Analysis of riboflavin

12110248 赵一菡
Southern University of Science and Technology

Contents

Chapter 1

Data Description

	y	x[, "AADK_at"]	x[, "AAPA_at"]	x[, "ABFA_at"]	x[, "ABH_at"]	x[, "ABNA_at"]	x[, "ABRB_at"]	x[, "ACCA_at"]	x[, "ACCB_at"]	x[, "ACCC_at"]	x[, "ACD
1	-6.643856	8.492403....	8.111450....	8.320841....	10.28711....	8.261278....	10.20827....	9.745474....	9.818821....	9.676227....	8.372
2	-6.947862	7.639379....	7.239965....	7.289050....	9.862287....	7.303496....	9.500023....	9.216008....	9.854945....	9.650078....	7.732
3	-7.930160	8.088340....	7.855510....	7.793395....	9.676720....	7.098273....	9.473917....	9.580384....	9.926076....	9.787129....	7.925
4	-8.287712	7.886820....	7.939513....	7.997587....	9.680562....	7.408493....	9.788725....	9.447722....	9.852772....	9.546914....	7.838
5	-7.310432	6.805762....	7.554522....	7.609902....	8.551952....	7.712406....	8.490846....	8.696248....	8.573272....	8.589660....	7.905
6	-7.643856	7.178876....	7.885714....	7.757972....	8.748767....	7.667774....	8.603118....	8.943573....	8.615258....	8.726849....	7.982
7	-6.137965	6.971955....	7.695355....	8.074744....	8.751813....	6.813235....	8.905995....	8.395549....	8.803129....	8.558474....	8.272
8	-6.559792	8.035707....	7.912118....	8.239717....	10.28784....	8.055294....	9.939948....	9.354785....	9.351944....	9.416944....	7.896
9	-8.333516	6.700876....	7.211641....	7.324891....	9.272632....	6.681405....	9.593209....	8.585038....	8.718382....	8.645985....	7.509
10	-6.310432	6.893788....	8.033640....	8.172674....	8.839805....	7.361545....	8.324694....	8.288879....	8.310754....	8.422998....	8.000
11	-6.615287	6.292866....	7.324719....	7.448579....	8.540632....	6.337293....	7.794797....	8.077470....	7.969102....	8.217208....	7.855
12	-7.117787	6.327641....	7.523628....	7.354843....	8.606369....	7.284397....	8.506239....	8.272068....	8.349110....	8.191391....	7.893
13	-6.333516	6.496524....	7.581779....	7.812213....	9.355401....	7.067614....	9.462897....	8.725616....	8.589603....	8.802301....	8.290
14	-6.702750	7.101828....	7.336413....	7.671436....	9.806513....	7.228840....	9.671816....	8.897812....	8.959056....	9.071168....	7.937
15	-7.310432	6.363434....	7.436675....	7.657328....	9.142939....	6.553575....	8.770141....	8.440705....	8.613963....	8.589429....	8.263
16	-5.930160	7.700054....	7.542364....	7.901635....	10.07514....	8.538412....	10.04080....	8.988470....	8.953778....	8.959054....	7.682
17	-6.287712	7.712096....	7.813581....	7.590394....	10.53596....	7.679025....	9.929397....	8.777188....	8.949379....	8.998269....	7.541
18	-8.158429	7.034472....	7.806579....	8.273655....	10.38531....	7.915671....	10.47558....	8.931827....	9.571879....	9.531392....	8.272
19	-5.673003	6.598248....	7.631168....	7.934089....	9.293582....	7.834631....	9.253776....	8.641318....	8.262214....	8.364870....	8.067
20	-6.137965	6.821492....	7.424457....	7.490057....	10.30227....	7.186265....	9.688242....	8.532955....	8.471036....	8.631843....	7.667
21	-7.333516	6.514229....	7.611659....	7.808278....	9.496962....	7.827977....	9.969995....	8.614048....	8.719779....	8.770864....	7.950
22	-6.011588	6.993716....	7.880063....	8.104790....	10.03200....	7.608811....	9.464803....	8.403315....	8.393172....	8.242090....	8.114
23	-7.265345	6.850524....	8.637262....	8.550220....	9.320293....	7.067916....	8.983469....	8.644782....	8.783839....	8.592625....	8.395
24	-6.299028	6.283384....	6.625481....	6.853930....	8.632215....	5.054136....	7.433181....	6.877303....	7.485159....	7.767801....	7.478
25	-6.429731	6.796129....	6.651750....	7.130069....	7.217780....	6.239585....	8.684817....	8.316517....	7.986551....	8.786225....	7.823

Showing 1 to 26 of 71 entries, 4089 total columns

Figure 1.1: Data

The data **riboflavin** comes from the package **hdi** of R, with 71 observations and 4088 continuous explanatory variables. So this is a $n << p$ problem.

Chapter 2

PCA

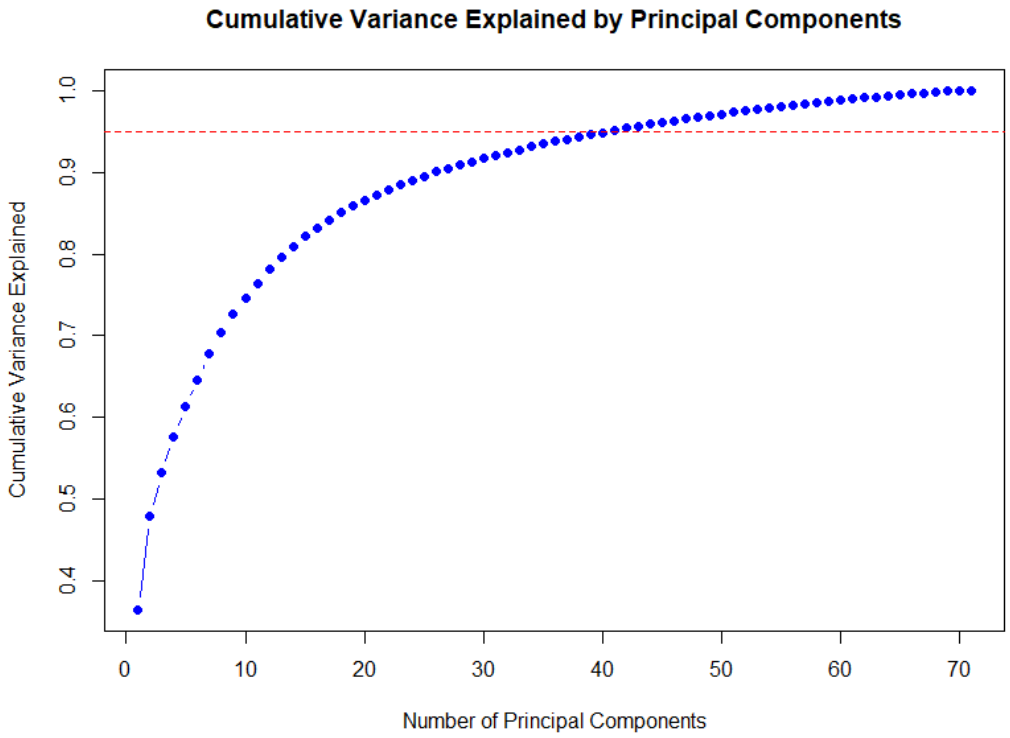


Figure 2.1: PCA Accumulated Variance

First, I try PCA to reduce the dimensionality. Considering that PCA is an unsupervised learning method and there is no clear standard how much cumulative variance should I choose, I choose 95% to keep more information. In this way, there are 41 variables left.

```r
riboflavin <- read.csv('riboflavin.csv')
X <- riboflavin[-1]
y <- riboflavin[1]
pca_result <- prcomp(X, scale. = TRUE)
explained_variance <- pca_result$sdev^2 / sum(pca_result$sdev^2)
cumulative_variance <- cumsum(explained_variance)
num_components <- 41
cumulative_variance[num_components]
plot(cumulative_variance, type = "b", pch = 19, col = "blue", xlab = "Number of Principal Components", ylab = "Cumulative Variance Explained", main = "Cumulative Variance Explained by Principal Components")
abline(h = 0.95, col = "red", lty = 2)
PC_variables <- list()
for (i in 1:41) {
  PC_index <- which.max(abs(pca_result$rotation[, i]))
  PC_variable <- rownames(pca_result$rotation)[PC_index]
  PC_variables[[paste0("PC", i)]] <- PC_variable
}
selected_columns <- riboflavin[, unlist(PC_variables)]
riboflavin2 <- cbind(y, selected_columns)
```

Chapter 3

Kernel Smoothing

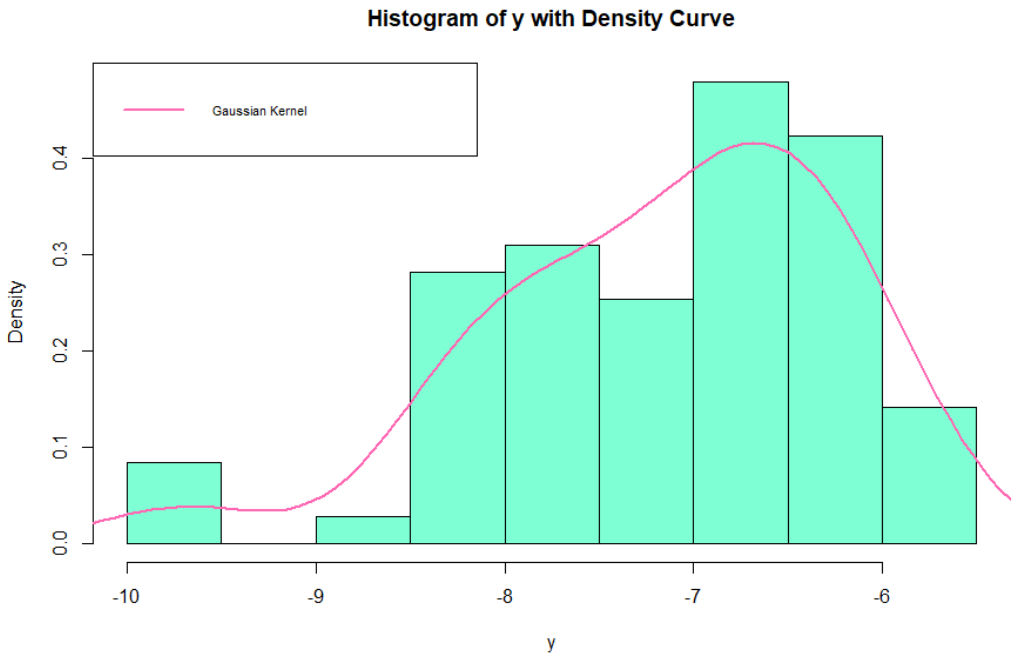


Figure 3.1: Kernel Smoothing of y

Next, I want to explore the distribution of y, so I make a histogram of y, and use Gaussian kernel with optimal bandwidth based on the cross validation to smooth. The plot shows that y is unimodal and left-skewed, so it might not be normal.

```
hcv_result <- sm.density(y, hcv = TRUE)
optimal_bandwidth <- hcv_result$h
hist(riboflavin$y, breaks = "FD", freq = FALSE, main = "
    Histogram of y with Density Curve", xlab = "y", col =
     "#7FFFD4", border = "black")
```

```r
density_estimate <- sm.density(y, h = optimal_bandwidth, display = "none")
lines(density_estimate$eval.points, density_estimate$estimate, col = "#FF69B4", lwd = 2)
legend("topleft", legend = c("Gaussian Kernel"), col = c("#FF69B4"), lwd = 2, cex = 0.7)

######################
##### Box-Cox #####
######################
y <- as.matrix(y)
min_y <- min(y)
if (min_y <= 0) {
  y <- y - min_y + 1
}
boxcox_result <- boxcox(y ~ 1, lambda = seq(-2, 2, by = 0.1))
best_lambda <- boxcox_result$x[which.max(boxcox_result$y)]
print(paste("Best lambda:", best_lambda))
if (best_lambda == 0) {
  y_boxcox <- log(y)
} else {
  y_boxcox <- (y^best_lambda - 1) / best_lambda
}

################################
### New Kernel Estimation ###
################################
hcv_result <- sm.density(y, hcv = TRUE)
optimal_bandwidth <- hcv_result$h
hist(riboflavin2_boxcox$y, breaks = "FD", freq = FALSE, main = "Histogram of y with Density Curve",
     xlab = "y", col = "#7FFFD4", border = "black")
density_estimate <- sm.density(y, h = optimal_bandwidth, display = "none")
lines(density_estimate$eval.points, density_estimate$estimate, col = "#FF69B4", lwd = 2)
legend("topleft", legend = c("Gaussian Kernel"), col = c("#FF69B4"), lwd = 2, cex = 0.7)
```

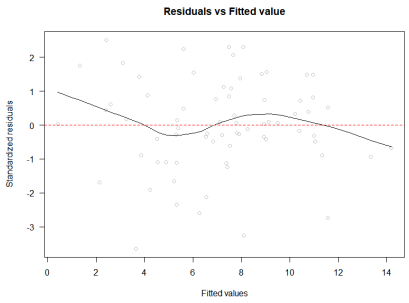
Chapter 4

Linear Model

4.1 Hypothesis Testing

Fitting a linear model and making inferences requires 5 assumptions, and I will test each one.

4.1.1 Linearity and Homoscedasticity



(a) r_i *vs* $\hat{y}$

```
> print(bp_test)

        studentized Breusch-Pagan test

data:  model
BP = 40.12, df = 40, p-value = 0.4649
```

(b) Breusch-Pagan Test

The residual plot shows that the $r_i's$ are randomly distribute around 0, so linearity is satisfied. And the p-value of Breusch-Pagan Test is very large, which means that we cannot reject H_0 : *Variance of the error term is a constant*, i.e. homoscedasticity is satisfied.

```
model <- lm(y ~ .-x.NADB_at.1, data = riboflavin2_boxcox)
summary(model)
scatter.smooth(resid(model) ~ fitted(model), col = "grey", las = 1, ylab = "Standardized residuals", xlab = "Fitted values")
title('Residuals vs Fitted value')
abline(h = 0, col = "red", lty = 2)
```

```
bp_test <- bptest(model)
print(bp_test)
```

4.1.2 Independence

There's no information of the time when the data is collected, so we just assume that the data is independent on the time dimension.

4.1.3 Multicollinearity

```
> vif(model)
   x.YNDJ_at    x.YACD_at    x.YXDJ_at    x.YWAE_at    x.XYLB_at    x.UREA_at  x.SPOVID_at    x.PHRK_at    x.ADHA_at
   12.969803     5.343119     4.195356     5.353218     6.829044     7.810606     3.949662     3.819092     3.705367
   x.RIBA_at    x.BOFA_at    x.YKPC_at    x.YCNK_at    x.PABB_at x.SPOIIGA_at    x.CSBA_at    x.HUTP_at    x.YCLK_at
    3.351136     5.023135     4.572193     6.114210     2.973912     3.164068     9.635928     2.753163     5.293606
   x.YUSJ_at    x.YKRP_at    x.XKDO_at    x.CHER_at    x.NADB_at    x.YURK_at    x.YTAB_at    x.YERO_at    x.YRHA_at
    2.617433     2.958422     3.326167     4.163400     2.050740     4.249037     7.776301     4.762744     3.834362
   x.ALST_at    x.PYRF_at    x.CYSE_at     x.FFH_at x.YHDS_r_at     x.YVFK_at     x.DPPD_at    x.YEFA_at    x.PRFA_at
    3.399612     2.926549     4.095913    13.977456     2.172475    62.333463    19.462113     5.829215     2.841525
   x.CSAA_at    x.YEZC_at     x.YVFO_at    x.GLNM_at
    4.162516     5.737347    55.285061     4.669691
```

Figure 4.2: VIF of the Original Model

```
model11 <- lm(y ~ .-x.NADB_at.1-x.YVFO_at, data = riboflavin2_boxcox)
model12 <- lm(y ~ .-x.NADB_at.1-x.FFH_at, data = riboflavin2_boxcox)
model13 <- lm(y ~ .-x.NADB_at.1-x.DPPD_at, data = riboflavin2_boxcox)
model14 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at, data = riboflavin2_boxcox)
```

(a) Deleting One Variable

```
> any(vif(model11)>10)
[1] TRUE
> any(vif(model12)>10)
[1] TRUE
> any(vif(model13)>10)
[1] TRUE
> any(vif(model14)>10)
[1] TRUE
```

(b) VIF

```
model21 <- lm(y ~ .-x.NADB_at.1-x.YVFO_at-x.FFH_at, data = riboflavin2_boxcox)
model22 <- lm(y ~ .-x.NADB_at.1-x.DPPD_at-x.YVFO_at, data = riboflavin2_boxcox)
model23 <- lm(y ~ .-x.NADB_at.1-x.FFH_at-x.DPPD_at, data = riboflavin2_boxcox)
model24 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at-x.DPPD_at, data = riboflavin2_boxcox)
model25 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at-x.FFH_at, data = riboflavin2_boxcox)
model26 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at-x.YVFO_at, data = riboflavin2_boxcox)
```

(a) Deleting Two Variables

```
> any(vif(model21)>10)
[1] FALSE
> any(vif(model22)>10)
[1] FALSE
> any(vif(model23)>10)
[1] TRUE
> any(vif(model24)>10)
[1] FALSE
> any(vif(model25)>10)
[1] FALSE
> any(vif(model26)>10)
[1] TRUE
```

(b) VIF

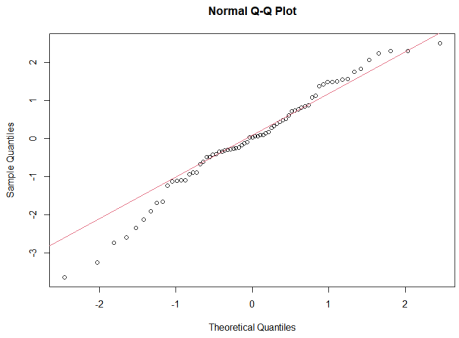
When I first fit the model, it shows that the **x.NADB_at.1** term can be completely represented by other variables, so I delete it and fit a new one. Then, I compute the VIF of each variable and check whether any of them is more than 10, and I find four of them with multicollinearity problem and none of them is significant. When I try to delete one, there still exists multicollinearty. So I test with two of the four deleted and choose one with the highest R^2, which turn out to be **x.DPPD_at** and **x.YVFO_at** are removed, and get the **model25**.

```
vif(model)
model11 <- lm(y ~ .-x.NADB_at.1-x.YVFO_at, data =
    riboflavin2_boxcox)
```

```
model12 <- lm(y ~ .-x.NADB_at.1-x.FFH_at, data =
    riboflavin2_boxcox)
model13 <- lm(y ~ .-x.NADB_at.1-x.DPPD_at, data =
    riboflavin2_boxcox)
model14 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at, data =
    riboflavin2_boxcox)

model21 <- lm(y ~ .-x.NADB_at.1-x.YVFO_at-x.FFH_at, data
    = riboflavin2_boxcox)
model22 <- lm(y ~ .-x.NADB_at.1-x.DPPD_at-x.YVFO_at,
    data = riboflavin2_boxcox)
model23 <- lm(y ~ .-x.NADB_at.1-x.FFH_at-x.DPPD_at, data
    = riboflavin2_boxcox)
model24 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at-x.DPPD_at,
    data = riboflavin2_boxcox)
model25 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at-x.FFH_at, data
    = riboflavin2_boxcox)
model26 <- lm(y ~ .-x.NADB_at.1-x.YVFK_at-x.YVFO_at,
    data = riboflavin2_boxcox)
```

4.1.4 Normality



(a) Q-Q Plot

```
> shapiro.test(riboflavin2_boxcox$y)

        Shapiro-Wilk normality test

data:  riboflavin2_boxcox$y
W = 0.97885, p-value = 0.2746
```

(b) Shapiro Test

We can learn from the kernel smoothing that y is unlikely normally distributed. So I make a Box-Cox transformation to y and then make the Shapiro test and draw the Q-Q plot of y. Since p-value of the test is large, we cannot reject H_0 : *y is normally distributed*. Hence, normality is satisfied.

```
shapiro.test(riboflavin25_boxcox$y)
qqnorm(resid(model25))
qqline(resid(model25), col = 2)
```

4.2 Model Comparison

```
Residual standard error: 1.975 on 32 degrees of freedom
Multiple R-squared:  0.8267,    Adjusted R-squared:  0.6209
F-statistic: 4.017 on 38 and 32 DF,  p-value: 6.055e-05
```

(a) R^2 and Adjusted R^2 of Full Model

```
> # prediction
> predicted <- predict(model2, x)
> lm_mse <- mean((riboflavin2_boxcox[,1]-predicted)^2)
> lm_mse
[1] 1.75762
```

(b) Prediction MSE of Full Model

```
Residual standard error: 1.61 on 55 degrees of freedom
Multiple R-squared:  0.802,     Adjusted R-squared:  0.748
F-statistic: 14.85 on 15 and 55 DF,  p-value: 3.155e-14
```

(a) R^2 and Adjusted R^2 of AIC Model

```
> prediction_AIC <- predict(modelAIC, newx = x)
> AIC_mse <- mean((prediction_AIC - riboflavin2_boxcox$y)^2)
> AIC_mse
[1] 2.007884
```

(b) Prediction MSE of AIC Model

```
Residual standard error: 1.667 on 58 degrees of freedom
Multiple R-squared:  0.7761,    Adjusted R-squared:  0.7298
F-statistic: 16.76 on 12 and 58 DF,  p-value: 1.163e-14
```

(a) R^2 and Adjusted R^2 of BIC Model

```
> prediction_BIC <- predict(modelBIC, newdata = riboflavin2_boxcox)
> BIC_mse <- mean((prediction_BIC - y)^2)
> BIC_mse
[1] 2.270461
```

(b) Prediction MSE of BIC Model

```
> ss_res <- sum((y - y_pred)^2)
> ss_tot <- sum((y - mean(y))^2)
> r_squared <- 1 - ss_res / ss_tot
> n <- length(y)
> lasso_coef <- coef(final_lasso_model, s = best_lambda)
> p <- sum(lasso_coef != 0) - 1
> adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
> r_squared
[1] 0.7219365
> adjusted_r_squared
[1] 0.6256838
```

(a) R^2 and Adjusted R^2 of Lasso Model

```
> y_pred <- predict(final_lasso_model, newx = as.matrix(x))
> LASSO_mse <- mean((y - y_pred)^2)
> LASSO_mse
[1] 2.819975
```

(b) Prediction MSE of Lasso Model

Here I try AIC, BIC and Lasso to choose variables and the whole dataset is taken as both train data and test data. Although the MSE of the full model seems to be the smallest, it only means that it performs the best in the original data. Thus, in general, consider both to minimize MSE and maximize adjusted R^2, the best model should be the AIC model.

```
### full model ###
predicted <- predict(model2, x)
lm_mse <- mean((riboflavin2_boxcox[,1]-predicted)^2)
lm_mse
summary(model2)
### AIC ###
modelAIC <- stepAIC(model2, direction = 'both')
prediction_AIC <- predict(modelAIC, newx = x)
AIC_mse <- mean((prediction_AIC - riboflavin2_boxcox$y)
    ^2)
AIC_mse
summary(modelAIC)
### BIC ###
modelBIC <- stepAIC(model2, direction = 'both', k = log(
    nrow(model.frame(model2))))
prediction_BIC <- predict(modelBIC, newdata =
    riboflavin2_boxcox)
BIC_mse <- mean((prediction_BIC - y)^2)
BIC_mse
summary(modelBIC)
### LASSO ###
lasso_model <- cv.glmnet(as.matrix(x), y, alpha = 1)
```

```
best_lambda <- lasso_model$lambda.min
final_lasso_model <- glmnet(x, y, alpha = 1, lambda =
    best_lambda)
lasso_coefficients <- coef(final_lasso_model)
y_pred <- predict(final_lasso_model, newx = as.matrix(x)
    )
LASSO_mse <- mean((y - y_pred)^2)
LASSO_mse
ss_res <- sum((y - y_pred)^2)
ss_tot <- sum((y - mean(y))^2)
r_squared <- 1 - ss_res / ss_tot
n <- length(y)
lasso_coef <- coef(final_lasso_model, s = best_lambda)
p <- sum(lasso_coef != 0) - 1
adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (
    n - p - 1))
r_squared
adjusted_r_squared
```

Chapter 5

Other Methods

5.1 Multivariate Adaptive Regression Splines(MARS)

```
> ss_res <- sum((y - predicted)^2)
> ss_tot <- sum((y - mean(y))^2)
> r_squared <- 1 - ss_res / ss_tot
> n <- length(y)
> p <- length(mars_model$selected.terms) - 1
> adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
> r_squared
[1] 0.8547289
> adjusted_r_squared
[1] 0.8246728
```

(a) R^2 and Adjusted R^2 of MARS Model

```
> predicted <- predict(mars_model, x)
> mars_mse  <- mean((riboflavin2_boxcox[,1]-predicted)^2)
> mars_mse
[1] 1.473264
```

(b) Prediction MSE of MARS Model

I use the MARS method to fit model, the R^2 is 0.85, adjusted R^2 is 0.82 and prediction MSE is 1.47.

```
1  mars_model <-  mars(x, y, degree = 2)
2  summary(mars_model)
3  predicted <- predict(mars_model, x)
4  mars_mse  <- mean((riboflavin2_boxcox[,1]-predicted)^2)
5
6  ss_res <- sum((y - predicted)^2)
7  ss_tot <- sum((y - mean(y))^2)
8  r_squared <- 1 - ss_res / ss_tot
9  n <- length(y)
10 p <- length(mars_model$selected.terms) - 1
```

```
adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (
    n - p - 1))
r_squared
adjusted_r_squared
```

5.2 Classification And Regression Tree(CART)

```
> ss_res <- sum((y - predicted)^2)
> ss_tot <- sum((y - mean(y))^2)
> r_squared <- 1 - ss_res / ss_tot
> n <- length(y)
> p <- length(unique(tree_model$frame$var[tree_model$frame$var != "<leaf>"]))
> adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
> r_squared
[1] 0.8319118
> adjusted_r_squared
[1] 0.8071119
```

(a) R^2 and Adjusted R^2 of CART Model

```
> predicted <- predict(tree_model, as.data.frame(x))
> tree_mse  <- mean((riboflavin2_boxcox[,1]-predicted)^2)
> tree_mse
[1] 1.704663
```

(b) Prediction MSE of CART Model

Then I use the CART method to fit model, the R^2 is 0.83, adjusted R^2 is 0.81 and prediction MSE is 1.7. But I think CART method is not highly explanatory for continuous variables, since there might be little difference between different classes.

```
tree_model <- tree(y ~ .,data = riboflavin2_boxcox)
plot(tree_model)
text(tree_model)

predicted <- predict(tree_model, as.data.frame(x))
tree_mse  <- mean((riboflavin2_boxcox[,1]-predicted)^2)

ss_res <- sum((y - predicted)^2)
ss_tot <- sum((y - mean(y))^2)
r_squared <- 1 - ss_res / ss_tot
n <- length(y)
p <- length(unique(tree_model$frame$var[tree_model$frame
    $var != "<leaf>"]))
adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (
    n - p - 1))
r_squared
adjusted_r_squared
```

Chapter 6

Elastic Net

Considering that PCA is not y oriented, I try Lasso and Elastic Net directly.

6.1 Lasso & Elastic Net

```
> ss_res <- sum((y - prediction_lasso)^2)
> ss_tot <- sum((y - mean(y))^2)
> r_squared <- 1 - ss_res / ss_tot
>
> n <- length(y)
> p <- length(which(coef(model_lasso, s = "lambda.min") != 0)) - 1
> adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
> r_squared
[1] 0.6046345
> adjusted_r_squared
[1] 0.4677773
```

(a) R^2 and Adjusted R^2 of Lasso Model

```
> mse_lasso <- mean((prediction_lasso - riboflavin$y)^2)
> mse_lasso
[1] 4.009591
```

(b) Prediction MSE of Lasso Model

```
> ss_res <- sum((y - prediction_elastic)^2)
> ss_tot <- sum((y - mean(y))^2)
> r_squared <- 1 - ss_res / ss_tot
>
> n <- length(y)
> p <- length(which(coef(model_elastic, s = "lambda.min") != 0)) - 1
> adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
> r_squared
[1] 0.7361478
> adjusted_r_squared
[1] 0.6070286
```

(a) R^2 and Adjusted R^2 of Elastic Net Model

```
> mse_elastic <- mean((prediction_elastic - riboflavin$y)^2)
> mse_elastic
[1] 2.675852
```

(b) Prediction MSE of Elastic Net Model

First, I use the whole data to fit the models. Comparing the two model, obviously

that Elastic Net is better than Lasso, since Lasso is a special case of Elastic Net.

```r
### Lasso ###
model_lasso <- cv.glmnet(as.matrix(x), y, alpha = 1)
coef(model_lasso, s = "lambda.min")
sum(coef(model_lasso, s = "lambda.min") != 0)

prediction_lasso <- predict(model_lasso, newx = as.
    matrix(riboflavin[, -1]))
print(prediction_lasso)
mse_lasso <- mean((prediction_lasso - riboflavin$y)^2)
mse_lasso

ss_res <- sum((y - prediction_lasso)^2)
ss_tot <- sum((y - mean(y))^2)
r_squared <- 1 - ss_res / ss_tot
n <- length(y)
p <- length(which(coef(model_lasso, s = "lambda.min") !=
    0)) - 1
adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (
    n - p - 1))
r_squared
adjusted_r_squared

### Elastic Net ###
alpha_values <- seq(0, 1, by = 0.1)
cv_results <- lapply(alpha_values, function(alpha) {
  cv.glmnet(as.matrix(x), y, alpha = alpha)
})
cv_mse <- sapply(cv_results, function(cv) min(cv$cvm))
best_alpha_index <- which.min(cv_mse)
best_alpha <- alpha_values[best_alpha_index]
best_lambda <- cv_results[[best_alpha_index]]$lambda.min
model_elastic <- glmnet(x, y, alpha = best_alpha, lambda
    = best_lambda)
elastic_net_coefficients <- coef(model_elastic)

sum(elastic_net_coefficients != 0)
elastic_net_coefficients[elastic_net_coefficients != 0]

prediction_elastic <- predict(model_elastic, newx = as.
    matrix(riboflavin[, -1]))
print(prediction_elastic)
```

```
mse_elastic <- mean((prediction_elastic - riboflavin$y)^2)
mse_elastic

ss_res <- sum((y - prediction_elastic)^2)
ss_tot <- sum((y - mean(y))^2)
r_squared <- 1 - ss_res / ss_tot
n <- length(y)
p <- length(which(coef(model_elastic, s = "lambda.min") != 0)) - 1
adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
r_squared
adjusted_r_squared
```

6.2 Elastic Net & Average Elastic Net

```
> sst <- sum((y_test - mean(y_test))^2)
> sse <- sum((y_test - y_test_pred)^2)
> r_squared <- 1 - (sse / sst)
>
> n <- length(y_train)
> p <- length(selected_variables)
> adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
> r_squared
[1] 0.8570204
> adjusted_r_squared
[1] 0.6387884
```

(a) R^2 and Adjusted R^2 of Average Elastic Net Model

```
> mse <- mean((y_test - y_test_pred)^2)
> mse
[1] 0.08749318
```

(b) Prediction MSE of Average Elastic Net Model

Next, I want to fit a model that can perform better, so I randomly divide 70% of the data into train data and the rest as test data to fit a best elastic net model. After repeat the procedure for 100 times, I choose the variables selected for more than 50 times and take their average coefficients as the new coefficients. In this term, the adjusted R^2 increases from 0.61 to 0.64 and prediction MSE decreases from 2.68 to 0.09. It shows that multiple biased classifiers can be combined to form one robust classifier.

```
num_repeats <- 100
variable_selection_frequency <- matrix(0, nrow = num_repeats, ncol = ncol(X))
alpha_values <- seq(0, 1, by = 0.1)
for (i in 1:num_repeats) {
```

```
  set.seed(i)
  train_indices <- sample(1:nrow(X), size = 0.7 * nrow(X))
  X_train <- X[train_indices, ]
  y_train <- y[train_indices]
  X_test <- X[-train_indices, ]
  y_test <- y[-train_indices]
  cv_results <- lapply(alpha_values, function(alpha) {
    cv.glmnet(as.matrix(X_train), y_train, alpha = alpha)
  })
  cv_mse <- sapply(cv_results, function(cv) min(cv$cvm))
  best_alpha_index <- which.min(cv_mse)
  best_alpha <- alpha_values[best_alpha_index]
  best_lambda <- cv_results[[best_alpha_index]]$lambda.min
  final_model <- glmnet(X_train, y_train, alpha = best_alpha, lambda = best_lambda)
  coefs <- coef(final_model, s = best_lambda)
  selected_vars <- as.numeric(coefs[-1] != 0)
  variable_selection_frequency[i, ] <- selected_vars
}
selection_frequency <- colMeans(variable_selection_frequency)
selected_variables <- which(selection_frequency > 0.5)
X_train_selected <- X_train[, selected_variables]
X_test_selected <- X_test[, selected_variables]
final_model <- glmnet(X_train_selected, y_train, alpha = best_alpha, lambda = best_lambda)
y_test_pred <- predict(final_model, newx = X_test_selected)
mean((y_test - y_test_pred)^2)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - y_test_pred)^2)
r_squared <- 1 - (sse / sst)
n <- length(y_train)
p <- length(selected_variables)
adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - p - 1))
r_squared
adjusted_r_squared
```

Chapter 7

Conlusion

I think the best model is PCA + MARS, because considering that there is a high possibility of the existence of correlations between gene expressions and MARS takes the interactions into account, therefore is better than simple linear model. And compared to directly Elastic Net Method, PCA can deal with multicollinearity better. So this model may have better predictions. Also, it is more interpretive than the Regression Tree.