Data Mining and Machine Learning Lab 3.

Instructions: Create a file called xxxxxxxx.doc where <xxxxxxxx> is your UCD student number. Write your answers in this file and save it to your own computer so you don't lose your answers. Then upload to the moodle before the end of the lab.

At the top of the file, fill in your details below (delete the 'x' where your information goes):

Name: x
BDIC Student Number: x
UCD Student Number: x

Data exploration – identification and handling of data quality issue with the 'dlookr' Package

To illustrate basic use of the 'dlookr' package we are going to use the `flights` data from the `nycflights13` package. The `flights` data frame is data about departure and arrival on all flights departing from NYC in 2013.

```
library(nycflights13)

dim(flights)

[1] 336776     19

flights
```

```
# A tibble: 336,776 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     1      517            515         2      830
2  2013     1     1      533            529         4      850
3  2013     1     1      542            540         2      923
4  2013     1     1      544            545        -1     1004
# … with 336,772 more rows, and 12 more variables: sched_arr_time
<int>,
#   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour
<dbl>,
#   minute <dbl>, time_hour <dttm>
```

Data diagnosis

dlookr aims to diagnose the data and to select variables that can not be used for data analysis or to find the variables that need to be calibrated.:

- `diagnose()` provides basic diagnostic information for variables.
- `diagnose_category()` provides detailed diagnostic information for categorical variables.
- `diagnose_numeric()` provides detailed diagnostic information for numeric variables.
- `diagnose_outlier()` and `plot_outlier()` provide information and visualization of outliers.

General diagnosis of all variables with `diagnose()`

`diagnose()` allows you to diagnosis a variables in a data frame. Like function of dplyr, the first argument is the tibble (or data frame). The second and subsequent arguments refer to variables within that data frame.

The variables of the `tbl_df` object returned by `diagnose ()` are as follows.

- `variables` : variable name
- `types` : the data type of the variable
- `missing_count` : number of missing values
- `missing_percent` : percentage of missing values
- `unique_count` : number of unique values
- `unique_rate` : rate of unique value. unique_count / number of observation

For example, we can diagnose all variables in `flights`:

```
diagnose(flights)
```

```
# A tibble: 19 x 6
  variables types   missing_count missing_percent unique_count
unique_rate
  <chr>     <chr>           <int>           <dbl>        <int>
<dbl>
1 year      integer             0             0                1
0.00000297
2 month     integer             0             0               12
0.0000356
3 day       integer             0             0               31
0.0000920
4 dep_time  integer          8255             2.45          1319
0.00392
# … with 15 more rows
```

- `Missing Value(NA)` : Variables with very large missing values, ie those with a `missing_percent` close to 100, should be excluded from the analysis.
- `Unique value` : Variables with a unique value (`unique_count` = 1) are considered to be excluded from data analysis. And if the data type is not numeric (integer, numeric) and

the number of unique values is equal to the number of observations (unique_rate = 1), then the variable is likely to be an identifier. Therefore, this variable is also not suitable for the analysis model.

`year` can be considered not to be used in the analysis model since `unique_count` is 1. However, you do not have to remove it if you configure `date` as a combination of `year`, `month`, and `day`.

For example, we can diagnose only a few selected variables:

```
# Select columns by name
diagnose(flights, year, month, day)
```

```
# A tibble: 3 x 6
  variables types   missing_count missing_percent unique_count
unique_rate
  <chr>     <chr>           <int>           <dbl>        <int>
<dbl>
1 year      integer             0               0            1
0.00000297
2 month     integer             0               0           12
0.0000356
3 day       integer             0               0           31
0.0000920
```

```
# Select all columns between year and day (inclusive)
diagnose(flights, year:day)
```

```
# A tibble: 3 x 6
  variables types   missing_count missing_percent unique_count
unique_rate
  <chr>     <chr>           <int>           <dbl>        <int>
<dbl>
1 year      integer             0               0            1
0.00000297
2 month     integer             0               0           12
0.0000356
3 day       integer             0               0           31
0.0000920
```

```
# Select all columns except those from year to day (inclusive)
diagnose(flights, -(year:day))
```

```
# A tibble: 16 x 6
  variables    types missing_count missing_percent unique_count
unique_rate
  <chr>        <chr>         <int>           <dbl>        <int>
<dbl>
1 dep_time     inte…          8255            2.45         1319
0.00392
2 sched_dep_t… inte…             0            0            1021
0.00303
3 dep_delay    nume…          8255            2.45          528
0.00157
4 arr_time     inte…          8713            2.59         1412
```

```
0.00419
# … with 12 more rows
```

By using dplyr, variables including missing values can be sorted by the weight of missing values.:

```
flights %>%
  diagnose() %>%
  select(-unique_count, -unique_rate) %>%
  filter(missing_count > 0) %>%
  arrange(desc(missing_count))
```

```
# A tibble: 6 x 4
  variables types   missing_count missing_percent
  <chr>     <chr>           <int>           <dbl>
1 arr_delay numeric          9430            2.80
2 air_time  numeric          9430            2.80
3 arr_time  integer          8713            2.59
4 dep_time  integer          8255            2.45
# … with 2 more rows
```

Diagnosis of numeric variables with `diagnose_numeric()`

`diagnose_numeric()` diagnoses numeric(continuous and discrete) variables in a data frame. Usage is the same as `diagnose()` but returns more diagnostic information. However, if you specify a non-numeric variable in the second and subsequent argument list, the variable is automatically ignored.

The variables of the `tbl_df` object returned by `diagnose_numeric()` are as follows.

- `min` : minimum value
- `Q1` : ¼ quartile, 25th percentile
- `mean` : arithmetic mean
- `median` : median, 50th percentile
- `Q3` : ¾ quartile, 75th percentile
- `max` : maximum value
- `zero` : number of observations with a value of 0
- `minus` : number of observations with negative numbers
- `outlier` : number of outliers

Applying the summary () function to a data frame can help you figure out the distribution of data by printing `min`, `Q1`, `mean`, `median`, `Q3`, and `max` give. However, the result is that analysts can only look at it with eyes. However, returning such information as a data frame structure like `tbl_df` widens the scope of utilization.

`zero`, `minus`, and `outlier` are useful for diagnosing the integrity of data. For example, numerical data in some cases may not have 0 or a negative number. Since the hypothetical numeric variable 'employee salary' can not have a negative or zero value, you should check for zero or negative numbers in the data diagnosis process.

`diagnose_numeric()` can diagnose all numeric variables of `flights` as follows.:

```
diagnose_numeric(flights)
```

```
# A tibble: 14 x 10
  variables   min    Q1    mean median    Q3    max  zero minus
outlier
  <chr>     <dbl> <dbl>   <dbl>  <dbl> <dbl> <dbl> <int> <int>
<int>
1 year       2013  2013 2013      2013  2013  2013     0     0
0
2 month         1     4    6.55      7    10    12     0     0
0
3 day           1     8   15.7      16    23    31     0     0
0
4 dep_time      1   907 1349.     1401  1744  2400     0     0
0
# … with 10 more rows
```

If a numeric variable can not logically have a negative or zero value, it can be used with `filter()` to easily find a variable that does not logically match:

```
diagnose_numeric(flights) %>%
  filter(minus > 0 | zero > 0)
```

```
# A tibble: 3 x 10
  variables   min    Q1  mean median    Q3    max  zero   minus
outlier
  <chr>     <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <int>   <int>
<int>
1 dep_delay   -43    -5 12.6      -2    11  1301 16514  183575
43216
2 arr_delay   -86   -17  6.90     -5    14  1272  5409  188933
27880
3 minute        0     8 26.2      29    44    59 60696       0
0
```

Diagnosis of categorical variables with `diagnose_category()`

`diagnose_category()` diagnoses the categorical(factor, ordered, character) variables of a data frame. The usage is similar to `diagnose ()` but returns more diagnostic information. If you specify a non-categorical variable in the second and subsequent argument list, the variable is automatically ignored. The top argument specifies the number of levels to return per variable. The default value is 10, which returns the top 10 level. Of course, if the number of levels is less than 10, all levels are returned.

The variables of the `tbl_df` object returned by `diagnose_category()` are as follows.

- `variables` : variable names
- `levels`: level names
- `N` : Number of observation
- `freq` : Number of observation at the levles
- `ratio` : Percentage of observation at the levles

- `rank` : Rank of occupancy ratio of levels

`diagnose_category()` can diagnose all categorical variables of `flights` as follows.:

```
diagnose_category(flights)
```

```
# A tibble: 33 x 6
  variables levels      N  freq ratio  rank
  <chr>     <chr>   <int> <int> <dbl> <int>
1 carrier   UA     336776 58665  17.4     1
2 carrier   B6     336776 54635  16.2     2
3 carrier   EV     336776 54173  16.1     3
4 carrier   DL     336776 48110  14.3     4
# … with 29 more rows
```

In collaboration with `filter()` in the `dplyr` package, we can see that the `tailnum` variable is ranked in top 1 with 2,512 missing values in the case where the missing value is included in the top 10:

```
diagnose_category(flights) %>%
  filter(is.na(levels))
```

```
# A tibble: 1 x 6
  variables levels      N  freq ratio  rank
  <chr>     <chr>   <int> <int> <dbl> <int>
1 tailnum   <NA>   336776  2512 0.746     1
```

The following returns a list of levels less than or equal to 0.01%. It should be noted that the top argument has a generous specification of 500. If you use the default value of 10, values below 0.01% would not be included in the list:

```
flights %>%
  diagnose_category(top = 500)  %>%
  filter(ratio <= 0.01)
```

```
# A tibble: 10 x 6
  variables levels      N  freq   ratio  rank
  <chr>     <chr>   <int> <int>   <dbl> <int>
1 carrier   OO     336776    32 0.00950    16
2 dest      JAC    336776    25 0.00742    97
3 dest      PSP    336776    19 0.00564    98
4 dest      EYW    336776    17 0.00505    99
# … with 6 more rows
```

In the analytical model, it is also possible to consider removing the small percentage of observations in the observations or joining them together.

Diagnosing outliers with `diagnose_outlier()`

`diagnose_outlier()` diagnoses the outliers of the numeric (continuous and discrete) variables of the data frame. The usage is the same as `diagnose()`.

The variables of the `tbl_df` object returned by `diagnose_outlier()` are as follows.

- `outliers_cnt` : Count of outliers
- `outliers_ratio` : Percent of outliers
- `outliers_mean` : Arithmetic Average of outliers
- `with_mean` : Arithmetic Average of with outliers
- `without_mean` : Arithmetic Average of without outliers

`diagnose_outlier()` can diagnose anomalies of all numeric variables of `flights` as follows:

```
diagnose_outlier(flights)
```

```
# A tibble: 14 x 6
  variables outliers_cnt outliers_ratio outliers_mean with_mean
  <chr>            <int>          <dbl>         <dbl>     <dbl>
1 year                 0              0           NaN    2013
2 month                0              0           NaN       6.55
3 day                  0              0           NaN      15.7
4 dep_time             0              0           NaN    1349.
# … with 10 more rows, and 1 more variable: without_mean <dbl>
```

Numeric variables that contain anomalies are easily found with `filter()`.:

```
diagnose_outlier(flights) %>%
  filter(outliers_cnt > 0)
```

```
# A tibble: 5 x 6
  variables outliers_cnt outliers_ratio outliers_mean with_mean
  <chr>            <int>          <dbl>         <dbl>     <dbl>
1 dep_delay        43216      12.8               93.1      12.6
2 arr_delay        27880       8.28             121.        6.90
3 flight               1       0.000297        8500      1972.
4 air_time          5448       1.62             400.      151.
# … with 1 more row, and 1 more variable: without_mean <dbl>
```

The following is a list of numeric variables with anomalies greater than 5%.:

```
diagnose_outlier(flights) %>%
  filter(outliers_ratio > 5) %>%
  mutate(rate = outliers_mean / with_mean) %>%
  arrange(desc(rate)) %>%
  select(-outliers_cnt)
```

```
# A tibble: 2 x 6
  variables outliers_ratio outliers_mean with_mean without_mean
rate
  <chr>              <dbl>         <dbl>     <dbl>        <dbl>
<dbl>
1 arr_delay           8.28          121.      6.90        -3.69
17.5
2 dep_delay          12.8            93.1    12.6          0.444
7.37
```

If the outlier is larger than the average of all observations, it may be desirable to replace or remove the outlier in the data analysis process.

Visualization of outliers using `plot_outlier()`

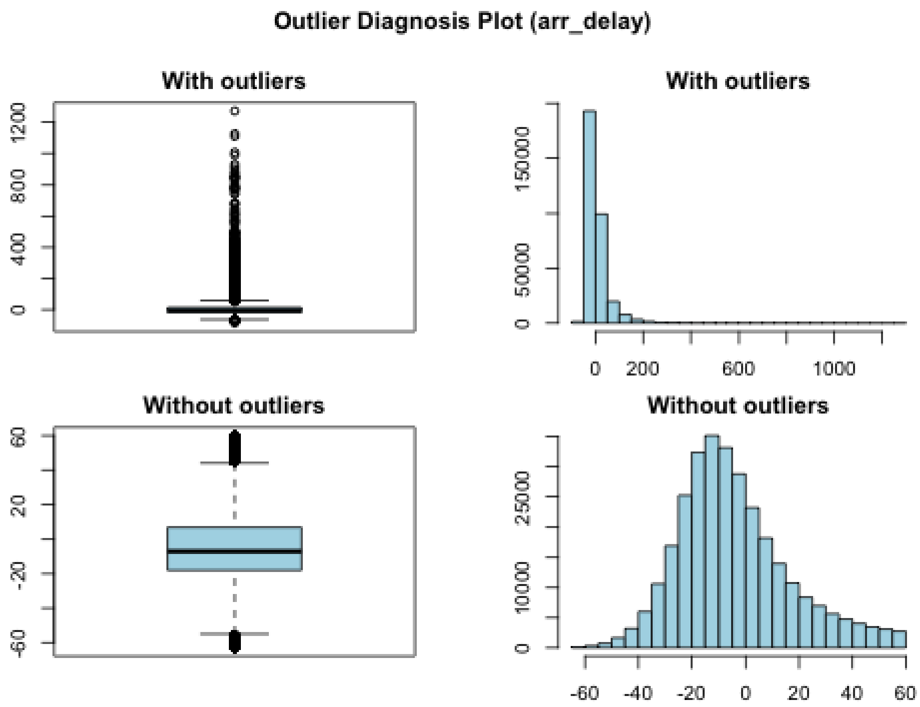
`plot_outlier()` visualizes outliers of numarical variables(continious and discrete) of data.frame. Usage is the same `diagnose()`.

The plot derived from the numerical data diagnosis is as follows.

- With outliers box plot
- Without outliers box plot
- With outliers histogram
- Without outliers histogram

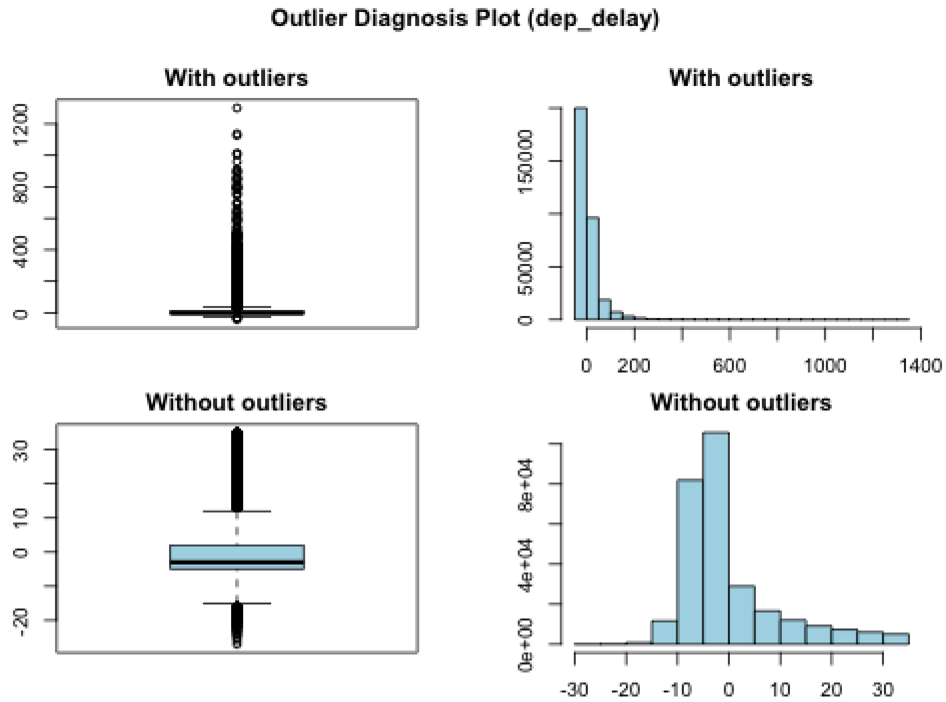
`plot_outlier()` can visualize an anomaly in the `arr_delay` variable of `flights` as follows:

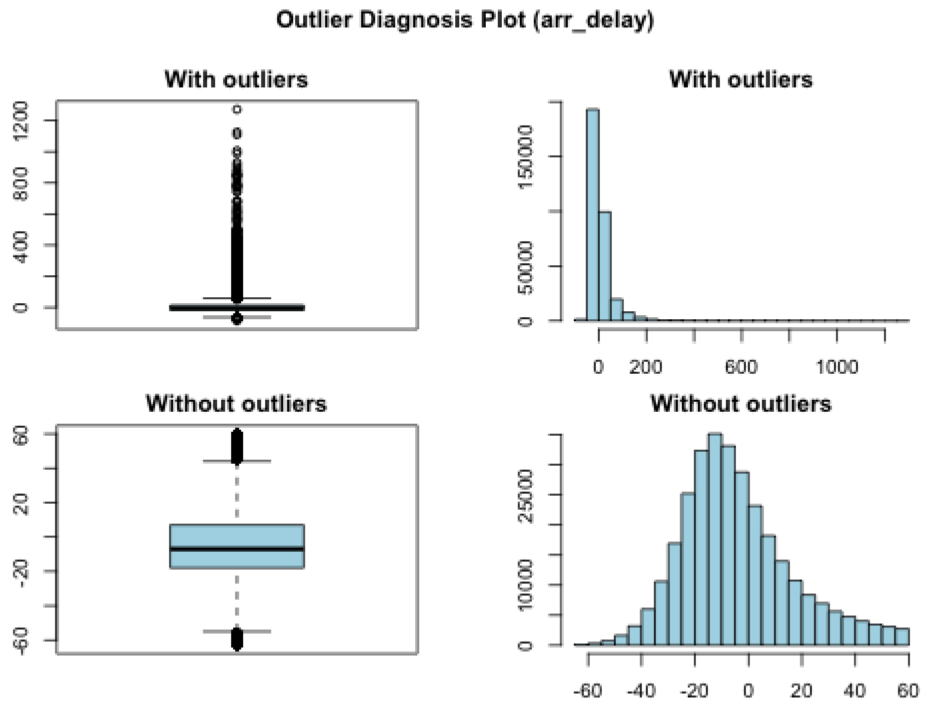
```
flights %>%
  plot_outlier(arr_delay)
```

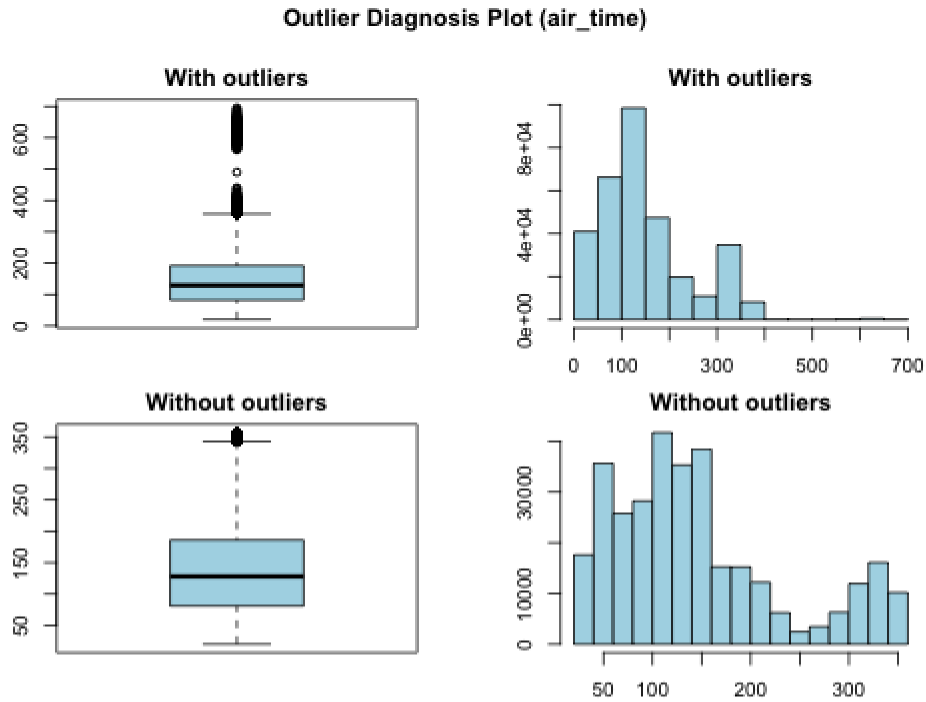


Use the function of the dplyr package and `plot_outlier()` and `diagnose_outlier()` to visualize anomaly values of all numeric variables with an outlier ratio of 0.5% or more.:

```
flights %>%
  plot_outlier(diagnose_outlier(flights) %>%
               filter(outliers_ratio >= 0.5) %>%
               select(variables) %>%
               unlist())
```


Outlier Diagnosis Plot (dep_delay)
With outliers
With outliers
Without outliers
Without outliers
0
400
800
1200
0
50000
150000
0
200
600
1000
1400
-20
0
10
30
0e+00
4e+04
8e+04
-30
-20
-10
0
10
20
30


Outlier Diagnosis Plot (arr_delay)
With outliers
With outliers
Without outliers
Without outliers
0
400
800
1200
0
50000
150000
0
200
600
1000
-60
-20
20
60
0
10000
25000
-60
-40
-20
0
20
40
60



You should look at the visualization results and decide whether to remove or replace the outliers. In some cases, it is important to consider removing the variables that contain anomalies from the data analysis model.

In the visualization results, `arr_delay` has similar distributions to the normal distribution of the observed values. In the case of linear models, we can also consider removing or replacing anomalies. And `air_time` shows a roughly similar distribution before and after removing anomalies.

For more details on the dlookr package see the pdf on Moodle (dlookr.pdf) and

https://cran.r-project.org/web/packages/dlookr/vignettes/diagonosis.html