riboflavin

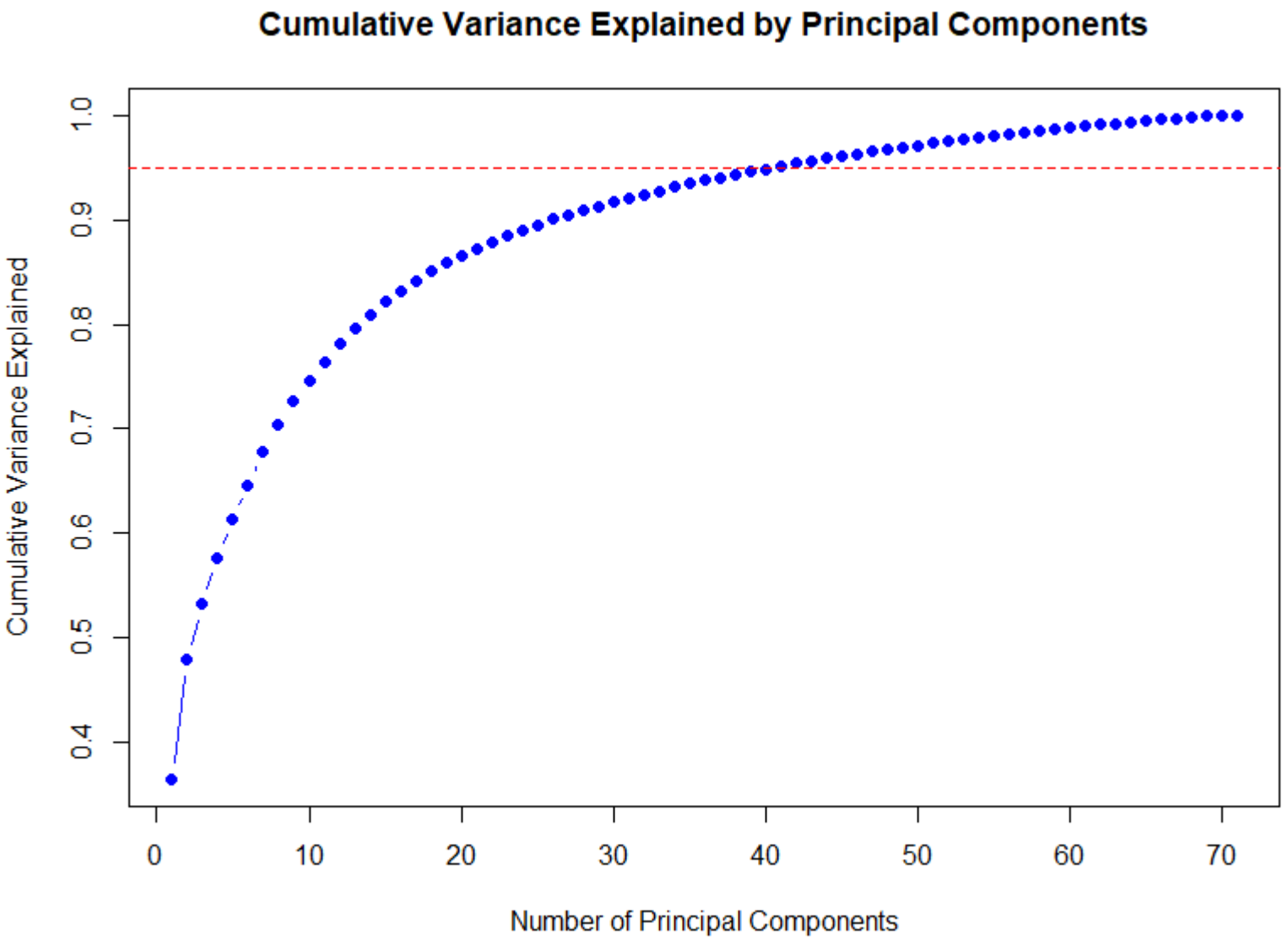
12110248 赵一菡

	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[, "ACC
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...	7.836
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...	7.505
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...	7.806278...	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

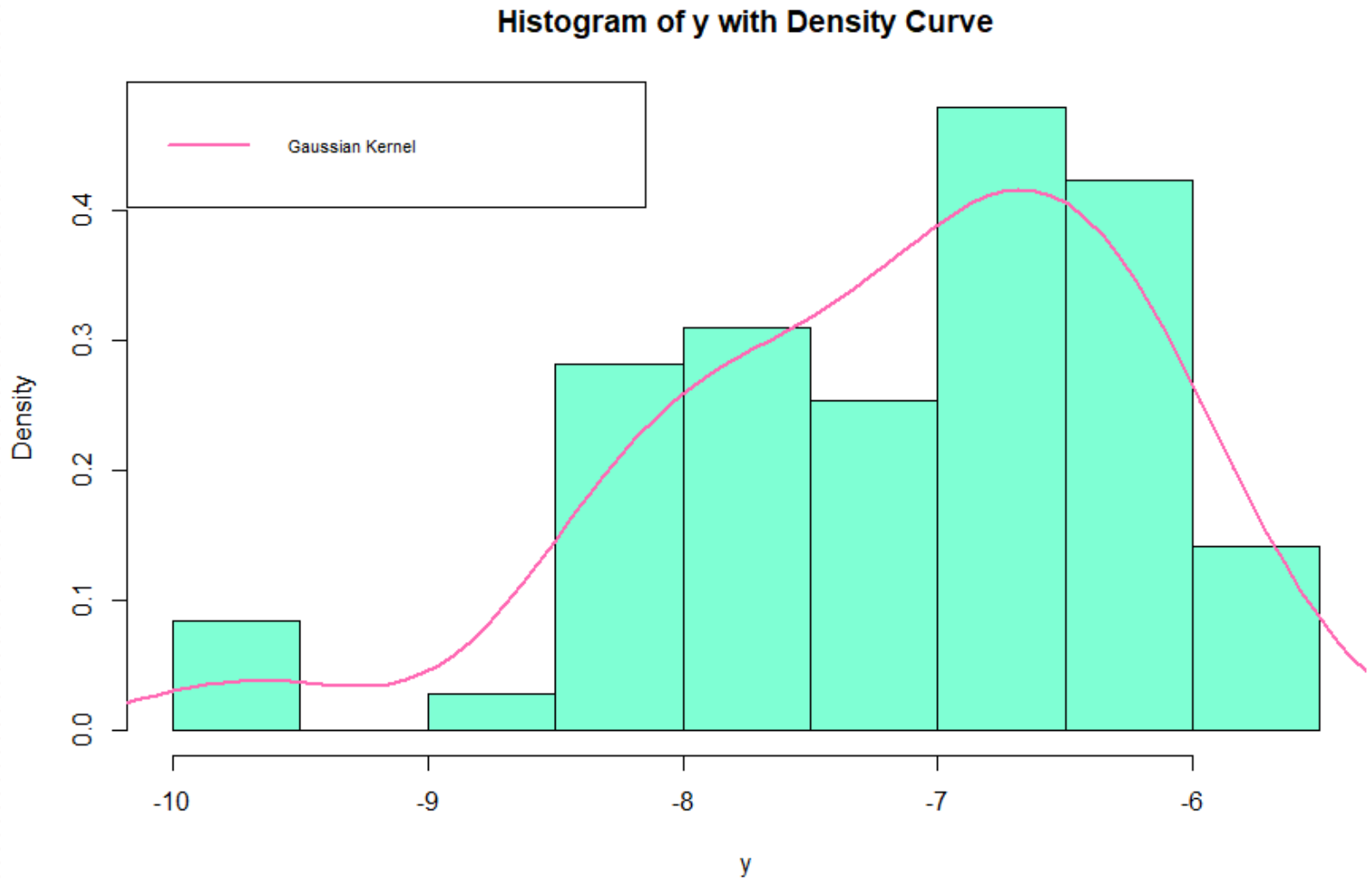
- **n=71, p=4088**
- **n<<p**
- **Dimensionality Reduction**

PCA



- **41 explanatory variables**
- **95% of the variance**

Kernel Smoothing


Histogram of y with Density Curve
Gaussian Kernel
Density
0.0
0.1
0.2
0.3
0.4
-10
-9
-8
-7
-6
y

$$h^* \simeq 3.491 s n^{-1/3}.$$

Freedman–Diaconis rule

alexdonkey 2009年9月22日

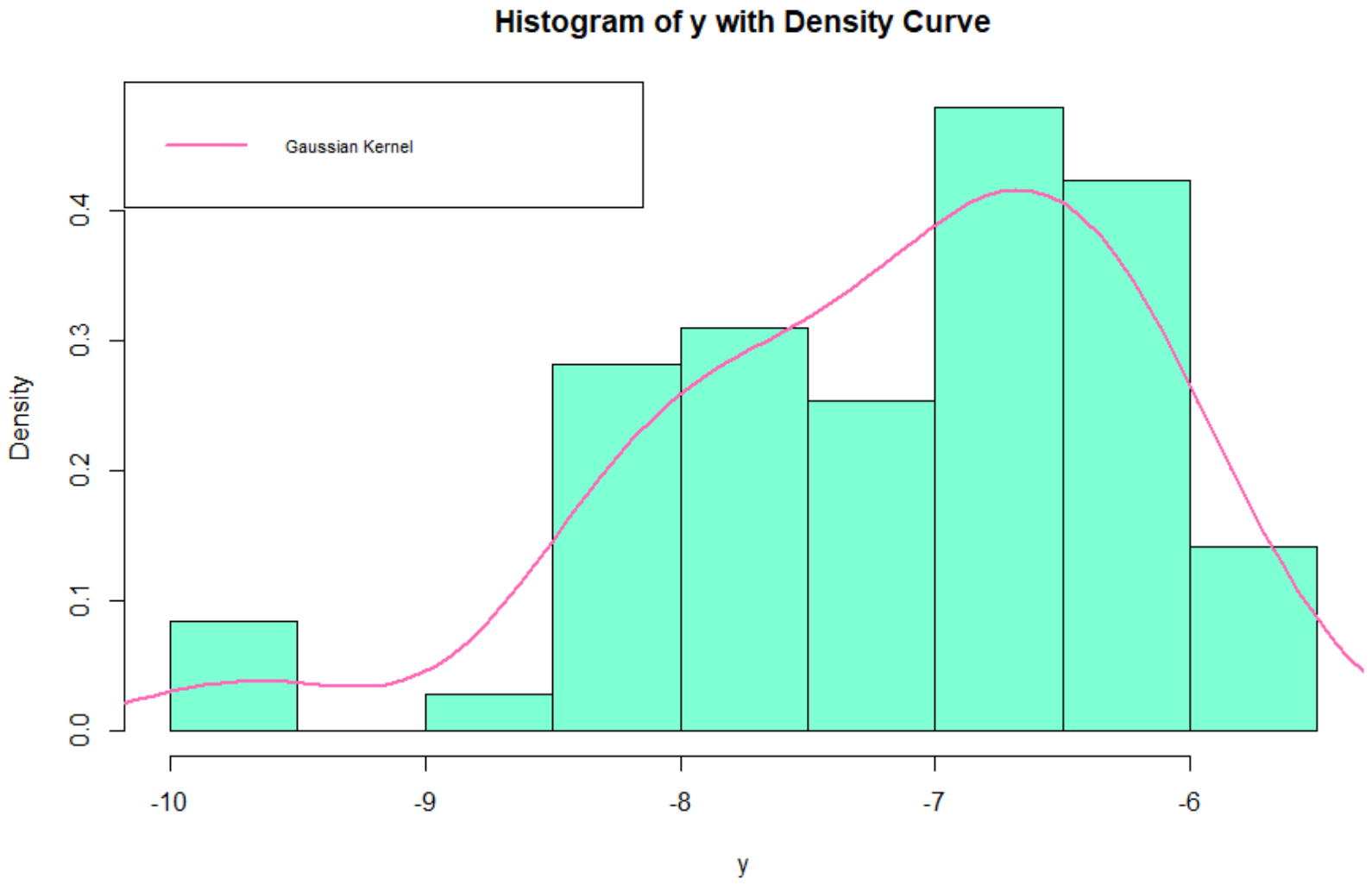
似乎有点知道了，请大家指教

先检验下样本的分布情况，

如果样本分布服从正态分布的话，那么就用正态核来估计吧，hopt= 1.06*sigma*n^(-1/5)，当然也可以选择 ?density 里的那些选择 nrd, bcv, ucv 等等

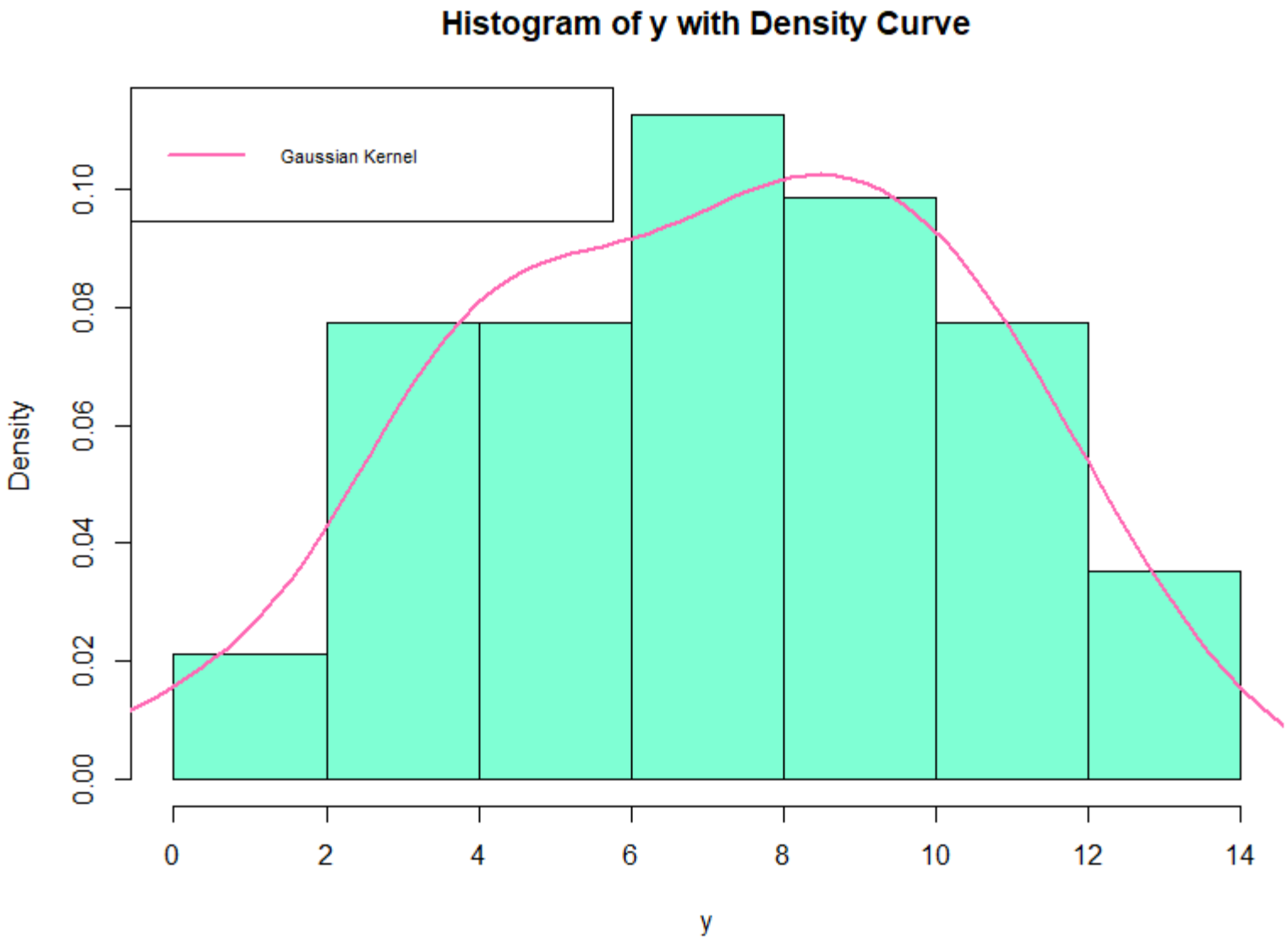
sinon, 那就可以用CV(h): validation croisée 来估计hopt 可以安装程序包 sm，用hcv 就OK了，还以用 Plug-in方法，不过在R里好像也只找到 noyau gaussien 的方法，这点不是很明白?

回复




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

        Shapiro-Wilk normality test

data:  riboflavin$y
W = 0.94412, p-value = 0.003326
```

- **Left skewed**
- **Unimodal**
- **Not normal**




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

        Shapiro-Wilk normality test

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

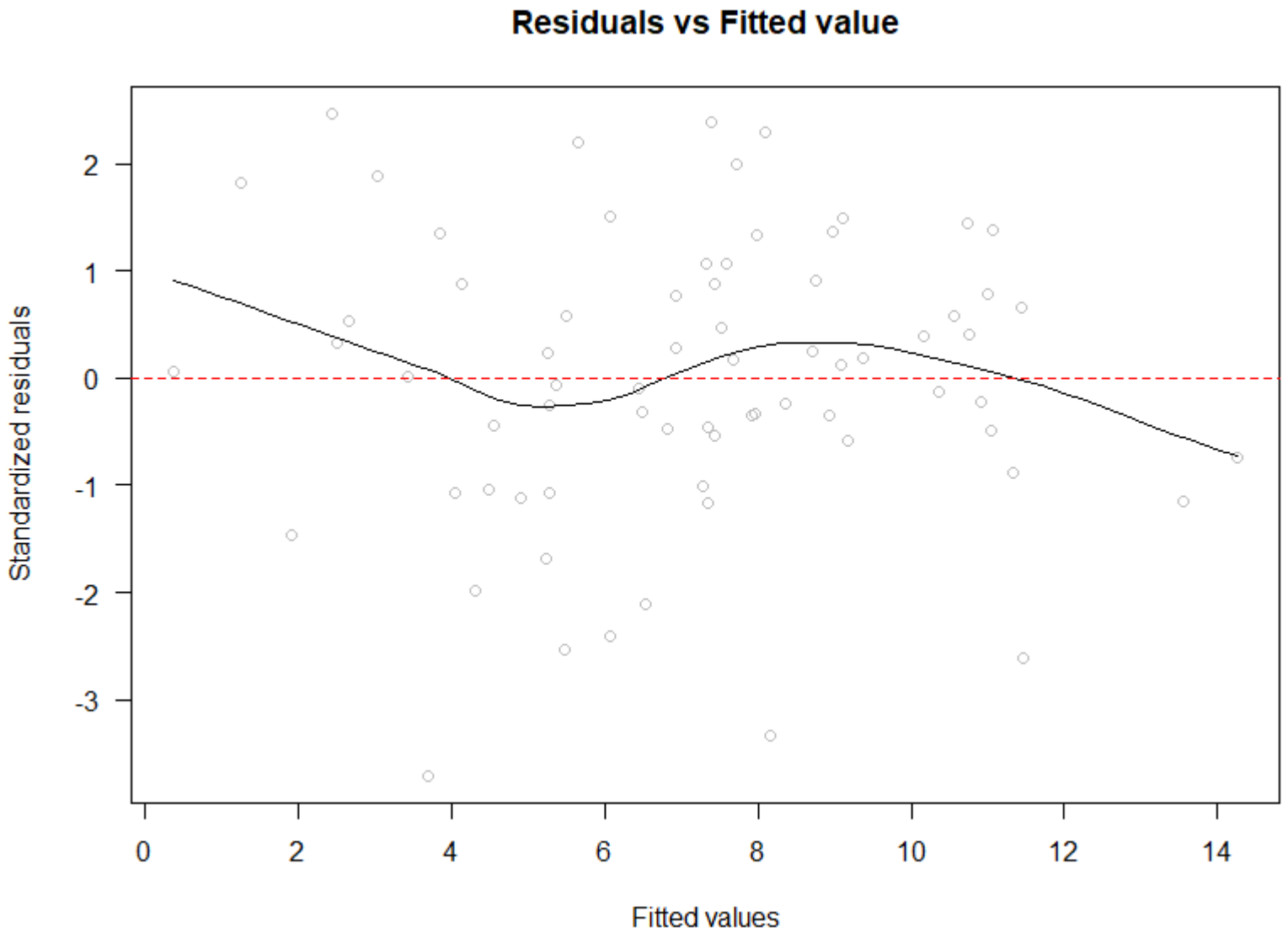
- **Almost no skewness**
- **Unimodal**
- **Normal**

Linear Model

Linear Regression

```
> summary(model)

Call:
lm(formula = y ~ ., data = riboflavin2_boxcox)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7113 -0.6690  0.0586  0.8813  2.4658

Coefficients: (1 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -194.00317   65.79566  -2.949  0.00613 **
x.YNDJ_at      0.34332    1.61964   0.212  0.83356
x.YACD_at     -0.50070    1.71829  -0.291  0.77275
x.YXDJ_at      2.26359    2.11218   1.072  0.29241
x.YWAE_at     -0.22895    2.02556  -0.113  0.91076
x.XYLB_at     -2.03679    1.30453  -1.561  0.12894
x.UREA_at     -2.58534    1.53247  -1.687  0.10197
x.SPOVID_at   -0.43406    1.76460  -0.246  0.80737
x.PHRK_at      0.25504    0.89546   0.285  0.77775
x.ADHA_at      2.63040    1.27818   2.058  0.04838 *
x.RIBA_at      5.91973    1.83956   3.218  0.00309 **
x.BOFA_at      2.16976    1.69554   1.280  0.21046
x.YKPC_at     -0.06068    1.26954  -0.048  0.96220
x.YCNK_at      1.92565    2.52561   0.762  0.45174
x.PABB_at      1.50610    1.68541   0.894  0.37864
x.SPOIIGA_at  -0.92676    1.57295  -0.589  0.56015
x.CSBA_at      0.53573    1.67204   0.320  0.75088
x.HUTP_at      1.47673    2.47318   0.597  0.55492
x.YCLK_at      1.05164    3.00917   0.349  0.72917
x.YUSJ_at     -0.36367    1.30518  -0.279  0.78244
x.YKRP_at     -0.84267    1.00664  -0.837  0.40915
x.XKDO_at      2.80358    0.85764   3.269  0.00271 **
x.CHER_at     -0.76801    1.80022  -0.427  0.67270
x.NADB_at     -0.42792    0.29608  -1.445  0.15875
x.YURK_at     -0.40301    1.88493  -0.214  0.83214
x.YTAB_at      1.17299    1.97921   0.593  0.55785
x.YERO_at     -0.79305    2.55682  -0.310  0.75858
x.NADB_at.1          NA         NA      NA       NA
x.YRHA_at      0.03247    1.46138   0.022  0.98242
x.ALST_at     -0.21853    1.78775  -0.122  0.90352
x.PYRF_at      1.84829    1.41844   1.303  0.20247
x.CYSE_at      1.36118    1.88739   0.721  0.47637
x.FFH_at      -0.06067    1.28794  -0.047  0.96274
x.YHDS_r_at    2.33108    0.89617   2.601  0.01429 *
x.YVFK_at     -1.20590    2.89078  -0.417  0.67954
x.DPPD_at     -1.03051    2.17981  -0.473  0.63981
x.YEFA_at      2.48013    2.27678   1.089  0.28469
x.PRFA_at      0.29023    1.50563   0.193  0.84844
x.CSAA_at      1.12029    1.78597   0.627  0.53522
x.YEZC_at     -1.02434    2.03714  -0.503  0.61875
x.YVFO_at      1.75717    2.98629   0.588  0.56066
x.GLNM_at     -0.24351    1.54271  -0.158  0.87564
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.034 on 30 degrees of freedom
Multiple R-squared:  0.8277,    Adjusted R-squared:  0.598
F-statistic: 3.603 on 40 and 30 DF,  p-value: 0.0002349
```

Linear Regression




```
> print(bp_test)

        studentized Breusch-Pagan test

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

- **Linearity**
- **Homoscedasticity**

Linear Regression

```
> 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
```

- **Multicollinearity**

Linear Regression

```
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)
```

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

- **Multicollinearity**

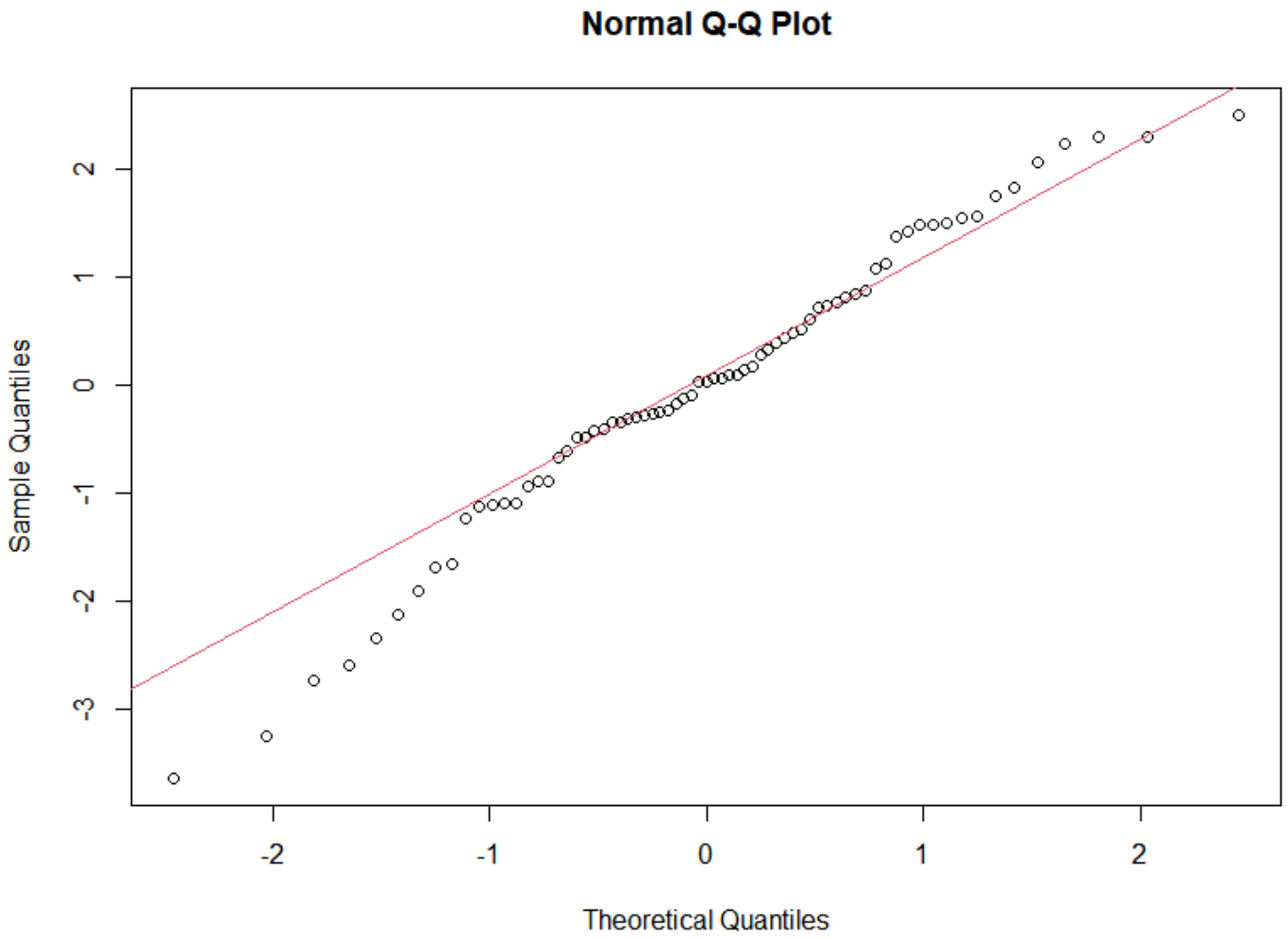
Linear Regression

```
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)
```

```
> 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
```

- **Multicollinearity**

Linear Regression




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

        Shapiro-Wilk normality test

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

- **Normality**

- **Full Model**

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

- **AIC**

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

- **BIC**

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

- **LASSO**

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

- **Full Model**

```
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
```

- **AIC✓**

```
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
```

- **BIC**

```
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
```

- **LASSO**

```
> 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
```

Other Methods

Multivariate Adaptive Regression Splines(MARS)

- R^2
- Adjusted R^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(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
```

- MSE

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

Classification And Regression Tree(CART)

- R^2
- Adjusted R^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
[1] 0.8319118
> adjusted_r_squared
[1] 0.8071119
```

- MSE

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

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

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

- LASSO

```
> 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

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

- Elastic Net

```
> 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
```

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

- **Elastic Net**

21

```
> 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
```

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

- **Average Elastic Net**

29

PCA+MARS v.s. Average Elastic Net

THANKS FOR LISTENING!