Dplyr Package

Srijan Kundu

2022-11-07

> `Dplyr` is a grammar of data manipulation, providing a consistent set of verbs that help users solve the most common data manipulation challenges in R, and according to its official documentation, is "a fast, consistent tool for working with data frame like objects, both in memory and out of memory."

A list of functions under the `dplyr` package can be accessed from here, although the focus here is to talk about some of the more frequently used ones among those, viz.,

SI. No.	`dplyr` function	Description
01	`filter()`	Subset rows using column values
02	`arrange()`	Orders the rows of a data frame by the values of selected columns
03	`select()`	Subset columns using their names and types
04	`rename()`	Rename columns
05	`slice()`	Subset rows using their positions
06	`mutate()`	Adds new variables and preserves existing ones
07	`transmute()`	Adds new variables and drops existing ones
08	`summarise()`	Summarise each group to fewer rows
09	`group_by()`	Takes an existing tbl and converts it into a grouped tbl where operations are performed "by group"

First, let's load the `dplyr` package.

```
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
```

We will load a dataset and demonstrate each of the functions mentioned above using that dataset. A good example is the "starwars dataset" which comes with the `dplyr` library itself. The dataset contains 87 rows and 14 variables.

```
starwars
```

```
## # A tibble: 87 x 14
##    name        height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##    <chr>        <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
##  1 Luke Skywa~    172    77 blond   fair    blue       19   male  mascu~ Tatooi~
##  2 C-3PO          167    75 <NA>    gold    yellow    112   none  mascu~ Tatooi~
##  3 R2-D2           96    32 <NA>    white,~ red        33   none  mascu~ Naboo
##  4 Darth Vader    202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
##  5 Leia Organa    150    49 brown   light   brown      19   fema~ femin~ Aldera~
##  6 Owen Lars      178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
##  7 Beru White~    165    75 brown   light   blue       47   fema~ femin~ Tatooi~
##  8 R5-D4           97    32 <NA>    white,~ red        NA   none  mascu~ Tatooi~
##  9 Biggs Dark~    183    84 black   light   brown      24   male  mascu~ Tatooi~
## 10 Obi-Wan Ke~    182    77 auburn~ fair    blue-g~    57   male  mascu~ Stewjon
## # ... with 77 more rows, 4 more variables: species <chr>, films <list>,
## #   vehicles <list>, starships <list>, and abbreviated variable names
## #   1: hair_color, 2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

```
head(starwars)
```

```
## # A tibble: 6 x 14
##   name         height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>         <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
## 1 Luke Skywal~    172    77 blond   fair    blue       19   male  mascu~ Tatooi~
## 2 C-3PO           167    75 <NA>    gold    yellow    112   none  mascu~ Tatooi~
## 3 R2-D2            96    32 <NA>    white,~ red        33   none  mascu~ Naboo
## 4 Darth Vader     202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
## 5 Leia Organa     150    49 brown   light   brown      19   fema~ femin~ Aldera~
## 6 Owen Lars       178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `colnames()` to see all variable names
```

```
dim(starwars)
```

```
## [1] 87 14
```

The `filter()` function:

`filter()` allows us select a subset of rows in a data frame. Like all single verbs, the first argument is the tibble (or data frame). The second and subsequent arguments refer to variables within that data frame, selecting rows where the expression is `TRUE`.

```
# Filtering by one criterion
filter(starwars, species == "Human")
```

```
## # A tibble: 35 x 14
##    name        height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##    <chr>        <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
##  1 Luke Skywa~    172    77 blond   fair    blue       19   male  mascu~ Tatooi~
##  2 Darth Vader    202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
##  3 Leia Organa    150    49 brown   light   brown      19   fema~ femin~ Aldera~
##  4 Owen Lars      178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
##  5 Beru White~    165    75 brown   light   blue       47   fema~ femin~ Tatooi~
##  6 Biggs Dark~    183    84 black   light   brown      24   male  mascu~ Tatooi~
##  7 Obi-Wan Ke~    182    77 auburn~ fair    blue-g~    57   male  mascu~ Stewjon
```

```
##  8 Anakin Sky~    188    84 blond   fair    blue       41.9 male  mascu~ Tatooi~
##  9 Wilhuff Ta~    180    NA auburn~ fair    blue       64   male  mascu~ Eriadu
## 10 Han Solo       180    80 brown   fair    brown      29   male  mascu~ Corell~
## # ... with 25 more rows, 4 more variables: species <chr>, films <list>,
## #   vehicles <list>, starships <list>, and abbreviated variable names
## #   1: hair_color, 2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

```
# Filtering by multiple criteria within a single logical expression
filter(starwars, hair_color == "none" & eye_color == "black")
```

```
## # A tibble: 9 x 14
##   name       height  mass hair_co~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>       <int> <dbl> <chr>     <chr>   <chr>     <dbl> <chr> <chr>  <chr>
## 1 Nien Nunb     160    68 none      grey    black        NA male  mascu~ Sullust
## 2 Gasgano       122    NA none      white,~ black        NA male  mascu~ Troiken
## 3 Kit Fisto     196    87 none      green   black        NA male  mascu~ Glee A~
## 4 Plo Koon      188    80 none      orange  black        22 male  mascu~ Dorin
## 5 Lama Su       229    88 none      grey    black        NA male  mascu~ Kamino
## 6 Taun We       213    NA none      grey    black        NA fema~ femin~ Kamino
## 7 Shaak Ti      178    57 none      red, b~ black        NA fema~ femin~ Shili
## 8 Tion Medon    206    80 none      grey    black        NA male  mascu~ Utapau
## 9 BB8            NA    NA none      none    black        NA none  mascu~ <NA>
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `colnames()` to see all variable names
```

```
# Filtering entries with missing values in hair_color column
filter(starwars, is.na(hair_color))
```

```
## # A tibble: 5 x 14
##   name         height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>         <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
## 1 C-3PO           167    75 <NA>    gold    yellow      112 none  mascu~ Tatooi~
## 2 R2-D2            96    32 <NA>    white,~ red          33 none  mascu~ Naboo
## 3 R5-D4            97    32 <NA>    white,~ red          NA none  mascu~ Tatooi~
## 4 Greedo          173    74 <NA>    green   black        44 male  mascu~ Rodia
## 5 Jabba Desil~    175  1358 <NA>    green-~ orange      600 herm~ mascu~ Nal Hu~
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `colnames()` to see all variable names
```

The `arrange()` function:

`arrange()` works similarly to `filter()` except that instead of filtering or selecting rows, it reorders them. It takes a data frame, and a set of column names (or more complicated expressions) to order by. If more than one column name is provided, each additional column will be used to break ties in the values of preceding columns.

```
#Arrange entries by increasing mass
arrange(starwars, mass)
```

```
## # A tibble: 87 x 14
##    name        height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##    <chr>        <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
```

```
##  1 Ratts Tyer~     79    15 none    grey, ~ unknown      NA male  mascu~ Aleen ~
##  2 Yoda            66    17 white   green   brown       896 male  mascu~ <NA>
##  3 Wicket Sys~     88    20 brown   brown   brown         8 male  mascu~ Endor
##  4 R2-D2           96    32 <NA>    white,~ red          33 none  mascu~ Naboo
##  5 R5-D4           97    32 <NA>    white,~ red          NA none  mascu~ Tatooi~
##  6 Sebulba        112    40 none    grey, ~ orange       NA male  mascu~ Malast~
##  7 Dud Bolt        94    45 none    blue, ~ yellow       NA male  mascu~ Vulpter
##  8 Padmé Amid~    165    45 brown   light   brown        46 fema~ femin~ Naboo
##  9 Wat Tambor     193    48 none    green,~ unknown      NA male  mascu~ Skako
## 10 Sly Moore      178    48 none    pale    white        NA <NA>  <NA>   Umbara
## # ... with 77 more rows, 4 more variables: species <chr>, films <list>,
## #   vehicles <list>, starships <list>, and abbreviated variable names
## #   1: hair_color, 2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

The `select()` function:

Often we have to work with large datasets with many columns but only a few are actually of interest at the time. `select()` allows us rapidly zoom in on a useful subset using operations that usually only work on numeric variable positions.

```
#Select variables by name:
select(starwars, height)
```

```
## # A tibble: 87 x 1
##    height
##     <int>
##  1    172
##  2    167
##  3     96
##  4    202
##  5    150
##  6    178
##  7    165
##  8     97
##  9    183
## 10    182
## # ... with 77 more rows
## # i Use `print(n = ...)` to see more rows
```

```
#Select multiple variables by separating them with commas:
select(starwars, homeworld, height, mass)
```

```
## # A tibble: 87 x 3
##    homeworld height  mass
##    <chr>      <int> <dbl>
##  1 Tatooine     172    77
##  2 Tatooine     167    75
##  3 Naboo         96    32
##  4 Tatooine     202   136
##  5 Alderaan     150    49
##  6 Tatooine     178   120
##  7 Tatooine     165    75
##  8 Tatooine      97    32
##  9 Tatooine     183    84
## 10 Stewjon      182    77
```

```
## # ... with 77 more rows
## # i Use `print(n = ...)` to see more rows
```

The `rename()` function:

Variables can be renamed with `select()` by using named arguments:

```
select(starwars, home_world = homeworld)
```

```
## # A tibble: 87 x 1
##    home_world
##    <chr>
##  1 Tatooine
##  2 Tatooine
##  3 Naboo
##  4 Tatooine
##  5 Alderaan
##  6 Tatooine
##  7 Tatooine
##  8 Tatooine
##  9 Tatooine
## 10 Stewjon
## # ... with 77 more rows
## # i Use `print(n = ...)` to see more rows
```

But because `select()` drops all the variables not explicitly mentioned, it's not that useful. Instead, `rename()` can be used.

```
#Changes the names of individual variables using new_name = old_name syntax
rename(starwars, movies = films)
```

```
## # A tibble: 87 x 14
##    name        height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##    <chr>        <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
##  1 Luke Skywa~    172    77 blond   fair    blue       19   male  mascu~ Tatooi~
##  2 C-3PO          167    75 <NA>    gold    yellow    112   none  mascu~ Tatooi~
##  3 R2-D2           96    32 <NA>    white,~ red        33   none  mascu~ Naboo
##  4 Darth Vader    202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
##  5 Leia Organa    150    49 brown   light   brown      19   fema~ femin~ Aldera~
##  6 Owen Lars      178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
##  7 Beru White~    165    75 brown   light   blue       47   fema~ femin~ Tatooi~
##  8 R5-D4           97    32 <NA>    white,~ red        NA   none  mascu~ Tatooi~
##  9 Biggs Dark~    183    84 black   light   brown      24   male  mascu~ Tatooi~
## 10 Obi-Wan Ke~    182    77 auburn~ fair    blue-g~    57   male  mascu~ Stewjon
## # ... with 77 more rows, 4 more variables: species <chr>, movies <list>,
## #   vehicles <list>, starships <list>, and abbreviated variable names
## #   1: hair_color, 2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

The `slice()` function:

It chooses rows based on location.

```
slice(starwars, 1:7)
```

```
## # A tibble: 7 x 14
##   name         height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>         <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
```

```
## 1 Luke Skywal~    172    77 blond   fair    blue       19   male  mascu~ Tatooi~
## 2 C-3PO           167    75 <NA>    gold    yellow    112   none  mascu~ Tatooi~
## 3 R2-D2            96    32 <NA>    white,~ red        33   none  mascu~ Naboo
## 4 Darth Vader     202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
## 5 Leia Organa     150    49 brown   light   brown      19   fema~ femin~ Aldera~
## 6 Owen Lars       178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
## 7 Beru Whites~    165    75 brown   light   blue       47   fema~ femin~ Tatooi~
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `colnames()` to see all variable names
```

```
#Slice a random sample from the dataset
slice_sample(starwars, n = 5)
```

```
## # A tibble: 5 x 14
##   name        height  mass hair_c~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>        <int> <dbl> <chr>    <chr>   <chr>     <dbl> <chr> <chr>  <chr>
## 1 R5-D4           97    32 <NA>     white,~ red          NA none  mascu~ Tatooi~
## 2 Lama Su        229    88 none     grey    black        NA male  mascu~ Kamino
## 3 Ayla Secura    178    55 none     blue    hazel        48 fema~ femin~ Ryloth
## 4 Yoda            66    17 white    green   brown       896 male  mascu~ <NA>
## 5 Darth Maul     175    80 none     red     yellow       54 male  mascu~ Dathom~
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `colnames()` to see all variable names
```

```
slice_sample(starwars, prop = 0.1)
```

```
## # A tibble: 8 x 14
##   name         height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>         <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
## 1 Nute Gunray     191    90 none    mottle~ red          NA male  mascu~ Cato N~
## 2 Palpatine       170    75 grey    pale    yellow       82 male  mascu~ Naboo
## 3 Barriss Off~    166    50 black   yellow  blue         40 fema~ femin~ Mirial
## 4 Grievous        216   159 none    brown,~ green,~      NA male  mascu~ Kalee
## 5 Wat Tambor      193    48 none    green,~ unknown      NA male  mascu~ Skako
## 6 Luke Skywal~    172    77 blond   fair    blue         19 male  mascu~ Tatooi~
## 7 Roos Tarpals    224    82 none    grey    orange       NA male  mascu~ Naboo
## 8 Mon Mothma      150    NA auburn  fair    blue         48 fema~ femin~ Chandr~
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
## # i Use `colnames()` to see all variable names
```

The `mutate()` function:

Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns. This is the job of `mutate()`.

```
#Add a new column
mutate(starwars, height_m = height / 100)
```

```
## # A tibble: 87 x 15
##    name        height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##    <chr>        <int> <dbl> <chr>   <chr>   <chr>     <dbl> <chr> <chr>  <chr>
```

```
##  1 Luke Skywa~    172    77 blond   fair    blue       19   male  mascu~ Tatooi~
##  2 C-3PO          167    75 <NA>    gold    yellow    112   none  mascu~ Tatooi~
##  3 R2-D2           96    32 <NA>    white,~ red        33   none  mascu~ Naboo
##  4 Darth Vader    202   136 none    white   yellow     41.9 male  mascu~ Tatooi~
##  5 Leia Organa    150    49 brown   light   brown      19   fema~ femin~ Aldera~
##  6 Owen Lars      178   120 brown,~ light   blue       52   male  mascu~ Tatooi~
##  7 Beru White~    165    75 brown   light   blue       47   fema~ femin~ Tatooi~
##  8 R5-D4           97    32 <NA>    white,~ red        NA   none  mascu~ Tatooi~
##  9 Biggs Dark~    183    84 black   light   brown      24   male  mascu~ Tatooi~
## 10 Obi-Wan Ke~    182    77 auburn~ fair    blue-g~    57   male  mascu~ Stewjon
## # ... with 77 more rows, 5 more variables: species <chr>, films <list>,
## #   vehicles <list>, starships <list>, height_m <dbl>, and abbreviated variable
## #   names 1: hair_color, 2: skin_color, 3: eye_color, 4: birth_year,
## #   5: homeworld
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

The `transmute()` funciton:

If you only want to keep the new variables, use `transmute()`.

```
#To keep the new variables
transmute(starwars,
    height_m = height / 100,
    BMI = mass / (height_m^2)
  )
```

```
## # A tibble: 87 x 2
##    height_m   BMI
##       <dbl> <dbl>
##  1     1.72  26.0
##  2     1.67  26.9
##  3     0.96  34.7
##  4     2.02  33.3
##  5     1.5   21.8
##  6     1.78  37.9
##  7     1.65  27.5
##  8     0.97  34.0
##  9     1.83  25.1
## 10     1.82  23.2
## # ... with 77 more rows
## # i Use `print(n = ...)` to see more rows
```

The `summarise()` function:

It collapses a data frame to a single row.

```
summarise(starwars, height = mean(height, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   height
##    <dbl>
## 1   174.
```

The `group_by()` function:

The dplyr API is functional in the sense that function calls don't have side-effects. One must always save their results. This doesn't lead to particularly elegant code, especially if they want to do many operations at

once.

```
a1 <- group_by(starwars, species, sex)
a2 <- select(a1, height, mass)
```

```
## Adding missing grouping variables: `species`, `sex`
```

```
a3 <- summarise(a2,
  height = mean(height, na.rm = TRUE),
  mass = mean(mass, na.rm = TRUE)
)
```

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

```
a3
```

```
## # A tibble: 41 x 4
## # Groups:   species [38]
##    species   sex    height  mass
##    <chr>     <chr>   <dbl> <dbl>
##  1 Aleena    male      79   15
##  2 Besalisk  male     198  102
##  3 Cerean    male     198   82
##  4 Chagrian  male     196  NaN
##  5 Clawdite  female   168   55
##  6 Droid     none     131.  69.8
##  7 Dug       male     112   40
##  8 Ewok      male      88   20
##  9 Geonosian male     183   80
## 10 Gungan    male     209.  74
## # ... with 31 more rows
## # i Use `print(n = ...)` to see more rows
```

This is difficult to read because the order of the operations is from inside to out. Thus, the arguments are a long way away from the function. To get around this problem, dplyr provides the `%\>%` operator from magrittr. x `%\>%` f(y) turns into f(x, y) so it can be used to rewrite multiple operations that one can read left-to-right, top-to-bottom (reading the pipe operator as "then"):

```
starwars %>%
  group_by(species, sex) %>%
  select(height, mass) %>%
  summarise(
    height = mean(height, na.rm = TRUE),
    mass = mean(mass, na.rm = TRUE)
  )
```

```
## Adding missing grouping variables: `species`, `sex`
## `summarise()` has grouped output by 'species'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 41 x 4
## # Groups:   species [38]
##    species   sex    height  mass
##    <chr>     <chr>   <dbl> <dbl>
##  1 Aleena    male      79   15
##  2 Besalisk  male     198  102
##  3 Cerean    male     198   82
```

```
##  4 Chagrian  male     196  NaN
##  5 Clawdite  female   168   55
##  6 Droid     none     131.  69.8
##  7 Dug       male     112   40
##  8 Ewok      male      88   20
##  9 Geonosian male     183   80
## 10 Gungan    male     209.  74
## # ... with 31 more rows
## # i Use `print(n = ...)` to see more rows
```