Locally Adaptive Bayesian Function Estimation

Preloaded Function

Functions to generate data.

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:MASS':
##
##     select

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

Glmnet Ridge Regression

This is using the glmnet packages to use cross validate wether I've implemented Ridge regression correctly, which I think I have. This could be used later for Lasso regression in our method to reconstruct our parameter.

```
FitBridge <- function(X, data,sigma, a){
  # X:  "Design" matrix.
  # data: observation passed in.
  # a:  the alpha parameter, 0 for ridge regression and 1 for lasso. Anything in input between 0
  #     and 1 is a combination of ridge and fused
  return.data <- list()
  library(glmnet)
  lambdas_to_try <- 10^seq(-3, 5, length.out = 100)
  # Setting alpha = 0 implements ridge regression
  kfold <- length(data)*0.2
  if(kfold < 4){
    kfold <- 4
  }
  ridge_cv <- cv.glmnet(X,
                        data,
                        alpha = a,
                        lambda = lambdas_to_try,
                        standardize = TRUE,
                        nfolds = kfold)
  # Plot cross-validation results
  lambda_cv <- ridge_cv$lambda.min
  chosenTau <- lambda_cv
  model_cv <- glmnet(X, data, alpha = a, lambda = lambda_cv, standardize = TRUE)
```

```r
  y_hat_cv <- predict(model_cv, X)
  BIC <- -0.5*sigma^(-2)*(t(data-y_hat_cv)%*%(data-y_hat_cv)) - 0.5*log1p(length(data))*model_cv$df
  #print(paste("FIT: ", -0.5*sigma^(-2)*(t(data-y_hat_cv)%*%(data-y_hat_cv))))
  #print(paste("COMPLEXITY: ", 0.5*log1p(length(data))*model_cv$df))

  return.data$beta <- coef(model_cv, s=model_cv$lambda.min)
  return.data$lambda <- chosenTau
  return.data$max <- BIC
  return.data$fit <- y_hat_cv
  return.data$df <- model_cv$df
  return.data$ridge <- ridge_cv
  return(return.data)
}
```

Ridge Regression for constant function

This is a Ridge regression for a constant regression. It produce a horizontal line but has the property that as $\lambda \to \infty$ this horizontal line shrink to 0. On the other hand as $\lambda \to 0$ then horizontal line is at $\bar{y}$.

```r
FitConstant <- function(X,ynew, numberOfSteps,sigma){
  return.data <- list()

  n <- length(ynew)
  p <- ncol(X)
  L <- diag(p)
  BIC <- rep(NA,numberOfSteps)
  tau <- rep(NA,numberOfSteps)
  for(i in 1:numberOfSteps){
    tau[i] <- (i/numberOfSteps)*100
    beta <- solve(t(rep(1,p))%*%(t(X)%*%X + ((sigma^2)/(tau[i]^2)*t(L)%*%L))%*%rep(1,p))%*%t(rep(1,p))%
    beta <- matrix(beta*rep(1,p),ncol=1)
    BIC[i] <- -1/(2*sigma^2)*t(ynew-X%*%beta)%*%(ynew-X%*%beta) - (1/2)*log1p(n)
  }
  max = max(BIC, na.rm=TRUE)
  maxTau <- 0
  for(j in 1:numberOfSteps){
    if(!is.na(BIC[j])){
      if(BIC[j]==max){
        maxTau <- tau[j]
        B <- solve(t(rep(1,p))%*%(t(X)%*%X + ((sigma^2)/(maxTau^2)*t(L)%*%L))%*%rep(1,p))%*%t(rep(1,p))
        beta <- B%*%ynew
        Cov <- sigma*B%*%t(B)
      }
    }
  }
  beta <- matrix(beta*rep(1,p),ncol=1)
  return.data$beta <- beta
  return.data$maxBIC <- max
  return.data$BIC <- BIC
  return.data$maxTau <- maxTau
  return.data$Cov <- diag(Cov)
  return.data$fit <- X%*%beta
  return(return.data)
```

```
}
```

Generalised Ridge Regression

FitRidge function is a generalised ridge regression where if L matrix is not specified then this is a ordinary Ridge regression. The formula for generalised ridge regression is:

$$\underline{\hat{f}}_{p\times 1} = (X^T_{p\times n}X_{n\times p} + \lambda * L^T_{p\times m}L_{m\times p})^{-1}X^T_{p\times n}\underline{y}_{n\times 1}$$

Efficient Implementation. The problem here is computing the inverse when p is large. There is a small trick in order to reduce the time it takes to invert. This is only appropriet when n is small. The trick is to use single value decomposition on $X_{n\times p} = U_{n\times k}D_{k\times n}V^T_{n\times p}$. SVD of $X_{n\times p}$ means the matrix $V_{p\times n}$ is orthogonal hence our estimates is:

$$\underline{\hat{f}}_{p\times 1} = (V_{p\times n}R^T_{n\times n}R_{n\times n}V^T_{n\times p} + \lambda * L^T_{p\times m}L_{m\times p})^{-1}X^T_{p\times n}\underline{y}_{n\times 1}$$

let $\Delta = L^T_{p\times m}L_{m\times p}$ then we can apply the same decomposition to Δ using $V_{p\times n}$ hence $\Delta = V_{p\times n}\Lambda V^T_{n\times p}$ hence:

$$\underline{\hat{f}}_{p\times 1} = V_{p\times n}(D^T_{n\times k}D_{k\times n} + \lambda * \Lambda_{n\times n})^{-1}D^T_{n\times k}U^T_{k\times n}\underline{y}_{n\times 1}$$

Hence $(R^T_{n\times n}R_{n\times n} + \lambda * \Lambda)^{-1}$ is a $n \times n$ matrix which is the number of observation. Hence we can controll the size of the inverse matrix.

SVD Implementation

```
checkInverse <- function(m) class(try(solve(m),silent=T))=="matrix"
FitRidge <- function(X, y, L = NA, sigma, min , max){
  return.data <- list()
  SVD <- svd(K)
  if(is.na(L)){
    L <- diag(ncol(K))
  }

  tau_to_try <- 10^seq(min, max, length.out = 100)
  BIC <- rep(NA, length(tau_to_try))

  for(i in 1:length(tau_to_try)){
    beta <- SVD$v%*%solve(diag(SVD$d)%*%diag(SVD$d) + (sigma^2/tau_to_try[i]^2)*t(SVD$v)%*%t(L1)%*%L1%*
    BIC[i] <- -0.5*sigma^(-2)*t(y-K%*%beta)%*%(y-K%*%beta) - 0.5*log1p(length(y))*sum(diag(K%*%solve(t(
  }

  maxBIC <- max(BIC, na.rm = TRUE)
  maxTau <- NA
  fit <- rep(NA,length(y))
  beta <- rep(NA, ncol(X))
  for (i in 1:length(BIC)) {
    if(!is.na(BIC[i])){
      if(BIC[i]==maxBIC){
        maxTau <- tau_to_try[i]
        #print(maxTau)
        beta <- solve(t(K)%*%K + (sigma^2/maxTau^2)*t(L)%*%L,t(K)%*%y)
        fit <- K%*%beta
      }
    }
```

```
  }

  return.data$maxBIC <- maxBIC
  return.data$BIC <- BIC
  return.data$maxTau <- maxTau
  return.data$beta <- beta
  return.data$fit <- fit
  #print(return.data)
  return(return.data)
}
```

Non-SVD Implementation

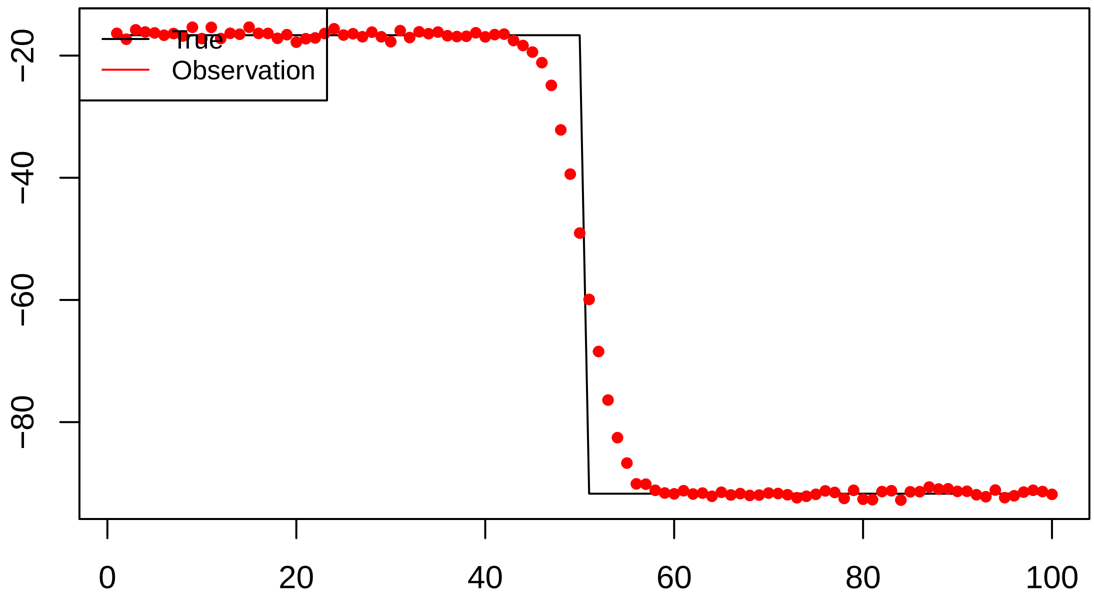
```
FitRidge <- function(K, y, L = NA, sigma, min , max){
  return.data <- list()

  if(is.na(L)){
    L <- diag(ncol(K))
  }

  tau_to_try <- 10^seq(min, max, length.out = 100)
  BIC <- rep(NA, length(tau_to_try))

  for(i in 1:length(tau_to_try)){
    if(checkInverse(t(K)%*%K + (sigma^2/tau_to_try[i]^2)*t(L)%*%L)){
      beta <- solve(t(K)%*%K + (sigma^2/tau_to_try[i]^2)*t(L)%*%L, t(K)%*%y)
      BIC[i] <- -0.5*sigma^(-2)*t(y-K%*%beta)%*%(y-K%*%beta) - 0.5*log1p(length(y))*sum(diag(K%*%solve(t
    }else{
      #print(tau_to_try[i])
    }
  }

  maxBIC <- max(BIC, na.rm = TRUE)
  maxTau <- 0.1
  fit <- rep(NA,length(y))
  beta <- rep(NA, ncol(K))
  for (i in 1:length(BIC)) {
    if(!is.na(BIC[i])){
      if(BIC[i]==maxBIC){
        maxTau <- tau_to_try[i]
        #print(maxTau)
        beta <- solve(t(K)%*%K + (sigma^2/maxTau^2)*t(L)%*%L,t(K)%*%y)
        fit <- K%*%beta
      }
    }
  }

  return.data$maxBIC <- maxBIC
  return.data$BIC <- BIC
  return.data$maxTau <- maxTau
  return.data$beta <- beta
  return.data$fit <- fit
  #print(return.data)
  return(return.data)
```

```
}
```

Simulated Data

Randomly generated data for our method to use.

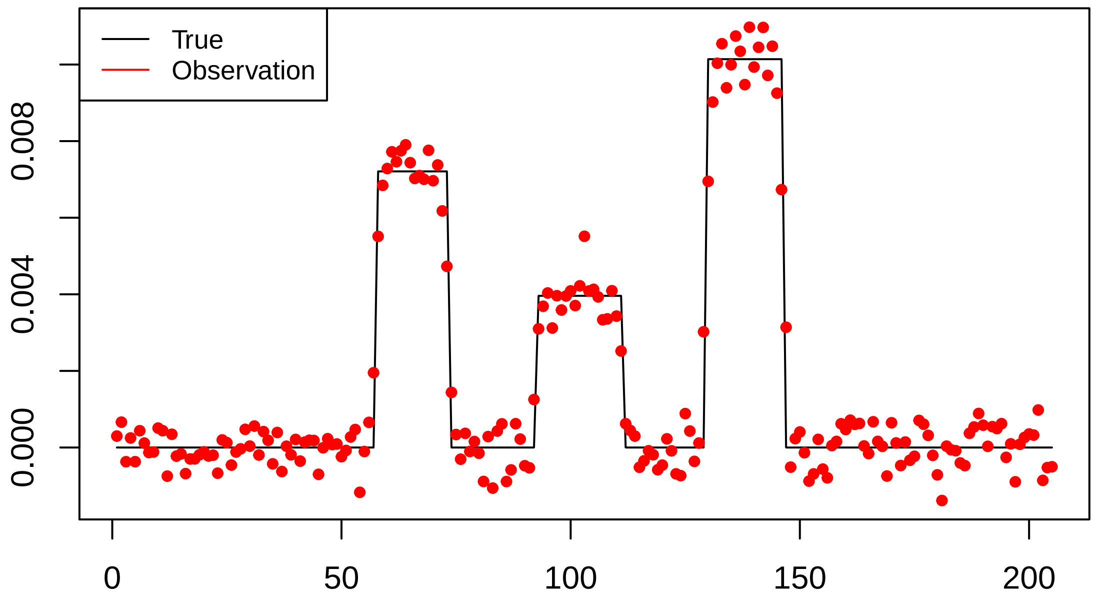
```
m <- 100
# the width of each function, default is 10. This mean each function generated occupy 10 values.
numberOfFunction <- 2
functionWidth <- m/numberOfFunction
y <- rep(0,m)
for(i in 1:numberOfFunction){
  ran <- 1#round(runif(1,0.75,3.25))
  if(ran==1){
    temp <- ConstantFunction(functionWidth)
    for(j in 1:length(temp)){
      y[(i-1)*functionWidth+j] <- temp[j]
    }
  }
  if(ran==2){
    temp <- c(LinearFunction(functionWidth))
    for(j in 1:length(temp)){
      y[(i-1)*functionWidth+j] <- temp[j]
    }
  }
  if(ran==3){
    temp <- c(QuadraticFunction(functionWidth))
    for(j in 1:length(temp)){
      y[(i-1)*functionWidth+j] <- temp[j]
    }
  }
}

# Add noise to the data
sigma <- 0.5
K <- getKG1D(n=length(y),delta = 3)
n <- m
Covariance <- sigma^2*diag(n)
mu <- rep(0,m)
f <- y
ynew <- K%*%f + mvrnorm(1,mu,Covariance)
min <- min(c(y,ynew))
max <- max(c(y,ynew))
plot(y=y,x=c(1:m), type="l", xlab = "", ylim = c(min,max), ylab = "")
points(ynew,x=c(1:m),type="p", col = "red", pch=20)
legend("topleft", legend=c("True", "Observation"),col=c("black","red"),lty=1, cex=0.8)
```




```
#bench_mark_obs <- ynew
#bench_mark_f <- f
#bench_mark_K <- K
#bench_mark_sigma <- sigma
```

Real data

```
truth1 <- read.table("truth1.dat")
truth2 <- read.table("truth2.dat")
sigma <- 0.0005
y <- truth2[,1]
K <- getKG1D(n=length(y),delta = 1)
Covariance <- sigma^2*diag(length(y))
mu <- rep(0,length(y))
ynew <- K%*%y + mvrnorm(1,mu,Covariance)
plot(y=y,x=c(1:length(y)), type="l", xlab = "", ylim = c(min(ynew),max(ynew)), ylab = "")
points(ynew,x=c(1:length(ynew)),type="p", col = "red", pch=20)
legend("topleft", legend=c("True", "Observation"),col=c("black","red"),lty=1, cex=0.8)
```



Fitting Using Generalised Ridge's Regression

As comparison we wish to use these to compared our method vs generalised Ridge's regression.

```
n <- length(ynew)
p <- ncol(K)
L0 <- diag(p)
L1 <- matrix(c(-1,1,rep(0,p-1)),n-1,p, byrow <- T)
```

```
## Warning in matrix(c(-1, 1, rep(0, p - 1)), n - 1, p, byrow <- T): data
## length [206] is not a sub-multiple or multiple of the number of rows [204]
```

```
L2 <- matrix(c(-1,2,-1,rep(0,p-2)),n-2,p, byrow <- T)
```

```
## Warning in matrix(c(-1, 2, -1, rep(0, p - 2)), n - 2, p, byrow <- T): data
## length [206] is not a sub-multiple or multiple of the number of rows [203]
```

```
naiveRidgeL0 <- FitRidge(K,ynew,L=L0,sigma,-5,5)
```

```
## Warning in if (is.na(L)) {: the condition has length > 1 and only the first
## element will be used
```

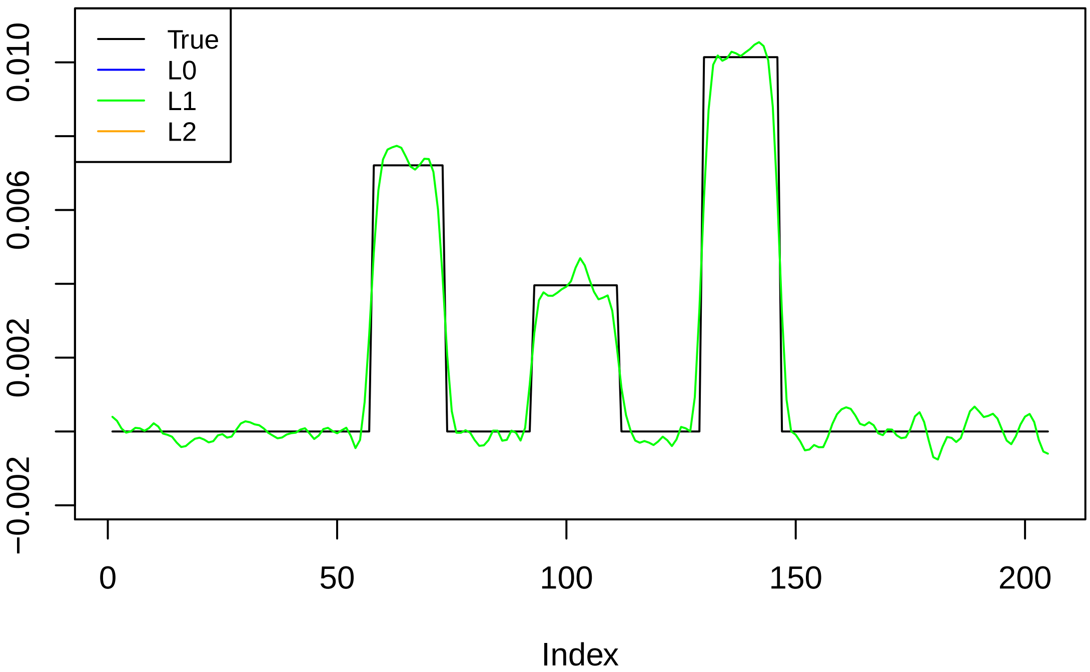
```
naiveRidgeL1 <- FitRidge(K,ynew,L=L1,sigma,-5,5)
```

```
## Warning in if (is.na(L)) {: the condition has length > 1 and only the first
## element will be used
```

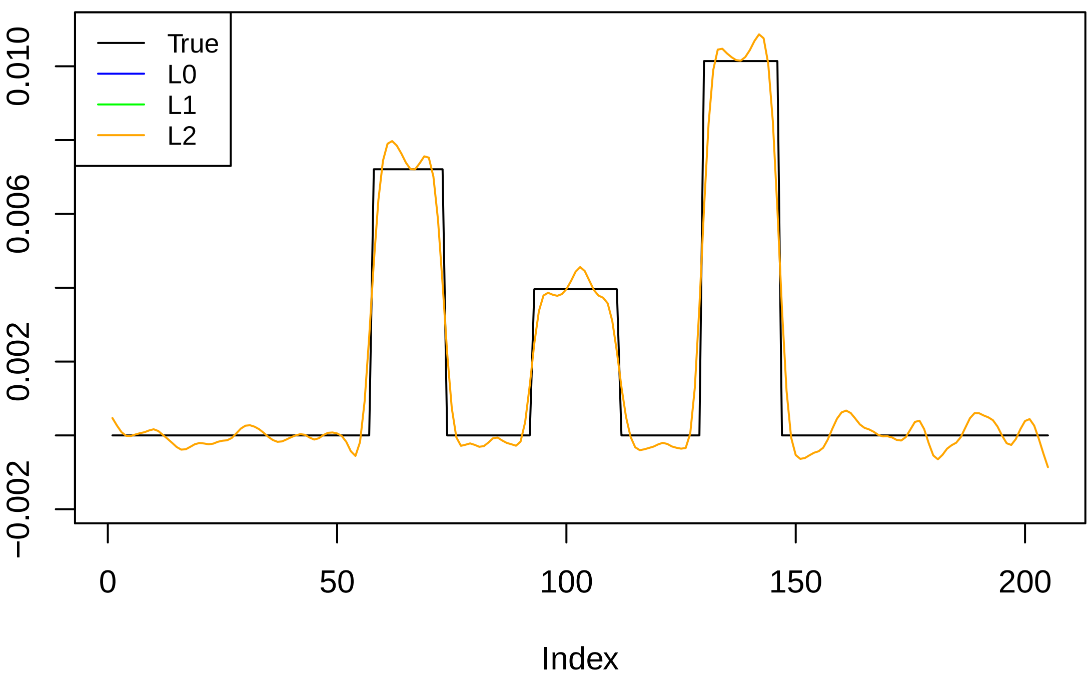
```
naiveRidgeL2 <- FitRidge(K,ynew,L=L2,sigma,-5,5)
```

```
## Warning in if (is.na(L)) {: the condition has length > 1 and only the first
## element will be used
```

```
min <- min(c(y,naiveRidgeL0$beta,naiveRidgeL1$beta,naiveRidgeL2$beta))
max <- max(c(y,naiveRidgeL0$beta,naiveRidgeL1$beta,naiveRidgeL2$beta))
plot(y, type="l", col="black", ylab = "", ylim=c(min,max))
lines(naiveRidgeL1$beta, col="green")
legend("topleft", legend=c("True", "L0","L1","L2"),col=c("black","blue","green","orange"),lty=1, cex=0.8
```




```
plot(y, type="l", col="black", ylab = "", ylim=c(min,max))
lines(naiveRidgeL2$beta, col="orange")
legend("topleft", legend=c("True", "L0","L1","L2"),col=c("black","blue","green","orange"),lty=1, cex=0.8
```




```
print(paste("L0 MSE: ", sum((naiveRidgeL0$beta-y)^2)," L1 MSE: ", sum((naiveRidgeL1$beta-y)^2)," L2 MSE
```

```
## [1] "L0 MSE:  0.000137394473450242  L1 MSE:  0.000107878347468459  L2 MSE:  0.000124557902954073"
```

Animating Effect of τ

We can produce an animation of the effects of τ on the estimates.

NOTE: This only work in R since the package used cannot animate in pdf or html.

Locally Adaptive Bayesian Model Selection

BIC

Here we is the main code to produce estimates, fitted, τ and BIC values of each models for each window. The models here includes constant model, L0, L1 and L2 we could further add Lasso and Bridge regression but further amendment to the code will be required.

```
estimate.data <- data.frame()
fitted.data <- data.frame()
BIC.data <- data.frame()
BICTau.data <- data.frame()
MSE.data <- data.frame()


windowWidth <- 10
for (i in 1:length(ynew)) {
  #print((i/length(ynew))*100)
  centered <- i
  lower <- centered - windowWidth/2
  upper <- centered + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  data <- ynew[c(lower:upper)]
  X <- K[c(lower:upper),]

  n <- length(ynew)
  p <- ncol(X)
  L0 <- diag(p)
  L1 <- matrix(c(-1,1,rep(0,p-1)),n-1,p, byrow <- T)
  L2 <- matrix(c(-1,2,-1,rep(0,p-2)),n-2,p, byrow <- T)


  constant <- FitConstant(X,data,1000,sigma)
  naiveRidgeL0 <- FitRidge(X,data, L = L0, sigma = sigma,-5,2)
  #naiveRidgeL0 <- FitConstant(X,data,1000,sigma)
  naiveRidgeL1 <- FitRidge(X,data, L = L1, sigma = sigma,-5,2)
  #naiveRidgeL1 <- FitConstant(X,data,1000,sigma)
  naiveRidgeL2 <- FitRidge(X,data, L = L2, sigma = sigma,-5,2)
  #naiveRidgeL2 <- FitConstant(X,data,1000,sigma)
  #BICVec <- c(constant$max)
  BICVec <- c(constant$max,naiveRidgeL0$maxBIC, naiveRidgeL1$maxBIC, naiveRidgeL2$maxBIC)
  prob <- ToProb(BICVec)

  newrow <- data.frame(
    "window" = i,
    "X" = c(1:length(ynew)),
```

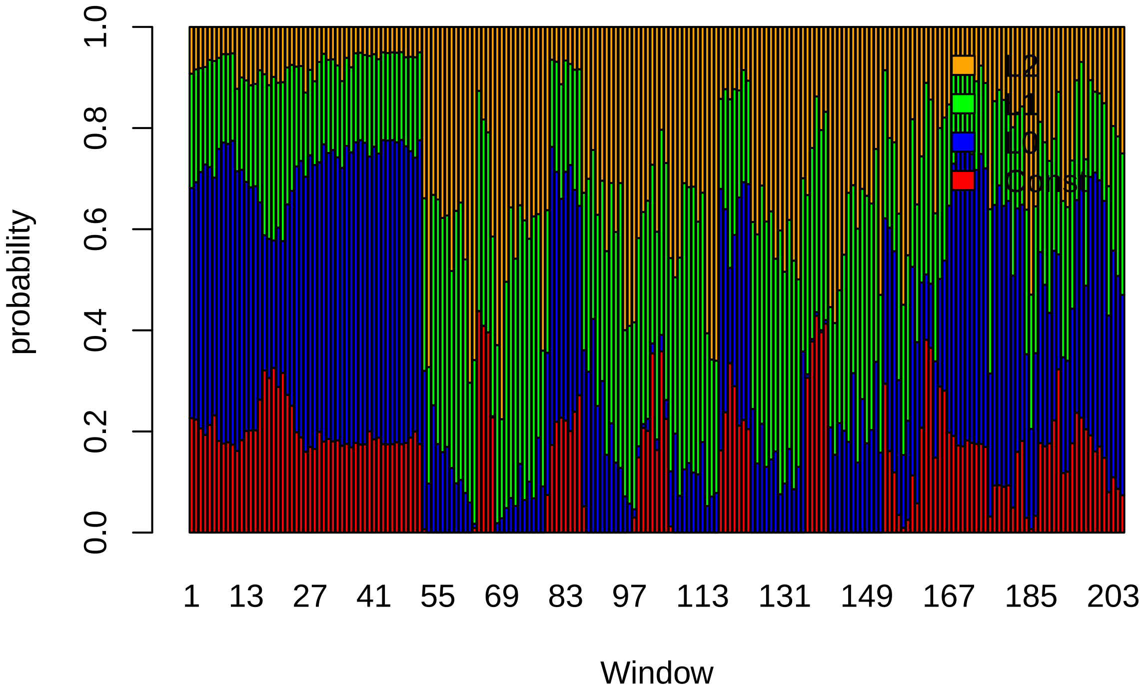
```r
    #"X" = c(lower:upper),
    "constantBeta" = constant$beta,
    "L0Beta" = naiveRidgeL0$beta,
    "L1Beta" = naiveRidgeL1$beta,
    "L2Beta" = naiveRidgeL2$beta
  )

  estimate.data <- rbind(estimate.data,newrow)

  newrow <- data.frame(
    "window" = i,
    "X" = c(lower:upper),
    "constantFit" = constant$fit,
    "L0Fit" = naiveRidgeL0$fit,
    "L1Fit" = naiveRidgeL1$fit,
    "L2Fit" = naiveRidgeL2$fit
  )

  fitted.data <- rbind(fitted.data, newrow)

  newrow <- data.frame(
    "window" = i,
    "constantBIC" = constant$BIC,
    "L0BIC" = naiveRidgeL0$BIC,
    "L1BIC" = naiveRidgeL1$BIC,
    "L2BIC" = naiveRidgeL2$BIC
  )

  BIC.data <- rbind(BIC.data, newrow)
  #print(paste("window: ", i, ",constantTau: ", naiveRidgeL0$maxTau, ",L0Tau: ", naiveRidgeL0$maxTau))
  newrow <- data.frame(
    "window" = i,
    "constantTau" = constant$maxTau,
    "L0Tau" = naiveRidgeL0$maxTau,
    "L1Tau" = naiveRidgeL1$maxTau,
    "L2Tau" = naiveRidgeL2$maxTau,
    "constantBIC" = constant$maxBIC,
    "L0BIC" = naiveRidgeL0$maxBIC,
    "L1BIC" = naiveRidgeL1$maxBIC,
    "L2BIC" = naiveRidgeL2$maxBIC
  )

  BICTau.data <- rbind(BICTau.data, newrow)

  newrow <- data.frame(
    "window"=i,
    "constantMSE"= sum((constant$beta[lower:upper]-data)^2),
    "L0MSE"= sum((naiveRidgeL0$beta[lower:upper]-data)^2),
    "L1MSE"= sum((naiveRidgeL1$beta[lower:upper]-data)^2),
    "L2MSE"= sum((naiveRidgeL2$beta[lower:upper]-data)^2)
      )
  MSE.data <- rbind(MSE.data, newrow)
}
```

Animated Estimates

Here we produce an animation plot to show the intermediate stages of fitting each models to each window. NOTE: this only work in R.

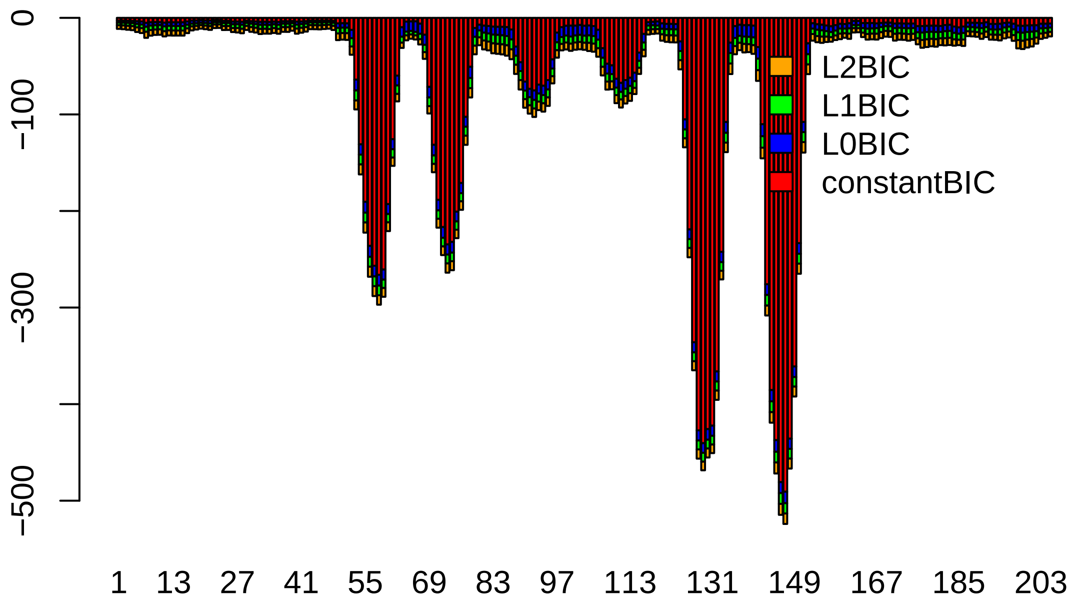
Stacked Percentage Plot of Posterior with $\alpha \to \infty$

```
prob.data <- data.frame()
temp <- BICTau.data[,c("constantBIC","L0BIC","L1BIC","L2BIC")]
for (row in 1:nrow(temp)){
  #print(row)
  BIC <- temp[row,]
  prob <- rep(NA, length(BIC))
  prob[1] <- exp(BIC[1])/sum(exp(BIC))
  prob[2] <- exp(BIC[2])/sum(exp(BIC))
  prob[3] <- exp(BIC[3])/sum(exp(BIC))
  prob[4] <- exp(BIC[4])/sum(exp(BIC))
  #print(prob)
  prob.data <- rbind(prob.data,prob)
}
colnames(prob.data) <- c("Const","L0","L1","L2")
rownames(prob.data) <- c(1:nrow(prob.data))
col <- rep(c("red","blue","green","orange"),40)
barplot(t(prob.data),
        xlab = "Window",
        ylab = "probability",
        col = col,
        legend.text = TRUE,
        args.legend=list(
          x=ncol(t(prob.data)) + 35,
          y=max(colSums(t(prob.data))),
          bty = "n"
        ))
```



BIC Plot of Model

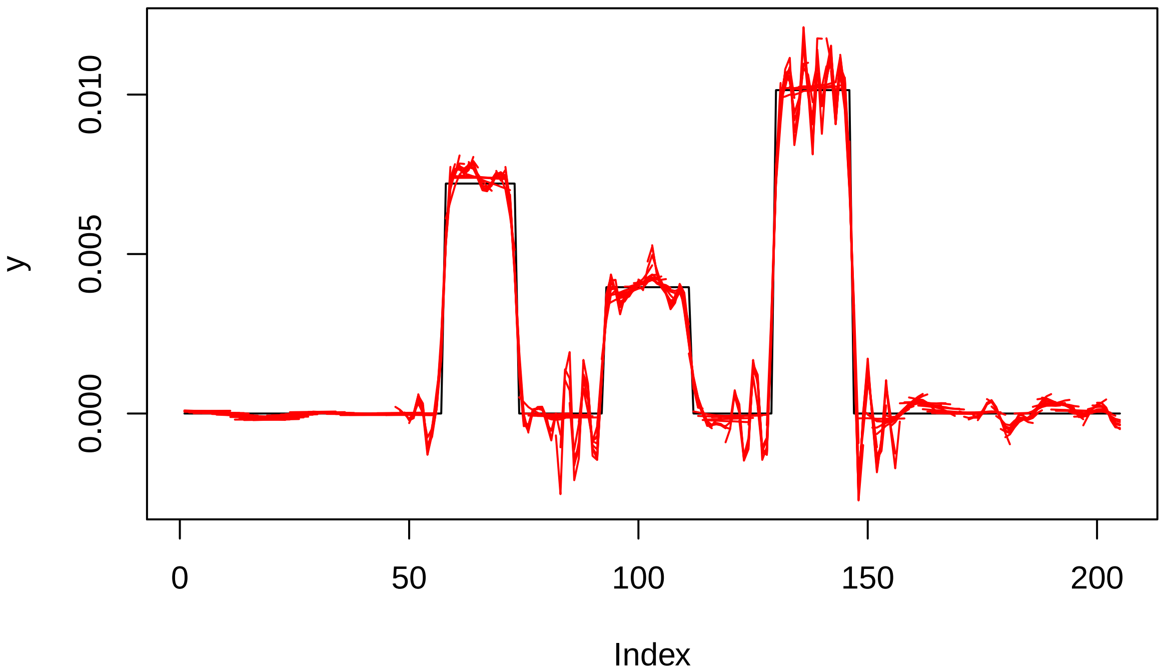
```
temp <- BICTau.data[,c("constantBIC","L0BIC","L1BIC","L2BIC")]
rownames(temp) <- c(1:nrow(prob.data))
col <- rep(c("red","blue","green","orange"),40)
barplot(t(temp),
        col=col,
        legend.text = TRUE,
        args.legend=list(
          x=ncol(t(temp)) + 30,
          y=max(colSums(t(temp))),
          bty = "n"
        ))
```



Within-Window Posterior Average

```
interWindowAverage.data <- data.frame()
for (w in 1:length(ynew)) {
  lower <- w - windowWidth/2
  upper <- w + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  estimate <- prob.data[w,]$Const*estimate.data[estimate.data$window==w,"constantBeta"] + prob.data[w,]$
  newrow <- data.frame(
    "window" = w,
    "X" = c(lower:upper),
    "estimate" = estimate[c(lower:upper)]
  )
  interWindowAverage.data <- rbind(interWindowAverage.data, newrow)
}
```

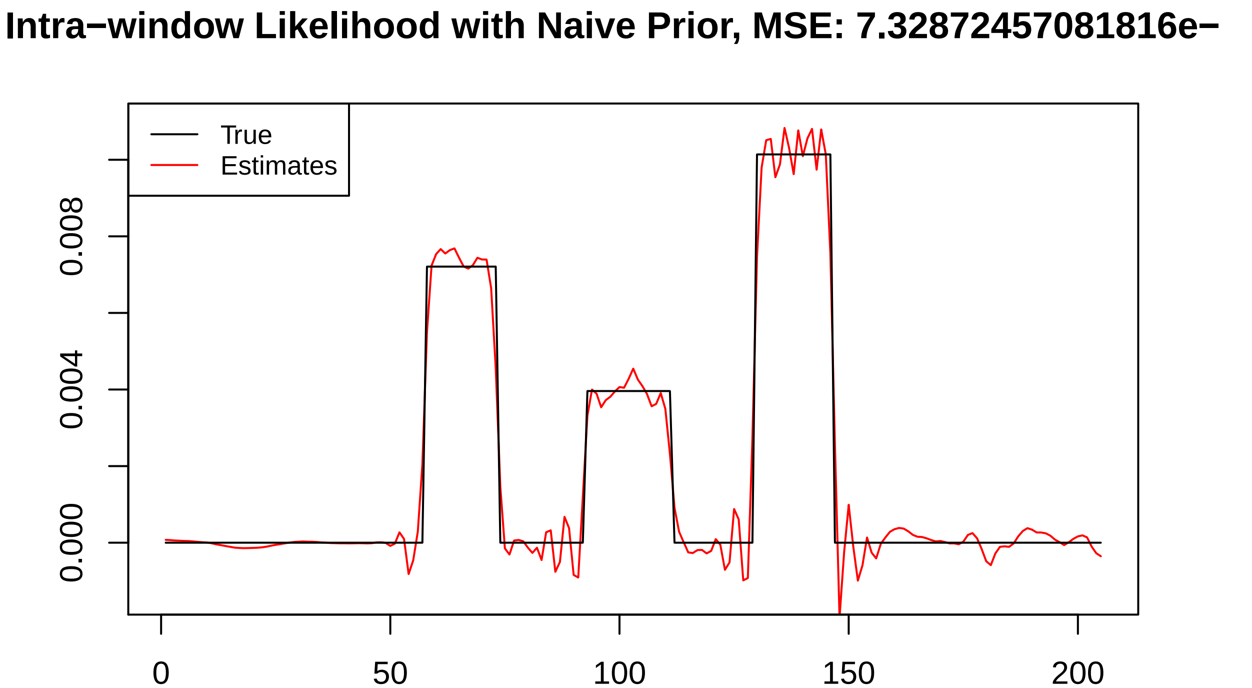
```
min <- min(interWindowAverage.data[,"estimate"])
max <- max(interWindowAverage.data[,"estimate"])
plot(y, type="l", col="black", ylim = c(min,max))
for (w in 1:length(y)) {
  lines(x=subset(interWindowAverage.data, window == w)[,"X"], y=subset(interWindowAverage.data, window =
}
```

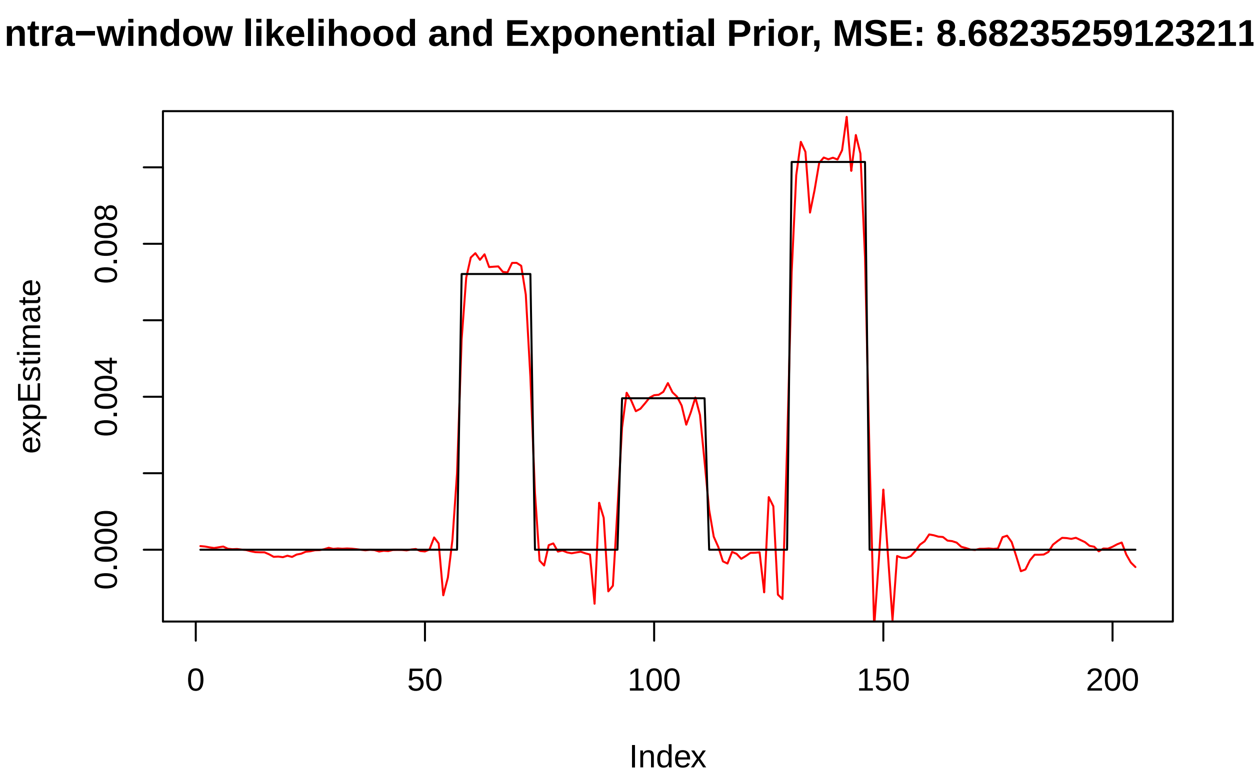


Comparing Two Different Value of α

Posterior with $\alpha \to \infty$

```
simpleEstimate <- rep(NA, length(ynew))
for(x in 1:length(ynew)){
  est <- subset(interWindowAverage.data,X==x)[,"estimate"]
  simpleEstimate[x] <- mean(est)
}
#par(mfrow=c(1,2))
min <- min(c(ynew, y))
max <- max(c(ynew, y))
plot(simpleEstimate, type="l", col="red", main = paste("Intra-window Likelihood with Naive Prior, MSE:"
lines(y, col="black")
#plot(K%*%simpleEstimate, col="red")
#points(ynew, col="black")
#points(ynew, col="blue")
#lines(y, col="blue")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```

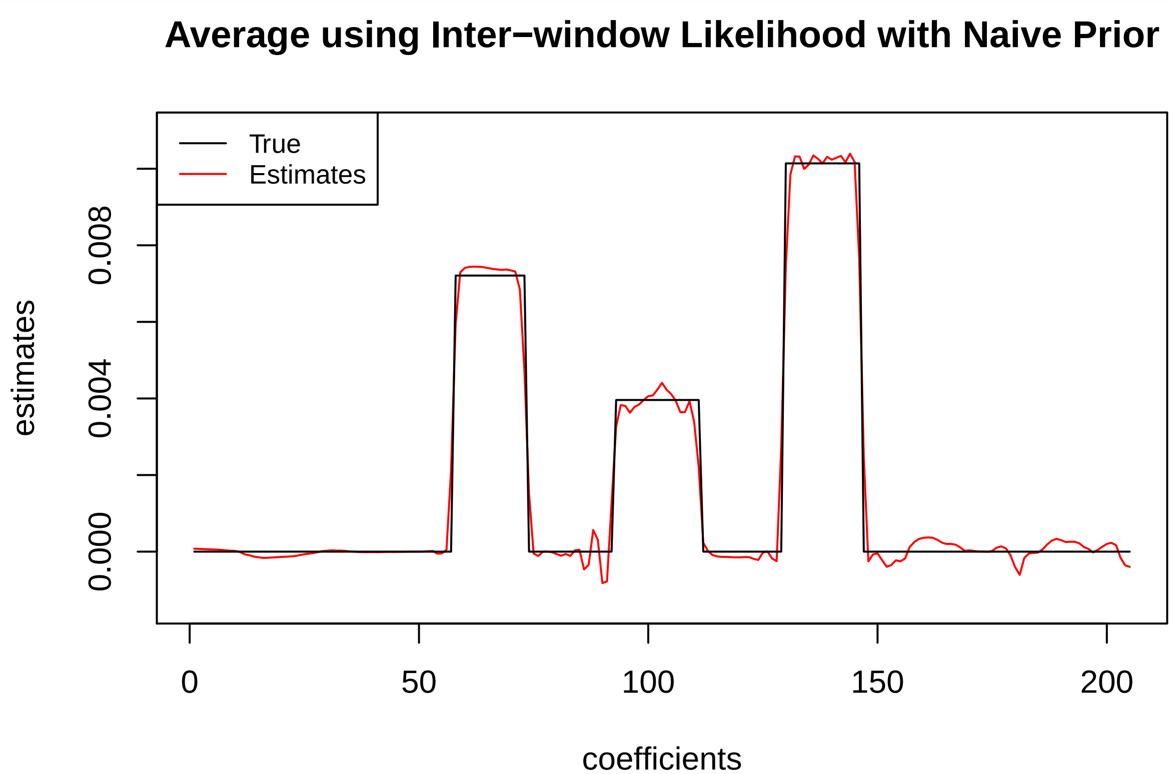




Posterior when $\alpha = 0$

```
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
```

```
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC","L0BIC","L1BIC","L2BIC")]
  prob <- cbind("window"=c(lower:upper),exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[prob$window==w,]$constantBIC*estimate$constantBeta + prob[p
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive P
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



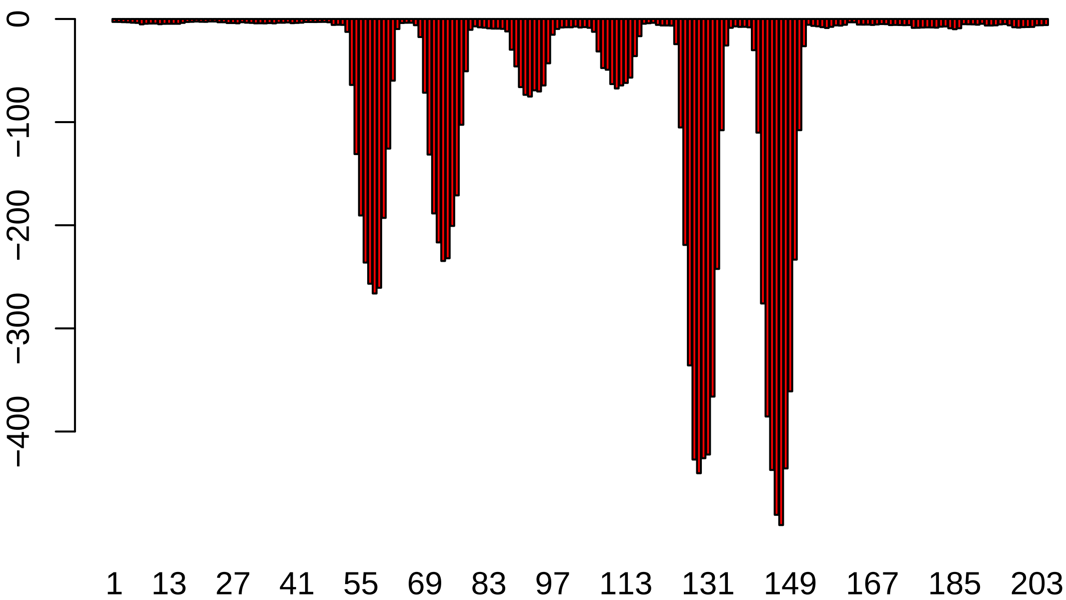
Average with Different Combination of Models

Single Model

Constant

BIC Plot

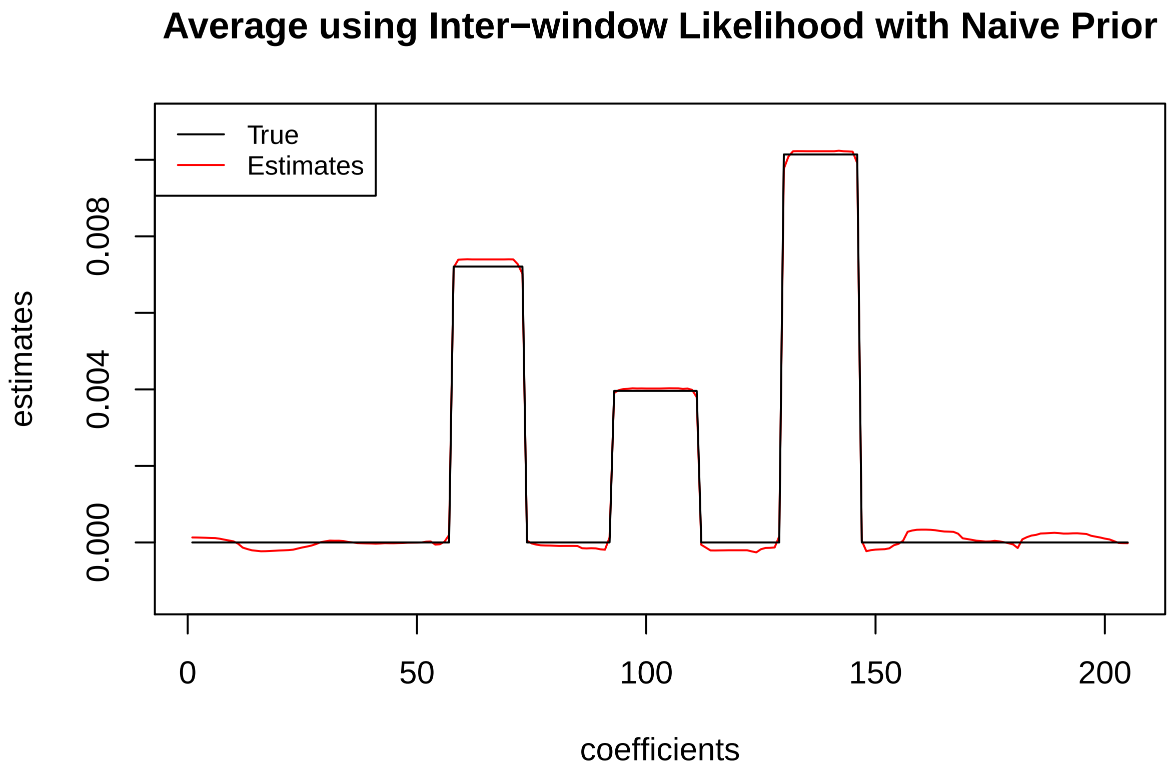
```r
temp <- BICTau.data[,c("constantBIC")]
col <- rep(c("red"),40)
barplot(t(temp),
        names.arg = c(1:length(temp)),
        col=col,
        legend.text = TRUE,
        args.legend=list(
          x=ncol(t(temp)) + 30,
          y=max(colSums(t(temp))),
          bty = "n"
        ))
```



Inter-window Likelihood with Naive Prior

```r
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC")]
  prob <- cbind("window"=c(lower:upper),"constantBIC"=exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$constantBeta #+ prob[prob$window==w,]
  }
}
min <- min(c(BICEstimate, ynew,y))
```

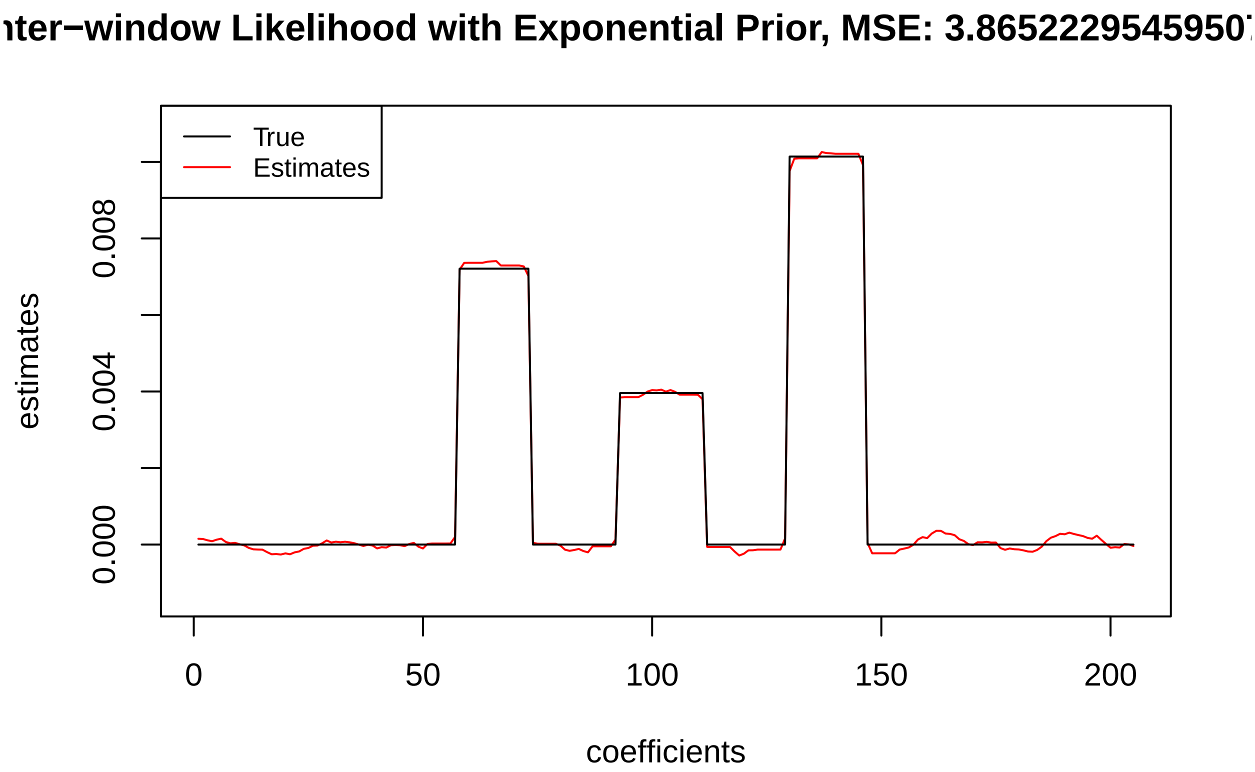
```
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive P
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



Inter-window Likelihood with Exponential Prior

```
BICEstimate <- rep(0,length(ynew))
alpha <- 4
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC")]
  prob <- cbind("window"=c(lower:upper),"constantBIC"=exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)
```

```
  for(w in lower:upper){
    prob[w-lower+1,2] <- prior[w - lower + 1]*prob[w-lower+1,2]
  }
  prob[,2] <- prob[,2]/sum(prob[,2])

  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$constantBeta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```

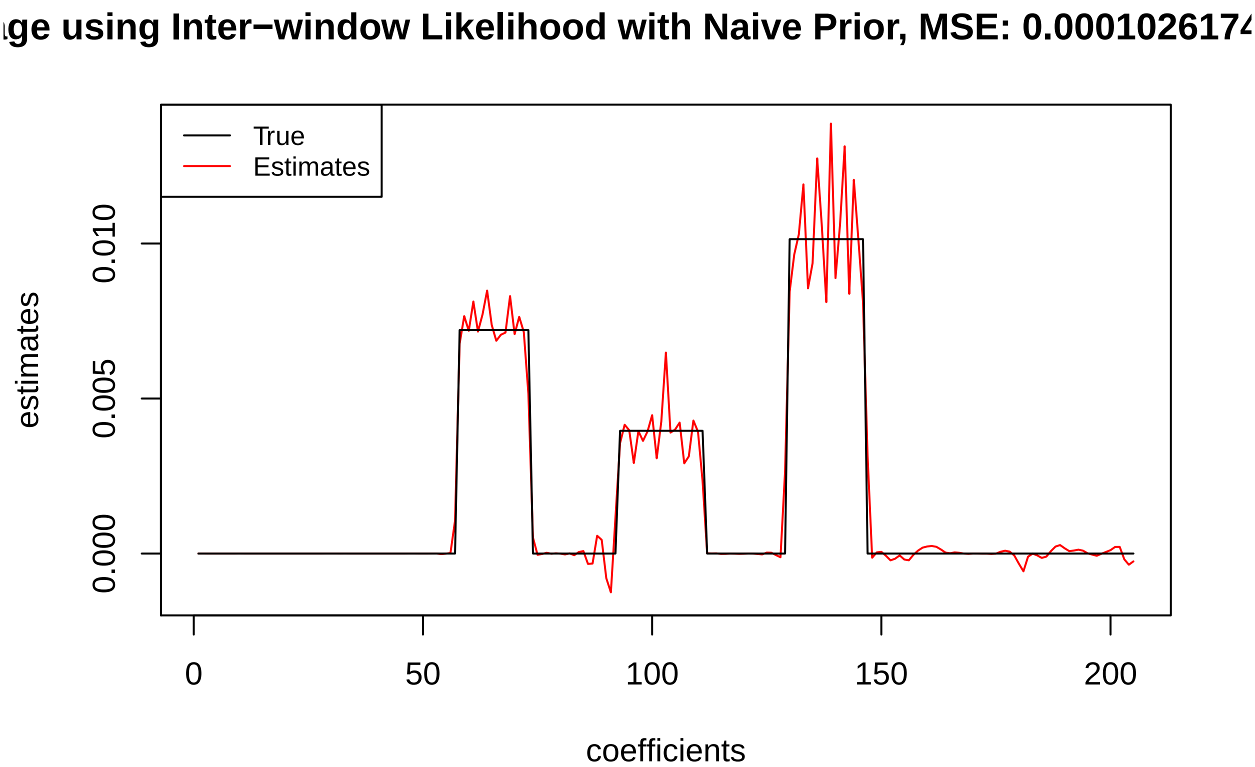


L0 Model

Inter-window Likelihood with Naive Prior

```
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
```

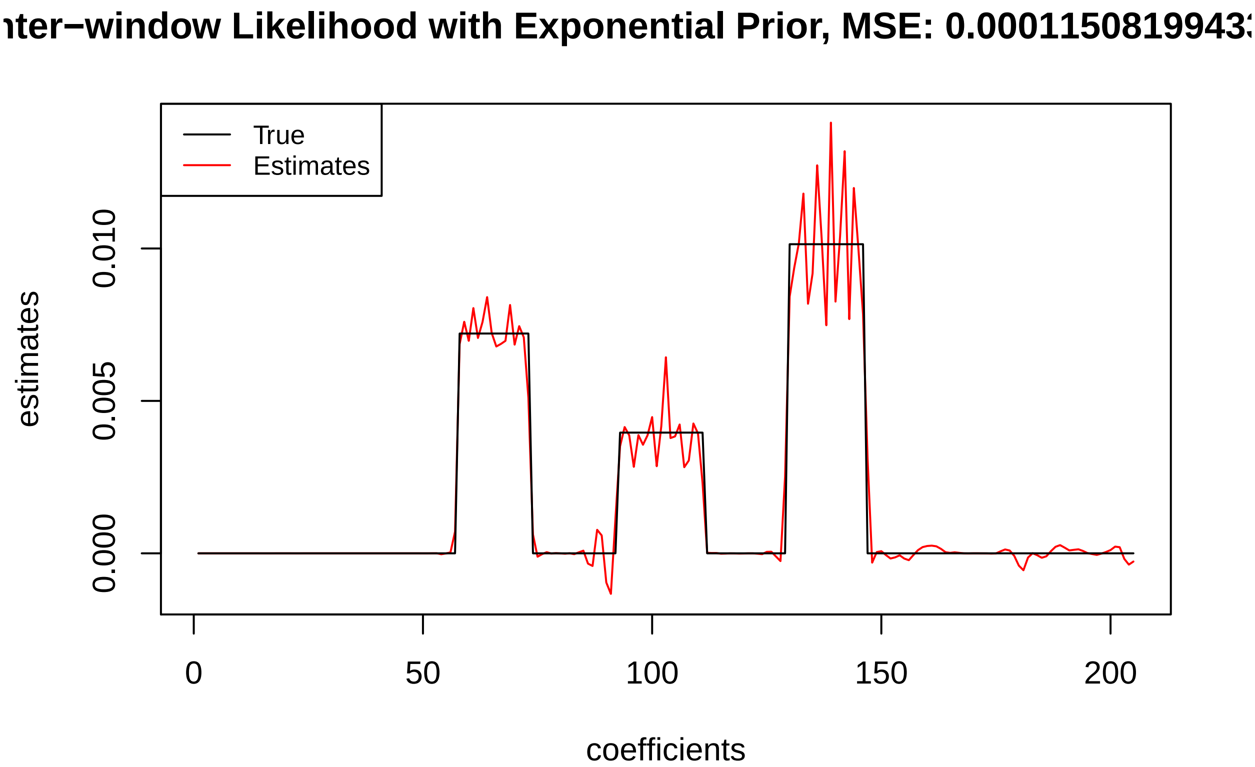
```
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("LOBIC")]
  prob <- cbind("window"=c(lower:upper),"LOBIC"=exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$LOBeta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive P
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



Inter-window Likelihood with Exponential Prior

```
BICEstimate <- rep(0,length(ynew))
alpha <- 0.3
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
```

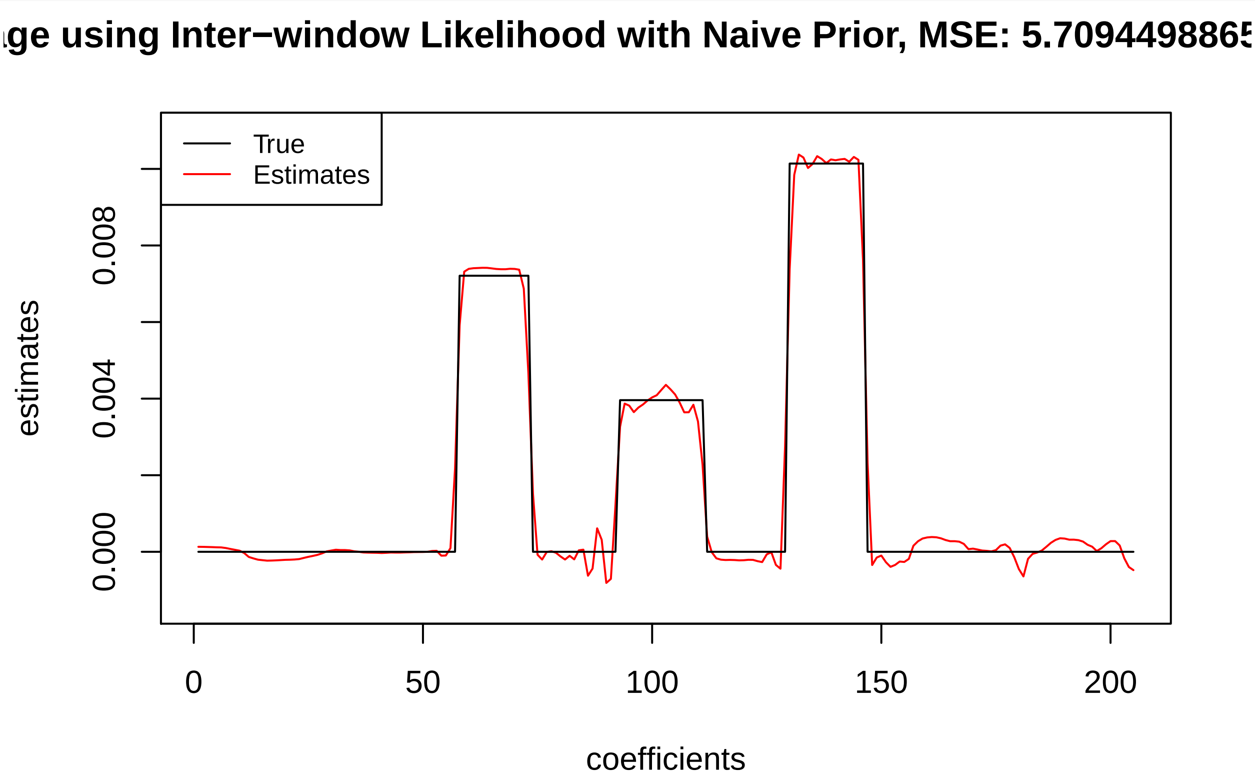
```
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("LOBIC")]
  prob <- cbind("window"=c(lower:upper),"LOBIC"=exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)

  for(w in lower:upper){
    prob[w-lower+1,2] <- prior[w - lower + 1]*prob[w-lower+1,2]
  }
  prob[,2] <- prob[,2]/sum(prob[,2])

  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$LOBeta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



L1 Model

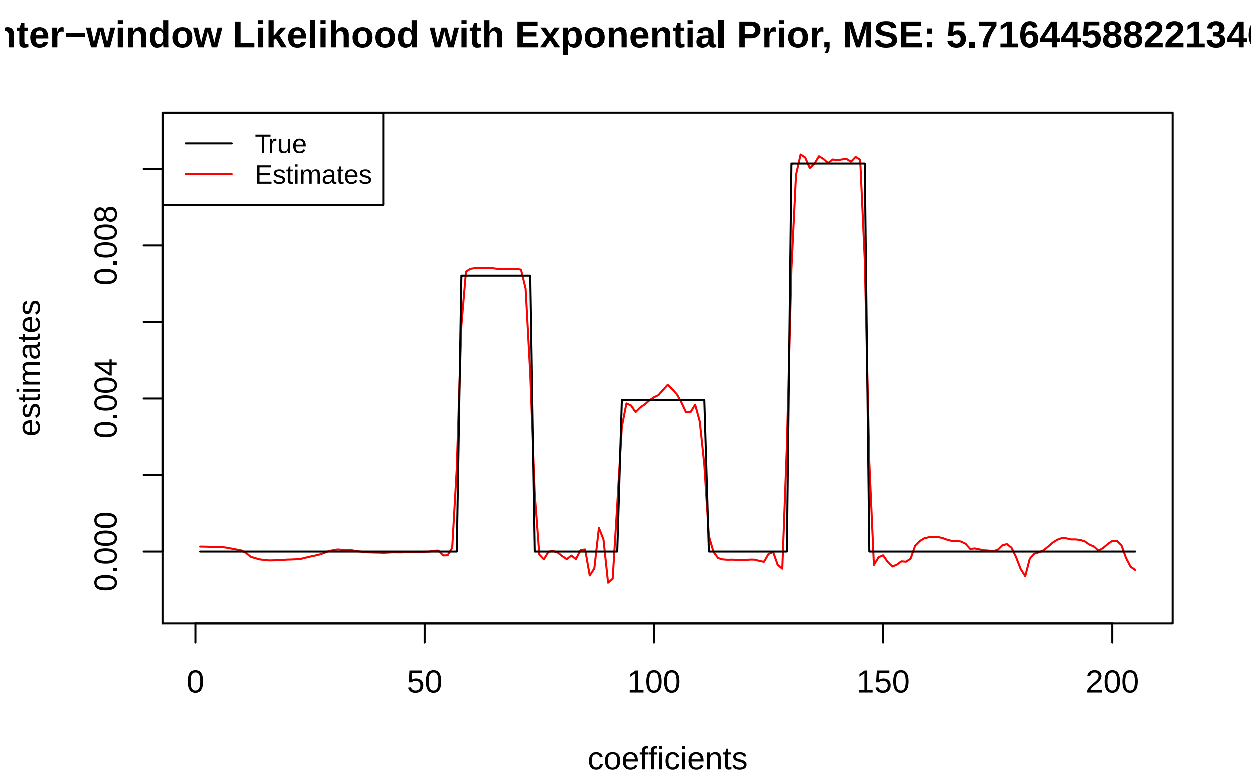
Inter-window Likelihood with Naive Prior

```r
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("L1BIC")]
  prob <- cbind("window"=c(lower:upper),"L1BIC"=exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$L1Beta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive P
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



Inter-window Likelihood with Exponential Prior

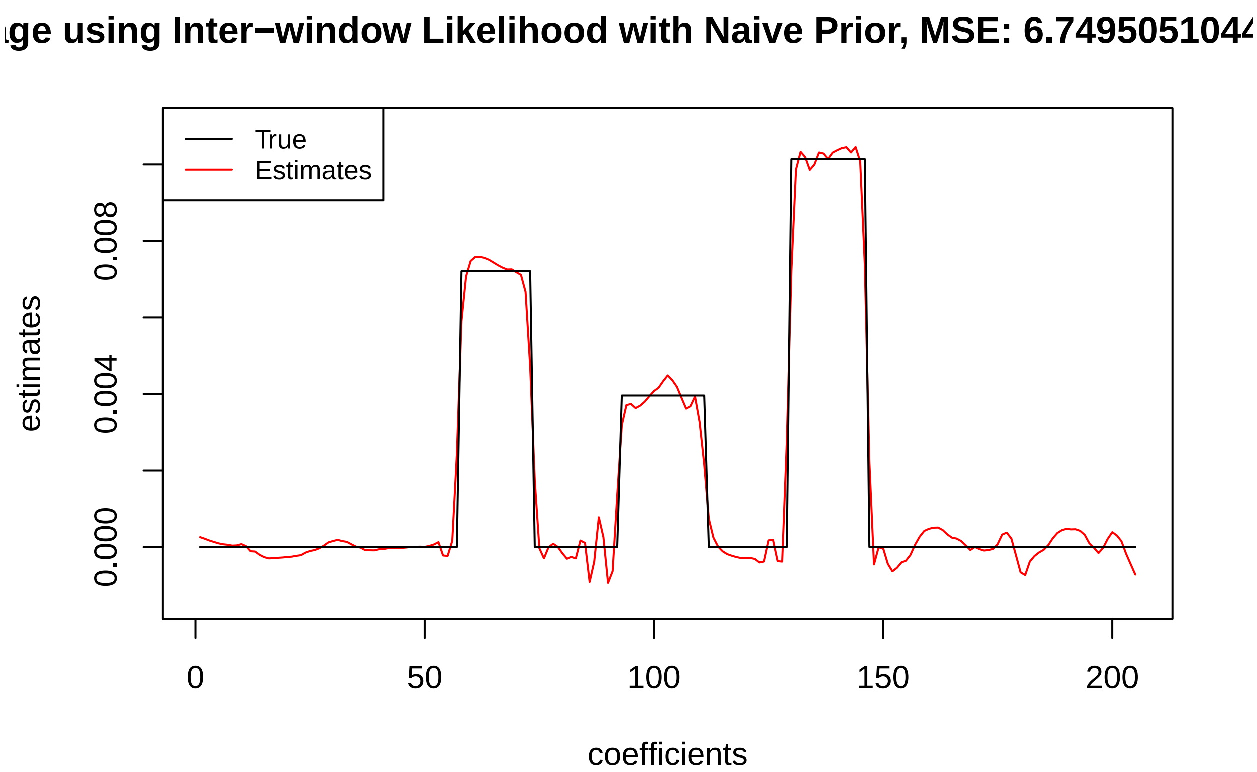
```
BICEstimate <- rep(0,length(ynew))
alpha <- 0.01
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("L1BIC")]
  prob <- cbind("window"=c(lower:upper),"L1BIC"=exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)

  for(w in lower:upper){
    prob[w-lower+1,2] <- prior[w - lower + 1]*prob[w-lower+1,2]
  }
  prob[,2] <- prob[,2]/sum(prob[,2])

  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$L1Beta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



L2 Model

Inter-window Likelihood with Naive Prior

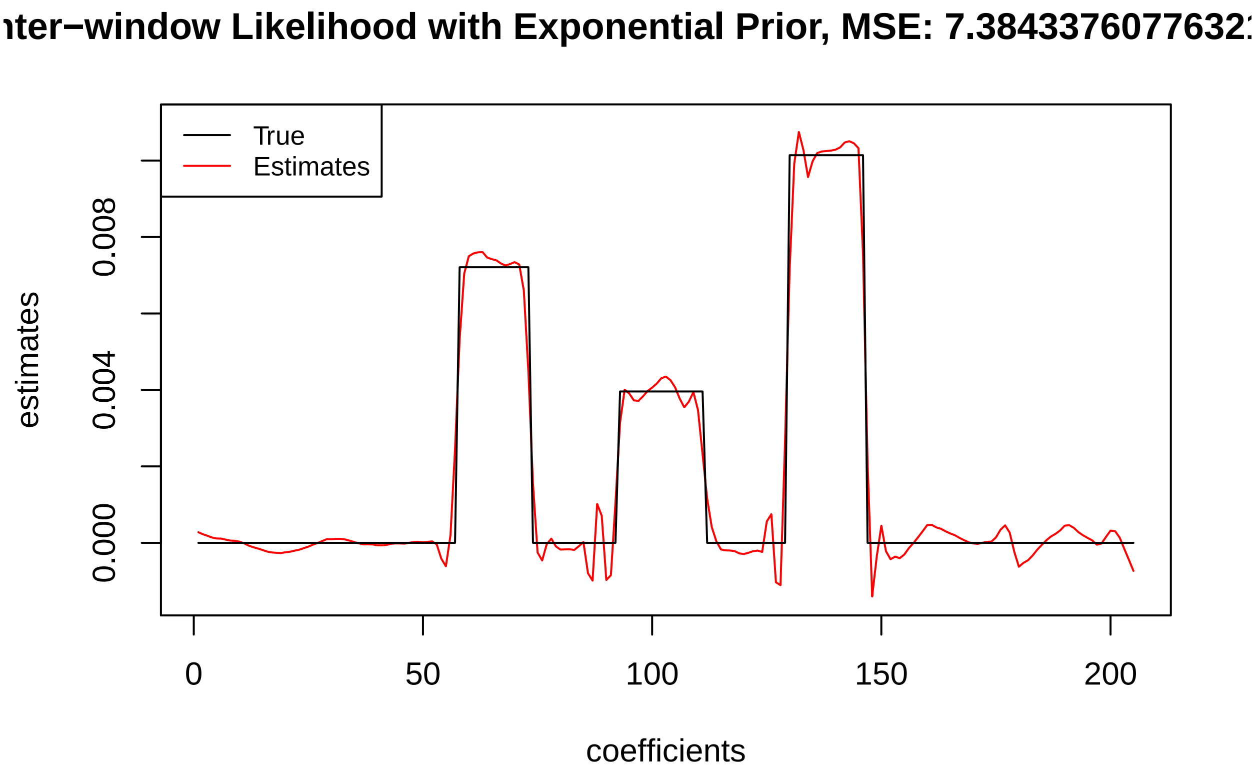
```
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("L2BIC")]
  prob <- cbind("window"=c(lower:upper),"L2BIC"=exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$L2Beta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive P
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



Inter-window Likelihood with Exponential Prior

```
BICEstimate <- rep(0,length(ynew))
alpha <- 1
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("L2BIC")]
  prob <- cbind("window"=c(lower:upper),"L2BIC"=exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)

  for(w in lower:upper){
    prob[w-lower+1,2] <- prior[w - lower + 1]*prob[w-lower+1,2]
  }
  prob[,2] <- prob[,2]/sum(prob[,2])
```

```
  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$L2Beta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



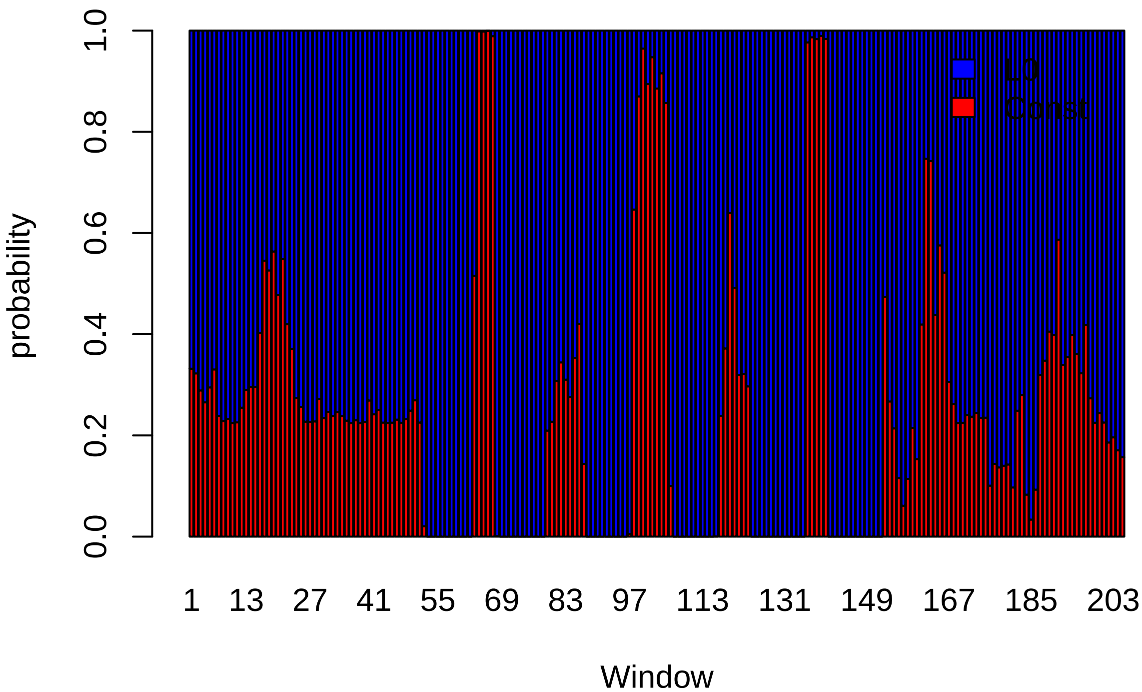
Two Models

Constant with L0

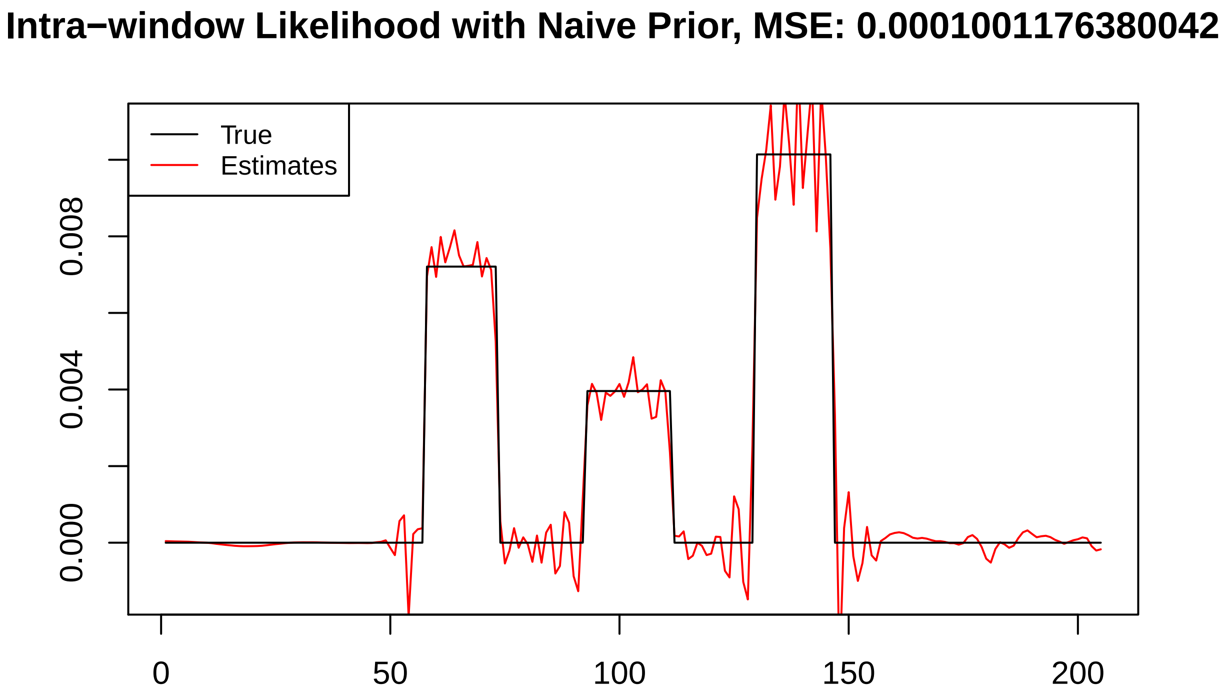
Intra-window likelihood

```
prob.data <- data.frame()
temp <- BICTau.data[,c("constantBIC","L0BIC")]
for (row in 1:nrow(temp)){
  #print(row)
  BIC <- temp[row,]
  prob <- rep(NA, length(BIC))
  prob[1] <- exp(BIC[1])/sum(exp(BIC))
  prob[2] <- exp(BIC[2])/sum(exp(BIC))
  #prob[3] <- exp(BIC[3])/sum(exp(BIC))
  #prob[4] <- exp(BIC[4])/sum(exp(BIC))
  #print(prob)
```

```r
  prob.data <- rbind(prob.data,prob)
}
colnames(prob.data) <- c("Const","L0")
rownames(prob.data) <- c(1:nrow(prob.data))
col <- rep(c("red","blue"),40)
barplot(t(prob.data),
        xlab = "Window",
        ylab = "probability",
        col = col,
        legend.text = TRUE,
        args.legend=list(
          x=ncol(t(prob.data)) + 35,
          y=max(colSums(t(prob.data))),
          bty = "n"
        ))
```

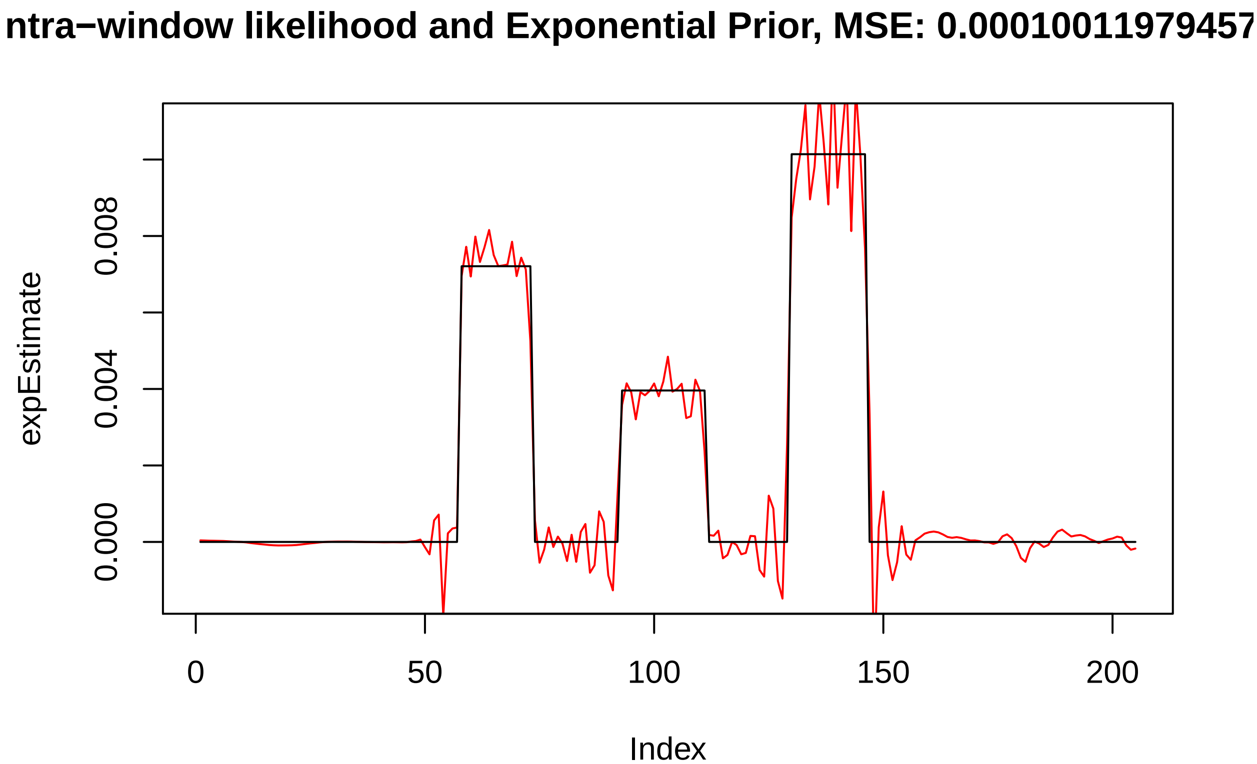


Naive Prior



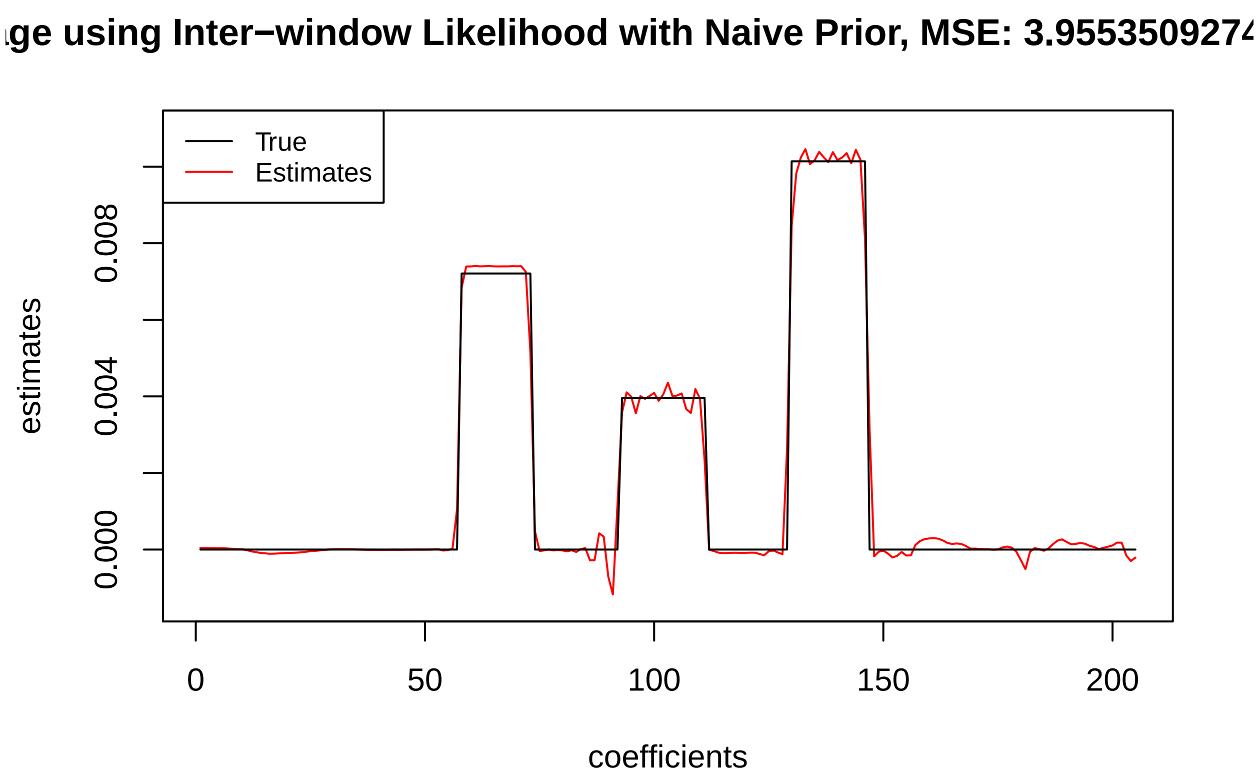
Exponential Prior

```
decayRate <- 0.0001
expEstimate <- rep(NA, length(ynew))
for(x in 1:length(ynew)){
  est <- subset(interWindowAverage.data,X==x)[,c("window", "estimate")]
  weight <- rep(NA,nrow(est))
  for (w in 1:nrow(est)) {
    weight[w] <- exp(-decayRate*abs(x-est[w,"window"]))
  }
  total <- sum(weight)
  for (w in 1:length(weight)) {
    weight[w] <- weight[w]/total
  }
  expEstimate[x] <- sum(weight*est[,"estimate"])
  #if(is.na(sum(weight*est[,"estimate"]))){
  #  print(paste("weight: ", weight))
  #  print(paste("estimate: ", est[,"estimate"]))
  #}
}
#par(mfrow=c(1,2))
min <- min(c(ynew, y))
max <- max(c(ynew, y))
plot(expEstimate, type="l", col="red", main = paste("Intra-window likelihood and Exponential Prior, MSE
lines(y, col="black")
```



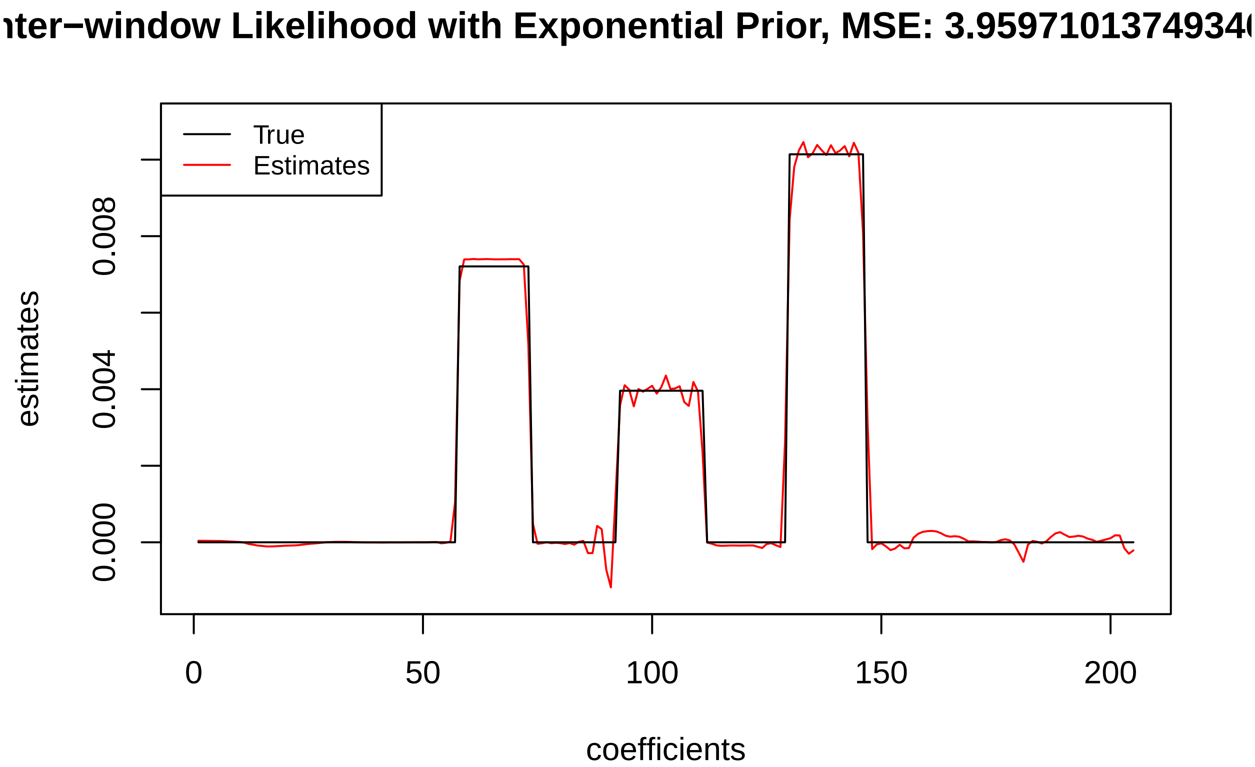
Inter-window Likelihood

Naive Prior



Exponential Prior

```
BICEstimate <- rep(0,length(ynew))
alpha <- 0.01
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC","LOBIC")]
  prob <- cbind("window"=c(lower:upper),exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)

  for(w in lower:upper){
    prob[prob$window==w,c("constantBIC","LOBIC")] <- prior[w - lower + 1]*prob[prob$window==w,c("consta
  }
  prob[,c("constantBIC","LOBIC")] <- prob[,c("constantBIC","LOBIC")]/sum(prob[,c("constantBIC","LOBIC")
  
  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[prob$window==w,"constantBIC"]*estimate$constantBeta + prob[
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```

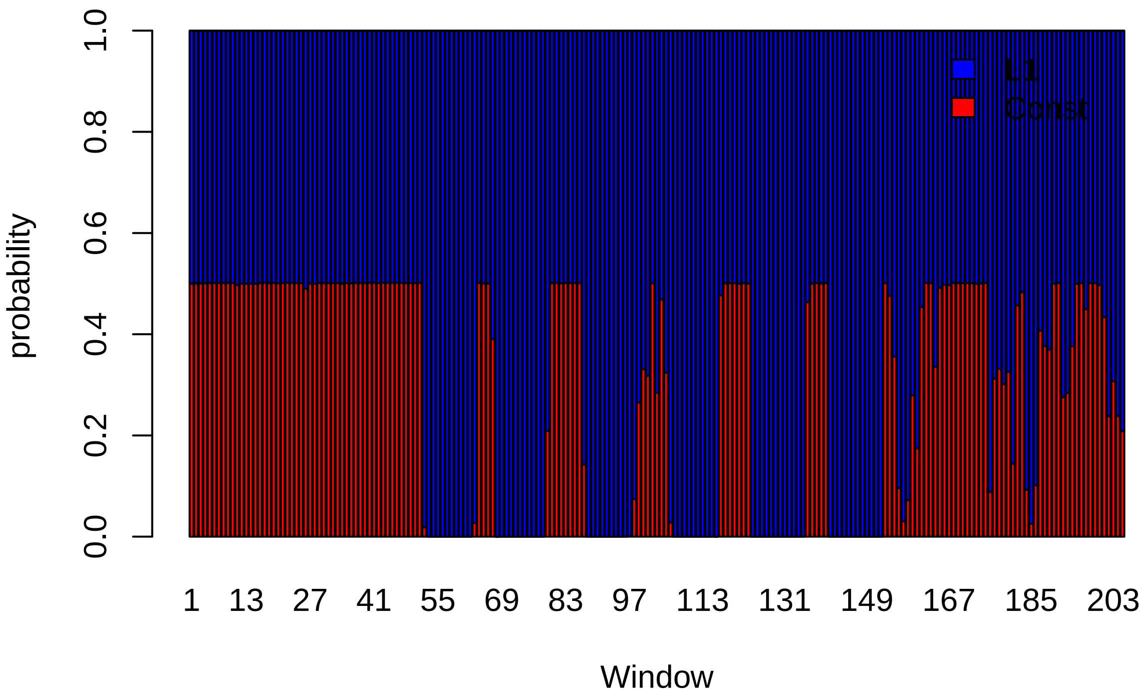


Constant with L1

Intra-window likelihood

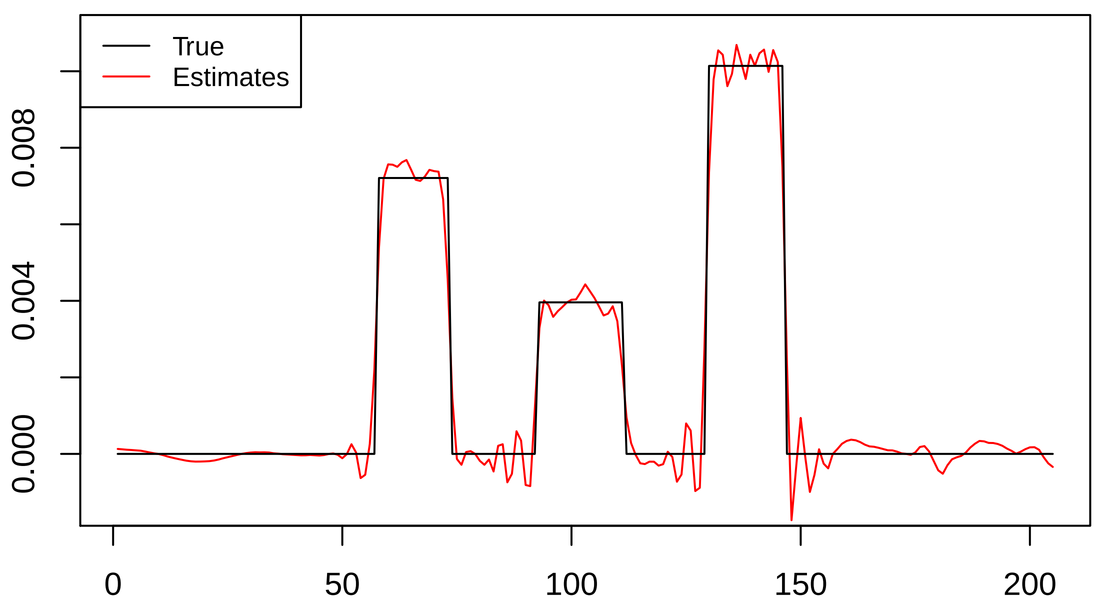
```
prob.data <- data.frame()
temp <- BICTau.data[,c("constantBIC","L1BIC")]
for (row in 1:nrow(temp)){
  #print(row)
  BIC <- temp[row,]
  prob <- rep(NA, length(BIC))
  prob[1] <- exp(BIC[1])/sum(exp(BIC))
  prob[2] <- exp(BIC[2])/sum(exp(BIC))
  #prob[3] <- exp(BIC[3])/sum(exp(BIC))
  #prob[4] <- exp(BIC[4])/sum(exp(BIC))
  #print(prob)
  prob.data <- rbind(prob.data,prob)
}
colnames(prob.data) <- c("Const","L1")
rownames(prob.data) <- c(1:nrow(prob.data))
col <- rep(c("red","blue"),40)
barplot(t(prob.data),
        xlab = "Window",
        ylab = "probability",
        col = col,
        legend.text = TRUE,
        args.legend=list(
          x=ncol(t(prob.data)) + 35,
          y=max(colSums(t(prob.data))),
```

```
        bty = "n"
      ))
```



Naive Prior

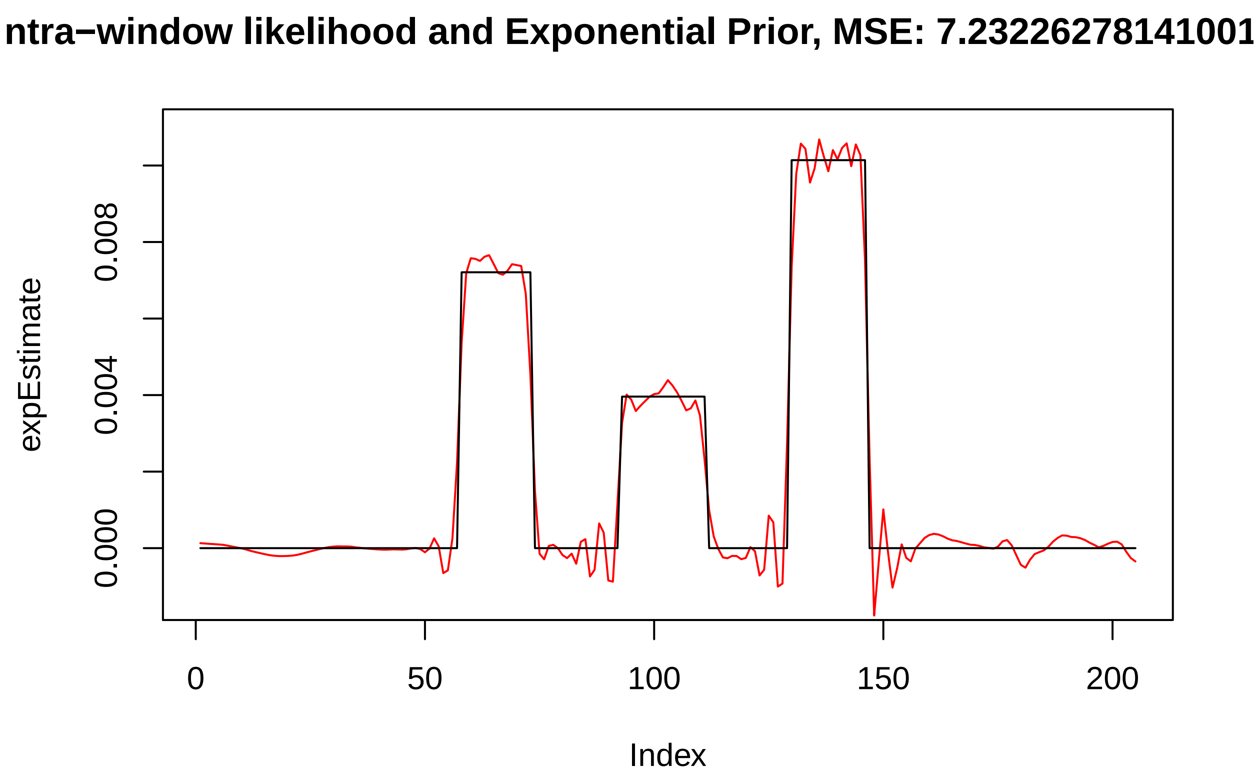
Intra−window Likelihood with Naive Prior, MSE: 7.12168869343278e−



Exponential Prior

```
decayRate <- 0.1
expEstimate <- rep(NA, length(ynew))
for(x in 1:length(ynew)){
  est <- subset(interWindowAverage.data,X==x)[,c("window", "estimate")]
```

```
  weight <- rep(NA,nrow(est))
  for (w in 1:nrow(est)) {
    weight[w] <- exp(-decayRate*abs(x-est[w,"window"]))
  }
  total <- sum(weight)
  for (w in 1:length(weight)) {
    weight[w] <- weight[w]/total
  }
  expEstimate[x] <- sum(weight*est[,"estimate"])
  #if(is.na(sum(weight*est[,"estimate"]))){
  #  print(paste("weight: ", weight))
  #  print(paste("estimate: ", est[,"estimate"]))
  #}
}
#par(mfrow=c(1,2))
min <- min(c(ynew, y))
max <- max(c(ynew, y))
plot(expEstimate, type="l", col="red", main = paste("Intra-window likelihood and Exponential Prior, MSE
lines(y, col="black")
```

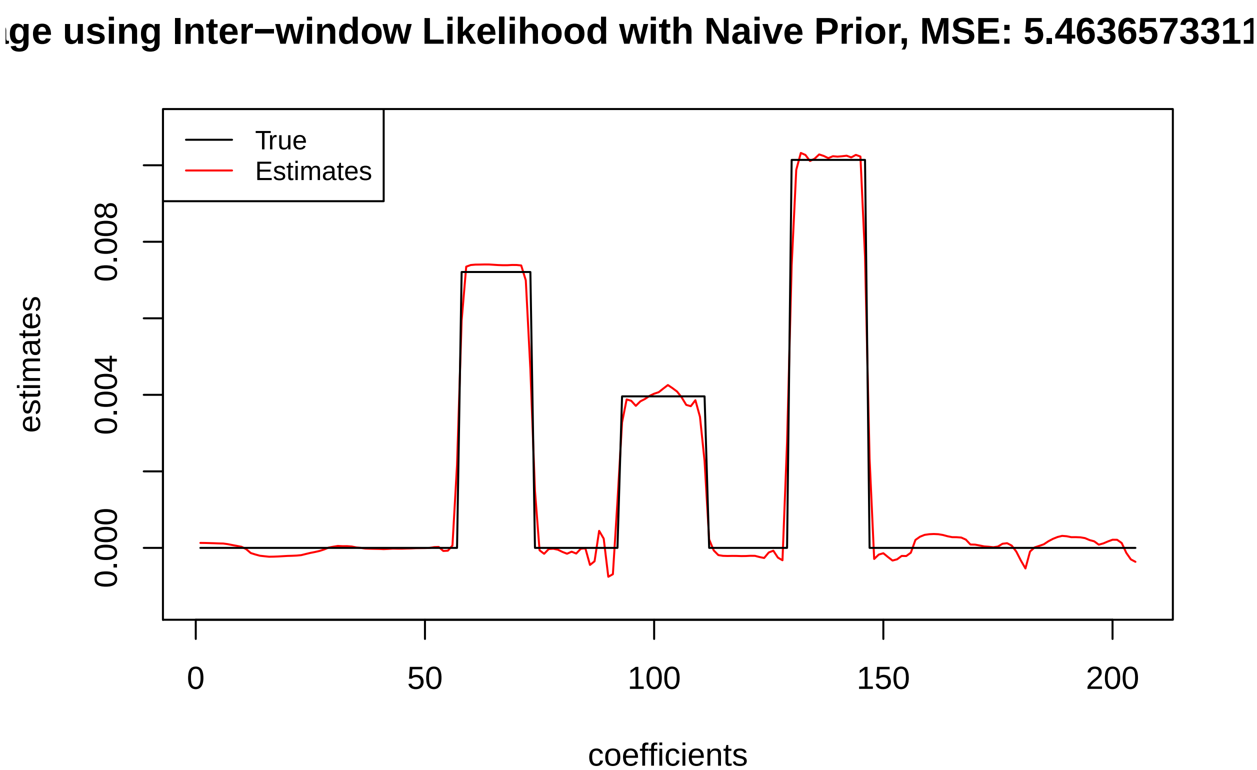


Inter-window Likelihood

Naive Prior

```
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
```

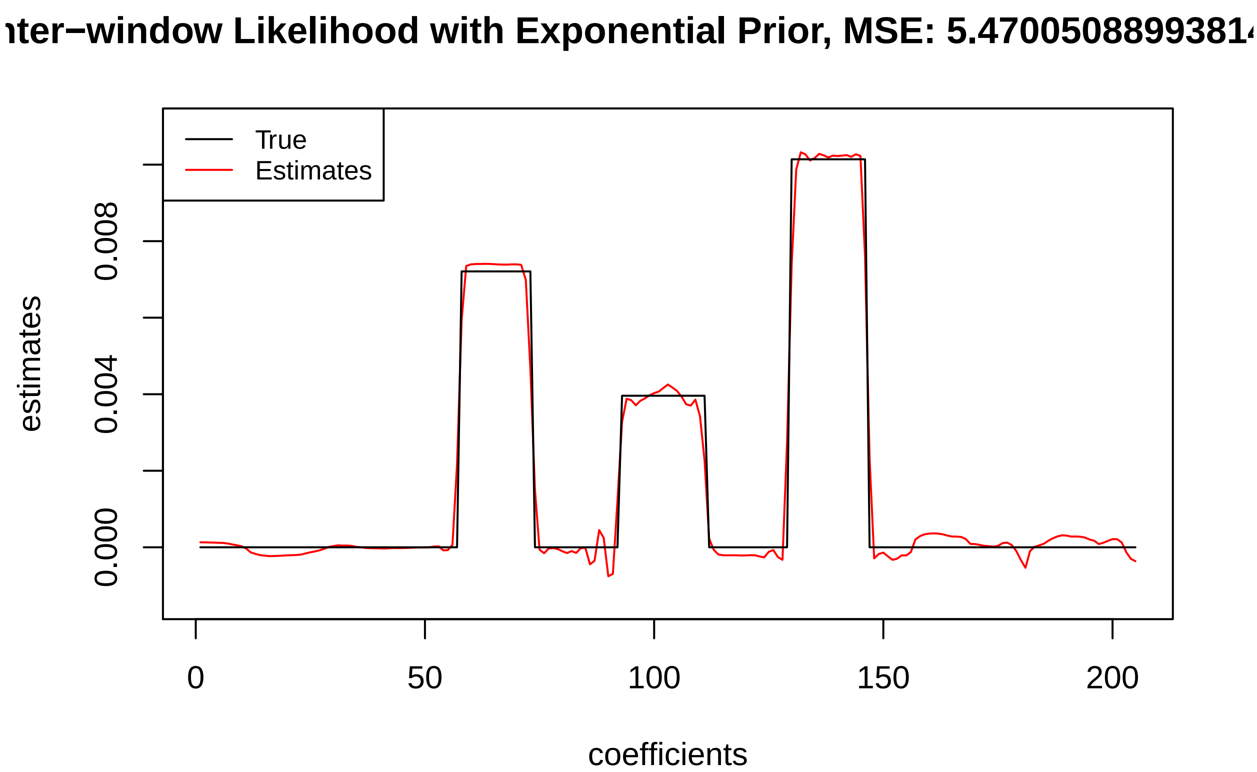
```
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC","L1BIC")]
  prob <- cbind("window"=c(lower:upper),exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[prob$window==w,"constantBIC"]*estimate$constantBeta +prob[pr
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive Pr
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



Exponential Prior

```
BICEstimate <- rep(0,length(ynew))
alpha <- 0.01
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
```

```
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC","L1BIC")]
  prob <- cbind("window"=c(lower:upper),exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)

  for(w in lower:upper){
    prob[prob$window==w,c("constantBIC","L1BIC")] <- prior[w - lower + 1]*prob[prob$window==w,c("consta
  }
  prob[,c("constantBIC","L1BIC")] <- prob[,c("constantBIC","L1BIC")]/sum(prob[,c("constantBIC","L1BIC")

  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[prob$window==w,"constantBIC"]*estimate$constantBeta + prob[p
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```

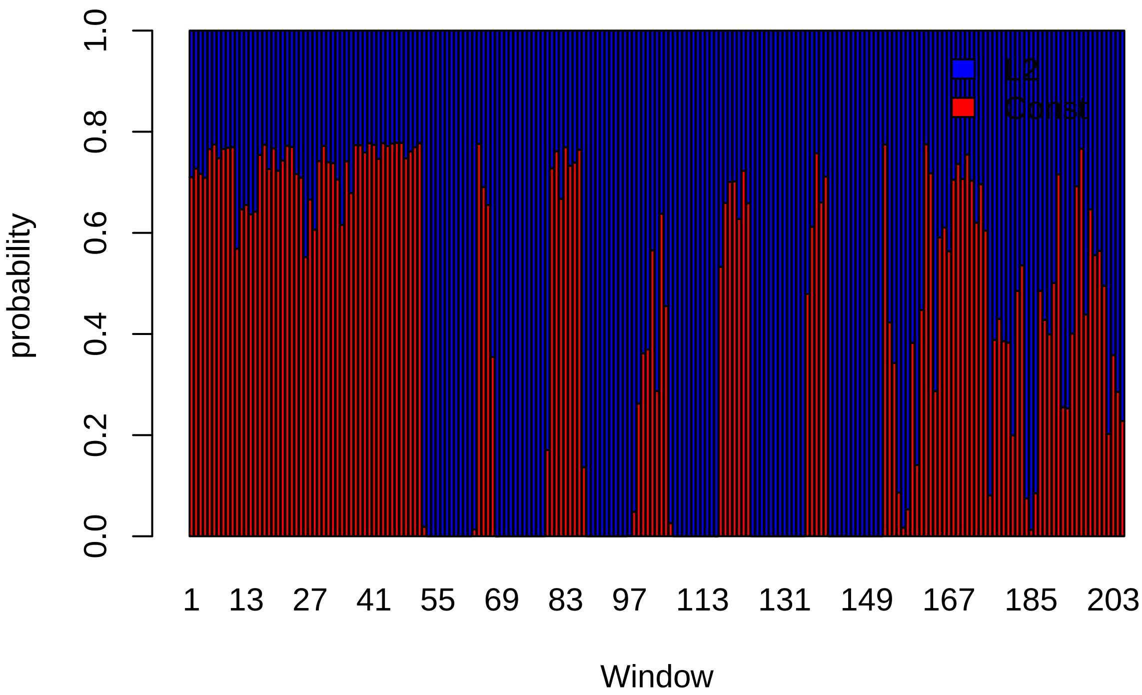


Constant with L2

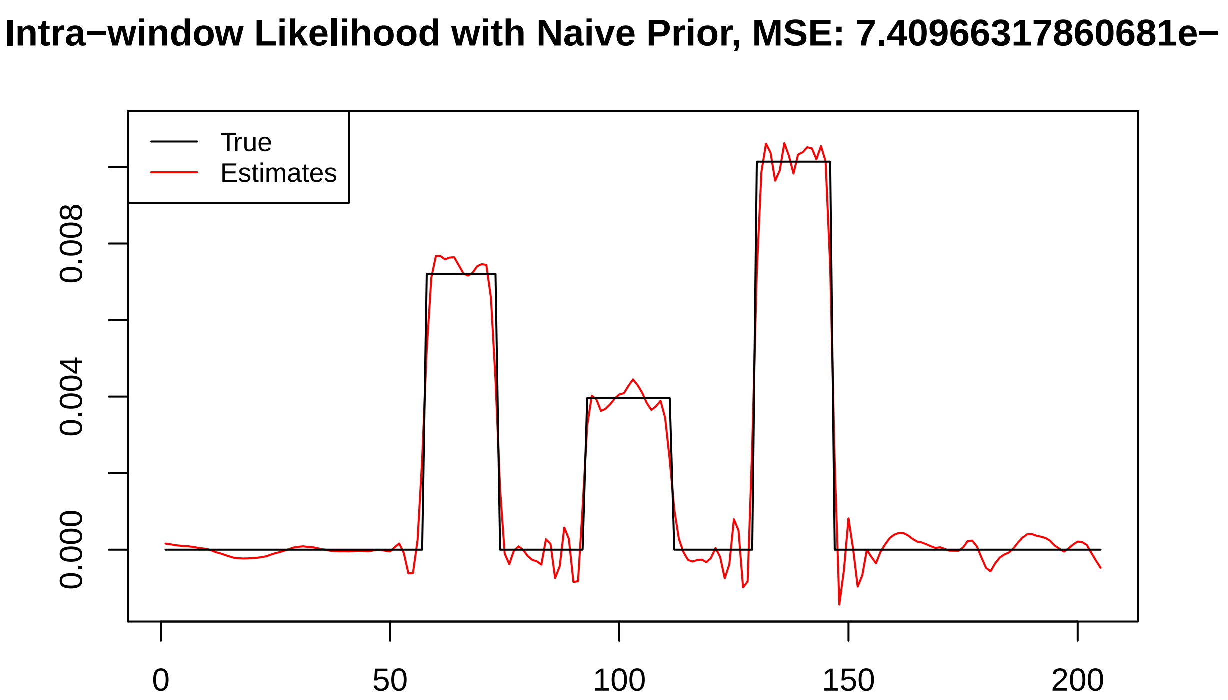
Intra-window likelihood

```
prob.data <- data.frame()
temp <- BICTau.data[,c("constantBIC","L2BIC")]
for (row in 1:nrow(temp)){
  #print(row)
  BIC <- temp[row,]
  prob <- rep(NA, length(BIC))
  prob[1] <- exp(BIC[1])/sum(exp(BIC))
  prob[2] <- exp(BIC[2])/sum(exp(BIC))
  #prob[3] <- exp(BIC[3])/sum(exp(BIC))
  #prob[4] <- exp(BIC[4])/sum(exp(BIC))
  #print(prob)
  prob.data <- rbind(prob.data,prob)
}
colnames(prob.data) <- c("Const","L2")
rownames(prob.data) <- c(1:nrow(prob.data))
col <- rep(c("red","blue"),40)
barplot(t(prob.data),
        xlab = "Window",
        ylab = "probability",
        col = col,
        legend.text = TRUE,
        args.legend=list(
          x=ncol(t(prob.data)) + 35,
          y=max(colSums(t(prob.data))),
```

```
        bty = "n"
      ))
```



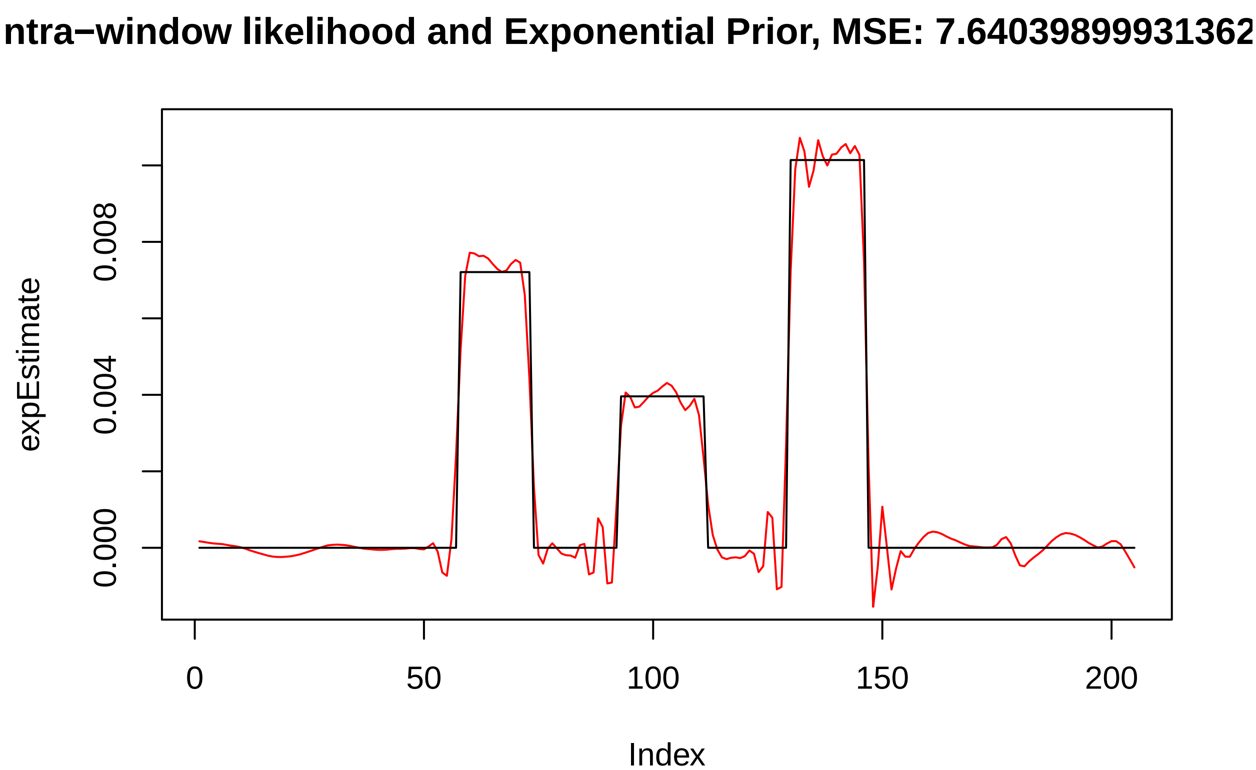
Naive Prior



Exponential Prior

```
decayRate <- 0.4
expEstimate <- rep(NA, length(ynew))
for(x in 1:length(ynew)){
  est <- subset(interWindowAverage.data,X==x)[,c("window", "estimate")]
```

```r
  weight <- rep(NA,nrow(est))
  for (w in 1:nrow(est)) {
    weight[w] <- exp(-decayRate*abs(x-est[w,"window"]))
  }
  total <- sum(weight)
  for (w in 1:length(weight)) {
    weight[w] <- weight[w]/total
  }
  expEstimate[x] <- sum(weight*est[,"estimate"])
  #if(is.na(sum(weight*est[,"estimate"]))){
  #  print(paste("weight: ", weight))
  #  print(paste("estimate: ", est[,"estimate"]))
  #}
}
#par(mfrow=c(1,2))
min <- min(c(ynew, y))
max <- max(c(ynew, y))
plot(expEstimate, type="l", col="red", main = paste("Intra-window likelihood and Exponential Prior, MSE
lines(y, col="black")
```

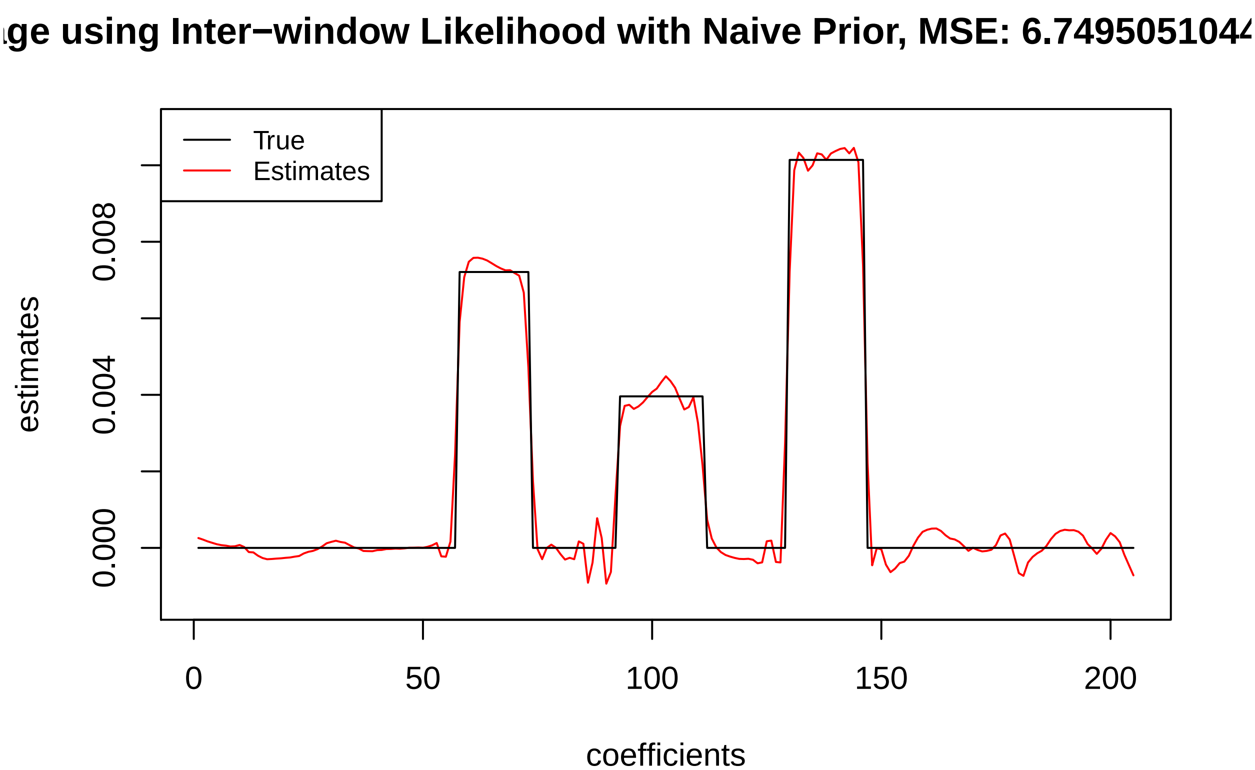


Inter-window Likelihood

Naive Prior

```r
BICEstimate <- rep(0,length(ynew))
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
```

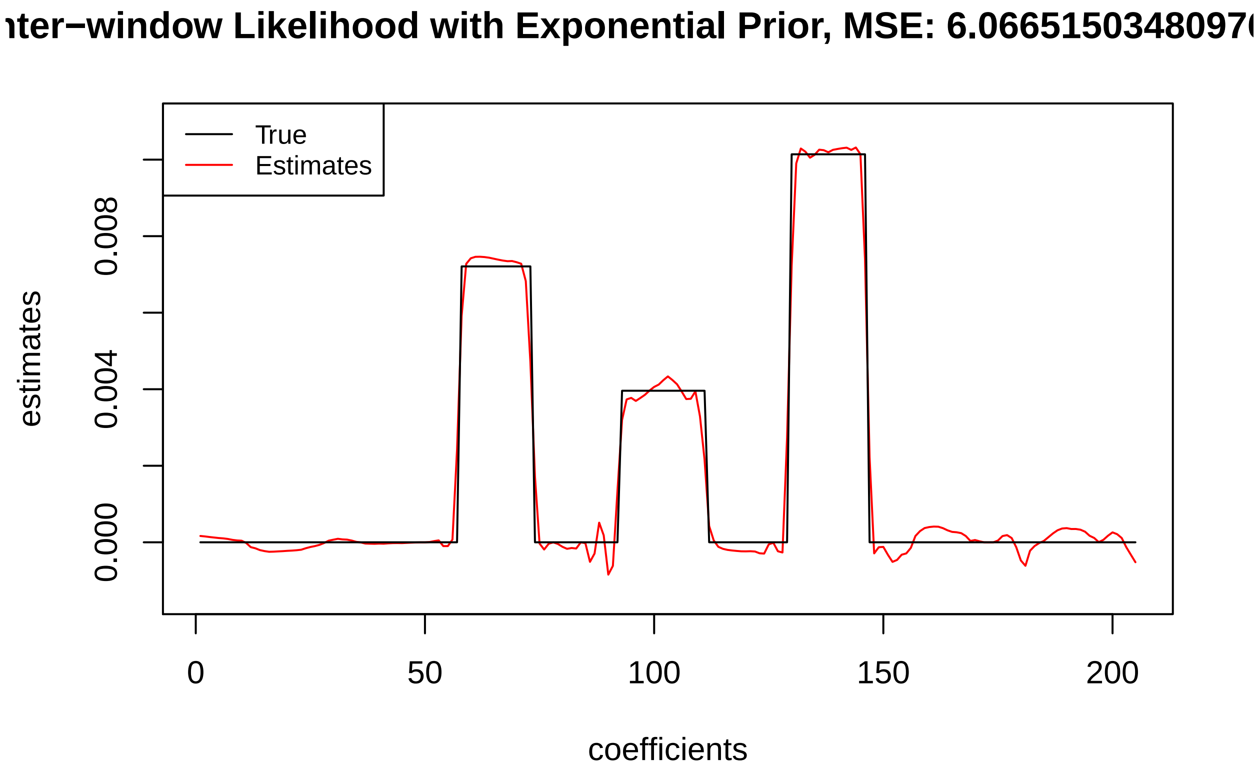
```
  upper <- x + windowWidth/2
  if(lower < 1){
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("L2BIC")]
  prob <- cbind("window"=c(lower:upper),"L2BIC"=exp(BIC)/sum(exp(BIC)))
  for (w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[w-lower+1,2]*estimate$L2Beta
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Average using Inter-window Likelihood with Naive Pr
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```



Exponential Prior

```
BICEstimate <- rep(0,length(ynew))
alpha <- 0.01
for (x in 1:length(ynew)) {
  #print(paste("x: ", x))
  prob <- data.frame()
  lower <- x - windowWidth/2
  upper <- x + windowWidth/2
  if(lower < 1){
```

```
    lower <- 1
  }
  if(upper > length(ynew)){
    upper <- length(ynew)
  }
  BIC <- BICTau.data[c(lower:upper),c("constantBIC","L2BIC")]
  prob <- cbind("window"=c(lower:upper),exp(BIC)/sum(exp(BIC)))

  # Exponential Weighting
  prior <- c(lower:upper)
  for(i in 1:length(prior)){
    prior[i] <- exp(-alpha*abs(prior[i]-x))
  }
  prior <- prior/sum(prior)

  for(w in lower:upper){
    prob[prob$window==w,c("constantBIC","L2BIC")] <- prior[w - lower + 1]*prob[prob$window==w,c("consta
  }
  prob[,c("constantBIC","L2BIC")] <- prob[,c("constantBIC","L2BIC")]/sum(prob[,c("constantBIC","L2BIC")]

  for(w in lower:upper) {
    estimate <- estimate.data[estimate.data$X==x & estimate.data$window==w,]
    BICEstimate[x] <- BICEstimate[x] + prob[prob$window==w,"constantBIC"]*estimate$constantBeta + prob[p
  }
}
min <- min(c(BICEstimate, ynew,y))
max <- max(c(BICEstimate, ynew,y))
plot(BICEstimate, type="l", col="red", main = paste("Inter-window Likelihood with Exponential Prior, MSE
lines(y, col="black")
legend("topleft", legend=c("True", "Estimates"),col=c("black","red"),lty=1, cex=0.8)
```


nter−window Likelihood with Exponential Prior, MSE: 6.0665150348097(
True
Estimates
estimates
0.000
0.004
0.008
0
50
100
150
200
coefficients