

Data Wrangling Part2: dplyr

Installing Packages

- To use an R package, you must first **install** it on your computer
- **install.packages()**
- Package names should be in quotes
- Search CRAN server, download, and install

```
install.packages("<package name>")
```

- Multiple packages? Vectorize **c()**.

```
install.packages(c("ggplot2", "data.table", "dplyr"))
```

Loading Packages

- To actually use the packages, we have to **load** them first
- `library()`
- Every new R session -> Fresh -> Have to load packages again
- Goes at the top
- Quotation marks are optional for `library()`, but personally I use them every time
- Load everything in the package
 - ▶ datasets, functions, objects, help documents

```
library(<package name>)
library("<package name>")
```

Loading Packages

- Loading everything may not be a good news:
 - ▶ Takes up computing resources (RAM)
 - ▶ Some R objects have same names -> mask the previously loaded ones
- You can manually use R object inside a package by
`package_name::object_name`
- Usually, do not have to worry about this

```
# Use lag function from dplyr
lag_gdp <- dplyr::lag(df$gdp)
```

```
# Use lag function from stats
lag_gdp <- stats::lag(df$gdp)
```

Help Documents

- `help()` or `?`

```
help(lm)
help("lm") # quotation marks are optional
?lm
?"lm"
?sample
```

```
# Specific function inside a specific package
help(rlm, package = "MASS")
# For packages
help(package="MASS")
```

Help Documents

- Description: sometimes helpful
- Arguments
- Values: these are outputs
- **Examples:** very useful

In-class exercises:

- ① Install the packages **dplyr** and **ggplot2**.
- ② Load the packages into your current R Session.

Data Wrangling Part 2: dplyr

- We have learned how to play or wrangle with R data in base R way:
 - ▶ `object[row, col]`
- `dplyr` package offers some convenient alternatives

dplyr Verbs

① Rows:

- `filter()` chooses rows based on column values.
- `slice()` chooses rows based on location.
- `arrange()` changes the order of the rows.

② 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.

3 Groups of rows:

- `summarise()` collapses a group into a single row.

The Pipe

- The single most important feature of dplyr
- %>% or |>
- x %>% function(y): Passes A as **the first argument** of the function()
- Same as function(x, y)
- You can *chain* functions

```
data <- original %>%  
  filter() %>% # pass filtered data to select()  
  select() %>% # pass selected data to group_by()  
  group_by() %>% # pass grouped data to summarize()  
  summarize()
```

Filter rows with filter()

- `filter(data, index / conditions)` subset rows
- For example, we can select all rows with STATE as Missouri:

```
library("dplyr")
worldTFR <- read.csv('~/worldTFR.csv')
```

```
filter(worldTFR, Year == "1950")
```

```
worldTFR %>%
  filter(Year == "1950")
```

- This is roughly equivalent to this base R code:

```
worldTFR[worldTFR$Year == "1950", ] # Don't forget the comma!
```

Filter rows with filter()

- More conditions

```
worldTFR %>%
  filter(
    Year == "1950",
    TFR >= 7.5,
    Uncode > 500
  )
```

- Filter NAs

```
worldTFR %>%
  filter(!is.na(GDPpc))

# With other conditions here:
worldTFR %>%
  filter(!is.na(GDPpc), TFR > 5)
```

Arrange rows with arrange()

- `arrange()`: reorders rows

```
worldTFR.na <- worldTFR %>%  
  filter(complete.cases(.)) # na.omit()  
worldTFR.na %>%  
  arrange(Year, TFR)
```

- Use `desc()` to order a column in descending order:

```
# desc in Year, but still ascending in TFR  
worldTFR.na %>%  
  arrange(desc(Year), TFR) %>%  
  head(2)
```

Choose rows using their position with slice()

- `slice()`: index rows by their (integer) locations

```
worldTFR.na %>%  
  slice(5:10)
```

```
worldTFR[5:10, ]
```

Other slice-likes()

- `slice_head()` and `slice_tail()` select the first or last rows
- `slice_sample(n, prop, replace)`
- `slice_min()` and `slice_max()` select rows with highest or lowest values of a variable
 - ▶ Watch out for missing values (NAs)

Select columns with `select()` by Names

```
worldTFR %>% select(CountryCode, Year, TFR)
```

- WARNING: MASS package also has `select()` function
- Load `dplyr` later than MASS
- Or use `dplyr::select()` and `MASS::select()`

Select all columns between two columns

- Select all columns between Year and LifeExpB

```
worldTFR.na %>%  
  select(Year:LifeExpB)
```

Select all columns except some

Select all columns except those from NAME to STATE

```
worldTFR.na %>%  
  select(!(NAME:STATE))
```

Select columns with an expression

```
worldTFR.na %>% select(starts_with("Country"))
worldTFR.na %>% select(ends_with("e"))
worldTFR.na %>% select(contains("Country"))
```

Add new columns with mutate()

- `mutate()`

```
worldTFR.na %>%
  mutate(Childbearing_r = round(Childbearing, 0))

worldTFR$Childbearing_r = round(worldTFR$Childbearing, 0)
```

Group operation

- You can group the data and operate based on the group.

```
# Create new variable based on Pop1564Female
worldTFR <- worldTFR %>%
  mutate(female50 = ifelse(Pop1564Female > 50, 1, 0))

# Now calculate group mean of LifeExpB based on the female 50
worldTFR %>%
  group_by(female50) %>%
  summarize(mean(LifeExpB))
```

In-class exercises:

Let's apply the tools we learned with the **worldTFR** dataset!

- ① Select all rows where Year is 1951 and TFR is greater than 6.
- ② Select all columns except CountryCode.
- ③ Create a new column called **ChildBearing_sd** where you subtract the mean of the column from its original value and divide the result by the standard deviation of the column.

Intro to R Markdown

- Markdown + R output
- Problem sets -> probably in this format
- Easy to convert into different output formats:
 - ▶ HTML, PDF, LaTex, Beamer, etc.
- knitr package is needed
- A good guide: <https://bookdown.org/yihui/rmarkdown/>

Compile

- Hit the “knit” button in Rstudio
- `rmarkdown::render(file.rmd)`

The YAML Header

- The beginning of the R markdown
- Includes metadata
- Surrounded by three dashes
- All are optional

```
title: "TITLE"
author: "AUTHOR"
date: "DATE"
output:
  pdf_document / html_document / beamer_presentation:
    output_options such as toc
---
```

Markdown part

- knitr uses pandoc in as its backend
- pandoc markdown style will mostly be respected
- Headings

```
# Header 1  
## Header 2  
### Header 3  
#### Header 4
```

Markdown part

- italic with *asterisks*
- bold with **double asterisks**
- List

1. item 1
 1. Second level
 - Or this
2. item 2
3. item 3

- List 2

- * item 1
- * item 2
- * item 3

Markdown part

- LaTex style math
 - ▶ `$inline_math$`
 - ▶ `$$display_math$$`

R code chunks

- Similar to plain markdown codechunk, but the outputs will be executed
- Three backticks followed by {r}
- Without {r} it will be treated as normal code chunk and will not be executed

```
```{r}
your code
```
```

Code chunk options

- You can set options for code chunks
 - ▶ name, execute or not, display or not etc
- For global setting,

```
```{r}
knitr::opts_chunk$set(options)
```
```

- For a code chunk

```
```{r codechuck_name, option1 = TRUE, option2 = FALSE}
your code
```
```

Code chunk options

| option | default | effect |
|---------|---------|---|
| eval | TRUE | Whether to evaluate |
| echo | TRUE | Whether to display code |
| include | TRUE | Whether to include the code and its results |
| results | TRUE | How to show results |
| warning | TRUE | Whether to display warnings |
| error | FALSE | Whether to display errors |
| message | TRUE | Whether to display messages |
| cache | FALSE | Whether to save cache files |

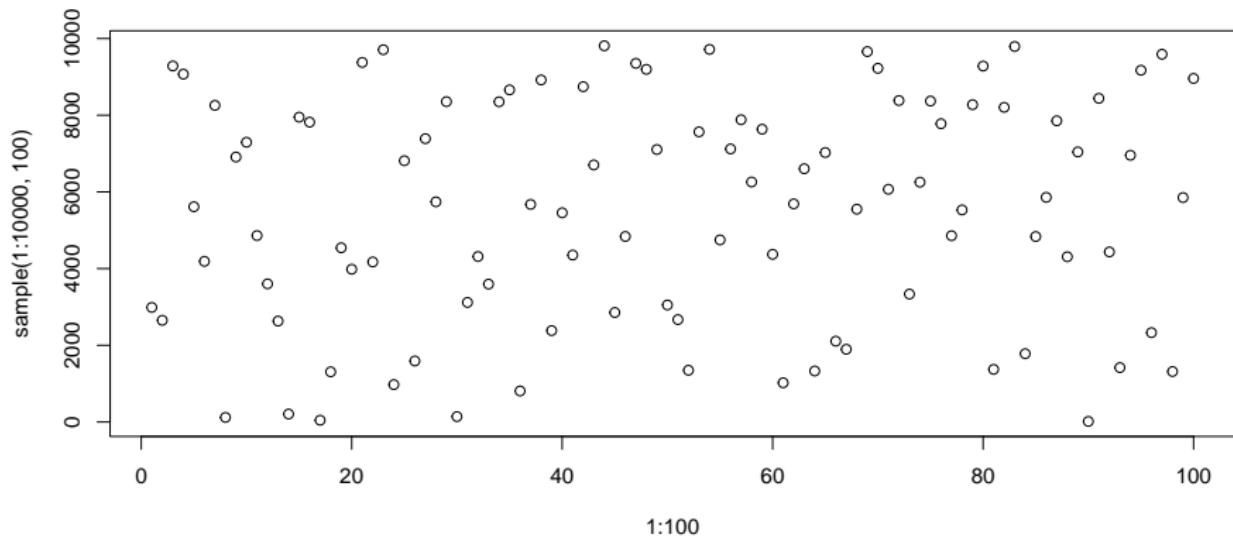
- The full list of options are available here:
<https://yihui.org/knitr/options/>

Adding Images

- When you create an image in your code block in R Markdown

For example,

```
plot(1:100, sample(1:10000, 100))
```



Adding Images

- External image files:

```
! [name of your image] (path-to-image-here)
```

