Data Wrangling

Introduction

This documents introduce you some functions for data cleaning, such as sub-setting, making new variables, and summarizing. We mainly focus on `tidyverse` package in R, which is the most popular package designed for data science.

Download Data and Install Package

We need to download the Data_Wrangling.zip file and unzip it. Then, we need to install the following packages.

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

Read Data From Different Formates

```
library(tidyverse)
library(haven) # for reading stata file
library(readxl) # for reading excel file

# First thing first, set working directory

# setwd()

# Different format

data_txt <- read.delim("anes2016.txt")
```

```r
data_csv <- read_csv("anes2016.csv")

data_excel <- read_excel("anes2016.xlsx")

data_dta <- read_dta("anes2016.dta")

data_rds <- readRDS("anes2016.rds")

load("anes2016.rds") # read R data file
```

Data Cleaning

Pipes!

Pipes let you take the output of one function and send it directly to the next, which is useful when you need to do many things to the same dataset. Pipes in R look like `%>%` or `|>` in the newer version.

Subsetting columns and rows

1. select columns: `select()`
2. select rows: `filter()`

```r
data <- read_csv("anes2016.csv")

glimpse(data)


# Let's select age, education, votein2016

data_sub <- data %>% select(age, education, votein2016)

# Filter non-missing value and age between 45 and 65

data_sub %>% filter(!is.na(votein2016)) %>% filter(age >= 45 & age <= 65)

# Filter specific value

educ <- data %>% filter(education == 5)
```

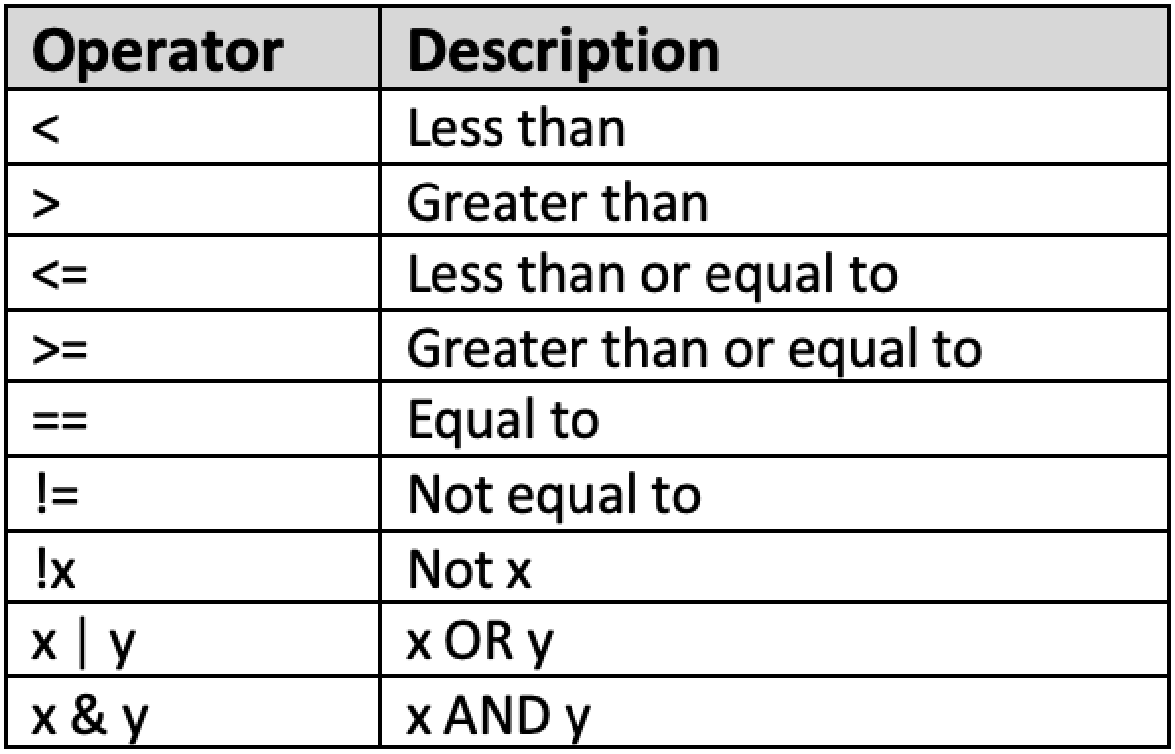
```r
# Remove duplicate row

unqi <- data %>% distinct(education, .keep_all = T)

# Put all lines of code together

data_sub <- data %>% select(age, education, votein2016) %>%
  filter(!is.na(votein2016)) %>% filter(age >= 45 & age <= 65) %>%
  filter(education == 5)


# Advance usage of select

data %>% select(contains("vote"))

data %>% select(starts_with("vote"))

data %>% select(ends_with("feel"))

data %>% select(latino:asian)
```

Logical Operators in R

Sort Data and Handle Missing

1. Sort data: `arrange()`
2. High to low: `desc()`
3. Drop missing: `drop_na`
4. Fill missing: `fill()`
5. Replace missing: `replace_na()`

```r
# Sort data

data %>% arrange(attentiontopolitics)

data %>% arrange(desc(attentiontopolitics))

# Drop or replace missing
```



Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
!x	Not x
x | y	x OR y
x & y	x AND y

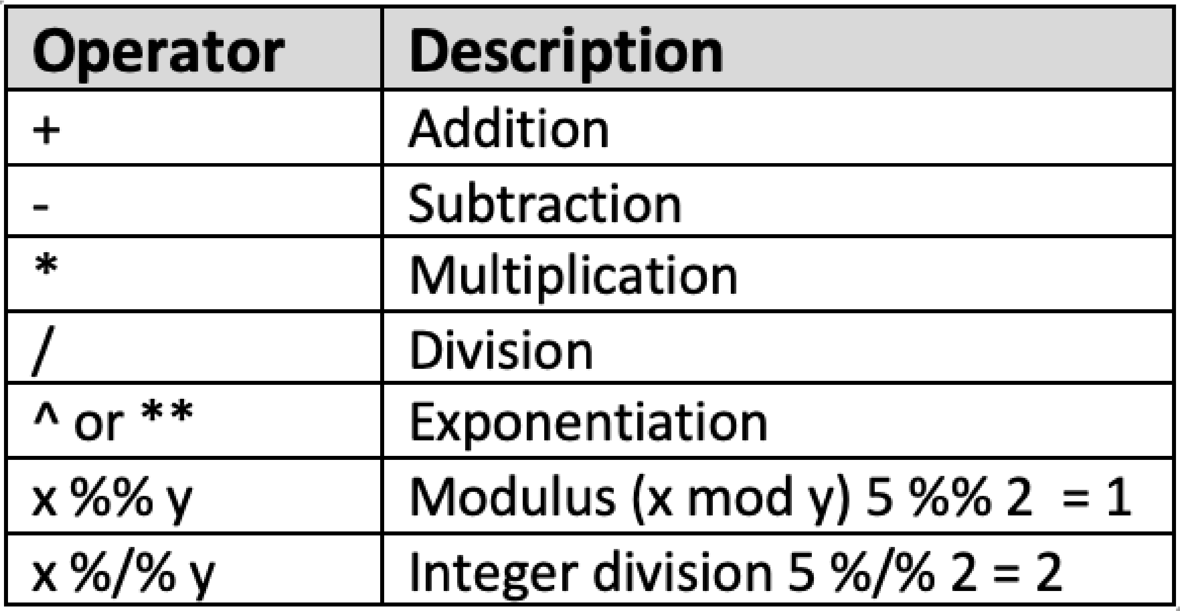
Figure 1: Logical Operators in R

```r
data_no_missing <- data %>% drop_na(votetrump)

data_replace_missing <- data %>% replace_na(list( votetrump = 0) )
```

Make New Variables

1. Create new variable: `mutate()`
2. Rename variable: `rename()`
3. Create new variable based on condition: `if_else()`
4. Recode data: `recode()`

```r
# Crete a new variable

data_sub <- data %>% mutate(vote_all = votein2012 + votein2016) %>%
  mutate(vote_all = if_else(vote_all > 1, 1, 0)) %>%
  rename(vote_both = vote_all) %>%
  mutate(latino = recode(latino, `1` = "Yes", `0` = "No"))
```



Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation
x %% y	Modulus (x mod y) 5 %% 2 = 1
x %/% y	Integer division 5 %/% 2 = 2

Figure 2: Arithmetic Operators in R

Group and Summarise Data

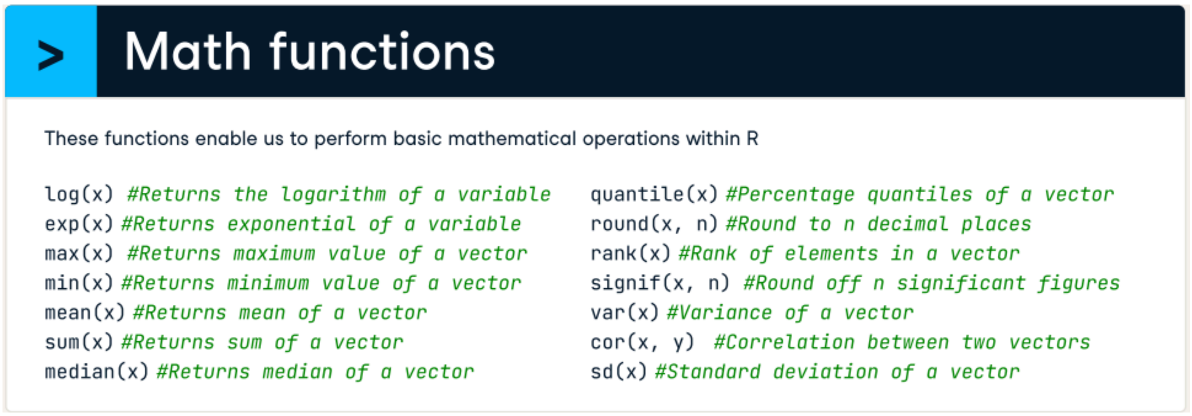
1. Summarize data: `summarise()`
2. Group data: `group_by()`

```
# Quick summaries of your data

sum_stat <- data_sub %>% filter(latino == "Yes") %>%
  drop_na() %>%
  summarise(ave_age = mean(age),
            max_age = max(age),
            min_age = min(age),
            sd_age = sd(age),
         ave_edu = mean(education),
         votetrump_number = n(),
         votetrump_percent = mean(votetrump)*100,
         )

# What happens if we do not use drop_na?

# Summarize by group

data_sub %>% group_by(latino) %>%
  drop_na() %>%
  summarise(ave_age = mean(age),
            max_age = max(age),
            min_age = min(age),
            sd_age = sd(age),
         ave_edu = mean(education),
         votetrump_number = n(),
         votetrump_percent = mean(votetrump)*100,
         )
```

Combine Data

1. Row bind and column bind: `bind_rows()` and `bind_cols()`
2. Join two data set: 1) `left_join()`, 2) `right_join()`, 3) `inner_join()`, 4) `full_join()`

```
# Let's split data by row first
```



Math functions

These functions enable us to perform basic mathematical operations within R

```
log(x) #Returns the logarithm of a variable
exp(x) #Returns exponential of a variable
max(x) #Returns maximum value of a vector
min(x) #Returns minimum value of a vector
mean(x) #Returns mean of a vector
sum(x) #Returns sum of a vector
median(x) #Returns median of a vector
quantile(x) #Percentage quantiles of a vector
round(x, n) #Round to n decimal places
rank(x) #Rank of elements in a vector
signif(x, n) #Round off n significant figures
var(x) #Variance of a vector
cor(x, y)  #Correlation between two vectors
sd(x) #Standard deviation of a vector
```

Figure 3: Math Funtions in R

```
data1 <- data %>% filter(unique_id <= 1000)

data2 <- data %>% filter(unique_id >= 1000)

# If we want to bind them together

data_bind <- bind_rows(data1, data2)

data_bind <- data2 %>% bind_rows(data1) %>% arrange(unique_id) # why we have one more


# Let's split data by column


data1 <- data %>% select(age, education, votein2016)
data2 <- data %>% select(-education, -votein2016)

data_bind <- bind_cols(data1, data2) # what happens?
```

Join two data

```
# Let's read two data set first

data_main <- read_csv("data_main.csv")
```

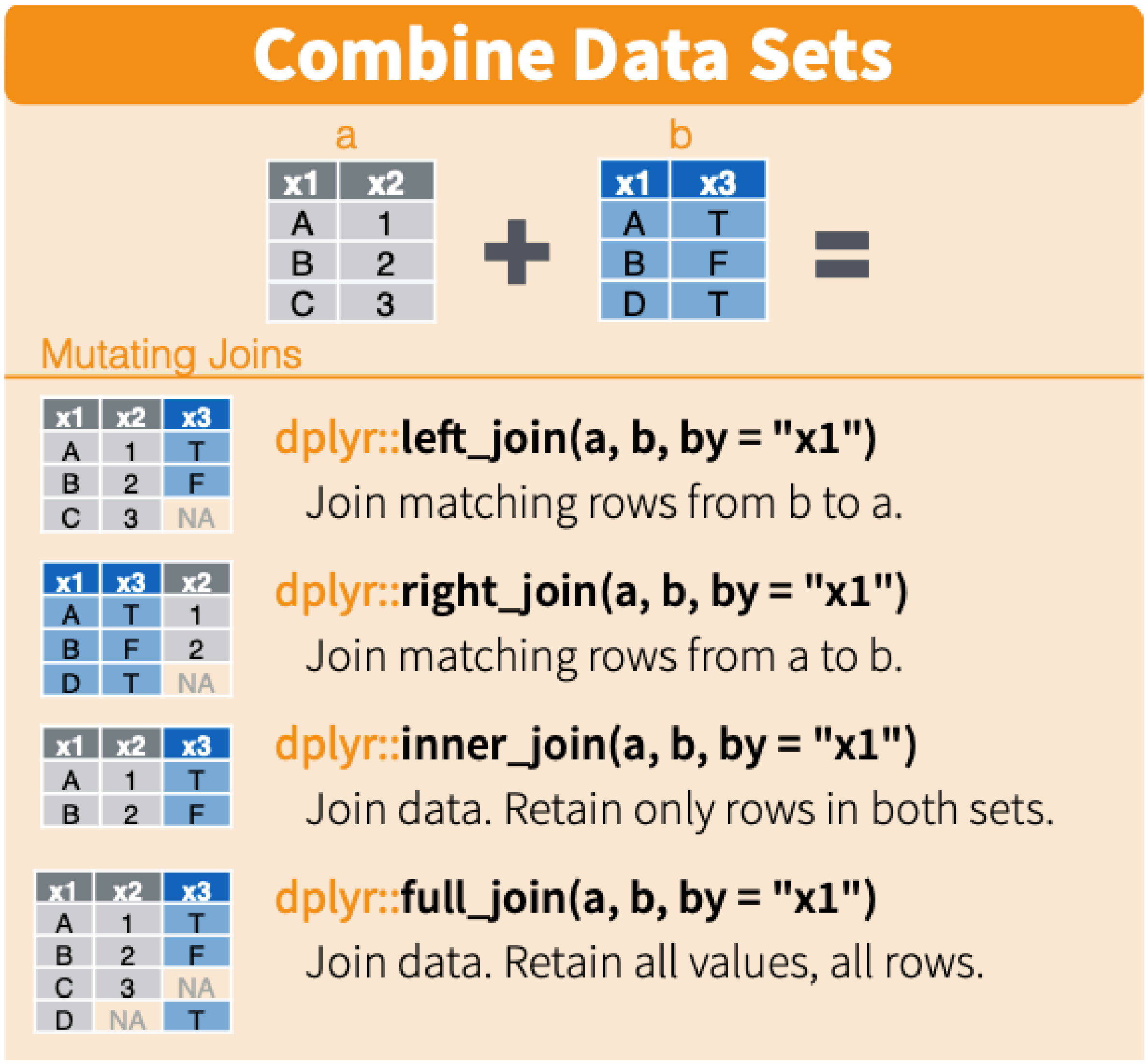
```r
data_merge <- read_csv("data_merge.csv")

# We can combine by unique id

## observations that are not in data_merge will be missing
data_com <- left_join(data_main, data_merge, by = "unique_id" )

## observations that are not in data_main will be missing
data_com <- right_join(data_main, data_merge, by = "unique_id" )

## observations that are only in both
data_com <- inner_join(data_main, data_merge, by = "unique_id" )

## observations that are in both
data_com <- full_join(data_main, data_merge, by = "unique_id" )


# What else unique ids in the real word?
```

Distribution Functions

Let's take normal functions as examples.

1. `dnorm()`: returns the value of the probability density function (pdf).
2. `pnorm()`: returns the value of the cumulative density function (cdf). For statistical testingm, after you have critical value, you could use CDF function to calculate probability.
3. `qnorm()`: returns the value of the inverse cumulative density function.
4. `rnorm()`: generates a vector of normally distributed random variables

```r
# Density function

dnorm(1.9)

dnorm(1.9, mean = 1, sd = 2)

# CDF function

pnorm(1.5)
```



Combine Data Sets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

```
dplyr::left_join(a, b, by = "x1")
```

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

```
dplyr::right_join(a, b, by = "x1")
```

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

```
dplyr::inner_join(a, b, by = "x1")
```

Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

```
dplyr::full_join(a, b, by = "x1")
```

Join data. Retain all values, all rows.

Figure 4: Merge two dataset in R

```
pnorm(1.5, mean = 0, sd = 1, lower.tail = F)

# Quantile Function

qnorm(0.95)

qnorm(0.95, mean = 0, sd = 1, lower.tail = F)

# Random generator

rnorm(100)

rnorm(100, mean = 1, sd = 2)
```

Questions

1. Read the `In_class.csv` into R

Answer:

```
data_in <- read_csv("In_class.csv")
```

2. Please do the following steps in one pipe. First, filter observations which education levels are "BAdeg" and "CCdeg". Second, rename "income.num" to "income_num". Third, create a new variable called "trump_distance" by calculating the difference between selfLR and TrumpLR. Finally, select columns: age, education, income_num, PreVote, and trump_distance.

Answer:

```
data_sub <- data_in %>% filter(education == "BAdeg" | education == "CCdeg") %>%
  rename(income_num = income.num ) %>%
  mutate(trump_distance =  selfLR - TrumpLR) %>%
  select(age, education, income_num, PreVote, trump_distance)
```

3. Based on the data from Q2, can you calculate how many people intend to vote for Trump? What is percentage of people intend to vote for Trump? Can you calculate the mean and standard deviation of age, income_num, and trump_distance by whether they vote for Trump or Clinton?

Answer:

```
data_sub %>% mutate(vote = if_else(PreVote == "DonaldTrump", 1, 0)) %>%
  summarise(vote_num = sum(vote),
            vote_percent = mean(vote))

data_sub %>% group_by(PreVote) %>%
  summarise(ave_age = mean(age),
            sd_age = sd(age),
            ave_income = mean(income_num),
            sd_income = sd(income_num),
            ave_trump = mean(trump_distance))
```

3.1 Advanced Question: can you conduct a two sample T-test to determine if there was a statistically difference in income between Trump voters and Clinton voters.

Answer:

```
data_in %>% rename(income_num = income.num ) %>%
  group_by(PreVote) %>%
  summarise(ave_income = mean(income_num),
            sd_income = sd(income_num),
            vote_num = n())

x1 <- 79.63

x2 <- 84.05

n1 <- 1065

n2 <- 1123

s1 <- 58.81

s2 <- 66.55

sp <- sqrt( ((n1 - 1)*s1^2 + (n2 - 1)*s2^2)/(n1 + n2 - 2) )

t <- (x1 - x2)/(sp*(sqrt(1/n1 + 1/n2)))

p_value <- 2*pt(t, df = n1 + n2 - 2, lower.tail=T)
```

4. What happens if we want to row bind two data set with different columns

Answer:

```
# Let's split data by row first but in the second dataset, we omit age column

data1 <- data_in %>% filter(unique_id <= 1000)

data2 <- data_in %>% filter(unique_id >= 1000) %>% select(-age)

# If we want to bind them together

data_bind <- bind_rows(data1, data2)
```

5. What happens if we want to column bind two data set with different rows

Answer:

```
# Let's split data by columns first but in the second dataset,
# we only keep 2000 rows

data1 <- data_in %>% select(age, education)
data2 <- data_in %>% select(-age, -education) %>% filter(unique_id <= 2000)

data_bind <- bind_cols(data1, data2)
```