Hands-On Data Analysis for ININ Using R

Prof. Dr. Cornelia Storz, M.Sc. Fei (Michael) Wang

Management and Microeconomics, Goethe-Universität Frankfurt

This document was prepared for students who are taking ININ course and planning to take the exam. It is a collection of notes and codes for the course. The notes are based on the tutorials we had in the course. I am trying to make it concise and easy to understand. I hope it can help you to review the course and prepare for the exam. *We are living in a very noisy world, therefore let's keep it simple and clear.* I setup a challenge for myself to deliver a clear and concise review notes within 15 pages. This brings the trade-off, which means some figures and tables are not included in the notes. Therefore, you have to run the codes to see the results.

I hope you enjoy reading it. I also hope you will have this notes with you whenever you want to do some data analysis. If one day, you still refer to this notes and find it still useful, I would be very happy to hear that.

Keywords: econometrics, data analysis, regression models, empirical research, innovation, management

---

Contents

1 Introduction

All statistical or econometric or machine learning models are based on the following assumptions:

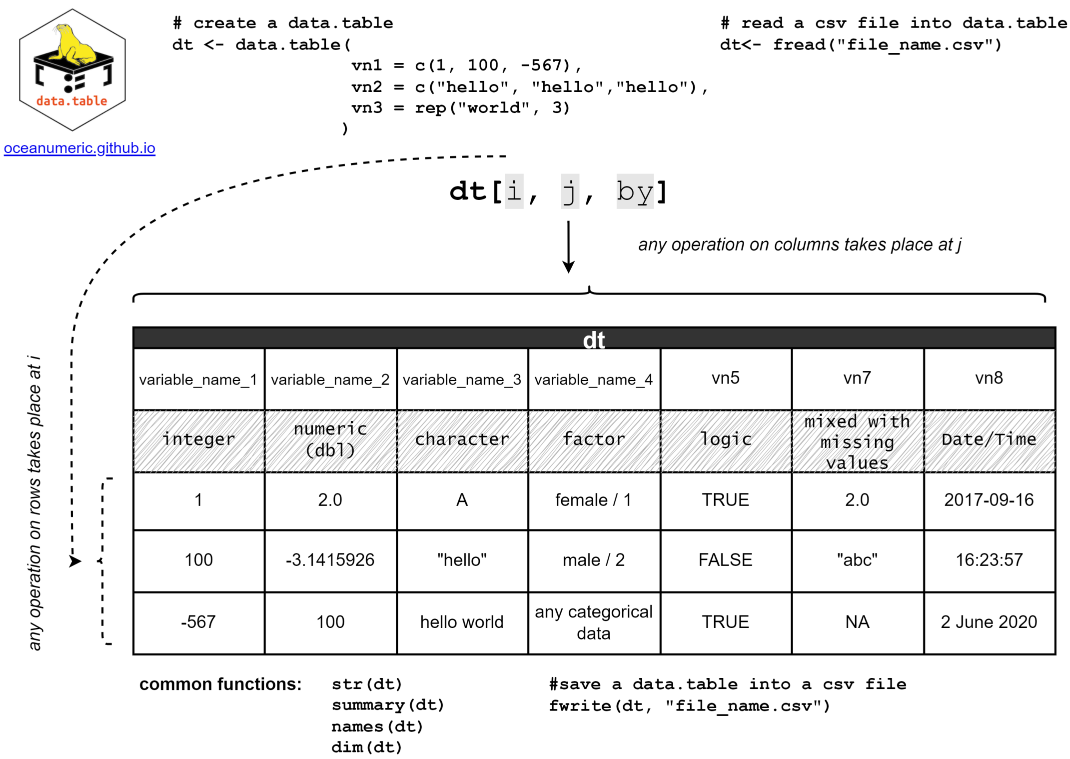
- there are something we know - **data**
- and something we don't know - **error** ε.

In summary, according to confucius, *to know what we know and what we do not know*, that is called **wisdom**. Or like Plato said, *I know that I know nothing*. To help you to review the course, the notes will be organized as follows:

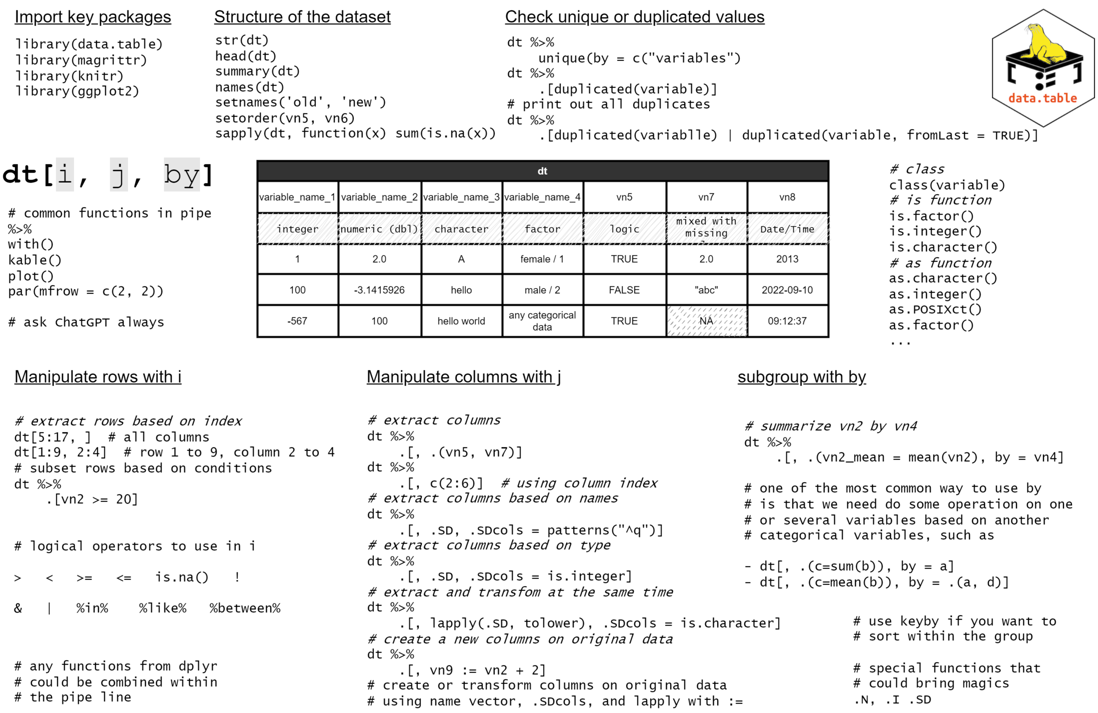
1. **Data**: using `data.table` to get familiar with the data
2. **Simple linear regression**: how to estimate a simple linear regression model, how to interpret the results
3. **Multiple linear regression**: how to estimate a multiple linear regression model, how to interpret the results, how to test the model
4. **Introduction to logistic regression**: why do we need logistic regression
5. **Data manipulation**: will not be tested in the exam, but it is very useful for your future work or research

2 Introduction to Data and `data.table`

Broadly speaking, there are two kinds of data: **structured data** and **unstructured data**. Structured data is data that has a structure, such as a table, whereas unstructured data is data that does not have a structure, such as a text file. In this course, we focus on structured data. This means all the data we will use look like tables, such as the following one:



The basic syntax of data.table is summarized in the following illustration. **You will not be tested on the syntax of data.table in the exam**. However, you will be tested on the underlying concepts of data.table, such as the type of variables (integer, character, factor, etc.). In the future if you will be working as a data scientist, you can use data.table to do big data analysis. You will need to know the syntax of data.table for practical use not for the exam.



Import key packages

```
library(data.table)
library(magrittr)
library(knitr)
library(ggplot2)
```

Structure of the dataset

```
str(dt)
head(dt)
summary(dt)
names(dt)
setnames('old', 'new')
setorder(vn5, vn6)
sapply(dt, function(x) sum(is.na(x)))
```

Check unique or duplicated values

```
dt %>%
    unique(by = c("variables")
dt %>%
    .[duplicated(variable)]
# print out all duplicates
dt %>%
    .[duplicated(variablle) | duplicated(variable, fromLast = TRUE)]
```

dt[i, j, by]

```
# common functions in pipe
%>%
with()
kable()
plot()
par(mfrow = c(2, 2))

# ask ChatGPT always
```

dt						
variable_name_1	variable_name_2	variable_name_3	variable_name_4	vn5	vn7	vn8
integer	numeric (dbl)	character	factor	logic	mixed with missing	Date/Time
1	2.0	A	female / 1	TRUE	2.0	2013
100	-3.1415926	hello	male / 2	FALSE	"abc"	2022-09-10
-567	100	hello world	any categorical data	TRUE	NA	09:12:37

```
# class
class(variable)
# is function
is.factor()
is.integer()
is.character()
# as function
as.character()
as.integer()
as.POSIXct()
as.factor()
...
```

Manipulate rows with i

```
# extract rows based on index
dt[5:17, ]  # all columns
dt[1:9, 2:4]  # row 1 to 9, column 2 to 4
# subset rows based on conditions
dt %>%
    .[vn2 >= 20]


# logical operators to use in i

>   <   >=   <=   is.na()   !

&   |   %in%   %like%   %between%


# any functions from dplyr
# could be combined within
# the pipe line
```

Manipulate columns with j

```
# extract columns
dt %>%
    .[, .(vn5, vn7)]
dt %>%
    .[, c(2:6)]  # using column index
# extract columns based on names
dt %>%
    .[, .SD, .SDcols = patterns("^q")]
# extract columns based on type
dt %>%
    .[, .SD, .SDcols = is.integer]
# extract and transfom at the same time
dt %>%
    .[, lapply(.SD, tolower), .SDcols = is.character]
# create a new columns on original data
dt %>%
    .[, vn9 := vn2 + 2]
# create or transform columns on original data
# using name vector, .SDcols, and lapply with :=
```

subgroup with by

```
# summarize vn2 by vn4
dt %>%
    .[, .(vn2_mean = mean(vn2), by = vn4]

# one of the most common way to use by
# is that we need do some operation on one
# or several variables based on another
# categorical variables, such as

- dt[, .(c=sum(b)), by = a]
- dt[, .(c=mean(b)), by = .(a, d)]

# use keyby if you want to
# sort within the group

# special functions that
# could bring magics
.N, .I .SD
```

Let's start from installing and loarding the packages we need for the course.

```
# install packages
install.packages("stargazer")
# install ISLR if you don't have it
# install.packages("ISLR")
install.packages("corrplot")
# sometimes you need to install other packages
# hopefully you can figure it out by yourself
```

After you install the packages, you can load them into R.

```
# library for data analysis
library(data.table)
library(magrittr)
library(ggplot2)
library(knitr)
library(stargazer)
library(MASS)
library(ISLR)
library(corrplot)
```

Now, we can load the data into R and manipulate the data.

```
# read the dataset
cis ← fread("https://shorturl.at/wBESZ")
# check structure of the dataset
str(cis)
# check the first 10 rows of the dataset
head(cis, 10)
```

```r
# check the number of missing values in each column
cis %>%
    .[, .SD, .SDcols = is.double] %>%
    # check the number of missing values in each column
    sapply(function(x) sum(is.na(x))) %>%
    # sort the number of missing values in each column
    sort(decreasing = TRUE) %>%
    # convert to data.table and keep rownames as a column
    as.data.table(keep.rownames = TRUE) %>%
    # set variable names
    setnames(c("variables", "Numbe of NAs")) %>%
    # check the first 10
    head(10) %>%
    kable("pipe)
```

2.1 Data Visualization

It is important to visualize the data before you start to do the analysis. To choose the right figure, you need to know the type of variables. Here is the summary:

- **Categorical variables**: bar chart, pie chart, etc.
- **Continuous variables**: histogram, boxplot, etc.
- **Categorical vs. continuous variables**: boxplot or violin or histogram with different colors

Here is a demo of how to visualize the data. You can try to run the code and see the results.

```r
# four figures in one page
# bar chart, histogram, box plot and box plot compared
# figure size
options(repr.plot.width = 10, repr.plot.height = 10)
par(mfrow = c(2, 2))
cis %>%
    # group by branche and .N calculates the number of frequency
    .[, .N, by = branche] %>%
    # order it in a descending way
    .[order(-N)] %>%
    # get the top 10 branche (industry)
    head(10) %>%
    # plot it
    with(pie(N, labels = branche, main = "Distribution of Industries"))

# histgorma for number of employees
cis %>%
    with(hist(bges, main = "number of employees"))
# boxplot for sales
cis %>%
    with(boxplot(um18,  main = "Boxplot of Sales in 2018"))
# boxplot for log(1+sales)
cis %>%
    with(boxplot(log(1+um18), main = "Boxplot of Log Transform "))
```

When you visualize the data, it is important to check two things for continuous variables:

- **shape**: whether the distribution is symmetric or skewed (ideally, we prefer symmetric distribution)
- **outliers**: outliers are extreme values that deviate from other observations on data , they may indicate a variability in a measurement, experimental errors or a novelty.

2.2 Log Transformation

Log transformation is a very useful tool to deal with skewed data. When you have a skewed data, you can try to log transform it. However, it could be tricky. You need to know the underlying theory of log transformation. Generally speaking, log transformation is used to make the data more symmetric. To avoid the negative values, we usually use $\log(1+x)$ instead of $\log(x)$.

```
# log transform
options(repr.plot.width = 10, repr.plot.height = 10)
par(mfrow = c(2, 2))
cis %>%
    with(hist(um18, main = "Histgoram of Sales (2018)"))

cis %>%
    with(hist(log(1+um18), main = "Histogram of Log (1+sales)"))

cis %>%
    with(boxplot(um18, main = "Boxplot of Sales (2018)"))
cis %>%
    with(boxplot(log(1+um18), main = "Boxplot of Log Transform "))
```

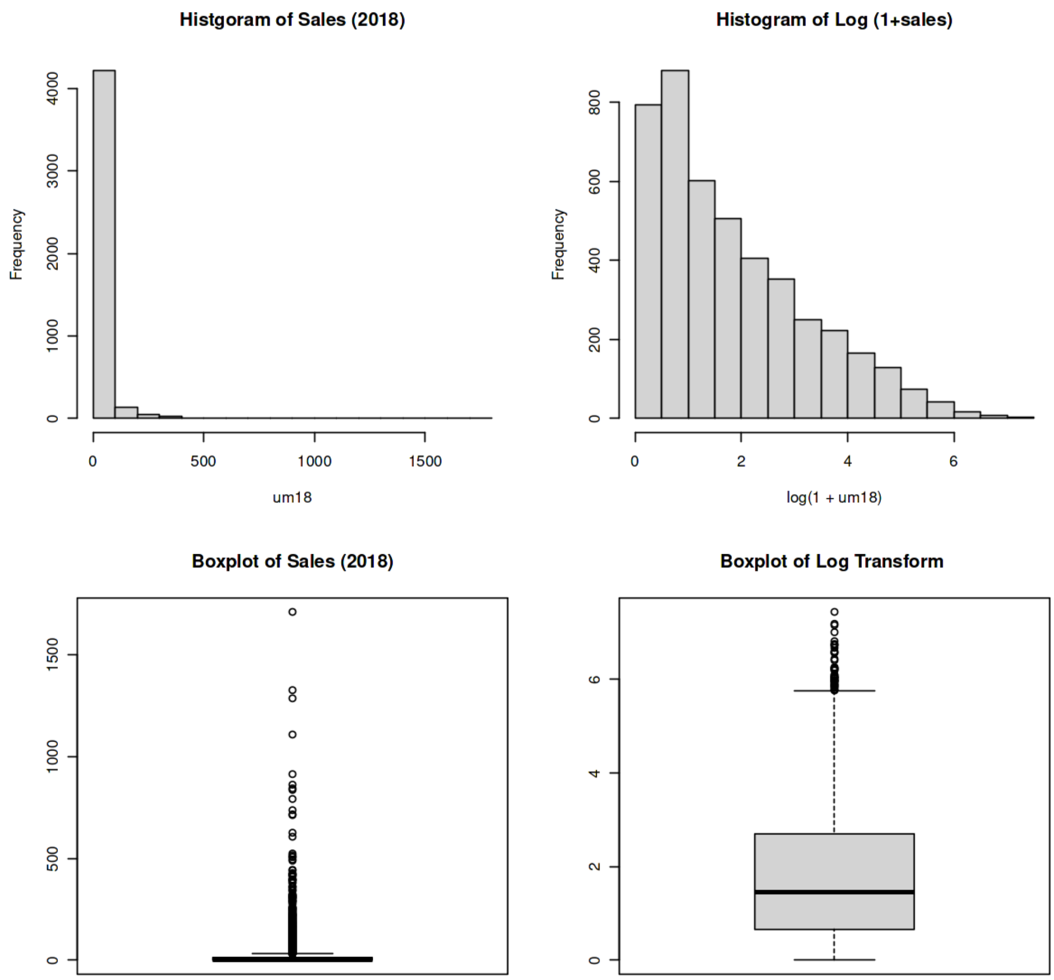


Figure 1: Log transformation of sales