Assignment 4 Question 2 b)

Sheen Thusoo

30/11/2021

Generate $M = 50$ samples $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{50}$ of size $n = 100$. You are encouraged (but don't have to) use functions `getSampleComp` and `getXYSample` from the lectures. Fit polynomials of degree 1, 2, 5, 10, 15, and 20 to every sample.

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

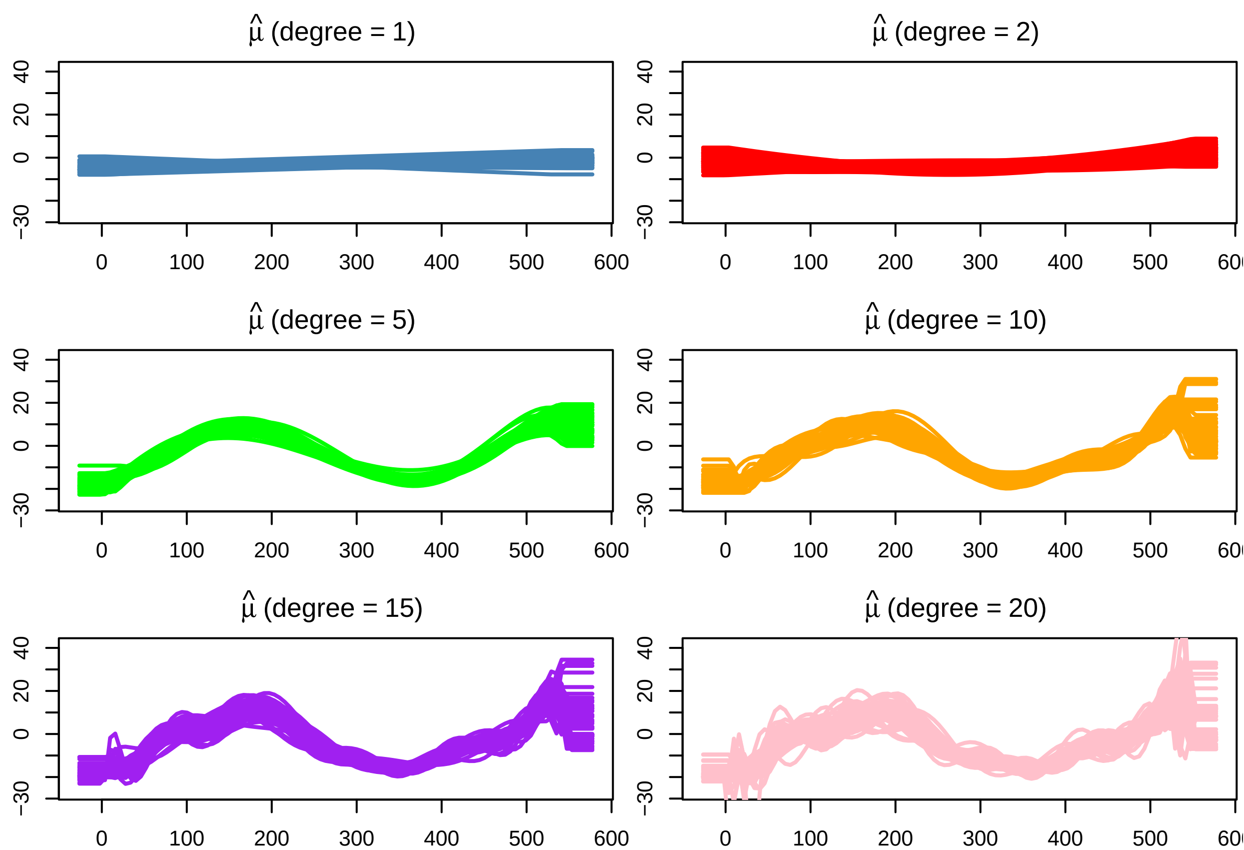
```
ave_y_mu_sq <- function(sample, predfun, na.rm = TRUE){
  mean((sample$y - predfun(sample$x))^2, na.rm = na.rm)
}

ave_mu_mu_sq <- function(predfun1, predfun2, x, na.rm = TRUE){
  mean((predfun1(x) - predfun2(x))^2, na.rm = na.rm)
}

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)}

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

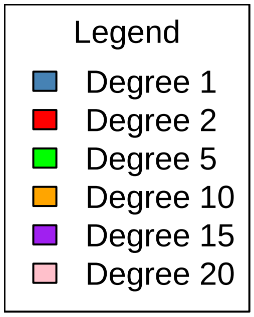
```r
  fit <- lm(as.formula(formula), data = sampleXY)
  tx = sampleXY$x
  ty = fit$fitted.values

  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){
    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
}


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)
}
```

```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)
})
```

```
par(mfrow = c(3, 2), mar = 2.5 * c(1, 1, 1, 0.1))

dset = c(1,2,5,10,15,20)
colors = c("steelblue", "red", "green", "orange", "purple", "pink")

xlim <- extendrange(data[, xnam])
ylim <- extendrange(data[, ynam])

for (i in 1:6) {
  muhats <- lapply(Ssam, getmuhat, complexity = dset[i])

  plot(Ssam[[i]], main = bquote(hat(mu) ~ "(degree =" ~ .(dset[i]) * ") "),
    xlab = xnam, ylab = ynam, pch = 19, col = 0, ylim = ylim, xlim = xlim)

  for (j in 1:N_S) {
      tempfn = muhats[[j]]
      curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col = colors[i],
          lwd = 2)
  }
}
```




Legend
Degree 1
Degree 2
Degree 5
Degree 10
Degree 15
Degree 20