

R Review

尹超

中国科学院大学，北京 100049

Carter Yin

University of Chinese Academy of Sciences, Beijing 100049, China

2024.8 – 2025.1

序言

R Review

目录

序言	I
目录	II
1 问题 1: 使用最大似然估计 (MLE) 进行自助法 (Bootstrapping)	1
2 问题 2: 使用矩估计进行自助法 (Bootstrapping)	2
3 问题 3: 单样本 Z 检验 (已知方差)	4
4 问题 4: Wilcoxon 秩和检验、t 检验和 Hardy-Weinberg 平衡检验	6
5 问题 5	7
6 12.02 数据处理与可视化	9
7 12.04 数据处理与回归分析	11
8 12.09	13
9 12.25	15

问题 1：使用最大似然估计 (MLE) 进行自助法 (Bootstrapping)

```
1  # 定义样本大小 (观察值的数量)
2  mysamplesize = 115
3
4  # 构造置信区间时的显著性水平 (alpha)
5  alpha = 0.05
6
7  # 定义真实参数值 (指数分布的偏移量)
8  mytheta = 1
9
10 # 从指数分布生成一个大小为 `mysamplesize` 的随机样本, 并加上偏移量 `mytheta`
11 mysample = rexp(mysamplesize, rate = 1) + mytheta
12
13 # 计算最大似然估计 (MLE), 这里采用排序后样本中的最小值 (指数分布的MLE就是样本中的最小值)
14 mymle = sort(mysample)[1]
15
16 # 自助法样本的数量
17 bootstrapsize = 1000
18
19 # 初始化一个向量来存储每次自助法估计的结果
20 bootstrapestimates = rep(0, bootstrapsize)
21
22 # 执行自助法: 生成自助样本并计算它们的估计值
23 for (ii in 1:bootstrapsize) {
24   # 使用MLE作为偏移量生成一个新的自助样本
25   bootstrapsample = rexp(mysamplesize, rate = 1) + mymle
26
27   # 每个自助样本的估计值是排序后的最小值 (类似MLE)
28   bootstrapestimates[ii] = sort(bootstrapsample)[1]
29 }
30
31 # 排序自助法的估计值, 为计算分位数做准备
32 bootstrapquantiles = sort(bootstrapestimates - mymle)
33
34 # 计算置信区间的下分位数 (alpha/2)
35 lowerquantile = bootstrapquantiles[round(bootstrapsize*alpha*0.5)]
36
37 # 计算置信区间的上分位数 (1-alpha/2)
38 upperquantile = bootstrapquantiles[round(bootstrapsize*(1-alpha*0.5))]
39
40 # 计算置信区间的下界, 通过从MLE中减去上分位数
41 lowerbound = mymle - upperquantile
42
43 # 计算置信区间的上界, 通过从MLE中减去下分位数
44 upperbound = mymle - lowerquantile
45
46 # 输出置信区间的下界
47 lowerbound
48
49 # 输出置信区间的上界
50 upperbound
51
52 > lowerbound
53 [1] 0.9730528
54 > upperbound
55 [1] 1.003367
```

问题 2：使用矩估计进行自助法（Bootstrapping）

```
1      # 样本大小与问题1相同
2  mysamplesize = 115
3
4  # 构造置信区间时的显著性水平（alpha）
5  alpha = 0.05
6
7  # 定义真实参数值（指数分布的偏移量）
8  mytheta = 1
9
10 # 从指数分布生成一个样本，并加上偏移量 `mytheta`
11 mysample = rexp(mysamplesize, rate = 1) + mytheta
12
13 # 计算矩估计，这里是样本均值减去1（对于指数分布，矩估计是样本均值减去1）
14 mymomentestimation = mean(mysample) - 1
15
16 # 自助法样本的数量
17 bootstrapsize = 1000
18
19 # 初始化一个向量来存储每次自助法估计的结果
20 bootstrapestimates = rep(0, bootstrapsize)
21
22 # 执行自助法：生成自助样本并计算它们的估计值
23 for (ii in 1:bootstrapsize) {
24     # 使用矩估计作为偏移量生成一个新的自助样本
25     bootstrapsample = rexp(mysamplesize, rate = 1) + mymomentestimation
26
27     # 每个自助样本的矩估计值是样本均值减去1
28     bootstrapestimates[ii] = mean(bootstrapsample) - 1
29 }
30
31 # 排序自助法的估计值，为计算分位数做准备
32 bootstrapquantiles = sort(bootstrapestimates - mymomentestimation)
33
34 # 计算置信区间的下分位数（alpha/2）
35 lowerquantile = bootstrapquantiles[round(bootstrapsize*alpha*0.5)]
36
37 # 计算置信区间的上分位数（1-alpha/2）
38 upperquantile = bootstrapquantiles[round(bootstrapsize*(1-alpha*0.5))]
39
40 # 计算置信区间的下界，通过从矩估计中减去上分位数
41 lowerbound = mymomentestimation - upperquantile
42
43 # 计算置信区间的上界，通过从矩估计中减去下分位数
44 upperbound = mymomentestimation - lowerquantile
45
46 # 输出置信区间的下界
47 lowerbound
48
49 # 输出置信区间的上界
50 upperbound
51
52
53 > lowerbound
54 [1] 0.8166889
55 > upperbound
56 [1] 1.165083
```

解释

随机样本生成

`rexp(mysamplesize, rate = 1)` 用于从指数分布中生成随机数, `rate = 1` 指的是指数分布的速率参数为 1。在两种情况中, 都将 `mytheta` (MLE 中的真实值) 或 `mymomentestimation` (矩估计中的真实值) 加到样本中。

最大似然估计 (MLE)

对于指数分布, MLE 是排序后样本中的最小值 (`sort(mysample)[1]`), 因为指数分布的 MLE 就是样本中的最小值。

矩估计

矩估计是计算样本均值减去 1 (`mean(mysample) - 1`), 这是根据矩估计法得到的结果, 用于指数分布。

自助法 (Bootstrapping)

自助法通过对样本进行有放回抽样, 再次计算估计值 (MLE 或矩估计), 并计算这些估计值的分布。置信区间是基于自助法估计值的分位数计算的 (`lowerquantile` 和 `upperquantile`)。

置信区间计算

最终步骤是通过自助法估计值的分位数计算置信区间的上下界。计算方式是将分位数从原始估计值 (MLE 或矩估计) 中减去。

这个代码实现了通过自助法估计最大似然估计 (MLE) 和矩估计的置信区间。两者的逻辑类似, 但初始估计的计算方法不同。

问题 3：单样本 Z 检验（已知方差）

```
1 # one-sample z-test (variance known) -----
2
3 nail = c(2.942371, 2.988662, 3.106234, 3.109316, 3.118427, 3.132254,
4         3.140042, 3.170188, 2.902562, 3.128003, 3.146441, 2.978240,
5         3.103600, 3.003394, 3.044384, 2.849916) # 样本数据
6 sigma = 0.1 # 方差已知, sigma为已知的标准差
7 mu = 3 # 零假设下的均值
8 n = length(nail) # 样本数量
9 # 检验统计量, 零假设下服从标准正态分布
10 Z_statistic = sqrt(n) * (mean(nail) - mu) / sigma # 计算Z统计量
11 # help(pnorm) # 查询pnorm函数的帮助文档
12 p_value = 2 * pnorm(Z_statistic, lower.tail = F) # 计算双尾p值
13 p_value # 输出p值
14 # 0.03076609 # 计算得到的p值
15
16 # 方法2: 使用BSDA包进行Z检验
17 # install.packages("BSDA") # 安装BSDA包
18 library(BSDA) # 加载BSDA包
19 # help(z.test) # 查询z.test函数的帮助文档
20 result = z.test(nail, mu = mu, alternative = c("two.sided"), sigma.x = sigma) # 使用z.test进行Z检验
21 p_value = result$p.value # 提取p值
22 p_value # 输出p值
23 # 0.03076609 # 计算得到的p值
24
25 ### 单样本t检验 (未知方差)
26
27 # help(t.test) # 查询t.test函数的帮助文档
28 result = t.test(nail, mu = mu, alternative = c("two.sided")) # 进行单样本t检验
29 p_value = result$p.value # 提取p值
30 p_value # 输出p值
31 # 0.04297243 # 计算得到的p值
32
33 ### 双样本t检验
34
35 # H0: 磷肥没有效果 mu1=mu2
36 # H1: 磷肥有正面效果 (提高产量) mu1-mu2>0
37 # 设置两组数据
38 x = c(62, 57, 65, 60, 63, 58, 57, 60, 60, 58) # 第一组数据
39 y = c(50, 59, 56, 57, 58, 57, 56, 55, 57) # 第二组数据
40 # 假设两个总体方差相等, 进行双样本t检验
41 result = t.test(x, y, alternative = c("greater"), var.equal = T) # 方差相等假设
42 p_value = result$p.value # 提取p值
43 p_value # 输出p值
44 # 0.002471312 # 计算得到的p值
45
46 # 假设两个总体方差不相等, 进行双样本t检验
47 result = t.test(x, y, alternative = c("greater"), var.equal = F) # 方差不等假设
48 p_value = result$p.value # 提取p值
49 p_value # 输出p值
50 # 0.002450701 # 计算得到的p值
51
52 ### 检验
53
54 # 不使用函数计算p值
55
56 # help(pt) # 查询pt函数的帮助文档
```

```
57
58 # 单样本Z检验 (已知方差)
59 Z_statistic <- 4 * (mean(nail) - 3) / 0.1 # 计算Z统计量
60 p_value <- 2 * pnorm(Z_statistic, lower.tail = F) # 计算双尾p值
61 p_value # 输出p值
62
63 # 单样本t检验 (未知方差)
64 S <- sqrt(sum((nail - mean(nail))^2) / 15) # 计算样本的标准差
65 T_statistic <- 4 * (mean(nail) - 3) / S # 计算t统计量
66 p_value <- 2 * pt(T_statistic, df = 15, lower.tail = F) # 计算双尾p值
67 p_value # 输出p值
68
69 # 双样本t检验 (方差相等)
70 S1 <- sqrt(sum((x - mean(x))^2) / 9) # 计算第一组样本的标准差
71 S2 <- sqrt(sum((y - mean(y))^2) / 8) # 计算第二组样本的标准差
72 T_statistic <- (mean(x) - mean(y)) / sqrt(1/9 + 1/10) / sqrt((9 * S1^2 + 8 * S2^2) / 17) # 计算t统计量
73 p_value <- pt(T_statistic, df = 17, lower.tail = F) # 计算p值
74 p_value # 输出p值
75
76 # 双样本t检验 (方差不等)
77 SEw <- sqrt(1/10 * S1^2 + 1/9 * S2^2) # 计算标准误差
78 df <- SEw^4 / (S1^4/10/10/9 + S2^4/9/9/8) # 计算自由度
79 T_statistic <- (mean(x) - mean(y)) / SEw # 计算t统计量
80 p_value <- pt(T_statistic, df = df, lower.tail = F) # 计算p值
81 p_value # 输出p值
```

解释

单样本 Z 检验 (已知方差)

使用已知的总体标准差进行 Z 检验，通过计算 Z 统计量 (`Z_statistic`) 并从标准正态分布中查找 p 值。

方法 2: BSDA 包的 Z 检验

使用 BSDA 包中的 `z.test` 函数来进行 Z 检验，并提取 p 值。

单样本 t 检验 (未知方差)

在样本方差未知的情况下，使用 t 检验并从结果中提取 p 值。

双样本 t 检验

通过假设方差相等或不等，进行双样本 t 检验，检验两个组的均值差异。

检验部分

不使用内建函数手动计算 Z 统计量、t 统计量和 p 值，分别进行了单样本和双样本检验的计算过程。

问题 4: Wilcoxon 秩和检验、t 检验和 Hardy-Weinberg 平衡检验

```
1 Wilcoxon 检验
2 # help(wilcox.test) # 查询 wilcox.test 函数的帮助文档
3 Aarea = c(62, 57.04, 65, 60, 63, 58.1, 57.001, 60.1, 60.2, 58) # 样本A数据
4 Barea = c(50, 59, 56.01, 57.01, 58.05, 57.02, 56, 55, 57.03) # 样本B数据
5 result = wilcox.test(Aarea, Barea, exact = TRUE, alternative = c("two.sided")) # 进行Wilcoxon秩和检验
6 # (双尾检验)
7 p_value = result$p.value # 提取p值
8 p_value # 输出p值
9 # p_value = 0.002987724 # 计算得到的p值
10
11 t 检验
12 result = t.test(Aarea, Barea, alternative = c("greater")) # 进行t检验 (单尾检验, 假设样本A的均值大于样
13 # 本B的均值)
14 p_value = result$p.value # 提取p值
15 p_value # 输出p值
16 # p_value = 0.002307728 # 计算得到的p值
17
18 Hardy-Weinberg 平衡检验
19 # MLE of theta
20 theta <- 874/(1184+874) # 计算最大似然估计 (MLE) 的theta值, 这里假设是基因型频率
21 # MLE of p_M
22 p1 <- (1-theta)^2 # MLE of p_M, 计算纯合隐性基因型 (MM) 的频率
23 # MLE of p_MN
24 p2 <- 2*theta*(1-theta) # MLE of p_MN, 计算杂合基因型 (MN) 的频率
25 # MLE of p_N
26 p3 <- theta^2 # MLE of p_N, 计算纯合显性基因型 (NN) 的频率
27 # expression of test statistic
28 Z <- (342-1029*p1)^2/(1029*p1) + (500-1029*p2)^2/(1029*p2) + (187-1029*p3)^2/(1029*p3) # 计算卡方检验统
29 # 计量
30 p_value <- pchisq(Z, df = 1, lower.tail = FALSE) # 计算卡方分布的p值, df=1是自由度
31 p_value # 输出p值
32 # p_value = 0.8569258 # 计算得到的p值
```

解释

Wilcoxon 检验

`wilcox.test()` 用于执行非参数检验,比较两组数据的分布是否相同。此处为双尾检验。通过 `result$p.value` 提取 p 值, 结果表明 p 值为 0.002987724, 表明两组数据差异显著。

t 检验

`t.test()` 用于执行传统的 t 检验, 这里使用单尾检验, 假设组 A 的均值大于组 B。通过 `result$p.value` 提取并输出 t 检验的 p 值, 结果为 0.002307728, 表明组 A 的均值显著大于组 B。

Hardy-Weinberg 平衡检验

使用最大似然估计 (MLE) 方法计算等位基因频率, 并基于这些频率计算卡方检验的统计量。通过 `pchisq()` 计算卡方分布的 p 值, 结果为 0.8569258, 表示未拒绝零假设, 基因频率符合 Hardy-Weinberg 平衡。

问题 5

```
1 Jane Austen 数据分析
2 table1 <- matrix(c(14, 133, 12, 241, 11, 259, 16, 180, 14, 285, 6, 265, 8, 93,
3                   12, 139, 8, 221, 2, 81, 1, 153, 17, 204),
4                   nrow = 6) # 创建一个6行4列的矩阵table1, 包含Jane Austen数据
5 result <- chisq.test(table1[, 1:3]) # 对table1的前三列进行卡方检验
6 p_value <- result$p.value # 提取p值
7 p_value # 输出p值
8 # p_value = 0.009734904 # 计算得到的p值
9
10 # 前三列求行和作为一列
11 table2 <- matrix(c(38, 406, 38, 665, 25, 745, 2, 81, 1, 153, 17, 204), nrow = 6) # 创建一个新的6行4列矩阵table2
12 result <- chisq.test(table2) # 对table2进行卡方检验
13 p_value <- result$p.value # 提取p值
14 p_value # 输出p值
15 # p_value = 1.669517e-05 # 计算得到的p值
16
17 探索性数据分析 (EDA)
18 library(MASS) # 加载MASS包, 提供统计工具和数据集
19 library(lattice) # 加载lattice包, 提供绘图功能
20
21 # base:plot, ggplot2, lattice # 介绍常用的绘图工具包
22
23 # R内置数据集hills, 包含1984年35场苏格兰山地赛跑的记录时间数据
24 hills # 查看hills数据
25 ?hills # 查看hills数据集的帮助文档, 了解数据集的背景和描述
26
27 splom(~ hills) # 散点图矩阵, 展示数据中各个变量的关系
28
29 attach(hills) # 使用attach函数, 使数据集hills中的变量可以直接使用
30
31 plot(hills$dist, hills$time) # 绘制散点图, 展示distance (距离) 与time (时间) 的关系
32
33 # interactive()测试当前是否是R的交互模式
34 # 如果是, 则运行identify函数打开一个交互窗口
35 # 该交互窗口可以在图上的点上直接点击, 显示标签
36 # 在原版R中可以正常操作, 但是Rstudio里出现了偏差, 原因不明
37 # if(interactive()) identify(dist, time, row.names(hills)) # 在交互模式下显示数据点的标签
38
39 # detach(hills) # 取消attach, 避免之后引用变量时出现混淆
40
41 4.5 Trellis 图形
42 xyplot(time ~ dist, data = hills, # 创建一个Trellis散点图, 展示时间与距离的关系
43        panel = function(x, y, ...) { # 定义绘图面板函数
44          panel.xyplot(x, y, ...) # 绘制散点图
45          panel.lmline(x, y, type = "l") # 添加最小二乘回归线
46          panel.abline(lqs(y ~ x), lty = 3) # 添加resistant regression线 (稳健回归线), lty控制线条类型
47        }
48 )
49
50 median(hills$time) # 计算时间数据的中位数
51 mad(hills$time) # 计算时间数据的绝对中位差
52 histogram(~time, data = hills) # 绘制时间数据的直方图
53 densityplot(~time, data = hills) # 绘制时间数据的密度图
54 stem(hills$time) # 绘制时间数据的茎叶图
55 qqnorm(hills$time) # 绘制时间数据的QQ图, 用于检验数据是否符合正态分布
```

```
56  
57  
58 Michelson 实验数据  
59 michelson # 查看michelson数据集，包含光速实验数据  
60 ?michelson # 查看michelson数据集的帮助文档  
61  
62 bwplot(Expt ~ Speed, data = michelson, main = "Speed of Light Data", ylab = "Experiment No.") # 绘制箱线图，展示不同实验光速的分布  
63 # title("Speed of Light Data") # 添加标题，但与lattice包的绘图函数一起使用时可能会出现问题
```

解释

Jane Austen 数据分析

使用 `chisq.test()` 进行卡方检验，比较不同类别的数据频率。创建了两个数据表 `table1` 和 `table2`，并对其进行卡方检验，提取并输出 `p` 值，判断数据是否存在显著差异。

探索性数据分析（EDA）

使用 `MASS` 和 `lattice` 包进行数据分析和可视化。展示了如何使用散点图、散点图矩阵、Trellis 图等可视化方法探索数据，帮助理解数据分布和关系。计算了数据的中位数、绝对中位差，并绘制了直方图、密度图、茎叶图和 QQ 图。

Michelson 实验数据

使用 `michelson` 数据集，绘制光速实验的箱线图，展示不同实验中光速测量的分布。

12.02 数据处理与可视化

```
1 # 请注意R的工作目录和Pearson.csv文件所在的目录
2
3 # 获取当前工作目录
4 getwd()
5 # 设置工作目录, 确保R能找到文件
6 setwd("C:/Users/WangKe/Desktop/TA/2024TA/exercise class")
7 # 读取CSV文件“Pearson.csv”, 并将数据存储在height中
8 height = read.csv("Pearson.csv")
9 # 如果文件是txt格式的, 可以用read.table方法读取
10 # height = read.table("Pearson.txt", header = TRUE)
11
12 # 另一种方法: 使用绝对路径直接读取CSV文件
13 height = read.csv("C:/Users/WangKe/Desktop/TA/2024TA/exercise class/Pearson.csv")
14 # 另一种方法, 读取txt文件
15 # height = read.table("C:/Users/WangKe/Desktop/TA/2024TA/exercise class/Pearson.txt", header = TRUE)
16
17 # 绘制散点图, bty控制图的边框类型, pch控制点的形状
18 plot(Son ~ Father, data = height, bty = "n", pch = 20)
19 # 添加直线, a为截距, b为斜率, lty控制线条类型, lwd控制线条宽度
20 abline(a = 0, b = 1, lty = 1, lwd = 2)
21 # 画出线性回归模型的回归线, Son为y, Father为x
22 abline(lm(Son ~ Father, data = height), lty = 1, lwd = 2)
23
24 # 计算Father列的标准差
25 Father.sd = sd(height$Father)
26 # 计算Son列的标准差
27 Son.sd = sd(height$Son)
28 # 计算Father列的均值
29 Father.ave = mean(height$Father)
30 # 计算Son列的均值
31 Son.ave = mean(height$Son)
32
33 # 计算Pearson相关系数, 衡量Father和Son之间的线性关系
34 Pearson.cor = cor(height$Father, height$Son)
35
36 # 绘制Father数据的直方图, 检查其分布情况
37 hist(height$Father)
38 # 绘制Son数据的直方图, 检查其分布情况
39 hist(height$Son)
40
41 # 绘制Father数据的核密度图, 进一步查看分布
42 plot(density(height$Father))
43 # 绘制Son数据的核密度图, 进一步查看分布
44 plot(density(height$Son))
45
46 # 绘制QQ图, 检查Father数据是否服从正态分布
47 qqnorm(height$Father)
48 qqline(height$Father) # 绘制QQ图的参考线
49 # 绘制QQ图, 检查Son数据是否服从正态分布
50 qqnorm(height$Son)
51 qqline(height$Son) # 绘制QQ图的参考线
```

12.02

读取数据

通过 `read.csv()` 或 `read.table()` 函数读取 CSV 或 TXT 文件，并设置工作目录。

可视化

使用 `plot()` 绘制散点图，`abline()` 添加回归线，`lm()` 进行线性回归。计算数据的标准差、均值和 Pearson 相关系数。

分布检验

通过直方图、核密度图和 QQ 图检查数据的分布情况，检查是否符合正态分布。

12.04 数据处理与回归分析

```
1 # 设置工作目录，确保R能找到文件
2 setwd("C:/Users/WangKe/Desktop/TA/2024TA/exercise class")
3 # 读取CSV文件“Pearson.csv”，并将数据存储在height中
4 height = read.csv("Pearson.csv")
5 # 绘制Son和Father的散点图
6 plot(Son ~ Father, data = height, bty = "l", pch = 20)
7
8 # 计算Son和Father的均值
9 Son.ave = mean(height$Son)
10 Father.ave = mean(height$Father)
11 # 绘制Son和Father的回归线，截距为Son的均值减去Father的均值，斜率为1
12 abline(a = Son.ave - Father.ave, b = 1, lty = 1, lwd = 2)
13 # 绘制Son和Father的回归线，使用线性回归模型
14 abline(lm(Son ~ Father, data = height), lty = 1, lwd = 2)
15
16 # 查看线性回归的结果，显示回归系数等信息
17 summary(lm(Son ~ Father, data = height))
18
19 # 读取另一份数据文件“reading.csv”，并指定列名
20 reading <- read.csv("reading.csv", col.names = c("S1982", "S1983"))
21 # 绘制S1982和S1983的散点图
22 plot(S1983 ~ S1982, data = reading, bty = "l", pch = 20)
23
24 # 计算S1983和S1982的均值
25 S1983.ave = mean(reading$S1983)
26 S1982.ave = mean(reading$S1982)
27 # 绘制S1983和S1982的回归线，截距为S1983的均值减去S1982的均值，斜率为1
28 abline(a = S1983.ave - S1982.ave, b = 1, lty = 1, lwd = 2)
29 # 绘制S1983和S1982的回归线，使用线性回归模型
30 abline(lm(S1983 ~ S1982, data = reading), lty = 1, lwd = 2)
31
32 # 创建线性回归模型，预测S1983基于S1982
33 model <- lm(S1983 ~ S1982, data = reading)
34 # 输出回归模型的摘要
35 model
36 # 显示线性回归模型的详细统计信息
37 summary(model)
38
39 # 设置显著性水平alpha为0.05
40 alpha <- 0.05
41 # 计算回归系数的置信区间，置信度为1-alpha
42 confint(model, level = 1 - alpha)
43
44
```

12.04

回归分析

在两组数据上绘制回归线，使用 `lm()` 进行线性回归，计算回归系数，并生成回归模型的摘要。

置信区间

通过 `confint()` 计算回归系数的置信区间，设定显著性水平为 0.05。

12.09

```
1 # 安装并加载 deming 包（如果尚未安装，可以取消注释进行安装）
2 # install.packages('deming')
3 library(deming) # 加载deming包，用于Deming回归
4
5 # 绘制散点图，bty控制图的边框类型，pch控制点的形状
6 plot(aes ~ aas, data = arsenate, bty = "l", pch = 20)
7
8 # 添加线性回归直线，使用 lm() 计算线性回归，并用 abline() 绘制回归线
9 abline(lm(aes ~ aas, data = arsenate), lty = 1, lwd = 2, col = 2)
10
11 # 添加Deming回归直线，使用 deming() 函数计算Deming回归，并用 abline() 绘制回归线
12 abline(deming(aes ~ aas, data = arsenate, xstd = se.aas, ystd = se.aes), lty = 1, lwd = 2, col = 4)
13
14 # 使用 lm() 建立线性回归模型，并存储为 fit1
15 fit1 = lm(aes ~ aas, data = arsenate)
16
17 # 使用 deming() 建立Deming回归模型，并存储为 fit2
18 fit2 = deming(aes ~ aas, data = arsenate, xstd = se.aas, ystd = se.aes)
19
20 # 打印线性回归的结果
21 print(fit1)
22
23 # 打印Deming回归的结果
24 print(fit2)
25
26 # 计算线性回归模型的斜率
27 slope_lm = coefficients(fit1)[2]
28
29 # 计算Deming回归模型的斜率
30 slope_deming = coefficients(fit2)[2]
31
32 # 计算两者斜率的差值
33 slope_lm - slope_deming # -0.1283495
```

解释

安装和加载 deming 包

deming 包提供了 Deming 回归分析工具，该方法是处理误差变量的回归分析。

library(deming)：加载该包，使得后续代码可以使用其中的函数。

绘制散点图

plot(aes ~ aas, data = arsenate, bty = "l", pch = 20)：这行代码用于绘制 aes 和 aas 之间的散点图。bty = "l" 控制图形的边框样式，pch = 20 设置点的形状为实心圆。

添加回归线

abline(lm(aes ~ aas, data = arsenate), lty = 1, lwd = 2, col = 2)：这行代码使用普通最小二乘法（OLS）拟合线性回归，并将回归直线添加到散点图中。lty = 1 为线型，lwd = 2 设置线条宽度，

`col = 2` 设置颜色为红色。

`abline(deming(aes(aas, data = arsenate, xstd = se.aas, ystd = se.aes), lty = 1, lwd = 2, col = 4))`: 这行代码使用 Deming 回归拟合, 并将回归直线添加到散点图中, 颜色设置为蓝色。

建立回归模型

`fit1 = lm(aes(aas, data = arsenate))`: 使用线性回归 (OLS) 建立 aes 对 aas 的回归模型。

`fit2 = deming(aes(aas, data = arsenate, xstd = se.aas, ystd = se.aes))`: 使用 Deming 回归建立相同的回归模型。

打印回归结果

`print(fit1)`: 打印线性回归模型的结果。

`print(fit2)`: 打印 Deming 回归模型的结果。

计算斜率差异

`slope_lm = coefficients(fit1)[2]`: 提取线性回归模型的斜率。

`slope_deming = coefficients(fit2)[2]`: 提取 Deming 回归模型的斜率。

`slope_lm - slope_deming`: 计算两个模型斜率的差异, 结果为 -0.1283495, 表明两种方法的回归斜率存在一定差异。

通过这段代码, 你可以比较普通最小二乘法 (OLS) 回归和 Deming 回归在处理具有误差的自变量时的不同结果。

12.25

```
1 # 1. 加载数据
2 brainsize = read.csv("C:/Users/78003/Desktop/brainsize.csv") # 读取存储在桌面上的 CSV 文件，包含脑大小（
   BRAIN）、体重（BODY）、妊娠期（GESTATION）、胎数（LITTER）等数据
3
4 # 2. 先运行散点图矩阵（可选）
5 pairs(brainsize[, 2:5]) # 绘制数据集中第2至第5列（BODY、GESTATION、LITTER、BRAIN）之间的散点图矩阵，帮助检
   查变量之间的关系
6
7 # 3. 创建线性回归模型
8 linearmodel = lm(BRAIN ~ BODY + GESTATION + LITTER, data = brainsize) # 创建一个线性回归模型，预测BRAIN（脑
   大小）基于BODY（体重）、GESTATION（妊娠期）和LITTER（胎数）
9
10 # 4. 查看回归模型摘要
11 summary(linearmodel) # 输出线性回归模型的摘要，包括回归系数、标准误差、t值、p值等信息
12
13 # 5. 绘制BRAIN的密度图
14 plot(density(brainsize$BRAIN)) # 绘制脑大小（BRAIN）的密度图，帮助观察数据分布的形态
15
16 # 6. 绘制log(BRAIN)的密度图
17 plot(density(log(brainsize$BRAIN))) # 绘制对数变换后的脑大小（log(BRAIN)）的密度图，检查数据是否服从对数正
   态分布
18
19 # 7. Q-Q图，检查log(BRAIN)的正态性
20 qqnorm(log(brainsize$BRAIN)) # 绘制log(BRAIN)的Q-Q图，检查数据是否接近正态分布
21 qqline(log(brainsize$BRAIN)) # 添加Q-Q图的参考线，帮助判断数据的正态性
22
23 # 8. 对数变换后的散点图矩阵
24 pairs(cbind(log(brainsize[,2]), log(brainsize[,3]), log(brainsize[,4]), log(brainsize[,5]))) # 绘制对数变换
   后的散点图矩阵，检查log变换后的各个变量之间的关系
25
26 # 9. 使用对数变换创建新的线性回归模型
27 linearmodel = lm(log(BRAIN) ~ log(BODY) + log(GESTATION) + log(LITTER), data = brainsize) # 创建新的线性回
   归模型，预测对数变换后的BRAIN基于对数变换后的BODY、GESTATION和LITTER
28
29 summary(linearmodel) # 输出新的线性回归模型的摘要，检查回归系数和其他统计信息
30
31 # 10. 查看模型残差的Q-Q图
32 qqnorm(linearmodel$residuals) # 绘制回归模型残差的Q-Q图，检查残差是否接近正态分布
33 qqline(linearmodel$residuals) # 添加Q-Q图的参考线，帮助判断残差的正态性
```

解释

加载数据

`read.csv()` 用于读取 CSV 格式的文件，将数据加载到 R 中的 `brainsize` 数据框中。

散点图矩阵

`pairs()` 用于生成一组变量之间的散点图矩阵。通过查看散点图矩阵，你可以初步了解变量之间的关系。

线性回归模型

`lm()` 用于建立线性回归模型，指定自变量（`BODY`、`GESTATION`、`LITTER`）和因变量（`BRAIN`）。该模型假设脑大小是体重、妊娠期和胎数的线性函数。

回归模型摘要

`summary()` 提供模型的详细信息，包括回归系数、标准误差、`t` 值、`p` 值等统计量。通过这些信息，你可以了解模型的拟合情况。

密度图

`density()` 用于估计并绘制数据的密度图，帮助理解数据的分布情况。这里分别绘制了脑大小和其对数值的密度图。

Q-Q 图

`qqnorm()` 用于绘制 Q-Q 图，以检查数据是否符合正态分布。`qqline()` 添加参考线，用于判断数据是否偏离正态分布。

对数变换后的散点图矩阵

`cbind()` 函数将数据的对数变换合并在一起，`pairs()` 绘制对数变换后的散点图矩阵。对数变换有助于稳定数据的方差，并可能使数据分布更接近正态分布。

新的线性回归模型

使用对数变换后的数据建立新的回归模型，帮助缓解原数据可能存在的非正态性或异方差问题。

残差 Q-Q 图

通过分析模型残差的 Q-Q 图，检查残差是否服从正态分布，进而判断回归模型的适用性。

通过这些步骤，你可以全面地检查数据的分布、回归模型的拟合情况以及变换后数据的正态性。这些是进行线性回归分析和验证的重要步骤。

终点

知识点梳理与解释

文档设置与宏包导入

- `documentclass[UTF8]{report}`: 设定文档类型为报告，编码格式为 UTF-8。
- `usepackage{...}`: 导入各种宏包，如 `ctex`（中文支持）、`amsmath`（数学公式）、`graphicx`（插图）、`listings`（代码高亮）等。
- `geometry{...}`: 设置页面边距。
- `fancyhdr`: 设置页眉页脚。
- `titlesec`: 自定义标题格式。

数据读取与处理

- `read.csv()` 和 `read.table()`: 读取 CSV 或 TXT 文件，将数据加载到 R 中的数据框中。例如：

```
1 brainsize = read.csv("C:/Users/78003/Desktop/brainsize.csv")
2
```

- `setwd()`: 设置工作目录，确保 R 能找到需要读取的文件。例如：

```
1 setwd("C:/Users/WangKe/Desktop/TA/2024TA/exercise class")
2
```

- `getwd()`: 获取当前工作目录。例如：

```
1 getwd()
2
```

数据可视化

- `plot()`: 绘制散点图，展示两个变量之间的关系。例如：

```
1 plot(Son ~ Father, data = height, bty = "n", pch = 20)
2
```

- `abline()`: 在散点图上添加回归线。例如：

```
1 abline(lm(Son ~ Father, data = height), lty = 1, lwd = 2)
2
```

- `hist()`: 绘制直方图，检查数据分布。例如：

```
1 hist(height$Father)
2
```

- `density()`: 绘制密度图，估计数据分布。例如：

```
1 plot(density(height$Father))
2
```

- `qqnorm()` 和 `qqline()`: 绘制 Q-Q 图, 检查数据是否符合正态分布。例如:

```
1 qqnorm(height$Father)
2 qqline(height$Father)
3
```

- `pairs()`: 生成散点图矩阵, 展示多个变量之间的关系。例如:

```
1 pairs(brainsize[, 2:5])
2
```

- `bwplot()`: 绘制箱线图, 展示数据分布。例如:

```
1 bwplot(Expt ~ Speed, data = michelson, main = "Speed of Light Data", ylab = "Experiment No.")
2
```

统计检验与回归分析

- `lm()`: 建立线性回归模型, 指定自变量和因变量。例如:

```
1 linearmodel = lm(BRAIN ~ BODY + GESTATION + LITTER, data = brainsize)
2
```

- `summary()`: 提供模型的详细信息, 包括回归系数、标准误差、t 值、p 值等统计量。例如:

```
1 summary(linearmodel)
2
```

- `confint()`: 计算回归系数的置信区间。例如:

```
1 confint(model, level = 1 - alpha)
2
```

- `z.test()`: 使用 BSDA 包进行 Z 检验。例如:

```
1 result = z.test(nail, mu = mu, alternative = c("two.sided"), sigma.x = sigma)
2
```

- `t.test()`: 进行 t 检验。例如:

```
1 result = t.test(nail, mu = mu, alternative = c("two.sided"))
2
```

- `wilcox.test()`: 进行 Wilcoxon 秩和检验。例如:

```
1 result = wilcox.test(Aarea, Barea, exact = TRUE, alternative = c("two.sided"))
2
```

- `chisq.test()`: 进行卡方检验。例如:

```
1 result = chisq.test(table1[, 1:3])
2
```

- `deming()`: 进行 Deming 回归, 处理误差变量的回归分析。例如:

```
1 fit2 = deming(aes ~ aas, data = arsenate, xstd = se.aas, ystd = se.aes)
2
```

自助法 (Bootstrapping)

- 通过对样本进行有放回抽样，再次计算估计值，并计算这些估计值的分布，用于估计统计量的置信区间。例如：

```

1  bootstrapestimates = rep(0, bootstrapsize)
2  for (ii in 1:bootstrapsize) {
3      bootstrapsample = rexp(mysamplesize, rate = 1) + mymle
4      bootstrapestimates[ii] = sort(bootstrapsample)[1]
5  }
6  bootstrapquantiles = sort(bootstrapestimates - mymle)
7  lowerquantile = bootstrapquantiles[round(bootstrapsize*alpha*0.5)]
8  upperquantile = bootstrapquantiles[round(bootstrapsize*(1-alpha*0.5))]
9  lowerbound = mymle - upperquantile
10 upperbound = mymle - lowerquantile
11

```

- 计算置信区间的上下界，计算方式是将分位数从原始估计值中减去。例如：

```

1  lowerbound = mymomentestimation - upperquantile
2  upperbound = mymomentestimation - lowerquantile
3

```

数据变换与分布检验

- 对数变换：使用 `log()` 函数对数据进行对数变换，稳定数据的方差，使数据分布更接近正态分布。例如：

```

1  plot(density(log(brainsize$BRAIN)))
2

```

- `stem()`：绘制茎叶图，展示数据分布。例如：

```

1  stem(hills$time)
2

```

- `splom()`：绘制散点图矩阵，展示数据中各个变量的关系。例如：

```

1  splom(~ hills)
2

```

- `attach()` 和 `detach()`：使数据集中的变量可以直接使用，避免变量名冲突。例如：

```

1  attach(hills)
2  detach(hills)
3

```

模型残差分析

- 通过分析模型残差的 Q-Q 图，检查残差是否服从正态分布，进而判断回归模型的适用性。例如：

```

1  qqnorm(linearmodel$residuals)
2  qqline(linearmodel$residuals)
3

```

示例代码与解释

- 通过具体的 R 代码示例，展示如何进行数据读取、可视化、统计检验、回归分析、自助法等操作。
- 对每段代码进行详细解释，帮助理解每个步骤的目的和实现方法。

总结

通过这些步骤和知识点的梳理，你可以全面地进行数据分析和回归建模，包括数据读取、可视化、统计检验、自助法、数据变换、分布检验和模型残差分析等。这些是进行数据分析和验证的重要步骤。