R Notebook

Utility function to read csv

```
read_csv_func = function(x){
  df = read.csv(x, header=TRUE) # Read CSV
  return (df)
}
```

Custom utility function to extract specific column from dataframe

```
extract_column_func = function(df, column_name, keep_char=FALSE){
  column_names_list = names(df)
  column_index = match(column_name, column_names_list)
  column_data = df[, column_index]
  if(typeof(column_data) == "character" && keep_char==FALSE)
  {
    column_data = as.double(column_data)
  }
  return (column_data)
}
```

Utility function t0 extract rows from df based on condition

```
extract_rows_based_on_condition = function(df, column_name, condition){
  columns_list = names(df) # Getting list of column names in dataframe

  column_index = match(column_name, columns_list)
  # Getting index of column where are looking for from the column list

  df_constraint = df[df[,column_index] == condition,]
  # Applying constraint to get rows which are matching a condition
  return (df_constraint)
}
```

Solution 1 (a)

```
bodytemp_heartrate_df =
  read_csv_func("/Users/karthik_ragunath/Desktop/Stats/bodytemp-heartrate.csv")
```

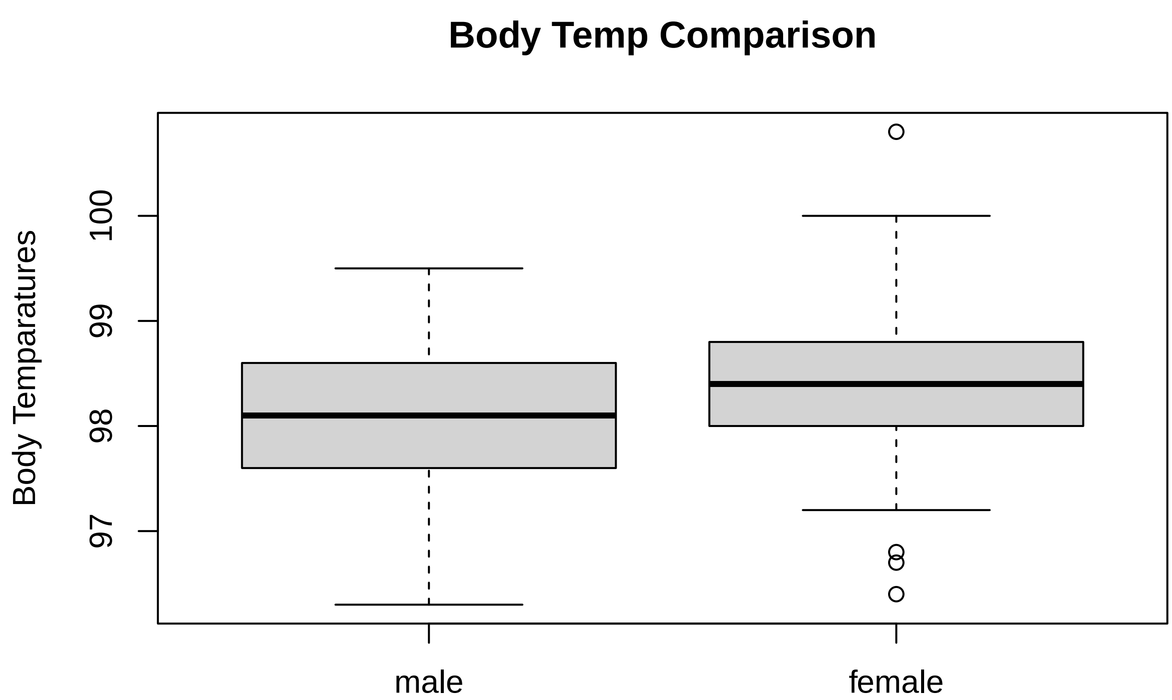
```
temp_heartrate_male_df = extract_rows_based_on_condition(df=bodytemp_heartrate_df,
                                                  column_name="gender",
                                                  condition=1)
temp_heartrate_female_df = extract_rows_based_on_condition(df=bodytemp_heartrate_df,
                                                  column_name="gender",
                                                  condition=2)
```

Exploratory Analysis of the data

Printing summary statistics

Plotiing side by side box plot

```
bodytemp_male = extract_column_func(df=temp_heartrate_male_df,
                                  column_name="body_temperature")
bodytemp_female = extract_column_func(df=temp_heartrate_female_df,
                                  column_name="body_temperature")
boxplot(bodytemp_male, bodytemp_female, main="Body Temp Comparison",
        ylab="Body Temparatures", names=c("male", "female"))
```



Printing Male Body Temparature summary Statistics

```
summary(bodytemp_male)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    96.3    97.6    98.1    98.1    98.6    99.5
```

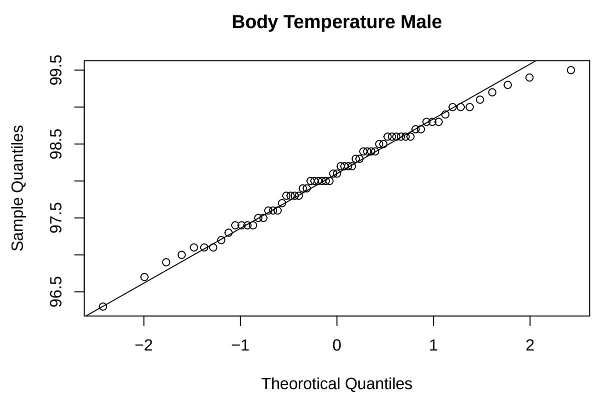
Printing Male Body Temparature summary Statistics

```
summary(bodytemp_female)
```

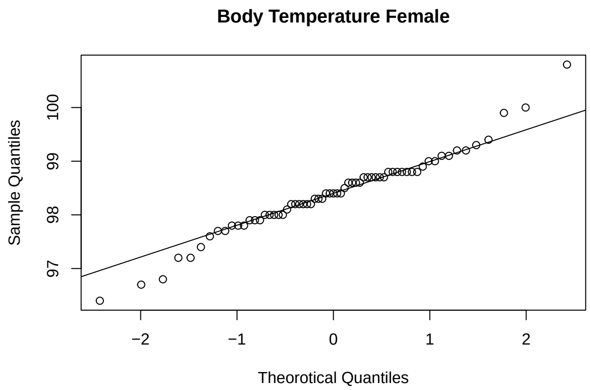
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   96.40   98.00   98.40   98.39   98.80  100.80
```

Plotting QQ Plots

```
qqnorm(bodytemp_male,main="Body Temperature Male",
       xlab="Theorotical Quantiles", ylab="Sample Quantiles")
qqline(bodytemp_male)
```




```
qqnorm(bodytemp_female,main="Body Temperature Female",
       xlab="Theorotical Quantiles", ylab="Sample Quantiles")
qqline(bodytemp_female)
```



From QQ-Plots distributions of male and female body temparatures, we can see that both male and female body tempatures can be assumed to be of Normal Distribution since sample quantiles mostly overlaps with theorotical quantiles in both the cases.

Since, the two samples are independent and has unequal variances and also come from approximate normnal distribution (from QQ-Plots), we can perform two sample T-Test to identify whether two distributions have equal mean.

Null Hypothesis implies that difference between means of male and female body temparatures are zero. Alternate Hypothesis implies that difference between of male and female body temparatures are not zero.

Performing T-Test to understand whether male and

female bodytemparatures are equal

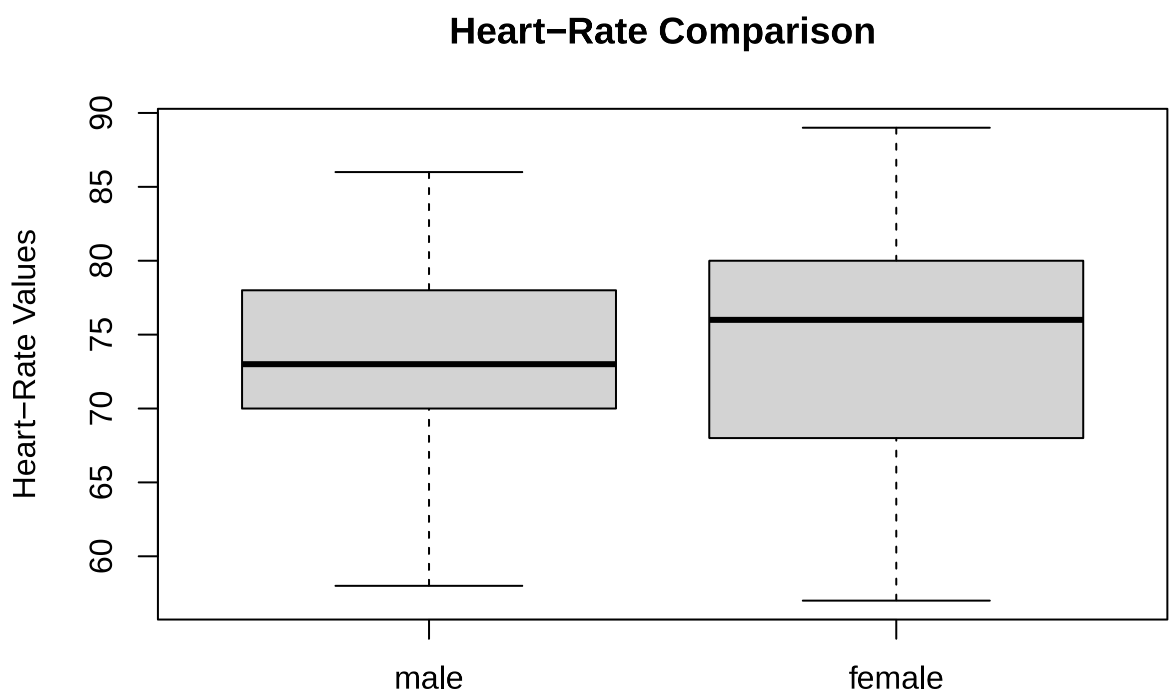
```
t.test(bodytemp_male, bodytemp_female, alternative = "two.sided",
       conf.level=0.95, var.equal=FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  bodytemp_male and bodytemp_female
## t = -2.2854, df = 127.51, p-value = 0.02394
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.53964856 -0.03881298
## sample estimates:
## mean of x mean of y
##  98.10462  98.39385
```

```
We could see that the p-value is less than 0.05 and also 0 does not lie on the
95% confidence interval range. Therefore, we can safely assume that the male and
female body temperatures have different mean values and from the summary
statistics, it is clear that mean of the body temparature of the female is
slightly higher than the male.
```

Solution 1(b)

```
heartrate_male = extract_column_func(df=temp_heartrate_male_df,
                                     column_name="heart_rate")
heartrate_female = extract_column_func(df=temp_heartrate_female_df,
                                       column_name="heart_rate")
boxplot(heartrate_male, heartrate_female, main="Heart-Rate Comparison",
        ylab="Heart-Rate Values", names=c("male", "female"))
```



Printing summary statistics

Printing Male Body Temparature summary Statistics

```
summary(heartrate_male)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   58.00   70.00   73.00   73.37   78.00   86.00
```

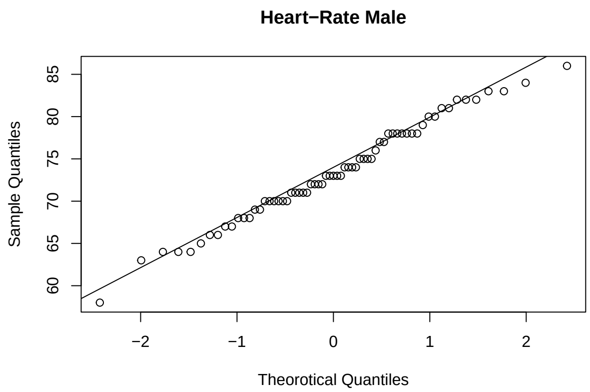
Printing Male Body Temparature summary Statistics

```
summary(heartrate_female)
```

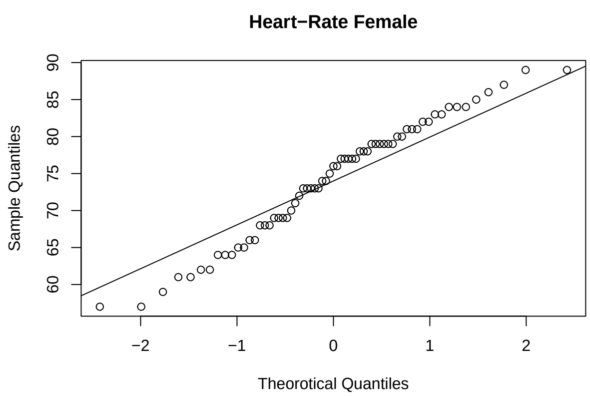
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   57.00   68.00   76.00   74.15   80.00   89.00
```

Plotting QQ Plots

```
qqnorm(heartrate_male,main="Heart-Rate Male",
       xlab="Theorotical Quantiles", ylab="Sample Quantiles")
qqline(heartrate_male)
```




```
qqnorm(heartrate_female,main="Heart-Rate Female",
       xlab="Theorotical Quantiles", ylab="Sample Quantiles")
qqline(heartrate_male)
```



From QQ-Plots distributions of male and female body temparatures, we can see that both male and female heartrates can be assumed to be of Normal Distribution since sample quantiles mostly overlaps with theorotical quantiles in both the cases.

Since, the two samples are independent and has unequal variances and also come from approximate normnal distribution (from QQ-Plots), we can perform two sample T-Test to identify whether two distributions have equal mean.

Null Hypothesis implies that difference between means of male and female heartrates are zero. (i.e. Male and female feartrates are same). Alternate Hypothesis implies that difference between of male and female body temparatures are not zero. (i.e. Male and Female heartrates are not same).

Performing T-Test to understand whether male and

female bodytemparatures are equal

```
t.test(heartrate_male, heartrate_female, alternative = "two.sided",
       conf.level=0.95, var.equal=FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  heartrate_male and heartrate_female
## t = -0.63191, df = 116.7, p-value = 0.5287
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.243732  1.674501
## sample estimates:
## mean of x mean of y
##  73.36923  74.15385
```

```
We could see that the p-value is greater than 0.05 and also 0 lies on the
95% confidence interval range. Therefore, we can safely assume that the male and
female heartrate values have the same mean value.
```

Solution 1 (c)

Plotting scatter-plots with regression line to identify linear relationship

```
Checking if there is relationship between body temperatures and heart-rate
irrespective of gender (overall data)
```

```
heartrate_values = extract_column_func(df=bodytemp_heartrate_df,
                                       column_name='heart_rate')
bodytemp_values = extract_column_func(df=bodytemp_heartrate_df,
                                      column_name='body_temperature')
```

The positive slope of the regression line suggests that there is a positive linear association between heart rate and body temperature. Further, from scatter plot, we can see that linear relationship is weak.

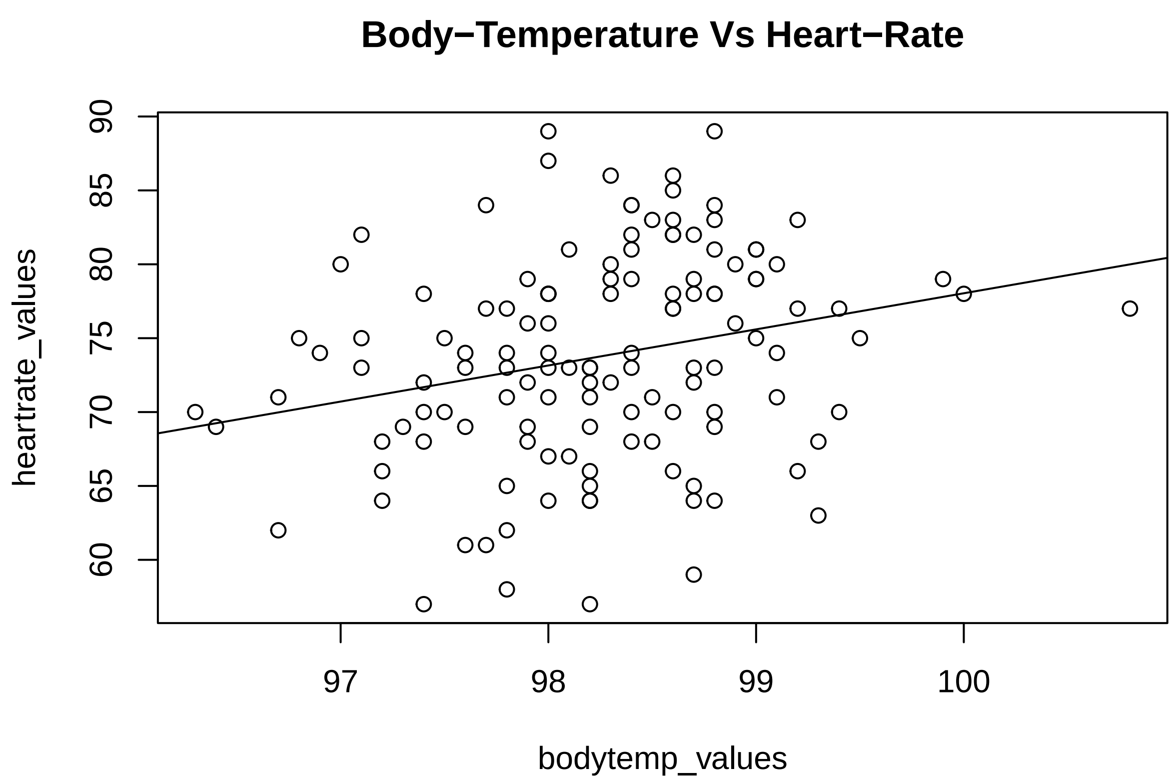
Correlation Value between Body Temperature and Heartbeat of the overall data

```
cor(bodytemp_values, heartrate_values)
```

```
## [1] 0.2536564
```

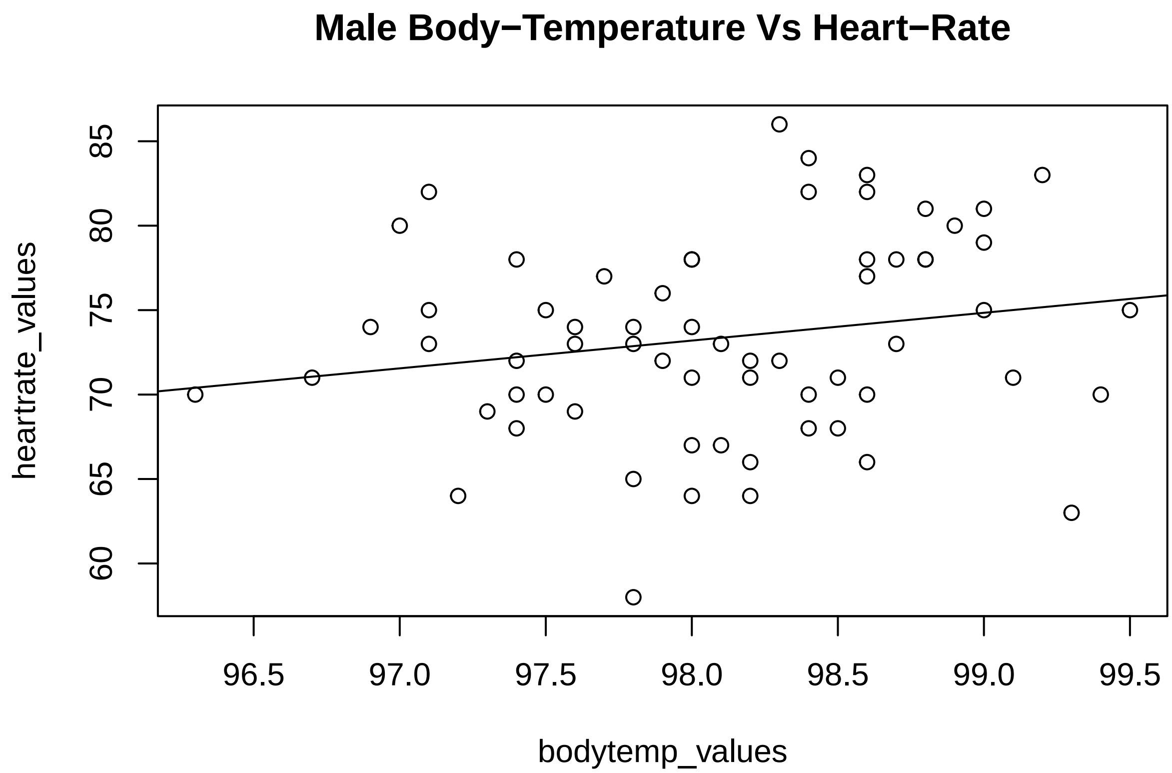
```
Checking if there is relationship between body temperatures and heart-rate
for male gender
```

```
plot(bodytemp_values, heartrate_values, xlab = "bodytemp_values",
     ylab = "heartrate_values",  main="Body-Temperature Vs Heart-Rate")
abline(lm(heartrate_values ~ bodytemp_values))
```



The positive slope of the regression line suggests that there is a positive linear association between heart rate and body temperature. Further, from scatter plot, we can see that linear relationship is weak.

```
plot(bodytemp_male, heartrate_male, xlab = "bodytemp_values",
     ylab = "heartrate_values",  main="Male Body-Temperature Vs Heart-Rate")
abline(lm(heartrate_male ~ bodytemp_male))
```



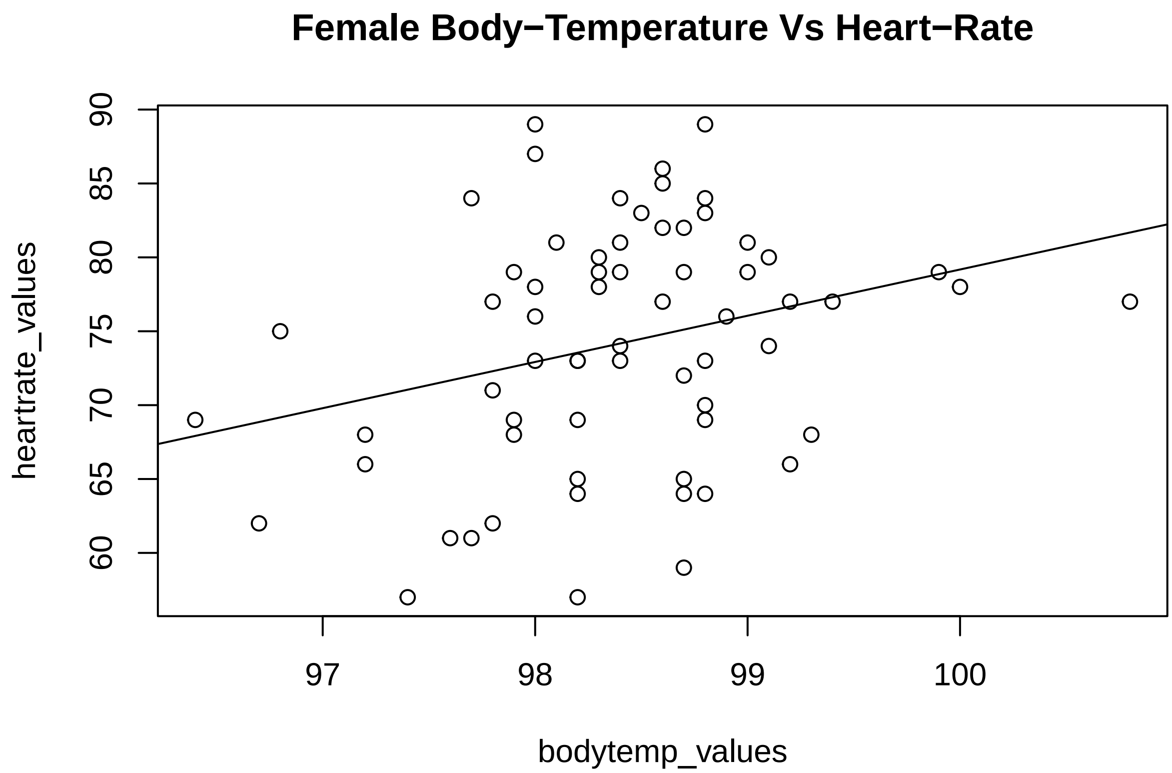
Correlation Value between Body Temperature and Heartbeat for the male gender

```
cor(bodytemp_male, heartrate_male)
```

```
## [1] 0.1955894
```

```
Checking if there is relationship between body temperatures and heart-rate
for male gender
```

```
plot(bodytemp_female, heartrate_female, xlab = "bodytemp_values",
     ylab = "heartrate_values",  main="Female Body-Temperature Vs Heart-Rate")
abline(lm(heartrate_female ~ bodytemp_female))
```



The positive slope of the regression line suggests that there is a positive linear association between heart rate and body temperature. Further, from scatter plot, we can see that linear relationship is weak.

Correlation Value between Body Temperature and Heartbeat for the female gender

```
cor(bodytemp_female, heartrate_female)
```

```
## [1] 0.2869312
```

From the correlation between body temperature and heart-beat for the overall data, for male and female genders alone, we could see that, there is higher correlation between body temperature and heartbeat for female gender when compared to correlation for the overall data which inturn is higher when compared to correlation for male gender.

Therefore, we can infer that there is a general positive correlation between body temperature and heartbeat and the correlation is greater for female gender when compared to male gender.

Solution 2(a)

```
u_star_computation = function(lambda, n){
  exponential_random_variables = rexp(n, rate = lambda)
  return (mean(exponential_random_variables))
}
```

```
generate_exponential_random_variables = function(lambda, n){
  percentile_bootstrap_coverage_prob = c()
  z_distribution_coverage_prob = c()
  exponential_random_variables = rexp(n, rate = lambda)
  star_lambda = 1/mean(exponential_random_variables)
  z_distribution_lower_bound = (mean(exponential_random_variables)
  - (qnorm(0.975) * sd(exponential_random_variables) / sqrt(n)))
  z_distribution_upper_bound = (mean(exponential_random_variables)
  + (qnorm(0.975) * sd(exponential_random_variables) / sqrt(n)))
  exponential_mean = (1 / lambda)
  if(z_distribution_upper_bound >= exponential_mean
     & z_distribution_lower_bound <= exponential_mean)
  {
    z_distribution_coverage_prob = 1
  }
  else
  {
    z_distribution_coverage_prob = 0
  }

  bootstrap_samples = replicate(1000, u_star_computation(lambda=star_lambda, n=n))
  sorted_list = sort(bootstrap_samples)
  c_limits = c(sorted_list[25], sorted_list[975])
  if(c_limits[2] >= exponential_mean & c_limits[1] <= exponential_mean)
  {
    percentile_bootstrap_coverage_prob = 1
  }
  else
  {
    percentile_bootstrap_coverage_prob = 0
  }
  return (c(z_distribution_coverage_prob, percentile_bootstrap_coverage_prob))
}
```

```
compute_coverage_probability = function(lambda, n){
  number_of_replications = 5000
  z_coverage_probs = c()
  percentile_bootstrap_coverage_probs = c()
  for(i in 1:number_of_replications)
  {
    estimated_coverage_possibilities = generate_exponential_random_variables(
      lambda=lambda, n=n)
    z_coverage_probs = append(z_coverage_probs,
                              estimated_coverage_possibilities[1])
    percentile_bootstrap_coverage_probs = append(
      percentile_bootstrap_coverage_probs, estimated_coverage_possibilities[2])
  }
  z_coverage_probability = mean(z_coverage_probs)
```

```
  percentile_bootstrap_coverage_probability = mean(
    percentile_bootstrap_coverage_probs)
  return (c(z_coverage_probability, percentile_bootstrap_coverage_probability))
}
```

```
lambda = 0.01
n = 5
coverage_probs = compute_coverage_probability(lambda=lambda, n=n)
z_coverage_probability = coverage_probs[1]
percentile_bootstrap_coverage_probability = coverage_probs[2]
```

```
paste("Large Sample Z-Interval Coverage Probability:", z_coverage_probability)
```

```
## [1] "Large Sample Z-Interval Coverage Probability: 0.8116"
```

```
paste("Parametric Bootstrap Coverage Probability:",
      percentile_bootstrap_coverage_probability)
```

```
## [1] "Parametric Bootstrap Coverage Probability: 0.9016"
```

Solution 2 (b)

Repating for 16 combinations mentioned

```
lambda_list = c(0.01, 0.1, 1, 10)
n_list = c(5, 10, 30, 100)

summary_df = data.frame(matrix(ncol = 5, nrow = 0))
x = c("Index", "lambda", "n", "Large Sample Z-Interval Coverage Probability",
      "Parametric Bootstrap Coverage Probability")
colnames(summary_df) = x

z_prob_matrix = matrix(nrow=4,ncol=4)
percentile_bootstrap_prob_matrix = matrix(nrow=4,ncol=4)
index_counter = 1
for(i in 1:length(lambda_list))
{
  for(j in 1:length(n_list))
  {
    coverage_probs = compute_coverage_probability(lambda=lambda_list[i],
                                                  n=n_list[j])
    z_coverage_probability = coverage_probs[1]
    percentile_bootstrap_coverage_probability = coverage_probs[2]
    summary_df[nrow(summary_df) + 1,] = c(index_counter,
                          lambda_list[i], n_list[j],
                          z_coverage_probability,
                          percentile_bootstrap_coverage_probability)
```

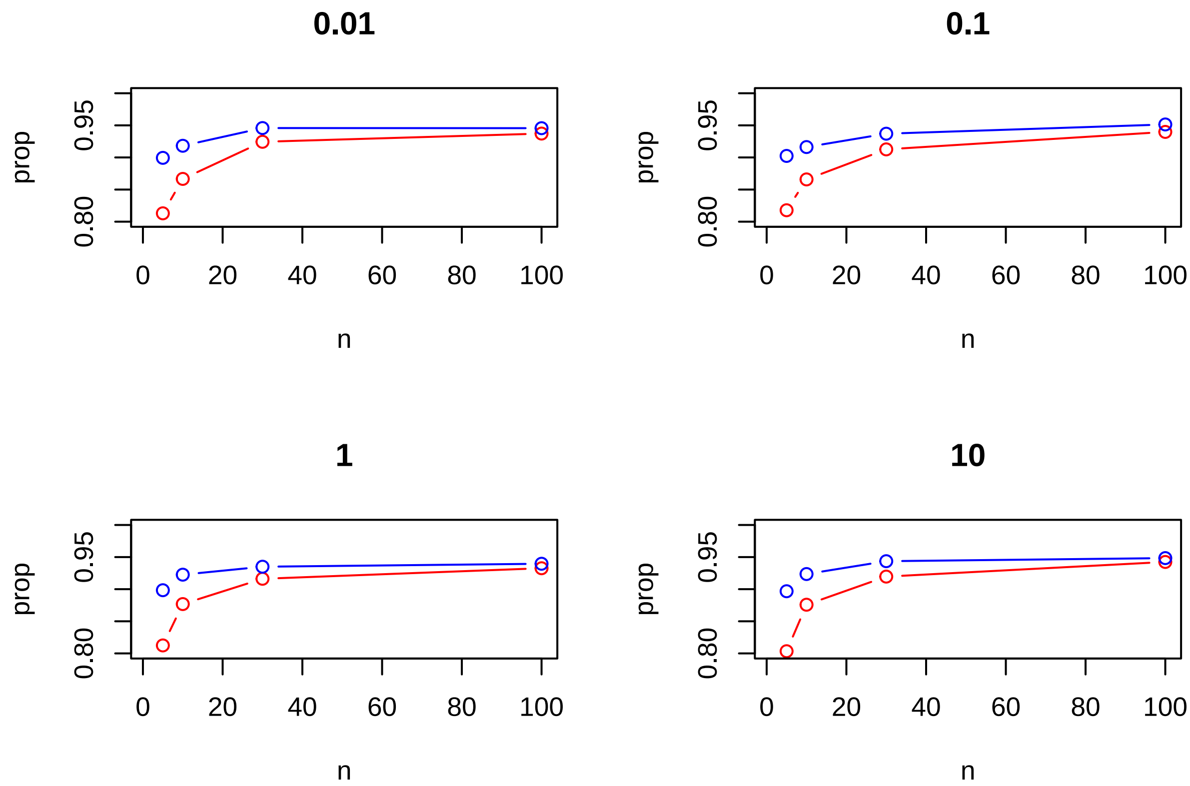
```
    z_prob_matrix[i,j] = z_coverage_probability
    percentile_bootstrap_prob_matrix[i,j] =
      percentile_bootstrap_coverage_probability
    index_counter = index_counter + 1
  }
}
kable(summary_df, caption="Coverage Probability Summary Table")
```

Table 1: Coverage Probability Summary Table

Index	lambda	n	Large Sample Z-Interval Coverage Probability	Parametric Bootstrap Coverage Probability
1	0.01	5	0.8130	0.8994
2	0.01	10	0.8666	0.9182
3	0.01	30	0.9244	0.9458
4	0.01	100	0.9372	0.9456
5	0.10	5	0.8178	0.9024
6	0.10	10	0.8658	0.9162
7	0.10	30	0.9126	0.9370
8	0.10	100	0.9398	0.9514
9	1.00	5	0.8124	0.8984
10	1.00	10	0.8768	0.9226
11	1.00	30	0.9162	0.9350
12	1.00	100	0.9326	0.9396
13	10.00	5	0.8034	0.8968
14	10.00	10	0.8758	0.9236
15	10.00	30	0.9196	0.9436
16	10.00	100	0.9424	0.9484

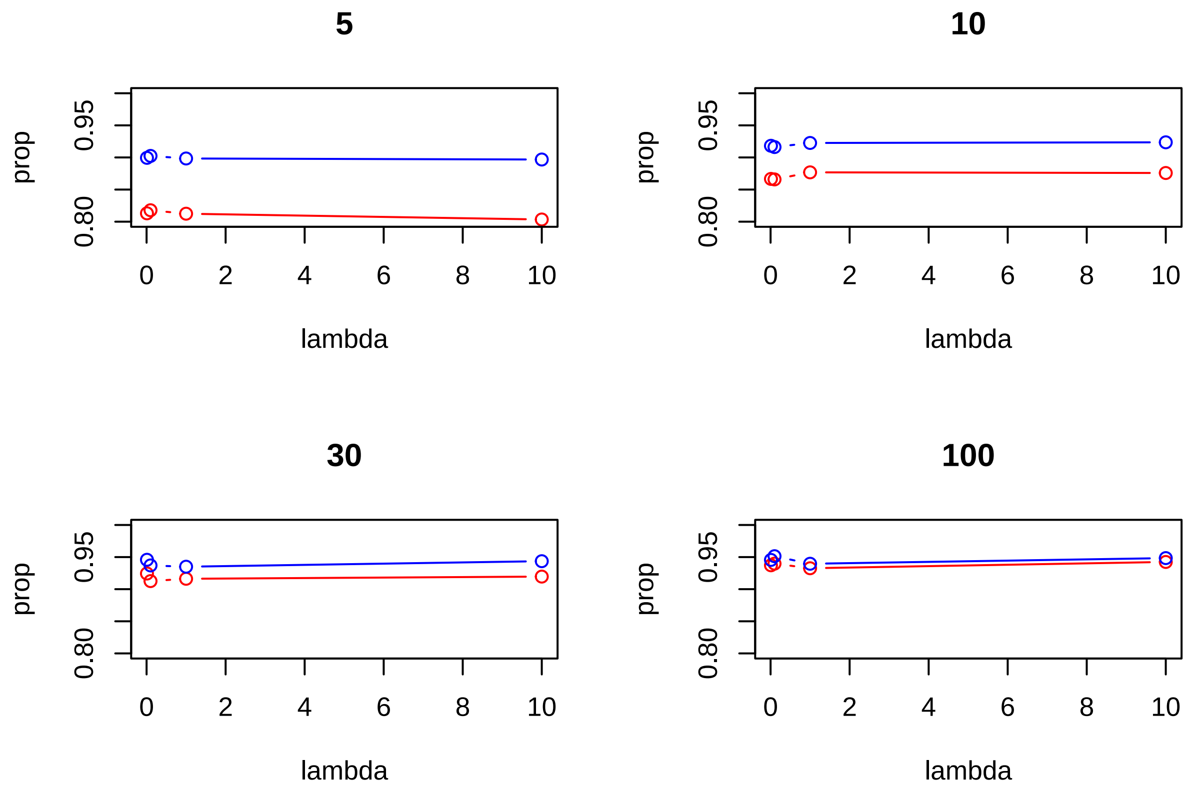
Plotting for constant lambda values

```
j = c(1,2,3,4)
par(mfrow=c(2,2))
for (i in j){
  plot(n_list,z_prob_matrix[i,], main=lambda_list[i], xlab = "n", ylab = "prop",
       col = 'red', type= 'b', xlim = c(1,100), ylim = c(0.8,1))
  lines(n_list,percentile_bootstrap_prob_matrix[i,], col='blue',type='b')
}
```



Plotting for constant n values

```
x = c(1,2,3,4)
par(mfrow=c(2,2))
for (i in x){
  plot(lambda_list,z_prob_matrix[,i], main=n_list[i], xlab = "lambda",
       ylab = "prop", col = 'red', type= 'b', xlim = c(0.01,10), ylim = c(0.8,1))
  lines(lambda_list,percentile_bootstrap_prob_matrix[,i], col='blue',type='b')
}
```



Solution 2 (c)

For large samples, from summary tables, we could see that, the coverage probabilities for Large Sample Z interval is same as the coverage probabilities from Parametric Bootstrap Perentile method when n = 100 (i.e. when n is large).

For Parametric Bootstrap Percentile method, from n = 30, the coverage probabilities are higher.

From 2nd set of graphs (plots for constant n values) we can see that there is not much change in coverage probability.Therefore, the coverage probability doesn't depend on lambda.

Taking, into account all possible combinations in Coverage Probability Summary Table, coverage probability values for parametric bootstrap percentile method is hgher than that of Coverage Probability for Large Sample Z-Test method. Therefore, Parametric Bootstrap Percentile method is recommended.

Solution 2 (d)

The conclusions in 2(c) is not specific to particular values of lambda fixed in advance.

It holds generically for all values of lambda as we saw in "Coverage Probability Summary Table" and graphs.