LSD ANOVA with missing values

Pranav Gopalkrishna, 1940223

2021-11-08

# INTRO TO LSD

Latin square design or LSD is an experimental design. Hence, it is applied for the data collection process, not the analysis. To do this, we must have at least three factors and a response. For best results, all factors must have equal number of levels. Two of these factors may be usable in classify the experimental units. If yes, then these are the blocking factors. Together, they form a grid for experimental units to sit in. The third factor is considered as the treatment, and each level is applied randomly to the experimental units so that every cell of th grid i.e. blocking factor combo is subjected to every treatment at only once. It is assumed that there is no interaction between any of the factors.

# **DATA SET**

This dataset is for basic data analysis, and contains basic information about various different computer models, including RAM size, screen width and price.

I want to find out how a computer’s screen size can affect the computer’s price on average.

I pick RAM size as the blocking factor since it is generally the most significant factor in determining the price of the computer. Hence, we would expect there to be substantial mean price differences between groups of computers having different RAM sizes, implying relatively more homogeneity in prices within each group (this is visually confirmed in the box plots below).

setwd("~/Documents/Study/computerScience/programming/r/data/")  
myData = read.csv("computersBasicsWithMissingValues.csv")  
head(myData)

## X price speed hd ram screen cd multi premium ads trend  
## 1 1 1499 25 80 4 14 no no yes 94 1  
## 2 2 1795 33 85 2 14 no no yes 94 1  
## 3 3 1595 25 170 4 15 no no yes 94 1  
## 4 4 1849 25 170 8 14 no no no 94 1  
## 5 5 3295 33 340 16 14 no no yes 94 1  
## 6 6 NA 66 340 16 14 no no yes 94 1

y = myData$price  
bf\_1 = as.factor(myData$ram) # Blocking factor

RAM size is chosen as a blocking factor since it is generally the most significant factor in determining the price of the computer. Hence, we would expect there to be substantial mean price differences between groups of computers having different RAM sizes.

bf\_2 = myData$premium

Whether or not a model is premium obviously affects its price. Hence, this is the second blocking factor.

tr = as.factor(myData$screen) # Treatment

Hence, we have

***treatment****: screen size*

***blocking factor 1****: RAM size*

***blocking factor 2****: premium (yes/no)*

# FUNCTIONS

***Functions to sum all prices for a certain level of a factor.***

categorySum = function(searchRange, category, sumRange)  
{  
 n = 1  
 sum = 0  
 for(x in searchRange)  
 {  
 if(x == category & !is.na(sumRange[n])){sum = sum + sumRange[n]}  
 n = n + 1  
 }  
 return(sum)  
}  
categoryMean = function(searchRange, category, sumRange)  
{  
 n = 1  
 sum = 0  
 notNullCount = 0  
 for(x in searchRange)  
 {  
 if(x == category & !is.na(sumRange[n]))  
 {  
 sum = sum + sumRange[n]  
 notNullCount = notNullCount + 1  
 }  
 n = n + 1  
 }  
 return(sum/notNullCount)  
}  
categoryCount = function(searchRange, category)  
{  
 return(sum(searchRange == category))  
}  
categoryCountNotNull = function(searchRange1, category, searchRange2)  
{  
 return(sum(searchRange1 == category & !is.null(searchRange2)))  
}

***Missing value estimation functions.***

missingValueEstimate = function(t, b1, b2, r, i)  
{  
 # r is the observed response values.  
 # i is the index of the missing value in the response that you want to estimate.  
 # t is the treatment levels corresponding to each response.  
 # b1 is the 1st category of blocks corresponding to each response.  
 # b2 is the 2nd category of blocks corresponding to each response.  
 treatmentMean = categoryMean(t, t[i], r)  
 block1\_mean = categoryMean(b1, b1[i], r)  
 block2\_mean = categoryMean(b2, b2[i], r)  
 return((treatmentMean + block1\_mean + block2\_mean) / 2)  
}  
getMissingValueIndices = function(r)  
{  
 mvi = c() # Missing value indices  
 max = length(r) # Total number of observations.  
 for(i in c(1:max)){if(is.na(r[i])){mvi = c(mvi, i)}}  
 return(mvi)  
}  
missingValueEstimation = function(t, b1, b2, r)  
{  
 # Getting the indices of all missing values...  
 mvi = getMissingValueIndices(r)  
 #------------------------  
 # Initial estimation for all missing values but one...  
 rCopy = r # This is important, as it helps us ensure that previous initial estimates must not affect other intial estiamtes  
 for(i in mvi[2:length(mvi)]){r[i] = missingValueEstimate(t, b1, b2, rCopy, i)}  
 # We have left one missing value at the start.  
 # A missing value is left at the start and not at the end for convenience in coding in the final estimation.  
 #------------------------  
 # Final estimation for all missing values...  
 for(i in mvi)  
 {  
 r[i] = NA  
 r[i] = missingValueEstimate(tr, b1, b2, r, i)  
 }  
 return(r)  
}  
hasMissingValues = function(sov, r)  
{  
 i = 1  
 for(x in sov)  
 {  
 if(is.na(r[i]))  
 {  
 return(TRUE)  
 }  
 i = i + 1  
 }  
 return(FALSE)  
}

# MISSING VALUE ESTIMATION

If the data set has missing values, we will estimate them.

if(hasMissingValues(tr, y))  
{  
 #------------------------  
 # Getting all the missing values' indices  
 mvi = getMissingValueIndices(y)  
 #------------------------  
 # Showing all the missing value rows  
 for(i in mvi)  
 {  
 print(paste("Index", i, ", tr:", tr[i], ", bf\_1:", bf\_1[i], ", bf\_2:", bf\_2[i]))  
 # Note that tr is screen size, bf\_1 is RAM capacity and bf\_2 is whether or not the product is premium.  
 }  
 #------------------------  
 # Estimating the missing values  
 y = missingValueEstimation(tr, bf\_1, bf\_2, y)  
 #------------------------  
 #' Showing all the estimated missing values  
 for(i in mvi)  
 {  
 print(paste("Index", i, ", y:", y[i]))  
 # Note that tr is screen size, bf\_1 is RAM capacity and bf\_2 is whether or not the product is premium.  
 }  
}

**MISSING VALUE ROWS**

## [1] "Index 6 , tr: 14 , bf\_1: 16 , bf\_2: yes"  
## [1] "Index 18 , tr: 15 , bf\_1: 4 , bf\_2: yes"  
## [1] "Index 33 , tr: 17 , bf\_1: 8 , bf\_2: yes"  
## [1] "Index 38 , tr: 14 , bf\_1: 8 , bf\_2: yes"  
## [1] "Index 52 , tr: 14 , bf\_1: 4 , bf\_2: yes"  
## [1] "Index 101 , tr: 14 , bf\_1: 4 , bf\_2: yes"  
## [1] "Index 410 , tr: 14 , bf\_1: 16 , bf\_2: yes"  
## [1] "Index 433 , tr: 14 , bf\_1: 16 , bf\_2: yes"

**ESTIMATED MISSING VALUES**

## [1] "Index 6 , y: 3536.05954161876"  
## [1] "Index 18 , y: 3214.27084594255"  
## [1] "Index 33 , y: 3557.42747632303"  
## [1] "Index 38 , y: 3293.34652334805"  
## [1] "Index 52 , y: 3081.29086905404"  
## [1] "Index 101 , y: 3081.29185843162"  
## [1] "Index 410 , y: 3536.06280874299"  
## [1] "Index 433 , y: 3536.06447468304"

# SPECIFIC DATA FRAME FOR THE CHOSEN FIELDS

**(INCLUDING ESTIMATED MISSING VALUES)**

data = data.frame(tr, bf\_1, bf\_2, y)  
head(data)

## tr bf\_1 bf\_2 y  
## 1 14 4 yes 1499.00  
## 2 14 2 yes 1795.00  
## 3 15 4 yes 1595.00  
## 4 14 8 no 1849.00  
## 5 14 16 yes 3295.00  
## 6 14 16 yes 3536.06

# ANOVA TEST

Hypotheses (for all factors in general)

**H0: Factor's level's means are not significantly different  
H1: Factor's level's means are significantly different**

model = lm(y~., data)  
aovModel = aov(model)  
summary(aovModel)

## Df Sum Sq Mean Sq F value Pr(>F)   
## tr 2 1.953e+08 97671554 576.4 <2e-16 \*\*\*  
## bf\_1 5 7.553e+08 151055050 891.4 <2e-16 \*\*\*  
## bf\_2 1 1.065e+08 106535523 628.7 <2e-16 \*\*\*  
## Residuals 6250 1.059e+09 169460   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

As we can see, all factors, including the blocking factors, have significant impact on prices. Note that Pr(>F) is the probability of getting the particular calculated F value for that factor, given a standard F distribution. Essentially, we are checking if the particular F value can be said to be present in the F distribution (with 0.05 level of significance).

F value is a statistic drawn using the variation between the means of the various levels in each factor individually, and the smaller this variance is, the more likely it is that the F statistic will lie in the 95% confidence interval of the F distribution.

# CONCLUSIONS

Since the p-value for the treatment (screen size) is below the significance level 0.05, we may reject the null hypothesis for screen size. Hence, we must perform a post-hoc analysis to determine which levels of the treatment i.e. which screen sizes have significantly different mean prices.

# POST HOC ANALYSIS

library(lsmeans)

## Loading required package: emmeans

## Warning: package 'emmeans' was built under R version 3.6.2

## The 'lsmeans' package is now basically a front end for 'emmeans'.  
## Users are encouraged to switch the rest of the way.  
## See help('transition') for more information, including how to  
## convert old 'lsmeans' objects and scripts to work with 'emmeans'.

pairs(lsmeans(aovModel, "tr"))

## contrast estimate SE df t.ratio p.value  
## 14 - 15 -17.2 12.1 6250 -1.424 0.3287  
## 14 - 17 -329.7 18.3 6250 -17.987 <.0001  
## 15 - 17 -312.4 19.2 6250 -16.256 <.0001  
##   
## Results are averaged over the levels of: bf\_1, bf\_2   
## P value adjustment: tukey method for comparing a family of 3 estimates

Given a 0.05 significance level, only screen sizes 14 and 15 don’t have significantly different price levels.