Assignment 4

Fenglin Chen

05/12/2021

### Question 1 a)

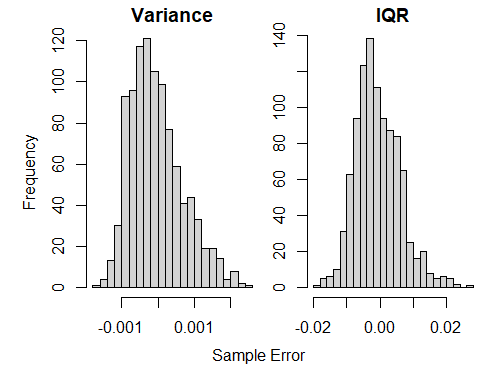
econ <- read.csv("EconomicMobility.csv", header=TRUE)  
  
VarIQR <- function(pop) {  
 pop.var <- sum((pop - mean(pop))^2) / length(pop)  
 pop.iqr <- IQR(pop)  
 c(pop.var, pop.iqr)  
}  
pop.res <- VarIQR(econ$Mobility)  
pop.res

## [1] 0.002768671 0.053418368

The population variance is 0.002768671 and the population IQR is 0.053418368.

### Question 1 b)

M <- 1000  
n <- 100  
set.seed(341)  
samples <- sapply(1:M, FUN = function(m) sample(econ$Mobility, n, replace=FALSE))  
  
samp.res <- apply(samples, MARGIN=2, FUN=VarIQR)  
samp.var.err <- samp.res[1,] - pop.res[1]  
samp.iqr.err <- samp.res[2,] - pop.res[2]  
  
par(mfrow=c(1,2), mar=c(2.5, 2.5, 1.5, 0), oma=c(2, 2, 0, 0))  
hist(samp.var.err, breaks=20, main='Variance')  
hist(samp.iqr.err, breaks=20, main='IQR')  
mtext("Sample Error", side=1, line=0, outer=TRUE, cex=1)  
mtext("Frequency", side=2, line=0, outer=TRUE, cex=1, las=0)



### Question 1 c)

#### i)

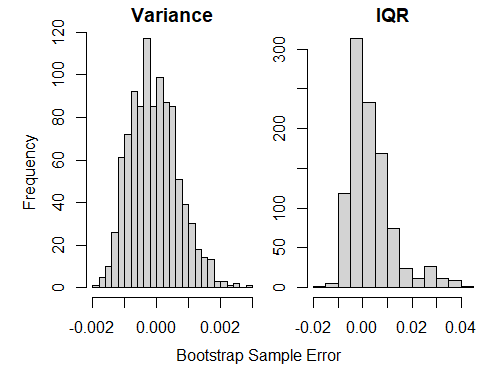
comm.samp <- econ$Mobility[CommunitiesSample]  
comm.samp.res <- VarIQR(comm.samp)  
comm.samp.res

## [1] 0.003185652 0.051400134

The community sample variance is 0.003185652, and the IQR is 0.051400134.

#### ii)

B <- 1000  
n <- length(comm.samp)  
set.seed(341)  
boot.samples <- sapply(1:B, FUN = function(m) sample(comm.samp, n, replace=TRUE))  
  
boot.samp.res <- apply(boot.samples, MARGIN=2, FUN=VarIQR)  
boot.samp.var.err <- boot.samp.res[1,] - comm.samp.res[1]  
boot.samp.iqr.err <- boot.samp.res[2,] - comm.samp.res[2]  
  
par(mfrow=c(1,2), mar=c(2.5, 2.5, 1.5, 0), oma=c(2, 2, 0, 0))  
hist(boot.samp.var.err, breaks=20, main='Variance')  
hist(boot.samp.iqr.err, breaks=20, main='IQR')  
mtext("Bootstrap Sample Error", side=1, line=0, outer=TRUE, cex=1)  
mtext("Frequency", side=2, line=0, outer=TRUE, cex=1, las=0)



#### iii)

se.var <- sd(boot.samp.res[1,])  
se.iqr <- sd(boot.samp.res[2,])  
c(se.var, se.iqr)

## [1] 0.0007497525 0.0090446275

The standard error of the sample estimate of the variance is 0.0007497525, and 0.0090446275 for IQR.

quantile(boot.samp.res[1,], probs=c(0.025, 0.975))

## 2.5% 97.5%   
## 0.001889944 0.004726023

quantile(boot.samp.res[2,], probs=c(0.025, 0.975))

## 2.5% 97.5%   
## 0.04272616 0.08065662

The 95% confidence interval for the population variance and IQR via the percentile method are [0.001889944, 0.004726023] and [0.04272616, 0.08065662] respectively. The population attributes are indeed within these intervals, which validates the results.

### Question 1 d)

set.seed(341)  
numIntervals <- 100  
fit.var <- 0  
fit.iqr <- 0  
for (i in 1:numIntervals){  
 B <- 1000  
 n <- 100  
 one.samp <- sample(econ$Mobility, n, replace=FALSE)  
 boot.samps <- sapply(1:B, FUN = function(m) sample(one.samp, n, replace=TRUE))  
 boot.samps.res <- apply(boot.samps, MARGIN=2, FUN=VarIQR)  
 boot.samps.var <- boot.samps.res[1,]  
 boot.samps.iqr <- boot.samps.res[2,]  
 var.percent <- quantile(boot.samps.var, probs=c(0.025, 0.975))  
 iqr.percent <- quantile(boot.samps.iqr, probs=c(0.025, 0.975))  
 if (pop.res[1] >= var.percent[1] && pop.res[1] <= var.percent[2]) {  
 fit.var <- fit.var + 1  
 }  
 if (pop.res[2] >= iqr.percent[1] && pop.res[2] <= iqr.percent[2]) {  
 fit.iqr <- fit.iqr + 1  
 }  
}  
c(fit.var, fit.iqr)

## [1] 84 95

The 95% percentile-based confidence interval coverage probability of variance is about 84% , and the coverage probability of IQR is about 95%.

sdN = function(x) {  
 sqrt(var(x) \* (length(x) - 1)/length(x))  
}  
  
temp.var <- rep(0, numIntervals)  
temp.var[1:fit.var] <- 1  
sdN(temp.var) / sqrt(numIntervals)

## [1] 0.03666061

temp.iqr <- rep(0, numIntervals)  
temp.iqr[1:fit.iqr] <- 1  
sdN(temp.iqr) / sqrt(numIntervals)

## [1] 0.02179449

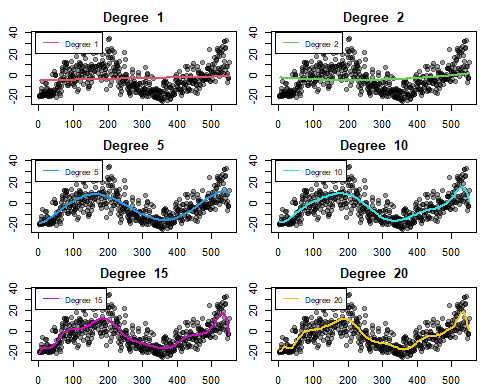
The coverage probability is the count of TRUE/FALSE divided by the number of iterations, so it is a mean. The standard error of a mean is the standard deviation divided by the square root of the sample size.

The coverage probability of variance was far below what was predicted (84 instead of 95) and with a higher standard error (3.666%). Therefore, this attribute is more sensitive to the effects of sampling and bootstrapping. On the other hand, IQR had exactly the predicted coverage probability with a much smaller standard error (2.179%), so it is better suited for bootstrapping.

### Question 2 a)

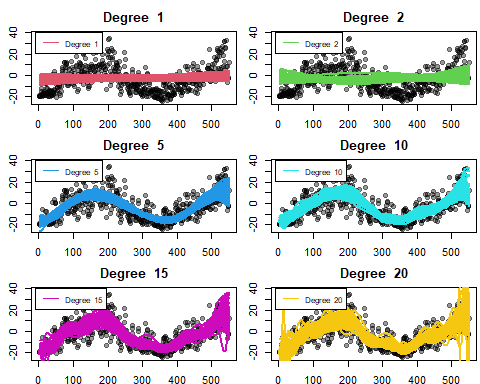
getmuhat <- function(sampleXY, complexity = 1) {  
 formula <- paste0("y ~ ",  
 if (complexity==0) { "1" }   
 else paste0("poly(x, ", complexity, ", raw = FALSE)") )  
 fit <- lm(as.formula(formula), data = sampleXY)  
 tx = sampleXY$x  
 ty = fit$fitted.values  
 range.X = range(tx)  
 val.rY = c( mean(ty[tx == range.X[1]]),   
 mean(ty[tx == range.X[2]]) )  
 muhat <- function(x){  
 if ("x" %in% names(x)) { newdata <- x }   
 else { newdata <- data.frame(x = x) }  
 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  
}  
  
ozone <- read.csv("OzoneData.csv", header=TRUE)  
names(ozone) <- c("x", "y")

par(mfrow = c(3, 2), mar = 2.5 \* c(0.75, 1, 1, 0.1))  
degrees <- c(1, 2, 5, 10, 15, 20)  
colours <- c(2, 3, 4, 5, 6, 7)  
for (i in 1:6) {  
 muhat <- getmuhat(ozone, complexity = degrees[i])  
 plot(y ~ x, data = ozone, main = paste("Degree ", degrees[i]), pch = 19,  
 xlab = "Day", ylab = "Ozone Level", col = adjustcolor("black", 0.4))  
 curve(muhat, add = TRUE, lwd = 2, col = colours[i])  
 legend(-10, 40, legend=paste("Degree ", degrees[i]), col = colours[i], cex=0.8, lty=1)  
}



### Question 2 b)

popSize <- function(pop) {nrow(as.data.frame(pop))}  
getSampleComp <- function(pop, size, replace=FALSE) {  
 N <- popSize(pop)  
 samp <- rep(FALSE, N)  
 samp[sample(1:N, size, replace = replace)] <- TRUE  
 samp  
}  
  
M <- 50  
n <- 100  
set.seed(341)  
samples <- lapply(1:M, FUN = function(i) {getSampleComp(ozone, n)})  
Ssam <- lapply(samples, FUN = function(Si) {ozone[Si,]})  
Tsam <- lapply(samples, FUN = function(Si) {ozone[!Si,]})  
  
par(mfrow = c(3, 2), mar = 2.5 \* c(0.75, 1, 1, 0.1))  
degrees <- c(1, 2, 5, 10, 15, 20)  
colours <- c(2, 3, 4, 5, 6, 7)  
for (i in 1:6) {  
 fits <- lapply(Ssam, FUN=getmuhat, complexity = degrees[i])  
 plot(y ~ x, data = ozone, main = paste("Degree ", degrees[i]), pch = 19,  
 xlab = "Day", ylab = "Ozone Level", col = adjustcolor("black", 0.4))  
 for (fit in fits) {  
 curve(fit, add = TRUE, lwd = 2, col = colours[i])  
 }  
 legend(-10, 40, legend=paste("Degree ", degrees[i]), col = colours[i], cex=0.8, lty=1)  
}



### Question 2 c)

library(knitr)  
getmuFun <- function(pop) {  
 pop = na.omit(pop)  
 tauFun = approxfun(pop$x, pop$y, rule = 2, ties = mean)  
 tauFun  
}  
  
getmubar <- function(muhats) {  
 function(x) {  
 Ans <- sapply(muhats, FUN = function(muhat) { muhat(x) })  
 apply(Ans, MARGIN = 1, FUN = mean)  
 }  
}  
  
apse\_all <- function(Ssamples, Tsamples, complexity, tau) {  
 N\_S <- length(Ssamples)  
 muhats <- lapply(Ssamples, FUN = function(sample) getmuhat(sample, complexity))  
 mubar <- getmubar(muhats)  
   
 rowMeans(sapply(1:N\_S, FUN = function(j) {  
 T\_j <- Tsamples[[j]]  
 S\_j <- Ssamples[[j]]  
 muhat <- muhats[[j]]  
 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  
 rowMeans(squares)  
 }))  
}  
  
degrees <- 0:15  
tauFun = getmuFun(ozone)  
apse\_vals <- sapply(degrees, FUN = function(deg) {  
 apse\_all(Ssam, Tsam, complexity = deg, tau = tauFun)  
})  
colnames(apse\_vals) = paste("deg", degrees, sep = " ")

kable(round(apse\_vals, 3))

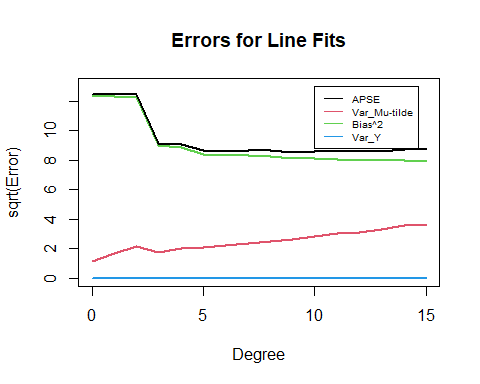
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | deg 0 | deg 1 | deg 2 | deg 3 | deg 4 | deg 5 |
| apse | 154.899 | 155.162 | 155.372 | 83.990 | 82.539 | 74.452 |
| var\_mutilde | 1.283 | 2.741 | 4.555 | 3.171 | 4.207 | 4.279 |
| bias2 | 153.616 | 152.421 | 150.817 | 80.819 | 78.332 | 70.174 |
| var\_y | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | deg 6 | deg 7 | deg 8 | deg 9 | deg 10 | deg 11 |
| apse | 74.382 | 74.681 | 74.420 | 72.921 | 73.826 | 74.216 |
| var\_mutilde | 4.906 | 5.614 | 6.408 | 6.970 | 7.923 | 9.305 |
| bias2 | 69.476 | 69.068 | 68.012 | 65.951 | 65.903 | 64.911 |
| var\_y | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | deg 12 | deg 13 | deg 14 | deg 15 |
| apse | 73.807 | 74.382 | 75.980 | 76.408 |
| var\_mutilde | 9.727 | 11.047 | 12.622 | 13.297 |
| bias2 | 64.080 | 63.335 | 63.357 | 63.111 |
| var\_y | 0.000 | 0.000 | 0.000 | 0.000 |

### Question 2 d)

plot(degrees, sqrt(apse\_vals[3,]), main="Errors for Line Fits", xlab = "Degree",   
 ylab = "sqrt(Error)", type = "l", ylim = c(0, 13), col = 3, lwd = 2)  
lines(degrees, sqrt(apse\_vals[2,]), col = 2, lwd = 2)  
lines(degrees, sqrt(apse\_vals[1,]), col = 1, lwd = 2)  
lines(degrees, sqrt(apse\_vals[4,]), col = 4, lwd = 2)  
legend(10, 13, legend=c("APSE", "Var\_Mu-tilde", "Bias^2", "Var\_Y"),   
 col = 1:4, cex=0.65, lty=1)



The plot shows that APSE decreases dramatically from deg=2 to deg=3, mainly due to decreasing bias^2. It is lowest at deg=9, but increases slightly after due to increasing var\_mutilde. Clearly, there is a trade-off between bias and variance. On the other hand, var\_y stays constant at 0 for the entire plot.

### Question 2 e)

Degree 9 has the best predictive accuracy because it has the lowest APSE.

muhat <- getmuhat(ozone, complexity = 9)  
plot(y ~ x, data = ozone, main = "Degree 9", pch = 19,  
 xlab = "Day", ylab = "Ozone Level", col = adjustcolor("black", 0.4))  
curve(muhat, add = TRUE, lwd = 2, col = 2)

