Introduction to dplyr for Faster Data Manipulation in R

Masumbuko Semba

On 19 October, 2018 at 08:26:26.654

### Setting Working Directory

A working directory is a folder in a computer or server that stores the data and other asset. This folder is important in programming because it allows to read and write files to working directory. In R you can set working directory with setwd() function and check whether you are in the right working directory with the getwd() function.

getwd()

## [1] "E:/Data Manipulation/R\_dege"

setwd("E:/Data Manipulation/R\_dege/")  
# getwd()

## Introduction to dplyr package

The dplyr package provides a set of functions for efficiently manipulating datasets in R writen by @dplyr. The package make it easy to transform and summarise tabular data with rows and columns. The dplyr packages contains set of functions—verbs that perfom most common data manipulation tasks like

### Why use dplyr?

1. Great for data exploration and manipulation
2. Intuitive to write and easy to read, especially when using the chaining syntax
3. Fast on data frame—tabular dataset

### dplyr functionality

* Five basic verbs:
* select() to select columns based on their names
* filter() to rows in data frame
* arrange() to re-order or arrange the rows in ascending or descending order
* mutate() to create new columns—add new variable
* summarise() to make a summary of variable(s)
* group\_by() to group observation
* sample\_n() and rename()to make random sample from the data set

The group\_by() function perform other common task which are related to the *split-apply-combine* concept. The dplyr package comes with the pipe operateor %>% from the magrittr package. The pipe operator is very useful for combining several functions in a chain.

## Loading packages

Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into the session to be used. We load the dplyr package and nycflights13 package.

require(dplyr)

## Loading required package: dplyr

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

require(nycflights13)

## Loading required package: nycflights13

flights

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 336,766 more rows, and 12 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

The nycflights13 contains information of the flights departed from New York City in 2013 [@nyc]. Also includes useful metadata on airlines, airports, weather, and planes. Thenycflights13` packages comes with five tabular datasets, which are:

* airlines Airline names.
* airports Airport metadata
* flights Flights data
* planes Plane metadata.
* weather Hourly weather data

# assign the flights dataset  
flights = flights

### explore the structure of the flights dataset

dim(flights)

## [1] 336776 19

nrow(flights)

## [1] 336776

ncol(flights)

## [1] 19

length(flights)

## [1] 19

# head(flights)  
# tail(flights)  
flights

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 336,766 more rows, and 12 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

glimpse(flights)

## Observations: 336,776  
## Variables: 19  
## $ year <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013,...  
## $ month <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...  
## $ day <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...  
## $ dep\_time <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 55...  
## $ sched\_dep\_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 60...  
## $ dep\_delay <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2...  
## $ arr\_time <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 7...  
## $ sched\_arr\_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 7...  
## $ arr\_delay <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -...  
## $ carrier <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV",...  
## $ flight <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79...  
## $ tailnum <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN...  
## $ origin <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR"...  
## $ dest <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL"...  
## $ air\_time <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138...  
## $ distance <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 94...  
## $ hour <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5,...  
## $ minute <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, ...  
## $ time\_hour <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013...

## filter: Keep rows matching criteria

* Base R approach to filtering forces you to repeat the data frame’s name
* dplyr approach is simpler to write and read
* Command structure (for all dplyr verbs):
  + first argument is a data frame
  + return value is a data frame
  + nothing is modified in place
* Note: dplyr generally does not preserve row names

# base R approach to view all flights on January 1  
flights[flights$month==1 & flights$day==1, ]

# dplyr approach  
# note: you can use comma or ampersand to represent AND condition  
filter(flights, month==1, day==1)

## # A tibble: 842 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 832 more rows, and 12 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

# use pipe for OR condition

## select: Pick columns by name

* Base R approach is awkward to type and to read
* dplyr approach uses similar syntax to filter
* Like a SELECT in SQL

# base R approach to select DepTime, and ArrTime  
flights[, c("dep\_time", "arr\_time")]

# dplyr approach  
select(flights, dep\_time, arr\_time)

## # A tibble: 336,776 x 2  
## dep\_time arr\_time  
## <int> <int>  
## 1 517 830  
## 2 533 850  
## 3 542 923  
## 4 544 1004  
## 5 554 812  
## 6 554 740  
## 7 555 913  
## 8 557 709  
## 9 557 838  
## 10 558 753  
## # ... with 336,766 more rows

# use colon to select multiple contiguous columns, and use `contains` to match columns by name  
# note: `starts\_with`, `ends\_with`, and `matches` (for regular expressions) can also be used to match columns by name  
select(flights, year:day, contains("taxi"), contains("delay"))

## # A tibble: 336,776 x 5  
## year month day dep\_delay arr\_delay  
## <int> <int> <int> <dbl> <dbl>  
## 1 2013 1 1 2 11  
## 2 2013 1 1 4 20  
## 3 2013 1 1 2 33  
## 4 2013 1 1 -1 -18  
## 5 2013 1 1 -6 -25  
## 6 2013 1 1 -4 12  
## 7 2013 1 1 -5 19  
## 8 2013 1 1 -3 -14  
## 9 2013 1 1 -3 -8  
## 10 2013 1 1 -2 8  
## # ... with 336,766 more rows

## “Chaining” or “Pipelining”

* Usual way to perform multiple operations in one line is by nesting
* Can write commands in a natural order by using the %>% infix operator (which can be pronounced as “then”)

# nesting method to select UniqueCarrier and DepDelay columns and filter for delays over 60 minutes  
filter(select(flights, carrier, dep\_delay), dep\_delay > 60)

# chaining method  
flights %>%  
 select(carrier, dep\_delay) %>%  
 filter(dep\_delay > 60)

## # A tibble: 26,581 x 2  
## carrier dep\_delay  
## <chr> <dbl>  
## 1 MQ 101  
## 2 AA 71  
## 3 MQ 853  
## 4 UA 144  
## 5 UA 134  
## 6 EV 96  
## 7 MQ 71  
## 8 B6 77  
## 9 EV 70  
## 10 EV 115  
## # ... with 26,571 more rows

* Chaining increases readability significantly when there are many commands
* Operator is automatically imported from the [magrittr](https://github.com/smbache/magrittr) package
* Can be used to replace nesting in R commands outside of dplyr. For example, we can create two vectors and calculate Euclidian distance between them using the mathematical equation (1)

$$

$$

x1 <- 1:5; x2 <- 2:6  
sqrt(sum((x1-x2)^2))

# chaining method  
(x1-x2)^2 %>% sum() %>% sqrt()

## [1] 2.236068

# Pipping with

## Choosing columns: select, rename

# besides just using select() to pick columns...  
flights %>%   
 select(carrier, flight)

## # A tibble: 336,776 x 2  
## carrier flight  
## <chr> <int>  
## 1 UA 1545  
## 2 UA 1714  
## 3 AA 1141  
## 4 B6 725  
## 5 DL 461  
## 6 UA 1696  
## 7 B6 507  
## 8 EV 5708  
## 9 B6 79  
## 10 AA 301  
## # ... with 336,766 more rows

# ...you can use the minus sign to hide columns  
flights %>%   
 select(-month, -day)

## # A tibble: 336,776 x 17  
## year dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <dbl> <int> <int>  
## 1 2013 517 515 2 830 819  
## 2 2013 533 529 4 850 830  
## 3 2013 542 540 2 923 850  
## 4 2013 544 545 -1 1004 1022  
## 5 2013 554 600 -6 812 837  
## 6 2013 554 558 -4 740 728  
## 7 2013 555 600 -5 913 854  
## 8 2013 557 600 -3 709 723  
## 9 2013 557 600 -3 838 846  
## 10 2013 558 600 -2 753 745  
## # ... with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,  
## # time\_hour <dttm>

# hide a range of columns  
flights %>%   
 select(-(dep\_time:arr\_delay))

# hide any column with a matching name  
flights %>%   
 select(-contains("time"))

## # A tibble: 336,776 x 13  
## year month day dep\_delay arr\_delay carrier flight tailnum origin  
## <int> <int> <int> <dbl> <dbl> <chr> <int> <chr> <chr>   
## 1 2013 1 1 2 11 UA 1545 N14228 EWR   
## 2 2013 1 1 4 20 UA 1714 N24211 LGA   
## 3 2013 1 1 2 33 AA 1141 N619AA JFK   
## 4 2013 1 1 -1 -18 B6 725 N804JB JFK   
## 5 2013 1 1 -6 -25 DL 461 N668DN LGA   
## 6 2013 1 1 -4 12 UA 1696 N39463 EWR   
## 7 2013 1 1 -5 19 B6 507 N516JB EWR   
## 8 2013 1 1 -3 -14 EV 5708 N829AS LGA   
## 9 2013 1 1 -3 -8 B6 79 N593JB JFK   
## 10 2013 1 1 -2 8 AA 301 N3ALAA LGA   
## # ... with 336,766 more rows, and 4 more variables: dest <chr>,  
## # distance <dbl>, hour <dbl>, minute <dbl>

# pick columns using a character vector of column names  
cols <- c("carrier", "flight", "tailnum")  
flights %>%   
 select(one\_of(cols))

## # A tibble: 336,776 x 3  
## carrier flight tailnum  
## <chr> <int> <chr>   
## 1 UA 1545 N14228   
## 2 UA 1714 N24211   
## 3 AA 1141 N619AA   
## 4 B6 725 N804JB   
## 5 DL 461 N668DN   
## 6 UA 1696 N39463   
## 7 B6 507 N516JB   
## 8 EV 5708 N829AS   
## 9 B6 79 N593JB   
## 10 AA 301 N3ALAA   
## # ... with 336,766 more rows

# select() can be used to rename columns, though all columns not mentioned are dropped  
flights %>%   
 select(tail = tailnum)

## # A tibble: 336,776 x 1  
## tail   
## <chr>   
## 1 N14228  
## 2 N24211  
## 3 N619AA  
## 4 N804JB  
## 5 N668DN  
## 6 N39463  
## 7 N516JB  
## 8 N829AS  
## 9 N593JB  
## 10 N3ALAA  
## # ... with 336,766 more rows

# rename() does the same thing, except all columns not mentioned are kept  
flights %>%   
 rename(tail = tailnum)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 336,766 more rows, and 12 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tail <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

## Choosing rows: filter, between, slice, sample\_n, top\_n, distinct

# filter() supports the use of multiple conditions  
flights %>%   
 filter(dep\_time >= 600, dep\_time <= 605)

## # A tibble: 2,460 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 600 600 0 851  
## 2 2013 1 1 600 600 0 837  
## 3 2013 1 1 601 600 1 844  
## 4 2013 1 1 602 610 -8 812  
## 5 2013 1 1 602 605 -3 821  
## 6 2013 1 2 600 600 0 814  
## 7 2013 1 2 600 605 -5 751  
## 8 2013 1 2 600 600 0 819  
## 9 2013 1 2 600 600 0 846  
## 10 2013 1 2 600 600 0 737  
## # ... with 2,450 more rows, and 12 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

# between() is a concise alternative for determing if numeric values fall in a range  
flights %>%   
 filter(between(dep\_time, 600, 605))

# side note: is.na() can also be useful when filtering  
flights %>%   
 filter(!is.na(dep\_time))

## # A tibble: 328,521 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 328,511 more rows, and 12 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

## Adding new variables: mutate, transmute, add\_rownames

# mutate() creates a new variable (and keeps all existing variables)  
flights %>% mutate(speed = distance/air\_time\*60)

## # A tibble: 336,776 x 20  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 336,766 more rows, and 13 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>, speed <dbl>

# transmute() only keeps the new variables  
flights %>% transmute(speed = distance/air\_time\*60)

## # A tibble: 336,776 x 1  
## speed  
## <dbl>  
## 1 370.  
## 2 374.  
## 3 408.  
## 4 517.  
## 5 394.  
## 6 288.  
## 7 404.  
## 8 259.  
## 9 405.  
## 10 319.  
## # ... with 336,766 more rows

# example data frame with row names  
mtcars %>% head()

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

# add\_rownames() turns row names into an explicit variable  
mtcars %>% add\_rownames("model") %>% head()

## Warning: Deprecated, use tibble::rownames\_to\_column() instead.

## # A tibble: 6 x 12  
## model mpg cyl disp hp drat wt qsec vs am gear carb  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Mazda~ 21 6 160 110 3.9 2.62 16.5 0 1 4 4  
## 2 Mazda~ 21 6 160 110 3.9 2.88 17.0 0 1 4 4  
## 3 Datsu~ 22.8 4 108 93 3.85 2.32 18.6 1 1 4 1  
## 4 Horne~ 21.4 6 258 110 3.08 3.22 19.4 1 0 3 1  
## 5 Horne~ 18.7 8 360 175 3.15 3.44 17.0 0 0 3 2  
## 6 Valia~ 18.1 6 225 105 2.76 3.46 20.2 1 0 3 1

# side note: dplyr no longer prints row names (ever) for local data frames  
mtcars %>% tbl\_df()

## # A tibble: 32 x 11  
## mpg cyl disp hp drat wt qsec vs am gear carb  
## \* <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 21 6 160 110 3.9 2.62 16.5 0 1 4 4  
## 2 21 6 160 110 3.9 2.88 17.0 0 1 4 4  
## 3 22.8 4 108 93 3.85 2.32 18.6 1 1 4 1  
## 4 21.4 6 258 110 3.08 3.22 19.4 1 0 3 1  
## 5 18.7 8 360 175 3.15 3.44 17.0 0 0 3 2  
## 6 18.1 6 225 105 2.76 3.46 20.2 1 0 3 1  
## 7 14.3 8 360 245 3.21 3.57 15.8 0 0 3 4  
## 8 24.4 4 147. 62 3.69 3.19 20 1 0 4 2  
## 9 22.8 4 141. 95 3.92 3.15 22.9 1 0 4 2  
## 10 19.2 6 168. 123 3.92 3.44 18.3 1 0 4 4  
## # ... with 22 more rows

## Grouping and counting: summarise, tally, count, group\_size, n\_groups, ungroup

# summarise() can be used to count the number of rows in each group  
flights %>% group\_by(month) %>% summarise(cnt = n())

## # A tibble: 12 x 2  
## month cnt  
## <int> <int>  
## 1 1 27004  
## 2 2 24951  
## 3 3 28834  
## 4 4 28330  
## 5 5 28796  
## 6 6 28243  
## 7 7 29425  
## 8 8 29327  
## 9 9 27574  
## 10 10 28889  
## 11 11 27268  
## 12 12 28135

# tally() and count() can do this more concisely  
flights %>% group\_by(month) %>% tally()

# you can sort by the count  
flights %>% group\_by(month) %>% summarise(cnt = n()) %>% arrange(desc(cnt))

## # A tibble: 12 x 2  
## month cnt  
## <int> <int>  
## 1 7 29425  
## 2 8 29327  
## 3 10 28889  
## 4 3 28834  
## 5 5 28796  
## 6 4 28330  
## 7 6 28243  
## 8 12 28135  
## 9 9 27574  
## 10 11 27268  
## 11 1 27004  
## 12 2 24951

# you can sum over a specific variable instead of simply counting rows  
flights %>% group\_by(month) %>% summarise(dist = sum(distance))

## # A tibble: 12 x 2  
## month dist  
## <int> <dbl>  
## 1 1 27188805  
## 2 2 24975509  
## 3 3 29179636  
## 4 4 29427294  
## 5 5 29974128  
## 6 6 29856388  
## 7 7 31149199  
## 8 8 31149334  
## 9 9 28711426  
## 10 10 30012086  
## 11 11 28639718  
## 12 12 29954084

# tally() and count() have a wt parameter for this purpose  
flights %>% group\_by(month) %>% tally(wt = distance)  
flights %>% count(month, wt = distance)

# group\_size() returns the counts as a vector  
flights %>% group\_by(month) %>% group\_size()

## [1] 27004 24951 28834 28330 28796 28243 29425 29327 27574 28889 27268  
## [12] 28135

# n\_groups() simply reports the number of groups  
flights %>% group\_by(month) %>% n\_groups()

## [1] 12

# group by two variables, summarise, arrange (output is possibly confusing)  
flights %>% group\_by(month, day) %>% summarise(cnt = n()) %>% arrange(desc(cnt)) %>% print(n = 40)

## # A tibble: 365 x 3  
## # Groups: month [12]  
## month day cnt  
## <int> <int> <int>  
## 1 11 27 1014  
## 2 7 11 1006  
## 3 7 8 1004  
## 4 7 10 1004  
## 5 12 2 1004  
## 6 7 18 1003  
## 7 7 25 1003  
## 8 7 12 1002  
## 9 7 9 1001  
## 10 7 17 1001  
## 11 7 31 1001  
## 12 8 7 1001  
## 13 8 8 1001  
## 14 8 12 1001  
## 15 7 22 1000  
## 16 7 24 1000  
## 17 8 1 1000  
## 18 8 5 1000  
## 19 8 15 1000  
## 20 11 21 1000  
## 21 7 15 999  
## 22 7 19 999  
## 23 7 26 999  
## 24 7 29 999  
## 25 8 2 999  
## 26 8 9 999  
## 27 11 22 999  
## 28 8 16 998  
## 29 7 23 997  
## 30 7 30 997  
## 31 8 14 997  
## 32 7 16 996  
## 33 8 6 996  
## 34 8 19 996  
## 35 9 13 996  
## 36 9 26 996  
## 37 9 27 996  
## 38 4 15 995  
## 39 6 20 995  
## 40 6 26 995  
## # ... with 325 more rows

# ungroup() before arranging to arrange across all groups  
flights %>% group\_by(month, day) %>% summarise(cnt = n()) %>% ungroup() %>% arrange(desc(cnt))

## # A tibble: 365 x 3  
## month day cnt  
## <int> <int> <int>  
## 1 11 27 1014  
## 2 7 11 1006  
## 3 7 8 1004  
## 4 7 10 1004  
## 5 12 2 1004  
## 6 7 18 1003  
## 7 7 25 1003  
## 8 7 12 1002  
## 9 7 9 1001  
## 10 7 17 1001  
## # ... with 355 more rows

## Creating data frames: data\_frame

data\_frame() is a better way than data.frame() for creating data frames. Benefits of data\_frame():

* You can use previously defined columns to compute new columns.
* It never coerces column types.
* It never munges column names.
* It never adds row names.
* It only recycles length 1 input.
* It returns a local data frame (a tbl\_df).

# data\_frame() example  
data\_frame(a = 1:6, b = a\*2, c = 'string', 'd+e' = 1) %>% glimpse()

## Observations: 6  
## Variables: 4  
## $ a <int> 1, 2, 3, 4, 5, 6  
## $ b <dbl> 2, 4, 6, 8, 10, 12  
## $ c <chr> "string", "string", "string", "string", "string", "string"  
## $ `d+e` <dbl> 1, 1, 1, 1, 1, 1

# data.frame() example  
data.frame(a = 1:6, c = 'string', 'd+e' = 1) %>% glimpse()

## Observations: 6  
## Variables: 3  
## $ a <int> 1, 2, 3, 4, 5, 6  
## $ c <fct> string, string, string, string, string, string  
## $ d.e <dbl> 1, 1, 1, 1, 1, 1

## Joining (merging) tables: left\_join, right\_join, inner\_join, full\_join, semi\_join, anti\_join

# create two simple data frames  
(a <- data\_frame(color = c("green","yellow","red"), num = 1:3))

## # A tibble: 3 x 2  
## color num  
## <chr> <int>  
## 1 green 1  
## 2 yellow 2  
## 3 red 3

(b <- data\_frame(color = c("green","yellow","pink"), size = c("S","M","L")))

## # A tibble: 3 x 2  
## color size   
## <chr> <chr>  
## 1 green S   
## 2 yellow M   
## 3 pink L

# only include observations found in both "a" and "b" (automatically joins on variables that appear in both tables)  
inner\_join(a, b)

## Joining, by = "color"

## # A tibble: 2 x 3  
## color num size   
## <chr> <int> <chr>  
## 1 green 1 S   
## 2 yellow 2 M

# include observations found in either "a" or "b"  
full\_join(a, b)

## Joining, by = "color"

## # A tibble: 4 x 3  
## color num size   
## <chr> <int> <chr>  
## 1 green 1 S   
## 2 yellow 2 M   
## 3 red 3 <NA>   
## 4 pink NA L

# include all observations found in "a"  
left\_join(a, b)

## Joining, by = "color"

## # A tibble: 3 x 3  
## color num size   
## <chr> <int> <chr>  
## 1 green 1 S   
## 2 yellow 2 M   
## 3 red 3 <NA>

# include all observations found in "b"  
right\_join(a, b)

## Joining, by = "color"

## # A tibble: 3 x 3  
## color num size   
## <chr> <int> <chr>  
## 1 green 1 S   
## 2 yellow 2 M   
## 3 pink NA L

# right\_join(a, b) is identical to left\_join(b, a) except for column ordering  
left\_join(b, a)

## Joining, by = "color"

## # A tibble: 3 x 3  
## color size num  
## <chr> <chr> <int>  
## 1 green S 1  
## 2 yellow M 2  
## 3 pink L NA

# filter "a" to only show observations that match "b"  
semi\_join(a, b)

## Joining, by = "color"

## # A tibble: 2 x 2  
## color num  
## <chr> <int>  
## 1 green 1  
## 2 yellow 2

# filter "a" to only show observations that don't match "b"  
anti\_join(a, b)

## Joining, by = "color"

## # A tibble: 1 x 2  
## color num  
## <chr> <int>  
## 1 red 3

# sometimes matching variables don't have identical names  
b <- b %>% rename(col = color)  
# specify that the join should occur by matching "color" in "a" with "col" in "b"  
inner\_join(a, b, by=c("color" = "col"))

## # A tibble: 2 x 3  
## color num size   
## <chr> <int> <chr>  
## 1 green 1 S   
## 2 yellow 2 M

## Viewing more output: print, View

# specify that you want to see more rows  
flights %>% print(n = 15)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## 11 2013 1 1 558 600 -2 849  
## 12 2013 1 1 558 600 -2 853  
## 13 2013 1 1 558 600 -2 924  
## 14 2013 1 1 558 600 -2 923  
## 15 2013 1 1 559 600 -1 941  
## # ... with 3.368e+05 more rows, and 12 more variables:  
## # sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

# specify that you want to see ALL rows (don't run this!)  
flights %>% print(n = Inf)

# specify that you want to see all columns  
flights %>% print(width = Inf)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## sched\_arr\_time arr\_delay carrier flight tailnum origin dest air\_time  
## <int> <dbl> <chr> <int> <chr> <chr> <chr> <dbl>  
## 1 819 11 UA 1545 N14228 EWR IAH 227  
## 2 830 20 UA 1714 N24211 LGA IAH 227  
## 3 850 33 AA 1141 N619AA JFK MIA 160  
## 4 1022 -18 B6 725 N804JB JFK BQN 183  
## 5 837 -25 DL 461 N668DN LGA ATL 116  
## 6 728 12 UA 1696 N39463 EWR ORD 150  
## 7 854 19 B6 507 N516JB EWR FLL 158  
## 8 723 -14 EV 5708 N829AS LGA IAD 53  
## 9 846 -8 B6 79 N593JB JFK MCO 140  
## 10 745 8 AA 301 N3ALAA LGA ORD 138  
## distance hour minute time\_hour   
## <dbl> <dbl> <dbl> <dttm>   
## 1 1400 5 15 2013-01-01 05:00:00  
## 2 1416 5 29 2013-01-01 05:00:00  
## 3 1089 5 40 2013-01-01 05:00:00  
## 4 1576 5 45 2013-01-01 05:00:00  
## 5 762 6 0 2013-01-01 06:00:00  
## 6 719 5 58 2013-01-01 05:00:00  
## 7 1065 6 0 2013-01-01 06:00:00  
## 8 229 6 0 2013-01-01 06:00:00  
## 9 944 6 0 2013-01-01 06:00:00  
## 10 733 6 0 2013-01-01 06:00:00  
## # ... with 336,766 more rows

# show up to 1000 rows and all columns  
flights %>% View()  
# set option to see all columns and fewer rows  
options(dplyr.width = Inf, dplyr.print\_min = 6)  
# reset options (or just close R)  
options(dplyr.width = NULL, dplyr.print\_min = 10)

## Resources