

R Basics 2 -Data-

Instructor: Yuta Toyama

Last updated: 2020-03-31

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.6.2
```

```
hook_output = knit_hooks$get('output')
knit_hooks$set(output = function(x, options) {
  # this hook is used only when the linewidth option is not NULL
  if (!is.null(n <- options$linewidth)) {
    x = knitr:::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n)) x = strwrap(x, width = n)
    x = paste(x, collapse = '\n')
  }
  hook_output(x, options)
})
```

```
knitr::opts_chunk$set(collapse = FALSE, tidy = FALSE, linewidth = 75)
```

Section 1

Data frame

Acknowledgement

This note is largely based on Applied Statistics with R.
<https://davidalpiaz.github.io/appliedstats/>

Introduction

- ▶ A **data frame** is the most common way that we store and interact with data in this course.

```
example_data = data.frame(x = c(1, 3, 5, 7, 9, 1, 3, 5, 7, 9),  
                           y = c(rep("Hello", 9), "Goodbye"),  
                           z = rep(c(TRUE, FALSE), 5))
```

- ▶ A data frame is a **list** of vectors.
 - ▶ Each vector must contain the same data type
 - ▶ The difference vectors can store different data types.

example_data

```
## # A tibble: 10 x 3
##       x y       z
##   <dbl> <fct> <lgl>
## 1     1 1 Hello TRUE
## 2     2 3 Hello FALSE
## 3     3 5 Hello TRUE
## 4     4 7 Hello FALSE
## 5     5 9 Hello TRUE
## 6     6 1 Hello FALSE
## 7     7 3 Hello TRUE
## 8     8 5 Hello FALSE
## 9     9 7 Hello TRUE
## 10    9 9 Goodbye FALSE
```

- ▶ `write.csv` save (or export) the dataframe in `.csv` format.

Load csv file

- ▶ We can also import data from various file types into R, as well as use data stored in packages.
- ▶ Read csv file into R.
 - ▶ `read.csv()` function as default
 - ▶ `read_csv()` function from the `readr` package. This is faster for larger data.

```
# install.packages("readr")  
#library(readr)  
#example_data_from_csv = read_csv("example-data.csv")  
example_data_from_csv = read.csv("example-data.csv")
```


- ▶ Note: This particular line of code assumes that the file `example_data.csv` exists in your current working directory.
- ▶ The current working directory is the folder that you are working with. To see this, you type

```
getwd()
```

```
## [1]  
"C:/Users/Yuta/Dropbox/Teaching/2020_1_3_4_Applied_Metrics/Note_Github/02_R"
```

- ▶ If you want to set the working directory, use `setwd()` function

```
setwd(dir = "directory path" )
```

Examine dataframe

- ▶ Inside the `ggplot2` package is a dataset called `mpg`. By loading the package using the `library()` function, we can now access `mpg`.

```
library(ggplot2)
```

- ▶ Three things we would generally like to do with data:
 - ▶ Look at the raw data.
 - ▶ Understand the data. (Where did it come from? What are the variables? Etc.)
 - ▶ Visualize the data.
- ▶ To look at the data, we have two useful commands: `head()` and `str()`

```
head(mpg, n = 10)
```

```
## # A tibble: 10 x 11
## manufacturer model displ year cyl trans drv cty hwy fl class
## <chr> <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
## 1 audi a4 1.8 1999 4 auto(l~ f 18 29 p comp~
## 2 audi a4 1.8 1999 4 manual~ f 21 29 p comp~
## 3 audi a4 2 2008 4 manual~ f 20 31 p comp~
## 4 audi a4 2 2008 4 auto(a~ f 21 30 p comp~
## 5 audi a4 2.8 1999 6 auto(l~ f 16 26 p comp~
## 6 audi a4 2.8 1999 6 manual~ f 18 26 p comp~
## 7 audi a4 3.1 2008 6 auto(a~ f 18 27 p comp~
## 8 audi a4 quat~ 1.8 1999 4 manual~ 4 18 26 p comp~
## 9 audi a4 quat~ 1.8 1999 4 auto(l~ 4 16 25 p comp~
## 10 audi a4 quat~ 2 2008 4 manual~ 4 20 28 p comp~
```

- ▶ The function `str()` will display the “structure” of the data frame.
 - ▶ It will display the number of **observations** and **variables**, list the variables, give the type of each variable, and show some elements of each variable.
 - ▶ This information can also be found in the “Environment” window in RStudio.

```
str(mpg)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame': 234 obs. of 11 variables:
## $ manufacturer: chr "audi" "audi" "audi" "audi" ...
## $ model : chr "a4" "a4" "a4" "a4" ...
## $ displ : num 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
## $ year : int 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
## $ cyl : int 4 4 4 4 6 6 6 4 4 4 ...
## $ trans : chr "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
## $ drv : chr "f" "f" "f" "f" ...
## $ cty : int 18 21 20 21 16 18 18 18 16 20 ...
## $ hwy : int 29 29 31 30 26 26 27 26 25 28 ...
## $ fl : chr "p" "p" "p" "p" ...
## $ class : chr "compact" "compact" "compact" "compact" ...
```

▶ `names()` function to obtain names of the variables in the dataset

```
names(mpg)
```

```
## [1] "manufacturer" "model" "displ" "year" "cyl"  
## [6] "trans" "drv" "cty" "hwy" "fl"  
## [11] "class"
```

▶ To access one of the variables **as a vector**, we use the `$` operator.

```
mpg$year
```

```
## [1] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 2008 1999 1999  
2008 2008  
## [16] 1999 2008 2008 2008 2008 2008 1999 2008 1999 1999 2008 2008 2008  
2008 2008  
## [31] 1999 1999 1999 2008 1999 2008 2008 1999 1999 1999 1999 2008 2008  
2008 1999  
## [46] 1999 2008 2008 2008 2008 1999 1999 2008 2008 2008 1999 1999 1999  
2008 2008  
## [61] 2008 1999 2008 1999 2008 2008 2008 2008 2008 2008 1999 1999 2008  
1999 1999  
## [76] 1999 2008 1999 1999 1999 2008 2008 1999 1999 1999 1999 1999 2008  
1999 2008
```

- ▶ We can use the `dim()`, `nrow()` and `ncol()` functions to obtain information about the dimension of the data frame.

```
dim(mpg)
```

```
## [1] 234  11
```

```
nrow(mpg)
```

```
## [1] 234
```

```
ncol(mpg)
```

```
## [1] 11
```

Subsetting data

- ▶ Subsetting data frames can work much like subsetting matrices using square brackets, `[,]`.
- ▶ Here, we find fuel efficient vehicles earning over 35 miles per gallon and only display manufacturer, model and year.

```
mpg[mpg$hwyl > 35, c("manufacturer", "model", "year")]
```

```
## # A tibble: 6 x 3
##   manufacturer model      year
##   <chr>         <chr>   <int>
## 1 honda         civic    2008
## 2 honda         civic    2008
## 3 toyota        corolla  2008
## 4 volkswagen    jetta    1999
## 5 volkswagen    new beetle 1999
## 6 volkswagen    new beetle 1999
```

- ▶ An alternative would be to use the `subset()` function, which has a much more readable syntax.

```
subset(mpg, subset = hwy > 35, select = c("manufacturer", "model", "year"))
```


- ▶ Lastly, we could use the `filter` and `select` functions from the `dplyr` package which introduces the `%>%` operator from the `magrittr` package.

```
library(dplyr)
mpg %>%
  filter(hwy > 35) %>%
  select(manufacturer, model, year)
```

- ▶ I will give you an assignment about `dplyr` package in the DataCamp as a makeup lecture.