Data Cleaning for Preparation of Analysis in R

by Matthias Langenhan

03.10.2021

This R-markdown-file documents all data cleaning and transformation done in R for the case study

Loading packages

For basic data wrangling, manipulation and plotting (via ggplot2) we install the tidyverse package that itself contains a lot of useful packages. You will need to install the tidyverse package manually via install.packages("tidyverse"), since doing so through a knitr document causes issues. Then we load it into the environment:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Check and set current working directory

If you want run this markdown file, it is important that you change the absolute file path to wherever you saved the data files!

```
setwd("C:/Users/Matthias/Desktop/Google_Data_Analytics_Capstone_Project_Container")
getwd()
```

```
## [1] "C:/Users/Matthias/Desktop/Google_Data_Analytics_Capstone_Project_Container"
```

Importing, inspection and transformation of the data

```
ContainerData <- read_csv2("rawDataContainerUnloading2019v2.csv")
```

```
## i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.

## Rows: 235 Columns: 14

## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr   (1): ItemDimensionsCentimeter
## dbl  (12): PaletteCompletionTimeMinutes, PaletteCompletionTimeSeconds, Total...
## date  (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(ContainerData)
```

```
## Rows: 235
## Columns: 14
## $ Date                                  <date> 2019-09-09, 2019-09-09, 2019-09~
## $ PaletteCompletionTimeMinutes          <dbl> 1, 5, 10, 12, 15, 17, 20, 22, 24~
## $ PaletteCompletionTimeSeconds          <dbl> 40, 11, 26, 50, 26, 47, 0, 25, 4~
## $ TotalPaletteCompletionTimeSeconds     <dbl> 100, 311, 626, 770, 926, 1067, 1~
## $ DiffTotalPaletteCompletionTimeSeconds <dbl> 100, 211, 315, 144, 156, 141, 13~
## $ TeamSize                              <dbl> 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ ItemWeight                            <dbl> 4.35, 4.35, 4.35, 4.35, 4.35, 4.~
## $ ItemDimensionsCentimeter              <chr> NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ ItemHeightCentimeter                  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ ItemLengthCentimeter                  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ ItemDepthCentimeter                   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ ItemVolumeCubiccentimeter             <dbl> 41538, 41538, 41538, 41538, 4153~
## $ ItemQuantityPerPalette                <dbl> 24, 24, 24, 24, 24, 24, 24, 24, ~
## $ PaletteNumber                         <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1~
```

The number of rows and columns is correct and so are the data types. We want to narrow the data down to the columns of interest and rename the variables at the same time:

```
core_data <- select(ContainerData,
                    DiffTotalPaletteCompletionTimeSeconds,
                    TeamSize,
                    ItemWeight,
                    ItemVolumeCubiccentimeter,
                    ItemQuantityPerPalette,
                    PaletteNumber)
core_data <- rename(core_data,
                    time_seconds = DiffTotalPaletteCompletionTimeSeconds,
                    team_size = TeamSize,
                    item_weight = ItemWeight,
                    item_volume = ItemVolumeCubiccentimeter,
                    palette_quantity = ItemQuantityPerPalette,
                    palette_number = PaletteNumber)
glimpse(core_data)
```

```
## Rows: 235
## Columns: 6
## $ time_seconds     <dbl> 100, 211, 315, 144, 156, 141, 133, 145, 135, 180, 138~
## $ team_size        <dbl> 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5,~
## $ item_weight      <dbl> 4.35, 4.35, 4.35, 4.35, 4.35, 4.35, 4.35, 4.35, 4.35,~
## $ item_volume      <dbl> 41538, 41538, 41538, 41538, 41538, 41538, 41538, 4153~
## $ palette_quantity <dbl> 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 2~
## $ palette_number   <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
```

Now we check if values in columns “make sense”. First, we want to know if the unique values in the columns are reasonable:

```
unique(core_data$team_size)
```

```
## [1] 3 4 5 6
```

```
unique(core_data$palette_quantity)
```

```
## [1] 24 30 50 14
```

```
unique(core_data$palette_number)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## [51] 51 52 53 54 55 56 57 58 59
```

Is the data range from lowest to highest value reasonable?

```
summary(core_data)
```

```
##   time_seconds     team_size      item_weight     item_volume
##  Min.   :  5.0   Min.   :3.000   Min.   :3.040   Min.   :30723
##  1st Qu.: 86.0   1st Qu.:5.000   1st Qu.:4.520   1st Qu.:43988
##  Median :122.0   Median :5.000   Median :8.140   Median :64276
##  Mean   :131.4   Mean   :5.157   Mean   :6.802   Mean   :60693
##  3rd Qu.:160.0   3rd Qu.:5.000   3rd Qu.:8.490   3rd Qu.:75038
##  Max.   :580.0   Max.   :6.000   Max.   :8.840   Max.   :85800
##  palette_quantity palette_number
##  Min.   :14.00    Min.   : 1.00
##  1st Qu.:19.00    1st Qu.:12.00
##  Median :24.00    Median :24.00
##  Mean   :23.51    Mean   :24.42
##  3rd Qu.:24.00    3rd Qu.:36.00
##  Max.   :50.00    Max.   :59.00
```

Column time_seconds shows some questionable values (lowest value of five seconds to fully stack a palette is humanly impossible). Further investigation is needed. Save a copy of the original data file:

```
write.csv2(core_data, "core_data_v1.csv")
```

Sort data by time_seconds, ascending:

```
sorted <- arrange(core_data, time_seconds)
```

Output shows that there are multiple observations in the time_seconds column that seem nonsensical:

```
print(sorted$time_seconds)
```

```
##   [1]   5   5   5  10  10  13  15  15  20  20  30  30  35  35  40  40  41  42
##  [19]  42  43  44  47  48  50  50  52  55  56  57  58  58  58  59  60  60  61
##  [37]  61  63  64  64  65  65  65  66  66  69  69  72  73  75  75  78  82  82
##  [55]  85  85  85  86  86  86  88  90  90  90  90  91  92  92  93  94  94  95
##  [73]  95  96  98  99  99  99 100 100 100 101 101 101 102 102 102 102 104 104
##  [91] 106 106 106 107 107 109 110 110 110 110 111 111 111 112 112 113 113 113
## [109] 113 114 115 115 117 117 118 121 121 122 122 122 123 124 125 125 125 125
## [127] 125 126 128 129 130 133 133 133 135 135 137 137 137 138 138 138 139 140
## [145] 141 142 142 142 143 144 144 144 145 145 145 146 147 147 147 148 148 150
## [163] 150 151 154 155 155 155 155 156 156 156 157 158 159 160 160 163 167 167
## [181] 170 171 171 172 177 178 178 180 180 182 183 183 183 189 190 190 192 195
## [199] 195 198 201 201 204 209 209 210 211 215 215 216 217 223 225 227 228 228
## [217] 231 237 237 244 245 250 253 255 262 268 287 310 312 315 319 342 382 427
## [235] 580
```

To gage which nonsensical observations to drop, we introduce a new column that shows the time it takes to stack only one item onto a palette:

```
core_data_v2 <- core_data %>%
  mutate(item_time = time_seconds / palette_quantity)
```

Show the properties of the new column:

```
summary(core_data_v2)
```

```
##   time_seconds     team_size      item_weight     item_volume
##  Min.   :  5.0   Min.   :3.000   Min.   :3.040   Min.   :30723
##  1st Qu.: 86.0   1st Qu.:5.000   1st Qu.:4.520   1st Qu.:43988
##  Median :122.0   Median :5.000   Median :8.140   Median :64276
##  Mean   :131.4   Mean   :5.157   Mean   :6.802   Mean   :60693
##  3rd Qu.:160.0   3rd Qu.:5.000   3rd Qu.:8.490   3rd Qu.:75038
##  Max.   :580.0   Max.   :6.000   Max.   :8.840   Max.   :85800
##  palette_quantity palette_number    item_time
##  Min.   :14.00    Min.   : 1.00   Min.   : 0.2083
##  1st Qu.:19.00    1st Qu.:12.00   1st Qu.: 3.5375
##  Median :24.00    Median :24.00   Median : 5.5417
##  Mean   :23.51    Mean   :24.42   Mean   : 5.9116
##  3rd Qu.:24.00    3rd Qu.:36.00   3rd Qu.: 7.3958
##  Max.   :50.00    Max.   :59.00   Max.   :30.5000
```

```
sorted_2 <- arrange(core_data_v2, item_time)
print(sorted_2$item_time)
```

```
##   [1]  0.2083333  0.2083333  0.3571429  0.4166667  0.4166667  0.6250000
##   [7]  0.6250000  0.8333333  0.8333333  0.9285714  1.2500000  1.2500000
##  [13]  1.6666667  1.7500000  1.7500000  1.7916667  1.9583333  2.0000000
##  [19]  2.0000000  2.0833333  2.0833333  2.4166667  2.4166667  2.4583333
##  [25]  2.5000000  2.5000000  2.5000000  2.5416667  2.5416667  2.6250000
##  [31]  2.6666667  2.6666667  2.7083333  2.8571429  2.8666667  2.9285714
##  [37]  3.0000000  3.0000000  3.0000000  3.0666667  3.1000000  3.1250000
##  [43]  3.1333333  3.1428571  3.2500000  3.2666667  3.3000000  3.3000000
##  [49]  3.3000000  3.3333333  3.3666667  3.3666667  3.3666667  3.4000000
##  [55]  3.4166667  3.4666667  3.4666667  3.5333333  3.5333333  3.5416667
##  [61]  3.5666667  3.5666667  3.5833333  3.6600000  3.6666667  3.7142857
##  [67]  3.7500000  3.7666667  3.7666667  3.7916667  3.8333333  3.8400000
##  [73]  3.9000000  3.9000000  3.9166667  3.9285714  3.9333333  3.9600000
##  [79]  4.0000000  4.0333333  4.0714286  4.1428571  4.1666667  4.1666667
##  [85]  4.1800000  4.3000000  4.3200000  4.3400000  4.4166667  4.5000000
##  [91]  4.5833333  4.6200000  4.6250000  4.6428571  4.6428571  4.6666667
##  [97]  4.7083333  4.7142857  4.7142857  4.7333333  4.7500000  4.7916667
## [103]  4.8000000  4.9285714  4.9285714  5.0416667  5.0833333  5.1666667
## [109]  5.2083333  5.2083333  5.2083333  5.2142857  5.3571429  5.3750000
## [115]  5.4166667  5.5416667  5.5416667  5.5416667  5.6250000  5.7083333
## [121]  5.7083333  5.7083333  5.7400000  5.7500000  5.7500000  5.7500000
## [127]  5.7916667  5.8333333  5.8571429  5.8750000  5.9166667  5.9166667
## [133]  5.9583333  6.0000000  6.0000000  6.0416667  6.0416667  6.0416667
## [139]  6.0714286  6.0714286  6.1250000  6.1250000  6.1250000  6.1428571
## [145]  6.1666667  6.2500000  6.2857143  6.2916667  6.3333333  6.4166667
## [151]  6.4285714  6.4583333  6.4583333  6.5000000  6.5000000  6.5000000
## [157]  6.5416667  6.5714286  6.5833333  6.6250000  6.6666667  6.7000000
## [163]  6.7857143  6.7857143  6.7916667  6.8571429  6.9583333  6.9583333
## [169]  7.0833333  7.1250000  7.1250000  7.1666667  7.2857143  7.2857143
## [175]  7.2857143  7.3750000  7.4166667  7.4166667  7.5000000  7.5000000
## [181]  7.5833333  7.6250000  7.6250000  7.7857143  7.8571429  7.8571429
## [187]  7.8750000  7.9166667  7.9285714  7.9285714  8.0000000  8.0714286
## [193]  8.3571429  8.3750000  8.5000000  8.7083333  8.7142857  8.7142857
## [199]  8.7500000  8.7857143  8.7916667  8.9285714  8.9285714  8.9583333
## [205]  9.0000000  9.1428571  9.2916667  9.3750000  9.4583333  9.8750000
## [211]  9.8750000 10.1666667 10.2083333 10.4166667 10.4285714 10.5416667
## [217] 10.5714286 10.6250000 10.7142857 10.9166667 11.0714286 11.0714286
## [223] 11.1666667 11.4000000 11.4285714 13.0000000 13.1250000 13.2916667
## [229] 13.9285714 15.9166667 16.2857143 16.2857143 19.3333333 22.1428571
## [235] 30.5000000
```

We make the ad-hoc decision to drop all observations with item_time < 3 seconds and save a copy of the original data file:

```
write.csv2(core_data_v2, "core_data_v2.csv")
core_data_v3 <- filter(core_data_v2, item_time >= 3)
core_data_v3 <- rename(core_data_v3,
                       item_weight_kg = item_weight,
                       item_volume_cc = item_volume,
                       item_time_sec = item_time,
                       time_sec = time_seconds)
```

Now we have 199 observations left:

```
summary(core_data_v3)
```

```
##     time_sec       team_size     item_weight_kg  item_volume_cc
##  Min.   : 44.0   Min.   :3.000   Min.   :3.040   Min.   :30723
##  1st Qu.:102.0   1st Qu.:5.000   1st Qu.:4.350   1st Qu.:41538
##  Median :135.0   Median :5.000   Median :8.140   Median :64276
##  Mean   :147.9   Mean   :5.156   Mean   :6.633   Mean   :59943
##  3rd Qu.:177.5   3rd Qu.:5.000   3rd Qu.:8.840   3rd Qu.:85800
##  Max.   :580.0   Max.   :6.000   Max.   :8.840   Max.   :85800
##  palette_quantity palette_number item_time_sec
##  Min.   :14.00    Min.   : 1.0   Min.   : 3.000
##  1st Qu.:14.00    1st Qu.:11.5   1st Qu.: 4.310
##  Median :24.00    Median :23.0   Median : 6.042
##  Mean   :23.67    Mean   :23.5   Mean   : 6.658
##  3rd Qu.:24.00    3rd Qu.:34.0   3rd Qu.: 7.857
##  Max.   :50.00    Max.   :59.0   Max.   :30.500
```

```
glimpse(core_data_v3)
```

```
## Rows: 199
## Columns: 7
## $ time_sec         <dbl> 100, 211, 315, 144, 156, 141, 133, 145, 135, 180, 138~
## $ team_size        <dbl> 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5,~
## $ item_weight_kg   <dbl> 4.35, 4.35, 4.35, 4.35, 4.35, 4.35, 4.35, 4.35, 4.35,~
## $ item_volume_cc   <dbl> 41538, 41538, 41538, 41538, 41538, 41538, 41538, 4153~
## $ palette_quantity <dbl> 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 2~
## $ palette_number   <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
## $ item_time_sec    <dbl> 4.166667, 8.791667, 13.125000, 6.000000, 6.500000, 5.~
```

```
write.csv2(core_data_v3, "core_data_v3.csv")
```

Now our data set is ready to be analyzed.