Data manipulation with dplyr

Matthew Castelo

June, 2021

The **dplyr** package in the **Tidyverse** is an alternative grammar to base R for data manipulation. For those coming from SAS, it may be more intuitive, especially when combined with the pipe (%>%).

The main verbs or functions include:

```
select() #include or exclude columns/variables
filter() #include or exclude rows/observations

mutate() #alter/add variables
summarize() #reduce datasets to summary statistics
rename() #change column/variable names

arrange() #change the order of the dataset
group_by() #perform operations by groups
```

Many of these functions are analogous to those in other statistical software. They will be briefly introduced with examples.

We will need to install the following packages.

```
install.packages("tidyverse")
```

As usual, we will load our packages and set the working directory.

```
library(tidyverse)

#The working directory saves the following path when loading files, saving plots, etc
setwd("C:/Users/matth/Documents/R/R code and education/Tutorials/")
```

We will be working with a simulated dataset representing 80 generic cancer patients. The dataset can be loaded directly from GitHub so all the subsequent code will run on your computer.

```
urlRemote  <- "https://raw.githubusercontent.com/"
pathGithub <- "mcas-surg/Tutorials/main/Datasets/"
fileName   <- "generic_cancer.csv"

cancer <- read_csv(paste0(urlRemote, pathGithub, fileName))
```

The first few rows and data structure can be seen.

```
head(cancer)
```

```
## # A tibble: 6 x 13
##   male   pt_id   age high_grade adverse_tumour_marker tumour_size diabetes
##   <chr>  <dbl> <dbl> <chr>      <chr>                        <dbl> <chr>
## 1 Male       1    62 No         No                            0.41 No
## 2 Male       2    57 No         No                            2.84 No
## 3 Female     3    66 No         No                            3.35 No
## 4 Male       4    54 No         No                            4.63 No
## 5 Male       5    49 No         No                            1.29 No
## 6 Male       6    63 No         Yes                           1.99 No
## # ... with 6 more variables: heart_disease <chr>, hospital <chr>,
## #   extended_resection <chr>, postop_complication <chr>, vital_status <chr>,
## #   follow_up <dbl>
```

The `select()` verb

`select()` will include or exclude columns/variables. If we only want the *male* variable.

```
cancer %>%
  select(male)
```

```
## # A tibble: 80 x 1
##    male
##    <chr>
##  1 Male
##  2 Male
##  3 Female
##  4 Male
##  5 Male
##  6 Male
##  7 Male
##  8 Female
##  9 Male
## 10 Male
## # ... with 70 more rows
```

Note this simply prints the resulting dataset. If we want to save this selection to a new dataset, we should pass a new dataset name.

```
cancer_only_sex <- cancer %>%
  select(male)
```

If we would like to keep only the *male* and *age* variables, multiple columns can be given to `select()`.

```
cancer %>%
  select(male, age)
```

```
## # A tibble: 80 x 2
##    male     age
```

```
##    <chr>  <dbl>
##  1 Male      62
##  2 Male      57
##  3 Female    66
##  4 Male      54
##  5 Male      49
##  6 Male      63
##  7 Male      54
##  8 Female    72
##  9 Male      57
## 10 Male      65
## # ... with 70 more rows
```

We can also ask to keep a range of variables.

```
cancer %>%
  select(male:high_grade)
```

```
## # A tibble: 80 x 4
##    male   pt_id   age high_grade
##    <chr>  <dbl> <dbl> <chr>
##  1 Male       1    62 No
##  2 Male       2    57 No
##  3 Female     3    66 No
##  4 Male       4    54 No
##  5 Male       5    49 No
##  6 Male       6    63 No
##  7 Male       7    54 No
##  8 Female     8    72 No
##  9 Male       9    57 No
## 10 Male      10    65 Yes
## # ... with 70 more rows
```

Removing columns works in much the same way, except using a - sign.

```
cancer %>%
  select(-male, -age, -hospital)

#Instead of putting a "-" sign in front of each variable, we can use the following notation

cancer %>%
  select(-c(male, age, hospital))
```

It is often convenient to select based on the common naming of variables. Say we have a dataset of bloodwork with the following form.

```
## # A tibble: 10 x 8
##    pt_id cbc_hb cbc_leuk cbc_plt lft_alt lft_bili_tot lft_bili_dir vbg_ph
##    <int>  <dbl>    <dbl>   <dbl>   <dbl>        <dbl>        <dbl>  <dbl>
##  1     1    147        8     235      43            7            4    6.6
##  2     2    155        4     205      54           10            5    6.6
##  3     3    139       13     206      49            7            6    7
```

```
##  4     4    132        6     248      44            7            3    6.7
##  5     5    136        8     195      54            8            6    6.9
##  6     6    156       12     238      44            8            7    6.9
##  7     7    121        6     157      49            7            5    6.5
##  8     8    143        5     151      41            7            3    6.9
##  9     9    168        9     204      45           10            6    6.7
## 10    10    151        5     189      42            9            6    7.1
```

Notice the values for hemoglobin, white blood cell count, and platelet all begin with "cbc". We can select just those using `starts_with()`.

```
blood_example %>%
  select(starts_with("cbc"))
```

```
## # A tibble: 10 x 3
##    cbc_hb cbc_leuk cbc_plt
##     <dbl>    <dbl>   <dbl>
##  1    147        8     235
##  2    155        4     205
##  3    139       13     206
##  4    132        6     248
##  5    136        8     195
##  6    156       12     238
##  7    121        6     157
##  8    143        5     151
##  9    168        9     204
## 10    151        5     189
```

If we want to keep the *pt_id* variable as well.

```
blood_example %>%
  select(pt_id,
         starts_with("cbc"))
```

```
## # A tibble: 10 x 4
##    pt_id cbc_hb cbc_leuk cbc_plt
##    <int>  <dbl>    <dbl>   <dbl>
##  1     1    147        8     235
##  2     2    155        4     205
##  3     3    139       13     206
##  4     4    132        6     248
##  5     5    136        8     195
##  6     6    156       12     238
##  7     7    121        6     157
##  8     8    143        5     151
##  9     9    168        9     204
## 10    10    151        5     189
```

Also note the liver enzyme variables begin with "lft". We can pass multiple strings to `starts_with()`.

```
blood_example %>%
  select(starts_with(c("cbc", "lft")))
```

```
## # A tibble: 10 x 6
##    cbc_hb cbc_leuk cbc_plt lft_alt lft_bili_tot lft_bili_dir
##     <dbl>    <dbl>   <dbl>   <dbl>        <dbl>        <dbl>
##  1    147        8     235      43            7            4
##  2    155        4     205      54           10            5
##  3    139       13     206      49            7            6
##  4    132        6     248      44            7            3
##  5    136        8     195      54            8            6
##  6    156       12     238      44            8            7
##  7    121        6     157      49            7            5
##  8    143        5     151      41            7            3
##  9    168        9     204      45           10            6
## 10    151        5     189      42            9            6
```

You can imagine situations where the number of strings may be quite large to select. To avoid the code becoming too cluttered, you can assign the strings to a vector and pass that to `starts_with()`. The following is equivalent to what we just did.

```
vars_wanted <- c("cbc", "lft")

blood_example %>%
  select(starts_with(vars_wanted))
```

```
## # A tibble: 10 x 6
##    cbc_hb cbc_leuk cbc_plt lft_alt lft_bili_tot lft_bili_dir
##     <dbl>    <dbl>   <dbl>   <dbl>        <dbl>        <dbl>
##  1    147        8     235      43            7            4
##  2    155        4     205      54           10            5
##  3    139       13     206      49            7            6
##  4    132        6     248      44            7            3
##  5    136        8     195      54            8            6
##  6    156       12     238      44            8            7
##  7    121        6     157      49            7            5
##  8    143        5     151      41            7            3
##  9    168        9     204      45           10            6
## 10    151        5     189      42            9            6
```

If we wanted just the two bilirubin values and used `starts_with()` we would also select for *lft_alt*. We can use `contains()` instead.

```
blood_example %>%
  select(contains("bili"))
```

```
## # A tibble: 10 x 2
##    lft_bili_tot lft_bili_dir
##           <dbl>        <dbl>
##  1            7            4
##  2           10            5
```

```
##  3            7            6
##  4            7            3
##  5            8            6
##  6            8            7
##  7            7            5
##  8            7            3
##  9           10            6
## 10            9            6
```

Even more complex selecting can be achieved. To select variables based on logical statements we can use `where()`. For example, to keep only numeric variables.

```
cancer %>%
  select(where(is.numeric))
```

```
## # A tibble: 80 x 4
##    pt_id   age tumour_size follow_up
##    <dbl> <dbl>       <dbl>     <dbl>
##  1     1    62        0.41        58
##  2     2    57        2.84        52
##  3     3    66        3.35        33
##  4     4    54        4.63        58
##  5     5    49        1.29        56
##  6     6    63        1.99        41
##  7     7    54        1.86        64
##  8     8    72        1.01        32
##  9     9    57        1.86        30
## 10    10    65        3.91        40
## # ... with 70 more rows
```

```
#This obviously extends to
cancer %>%
  select(where(is.character))

#And
cancer %>%
  select(where(is.factor))
```

`where()` can be used in even more complex situations. Say we have 10 patient's systolic blood pressure across multiple visits, where in each visit a different pain strategy was used for a painful procedure.

```
## # A tibble: 10 x 6
##    pt_id visit_1 visit_2 visit_3 visit_4 visit_5
##    <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1     1     105     119     122     147     159
##  2     2     124     118     122     149     156
##  3     3     116     114     127     145     156
##  4     4     121     116     132     154     167
##  5     5     111     120     125     145     157
##  6     6     124     117     122     150     163
##  7     7     126     120     118     147     151
##  8     8     119     116     123     143     157
##  9     9     111     112     124     140     148
## 10    10     122     114     117     139     155
```

If we wanted to keep only the patient ID and visits where the mean blood pressure is greater than 130 (perhaps suggesting the pain management in those visits was suboptimal), a `where()` solution would be.

```
bp_example %>%
  select(pt_id,
         starts_with("visit") & where(function(x) mean(x)>130))
```

```
## # A tibble: 10 x 3
##    pt_id visit_4 visit_5
##    <int>   <dbl>   <dbl>
##  1     1     147     159
##  2     2     149     156
##  3     3     145     156
##  4     4     154     167
##  5     5     145     157
##  6     6     150     163
##  7     7     147     151
##  8     8     143     157
##  9     9     140     148
## 10    10     139     155
```