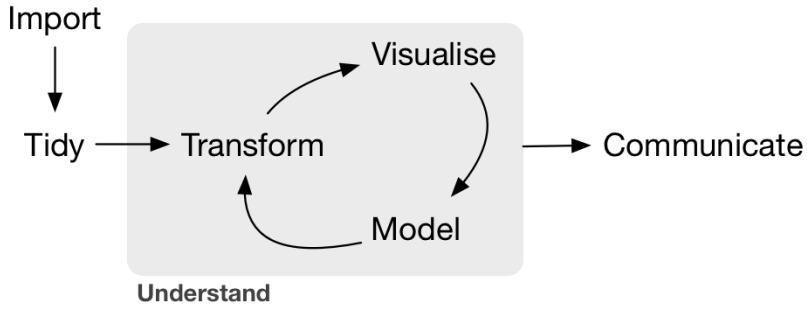
Data Wrangling with `dplyr`

Stephanie Hicks, Rafael Irizarry

Once data has been transformed into a **tidy** tabular format, an important part of "data wrangling" or "data munging" is data transformation.



We have learned about some functions in the R package `dplyr` that can transform and summarize tabular data with rows and columns. For example, when data is a tabular format, you have seen how we can use `dplyr` to `filter()` rows, `select()` columns and add new columns using `mutate()`. Now we will explore some more advanced `dplyr` functionality.

Brief recap of dplyr

`dplyr` is a powerful R-package to transform and summarize tabular data with rows and columns.

The package contains a set of functions (or "verbs") to perform common data manipulation operations such as filtering for rows, selecting specific columns, re-ordering rows, adding new columns and summarizing data. In addition, `dplyr` contains a useful function to perform another common task which is the is the "split-apply-combine" concept. We will discuss that in a little bit.

Data

mammals sleep

The `msleep` (mammals sleep) data set contains the sleeptimes and weights for a set of mammals and is available in the `data` repository on GitHub. This data set contains 83 rows and 11 variables.

To load the `msleep` data set

```
library(readr)
library(dplyr)
library(ggplot2)

msleep <- read_csv("https://raw.githubusercontent.com/datasciencelabs/data/master/msleep_ggplot2.csv")
msleep
```

```
## Source: local data frame [83 x 11]
##
##                         name       genus  vore        order conservation
```

```
##                          (chr)       (chr) (chr)        (chr)        (chr)
## 1                      Cheetah    Acinonyx carni    Carnivora           lc
## 2                  Owl monkey       Aotus  omni     Primates           NA
## 3              Mountain beaver  Aplodontia herbi     Rodentia           nt
## 4  Greater short-tailed shrew     Blarina  omni Soricomorpha           lc
## 5                          Cow         Bos herbi Artiodactyla domesticated
## 6             Three-toed sloth    Bradypus herbi       Pilosa           NA
## 7           Northern fur seal Callorhinus carni    Carnivora           vu
## 8                Vesper mouse     Calomys    NA     Rodentia           NA
## 9                         Dog       Canis carni    Carnivora domesticated
## 10                    Roe deer   Capreolus herbi Artiodactyla           lc
## ..                         ...         ...   ...          ...          ...
## Variables not shown: sleep_total (dbl), sleep_rem (dbl), sleep_cycle
##   (dbl), awake (dbl), brainwt (dbl), bodywt (dbl)
```

The columns (in order) correspond to the following:

column name	Description
`name`	common name
`genus`	taxonomic rank
`vore`	carnivore, omnivore or herbivore?
`order`	taxonomic rank
`conservation`	the conservation status of the mammal
`sleep_total`	total amount of sleep, in hours
`sleep_rem`	rem sleep, in hours
`sleep_cycle`	length of sleep cycle, in hours
`awake`	amount of time spent awake, in hours
`brainwt`	brain weight in kilograms
`bodywt`	body weight in kilograms

Important `dplyr` verbs to remember

`dplyr` verbs	Description
`select()`	select columns
`mutate()`	create new columns
`filter()`	filter rows
`arrange()`	re-order or arrange rows
`summarise()`	summarise values
`group_by()`	allows for group operations in the "split-apply-combine" concept

`dplyr` verbs in action

`select()` and `mutate()` columns; `filter()` rows

The two most basic functions are `select()` and `filter()` which selects columns and filters rows, respectively. The function `mutate()` can be used to create new columns. We have already seen examples of all of these in class.

For example, to select a range of columns by name, use the ":" (colon) operator

```
msleep %>% select(name:order)
```

```
## Source: local data frame [83 x 4]
##
##                          name       genus  vore        order
##                         (chr)       (chr) (chr)        (chr)
## 1                     Cheetah    Acinonyx carni    Carnivora
## 2                 Owl monkey       Aotus  omni     Primates
## 3             Mountain beaver  Aplodontia herbi     Rodentia
## 4  Greater short-tailed shrew     Blarina  omni Soricomorpha
## 5                         Cow         Bos herbi Artiodactyla
## 6            Three-toed sloth    Bradypus herbi       Pilosa
## 7           Northern fur seal Callorhinus carni    Carnivora
## 8                Vesper mouse     Calomys    NA     Rodentia
## 9                         Dog       Canis carni    Carnivora
## 10                  Roe deer   Capreolus herbi Artiodactyla
## ..                       ...         ...   ...          ...
```

To select all columns that start with the character string "sl", use the function `starts_with()`

```
msleep %>% select(starts_with("sl"))
```

```
## Source: local data frame [83 x 3]
##
##    sleep_total sleep_rem sleep_cycle
##          (dbl)     (dbl)       (dbl)
## 1         12.1        NA          NA
## 2         17.0       1.8          NA
## 3         14.4       2.4          NA
## 4         14.9       2.3   0.1333333
## 5          4.0       0.7   0.6666667
## 6         14.4       2.2   0.7666667
## 7          8.7       1.4   0.3833333
## 8          7.0        NA          NA
## 9         10.1       2.9   0.3333333
## 10         3.0        NA          NA
## ..         ...       ...         ...
```

Some additional options to select columns based on a specific criteria include

1. `ends_with()` = Select columns that end with a character string
2. `contains()` = Select columns that contain a character string
3. `matches()` = Select columns that match a regular expression
4. `one_of()` = Select columns names that are from a group of names

Assessment Select all columns except those from `genus` to `conservation` and filter the rows for mammals that sleep a total of more than 16 hours and have a body weight of greater than 1 kilogram

```
## Provide your code here

msleep %>%
    select(-(genus:conservation)) %>%
    filter(sleep_total >= 16, bodywt >= 1)
```

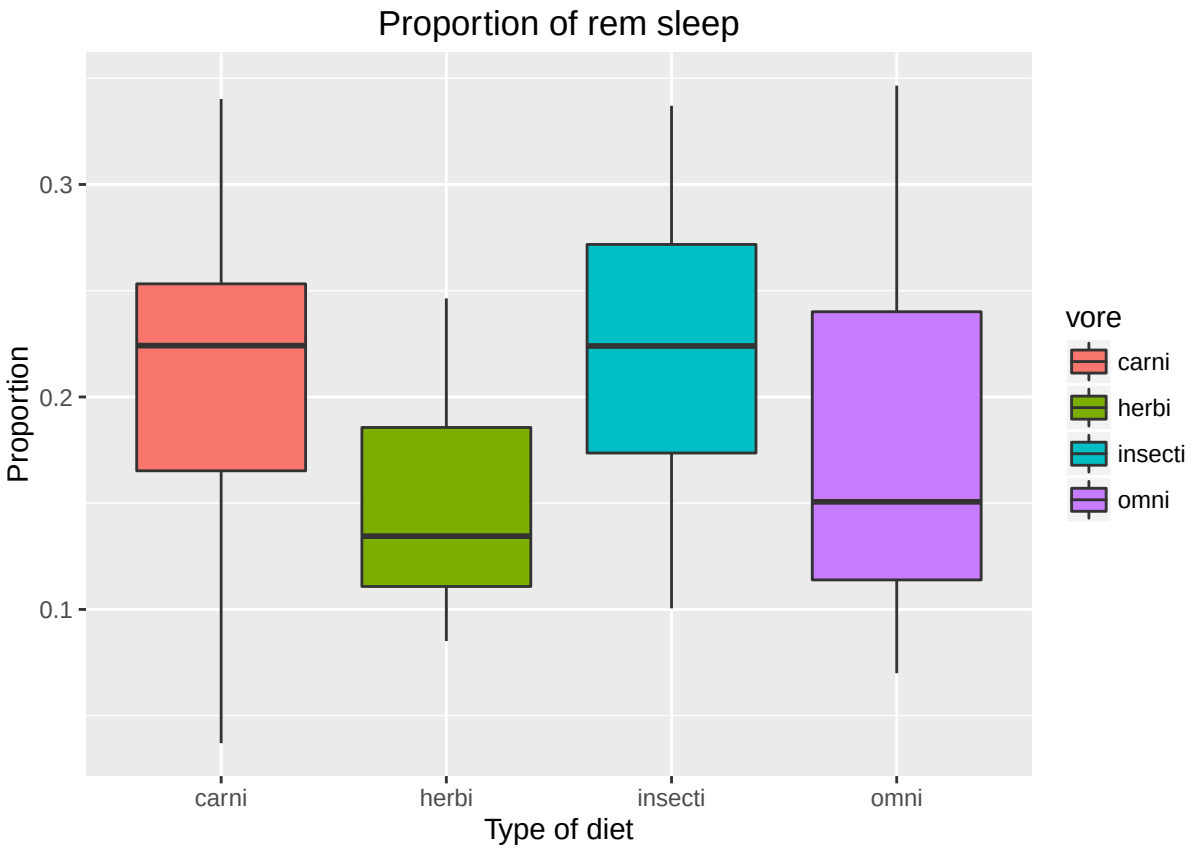
```
## Source: local data frame [3 x 7]
##
##                     name sleep_total sleep_rem sleep_cycle awake brainwt
##                    (chr)       (dbl)     (dbl)       (dbl) (dbl)   (dbl)
## 1   Long-nosed armadillo        17.4       3.1   0.3833333   6.6  0.0108
## 2 North American Opossum        18.0       4.9   0.3333333   6.0  0.0063
## 3        Giant armadillo        18.1       6.1          NA   5.9  0.0810
## Variables not shown: bodywt (dbl)
```

Assessment Create a new column called `rem_proportion` which is the ratio of rem sleep to total amount of sleep and create boxplots of the `rem_proportion` column split and colored by the `vore` column. Create labels for the `x` and `y` axis.

```
## Provide your code here

msleep %>%
    mutate(rem_proportion = sleep_rem / sleep_total) %>%
    ggplot(aes(x = vore, y = rem_proportion, fill = vore)) +
        geom_boxplot() + xlab("Type of diet") +
        ylab("Proportion") + labs(title = "Proportion of rem sleep")
```

```
## Warning: Removed 27 rows containing non-finite values (stat_boxplot).
```



Assessment (optional) Select all columns that start with the character string "sl" or ends with the character string "wt", create a new column called `rem_proportion` which is the ratio of rem sleep to total amount of sleep, create a second column `bodywt_grams` which is the bodywt column in grams and filter for the rows 20 to 30 in the msleep data set by numerical position.

Hint: Look at the `slice()` help file to filter for rows by numerical position.

```
## Provide your code here

msleep %>%
    select(starts_with("sl"), ends_with("wt")) %>%
    mutate(rem_proportion = sleep_rem / sleep_total,
           bodywt_grams = bodywt * 1000) %>%
    slice(20:30)
```

```
## Source: local data frame [11 x 7]
##
##    sleep_total sleep_rem sleep_cycle brainwt   bodywt rem_proportion
##          (dbl)     (dbl)       (dbl)   (dbl)    (dbl)          (dbl)
## 1         18.0       4.9   0.3333333  0.0063    1.700      0.2722222
## 2          3.9        NA          NA  4.6030 2547.000             NA
## 3         19.7       3.9   0.1166667  0.0003    0.023      0.1979695
## 4          2.9       0.6   1.0000000  0.6550  521.000      0.2068966
## 5          3.1       0.4          NA  0.4190  187.000      0.1290323
## 6         10.1       3.5   0.2833333  0.0035    0.770      0.3465347
## 7         10.9       1.1          NA  0.1150   10.000      0.1009174
```

```
## 8         14.9        NA          NA      NA    0.071             NA
## 9         12.5       3.2   0.4166667  0.0256    3.300      0.2560000
## 10         9.8       1.1   0.5500000  0.0050    0.200      0.1122449
## 11         1.9       0.4          NA      NA  899.995      0.2105263
## Variables not shown: bodywt_grams (dbl)
```

Arrange or re-order rows using `arrange()`

To arrange (or re-order) rows by a particular column such as the taxonomic order, list the name of the column you want to arrange the rows by

```
msleep %>%
    arrange(order)
```

```
## Source: local data frame [83 x 11]
##
##                 name       genus  vore        order conservation
##                (chr)       (chr) (chr)        (chr)        (chr)
## 1             Tenrec      Tenrec  omni Afrosoricida           NA
## 2                Cow         Bos herbi Artiodactyla domesticated
## 3          Roe deer   Capreolus herbi Artiodactyla           lc
## 4               Goat       Capri herbi Artiodactyla           lc
## 5            Giraffe     Giraffa herbi Artiodactyla           cd
## 6              Sheep        Ovis herbi Artiodactyla domesticated
## 7                Pig         Sus  omni Artiodactyla domesticated
## 8            Cheetah    Acinonyx carni    Carnivora           lc
## 9  Northern fur seal Callorhinus carni    Carnivora           vu
## 10               Dog       Canis carni    Carnivora domesticated
## ..               ...         ...   ...          ...          ...
## Variables not shown: sleep_total (dbl), sleep_rem (dbl), sleep_cycle
##   (dbl), awake (dbl), brainwt (dbl), bodywt (dbl)
```

Assessment Select all columns names with the characters "sleep" and arrange the rows for the `sleep_rem` in a decreasing order.

Hint: look at the `?arrange` help file for the `desc()` option.

```
## Provide your code here

msleep %>%
    select(matches("sleep")) %>%
    arrange(desc(sleep_rem))
```

```
## Source: local data frame [83 x 3]
##
##    sleep_total sleep_rem sleep_cycle
##          (dbl)     (dbl)       (dbl)
## 1         19.4       6.6          NA
## 2         18.1       6.1          NA
## 3         18.0       4.9   0.3333333
## 4         19.7       3.9   0.1166667
## 5         10.1       3.5   0.2833333
```

```
## 6         13.8       3.4   0.2166667
## 7         12.5       3.2   0.4166667
## 8         17.4       3.1   0.3833333
## 9         14.3       3.1   0.2000000
## 10        15.9       3.0          NA
## ..         ...       ...         ...
```

Assessment Select three columns from `msleep` (`name`, `order`, `sleep_total`), arrange the rows in the `sleep_total` column in a descending order, and filter the rows for mammals that sleep for a total of 16 or more hours.

```
## Provide your code here

msleep %>%
    select(name, order, sleep_total) %>%
    arrange(order, desc(sleep_total)) %>%
    filter(sleep_total >= 16)
```

```
## Source: local data frame [8 x 3]
##
##                     name           order sleep_total
##                    (chr)           (chr)       (dbl)
## 1       Little brown bat      Chiroptera        19.9
## 2          Big brown bat      Chiroptera        19.7
## 3        Giant armadillo       Cingulata        18.1
## 4   Long-nosed armadillo       Cingulata        17.4
## 5   Thick-tailed opposum Didelphimorphia        19.4
## 6 North American Opossum Didelphimorphia        18.0
## 7             Owl monkey        Primates        17.0
## 8 Arctic ground squirrel        Rodentia        16.6
```

Create summaries of the data frame using `summarise()`

The `summarise()` function will create summary statistics for a given column in the data frame such as finding the mean. For example, to compute the average number of hours of sleep, apply the `mean()` function to the column `sleep_total` and call the summary value `avg_sleep`.

```
msleep %>%
    summarise(avg_sleep = mean(sleep_total))
```

```
## Source: local data frame [1 x 1]
##
##   avg_sleep
##       (dbl)
## 1  10.43373
```

There are many other summary statistics you could consider such `sd()`, `min()`, `max()`, `median()`, `sum()`, `n()` (returns the length of vector), `first()` (returns first value in vector), `last()` (returns last value in vector) and `n_distinct()` (number of distinct values in vector).

Assessment Summarize `sleep_total` column in the `msleep` data set with the average sleep, the minimum and maximum amount of sleep, and the total number of mammals.

```
## Provide your code here

msleep %>%
    summarise(avg_sleep = mean(sleep_total),
              min_sleep = min(sleep_total),
              max_sleep = max(sleep_total),
              total = n())
```

```
## Source: local data frame [1 x 4]
##
##   avg_sleep min_sleep max_sleep total
##       (dbl)     (dbl)     (dbl) (int)
## 1  10.43373       1.9      19.9    83
```

Group operations using `group_by()`

The `group_by()` verb is an important function in `dplyr`. As we mentioned before it's related to concept of "split-apply-combine". We literally want to split the data frame by some variable (e.g. `vore`), apply a function to the individual data frames and then combine the output.

Say we wanted to calculate the standard deviation of the body and brain weights for each of factor in the `vore` column. First, we can look at the types of factors in the `vore` column

```
table(msleep$vore)
```

```
##
##   carni   herbi insecti    omni
##      19      32       5      20
```

Then, we could use `filter()` to filter for rows that contain "carni" in the `vore` column and summarize with the mean of the brain and body weights.

```
msleep %>%
    filter(vore == "carni") %>%
    summarize("bodywt_sd" = mean(bodywt),
              "brainwt_sd" = mean(brainwt, na.rm = TRUE))
```

```
## Source: local data frame [1 x 2]
##
##   bodywt_sd brainwt_sd
##       (dbl)      (dbl)
## 1  90.75111 0.07925556
```

We could repeat this for each factor in `vore`, which is a bit teadious. Instead, we could use this using the `group_by()` function.

Let's do that: split the `msleep` data frame by the `vore` column, then calculate the mean of body weight and brain weight for each invididual data frame. (hint: We expect a set of summary statistics for each level in `vore`.)

```
msleep %>%
    group_by(vore) %>%
    summarize("bodywt_sd" = mean(bodywt),
              "brainwt_sd" = mean(brainwt, na.rm = TRUE))
```

```
## Source: local data frame [5 x 3]
##
##      vore bodywt_sd brainwt_sd
##     (chr)     (dbl)      (dbl)
## 1   carni  90.75111 0.07925556
## 2   herbi 366.87725 0.62159750
## 3 insecti  12.92160 0.02155000
## 4    omni  12.71800 0.14573118
## 5      NA   0.85800 0.00762600
```

Assessment Split the `msleep` data frame by the taxonomic order, then for each taxonomic order summarize the `sleep_total` with the average sleep, the minimum and maximum amount of sleep, and the total number of mammals in each order.

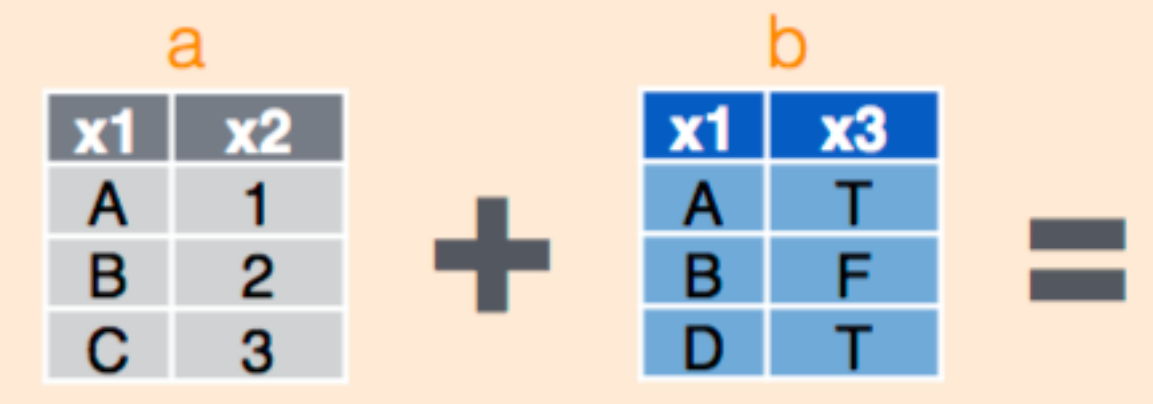
```
## Provide your code here

msleep %>%
    group_by(order) %>%
    summarise(avg_sleep = mean(sleep_total),
              min_sleep = min(sleep_total),
              max_sleep = max(sleep_total),
              total = n())
```

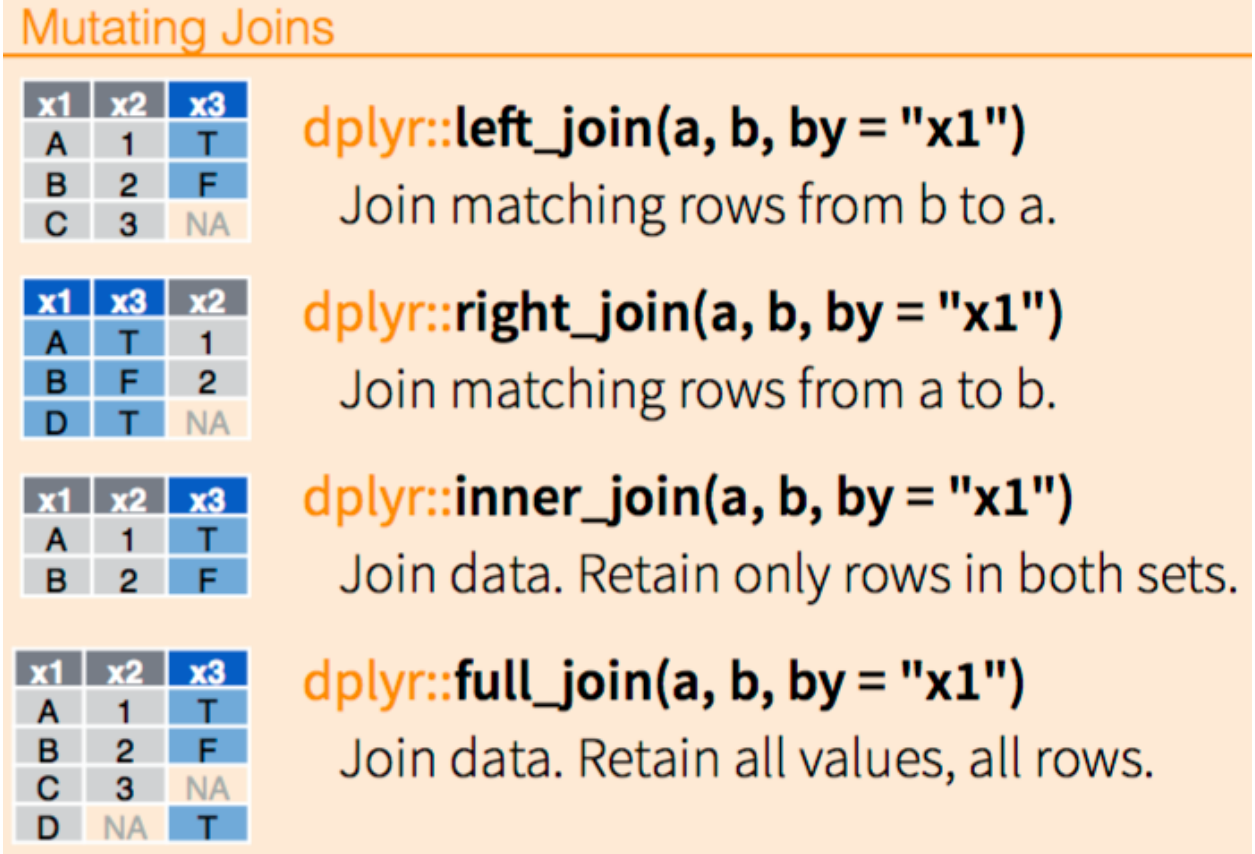
```
## Source: local data frame [19 x 5]
##
##              order avg_sleep min_sleep max_sleep total
##              (chr)     (dbl)     (dbl)     (dbl) (int)
## 1     Afrosoricida 15.600000      15.6      15.6     1
## 2     Artiodactyla  4.516667       1.9       9.1     6
## 3        Carnivora 10.116667       3.5      15.8    12
## 4          Cetacea  4.500000       2.7       5.6     3
## 5       Chiroptera 19.800000      19.7      19.9     2
## 6        Cingulata 17.750000      17.4      18.1     2
## 7  Didelphimorphia 18.700000      18.0      19.4     2
## 8    Diprotodontia 12.400000      11.1      13.7     2
## 9   Erinaceomorpha 10.200000      10.1      10.3     2
## 10      Hyracoidea  5.666667       5.3       6.3     3
## 11      Lagomorpha  8.400000       8.4       8.4     1
## 12     Monotremata  8.600000       8.6       8.6     1
## 13  Perissodactyla  3.466667       2.9       4.4     3
## 14          Pilosa 14.400000      14.4      14.4     1
## 15        Primates 10.500000       8.0      17.0    12
## 16     Proboscidea  3.600000       3.3       3.9     2
## 17        Rodentia 12.468182       7.0      16.6    22
## 18      Scandentia  8.900000       8.9       8.9     1
## 19    Soricomorpha 11.100000       8.4      14.9     5
```

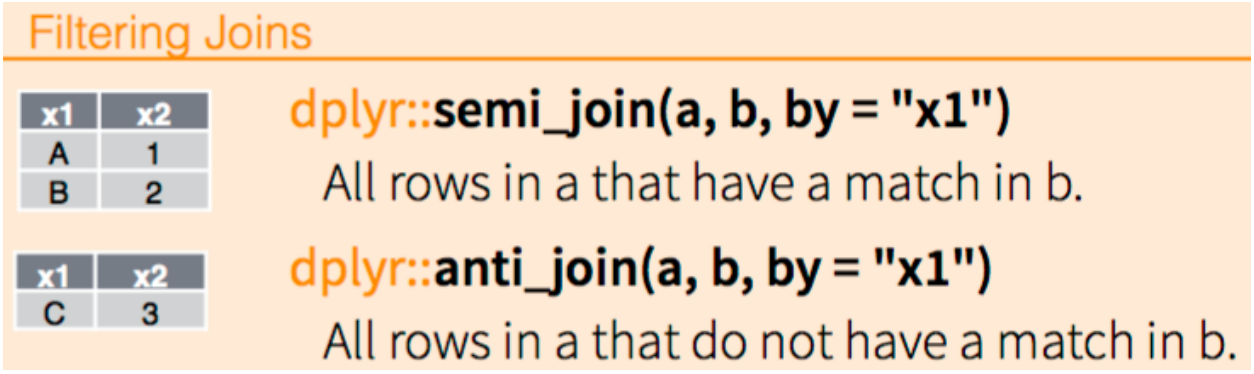
joining two data frames in `dplyr`

The last part of `dplyr` that we will discuss are a set of `dplyr` verbs that allow you to join two data sets. The following are cartoons extracted from Data Wrangling with dplyr and tidyr Cheatsheet from RStudio to illustrate the different ways to join data frames using `dplyr`.



Two types of functions to join together data frames





Data

For this section, we will work with two small data sets related to the 2016 Oscars Nominations. The two data sets are `oscars` and `movies`. The first data set contains information about the the name of the actor/actress, the name of the movie and the category for nomination. The second data set contains a list of movies and the length of the movie in minutes.

We will explore the `dplyr` verbs to join the two tables. First let's load the data.

```
library(readr)

oscars <-"
name,movie,category
Adam McKay,The Big Short,Best Director
Alejandro González Iñárritu,The Revenant,Best Director
Lenny Abrahamson,Room,Best Director
Tom McCarthy,Spotlight,Best Director
George Miller,Mad Max: Fury Road,Best Director
Bryan Cranston,Trumbo,Best Actor
Matt Damon,The Martian,Best Actor
Michael Fassbender,Steve Jobs,Best Actor
Leonardo DiCaprio,The Revenant,Best Actor
Eddie Redmayne,The Danish Girl,Best Actor
Cate Blanchett,Carol,Best Actress
Brie Larson,Room,Best Actress
Jennifer Lawrence,Joy,Best Actress
Charlotte Rampling,45 Years,Best Actress
Saoirse Ronan,Brooklyn,Best Actress
"
oscars <- read_csv(oscars, trim_ws = TRUE, skip = 1)
oscars
```

```
## Source: local data frame [15 x 3]
##
##                          name              movie      category
##                         (chr)              (chr)         (chr)
## 1                  Adam McKay      The Big Short Best Director
## 2  Alejandro González Iñárritu       The Revenant Best Director
## 3             Lenny Abrahamson               Room Best Director
## 4                 Tom McCarthy          Spotlight Best Director
## 5               George Miller Mad Max: Fury Road Best Director
```

```
## 6               Bryan Cranston             Trumbo    Best Actor
## 7                   Matt Damon        The Martian    Best Actor
## 8           Michael Fassbender         Steve Jobs    Best Actor
## 9            Leonardo DiCaprio       The Revenant    Best Actor
## 10              Eddie Redmayne    The Danish Girl    Best Actor
## 11              Cate Blanchett              Carol  Best Actress
## 12                 Brie Larson               Room  Best Actress
## 13           Jennifer Lawrence                Joy  Best Actress
## 14          Charlotte Rampling           45 Years  Best Actress
## 15               Saoirse Ronan           Brooklyn  Best Actress
```

```
movies <-"
movie,length_mins
The Big Short,130
Star Wars: The Force Awakens,135
Brooklyn,111
Mad Max: Fury Road,120
Room,118
The Martian,144
The Revenant,156
Spotlight,128
"
movies <- read_csv(movies, trim_ws = TRUE, skip = 1)
movies
```

```
## Source: local data frame [8 x 2]
##
##                          movie length_mins
##                          (chr)       (int)
## 1                The Big Short         130
## 2 Star Wars: The Force Awakens         135
## 3                     Brooklyn         111
## 4           Mad Max: Fury Road         120
## 5                         Room         118
## 6                  The Martian         144
## 7                 The Revenant         156
## 8                    Spotlight         128
```

inner_join(x,y)

This function joins all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.

```
inner_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [9 x 4]
##
##                          name              movie      category length_mins
##                         (chr)              (chr)         (chr)       (int)
## 1                  Adam McKay      The Big Short Best Director         130
## 2 Alejandro González Iñárritu       The Revenant Best Director         156
## 3            Lenny Abrahamson               Room Best Director         118
```

```
## 4                Tom McCarthy          Spotlight Best Director         128
## 5              George Miller Mad Max: Fury Road Best Director         120
## 6                 Matt Damon        The Martian    Best Actor         144
## 7           Leonardo DiCaprio       The Revenant    Best Actor         156
## 8                Brie Larson               Room  Best Actress         118
## 9              Saoirse Ronan           Brooklyn  Best Actress         111
```

semi_join(x,y)

This function returns all rows from x where there are matching values in y, keeping just columns from x. A semi join differs from an inner join because an inner join will return one row of x for each matching row of y, where a semi join will never duplicate rows of x.

```
semi_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [9 x 3]
##
##                          name              movie      category
##                         (chr)              (chr)         (chr)
## 1                  Adam McKay      The Big Short Best Director
## 2               Saoirse Ronan           Brooklyn  Best Actress
## 3               George Miller Mad Max: Fury Road Best Director
## 4            Lenny Abrahamson               Room Best Director
## 5                 Brie Larson               Room  Best Actress
## 6                  Matt Damon        The Martian    Best Actor
## 7 Alejandro González Iñárritu       The Revenant Best Director
## 8           Leonardo DiCaprio       The Revenant    Best Actor
## 9                 Tom McCarthy          Spotlight Best Director
```

Assessment Try applying the `semi_join()` function with `x=movies` and `y=oscars`. What is the difference?

```
## Provide your code here

semi_join(movies, oscars, by = "movie")
```

```
## Source: local data frame [7 x 2]
##
##                movie length_mins
##                (chr)       (int)
## 1      The Big Short         130
## 2       The Revenant         156
## 3               Room         118
## 4          Spotlight         128
## 5 Mad Max: Fury Road         120
## 6        The Martian         144
## 7           Brooklyn         111
```

Assessment

Using the `dplyr` join functions, combine the columns from the `oscars` and `movies` data sets and return all rows from the `oscars` data set and all columns in both the `oscars` and `movies` columns.

Hint: read the help file for `left_join()` or `right_join()`.

```
## Provide your code here

left_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [15 x 4]
##
##                           name              movie      category
##                          (chr)              (chr)         (chr)
## 1                   Adam McKay      The Big Short Best Director
## 2  Alejandro González Iñárritu       The Revenant Best Director
## 3             Lenny Abrahamson               Room Best Director
## 4                 Tom McCarthy          Spotlight Best Director
## 5                George Miller Mad Max: Fury Road Best Director
## 6               Bryan Cranston             Trumbo    Best Actor
## 7                   Matt Damon        The Martian    Best Actor
## 8           Michael Fassbender         Steve Jobs    Best Actor
## 9            Leonardo DiCaprio       The Revenant    Best Actor
## 10              Eddie Redmayne    The Danish Girl    Best Actor
## 11              Cate Blanchett              Carol  Best Actress
## 12                 Brie Larson               Room  Best Actress
## 13           Jennifer Lawrence                Joy  Best Actress
## 14          Charlotte Rampling           45 Years  Best Actress
## 15                Saoirse Ronan           Brooklyn  Best Actress
## Variables not shown: length_mins (int)
```

```
## or
right_join(movies, oscars, by = "movie")
```

```
## Source: local data frame [15 x 4]
##
##                 movie length_mins                        name
##                 (chr)       (int)                       (chr)
## 1       The Big Short         130                  Adam McKay
## 2        The Revenant         156 Alejandro González Iñárritu
## 3                Room         118            Lenny Abrahamson
## 4           Spotlight         128                Tom McCarthy
## 5  Mad Max: Fury Road         120              George Miller
## 6              Trumbo          NA              Bryan Cranston
## 7         The Martian         144                  Matt Damon
## 8          Steve Jobs          NA          Michael Fassbender
## 9        The Revenant         156           Leonardo DiCaprio
## 10    The Danish Girl          NA              Eddie Redmayne
## 11              Carol          NA              Cate Blanchett
## 12               Room         118                 Brie Larson
## 13                Joy          NA           Jennifer Lawrence
## 14          45 Years          NA          Charlotte Rampling
## 15          Brooklyn         111                Saoirse Ronan
## Variables not shown: category (chr)
```

Why are there NAs?

Assessment

Using the `dplyr` join functions, combine the columns from the `oscars` and `movies` data sets and return all rows from the `movies` data set and all columns in both the `oscars` and `movies` columns.

```
## Provide your code here

right_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [10 x 4]
##
##                           name                        movie      category
##                          (chr)                        (chr)         (chr)
## 1                   Adam McKay                The Big Short Best Director
## 2                           NA Star Wars: The Force Awakens            NA
## 3               Saoirse Ronan                      Brooklyn  Best Actress
## 4               George Miller           Mad Max: Fury Road Best Director
## 5            Lenny Abrahamson                        Room Best Director
## 6                 Brie Larson                        Room  Best Actress
## 7                  Matt Damon                  The Martian    Best Actor
## 8  Alejandro González Iñárritu                The Revenant Best Director
## 9           Leonardo DiCaprio                The Revenant    Best Actor
## 10               Tom McCarthy                   Spotlight Best Director
## Variables not shown: length_mins (int)
```

```
## or
left_join(movies, oscars, by = "movie")
```

```
## Source: local data frame [10 x 4]
##
##                           movie length_mins                       name
##                           (chr)       (int)                      (chr)
## 1                 The Big Short         130                 Adam McKay
## 2  Star Wars: The Force Awakens         135                         NA
## 3                      Brooklyn         111              Saoirse Ronan
## 4            Mad Max: Fury Road         120              George Miller
## 5                          Room         118           Lenny Abrahamson
## 6                          Room         118                Brie Larson
## 7                   The Martian         144                 Matt Damon
## 8                 The Revenant         156 Alejandro González Iñárritu
## 9                 The Revenant         156          Leonardo DiCaprio
## 10                   Spotlight         128               Tom McCarthy
## Variables not shown: category (chr)
```

full_join(x,y)

This function returns all rows and all columns from both x and y. When there are not matching values, it will return NA for the one missing.

```
full_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [16 x 4]
##
##                          name                        movie      category
##                         (chr)                        (chr)         (chr)
## 1                   Adam McKay                The Big Short Best Director
## 2  Alejandro González Iñárritu                The Revenant Best Director
## 3              Lenny Abrahamson                        Room Best Director
## 4                  Tom McCarthy                   Spotlight Best Director
## 5                 George Miller           Mad Max: Fury Road Best Director
## 6                Bryan Cranston                      Trumbo    Best Actor
## 7                    Matt Damon                  The Martian    Best Actor
## 8            Michael Fassbender                   Steve Jobs    Best Actor
## 9             Leonardo DiCaprio                 The Revenant    Best Actor
## 10               Eddie Redmayne               The Danish Girl    Best Actor
## 11               Cate Blanchett                        Carol  Best Actress
## 12                  Brie Larson                         Room  Best Actress
## 13            Jennifer Lawrence                          Joy  Best Actress
## 14           Charlotte Rampling                    45 Years  Best Actress
## 15                Saoirse Ronan                     Brooklyn  Best Actress
## 16                           NA Star Wars: The Force Awakens            NA
## Variables not shown: length_mins (int)
```

```
full_join(movies, oscars, by = "movie")
```

```
## Source: local data frame [16 x 4]
##
##                           movie length_mins                       name
##                           (chr)       (int)                      (chr)
## 1                 The Big Short         130                  Adam McKay
## 2  Star Wars: The Force Awakens         135                          NA
## 3                       Brooklyn         111               Saoirse Ronan
## 4            Mad Max: Fury Road         120               George Miller
## 5                           Room         118             Lenny Abrahamson
## 6                           Room         118                 Brie Larson
## 7                    The Martian         144                   Matt Damon
## 8                   The Revenant         156 Alejandro González Iñárritu
## 9                   The Revenant         156            Leonardo DiCaprio
## 10                     Spotlight         128                 Tom McCarthy
## 11                       Trumbo          NA               Bryan Cranston
## 12                    Steve Jobs          NA           Michael Fassbender
## 13                The Danish Girl          NA               Eddie Redmayne
## 14                          Carol          NA               Cate Blanchett
## 15                            Joy          NA            Jennifer Lawrence
## 16                       45 Years          NA           Charlotte Rampling
## Variables not shown: category (chr)
```

Assessment

Using the `dplyr` join functions, return all rows from `oscars` data set where there are not matching values in `movies`, only keeping the columns from the `oscars` data set.

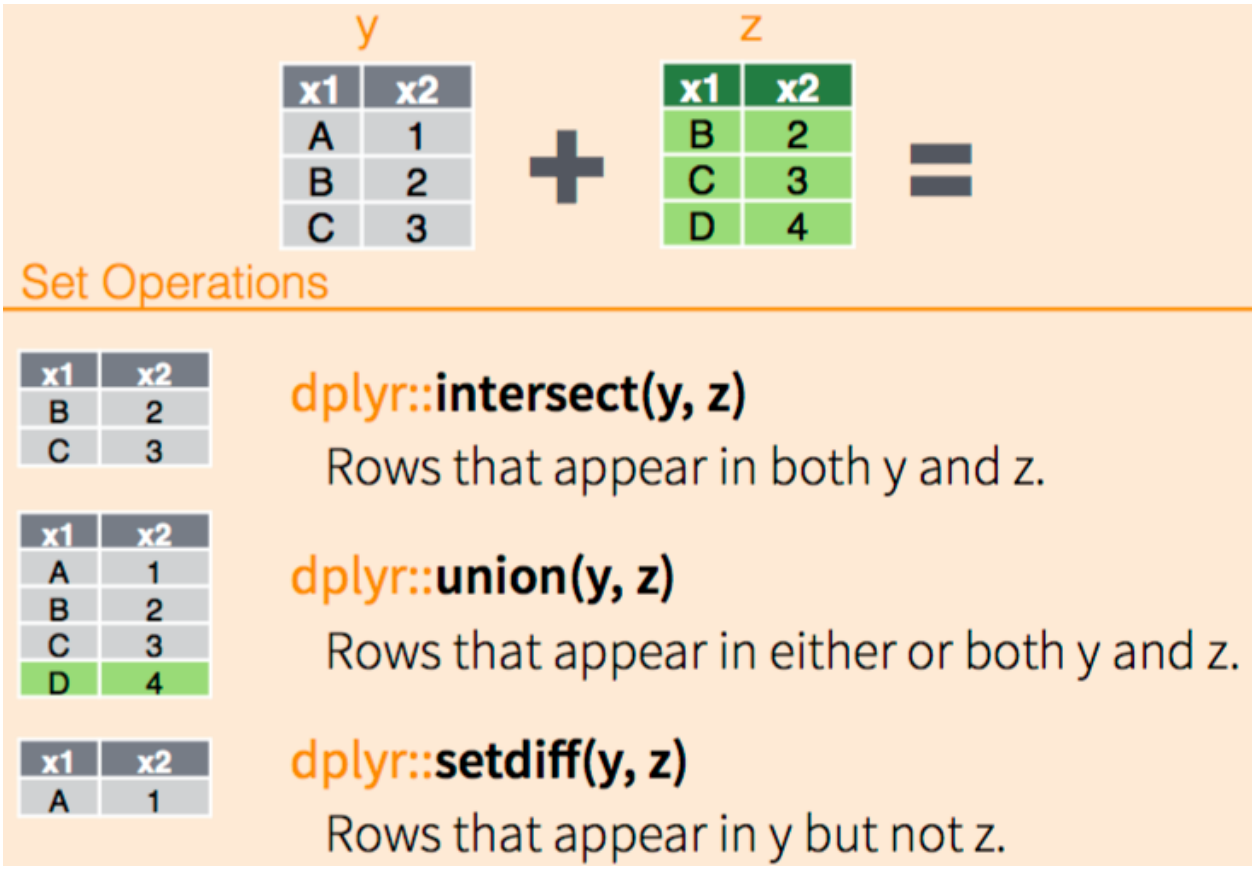
Hint: Read the help file for `anti_join()`.

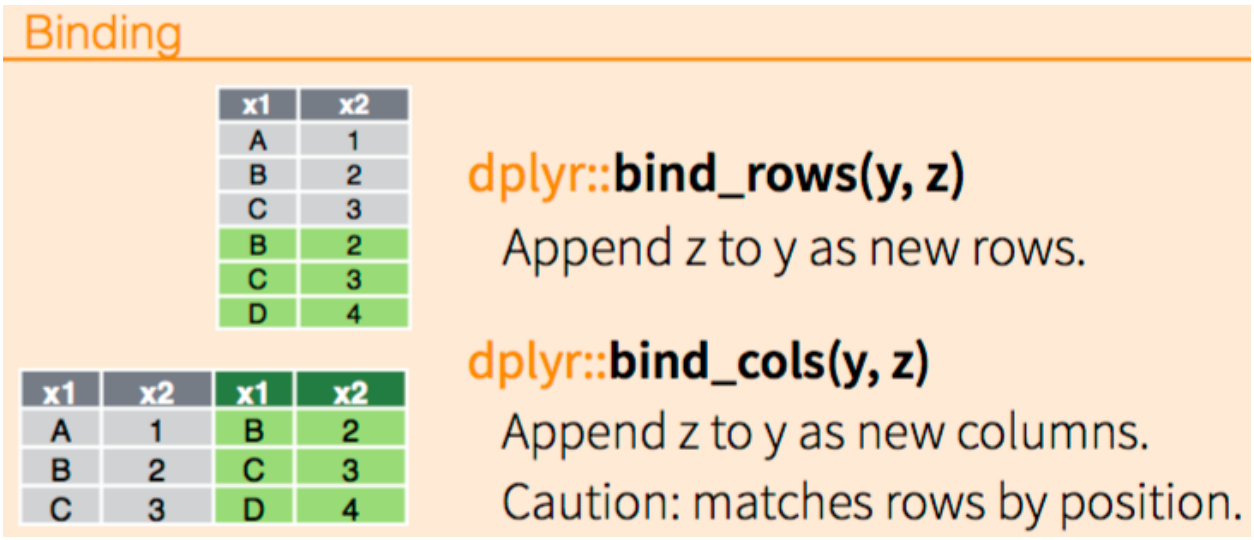
```
## Provide your code here

anti_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [6 x 3]
##
##                  name           movie     category
##                 (chr)           (chr)        (chr)
## 1 Charlotte Rampling        45 Years Best Actress
## 2  Jennifer Lawrence             Joy Best Actress
## 3     Cate Blanchett           Carol Best Actress
## 4     Eddie Redmayne The Danish Girl   Best Actor
## 5 Michael Fassbender      Steve Jobs   Best Actor
## 6     Bryan Cranston          Trumbo   Best Actor
```

Other functions in `dplyr` to join together data frames





Cheatsheets

- Data Wrangling with dplyr and tidyr from RStudio