Assignment 4 Question 2 d)

Sheen Thusoo

01/12/2021

Using your results from part (c) construct a plot whose x-axis is degree and which has four lines: one for `apse`, one for `var_mutilde`, one for `bias2` and one for `var_y`. Specifically, and for interpretability, plot `sqrt(apse)`, `sqrt(var_mutilde)`, `sqrt(bias2)` and `sqrt(var_y)` vs. `degree`. Be sure to distinguish the lines with different colours and a legend. Briefly describe the trends you see in the plot.

```r
data <- read.csv("OzoneData.csv")
```

```r
apse_all <- function(Ssamples, Tsamples, complexity, tau) {
    ## average over the samples S
    N_S <- length(Ssamples)
    muhats <- lapply(Ssamples, FUN = function(sample) getmuhat(sample, complexity))
    ## get the average of these, mubar
    mubar <- getmubar(muhats)

    rowMeans(sapply(1:N_S, FUN = function(j) {
        T_j <- Tsamples[[j]]
        S_j <- Ssamples[[j]]
        muhat <- muhats[[j]]
        ## Take care of any NAs
        T_j <- na.omit(T_j)
        y <- c(S_j$y, T_j$y)
        x <- c(S_j$x, T_j$x)

        tau_x <- tau(x)
        muhat_x <- muhat(x)
        mubar_x <- mubar(x)

        apse <- (y - muhat_x)
        bias2 <- (mubar_x - tau_x)
        var_mutilde <- (muhat_x - mubar_x)
        var_y <- (y - tau_x)

        squares <- rbind(apse, var_mutilde, bias2, var_y)^2

        ## return means
        rowMeans(squares)
    }))
}
```

```r
getmubar <- function(muhats) {
    function(x) {
        Ans <- sapply(muhats, FUN = function(muhat) {
            muhat(x)
        })
        apply(Ans, MARGIN = 1, FUN = mean)
    }
}

getmuFun <- function(pop, xvarname, yvarname){
  pop   = na.omit(pop[, c(xvarname, yvarname)])

  # rule = 2 means return the nearest y-value when extrapolating, same as above.
  # ties = mean means that repeated x-values have their y-values averaged, as above.
  tauFun = approxfun(pop[,xvarname], pop[,yvarname], rule = 2, ties = mean)
  return(tauFun)
}

getSampleComp <- function(pop, size, replace=FALSE) {
  N <- popSize(pop)
  samp <- rep(FALSE, N)
  samp[sample(1:N, size, replace = replace)] <- TRUE
  samp
}

getXYSample <- function(xvarname, yvarname, samp, pop) {
  sampData <- pop[samp, c(xvarname, yvarname)]
  names(sampData) <- c("x", "y")
  sampData
}

popSize <- function(pop) {nrow(as.data.frame(pop))}
sampSize <- function(samp) {popSize(samp)}
```

```r
getmuhat <- function(sampleXY, complexity = 1) {
  formula <- paste0("y ~ ",
                    if (complexity==0) {
                      "1"
                      } else
                      paste0("poly(x, ", complexity, ", raw = FALSE)")
                      #paste0("bs(x, ", complexity, ")")
  )

  fit <- lm(as.formula(formula), data = sampleXY)
  tx = sampleXY$x
  ty = fit$fitted.values

  range.X = range(tx)
  val.rY  = c( mean(ty[tx == range.X[1]]),
             mean(ty[tx == range.X[2]]) )

  ## From this we construct the predictor function
  muhat <- function(x){
```

```r
    if ("x" %in% names(x)) {
      ## x is a dataframe containing the variate named
      ## by xvarname
      newdata <- x
    } else
      ## x is a vector of values that needs to be a data.frame
    { newdata <- data.frame(x = x) }
    ## The prediction
    ##
    val = predict(fit, newdata = newdata)
    val[newdata$x < range.X[1]] = val.rY[1]
    val[newdata$x > range.X[2]] = val.rY[2]
    val
  }
  ## muhat is the function that we need to calculate values
  ## at any x, so we return this function from getmuhat
  muhat
}
```

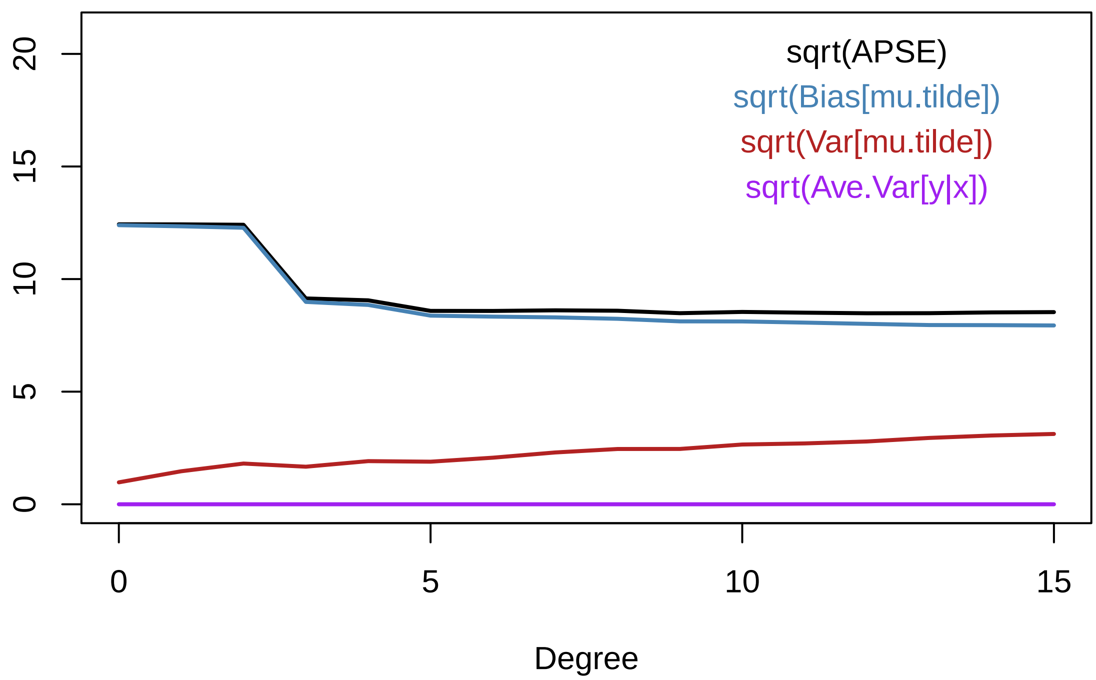
```r
xnam <- "Day"
ynam <- "Ozone"
pop <- data
n <- 100
N_S <- 50

set.seed(1)  # for reproducibility
samples <- lapply(1:N_S, FUN = function(i) {
    getSampleComp(pop, n)
})
Ssam <- lapply(samples, FUN = function(Si) {
    getXYSample(xnam, ynam, Si, pop)
})
Tsam <- lapply(samples, FUN = function(Si) {
    getXYSample(xnam, ynam, !Si, pop)
})
```

```r
degrees <- 0:15
tau.annual = getmuFun(pop, xnam, ynam)

apse_vals <- sapply(degrees, FUN = function(deg) {
    apse_all(Ssam, Tsam, complexity = deg, tau = tau.annual)
})

colnames(apse_vals) = paste("deg=", degrees, sep = "")
round(apse_vals, 3)
```

```
##               deg=0   deg=1   deg=2  deg=3  deg=4  deg=5  deg=6  deg=7  deg=8
## apse        154.563 154.513 154.039 83.544 82.017 73.745 73.710 74.178 73.858
## var_mutilde   0.950   2.147   3.270  2.782  3.671  3.573  4.279  5.291  6.024
## bias2       153.613 152.366 150.769 80.762 78.346 70.172 69.431 68.887 67.834
## var_y         0.000   0.000   0.000  0.000  0.000  0.000  0.000  0.000  0.000
##              deg=9 deg=10 deg=11 deg=12 deg=13 deg=14 deg=15
## apse        72.010 72.952 72.393 71.939 72.014 72.596 72.808
```

```
## var_mutilde  6.033  7.040  7.298  7.774  8.679  9.327  9.751
## bias2       65.977 65.912 65.095 64.165 63.336 63.269 63.057
## var_y        0.000  0.000  0.000  0.000  0.000  0.000  0.000
```

```
plot(degrees, sqrt(apse_vals[2, ]), xlab = "Degree", ylab = "", type = "l", ylim=c(0, 21),
    col = "firebrick", lwd = 2)
lines(degrees, sqrt(apse_vals[1, ]), xlab = "Degree", ylab = "", col = "black", lwd = 2)
lines(degrees, sqrt(apse_vals[3, ]), xlab = "Degree", ylab = "", col = "steelblue", lwd = 2)
lines(degrees, sqrt(apse_vals[4, ]), col = "purple", lwd = 2)
text(12, 20, "sqrt(APSE)", col = "black")
text(12, 18, "sqrt(Bias[mu.tilde])", col = "steelblue")
text(12, 16, "sqrt(Var[mu.tilde])", col = "firebrick")
text(12, 14, "sqrt(Ave.Var[y|x])", col = "purple")
```



Briefly describe the trends you see in the plot.

`sqrt(var_y)` remains at a constant value of zero since the Ozone data had unique x values (no duplicates). `sqrt(apse)` seems to decrease as the degree increases and levels off to a steady-state value just below 10 when the degree is greater than 5. `sqrt(var_mutilde)` seems to increase as the degree increases. It goes from a value near 1 to almost 5 at degree 15. `sqrt(bias2)` seems to follow a similar pattern to `sqrt(apse)` where it decreases as degree increases and levels-off when the degree is greater than 5.