Statistical Learning Problem Set # 2

John Gaebler – joga5948@colorado.edu

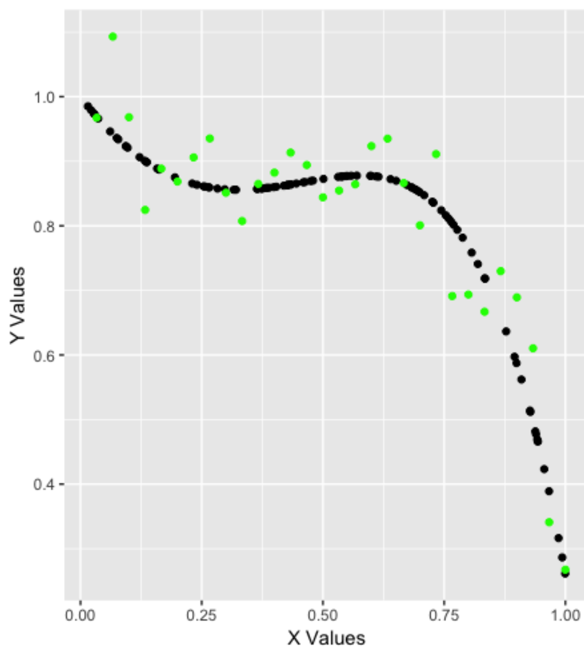
February 10, 2020

---

1. Creation of Datasets

   (a) Write a function that generates samples - see appendix code

   (b) Display the true function f(x) using 100 uniformly spaces samples on [0,1] and the 30 noisy samples on the same plot.



The black points correspond to the true function, and the green points are the noisy data sample

2. Estimation of the Polynomial

(a) Write a function that computes the optimal estimator $\hat{y}$ given a fixed order p. Give the coefficients generated for p = 1,4, and 15

Coefficients: p = 1

Intercept	x
1.0395355	-0.4407709

Coefficients: p = 4

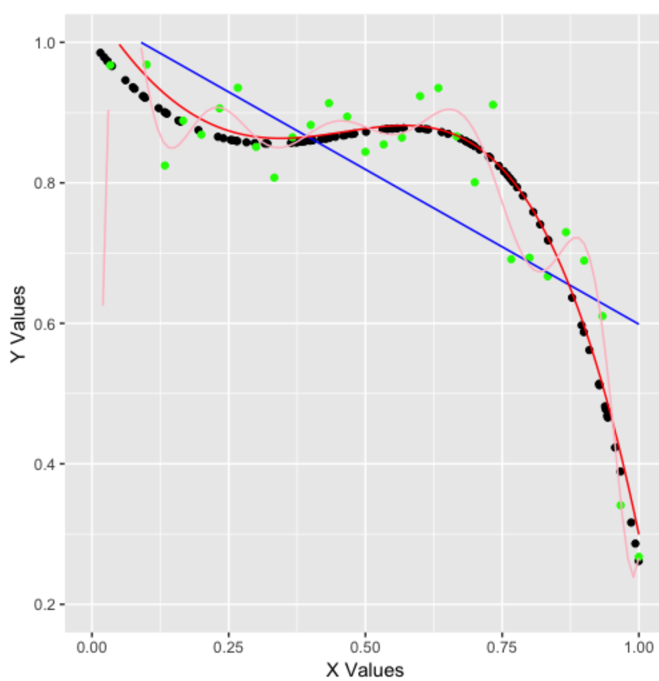
Intercept	x	x^2	x^3	x^4
1.05401141	-1.24916710	2.23042888	-0.02250314	-1.71335123

Coefficients: p = 15

Intercept	x	x^2	x^3	x^4	-
-5.228157e-01	8.330017e+01	-1.528828e+03	1.263023e+04	-4.689995e+04	-
x^5	x^6	x^7	x^8	x^9	-
4.022791e+03	6.963936e+05	-3.157657e+06	7.470297e+06	-1.087213e+07	-
x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}
9.938791e+06	-5.401311e+06	1.415607e+06	NA	-5.829778e+04	NA

Note: The x^{13} and x^{15} coefficients in the p = 15 model are NA because they are linearly related to other components of the model. Past degree 14, NAs appear for the coefficents with very high frequency.

(b) Display the three optimal polynomials as well as the true function and the noisy data on the same plot



Blue line: degree 1, Red line: degree 4, Pink line: degree 15

(c) Based on visual inspection, what is the optimal degree polynomial for:

   i. The best fit to the noisy data?
   Answer: The degree 15 polynomial best fits the noisy data points.

   ii. The best fit to the true function?
   Answer: The degree 4 polynomial best fits the path of the true function.

The higher degree polynomial can do a better job of fitting the variation in the data points, but it fails to capture the actual movement of the function. The lower degree polynomial may actually better approximate the true function even if it doesn't appear to follow the noisy data as well. We call the effect of matching the noisy data too closely and as a result loosing the true function "over-fitting the model."

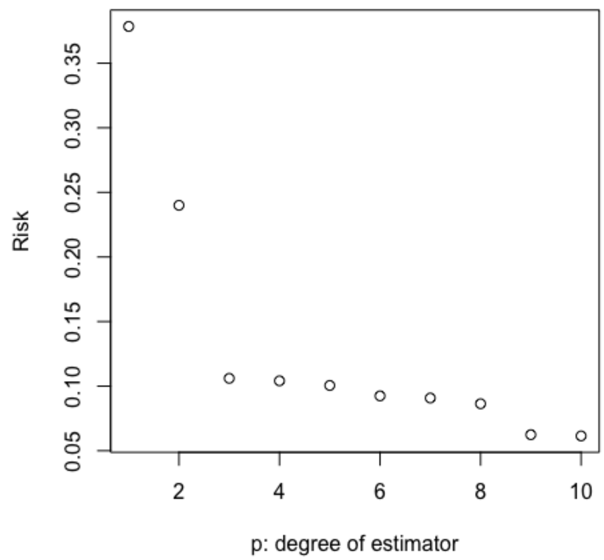
3. Choice of the Optimal Degree

(a) For p = 1, ... , 10 compute the values of $\hat{R}$. Display $\hat{R}$ as a function of p, and find the value of p that minimizes $\hat{R}$.

Values of $\hat{R}$ for each value of p:

p = 1	p = 2	p = 3	p = 4	p = 5
0.37845020	0.23997598	0.10603381	0.10410272	0.10051470
p = 6	p = 7	p = 8	p = 9	p = 10
0.09244547	0.09079719	0.08636445	0.06241269	0.06144891

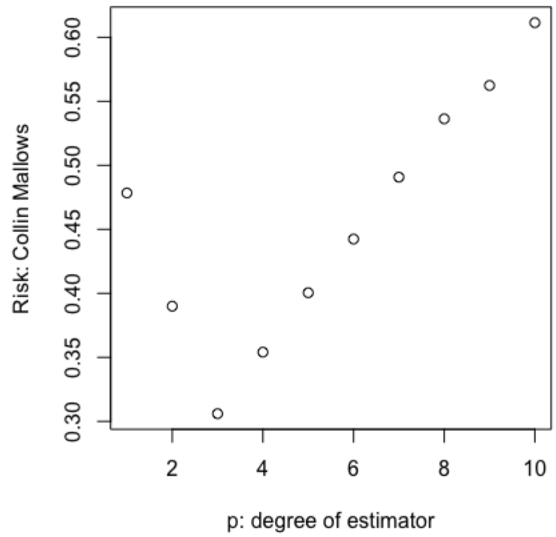
The value of p that minimizes $\hat{R}$ is p = 10.



(b) To penalize overly complex models, use the Collin Mallows statistic C_p to choose the optimal model. Display C_p as a function of , and find the value of p that minimizes C_p.

p = 1	p = 2	p = 3	p = 4	p = 5
0.4784502	0.3899760	0.3060338	0.3541027	0.4005147
p = 6	p = 7	p = 8	p = 9	p = 10
0.4424455	0.4907972	0.5363644	0.5624127	0.6114489

The value of p that minimizes C_p is p = 3.



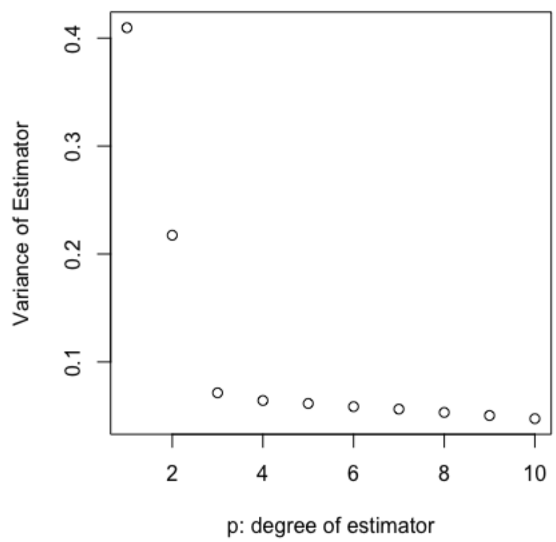
Risk: Collin Mallows
0.30
0.35
0.40
0.45
0.50
0.55
0.60
2
4
6
8
10
p: degree of estimator

4. Estimate of the Bias and Variance of the Estimator

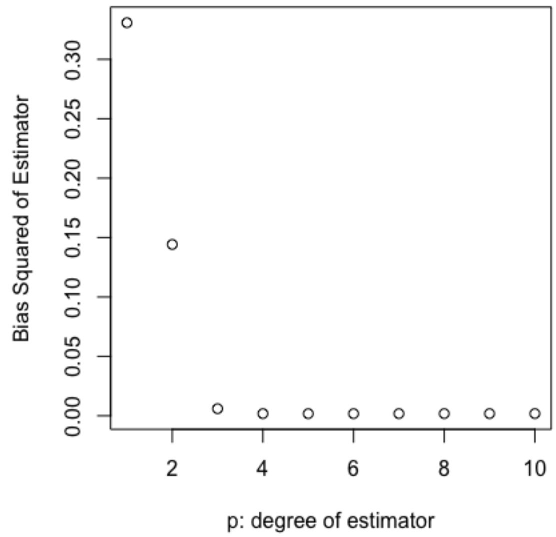
(a) Let S = 100. Generate S random realizations of $y^{(s)}$ using the code that you wrote in the first problem. For each $y^{(s)}$, compute the optimal estimate $\hat{y}^{(s)}$. See appendix code Finally, compute the Variance and the Bias of the estimator, and combine theses statistics to to compute the risk:

$$\tilde{R} = Var(\hat{y}) + bias^2(\hat{y})$$

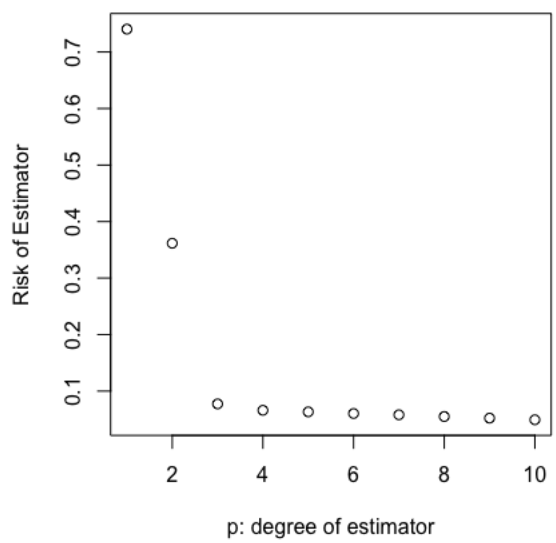
for p = 1,...,10. Then display $Var(\hat{y})$, $bias^2(\hat{y})$, and $\tilde{R}$ as functions of p. Describe and explain the behavior of the curves as p increases. What is the value of p that minimizes $\tilde{R}$?



The value of p that minimizes variance is p = 10



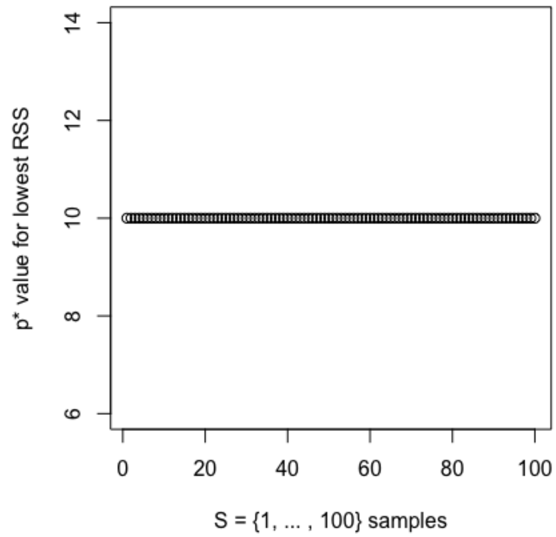
The value of p that minimizes bias squared is p = 10



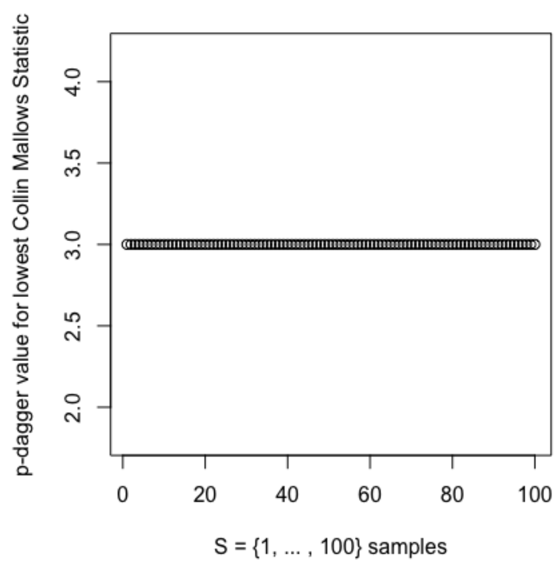
The value of p that minimizes Risk is p = 10

For all of the statistics, the values decreases as p increases. This is because a function with higher degree is going to be able to better approximate the sample data. For Bias squared in particular, the value rapidly approaches zero as early as p = 3. However, the higher degree estimator leads to an overly complex model that is likely to over fit the data and produce predictions that are less accurate than a more simple model.

(b) For each of the S data sets, compute the value p^* that minimizes the residual sum of squares. Additionally, compute the value p^+ that minimize the the statistic C_p.



For all the S = 100 samples, the degree 10 polynomial estimator produces the least value of Residual Sum of Squares



For all the S = 100 samples, the degree 3 polynomial estimator produces the least value of the Collin Mallows statistic.

The differences between these measures of risk demonstrates the importance of penalizing excessively complex models. There is an apparent large difference in the optimal degree of estimator between these two metrics of risk (3 vs 10). These estimates are very consistent across the S = 100 realizations of testing data, considering there is not a single deviation from p = 10 or p = 3 respectively.

Appendix: Code

```
#======================================================================
##----------------------Homework #2-------------------------------------
library(ggplot2)
#Generating data
x_Data <- seq(1/30,1, by = 1/30)
true_fun_in <- runif(100, 0 ,1)

#function to generate the samples
generate_Data <- function(x, sd =0) {
  data <- 1 - x + (2 * x ^2) - (.8 *  x ^ 3) + (.06 * x ^4) - (x ^ 5)
  data <- data + rnorm(30, 0, sd)
  return(data)
}


y_Sample <- generate_Data(x_Data, .05)
true_fun_out <- generate_Data(true_fun_in)
both_in <- append(true_fun_in, x_Data)
both_out<- append(true_fun_out, y_Sample)
y_Data

bigplot <- ggplot()+
  geom_point(aes(x = true_fun_in, y = true_fun_out))+
  geom_point(aes(x = x_Data, y = y_Sample), color = "green")+
  labs(x = "X Values", y = "Y Values")

##-------------------------------------------------------------

estimator_degree <- function(n, data = x_Data) {
  i = 1
  data_matrix <-  array(data = rep(x_Data, n), dim = c(length(x_Data),n))
  estimator <- c(rep(x_Data, n))
  while (i <= n) {
      data_matrix[,i] <- data_matrix[,i]^i
      i = i +1
  }
  return(data_matrix)
}

#----------------------------
p1_data <- estimator_degree(1)
p1 <- lm(y_Sample ~ p1_data)

degree1 <- function(x) 1.0395 + -0.4408 *x

```

```r
degree1 <- function(x) 1.0395 + -0.4408 *x

p2_data <- estimator_degree(2)
p2 <- lm(y_Sample ~ p2_data)

p3_data <- estimator_degree(3)
p3 <- lm(y_Sample ~ p3_data)

p4_data <- estimator_degree(4)
p4 <- lm(y_Sample ~ p4_data)

p5_data <- estimator_degree(5)
p5 <- lm(y_Sample ~ p5_data)

p6_data <- estimator_degree(6)
p6 <- lm(y_Sample ~ p6_data)

p7_data <- estimator_degree(7)
p7 <- lm(y_Sample ~ p7_data)

p8_data <- estimator_degree(8)
p8 <- lm(y_Sample ~ p8_data)

p9_data <- estimator_degree(9)
p9 <- lm(y_Sample ~ p9_data)

p10_data <- estimator_degree(10)
p10 <- lm(y_Sample ~ p10_data)

data_vec <- c(p1_data, p2_data, p3_data, p4_data, p5_data, p6_data, p7_data, p8_data, p9_data, p10_data)



degree4 <- function(x) 1.05401 -1.24917*x +2.23043 *x^2 -0.02250 * x^3 -1.71335 *x^4

p15_data <- estimator_degree(15)
p15 <- lm(y_Sample ~ p15_data)

degree4 <- function(x) {
  output <- 0
  for(i in 1:length(coef(p4))) {
    output <- output + as.numeric(coef(p4)[i]) * x ^(i-1)
  }
  return(output)
}
```

```r
degree4 <- function(x) {
  output <- 0
  for(i in 1:length(coef(p4))) {
    output <- output + as.numeric(coef(p4)[i]) * x ^(i-1)
  }
  return(output)
}
degree15 <- function(x) {
  output <- 0
  for(i in 1:length(coef(p15))) {
    output <- output + ifelse(is.na(as.numeric(coef(p15)[i]) * x ^(i-1)), 0,  as.numeric(coef(p15)[i]) * x ^(i-1))
  }
  return(output)
}

degree_n <- function(x, degree) {
  output <- 0
  reg <- eval(parse(text = paste("p", degree, sep = "")))
  for(i in 1:length(coef(reg))) {
    output <- output + ifelse(is.na(as.numeric(coef(reg)[i]) * x ^(i-1)), 0,  as.numeric(coef(reg)[i]) * x ^(i-1))
  }
  return(output)
}

bigplot + stat_function(fun = degree1, aes(x = x), data = data.frame(x = 0), color = "blue") +
  stat_function(fun = degree4, aes(x = x), data = data.frame(x = 0), color = "red") +
  stat_function(fun = degree15, aes(x = x), data = data.frame(x = 0), color = "pink") +
  ylim(.2,1)



Risk <- function(deg) {
  r_hat <- sum((y_Sample - degree_n(x_Data, deg))^2)
  return(r_hat)
}

risk_vec <- sapply(c(1:10), Risk)
plot(x = 1:10, y = risk_vec,
     xlab = "p: degree of estimator",
     ylab = "Risk")


c_p <- risk_vec + 2 * (c(2:11)) * .025

```

```r
c_p <- risk_vec + 2 * (c(2:11)) * .025

plot(x = 1:10, y = c_p,
     xlab = "p: degree of estimator",
     ylab = "Risk: Collin Mallows")
##-------- Looks like the optimal degree is 3


##-- Creatin the S = 100 random realization of y(s)
y_s <- rerun(100, generate_Data(x_Data, .05))

model_creator <- function(y, data) {
  return(lm(y ~ data))
}
## generating the p = {1,...10} models for each of the S = 100 realizations, 1000 total models
y_s_model_p1 <- lapply(y_s, model_creator, data = p1_data)
y_s_model_p2 <- lapply(y_s, model_creator, data = p2_data)
y_s_model_p3 <- lapply(y_s, model_creator, data = p3_data)
y_s_model_p4 <- lapply(y_s, model_creator, data = p4_data)
y_s_model_p5 <- lapply(y_s, model_creator, data = p5_data)
y_s_model_p6 <- lapply(y_s, model_creator, data = p6_data)
y_s_model_p7 <- lapply(y_s, model_creator, data = p7_data)
y_s_model_p8 <- lapply(y_s, model_creator, data = p8_data)
y_s_model_p9 <- lapply(y_s, model_creator, data = p9_data)
y_s_model_p10 <- lapply(y_s, model_creator, data = p10_data)

model_vec <- list(y_s_model_p1, y_s_model_p2, y_s_model_p3, y_s_model_p4, y_s_model_p5, y_s_model_p6,
                  y_s_model_p7, y_s_model_p8, y_s_model_p9, y_s_model_p10)

mass_predictor <- function(model) {
  fitted <- 0
  reg <- model
  for(i in 1:length(coef(reg))) {
    fitted <- fitted + ifelse(is.na(as.numeric(coef(reg)[i]) * x_Data ^(i-1)), 0,
                              as.numeric(coef(reg)[i]) * x_Data ^(i-1))
  }
 return(fitted)
}
## creating fitted data with the models, 10 fitted sets per S = 100 datasets, 1000 fitted datasets
fitted_vec_p1 <- lapply(y_s_model_p1, mass_predictor)
fitted_vec_p2 <- lapply(y_s_model_p2, mass_predictor)
fitted_vec_p3 <- lapply(y_s_model_p3, mass_predictor)
fitted_vec_p4 <- lapply(y_s_model_p4, mass_predictor)
fitted_vec_p5 <- lapply(y_s_model_p5, mass_predictor)
```

```r
## creating fitted data with the models, 10 fitted sets per S = 100 datasets, 1000 fitted datasets
fitted_vec_p1 <- lapply(y_s_model_p1, mass_predictor)
fitted_vec_p2 <- lapply(y_s_model_p2, mass_predictor)
fitted_vec_p3 <- lapply(y_s_model_p3, mass_predictor)
fitted_vec_p4 <- lapply(y_s_model_p4, mass_predictor)
fitted_vec_p5 <- lapply(y_s_model_p5, mass_predictor)
fitted_vec_p6 <- lapply(y_s_model_p6, mass_predictor)
fitted_vec_p7 <- lapply(y_s_model_p7, mass_predictor)
fitted_vec_p8 <- lapply(y_s_model_p8, mass_predictor)
fitted_vec_p9 <- lapply(y_s_model_p9, mass_predictor)
fitted_vec_p10 <- lapply(y_s_model_p10, mass_predictor)

fitted_vec <- list(fitted_vec_p1, fitted_vec_p2, fitted_vec_p3, fitted_vec_p4, fitted_vec_p5,
                fitted_vec_p6, fitted_vec_p7, fitted_vec_p8, fitted_vec_p9, fitted_vec_p10)


Var_calc <- function(fit) {
  b <- mapply('-', y_s, fit)
  RSS <- 0
  for(i in 1:100) {
    RSS <- RSS + sqrt(sum(b[,i]^2))^2
  }
  return(RSS / 99)
}

Variance_vec <- lapply(fitted_vec, Var_calc)
# creating the sample means for the optimal estimated for each p = {1, ... , 10}
y_bar <- lapply(fitted_vec, function(x) Reduce("+", x)/ 99)

true_Values <- generate_Data(x_Data)
true_Values_vec <- list(true_Values, true_Values, true_Values, true_Values, true_Values,
                        true_Values, true_Values, true_Values, true_Values, true_Values)

bias <- mapply('-', true_Values_vec, y_bar)
for(i in 1:10) {
  bias_squared[i] <- sqrt(sum(bias[,i]^2))^2
}


Risk_Model <- mapply('+', Variance_vec, bias_squared)
```

```r
##  Plots

plot(x = 1:10, y = Variance_vec,
     xlab = "p: degree of estimator",
     ylab = "Variance of Estimator")

plot(x = 1:10, y = bias_squared,
     xlab = "p: degree of estimator",
     ylab = "Bias Squared of Estimator")

plot(x = 1:10, y = Risk_Model,
     xlab = "p: degree of estimator",
     ylab = "Risk of Estimator")

match(min(Risk_Model), Risk_Model)

## complute the RSS for the models p 1-10

RSS <- function( model) {
  output <- mapply('-', y_s, model)
  output1 <- c(1:100)
  for(i in 1:100) {
    output[,i] <-output[,i]^2
    output1[i] <- sum(output[,i])
  }
  return(output1)
}

p_RSS <- lapply(fitted_vec, RSS)

p_star <- c(1:100)

for(i in 1:100) {
  vec <- c(1:10)
  for(j in 1:10) {
    vec[j] <- p_RSS[[j]][i]
  }
  p_star[i] <- match(min(vec), vec)
}
```

```r
## complute the RSS for the models p 1-10

RSS <- function( model) {
  output <- mapply('-', y_s, model)
  output1 <- c(1:100)
  for(i in 1:100) {
    output[,i] <-output[,i]^2
    output1[i] <- sum(output[,i])
  }
  return(output1)
}

p_RSS <- lapply(fitted_vec, RSS)

p_star <- c(1:100)

for(i in 1:100) {
  vec <- c(1:10)
  for(j in 1:10) {
    vec[j] <- p_RSS[[j]][i]
  }
  p_star[i] <- match(min(vec), vec)
}


plot(x = 1:100, y = p_star,
     ylab = "p* value for lowest RSS",
     xlab = "S = {1, ... , 100} samples")



c_of_p <- c(1:100)

for(i in 1:100) {
  vec <- c(1:10)
  for(j in 1:10) {
    vec[j] <- p_RSS[[j]][i] + 2* Variance_vec[[j]] * (j +1)
  }
  c_of_p[i] <- match(min(vec), vec)
}

plot(x = 1:100, y = c_of_p,
     ylab = "p-dagger value for lowest Collin Mallows Statistic",
     xlab = "S = {1, ... , 100} samples")

```