Chapter 3: Interactive Notebook for Instructors

Ram Gopal, Dan Philps, and Tillman Weyde

Summer 2022

Table of Contents

# Introduction

In this lesson, we will explore a very popular package called dplyr. Similar to ggplot it provides common verbs useful for data manipulation and management. If you already installed tidyverse, you do not have to install dplyr because it is included in this package. Once installed, load the package.

library(dplyr)  
library(tidyverse)  
library(readxl)  
library(ggplot2)  
library(hflights)

Some of the common verbs used in dplyr are as follows:

select() select columns  
filter() filter rows  
arrange() re-order or arrange rows  
mutate() create new columns  
summarise() summarize values  
group\_by() allows for operating on groups

To see the power of dplyr, we will work with a fairly complex data set. There is a data set called hflights in a package also called hflights. Let us load and take a look at this data set.

glimpse(hflights)

## Rows: 227,496  
## Columns: 21  
## $ Year <int> 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011…  
## $ Month <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ DayofMonth <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1…  
## $ DayOfWeek <int> 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2…  
## $ DepTime <int> 1400, 1401, 1352, 1403, 1405, 1359, 1359, 1355, 1443…  
## $ ArrTime <int> 1500, 1501, 1502, 1513, 1507, 1503, 1509, 1454, 1554…  
## $ UniqueCarrier <chr> "AA", "AA", "AA", "AA", "AA", "AA", "AA", "AA", "AA"…  
## $ FlightNum <int> 428, 428, 428, 428, 428, 428, 428, 428, 428, 428, 42…  
## $ TailNum <chr> "N576AA", "N557AA", "N541AA", "N403AA", "N492AA", "N…  
## $ ActualElapsedTime <int> 60, 60, 70, 70, 62, 64, 70, 59, 71, 70, 70, 56, 63, …  
## $ AirTime <int> 40, 45, 48, 39, 44, 45, 43, 40, 41, 45, 42, 41, 44, …  
## $ ArrDelay <int> -10, -9, -8, 3, -3, -7, -1, -16, 44, 43, 29, 5, -9, …  
## $ DepDelay <int> 0, 1, -8, 3, 5, -1, -1, -5, 43, 43, 29, 19, -2, -3, …  
## $ Origin <chr> "IAH", "IAH", "IAH", "IAH", "IAH", "IAH", "IAH", "IA…  
## $ Dest <chr> "DFW", "DFW", "DFW", "DFW", "DFW", "DFW", "DFW", "DF…  
## $ Distance <int> 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 22…  
## $ TaxiIn <int> 7, 6, 5, 9, 9, 6, 12, 7, 8, 6, 8, 4, 6, 5, 6, 12, 8,…  
## $ TaxiOut <int> 13, 9, 17, 22, 9, 13, 15, 12, 22, 19, 20, 11, 13, 15…  
## $ Cancelled <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ CancellationCode <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", …  
## $ Diverted <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…

As you can see, it’s a fairly complex data set, with 227496 rows and 21 columns. We can use a special type of table called kable to view parts of the data.

knitr::kable(hflights[1:5,1:8],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 1 | 6 | 1400 | 1500 | AA | 428 |
| 2011 | 1 | 2 | 7 | 1401 | 1501 | AA | 428 |
| 2011 | 1 | 3 | 1 | 1352 | 1502 | AA | 428 |
| 2011 | 1 | 4 | 2 | 1403 | 1513 | AA | 428 |
| 2011 | 1 | 5 | 3 | 1405 | 1507 | AA | 428 |

We will start to explore each of the important “verbs” in dplyr.

# Select

Select is used to select columns in a data frame and the syntax is  
select(data, columns).  
From this data set, here are some examples.

d1 = select(hflights, ArrTime, DepTime)  
knitr::kable(d1[1:5,])

|  | ArrTime | DepTime |
| --- | --- | --- |
| 5424 | 1500 | 1400 |
| 5425 | 1501 | 1401 |
| 5426 | 1502 | 1352 |
| 5427 | 1513 | 1403 |
| 5428 | 1507 | 1405 |

You can also search based on names of the columns. The following helper functions can be useful sometimes.

## Helper Functions

dplyr comes with a set of helper functions that can help you select groups of variables inside a select() call:

starts\_with("X") every name that starts with “X”,  
ends\_with("X") every name that ends with “X”,  
contains("X") every name that contains “X”,  
matches("X") every name that matches “X”, where “X” can be a regular expression,  
num\_range("x", 1:5) the variables named x01, x02, x03, x04 and x05,  
one\_of(x) every name that appears in x, which should be a character vector.

When you refer to columns directly inside select(), you don’t use quotes. If you use the helper functions, you do use quotes.

For example, let us say you want to see all the columns which contain the word “time” in the name.

d2 = select(hflights, contains("time"))  
knitr::kable(d2[1:5,])

|  | DepTime | ArrTime | ActualElapsedTime | AirTime |
| --- | --- | --- | --- | --- |
| 5424 | 1400 | 1500 | 60 | 40 |
| 5425 | 1401 | 1501 | 60 | 45 |
| 5426 | 1352 | 1502 | 70 | 48 |
| 5427 | 1403 | 1513 | 70 | 39 |
| 5428 | 1405 | 1507 | 62 | 44 |

All of these columns contain “time”. Keep in mind that it is not case sensitive.

Now suppose you want to select columns that contain either “time” or “delay.”

d3 = select(hflights, contains("time"), contains("delay"))  
knitr::kable(d3[1:5,])

|  | DepTime | ArrTime | ActualElapsedTime | AirTime | ArrDelay | DepDelay |
| --- | --- | --- | --- | --- | --- | --- |
| 5424 | 1400 | 1500 | 60 | 40 | -10 | 0 |
| 5425 | 1401 | 1501 | 60 | 45 | -9 | 1 |
| 5426 | 1352 | 1502 | 70 | 48 | -8 | -8 |
| 5427 | 1403 | 1513 | 70 | 39 | 3 | 3 |
| 5428 | 1405 | 1507 | 62 | 44 | -3 | 5 |

You will notice that some columns are numeric while others are not. The following can be useful to pull out columns that are of a particular data type. Let us bring out all of the numeric columns.

d4 = select\_if(hflights, is.numeric)  
knitr::kable(d4[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | FlightNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Distance | TaxiIn | TaxiOut | Cancelled | Diverted |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 1 | 6 | 1400 | 1500 | 428 | 60 | 40 | -10 | 0 | 224 | 7 | 13 | 0 | 0 |
| 2011 | 1 | 2 | 7 | 1401 | 1501 | 428 | 60 | 45 | -9 | 1 | 224 | 6 | 9 | 0 | 0 |
| 2011 | 1 | 3 | 1 | 1352 | 1502 | 428 | 70 | 48 | -8 | -8 | 224 | 5 | 17 | 0 | 0 |
| 2011 | 1 | 4 | 2 | 1403 | 1513 | 428 | 70 | 39 | 3 | 3 | 224 | 9 | 22 | 0 | 0 |
| 2011 | 1 | 5 | 3 | 1405 | 1507 | 428 | 62 | 44 | -3 | 5 | 224 | 9 | 9 | 0 | 0 |

vice versa, let us bring out all the character columns.

d5 = select\_if(hflights, is.character)  
knitr::kable(d5[1:5,],row.names = F)

| UniqueCarrier | TailNum | Origin | Dest | CancellationCode |
| --- | --- | --- | --- | --- |
| AA | N576AA | IAH | DFW |  |
| AA | N557AA | IAH | DFW |  |
| AA | N541AA | IAH | DFW |  |
| AA | N403AA | IAH | DFW |  |
| AA | N492AA | IAH | DFW |  |

# Filter Function

The filter() function is used to select rows. For example, let us subset the rows with a distance over 1000 miles.

d6 = filter(hflights, Distance>1000)  
dim(d6)

## [1] 65389 21

Suppose you want to get all the flights that were canceled because of weather or security. These are labelled with the characters “B” and “D” in the CancellationCode column.

d7 = filter(hflights, CancellationCode == "D" | CancellationCode == "B")  
dim(d7)

## [1] 1653 21

Another way to write this is:

d8 = filter(hflights, CancellationCode %in% c("B","D"))  
dim(d8)

## [1] 1653 21

%in% identifies the elements belonging to the vector CancellationCode. Note that we always use c(), even if we are only looking for the rows which contain one element.

Let us do two more.  
1. Select all the flights where the total delay, which is ArrDelay + DepDelay, is more than one hour. Note that the time is in minutes.  
2. Select all flights where AirTime is less than the total delay (what we calculated before).

hflights$TotalDelay = hflights$ArrDelay + hflights$DepDelay   
d9 = filter(hflights, TotalDelay>60)  
dim(d9)

## [1] 22367 22

knitr::kable(d9[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 9 | 7 | 1443 | 1554 | AA | 428 | N476AA | 71 | 41 | 44 | 43 | IAH | DFW | 224 | 8 | 22 | 0 |  | 0 | 87 |
| 2011 | 1 | 10 | 1 | 1443 | 1553 | AA | 428 | N504AA | 70 | 45 | 43 | 43 | IAH | DFW | 224 | 6 | 19 | 0 |  | 0 | 86 |
| 2011 | 1 | 17 | 1 | 1530 | 1634 | AA | 428 | N518AA | 64 | 48 | 84 | 90 | IAH | DFW | 224 | 8 | 8 | 0 |  | 0 | 174 |
| 2011 | 1 | 20 | 4 | 1507 | 1622 | AA | 428 | N425AA | 75 | 42 | 72 | 67 | IAH | DFW | 224 | 9 | 24 | 0 |  | 0 | 139 |
| 2011 | 1 | 31 | 1 | 1441 | 1553 | AA | 428 | N505AA | 72 | 39 | 43 | 41 | IAH | DFW | 224 | 8 | 25 | 0 |  | 0 | 84 |

d10 = filter(hflights, AirTime<TotalDelay)  
knitr::kable(d10[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 9 | 7 | 1443 | 1554 | AA | 428 | N476AA | 71 | 41 | 44 | 43 | IAH | DFW | 224 | 8 | 22 | 0 |  | 0 | 87 |
| 2011 | 1 | 10 | 1 | 1443 | 1553 | AA | 428 | N504AA | 70 | 45 | 43 | 43 | IAH | DFW | 224 | 6 | 19 | 0 |  | 0 | 86 |
| 2011 | 1 | 11 | 2 | 1429 | 1539 | AA | 428 | N565AA | 70 | 42 | 29 | 29 | IAH | DFW | 224 | 8 | 20 | 0 |  | 0 | 58 |
| 2011 | 1 | 17 | 1 | 1530 | 1634 | AA | 428 | N518AA | 64 | 48 | 84 | 90 | IAH | DFW | 224 | 8 | 8 | 0 |  | 0 | 174 |
| 2011 | 1 | 20 | 4 | 1507 | 1622 | AA | 428 | N425AA | 75 | 42 | 72 | 67 | IAH | DFW | 224 | 9 | 24 | 0 |  | 0 | 139 |

Another way to do this is as follows:

d9 = filter(hflights, (ArrDelay + DepDelay)>60)  
dim(d9)

## [1] 22367 22

knitr::kable(d9[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 9 | 7 | 1443 | 1554 | AA | 428 | N476AA | 71 | 41 | 44 | 43 | IAH | DFW | 224 | 8 | 22 | 0 |  | 0 | 87 |
| 2011 | 1 | 10 | 1 | 1443 | 1553 | AA | 428 | N504AA | 70 | 45 | 43 | 43 | IAH | DFW | 224 | 6 | 19 | 0 |  | 0 | 86 |
| 2011 | 1 | 17 | 1 | 1530 | 1634 | AA | 428 | N518AA | 64 | 48 | 84 | 90 | IAH | DFW | 224 | 8 | 8 | 0 |  | 0 | 174 |
| 2011 | 1 | 20 | 4 | 1507 | 1622 | AA | 428 | N425AA | 75 | 42 | 72 | 67 | IAH | DFW | 224 | 9 | 24 | 0 |  | 0 | 139 |
| 2011 | 1 | 31 | 1 | 1441 | 1553 | AA | 428 | N505AA | 72 | 39 | 43 | 41 | IAH | DFW | 224 | 8 | 25 | 0 |  | 0 | 84 |

d10 = filter(hflights, AirTime<(ArrDelay + DepDelay))  
knitr::kable(d10[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 9 | 7 | 1443 | 1554 | AA | 428 | N476AA | 71 | 41 | 44 | 43 | IAH | DFW | 224 | 8 | 22 | 0 |  | 0 | 87 |
| 2011 | 1 | 10 | 1 | 1443 | 1553 | AA | 428 | N504AA | 70 | 45 | 43 | 43 | IAH | DFW | 224 | 6 | 19 | 0 |  | 0 | 86 |
| 2011 | 1 | 11 | 2 | 1429 | 1539 | AA | 428 | N565AA | 70 | 42 | 29 | 29 | IAH | DFW | 224 | 8 | 20 | 0 |  | 0 | 58 |
| 2011 | 1 | 17 | 1 | 1530 | 1634 | AA | 428 | N518AA | 64 | 48 | 84 | 90 | IAH | DFW | 224 | 8 | 8 | 0 |  | 0 | 174 |
| 2011 | 1 | 20 | 4 | 1507 | 1622 | AA | 428 | N425AA | 75 | 42 | 72 | 67 | IAH | DFW | 224 | 9 | 24 | 0 |  | 0 | 139 |

Now, suppose we want to create a data set with only the columns FlightNum, DayOfWeek, ArrTime, ArrDelay, and DepDelay for all the flights where AirTime is less than the total delay (ArrDelay + DepDelay).

d11 = filter(hflights, AirTime<(ArrDelay + DepDelay))  
d12 = select(d11, FlightNum, DayOfWeek, AirTime, ArrDelay, DepDelay)  
knitr::kable(d12[1:5,],row.names = F)

| FlightNum | DayOfWeek | AirTime | ArrDelay | DepDelay |
| --- | --- | --- | --- | --- |
| 428 | 7 | 41 | 44 | 43 |
| 428 | 1 | 45 | 43 | 43 |
| 428 | 2 | 42 | 29 | 29 |
| 428 | 1 | 48 | 84 | 90 |
| 428 | 4 | 42 | 72 | 67 |

Note that if you have more than one condition, you can simply add them after a comma.

# Piping

Sometimes, doing a series of operations on a variable can become complicated, especially when many parenthesis are involved. For example, the following can be difficult to understand. Notice the number of parenthesis involved in creating the function.

x = c(25,2,72,456,8,34,6,7,2,3,4)  
round(exp(sqrt(log(x))), 1)

## [1] 6.0 2.3 7.9 11.9 4.2 6.5 3.8 4.0 2.3 2.9 3.2

Piping is an important operator that can make writing such expressions simpler. The operator for piping is %>% and it comes with the dplyr package. The above example can be written using piping as follows.

x %>% log() %>%  
 sqrt() %>%  
 exp() %>%  
 round(1)

## [1] 6.0 2.3 7.9 11.9 4.2 6.5 3.8 4.0 2.3 2.9 3.2

The mathematical reasoning behind piping has to do with composite functions. When solving f(g(x)), the input x is run through the function g, and then g(x) is run through f. This would be written like so:  
x %>% g() %>% f()

To explain further, function(argument) is rewritten as argument %>% function().

Now, let us rewrite the last example we did in the previous section using piping.

d12 = hflights %>%   
 filter(ArrTime<(ArrDelay + DepDelay)) %>%  
 select(FlightNum, DayOfWeek, ArrTime, ArrDelay, DepDelay)  
knitr::kable(d12[1:5,])

| FlightNum | DayOfWeek | ArrTime | ArrDelay | DepDelay |
| --- | --- | --- | --- | --- |
| 310 | 1 | 2 | 33 | 49 |
| 467 | 1 | 53 | 58 | 42 |
| 209 | 7 | 19 | 20 | 37 |
| 310 | 7 | 5 | 36 | 25 |
| 467 | 7 | 10 | 15 | 26 |

To the extent possible, try to use the pipe operator to write your code. Check out the further reading for a more in-depth piping tutorial.

# Rename Function

The rename function can be used to rename the columns of a data frame. The syntax is:  
rename(dataframe, new name = old name)

Let us say we want to change the columns DepTime to DepartureTime and ArrTime to ArrivalTime. We will use the piping method.

d13 = hflights %>%   
 rename(DepartureTime = DepTime, ArrivalTime=ArrTime)  
knitr::kable(d13[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepartureTime | ArrivalTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 1 | 6 | 1400 | 1500 | AA | 428 | N576AA | 60 | 40 | -10 | 0 | IAH | DFW | 224 | 7 | 13 | 0 |  | 0 | -10 |
| 2011 | 1 | 2 | 7 | 1401 | 1501 | AA | 428 | N557AA | 60 | 45 | -9 | 1 | IAH | DFW | 224 | 6 | 9 | 0 |  | 0 | -8 |
| 2011 | 1 | 3 | 1 | 1352 | 1502 | AA | 428 | N541AA | 70 | 48 | -8 | -8 | IAH | DFW | 224 | 5 | 17 | 0 |  | 0 | -16 |
| 2011 | 1 | 4 | 2 | 1403 | 1513 | AA | 428 | N403AA | 70 | 39 | 3 | 3 | IAH | DFW | 224 | 9 | 22 | 0 |  | 0 | 6 |
| 2011 | 1 | 5 | 3 | 1405 | 1507 | AA | 428 | N492AA | 62 | 44 | -3 | 5 | IAH | DFW | 224 | 9 | 9 | 0 |  | 0 | 2 |

The basic way to do this is as follows:

d14 = rename(hflights, DepartureTime = DepTime, ArrivalTime=ArrTime)  
knitr::kable(d14[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepartureTime | ArrivalTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 1 | 6 | 1400 | 1500 | AA | 428 | N576AA | 60 | 40 | -10 | 0 | IAH | DFW | 224 | 7 | 13 | 0 |  | 0 | -10 |
| 2011 | 1 | 2 | 7 | 1401 | 1501 | AA | 428 | N557AA | 60 | 45 | -9 | 1 | IAH | DFW | 224 | 6 | 9 | 0 |  | 0 | -8 |
| 2011 | 1 | 3 | 1 | 1352 | 1502 | AA | 428 | N541AA | 70 | 48 | -8 | -8 | IAH | DFW | 224 | 5 | 17 | 0 |  | 0 | -16 |
| 2011 | 1 | 4 | 2 | 1403 | 1513 | AA | 428 | N403AA | 70 | 39 | 3 | 3 | IAH | DFW | 224 | 9 | 22 | 0 |  | 0 | 6 |
| 2011 | 1 | 5 | 3 | 1405 | 1507 | AA | 428 | N492AA | 62 | 44 | -3 | 5 | IAH | DFW | 224 | 9 | 9 | 0 |  | 0 | 2 |

# Mutate Function

This option is used to create new columns based on existing columns. The basic syntax is:  
mutate(dataframe, new column name = expression)

Suppose we want to create the following columns using piping:  
ActualGroundTime = ActualElapsedTime - AirTime  
TaxiTime = TaxiIn + TaxiOut  
AvgSpeed = 60 (Distance / AirTime)

d15 = hflights %>%  
 mutate(ActualGroundTime = ActualElapsedTime - AirTime, TaxiTime = TaxiIn +   
 TaxiOut, AvgSpeed = 60 \* (Distance/AirTime))  
knitr::kable(d15[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay | ActualGroundTime | TaxiTime | AvgSpeed |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 1 | 6 | 1400 | 1500 | AA | 428 | N576AA | 60 | 40 | -10 | 0 | IAH | DFW | 224 | 7 | 13 | 0 |  | 0 | -10 | 20 | 20 | 336.0 |
| 2011 | 1 | 2 | 7 | 1401 | 1501 | AA | 428 | N557AA | 60 | 45 | -9 | 1 | IAH | DFW | 224 | 6 | 9 | 0 |  | 0 | -8 | 15 | 15 | 298.7 |
| 2011 | 1 | 3 | 1 | 1352 | 1502 | AA | 428 | N541AA | 70 | 48 | -8 | -8 | IAH | DFW | 224 | 5 | 17 | 0 |  | 0 | -16 | 22 | 22 | 280.0 |
| 2011 | 1 | 4 | 2 | 1403 | 1513 | AA | 428 | N403AA | 70 | 39 | 3 | 3 | IAH | DFW | 224 | 9 | 22 | 0 |  | 0 | 6 | 31 | 31 | 344.6 |
| 2011 | 1 | 5 | 3 | 1405 | 1507 | AA | 428 | N492AA | 62 | 44 | -3 | 5 | IAH | DFW | 224 | 9 | 9 | 0 |  | 0 | 2 | 18 | 18 | 305.5 |

The basic way to do this is as follows:

d16 = mutate(hflights, ActualGroundTime = ActualElapsedTime - AirTime, TaxiTime = TaxiIn +   
 TaxiOut, AvgSpeed = 60 \* (Distance/AirTime))  
knitr::kable(d16[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay | ActualGroundTime | TaxiTime | AvgSpeed |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 1 | 1 | 6 | 1400 | 1500 | AA | 428 | N576AA | 60 | 40 | -10 | 0 | IAH | DFW | 224 | 7 | 13 | 0 |  | 0 | -10 | 20 | 20 | 336.0 |
| 2011 | 1 | 2 | 7 | 1401 | 1501 | AA | 428 | N557AA | 60 | 45 | -9 | 1 | IAH | DFW | 224 | 6 | 9 | 0 |  | 0 | -8 | 15 | 15 | 298.7 |
| 2011 | 1 | 3 | 1 | 1352 | 1502 | AA | 428 | N541AA | 70 | 48 | -8 | -8 | IAH | DFW | 224 | 5 | 17 | 0 |  | 0 | -16 | 22 | 22 | 280.0 |
| 2011 | 1 | 4 | 2 | 1403 | 1513 | AA | 428 | N403AA | 70 | 39 | 3 | 3 | IAH | DFW | 224 | 9 | 22 | 0 |  | 0 | 6 | 31 | 31 | 344.6 |
| 2011 | 1 | 5 | 3 | 1405 | 1507 | AA | 428 | N492AA | 62 | 44 | -3 | 5 | IAH | DFW | 224 | 9 | 9 | 0 |  | 0 | 2 | 18 | 18 | 305.5 |

# Arrange Function

This function is used to reorder the rows of a data frame in ascending or descending order. The basic syntax is:  
arrange(dataframe, columns to arrange by)

Let us say we want to sort the data frame hflights by AirTime.

d17 = hflights %>%  
 arrange(AirTime)  
knitr::kable(d17[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 8 | 31 | 3 | 1255 | 1346 | CO | 1646 | N11612 | 51 | 11 | -7 | 0 | IAH | AUS | 140 | 8 | 32 | 0 |  | 0 | -7 |
| 2011 | 6 | 5 | 7 | 1145 | 1225 | CO | 405 | N32626 | 40 | 19 | -16 | 0 | IAH | SAT | 191 | 6 | 15 | 0 |  | 0 | -16 |
| 2011 | 2 | 23 | 3 | 947 | 1030 | XE | 2204 | N17928 | 43 | 22 | 29 | 32 | IAH | BPT | 79 | 5 | 16 | 0 |  | 0 | 61 |
| 2011 | 4 | 8 | 5 | 2017 | 2057 | XE | 2418 | N14925 | 40 | 22 | 7 | 17 | IAH | LCH | 127 | 6 | 12 | 0 |  | 0 | 24 |
| 2011 | 1 | 3 | 1 | 1951 | 2026 | XE | 2814 | N27152 | 35 | 23 | -19 | -4 | IAH | LCH | 127 | 4 | 8 | 0 |  | 0 | -23 |

You can sort by multiple columns and in descending order using desc(). Note that ascending order is the default.

d18 = hflights %>%  
 arrange(AirTime, UniqueCarrier, desc(Month))  
knitr::kable(d18[1:5,],row.names = F)

| Year | Month | DayofMonth | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | ActualElapsedTime | AirTime | ArrDelay | DepDelay | Origin | Dest | Distance | TaxiIn | TaxiOut | Cancelled | CancellationCode | Diverted | TotalDelay |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2011 | 8 | 31 | 3 | 1255 | 1346 | CO | 1646 | N11612 | 51 | 11 | -7 | 0 | IAH | AUS | 140 | 8 | 32 | 0 |  | 0 | -7 |
| 2011 | 6 | 5 | 7 | 1145 | 1225 | CO | 405 | N32626 | 40 | 19 | -16 | 0 | IAH | SAT | 191 | 6 | 15 | 0 |  | 0 | -16 |
| 2011 | 4 | 8 | 5 | 2017 | 2057 | XE | 2418 | N14925 | 40 | 22 | 7 | 17 | IAH | LCH | 127 | 6 | 12 | 0 |  | 0 | 24 |
| 2011 | 2 | 23 | 3 | 947 | 1030 | XE | 2204 | N17928 | 43 | 22 | 29 | 32 | IAH | BPT | 79 | 5 | 16 | 0 |  | 0 | 61 |
| 2011 | 7 | 19 | 2 | 1043 | 1119 | CO | 1583 | N16713 | 36 | 23 | 70 | 93 | IAH | AUS | 140 | 3 | 10 | 0 |  | 0 | 163 |

# Summarize Function

This function is used to summarize data, but it usually goes hand-in-hand with the group\_by() function. The syntax is:

g1 = group\_by(dataframe, factor variables you want to group by)  
s1 = summarize(g1, variable name = some aggregating function)

Let us say we want to get the average air time for each unique carrier.

g1 = group\_by(hflights, UniqueCarrier)  
s1 = summarize(g1, AvgTime = mean(AirTime, na.rm = T))  
knitr::kable(s1)

| UniqueCarrier | AvgTime |
| --- | --- |
| AA | 69.65 |
| AS | 254.18 |
| B6 | 183.99 |
| CO | 145.46 |
| DL | 97.80 |
| EV | 103.66 |
| F9 | 125.34 |
| FL | 92.71 |
| MQ | 93.84 |
| OO | 113.41 |
| UA | 157.41 |
| US | 133.86 |
| WN | 86.73 |
| XE | 83.22 |
| YV | 121.94 |

Remember to add na.rm = T if the data contains missing values.

You can also pipe this to make it read better.

s2 = hflights %>%  
 group\_by(UniqueCarrier) %>%  
 summarize(AvgTime = mean(AirTime, na.rm = T))  
knitr::kable(s2)

| UniqueCarrier | AvgTime |
| --- | --- |
| AA | 69.65 |
| AS | 254.18 |
| B6 | 183.99 |
| CO | 145.46 |
| DL | 97.80 |
| EV | 103.66 |
| F9 | 125.34 |
| FL | 92.71 |
| MQ | 93.84 |
| OO | 113.41 |
| UA | 157.41 |
| US | 133.86 |
| WN | 86.73 |
| XE | 83.22 |
| YV | 121.94 |

You can add a number of other statistics in addition to the mean. You can use the basic aggregating functions that R provides, such as:

min(x) - minimum value of vector x  
max(x) - maximum value of vector x  
mean(x) - mean value of vector x  
median(x) - median value of vector x  
quantile(x, p) - pth quantile of vector x  
sd(x) - standard deviation of vector x  
var(x) - variance of vector x  
IQR(x) - Inter Quartile Range (IQR) of vector x  
diff(range(x)) - total range of vector x

In addition, dplyr provides a number of other aggregate functions which include:

first(x) - The first element of vector x  
last(x) - The last element of vector x  
nth(x, n) - The nth element of vector x  
n() - The number of rows in the data.frame or group of observations that summarise() describes  
n\_distinct(x) - The number of unique values in vector x

Additionally, you can get multiple statistics from multiple groups. Here is an example.

s3 = hflights %>%  
 group\_by(UniqueCarrier, Origin) %>%  
 summarize(AvgTime = mean(AirTime, na.rm = T), NumFlights = n(),   
 LongestFlightTime = max(AirTime, na.rm = T))

## `summarise()` has grouped output by 'UniqueCarrier'. You can override using the  
## `.groups` argument.

knitr::kable(s3)

| UniqueCarrier | Origin | AvgTime | NumFlights | LongestFlightTime |
| --- | --- | --- | --- | --- |
| AA | IAH | 69.65 | 3244 | 161 |
| AS | IAH | 254.18 | 365 | 315 |
| B6 | HOU | 183.99 | 695 | 258 |
| CO | IAH | 145.46 | 70032 | 549 |
| DL | HOU | 92.39 | 388 | 135 |
| DL | IAH | 98.71 | 2253 | 188 |
| EV | HOU | 92.88 | 472 | 117 |
| EV | IAH | 106.48 | 1732 | 173 |
| F9 | HOU | 125.34 | 838 | 190 |
| FL | HOU | 92.71 | 2139 | 186 |
| MQ | HOU | 47.55 | 2424 | 88 |
| MQ | IAH | 144.11 | 2224 | 220 |
| OO | IAH | 113.41 | 16061 | 225 |
| UA | IAH | 157.41 | 2072 | 276 |
| US | IAH | 133.86 | 4082 | 212 |
| WN | HOU | 86.73 | 45343 | 288 |
| XE | IAH | 83.22 | 73053 | 204 |
| YV | IAH | 121.94 | 79 | 150 |

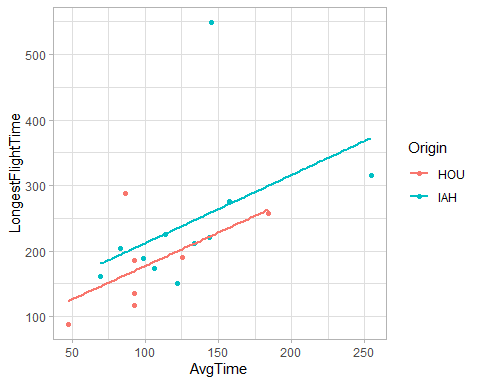
library(ggplot2)  
s3 = hflights %>%  
 group\_by(UniqueCarrier, Origin) %>%  
 summarize(AvgTime = mean(AirTime, na.rm = T), NumFlights = n(),   
 LongestFlightTime = max(AirTime, na.rm = T))

## `summarise()` has grouped output by 'UniqueCarrier'. You can override using the  
## `.groups` argument.

Once a summary is created, you can easily plot this using ggplot2.

ggplot(s3, aes(x=AvgTime, y = LongestFlightTime, col = Origin)) +  
 geom\_point() +  
 theme\_light() +  
 geom\_smooth(method = lm, se = F)

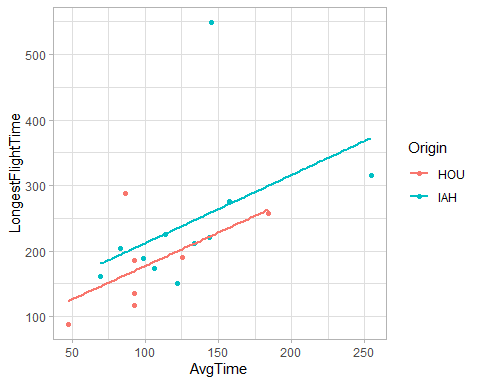
## `geom\_smooth()` using formula 'y ~ x'



Now, we can pipe this altogether,

hflights %>%  
 group\_by(UniqueCarrier, Origin) %>%  
 summarize(AvgTime = mean(AirTime, na.rm = T), NumFlights = n(),   
 LongestFlightTime = max(AirTime, na.rm = T)) %>%  
 ggplot(aes(x=AvgTime, y = LongestFlightTime, col = Origin)) +  
 geom\_point()+  
 theme\_light() +  
 geom\_smooth(method = lm, se = F)

## `summarise()` has grouped output by 'UniqueCarrier'. You can override using the  
## `.groups` argument.  
## `geom\_smooth()` using formula 'y ~ x'



If you want one summary for the entire data frame, use the following syntax:

summarize(dataframe, variable name = some aggregating function)

# Data Reshaping

There are times when the data is not stored in the format you need for analysis. Let us take a look at an example. We will work with a data frame called Rainfall. This is an Excel file you can download from the link at the bottom of this page.

Rainfall <- read\_excel("../../data/Rainfall.xlsx")  
knitr::kable(Rainfall[1:5,])

| State No. | Station Name | Class | Index | Year | Annual total rainfall (mm) | Winter total rainfall (mm) | Summer total rainfall (mm) | Monsoon total rainfall (mm) | Post monsoon total rainfall (mm) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | BOATH | REV | 90014 | 1952 | 714.9 | 64.4 | 18.6 | 526.1 | 103.8 |
| 1 | BOATH | REV | 90014 | 1953 | 1305.7 | 2.0 | 20.1 | 1209.2 | 76.4 |
| 1 | BOATH | REV | 90014 | 1954 | 808.9 | 0.0 | 40.6 | 768.3 | 0.0 |
| 1 | BOATH | REV | 90014 | 1955 | 1245.2 | 5.1 | 18.6 | 1159.2 | 62.3 |
| 1 | BOATH | REV | 90014 | 1956 | 1000.8 | 4.8 | 109.8 | 731.1 | 155.1 |

The last five columns of the data frame contain information about amount of rainfall. We may want to compress these five columns into two columns (this is called the long form). The first column can be called “TotalRainfall”, which will indicate annual, monsoon, etc. The second column, which we will call “Amount(mm)”, will contain the actual rainfall amount.

The dplyr package contains two important verbs for data reshaping.

1. gather() This will gather multiple columns into two columns.
2. spread() This does the opposite. It takes two columns and spreads them into multiple columns.

To create the data in the long form, the code is:

Rainfall\_long = Rainfall %>%  
 gather(key = "TotalRainfall", value = "Amount(mm)",  
 `Annual total rainfall (mm)`: `Post monsoon total rainfall (mm)`)  
  
knitr::kable(Rainfall\_long[1:10,])

| State No. | Station Name | Class | Index | Year | TotalRainfall | Amount(mm) |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | BOATH | REV | 90014 | 1952 | Annual total rainfall (mm) | 714.9 |
| 1 | BOATH | REV | 90014 | 1953 | Annual total rainfall (mm) | 1305.7 |
| 1 | BOATH | REV | 90014 | 1954 | Annual total rainfall (mm) | 808.9 |
| 1 | BOATH | REV | 90014 | 1955 | Annual total rainfall (mm) | 1245.2 |
| 1 | BOATH | REV | 90014 | 1956 | Annual total rainfall (mm) | 1000.8 |
| 1 | BOATH | REV | 90014 | 1957 | Annual total rainfall (mm) | 1074.7 |
| 1 | BOATH | REV | 90014 | 1958 | Annual total rainfall (mm) | 1049.0 |
| 1 | BOATH | REV | 90014 | 1959 | Annual total rainfall (mm) | 1189.8 |
| 1 | BOATH | REV | 90014 | 1960 | Annual total rainfall (mm) | 690.9 |
| 1 | BOATH | REV | 90014 | 1961 | Annual total rainfall (mm) | 1003.7 |

It’s as simple as that!

You can also convert data in the long format to the wide format. The syntax is:

Rainfall\_wide = Rainfall\_long %>%  
 spread(key = "TotalRainfall", value = "Amount(mm)")  
knitr::kable(Rainfall\_wide[1:10,])

| State No. | Station Name | Class | Index | Year | Annual total rainfall (mm) | Monsoon total rainfall (mm) | Post monsoon total rainfall (mm) | Summer total rainfall (mm) | Winter total rainfall (mm) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | BOATH | REV | 90014 | 1952 | 714.9 | 526.1 | 103.8 | 18.6 | 64.4 |
| 1 | BOATH | REV | 90014 | 1953 | 1305.7 | 1209.2 | 76.4 | 20.1 | 2.0 |
| 1 | BOATH | REV | 90014 | 1954 | 808.9 | 768.3 | 0.0 | 40.6 | 0.0 |
| 1 | BOATH | REV | 90014 | 1955 | 1245.2 | 1159.2 | 62.3 | 18.6 | 5.1 |
| 1 | BOATH | REV | 90014 | 1956 | 1000.8 | 731.1 | 155.1 | 109.8 | 4.8 |
| 1 | BOATH | REV | 90014 | 1957 | 1074.7 | 978.2 | 27.9 | 68.6 | 0.0 |
| 1 | BOATH | REV | 90014 | 1958 | 1049.0 | 943.1 | 53.8 | 52.1 | 0.0 |
| 1 | BOATH | REV | 90014 | 1959 | 1189.8 | 1053.6 | 88.5 | 47.7 | 0.0 |
| 1 | BOATH | REV | 90014 | 1960 | 690.9 | 582.2 | 93.3 | 15.4 | 0.0 |
| 1 | BOATH | REV | 90014 | 1961 | 1003.7 | 845.2 | 96.0 | 62.5 | 0.0 |

# Recoding Factor Variables

In the long format, the data in the TotalRainfall column are too long. Let us suppose we want to recode these values as follows:

| TotalRainfall | TotalRainfall\_New |
| --- | --- |
| Annual total rainfall (mm) | Annual |
| Winter total rainfall (mm) | Winter |
| Summer total rainfall (mm) | Summer |
| Monsoon total rainfall (mm) | Monsoon |
| Post monsoon total rainfall (mm) | PostMonsoon |

The strategy is as follows:

1. Create a recode table with the existing values and new values of the factor variable.
2. Join the data frame with the recode table.

TotalRainfall = c("Annual total rainfall (mm)", "Winter total rainfall (mm)",   
 "Summer total rainfall (mm)","Monsoon total rainfall (mm)",   
 "Post monsoon total rainfall (mm)")  
  
TotalRainfall\_new = c("Annual","Winter","Summer","Monsoon","PostMonsoon")  
  
RecodeTable = data.frame(TotalRainfall,TotalRainfall\_new)  
  
knitr::kable(RecodeTable)

| TotalRainfall | TotalRainfall\_new |
| --- | --- |
| Annual total rainfall (mm) | Annual |
| Winter total rainfall (mm) | Winter |
| Summer total rainfall (mm) | Summer |
| Monsoon total rainfall (mm) | Monsoon |
| Post monsoon total rainfall (mm) | PostMonsoon |

Now, we need to join Rainfall\_long and RecodeTable.

NewRainfall = left\_join(Rainfall\_long, RecodeTable, by = "TotalRainfall")  
  
NewRainfall = NewRainfall %>%  
 select(-TotalRainfall)  
  
knitr::kable(NewRainfall[1:10,])

| State No. | Station Name | Class | Index | Year | Amount(mm) | TotalRainfall\_new |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | BOATH | REV | 90014 | 1952 | 714.9 | Annual |
| 1 | BOATH | REV | 90014 | 1953 | 1305.7 | Annual |
| 1 | BOATH | REV | 90014 | 1954 | 808.9 | Annual |
| 1 | BOATH | REV | 90014 | 1955 | 1245.2 | Annual |
| 1 | BOATH | REV | 90014 | 1956 | 1000.8 | Annual |
| 1 | BOATH | REV | 90014 | 1957 | 1074.7 | Annual |
| 1 | BOATH | REV | 90014 | 1958 | 1049.0 | Annual |
| 1 | BOATH | REV | 90014 | 1959 | 1189.8 | Annual |
| 1 | BOATH | REV | 90014 | 1960 | 690.9 | Annual |
| 1 | BOATH | REV | 90014 | 1961 | 1003.7 | Annual |

# Further Reading

[link] <https://www.listendata.com/2016/08/dplyr-tutorial.html#select-Function>  
[link] <https://dereksonderegger.github.io/570L/7-data-manipulation.html>  
[link] <https://genomicsclass.github.io/book/pages/dplyr_tutorial.html>  
[link] <https://www.datacamp.com/community/tutorials/pipe-r-tutorial>