Lab2 md

2023-11-16

Task 0

```
setwd("/Users/viktorsjoberg/Desktop/high-dimensional/Assignment 2")
BG=read.table("BankGenuine.txt")
library(gt)
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(patchwork)
library(ellipse)
```

```
##
## Attaching package: 'ellipse'
```

```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
library(scales)
library(webshot)
```

```
x1_bar <- mean(BG$V1)
x2_bar <- mean(BG$V2)
```

```r
x3_bar <- mean(BG$V3)
x4_bar <- mean(BG$V4)
x5_bar <- mean(BG$V5)
x6_bar <- mean(BG$V6)

x1_s <- var(BG$V1)
x2_s <- var(BG$V2)
x3_s <- var(BG$V3)
x4_s <- var(BG$V4)
x5_s <- var(BG$V5)
x6_s <- var(BG$V6)

data <- data.frame(
  Variable = c("Length", "Height L", "Height R", "Distance inner to lower", "Distance inner to upper", "
  Mean = c(x1_bar, x2_bar, x3_bar, x4_bar, x5_bar, x6_bar),
  Variance = c(x1_s, x2_s, x3_s, x4_s, x5_s, x6_s)
)

gt_table <- gt(data)
# print(gt_table)
gt_output <- "gt img/gt_table.png"
gtsave(gt_table, gt_output)
```

```
## Registered S3 method overwritten by 'webshot2':
##   method        from
##   print.webshot webshot
```

```r
temp_data <- BG %>%
  rename(
    Length = V1,
    Left = V2,
    Right = V3,
    Lower = V4,
    Upper = V5,
    Diagonal = V6
  )
```

Variable	Mean	Variance
Length	214.969	0.1502414
Height L	129.943	0.1325768
Height R	129.720	0.1262626
Distance inner to lower	8.305	0.4132071
Distance inner to upper	10.168	0.4211879
Length of diagonal	141.517	0.1998091

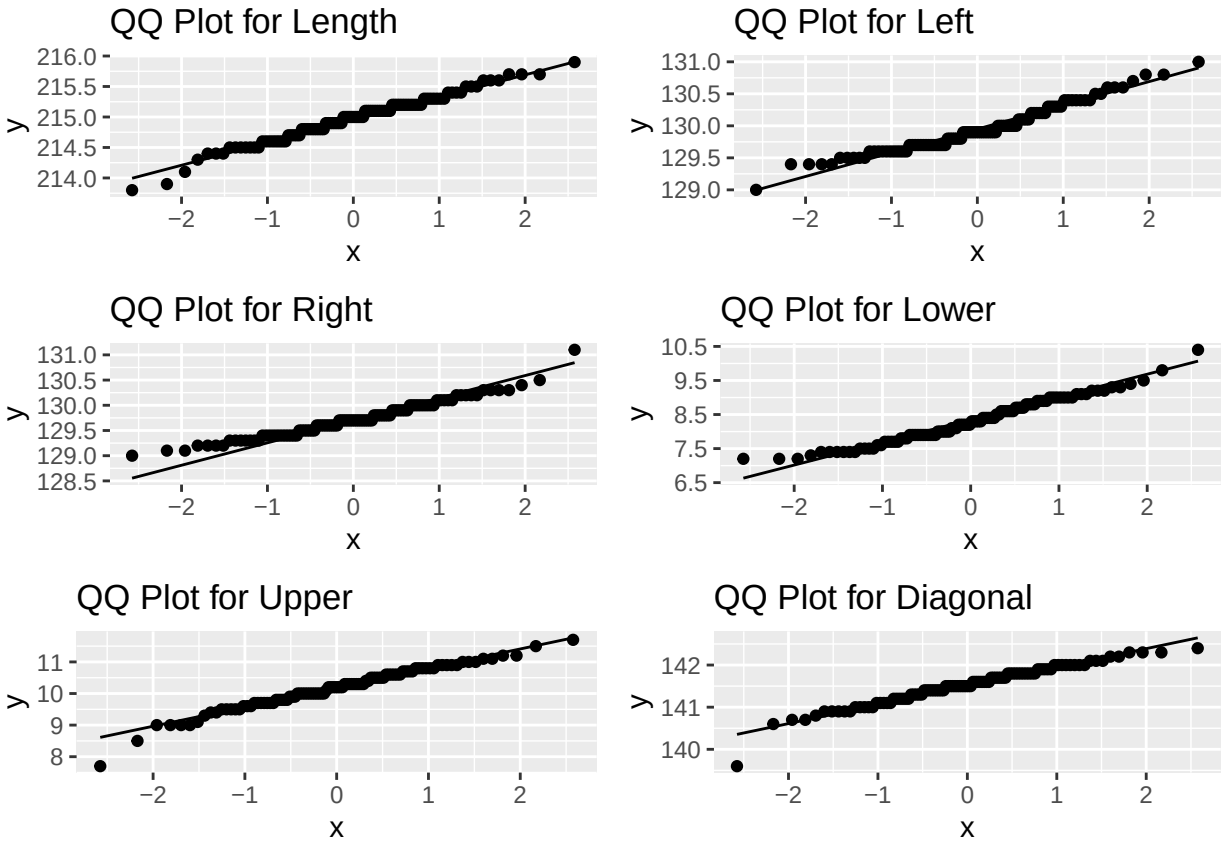
Task 1 Printing Bank Notes

1.1

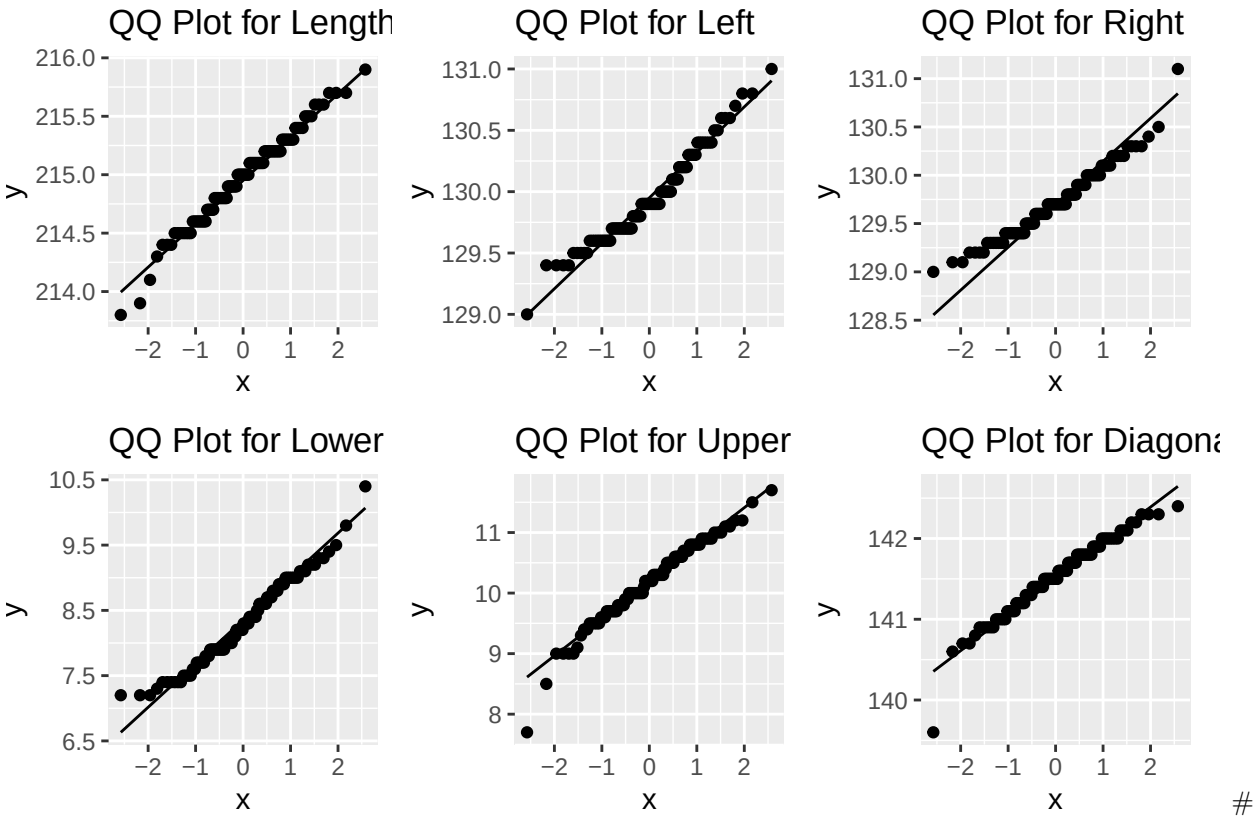
```
qq_plot <- function(variable, data) {
  ggplot(data, aes_string(sample = variable)) +
    stat_qq() +
    stat_qq_line() +
    ggtitle(paste("QQ Plot for", variable))
}

plot_list <- lapply(names(temp_data), function(var) qq_plot(var, temp_data))
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
grid.arrange(grobs = plot_list, ncol = 2, nrow = 3)
```

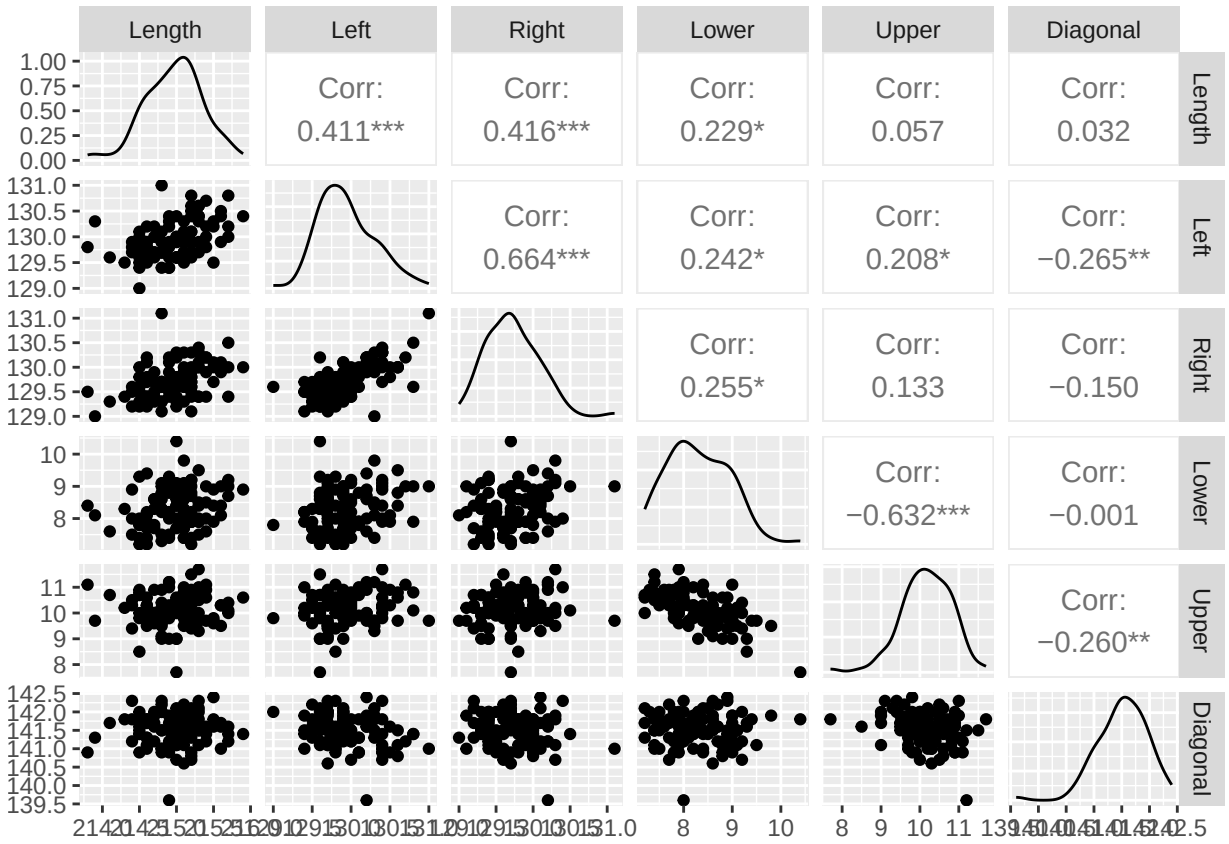



```
plot_layout <- plot_list[[1]] + plot_list[[2]] + plot_list[[3]] +
               plot_list[[4]] + plot_list[[5]] + plot_list[[6]]
plot_layout + plot_layout(3, 2)
```

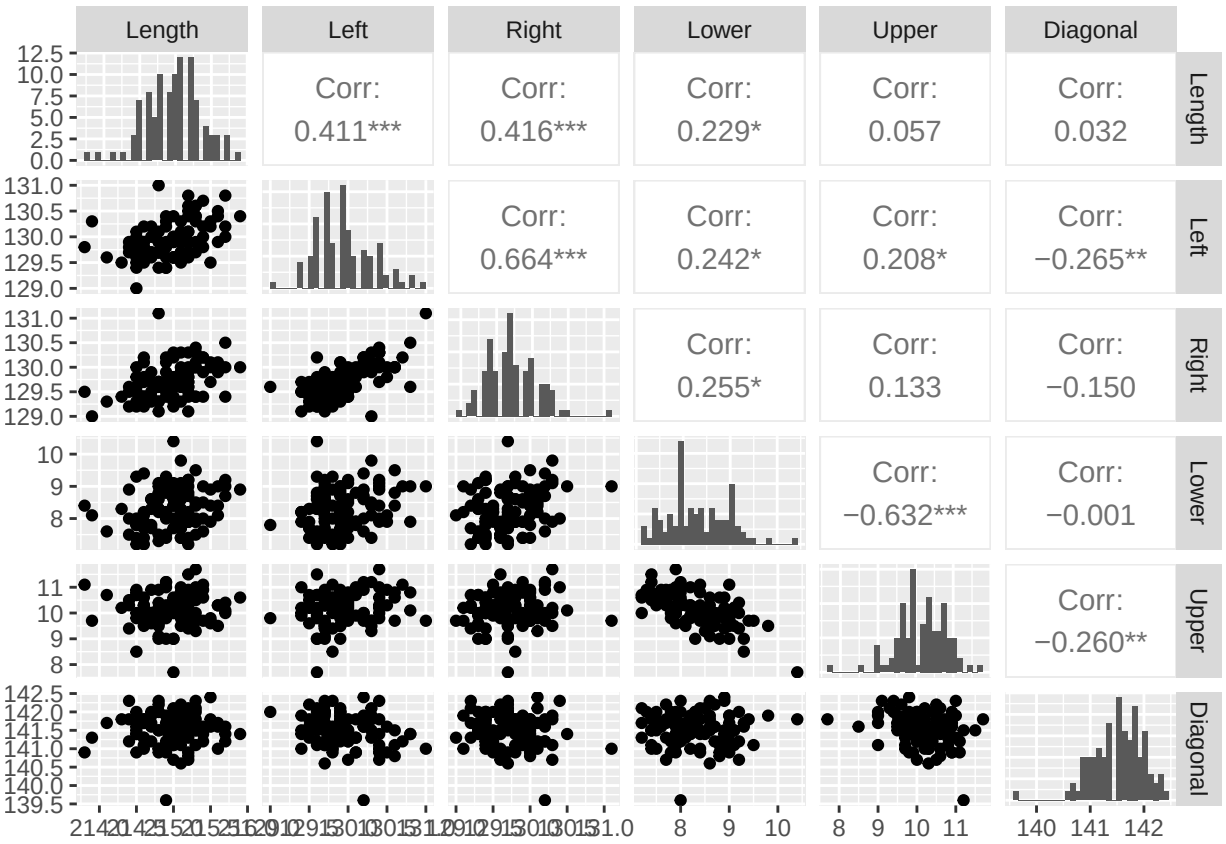


1.2

```
cov_matrix <- cov(temp_data)
cov_df <- as.data.frame(cov_matrix)
cov_df <- cbind(Variable = rownames(cov_df), cov_df)
rownames(cov_df) <- NULL
cov_matrix_table <- gt(cov_df)
for (i in 1:ncol(cov_df) - 1) {  # -1 because the first column is now 'Variable'
  column_name <- names(cov_df)[i + 1]  # +1 to skip the 'Variable' column
  cov_matrix_table <- cov_matrix_table %>%
    tab_style(
      style = cell_text(color = "red"),
      locations = cells_body(
        columns = column_name,
        rows = Variable == column_name
      )
    )
}


ggpairs(temp_data)
```




```
custom_diag <- function(data, mapping, ...) {
  p <- ggplot(data = data, mapping = mapping) +
  geom_histogram(...)
}

ggpairs(temp_data, diag = list(continuous = wrap(custom_diag, bins = 30)))
```




```
# print(cov_matrix_table)

gt_output2 <- "gt img/cov_matrix_table.png"
gtsave(cov_matrix_table, gt_output2)
```

Variable	Length	Left	Right	Lower	Upper	Diagonal
Length	0.150241414	0.05801313	0.05729293	0.0571262626	0.01445253	0.0054818182
Left	0.058013131	0.13257677	0.08589899	0.0566515152	0.04906667	-0.0430616162
Right	0.057292929	0.08589899	0.12626263	0.0581818182	0.03064646	-0.0237777778
Lower	0.057126263	0.05665152	0.05818182	0.4132070707	-0.26347475	-0.0001868687
Upper	0.014452525	0.04906667	0.03064646	-0.2634747475	0.42118788	-0.0753090909
Diagonal	0.005481818	-0.04306162	-0.02377778	-0.0001868687	-0.07530909	0.1998090909

1.3

```
BG=read.table("BankGenuine.txt")
n=dim(BG)[1]
p=dim(BG)[2]
S=cov(BG)
IS=solve(S)
barX=colMeans(BG)
```

```r
a=(n-1)*p/(n-p)*qf(0.95,p,n-p)
q=3
mu0=c(210,130,130,10,10,140)
t2=n*(barX-mu0)%*%IS%*%(barX-mu0)
cx=as.matrix(barX-mu0)
T2=n*t(cx)%*%IS%*%cx

T2
```

```
##          [,1]
## [1,] 24823.14
```

```r
t2
```

```
##          [,1]
## [1,] 24823.14
```

The fact that both t2 and T2 returns the same result demonstrates that R is treating a one-column matrix and a vector the same.

```r
data2 <- data.frame(
  Row = c(2, 3, 4, 5, 6, 7, 8, 9, "10 & 12"),
   Explanation = c(
    "n - Count observations",
    "p - Count Variables",
    "S - Variance-Covariance matrix",
    "IS - Inverse Variance-Covariance matrix",
    "Mean Calculation per variable",
    "Critical value with alpha = 0.05. (Of F-distribution)",
    "q = ?",
    "m0 - True value under h0.",
    "Hotelling t-squared statistics for a single variable"
  )
)
gt_table2 <- gt(data2)

# print(gt_table2)

gt_output3 <- "gt img/gt_table2.png"
gtsave(gt_table2, gt_output3)
```

Row	Explanation
2	n - Count observations
3	p - Count Variables
4	S - Variance-Covariance matrix
5	IS - Inverse Variance-Covariance matrix
6	Mean Calculation per variable
7	Critical value with alpha = 0.05. (Of F-distribution)
8	q = ?
9	m0 - True value under h0.
10 & 12	Hotelling t-squared statistics for a single variable

1.4 - 1.6

```
InHotCR=function(mu0,barX,IS,n){
  mu0=as.vector(mu0)   #mu0 in the code below
  barX=as.vector(barX) # must be a vector not a matrix!
  p=dim(IS)[1]
  a=(n-1)*p/(n-p)*qf(0.95,p,n-p)
  cx=as.matrix(barX-mu0)
  T2=n*t(cx)%*%IS%*%cx
  InHotCR=T2<a
  InHotCR}

m0=c(214.97,  130, 129.67,    8.3, 10.16,   141.52)
InHotCR(m0,barX,IS,n)
```

```
##       [,1]
## [1,] FALSE
```

```
InBonCR=function(mu0,barX,S,n){
  mu0=as.vector(mu0)  #mu0 must be a vector not a matrix!
  barX=as.vector(barX)
```

```
  p=dim(S)[1]
  alpha=0.05/p
  a=qf(1-alpha,1,n-1)*diag(S)/n
  cx=as.matrix(barX-mu0)
  T2=cx^2
  T2<a
  InBonCR=T2<a
  InBonCR
}

prod(InBonCR(m0,barX,S,n))==1
```

```
## [1] TRUE
```

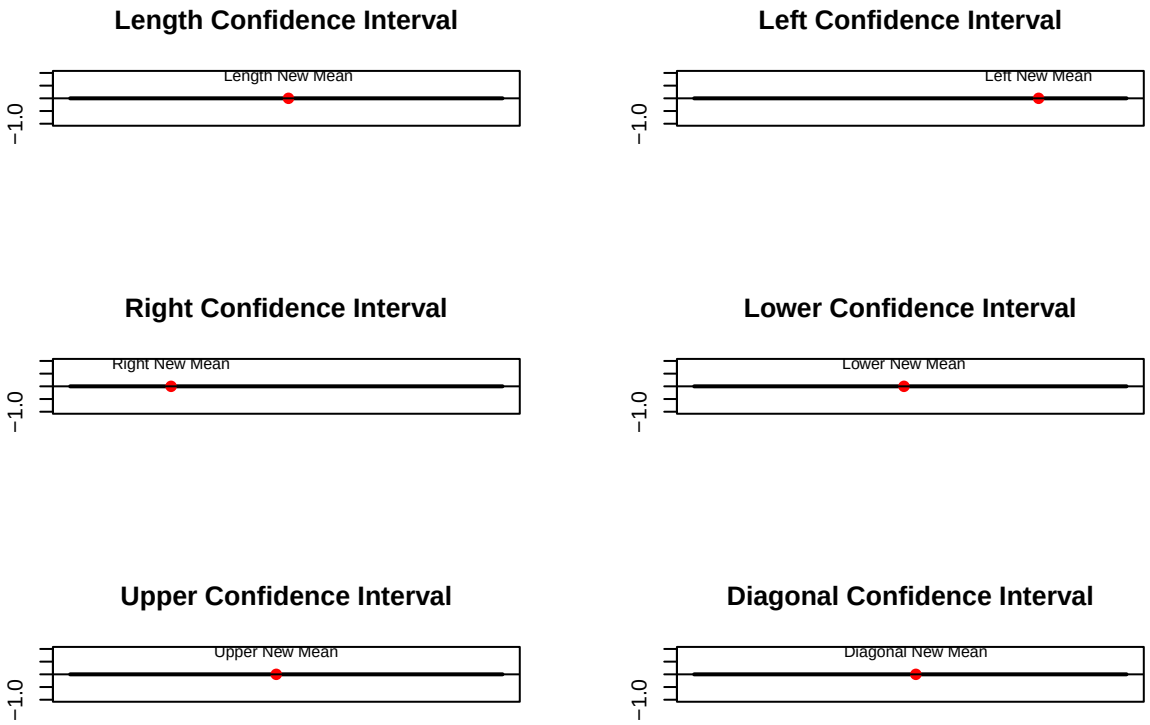
The R code you've provided defines two functions, InHotCR and InBonCR, which are used for multivariate hypothesis testing to determine if a sample mean vector is significantly different from a hypothesized mean vector (mu0). The functions also test if the sample mean vector lies within a certain confidence region.

InHotCR function suggests that when considering all variables together, the difference between the sample means and the hypothesized means is significant. However, the InBonCR function suggests that when examining each variable individually and adjusting for multiple comparisons, there is no significant difference between the sample mean(s) and the hypothesized mean(s).

The second function also takes in m0 which is the mean for the new production line and it test if this dataset reject or does not reject that the m0 is the same for our dataset using the Bonferoni method.

1.7

```
alpha <- 0.05
p <- length(barX)
bonferroni_alpha <- alpha / p
z <- qnorm(1 - bonferroni_alpha / 2)

se <- sqrt(diag(S) / n)

ci_lower <- barX - z * se
ci_upper <- barX + z * se

par(mfrow = c(3, 2))
variable_names <- c("Length", "Left", "Right", "Lower", "Upper", "Diagonal")

for (i in 1:p) {
  plot(c(ci_lower[i], ci_upper[i]), c(0, 0), type = "n", xlab = "", ylab = "", xaxt = "n",
       main = paste(variable_names[i], "Confidence Interval"), ylim = c(-1, 1))
  segments(ci_lower[i], 0, ci_upper[i], 0, lwd = 2)
  points(m0[i], 0, pch = 19, col = "red")
  abline(h = 0)
  text(m0[i], 0.2, labels = paste(variable_names[i], "New Mean"), pos = 3, cex = 0.8)
}
```

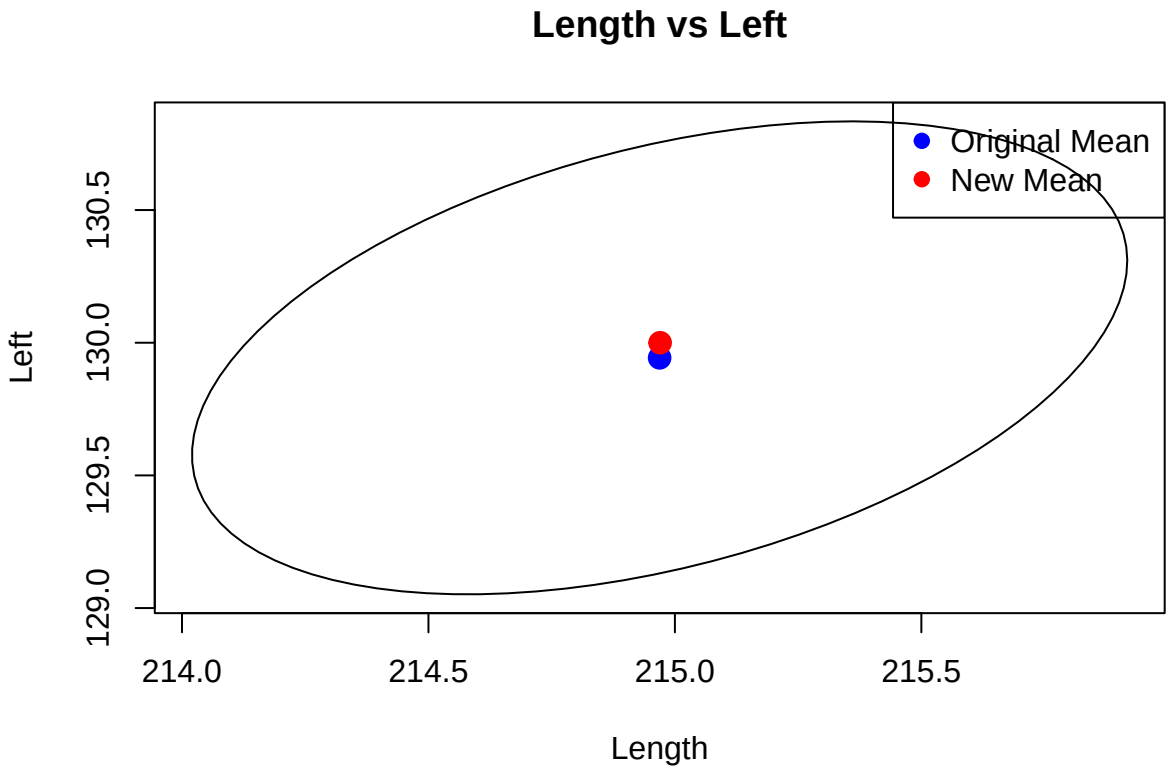


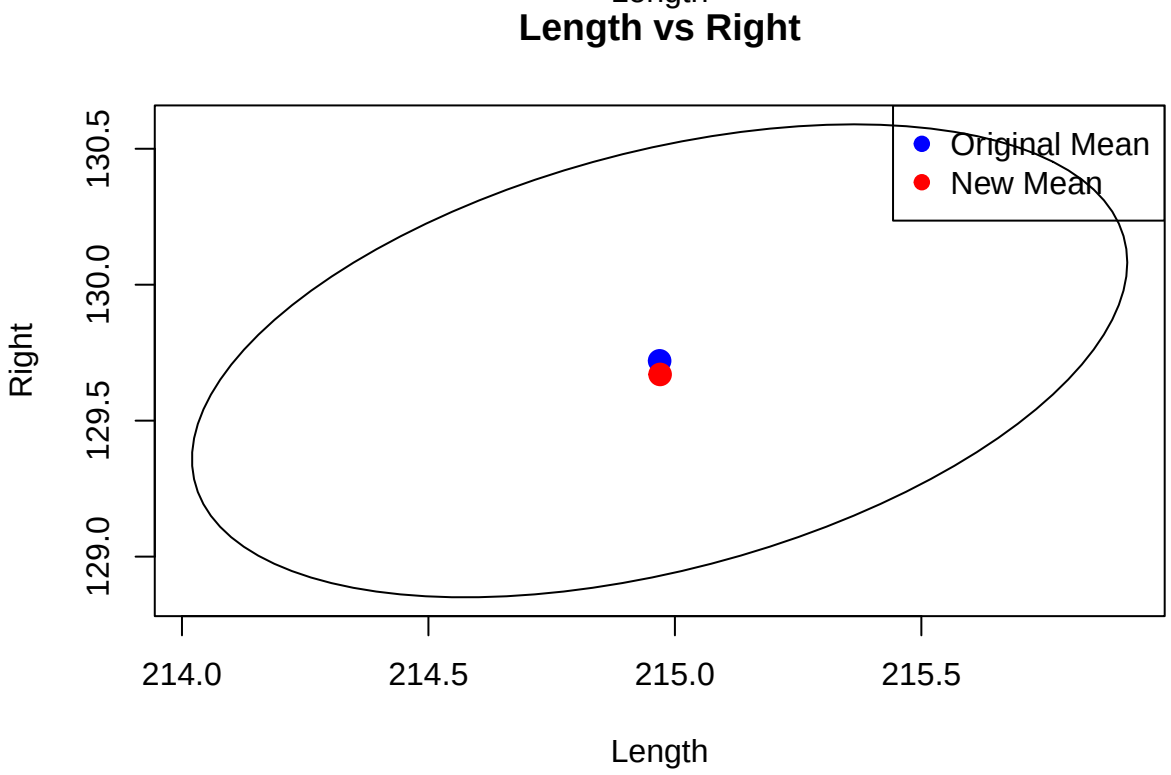

```
par(mfrow = c(1, 1))
```

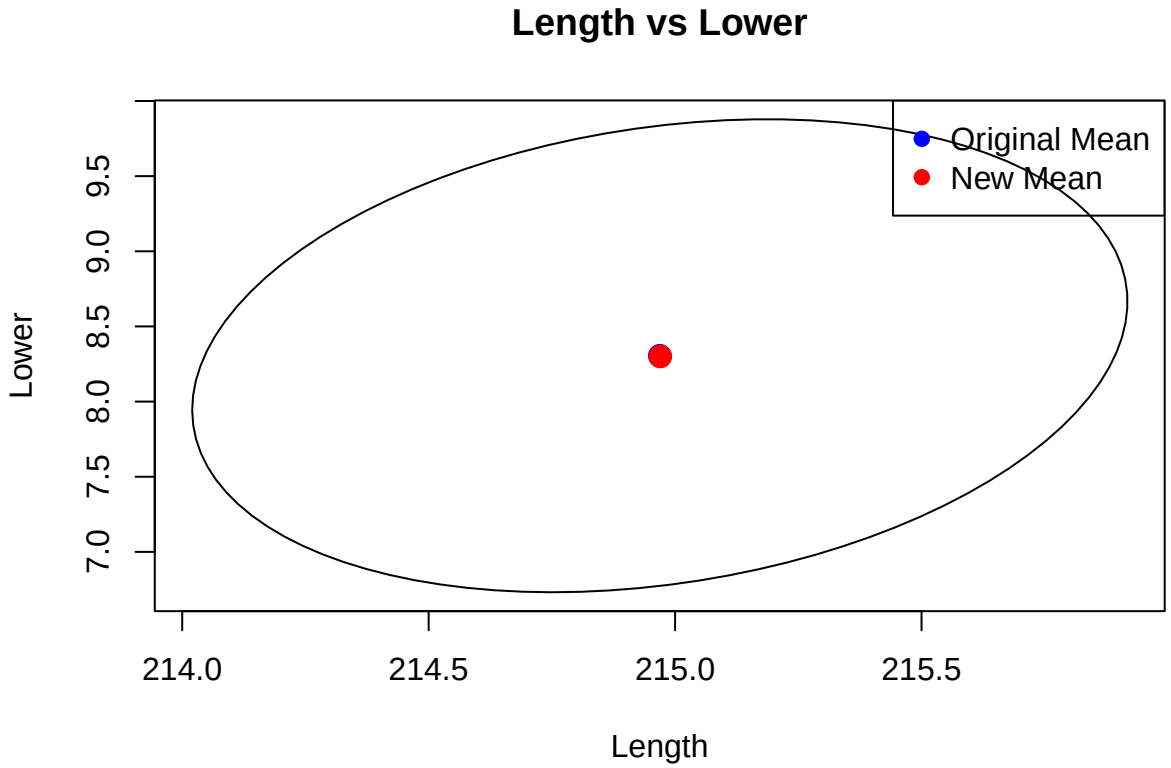
If the red dots are within the black lines in all the plots, it indicates that the means of the new production line for all dimensions fall within the confidence intervals calculated from the original sample. This would suggest that the new production line is producing bank notes with dimensions that are statistically comparable to those of the original sample.

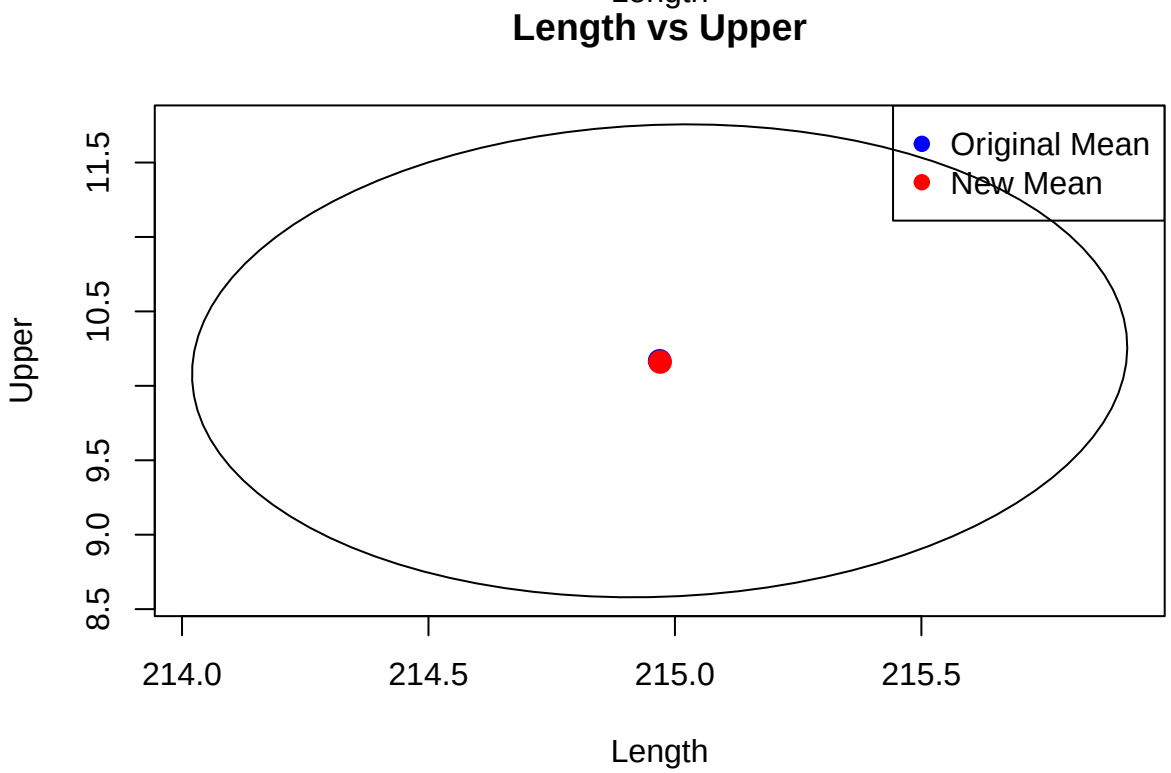
1.8

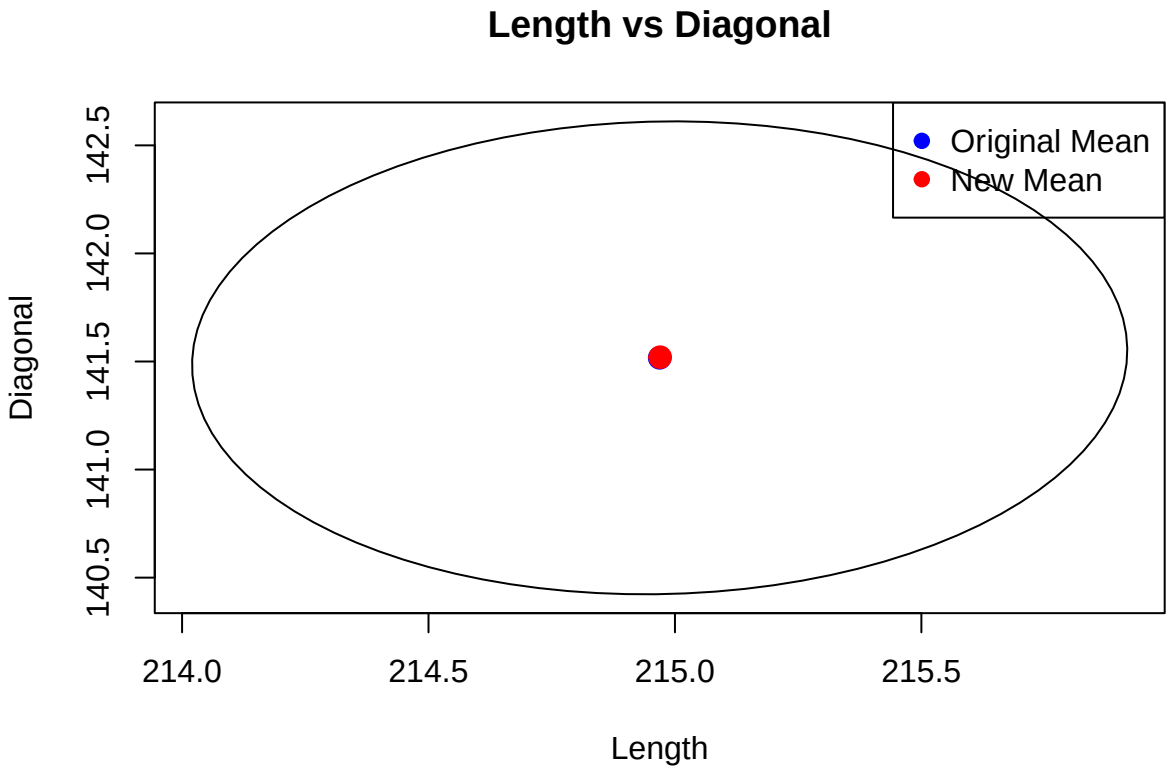
```
conf_level <- 0.95

for (i in 1:(length(barX) - 1)) {
  for (j in (i + 1):length(barX)) {
    ellipse_data <- ellipse(S[c(i, j), c(i, j)], centre = barX[c(i, j)], level = conf_level)

    plot(ellipse_data, type = 'l', xlab = variable_names[i], ylab = variable_names[j],
         main = paste(variable_names[i], "vs", variable_names[j]))

    points(barX[i], barX[j], pch = 19, col = "blue", cex = 1.5)
    points(m0[i], m0[j], pch = 19, col = "red", cex = 1.5)

    legend("topright", legend = c("Original Mean", "New Mean"), pch = 19, col = c("blue", "red"))
  }
}
```

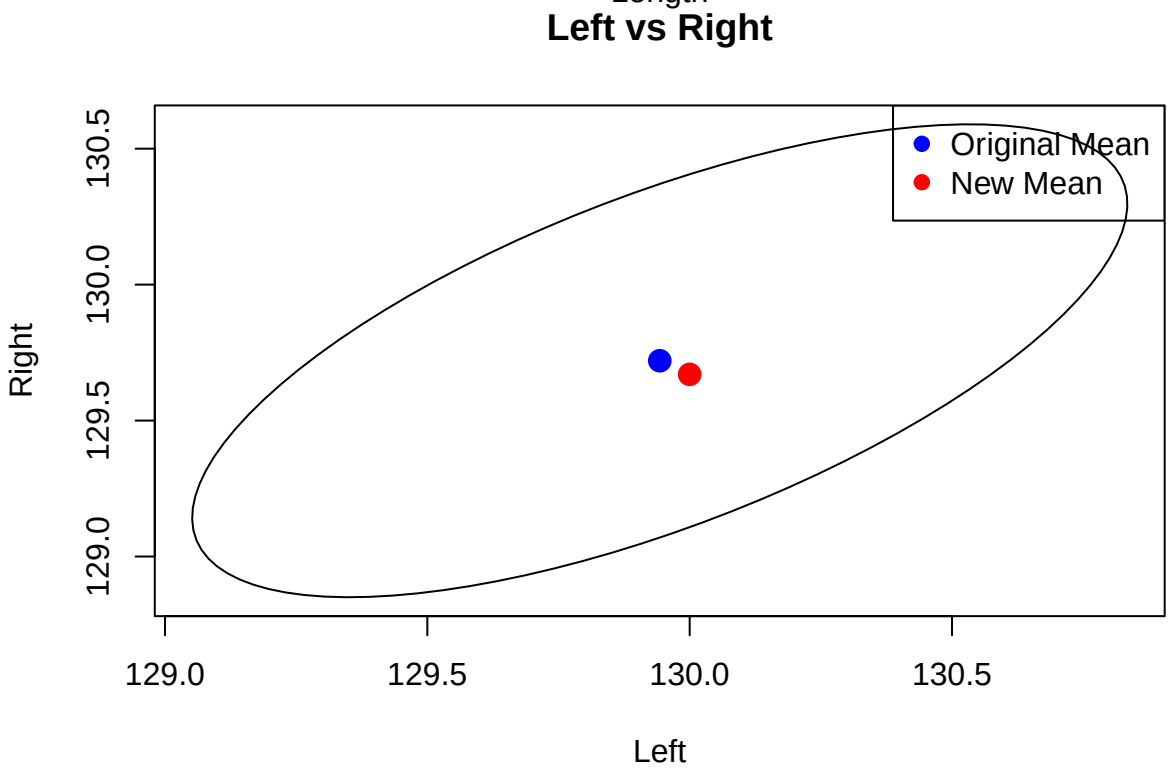

Length vs Left
Original Mean
New Mean
Left
130.5
130.0
129.5
129.0
214.0
214.5
215.0
215.5
Length

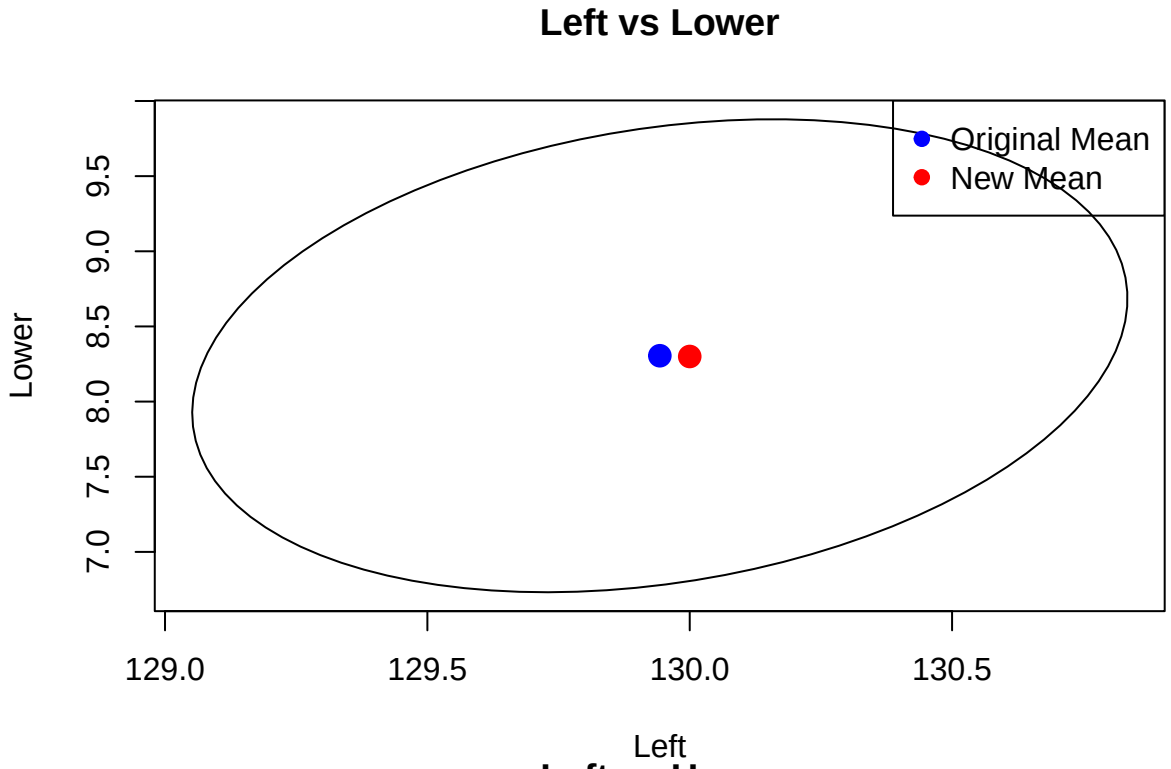

Length vs Right
Original Mean
New Mean
Right
130.5
130.0
129.5
129.0
214.0
214.5
215.0
215.5
Length

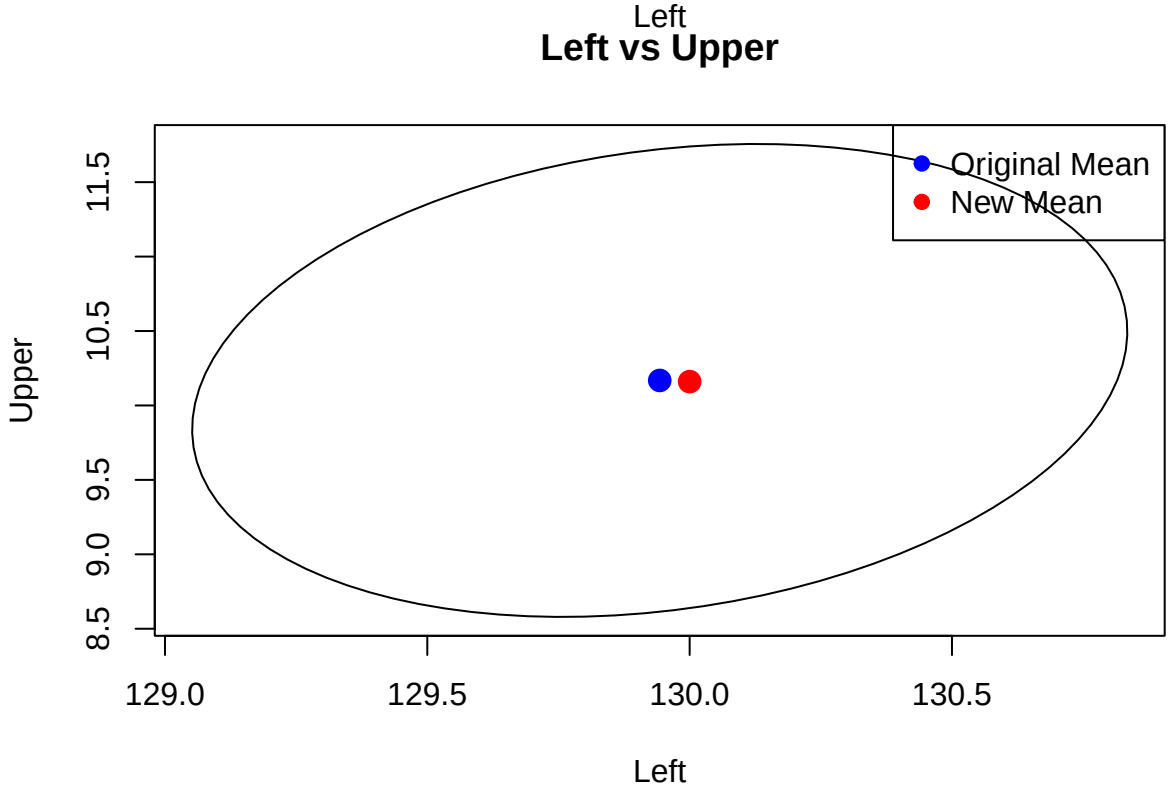

Length vs Lower
Original Mean
New Mean
Lower
7.0
7.5
8.0
8.5
9.0
9.5
214.0
214.5
215.0
215.5
Length

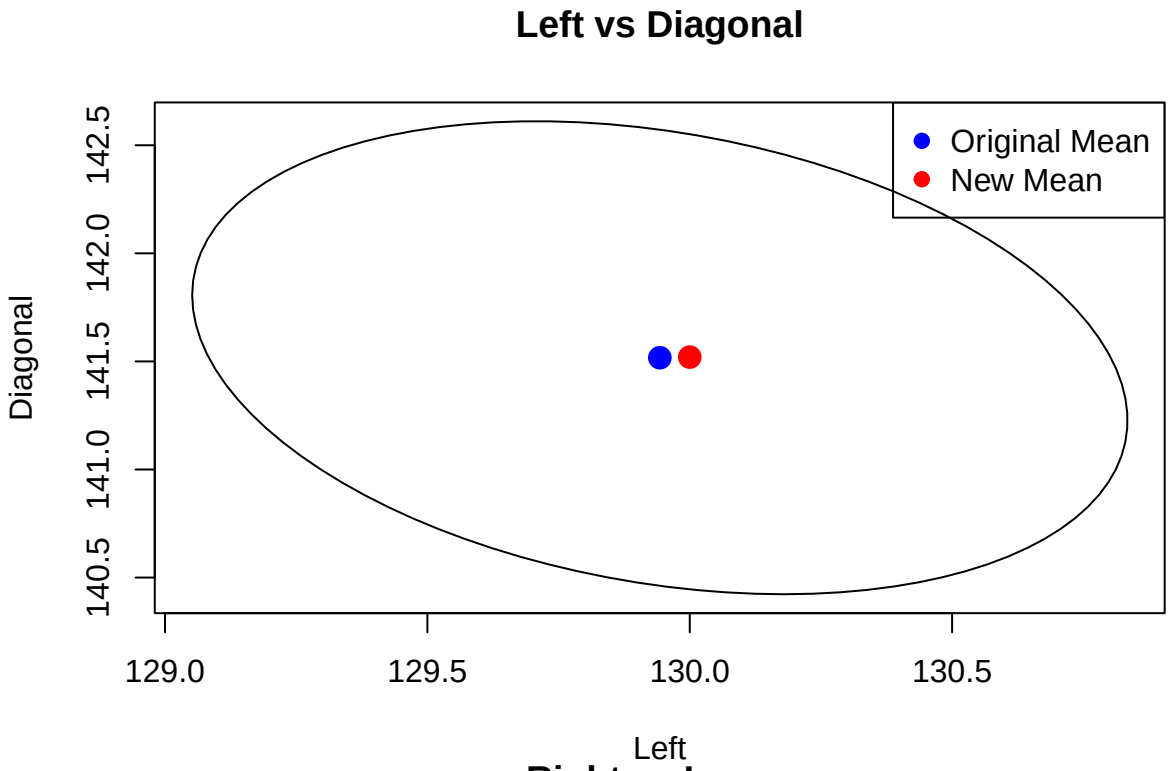

Length vs Upper
Original Mean
New Mean
Upper
8.5
9.0
9.5
10.5
11.5
214.0
214.5
215.0
215.5
Length

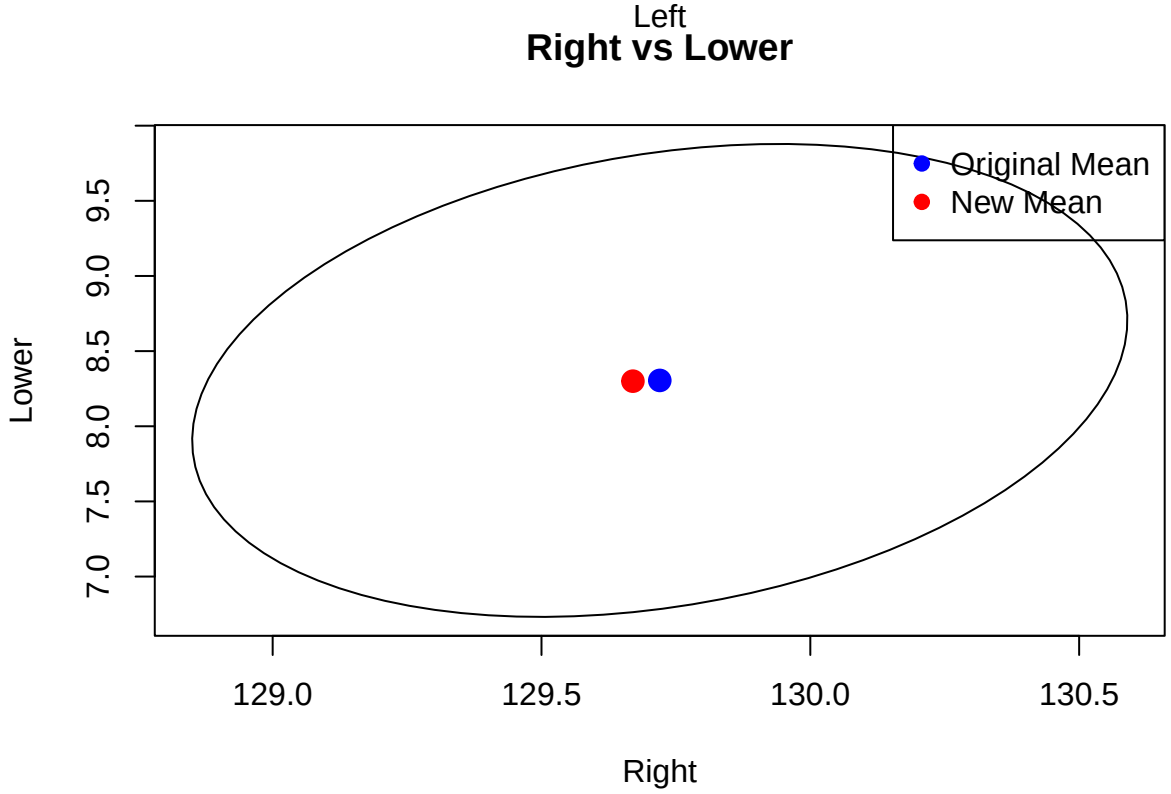

Length vs Diagonal
Original Mean
New Mean
Diagonal
142.5
142.0
141.5
141.0
140.5
214.0
214.5
215.0
215.5
Length

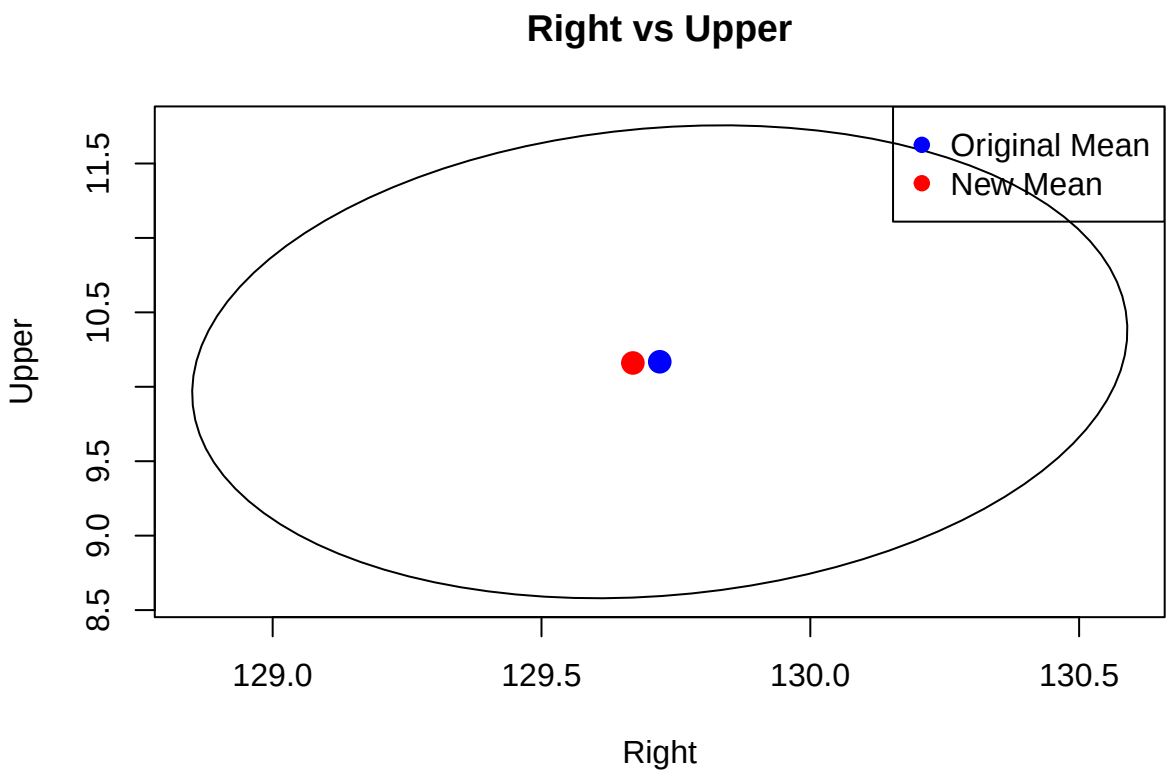

Left vs Right
Original Mean
New Mean
Right
130.5
130.0
129.5
129.0
129.0
129.5
130.0
130.5
Left

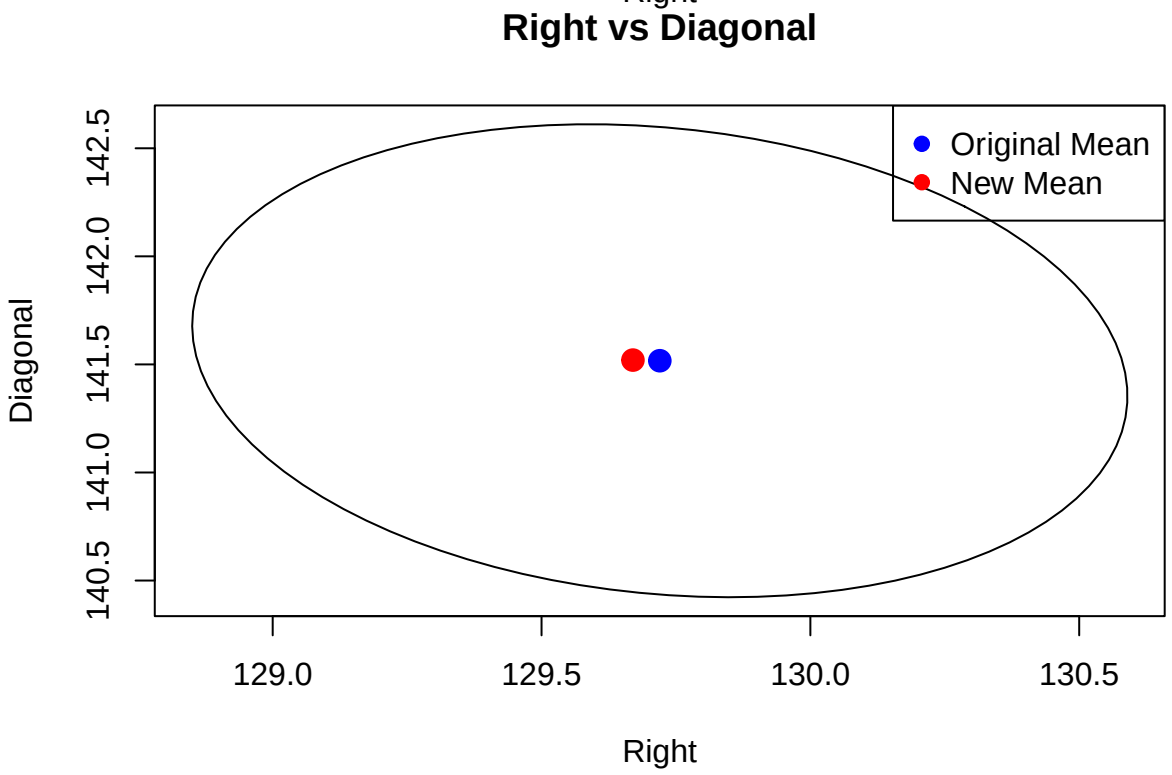

Left vs Lower
Original Mean
New Mean
Lower
7.0
7.5
8.0
8.5
9.0
9.5
129.0
129.5
130.0
130.5
Left

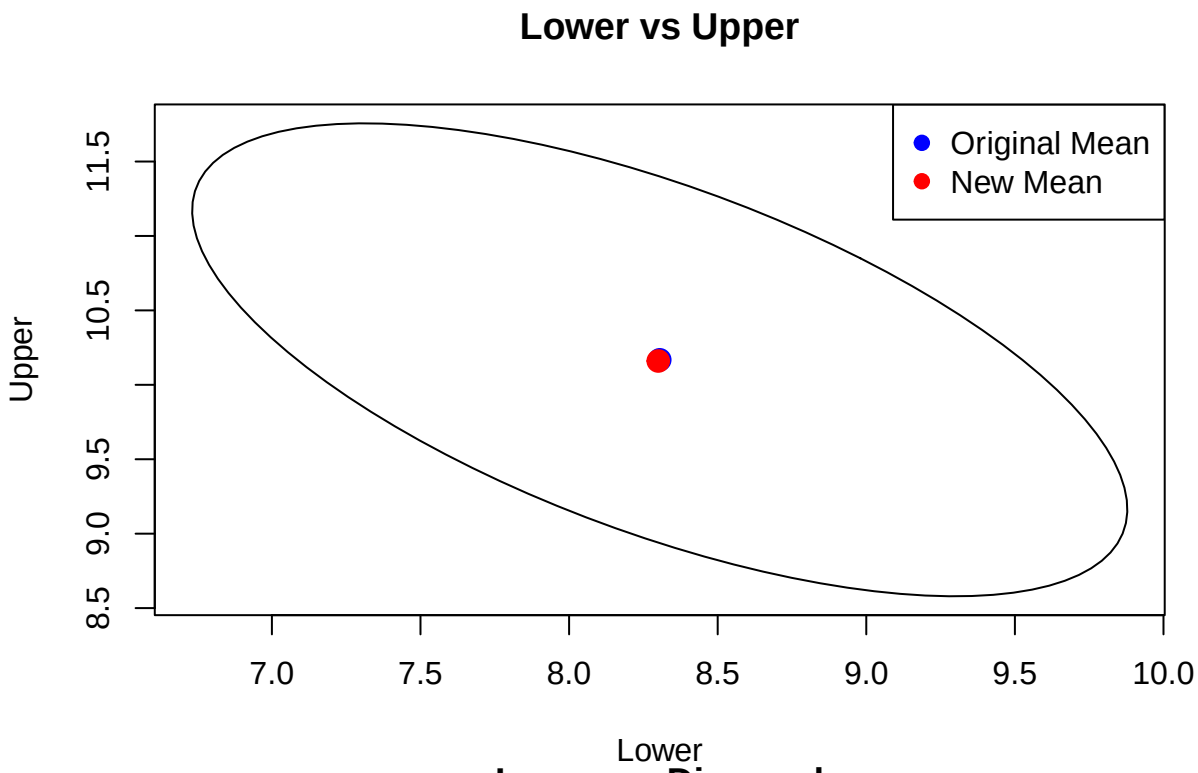

Left vs Upper
Original Mean
New Mean
Upper
8.5
9.0
9.5
10.5
11.5
129.0
129.5
130.0
130.5
Left

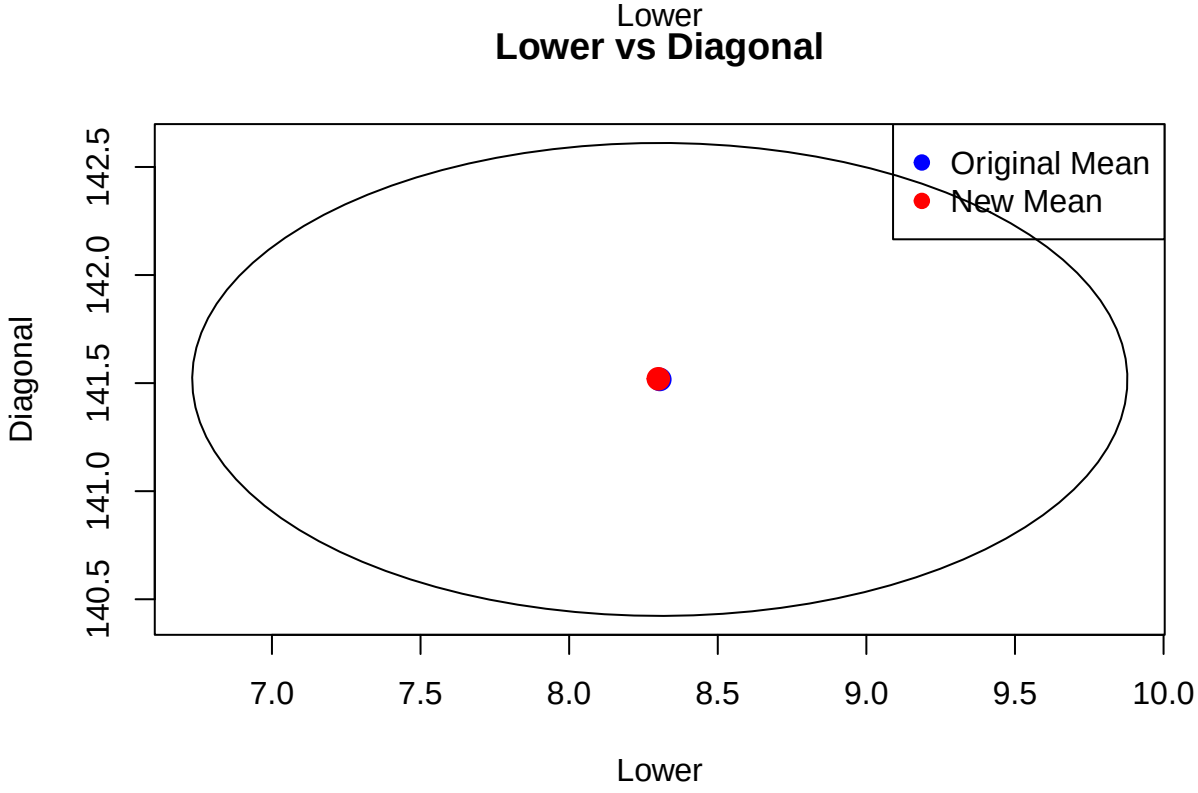

Left vs Diagonal
Original Mean
New Mean
Diagonal
142.5
142.0
141.5
141.0
140.5
129.0
129.5
130.0
130.5
Left

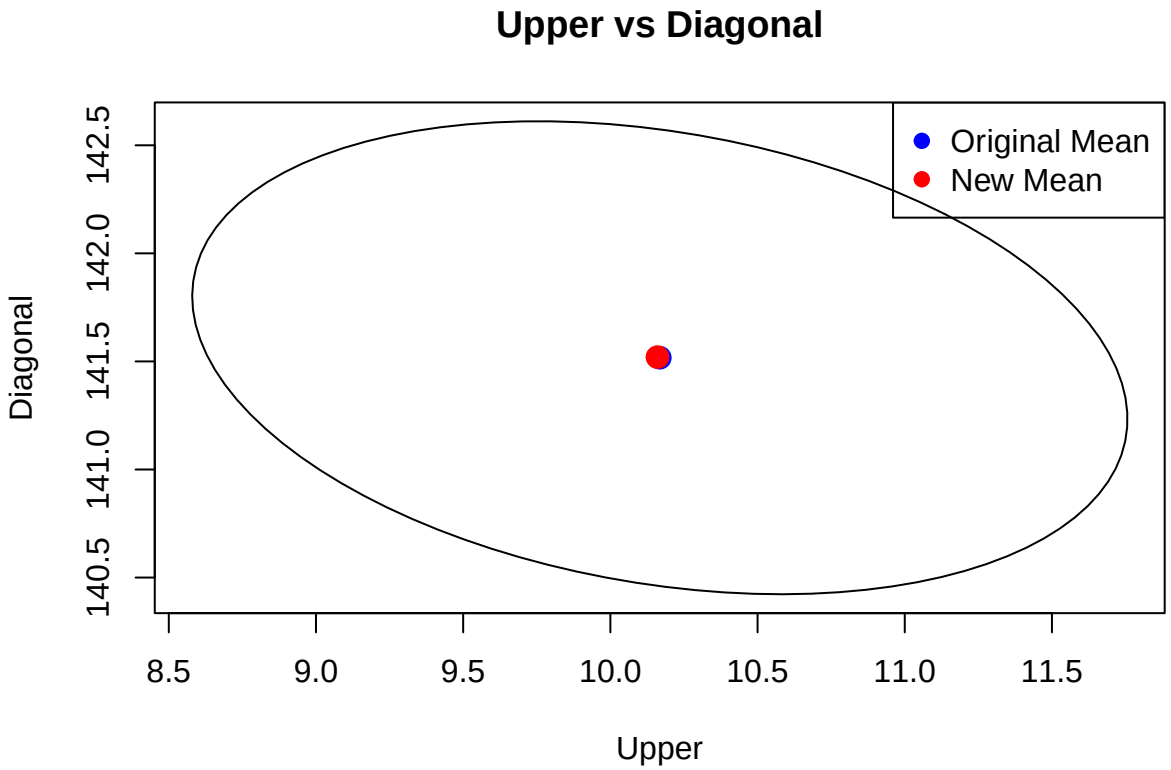

Right vs Lower
Original Mean
New Mean
Lower
9.5
9.0
8.5
8.0
7.5
7.0
129.0
129.5
130.0
130.5
Right


Right vs Upper
Original Mean
New Mean
Upper
11.5
10.5
9.5
9.0
8.5
129.0
129.5
130.0
130.5
Right


Right vs Diagonal
Original Mean
New Mean
Diagonal
142.5
142.0
141.5
141.0
140.5
129.0
129.5
130.0
130.5
Right


Lower vs Upper
Original Mean
New Mean
Upper
11.5
10.5
9.5
9.0
8.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
Lower


Lower vs Diagonal
Original Mean
New Mean
Diagonal
142.5
142.0
141.5
141.0
140.5
7.0
7.5
8.0
8.5
9.0
9.5
10.0
Lower

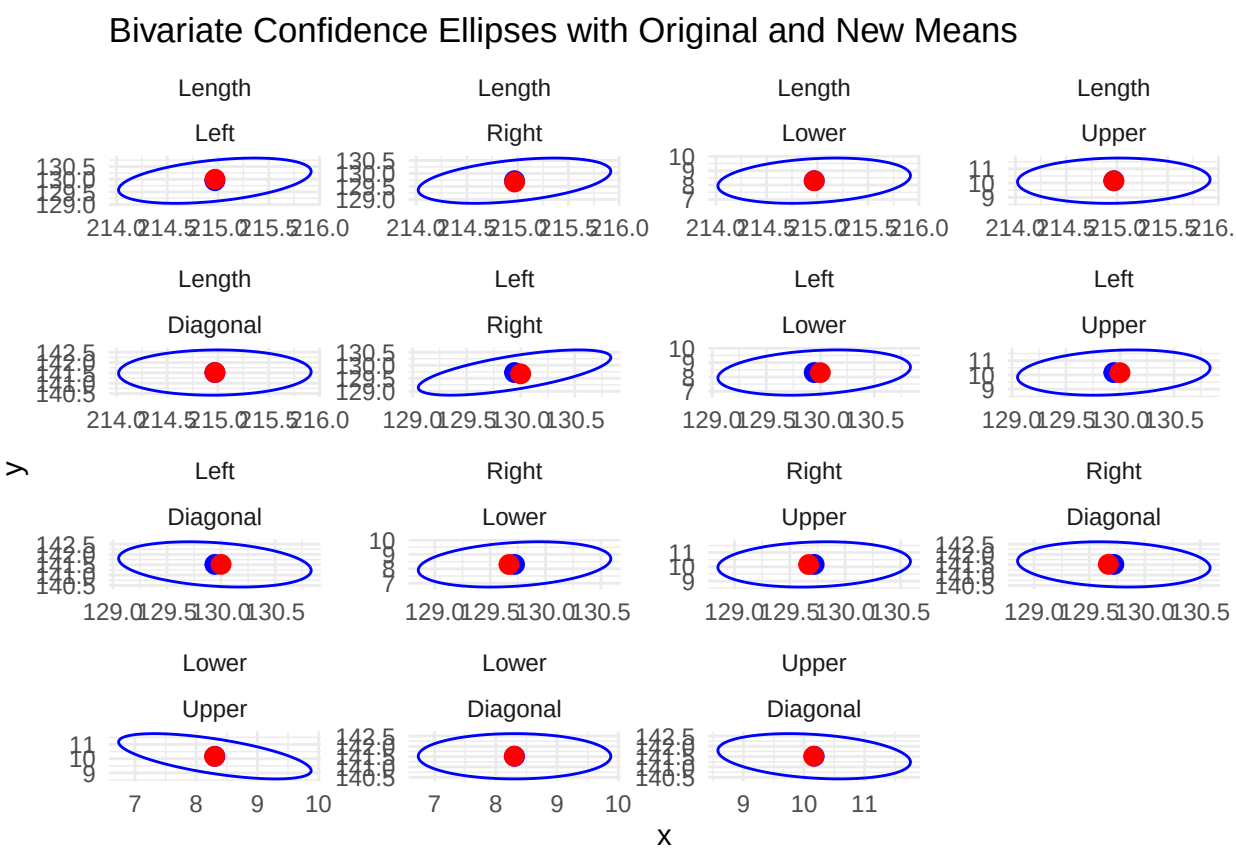


If the red point (new mean) lies within the ellipse (confidence region of the original sample) for each pair of variables, it suggests that the new production line's means are within the multivariate confidence region of the original sample.

1.9

```
ellipse_points <- data.frame()
means <- data.frame()

ellipse_points <- data.frame(x = numeric(), y = numeric(),
                             Variable1 = factor(levels = variable_names),
                             Variable2 = factor(levels = variable_names))

means <- data.frame(Variable1 = factor(), Variable2 = factor(),
                    Original_Mean_X = numeric(), Original_Mean_Y = numeric(),
                    New_Mean_X = numeric(), New_Mean_Y = numeric())

for (i in 1:(length(barX) - 1)) {
  for (j in (i + 1):length(barX)) {
    # Calculate the ellipse data
    ellipse_data <- ellipse(S[c(i, j), c(i, j)], centre = barX[c(i, j)], level = conf_level)
    current_ellipse <- data.frame(x = ellipse_data[, 1], y = ellipse_data[, 2],
                                  Variable1 = variable_names[i], Variable2 = variable_names[j])
    ellipse_points <- rbind(ellipse_points, current_ellipse)

    # Add the original and new means to the means data frame
    current_means <- data.frame(Variable1 = variable_names[i], Variable2 = variable_names[j],
                                Original_Mean_X = barX[i], Original_Mean_Y = barX[j],
                                New_Mean_X = m0[i], New_Mean_Y = m0[j])
```

```
    means <- rbind(means, current_means)
  }
}

ellipse_points$Variable1 <- factor(ellipse_points$Variable1, levels = variable_names)
ellipse_points$Variable2 <- factor(ellipse_points$Variable2, levels = variable_names)
means$Variable1 <- factor(means$Variable1, levels = variable_names)
means$Variable2 <- factor(means$Variable2, levels = variable_names)


ggplot() +
  geom_polygon(data = ellipse_points, aes(x = x, y = y, group = interaction(Variable1, Variable2)), fil
  geom_point(data = means, aes(x = Original_Mean_X, y = Original_Mean_Y), color = 'blue', size = 3) +
  geom_point(data = means, aes(x = New_Mean_X, y = New_Mean_Y), color = 'red', size = 3) +
  facet_wrap(~ Variable1 + Variable2, scales = 'free') +
  theme_minimal() +
  labs(title = 'Bivariate Confidence Ellipses with Original and New Means')
```



1.10

```
m1 <- c(214.99, 129.95, 129.73, 8.51, 9.96, 141.55)


hotelling_result <- InHotCR(m1, barX, IS, n)
print(hotelling_result)
```

```
##       [,1]
## [1,] TRUE
```

```r
bonferroni_result <- InBonCR(m1, barX, S, n)
print(bonferroni_result)
```

```
##       [,1]
## [1,]  TRUE
## [2,]  TRUE
## [3,]  TRUE
## [4,] FALSE
## [5,] FALSE
## [6,]  TRUE
```

```r
# Interpret the results
if (all(hotelling_result)) {
  print("The mean vector m1 is within the Hotelling's confidence region.")
} else {
  print("The mean vector m1 is not within the Hotelling's confidence region.")
}
```

```
## [1] "The mean vector m1 is within the Hotelling's confidence region."
```

```r
if (all(bonferroni_result)) {
  print("The mean vector m1 is within all individual Bonferroni's confidence intervals.")
} else {
  print("The mean vector m1 is not within all individual Bonferroni's confidence intervals.")
}
```

```
## [1] "The mean vector m1 is not within all individual Bonferroni's confidence intervals."
```

1.11

```r
m2 <- c(214.9473, 129.9243, 129.6709, 8.3254, 10.0389, 141.4954)

hotelling_result2 <- InHotCR(m2, barX, IS, n)
print(hotelling_result2)
```

```
##      [,1]
## [1,] TRUE
```

```r
bonferroni_result2 <- InBonCR(m2, barX, S, n)
print(bonferroni_result2)
```

```
##      [,1]
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
```

```r
if (all(hotelling_result2)) {
  print("The mean vector m2 is within the Hotelling's confidence region.")
} else {
  print("The mean vector m2 is not within the Hotelling's confidence region.")
}
```

```
## [1] "The mean vector m2 is within the Hotelling's confidence region."
```

```r
if (all(bonferroni_result2)) {
  print("The mean vector m2 is within all individual Bonferroni's confidence intervals.")
} else {
  print("The mean vector m2 is not within all individual Bonferroni's confidence intervals.")
}
```

```
## [1] "The mean vector m2 is within all individual Bonferroni's confidence intervals."
```

It does not need any more tuning now every test returns that the m0 is ture

Task 2 Simulation

2.1 Tesing for global null

```r
set.seed(123)
p <- 5000
n_simulations <- 1000
alpha <- 0.05
power_bonferroni <- numeric(3)
power_chi_square <- numeric(3)

simulate_and_test <- function(mu_vector) {
  X <- mapply(rnorm, n = 1, mean = mu_vector, sd = 1)

  p_values <- 2 * pnorm(-abs(X))
  bonferroni_rejected <- any(p_values < (alpha / p))

  chi_square_statistic <- sum((X - mu_vector)^2)
  chi_square_rejected <- chi_square_statistic > qchisq(1 - alpha, df = p)

  return(c(bonferroni = bonferroni_rejected, chi_square = chi_square_rejected))
}

for (i in 1:3) {
  mu <- rep(0, p)
  if (i == 1) {
    mu[1:100] <- 1.2 * sqrt(2 * log(p))
  } else if (i == 2) {
    mu[1:100] <- 0.5 * sqrt(2 * log(p))
    mu[101:5000] <- 0
  } else if (i == 3) {
    mu[1:1000] <- 0.2 * sqrt(2 * log(p))
    mu[1001:5000] <- 0
  }

  results <- replicate(n_simulations, simulate_and_test(mu))

  power_bonferroni[i] <- mean(results["bonferroni", ])
  power_chi_square[i] <- mean(results["chi_square", ])
}

conclusion <- data.frame(
  Scenario = c("a", "b", "c"),
```

```
  Power_Bonferroni = power_bonferroni,
  Power_Chi_Square = power_chi_square
)

gt_conclusion <- gt(conclusion)

# print(gt_conclusion)

gt_output4 <- "gt img/gt_conclusion.png"
gtsave(gt_conclusion, gt_output4)
```

Scenario	Power_Bonferroni	Power_Chi_Square
a	1.000	0.047
b	0.638	0.052
c	0.172	0.046

2.2 Multiple testing

```
simulate_and_calculate_errors <- function(mu_vector) {
  X <- rnorm(p, mean = mu_vector, sd = 1)

  p_values <- 2 * pnorm(-abs(X))

  bonferroni_adjusted <- p.adjust(p_values, method = "bonferroni") < alpha
  fwer_bonferroni <- any(bonferroni_adjusted[1:10])  # Adjust according to non-zero mean indices

  bh_adjusted <- p.adjust(p_values, method = "BH") < alpha
  fwer_bh <- any(bh_adjusted[1:10])  # Adjust according to non-zero mean indices

  fdr_bonferroni <- sum(bonferroni_adjusted & mu_vector == 0) / max(1, sum(bonferroni_adjusted))
  fdr_bh <- sum(bh_adjusted & mu_vector == 0) / max(1, sum(bh_adjusted))

  return(c(fwer_bonferroni, fwer_bh, fdr_bonferroni, fdr_bh))
}

fwer_bonferroni <- numeric(n_simulations)
fwer_bh <- numeric(n_simulations)
fdr_bonferroni <- numeric(n_simulations)
fdr_bh <- numeric(n_simulations)

for (scenario in 1:2) {
```

```r
  mu <- rep(0, p)
  if (scenario == 1) {
    mu[1:10] <- sqrt(2 * log(p))
  } else {
    mu[1:500] <- sqrt(2 * log(p))
  }

  errors <- t(replicate(n_simulations, simulate_and_calculate_errors(mu)))

  fwer_bonferroni[scenario] <- mean(errors[, 1])
  fwer_bh[scenario] <- mean(errors[, 2])
  fdr_bonferroni[scenario] <- mean(errors[, 3])
  fdr_bh[scenario] <- mean(errors[, 4])
}

conclusion2 <- data.frame(
  Scenario = c("a", "b"),
  FWER_Bonferroni = fwer_bonferroni[1:2],
  FWER_BH = fwer_bh[1:2],
  FDR_Bonferroni = fdr_bonferroni[1:2],
  FDR_BH = fdr_bh[1:2]
)

gt_conclusion2 <- gt(conclusion2)

# print(gt_conclusion2)

gt_output5 <- "gt img/gt_conclusion2.png"
gtsave(gt_conclusion2, gt_output5)
```

Scenario	FWER_Bonferroni	FWER_BH	FDR_Bonferroni	FDR_BH
a	0.990	0.995	0.015793254	0.05210639
b	0.991	1.000	0.000169481	0.04495750