MA213 Basic Statistics and Probability - Lab2 Guide

Lab 2: Data transformation

Data transformation is a crucial part in exploratory data analysis (EDA). EDA includes the processes of importing, cleaning, and visualizing data to uncover the patterns, relationships and hidden insights within the data.

In this lab, we will learn data processing and subsetting to see more insights from the data.

Learning Objectives

- Use R for Data Management and Exploration: Utilize R to load, pre-process, and explore data through visualization and summarization techniques.
- Classify and Analyze Variables: Categorize variables based on their types (e.g., numerical/categorical, continuous/discrete, ordinal), assess their association (positive, negative, or independent), and determine which make sense as explanatory vs. response variables.

Load packages for Data

If you didn't install `tidyverse` package

```
install.packages("tidyverse")
```

Load package

```
library(tidyverse)
```

New York Air Quality Data

Daily air quality measurements in New York, May to September 1973 stored within a data frame with 153 observations on 6 variables. It is obtained from the New York State Department of Conservation (ozone data) and the National Weather Service (meteorological data). It is cited by Chambers (2018).

You can find the details fo the data description here.

Import the Data

You can use `airquality` object from base R data. You can also `airquality.csv` file where the data is stored as a comma separated value (CSV) file. Each row holds information for a single observation.

Import the data from `airquality.csv` and show first 6 rows to see how this data is organized.

```
df = read.csv("Lab2/data/airquality.csv")
```

`glimpse()` to take a quick peek at your data

```
glimpse(df)
```

```
## Rows: 153
## Columns: 7
## $ X       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,~
## $ Ozone   <int> 41, 36, 12, 18, NA, 28, 23, 19, 8, NA, 7, 16, 11, 14, 18, 14, ~
## $ Solar.R <int> 190, 118, 149, 313, NA, NA, 299, 99, 19, 194, NA, 256, 290, 27~
## $ Wind    <dbl> 7.4, 8.0, 12.6, 11.5, 14.3, 14.9, 8.6, 13.8, 20.1, 8.6, 6.9, 9~
## $ Temp    <int> 67, 72, 74, 62, 56, 66, 65, 59, 61, 69, 74, 69, 66, 68, 58, 64~
## $ Month   <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,~
## $ Day     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,~
```

To view the names of the variables

```
names(df)
```

```
## [1] "X"       "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

To view the first 6 rows

```
head(df)
```

```
##   X Ozone Solar.R Wind Temp Month Day
## 1 1    41     190  7.4   67     5   1
## 2 2    36     118  8.0   72     5   2
## 3 3    12     149 12.6   74     5   3
## 4 4    18     313 11.5   62     5   4
## 5 5    NA      NA 14.3   56     5   5
## 6 6    28      NA 14.9   66     5   6
```

The pipe operator

`dplyr` (one of data packages in `tidyverse`) provides the `%>%` operator from magrittr. `x %>% f(y)` turns into `f(x, y)` so the result from one step is then "piped" into the next step.

`df %>% filter(Month==5)` means "Select all rows with May"

```
df %>% filter(Month==5)
```

```
##     X Ozone Solar.R Wind Temp Month Day
## 1   1    41     190  7.4   67     5   1
## 2   2    36     118  8.0   72     5   2
## 3   3    12     149 12.6   74     5   3
## 4   4    18     313 11.5   62     5   4
## 5   5    NA      NA 14.3   56     5   5
## 6   6    28      NA 14.9   66     5   6
## 7   7    23     299  8.6   65     5   7
## 8   8    19      99 13.8   59     5   8
## 9   9     8      19 20.1   61     5   9
## 10 10    NA     194  8.6   69     5  10
## 11 11     7      NA  6.9   74     5  11
## 12 12    16     256  9.7   69     5  12
## 13 13    11     290  9.2   66     5  13
## 14 14    14     274 10.9   68     5  14
## 15 15    18      65 13.2   58     5  15
## 16 16    14     334 11.5   64     5  16
## 17 17    34     307 12.0   66     5  17
```

```
## 18 18     6      78 18.4   57     5  18
## 19 19    30     322 11.5   68     5  19
## 20 20    11      44  9.7   62     5  20
## 21 21     1       8  9.7   59     5  21
## 22 22    11     320 16.6   73     5  22
## 23 23     4      25  9.7   61     5  23
## 24 24    32      92 12.0   61     5  24
## 25 25    NA      66 16.6   57     5  25
## 26 26    NA     266 14.9   58     5  26
## 27 27    NA      NA  8.0   57     5  27
## 28 28    23      13 12.0   67     5  28
## 29 29    45     252 14.9   81     5  29
## 30 30   115     223  5.7   79     5  30
## 31 31    37     279  7.4   76     5  31
```

The pipe operator can be added

You can use the pipe to rewrite multiple operations that you can read left-to-right, top-to-bottom (reading the pipe operator as "then") (`%>%` and `|>` are equivalent)

```
df |>
  select(Ozone, Wind, Month) |>
  filter(Month==5)
```

```
##    Ozone Wind Month
## 1     41  7.4     5
## 2     36  8.0     5
## 3     12 12.6     5
## 4     18 11.5     5
## 5     NA 14.3     5
## 6     28 14.9     5
## 7     23  8.6     5
## 8     19 13.8     5
## 9      8 20.1     5
## 10    NA  8.6     5
## 11     7  6.9     5
## 12    16  9.7     5
## 13    11  9.2     5
## 14    14 10.9     5
## 15    18 13.2     5
## 16    14 11.5     5
## 17    34 12.0     5
## 18     6 18.4     5
## 19    30 11.5     5
## 20    11  9.7     5
## 21     1  9.7     5
## 22    11 16.6     5
## 23     4  9.7     5
## 24    32 12.0     5
## 25    NA 16.6     5
## 26    NA 14.9     5
## 27    NA  8.0     5
## 28    23 12.0     5
## 29    45 14.9     5
## 30   115  5.7     5
```

```
## 31    37  7.4     5
#
```

Dplyr data manipulation functions

- Rows:
  - `filter()` chooses rows based on column values.
  - `slice()` chooses rows based on location.
  - `arrange()` changes the order of the rows.
- Columns:
  - `select()` changes whether or not a column is included.
  - `rename()` changes the name of columns.
  - `mutate()` changes the values of columns and creates new columns.
  - `relocate()` changes the order of the columns.
- Groups of rows:
  - `summarise()` collapses a group into a single row.

Some useful resources

https://openintrostat.github.io/oilabs-tidy/02_intro_to_data/intro_to_data.html

https://dplyr.tidyverse.org/articles/dplyr.html

https://rstudio.github.io/cheatsheets/html/data-transformation.html

Creating new column using Dplyr

You can create new columns or modify existing ones using the `mutate()` function.

For example, to create a new column that categorizes temperature:

```
df <- df %>%
  mutate(Temp_Category = ifelse(Temp > 80, "High", "Moderate")) # if Temp>80 -> "High" otherwise "Moder

head(df)
```

```
##   X Ozone Solar.R Wind Temp Month Day Temp_Category
## 1 1    41     190  7.4   67     5   1      Moderate
## 2 2    36     118  8.0   72     5   2      Moderate
## 3 3    12     149 12.6   74     5   3      Moderate
## 4 4    18     313 11.5   62     5   4      Moderate
## 5 5    NA      NA 14.3   56     5   5      Moderate
## 6 6    28      NA 14.9   66     5   6      Moderate
```

Contingency Table using dplyr

A contingency table displays the frequency distribution of variables. You can create contingency tables using `dplyr` in combination with `tidyr`.

For example, to create a contingency table of Month vs. Ozone_Level:

```r
df <- df %>%
  mutate(Ozone_Level = ifelse(Ozone > median(Ozone, na.rm=TRUE), "High", "Low"))

contingency_table <- df %>%
  group_by(Month, Ozone_Level) %>%
  summarise(Count = n()) %>%
  pivot_wider(names_from = Ozone_Level, values_from = Count)
```

```
## `summarise()` has grouped output by
## 'Month'. You can override using the
## `.groups` argument.
```

```r
print(contingency_table)
```

```
## # A tibble: 5 x 4
## # Groups:   Month [5]
##   Month  High   Low  `NA`
##   <int> <int> <int> <int>
## 1     5     7    19     5
## 2     6     3     6    21
## 3     7    21     5     5
## 4     8    18     8     5
## 5     9     9    20     1
```

Contingency Table with new categorized variable

You can categorize `solar.R` by

```
Solar.R <= 115       : Low
115< Solar.R <= 185  : Medium
258< Solar.R         : High
```

Create a contingency table for categorized `solar.R` vs `Ozone_level`

```r
 contingency_table2 <- df %>%
  mutate(solar_cat = case_when(
    Solar.R <= 115 ~ "Low",
    Solar.R > 115 & Solar.R <= 185 ~ "Medium",
    Solar.R > 258 ~ "High"
  )) %>%
  group_by(solar_cat, Ozone_Level) %>%
  summarise(Count = n()) %>%
  pivot_wider(
    names_from = Ozone_Level,
    values_from = Count
  )
```

```
## `summarise()` has grouped output by
## 'solar_cat'. You can override using the
## `.groups` argument.
```

```r
 print(contingency_table2)
```

```
## # A tibble: 4 x 4
## # Groups:   solar_cat [4]
##   solar_cat  High   Low  `NA`
##   <chr>     <int> <int> <int>
```

```
## 1 High          15    12    10
## 2 Low            6    23     8
## 3 Medium         7     7     8
## 4 <NA>          30    16    11
```

Filtering out missing values

We have `NA` values in `Ozone`, `Solar.R` and so on.

```
print(head(df))
```

```
##   X Ozone Solar.R Wind Temp Month Day Temp_Category Ozone_Level
## 1 1    41     190  7.4   67     5   1      Moderate        High
## 2 2    36     118  8.0   72     5   2      Moderate        High
## 3 3    12     149 12.6   74     5   3      Moderate         Low
## 4 4    18     313 11.5   62     5   4      Moderate         Low
## 5 5    NA      NA 14.3   56     5   5      Moderate        <NA>
## 6 6    28      NA 14.9   66     5   6      Moderate         Low
```

Let's filter out those `NA` values from `df` and make it to `filtered_df` .

1. Filter out based on specific column. For example, we are going to remove `NA` values based on `Ozone` column.

```
filtered_df <- df %>%
  filter(!is.na(Ozone))


print(filtered_df[1:10,])
```

```
##     X Ozone Solar.R Wind Temp Month Day Temp_Category Ozone_Level
## 1   1    41     190  7.4   67     5   1      Moderate        High
## 2   2    36     118  8.0   72     5   2      Moderate        High
## 3   3    12     149 12.6   74     5   3      Moderate         Low
## 4   4    18     313 11.5   62     5   4      Moderate         Low
## 5   6    28      NA 14.9   66     5   6      Moderate         Low
## 6   7    23     299  8.6   65     5   7      Moderate         Low
## 7   8    19      99 13.8   59     5   8      Moderate         Low
## 8   9     8      19 20.1   61     5   9      Moderate         Low
## 9  11     7      NA  6.9   74     5  11      Moderate         Low
## 10 12    16     256  9.7   69     5  12      Moderate         Low
```

2. Filter out based on multiple columns. For example, we are going to remove `NA` values based on `Ozone` and `Solar.R`columns.

```
filtered_df <- df %>%
  filter(!is.na(Ozone)) %>%
  filter(!is.na(Solar.R))

print(filtered_df[1:10,])
```

```
##     X Ozone Solar.R Wind Temp Month Day Temp_Category Ozone_Level
## 1   1    41     190  7.4   67     5   1      Moderate        High
## 2   2    36     118  8.0   72     5   2      Moderate        High
## 3   3    12     149 12.6   74     5   3      Moderate         Low
## 4   4    18     313 11.5   62     5   4      Moderate         Low
## 5   7    23     299  8.6   65     5   7      Moderate         Low
## 6   8    19      99 13.8   59     5   8      Moderate         Low
```

```
## 7   9     8      19 20.1   61     5   9      Moderate         Low
## 8  12    16     256  9.7   69     5  12      Moderate         Low
## 9  13    11     290  9.2   66     5  13      Moderate         Low
## 10 14    14     274 10.9   68     5  14      Moderate         Low
```

```
# filtered_df <- df %>%
#   filter(!is.na(Ozone) & !is.na(Solar.R)). ## this is equivalent code
```

3. Filter out based on all the columns.

```
filtered_df <- df %>%
  filter(complete.cases(.))

print(filtered_df[1:10,])
```

```
##     X Ozone Solar.R Wind Temp Month Day Temp_Category Ozone_Level
## 1   1    41     190  7.4   67     5   1      Moderate        High
## 2   2    36     118  8.0   72     5   2      Moderate        High
## 3   3    12     149 12.6   74     5   3      Moderate         Low
## 4   4    18     313 11.5   62     5   4      Moderate         Low
## 5   7    23     299  8.6   65     5   7      Moderate         Low
## 6   8    19      99 13.8   59     5   8      Moderate         Low
## 7   9     8      19 20.1   61     5   9      Moderate         Low
## 8  12    16     256  9.7   69     5  12      Moderate         Low
## 9  13    11     290  9.2   66     5  13      Moderate         Low
## 10 14    14     274 10.9   68     5  14      Moderate         Low
```

Chambers, John M. 2018. *Graphical Methods for Data Analysis*. Chapman; Hall/CRC.