Using R for data analysis: SSA

Boerhaave Nascholing

November 26th, 2020

Important note

The primary goal of the assignment is to write an R Markdown document containing **the code** which calculates the answers to the questions below. Use `Knit` button regularly to check that your code does not produce errors.

Diamonds dataset

For this SSA you use the `diamonds` dataset which contains various attributes of sold diamonds (see also `?diamonds`). The dataset comes with the `tidyverse` package. After you load the `tidyverse` library you'll have access to the dataset in the `diamonds` variable. Make sure you put `library(tidyverse)` in the R chunk at the top of your R Markdown file.

```
library( tidyverse )
diamonds
```

```
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10 0.23  Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ... with 53,930 more rows
```

Diamonds dataset tibble

Each row of the `diamonds` tibble describes one sold diamond. There are the following variables (columns):

- `price`: Price in US dollars.
- `carat`: Weight of the diamond (in carat units: 1 carat = 0.2g).
- `cut`: Quality of the cut (`Fair`, `Good`, `Very Good`, `Premium`, `Ideal`).
- `color`: Diamond colour, from `J` (worst) to `D` (best).

- `clarity`: How clear the diamond is (`I1` (worst), `SI2`, `SI1`, `VS2`, `VS1`, `VVS2`, `VVS1`, `IF` (best)).
- `x`, `y`, `z`: Length, width, depth. Each in mm.
- `depth`: Total depth percentage = z / mean(x, y) = 2 * z / (x + y).
- `table`: Width of top of diamond relative to widest point.

Questions

Q1. [0.5p] Show the type/class of the `diamonds` table. [0.5p] Show the type of the column `clarity`.

```
class(diamonds)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

```
class(diamonds$clarity)
```

```
[1] "ordered" "factor"
```

Q2. [1p] Show the structure of the `diamonds` table.

```
str(diamonds) # alternatively: glimpse(diamonds)
```

```
tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
 $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Q3. [1p] Print the rows 7-10 (hint: combine `head` and `tail`).

```
diamonds %>% head( 10 ) %>% tail( 4 )
```

```
# A tibble: 4 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
2  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
3  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
4  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
```

Q4. [1p] Calculate the mean of the `price` column.

```
mean(diamonds$price)
```

```
[1] 3932.8
```

Q5. [1p] Give the **number** of levels of the factor in the `clarity` column.

```
nlevels(diamonds$clarity) # alternatively: length(levels(diamonds$clarity))
```

```
[1] 8
```

Q6. [3p] Make a `list` with two elements calculated as follows from the `diamonds` table. Name the first list element `medianDepth` and set it to the median diamond depth. Name the second list element `clarities` and set it to the levels of the column `clarity`.

```
list(
  medianDepth = median( diamonds$depth ),
  clarities = levels( diamonds$clarity )
)
```

```
$medianDepth
[1] 61.8

$clarities
[1] "I1"   "SI2"  "SI1"  "VS2"  "VS1"  "VVS2" "VVS1" "IF"
```

Q7. Frequencies and cross table.

a) [1p] Count all the combinations of the value pairs in columns `cut` and `clarity`.

```
diamonds %>% count(cut,clarity)
```

```
# A tibble: 40 x 3
   cut   clarity     n
   <ord> <ord>   <int>
 1 Fair  I1        210
 2 Fair  SI2       466
 3 Fair  SI1       408
 4 Fair  VS2       261
 5 Fair  VS1       170
 6 Fair  VVS2       69
 7 Fair  VVS1       17
 8 Fair  IF          9
 9 Good  I1         96
10 Good  SI2      1081
# ... with 30 more rows
```

b) [2p] Print a **cross table** of `cut` and `clarity`, with `cut` categories given in columns.

```
diamonds %>% count(cut,clarity) %>% spread(cut,n)
```

```
# A tibble: 8 x 6
  clarity  Fair  Good `Very Good` Premium Ideal
  <ord>   <int> <int>       <int>   <int> <int>
1 I1        210    96          84     205   146
2 SI2       466  1081        2100    2949  2598
3 SI1       408  1560        3240    3575  4282
4 VS2       261   978        2591    3357  5071
5 VS1       170   648        1775    1989  3589
6 VVS2       69   286        1235     870  2606
7 VVS1       17   186         789     616  2047
8 IF          9    71         268     230  1212
```

Q8. [3p] Group the `diamonds` table by `color`. In each group calculate min, max, median and mean `price`.

```
diamonds %>%
  group_by(color) %>%
  summarise(
    minPrice=min(price),
    maxPrice=max(price),
    meanPrice=mean(price),
    medianPrice=median(price),
    .groups='drop'
  )
```

```
# A tibble: 7 x 5
  color minPrice maxPrice meanPrice medianPrice
  <ord>    <int>    <int>     <dbl>       <dbl>
1 D          357    18693     3170.        1838
2 E          326    18731     3077.        1739
3 F          342    18791     3725.        2344.
4 G          354    18818     3999.        2242
5 H          337    18803     4487.        3460
6 I          334    18823     5092.        3730
7 J          335    18710     5324.        4234
```

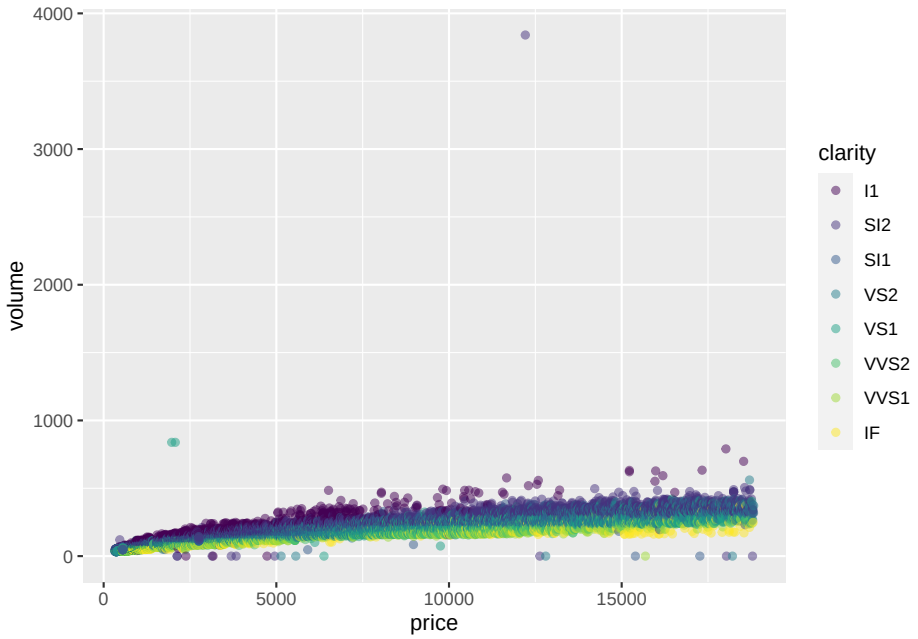
Q9. Diamond volume in a scatter plot.

a) [1p] Add a new column `volume` representing diamond's volume in cubic millimetres given the dimensions `x`, `y` and `z`. Store the tibble with the added column in the variable `diamonds_volume`.

```
diamonds_volume <- diamonds %>% mutate(volume=x*y*z)
```

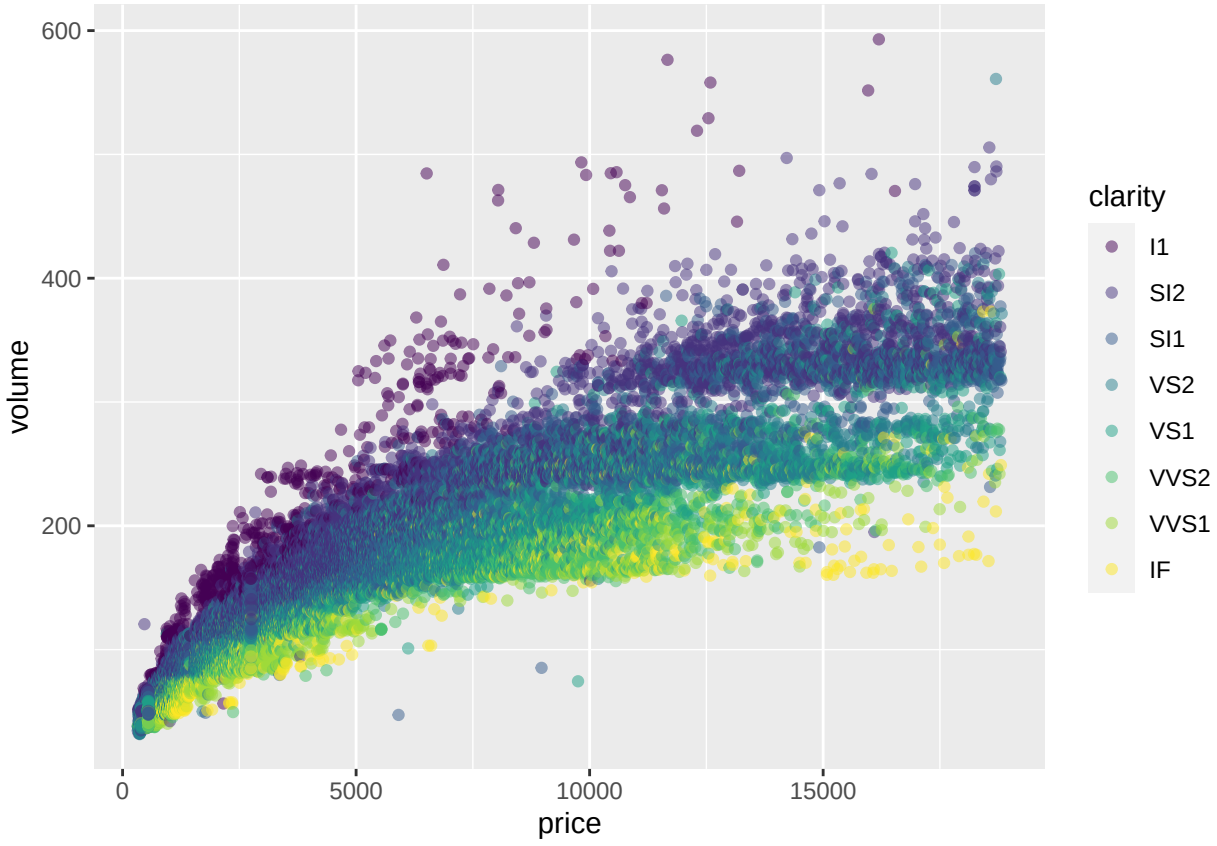
b) [2p] Use the data from `diamonds_volume` variable and plot the `volume` (vertical axis) against the `price` (horizontal axis) in a scatterplot. Colour points by `clarity`. Make points 0.5 transparent.

```
ggplot(diamonds_volume) +
  aes(x = price, y = volume, color = clarity) +
  geom_point( alpha = 0.5 )
```



b) [1p] Replot the scatterplot in Q9.b but now with rows where $volume > 0$ and $volume \leq 600$.

```
ggplot(diamonds_volume %>% filter(volume > 0, volume <= 600) ) +
  aes(x = price, y = volume, color = clarity) +
  geom_point( alpha = 0.5 )
```



Q10. Read/write CSV files.

a) [1p] Write the table `diamonds_volume` to a *comma-separated values* (CSV) file. Give the following name to the file: `diamonds_volume.csv`

```
write_csv(diamonds_volume, path = "diamonds_volume.csv")
```

b) [1p] Read the file `diamonds_volume.csv` back into variable `d` and show it.

```
d <- read_csv(file = "diamonds_volume.csv")
d
```

```
# A tibble: 53,940 x 11
   carat cut       color clarity depth table price     x     y     z volume
   <dbl> <chr>     <chr> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43   38.2
 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31   34.5
 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31   38.1
 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63   46.7
 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75   51.9
 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48   38.7
 7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47   38.8
 8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53   42.3
 9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49   36.4
10 0.23  Very Good H     VS1      59.4    61   338  4     4.05  2.39   38.7
# ... with 53,930 more rows
```