STA 141 HW 5

Jiewei Chen (999 494 235)

11/28/2016

1.

Function for (i) - (iii)

The function **Create.Data** uses **s** as input, which is a number used to set the random seed every time we run this funciton. In the function, it generates **X's**, **Y's** as described in the question 1(i). And this function will return a matrix with column representing **X's**, ϵ, **Y's**.

The function **Cal.beta** uses **X** and **Y** as inputs, it fits the least squares regression line to the data and outputs the estimate of (β_0, β_1, σ^2).

The function **Resampling** uses **data** as inputs, it resamples the residuals of a model and then create a new dataset by using these resampled residuals. Then it computes the two coefficients of the new model. The output of this function is the two coefficients calculated from the resampled dataset.

The function **step1to3** just concatenates Question (i) to (iii) together to get the estimate of (β_0, β_1, σ^2), and confidence intervals of two coefficients.

```
step1to3 = function (i) {
        library(tidyverse)
        library(readr)
        library(broom)

        Create.Data = function (s) {
                # s  is the seed
                # Generate Chi-squared distribution
                set.seed(s)
                X = rchisq(100, 6, ncp = 0)
                # Generate Normail distribution
                set.seed(s)
                epsilon = rnorm(100, mean = 0, sd = 1)

                Y = -5 + 2*X + epsilon

                result = as_data_frame(cbind(X, Y))
                return(result)
        }

        result_1 = Create.Data(i)

        Cal.beta = function(X,Y) {
                model1 = lm(Y ~ X)
                SSE = sum(model1$residuals^2)
                n = length(X)
                MSE = SSE/(n-2)
                return(c(coef(model1), MSE))
        }
```

```r
        result_2 = Cal.beta(result_1$X, result_1$Y)

        # Residual bootstrap (resample residuals)
        resid = augment(lm(Y~X, result_1))

        Resampling = function(data) {
                n = nrow(data)
                # Sample row numbers (i) rather than values (e_i)
                idx = sample(n, n, replace = TRUE)

                # Use row numbers to get new residuals (e2_i).
                res_samp = data$.resid[idx]

                # y2_i =  b_0 + b_1 * x_i    + e2_i
                y_samp =  data$.fitted + res_samp

                # Insert new response (y_i) into data frame, keeping old covariates (x_i)
                data$Y = y_samp

                # Fit the same model with new data (y2_i, x_i).
                new_mod = lm(Y ~ X, data)

                return (coef(new_mod))
        }

        # Bootstrap 400 times.
        boot = replicate(400, Resampling(resid))

        # Now compute statistics on the bootstrap samples. Each column is one bootstrap
        # sample and each row is one statistic.
        #
        # For 95% confidence intervals:

        ci_beta0 = quantile(boot[1, ], c(0.025, 0.975))
        ci_beta1 = quantile(boot[2, ], c(0.025, 0.975))

        # theoretical confidence intervals
        ci_beta0_theo = confint(lm(Y~X, result_1), level = 0.95)[1,]
        ci_beta1_theo = confint(lm(Y~X, result_1), level = 0.95)[2,]

        result = list("est.beta0" = result_2[1], "est.beta1" = result_2[2],
                      "est.sigma2" = result_2[3],
                      "CI.beta0" = ci_beta0, "CI.beta1" = ci_beta1,
                      "CI.beta0.theo" = ci_beta0_theo, "CI.beta1.theo" = ci_beta1_theo)

        return(result)
}

result.1 = step1to3(1)
```

(ii)

So the estimation of β_0, β_1, σ^2 are

Estimation of	**Value**
$\hat{\beta}_0$	-4.8417927
$\hat{\beta}_1$	1.9915432
$\hat{\sigma^2}$	0.8144113

(iii)

So the resampling-based 95% confidence intervals for β_0 and β_1 by using a parametric (i.e., residual-base) bootstrap procedure with 400 bootstrap replicates are

Coefficient	95% Confidence Interval
β_0	[-5.2453447, -4.4360414]
β_1	[1.9268544, 2.0577101]

(iv)

```
# Do step (i) to (iii) three times
result.10 = lapply(1:10, step1to3)

# Tabulate the confidence intervals and length of Beta0 from Bootstrap
CI.beta0.10 = as_data_frame(t(sapply(1:10, function(i) {
        cbind(result.10[[i]]$CI.beta0) } )))
colnames(CI.beta0.10) = c("Lower.Bound", "Higher.Bound")
CI.beta0.10$length = CI.beta0.10$Higher.Bound - CI.beta0.10$Lower.Bound

# Tabulate the theoretical confidence intervals and length of Beta0
CI.beta0.10.theo = as_data_frame(t(sapply(1:10, function(i) {
        cbind(result.10[[i]]$CI.beta0.theo) } )))
colnames(CI.beta0.10.theo) = c("Lower.Bound", "Higher.Bound")
CI.beta0.10.theo$length = CI.beta0.10.theo$Higher.Bound - CI.beta0.10.theo$Lower.Bound

# Tabulate the confidence intervals and length of Beta1 from Bootstrap
CI.beta1.10 = as_data_frame(t(sapply(1:10, function(i) {
        cbind(result.10[[i]]$CI.beta1) } )))
colnames(CI.beta1.10) = c("Lower.Bound", "Higher.Bound")
CI.beta1.10$length = CI.beta1.10$Higher.Bound - CI.beta1.10$Lower.Bound

# Tabulate the theoretical confidence intervals and length of Beta1
CI.beta1.10.theo = as_data_frame(t(sapply(1:10, function(i) {
        cbind(result.10[[i]]$CI.beta1.theo) } )))
colnames(CI.beta1.10.theo) = c("Lower.Bound", "Higher.Bound")
CI.beta1.10.theo$length = CI.beta1.10.theo$Higher.Bound - CI.beta1.10.theo$Lower.Bound
```

The theoretic CI for β_0 is

```
## # A tibble: 10 × 3
##    Lower.Bound Higher.Bound    length
##          <dbl>        <dbl>     <dbl>
## 1    -5.252405    -4.431180 0.8212255
## 2    -5.404225    -4.480598 0.9236275
## 3    -5.284119    -4.539377 0.7447422
## 4    -5.100959    -4.416712 0.6842476
## 5    -5.324348    -4.568587 0.7557607
## 6    -5.364946    -4.560291 0.8046550
## 7    -5.311670    -4.406385 0.9052844
## 8    -5.519438    -4.650472 0.8689663
## 9    -5.754756    -4.954005 0.8007502
## 10   -5.743432    -4.933970 0.8094612
```

The CI computed from Bootstrap for β_0 is

```
## # A tibble: 10 × 3
##    Lower.Bound Higher.Bound    length
##          <dbl>        <dbl>     <dbl>
## 1    -5.245345    -4.436041 0.8093033
## 2    -5.413617    -4.452198 0.9614191
## 3    -5.268578    -4.558963 0.7096157
## 4    -5.053472    -4.422843 0.6306299
## 5    -5.309086    -4.586302 0.7227839
## 6    -5.317996    -4.575300 0.7426959
## 7    -5.343331    -4.345186 0.9981445
## 8    -5.503191    -4.607643 0.8955479
## 9    -5.709065    -4.960073 0.7489921
## 10   -5.728658    -4.958214 0.7704442
```

The theoretic CI for β_1 is

```
## # A tibble: 10 × 3
##    Lower.Bound Higher.Bound     length
##          <dbl>        <dbl>      <dbl>
## 1     1.928186     2.054901 0.12671515
## 2     1.919421     2.051427 0.13200630
## 3     1.925896     2.046070 0.12017409
## 4     1.929861     2.023467 0.09360576
## 5     1.935884     2.056080 0.12019602
## 6     1.923331     2.058108 0.13477646
## 7     1.936442     2.062857 0.12641504
## 8     1.943226     2.054315 0.11108868
## 9     1.991836     2.102000 0.11016392
## 10    1.973933     2.093184 0.11925090
```

The CI computed from Bootstrap for β_1 is

```
## # A tibble: 10 × 3
##    Lower.Bound Higher.Bound     length
##          <dbl>        <dbl>      <dbl>
## 1     1.926854     2.057710 0.13085565
## 2     1.910842     2.061265 0.15042310
## 3     1.927382     2.054088 0.12670542
## 4     1.931213     2.015889 0.08467642
## 5     1.935712     2.053508 0.11779652
## 6     1.928704     2.055300 0.12659644
## 7     1.932337     2.069588 0.13725168
## 8     1.941236     2.053353 0.11211750
## 9     1.990092     2.097493 0.10740048
## 10    1.979442     2.093761 0.11431865
```

```
len.beta0.theo = mean(CI.beta0.10.theo$length)
len.beta0 = mean(CI.beta0.10$length)
len.beta1.theo = mean(CI.beta1.10.theo$length)
len.beta1 = mean(CI.beta1.10$length)
```

The average lengths of the bootstrap confidence intervals and that of corresponding theoretical confidence intervals are listed below

Coefficient	Length of CI from Bootstrap	Length of CI from Theoretics
β_0	0.7989576	0.8118721
β_1	0.1208142	0.1194392