MODERN DATA MANIPULATION: DPLYR AND DATA.TABLE

Tobias Niedermaier

DPLYR

BASICS

`dplyr` is an `R` package that implements an advanced version of standard data frames

From the official website:

"dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges"

- Part of the `tidyverse` (package collection)

```
1  library(dplyr)
```

PIPE %>%

The `%>%` is an operator to concatenate function calls. Originally from the `magrittr` package.

A short example will show this merits:

- Take a list of numbers.
- Square each number.
- Sum the squared values.

```
# Without using pipe operator

sum(sapply(list(1, 2, 3, 4),function(x) x^2))
```

```
[1] 30
```

```
# Use pipe
list(1, 2, 3, 4) %>%
  sapply(function(x) x^2) %>%
  sum()
```

```
[1] 30
```

There is a keyboard shortcut in Rstudio to insert a pipe operator. On my RStudio, it is `Ctrl + Shift + m`

The pipe was so popular, that there is an official implementation in base R (`|>`). However, I prefer `%>%` for reasons…

TIBBLE

TIBBLE

Here, we use the `starwars` data set from the `dplyr` package.

```
  1  head(starwars)
```

```
# A tibble: 6 × 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr> <chr>
1 Luke Sky…    172    77 blond      fair       blue             19   male  mascu…
2 C-3PO        167    75 <NA>       gold       yellow          112   none  mascu…
3 R2-D2         96    32 <NA>       white, bl… red              33   none  mascu…
4 Darth Va…    202   136 none       white      yellow           41.9 male  mascu…
5 Leia Org…    150    49 brown      light      brown            19   fema… femin…
6 Owen Lars    178   120 brown, gr… light      blue             52   male  mascu…
# ℹ 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- Data class is already `tibble`
- Slightly more informative than standard data frame
- We can define a tibble by using the according function, e.g. `tibble(iris)`
- A tibble behaves like a standard data frame (e.g. it is still list-like)

BASIC OPERATIONS

`tibble` is designed to work smoothly with the pipe

- Select rows and columns
  - We can use `filter` to select rows with defined conditions
  - We can use `select` to select columns

```
 1  starwars %>%
 2    filter(height > 170, mass < 130) %>%
 3    select(name, homeworld)
```

```
# A tibble: 37 × 2
   name               homeworld
   <chr>              <chr>
 1 Luke Skywalker     Tatooine
 2 Owen Lars          Tatooine
 3 Biggs Darklighter  Tatooine
 4 Obi-Wan Kenobi     Stewjon
 5 Anakin Skywalker   Tatooine
 6 Chewbacca          Kashyyyk
 7 Han Solo           Corellia
 8 Greedo             Rodia
 9 Jek Tono Porkins   Bestine IV
10 Boba Fett          Kamino
```

```
# ℹ 27 more rows
```

- Note that we can call variables from the data set without $

BASIC OPERATIONS

Define new variables

- We can use `mutate` to define new variables

```
starwars %>%
  mutate(bmi = mass / (height/100)^2)
```

```
# A tibble: 87 × 15
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr> <chr>
 1 Luke Sk…    172    77 blond      fair       blue             19   male  mascu…
 2 C-3PO       167    75 <NA>       gold       yellow          112   none  mascu…
 3 R2-D2        96    32 <NA>       white, bl… red              33   none  mascu…
 4 Darth V…    202   136 none       white      yellow           41.9 male  mascu…
 5 Leia Or…    150    49 brown      light      brown            19   fema… femin…
 6 Owen La…    178   120 brown, gr… light      blue             52   male  mascu…
 7 Beru Wh…    165    75 brown      light      blue             47   fema… femin…
 8 R5-D4        97    32 <NA>       white, red red              NA   none  mascu…
 9 Biggs D…    183    84 black      light      brown            24   male  mascu…
10 Obi-Wan…    182    77 auburn, w… fair       blue-gray        57   male  mascu…
# ℹ 77 more rows
# ℹ 6 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>, bmi <dbl>
```

- Note that we returned the data here!
- Hence, no *inplace* operation. Use `<-` or `%<>%` to for an assigment

GROUP AND SUMMARY OPERATIONS

- Use `group_by` to make a operations on subgroups
- Use `summarise` to summarize variables
- Sort results using `arrange` and possibly `desc`

Question: What is the mean height and weight for each species? Sort the result by weight in descending order.

```
starwars %>%
  group_by(species) %>%
  summarise(
    mean_height = mean(height, na.rm = TRUE),
    mean_weight = mean(mass, na.rm = TRUE),
  ) %>%
  arrange(desc(mean_weight))
```

```
# A tibble: 38 × 3
   species      mean_height mean_weight
   <chr>              <dbl>       <dbl>
 1 Hutt                 175        1358
 2 Kaleesh              216         159
 3 Wookiee              231         124
 4 Trandoshan           190         113
```

```
 5 Besalisk             198         102
 6 Neimodian            191          90
 7 Kaminoan             221          88
 8 Nautolan             196          87
 9 Mon Calamari         180          83
10 Cerean               198          82
# ℹ 28 more rows
```

A LOT MORE...

We can only scratch on the surface. See the webpage for more examples:

https://dplyr.tidyverse.org/index.html

DATA.TABLE

DATA. TABLE

- `data.table` is used for large data sets and is designed to be fast and memory efficient.
- Like a `tibble`, it is a 'list-like' object.
- We stick to the starwars data set:

```
1  library(data.table)
2
3  sw_dt <- starwars
4  setDT(sw_dt)
5  head(sw_dt)
```

```
            name height  mass  hair_color  skin_color eye_color birth_year
          <char>  <int> <num>      <char>      <char>    <char>      <num>
1: Luke Skywalker    172    77       blond        fair      blue       19.0
2:          C-3PO    167    75        <NA>        gold    yellow      112.0
3:          R2-D2     96    32        <NA> white, blue       red       33.0
4:    Darth Vader    202   136        none       white    yellow       41.9
5:    Leia Organa    150    49       brown       light     brown       19.0
6:      Owen Lars    178   120 brown, grey       light      blue       52.0
      sex    gender homeworld species
   <char>    <char>    <char>  <char>
1:   male masculine  Tatooine   Human
2:   none masculine  Tatooine   Droid
3:   none masculine     Naboo   Droid
4:   male masculine  Tatooine   Human
5: female  feminine  Alderaan   Human
6:   male masculine  Tatooine   Human
```

GENERAL SYNTAX

data.table uses as basic syntac

```
1  DT[i, j, by]
```

with an analogy to SQL:

```
1  DT[where | order by,  select | update ,  group by]
```

- `i` to filter rows or order
- `j` to select columns OR create new ones
- `by` to do it for subgroups

> **Warning**
>
> Wrap `j` in a `list()` or its *alias* in DT `.()` to ensure a data table object! → see exercise.

EXAMPLE FROM BEFORE

dplyr

```
1  starwars %>%
2    filter(height > 170, mass < 130) %>%
3    select(name, homeworld)
```

data.table

```
1  sw_dt[height > 170 & mass < 130, .(name, homeworld)]
```

```
                name      homeworld
              <char>         <char>
 1:    Luke Skywalker       Tatooine
 2:         Owen Lars       Tatooine
 3: Biggs Darklighter       Tatooine
 4:    Obi-Wan Kenobi        Stewjon
 5:  Anakin Skywalker       Tatooine
 6:         Chewbacca       Kashyyyk
 7:          Han Solo       Corellia
 8:            Greedo          Rodia
 9:  Jek Tono Porkins     Bestine IV
10:         Boba Fett         Kamino
11:             Bossk      Trandosha
12:  Lando Calrissian        Socorro
13:             Lobot         Bespin
14:            Ackbar       Mon Cala
15:     Qui-Gon Jinn            <NA>
```

SPECIAL ARGUMENT .N

- data.table uses `.N` to count rows (in a group)

```
starwars[species == "Human", .N]  # 35 humans in the data set
```

```
[1] 35
```

- we can combine it with `by` to count the number in each group

```
head(starwars[,.N, by = species])
```

```
   species     N
    <char> <int>
1:   Human    35
2:   Droid     6
3: Wookiee     2
4:  Rodian     1
5:    Hutt     1
6:    <NA>     4
```

MORE COMPLEX EXAMPLE FROM BEFORE

```
starwars %>%
  group_by(species) %>%
  summarise(
    mean_height = mean(height, na.rm = TRUE),
    mean_weight = mean(mass, na.rm = TRUE),
  ) %>%
  arrange(desc(mean_weight))
```

```
# A tibble: 38 × 3
   species      mean_height mean_weight
   <chr>              <dbl>       <dbl>
 1 Hutt                 175        1358
 2 Kaleesh              216         159
 3 Wookiee              231         124
 4 Trandoshan           190         113
 5 Besalisk             198         102
 6 Neimodian            191          90
 7 Kaminoan             221          88
 8 Nautolan             196          87
 9 Mon Calamari         180          83
10 Cerean               198          82
# ℹ 28 more rows
```

- Giving the list in `j` names to return a data.table with according names.
- We use the fact that the returned object is a data table

```
sw_dt[,.(
    mean_height = mean(height, na.rm = TRUE),
    mean_weight = mean(mass, na.rm = TRUE)
), by = species][order(-mean_weight),]
```

```
          species mean_height mean_weight
           <char>       <num>       <num>
 1:          Hutt    175.0000     1358.00
 2:       Kaleesh    216.0000      159.00
 3:       Wookiee    231.0000      124.00
 4:    Trandoshan    190.0000      113.00
 5:       Besalisk    198.0000      102.00
 6:     Neimodian    191.0000       90.00
 7:       Kaminoan    221.0000       88.00
 8:       Nautolan    196.0000       87.00
 9:  Mon Calamari    180.0000       83.00
10:         Cerean    198.0000       82.00
11:          Human    178.0000       81.31
12:           <NA>    175.0000       81.00
13:         Zabrak    173.0000       80.00
14:       Kel Dor    188.0000       80.00
15:     Geonosian    183.0000       80.00
```

EXAMPLE WITH MULTIPLE ARGUMENTS IN GROUP

We want to find out what the max and min value of height for each subgroup of species and gender is. We also want to know, how many data points were used to calculate the result:

```
sw_dt[,.(minimum = min(height, na.rm = T),
         maximum = (max(height, na.rm = T)),
         .N),   # no name used for .N!
       by = .(species, gender)]
```

```
          species    gender minimum maximum     N
           <char>    <char>   <int>   <int> <int>
 1:          Human masculine     170     202    26
 2:          Droid masculine      96     200     5
 3:          Human  feminine     150     185     9
 4:        Wookiee masculine     228     234     2
 5:         Rodian masculine     173     173     1
 6:           Hutt masculine     175     175     1
 7:           <NA>      <NA>     157     185     4
 8: Yoda's species masculine      66      66     1
 9:     Trandoshan masculine     190     190     1
10:   Mon Calamari masculine     180     180     1
11:           Ewok masculine      88      88     1
12:      Sullustan masculine     160     160     1
13:      Neimodian masculine     191     191     1
14:         Gungan masculine     196     224     3
15:      Toydarian masculine     137     137     1
```

NEW VARIABLES

- When we define new variables in base data sets, this invokes a deep copy (very inefficient)
- data.table relies on *reference* using a custom operator `:=`

We create a new variable `bmi`

```
1  sw_dt[, bmi := mass / (height/100)^2]
```

This is a lot more efficient as we fo not make a copy of the full data set.

DELETE VARIABLES

- We can delete variables from the data table by assigning a `NULL` to it:

```
1  sw_dt[, vehicles := NULL]
```

NEW VARIABLES CONT'D

`:=` can also be used…

- to assign multiple variables
- in combination with selection of rows
- in combination with `by` and `.N`

Here we make an example where we calculate the mean bmi (by hand) and its contribution (in %) to the common weight of all individuals from one species.

- We don't create a new data table, but add the information to the original one:

```
1  sw_dt[, c("mean_bmi", "mass_contrib") := .(sum(mass/(height/100)^2, na.rm=T)/.N,
2                                             mass/sum(mass, na.rm = T)*100),
3         by = species]
4
5  # look at the data set
```

```
6  sw_dt[, .(name, species, bmi, mean_bmi, mass_contrib)]
```

	name	species	bmi	mean_bmi	mass_contrib
	<char>	<char>	<num>	<num>	<num>
1:	Luke Skywalker	Human	26.02758	14.243690	4.734965
2:	C-3PO	Droid	26.89232	21.770756	26.881720
3:	R2-D2	Droid	34.72222	21.770756	11.469534
4:	Darth Vader	Human	33.33007	14.243690	8.363055
5:	Leia Organa	Human	21.77778	14.243690	3.013160
6:	Owen Lars	Human	37.87401	14.243690	7.379166
7:	Beru Whitesun Lars	Human	27.54821	14.243690	4.611979
8:	R5-D4	Droid	34.00999	21.770756	11.469534
9:	Biggs Darklighter	Human	25.08286	14.243690	5.165416
10:	Obi-Wan Kenobi	Human	23.24598	14.243690	4.734965
11:	Anakin Skywalker	Human	23.76641	14.243690	5.165416
12:	Wilhuff Tarkin	Human	NA	14.243690	NA
13:	Chewbacca	Wookiee	21.54509	23.191276	45.161290
14:	Han Solo	Human	24.69136	14.243690	4.919444
15:	Greedo	Rodian	24.72518	24.725183	100.000000

Note that the mean bmi here is so small because be calculated it by hand using `/.N`! So we ignore NAs here…It is rather an example to show a potential use for `.N`.