HW3

```
library(dplyr)
library(readr)
library(ggplot2)
library(splines)
library(Matrix)
library(mgcv)
```

Question 1

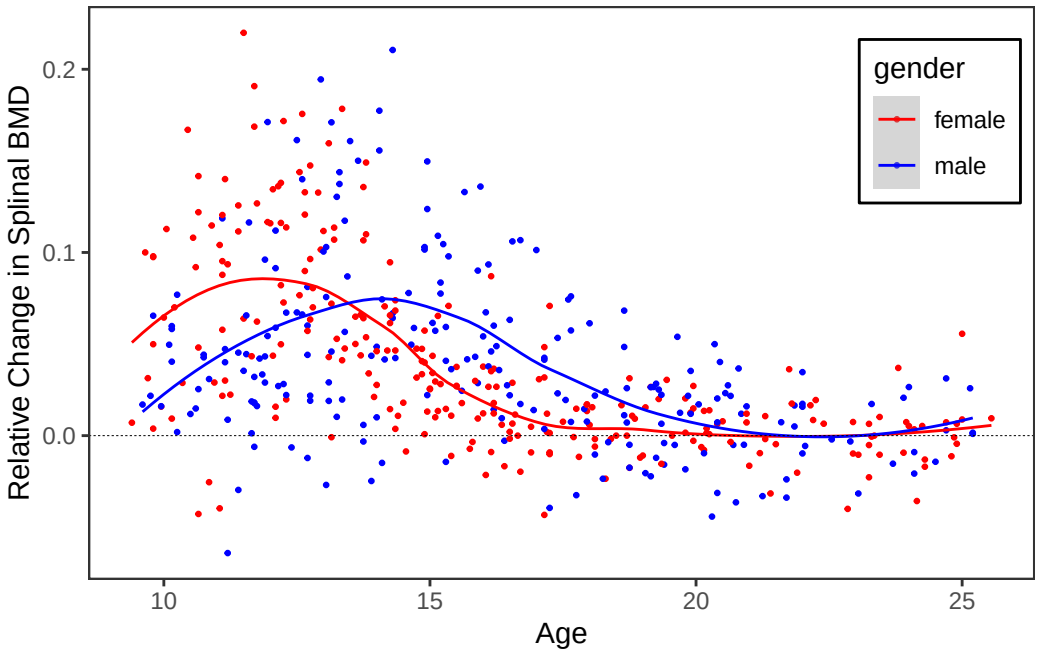
Replicate the figure 5.6

```
q1<-read_table(
"https://hastie.su.domains/ElemStatLearn/datasets/bone.data")%>%as.data.frame()
```

```
ggplot(data=q1)+
  geom_point(aes(x=age,y=spnbmd,color=gender),size=0.5)+
  geom_smooth(aes(x=age,y=spnbmd,color=gender),level=0,size=0.5)+
  scale_color_manual(values= c("male"="blue","female"="red"))+
  labs(x="Age",y="Relative Change in Splinal BMD")+
  theme_bw()+
  geom_hline(yintercept=0,linetype=2,size=0.1)+
  theme(
    panel.grid = element_blank(),
  # legend.background = element_blank(),
    legend.position = c(.9, .8),
   # legend.box.background = element_rect(color="black", size=0.05),
   # legend.box.margin = margin(6, 6, 6, 6),
    legend.background = element_rect(fill = "white", color = "black"))
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Question 2

```
df<-read_csv("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data")

View(df)
```

Spline basis

B-spline basis

```
b_spline<-bs(df$tobacco,df=6, intercept = FALSE)
```

Natural Spline Basis

```
n_spline<- ns(df$tobacco, df = 4, intercept = FALSE)
```

Truncated polynomial spline bases

```
t_spline<- bs(df$tobacco,df=6, intercept = TRUE)
```

Logistic Regressions

```
b_logistic<-glm(df$chd ~ b_spline,family = binomial(link = "logit"))
b_logistic
```

```
Call:  glm(formula = df$chd ~ b_spline, family = binomial(link = "logit"))

Coefficients:
(Intercept)    b_spline1    b_spline2    b_spline3    b_spline4    b_spline5
    -1.7800      -0.6808       1.7603       1.0070       3.0172       1.5492
  b_spline6
     5.6237

Degrees of Freedom: 461 Total (i.e. Null);  455 Residual
Null Deviance:      596.1
Residual Deviance: 538.4    AIC: 552.4
```

```
n_logistic<-glm(df$chd ~ n_spline,family = binomial(link = "logit"))
n_logistic
```

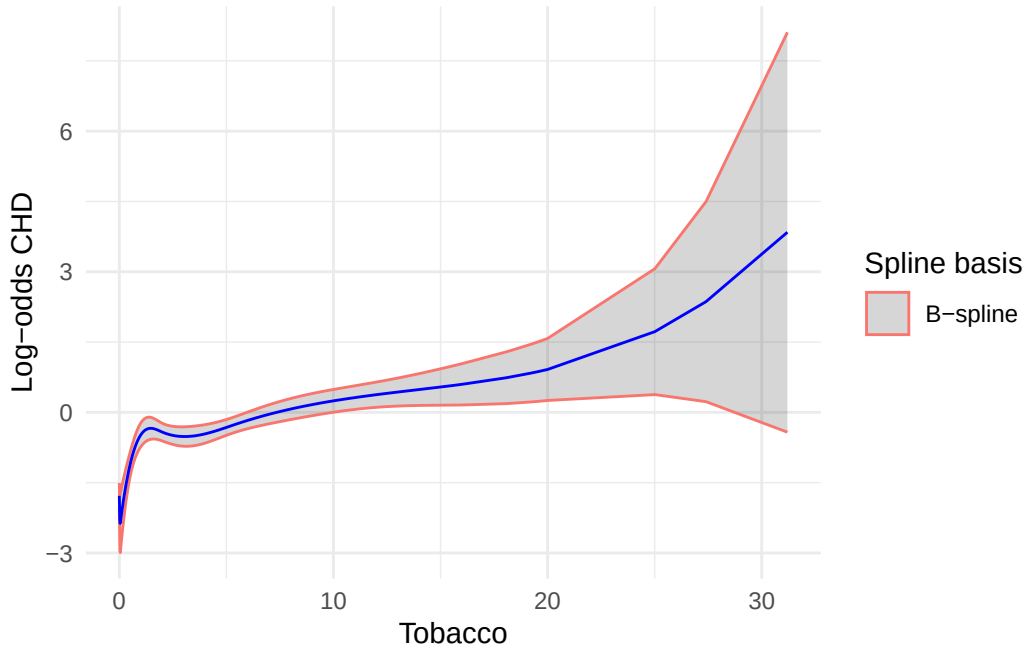
```
Call:  glm(formula = df$chd ~ n_spline, family = binomial(link = "logit"))

Coefficients:
(Intercept)    n_spline1    n_spline2    n_spline3    n_spline4
     -1.896        1.323        1.813        4.558        3.328

Degrees of Freedom: 461 Total (i.e. Null);  457 Residual
Null Deviance:      596.1
Residual Deviance: 539.8    AIC: 549.8
```

```r
t_logistic<-glm(df$chd ~ t_spline,family = binomial(link = "logit"))
t_logistic
```

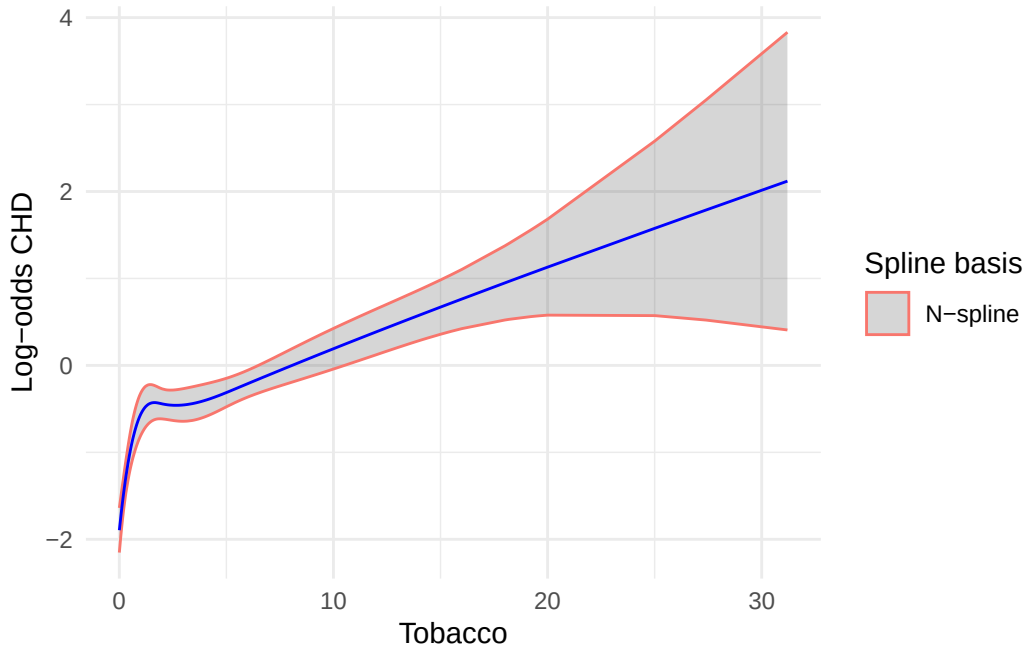
```
Call:  glm(formula = df$chd ~ t_spline, family = binomial(link = "logit"))

Coefficients:
(Intercept)    t_spline1    t_spline2    t_spline3    t_spline4    t_spline5
      2.610       -4.487       -3.749       -3.001       -2.631       -1.115
  t_spline6
         NA

Degrees of Freedom: 461 Total (i.e. Null);  456 Residual
Null Deviance:      596.1
Residual Deviance: 541.1    AIC: 553.1
```

Predicted Model and Variance

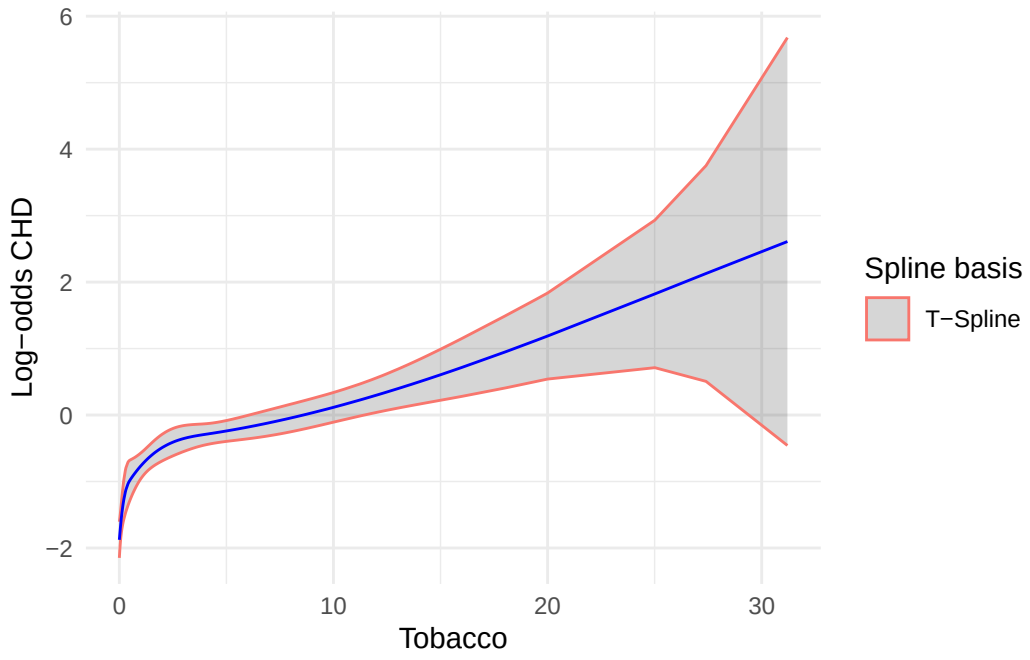
```r
fit_bspline <- glm(chd ~ b_spline, data = df, family = binomial())
pred_bspline <- predict(fit_bspline, type = "link")
var_bspline <- predict(fit_bspline, type = "link", se.fit = TRUE)$se.fit^2
plot_data_bspline <- data.frame(tobacco = df$tobacco, pred = pred_bspline,
                                upper = pred_bspline + sqrt(var_bspline),
                                lower = pred_bspline - sqrt(var_bspline), spline = "B-spli

ggplot(plot_data_bspline, aes(x = tobacco, y = pred, colour = spline)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
  geom_line(color="blue")+
  labs(x = "Tobacco", y = "Log-odds CHD", colour = "Spline basis") +
  theme_minimal()
```




```
fit_nspline <- glm(chd ~ n_spline, data = df, family = binomial())
pred_nspline <- predict(fit_nspline, type = "link")
var_nspline <- predict(fit_nspline, type = "link", se.fit = TRUE)$se.fit^2
plot_data_nspline <- data.frame(tobacco = df$tobacco, pred = pred_nspline,
                                upper = pred_nspline + sqrt(var_nspline),
                                lower = pred_nspline - sqrt(var_nspline), spline = "N-spli

ggplot(plot_data_nspline, aes(x = tobacco, y = pred, colour = spline)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
  geom_line(color="blue")+
  labs(x = "Tobacco", y = "Log-odds CHD", colour = "Spline basis") +
  theme_minimal()
```



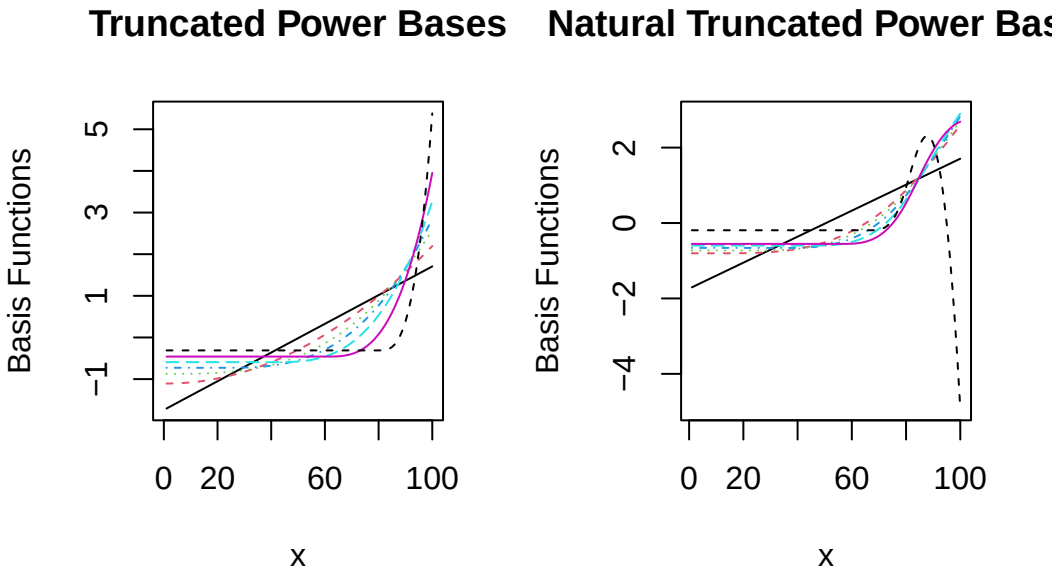

```
fit_tspline <- glm(chd ~ t_spline, data = df, family = binomial())
pred_tspline <- predict(fit_tspline, type = "link")
var_tspline <- predict(fit_tspline, type = "link", se.fit = TRUE)$se.fit^2
plot_data_tspline <- data.frame(tobacco = df$tobacco, pred = pred_tspline,
                                upper = pred_tspline + sqrt(var_tspline),
                                lower = pred_tspline - sqrt(var_tspline), spline = "T-Spli

ggplot(plot_data_tspline, aes(x = tobacco, y = pred, colour = spline)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
  geom_line(color="blue")+
  labs(x = "Tobacco", y = "Log-odds CHD", colour = "Spline basis") +
  theme_minimal()
```



Question 3

```r
polyspline_truncated <- function(input_x, df, natural_spline = FALSE) {
  calculate_trunc_fun <- function(knot) (input_x >= knot) * (input_x - knot)^3

  if (natural_spline) {
    knot_vals <- quantile(input_x, seq(0, 1, length = df + 3))
    xi_K = knot_vals[df + 2]
    xi_K_1 = knot_vals[df + 1]
    d_K_1 = ((input_x >= xi_K_1) * (input_x - xi_K_1)^3 - (input_x >= xi_K) * (input_x - x
    spline_matrix <- sapply(knot_vals[2:(df)], function(k) calculate_trunc_fun(k) - ((inpu
    spline_matrix <- as(spline_matrix, "CsparseMatrix")
    spline_matrix <- cbind(input_x, spline_matrix)
  } else {
    knot_vals <- quantile(input_x, seq(0, 1, length = df - 1))
    spline_matrix <- sapply(knot_vals[1:(df - 2)], function(k) calculate_trunc_fun(k))
    spline_matrix <- as(spline_matrix, "CsparseMatrix")
    spline_matrix <- cbind(input_x, input_x^2, spline_matrix)
  }

  return(spline_matrix)
}
```

```r
x <- seq(0, 2, length.out = 100)
df <- 7
#spline bases without natural constraints
regular_spline_bases <- polyspline_truncated(x, df, natural_spline = FALSE)

# spline bases with natural constraints
natural_spline_bases <- polyspline_truncated(x, df, natural_spline = TRUE)

# Plot
par(mfrow = c(1, 2))
matplot(scale(regular_spline_bases), type = "l", main = "Truncated Power Bases", xlab = "x
matplot(scale(natural_spline_bases), type = "l", main = "Natural Truncated Power Bases", x
```



Question 4

(a) generate data

To generate the desired data, we use

$$f(x,y) = \sum_j w_j \exp\left(-\frac{(x-x_j)^2 + (y-y_j)^2}{s_j^2}\right)$$

mentioned in the homework, where w_j is the weight and s_j is the scale.

```r
simulate_data <- function(n, noise_sd) {
  # Generate random x and y
  x <- runif(n)
  y <- runif(n)

  # random centers, scales, weights
  centers <- cbind(runif(2), runif(2))####
  scales <- runif(2)
  weights <- runif(2)
  weights <- weights / sum(weights)

  # Compute the function for each xy pairs
  f <- function(x, y, centers, scales, weights) {
    val <- 0
    for (i in 1:length(weights)) {
      val <- val + weights[i] * exp(-((x-centers[i,1])^2 + (y-centers[i,2])^2)/(2*scales[i
    }
    return(val)
  }
  z <- f(x, y, centers, scales, weights) + rnorm(n, mean = 0, sd = noise_sd)

  return(data.frame(x = x, y = y, z = z))
}

#simulated_data <- simulate_data(n = 5000, noise_sd = 0)
#ggplot(simulated_data, aes(x = x, y = y)) +
#  geom_point(aes(color = z), size = 1) +
#  scale_color_gradient(low = "white", high = "blue") +
#  theme_minimal()
```

(b) bench mark

We generate 250 different simulated dataset. For each dataset, we fit 2D spline, and benchmark the two methods for 250 times.

```r
library(microbenchmark)
#Initial setup
#Set the number of simulations to be 250 trials
n_sim<-250
results_gcv <- list()
```

```r
results_reml <- list()

#Generated simulated data
for (i in 1:n_sim){
simulated_data<-simulate_data(200,0.4)

#Functions of GCV.Cp and REML
gcv<-gam(
    z ~ te(x, y, bs = "gp"),
    data = simulated_data,
    family = gaussian(),
    method = "GCV.Cp",
    fit = TRUE
  )

reml<-gam(
    z ~ te(x, y, bs = "gp"),
    data = simulated_data,
    method = "REML",
    fit = TRUE
  )
# Benchmark GCV.Cp and REML
result_g <- microbenchmark(gcv, times = 250,unit="ms")
result_r <- microbenchmark(reml, times = 250,unit="ms")
# bias, variance, and mean squared errors
results_gcv[[i]] <- list(

    time = result_g$time,
    bias = mean(gcv$fitted.values - simulated_data$z),
    var = var(gcv$fitted.values),
    mse = mean((gcv$fitted.values - simulated_data$z)^2)
  )

  results_reml[[i]] <- list(
    time = result_r$time,#unit microseconds
    bias = mean(reml$fitted.values - simulated_data$z),
    var = var(reml$fitted.values),
    mse = mean((reml$fitted.values - simulated_data$z)^2)
  )
 }
```

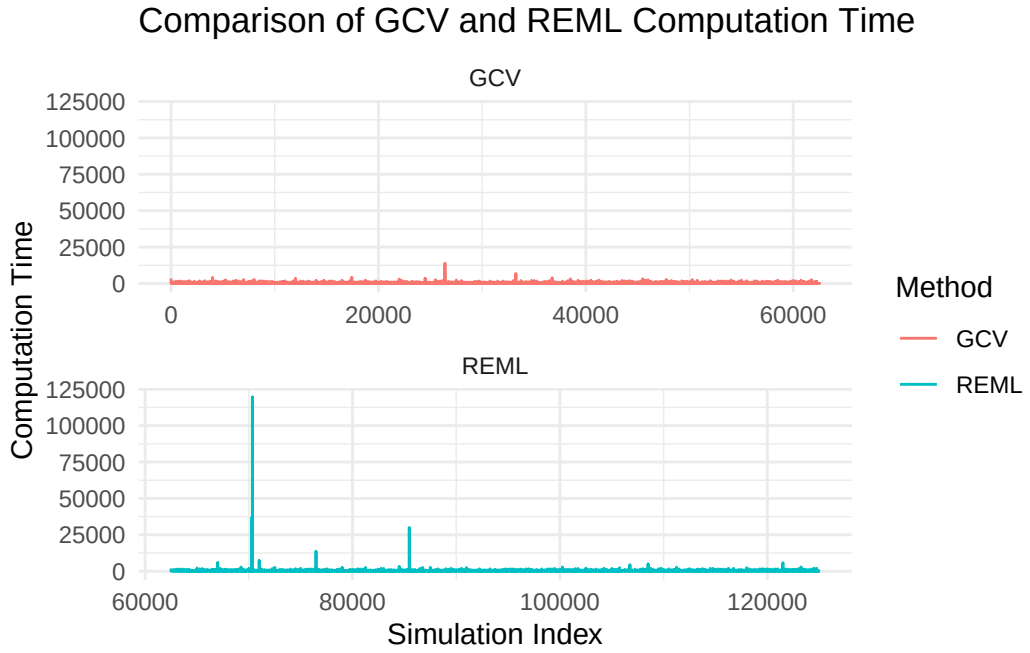
```r
# Convert the lists to df
df_gcv <- do.call(rbind, lapply(results_gcv, as.data.frame))
df_reml <- do.call(rbind, lapply(results_reml, as.data.frame))

# Set the row names to NULL for proper indexing
rownames(df_gcv) <- NULL
rownames(df_reml) <- NULL
```

```r
#average computation time
#mean(df_gcv$time)
#mean(df_reml$time)

#average bias
#mean(df_gcv$bias)
#mean(df_reml$bias)

#average variance
#mean(df_gcv$var)
#mean(df_reml$var)

#average mse
#mean(df_gcv$mse)
#mean(df_reml$mse)

library(knitr)

# Create df
avg_values <- data.frame(
  Metric = c("Avg. Time", "Avg. Bias", "Avg. Variance", "Avg. MSE"),
  GCV = c(mean(df_gcv$time), mean(df_gcv$bias), mean(df_gcv$var), mean(df_gcv$mse)),
  REML = c(mean(df_reml$time), mean(df_reml$bias), mean(df_reml$var), mean(df_reml$mse))
)

# Create a table using knitr::kable()
kable(avg_values, row.names = FALSE)
```

Metric	GCV	REML
Avg. Time	42.4814880	45.2980960
Avg. Bias	0.0000000	0.0000000
Avg. Variance	0.0297183	0.0268075

Metric	GCV	REML
Avg. MSE	0.1523066	0.1558443

```r
library(ggplot2)

df_gcv$Method <- "GCV"
df_reml$Method <- "REML"

combined_df <- rbind(df_gcv, df_reml)

# Create line plot for GCV and REML to compare the time
ggplot(combined_df, aes(x = 1:nrow(combined_df), y = time, group = Method, color = Method)) +
  geom_line() +
  facet_wrap(~Method, nrow = 2, scales = "free_x") +
  labs(title = "Comparison of GCV and REML Computation Time",
       x = "Simulation Index",
       y = "Computation Time") +
  theme_minimal()
```



After several attempt,with sample size = 200, `gcv` is not always faster than `reml`. However, it always have a slightly larger variance and smaller mse compared with `reml`. The bias of the two methods are almost the same.