An Introduction to dplyr

Phil Canham

30th October 2018

# An Introduction to dplyr

Dplyr is one of the most useful R librarys for manipulating data. Think of it like a Swiss Army knife for data.

## What does it do?

Once we have a data set to work with, almost always it is not in the format, or shape that we want it to be. In Excel we might for example use filters, formulas, lookups and pivot tables to make the data more useful. In R we can use the tools in the dplyr libarary to do these things and much more.

The other important thing to remember is that in R we normally provide a series of instructions which can then always be repeated and changed without having to start again each time, so we have total control of what happens to the data, and the original data should never be permanently overwritten unless we explicity ask R to do this (which is generally bad practice)

The author of the library, Hadley Wickham, has streamlined the process of data manipulation by following a common format using the concept of verbs, (ie words used to describe actions). The key dplyr verbs we will consider today are:

* filter
* select
* arrange
* mutate
* group\_by
* summarise

### Lets try them out

### First we clear the memory and any previously loaded work from R Studio

rm(list= ls())

### Second, set the working directory to allow us to load in our data

setwd('//dsfin/corpcom/LBBDR/TNG/Session 3')

### We can now load any librarys we are interested in using - here we load the dplyr library

library(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

# I also want to use another library which prints out much neater tables but we dont need to do this  
library(tibble)

### The next step is to read in the data from our csv file

flights <- read.csv('session3a.csv')

### and have a look at a summary of the data we have loaded

summary(flights)

## X Year Month DayofMonth   
## Min. : 5424 Min. :2011 Min. : 1.000 Min. : 1.00   
## 1st Qu.:1555452 1st Qu.:2011 1st Qu.: 4.000 1st Qu.: 8.00   
## Median :3114682 Median :2011 Median : 7.000 Median :16.00   
## Mean :3021458 Mean :2011 Mean : 6.514 Mean :15.74   
## 3rd Qu.:4556825 3rd Qu.:2011 3rd Qu.: 9.000 3rd Qu.:23.00   
## Max. :6083259 Max. :2011 Max. :12.000 Max. :31.00   
##   
## DayOfWeek DepTime ArrTime UniqueCarrier   
## Min. :1.000 Min. : 1 Min. : 1 XE :73053   
## 1st Qu.:2.000 1st Qu.:1021 1st Qu.:1215 CO :70032   
## Median :4.000 Median :1416 Median :1617 WN :45343   
## Mean :3.948 Mean :1396 Mean :1578 OO :16061   
## 3rd Qu.:6.000 3rd Qu.:1801 3rd Qu.:1953 MQ : 4648   
## Max. :7.000 Max. :2400 Max. :2400 US : 4082   
## NA's :2905 NA's :3066 (Other):14277   
## FlightNum TailNum ActualElapsedTime AirTime   
## Min. : 1 N14945 : 971 Min. : 34.0 Min. : 11.0   
## 1st Qu.: 855 N15926 : 960 1st Qu.: 77.0 1st Qu.: 58.0   
## Median :1696 N16927 : 951 Median :128.0 Median :107.0   
## Mean :1962 N12946 : 948 Mean :129.3 Mean :108.1   
## 3rd Qu.:2755 N14937 : 946 3rd Qu.:165.0 3rd Qu.:141.0   
## Max. :7290 N14942 : 946 Max. :575.0 Max. :549.0   
## (Other):221774 NA's :3622 NA's :3622   
## ArrDelay DepDelay Origin Dest   
## Min. :-70.000 Min. :-33.000 HOU: 52299 DAL : 9820   
## 1st Qu.: -8.000 1st Qu.: -3.000 IAH:175197 ATL : 7886   
## Median : 0.000 Median : 0.000 MSY : 6823   
## Mean : 7.094 Mean : 9.445 DFW : 6653   
## 3rd Qu.: 11.000 3rd Qu.: 9.000 LAX : 6064   
## Max. :978.000 Max. :981.000 DEN : 5920   
## NA's :3622 NA's :2905 (Other):184330   
## Distance TaxiIn TaxiOut Cancelled   
## Min. : 79.0 Min. : 1.000 Min. : 1.00 Min. :0.00000   
## 1st Qu.: 376.0 1st Qu.: 4.000 1st Qu.: 10.00 1st Qu.:0.00000   
## Median : 809.0 Median : 5.000 Median : 14.00 Median :0.00000   
## Mean : 787.8 Mean : 6.099 Mean : 15.09 Mean :0.01307   
## 3rd Qu.:1042.0 3rd Qu.: 7.000 3rd Qu.: 18.00 3rd Qu.:0.00000   
## Max. :3904.0 Max. :165.000 Max. :163.00 Max. :1.00000   
## NA's :3066 NA's :2947   
## CancellationCode Diverted   
## :224523 Min. :0.000000   
## A: 1202 1st Qu.:0.000000   
## B: 1652 Median :0.000000   
## C: 118 Mean :0.002853   
## D: 1 3rd Qu.:0.000000   
## Max. :1.000000   
##

This dataset contains all flights departing from Houston airports IAH (George Bush Intercontinental) and HOU (Houston Hobby) in 2011. The data comes from the Research and Innovation Technology Administration at the Bureau of Transportation statistics: <http://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=120&Link=0>

Information on the variables and the dataset can be found [here](https://cran.r-project.org/web/packages/hflights/hflights.pdf)

There is also a pdf of this document in the working directory folder

### now check the top few rows of the data

head(flights)

## X Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier  
## 1 5424 2011 1 1 6 1400 1500 AA  
## 2 5425 2011 1 2 7 1401 1501 AA  
## 3 5426 2011 1 3 1 1352 1502 AA  
## 4 5427 2011 1 4 2 1403 1513 AA  
## 5 5428 2011 1 5 3 1405 1507 AA  
## 6 5429 2011 1 6 4 1359 1503 AA  
## FlightNum TailNum ActualElapsedTime AirTime ArrDelay DepDelay Origin  
## 1 428 N576AA 60 40 -10 0 IAH  
## 2 428 N557AA 60 45 -9 1 IAH  
## 3 428 N541AA 70 48 -8 -8 IAH  
## 4 428 N403AA 70 39 3 3 IAH  
## 5 428 N492AA 62 44 -3 5 IAH  
## 6 428 N262AA 64 45 -7 -1 IAH  
## Dest Distance TaxiIn TaxiOut Cancelled CancellationCode Diverted  
## 1 DFW 224 7 13 0 0  
## 2 DFW 224 6 9 0 0  
## 3 DFW 224 5 17 0 0  
## 4 DFW 224 9 22 0 0  
## 5 DFW 224 9 9 0 0  
## 6 DFW 224 6 13 0 0

### lets use the tibble library to convert our data frame into a format that prints out nicely

flights <- as.tibble(flights) ## remember we dont need to do this step it is just for better display

## 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$Month==1 & flights$DayofMonth==1, ]

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

## # A tibble: 552 x 22  
## X Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier  
## <int> <int> <int> <int> <int> <int> <int> <fctr>  
## 1 5424 2011 1 1 6 1400 1500 AA  
## 2 6343 2011 1 1 6 728 840 AA  
## 3 19266 2011 1 1 6 1631 1736 AA  
## 4 23655 2011 1 1 6 1756 2112 AA  
## 5 33051 2011 1 1 6 1012 1347 AA  
## 6 35256 2011 1 1 6 1211 1325 AA  
## 7 39453 2011 1 1 6 557 906 AA  
## 8 46433 2011 1 1 6 1824 2106 AS  
## 9 57719 2011 1 1 6 654 1124 B6  
## 10 57720 2011 1 1 6 1639 2110 B6  
## # ... with 542 more rows, and 14 more variables: FlightNum <int>,  
## # TailNum <fctr>, ActualElapsedTime <int>, AirTime <int>,  
## # ArrDelay <int>, DepDelay <int>, Origin <fctr>, Dest <fctr>,  
## # Distance <int>, TaxiIn <int>, TaxiOut <int>, Cancelled <int>,  
## # CancellationCode <fctr>, Diverted <int>

# use pipe for OR condition  
filter(flights, UniqueCarrier=="AA" | UniqueCarrier=="UA")

## # A tibble: 5,316 x 22  
## X Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier  
## <int> <int> <int> <int> <int> <int> <int> <fctr>  
## 1 5424 2011 1 1 6 1400 1500 AA  
## 2 5425 2011 1 2 7 1401 1501 AA  
## 3 5426 2011 1 3 1 1352 1502 AA  
## 4 5427 2011 1 4 2 1403 1513 AA  
## 5 5428 2011 1 5 3 1405 1507 AA  
## 6 5429 2011 1 6 4 1359 1503 AA  
## 7 5430 2011 1 7 5 1359 1509 AA  
## 8 5431 2011 1 8 6 1355 1454 AA  
## 9 5432 2011 1 9 7 1443 1554 AA  
## 10 5433 2011 1 10 1 1443 1553 AA  
## # ... with 5,306 more rows, and 14 more variables: FlightNum <int>,  
## # TailNum <fctr>, ActualElapsedTime <int>, AirTime <int>,  
## # ArrDelay <int>, DepDelay <int>, Origin <fctr>, Dest <fctr>,  
## # Distance <int>, TaxiIn <int>, TaxiOut <int>, Cancelled <int>,  
## # CancellationCode <fctr>, Diverted <int>

# you can also use %in% operator  
filter(flights, UniqueCarrier %in% c("AA", "UA"))

## 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, ArrTime, and FlightNum columns  
flights[, c("DepTime", "ArrTime", "FlightNum")]

# dplyr approach  
select(flights, DepTime, ArrTime, FlightNum)

## # A tibble: 227,496 x 3  
## DepTime ArrTime FlightNum  
## <int> <int> <int>  
## 1 1400 1500 428  
## 2 1401 1501 428  
## 3 1352 1502 428  
## 4 1403 1513 428  
## 5 1405 1507 428  
## 6 1359 1503 428  
## 7 1359 1509 428  
## 8 1355 1454 428  
## 9 1443 1554 428  
## 10 1443 1553 428  
## # ... with 227,486 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:DayofMonth, contains("Taxi"), contains("Delay"))

## # A tibble: 227,496 x 7  
## Year Month DayofMonth TaxiIn TaxiOut ArrDelay DepDelay  
## <int> <int> <int> <int> <int> <int> <int>  
## 1 2011 1 1 7 13 -10 0  
## 2 2011 1 2 6 9 -9 1  
## 3 2011 1 3 5 17 -8 -8  
## 4 2011 1 4 9 22 3 3  
## 5 2011 1 5 9 9 -3 5  
## 6 2011 1 6 6 13 -7 -1  
## 7 2011 1 7 12 15 -1 -1  
## 8 2011 1 8 7 12 -16 -5  
## 9 2011 1 9 8 22 44 43  
## 10 2011 1 10 6 19 43 43  
## # ... with 227,486 more rows

## "Chaining" or "Piping"

* One way to perform multiple operations in one line is by nesting but this can be quite complex to follow
* We can also write commands in a natural order by using the %>% pipe operator (which can be pronounced as "then")

for example

eat food %>% drink beer

eat food THEN drink beer

# nesting method to select UniqueCarrier and DepDelay columns and filter for delays over 60 minutes  
filter(select(flights, UniqueCarrier, DepDelay), DepDelay > 600)

# chaining or piping method  
flights %>%  
 select(UniqueCarrier, DepDelay) %>%  
 filter(DepDelay > 600)

## # A tibble: 13 x 2  
## UniqueCarrier DepDelay  
## <fctr> <int>  
## 1 CO 780  
## 2 MQ 803  
## 3 CO 758  
## 4 UA 869  
## 5 MQ 814  
## 6 CO 981  
## 7 DL 730  
## 8 AA 677  
## 9 MQ 931  
## 10 AA 970  
## 11 AA 653  
## 12 MQ 691  
## 13 XE 628

* 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
* the shortcut key is ctrl/shift M

# create two vectors and calculate Euclidian distance between them. This is the less readable way without chaining  
x1 <- 1:5; x2 <- 2:6  
sqrt(sum((x1-x2)^2))

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

## [1] 2.236068

## arrange: Reorder rows

# base R approach to select UniqueCarrier and DepDelay columns and sort by DepDelay  
flights[order(flights$DepDelay), c("UniqueCarrier", "DepDelay")]

# dplyr approach  
flights %>%  
 select(UniqueCarrier, DepDelay) %>%  
 arrange(DepDelay)

## # A tibble: 227,496 x 2  
## UniqueCarrier DepDelay  
## <fctr> <int>  
## 1 OO -33  
## 2 MQ -23  
## 3 XE -19  
## 4 XE -19  
## 5 CO -18  
## 6 EV -18  
## 7 XE -17  
## 8 CO -17  
## 9 XE -17  
## 10 MQ -17  
## # ... with 227,486 more rows

# use `desc` for descending  
flights %>%  
 select(UniqueCarrier, DepDelay) %>%  
 arrange(desc(DepDelay))

## mutate: Add new variables

* Create new variables that are functions of existing variables

# base R approach to create a new variable Speed (in mph)  
flights$Speed <- flights$Distance / flights$AirTime\*60  
flights[, c("Distance", "AirTime", "Speed")]

# dplyr approach (prints the new variable but does not store it)  
flights %>%  
 select(Distance, AirTime) %>%  
 mutate(Speed = Distance/AirTime\*60)

## # A tibble: 227,496 x 3  
## Distance AirTime Speed  
## <int> <int> <dbl>  
## 1 224 40 336.0000  
## 2 224 45 298.6667  
## 3 224 48 280.0000  
## 4 224 39 344.6154  
## 5 224 44 305.4545  
## 6 224 45 298.6667  
## 7 224 43 312.5581  
## 8 224 40 336.0000  
## 9 224 41 327.8049  
## 10 224 45 298.6667  
## # ... with 227,486 more rows

# store the new variable  
flights <- flights %>% mutate(Speed = Distance/AirTime\*60)  
  
# look at the names of the variables in flights  
  
names(flights) # we can see that "Speed" has no been added to the dataframe at the end

## [1] "X" "Year" "Month"   
## [4] "DayofMonth" "DayOfWeek" "DepTime"   
## [7] "ArrTime" "UniqueCarrier" "FlightNum"   
## [10] "TailNum" "ActualElapsedTime" "AirTime"   
## [13] "ArrDelay" "DepDelay" "Origin"   
## [16] "Dest" "Distance" "TaxiIn"   
## [19] "TaxiOut" "Cancelled" "CancellationCode"   
## [22] "Diverted" "Speed"

## summarise: Reduce variables to values

* Primarily useful with data that has been grouped by one or more variables
* Think of pivot tables in Excel as a comparable idea
* group\_by creates the groups that will be operated on
* summarise uses the provided aggregation function to summarise each group

# base R approaches to calculate the average arrival delay to each destination  
head(with(flights, tapply(ArrDelay, Dest, mean, na.rm=TRUE)))  
head(aggregate(ArrDelay ~ Dest, flights, mean))

# dplyr approach: create a table grouped by Dest, and then summarise each group by taking the mean of ArrDelay  
flights %>%  
 group\_by(Dest) %>%  
 summarise(avg\_delay = mean(ArrDelay, na.rm=TRUE)) ### the na.rm = TRUE tells R to ignore any missing values to avoid an error

## # A tibble: 116 x 2  
## Dest avg\_delay  
## <fctr> <dbl>  
## 1 ABQ 7.226259  
## 2 AEX 5.839437  
## 3 AGS 4.000000  
## 4 AMA 6.840095  
## 5 ANC 26.080645  
## 6 ASE 6.794643  
## 7 ATL 8.233251  
## 8 AUS 7.448718  
## 9 AVL 9.973988  
## 10 BFL -13.198807  
## # ... with 106 more rows

### There are three other variants of summarise

* \_all affects every variable
* \_at affects variables selected with a character vector or vars()
* \_if affects variables selected with a predicate function:

### So for example:

* summarise\_at allows you to apply the same summary function to multiple columns at once
* Note: mutate\_at is also available

# for each carrier, calculate the percentage of flights cancelled or diverted  
flights %>%  
 group\_by(UniqueCarrier) %>%  
 summarise\_at(vars(Cancelled, Diverted), funs(mean))

## # A tibble: 15 x 3  
## UniqueCarrier Cancelled Diverted  
## <fctr> <dbl> <dbl>  
## 1 AA 0.018495684 0.001849568  
## 2 AS 0.000000000 0.002739726  
## 3 B6 0.025899281 0.005755396  
## 4 CO 0.006782614 0.002627370  
## 5 DL 0.015903067 0.003029156  
## 6 EV 0.034482759 0.003176044  
## 7 F9 0.007159905 0.000000000  
## 8 FL 0.009817672 0.003272557  
## 9 MQ 0.029044750 0.001936317  
## 10 OO 0.013946828 0.003486707  
## 11 UA 0.016409266 0.002413127  
## 12 US 0.011268986 0.001469868  
## 13 WN 0.015504047 0.002293629  
## 14 XE 0.015495599 0.003449550  
## 15 YV 0.012658228 0.000000000

#### so the above can be read as " for the dataframe called flights, group by the Unique Carrier id THEN summarise these groups  
# using the variables Cancelled and Diverted using the MEAN function "  
  
# we can also do more than one summary  
  
# for each carrier, calculate the minimum and maximum arrival and departure delays  
flights %>%  
 group\_by(UniqueCarrier) %>%  
 summarise\_at(vars(ArrDelay, DepDelay), funs(min, max), na.rm = TRUE)

## # A tibble: 15 x 5  
## UniqueCarrier ArrDelay\_min DepDelay\_min ArrDelay\_max DepDelay\_max  
## <fctr> <dbl> <dbl> <dbl> <dbl>  
## 1 AA -39 -15 978 970  
## 2 AS -43 -15 183 172  
## 3 B6 -44 -14 335 310  
## 4 CO -55 -18 957 981  
## 5 DL -32 -17 701 730  
## 6 EV -40 -18 469 479  
## 7 F9 -24 -15 277 275  
## 8 FL -30 -14 500 507  
## 9 MQ -38 -23 918 931  
## 10 OO -57 -33 380 360  
## 11 UA -47 -11 861 869  
## 12 US -42 -17 433 425  
## 13 WN -44 -10 499 548  
## 14 XE -70 -19 634 628  
## 15 YV -32 -11 72 54

* Helper function n() counts the number of rows in a group
* Helper function n\_distinct(vector) counts the number of unique items in that vector

# for each day of the year, count the total number of flights and sort in descending order  
flights %>%  
 group\_by(Month, DayofMonth) %>%  
 summarise(flight\_count = n()) %>%  
 arrange(desc(flight\_count))

## # A tibble: 365 x 3  
## # Groups: Month [12]  
## Month DayofMonth flight\_count  
## <int> <int> <int>  
## 1 8 4 706  
## 2 8 11 706  
## 3 8 12 706  
## 4 8 5 705  
## 5 8 3 704  
## 6 8 10 704  
## 7 1 3 702  
## 8 7 7 702  
## 9 7 14 702  
## 10 7 28 701  
## # ... with 355 more rows

# rewrite more simply with the `tally` function  
flights %>%  
 group\_by(Month, DayofMonth) %>%  
 tally(sort = TRUE)

## # A tibble: 365 x 3  
## # Groups: Month [12]  
## Month DayofMonth n  
## <int> <int> <int>  
## 1 8 4 706  
## 2 8 11 706  
## 3 8 12 706  
## 4 8 5 705  
## 5 8 3 704  
## 6 8 10 704  
## 7 1 3 702  
## 8 7 7 702  
## 9 7 14 702  
## 10 7 28 701  
## # ... with 355 more rows

# for each destination, count the total number of flights and the number of distinct planes that flew there  
flights %>%  
 group\_by(Dest) %>%  
 summarise(flight\_count = n(), plane\_count = n\_distinct(TailNum))

## # A tibble: 116 x 3  
## Dest flight\_count plane\_count  
## <fctr> <int> <int>  
## 1 ABQ 2812 716  
## 2 AEX 724 215  
## 3 AGS 1 1  
## 4 AMA 1297 158  
## 5 ANC 125 38  
## 6 ASE 125 60  
## 7 ATL 7886 983  
## 8 AUS 5022 1015  
## 9 AVL 350 142  
## 10 BFL 504 70  
## # ... with 106 more rows

* Grouping can sometimes be useful without summarising

# for each destination, show the number of cancelled and not cancelled flights  
flights %>%  
 group\_by(Dest) %>%  
 select(Dest,Cancelled) %>%  
 table() %>% #### table() is actually a base R function  
 head(20) ### show top 20 rows

## Cancelled  
## Dest 0 1  
## ABQ 2787 25  
## AEX 712 12  
## AGS 1 0  
## AMA 1265 32  
## ANC 125 0  
## ASE 120 5  
## ATL 7745 141  
## AUS 4995 27  
## AVL 347 3  
## BFL 503 1  
## BHM 2697 39  
## BKG 108 2  
## BNA 3451 30  
## BOS 1724 28  
## BPT 3 0  
## BRO 1665 27  
## BTR 1733 29  
## BWI 2527 24  
## CAE 547 14  
## CHS 1191 9