Randomized Block Design & ANOVA With Missing Values

Estimating which screen sizes have significantly different average prices, with RAM size as the blocking factor

Pranav Gopalkrishna, 1940223

2021-09-15

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 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).

Hence,  
**treatment (*tr*) = screen  
blocking factor (*bf*) = ram  
*y* = price**

This data set also has 8 missing values, which I will estimate and modify the ANOVA to correct for possible biases.

myData = read.csv("~/Documents/Study/computerScience/programming/r/data/computersBasicsWithMissingValues.csv")[c(2, 5, 6)]  
head(myData)

## price ram screen  
## 1 1499 4 14  
## 2 1795 2 14  
## 3 1595 4 15  
## 4 1849 8 14  
## 5 3295 16 14  
## 6 NA 16 14

y = myData$price  
bf = as.factor(myData$ram) # Blocking factor  
tr = as.factor(myData$screen) # Treatment

# FUNCTIONS

Some user defined functions that I will be using in the program...

***# To calculate the sum, mean & count of observations of a certain level...***

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)))  
}

***# To check for and estimate missing values...***

missingValueEstimate = function(t, b, 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.  
 # b is the blocks corresponding to each response.  
 treatmentMean = categoryMean(t, t[i], r)  
 blockMean = categoryMean(b, b[i], r)  
 return((treatmentMean + blockMean) / 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, b, 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 the initial estimates must not account for estimated missing values.  
 for(i in mvi[2:length(mvi)]){r[i] = missingValueEstimate(t, b, 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, bf, r, i)  
 }  
 return(r)  
}  
hasMissingValues = function(sov, level, r)  
{  
 i = 1  
 for(x in sov)  
 {  
 if(is.na(r[i]))  
 {  
 return(TRUE)  
 }  
 i = i + 1  
 }  
 return(FALSE)  
}

***# To calculate the sum of squares for different sources of variation...***

sumOfSquares = function(sovLevels, sov, r, cf)  
{  
 # sov is source of variation.  
 # r is the observed response values.  
 # mv the number of missing values.  
 # sovDf is the degrees of freedom for the sov.  
 # totalDf is the total degrees of freedom.  
 # errorDf is the error degrees of freedom.  
  
 # Treatment sum of squares...  
 ss = 0 # Sum of squares.  
 for(x in sovLevels)  
 {  
 n = categoryCount(sov, x) # No. of observations per level.  
 ss = ss + categorySum(sov, x, r)^2 / n  
 }  
 ss = ss - cf  
 print(ss)  
 return(ss)  
}

***# To perform post hoc analysis and compare each pair of levels...***

pairMeanDifferences = function(sov, sovLevels, r, errorDf)  
{  
 i = 1  
 j = 1  
 dStdError = 0  
 while(i < length(sovLevels))  
 {  
 j = i + 1  
 while(j <= length(sovLevels))  
 {  
 print(c(i, j))  
 d = abs(categoryMean(sov, sovLevels[i], y) - categoryMean(sov, sovLevels[j], y))  
 a = categoryCountNotNull(sov, sovLevels[i], y)  
 b = categoryCountNotNull(sov, sovLevels[j], y)  
 dStdError = sqrt(mess\*(1/a + 1/b))  
 cd = abs(qt(0.05, errorDf)) \* dStdError  
 print(paste(d, ">", cd, "?", d > cd))  
 j = j + 1  
 print("----")  
 }  
 i = i + 1  
 }  
}

# ESTIMATING MISSING VALUES

# The number of missing values are...  
sum(is.na(y))

## [1] 8

Estimating all missing values using the above functions

y = missingValueEstimation(tr, bf, y)  
# Confirming no missing values remain...  
sum(is.na(y))

## [1] 0

# Making a new data frame with these values...  
newData = data.frame(tr, bf, y)  
head(newData)

## tr bf y  
## 1 14 4 1499.000  
## 2 14 2 1795.000  
## 3 15 4 1595.000  
## 4 14 8 1849.000  
## 5 14 16 3295.000  
## 6 14 16 2431.833

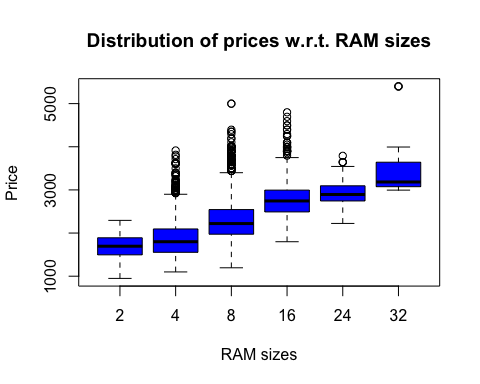
# VISUALISING THE DATA

newData = data.frame(tr, bf, y)  
summary(newData)

## tr bf y   
## 14:3661 2 : 394 Min. : 949   
## 15:1992 4 :2236 1st Qu.:1794   
## 17: 606 8 :2320 Median :2144   
## 16: 996 Mean :2219   
## 24: 297 3rd Qu.:2595   
## 32: 16 Max. :5399

## Visualising distribution of prices w.r.t. blocks i.e. RAM sizes…

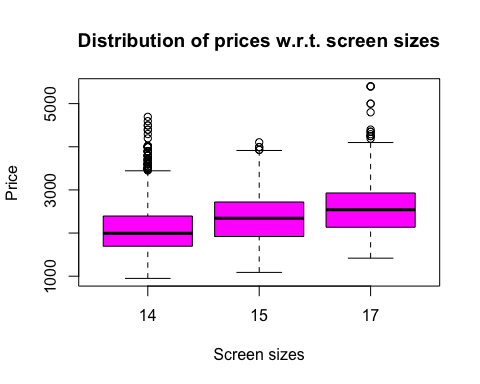
boxplot(y~bf, main = "Distribution of prices w.r.t. RAM sizes", xlab = "RAM sizes", ylab = "Price", col = "blue")



Here we see that each RAM has quite a visually distinct range of prices from other RAM sizes. This indicates homogeneity within blocks, and heterogeneity between blocks.

## Visualising distribution of prices w.r.t. treatment i.e. screen sizes…

boxplot(y~tr, main = "Distribution of prices w.r.t. screen sizes", xlab = "Screen sizes", ylab = "Price", col = "magenta")



# ANOVA TEST

## Hypotheses

First, we perform the test to see whether or not the treatment levels and the blocking factors have at least some significantly different means or not.

**For treatment levels**

H0: The means for all treatment levels are equal.

H1: The means of at least two treatment levels are unequal.

**For blocks**

H0: The means for all blocks are equal.

H1: The means of at least two blocks are unequal.

***Our level of significance will be 0.05.***

## ANOVA using user defined code and functions...

# Degrees of freedom and other basic measures...  
trLevels = unique(tr)  
bfLevels = unique(bf)

mvi = getMissingValueIndices(myData$price)  
totalDf = length(y) - length(mvi)  
trDf = length(trLevels) - 1  
bfDf = length(bfLevels) - 1  
errorDf = trDf\*bfDf - length(mvi)  
cf = sum(y)^2 / length(y)  
  
# Total sum of squares  
tss = sum(y^2) - cf  
  
# Treatment sum of squares...  
trss = sumOfSquares(trLevels, tr, y, cf)

## [1] 195991646

mtrss = trss / trDf  
  
# Block sum of squares...  
bfss = sumOfSquares(bfLevels, bf, y, cf)

## [1] 885297254

mbfss = bfss / bfDf  
  
# Error sum of squares...  
ess = tss - trss - bfss  
mess = ess / errorDf

## F statistics...

# Calculated...  
Ftr = mtrss / mess  
Fbl = mbfss / mess  
# Critical...  
Ftr\_crit = qf(0.05, trDf, errorDf)  
Fbl\_crit = qf(0.05, bfDf, errorDf)  
# Comparison  
Ftr > Ftr\_crit

## [1] TRUE

Fbl > Fbl\_crit

## [1] TRUE

# CONCLUSIONS

Both blocking factor and treatment are estimated to have significant differences between means of different pairs of levels.

**Hence, we reject null hypothesis for both treatment and blocking factor.**

# POST HOC ANALYSIS

pairMeanDifferences(tr, trLevels, y, errorDf)

## [1] 1 2  
## [1] "267.459958237114 > 1840.17075529048 ? FALSE"  
## [1] "----"  
## [1] 1 3  
## [1] "529.540181500884 > 2898.60013504888 ? FALSE"  
## [1] "----"  
## [1] 2 3  
## [1] "262.080223263769 > 3066.20902387744 ? FALSE"  
## [1] "----"

pairMeanDifferences(bf, bfLevels, y, errorDf)

## [1] 1 2  
## [1] "169.523320518496 > 3611.24561016449 ? FALSE"  
## [1] "----"  
## [1] 1 3  
## [1] "424.625369224151 > 1958.73249771829 ? FALSE"  
## [1] "----"  
## [1] 1 4  
## [1] "908.117952266069 > 2517.86980933647 ? FALSE"  
## [1] "----"  
## [1] 1 5  
## [1] "1739.28537237491 > 16582.5557231465 ? FALSE"  
## [1] "----"  
## [1] 1 6  
## [1] "1063.23970739174 > 4081.94016024721 ? FALSE"  
## [1] "----"  
## [1] 2 3  
## [1] "594.148689742647 > 3601.43832220832 ? FALSE"  
## [1] "----"  
## [1] 2 4  
## [1] "1077.64127278456 > 3933.62344001317 ? FALSE"  
## [1] "----"  
## [1] 2 5  
## [1] "1908.8086928934 > 16855.7076179646 ? FALSE"  
## [1] "----"  
## [1] 2 6  
## [1] "1232.76302791024 > 5078.97238256834 ? FALSE"  
## [1] "----"  
## [1] 3 4  
## [1] "483.492583041917 > 2503.78343881944 ? FALSE"  
## [1] "----"  
## [1] 3 5  
## [1] "1314.66000315075 > 16580.4227160527 ? FALSE"  
## [1] "----"  
## [1] 3 6  
## [1] "638.614338167589 > 4073.26633103918 ? FALSE"  
## [1] "----"  
## [1] 4 5  
## [1] "831.167420108837 > 16655.7333318635 ? FALSE"  
## [1] "----"  
## [1] 4 6  
## [1] "155.121755125672 > 4369.72928025613 ? FALSE"  
## [1] "----"  
## [1] 5 6  
## [1] "676.045664983165 > 16962.7833776541 ? FALSE"  
## [1] "----"