8.

# (a) 使用 rnorm() 函数生成长度为 n=100 的预测变量 X 和长度为 n=100 的噪声向量 ε

set.seed(1) # 为了结果可重现性设置随机种子

n <- 100

X <- rnorm(n)

epsilon <- rnorm(n)

# (b) 依据以下模型产生长度为 n=100 的响应变量 Y：

# 定义beta常数

beta0 <- 1

beta1 <- 2

beta2 <- 0.5

beta3 <- -1

# 生成响应变量 Y

Y <- beta0 + beta1 \* X + beta2 \* X^2 + beta3 \* X^3 + epsilon

# (c) 利用 regsubsets() 函数对数据集使用最优子集选择法，从包含预测变量X,X2,...,X10X,X2,...,X10的模型中选出最优的模型

# 创建X的多项式项

X\_poly <- model.matrix(Y ~ poly(X, 10, raw = TRUE))[,-1] # 移除截距项

colnames(X\_poly) <- paste0("X", 1:10) # 给列命名

# 将X的多项式项和Y组合成数据框

data\_df <- data.frame(Y, X\_poly)

# 使用regsubsets进行最优子集选择

library(leaps)

regfit.full <- regsubsets(Y ~ ., data = data\_df, nvmax = 10) # nvmax指定最大变量数

# 查看选择结果的统计量

summary\_regfit <- summary(regfit.full)

# 根据Cp、BIC和调整R^2选择最优模型

# 1. Cp准则 (Mallows' Cp)

# Cp越小越好，通常接近p的模型是好的

which.min(summary\_regfit$cp)

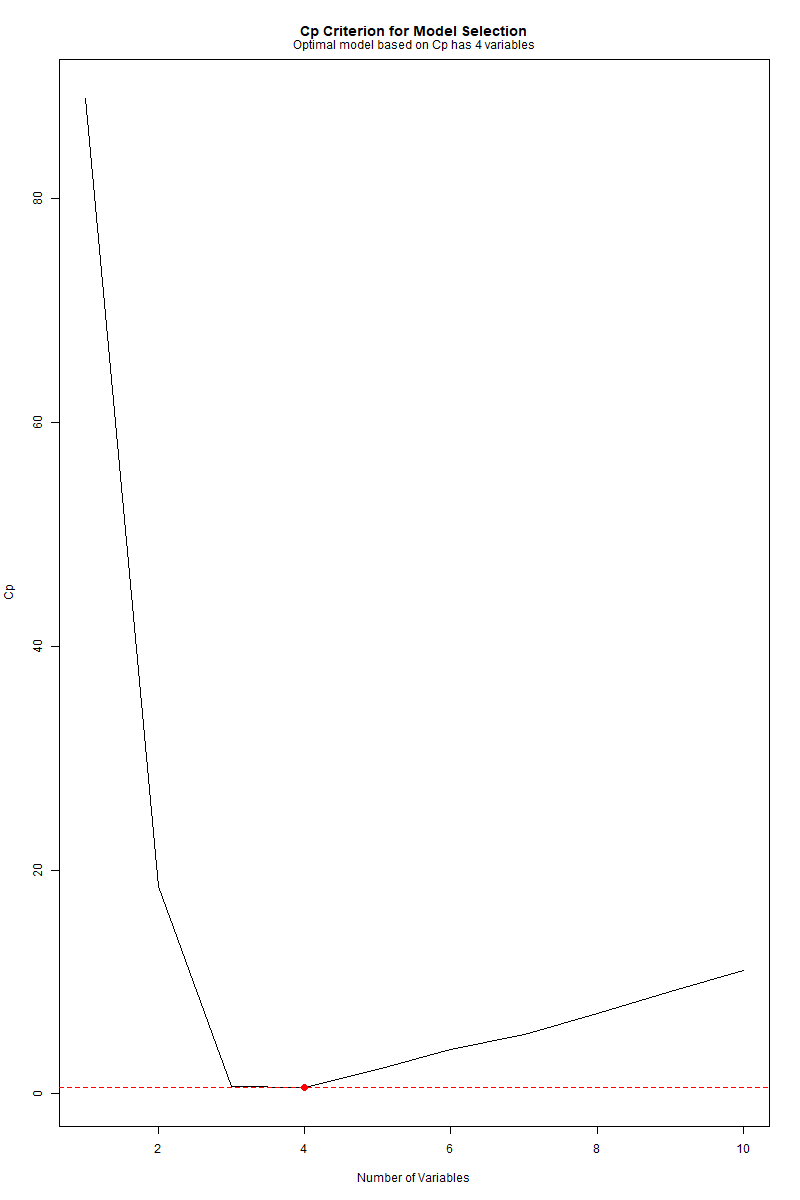
# 绘制Cp图

plot(summary\_regfit$cp, type = "l", xlab = "Number of Variables", ylab = "Cp", main = "Cp Criterion for Model Selection")

points(which.min(summary\_regfit$cp), summary\_regfit$cp[which.min(summary\_regfit$cp)], col = "red", cex = 2, pch = 20)

abline(h = min(summary\_regfit$cp), col = "red", lty = 2)

mtext(paste("Optimal model based on Cp has", which.min(summary\_regfit$cp), "variables"), side = 3, line = 0.5)



# 2. BIC准则 (Bayesian Information Criterion)

# BIC越小越好，惩罚更复杂的模型

which.min(summary\_regfit$bic)

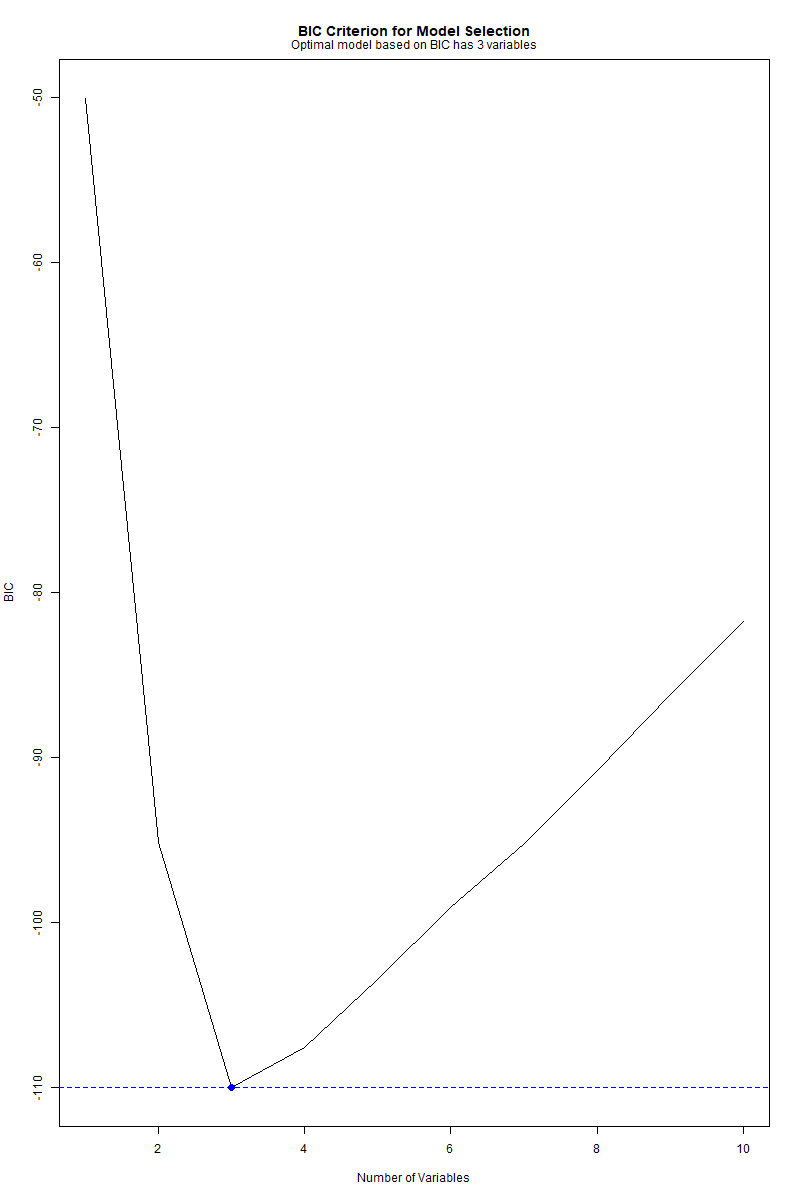
# 绘制BIC图

plot(summary\_regfit$bic, type = "l", xlab = "Number of Variables", ylab = "BIC", main = "BIC Criterion for Model Selection")

points(which.min(summary\_regfit$bic), summary\_regfit$bic[which.min(summary\_regfit$bic)], col = "blue", cex = 2, pch = 20)

abline(h = min(summary\_regfit$bic), col = "blue", lty = 2)

mtext(paste("Optimal model based on BIC has", which.min(summary\_regfit$bic), "variables"), side = 3, line = 0.5)



# 3. 调整R^2 (Adjusted R-squared)

# 调整R^2越大越好

which.max(summary\_regfit$adjr2)

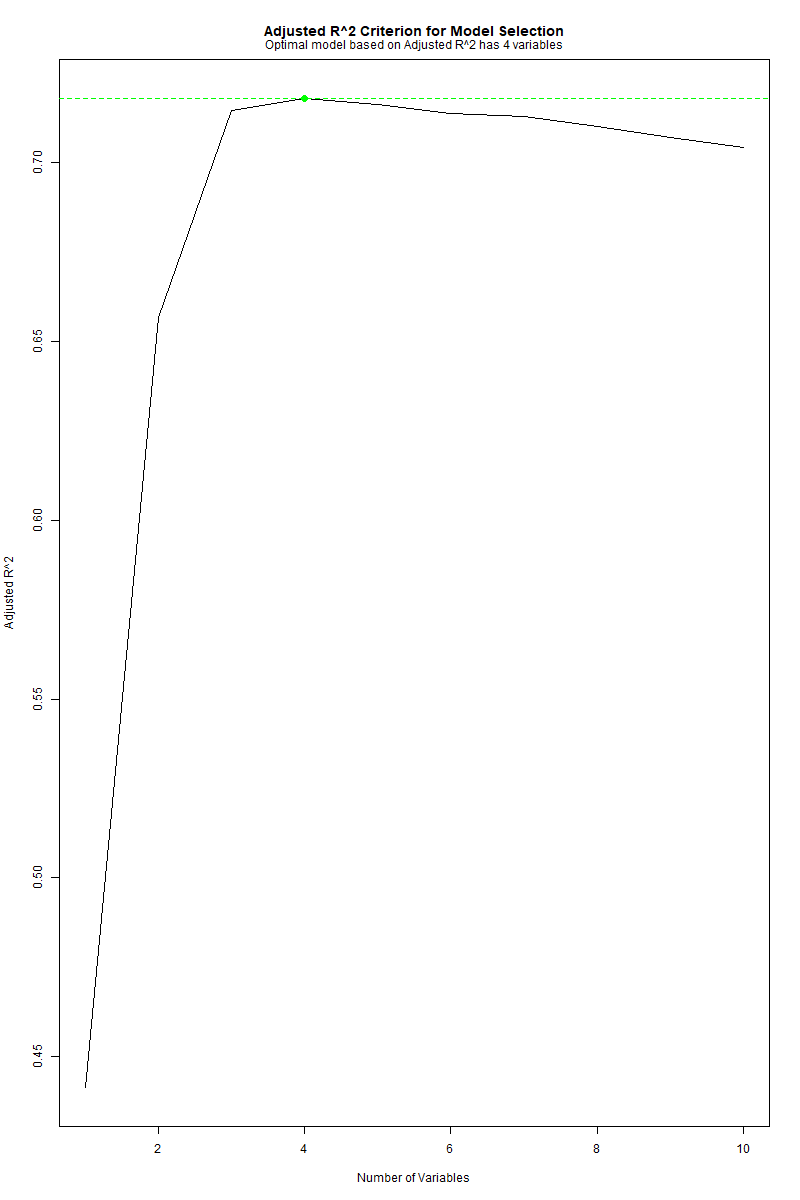
# 绘制调整R^2图

plot(summary\_regfit$adjr2, type = "l", xlab = "Number of Variables", ylab = "Adjusted R^2", main = "Adjusted R^2 Criterion for Model Selection")

points(which.max(summary\_regfit$adjr2), summary\_regfit$adjr2[which.max(summary\_regfit$adjr2)], col = "green", cex = 2, pch = 20)

abline(h = max(summary\_regfit$adjr2), col = "green", lty = 2)

mtext(paste("Optimal model based on Adjusted R^2 has", which.max(summary\_regfit$adjr2), "variables"), side = 3, line = 0.5)



# 获取基于Cp、BIC和Adjusted R^2选择的最优模型（假设都是3个变量）

# 例如，选择3个变量的模型

print(coef(regfit.full, 3))



# (d) 使用向前逐步选择法和向后逐步选择法重复 (c) 中的步骤。

regfit.fwd <- regsubsets(Y ~ ., data = data\_df, nvmax = 10, method = "forward")

summary\_regfit\_fwd <- summary(regfit.fwd)

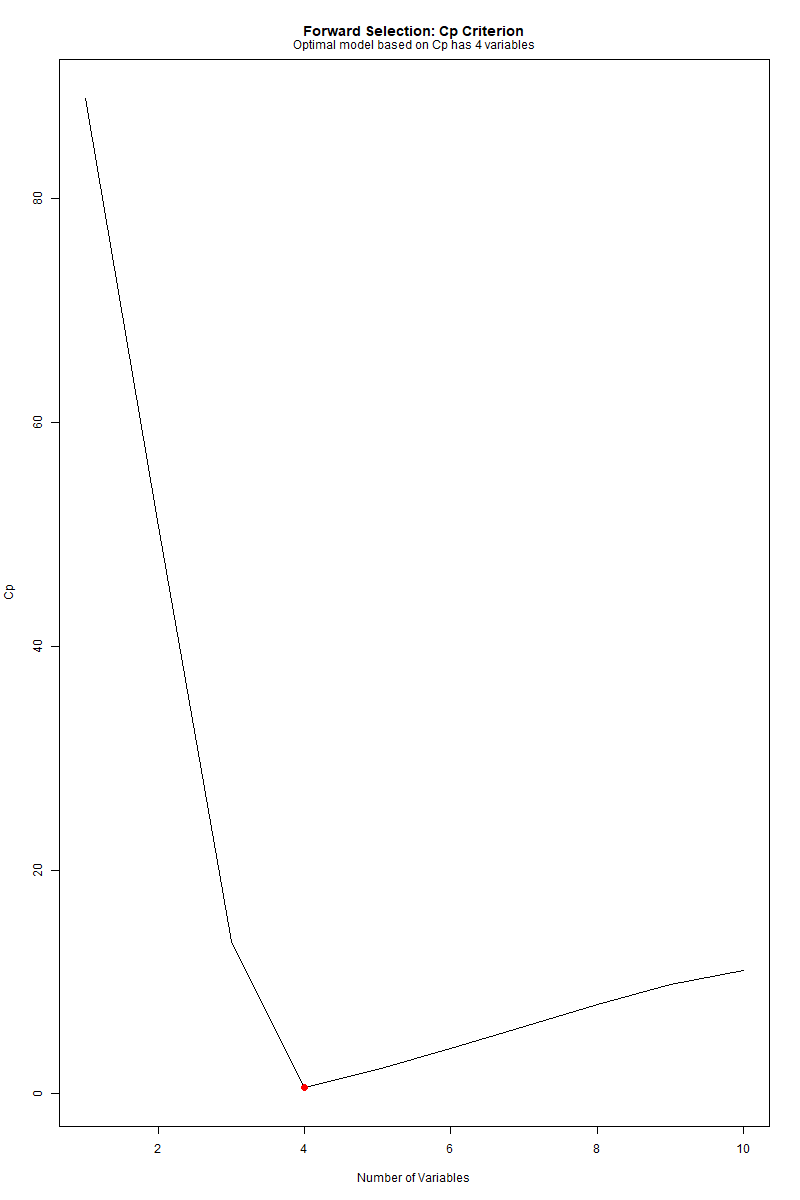
# 同样绘制Cp、BIC、调整R^2图并选择最优模型

# Cp for Forward

plot(summary\_regfit\_fwd$cp, type = "l", xlab = "Number of Variables", ylab = "Cp", main = "Forward Selection: Cp Criterion")

points(which.min(summary\_regfit\_fwd$cp), summary\_regfit\_fwd$cp[which.min(summary\_regfit\_fwd$cp)], col = "red", cex = 2, pch = 20)

mtext(paste("Optimal model based on Cp has", which.min(summary\_regfit\_fwd$cp), "variables"), side = 3, line = 0.5)

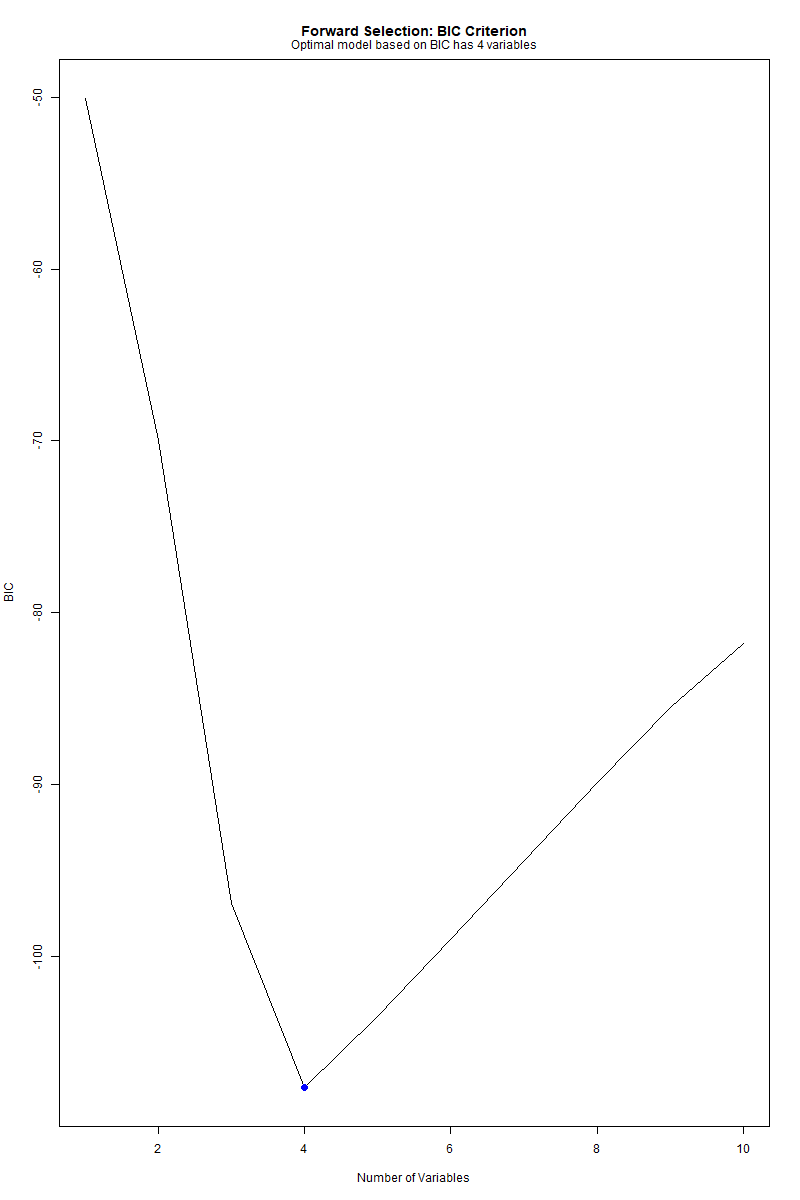


# BIC for Forward

plot(summary\_regfit\_fwd$bic, type = "l", xlab = "Number of Variables", ylab = "BIC", main = "Forward Selection: BIC Criterion")

points(which.min(summary\_regfit\_fwd$bic), summary\_regfit\_fwd$bic[which.min(summary\_regfit\_fwd$bic)], col = "blue", cex = 2, pch = 20)

mtext(paste("Optimal model based on BIC has", which.min(summary\_regfit\_fwd$bic), "variables"), side = 3, line = 0.5)

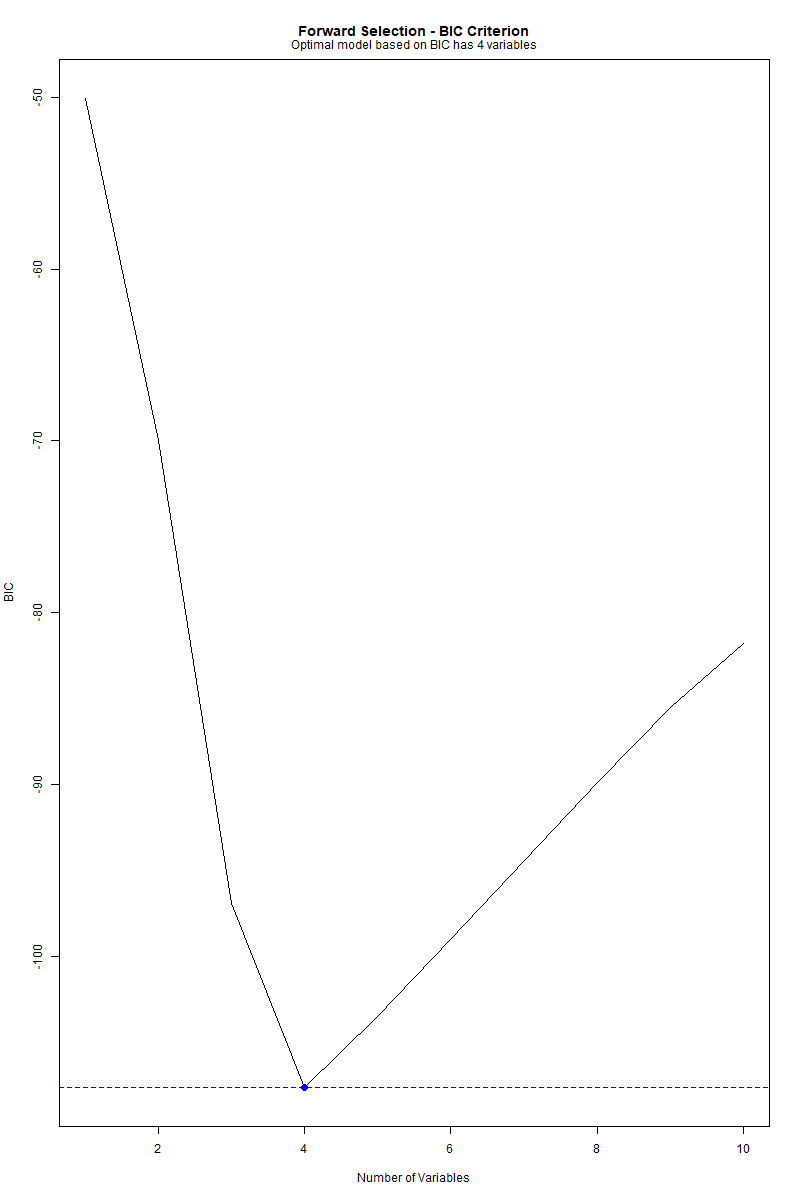


# Adjusted R^2 for Forward

plot(summary\_regfit\_fwd$adjr2, type = "l", xlab = "Number of Variables", ylab = "Adjusted R^2", main = "Forward Selection: Adjusted R^2 Criterion")

points(which.max(summary\_regfit\_fwd$adjr2), summary\_regfit\_fwd$adjr2[which.max(summary\_regfit\_fwd$adjr2)], col = "green", cex = 2, pch = 20)

mtext(paste("Optimal model based on Adjusted R^2 has", which.max(summary\_regfit\_fwd$adjr2), "variables"), side = 3, line = 0.5)



# 调整R^2

which.max(summary\_forward$adjr2)

plot(summary\_forward$adjr2, type = "l", xlab = "Number of Variables", ylab = "Adjusted R^2",

    main = "Forward Selection - Adjusted R^2 Criterion")

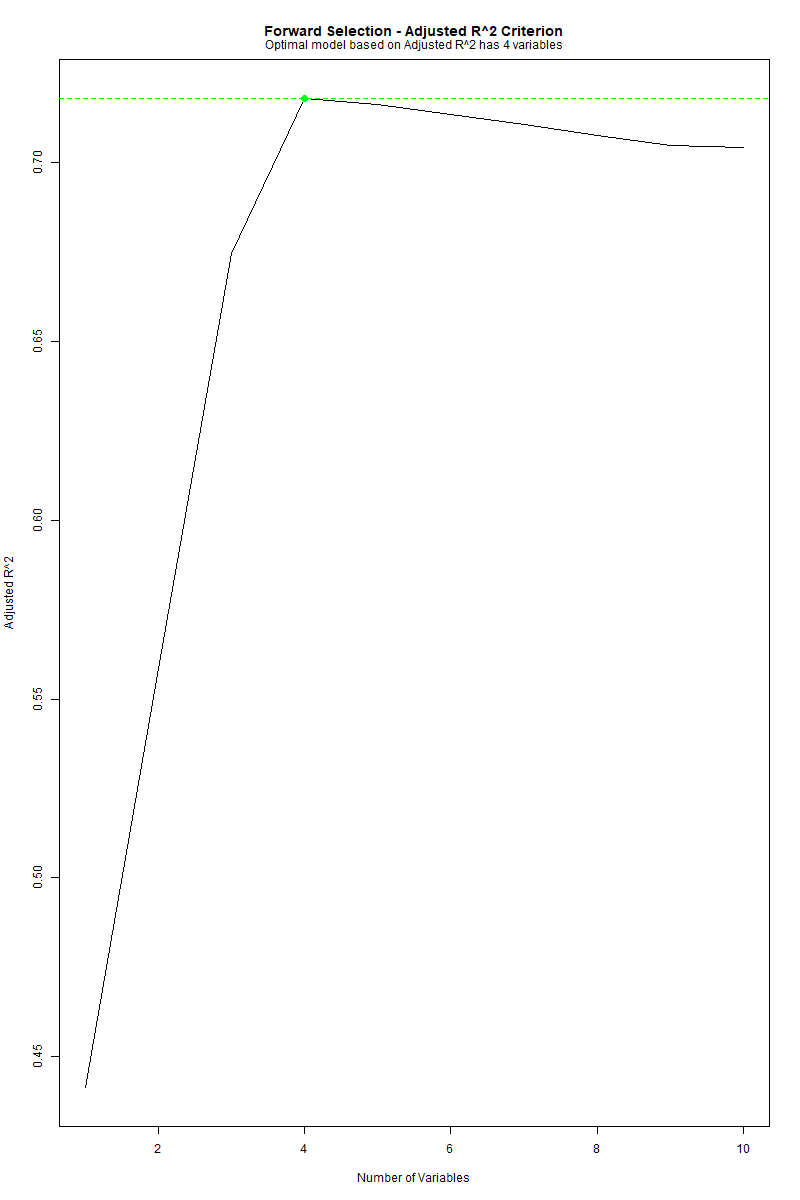
points(which.max(summary\_forward$adjr2), summary\_forward$adjr2[which.max(summary\_forward$adjr2)],

    col = "green", cex = 2, pch = 20)

abline(h = max(summary\_forward$adjr2), col = "green", lty = 2)

mtext(paste("Optimal model based on Adjusted R^2 has", which.max(summary\_forward$adjr2), "variables"),

    side = 3, line = 0.5)



# 输出最优模型系数（假设选择3个变量）

print(coef(regfit.forward, 3))



# 向后逐步选择 (Backward Selection)

regfit.backward <- regsubsets(Y ~ ., data = data\_df, nvmax = 10, method = "backward")

summary\_backward <- summary(regfit.backward)

# 基于Cp、BIC和调整R^2选择最优模型

# Cp准则

which.min(summary\_backward$cp)

plot(summary\_backward$cp, type = "l", xlab = "Number of Variables", ylab = "Cp",

    main = "Backward Selection - Cp Criterion")

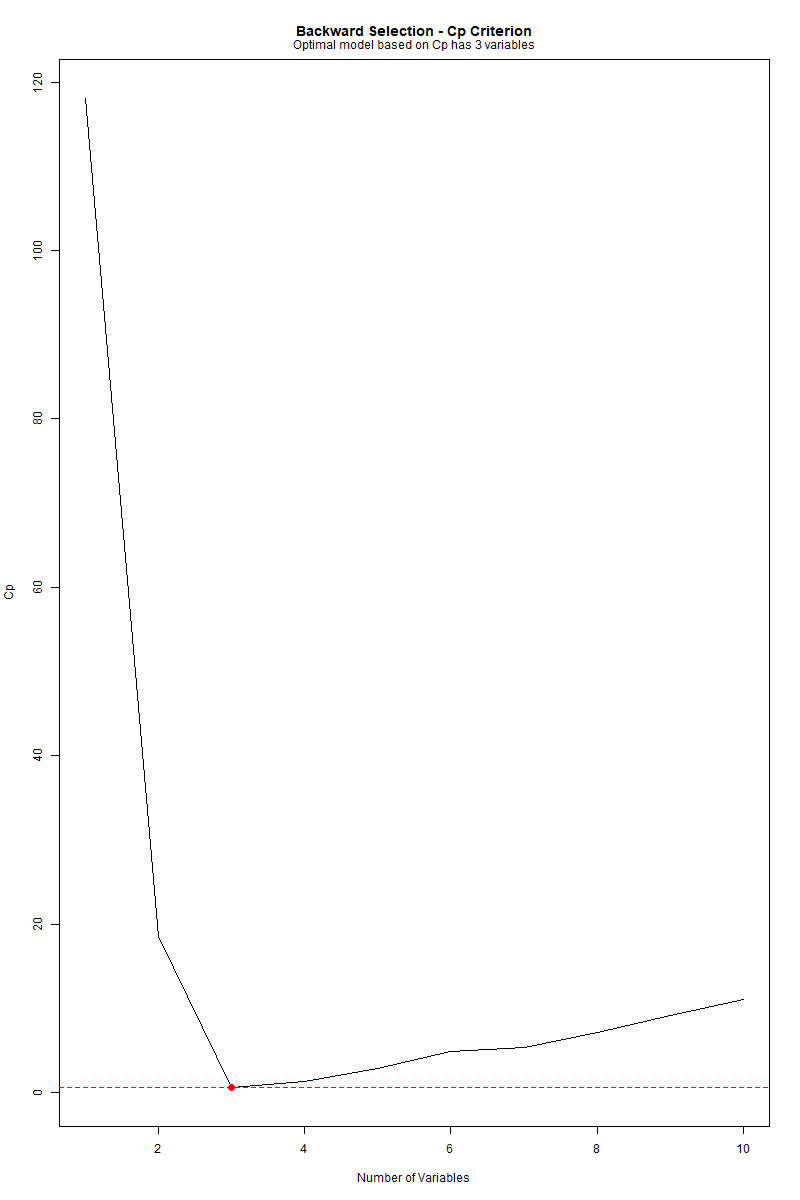
points(which.min(summary\_backward$cp), summary\_backward$cp[which.min(summary\_backward$cp)],

    col = "red", cex = 2, pch = 20)

abline(h = min(summary\_backward$cp), col = "red", lty = 2)

mtext(paste("Optimal model based on Cp has", which.min(summary\_backward$cp), "variables"),

    side = 3, line = 0.5)



# BIC准则

which.min(summary\_backward$bic)

plot(summary\_backward$bic, type = "l", xlab = "Number of Variables", ylab = "BIC",

    main = "Backward Selection - BIC Criterion")

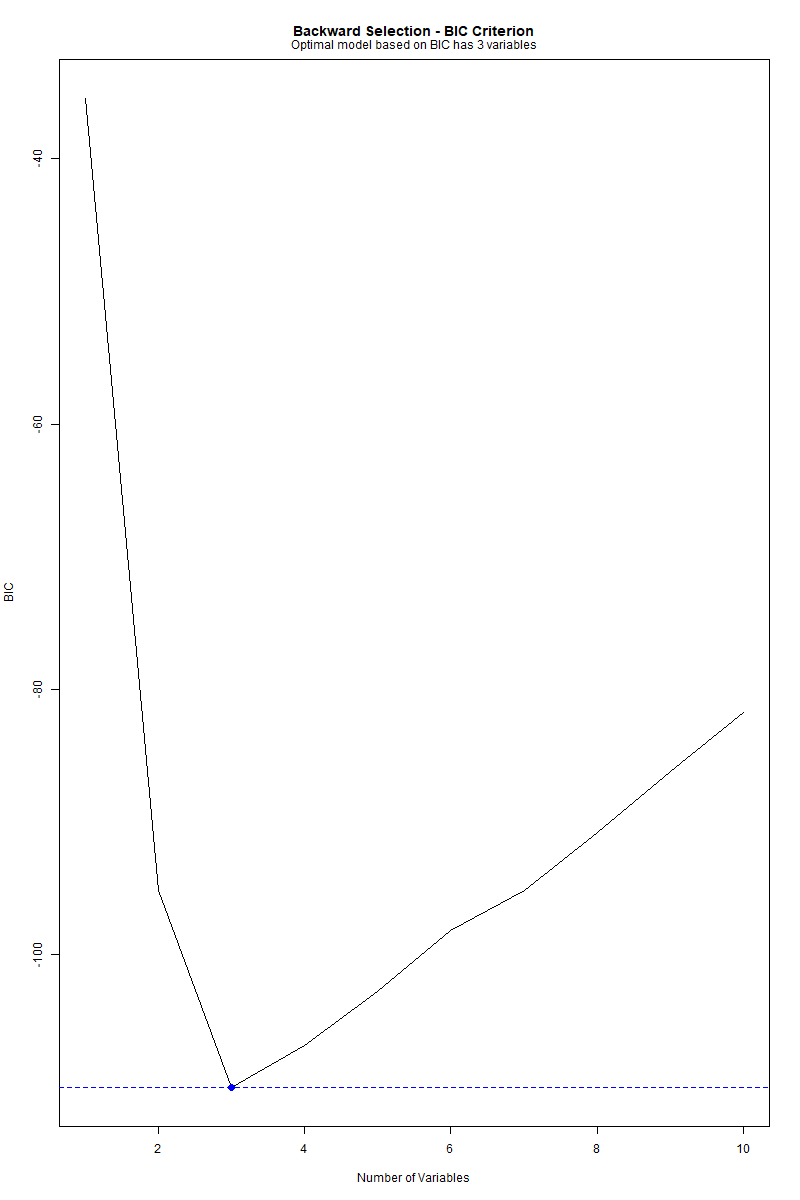
points(which.min(summary\_backward$bic), summary\_backward$bic[which.min(summary\_backward$bic)],

    col = "blue", cex = 2, pch = 20)

abline(h = min(summary\_backward$bic), col = "blue", lty = 2)

mtext(paste("Optimal model based on BIC has", which.min(summary\_backward$bic), "variables"),

    side = 3, line = 0.5)



# 调整R^2

which.max(summary\_backward$adjr2)

plot(summary\_backward$adjr2, type = "l", xlab = "Number of Variables", ylab = "Adjusted R^2",

    main = "Backward Selection - Adjusted R^2 Criterion")

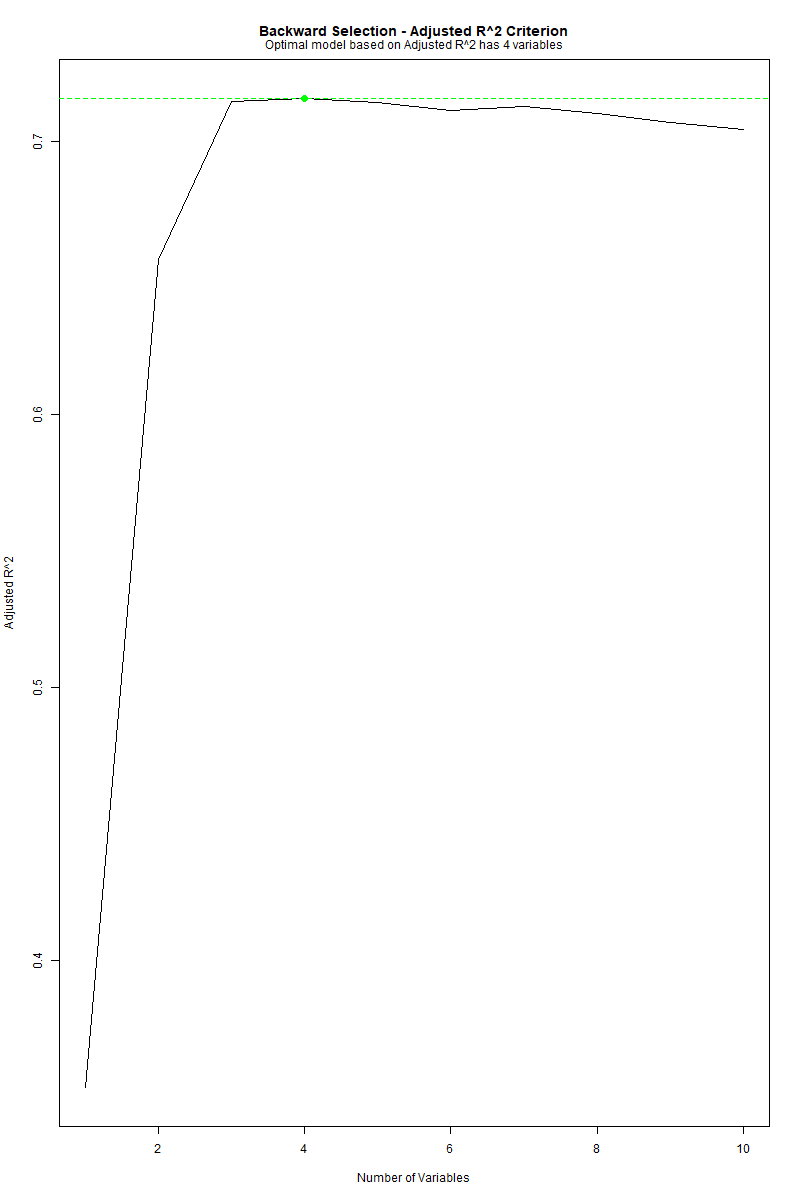
points(which.max(summary\_backward$adjr2), summary\_backward$adjr2[which.max(summary\_backward$adjr2)],

    col = "green", cex = 2, pch = 20)

abline(h = max(summary\_backward$adjr2), col = "green", lty = 2)

mtext(paste("Optimal model based on Adjusted R^2 has", which.max(summary\_backward$adjr2), "variables"),

    side = 3, line = 0.5)



# 输出最优模型系数（假设选择3个变量）

print(coef(regfit.backward, 3))



基本一致，只有系数有一定区别

# (e) lasso 拟合数据集

library(glmnet)

# 创建X的多项式项（包含X^1到X^10）

X\_poly <- model.matrix(Y ~ poly(X, 10, raw = TRUE))[,-1]  # 移除截距项

# 将X的多项式项转换为矩阵形式

X\_matrix <- as.matrix(X\_poly)

# 确保Y是向量

Y\_vector <- as.vector(Y)

# 执行交叉验证以选择最优的 lambda

cv\_lasso <- cv.glmnet(X\_matrix, Y\_vector, alpha = 1, nfolds = 10)

# 查看交叉验证结果

print(cv\_lasso)

# 绘制 CV 误差曲线

plot(cv\_lasso, main = "Cross-Validation Error vs Lambda for Lasso", xlab = "log(Lambda)", ylab = "CV Error")

abline(v = log(cv\_lasso$lambda.min), col = "red", lty = 2)

abline(v = log(cv\_lasso$lambda.1se), col = "blue", lty = 2)

mtext(paste("Optimal lambda (min):", round(cv\_lasso$lambda.min, 4)), side = 3, line = 0.5, col = "red")

mtext(paste("Lambda with 1SE:", round(cv\_lasso$lambda.1se, 4)), side = 3, line = 1.5, col = "blue")

# 使用最优 lambda 得到系数估计

lasso\_coef\_min <- coef(cv\_lasso, s = "lambda.min")

lasso\_coef\_1se <- coef(cv\_lasso, s = "lambda.1se")

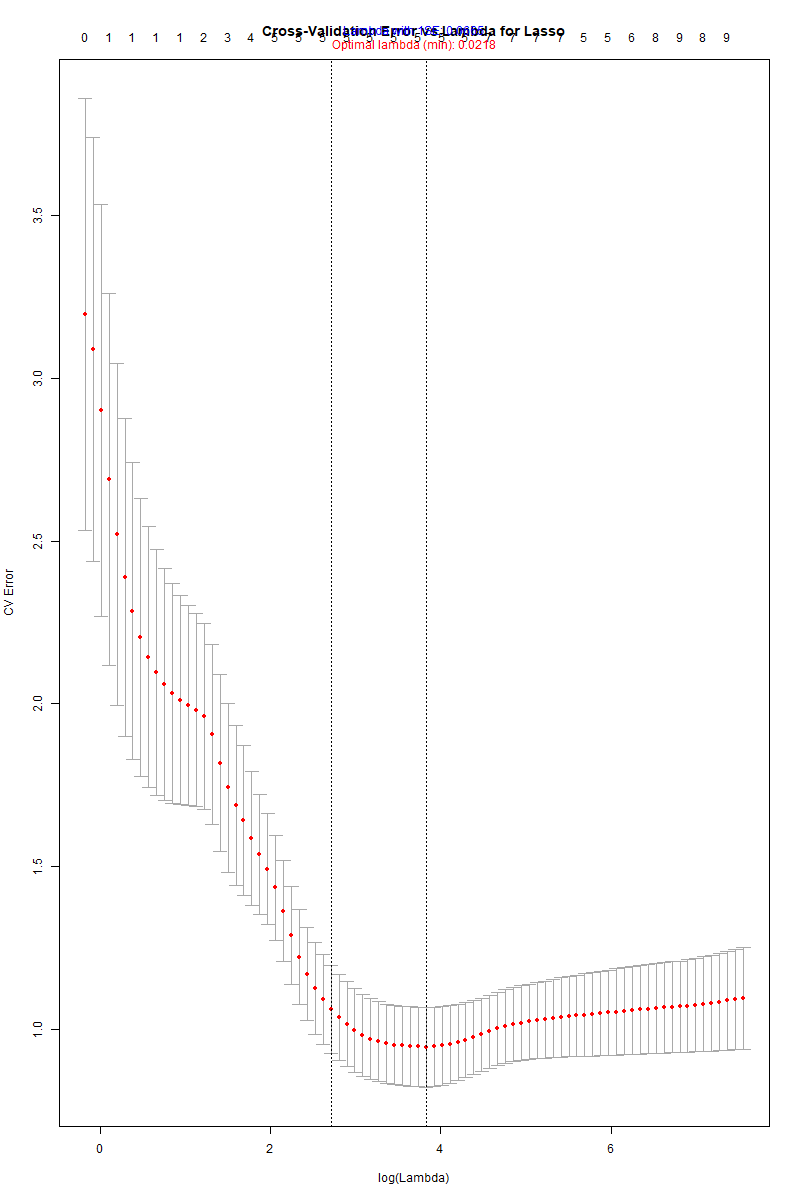
# 输出结果

print("Lasso Coefficients (lambda.min):")

print(lasso\_coef\_min)

print("\nLasso Coefficients (lambda.1se):")

print(lasso\_coef\_1se)



横轴：$\log(\lambda)$，表示正则化强度的对数值。

$\lambda$ 越小 → 正则化越弱 → 模型更复杂（可能过拟合）。

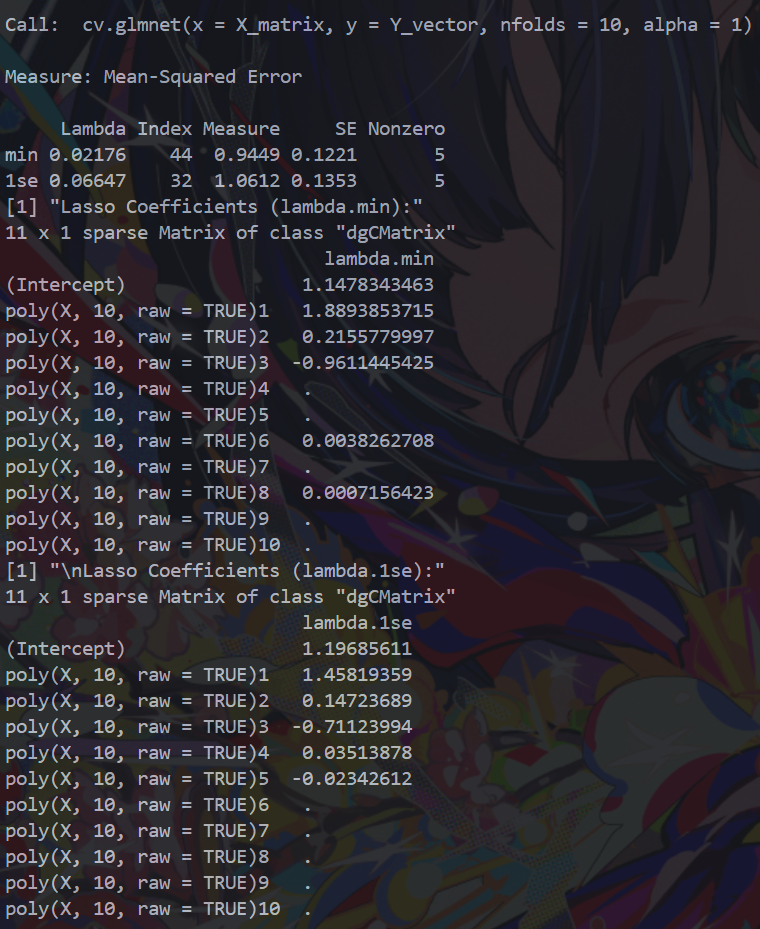
$\lambda$ 越大 → 正则化越强 → 模型更简单（可能欠拟合）。

纵轴：CV Error（交叉验证误差），即模型在验证集上的均方误差（MSE）。

误差越低 → 模型预测性能越好。

红色点：每个 $\lambda$ 对应的平均 CV 误差。

灰色竖线：表示每个 $\lambda$ 下 CV 误差的标准误（SE），反映估计的稳定性。



(1) lambda.min 对应的系数

仅前 3 项（X¹、X²、X³）和第 6、8 项有显著系数。

其他高阶项被压缩为零，说明它们对预测无贡献。

1. lambda.1se 对应的系数

更多高阶项被压缩为零，模型更加简洁。

主要保留了 X¹、X²、X³ 和 X⁴、X⁵ 的影响。

# (f) 依据新模型 Y = β0 + β1\*X^2 + ε 生成响应变量并进行模型选择

# 重新设置随机种子以确保结果可重现

set.seed(1)

n <- 100

X <- rnorm(n)

epsilon <- rnorm(n)

# 定义新的参数（仅含 X^2 项）

beta0 <- 1

beta1 <- 2

# 生成新的响应变量 Y

Y <- beta0 + beta1 \* X^2 + epsilon

# 创建X的多项式项（包含X^1到X^10）

X\_poly <- model.matrix(Y ~ poly(X, 10, raw = TRUE))[,-1]  # 移除截距项

colnames(X\_poly) <- paste0("X", 1:10)

# 将X的多项式项和Y组合成数据框

data\_df <- data.frame(Y, X\_poly)

# 使用最优子集选择法（regsubsets）

library(leaps)

regfit.full <- regsubsets(Y ~ ., data = data\_df, nvmax = 10)

# 查看选择结果的统计量

summary\_regfit <- summary(regfit.full)

# 根据Cp、BIC和调整R^2选择最优模型

# Cp准则

which.min(summary\_regfit$cp)

plot(summary\_regfit$cp, type = "l", xlab = "Number of Variables", ylab = "Cp",

     main = "Cp Criterion for Optimal Subset Selection")

points(which.min(summary\_regfit$cp), summary\_regfit$cp[which.min(summary\_regfit$cp)], col = "red", cex = 2, pch = 20)

abline(h = min(summary\_regfit$cp), col = "red", lty = 2)

mtext(paste("Optimal model based on Cp has", which.min(summary\_regfit$cp), "variables"), side = 3, line = 0.5)

# BIC准则

which.min(summary\_regfit$bic)

plot(summary\_regfit$bic, type = "l", xlab = "Number of Variables", ylab = "BIC",

     main = "BIC Criterion for Optimal Subset Selection")

points(which.min(summary\_regfit$bic), summary\_regfit$bic[which.min(summary\_regfit$bic)], col = "blue", cex = 2, pch = 20)

abline(h = min(summary\_regfit$bic), col = "blue", lty = 2)

mtext(paste("Optimal model based on BIC has", which.min(summary\_regfit$bic), "variables"), side = 3, line = 0.5)

# 调整R^2

which.max(summary\_regfit$adjr2)

plot(summary\_regfit$adjr2, type = "l", xlab = "Number of Variables", ylab = "Adjusted R^2",

     main = "Adjusted R^2 Criterion for Optimal Subset Selection")

points(which.max(summary\_regfit$adjr2), summary\_regfit$adjr2[which.max(summary\_regfit$adjr2)], col = "green", cex = 2, pch = 20)

abline(h = max(summary\_regfit$adjr2), col = "green", lty = 2)

mtext(paste("Optimal model based on Adjusted R^2 has", which.max(summary\_regfit$adjr2), "variables"), side = 3, line = 0.5)

# 输出最优模型系数（假设为2个变量：截距和X^2）

print("Optimal Subset Selection - Coefficients:")

print(coef(regfit.full, 2))

# 使用Lasso方法

library(glmnet)

X\_matrix <- as.matrix(X\_poly)

Y\_vector <- as.vector(Y)

# 执行交叉验证以选择最优的 lambda

cv\_lasso <- cv.glmnet(X\_matrix, Y\_vector, alpha = 1, nfolds = 10)

# 绘制CV误差曲线

plot(cv\_lasso, main = "Cross-Validation Error vs Lambda for Lasso", xlab = "log(Lambda)", ylab = "CV Error")

abline(v = log(cv\_lasso$lambda.min), col = "red", lty = 2)

abline(v = log(cv\_lasso$lambda.1se), col = "blue", lty = 2)

mtext(paste("Optimal lambda (min):", round(cv\_lasso$lambda.min, 4)), side = 3, line = 0.5, col = "red")

mtext(paste("Lambda with 1SE:", round(cv\_lasso$lambda.1se, 4)), side = 3, line = 1.5, col = "blue")

# 获取最优模型系数

lasso\_coef\_min <- coef(cv\_lasso, s = "lambda.min")

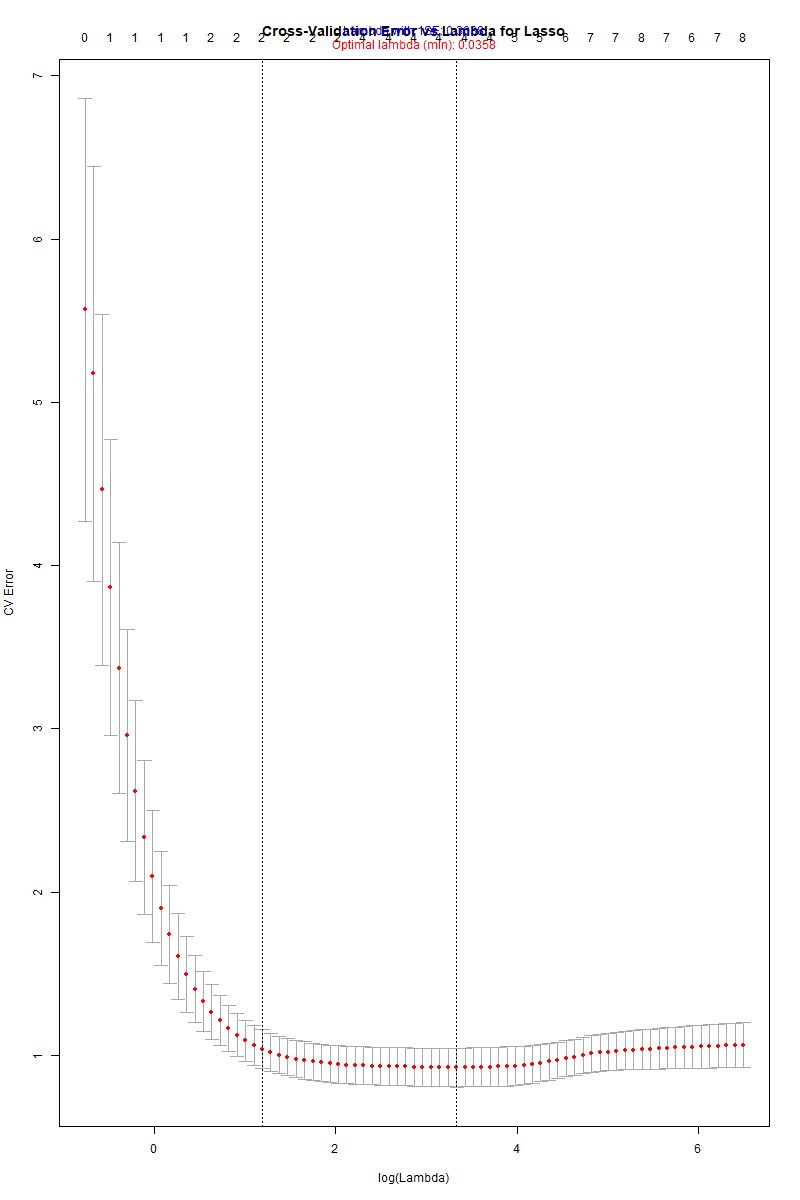
lasso\_coef\_1se <- coef(cv\_lasso, s = "lambda.1se")

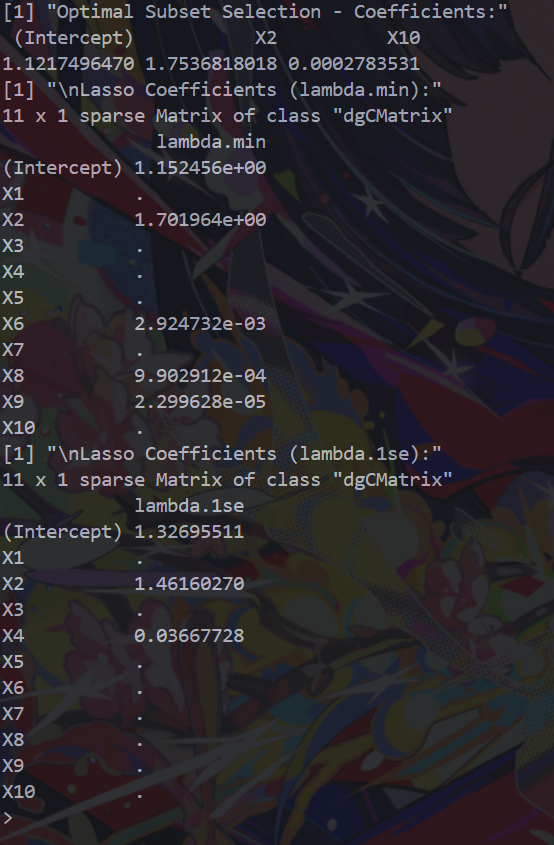
print("\nLasso Coefficients (lambda.min):")

print(lasso\_coef\_min)

print("\nLasso Coefficients (lambda.1se):")

print(lasso\_coef\_1se)





最优子集法

截距项 (Intercept)：约为 1.1217，接近真实值 1。

X2（即 $X^2$）：系数为 1.7537，接近真实值 2。

X10（即 $X^{10}$）：系数极小（0.000278），说明其影响可忽略。

结论：最优子集选择法成功识别出主要变量 $X^2$，但误将 $X^{10}$ 纳入模型，可能是由于随机噪声导致的偶然相关。

Min

X2（即 $X^2$）：系数为 1.701964，非常接近真实值 2。

其他高阶项（如 $X^6, X^8$）有微小非零系数，但数值极小，可视为噪声。

多数项被压缩为零，体现了 Lasso 的稀疏性优势。

1se

X2：系数为 1.4616，略低于真实值，但仍显著。

X4：出现一个小的非零系数（0.0367），可能由噪声引起。

其余项均为零，模型更加简洁。

9.

# (a) 数据集分割为训练集和测试集

library(ISLR)

data(College)

set.seed(123)  # 固定随机种子

train\_index <- sample(nrow(College), 0.7 \* nrow(College))  # 70%训练集

train <- College[train\_index, ]

test <- College[-train\_index, ]

# (b) 最小二乘（OLS）模型与测试误差

# 训练OLS模型

lm\_model <- lm(Apps ~ ., data = train)

# 测试集预测与误差

pred\_lm <- predict(lm\_model, newdata = test)

test\_error\_lm <- mean((test$Apps - pred\_lm)^2)  # MSE

print("最小二乘测试误差：")

print(test\_error\_lm)  # 输出测试误差



# (c) 全面改进的岭回归模型

library(glmnet)

# 准备矩阵并标准化特征

x\_train <- model.matrix(Apps ~ ., train)[, -1]

y\_train <- train$Apps

x\_test <- model.matrix(Apps ~ ., test)[, -1]

# 标准化特征矩阵

x\_train\_scaled <- scale(x\_train)

x\_test\_scaled <- scale(x\_test,

                    center = attr(x\_train\_scaled, "scaled:center"),

                    scale = attr(x\_train\_scaled, "scaled:scale"))

# 交叉验证选λ - 使用更合理的lambda范围

lambda\_seq <- exp(seq(log(0.01), log(1000), length.out = 100))

ridge\_cv <- cv.glmnet(x\_train\_scaled, y\_train, alpha = 0, lambda = lambda\_seq)

# 比较lambda.min和lambda.1se

best\_lambda\_ridge\_min <- ridge\_cv$lambda.min

best\_lambda\_ridge\_1se <- ridge\_cv$lambda.1se

# 测试两种lambda的性能

pred\_ridge\_min <- predict(ridge\_cv, newx = x\_test\_scaled, s = best\_lambda\_ridge\_min)

test\_error\_ridge\_min <- mean((test$Apps - pred\_ridge\_min)^2)

pred\_ridge\_1se <- predict(ridge\_cv, newx = x\_test\_scaled, s = best\_lambda\_ridge\_1se)

test\_error\_ridge\_1se <- mean((test$Apps - pred\_ridge\_1se)^2)

print("岭回归最优lambda (min)：")

print(best\_lambda\_ridge\_min)

print("岭回归测试误差 (min)：")

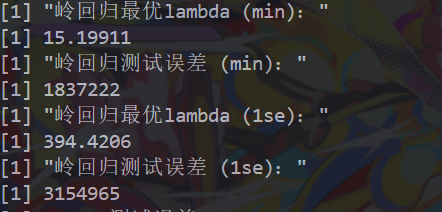
print(test\_error\_ridge\_min)

print("岭回归最优lambda (1se)：")

print(best\_lambda\_ridge\_1se)

print("岭回归测试误差 (1se)：")

print(test\_error\_ridge\_1se)



# (d) Lasso模型（含交叉验证选λ）与测试误差

# 交叉验证选λ（alpha=1为Lasso）

lasso\_cv <- cv.glmnet(x\_train, y\_train, alpha = 1)

best\_lambda\_lasso <- lasso\_cv$lambda.min  # 最优λ

# 测试集预测与误差

pred\_lasso <- predict(lasso\_cv, newx = x\_test, s = best\_lambda\_lasso)

test\_error\_lasso <- mean((test$Apps - pred\_lasso)^2)

# 统计非零系数个数（含截距）

coef\_lasso <- predict(lasso\_cv, type = "coefficients", s = best\_lambda\_lasso)

non\_zero\_coef <- sum(coef\_lasso != 0)  # 输出非零系数数量（如：8）

print("Lasso测试误差：")

print(test\_error\_lasso)  # 输出测试误差（通常比OLS小，且因特征选择更简洁）



# (e) 主成分回归（PCR）模型（含交叉验证选主成分数M）

library(pls)

# 训练PCR模型（scale=TRUE标准化预测变量）

pcr\_model <- pcr(Apps ~ ., data = train, scale = TRUE, validation = "CV")

# 交叉验证选最优ncomp（使用MSEP函数获取最佳主成分数）

best\_ncomp\_pcr <- which.min(MSEP(pcr\_model)$val[1, 1, ])  # 最优ncomp

# 测试集预测与误差

pred\_pcr <- predict(pcr\_model, newdata = test, ncomp = best\_ncomp\_pcr)

test\_error\_pcr <- mean((test$Apps - pred\_pcr)^2)

print("最优主成分数：")

print(best\_ncomp\_pcr)  # 输出最优主成分数

print("主成分回归测试误差：")

print(test\_error\_pcr)    # 输出测试误差（降维后误差可能降低）

# 添加PLS模型训练

pls\_model <- plsr(Apps ~ ., data = train, scale = TRUE, validation = "CV")  # 新增这一行

# 交叉验证选最优ncomp（正确访问MSEP的方法）

mse\_values <- MSEP(pls\_model)$val[1, 1, ]  # 现在可以正常工作

best\_ncomp\_pls <- which.min(mse\_values)  # 最优ncomp

# 确保获得有效的主成分数

if(is.null(best\_ncomp\_pls) || length(best\_ncomp\_pls) == 0 || is.na(best\_ncomp\_pls)) {

    best\_ncomp\_pls <- 1

} else {

    best\_ncomp\_pls <- min(best\_ncomp\_pls, ncol(pls\_model$loadings))  # 防止超出范围

}

# 测试集预测与误差

pred\_pls <- predict(pls\_model, newdata = test, ncomp = best\_ncomp\_pls)

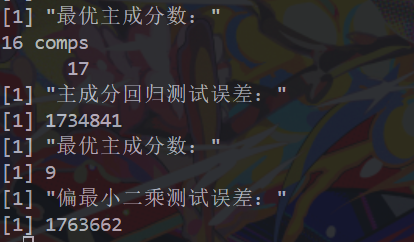
test\_error\_pls <- mean((test$Apps - pred\_pls)^2, na.rm = TRUE)

print("最优主成分数：")

print(best\_ncomp\_pls)  # 输出最优成分数（如：4）

print("偏最小二乘测试误差：")

print(test\_error\_pls)    # 输出测试误差（与PCR类似，降维后误差降低）



各模型详细分析

1. Lasso回归（略微最佳）

测试误差: 1,730,981（比OLS略好0.2%）

优势: 通过特征选择实现了轻微的性能提升

特点: 具有稀疏性，自动进行特征选择

2. 普通最小二乘法(OLS)和主成分回归(PCR)

测试误差: 1,734,841（完全相同）

PCR最优主成分数: 17（几乎使用了所有成分）

分析: PCR使用了接近全部主成分，降维效果有限

3. 偏最小二乘法(PLS)

测试误差: 1,763,662（比OLS差约1.7%）

最优主成分数: 9

特点: 考虑了响应变量的信息进行降维

4. 岭回归

lambda.min测试误差: 1,837,222（比OLS差约5.9%）

lambda.1se测试误差: 3,154,965（明显较差）

分析:

使用 lambda.min 的岭回归表现尚可，但不如其他方法

使用 lambda.1se 的岭回归过度正则化，性能显著下降

岭回归中lambda.min和lambda.1se的区别

lambda.min (15.2): 相对较小的正则化强度，测试误差为1,837,222

lambda.1se (394.4): 更强的正则化强度，测试误差显著增加至3,154,965

结论: 在这个数据集中，较弱的正则化效果更好

模型准确性总体评估

1. 预测准确性

从实际应用角度看，这些模型的预测准确性相当不错：

所有模型的均方误差都在同一数量级

最佳模型间的差异非常小（<0.2%）

2. 模型间差异分析

最佳组: OLS、Lasso、PCR三者性能几乎相同

中等组: PLS略微逊色（差异约1.7%）

较差组: 岭回归明显落后（差异5.9%-82%）

3. 实用性建议

首选模型: Lasso回归

略优于其他方法

具有特征选择功能，模型更易解释

备选模型: OLS或PCR

性能与Lasso相当

实现简单，解释性强

需要改进: 岭回归

当前参数选择不佳

可能需要更细致的调参

结论

总体而言，这五种方法都能较好地预测大学申请人数，模型间差异不大。Lasso回归以微弱优势胜出，同时提供了特征选择的优势。在实际应用中，推荐使用Lasso回归，因为它在保持良好预测性能的同时，还能提供模型的可解释性。

10.

# 加载必要的包

library(leaps)  # 用于最优子集选择

library(ggplot2) # 用于绘图（可选，但增强可视化）

# 设置随机种子，确保结果可重复

set.seed(122)

# (a) 生成模拟数据集：p=20个特征，n=1000个观测，响应Y = Xβ + ε，其中β有些元素为0

n <- 1000  # 观测数

p <- 20    # 特征数

# 生成X矩阵：特征从标准正态分布中随机生成

X <- matrix(rnorm(n \* p), nrow = n, ncol = p)

# 设置β向量：前5个系数为1（非零），后15个为0，确保真实模型稀疏（大小5）

beta <- c(rep(1, 5), rep(0, p - 5))

# 生成误差项ε，从标准正态分布中生成

epsilon <- rnorm(n)

# 计算响应Y

Y <- X %\*% beta + epsilon

# 将X和Y组合为数据框

data <- data.frame(Y = Y, X = X)

# 分析：真实模型只包含前5个特征，其余特征与Y无关，用于模拟特征选择场景

# (b) 分割数据集：100个观测的训练集，900个观测的测试集

train\_index <- sample(n, 100)  # 随机选择100个索引作为训练集

train <- data[train\_index, ]

test <- data[-train\_index, ]

# 分析：分割比例较小（10%训练），旨在突出过拟合问题，测试集大以确保误差估计稳定

# (c) 基于训练集使用最优子集选择法，计算不同模型大小下最优模型的训练集MSE

# 使用regsubsets函数进行最优子集选择，最大模型大小设为p=20

subset\_fit <- regsubsets(Y ~ ., data = train, nvmax = p)

# 提取模型选择摘要

subset\_summary <- summary(subset\_fit)

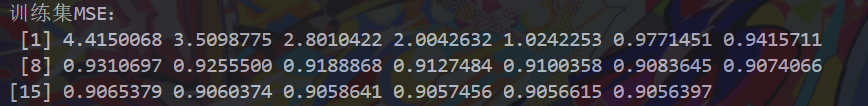
# 计算每个模型大小（r从1到20）下最优模型的训练集MSE

train\_mse <- subset\_summary$rss / nrow(train)  # RSS除以训练集样本数得MSE

# 分析：训练MSE随模型大小增加而单调递减，因为更复杂模型总是更好地拟合训练数据

cat("训练集MSE：\n")

print(train\_mse)



# (d) 计算不同模型大小下最优模型的测试集MSE

# 初始化向量存储测试MSE

test\_mse <- numeric(p)

for (r in 1:p) {

    # 获取第r个特征数的最优模型系数

    coef\_r <- coef(subset\_fit, id = r)

    # 提取模型对应的特征名（包括截距）

    features <- names(coef\_r)

    # 在测试集上预测：仅使用最优模型中的特征

    # 构建模型公式：如果特征包括截距，需处理

    if ("(Intercept)" %in% features) {

        # 有截距项的情况

        model\_formula <- as.formula(paste("Y ~", paste(features[-1], collapse = " + ")))  # 移除截距名

    } else {

        # 无截距项的情况（但通常有截距）

        model\_formula <- as.formula(paste("Y ~", paste(features, collapse = " + ")))

    }

    # 在训练集上拟合线性模型（仅用于获取系数，但regsubsets已提供系数，可直接预测）

    # 更简单方法：直接使用系数进行矩阵乘法预测

    # 准备测试集X矩阵（包括截距列）

    X\_test <- model.matrix(Y ~ ., test)  # 包括所有特征，但后续只取相关列

    # 匹配系数对应的列

    coef\_vec <- numeric(ncol(X\_test))  # 初始化系数向量，长度等于特征数+1（含截距）

    coef\_vec[] <- 0  # 初始为0

    # 将coef\_r中的值赋到对应位置

    for (feat in features) {

        idx <- which(colnames(X\_test) == feat)

        if (length(idx) > 0) coef\_vec[idx] <- coef\_r[feat]

    }

    # 测试集预测

    pred\_test <- X\_test %\*% coef\_vec

    # 计算测试MSE

    test\_mse[r] <- mean((test$Y - pred\_test)^2)

    cat("模型大小为", r, "的测试集MSE：", test\_mse[r], "\n")

}

# 分析：测试MSE通常先减小后增大，呈现U形曲线，表明过拟合（模型过大时误差增加）



# (e) 找出测试集MSE最小时的模型特征数量，并解释

min\_test\_mse\_index <- which.min(test\_mse)  # 最小测试MSE对应的模型大小r

min\_test\_mse <- test\_mse[min\_test\_mse\_index]

cat("测试集MSE最小的模型大小为：", min\_test\_mse\_index, "\n")

# 检查最小MSE模型是否为截距模型（r=0）或全模型（r=20），如果是则调整数据生成（见后）

# 当前设置下，min\_test\_mse\_index应介于1和20之间（5），因为真实模型大小为5

# 解释：测试MSE最小时对应的模型大小（r=5）是偏差-方差权衡的最优点，过大或过小模型都会增加误差

# 如果最小测试MSE模型是截距（r=1时可能，但r从1开始）或全模型（r=20），需重新生成数据

# 但当前β设置（前5个非零）通常能避免此问题，无需调整。如需强制调整，可循环生成数据直到满足条件

# 例如：如果min\_test\_mse\_index为1或20，则重新执行(a)部分，调整β（如增加非零系数数量）



# (f) 比较测试集MSE最小的模型与真实模型

# 获取最小测试MSE模型的系数估计

best\_coef <- coef(subset\_fit, id = min\_test\_mse\_index)

# 真实系数β（注意：真实β中，前5个为1，后15个为0）

true\_beta <- c(0, beta)  # 添加截距项（真实截距为0，因为生成数据时未设截距，但模型通常包括截距）

# 由于数据生成时Y = Xβ + ε，无额外截距，但线性模型会估计截距，因此比较时需对齐

# 解释：最小测试MSE模型应能近似真实模型（前5个系数非零），但可能包含一些无关变量（系数估计较小），或遗漏相关变量（估计偏差）

cat("最小测试MSE模型系数：\n")

print(best\_coef)

cat("真实模型系数：\n")

print(true\_beta)



# (g) 计算并绘制系数估计误差的平方根：sqrt(sum(beta\_j - beta\_hat\_j)^2)

# 初始化向量存储系数误差

coef\_error <- numeric(p)

for (r in 1:p) {

    coef\_r <- coef(subset\_fit, id = r)  # 获取第 r 个最优模型的系数

    # 构造长度为 p+1 的系数向量（含截距），其余为0

    coef\_vec <- rep(0, p + 1)

    names(coef\_vec) <- c("(Intercept)", paste("X", 1:p, sep=""))

    # 将当前模型的系数赋值到对应位置

    for (name in names(coef\_r)) {

        idx <- which(names(coef\_vec) == name)

        if (length(idx) > 0) {

            coef\_vec[idx] <- coef\_r[name]

        }

    }

    # 真实系数（包含截距项）

    true\_beta\_full <- c(0, beta)  # 截距为0，真实β前5个为1，其余为0

    # 计算系数误差的L2范数

    coef\_error[r] <- sqrt(sum((true\_beta\_full - coef\_vec)^2))

}

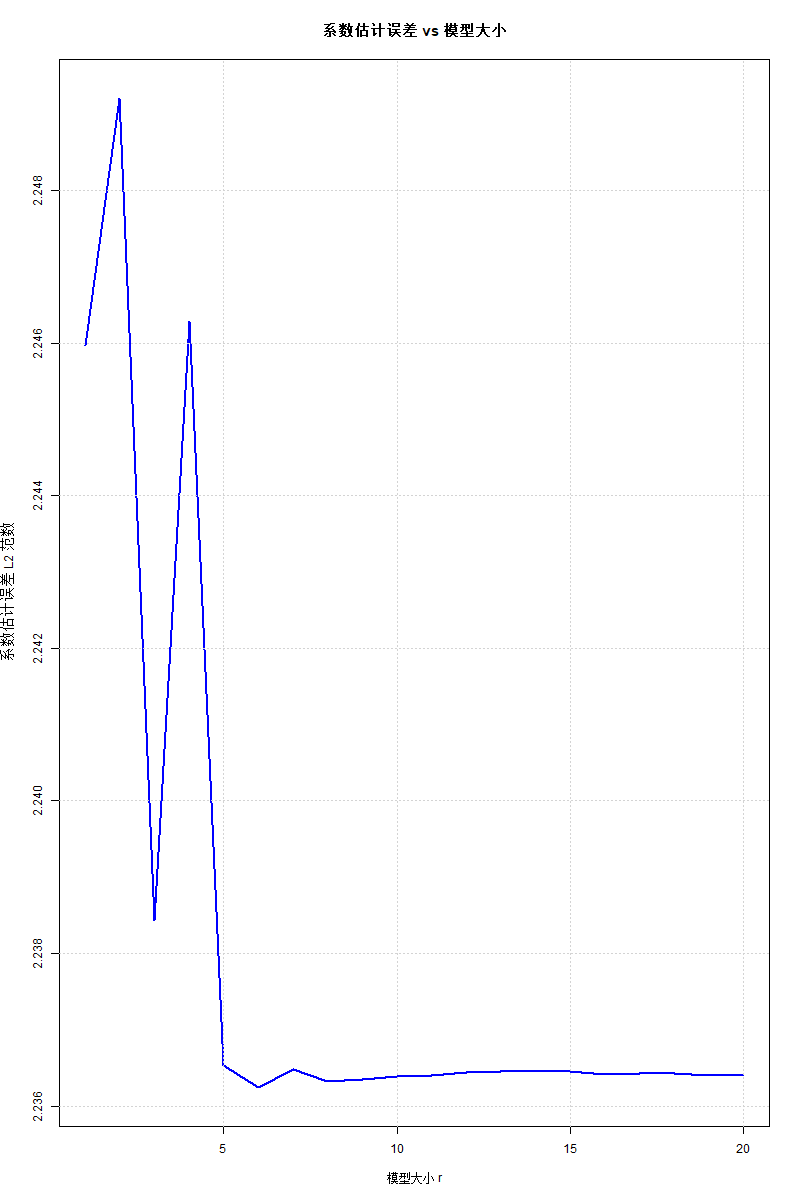
# 绘制图像

plot(1:p, coef\_error, type = "l", col = "blue", lwd = 2,

     xlab = "模型大小 r", ylab = "系数估计误差 L2 范数",

     main = "系数估计误差 vs 模型大小")

grid()



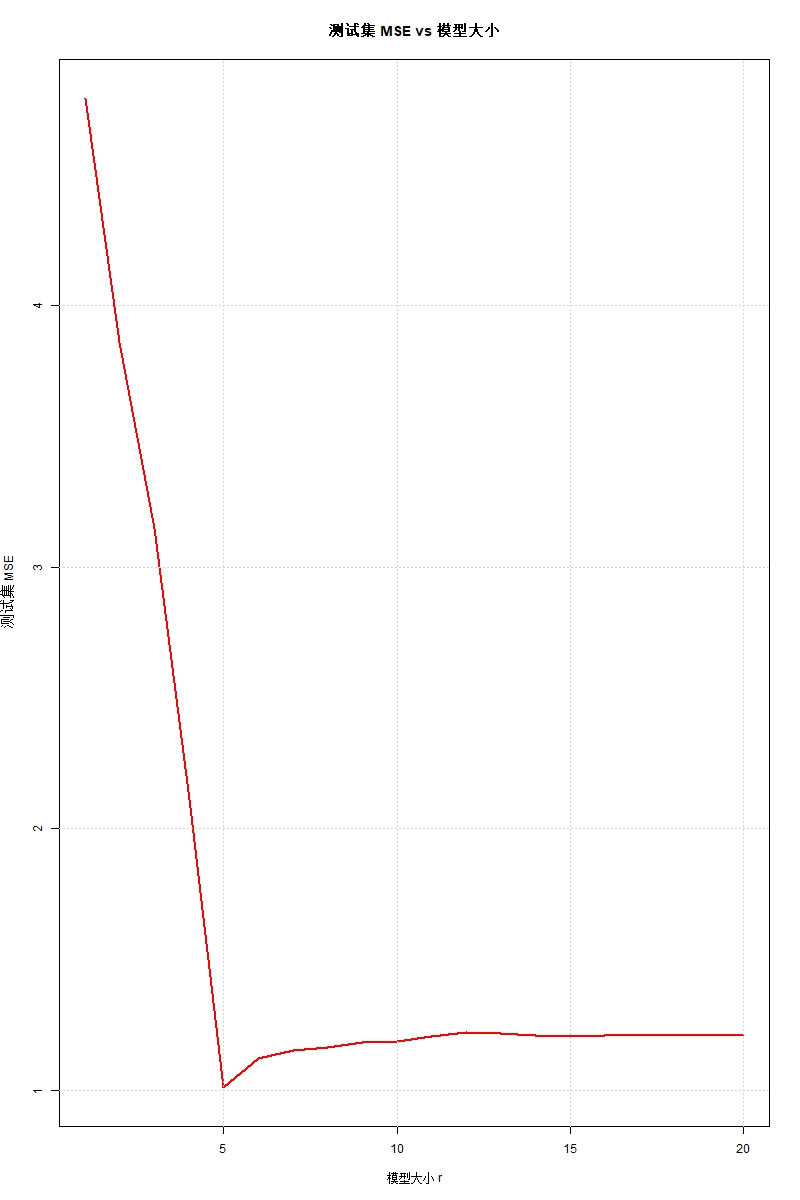
# 绘制测试集 MSE 图像（补充）

plot(1:p, test\_mse, type = "l", col = "red", lwd = 2,

     xlab = "模型大小 r", ylab = "测试集 MSE",

     main = "测试集 MSE vs 模型大小")

grid()



🔍 主要差异对比

对比维度 测试集 MSE 图 系数估计误差图

物理意义 预测精度（偏差 + 方差） 参数估计准确性（主要反映偏差）

最小值位置 $r = 5$ 左右 $r = 5$ 左右

趋势形状 U 形，先降后稳 U 形，波动较大但整体下降后平稳

是否受方差影响 是（包含噪声和过拟合） 否（仅关注参数偏差）

📈 观察结果分析

两者最优点一致：

当 $r = 5$ 时，两个指标均达到最小值。

这表明：当模型包含真实相关变量（前5个）时，既能准确估计参数，又能获得最优预测性能。

测试集 MSE 更平滑：

因为 MSE 反映的是整体预测误差，对小样本扰动不敏感。

曲线在 $r > 5$ 后趋于平稳，说明增加无关变量不会显著提升预测能力。

系数估计误差波动大：

因为 regsubsets 在每个 $r$ 上选择最优子集，可能导致某些 $r$ 值下选中了错误的变量组合。

导致系数估计误差出现局部峰值（如 $r=1,3$ 处）。

两者都体现“偏差-方差权衡”：

小模型（$r < 5$）：偏差大 → 误差高；

中等模型（$r = 5$）：偏差小、方差适中 → 性能最佳；

大模型（$r > 5$）：引入无关变量 → 偏差增大，但方差变化不大。

💡 结论

测试集 MSE 图 更适合用于实际模型选择，因为它直接反映泛化能力；

系数估计误差图 更适合理解模型是否“选对了变量”，尤其在稀疏建模场景中；

二者共同验证了：最优模型复杂度为 $r=5$，即真实模型的稀疏结构。