```r
## Load libraries
library(splines)
library(MASS)

## Define the number of tests
ntest <- 1000

## Set the value of lambda
lambda <- 0.8

## Set nuber of simulations
nSims <- 10000
```

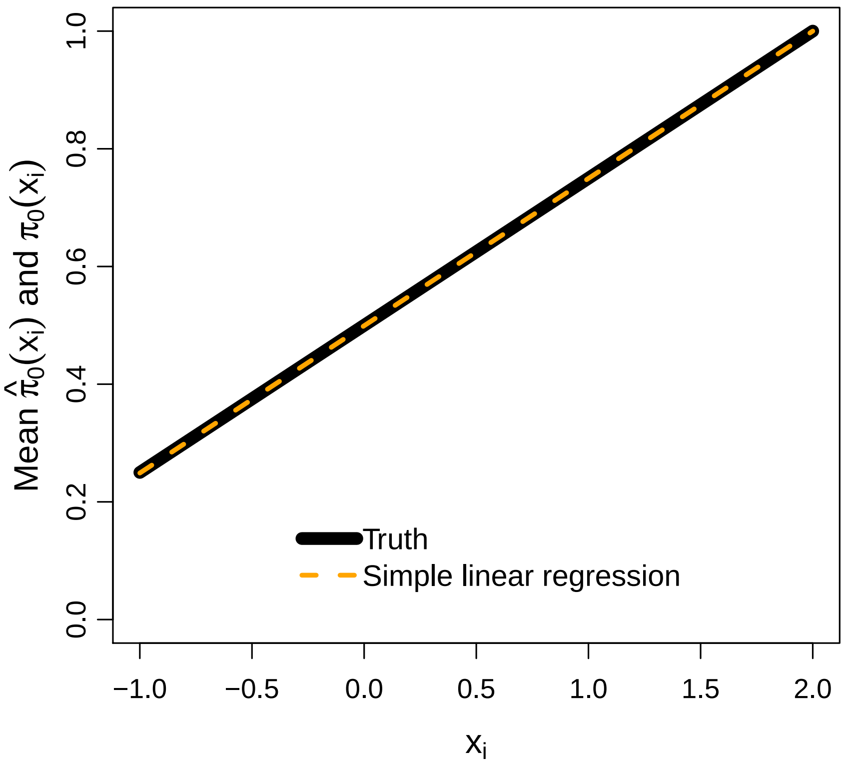
1 Probability of being a false positive as a linear function of time

```r
set.seed(1345)

## Set up the time vector and the probability of being null
tme <- seq(-1,2,length=ntest)
pi0 <- 1/4*tme+1/2

tmeInt <- cbind(1, tme)

##save the value of pi0hat for each simulation
pi0hatMat <- matrix(NA, nrow=nSims, ncol=ntest)

for(sim in 1:nSims)
  {
  ## Calculate a random variable indicating whether to draw
  ## the p-values from the null or alternative
  nullI <- rbinom(ntest,prob=pi0,size=1)> 0

  ## Sample the null P-values from U(0,1) and the alternatives
  ## from a beta distribution

  pValues <- rep(NA,ntest)
  pValues[nullI] <- runif(sum(nullI))
  pValues[!nullI] <- rbeta(sum(!nullI),1,50)

  ## Get the estimate
```

```r
  y <- pValues > lambda

  glm1 <- lsfit(tme, y)

  ## Get the estimated pi0 values
  pi0hatMat[sim, ] <-
    (tmeInt %*%
       matrix(glm1$coefficients, ncol=1))[,1]/(1-lambda)
}

## Get the mean values:
pi0hatMean <- colMeans(pi0hatMat)

##Get the variances:
pi0hatVar <- apply(pi0hatMat, 2, var)

##Get the variance bounds:
zMat <- tmeInt
S <- zMat%*%solve(t(zMat)%*%zMat)%*%t(zMat)
pi0hatVarBound <-
  diag(S)/(4*(1-lambda)^2)
```

1.1 Plot for means

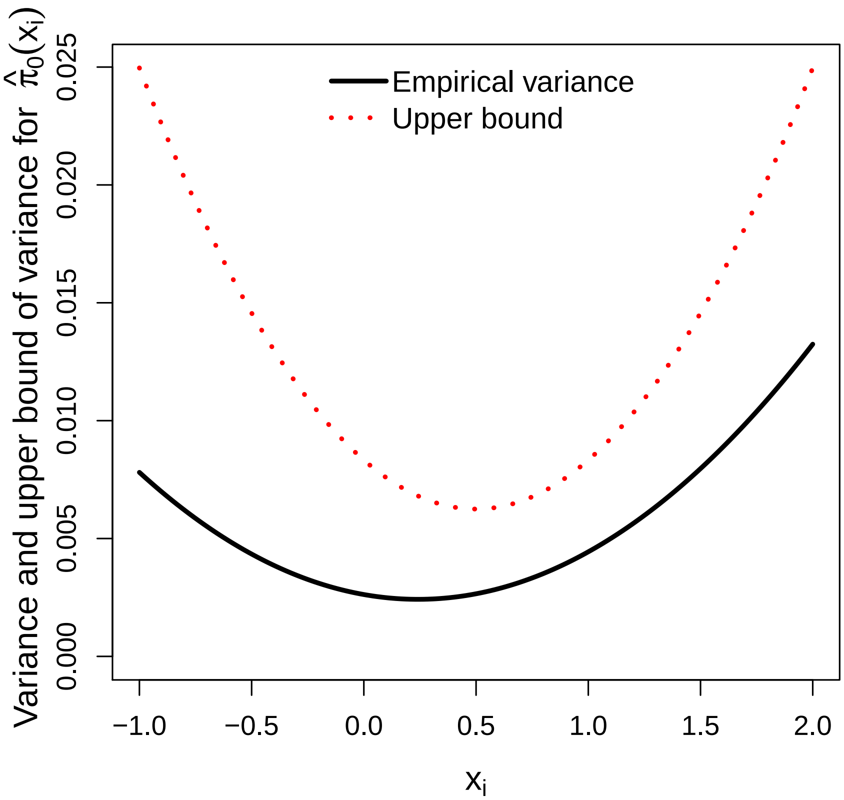
```r
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0 ~ tme,col="black",type="l",lwd=8, lty=1,
     xlab="", yaxt = "n",
     ylim=c(0,1), ylab="")
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Mean ", hat(pi)[0](x[i])," and ", pi[0](x[i]))), 2, line=2, cex=1.3)
points(pi0hatMean ~ tme,col="orange",type="l",lwd=3, lty=2)
legend(x=-0.4, y=0.2,
       legend=c("Truth", "Simple linear regression"),
       col=c("black", "orange"), bty="n",
       lwd=c(8,3), lty=c(1,2),
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
axis(side=2, at=(0:5)/5, mgp=c(3, 0.7, 0))
```



1.2 Plot for variances

```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0hatVarBound ~ tme, col="red", ylim=c(0, max(pi0hatVarBound)),
     lwd=3, lty=3,
     type="l",
     xlab="", ylab="")
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]), s
points(pi0hatVar ~ tme, col="black", type="l", lwd=3, lty=1)
legend("top", ##x=-0.4, y=0.2,
       legend=c("Empirical variance", "Upper bound"),
       col=c("black", "red"), bty="n",
       lwd=c(3,3), lty=c(1,3),
```

```
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
```



2 Probability of being a false positive as a smooth function of time

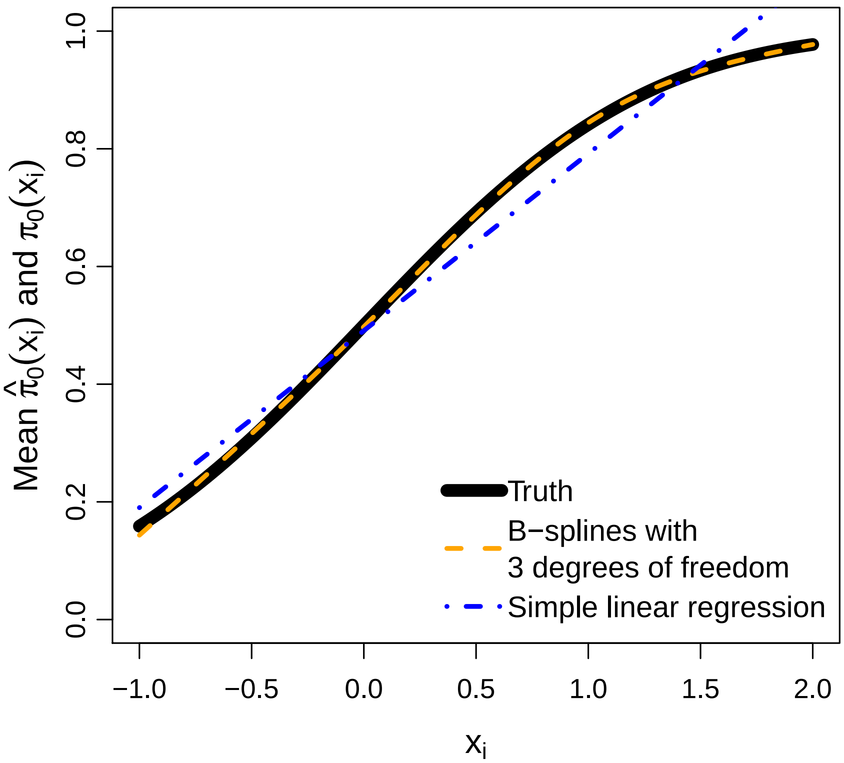
```
set.seed(1345)

## Set up the time vector and the probability of being null
tme <- seq(-1,2,length=ntest)
pi0 <- pnorm(tme)

##save the value of pi0hat for each simulation
##fitting splines:
```

```r
pi0hatMatFitSpl <- matrix(NA, nrow=nSims, ncol=ntest)
##fitting linear function:
pi0hatMatFitLin <- pi0hatMatFitSpl

splineMat <- ns(tme,df=3)
splineMatInt <- cbind(1, splineMat)

for(sim in 1:nSims)
  {
  ## Calculate a random variable indicating whether to draw
  ## the p-values from the null or alternative
  nullI <- rbinom(ntest,prob=pi0,size=1)> 0

  ## Sample the null P-values from U(0,1) and the alternatives
  ## from a beta distribution

  pValues <- rep(NA,ntest)
  pValues[nullI] <- runif(sum(nullI))
  pValues[!nullI] <- rbeta(sum(!nullI),1,50)

  ## Get the estimates

  y <- pValues > lambda

  glm1 <- lsfit(splineMat, y)
  ## Get the estimated pi0 values
  pi0hatMatFitSpl[sim, ] <-
    (splineMatInt %*%
       matrix(glm1$coefficients, ncol=1))[,1]/(1-lambda)

  glm2 <- lsfit(tme, y)
  ## Get the estimated pi0 values
  pi0hatMatFitLin[sim, ] <-
    (cbind(1, tme) %*%
       matrix(glm2$coefficients, ncol=1))[,1]/(1-lambda)
}

## Get the mean values:
pi0hatMeanFitSpl <- colMeans(pi0hatMatFitSpl)
pi0hatMeanFitLin <- colMeans(pi0hatMatFitLin)

##Get the variances:
pi0hatVarFitSpl <- apply(pi0hatMatFitSpl, 2, var)
pi0hatVarFitLin <- apply(pi0hatMatFitLin, 2, var)
```

```
##Get the variance bounds:
zMat <- splineMatInt
S <- zMat%*%solve(t(zMat)%*%zMat)%*%t(zMat)
pi0hatVarBoundFitSpl <-
  diag(S)/(4*(1-lambda)^2)

zMat <- cbind(1, tme)
S <- zMat%*%solve(t(zMat)%*%zMat)%*%t(zMat)
pi0hatVarBoundFitLin <-
  diag(S)/(4*(1-lambda)^2)
```

2.1 Plot for means

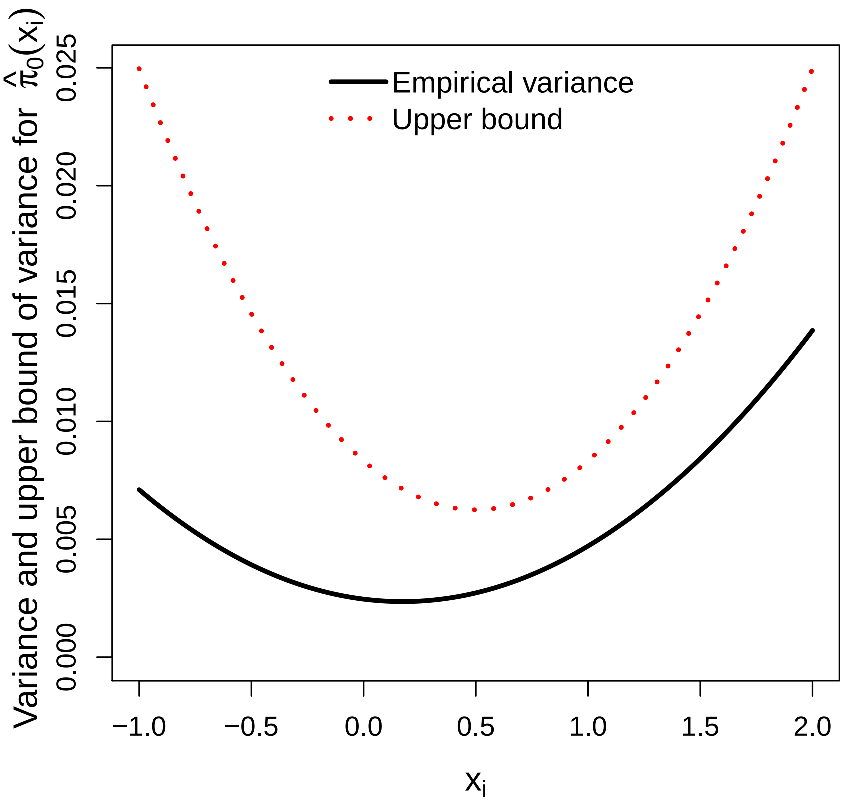
```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0 ~ tme,col="black",type="l",lwd=8, lty=1,
     xlab="", yaxt = "n",
     ylim=c(0,1), ylab="")
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Mean ", hat(pi)[0](x[i])," and ", pi[0](x[i]))), 2, line=2, cex=1.3
points(pi0hatMeanFitSpl ~ tme,
       col="orange",type="l",lwd=3, lty=2)
points(pi0hatMeanFitLin ~ tme,
       col="blue",type="l",lwd=3, lty=4)
legend("bottomright", ##x=-100, y=0.3,
       legend=c("Truth", "B-splines with\n3 degrees of freedom", "Simple linear regression"
       col=c("black", "orange", "blue"), bty="n",
       lwd=c(8,3,3), lty=c(1,2,4),
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
axis(side=2, at=(0:5)/5, mgp=c(3, 0.7, 0))
```



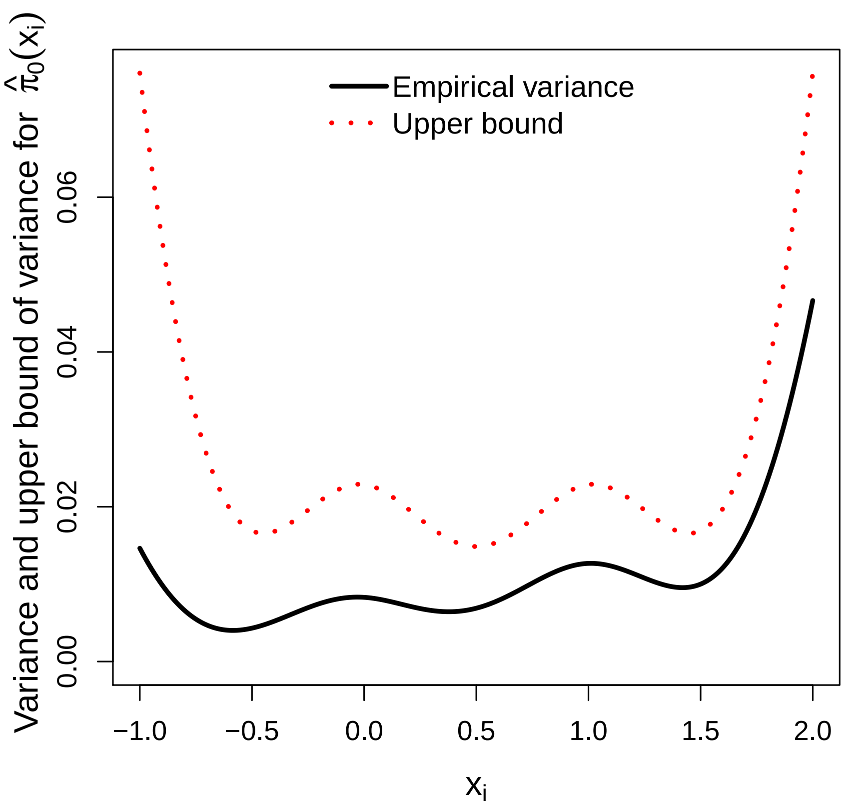
2.2 Plots for variances

```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0hatVarBoundFitLin ~ tme, col="red", ylim=c(0, max(pi0hatVarBoundFitLin)),
     lwd=3, lty=3,
     type="l",
     xlab="", ylab="")
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]), s
points(pi0hatVarFitLin ~ tme, col="black", type="l", lwd=3, lty=1)
legend("top", ##x=-0.4, y=0.2,
       legend=c("Empirical variance", "Upper bound"),
       col=c("black", "red"), bty="n",
       lwd=c(3,3), lty=c(1,3),
```

```
        cex=1.2, x.intersp=0.2, y.intersp=1.0)
```




```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0hatVarBoundFitSpl ~ tme, col="red", ylim=c(0, max(pi0hatVarBoundFitSpl)),
     lwd=3, lty=3,
     type="l",
     xlab="", ylab="")
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]), s
points(pi0hatVarFitSpl ~ tme, col="black", type="l", lwd=3, lty=1)
legend("top", ##x=-0.4, y=0.2,
       legend=c("Empirical variance", "Upper bound"),
       col=c("black", "red"), bty="n",
       lwd=c(3,3), lty=c(1,3),
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
```



3 Probability of being a false positive as a sine + step function

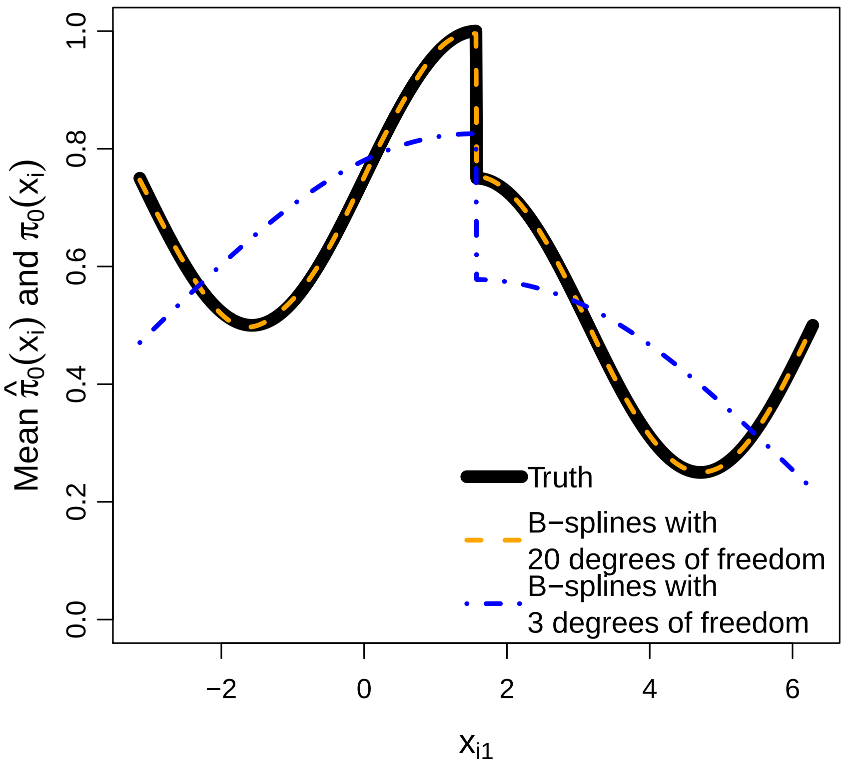
```
set.seed(1345)

## Set up the time vector and the probability of being null
tme1 <- seq(-1*pi,2*pi,length=ntest)
tme2 <- rep(1:0, each=ntest/2)
pi0 <- 1/4*sin(tme1) + tme2/4 + 1/2
##pi0 <- 1/4*sin(tme1) + 0.5
range(pi0)

## [1] 0.2500028 0.9999972
```

```
splineMat3 <- cbind(ns(tme1,df=3), tme2)
splineMat20 <- cbind(ns(tme1,df=20), tme2)
splineMatInt3 <- cbind(1, splineMat3)
splineMatInt20 <- cbind(1, splineMat20)

##save the value of pi0hat for each simulation
pi0hatMat3 <- pi0hatMat20 <-
  matrix(NA, nrow=nSims, ncol=ntest)

for(sim in 1:nSims)
{
  ## Calculate a random variable indicating whether to draw
  ## the p-values from the null or alternative
  nullI <- rbinom(ntest,prob=pi0,size=1)> 0

  ## Sample the null P-values from U(0,1) and the alternatives
  ## from a beta distribution

  pValues <- rep(NA,ntest)
  pValues[nullI] <- runif(sum(nullI))
  pValues[!nullI] <- rbeta(sum(!nullI),1,50)

  ## Get the estimate

  y <- pValues > lambda

  glm1 <- lsfit(splineMat3, y)
  ## Get the estimate pi0 values
  pi0hatMat3[sim, ] <- (splineMatInt3 %*%
                        matrix(glm1$coefficients,
                               ncol=1))[,1]/(1-lambda)

  glm2 <- lsfit(splineMat20, y)
  ## Get the estimate pi0 values
  pi0hatMat20[sim, ] <- (splineMatInt20 %*%
                         matrix(glm2$coefficients,
                                ncol=1))[,1]/(1-lambda)
}

## Get the mean values:
pi0hatMean3 <- colMeans(pi0hatMat3)
pi0hatMean20 <- colMeans(pi0hatMat20)

##Get the variances:
pi0hatVar3 <- apply(pi0hatMat3, 2, var)
```

```
pi0hatVar20 <- apply(pi0hatMat20, 2, var)

##Get the variance bounds:
zMat <- splineMatInt3
S <- zMat%*%ginv(t(zMat)%*%zMat)%*%t(zMat)
pi0hatVarBound3 <-
  diag(S)/(4*(1-lambda)^2)

zMat <- splineMatInt20
S <- zMat%*%ginv(t(zMat)%*%zMat)%*%t(zMat)
pi0hatVarBound20 <-
  diag(S)/(4*(1-lambda)^2)
```

3.1 Plot for means

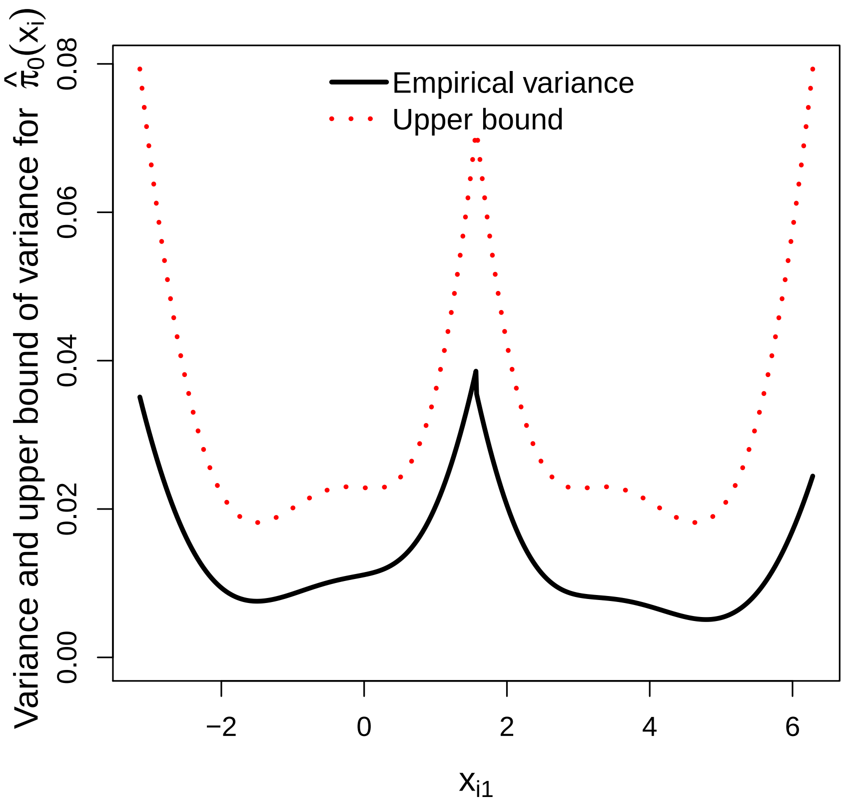
```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0 ~ tme1,col="black",type="l",lwd=8, lty=1,
     xlab="", yaxt = "n",
     ylim=c(0,1), ylab="")
mtext(expression(x[i1]), 1, line=3, cex=1.3)
mtext(expression(paste("Mean ", hat(pi)[0](x[i])," and ",
                       pi[0](x[i]))), 2, line=2, cex=1.3)
points(pi0hatMean20 ~ tme1,col="orange",type="l",lwd=3, lty=2)
points(pi0hatMean3 ~ tme1,col="blue",type="l",lwd=3, lty=4)

legend("bottomright", ##x=-100, y=0.3,
       legend=c("Truth", "B-splines with\n20 degrees of freedom",
                "B-splines with\n3 degrees of freedom"),
       col=c("black", "orange", "blue"), bty="n",
       lwd=c(8,3,3), lty=c(1,2,4),
       cex=1.2, x.intersp=0.2, y.intersp=1.15)
axis(side=2, at=(0:5)/5, mgp=c(3, 0.7, 0))
```



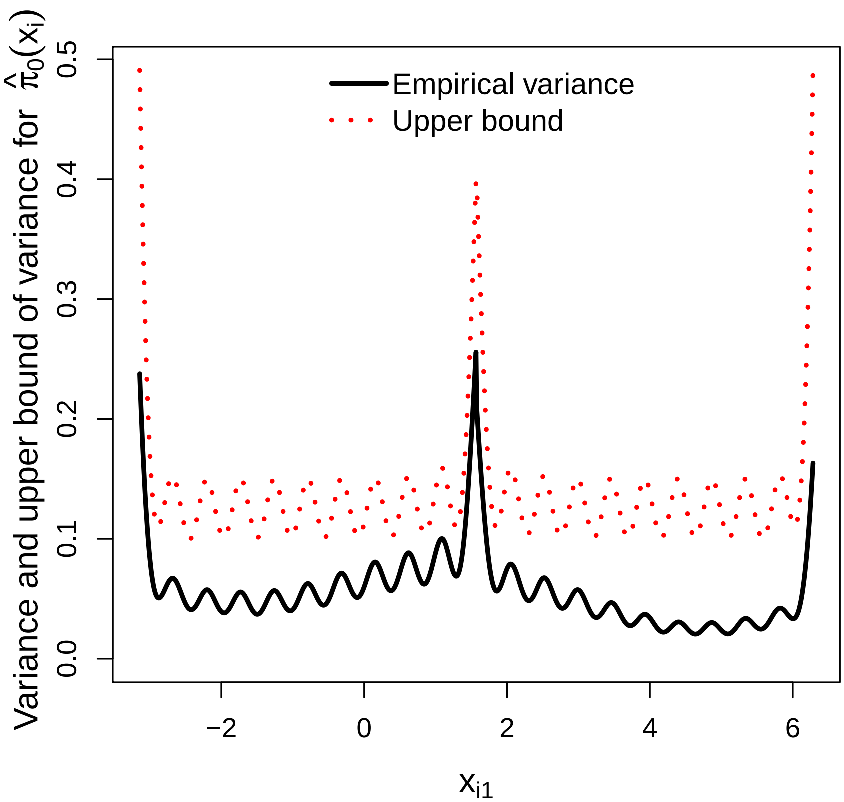
3.2 Plots for variances

```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0hatVarBound3 ~ tme1, col="red", ylim=c(0, max(pi0hatVarBound3)),
     lwd=3, lty=3,
     type="l",
     xlab="", ylab="")
mtext(expression(x[i1]), 1, line=3, cex=1.3)
mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]), s
points(pi0hatVar3 ~ tme1, col="black", type="l", lwd=3, lty=1)
legend("top", ##x=-0.4, y=0.2,
       legend=c("Empirical variance", "Upper bound"),
       col=c("black", "red"), bty="n",
       lwd=c(3,3), lty=c(1,3),
```

```
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
```




```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0hatVarBound20 ~ tme1, col="red", ylim=c(0, max(pi0hatVarBound20)),
     lwd=3, lty=3,
     type="l",
     xlab="", ylab="")
mtext(expression(x[i1]), 1, line=3, cex=1.3)
mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]), s
points(pi0hatVar20 ~ tme1, col="black", type="l", lwd=3, lty=1)
legend("top", ##x=-0.4, y=0.2,
       legend=c("Empirical variance", "Upper bound"),
       col=c("black", "red"), bty="n",
       lwd=c(3,3), lty=c(1,3),
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
```



Session info:

```
devtools::session_info()

## Session info -----------------------------------------------

##  setting  value
##  version  R version 3.1.2 (2014-10-31)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.1252
##  tz       America/New_York
##  date     2015-12-30

## Packages ---------------------------------------------------
```

```
##  package  * version date       source
##  devtools   1.9.1   2015-09-11 CRAN (R 3.1.3)
##  digest     0.6.8   2014-12-31 CRAN (R 3.1.2)
##  evaluate   0.5.5   2014-04-29 CRAN (R 3.1.1)
##  formatR    1.0     2014-08-25 CRAN (R 3.1.2)
##  highr      0.4     2014-10-23 CRAN (R 3.1.2)
##  knitr    * 1.9     2015-01-20 CRAN (R 3.1.2)
##  MASS     * 7.3-35  2014-09-30 CRAN (R 3.1.2)
##  memoise    0.2.1   2014-04-22 CRAN (R 3.1.3)
##  stringr    0.6.2   2012-12-06 CRAN (R 3.1.1)
```