R语言程序合集

```
#例4-5
#未提供样本点，使用公式正常计算(总体方差已知，提供的是σ^2,使用u检验，且n1和n2≥30)
#两组样本大小
n1=400
n2=200
#两组样本均值
mu1=69.5
mu2=70.3
#两组样本方差
var1=6.9^2
var2=6.9^2
#显著水平α(双边先÷2)
alpha=0.975
#零假设结果
mu=qnorm(alpha)
print("mu=")
print(mu)
#两组样本的方差
dealtVar=sqrt(var1/n1+var2/n2)
print("dealtVar=")
print(dealtVar)
#两组样本的u统计量
u=(mu1-mu2)/dealtVar
print("u=")
print(u)
#统计结果
if(!(abs(u)>mu))
{
  print("接受H0:u1=u2，拒绝H1:u1≠u2")
  print("两个样本无显著区别")
}else
{
  print("拒绝H0:u1=u2，接受H1:u1≠u2")
  print("两个样本存在显著区别")
}
```

```
#例4-6
#未提供样本点，使用公式正常计算(总体方差已知，使用u检验，提供的是σ^2,且n1或n2≤30)
#两组样本大小
n1=25
n2=30
#两组样本均值
mu1=3.71
mu2=3.46
#两组样本方差
var1=0.46
var2=0.37
```

```
#显著水平α(双边先÷2)
alpha=0.975
#零假设结果
mu=qnorm(alpha)
print("mu=")
print(mu)
#两组样本的方差
dealtVar=sqrt(var1/n1+var2/n2)
print("dealtVar=")
print(dealtVar)
#两组样本的u统计量
u=(mu1-mu2)/dealtVar
print("u=")
print(u)
#统计结果
if(!(abs(u)>mu))
{
  print("接受H0:u1=u2，拒绝H1:u1≠u2")
  print("两个样本无显著区别")
}else
{
  print("拒绝H0:u1=u2，接受H1:u1≠u2")
  print("两个样本存在显著区别")
}
```

```
#例4-7
#未提供样本点，使用公式正常计算(总体方差未知，提供的是样本方差s^2,使用u检验，且n1和n2≥30，为大样本，可近似)
#两组样本大小
n1=55
n2=107
#两组样本均值
mu1=95.4
mu2=77.6
#两组样本方差
var1=936.36
var2=800.89
#显著水平α(双边先÷2)
alpha=0.995
#零假设结果
mu=qnorm(alpha)
print("mu=")
print(mu)
#两组样本的方差
dealtVar=sqrt(var1/n1+var2/n2)
print("dealtVar=")
print(dealtVar)
#两组样本的u统计量
u=(mu1-mu2)/dealtVar
print("u=")
print(u)
#统计结果
```

```
if(!(abs(u)>mu))
{
  print("接受H0:u1=u2, 拒绝H1:u1≠u2")
  print("两个样本无显著区别")
}else
{
  print("拒绝H0:u1=u2, 接受H1:u1≠u2")
  print("两个样本存在显著区别")
}
```

```
# 例4-8
#样本数据1
x=c(134,146,106,119,124,161,107,83,113,129,97,123)
#样本数据2
y=c(70,118,101,85,107,132,94)
#F检验方差
var.test(x, y, ratio = 1,
         alternative ="two.sided",
         conf.level = 0.95)
#t检验均值
result=t.test(x, y ,
       alternative = "two.sided",
       mu = 0, paired = FALSE, var.equal = TRUE,
       conf.level = 0.95)
print(result)
if(result$p.value>1-alpha)
{
  print("接受H0:μd=0, 拒绝H1:μd≠0")
  print("两个样本无显著区别")
}else
{
  print("拒绝H0:μd=0, 接受H1:μd≠0")
  print("两个样本存在显著区别")
}
```

```
# 例4-9(t检验, 总体方差σ^2不同, 但是n相同)
#样本数据1
x=c(50,47,42,43,39,51,43,38,44,37)
#样本数据2
y=c(36,38,37,38,36,39,37,35,33,37)
#F检验方差
var.test(x, y, ratio = 1,
         alternative = "two.sided",
         conf.level = 0.99)
#t检验均值
result=t.test(x, y ,
       alternative = "two.sided",
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.99)
print(result)
if(result$p.value>1-alpha)
```

```
{
  print("接受H0:μd=0，拒绝H1:μd≠0")
  print("两个样本无显著区别")
}else
{
  print("拒绝H0:μd=0，接受H1:μd≠0")
  print("两个样本存在显著区别")
}
```

```
#例4-10
#未提供样本点，使用公式正常计算(总体方差未知，提供的是样本方差s^2和样本均值,使用t检验)
#两组样本大小
n1=10
n2=5
#两组样本均值
mu1=14.3
mu2=11.7
#两组样本方差
var1=1.621
var2=0.135
#平均化处理
var1real=var1/n1
var2real=var2/n2
#两组样本的R统计量
R=var1real/(var1real+var2real)
print("R=")
print(R)
#两组样本的R统计量
df=1/(R^2/(n1-1)+(1-R)^2/(n2-1))
print("df=")
print(df)
ndf=round(df)
#显著水平α(双边先÷2)
alpha=0.995
#t值表结果
tReal=qt(alpha,ndf)
print("tReal=")
print(tReal)
#两组样本的方差
dealtVar=sqrt(var1/n1+var2/n2)
print("dealtVar=")
print(dealtVar)
#两组样本的t统计量
t=(mu1-mu2)/dealtVar
print("t=")
print(t)
#统计结果
if(!(abs(t)>tReal))
{
  print("接受H0:u1=u2，拒绝H1:u1≠u2")
  print("两个样本无显著区别")
}else{
```

```
  print("拒绝H0:u1=u2，接受H1:u1≠u2")
  print("两个样本存在显著区别")}
```

```
#例4-11
#成对(paired)数据平均数比较的t检验
#样本数据1
x=c(3550,2000,3000,3950,3800,3750,3450,3050)
#样本数据2
y=c(2450,2400,1800,3200,3250,2700,2500,1750)
#置信度α
alpha=0.99
#t检验均值S
result=t.test(x, y ,
       alternative = "two.sided",
       mu = 0, paired = TRUE,
       conf.level = alpha)
print(result)
if(result$p.value>1-alpha)
{
  print("接受H0:μd=0，拒绝H1:μd≠0")
  print("两个样本无显著区别")
}else
{
  print("拒绝H0:μd=0，接受H1:μd≠0")
  print("两个样本存在显著区别")
}
```

```
#例5-5
x=c(22,24,25,27,32,35,37,39,43,45)
x_jitter <- x + runif(length(x), -0.01, 0.01)  # 给 x 添加微小扰动


wilcox.test(
  x_jitter, y_jitter=NULL,
  alternative = "greater",
  mu=28,
  paired = FALSE,
  exact = TRUE,  # 仍然要求精确计算
  correct = TRUE
)

wilcox.test(
  x,
  y=NULL,
  alternative = "greater",
  mu=28,
  paired = FALSE,exact = TRUE, correct = TRUE,
  conf.int = FALSE, conf.level = 0.95,
  tol.root = 1e-4, digits.rank = Inf
)
```

```r
#例5-6
library(coin)
x=c(94,88,83,92,87,95,90,90,86,84)
y=c(86,84,85,78,76,82,83,84,82,83)
x_jitter <- x + runif(length(x), -0.01, 0.01)  # 给 x 添加微小扰动
y_jitter <- y + runif(length(y), -0.01, 0.01)  # 给 y 添加微小扰动

wilcox.test(
  x_jitter, y_jitter,
  alternative = "two.sided",
  paired = TRUE,
  exact = TRUE,  # 仍然要求精确计算
  correct = TRUE
)

wilcox.test(
  x,
  y,
  alternative = "two.sided",
  paired = TRUE,exact = TRUE, correct = TRUE,
  conf.int = FALSE, conf.level = 0.95,
  tol.root = 1e-8, digits.rank = Inf
)

wilcox.test(
  x,
  y,
  alternative = "two.sided",
  paired = FALSE,exact = TRUE, correct = TRUE,
  conf.int = FALSE, conf.level = 0.95,
  tol.root = 1e-8, digits.rank = Inf
)

wilcoxsign_test(
  x ~ y,
  data = data.frame(x, y),
  alternative = "two.sided"
)
```

```r
#例5-8
x=c(148,143,138,145,142)
y=c(139,136,141,133,140)

wilcox.test(
  x,
  y,
  alternative = "greater",
  paired = FALSE,exact = TRUE, correct = TRUE,
  conf.int = FALSE, conf.level = 0.95,
  tol.root = 1e-8, digits.rank = Inf
)
```

```r
#例5-11
x <- c(49,50,54,58)
y <- c(36,41,38,42)
z <- c(33,34,31,35)
kruskal.test(list(x, y, z))
## Equivalently,
x <- c(x, y, z)
g <- factor(rep(1:3, c(4,4,4)),
            labels = c("A",
                       "B",
                       "C"))
kruskal.test(x, g)
```

```r
#例6.1 （函数自带的连续性矫正无法使用，采用手动矫正！）
# 输入数据
observed <- c(1503, 99)
expected_ratio <- c(3, 1)
alpha <- 0.99
df <- 1

# 计算理论频数
expected <- sum(observed) * expected_ratio / sum(expected_ratio)

# 手动计算连续性校正后的卡方值
corrected_chi <- sum( (abs(observed - expected) - 0.5 )^2 / expected )

# 计算p值
p_value <- pchisq(corrected_chi, df, lower.tail = FALSE)

# 输出结果
cat("连续性校正后的卡方值:", corrected_chi, "\n")
cat("p值:", p_value, "\n")
cat("理论频数:", expected, "\n")

# 检验结果判别
if (p_value > 1 - alpha) {
```

```
  print("接受H0, 拒绝H1")
  print("样本符合相应比率")
} else {
  print("拒绝H0, 接受H1")
  print("两个样本不符合相应比率")
}

# 可视化
barplot(
  rbind(observed, expected),
  beside = TRUE,
  names.arg = c("显性", "隐性"),
  col = c("skyblue", "orange"),
  main = "观测频数与理论频数对比",
  legend.text = c("观测值", "理论值")
)
```

```
#例6.2
# 输入数据
observed <- c(208, 81)
expected_ratio <- c(3, 1)
alpha=0.95
df=1
# 卡方检验
chi_test <- chisq.test(observed, p = expected_ratio, rescale.p = TRUE)
kaFang=qchisq(alpha, df)
# 输出结果
print(chi_test)
# 计算理论频数
expected <- sum(observed) * expected_ratio / sum(expected_ratio)

# 手动计算连续性校正后的卡方值
corrected_chi <- sum( (abs(observed - expected) - 0.5 )^2 / expected )

# 计算p值
p_value <- pchisq(corrected_chi, df, lower.tail = FALSE)

# 输出结果
cat("连续性校正后的卡方值:", corrected_chi, "\n")
cat("卡方real:",kaFang)
cat("理论频数:", sum(observed) * expected_ratio / sum(expected_ratio))

#检验结果判别
if(p_value>1-alpha)
{
  print("接受H0, 拒绝H1")
  print("样本符合相应比率")
}else
{
  print("拒绝H0, 接受H1")
  print("两个样本不符合相应比率")
}
```

```
# 可视化
barplot(
  rbind(observed, chi_test$expected),
  beside = TRUE,
  names.arg = c("显性", "隐性"),
  col = c("skyblue", "orange"),
  main = "观测频数与理论频数对比",
  legend.text = c("观测值", "理论值")
)
```

```
#例6.3
# 输入数据
observed <- c(315, 101,108,32)
expected_ratio <- c(9,3,3, 1)
alpha=0.95
df=3
# 卡方检验
chi_test <- chisq.test(observed, p = expected_ratio, rescale.p = TRUE)
kaFang=qchisq(alpha, df)
# 输出结果
print(chi_test)
cat("卡方real:",kaFang)
cat("理论频数:", sum(observed) * expected_ratio / sum(expected_ratio))

#检验结果判别
if(chi_test$p.value>1-alpha)
{
  print("接受H0, 拒绝H1")
  print("样本符合相应比率")
}else
{
  print("拒绝H0, 接受H1")
  print("两个样本不符合相应比率")
}

# 可视化
barplot(
  rbind(observed, chi_test$expected),
  beside = TRUE,
  names.arg = c("黄圆", "黄皱","绿圆","绿皱"),
  col = c("skyblue", "orange"),
  main = "观测频数与理论频数对比",
  legend.text = c("观测值", "理论值")
)
```

```
#例6-4
# 输入数据
data <- matrix(c(50, 250, 5, 195), nrow = 2, byrow = TRUE,
               dimnames = list(c("吸烟", "不吸烟"), c("患病", "未患病")))
```

```r
alpha=0.99
print(data)
# 卡方检验
chi_test <- chisq.test(data,correct=TRUE,rescale.p = TRUE)

# 输出结果
print(chi_test)

#检验结果判别
if(chi_test$p.value>1-alpha)
{
  print("接受H0, 拒绝H1")

  print("两组样本之间不存在相关性")
}else
{
  print("拒绝H0, 接受H1")
  print("两个样本之间存在相关性")
}
```

```r
#例6-5
# 输入数据
data <- matrix(c(37, 49,23, 150,100,57), nrow = 2, byrow = TRUE,
               dimnames = list(c("死亡", "未死亡"), c("甲", "乙","丙")))
alpha=0.95
print(data)
df=3
# 卡方检验
chi_test <- chisq.test(data,correct=TRUE,rescale.p = TRUE)

# 输出结果
print(chi_test)

#检验结果判别
if(chi_test$p.value>1-alpha)
{
  print("接受H0, 拒绝H1")

  print("两组样本不存在相关性")
}else
{
  print("拒绝H0, 接受H1")
  print("两组样本存在相关性")
}
```

```r
#相关系数test 使用例6-5的数据
library(DescTools)
# 输入数据
```

```r
data <- matrix(c(37, 49,23, 150,100,57), nrow = 2, byrow = TRUE,
               dimnames = list(c("死亡", "未死亡"), c("甲", "乙","丙")))
#ϕ相关系数
phi_coef <- Phi(data)
print("ϕ相关系数为：")
print(phi_coef)
#列联相关系数C
c_coef <- ContCoef(data)
cat("列联系数 C:", c_coef,"\n")
#V相关系数
cramer_v <- CramerV(data)
print("V相关系数为：")
print(cramer_v)
```

```r
#例6-6
# 输入数据
data <- matrix(c(67,9,10,5,32,23,20,4,10,11,23,5), nrow = 3, byrow = TRUE,
               dimnames = list(c("11-30", "31-50","50以上"), c("治愈", "显效","好转","无效")))
alpha=0.99
print(data)
df=6
# 卡方检验
chi_test <- chisq.test(data,correct=TRUE,rescale.p = TRUE)
fisher_test <- fisher.test(data,conf.level = alpha,simulate.p.value=TRUE,B=10000,alternative
= "two.sided",conf.int = TRUE)
# 输出结果
print(chi_test)
print(fisher_test)
#检验结果判别
if(chi_test$p.value>1-alpha)
{
  print("接受H0, 拒绝H1")

  print("样本不存在相关性")
}else
{
  print("拒绝H0, 接受H1")
  print("样本存在相关性")
}
```

```r
#例6-7
# 输入数据
data <- matrix(c(4,1,6,28), nrow = 2, byrow = TRUE,
               dimnames = list(c("死亡", "存活"), c("CMT", "ECMO")))
alpha=0.95
print(data)
# Fisher和卡方检验
fisher_test <- fisher.test(data,conf.level = alpha)
chi_test <- chisq.test(data,correct=TRUE,rescale.p = TRUE)
```

```
# 输出结果
print(fisher_test)
print(chi_test)
#检验结果判别
if(fisher_test$p.value>1-alpha)
{
  print("接受H0，拒绝H1")

  print("样本不存在相关性")
}else
{
  print("拒绝H0，接受H1")
  print("样本存在相关性")
}
```

```
#例6-8
# 输入数据
data <- matrix(c(20,43,21,16), nrow = 2, byrow = TRUE,
               dimnames = list(c("有效", "无效"), c("安慰剂", "新药")))
alpha=0.95
print(data)
# Fisher和卡方检验
fisher_test <- fisher.test(data,conf.level = alpha)
chi_test <- chisq.test(data,correct=TRUE,rescale.p = TRUE)
mcnemar_test<-mcnemar.test(data,correct=FALSE)
# 输出结果
print(fisher_test)
print(chi_test)
print(mcnemar_test)
```

```
#例6-9
# 输入数据
library(DescTools)
library(PropCIs)
data <- matrix(c(41,8,15,11), nrow =2, byrow = TRUE,
               dimnames = list(c("真手术", "假手术"),c("成功", "失败")))
alpha=0.95
#φ相关系数
phi_coef <- Phi(data)
print("φ相关系数为：")
print(phi_coef)
#列联相关系数C
c_coef <- ContCoef(data)
cat("列联系数 C:", c_coef,"\n")
#V相关系数
cramer_v <- CramerV(data)
print("V相关系数为：")
print(cramer_v)
```

```r
#wald风格的置信区间
prop.test(data, conf.level = 0.95, correct = TRUE)
# 计算 Agresti-Caffo 风格的置信区间
x1=41
n1=49
x2=15
n2=26
result <- diffscoreci(x1,n1,x2,n2,conf.level = alpha)

print(result)
# 输出结果
cat("比例差估计:", (x1/n1) - (x2/n2), "\n",
    "95% 置信区间: [", result$conf.int[1], ",", result$conf.int[2], "]")
```

```r
#例6-10
# 输入数据
library(DescTools)
library(PropCIs)
data <- matrix(c(89,37,6063,5711), nrow =2, byrow = TRUE,
               dimnames = list(c("是", "否"),c("吸烟者", "已戒烟者")))
alpha=0.95
#ϕ相关系数
phi_coef <- Phi(data)
print("ϕ相关系数为：")
print(phi_coef)
#列联相关系数C
c_coef <- ContCoef(data)
cat("列联系数 C:", c_coef,"\n")
#V相关系数
cramer_v <- CramerV(data)
print("V相关系数为：")
print(cramer_v)
chi_test <- chisq.test(data,correct=TRUE,rescale.p = TRUE)
print(chi_test)
```

```r
#例7-1
# 1. 构造观测矩阵
#    每列为一个处理组（group1-group4），每组 5 个观测值
mat <- matrix(c(
  18.1,18.6,18.7,18.9,18.3,
  17.4,17.9,17.1,16.5,17.5,
  17.3,16.9,18.5,18.2,16.2,
  15.6,15.8,16.7,15.3,16.8
), nrow = 4,byrow = TRUE)

df <- data.frame(
  value = as.vector(mat),
  group = factor(rep(paste0("G", 1:4), times = 5))
)

print(df)
```

```r
# 3. Fligner-Killeen 检验（对非正态更鲁棒）
fligner_res <- fligner.test(value ~ group, data = df)
print(fligner_res)

# 2. Bartlett 检验（假设正态分布）
bartlett_res <- bartlett.test(value ~ group, data = df)
print(bartlett_res)

library(car)
leveneTest(value ~ group, data = df)

# 执行单因素方差分析
model <- aov(value ~ group, data = df)
summary(model)

library(agricolae)

# 使用 LSD.test 进行多组均值比较
lsd_result <- LSD.test(model, "group",alpha=0.05,console = FALSE)

# 查看 LSD 检验结果
print(lsd_result)

# 使用 LSD.test 进行多组均值比较
lsd_result <- LSD.test(model, "group",alpha=0.01,console = FALSE)

# 查看 LSD 检验结果
print(lsd_result)

# 使用 SNK 检验（基于 SSR 分布）
snk_result <- SNK.test(model, "group", alpha = 0.05, console = FALSE)

# 查看结果（包括 LSR）
print(snk_result)

# 使用 SNK 检验（基于 q 分布）
snk_result <- SNK.test(model, "group", alpha = 0.01, console = FALSE)

# 查看结果（包括 LSR）
print(snk_result)

# Tukey HSD 多重比较<另一种SSR？>
hsd_result <- HSD.test(model,
                       trt = "group",    # 处理变量名
                       alpha = 0.05,     # 显著性水平
                       group = TRUE,     # 是否给出分组字母
                       console = FALSE)  # 在控制台打印结果
#查看结果
print(hsd_result)

# 使用 Duncan 检验（基于 Duncan法）
duncan_result <- duncan.test(model, "group", alpha = 0.05, console = FALSE)
```

```
#查看结果
print(duncan_result)

# 使用 Duncan 检验（基于 Duncan法）
duncan_result <- duncan.test(model, "group", alpha = 0.01, console = FALSE)

#查看结果
print(duncan_result)
```

```
library(car)
library(lme4)
#例7.2
# 1. 构造矩阵
# 创建因子组合和重复观测
M <- factor(rep(c("东北", "内蒙古","河北","安徽","贵州"), each = 4))



# 响应变量（模拟数据，替换为你的观测值）
value <- c(
  32.0,32.8,31.2,30.4,
  29.2,27.4,26.3,26.7,
  25.5,26.1,25.8,26.7,
  23.3,25.1,25.1,25.5,
  22.3,22.5,22.9,23.7
)

# 构建数据框
df <- data.frame(M, value)

print(df)
# 6. 构建单因素 ANOVA 模型
model <- aov(value ~ M, data = df)

# 7. 输出方差分析表
print(summary(model))

library(agricolae)

# 使用 LSD.test 进行多组均值比较
lsd_result <- LSD.test(model, "M",alpha=0.05,console = FALSE)

# 查看 LSD 检验结果
print(lsd_result)

# 使用 LSD.test 进行多组均值比较
lsd_result <- LSD.test(model, "M",alpha=0.01,console = FALSE)

# 查看 LSD 检验结果
print(lsd_result)

# 使用 SNK 检验（基于 SSR 分布）
snk_result <- SNK.test(model, "M", alpha = 0.05, console = FALSE)
```

```
# 查看结果（包括 LSR）
print(snk_result)

# 使用 SNK 检验（基于 q 分布）
snk_result <- SNK.test(model, "M", alpha = 0.01, console = FALSE)

# 查看结果（包括 LSR）
print(snk_result)

# Tukey HSD 多重比较<另一种SSR？>
hsd_result <- HSD.test(model,
                       trt = "M",    # 处理变量名
                       alpha = 0.05,    # 显著性水平
                       group = TRUE,    # 是否给出分组字母
                       console = FALSE)  # 在控制台打印结果
#查看结果
print(hsd_result)

# 使用 Duncan 检验（基于 Duncan法）
duncan_result <- duncan.test(model, "M", alpha = 0.05, console = FALSE)

#查看结果
print(duncan_result)

# 使用 Duncan 检验（基于 Duncan法）
duncan_result <- duncan.test(model, "M", alpha = 0.01, console = FALSE)

#查看结果
print(duncan_result)
```

```
#例7.4
# 1. 构造矩阵
data <- matrix(c(
  13,14,14,
  12,12,13,
  3, 3, 3,
  10, 9,10,
  2, 5, 4
), nrow = 5, byrow = TRUE,
dimnames = list(
  c("M1","M2","M3","M4","M5"),
  c("H1","H2","H3")
))

# 2. 转换为数据框并保留行名
df <- as.data.frame(data)
df$M <- rownames(df)        # 把行名存到M
rownames(df) <- NULL        # 可选：清空行名
```

```r
# 3. 用 tidyr 把宽表转换成长表
library(tidyr)
df_long <- pivot_longer(
  df,
  cols = starts_with("H"),
  names_to = "H",
  values_to = "value"
)

# 4. 将 M 和 H 设为因子
df_long$M <- factor(df_long$M)
df_long$H <- factor(df_long$H)

# 5. 查看长表
print(df_long)
#       M  H value
# 1    M1 H1    13
# 2    M1 H2    14
# 3    M1 H3    14
# 4    M2 H1    12
# ...

# 6. 构建二因素 ANOVA 模型（含交互作用）
#若不考虑交互作用只考虑主效应则使用 M + H
model <- aov(value ~ M * H, data = df_long)

# 7. 输出方差分析表
print(summary(model))

# 8. 事后检验（Tukey HSD），如有需要
print(TukeyHSD(model))

# 9. 绘制交互作用图
interaction.plot(
  x.factor     = df_long$H,
  trace.factor= df_long$M,
  response     = df_long$value,
  fun          = mean,
  type         = "b",
  pch          = 1:length(levels(df_long$M)),
  xlab         = "H 水平",
  ylab         = "平均值",
  trace.label  = "M 水平"
)
```

```r
library(car)
library(lme4)
#例7.5
# 1. 构造矩阵
# 创建因子组合和重复观测
M <- factor(rep(c("5h/d", "10h/d","15h/d"), each = 3 * 4))  # 3 levels × 3 H × 4 repeats
```

```r
H <- factor(rep(rep(c("25", "30", "35"), each = 4), times = 3))  # 3 repeats for each

# 响应变量（模拟数据，替换为你的观测值）
value <- c(
  143,138,120,107,  101,100,80,83,  89,93,101,76,
  96,103,78,91,     79,61,83,59,    80,76,61,67,
  79,83,96,98,      60,71,78,64,    67,58,71,83
)

# 构建数据框
df <- data.frame(M, H, value)

print(df)
# 6. 构建二因素 ANOVA 模型（含交互作用）
#若不考虑交互作用只考虑主效应则使用 M + H
model <- aov(value ~ M * H, data = df)

# 7. 输出方差分析表
print(summary(model))

modelnew <- lmer(value ~ M * H + (1|M:H), data = df)

# 查看模型结果
print(summary(modelnew))

Anova(model, type = "II")
# 8. 事后检验（Tukey HSD），如有需要
print(TukeyHSD(model))

# 9. 绘制交互作用图
interaction.plot(
  x.factor     = df_long$H,
  trace.factor= df_long$M,
  response     = df_long$value,
  fun          = mean,
  type         = "b",
  pch          = 1:length(levels(df_long$M)),
  xlab         = "H 水平",
  ylab         = "平均值",
  trace.label = "M 水平"
)
```

```r
# 例7-7
library(car)
library(lme4)

# 1. 构造矩阵
# 创建因子组合和重复观测
A <- factor(rep(c("0", "0.05", "0.10", "0.15"), each = 3 * 2 * 2))
B <- factor(rep(rep(c("0", "0.025", "0.050"), each = 2 * 2), times = 4))
C <- factor(rep(rep(rep(c("12", "14"), each = 2), times = 3), times = 4))
```

```r
# 响应变量
value <- c(
  1.11, 0.97, 1.52, 1.45, 1.09, 0.99, 1.27, 1.22, 0.85, 1.21, 1.67, 1.24,
  1.30, 1.00, 1.55, 1.53, 1.03, 1.21, 1.24, 1.34, 1.12, 0.96, 1.76, 1.27,
  1.22, 1.13, 1.38, 1.08, 1.34, 1.41, 1.40, 1.21, 1.34, 1.19, 1.46, 1.39,
  1.19, 1.03, 0.80, 1.29, 1.36, 1.16, 1.42, 1.39, 1.46, 1.03, 1.62, 1.27
)

# 构建数据框
df <- data.frame(A, B, C, value)
print(df)

# 6. 构建三因素固定效应 ANOVA 模型（含交互作用）
# 若不考虑交互作用只考虑主效应则使用 A+B+C
model_fixed <- aov(value ~ A * B * C, data = df)

# 7. 输出固定效应方差分析表
cat("=== 固定效应模型 ANOVA 结果 ===\n")
print(summary(model_fixed))



# ---------------------------
# 以下为随机模型多因素方差分析
# ---------------------------

# 8. 构建三因素随机效应混合模型
#    假设将 A、B、C 及它们的交互作用都视为随机效应
#    lmer 中 (1|因子) 表示该因子为随机截距
#    依次包括主效应和交互效应：(1|A), (1|B), (1|C), (1|A:B), (1|A:C), (1|B:C), (1|A:B:C)
model_random <- lmer(value ~
                       (1 | A) +
                       (1 | B) +
                       (1 | C) +
                       (1 | A:B) +
                       (1 | A:C) +
                       (1 | B:C) +
                       (1 | A:B:C),
                     data = df,
                     REML = FALSE)  # 通常用 REML=FALSE 做模型比较



# 9. 输出随机效应模型的摘要
cat("\n=== 随机效应模型 (lmer) 摘要 ===\n")
print(summary(model_random))


# 10. 提取并查看各随机效应的方差分量
cat("\n=== 随机效应方差分量 (VarCorr) ===\n")
print(VarCorr(model_random), comp = c("Variance", "Std.Dev"))

# 11. 对随机效应模型进行似然比检验（LR test），与仅含常数项的模型比较
```

```r
#    首先拟合仅含随机截距（无任何效应）的“空模型”：
model_null <- lmer(value ~ 1 +
                    (1 | A) +
                    (1 | B) +
                    (1 | C) +
                    (1 | A:B) +
                    (1 | A:C) +
                    (1 | B:C) +
                    (1 | A:B:C),
                  data = df,
                  REML = FALSE)

# 这里的 null 模型与 model_random 同结构，若想检验某一效应是否显著，可逐步去掉相应随机项再做比较，相当于检验所有随机效应是否都为 0。
cat("\n=== 随机效应模型 LRT 检验 ===\n")
anova(model_null, model_random)



# 12. 若用 REML 方法估计方差分量（一般用于报告最终模型），可：
model_random_REML <- lmer(value ~
                           (1 | A) +
                           (1 | B) +
                           (1 | C) +
                           (1 | A:B) +
                           (1 | A:C) +
                           (1 | B:C) +
                           (1 | A:B:C),
                         data = df,
                         REML = TRUE)

cat("\n=== 随机效应模型 (REML 估计) 方差分量 ===\n")
print(VarCorr(model_random_REML), comp = c("Variance", "Std.Dev"))
```

```r
# 例 8.1 数据及模型
x <- c(11.8,14.7,15.6,16.8,17.1,18.8,19.5,20.4)
y <- c(30.1,17.3,16.7,13.6,11.9,10.7,8.3,6.7)
data <- data.frame(x, y)

model <- lm(y ~ x, data = data)

# 打印模型摘要和置信区间、ANOVA
summary(model)
confint(model)          # 斜率和截距的 95% 置信区间
anova_res <- anova(model)
print(anova_res)
r <- cor.test(data$x, data$y,conf.level = 0.95)
print(r)

# 1. 绘制散点和空白画布
plot(data$x, data$y,
```

```r
     main  = "带 95% 置信区间的线性回归",
     xlab  = "平均温度",
     ylab  = "历期天数",
     pch   = 19)

# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata   = data.frame(x = x.seq),
                 interval  = "confidence",
                 level     = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
)

# 6. 残差-拟合值诊断图
plot(model, which = 1,
     main = "残差-拟合值诊断图")


# 创建新数据框用于预测
new_data <- data.frame(x = c(15))

# 进行预测，返回预测值及置信区间
predict(model, newdata = new_data, interval = "confidence",level = 0.95)
predict(model, newdata = new_data, interval = "prediction",level = 0.95)
```

```r
# -- 例 8.1 原始数据 + 引入随机数据 x2 --
set.seed(123)
x  <- c(11.8, 14.7, 15.6, 16.8, 17.1, 18.8, 19.5, 20.4)
```

```r
y  <- c(30.1, 17.3, 16.7, 13.6, 11.9, 10.7,  8.3,  6.7)

# 在 [0,100] 区间内纯随机生成正数
x2 <- runif(length(x), min = 0, max = 100)

data <- data.frame(x, x2, y)

# -- 多元线性回归 y ~ x + x2 --
model_multi <- lm(y ~ x + x2, data = data)

# -- 输出回归结果 --
summary(model_multi)    # 系数估计、t 值、p 值等
confint(model_multi)    # 斜率和截距的 95% 置信区间
anova(model_multi)      # 回归的方差分析表

# -- 回归诊断图 --
par(mfrow = c(1, 2))
plot(model_multi, which = 1, main = "残差-拟合值图")
plot(model_multi, which = 2, main = "正态 Q-Q 图")
par(mfrow = c(1, 1))

# -- 部分残差图（可选，需要安装 car 包） --
# install.packages("car")
library(car)
avPlots(model_multi, ask = FALSE, main = "Added-Variable Plots")
```

```r
# -- 例 8.1 原始数据 + 引入随机数据 x3(双向回归) --
library(ggplot2)
library(car)
library(plotly)

# 数据准备
set.seed(123)
x  <- c(11.8, 14.7, 15.6, 16.8, 17.1, 18.8, 19.5, 20.4)
y  <- c(30.1, 17.3, 16.7, 13.6, 11.9, 10.7,  8.3,  6.7)
x3 <- rnorm(length(x), mean = 10, sd = 5)
data <- data.frame(x, x3, y)

# 逐步回归
model_null <- lm(y ~ 1, data = data)
model_full <- lm(y ~ x + x3, data = data)
model_stepwise <- step(model_null,
                       scope = list(lower = model_null, upper = model_full),
                       direction = "both", trace = TRUE)
summary(model_stepwise)

# 获取最终模型变量
final_vars <- names(coef(model_stepwise))[-1]

# ---------- 可视化部分 ---------- #
```

```r
# 1️⃣ 如果只包含一个变量，用 ggplot2 画拟合线 + CI
if (length(final_vars) == 1) {
  predictor <- final_vars[1]

  ggplot(data, aes_string(x = predictor, y = "y")) +
    geom_point(size = 3, color = "black") +
    geom_smooth(method = "lm", se = TRUE, color = "blue", fill = "lightblue") +
    labs(title = paste("逐步回归结果：y ~", predictor),
         x = predictor,
         y = "y") +
    theme_minimal()
}

# 2️⃣ 如果是两个变量，做三维可视化和交互式图
if (length(final_vars) == 2) {
  var1 <- final_vars[1]
  var2 <- final_vars[2]

  # 生成网格用于拟合面
  grid <- expand.grid(
    var1 = seq(min(data[[var1]]), max(data[[var1]]), length.out = 30),
    var2 = seq(min(data[[var2]]), max(data[[var2]]), length.out = 30)
  )
  names(grid) <- c(var1, var2)
  grid$y_pred <- predict(model_stepwise, newdata = grid)

  # 构建三维图
  plot_ly() %>%
    add_markers(data = data, x = ~get(var1), y = ~get(var2), z = ~y,
                marker = list(color = 'black'),
                name = "原始数据") %>%
    add_surface(x = unique(grid[[var1]]),
                y = unique(grid[[var2]]),
                z = matrix(grid$y_pred, nrow = 30, byrow = TRUE),
                showscale = FALSE,
                opacity = 0.6,
                colorscale = 'Blues',
                name = "拟合面") %>%
    layout(scene = list(
      xaxis = list(title = var1),
      yaxis = list(title = var2),
      zaxis = list(title = "y"),
      camera = list(eye = list(x = 1.2, y = 1.2, z = 0.8))
    ),
    title = paste("三维拟合面图：y ~", var1, "+", var2))
}
```

```r
#例9-1
x <- c(399,329,247,191,145,119,90)
y <- c(0.380,0.379,0.371,0.343,0.317,0.301,0.248)
z <- x * y
```

```r
data <- data.frame(x, y)
plot(data$x, data$y,
     xlab  = "玉米株重",
     ylab  = "经济系数",
     pch   = 19,
     main = "玉米株重-经济系数 散点及倒数拟合")

# 2. 用 nls 拟合 y =(a+b*x)/x
exp.mod <- nls(y ~ (a+b*x)/x,
               data  = data,
               start = list(a = 1, b = 0.1))

# 查看拟合系数
coef(exp.mod)
#     a           b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)


data <- data.frame(x, z)
model <- lm(z ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$z,
     xlab  = "玉米株重",
     ylab  = "经济系数",
     pch   = 19)
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata    = data.frame(x = x.seq),
                 interval   = "confidence",
                 level      = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
```

```r
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
)
```

```r
#例9-2
x <- c(21,23,25,27,29,32,35)
y <- c(7,11,21,24,66,115,325)
z <- log(y)
data <- data.frame(x, y)
# 1. 画散点图
plot(data$x, data$y,
     xlab = "温度",
     ylab = "产卵数",
     pch  = 19,
     main = "温度-产卵数 散点及指数拟合")

# 2. 用 nls 拟合 y = a * exp(b * x)
exp.mod <- nls(y ~ a * exp(b * x),
               data  = data,
               start = list(a = 1, b = 0.1))

# 查看拟合系数
coef(exp.mod)
#      a            b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)


data <- data.frame(x, z)
model <- lm(z ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$z,
     xlab  = "温度",
     ylab  = "产卵数",
     pch   = 19)
```

```r
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata   = data.frame(x = x.seq),
                 interval  = "confidence",
                 level     = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
)
print(model$coefficients)
coef(model)
```

```r
#例9-3
x <-
c(7.2,7.9,11.8,12.0,16.9,18.7,18.9,20.2,21.8,22.7,22.9,23.1,23.3,23.6,23.8,27.0,27.6,28.6,30
.7,31.4)
y <-
c(13.8,21.4,24.9,32.3,33.6,39.5,40.1,36.9,40.2,42.6,44.6,36.6,35.1,44.4,44.1,43.9,48.3,48.5,
46.3,50.4)

data <- data.frame(x, y)

# 1. 画散点图
plot(data$x, data$y,
     xlab = "温度",
     ylab = "产卵数",
     pch  = 19,
     main = "温度-产卵数 散点及指数拟合")

# 2. 用 nls 拟合 y = a+b*log10(x)
```

```r
exp.mod <- nls(y ~ a+b*log10(x),
               data  = data,
               start = list(a = 1, b = 0.1))
summary(exp.mod)
# 查看拟合系数
coef(exp.mod)
#     a           b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)

x <- log10(x)
data <- data.frame(x, y)
model <- lm(y ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$y,
     xlab  = "膜内最高温度",
     ylab  = "室外最高温度",
     pch   = 19)
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata   = data.frame(x = x.seq),
                 interval  = "confidence",
                 level     = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
```

```
       title  = "图例",
       bty    = "n"
)
print(model$coefficients)
coef(model)

modelaov <- aov(y ~ x, data = data)

# 7. 输出方差分析表
print(summary(modelaov))
```

```
#例9-3 exp 改
x <-
c(7.2,7.9,11.8,12.0,16.9,18.7,18.9,20.2,21.8,22.7,22.9,23.1,23.3,23.6,23.8,27.0,27.6,28.6,30
.7,31.4)
y <-
c(13.8,21.4,24.9,32.3,33.6,39.5,40.1,36.9,40.2,42.6,44.6,36.6,35.1,44.4,44.1,43.9,48.3,48.5,
46.3,50.4)

data <- data.frame(x, y)

# 1. 画散点图
plot(data$x, data$y,
     xlab = "温度",
     ylab = "产卵数",
     pch  = 19,
     main = "温度-产卵数 散点及指数拟合")

# 2. 用 nls 拟合 y = a * exp(b * x)
exp.mod <- nls(y ~ a * exp(b * x),
               data  = data,
               start = list(a = 1, b = 0.1))

# 查看拟合系数
coef(exp.mod)
#     a           b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)

z <- log(y)
data <- data.frame(x, z)
model <- lm(z ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$z,
```

```r
     xlab  = "温度",
     ylab  = "产卵数",
     pch   = 19)
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata   = data.frame(x = x.seq),
                 interval  = "confidence",
                 level     = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
)
```

```r
#例9-4
x <- c(12,15,19,25,32,35,38,41,46,49,58)
y <-
c(0.17430,0.11080,0.06340,0.05310,0.04155,0.04080,0.04020,0.03998,0.03762,0.03538,0.03533)

data <- data.frame(x, y)

# 1. 画散点图
plot(data$x, data$y,
     xlab = "温度",
     ylab = "产卵数",
     pch  = 19,
     main = "温度-产卵数 散点及指数拟合")

# 2. 用 nls 拟合 y = a+b*log10(x)
exp.mod <- nls(y ~ a*x^b,
               data  = data,
               start = list(a = 1.4, b = -0.96))
```

```r
summary(exp.mod)
# 查看拟合系数
coef(exp.mod)
#      a            b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)

x <- log10(x)
y<-log10(y)
data <- data.frame(x, y)
model <- lm(y ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$y,
     xlab  = "膜内最高温度",
     ylab  = "室外最高温度",
     pch   = 19)
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata   = data.frame(x = x.seq),
                 interval  = "confidence",
                 level     = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
```

```
)
print(model$coefficients)
coef(model)

modelaov <- aov(y ~ x, data = data)

# 7. 输出方差分析表
print(summary(modelaov))
```

```
#例9-4 模型 倒数改
x <- c(12,15,19,25,32,35,38,41,46,49,58)
y <-
c(0.17430,0.11080,0.06340,0.05310,0.04155,0.04080,0.04020,0.03998,0.03762,0.03538,0.03533)
z <- x * y
data <- data.frame(x, y)
plot(data$x, data$y,
     xlab  = "玉米株重",
     ylab  = "经济系数",
     pch   = 19,
     main = "玉米株重-经济系数 散点及倒数拟合")

# 2. 用 nls 拟合 y =(a+b*x)/x
exp.mod <- nls(y ~ (a+b*x)/x,
               data  = data,
               start = list(a = 1, b = 0.1))

# 查看拟合系数
coef(exp.mod)
#     a           b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)


data <- data.frame(x, z)
model <- lm(z ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$z,
     xlab  = "玉米株重",
     ylab  = "经济系数",
     pch   = 19)
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata   = data.frame(x = x.seq),
```

```r
                  interval  = "confidence",
                  level     = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
)

print(model$coefficients)
coef(model)

modelaov <- aov(y ~ x, data = data)

# 7. 输出方差分析表
print(summary(modelaov))
```

```r
#例9-4 模型 指数改
x <- c(12,15,19,25,32,35,38,41,46,49,58)
y <-
c(0.17430,0.11080,0.06340,0.05310,0.04155,0.04080,0.04020,0.03998,0.03762,0.03538,0.03533)
data <- data.frame(x, y)

# 1. 画散点图
plot(data$x, data$y,
     xlab = "温度",
     ylab = "产卵数",
     pch  = 19,
     main = "温度-产卵数 散点及指数拟合")

# 2. 用 nls 拟合 y = a+b*log10(x)
exp.mod <- nls(y ~ a+b*log(x),
               data  = data,
               start = list(a = 1, b = 0.1))
```

```r
summary(exp.mod)
# 查看拟合系数
coef(exp.mod)
#     a           b
#（比如 a=1.23, b=0.20，具体值根据你的数据而定）

# 3. 在散点图上加上拟合曲线
#  3.1 生成细分的 x 供预测之用
x.seq <- seq(min(x), max(x), length.out = 200)
#  3.2 用拟合模型预测对应的 y 值
y.pred <- predict(exp.mod, newdata = data.frame(x = x.seq))
#  3.3 绘制曲线
lines(x.seq, y.pred, col = "red", lwd = 2)

x <- log(x)
data <- data.frame(x, y)
model <- lm(y ~ x, data = data)
summary(model)
confint(model)
plot(data$x, data$y,
     xlab  = "膜内最高温度",
     ylab  = "室外最高温度",
     pch   = 19)
# 2. 生成平滑的 x 序列，计算预测值和置信区间
x.seq <- seq(min(data$x), max(data$x), length.out = 100)
pred  <- predict(model,
                 newdata    = data.frame(x = x.seq),
                 interval   = "confidence",
                 level      = 0.95)

# 3. 绘制置信带（半透明多边形）
polygon(
  x      = c(x.seq, rev(x.seq)),
  y      = c(pred[,"lwr"], rev(pred[,"upr"])),
  col    = rgb(0, 0, 1, 0.2),
  border = NA
)

# 4. 添加回归拟合线
lines(x.seq, pred[,"fit"], col = "blue", lwd = 2, lty = 1)

# 5. 添加图例（包含置信区间说明）
legend("topright",
       legend = c("回归线", "95% 置信区间"),
       col    = c("blue", rgb(0, 0, 1, 0.2)),
       lty    = c(1, NA),
       lwd    = c(2, NA),
       pch    = c(NA, 15),
       pt.cex = c(NA, 2),
       pt.bg  = c(NA, rgb(0, 0, 1, 0.2)),
       title  = "图例",
       bty    = "n"
)
```

```r
print(model$coefficients)
coef(model)

modelaov <- aov(y ~ x, data = data)

# 7. 输出方差分析表
print(summary(modelaov))
```

```r
#例9-5
# 1. 准备数据
x <- c(2, 4, 6, 8, 10, 12, 14)
y <- c(0.30, 0.86, 1.73, 2.20, 2.47, 2.67, 2.80)
data <- data.frame(x, y)

# 2. 按“等间距”原则选取 y1, y2, y3:
n     <- length(y)
i1    <- 1                      # 第一组
i3    <- n                      # 最后一组
i2    <- floor((i1 + i3) / 2)   # 中间组
y1    <- y[i1]
y2    <- y[i2]
y3    <- y[i3]

# 3. 用三点公式计算 K
K_est <- ( y2^2 * (y1 + y3) - 2 * y1 * y2 * y3 ) / ( y2^2 - y1 * y3 )

# 4. 线性化变换: y' = ln((K - y) / y)
data$y_prime <- log((K_est - data$y) / data$y)

# 5. 线性回归 y' ~ x
lin.mod     <- lm(y_prime ~ x, data = data)
a_prime     <- coef(lin.mod)["(Intercept)"]
b_prime     <- coef(lin.mod)["x"]

# 6. 还原原始参数 a, b
a_est <- exp(a_prime)
b_est <- -b_prime

cat("估计参数: \n",
    "K =", round(K_est, 4), "\n",
    "a =", round(a_est, 4), "\n",
    "b =", round(b_est, 4), "\n")

# 7. 绘制散点和拟合曲线
plot(data$x, data$y,
     xlab = "x",
     ylab = "y",
     pch  = 19,
     main = "线性化 Logistic 拟合 (K 由等间距三点公式计算)")

x.seq <- seq(min(x), max(x), length.out = 200)
y.pred <- K_est / (1 + a_est * exp(-b_est * x.seq))
```

```r
lines(x.seq, y.pred, col = "red", lwd = 2)
legend("bottomright",
       legend = c("观测点", "线性化拟合曲线"),
       pch    = c(19, NA),
       lty    = c(NA, 1),
       col    = c("black", "red"),
       lwd    = c(NA, 2))
```

```r
#协方差test1
# —— 0. 加载包 & 设置对比方式 ——
# 安装（如有必要）: install.packages(c("car"))
library(car)
# 显式指定默认对比（Treatment 对比）
options(contrasts = c("contr.treatment", "contr.poly"))

# —— 1. 构造原始“长”表格数据 ——
A <- factor(rep(c("A1","A2","A3"), each = 2*8))
B <- factor(rep(rep(c("x","y"), each = 8), times = 3))
value <- c(
  18,16,11,14,14,13,17,17,
  85,89,65,80,78,83,91,85,
  17,18,18,19,21,21,16,22,
  95,100,94,98,104,97,90,106,
  18,23,23,20,24,25,25,26,
  91,89,98,82,100,98,102,108
)
df <- data.frame(A, B, value)

# —— 2. 重整为“每头猪一行” ——
df$pigID <- rep(1:8, times = 3, each = 2)
df2 <- reshape(df,
               idvar     = c("A","pigID"),
               timevar   = "B",
               direction = "wide")
names(df2)[names(df2) == "value.x"] <- "x"
names(df2)[names(df2) == "value.y"] <- "y"

# —— 3. 因子声明 ——
# 确保 A 是三水平因子
df2$A <- factor(df2$A, levels = c("A1","A2","A3"))

# —— 4. 协方差分析（ANCOVA） ——
# 强制在公式里用 factor(A)
anc <- aov(y ~ x + factor(A), data = df2)

cat("—— ANCOVA Type I SS ——\n")
print(summary(anc))

cat("\n—— ANCOVA Type III SS ——\n")
print(Anova(anc, type="III"))
```

```
# -- 5. 单因素方差分析（ANOVA） --
oneway <- aov(y ~ factor(A), data = df2)
cat("\n-- One-way ANOVA --\n")
print(summary(oneway))
```

```
#协方差test2
# 1. 构造数据
feed <- factor(rep(c("A1","A2","A3"), each=8))
x    <- c(18,16,11,14,14,13,17,17,
          17,18,18,19,21,21,16,22,
          18,23,23,20,24,25,25,26)
y    <- c(85,89,65,80,78,83,91,85,
          95,100,94,98,104,97,90,106,
          91,89,98,82,100,98,102,108)

df <- data.frame(feed, x, y)
print(df)
# 2. 一元方差分析 ANOVA
anova_mod <- aov(y ~ feed, data = df)
summary(anova_mod)                # F 值和 p 值

# 若要事后多重比较（Tukey HSD）
TukeyHSD(anova_mod, "feed")

# 3. 协方差分析 ANCOVA
ancova_mod <- aov(y ~ feed + x, data = df)
summary(ancova_mod)               # x（始重）与 feed（饲料）同时的显著性
str(ancova_mod)
# 4. 绘图检查模型假定（可选）
par(mfrow=c(2,2))
plot(anova_mod)                   # ANOVA 残差图
plot(ancova_mod)                  # ANCOVA 残差图

# 5. 校正后组均值（LS-means）（可选，需要加载 emmeans 包）
# install.packages("emmeans")
library(emmeans)
emm <- emmeans(ancova_mod, ~ feed, cov.reduce = mean)
summary(emm)                      # 给出校正初始体重后的各组预测均值
```

```
#协方差test3
# 1. 构造数据
feed <- factor(rep(c("A1","A2","A3"), each=8))
x    <- c(18,16,11,14,14,13,17,17,
          17,18,18,19,21,21,16,22,
          18,23,23,20,24,25,25,26)
y    <- c(85,89,65,80,78,83,91,85,
          95,100,94,98,104,97,90,106,
          91,89,98,82,100,98,102,108)
```

```
df <- data.frame(feed, x, y)

# 2. 一元方差分析 ANOVA
anova_mod <- aov(y ~ feed, data = df)
summary(anova_mod)                # F 值和 p 值
TukeyHSD(anova_mod, "feed")       # 事后多重比较

# 3. 计算 x 与 y 的协方差分析表（协方差矩阵）
cov_matrix <- cov(df[, c("x", "y")])
print("协方差矩阵 (x vs. y):")
print(cov_matrix)

# 如果还想做协方差检验（cov.test），可以参考：
# cov_test <- cov.test(df$x, df$y)  # 需要安装 psych 包：install.packages("psych")
# print(cov_test)

# 4. 协方差分析 ANCOVA
ancova_mod <- aov(y ~ feed+x, data = df)
summary(ancova_mod)               # x（始重）与 feed（饲料）同时的显著性

# 5. 绘图检查模型假定（可选）
par(mfrow=c(2,2))
plot(anova_mod)                   # ANOVA 残差图
plot(ancova_mod)                  # ANCOVA 残差图

# 6. 校正后组均值（LS-means）（可选，需要加载 emmeans 包）
# install.packages("emmeans")
library(emmeans)
emm <- emmeans(ancova_mod, ~ feed, cov.reduce = mean)
summary(emm)  # 给出校正初始体重后的各组预测均值
```

```
#协方差test4
# 1. 构造数据
feed <- factor(rep(c("A1","A2","A3"), each=8))
x    <- c(18,16,11,14,14,13,17,17,
          17,18,18,19,21,21,16,22,
          18,23,23,20,24,25,25,26)
y    <- c(85,89,65,80,78,83,91,85,
          95,100,94,98,104,97,90,106,
          91,89,98,82,100,98,102,108)

df <- data.frame(feed, x, y)
print(df)
# 2. 一元方差分析 ANOVA
anova_mod <- aov(y ~ feed, data = df)
summary(anova_mod)                # F 值和 p 值

# 若要事后多重比较（Tukey HSD）
TukeyHSD(anova_mod, "feed")

# 3. 协方差分析 ANCOVA
```

```r
ancova_mod <- lm(y ~ x+feed, data = df)
summary(ancova_mod)                # x（始重）与 feed（饲料）同时的显著性
anova(ancova_mod)
ancova_mod_null <- lm(y ~ 1, data = df)     # 仅截距的空模型
anova(ancova_mod_null, ancova_mod)           # 将 x1,x2,x3 三个一起看作“回归”来源
```

```r
#例11-2
# 1. 构造数据
chuli <- factor(rep(c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"),
each=2))
quzu<- factor(rep(rep(c("I","II"), each = 1), times = 14))
xy <- factor(rep(rep(rep(c("x", "y"), each = 1), times = 2),times= 14))
x <-
c(4.59,4.32,4.09,4.11,3.94,4.11,3.90,3.57,3.45,3.79,3.48,3.38,3.39,3.03,3.14,3.24,3.34,3.04,
4.12,4.76,4.12,4.75,3.84,3.60,3.96,4.50,3.03,3.01)
y <- c(58,61,65,62,64,64,66,69,71,67,71,72,71,74,72,69,69,69,61,54,63,56,67,62,64,60,75,71)

df <- data.frame(feed,quzu, x, y)
print(df)
# 2. 一元方差分析 ANOVA
anova_mod <- aov(y ~ chuli+quzu, data = df)
summary(anova_mod)                # F 值和 p 值

# 若要事后多重比较（Tukey HSD）
TukeyHSD(anova_mod, "chuli")

# 3. 协方差分析 ANCOVA
ancova_mod <- aov(lm(y ~ chuli+quzu, data = df))
summary(ancova_mod)                # x（始重）与 feed（饲料）同时的显著性
```

```r
# 例12-1
x1  <- c(2.80, 3.03, 3.17, 2.93, 2.42, 1.63, 2.90, 2.72)
x2  <- c(1.9, 4.6, 1.6, 7.8, 2.2, 5.2, 2.0, 1.4)
x3  <- c(32, 38, 18, 38, 27, 33, 29, 25)
y   <- c(2.56, 2.33, 3.35, 1.56, 2.25, 1.00, 3.10, 2.48)

data <- data.frame(x1, x2, x3, y)
print(data)

# -- 多元线性回归 y ~ x1 + x2 + x3 --
model_full <- lm(y ~ x1 + x2 + x3, data = data)
model_x3less <- lm(y ~ x1 + x2 , data = data)
model_x3x2less <- lm(y ~ x1 , data = data)
model_x3x1less <- lm(y ~ x2 , data = data)
anova(model_full)
# -- 输出常规模型结果 --

summary(model_full)    # 系数估计、t 值、p 值等
confint(model_full)    # 斜率和截距的 95% 置信区间
```

```r
# -- 方法一：构造仅含截距的“空模型”，再与“全模型”做嵌套比较 --
model_null <- lm(y ~ 1, data = data)    # 仅截距的空模型
anova(model_null, model_full)           # 将 x1,x2,x3 三个一起看作“回归”来源
AIC(model_full)
BIC(model_full)
library(ppcor)
pcor_all <- pcor(data)
# 查看偏相关系数矩阵
pcor_all$estimate

# 查看对应的 p 值矩阵
pcor_all$p.value

summary(model_x3less)    # 系数估计、t 值、p 值等
AIC(model_x3less)
BIC(model_x3less)
summary(model_x3x2less)    # 系数估计、t 值、p 值等
AIC(model_x3x2less)
BIC(model_x3x2less)
summary(model_x3x1less)    # 系数估计、t 值、p 值等
AIC(model_x3x1less)
BIC(model_x3x1less)
# 加载必要的包
library(ggplot2)

# 计算预测值
data$predicted <- predict(model_multi)

# 绘制散点图
ggplot(data, aes(x = predicted, y = y)) +
  geom_point(color = "blue", size = 3) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "预测值与实际值的比较",
       x = "预测值",
       y = "实际值") +
  theme_minimal()
```

```r
# 例13-1
x1  <- c(10,9,10,13,10,10,8,10,10,10,10,8,6,8,9)
x2  <- c(23,20,22,21,22,23,23,24,20,21,23,21,23,21,22)
x3  <- c(3.6,3.6,3.7,3.7,3.6,3.5,3.3,3.4,3.4,3.4,3.9,3.5,3.2,3.7,3.6)
x4 <- c(113,106,111,109,110,103,100,114,104,110,104,109,114,113,105)
y   <- c(15.7,14.5,17.5,22.5,15.5,16.9,8.6,17.0,13.7,13.4,20.3,10.2,7.4,11.6,12.3)

data <- data.frame(x1, x2, x3, x4,y)
print(data)

# -- 多元线性回归 y ~ x1 + x2 + x3 --
model_full <- lm(y ~ x1 + x2 + x3+x4, data = data)

anova(model_full)
```

```r
# -- 输出常规模型结果 --

summary(model_full)    # 系数估计、t 值、p 值等
confint(model_full)    # 斜率和截距的 95% 置信区间
model_less_backward1<-step(model_full,scope = list(lower = ~ 1,upper = ~ x1 + x2 + x3 +
x4),direction="backward",trace=2,steps = 1000,k = 2)
model_less_backward2<-step(model_full,scope = list(lower = ~ 1,upper = ~ x1 + x2 + x3 +
x4),direction="backward",trace=2,steps = 1000,k = log(nrow(data)))
model_less_both<-step(model_full,scope = list(lower = ~ 1,upper = ~ x1 + x2 + x3 +
x4),direction="both",trace=2,steps = 1000,k = 2)
#anova(model_less)
# -- 方法一：构造仅含截距的“空模型”，再与“全模型”做嵌套比较 --
model_null <- lm(y ~ 1, data = data)    # 仅截距的空模型
model_less_forward<-step(model_null,scope = list(lower = ~ 1,upper = ~ x1 + x2 + x3 +
x4),direction="forward",trace=2,steps = 1000,k = 2)
anova(model_null, model_full)          # 将 x1,x2,x3,x4 三个一起看作“回归”来源
AIC(model_full)
BIC(model_full)
library(ppcor)
pcor_all <- pcor(data)
# 查看偏相关系数矩阵
pcor_all$estimate



# 加载必要的包
library(ggplot2)

# 计算预测值
data$predicted <- predict(model_full)

# 绘制散点图
ggplot(data, aes(x = predicted, y = y)) +
  geom_point(color = "blue", size = 3) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "预测值与实际值的比较",
       x = "预测值",
       y = "实际值") +
  theme_minimal()



library(lm.beta)
fit.raw <- lm(y ~ x1 + x2 + x3, data = data)
fit.beta <- lm.beta(fit.raw)
summary(fit.beta)
# 2. 标准化（scale() 默认按每列处理：减均值、除以总体标准差）
data.std <- as.data.frame(scale(data))

# 3. 对标准化后的 y 做多元回归
fit.std <- lm(y ~ x1 + x2 + x3 , data = data.std)
summary(fit.std)
# 4. 提取标准化回归系数（去掉 intercept）
coef.std <- summary(fit.std)$coefficients
print(coef.std)
```

```
beta1 <- coef.std["x1","Estimate"]
beta2 <- coef.std["x2","Estimate"]
beta3 <- coef.std["x3","Estimate"]

# 3. 计算三个自变量之间的相关系数
r12 <- cor(data$x1, data$x2)
print(r12)
r13 <- cor(data$x1, data$x3)
print(r13)
r23 <- cor(data$x2, data$x3)
print(r23)

# 4. 计算自变量与因变量的零阶相关
r_yx1 <- cor(data$x1, data$y)
print(r_yx1)
r_yx2 <- cor(data$x2, data$y)
print(r_yx2)
r_yx3 <- cor(data$x3, data$y)
print(r_yx3)

# 5. 计算六个间接通径系数
#    例如 x1 通过 x2 的间接通径： r12 * β2
indirect_x1_via_x2 <- r12 * beta2
print(indirect_x1_via_x2)
indirect_x1_via_x3 <- r13 * beta3
print(indirect_x1_via_x3)
indirect_x2_via_x1 <- r12 * beta1
print(indirect_x2_via_x1)
indirect_x2_via_x3 <- r23 * beta3
print(indirect_x2_via_x3)
indirect_x3_via_x1 <- r13 * beta1
print(indirect_x3_via_x1)
indirect_x3_via_x2 <- r23 * beta2
print(indirect_x3_via_x2)
```

```
#例13-1new
# —— 准备数据 ——
x1  <- c(10,9,10,13,10,10,8,10,10,10,10,8,6,8,9)
x2  <- c(23,20,22,21,22,23,23,24,20,21,23,21,23,21,22)
x3  <- c(3.6,3.6,3.7,3.7,3.6,3.5,3.3,3.4,3.4,3.4,3.9,3.5,3.2,3.7,3.6)
x4  <- c(113,106,111,109,110,103,100,114,104,110,104,109,114,113,105)
y   <- c(15.7,14.5,17.5,22.5,15.5,16.9,8.6,17.0,13.7,13.4,20.3,10.2,7.4,11.6,12.3)

data <- data.frame(x1, x2, x3, x4, y)

# —— 加载 glmnet 包 ——
# install.packages("glmnet")  # 如未安装请取消注释
library(glmnet)

# —— 构建自变量矩阵 X 和响应变量 y ——
X <- as.matrix(data[, c("x1", "x2", "x3", "x4")])
```

```r
y <- data$y

# -- Lasso 回归交叉验证 --
lasso_cv <- cv.glmnet(X, y,family="gaussian",alpha = 1, standardize = TRUE)
print(lasso_cv)

# -- 最优 lambda --
best_lambda <- lasso_cv$lambda.min
cat("最优 lambda（使均方误差最小）：", best_lambda, "\n")

# -- 输出非零系数 --
lasso_coef <- coef(lasso_cv, s = "lambda.min")
print("非零系数：")
print(lasso_coef[lasso_coef != 0])

# -- 可选：计算 R² --
y_pred <- predict(lasso_cv, newx = X, s = "lambda.min")
rss <- sum((y - y_pred)^2)
tss <- sum((y - mean(y))^2)
rsq <- 1 - rss / tss
cat("Lasso 模型的 R²：", rsq, "\n")
fit_lasso <- glmnet(X, y,family="gaussian", alpha = 1)
plot(lasso_cv, xvar = "lambda",label = TRUE)
plot(fit_lasso, xvar = "lambda", label = TRUE)
```

```r
#先验后验
# 加载必要包
# 如果要使用 ggplot2 绘图，请先安装： install.packages("ggplot2")
library(ggplot2)

# -- 一元回归模拟设置 --
set.seed(0)
n       <- 100
a_true  <- 2.0
b_true  <- 3.5
sigma   <- 1.0
sigma_b <- 2.0  # 先验标准差

# 生成数据
x       <- runif(n, 0, 10)
epsilon <- rnorm(n, mean = 0, sd = sigma)
y       <- a_true + b_true * x + epsilon

# 去中心化
x_centered <- x - mean(x)
y_centered <- y - mean(y)

# 1. 经典 OLS 估计 (在去中心化后，无截距)
b_hat <- sum(x_centered * y_centered) / sum(x_centered^2)

# 2. 贝叶斯后验 (已知 sigma 和 sigma_b)
```

```r
# 后验精度 = (sum(x_c^2) / sigma^2) + (1 / sigma_b^2)
posterior_prec  <- sum(x_centered^2) / sigma^2 + 1 / sigma_b^2
posterior_var   <- 1 / posterior_prec
posterior_mean  <- (sum(x_centered * y_centered) / sigma^2) * posterior_var
posterior_sd    <- sqrt(posterior_var)

# 将一元结果整理到 data.frame
results_univariate <- data.frame(
  True_b          = b_true,
  OLS_b_hat       = b_hat,
  PosteriorMean   = posterior_mean,
  PosteriorSD     = posterior_sd
)

# 用 pipe 表格输出（Typora 支持这种表格语法）
# 如果想在 R Markdown 中渲染，可直接使用 knitr::kable()
cat("\n**一元回归结果（真实值、OLS 估计、后验均值、后验标准差）**\n\n")
cat("| True_b | OLS_b_hat | PosteriorMean | PosteriorSD |\n")
cat("|:------:|:---------:|:-------------:|:-----------:|\n")
cat(sprintf("| %.4f | %.4f | %.4f | %.4f |\n",
            results_univariate$True_b,
            results_univariate$OLS_b_hat,
            results_univariate$PosteriorMean,
            results_univariate$PosteriorSD))

# —— 多元回归模拟设置（p = 2） ——
p          <- 2
beta_true  <- c(1.5, -2.0)
X          <- matrix(rnorm(n * p), nrow = n, ncol = p)
epsilon_m  <- rnorm(n, mean = 0, sd = sigma)
y_multi    <- X %*% beta_true + epsilon_m   # 无截距情形

# 1. 经典 OLS 估计 (无截距)
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y_multi

# 2. 贝叶斯后验（独立先验 N(0, sigma_b^2)）
prior_prec_mat     <- (1 / sigma_b^2) * diag(p)     # 先验精度矩阵
posterior_prec_mat <- (t(X) %*% X) / sigma^2 + prior_prec_mat
posterior_cov_mat  <- solve(posterior_prec_mat)
posterior_mean_vec <- posterior_cov_mat %*% (t(X) %*% y_multi) / sigma^2
posterior_sd_vec   <- sqrt(diag(posterior_cov_mat))

# 将多元结果整理到 data.frame
results_multivariate <- data.frame(
  Coefficient     = c("beta1", "beta2"),
  True            = beta_true,
  OLS_Estimate    = as.numeric(beta_hat),
  PosteriorMean   = as.numeric(posterior_mean_vec),
  PosteriorSD     = posterior_sd_vec
)

# 输出多元表格
cat("\n**多元回归结果（真实值、OLS 估计、后验均值、后验标准差）**\n\n")
```

```r
cat("| Coefficient | True  | OLS_Estimate | PosteriorMean | PosteriorSD |\n")
cat("|:-----------:|:-----:|:------------:|:-------------:|:-----------:|\n")
for(i in 1:nrow(results_multivariate)) {
  cat(sprintf("| %s | %.4f | %.4f | %.4f | %.4f |\n",
              results_multivariate$Coefficient[i],
              results_multivariate$True[i],
              results_multivariate$OLS_Estimate[i],
              results_multivariate$PosteriorMean[i],
              results_multivariate$PosteriorSD[i]))
}

# -- 一元回归后验分布可视化 --
b_range      <- seq(b_hat - 3 * posterior_sd, b_hat + 3 * posterior_sd, length.out = 200)
posterior_df <- data.frame(
  b       = b_range,
  density = dnorm(b_range, mean = posterior_mean, sd = posterior_sd)
)

# 用 ggplot2 绘图
ggplot(posterior_df, aes(x = b, y = density)) +
  geom_line(size = 1) +
  geom_vline(xintercept = b_true, linetype = "dashed", color = "red", size = 0.8) +
  geom_vline(xintercept = b_hat, linetype = "dotted", color = "darkgreen", size = 0.8) +
  labs(
    title = "一元回归后验分布",
    x     = "b",
    y     = "密度"
  ) +
  theme_minimal(base_size = 12) +
  annotate("text", x = b_true + 0.1, y = max(posterior_df$density) * 0.9,
           label = "True b", color = "red") +
  annotate("text", x = b_hat - 0.1, y = max(posterior_df$density) * 0.8,
           label = "OLS b_hat", color = "darkgreen")
```