ILLINOIS

Data Wrangling with dplyr

Lecture 14

Dirk Eddelbuettel

STAT 430: Data Science Programming Methods (Fall 2019)

Department of Statistics, University of Illinois

Topics

- filter, select, mutate
- melt/cast
- pipe (magrittr)
- examples
- Chapter 10: Relational Data with dplyr in Wickham and Grolemund, *R for Data Science*, 2017
- Chapter 12: Faster Group Manipulation with dplyr in Lander, *R for Everone*, 2017.

Previous Lesson

- data.table
- `DT[i, j, by]`
- `fread()` / `fwrite()` and `rbindlist()`

Matrices

- Two-dimensional data structure
- All elements must be of the same type
- Indexing by position, name or logical expression:

```
M[1, 4:5]
M[1:10, "rates"]        # if col 'rates' exists
M[ M[,"rates"] > 0, 2:5]
```

- Also:

```
M <- matrix(1:9,3); I <- matrix(c(2,3,1,2),2)
M[ I ]
```

Data.Frame

- Core R Data Structure
- Different column types allowed, must have same length
- Indexing by position, name or logical expression:

```
DF[1, 4:5]
DF[1:10, "rates"]        # if a col 'rates' exists
DF[ DF[,"rates"] > 0, 2:5]
```

- `data.table` and `dplyr` can reference columns unquoted

data.table

- Extends `data.frame`
- Very fast and efficient
  - aggregation
  - (ordered) join
  - add/modify/delete by group
- Focus on
  - *efficiency* and
  - *performance*

DPLYR

Another Powerful and Popular Package

ILLINOIS

- Written by Hadley Wickham and collaborators
- Part of the tidyverse
- Powerful package to transform / summarize tabular data
- 'split-apply-combine' paradigm, cf Wickham (2011,JSS) (which describes earlier package `plyr`)
- Aims for easier and more consistent syntax than base R
- Vignettes, FAQ, ... at `https://cloud.r-project.org/package=dplyr` and `https://dplyr.tidyverse.org/`

(Current) List of Vignettes

- Introduction to dplyr
- Window functions and grouped mutate/filter
- Compatibility
- Programming (and NSE)
- Two-table verbse

See `https://cloud.r-project.org/package=dplyr`

When working with data you must:

- Figure out what you want to do.
- Describe those tasks in the form of a computer program.
- Execute the program.

The `dplyr` package makes these steps fast and easy:

- By constraining your options, it simplifies how you can think about common data manipulation tasks.
- It provides simple "verbs" [...]
- It uses efficient data storage backends [...]

Verbs

- `select()` to select columns (also: `rename()`)
- `filter()` to filter rows (also: `slice()`)
- `arrange()` to re-order or arrange rows
- `mutate()` to create new columns (also: `transmute()`)
- `summarise()` to summarise values
- `group_by()` for group operations (ie split-apply-combine)
- `distinct()`, `sample_n()`, `sample_frac()`

Data set

We use the `flights14` data we use in the last chapter (but this version has six more columns):

```
library(dplyr)
url <- paste0("https://github.com/arunsrinivasan/flights/",
              "wiki/NYCflights14/flights14.csv")
flights <- data.table::fread(url)
tbl <- as.tbl(flights)
```

Examples: select()

ILLINOIS

```
## to select one or more columns
select(tbl, carrier, flight)

## to exclude by name
select(tbl, -cancelled)

## for range from first to last
select(tbl, flight:distance)
```

Columns can be selected programmatically too – expressions like `starts_with("abc")`, `ends_with()`, `contains()`, `matches()`, `one_of()` as well as `distinct()` for unique values are available (and in associated package `tidyselect`).

filter() for row selections:

```
## for 1, 2, ... conditions, and corresponds to
## tbl[tbl$air_time >= 60 & tbl$arr_delay > 20,]
filter(tbl, air_time >= 60, arr_delay > 20)

## can use standard R functions and operators
filter(tbl, carrier %in% c("AA", "UA"))

## using OR
filter(tbl, carrier == "AA" | carrier == "UA")

## using slice() for row position
slice(tbl, 1:10)
```

All previous operations had one thing in common: first argument was a data object such as a `data.frame` or `data_frame` or `tbl` or `tibble`

Instead of nesting as in `head(select(data, colA, colB))` the *pipe operator* `%>%` operates on its *left-hand side argument* as if it were the first argument:

```
data %>% select(colA, colB) %>% head
```

arrange( ) for re-ordering by row(s):

```
## specify row-sort order using pipe
tbl %>% arrange(year, month, day)

## same but function call
arrange(tbl, year, month, day)

## sort descending
arrange(tbl, desc(arr_delay))
```

group_by() for grouping operations

```
## group-wise summaries
tbl %>%
    group_by(carrier) %>%
    summarise(avg_arr_delay = mean(arr_delay),
              avg_dep_delay = mean(dep_delay),
              total = n()) %>%
    arrange(desc(avg_arr_delay))
```

This is similar to **by=** in **data.table** as seen in the previous lecture.

`mutate()` to add new columns

```
## create new column via mutate
tbl %>% mutate(gain = arr_delay - dep_delay,
               speed = distance / air_time * 60)

## allows access to columns just created
tbl %>% mutate(gain = arr_delay - dep_delay,
               gain_per_hour = gain / (air_time / 60)
```

`transmute()` transform by returning only the new columns

Examples: `summarise()` etc

`summarise()` to summarise value

```
tbl %>%
    summarise(delay = mean(dep_delay, na.rm = TRUE))
```

`sample_n()` subsamples fixed size

`sample_frac()` subsamples fraction

```
by_tailnum <- group_by(tbl, tailnum)
delay <- summarise(by_tailnum,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE))
delay <- filter(delay, count > 20, dist < 2000)

library(ggplot2)
ggplot(delay, aes(dist, delay)) +
    geom_point(aes(size = count), alpha = 1/2) +
    geom_smooth() +
    scale_size_area()
```

Grouping and summaries

```
destinations <- group_by(flights, dest)
summarise(destinations,
  planes = n_distinct(tailnum),
  flights = n()
)
```

Comparison (in Hadley's words)

ILLINOIS

Compared to all existing options, dplyr:

- abstracts away data stores: data frames, data tables, DBs
- provides a thoughtful default `print()` method.

Compared to base functions:

- dplyr is more consistent; functions have the same interface.
- base functions (mostly) for vectors; dplyr mostly data frames

Compared to plyr, dplyr:

- is much much faster
- provides a better thought out set of joins
- only provides tools for working with data frames

Yet (also in Hadley's words)

ILLINOIS

> *If you're routinely working with larger data (10-100 Gb, say), you should learn more about data.table. This book doesn't teach data.table because it has a very concise interface which makes it harder to learn since it offers fewer linguistic cues. But if you're working with large data, the performance payoff is worth the extra effort required to learn it.*

Source: From the Introduction to *R for Data Science*

dplyr spawned a number of other packages

- `magrittr` and its `%>%` operator preceded it (but `dplyr` popularised it)
- `tidyr` with `spread()` and `gather()` replacing the `reshape` & `reshape2` packages – now replaced with `pivot_wide()` and `pivot_long()`
- `broom` converts statistical analysis objects into tidy data frames
- `purrr` for a functional programming approach
- and more ...
- and it keeps changing which has its plusses and minuses

Some Resources

- CRAN page
- GitHub repo
- R for Data Science (O'Reilly), source website here
- dplyr tutorial links (2014)

Appendix

More Tutorials

- https://www.listendata.com/2016/08/dplyr-tutorial.html
- http://stat545.com/block009_dplyr-intro.html and http://stat545.com/block010_dplyr-end-single-table.html