2019ADS2 Week5 DataCleaning ProblemSet

by Wanlu Liu, wanluliu@intl.zju.edu.cn (mailto:wanluliu@intl.zju.edu.cn)

2019-10-14

1. Introduction

This R Markdown file contains a tutorial of how to do data cleaning with R. The input data is orignally from R package ggplot2 (diamonds dataset). It contains the ID,carat,cut, color, clarity, depth, table, price, x,y,z for more than 50,000 diamonds. In order to do this demo, I randomly sampled 100 diamonds from the orignally dataset (with set.seed=12) and then modifed some of the entries.

2. Setting up working directory

Before we start, we need to set up our working directory.**This is a absolute or relative directory?**

```
setwd("/Users/wanluliu/Desktop/ADS2Week5/") #make sure change to your own corrected paths
```

3. Import data

The next thing we need to do is import our data. We have learnt in the lecture, there are serval function to import data (including: read.delim, read.delim2, read.csv, read.csv2). Since my data is in xxx.csv format, so I use **read.csv()** function

```
data=read.csv("Rdata_diamonds_samples100_mdf.csv")
head(data)
```

```
##   ID carat       cut color clarity depth table price    x    y    z
## 1 58  0.24 Very Good     G    VVS2  62.0    56   449 4.00 4.03 2.49
## 2  8  0.27 Very Good     D     VS1  60.4    59   470 4.15 4.20 2.52
## 3 53  0.29 Very Good     E    VVS1  60.9    61   629 4.23 4.27 2.59
## 4  1  0.30   Premium     E     VS2  61.7    60   570 4.28 4.31 2.65
## 5 42  0.30     Ideal     G     VS2  63.0    55   675 4.31 4.29 2.71
## 6 80  0.30     Ideal     H     SI1  62.2    53  1105 4.29 4.32 2.68
```

4. R data types

4.1 data types

Before we start data cleaning, let's get more ideas about R data types.

```
#check the data type of ID column
head(data$ID)
```

```
## [1] 58  8 53  1 42 80
```

```
typeof(data$ID)
```

```
## [1] "integer"
```

```
class(data$ID)
```

```
## [1] "integer"
```

```
#check the data type of carat column
head(data$carat)
```

```
## [1] 0.24 0.27 0.29 0.30 0.30 0.30
```

```
class(data$carat)
```

```
## [1] "numeric"
```

```
#check the data type of cut column
head(data$cut)
```

```
## [1] Very Good Very Good Very Good Premium   Ideal     Ideal
## Levels: Fair Good Idea Ideal Premium Very Good
```

```
class(data$cut)
```

```
## [1] "factor"
```

```
#What's the difference between -
#character and factor?
cut.chr=as.character(data$cut)
head(cut.chr)
```

```
## [1] "Very Good" "Very Good" "Very Good" "Premium"   "Ideal"     "Ideal"
```

```
class(cut.chr)
```

```
## [1] "character"
```

4.2 numeric vs integer?

Also, be aware of the difference between numeric vs integer

```
var1=c(2,3,5,6)
class(var1)
```

```
## [1] "numeric"
```

```
var2=c(2.0,3.9,5.1,6.9)
class(var2)
```

```
## [1] "numeric"
```

```
var3=c(2L,3L,5L,6L)
class(var3)
```

```
## [1] "integer"
```

5. R data structure

5.1 data structure

5.1.1 numeric vectors

Before we start data cleaning, let's get more ideas about R data structure. First let's create some *numeric vectors*.

```
############R data structure vector vs list
#create numeric vector
var.num=c(2.0,3.9,5.1,6.9)
print(var.num)
```

```
## [1] 2.0 3.9 5.1 6.9
```

```
class(var.num)
```

```
## [1] "numeric"
```

5.1.2 integer vectors

Then let's create some *integer vectors*.

```
#create integer vector
var.int=c(2L,3L,5L,6L)
print(var.int)
```

```
## [1] 2 3 5 6
```

```
class(var.int)
```

```
## [1] "integer"
```

5.1.3 character vectors

Then let's try to create some *character vectors*.

```
#create character vector
var.char=c("Hello",",","world","!")
class(var.char)
```

```
## [1] "character"
```

```
print(var.char)
```

```
## [1] "Hello" ","     "world" "!"
```

5.1.4 factor vectors

Then let's try to create some *factor vectors*.

```
#create factor vector
var.fac=factor(c("mid","mid","high","low"),
               levels=c("low","mid","high"))
print(var.fac)
```

```
## [1] mid  mid  high low
## Levels: low mid high
```

```
class(var.fac)
```

```
## [1] "factor"
```

5.1.5 list

Finally, let combine all the vectors we just created, to generate a *list*.

```
#combine all vector above to create list
data.list=list(var.num,var.int,var.char,var.fac)
head(data.list)
```

```
## [[1]]
## [1] 2.0 3.9 5.1 6.9
##
## [[2]]
## [1] 2 3 5 6
##
## [[3]]
## [1] "Hello" ","     "world" "!"
##
## [[4]]
## [1] mid  mid  high low
## Levels: low mid high
```

```
data.list[[1]]
```

```
## [1] 2.0 3.9 5.1 6.9
```

```
data.list[[1]][1]
```

```
## [1] 2
```

```
data.list[[3]]
```

```
## [1] "Hello" ","     "world" "!"
```

```
data.list[[3]][3]
```

```
## [1] "world"
```

```
length(data.list)
```

```
## [1] 4
```

```
dim(data.list)
```

```
## NULL
```

5.2 matrix vs dataframe?

What's the difference between **matrix vs data frame**?

```
#convert dataframe to matrix
data.matrix=as.matrix(data)
#look at the header of the matrix
head(data.matrix)
```

```
##      ID    carat  cut         color clarity depth  table  price   x
## [1,] " 58" "0.24" "Very Good" "G"   "VVS2"  "62.0" "56.0" "  449" "4.00"
## [2,] "  8" "0.27" "Very Good" "D"   "VS1"   "60.4" "59.0" "  470" "4.15"
## [3,] " 53" "0.29" "Very Good" "E"   "VVS1"  "60.9" "61.0" "  629" "4.23"
## [4,] "  1" "0.30" "Premium"   "E"   "VS2"   "61.7" "60.0" "  570" "4.28"
## [5,] " 42" "0.30" "Ideal"     "G"   "VS2"   "63.0" "55.0" "  675" "4.31"
## [6,] " 80" "0.30" "Ideal"     "H"   "SI1"   "62.2" "53.0" " 1105" "4.29"
##      y      z
## [1,] "4.03" "2.49"
## [2,] "4.20" "2.52"
## [3,] "4.27" "2.59"
## [4,] "4.31" "2.65"
## [5,] "4.29" "2.71"
## [6,] "4.32" "2.68"
```

```
#check the matrix class
class(data.matrix[,1])
```

```
## [1] "character"
```

```
class(data.matrix[,2])
```

```
## [1] "character"
```

6. Data Cleaning

Now let's try to clean our data. Still remember the **screen-diagnosis-treat-document** rules?

6.1 Missing data

6.1.1 Screen-Diagnosis

We first want to screen the missing data. We want to know, how the missing data is coded? (NA or NAN or ND or blank)? which rows have a missing data? which column have a missing data?

```
head(data)
```

```
##   ID carat       cut color clarity depth table price    x    y    z
## 1 58  0.24 Very Good     G    VVS2  62.0    56   449 4.00 4.03 2.49
## 2  8  0.27 Very Good     D     VS1  60.4    59   470 4.15 4.20 2.52
## 3 53  0.29 Very Good     E    VVS1  60.9    61   629 4.23 4.27 2.59
## 4  1  0.30   Premium     E     VS2  61.7    60   570 4.28 4.31 2.65
## 5 42  0.30     Ideal     G     VS2  63.0    55   675 4.31 4.29 2.71
## 6 80  0.30     Ideal     H     SI1  62.2    53  1105 4.29 4.32 2.68
```

```
head(is.na(data))
```

```
##         ID carat   cut color clarity depth table price     x     y     z
## [1,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [4,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [5,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [6,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
tail(data)
```

```
##     ID carat     cut color clarity depth table price    x    y    z
## 98  14  1.56 Premium     G     SI2  61.7    59  8858 7.51 7.43 4.61
## 99  27  1.57   Ideal     E     VS2  60.5    57 17548 7.53 7.57 4.57
## 100 39  1.59    <NA>     D     SI2  61.6    55  8975 7.48 7.45 4.60
## 101 52  1.67 Premium     H     SI2  62.6    60  8118 7.57 7.52 4.72
## 102 29  2.19    Good     I     SI2  63.7    57 11756 8.23 8.19 5.23
## 103 17    NA   Ideal     G     VS1  60.6    57 13034 7.47 7.36 4.49
```

```
tail(is.na(data))
```

```
##           ID carat   cut color clarity depth table price     x     y     z
##  [98,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [99,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100,] FALSE FALSE  TRUE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [101,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [102,] FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [103,] FALSE  TRUE FALSE FALSE   FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
apply(is.na(data),2,which) #this is to find the row,col of NA
```

```
## $ID
## integer(0)
##
## $carat
## [1] 103
##
## $cut
## [1]  60 100
##
## $color
## [1] 60
##
## $clarity
## [1] 19 42 60 92
##
## $depth
## [1] 55 60
##
## $table
## [1] 26 60
##
## $price
## [1] 49 60 69 72
##
## $x
## [1] 60
##
## $y
## [1] 60
##
## $z
## [1] 60
```

Looks like our missing data is coded with NA. And we don't know how to handle those missing data (no data to fill it up), so the treat we would like to take is to remove those entries.

6.1.2 Treat

```
dim(data)
```

```
## [1] 103  11
```

```
data.noNA=data[complete.cases(data),]
dim(data.noNA)
```

```
## [1] 92 11
```

After removing NA-containing rows, the number of rows change from 103 to 92. In order to documented this precisely, we need to know which rows were deleted.

6.1.3 Documentation

In this dataset, Wanlu found those entries contains missing data. Since there is no data or clues of how to fill up those missing data, Wanlu decidec to delete those data points. Before deleting those rows, in total there are 103 observations (rows), while only 92 remained after the cleaning.

```
print(data[!complete.cases(data),])
```

```
##     ID carat       cut color clarity depth table price    x    y    z
## 19  85  0.37      Idea     F    <NA>  61.7    56  1041 4.63 4.61 2.85
## 26  97  0.42     Ideal     E    VVS2  62.3    NA  1216 4.76 4.81 2.98
## 42  45  0.54     Ideal     F    <NA>  59.8    58  1680 5.29 5.34 3.18
## 49  56  0.72      Good     H     VS2  61.8    61    NA 5.68 5.75 3.53
## 55  66  0.90 Very Good     F     SI2    NA    61  4441 6.14 6.18 3.76
## 60  49  1.00      <NA>  <NA>    <NA>    NA    NA    NA   NA   NA   NA
## 69  90  1.02      Good     D     SI2  57.5    62    NA 6.60 6.62 3.80
## 72  71  1.03   Premium     G     VS1  60.7    58    NA 6.55 6.50 3.96
## 92  54  1.33   Premium     I    <NA>  61.5    58  6963 7.02 7.06 4.33
## 100 39  1.59      <NA>     D     SI2  61.6    55  8975 7.48 7.45 4.60
## 103 17    NA     Ideal     G     VS1  60.6    57 13034 7.47 7.36 4.49
```

6.2 Duplicated data

6.2.1 Screen-Diagnosis

We then want to screen the duplicated data. We want to know whether there is **duplicated rows** in the dataset?

```
duplicated(data.noNA)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [56]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [89] FALSE FALSE FALSE FALSE
```

```
frw.idx=which(duplicated(data.noNA)) #duplciated() will only give you the duplicated rows, but
 not the original rows, so we need the next line to get the originals
rvs.idx=which(duplicated(data.noNA,fromLast = TRUE))
data.noNA[c(frw.idx,rvs.idx),]
```

```
##    ID carat       cut color clarity depth table price    x    y    z
## 39 50  0.52 Very Good     G     SI1  59.6    62  1244 5.18 5.23 3.10
## 62 93  1.00     Ideal     E     SI1  62.6    56  4480 6.37 6.29 3.96
## 65 26  1.01     Ideal     D     SI1  62.0    57  5832 6.37 6.44 3.97
## 38 50  0.52 Very Good     G     SI1  59.6    62  1244 5.18 5.23 3.10
## 61 93  1.00     Ideal     E     SI1  62.6    56  4480 6.37 6.29 3.96
## 64 26  1.01     Ideal     D     SI1  62.0    57  5832 6.37 6.44 3.97
```

From this results, we can see *ID=50, 93, 26* are lines duplicated. So we want to delete the duplicated copy.

6.2.2 Treat

Since those duplicated entry are obviously error, we need to delete them from the table.

```
dim(data.noNA)
```

```
## [1] 92 11
```

```
data.noNA.noDup=data.noNA[!duplicated(data.noNA),]
dim(data.noNA.noDup)
```

```
## [1] 89 11
```

From this step, you can see, after removing the duplcated rows, the dimension of the dataframe decrease from 92 to 89.

6.2.3 Documentation

In this dataset, Wanlu found three entries (listed below) are duplicated.Wanlu decided to delete those duplicated data points. After deleting those duplicated rows, there are 89 observations left.

```
data.noNA[duplicated(data.noNA),]
```

```
##    ID carat       cut color clarity depth table price    x    y    z
## 39 50  0.52 Very Good     G     SI1  59.6    62  1244 5.18 5.23 3.10
## 62 93  1.00     Ideal     E     SI1  62.6    56  4480 6.37 6.29 3.96
## 65 26  1.01     Ideal     D     SI1  62.0    57  5832 6.37 6.44 3.97
```

6.3 Strange pattern

After removing the missing data and duplicated data, we now want to see whether there is any outliers or strange patterns. I will leave the outlier investigation in the problem set. Let's see there is any strange pattern together.

6.3.1 Screen

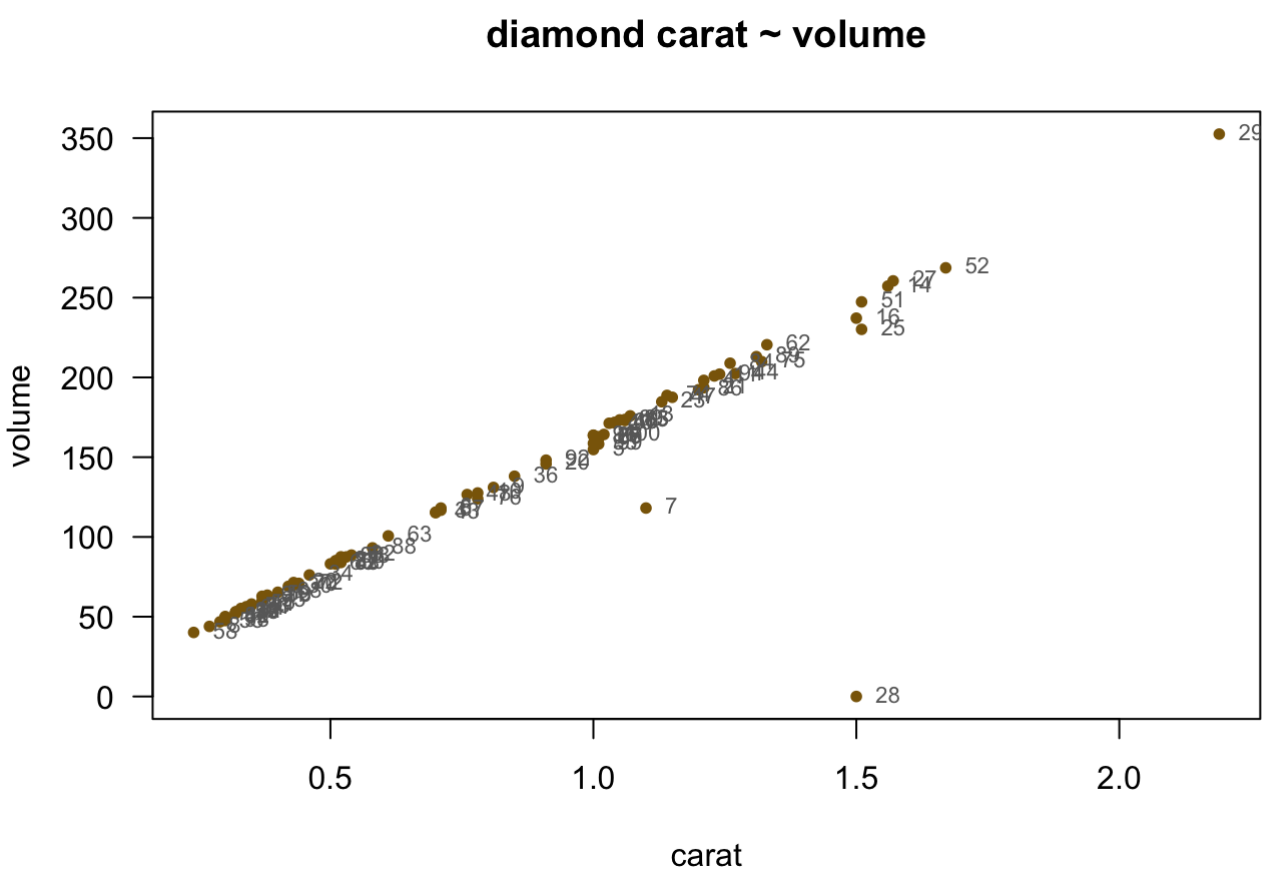
We know that diamond and it's volume have a linear relationship. Thus, we would like to investigate the relationship between carat vs. volume. In order to test this idea, we need to generate a new vector called volume=x*y*z.

```
data.noNA.noDup=data.frame(data.noNA.noDup,volume=data.noNA.noDup$x*data.noNA.noDup$y*data.noN
A.noDup$z)
head(data.noNA.noDup)
```

```
##   ID carat       cut color clarity depth table price    x    y    z
## 1 58  0.24 Very Good     G    VVS2  62.0    56   449 4.00 4.03 2.49
## 2  8  0.27 Very Good     D     VS1  60.4    59   470 4.15 4.20 2.52
## 3 53  0.29 Very Good     E    VVS1  60.9    61   629 4.23 4.27 2.59
## 4  1  0.30   Premium     E     VS2  61.7    60   570 4.28 4.31 2.65
## 5 42  0.30     Ideal     G     VS2  63.0    55   675 4.31 4.29 2.71
## 6 80  0.30     Ideal     H     SI1  62.2    53  1105 4.29 4.32 2.68
##     volume
## 1 40.13880
## 2 43.92360
## 3 46.78084
## 4 48.88402
## 5 50.10763
## 6 49.66790
```

we then plot scatterplot of carat vs volume to see whether they have a linear relationship.

```
plot(x=data.noNA.noDup$carat,y=data.noNA.noDup$volume,
     pch=20,col="darkgoldenrod4",
     las=1,xlab="carat",ylab="volume",
     main="diamond carat ~ volume")
text(data.noNA.noDup$carat, data.noNA.noDup$volume,
     labels=data.noNA.noDup$ID,col="dimgray",
     cex= 0.7, pos=4)
```



6.3.2 Diagnosis

Here we found two strange data point, ID=7 and ID=28. We decide to take a look at those two IDs.

```
print(data.noNA.noDup[which(data.noNA.noDup$ID=="7"),])
```

```
##    ID carat     cut color clarity depth table price    x    y    z
## 79  7   1.1 Premium     D     SI2  62.4    58  4640 5.74 5.78 3.56
##      volume
## 79 118.1108
```

```
print(data.noNA.noDup[which(data.noNA.noDup$ID=="28"),])
```

```
##    ID carat  cut color clarity depth table price    x    y z volume
## 95 28   1.5 Good     G      I1    64    61  4731 7.15 7.04 0      0
```

From this results, we found ID=28 have a z dimension =0, which is definitely wrong (There is no real TWO-Dimension diamond :). If we cannot find the correct data, we need to delete this data point.

How about ID=7? the x,y,z looks suspecious for ID=7, could we entered the data wrong for x.y.x? It indeed looks strange, so let's check whether there is any duplication of x y z. (Maybe we assigned some other diamonds' xyz to ID=7)?

Since x,y,z is stored in column 9 to 11, let's view the head of it first.

```
head(data.noNA.noDup[,9:11])
```

```
##      x    y    z
## 1 4.00 4.03 2.49
## 2 4.15 4.20 2.52
## 3 4.23 4.27 2.59
## 4 4.28 4.31 2.65
## 5 4.31 4.29 2.71
## 6 4.29 4.32 2.68
```

```
print(data.noNA.noDup[duplicated(data.noNA.noDup[,9:11],fromLast = "TRUE"),])
```

```
##    ID carat       cut color clarity depth table price    x    y    z
## 47  6  0.71 Very Good     E     VS1  61.8    56  3059 5.74 5.78 3.56
##      volume
## 47 118.1108
```

```
fwrd.dup.idx=which(duplicated(data.noNA.noDup[,9:11]))
rvse.dup.idx=which(duplicated(data.noNA.noDup[,9:11],fromLast = TRUE))
data.noNA.noDup[c(fwrd.dup.idx,rvse.dup.idx),]
```

```
##    ID carat       cut color clarity depth table price    x    y    z
## 79  7  1.10   Premium     D     SI2  62.4    58  4640 5.74 5.78 3.56
## 47  6  0.71 Very Good     E     VS1  61.8    56  3059 5.74 5.78 3.56
##      volume
## 79 118.1108
## 47 118.1108
```

From the results above, seems like ID=6 and ID=7 have exactly the same x,y,z so ID=7 is very likely be a errorous entry as well.

6.3.3 Treat

From our analysis on the carat vs volume, we found the data collection for ID=7 and ID=28 are both errorous. So we need to delete this two data point as well.

```
data.noNA.noDup.noStrg=data.noNA.noDup[-which(data.noNA.noDup$ID==7 | data.noNA.noDup$ID==28),]
dim(data.noNA.noDup)
```

```
## [1] 89 12
```

```
dim(data.noNA.noDup.noStrg)
```

```
## [1] 87 12
```

6.3.4 Documentation

When wanlu investigate the relationship between diamond carat and diamond volume, she found that ID=28 have no z dimension while ID=7 share the same x y z dimension with ID=6. She thinks this two entry is errorous during data collection or recording. So she decided to delete this two entry from the data. After removing those two strange pattern data point, there is 87 valid observations left.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).