STAD37

Richard Zhen

Due: November 23, 2021, 11:00pm

library("tidyverse")

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.0.6 v dplyr 1.0.4  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library("readr")  
library("dplyr")

## 1(a)

# Reads the data from "a3\_profile\_data.txt" and sets it to the variable plasma  
plasma <- read.table("a3\_profile\_data.txt", header = T)  
# Displays the data frame  
plasma

## Group x1 x2 x3  
## 1 1 125 137 121  
## 2 1 144 173 147  
## 3 1 105 119 125  
## 4 1 151 149 128  
## 5 1 137 139 109  
## 6 2 93 121 107  
## 7 2 116 135 106  
## 8 2 109 83 100  
## 9 2 89 95 83  
## 10 2 116 128 100

# Creates two different subsets for group 1 and group 2  
plasmag1 <- subset(plasma, Group == 1, select = c(x1,x2,x3))  
plasmag2 <- subset(plasma, Group == 2, select = c(x1,x2,x3))  
# Displays up to the first six entries of both group 1 and group 2  
head(plasmag1)

## x1 x2 x3  
## 1 125 137 121  
## 2 144 173 147  
## 3 105 119 125  
## 4 151 149 128  
## 5 137 139 109

head(plasmag2)

## x1 x2 x3  
## 6 93 121 107  
## 7 116 135 106  
## 8 109 83 100  
## 9 89 95 83  
## 10 116 128 100

# Set the values of the rows and columns and displays them  
p <- ncol(plasmag1)  
p

## [1] 3

n1 <- nrow(plasmag1)  
n1

## [1] 5

n2 <- nrow(plasmag2)  
n2

## [1] 5

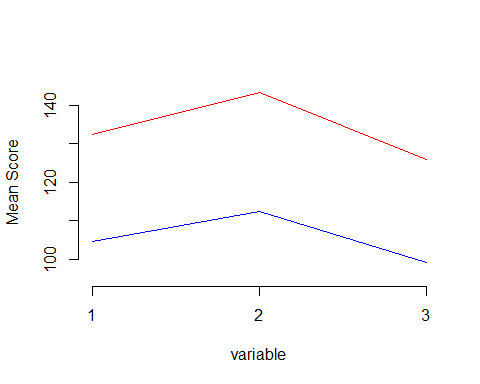
# Calculates the column means of Group 1 and Group 2 and displays them  
xbar1 <- colMeans(plasmag1)  
xbar1

## x1 x2 x3   
## 132.4 143.4 126.0

xbar2 <- colMeans(plasmag2)  
xbar2

## x1 x2 x3   
## 104.6 112.4 99.2

# Plots and displays the profile graph and a line for Group 1  
variable <- c(1,2,3)  
plot(variable,xbar1,col = "red",ylab = "Mean Score" , axes = FALSE, type = "l", ylim = range(c(95,145)))  
# Adjust the x-axis of the graph  
axis(side = 1, at = variable)  
# Adjust the y-axis of the graph  
axis(side = 2)  
# Creates a line for Graph 2  
lines(variable,xbar2,col="blue")



What I have noticed in the plot is that the slopes of the lines are parallel to each other, the mean scores of Group 1 are greater than Group 2 mean scores, and that the lines/profiles are not overlaping/coincidient.

## 1(b)

Testing for parallel profiles

# Calculates the variance-covariance of Group 1 and Group 2   
S1 <- cov(plasmag1)  
S2 <- cov(plasmag2)  
# Display the variance-covariance of Group 1, Group 2, and pooled  
S1

## x1 x2 x3  
## x1 326.80 285.8 66.75  
## x2 285.80 390.8 191.00  
## x3 66.75 191.0 190.00

S2

## x1 x2 x3  
## x1 164.30 119.45 63.1  
## x2 119.45 498.80 122.9  
## x3 63.10 122.90 92.7

Sp <- (1/(n1+n2-2))\*((n1-1)\*S1+(n2-1)\*S2)  
Sp

## x1 x2 x3  
## x1 245.550 202.625 64.925  
## x2 202.625 444.800 156.950  
## x3 64.925 156.950 141.350

# Creates the contrast matrix   
C<-matrix(c(-1,1,0,0,-1,1),byrow=TRUE,p-1,p)  
# Displays the contrast matrix  
C

## [,1] [,2] [,3]  
## [1,] -1 1 0  
## [2,] 0 -1 1

C%\*%(xbar1-xbar2)

## [,1]  
## [1,] 3.2  
## [2,] -4.2

# Sets to 90% confidence  
alpha <- 0.05  
# Calculates the Hotelling T^2 value for parallel  
T\_sq\_parallel <- ((1/n1+1/n2)^(-1))\*t(xbar1-  
xbar2)%\*%t(C)%\*%solve(C%\*%Sp%\*%t(C))%\*%C%\*%(xbar1-xbar2)  
# Displays the Hotelling T^2 value  
T\_sq\_parallel

## [,1]  
## [1,] 0.1716331

# Calculates the critcal value for parallel profiles  
CritValue\_parallel <- (((n1+n2-2)\*(p-1))/(n1+n2-p))\*qf(1-alpha, p-1, n1+n2-p)  
# Displays the critcal value for parallel profiles  
CritValue\_parallel

## [1] 10.82838

For H\_{01}: C{mu1} = C{mu2}, the critical value is greater than the T^2 value, we do not reject H01

Testing for coincident profiles

# creates a column matrix of ones in the entries  
one <- as.matrix(c(rep(1,p)))  
# Displays the one matrix  
one

## [,1]  
## [1,] 1  
## [2,] 1  
## [3,] 1

# Calculates the Hotelling T^2 value for coincident  
T\_sq\_coincident <- ((t(one)%\*%(xbar1- xbar2))/sqrt((1/n1+1/n2)\*(t(one)%\*%Sp%\*%one)))^2  
# Displays the Hotelling T^2 value for coincident  
T\_sq\_coincident

## [,1]  
## [1,] 10.89927

# Calulates the critical value for coincident  
CritValue\_coincident <- qf(1-alpha, 1, n1+n2-2)  
# Displays the critical value for coincident  
CritValue\_coincident

## [1] 5.317655

For H\_{02}: 1'(mu1 - mu2), the critical value is less than than the T^2 value, we do reject H02

Therefore, the profiles are parallel but not coincident then we do not need to test if the profiles are levelled.

## 2(a)

# Read the data from "1waymanova.txt" and sets it to the variable fish  
fish <- read.table("1waymanova.txt", header = T)  
# Displays the data   
fish

## x1 x2 x3 x4 Method  
## 1 5.4 6.0 6.3 6.7 1  
## 2 5.2 6.2 6.0 5.8 1  
## 3 6.1 5.9 6.0 7.0 1  
## 4 4.8 5.0 4.9 5.0 1  
## 5 5.0 5.7 5.0 6.5 1  
## 6 5.7 6.1 6.0 6.6 1  
## 7 6.0 6.0 5.8 6.0 1  
## 8 4.0 5.0 4.0 5.0 1  
## 9 5.7 5.4 4.9 5.0 1  
## 10 5.6 5.2 5.4 5.8 1  
## 11 5.8 6.1 5.2 6.4 1  
## 12 5.3 5.9 5.8 6.0 1  
## 13 5.0 5.3 5.3 6.5 2  
## 14 4.8 4.9 4.2 5.6 2  
## 15 3.9 4.0 4.4 5.0 2  
## 16 4.0 5.1 4.8 5.8 2  
## 17 5.6 5.4 5.1 6.2 2  
## 18 6.0 5.5 5.7 6.0 2  
## 19 5.2 4.8 5.4 6.0 2  
## 20 5.3 5.1 5.8 6.4 2  
## 21 5.9 6.1 5.7 6.0 2  
## 22 6.1 6.0 6.1 6.2 2  
## 23 6.2 5.7 5.9 6.0 2  
## 24 5.1 4.9 5.3 4.8 2  
## 25 4.8 5.0 6.5 7.0 3  
## 26 5.4 5.0 6.0 6.4 3  
## 27 4.9 5.1 5.9 6.5 3  
## 28 5.7 5.2 6.4 6.4 3  
## 29 4.2 4.6 5.3 6.3 3  
## 30 6.0 5.3 5.8 6.4 3  
## 31 5.1 5.2 6.2 6.5 3  
## 32 4.8 4.6 5.7 5.7 3  
## 33 5.3 5.4 6.8 6.6 3  
## 34 4.6 4.4 5.7 5.6 3  
## 35 4.5 4.0 5.0 5.9 3  
## 36 4.4 4.2 5.6 5.5 3

# Displays the sample means of each method  
aggregate(fish[, 1:4], list(fish$Method), mean)

## Group.1 x1 x2 x3 x4  
## 1 1 5.383333 5.708333 5.441667 5.983333  
## 2 2 5.258333 5.233333 5.308333 5.875000  
## 3 3 4.975000 4.833333 5.908333 6.233333

## 2(b)

#Creates subsets for each method  
f1 <- subset(fish, Method==1)  
f2 <- subset(fish, Method==2)  
f3 <- subset(fish, Method==3)  
# Sets the value of each row of the methods and displays them  
n1 <- nrow(f1)  
n2 <- nrow(f2)  
n3 <- nrow(f3)  
n <- n1+n2+n3  
n1

## [1] 12

n2

## [1] 12

n3

## [1] 12

n

## [1] 36

# Variance-covariance matrix for method 1  
S1 <- cov(f1[,1:4])  
# Displays variance-covariance matrix for method 1  
S1

## x1 x2 x3 x4  
## x1 0.3415152 0.1537879 0.2616667 0.2306061  
## x2 0.1537879 0.1953788 0.2296212 0.2283333  
## x3 0.2616667 0.2296212 0.4408333 0.3271212  
## x4 0.2306061 0.2283333 0.3271212 0.4851515

# Variance-covariance matrix for method 2  
S2 <- cov(f2[,1:4])  
# Displays variance-covariance matrix for method 2  
S2

## x1 x2 x3 x4  
## x1 0.5826515 0.3587879 0.3721970 0.1915909  
## x2 0.3587879 0.3296970 0.2496970 0.1781818  
## x3 0.3721970 0.2496970 0.3535606 0.1611364  
## x4 0.1915909 0.1781818 0.1611364 0.2675000

# Variance-covariance matrix for method 3  
S3 <- cov(f3[,1:4])  
# Displays variance-covariance matrix for method 3  
S3

## x1 x2 x3 x4  
## x1 0.2947727 0.1945455 0.1547727 0.1109091  
## x2 0.1945455 0.2115152 0.1896970 0.1633333  
## x3 0.1547727 0.1896970 0.2608333 0.1515152  
## x4 0.1109091 0.1633333 0.1515152 0.2078788

## 2(c)

# Calculates sum of squares and cross products matrix W  
W <- (n1-1)\*S1+(n2-1)\*S2+(n3-1)\*S3  
# Displays W  
W

## x1 x2 x3 x4  
## x1 13.408333 7.778333 8.675000 5.864167  
## x2 7.778333 8.102500 7.359167 6.268333  
## x3 8.675000 7.359167 11.607500 7.037500  
## x4 5.864167 6.268333 7.037500 10.565833

## 2(d)

# Total SS  
T <- (n-1)\*cov(fish[,1:4])  
# Displays Total SS  
T

## x1 x2 x3 x4  
## x1 14.458889 9.898333 7.299444 5.103889  
## x2 9.898333 12.707500 5.019167 5.025833  
## x3 7.299444 5.019167 13.989722 8.421944  
## x4 5.103889 5.025833 8.421944 11.376389

# Calculates sum of squares and cross products, B  
B <- T-W  
# Displays B  
B

## x1 x2 x3 x4  
## x1 1.0505556 2.1200 -1.375556 -0.7602778  
## x2 2.1200000 4.6050 -2.340000 -1.2425000  
## x3 -1.3755556 -2.3400 2.382222 1.3844444  
## x4 -0.7602778 -1.2425 1.384444 0.8105556

## 2(e)

# Lambda value  
lambda <- det(W)/(det(B+W))  
lambda

## [1] 0.2200479

# Populations  
g <- 3   
# Varibles  
p <- 4   
# Squaroots lambda value  
lambda.sqrt <- sqrt(lambda)  
# Calculates and displays test statistic  
TestStat <- ((n-p-2)/(p))\*((1-lambda.sqrt)/lambda.sqrt)  
TestStat

## [1] 8.488314

# Calculates and displays critical value  
CritValue <- qf(1-0.05, 2\*p, 2\*(n-p-2))  
CritValue

## [1] 2.096968

For H0: mu1 = mu2 = mu3, the critical value is less than the test statistic, we do reject H0.

## 2(f)

# Calculates and displays test statistic  
TestStat <- -(n-1-((p+g)/2))\*log(lambda)  
TestStat

## [1] 47.68817

# Calculates and displays critical value  
CritValue <- qchisq(p=1-0.05,df=p\*(g-1))  
CritValue

## [1] 15.50731

For H0: mu1 = mu2 = mu3, the critical value is less than the test statistic, we do reject H0.

## 3(a)

# Read the data from "2sample\_data.txt" and sets it to the variable skull  
skull <- read.table("2sample\_data.txt", header = T)

## Warning in if (!header) rlabp <- FALSE: the condition has length > 1 and only  
## the first element will be used

## Warning in if (header) {: the condition has length > 1 and only the first  
## element will be used

# Displays the data  
skull

## x1 x2 x3 x4 Period  
## 1 131 138 89 49 1  
## 2 125 131 92 48 1  
## 3 131 132 99 50 1  
## 4 119 132 96 44 1  
## 5 136 143 100 54 1  
## 6 138 137 89 56 1  
## 7 139 130 108 48 1  
## 8 125 136 93 48 1  
## 9 131 134 102 51 1  
## 10 134 134 99 51 1  
## 11 129 138 95 50 1  
## 12 134 121 95 53 1  
## 13 126 129 109 51 1  
## 14 132 136 100 50 1  
## 15 141 140 100 51 1  
## 16 131 134 97 54 1  
## 17 135 137 103 50 1  
## 18 132 133 93 53 1  
## 19 139 136 96 50 1  
## 20 132 131 101 49 1  
## 21 126 133 102 51 1  
## 22 135 135 103 47 1  
## 23 134 124 93 53 1  
## 24 128 134 103 50 1  
## 25 130 130 104 49 1  
## 26 138 135 100 55 1  
## 27 128 132 93 53 1  
## 28 127 129 106 48 1  
## 29 131 136 114 54 1  
## 30 124 138 101 46 1  
## 31 126 137 100 47 2  
## 32 135 133 96 47 2  
## 33 140 133 97 44 2  
## 34 150 128 103 50 2  
## 35 128 123 94 44 2  
## 36 137 135 97 51 2  
## 37 134 144 99 53 2  
## 38 135 129 101 47 2  
## 39 133 133 95 49 2  
## 40 135 124 93 45 2  
## 41 135 135 102 52 2  
## 42 133 138 97 50 2  
## 43 133 135 98 55 2  
## 44 140 133 97 48 2  
## 45 132 135 103 52 2  
## 46 133 127 97 44 2  
## 47 140 128 106 52 2  
## 48 125 130 100 50 2  
## 49 132 128 104 46 2  
## 50 136 129 92 53 2  
## 51 139 135 105 48 2  
## 52 128 130 99 47 2  
## 53 137 135 96 51 2  
## 54 131 125 90 49 2  
## 55 136 138 100 48 2  
## 56 133 133 89 52 2  
## 57 134 129 103 49 2  
## 58 132 131 92 51 2  
## 59 137 131 97 53 2  
## 60 132 127 100 50 2

# Displays the sample means of each period  
aggregate(skull[,1:4], list(skull$Period), mean)

## Group.1 x1 x2 x3 x4  
## 1 1 131.3667 133.6 99.16667 50.53333  
## 2 2 134.3667 131.7 98.06667 49.23333

## 3(b)

#Creates subsets for each period  
k1 <- subset(skull, Period==1)  
k2 <- subset(skull, Period==2)  
# Sets the value of each row of the periods and displays them  
n1 <- nrow(f1)  
n2 <- nrow(f2)  
n3 <- nrow(f3)  
n <- n1+n2+n3  
n1

## [1] 12

n2

## [1] 12

n3

## [1] 12

n

## [1] 36

# Variance-covariance matrix for period 1  
S1 <- cov(k1[,1:4])  
# Displays variance-covariance matrix for period 1  
S1

## x1 x2 x3 x4  
## x1 26.309195 4.1517241 0.4540230 7.2459770  
## x2 4.151724 19.9724138 -0.7931034 0.3931034  
## x3 0.454023 -0.7931034 34.6264368 -1.9195402  
## x4 7.245977 0.3931034 -1.9195402 7.6367816

# Variance-covariance matrix for period 2  
S2 <- cov(k2[,1:4])  
# Displays variance-covariance matrix for period 2  
S2

## x1 x2 x3 x4  
## x1 23.136782 1.010345 4.7678161 1.8425287  
## x2 1.010345 21.596552 3.3655172 5.6241379  
## x3 4.767816 3.365517 18.8919540 0.1908046  
## x4 1.842529 5.624138 0.1908046 8.7367816

# Pooled variance-covariance matrix  
Sp <- (1/(n1-1 + n2-1))\*((n1-1)\*S1 + (n2-1)\*S2)  
# Displays pooled variance-covariance matrix  
Sp

## x1 x2 x3 x4  
## x1 24.722989 2.581034 2.6109195 4.5442529  
## x2 2.581034 20.784483 1.2862069 3.0086207  
## x3 2.610920 1.286207 26.7591954 -0.8643678  
## x4 4.544253 3.008621 -0.8643678 8.1867816

## 3(c)

Box's M test

# Population  
g <- 2   
# Variables  
p <- 4   
# Calculates and displays M  
M <- ((n1+n2-2)\*log(det(Sp))) - (((n1-1)\*log(det(S1))) + ((n2-1)\*log(det(S2))))  
M

## [1] 4.155505

# Calculates and displays u  
u <- (1/(n1-1) + (1/(n2-1) -1/(n1+n2-2))) \* (((2\*p\*\*2)+3\*p-1)/(6\*(p+1)\*(g-1)))  
u

## [1] 0.1954545

# Calculates and displays the test statistic  
TestStat <- (1-u)\*M  
TestStat

## [1] 3.343293

# Calculates and displays the critical value  
CritValue <- qchisq(p=1-0.05,df=(1/2)\*p\*(p+1)\*(g-1))  
CritValue

## [1] 18.30704

For H\_0: Sum\_1 = Sum\_2, the critical value is greater than the test statistic, we do not reject H0

## 3(d)

# Column means of each period  
xbar1 <- colMeans(k1[,1:4])  
xbar2 <- colMeans(k2[,1:4])  
xbar1

## x1 x2 x3 x4   
## 131.36667 133.60000 99.16667 50.53333

xbar2

## x1 x2 x3 x4   
## 134.36667 131.70000 98.06667 49.23333

# create delta row matrix of zeros  
delta <- c(0,0,0,0)  
delta

## [1] 0 0 0 0

# Calculates and displays T^2 value  
T\_sq <- t(xbar1-xbar2-delta)%\*%solve((1/n1+1/n2)\*Sp)%\*%(xbar1-xbar2-delta)  
T\_sq

## [,1]  
## [1,] 6.3549

# Calculates and displays critical value  
criticalValue <- (((n1+n2-2)\*p)/(n1+n2-p-1))\*qf(1-0.05,p, n1+n2-p-1 )  
criticalValue

## [1] 13.40892

# Checks which linear combination is most responsible for the rejection  
a <- solve(Sp)%\*%(xbar1-xbar2)   
a

## [,1]  
## x1 -0.17945450  
## x2 0.07547477  
## x3 0.06265372  
## x4 0.23728097

For H0 mu1 - mu2 =0, the critical value is less than the T^2 value, we do reject H0

## 3(e)

Bonferroni SCIs

t <- qt(1-0.05/(p\*g\*(g-1)),df=n-g)  
for(i in 1:p){  
 LCI12 <- (xbar1[i]-xbar2[i])-t\*sqrt(Sp[i,i]\*(1/n1+1/n2))  
 UCI12 <- (xbar1[i]-xbar2[i])+t\*sqrt(Sp[i,i]\*(1/n1+1/n2))  
 cat("Bon SCI for mu1[",i,"]-mu2[",i,"] = (",LCI12,",",UCI12,")\n",sep="")  
}

## Bon SCI for mu1[1]-mu2[1] = (-8.354074,2.354074)  
## Bon SCI for mu1[2]-mu2[2] = (-3.009119,6.809119)  
## Bon SCI for mu1[3]-mu2[3] = (-4.470196,6.670196)  
## Bon SCI for mu1[4]-mu2[4] = (-1.780992,4.380992)

Bon SCI for mu1[1]-mu2[1] = (-8.354074,2.354074) Bon SCI for mu1[2]-mu2[2] = (-3.009119,6.809119) Bon SCI for mu1[3]-mu2[3] = (-4.470196,6.670196) Bon SCI for mu1[4]-mu2[4] = (-1.780992,4.380992)

SCIs

c\_square <- ((n1+n2-2)\*p)/(n1+n2-p-1)\*qf(1-0.05,df1 = p,df2 = n1+n2-p-1)  
c <- sqrt(c\_square)  
for(i in 1:p){  
LCI12 <- (xbar1[i]-xbar2[i])-c\*sqrt(Sp[i,i]\*(1/n1+1/n2))  
UCI12 <- (xbar1[i]-xbar2[i])+c\*sqrt(Sp[i,i]\*(1/n1+1/n2))  
cat("SCI for mu1[",i,"]-mu2[",i,"] = (",LCI12,",",UCI12,")\n",sep="")  
}

## SCI for mu1[1]-mu2[1] = (-10.43313,4.43313)  
## SCI for mu1[2]-mu2[2] = (-4.915392,8.715392)  
## SCI for mu1[3]-mu2[3] = (-6.633174,8.833174)  
## SCI for mu1[4]-mu2[4] = (-2.97738,5.57738)

SCI for mu1[1]-mu2[1] = (-10.43313,4.43313) SCI for mu1[2]-mu2[2] = (-4.915392,8.715392) SCI for mu1[3]-mu2[3] = (-6.633174,8.833174) SCI for mu1[4]-mu2[4] = (-2.97738,5.57738)

## 4(a)

# Read the data from "t6\_17.txt" and sets it to the variable nut  
nut <- read.table("t6\_17.txt", header = T)

## Warning in if (!header) rlabp <- FALSE: the condition has length > 1 and only  
## the first element will be used

## Warning in if (header) {: the condition has length > 1 and only the first  
## element will be used

# Displays the data  
nut

## Fac1 Fac2 x1 x2 x3  
## 1 1 5 195.3 153.1 51.4  
## 2 1 5 194.3 167.7 53.7  
## 3 2 5 189.7 139.5 55.5  
## 4 2 5 180.4 121.1 44.4  
## 5 1 6 203.0 156.8 49.8  
## 6 1 6 195.9 166.0 45.8  
## 7 2 6 202.7 166.1 60.4  
## 8 2 6 197.6 161.8 54.1  
## 9 1 8 193.5 164.5 57.8  
## 10 1 8 187.0 165.1 58.6  
## 11 2 8 201.5 166.8 65.0  
## 12 2 8 200.0 173.8 67.2

# Observation vectors  
p <- 3   
# Levels for factor 1  
g <- 2   
# Levels for factor 2  
b <- 3   
# Observations per treatment  
n <- 2

fac1 <- factor(nut$Fac1)  
fac2 <- factor(nut$Fac2)  
nut345 <- as.matrix(nut[,3:5])  
# Creates the fitted model  
fit <- manova(nut345~fac1\*fac2)  
# Simulates the Wilks test  
summary(fit,test = "Wilks")

## Df Wilks approx F num Df den Df Pr(>F)   
## fac1 1 0.106516 11.1843 3 4 0.020502 \*   
## fac2 2 0.012444 10.6191 6 8 0.001928 \*\*  
## fac1:fac2 2 0.074300 3.5582 6 8 0.050794 .   
## Residuals 6   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

For fac1, H0: tau1 = tau2 = 0, the p-value 0.020502 is less than alpha = 0.05, we do reject H0

For fac2, H0: beta1 = beta2 = beta3 = 0, the p-value 0.001928 is less than alpha = 0.05, we do reject H0

For fac1:fac2, H0: gamma11 = gamma12 = gammaab = 0, the p-value 0.050794 is greater than alpha = 0.05, we do not reject H0

## 4(b)

# Sum of squares and cross products matrices for SSPfac1  
moreinfo <- summary(fit)  
SSPfac1 <- moreinfo$SS[1]  
SSPfac1

## $fac1  
## x1 x2 x3  
## x1 0.7008333 -10.6575 7.129167  
## x2 -10.6575000 162.0675 -108.412500  
## x3 7.1291667 -108.4125 72.520833

# Sum of squares and cross products matrices for SSPfac2  
SSPfac2 <- moreinfo$SS[2]  
SSPfac2

## $fac2  
## x1 x2 x3  
## x1 196.1150 365.1825 42.6275  
## x2 365.1825 1089.0150 414.6550  
## x3 42.6275 414.6550 284.1017

# Sum of squares and cross products matrices for SSPint  
SSPint <- moreinfo$SS[[3]]  
SSPint

## x1 x2 x3  
## x1 205.1017 363.6675 107.78583  
## x2 363.6675 780.6950 254.22000  
## x3 107.7858 254.2200 85.95167

# Sum of squares and cross products matrices for SSPres  
SSPres <- moreinfo$SS[[4]]  
SSPres

## x1 x2 x3  
## x1 104.205 49.365 76.480  
## x2 49.365 352.105 121.995  
## x3 76.480 121.995 94.835

## 4(c)

# Creates the fitted model  
fit2 <- manova(nut345~fac1+fac2)  
moreinfo2 <- summary(fit2)

# Sum of squares and cross products matrices for SSPfac1  
SSPfac1\_2 <- moreinfo2$SS[1]  
SSPfac1\_2

## $fac1  
## x1 x2 x3  
## x1 0.7008333 -10.6575 7.129167  
## x2 -10.6575000 162.0675 -108.412500  
## x3 7.1291667 -108.4125 72.520833

# Sum of squares and cross products matrices for SSPfac2  
SSPfac2\_2 <- moreinfo2$SS[2]  
SSPfac2\_2

## $fac2  
## x1 x2 x3  
## x1 196.1150 365.1825 42.6275  
## x2 365.1825 1089.0150 414.6550  
## x3 42.6275 414.6550 284.1017

# Sum of squares and cross products matrices for SSPint  
SSPres\_2 <- moreinfo2$SS[3]  
SSPres\_2

## $Residuals  
## x1 x2 x3  
## x1 309.3067 413.0325 184.2658  
## x2 413.0325 1132.8000 376.2150  
## x3 184.2658 376.2150 180.7867

# SSPres + SSPint from the previouys part  
SSPres+SSPint

## x1 x2 x3  
## x1 309.3067 413.0325 184.2658  
## x2 413.0325 1132.8000 376.2150  
## x3 184.2658 376.2150 180.7867

What I have observed is that SSPfac1 and SSPfac2 in this part and part b are the same. SSprese is equal to SSPres + SSPint from part with interaction of fac1 and fac2.