Chapter 4 Exercise 4

Allen Church

10/9/2019

1. Run simulation code in the initial part of the education and salary question from the exercises in Ch 3 (p 87)

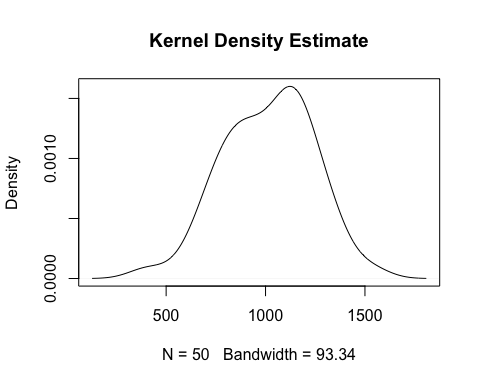
## Set model and simulation parameters  
Obs = 100 # Number of observations in each simulation  
Reps = 50 # Number of times we run the simulation  
TrueBeta0 = 12000 # "True" beta0 for the simulated  
TrueBeta1 = 1000 # "True" beta1 for the simulated  
SD = 10000 # The standard deviation of the error. The bigger this is, the larger the average value of epsilon  
Ed = 16 \* runif(Obs)# Simulate years of education as being between 0 and 16  
 # "runif" is a uniform random variable between 0 and 1, with all values having equal probability  
CoefMatrix = matrix(NA, Reps, 2)  
SEMatrix = matrix(NA, Reps, 2)  
# Matrix to store our results - for both coefficients and std errors  
 # 1st argument is NA, meaning we store "not available" as initial values in the matrix  
 # 2nd argument is Reps, meaning the number of rows is equal to number of times we run the simulations  
 # 3rd argument is 2 meaning we have 2 columns, one for storing the beta0 estimate and one for storing the beta1 estimate  
  
# Loop: repeat the commands between the brackets multiple times  
for (ii in 1:Reps) {   
 Salary = TrueBeta0+ TrueBeta1\* Ed + SD\*rnorm(Obs)   
 # Generate Salary = beta0 + beta1\*Ed + epsilon  
 # beta0 is the constant  
 # beta1 is the number multiplied by the X variable  
 # Epsilon has 2 parts: SD is the standard deviation; the bigger it is, the more epsilon varies.  
 # "runif" is a uniform random variable between 0 and 1, with all values having equal probability  
 OLS.result = lm(Salary ~ Ed) # Run a regression using simulated values of Y  
 CoefMatrix[ii, ] = coefficients(OLS.result) # Put OLS.result coefficients in row ii of CoefMatrix  
 SEMatrix[ii, ] = coef(summary(OLS.result))[, "Std. Error"]  
 ## For fun: plot results for each survey  
 ## plot(Ed, Salary, pch = 19, col= "darkgreen")   
 ## abline(OLS.result, lwd = 3, col= "darkgreen")  
 ## Sys.sleep(0.075) ## Include to slow down calculations so we can see each plot (briefly); not necessary  
} # This closes the "loop"  
  
c(mean(CoefMatrix[,1]), min(CoefMatrix[,1]), max(CoefMatrix[,1]))

## [1] 11475.157 7389.251 14996.599

# Average, min and max of beta\_0 estimates  
  
c(mean(CoefMatrix[,2]), min(CoefMatrix[,2]), max(CoefMatrix[,2]))

## [1] 1013.7543 407.5665 1528.6311

# Average, min and max of beta\_1 estimates  
  
#  
# For use in Chapter 3, #2 part (g):  
#  
# Kernel Density Plot  
plot(density(CoefMatrix[,2]), main = 'Kernel Density Estimate')



summary(OLS.result)

##   
## Call:  
## lm(formula = Salary ~ Ed)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -20587.2 -6109.7 191.5 6536.7 22935.0   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 10889.0 2084.3 5.224 9.84e-07 \*\*\*  
## Ed 1120.9 222.8 5.030 2.22e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 9880 on 98 degrees of freedom  
## Multiple R-squared: 0.2052, Adjusted R-squared: 0.1971   
## F-statistic: 25.3 on 1 and 98 DF, p-value: 2.222e-06

t stat = beta 1 hat / standard error of beta 1 hat tmatrix <- CoefMatrix/SEmatrix

#Create a T matrix, which is results from the coefficient matrix divided by results from the standard error matrix  
T.matrix = CoefMatrix/SEMatrix  
  
## Minimum t-stats for both coefficients (b0, b1)  
c(min(T.matrix[, 1]), min(T.matrix[, 2]) )

## [1] 3.435126 1.939765

#Print minimum beta 0 t-stat to user  
print(paste0("The minimum t-stat for beta 0 is ", min(T.matrix[, 1])))

## [1] "The minimum t-stat for beta 0 is 3.43512578577799"

#Print minimum beta 1 t-stat to user  
print(paste0("The minimum t-stat for beta 1 is ", min(T.matrix[, 2])))

## [1] "The minimum t-stat for beta 1 is 1.93976534143009"

## Maximum t-stats for both coefficients (b0, b1)  
c(max(T.matrix[, 1]), max(T.matrix[, 2]) )

## [1] 7.173698 6.929594

#Print maximum beta 0 t-stat to user  
print(paste0("The maximum t-stat for beta 0 is ", max(T.matrix[, 1])))

## [1] "The maximum t-stat for beta 0 is 7.17369794854556"

#Print maximum beta 1 t-stat to user  
print(paste0("The maximum t-stat for beta 1 is ", max(T.matrix[, 2])))

## [1] "The maximum t-stat for beta 1 is 6.92959357960424"

4b. Generate two-sided p values for the coefficient on education for each simulation. What are minimal and maximal values of these p values?

#Create two sided p values, and assign degrees of freedom from lm summary in 4a.  
deg\_freedom <- 98  
pvals <- 2\*(1-pt(abs(T.matrix),deg\_freedom))  
  
#Create P matrix with necessary shape to store results below  
P.Matrix = matrix(NA, Reps, 2)  
  
# For loop to assign p vals into the P Matrix  
for (ii in 1:Reps) {   
 #OLS.result = lm(Salary ~ Ed) # Run a regression using simulated values of Y  
 #CoefMatrix[ii, ] = coefficients(OLS.result) # Put OLS.result coefficients in row ii of CoefMatrix  
 #SEMatrix[ii, ] = coef(summary(OLS.result))[, "Std. Error"]  
 P.Matrix = pvals  
   
}  
  
## Minimum t-stats for both coefficients (b0, b1)  
c(min(P.Matrix[, 1]), min(P.Matrix[, 2]) )

## [1] 1.402496e-10 4.489902e-10

#Print minimum beta 0 p value to user  
print(paste0("The minimum p value for beta 0 is ", min(P.Matrix[, 1])))

## [1] "The minimum p value for beta 0 is 1.40249589719588e-10"

#Print minimum beta 1 p value to user  
print(paste0("The minimum p value for beta 1 is ", min(P.Matrix[, 2])))

## [1] "The minimum p value for beta 1 is 4.48990178369968e-10"

## Maximum p values for both coefficients (b0, b1)  
c(max(P.Matrix[, 1]), max(P.Matrix[, 2]) )

## [1] 0.0008698961 0.0552832335

#Print maximum beta 0 p value to user  
print(paste0("The maximum p value for beta 0 is ", max(P.Matrix[, 1])))

## [1] "The maximum p value for beta 0 is 0.00086989605400456"

#Print maximum beta 1 p value to user  
print(paste0("The maximum p value for beta 1 is ", max(P.Matrix[, 2])))

## [1] "The maximum p value for beta 1 is 0.0552832335226512"

4c. In what percent of simulations do we reject null hypothesis that Beta education = 0 at alpha = 0.05 level with a two-sided alternative hypothesis?

The maximum p value for Beta 1 above is 0.032. This indicates a 0.032 probability of observing the Beta education coefficient as large as observed if the Beta education coefficient equals 0.

4d. Re-run simulations and set true value of Beta education to 0. Do this for 500 simulations and report what percent of time we reject the null at alpha = 0.05 level with a two-sided alternative hypothesis

## Set model and simulation parameters  
Obs\_1 = 100 # Number of observations in each simulation  
Reps\_1 = 500 # Number of times we run the simulation  
TrueBeta0\_1 = 48 # "True" beta0 for the simulated  
TrueBeta1\_1 = 0 # "True" beta1 for the simulated  
SD = 10000 # The standard deviation of the error. The bigger this is, the larger the average value of epsilon  
Ed\_1 = 16 \* runif(Obs)# Simulate years of education as being between 0 and 16  
 # "runif" is a uniform random variable between 0 and 1, with all values having equal probability  
CoefMatrix\_1 = matrix(NA, Reps\_1, 2)  
SEMatrix\_1 = matrix(NA, Reps\_1, 2)  
  
# Matrix to store our results - for both coefficients and std errors  
 # 1st argument is NA, meaning we store "not available" as initial values in the matrix  
 # 2nd argument is Reps, meaning the number of rows is equal to number of times we run the simulations  
 # 3rd argument is 2 meaning we have 2 columns, one for storing the beta0 estimate and one for storing the beta1 estimate  
  
# Loop: repeat the commands between the brackets multiple times  
for (ii in 1:Reps\_1) {   
 Salary\_1 = TrueBeta0\_1+ TrueBeta1\_1\* Ed\_1 + SD\*rnorm(Obs\_1)   
 # Generate Salary = beta0 + beta1\*Ed + epsilon  
 # beta0 is the constant  
 # beta1 is the number multiplied by the X variable  
 # Epsilon has 2 parts: SD is the standard deviation; the bigger it is, the more epsilon varies.  
 # "runif" is a uniform random variable between 0 and 1, with all values having equal probability  
 OLS.result\_1 = lm(Salary\_1 ~ Ed\_1) # Run a regression using simulated values of Y  
 CoefMatrix\_1[ii, ] = coefficients(OLS.result\_1) # Put OLS.result coefficients in row ii of CoefMatrix  
 SEMatrix\_1[ii, ] = coef(summary(OLS.result\_1))[, "Std. Error"]  
}  
  
T.matrix\_1 = CoefMatrix\_1/SEMatrix\_1  
  
#Create two sided p values, and assign degrees of freedom from lm summary in 4a.  
deg\_freedom\_1 <- 98  
pvals\_1 <- 2\*(1-pt(abs(T.matrix\_1),deg\_freedom\_1))  
  
#Create P matrix with necessary shape to store results below  
P.Matrix\_1 = matrix(NA, Reps\_1, 2)  
  
# For loop to assign p vals into the P Matrix  
for (ii in 1:Reps\_1) {   
 #OLS.result = lm(Salary ~ Ed) # Run a regression using simulated values of Y  
 #CoefMatrix[ii, ] = coefficients(OLS.result) # Put OLS.result coefficients in row ii of CoefMatrix  
 #SEMatrix[ii, ] = coef(summary(OLS.result))[, "Std. Error"]  
 P.Matrix\_1 = pvals\_1  
   
}  
  
## Minimum t-stats for both coefficients (b0, b1)  
c(min(P.Matrix\_1[, 1]), min(P.Matrix\_1[, 2]) )

## [1] 0.002944679 0.003190032

#Print minimum beta 0 p value to user  
print(paste0("The minimum p value for beta 0 is ", min(P.Matrix\_1[, 1])))

## [1] "The minimum p value for beta 0 is 0.00294467935780052"

#Print minimum beta 1 p value to user  
print(paste0("The minimum p value for beta 1 is ", min(P.Matrix\_1[, 2])))

## [1] "The minimum p value for beta 1 is 0.00319003186935474"

## Maximum p values for both coefficients (b0, b1)  
c(max(P.Matrix\_1[, 1]), max(P.Matrix\_1[, 2]) )

## [1] 0.9982206 0.9997671

#Print maximum beta 0 p value to user  
print(paste0("The maximum p value for beta 0 is ", max(P.Matrix\_1[, 1])))

## [1] "The maximum p value for beta 0 is 0.998220561599399"

#Print maximum beta 1 p value to user  
print(paste0("The maximum p value for beta 1 is ", max(P.Matrix\_1[, 2])))

## [1] "The maximum p value for beta 1 is 0.999767129286395"

After re-running the simulations with 500 reps and setting a true beta 1 at 0, the minimum p value for Beta 1 above is 0.004, however maximum p value for Beta 1 is 0.99. Due to this range of values, we cannot decisively reject the null.