Data Wrangling with dplyr and tidyr

Aboubacar HEMA

June 26, 2020

# Data Import with readr, tibble, and tidyr

R’s **tidyverse** is built around **tidy data** stored in **tibbles**, an enhanced version of a data frame.

## Read functions

### Read tabular data to tibbles

These functions share the common arguments: **read\_**

#### read\_csv()

Reads comma delimited files.

# Tidy Data

Tidy Data is a foundation for wrangling in R. In a tidy data set, each variable is saved in its own column and each observation is saved in its own row.Tidy data complements R’s vectorized operations. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

## Syntax - Helpful conventions for wrangling

### tbl\_df()

Converts data to tbl class. tbl’s are easier to examine than data frames. R displays only the data that fits on screen.

dplyr::tbl\_df(iris)

## # A tibble: 150 x 5  
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## <dbl> <dbl> <dbl> <dbl> <fct>   
## 1 5.1 3.5 1.4 0.2 setosa   
## 2 4.9 3 1.4 0.2 setosa   
## 3 4.7 3.2 1.3 0.2 setosa   
## 4 4.6 3.1 1.5 0.2 setosa   
## 5 5 3.6 1.4 0.2 setosa   
## 6 5.4 3.9 1.7 0.4 setosa   
## 7 4.6 3.4 1.4 0.3 setosa   
## 8 5 3.4 1.5 0.2 setosa   
## 9 4.4 2.9 1.4 0.2 setosa   
## 10 4.9 3.1 1.5 0.1 setosa   
## # ... with 140 more rows

### glimpse()

Information dense summary of tbl data.

dplyr::glimpse(iris)

## Rows: 150  
## Columns: 5  
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4,...  
## $ Sepal.Width <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7,...  
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5,...  
## $ Petal.Width <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2,...  
## $ Species <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa...

### View()

View data set in spreadsheet-like display (note capital V).

#utils::View(iris)

### %>%

Passes object on lef hand side as first argument (or . argument) of function on righthand side. x %>% f(y) is the same as f(x, y) and y %>% f(x, ., z) is the same as f(x, y, z ). “Piping” with %>% makes code more readable, e.g:

library(dplyr)  
iris %>%  
 group\_by(Species) %>%  
 summarise(avg = mean(Sepal.Width)) %>%  
 arrange(avg)

## # A tibble: 3 x 2  
## Species avg  
## <fct> <dbl>  
## 1 versicolor 2.77  
## 2 virginica 2.97  
## 3 setosa 3.43

## Reshaping Data - Change the layout of a data set

### gather()

Gather columns into rows.

library(tidyr)  
mini\_iris <-  
 iris %>%  
 group\_by(Species) %>%  
 slice(1)  
mini\_gather<-mini\_iris %>% gather(key = "flower\_att", value = "measurement", -Species)  
#utils::View(mini\_iris)

### spread()

Spread rows into columns

mini\_gather %>% spread(flower\_att,measurement)

## # A tibble: 3 x 5  
## # Groups: Species [3]  
## Species Petal.Length Petal.Width Sepal.Length Sepal.Width  
## <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 setosa 1.4 0.2 5.1 3.5  
## 2 versicolor 4.7 1.4 7 3.2  
## 3 virginica 6 2.5 6.3 3.3

### separate()

Separate one column into several.

library(dplyr)  
df <- data.frame(x = c(NA, "a.b", "a.d", "b.c"))  
df

## x  
## 1 <NA>  
## 2 a.b  
## 3 a.d  
## 4 b.c

#utils::View(df)  
df %>% separate(x, c("A", "B"))

## A B  
## 1 <NA> <NA>  
## 2 a b  
## 3 a d  
## 4 b c

If you just want the second variable:

df %>% separate(x, c(NA, "B"))

## B  
## 1 <NA>  
## 2 b  
## 3 d  
## 4 c

### unite()

Unite several columns into one.

df<-df %>% separate(x, c("A", "B"))  
df %>% unite("x", A:B, remove = FALSE)

## x A B  
## 1 NA\_NA <NA> <NA>  
## 2 a\_b a b  
## 3 a\_d a d  
## 4 b\_c b c

To remove missing values:

df %>% unite("x", A:B, na.rm = TRUE, remove = FALSE)

## x A B  
## 1 <NA> <NA>  
## 2 a\_b a b  
## 3 a\_d a d  
## 4 b\_c b c

## Subset Observations (Rows)

### filter()

Extract rows that meet logical criteria.

### distinct()

Remove duplicate rows.

### sample\_frac()

Randomly select fraction of rows.

### sample\_n()

Randomly select n rows.

### slice()

Select rows by position.

### top\_n()

Select and order top n entries (by group if grouped data).

## Subset Variables (Columns)

### select()

Select columns by name or helper function.Helper functions for select.

#### select(iris,contains(“.”))

Select columns whose name contains a character string.

#### select(iris, ends\_with(“Length”))

Select columns whose name ends with a character string.

#### select(iris, everything())

Select every column.

#### select(iris, matches(“.t.”))

Select columns whose name matches a regular expression.

#### select(iris, num\_range(“x”, 1:5))

Select columns named x1, x2, x3, x4, x5.

#### select(iris, one\_of(c(“Species”, “Genus”)))

Select columns whose names are in a group of names.

#### select(iris, starts\_with(“Sepal”))

Select columns whose name starts with a character string.

#### select(iris, Sepal.Length:Petal.Width)

Select all columns between Sepal.Length and Petal.Width (inclusive).

#### select(iris, -Species)

Select all columns except Species.

## Summarise Data

### summarise()

Summarise data into single row of values.

### summarise\_each(iris,funs(mean))

Apply summary function to each column.

### count()

Count number of rows with each unique value of variable (with or without weights).

## Make New Variables

### mutate()

Compute and append one or more new columns.

### mutate\_each(iris,funs(min\_rank))

Apply window function to each column.

### transmute()

Compute one or more new columns. Drop original columns.

## Combine Data Sets

### Mutating Joins

#### left\_join()

Join matching rows from b to a.

#### right\_join()

Join matching rows from a to b.

#### inner\_join()

Join data. Retain only rows in both sets.

#### full\_join()

Join data. Retain all values, all rows.

### Filtering Joins

#### semi\_join()

All rows in **a** that have a match in **b**.

#### anti\_join()

All rows in **a** that do not have a match in **b**.

### Set Operations

#### intersect()

Rows that appear in both **y** and **z**.

#### union()

Rows that appear in either or both **y** and **z**.

#### setdiff()

Rows that appear in **y** but not **z**.

### Binding

#### bind\_rows()

Append **z** to **y** as new rows.

#### bind\_cols()

Append z to y as new columns. Caution: matches rows by position.

## Group Data

### group\_by(iris,Species)

Group data into rows with the same value of Species.

### ungroup(iris)

Remove grouping information from data frame.

### iris %>% group\_by(Species) %>% summarise(.)

Compute separate summary row for each group.

### iris %>% group\_by(Species) %>% mutate(.)

Compute new variables by group.