <- -2 * log_value + 4
  bic <- -2 * log_value + log(n) * 2
  return(c(aic,bic,par))
}
```

3　分工情况

董宇坤主要负责数据收集、协助其余二人工作。

林瑞洋主要负责框架构建、论文撰写。

李天恺主要负责代码实现。

三人均参与文献查找等工作。

总的来说三人工作量基本相等。