Data Transformation

Choonghyun Ryu

2020-01-21

Preface

After you have acquired the data, you should do the following:

- Diagnose data quality.
  - If there is a problem with data quality,
  - The data must be corrected or re-acquired.
- Explore data to understand the data and find scenarios for performing the analysis.
- **Derive new variables or perform variable transformations.**

The dlookr package makes these steps fast and easy:

- Performs an data diagnosis or automatically generates a data diagnosis report.
- Discover data in a variety of ways, and automatically generate EDA(exploratory data analysis) report.
- **Imputate missing values and outliers, resolve skewed data, and binarize continuous variables into categorical variables. And generates an automated report to support it.**

This document introduces **data transformation** methods provided by the dlookr package. You will learn how to transform of `tbl_df` data that inherits from data.frame and `data.frame` with functions provided by dlookr.

dlookr synergy with `dplyr` increases. Particularly in data transformation and data wrangle, it increases the efficiency of the `tidyverse` package group.

datasets

To illustrate the basic use of EDA in the dlookr package, I use a `Carseats` datasets. `Carseats` in the `ISLR` package is simulation dataset that sells children's car seats at 400 stores. This data is a data.frame created for the purpose of predicting sales volume.

```
library(ISLR)
str(Carseats)
'data.frame':   400 obs. of  11 variables:
 $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
 $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
 $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
 $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
 $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
 $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
 $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
 $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
 $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
 $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
 $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

The contents of individual variables are as follows. (Refer to ISLR::Carseats Man page)

- Sales
  - Unit sales (in thousands) at each location
- CompPrice
  - Price charged by competitor at each location

- Income
  - Community income level (in thousands of dollars)
- Advertising
  - Local advertising budget for company at each location (in thousands of dollars)
- Population
  - Population size in region (in thousands)
- Price
  - Price company charges for car seats at each site
- ShelveLoc
  - A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site
- Age
  - Average age of the local population
- Education
  - Education level at each location
- Urban
  - A factor with levels No and Yes to indicate whether the store is in an urban or rural location
- US
  - A factor with levels No and Yes to indicate whether the store is in the US or not

When data analysis is performed, data containing missing values is often encountered. However, Carseats is complete data without missing. Therefore, the missing values are generated as follows. And I created a data.frame object named carseats.

```
carseats <- ISLR::Carseats

suppressWarnings(RNGversion("3.5.0"))
set.seed(123)
carseats[sample(seq(NROW(carseats)), 20), "Income"] <- NA

suppressWarnings(RNGversion("3.5.0"))
set.seed(456)
carseats[sample(seq(NROW(carseats)), 10), "Urban"] <- NA
```

Data Transformation

dlookr imputates missing values and outliers and resolves skewed data. It also provides the ability to bin continuous variables as categorical variables.

Here is a list of the data conversion functions and functions provided by dlookr:

- `find_na()` finds a variable that contains the missing values variable, and `imputate_na()` imputates the missing values.
- `find_outliers()` finds a variable that contains the outliers, and `imputate_outlier()` imputates the outlier.
- `summary.imputation()` and `plot.imputation()` provide information and visualization of the imputated variables.
- `find_skewness()` finds the variables of the skewed data, and `transform()` performs the resolving of the skewed data.
- `transform()` also performs standardization of numeric variables.
- `summary.transform()` and `plot.transform()` provide information and visualization of transformed variables.
- `binning()` and `binning_by()` convert binational data into categorical data.
- `print.bins()` and `summary.bins()` show and summarize the binning results.
- `plot.bins()` and `plot.optimal_bins()` provide visualization of the binning result.
- `transformation_report()` performs the data transform and reports the result.

Imputation of missing values

Imputates the missing value with `imputate_na()`

`imputate_na()` imputates the missing value in the variable. The predictor with missing values supports both numeric and categorical variables and supports the following methods.

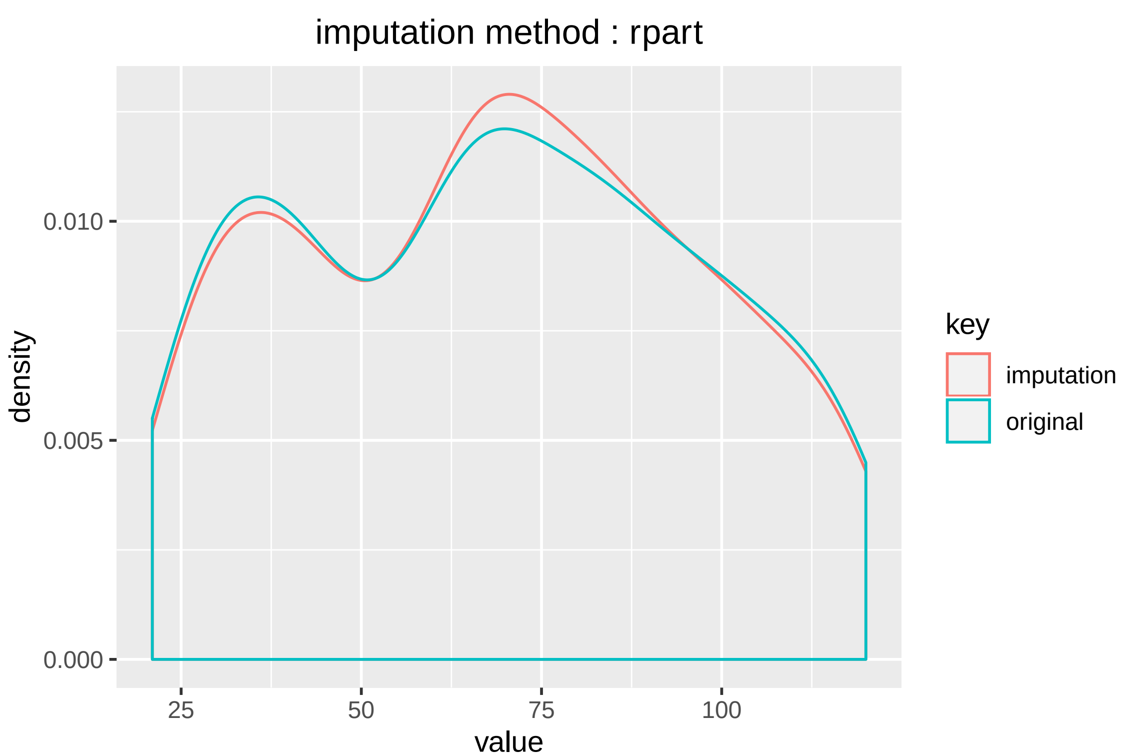
- predictor is numerical variable
  - "mean" : arithmetic mean
  - "median" : median
  - "mode" : mode
  - "knn" : K-nearest neighbors
    - target variable must be specified
  - "rpart" : Recursive Partitioning and Regression Trees
    - target variable must be specified
  - "mice" : Multivariate Imputation by Chained Equations
    - target variable must be specified
    - random seed must be set
- predictor is categorical variable
  - "mode" : mode
  - "rpart" : Recursive Partitioning and Regression Trees
    - target variable must be specified
  - "mice" : Multivariate Imputation by Chained Equations
    - target variable must be specified
    - random seed must be set

`imputate_na()` imputates the missing value with "rpart" for the numeric variable, `Income`. `summary()` summarizes missing value imputation information, and `plot()` visualizes imputation information.

```
income <- imputate_na(carseats, Income, US, method = "rpart")

# result of imputate
income
  [1]  73.00000  48.00000  35.00000 100.00000  64.00000 113.00000 105.00000  81.00000
  [9] 110.00000 113.00000  78.00000  94.00000  35.00000  28.00000 117.00000  95.00000
 [17]  76.75000  68.70968 110.00000  76.00000  90.00000  29.00000  46.00000  31.00000
 [25] 119.00000  32.00000 115.00000 118.00000  74.00000  99.00000  94.00000  58.00000
 [33]  32.00000  38.00000  54.00000  84.00000  76.00000  41.00000  73.00000  69.27778
 [41]  98.00000  53.00000  69.00000  42.00000  79.00000  63.00000  90.00000  98.00000
 [49]  52.00000  93.00000  32.00000  90.00000  40.00000  64.00000 103.00000  81.00000
 [57]  82.00000  91.00000  93.00000  71.00000 102.00000  32.00000  45.00000  88.00000
 [65]  67.00000  26.00000  92.00000  61.00000  69.00000  59.00000  81.00000  51.00000
 [73]  45.00000  90.00000  68.00000 111.00000  87.00000  71.00000  48.00000  67.00000
 [81] 100.00000  72.00000  83.00000  36.00000  25.00000 103.00000  84.00000  67.00000
 [89]  42.00000  66.00000  22.00000  46.00000 113.00000  30.00000  88.93750  25.00000
 [97]  42.00000  82.00000  77.00000  47.00000  69.00000  93.00000  22.00000  91.00000
[105]  96.00000 100.00000  33.00000 107.00000  79.00000  65.00000  62.00000 118.00000
[113]  99.00000  29.00000  87.00000  68.70968  75.00000  53.00000  88.00000  94.00000
[121] 105.00000  89.00000 100.00000 103.00000 113.00000  98.33333  68.00000  48.00000
[129] 100.00000 120.00000  84.00000  69.00000  87.00000  98.00000  31.00000  94.00000
[137]  75.00000  42.00000 103.00000  62.00000  60.00000  42.00000  84.00000  88.00000
[145]  68.00000  63.00000  83.00000  54.00000 119.00000 120.00000  84.00000  58.00000
```

```
[153]  78.00000  36.00000  69.00000  72.00000  34.00000  58.00000  90.00000  60.00000
[161]  28.00000  21.00000  83.53846  64.00000  64.00000  58.00000  67.00000  73.00000
[169]  89.00000  41.00000  39.00000 106.00000 102.00000  91.00000  24.00000  89.00000
[177]  69.27778  72.00000  85.00000  25.00000 112.00000  83.00000  60.00000  74.00000
[185]  33.00000 100.00000  51.00000  32.00000  37.00000 117.00000  37.00000  42.00000
[193]  26.00000  70.00000  98.00000  93.00000  28.00000  61.00000  80.00000  88.00000
[201]  92.00000  83.00000  78.00000  82.00000  80.00000  22.00000  67.00000 105.00000
[209]  98.33333  21.00000  41.00000 118.00000  69.00000  84.00000 115.00000  83.00000
[217]  43.75000  44.00000  61.00000  79.00000 120.00000  73.47368 119.00000  45.00000
[225]  82.00000  25.00000  33.00000  64.00000  73.00000 104.00000  60.00000  69.00000
[233]  80.00000  76.00000  62.00000  32.00000  34.00000  28.00000  24.00000 105.00000
[241]  80.00000  63.00000  46.00000  25.00000  30.00000  43.00000  56.00000 114.00000
[249]  52.00000  67.00000 105.00000 111.00000  97.00000  24.00000 104.00000  81.00000
[257]  40.00000  62.00000  38.00000  36.00000 117.00000  42.00000  73.47368  26.00000
[265]  29.00000  35.00000  93.00000  82.00000  57.00000  69.00000  26.00000  56.00000
[273]  33.00000 106.00000  93.00000 119.00000  69.00000  48.00000 113.00000  57.00000
[281]  86.00000  69.00000  96.00000 110.00000  46.00000  26.00000 118.00000  44.00000
[289]  40.00000  77.00000 111.00000  70.00000  66.00000  84.00000  76.00000  35.00000
[297]  44.00000  83.00000  63.00000  40.00000  78.00000  93.00000  77.00000  52.00000
[305]  98.00000  29.00000  32.00000  92.00000  80.00000 111.00000  65.00000  68.00000
[313] 117.00000  81.00000  56.57895  21.00000  36.00000  30.00000  72.00000  45.00000
[321]  70.00000  39.00000  50.00000 105.00000  65.00000  69.00000  30.00000  38.00000
[329]  66.00000  54.00000  59.00000  63.00000  33.00000  60.00000 117.00000  70.00000
[337]  35.00000  38.00000  24.00000  44.00000  29.00000 120.00000 102.00000  42.00000
[345]  80.00000  68.00000  76.75000  39.00000 102.00000  27.00000  51.83333 115.00000
[353] 103.00000  67.00000  31.00000 100.00000 109.00000  73.00000  96.00000  62.00000
[361]  86.00000  25.00000  55.00000  51.83333  21.00000  30.00000  56.00000 106.00000
[369]  22.00000 100.00000  41.00000  81.00000  68.66667  68.88889  47.00000  46.00000
[377]  60.00000  61.00000  88.00000 111.00000  64.00000  65.00000  28.00000 117.00000
[385]  37.00000  73.00000 116.00000  73.00000  89.00000  42.00000  75.00000  63.00000
[393]  42.00000  51.00000  58.00000 108.00000  81.17647  26.00000  79.00000  37.00000
attr(,"var_type")
[1] "numerical"
attr(,"method")
[1] "rpart"
attr(,"na_pos")
 [1]  17  18  40  95 116 126 163 177 179 209 217 222 263 315 347 351 364 373 374 397
attr(,"type")
[1] "missing values"
attr(,"message")
[1] "complete imputation"
attr(,"success")
[1] TRUE
attr(,"class")
[1] "imputation" "numeric"

# summary of imputate
summary(income)
* Impute missing values based on Recursive Partitioning and Regression Trees
 - method : rpart

* Information of Imputation (before vs after)
           Original   Imputation
```

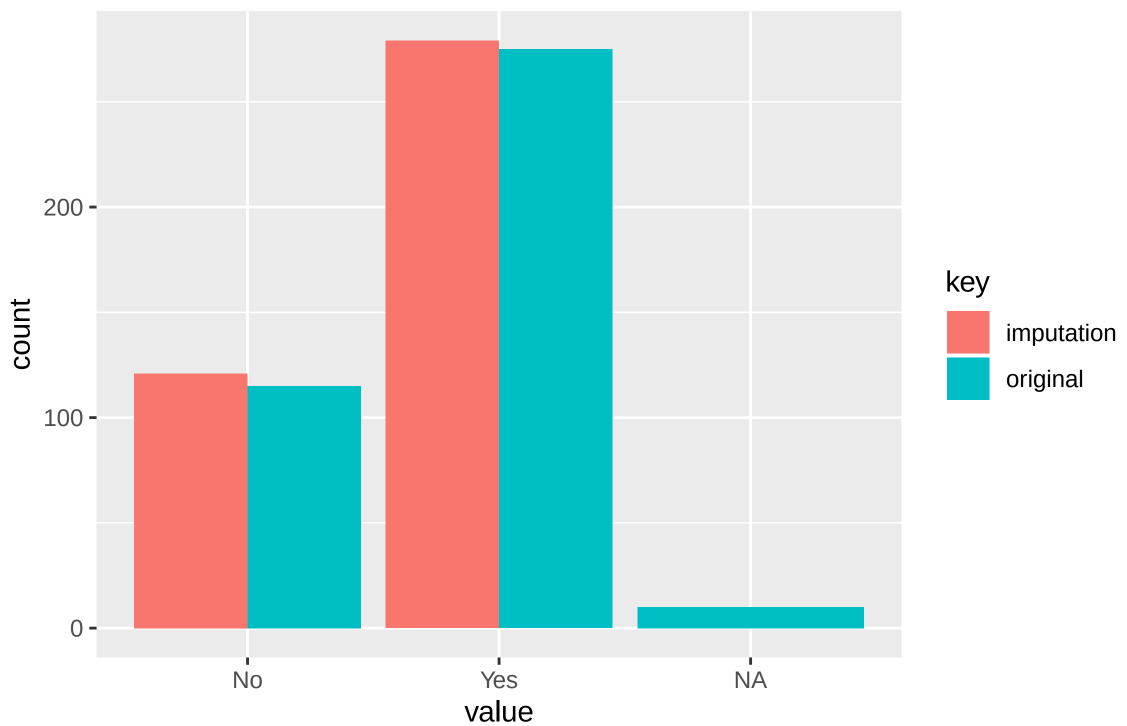
```
n         380.000000 400.00000000
na         20.000000   0.00000000
mean       68.860526  69.05073137
sd         28.091615  27.57381661
se_mean     1.441069   1.37869083
IQR        48.250000  46.00000000
skewness    0.044906   0.02935732
kurtosis   -1.089201  -1.03508622
p00        21.000000  21.00000000
p01        21.790000  21.99000000
p05        26.000000  26.00000000
p10        30.000000  30.90000000
p20        39.000000  40.00000000
p25        42.750000  44.00000000
p30        48.000000  51.58333333
p40        62.000000  63.00000000
p50        69.000000  69.00000000
p60        78.000000  77.40000000
p70        86.300000  84.30000000
p75        91.000000  90.00000000
p80        96.200000  96.00000000
p90       108.100000 106.10000000
p95       115.050000 115.00000000
p99       119.210000 119.01000000
p100      120.000000 120.00000000

# viz of imputate
plot(income)
```



The following imputates the categorical variable `urban` by the "mice" method.

```r
library(mice)

urban <- imputate_na(carseats, Urban, US, method = "mice")

 iter imp variable
  1   1  Income  Urban
  1   2  Income  Urban
  1   3  Income  Urban
  1   4  Income  Urban
  1   5  Income  Urban
  2   1  Income  Urban
  2   2  Income  Urban
  2   3  Income  Urban
  2   4  Income  Urban
  2   5  Income  Urban
  3   1  Income  Urban
  3   2  Income  Urban
  3   3  Income  Urban
  3   4  Income  Urban
  3   5  Income  Urban
  4   1  Income  Urban
  4   2  Income  Urban
  4   3  Income  Urban
  4   4  Income  Urban
  4   5  Income  Urban
  5   1  Income  Urban
  5   2  Income  Urban
  5   3  Income  Urban
  5   4  Income  Urban
  5   5  Income  Urban

# result of imputate
urban
  [1] Yes Yes Yes Yes Yes No  Yes Yes No  No  No  Yes Yes Yes Yes No  Yes Yes No  Yes Yes No
 [23] Yes Yes Yes No  No  Yes Yes Yes Yes Yes Yes Yes Yes Yes No  Yes Yes No  No  Yes Yes Yes
 [45] Yes Yes No  Yes Yes Yes Yes Yes Yes Yes No  Yes Yes Yes Yes Yes Yes No  Yes Yes No  No
 [67] Yes Yes Yes Yes Yes No  Yes No  No  No  Yes No  Yes Yes Yes Yes Yes No  No  No  Yes No
 [89] Yes No  No  Yes Yes No  Yes Yes No  Yes No  No  No  Yes No  Yes Yes Yes No  Yes Yes No
[111] Yes Yes No  Yes Yes Yes No  Yes Yes Yes Yes Yes Yes No  Yes No  Yes Yes Yes No  Yes No
[133] Yes Yes Yes No  No  Yes Yes No  Yes Yes Yes Yes No  Yes Yes No  No  Yes Yes No  No  No
[155] No  Yes Yes No  No  No  No  No  Yes No  No  Yes Yes Yes Yes Yes Yes Yes Yes Yes No  Yes
[177] No  Yes No  Yes Yes Yes Yes Yes No  Yes No  Yes Yes No  No  Yes No  Yes Yes Yes Yes Yes
[199] Yes Yes No  Yes No  Yes Yes Yes Yes No  Yes No  No  Yes Yes Yes Yes Yes Yes No  Yes Yes
[221] Yes Yes Yes Yes No  Yes Yes Yes No  No  No  No  Yes No  No  Yes Yes Yes Yes Yes Yes Yes
[243] No  Yes Yes No  Yes Yes Yes Yes Yes Yes Yes No  Yes Yes Yes Yes No  No  Yes Yes Yes Yes
[265] Yes Yes No  No  Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes No  Yes Yes No  Yes No  No  Yes
[287] No  Yes No  Yes No  Yes Yes Yes Yes No  Yes Yes Yes No  Yes Yes Yes Yes Yes Yes Yes Yes
[309] Yes Yes Yes Yes No  Yes Yes Yes Yes No  No  No  Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
[331] No  Yes Yes Yes Yes Yes Yes Yes No  Yes Yes No  No  Yes No  Yes No  No  Yes No  No  No
[353] Yes No  Yes Yes Yes Yes Yes Yes No  No  Yes Yes Yes No  No  Yes No  Yes Yes Yes No  Yes
[375] Yes Yes Yes No  Yes Yes Yes Yes Yes Yes Yes Yes Yes No  Yes Yes Yes Yes Yes No  Yes Yes
[397] No  Yes Yes Yes
attr(,"var_type")
```

```
[1] categorical
attr(,"method")
[1] mice
attr(,"na_pos")
 [1]  33  36  84  94 113 132 151 292 313 339
attr(,"type")
[1] missing values
attr(,"message")
[1] complete imputation
attr(,"success")
[1] TRUE
Levels: No Yes

# summary of imputate
summary(urban)
* Information of Imputation (before vs after)
     original imputation original_percent imputation_percent
No        115        121            28.75              30.25
Yes       275        279            68.75              69.75
<NA>       10          0             2.50               0.00

# viz of imputate
plot(urban)
```



Collaboration with dplyr

The following is an example of calculating the arithmetic mean of `US` variables by using the `Income` variable that imputates the missing value with `dplyr`.

```
# The mean before and after the imputation of the Income variable
carseats %>%
```

```
  mutate(Income_imp = imputate_na(carseats, Income, US, method = "knn")) %>%
  group_by(US) %>%
  summarise(orig = mean(Income, na.rm = TRUE),
    imputation = mean(Income_imp))
# A tibble: 2 x 3
  US     orig imputation
  <fct> <dbl>      <dbl>
1 No     65.8       66.1
2 Yes    70.4       70.5
```

Imputation of outliers

Imputates thr outliers with `imputate_outlier()`

`imputate_outlier()` imputates the outliers value. The predictor with outliers supports only numeric variables and supports the following methods.

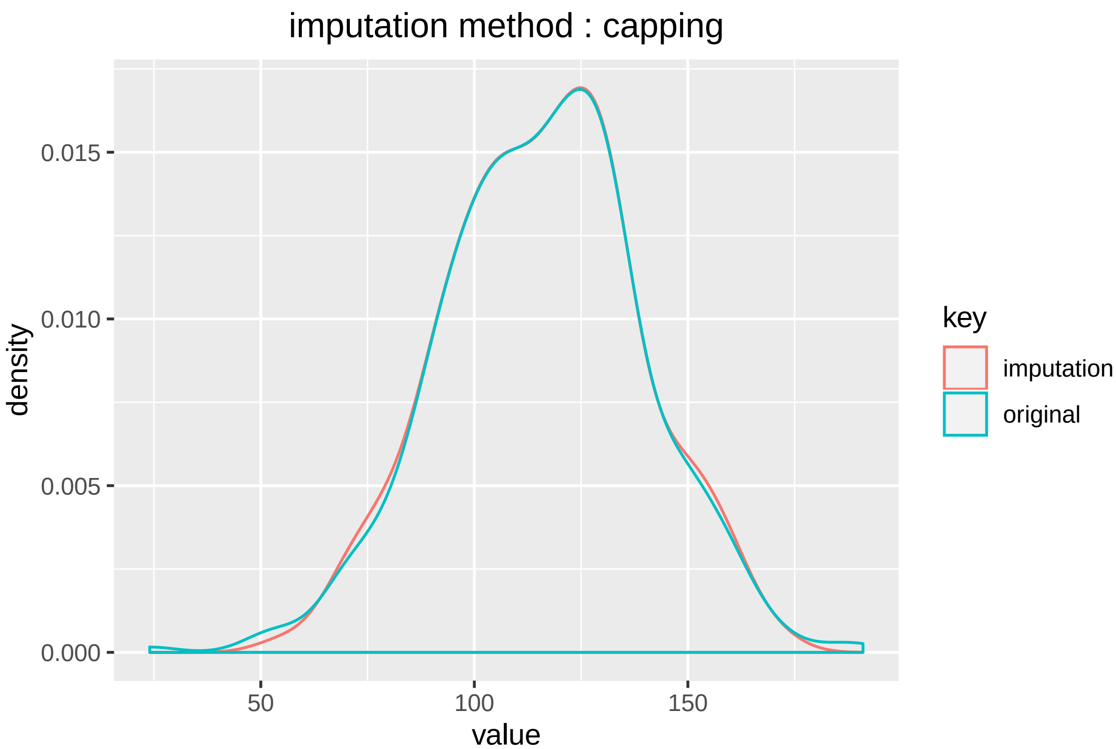
- predictor is numerical variable
  - "mean" : arithmetic mean
  - "median" : median
  - "mode" : mode
  - "capping" : Imputate the upper outliers with 95 percentile, and Imputate the bottom outliers with 5 percentile.

`imputate_outlier()` imputates the outliers with the numeric variable `Price` as the "capping" method, as follows. `summary()` summarizes outliers imputation information, and `plot()` visualizes imputation information.

```
price <- imputate_outlier(carseats, Price, method = "capping")

# result of imputate
price
  [1] 120.00  83.00  80.00  97.00 128.00  72.00 108.00 120.00 124.00 124.00 100.00  94.00
 [13] 136.00  86.00 118.00 144.00 110.00 131.00  68.00 121.00 131.00 109.00 138.00 109.00
 [25] 113.00  82.00 131.00 107.00  97.00 102.00  89.00 131.00 137.00 128.00 128.00  96.00
 [37] 100.00 110.00 102.00 138.00 126.00 124.00  77.00 134.00  95.00 135.00  70.00 108.00
 [49]  98.00 149.00 108.00 108.00 129.00 119.00 144.00 154.00  84.00 117.00 103.00 114.00
 [61] 123.00 107.00 133.00 101.00 104.00 128.00  91.00 115.00 134.00  99.00  99.00 150.00
 [73] 116.00 104.00 136.00  92.00  70.00  89.00 145.00  90.00  79.00 128.00 139.00  94.00
 [85] 121.00 112.00 134.00 126.00 111.00 119.00 103.00 107.00 125.00 104.00  84.00 148.00
 [97] 132.00 129.00 127.00 107.00 106.00 118.00  97.00  96.00 138.00  97.00 139.00 108.00
[109] 103.00  90.00 116.00 151.00 125.00 127.00 106.00 129.00 128.00 119.00  99.00 128.00
[121] 131.00  87.00 108.00 155.00 120.00  77.00 133.00 116.00 126.00 147.00  77.00  94.00
[133] 136.00  97.00 131.00 120.00 120.00 118.00 109.00  94.00 129.00 131.00 104.00 159.00
[145] 123.00 117.00 131.00 119.00  97.00  87.00 114.00 103.00 128.00 150.00 110.00  69.00
[157] 157.00  90.00 112.00  70.00 111.00 160.00 149.00 106.00 141.00 155.05 137.00  93.00
[169] 117.00  77.00 118.00  55.00 110.00 128.00 155.05 122.00 154.00  94.00  81.00 116.00
[181] 149.00  91.00 140.00 102.00  97.00 107.00  86.00  96.00  90.00 104.00 101.00 173.00
[193]  93.00  96.00 128.00 112.00 133.00 138.00 128.00 126.00 146.00 134.00 130.00 157.00
[205] 124.00 132.00 160.00  97.00  64.00  90.00 123.00 120.00 105.00 139.00 107.00 144.00
[217] 144.00 111.00 120.00 116.00 124.00 107.00 145.00 125.00 141.00  82.00 122.00 101.00
[229] 163.00  72.00 114.00 122.00 105.00 120.00 129.00 132.00 108.00 135.00 133.00 118.00
[241] 121.00  94.00 135.00 110.00 100.00  88.00  90.00 151.00 101.00 117.00 156.00 132.00
[253] 117.00 122.00 129.00  81.00 144.00 112.00  81.00 100.00 101.00 118.00 132.00 115.00
[265] 159.00 129.00 112.00 112.00 105.00 166.00  89.00 110.00  63.00  86.00 119.00 132.00
```

```
[277] 130.00 125.00 151.00 158.00 145.00 105.00 154.00 117.00  96.00 131.00 113.00  72.00
[289]  97.00 156.00 103.00  89.00  74.00  89.00  99.00 137.00 123.00 104.00 130.00  96.00
[301]  99.00  87.00 110.00  99.00 134.00 132.00 133.00 120.00 126.00  80.00 166.00 132.00
[313] 135.00  54.00 129.00 171.00  72.00 136.00 130.00 129.00 152.00  98.00 139.00 103.00
[325] 150.00 104.00 122.00 104.00 111.00  89.00 112.00 134.00 104.00 147.00  83.00 110.00
[337] 143.00 102.00 101.00 126.00  91.00  93.00 118.00 121.00 126.00 149.00 125.00 112.00
[349] 107.00  96.00  91.00 105.00 122.00  92.00 145.00 146.00 164.00  72.00 118.00 130.00
[361] 114.00 104.00 110.00 108.00 131.00 162.00 134.00  77.00  79.00 122.00 119.00 126.00
[373]  98.00 116.00 118.00 124.00  92.00 125.00 119.00 107.00  89.00 151.00 121.00  68.00
[385] 112.00 132.00 160.00 115.00  78.00 107.00 111.00 124.00 130.00 120.00 139.00 128.00
[397] 120.00 159.00  95.00 120.00
attr(,"method")
[1] "capping"
attr(,"var_type")
[1] "numerical"
attr(,"outlier_pos")
[1]  43 126 166 175 368
attr(,"outliers")
[1]  24  49 191 185  53
attr(,"type")
[1] "outliers"
attr(,"message")
[1] "complete imputation"
attr(,"success")
[1] TRUE
attr(,"class")
[1] "imputation" "numeric"

# summary of imputate
summary(price)
Impute outliers with capping

* Information of Imputation (before vs after)
            Original  Imputation
n        400.0000000 400.0000000
na         0.0000000   0.0000000
mean     115.7950000 115.8927500
sd        23.6766644  22.6109187
se_mean    1.1838332   1.1305459
IQR       31.0000000  31.0000000
skewness  -0.1252862  -0.0461621
kurtosis   0.4518850  -0.3030578
p00       24.0000000  54.0000000
p01       54.9900000  67.9600000
p05       77.0000000  77.0000000
p10       87.0000000  87.0000000
p20       96.8000000  96.8000000
p25      100.0000000 100.0000000
p30      104.0000000 104.0000000
p40      110.0000000 110.0000000
p50      117.0000000 117.0000000
p60      122.0000000 122.0000000
p70      128.3000000 128.3000000
```

```
p75      131.0000000 131.0000000
p80      134.0000000 134.0000000
p90      146.0000000 146.0000000
p95      155.0500000 155.0025000
p99      166.0500000 164.0200000
p100     191.0000000 173.0000000

# viz of imputate
plot(price)
```



Collaboration with dplyr

The following is an example of calculating the arithmetic mean of `US` variables by using the `Price` variable that imputates the outlier with `dplyr`.

```
# The mean before and after the imputation of the Price variable
carseats %>%
  mutate(Price_imp = imputate_outlier(carseats, Price, method = "capping")) %>%
  group_by(US) %>%
  summarise(orig = mean(Price, na.rm = TRUE),
    imputation = mean(Price_imp, na.rm = TRUE))
# A tibble: 2 x 3
  US     orig imputation
  <fct> <dbl>      <dbl>
1 No     114.       114.
2 Yes    117.       117.
```

Standardization and Resolving Skewness

Introduction to the use of transform()

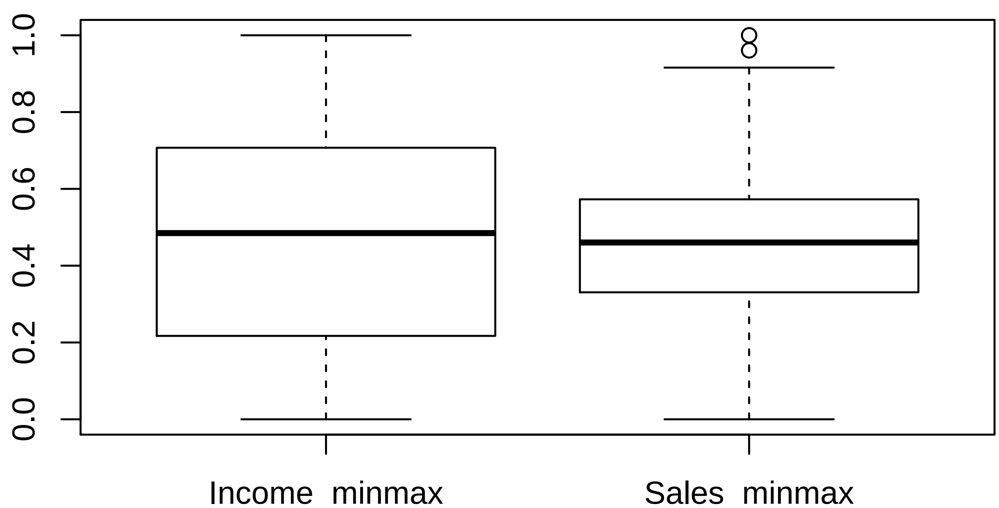
`transform()` performs data transformation. Only numeric variables are supported, and the following methods are provided.

- Standardization
  - "zscore" : z-score transformation. (x - mu) / sigma
  - "minmax" : minmax transformation. (x - min) / (max - min)
- Resolving Skewness
  - "log" : log transformation. log(x)
  - "log+1" : log transformation. log(x + 1). Used for values that contain 0.
  - "sqrt" : square root transformation.
  - "1/x" : 1 / x transformation
  - "x^2" : x square transformation
  - "x^3" : x^3 square transformation

Standardization with transform()

Use the methods "zscore" and "minmax" to perform standardization.

```
carseats %>%
  mutate(Income_minmax = transform(carseats$Income, method = "minmax"),
    Sales_minmax = transform(carseats$Sales, method = "minmax")) %>%
  select(Income_minmax, Sales_minmax) %>%
  boxplot()
```



Resolving Skewness data with transform()

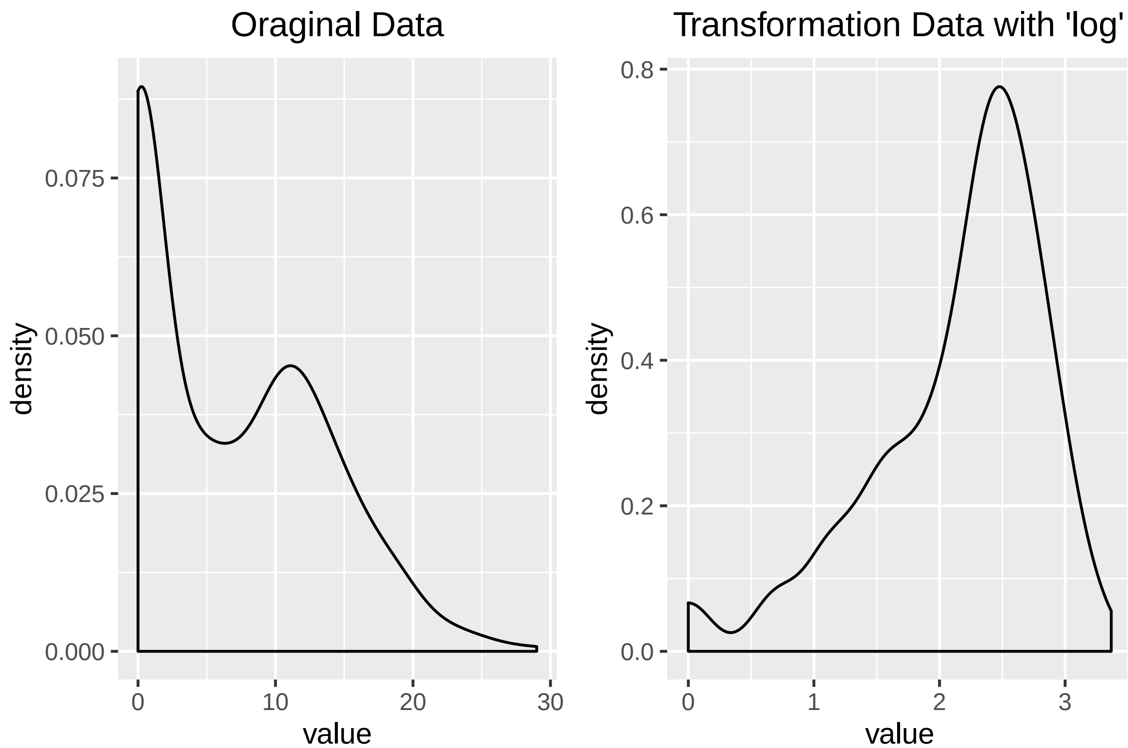
`find_skewness()` calculates the skewness and finds the skewed data.

```
# find index of skewed variables
find_skewness(carseats)
[1] 4

# find names of skewed variables
find_skewness(carseats, index = FALSE)
[1] "Advertising"

# compute the skewness
```

```
find_skewness(carseats, value = TRUE)
      Sales   CompPrice      Income Advertising  Population       Price         Age
      0.185      -0.043          NA       0.637      -0.051      -0.125      -0.077
  Education
      0.044

# compute the skewness & filtering with threshold
find_skewness(carseats, value = TRUE, thres = 0.1)
      Sales Advertising       Price
      0.185       0.637      -0.125
```

The skewness of `Advertising` is 0.637, which is a little slanted to the left, so I use `transformation ()` to convert it to `log`. `summary()` summarizes the transformation information, and `plot()` visualizes the transformation information.

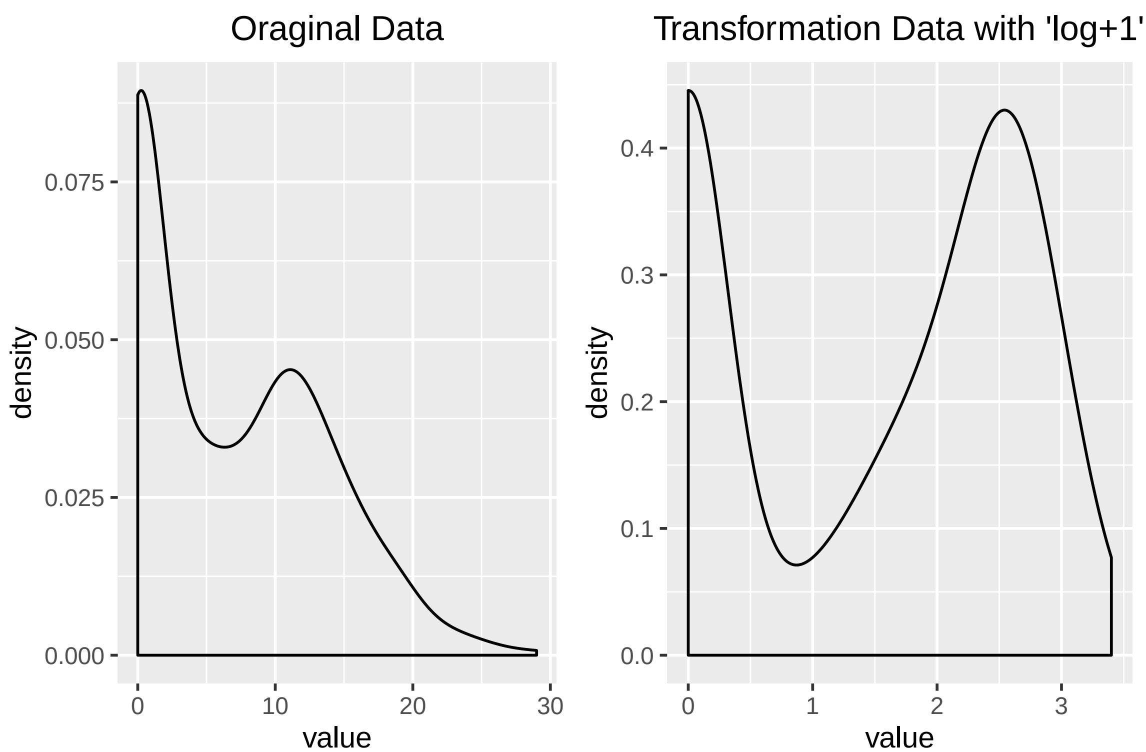
```
Advertising_log = transform(carseats$Advertising, method = "log")

# result of transformation
head(Advertising_log)
[1] 2.397895 2.772589 2.302585 1.386294 1.098612 2.564949
# summary of transformation
summary(Advertising_log)
* Resolving Skewness with log

* Information of Transformation (before vs after)
            Original Transformation
n        400.0000000    400.0000000
na         0.0000000      0.0000000
mean       6.6350000           -Inf
sd         6.6503642            NaN
se_mean    0.3325182            NaN
IQR       12.0000000            Inf
skewness   0.6395858            NaN
kurtosis  -0.5451178            NaN
p00        0.0000000           -Inf
p01        0.0000000           -Inf
p05        0.0000000           -Inf
p10        0.0000000           -Inf
p20        0.0000000           -Inf
p25        0.0000000           -Inf
p30        0.0000000           -Inf
p40        2.0000000      0.6931472
p50        5.0000000      1.6094379
p60        8.4000000      2.1265548
p70       11.0000000      2.3978953
p75       12.0000000      2.4849066
p80       13.0000000      2.5649494
p90       16.0000000      2.7725887
p95       19.0000000      2.9444390
p99       23.0100000      3.1359198
p100      29.0000000      3.3672958
# viz of transformation
plot(Advertising_log)
```



It seems that the raw data contains 0, as there is a -Inf in the log converted value. So this time we convert it to "log + 1".

```
Advertising_log <- transform(carseats$Advertising, method = "log+1")

# result of transformation
head(Advertising_log)
[1] 2.484907 2.833213 2.397895 1.609438 1.386294 2.639057
# summary of transformation
summary(Advertising_log)
* Resolving Skewness with log+1

* Information of Transformation (before vs after)
            Original Transformation
n        400.0000000   400.00000000
na         0.0000000     0.00000000
mean       6.6350000     1.46247709
sd         6.6503642     1.19436323
se_mean    0.3325182     0.05971816
IQR       12.0000000     2.56494936
skewness   0.6395858    -0.19852549
kurtosis  -0.5451178    -1.66342876
p00        0.0000000     0.00000000
p01        0.0000000     0.00000000
p05        0.0000000     0.00000000
p10        0.0000000     0.00000000
p20        0.0000000     0.00000000
p25        0.0000000     0.00000000
p30        0.0000000     0.00000000
p40        2.0000000     1.09861229
p50        5.0000000     1.79175947
```

```
p60        8.4000000     2.23936878
p70       11.0000000     2.48490665
p75       12.0000000     2.56494936
p80       13.0000000     2.63905733
p90       16.0000000     2.83321334
p95       19.0000000     2.99573227
p99       23.0100000     3.17846205
p100      29.0000000     3.40119738
# viz of transformation
plot(Advertising_log)
```



Binning

Binning of individual variables using `binning()`

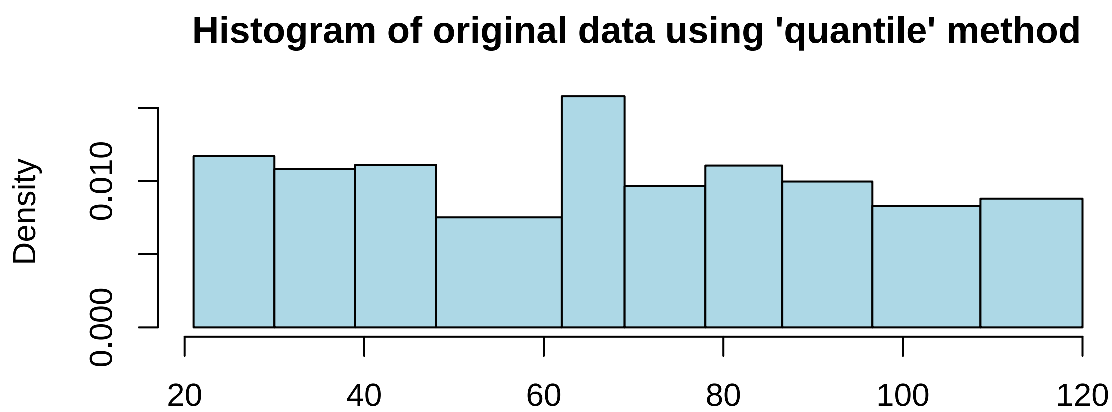
`binning()` transforms a numeric variable into a categorical variable by binning it. The following types of binning are supported.

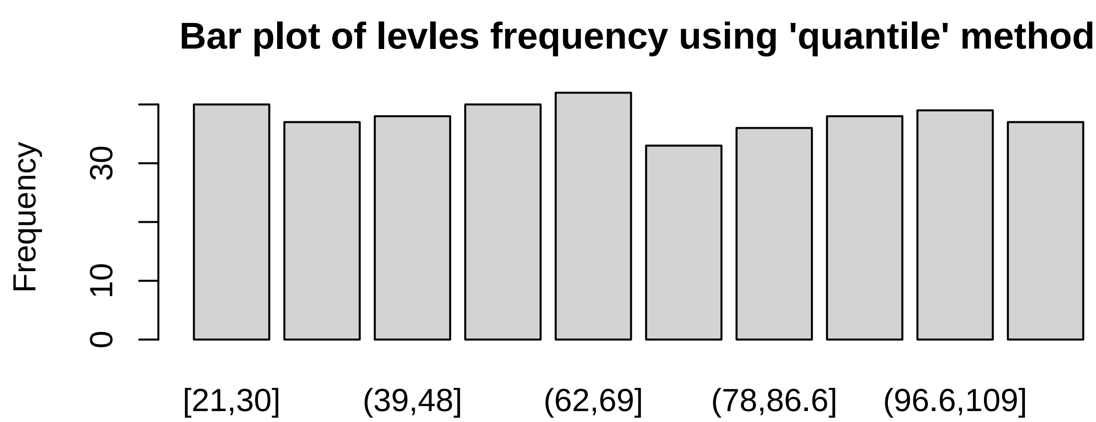
- "quantile" : categorize using quantile to include the same frequencies
- "equal" : categorize to have equal length segments
- "pretty" : categorized into moderately good segments
- "kmeans" : categorization using K-means clustering
- "bclust" : categorization using bagged clustering technique

The following example illustrates some ways to `Income` binning using `binning()`.:

```
# Binning the carat variable. default type argument is "quantile"
bin <- binning(carseats$Income)
# Print bins class object
bin
binned type: quantile
number of bins: 10
```

```
x
    [21,30]     (30,39]     (39,48]     (48,62]     (62,69]     (69,78]   (78,86.6]
         40          37          38          40          42          33          36
(86.6,96.6]  (96.6,109]   (109,120]        <NA>
         38          39          37          20
# Summarise bins class object
summary(bin)
        levels freq   rate
1       [21,30]   40 0.1000
2       (30,39]   37 0.0925
3       (39,48]   38 0.0950
4       (48,62]   40 0.1000
5       (62,69]   42 0.1050
6       (69,78]   33 0.0825
7     (78,86.6]   36 0.0900
8   (86.6,96.6]   38 0.0950
9    (96.6,109]   39 0.0975
10    (109,120]   37 0.0925
11         <NA>   20 0.0500
# Plot bins class object
plot(bin)
```






```
# Using labels argument
bin <- binning(carseats$Income, nbins = 4,
               labels = c("LQ1", "UQ1", "LQ3", "UQ3"))
bin
```

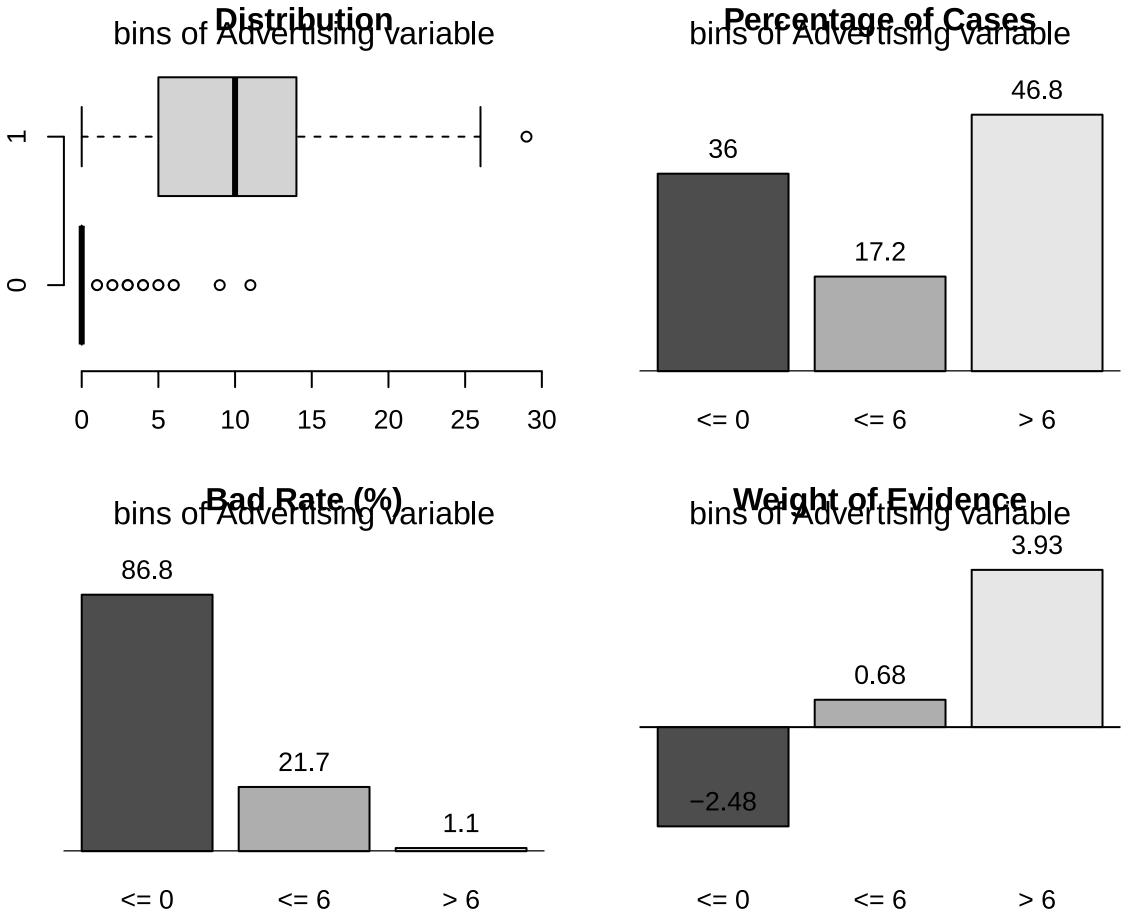
```
binned type: quantile
number of bins: 4
x
 LQ1  UQ1  LQ3  UQ3 <NA> 
  95  102   89   94   20 
# Using another type argument
binning(carseats$Income, nbins = 5, type = "equal")
binned type: equal
number of bins: 5
x
  [21,40.8] (40.8,60.6] (60.6,80.4]  (80.4,100]   (100,120]        <NA> 
         81          65          94          80          60          20 
binning(carseats$Income, nbins = 5, type = "pretty")
binned type: pretty
number of bins: 5
x
  [20,40]   (40,60]   (60,80]  (80,100] (100,120]      <NA> 
       81        65        94        80        60        20 
binning(carseats$Income, nbins = 5, type = "kmeans")
binned type: kmeans
number of bins: 5
x
    [21,49]   (49,70.5] (70.5,86.5]  (86.5,104]   (104,120]        <NA> 
        115          86          65          65          49          20 
binning(carseats$Income, nbins = 5, type = "bclust")
binned type: bclust
number of bins: 5
x
  [21,46.5] (46.5,64.5] (64.5,78.5] (78.5,95.5]  (95.5,120]        <NA> 
        109          58          63          71          79          20 

# -------------------------
# Using pipes & dplyr
# -------------------------
library(dplyr)

carseats %>%
 mutate(Income_bin = binning(carseats$Income)) %>%
 group_by(ShelveLoc, Income_bin) %>%
 summarise(freq = n()) %>%
 arrange(desc(freq)) %>%
 head(10)
Warning: Factor `Income_bin` contains implicit NA, consider using `forcats::fct_explicit_na`
# A tibble: 10 x 3
# Groups:   ShelveLoc [1]
  ShelveLoc Income_bin  freq
  <fct>     <ord>      <int>
1 Medium    [21,30]       25
2 Medium    (62,69]       24
3 Medium    (48,62]       23
4 Medium    (39,48]       21
# ... with 6 more rows
```

Optimal Binning with `binning_by()`

`binning_by()` converts a numeric variable into a categorical variable by optimal binning. This method is often used when developing a `scorecard model`.

The following `binning_by()` example optimally binning `Advertising` if `US` is a target variable with a binary class.

```
# optimal binning
bin <- binning_by(carseats, "US", "Advertising")
Warning in binning_by(carseats, "US", "Advertising"): The factor y has been changed to a
numeric vector consisting of 0 and 1.
bin
binned type: optimal
number of bins: 3
x
[-1,0]  (0,6] (6,29]
   144     69    187

# summary optimal_bins class
summary(bin)
  levels freq   rate
1 [-1,0]  144 0.3600
2  (0,6]   69 0.1725
3 (6,29]  187 0.4675

# information value
attr(bin, "iv")
[1] 4.8349

# information value table
attr(bin, "ivtable")
  Cutpoint CntRec CntGood CntBad CntCumRec CntCumGood CntCumBad PctRec GoodRate BadRate
1     <= 0    144      19    125       144         19       125 0.3600   0.1319  0.8681
2     <= 6     69      54     15       213         73       140 0.1725   0.7826  0.2174
3      > 6    187     185      2       400        258       142 0.4675   0.9893  0.0107
4  Missing      0       0      0       400        258       142 0.0000      NaN     NaN
5    Total    400     258    142        NA         NA        NA 1.0000   0.6450  0.3550
     Odds  LnOdds     WoE     IV
1  0.1520 -1.8839 -2.4810 2.0013
2  3.6000  1.2809  0.6838 0.0709
3 92.5000  4.5272  3.9301 2.7627
4     NaN     NaN     NaN    NaN
5  1.8169  0.5971  0.0000 4.8349

# visualize optimal_bins class
plot(bin, sub = "bins of Advertising variable")
```



Creating a data transformation report using `transformation_report()`

`transformation_report()` creates a data transformation report for all the variables in the data frame or objects that inherit the data frame (`tbl_df`, `tbl`, etc.).

`transformation_report()` creates a data transformation report in two forms:

- pdf file based on Latex
- html file

The contents of the report are as follows.:

- Imputation
  - Missing Values
    - Missing values imputation information
    - (variable names)
  - Outliers
    - Outliers imputation information
    - (variable names)
- Resolving Skewness
  - Skewed variables information
    - (variable names)
- Binning
  - Numerical Variables for Binning
  - Binning
    - (variable names)

- Optimal Binning
  * (variable names)

The following creates a data transformation report for `carseats`. The file format is pdf, and the file name is `Transformation_Report.pdf`.

```
carseats %>%
  transformation_report(target = US)
```

The following generates a report in html format called `transformation.html`.

```
carseats %>%
  transformation_report(target = US, output_format = "html",
    output_file = "transformation.html")
```

Data transformation reports are automated reports to assist in the data transformation process. Design data conversion scenarios by referring to the report results.

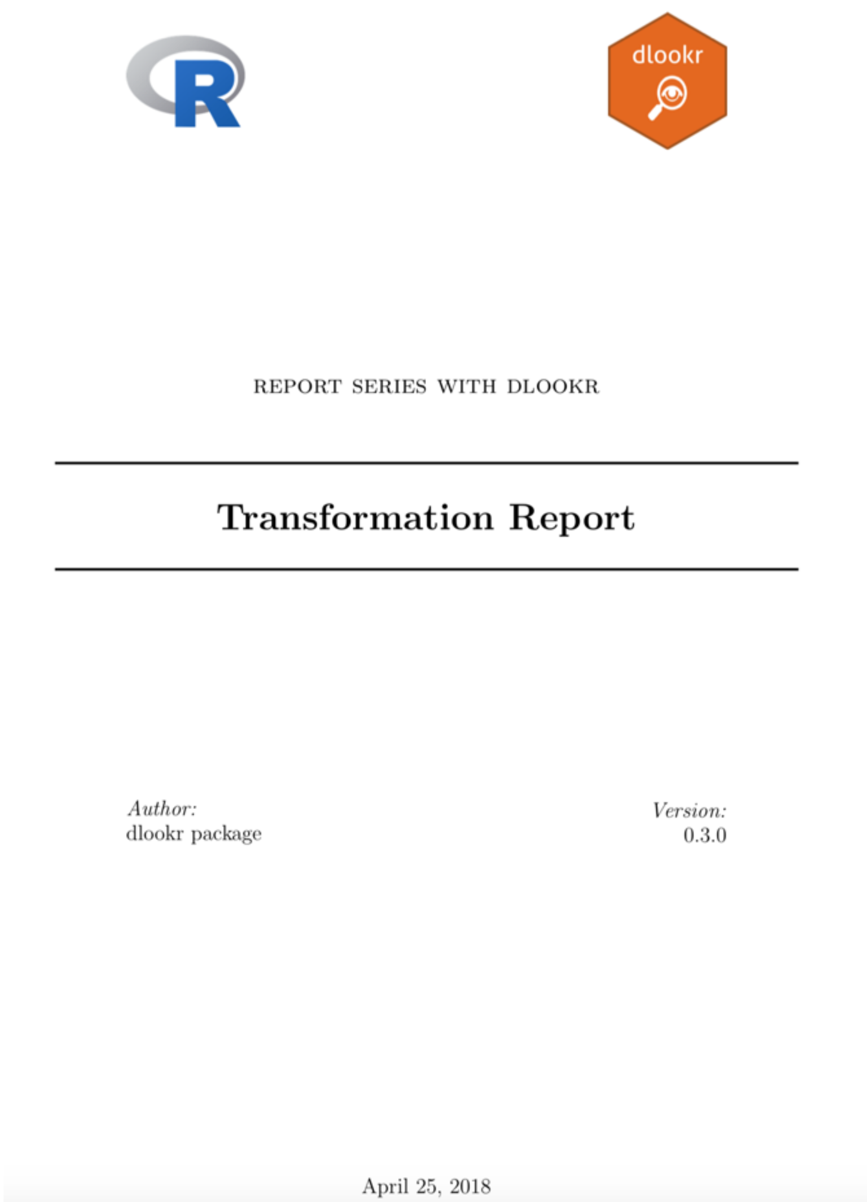
Data transformation report contents

Contents of pdf file

- The cover of the report is shown in the following figure.
- The report's argenda is shown in the following figure.
- Much of the information is displayed in tables and visualization results in reports. An example is shown in the following figure.

Contents of html file

- The title and contents of the report are shown in the following figure.
- Much information is represented in tables in the report. An example of a table in an html file is shown in the following figure.
- Binning information in the data transformation report includes visualization results. The result of the html file is shown in the following figure.



REPORT SERIES WITH DLOOKR

Transformation Report

Author:
dlookr package

Version:
0.3.0

April 25, 2018

Figure 1: Data transformation report cover

Contents

Figure 2: Table of Contents

Imputate missing values with mice

* Imputate missing values based on Multivariate Imputation by Chained Equations
- random seed : 17504

Table 1.9: Descriptive Statistics : Urban with 'mice'

	original	imputation	original_percent	imputation_percent
No	118	120	29.50	30
Yes	277	280	69.25	70
NA	5	0	1.25	0

Information of Imputation (before vs after)

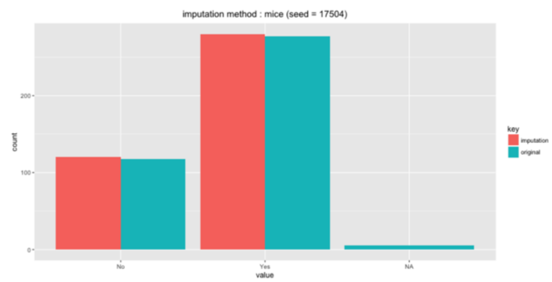


Figure 1.9: Urban - mice

Figure 3: Data Transformation Report Table and Visualization Example

Transformation Information Report

Report by dlookr package

2018-04-25

Figure 4: Data transformation report titles and table of contents

3.2.3.5 Binning with 'bclust'

levels	freq	rate
(21,28.5]	25	0.0625
(28.5,37.5]	42	0.1050
(37.5,49]	43	0.1075
(49,55.5]	15	0.0375
(55.5,67.5]	52	0.1300
(67.5,75.5]	38	0.0950
(75.5,85]	43	0.1075
(85,99.5]	49	0.1225
(99.5,108]	31	0.0775
(108,120]	38	0.0950
NA	**24**	**0.0600**

Figure 5: Report table example (Web)

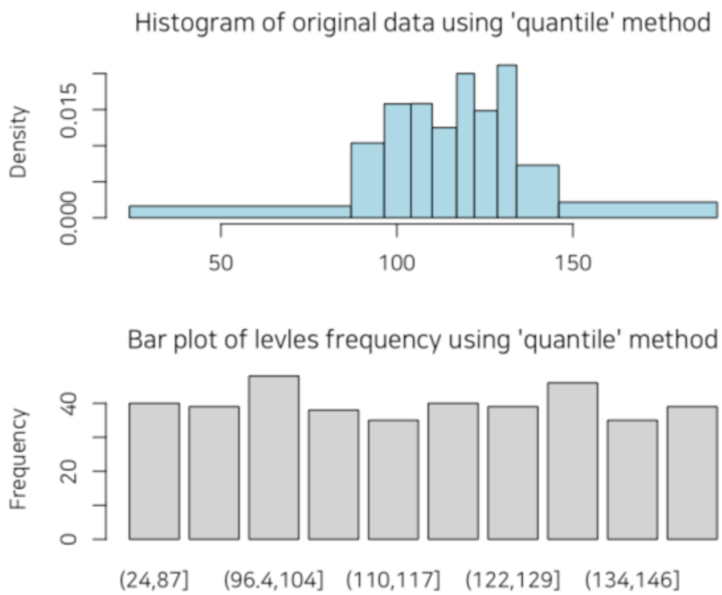


Figure 6: Data transformation report Binning information (web)