第三次上机作业

斯蓬 220810332

2024-11-17

目录

1 第三章课后习题编程部分

1.1 Task3.1

```
# 设置参数
lambda <- 1  # 泊松分布参数
n <- 500     # 样本大小
num_simulations <- 1000  # 模拟次数

# 存储结果
means <- numeric(num_simulations)
variances <- numeric(num_simulations)

# 模拟实验
set.seed(123)  # 设置随机种子，保证可复现性
for (i in 1:num_simulations) {
  sample_data <- rpois(n, lambda)
  means[i] <- mean(sample_data)
  variances[i] <- var(sample_data)
}

# 输出期望和方差的估计
mean(means)  # 平均值的期望
```

```
## [1] 1.000486
```

```
mean(variances)  # 方差的期望
```

```
## [1] 1.00214
```

1.2 Task3.2

```r
# 设置参数
n <- 500
alpha_true <- 0.7
num_simulations <- 1000  # 模拟次数

# 定义密度函数的随机生成器
generate_data <- function(alpha, n) {
  runif(n)^(1 / (alpha + 1))
}

# 定义估计函数
mse <- function(estimate, true_value) {
  mean((estimate - true_value)^2)
}

# 矩估计和最大似然估计
moment_estimates <- numeric(num_simulations)
mle_estimates <- numeric(num_simulations)

set.seed(123)
for (i in 1:num_simulations) {
  sample_data <- generate_data(alpha_true, n)

  # 矩估计：根据样本均值推导
  moment_estimates[i] <- 1 / mean(sample_data) - 1

  # 最大似然估计：优化得到
  log_likelihood <- function(alpha) {
    sum(log((alpha + 1) * sample_data^alpha))
  }
  mle_estimates[i] <- optimize(log_likelihood, interval = c(0, 5), maximum = TRUE)$maximum
}
```

```
# 计算均方误差
mse(moment_estimates, alpha_true)
```

```
## [1] 0.01306736
```

```
mse(mle_estimates, alpha_true)
```

```
## [1] 0.005264249
```

1.3 Task3.3

```
# 设置参数
n <- 300
alpha <- 1
beta <- 1

# 生成数据
set.seed(123)
x <- rnorm(n, 0, 1)  # 自变量 x
epsilon <- rnorm(n, 0, 1)  # 随机误差
y <- alpha + beta * x + epsilon  # 因变量 y

# 手动计算线性回归参数
x_mean <- mean(x)
y_mean <- mean(y)

# 计算 beta 和 alpha
beta_hat_manual <- sum((x - x_mean) * (y - y_mean)) / sum((x - x_mean)^2)
alpha_hat_manual <- y_mean - beta_hat_manual * x_mean

# 使用 R 的 lm 函数计算
model <- lm(y ~ x)

# 比较手动计算结果与 lm 结果
```

```
manual_result <- c(alpha_hat_manual, beta_hat_manual)
lm_result <- coef(model)

# 打印结果
cat(" 手动计算结果:\n")
```

```
## 手动计算结果:
```

```
print(manual_result)
```

```
## [1] 1.0112948 0.9365451
```

```
cat("\nR lm 函数结果:\n")
```

```
##
## R lm 函数结果:
```

```
print(lm_result)
```

```
## (Intercept)           x
##   1.0112948   0.9365451
```

可以看到编程计算的结果和自带的结果一致。

2 第三章例题复现部分

2.1 3.1.2

```
set.seed(220810332)
library(MASS)
K=1000
n=50
theta1=matrix(0,K,1)
theta2=matrix(0,K,1)
for (i in 1:K){
```

```
data=mvrnorm(n,c(0,0),diag(2))
theta1[i]=mean(abs(data[,1]-data[,2]))
theta2[i]=var(abs(data[,1]-data[,2]))
}
c(mean(theta1),2/(sqrt(pi)))
```

```
## [1] 1.134533 1.128379
```

```
c(mean(theta2),2-4/pi)
```

```
## [1] 0.7256894 0.7267605
```

2.2 3.2.2

```
set.seed(1)
K=1000      # 循环次数
n=50          # 样本量
a=0.05
mu=0
sigma=1
inter=matrix(0,K,2)
prob=matrix(0,K,1)
for(i in 1:K){
data=rnorm(n,mu,sigma)
Q=var(data)*(n-1)
chi1=qchisq(1-a/2,n-1)
chi2=qchisq(a/2,n-1)
inter[i,]=c(Q/chi1,Q/chi2)
prob[i]=(sigma>inter[i,1])&(sigma<inter[i,2])
}
colMeans(inter)
```

```
## [1] 0.7065346 1.5723243
```

```r
mean(inter[,2]-inter[,1])                    # 平均长度
```

```
## [1] 0.8657898
```

```r
mean(prob)
```

```
## [1] 0.948
```

3 第四章课后习题编程部分

3.1 Task4.1

```r
# 设置参数
n <- 100
m <- 50
mu1 <- 0
mu2 <- 2
sigma <- 1
alpha <- 0.05  # 显著性水平
num_simulations <- 100  # 模拟次数

# 检验功效计算
set.seed(123)
reject_null <- numeric(num_simulations)

for (i in 1:num_simulations) {
  x <- rnorm(n, mu1, sigma)  # 样本 X
  y <- rnorm(m, mu2, sigma)  # 样本 Y

  # 两独立样本 t 检验
  t_test <- t.test(x, y, var.equal = TRUE)

  # 检查是否拒绝 H0
```

```r
  reject_null[i] <- ifelse(t_test$p.value < alpha, 1, 0)
}

# 计算功效
power <- mean(reject_null)
cat(" 检验功效为: ", power, "\n")
```

```
## 检验功效为:  1
```

3.2 Task4.2

```r
# 血糖浓度数据
data <- c(87, 77, 92, 68, 80, 78, 84, 80, 77, 92, 86, 76, 80, 81, 75, 92, 78, 80, 88, 86, 77, 87)

# 参数
mu <- 80
sigma <- 6

# 1. 卡方检验
# 将数据分组（注意分组观测值数量不得少于 5 个）
breaks <- c(-Inf, 74, 80, 86, Inf)
observed <- table(cut(data, breaks))

# 理论频数
theoretical <- length(data) * diff(pnorm(breaks, mean = mu, sd = sigma))

# 卡方统计量
chisq_stat <- sum((observed - theoretical)^2 / theoretical)
chisq_p_value <- pchisq(chisq_stat, df = length(breaks) - 1 - 1, lower.tail = FALSE)

cat(" 卡方检验统计量: ", chisq_stat, "\n")
```

```
## 卡方检验统计量:  6.84381
```

```
cat(" 卡方检验 p 值: ", chisq_p_value, "\n")
```

```
## 卡方检验p值:  0.07704626
```

```
# 2. K-S 检验
ks_test <- ks.test(data, "pnorm", mean = mu, sd = sigma)

cat("K-S 检验统计量: ", ks_test$statistic, "\n")
```

```
## K-S检验统计量:  0.2049811
```

```
cat("K-S 检验 p 值: ", ks_test$p.value, "\n")
```

```
## K-S检验p值:  0.3136521
```

3.3 4.4

```
# 设置参数
set.seed(123)
m <- 50  # 样本 X 的大小
n <- 30  # 样本 Y 的大小

# 样本分布
x <- rnorm(m, mean = 0, sd = 1)  # X ~ N(0, 1)
y <- rnorm(n, mean = 0, sd = 2)  # Y ~ N(0, 2)

# K-S 检验
ks_test <- ks.test(x, y)

# 输出结果
cat("K-S 检验统计量: ", ks_test$statistic, "\n")
```

```
## K-S检验统计量:  0.1933333
```

```
cat("K-S 检验 P 值: ", ks_test$p.value, "\n")
```

```
## K-S检验P值:  0.4371957
```

```
# 设置模拟参数
num_simulations <- 100  # 模拟次数
alpha <- 0.05  # 显著性水平
power_count <- 0  # 用于记录拒绝 H0 的次数

# 模拟计算检验功效
for (i in 1:num_simulations) {
  x_sim <- rnorm(m, mean = 0, sd = 1)  # 样本 X
  y_sim <- rnorm(n, mean = 0, sd = 2)  # 样本 Y

  ks_sim <- ks.test(x_sim, y_sim)

  if (ks_sim$p.value < alpha) {
    power_count <- power_count + 1  # 拒绝 H0 计数
  }
}

# 计算功效
power <- power_count / num_simulations
cat("K-S 检验的功效为: ", power, "\n")
```

```
## K-S检验的功效为:  0.35
```

4 第四章例题复现部分

4.1 4.1.1

```
set.seed(1)
n=100
```

```r
res=c()
mu=0.3
for (i in 1:1000) {
  data=rnorm(n)
  E_data=mean(data)+mu
  stat=E_data*sqrt(n)
  res[i]=as.numeric(abs(stat)>=qnorm(0.975,0,1))
}
result=mean(res)    # 数值模拟估计的统计功效
criti=qnorm(0.975,0,1)
power=2-pnorm(criti-sqrt(n)*mu,0,1)-pnorm(criti+sqrt(n)*mu,0,1)    # 统计功效
c(result, power)
```

```
## [1] 0.8470000 0.8508388
```

4.2 P58 单样本 KS 检验

```r
set.seed(220810332)
n = 35
stat1 = NULL
res1 = NULL
res2 = NULL
for (i in 1:1000){
  data=rt(n,1)
  data=sort(data)
  D_splus=max(abs (c (1 : n) /n-pnorm(data)))
  D_minus=max(abs(pnorm(data)- (c(1:n)-1 )/n))
  stat1=max(D_splus,D_minus)
  res1[i]=as.numeric(stat1>0.23)
  index=seq(1,10000,1)
  p_val=2*sum((-1)^(index-1)*exp(-2*n*index^2*stat1^2))
  res2[i]=as.numeric(p_val<0.05)
}
```

```r
c(mean(res1),mean(res2))
```

```
## [1] 0.268 0.270
```