SOLES codeRs

Data Wrangling with the `tidyverse`

Thomas White

Outline

- Describe the structure of tidy data, and understand how to load and save it in R
- Understand the function and operation of key `dplyr` verbs
- Flexibly and elegantly manipulate real-world data using piped commands (in R)

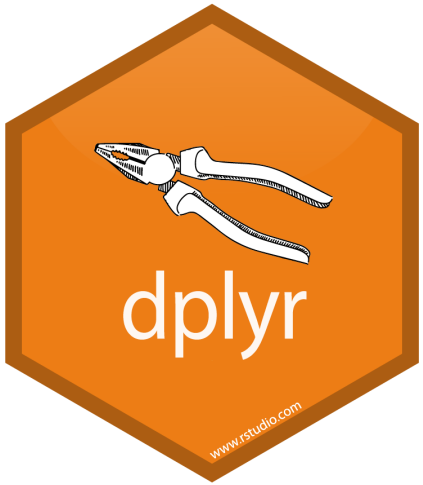
The problem

Data are often an absolute mess and/or not structured as we need!

How do we flexibly manipulate our data in a **transparent**, **reproducible** way?
(Remember: *we don't touch raw data*)

The solution

We wrangle!



the 'tidyverse'

R packages for data science

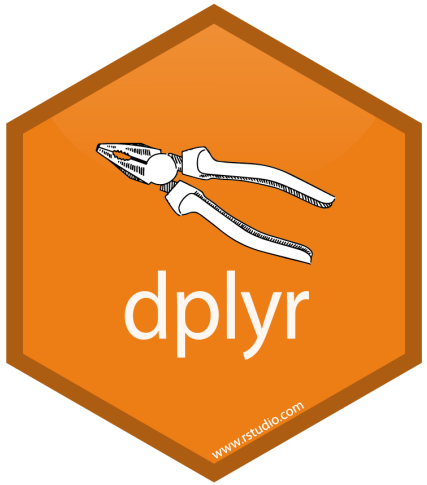
The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

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

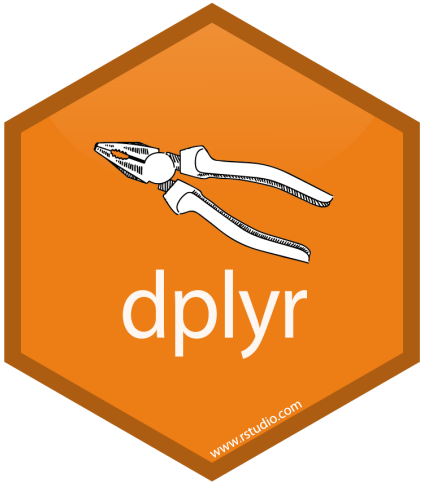
dplyr

`dplyr` is based on the concepts of functions as **verbs** that manipulate data frames.



dplyr

`dplyr` is based on the concepts of functions as **verbs** that manipulate data frames.



- The first argument is always a data frame
- Subsequent arguments say what to do with that data frame
- Always returns a data frame (technically a `tibble`)

The verbs

Reading

- `read_csv()`: loads `.csv` files

Wrangling

- `|>`: pipe data through functions
- `select()`: subset columns
- `filter()`: subset rows
- `mutate()`: create new columns using information from existing columns
- `summarise()` and `group_by()`: create summary statistics

Writing

- `write_csv()`: generates `.csv` files

read & pipe

read_csv()

```
# Load the tidyverse
library(tidyverse)

# Load up the data
dat_eco <- read_csv('ecol_survey_data.csv')
```

Inspect the data

```
head(dat_eco)
```

```
## # A tibble: 6 × 13
##   record_id month   day  year plot_id species_id sex   hindfoot_length weight genus   species  taxa   plot_type
##       <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr>   <chr>    <chr>  <chr>
## 1         1     7    16  1977       2 NL         M                  32     NA Neotoma albigula Rodent Control
## 2        72     8    19  1977       2 NL         M                  31     NA Neotoma albigula Rodent Control
## 3       224     9    13  1977       2 NL         <NA>               NA     NA Neotoma albigula Rodent Control
## 4       266    10    16  1977       2 NL         <NA>               NA     NA Neotoma albigula Rodent Control
## 5       349    11    12  1977       2 NL         <NA>               NA     NA Neotoma albigula Rodent Control
## 6       363    11    12  1977       2 NL         <NA>               NA     NA Neotoma albigula Rodent Control
```

|> pipe data

Consider a sequence of actions:

1. find key
2. unlock car
3. start car
4. drive to uni
5. park

Expressed as a set of nested functions in make-believe R code this could look like:

```
park(drive(start_car(unlock(find("keys")), which = 'car'), to = "campus"))
```

|> pipe data

Writing it out using **pipes** looks like:

```
me |>
  find("keys") |>
  unlock(which = 'car') |>
  start_car() |>
  drive(to = "campus") |>
  park()
```

|> pipe data

pipes create a more natural and easy-to-read structure.

It's ultimately down to preference, but they play particularly well with the `tidyverse` and are now part of base R (as of `4.1.0`) so let's get comfortable with them!

|> pipe data

Nested:

```
h(g(f(your_data), y = 1), z = 1)
```

Intermediate:

```
res1 <- f(your_data)
res2 <- g(res1, y = 1)
res3 <- h(res2, z = 1)
```

Piped:

```
your_data |>
  f() |>
  g(y = 1) |>
  h(z = 1)
```

|> pipe data

N.B. Remember to omit the `data` argument from functions when piping.

Yes:

```
  subset(the_data, variable_a > 3)
```

Yes:

```
the_data |>
  subset(variable_a > 3)
```

No:

```
the_data |>
  subset(the_data, variable_a > 3)
```

select() & filter()

Definitions

`select()`: select *variables* (columns) in a data frame

`filter()`: subset *rows* in a data frame

select() columns

To **keep** columns, simply name them

```
dat_eco |>
  select(plot_id, species_id)
```

```
## # A tibble: 34,786 × 2
##    plot_id species_id
##      <dbl> <chr>
##  1       2 NL
##  2       2 NL
##  3       2 NL
##  4       2 NL
##  5       2 NL
##  6       2 NL
##  7       2 NL
##  8       2 NL
##  9       2 NL
## 10       2 NL
## # ℹ 34,776 more rows
```

select() columns

To **drop** columns, use a minus sign (–)

```
dat_eco |>
  select(-plot_id, -species_id)
```

```
## # A tibble: 34,786 × 11
##    record_id month   day  year sex   hindfoot_length weight genus   species  taxa   plot_type
##        <dbl> <dbl> <dbl> <dbl> <chr>           <dbl>  <dbl> <chr>   <chr>    <chr>  <chr>
##  1         1     7    16  1977 M                  32     NA Neotoma albigula Rodent Control
##  2        72     8    19  1977 M                  31     NA Neotoma albigula Rodent Control
##  3       224     9    13  1977 <NA>               NA     NA Neotoma albigula Rodent Control
##  4       266    10    16  1977 <NA>               NA     NA Neotoma albigula Rodent Control
##  5       349    11    12  1977 <NA>               NA     NA Neotoma albigula Rodent Control
##  6       363    11    12  1977 <NA>               NA     NA Neotoma albigula Rodent Control
##  7       435    12    10  1977 <NA>               NA     NA Neotoma albigula Rodent Control
##  8       506     1     8  1978 <NA>               NA     NA Neotoma albigula Rodent Control
##  9       588     2    18  1978 M                  NA    218 Neotoma albigula Rodent Control
## 10       661     3    11  1978 <NA>               NA     NA Neotoma albigula Rodent Control
## # ℹ 34,776 more rows
```

filter() rows

Use **relational expressions** to filter rows

- == equal
- != does not equal
- < less than
- > greater than
- <= less than or equal to
- >= greater than or equal to
- || or
- && and

filter() rows

Use **relational expressions** to filter rows

```
dat_eco |>
  filter(year == 1995)
```

```
## # A tibble: 1,180 × 13
##    record_id month   day  year plot_id species_id sex   hindfoot_length weight genus     species  taxa   plot_type
##        <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr>     <chr>    <chr>  <chr>
##  1     22314     6     7  1995       2 NL         M                  34     NA Neotoma   albigula Rodent Control
##  2     22728     9    23  1995       2 NL         F                  32    165 Neotoma   albigula Rodent Control
##  3     22899    10    28  1995       2 NL         F                  32    171 Neotoma   albigula Rodent Control
##  4     23032    12     2  1995       2 NL         F                  33     NA Neotoma   albigula Rodent Control
##  5     22003     1    11  1995       2 DM         M                  37     41 Dipodomys merriami Rodent Control
##  6     22042     2     4  1995       2 DM         F                  36     45 Dipodomys merriami Rodent Control
##  7     22044     2     4  1995       2 DM         M                  37     46 Dipodomys merriami Rodent Control
##  8     22105     3     4  1995       2 DM         F                  37     49 Dipodomys merriami Rodent Control
##  9     22109     3     4  1995       2 DM         M                  37     46 Dipodomys merriami Rodent Control
## 10     22168     4     1  1995       2 DM         M                  36     48 Dipodomys merriami Rodent Control
## # ℹ 1,170 more rows
```

filter() rows

Use **relational expressions** to filter rows

```
dat_eco |>
  filter(year == 1995, month == 12)
```

```
## # A tibble: 218 × 13
##    record_id month   day  year plot_id species_id sex   hindfoot_length weight genus     species  taxa   plot_type
##        <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr>     <chr>    <chr>  <chr>
##  1     23032    12     2  1995       2 NL         F                  33     NA Neotoma   albigula Rodent Control
##  2     22997    12     2  1995       2 DM         M                  36     50 Dipodomys merriami Rodent Control
##  3     23002    12     2  1995       2 DM         M                  35     40 Dipodomys merriami Rodent Control
##  4     23003    12     2  1995       2 DM         F                  35     43 Dipodomys merriami Rodent Control
##  5     23041    12     2  1995       2 DM         M                  36     47 Dipodomys merriami Rodent Control
##  6     23044    12     2  1995       2 DM         M                  37     43 Dipodomys merriami Rodent Control
##  7     23115    12    21  1995       2 DM         F                  37     51 Dipodomys merriami Rodent Control
##  8     23117    12    21  1995       2 DM         M                  35     23 Dipodomys merriami Rodent Control
##  9     23120    12    21  1995       2 DM         M                  35     42 Dipodomys merriami Rodent Control
## 10     23136    12    21  1995       2 DM         M                  33     27 Dipodomys merriami Rodent Control
## # ℹ 208 more rows
```

select() and filter()

Chain chain chaaaain

Extract the `species_id`, `sex`, and `weight` of animals < 5 g

select() and filter()

Chain chain chaaaain

Extract the `species_id`, `sex`, and `weight` of animals < 5 g

```
dat_eco |>
  filter(weight < 5) |>
  select(species_id, sex, weight)
```

```
## # A tibble: 17 × 3
##    species_id sex   weight
##    <chr>      <chr>  <dbl>
##  1 PF         F          4
##  2 PF         F          4
##  3 PF         M          4
##  4 RM         F          4
##  5 RM         M          4
##  6 PF         <NA>       4
##  7 PP         M          4
##  8 RM         M          4
##  9 RM         M          4
## 10 RM         M          4
## 11 PF         M          4
## 12 PF         F          4
## 13 RM         M          4
```

mutate()

Definition

`mutate()`: create and add new variables to a data.frame, while preserving existing ones (by default, though optional)

mutate()

Let's convert our `weights` (in g) to kg:

```
dat_eco |>
  mutate(weight_kg = weight / 1000) |>
  head()
```

```
## # A tibble: 6 × 4
##   genus   species  weight weight_kg
##   <chr>   <chr>     <dbl>     <dbl>
## 1 Neotoma albigula    218     0.218
## 2 Neotoma albigula    204     0.204
## 3 Neotoma albigula    200     0.2
## 4 Neotoma albigula    199     0.199
## 5 Neotoma albigula    197     0.197
## 6 Neotoma albigula    218     0.218
```

mutate()

We can also build on columns created *within* `mutate`:

```
dat_eco |>
  mutate(weight_kg = weight / 1000,
         weight_lb = weight_kg * 2.2) |>
  head()
```

```
## # A tibble: 6 × 5
##   genus   species  weight weight_kg weight_lb
##   <chr>   <chr>     <dbl>     <dbl>     <dbl>
## 1 Neotoma albigula    218     0.218     0.480
## 2 Neotoma albigula    204     0.204     0.449
## 3 Neotoma albigula    200     0.2       0.44
## 4 Neotoma albigula    199     0.199     0.438
## 5 Neotoma albigula    197     0.197     0.433
## 6 Neotoma albigula    218     0.218     0.480
```

summarise and group_by

summarise() and group_by()

`summarise()`: create a new data.frame containing specified summary data (by group, if desired) from another data.frame

`group_by()`: group your data by a variable or variables

summarise() and group_by()

Many tasks require a *split-apply-combine* strategy:

1. *split* the data into groups
2. *apply* some analysis to each group
3. *combine* the results

This is what the **group_by()** and **summarise()** functions make easy!

summarise()

Let's start by calculating the mean & sd weight across all entries

```
dat_eco |>
  summarise(mean_weight = mean(weight, na.rm = TRUE),
            sd_weight = sd(weight, na.rm = TRUE))
```

```
## # A tibble: 1 × 2
##   mean_weight sd_weight
##         <dbl>     <dbl>
## 1        42.7      36.6
```

summarise() with group_by()

But what if we want to do this *by group*? Lets do the same again, but grouped by **sex**

```
dat_eco |>
  group_by(sex) |>
  summarise(mean_weight = mean(weight, na.rm = TRUE),
            sd_weight = sd(weight, na.rm = TRUE))
```

```
## # A tibble: 3 × 3
##   sex   mean_weight sd_weight
##   <chr>       <dbl>     <dbl>
## 1 F            42.2      36.8
## 2 M            43.0      36.2
## 3 <NA>         64.7      62.2
```

summarise() with group_by()

We can also specify **multiple groups** to get every pairwise combination

```
dat_eco |>
  group_by(sex, species_id) |>
  summarise(mean_weight = mean(weight, na.rm = TRUE),
            sd_weight = sd(weight, na.rm = TRUE))
```

```
## # A tibble: 6 × 4
## # Groups:   sex [1]
##   sex   species_id mean_weight sd_weight
##   <chr> <chr>            <dbl>     <dbl>
## 1 F     BA                9.16      2.24
## 2 F     DM               41.6       6.74
## 3 F     DO               48.5       8.24
## 4 F     DS              118.       21.5
## 5 F     NL              154.       39.2
## 6 F     OL               31.1       6.78
```

Writing & tying it all together

write_csv()

We've read & wrangled our data, let's close the loop and write it too.

Let's create a tidied-up dataset that doesn't include any **missing data**:

```
dat_eco_clean <-
  dat_eco |>
    drop_na()  # remove any rows containing missing data
```

write_csv()

Have a quick look at it

```
head(dat_eco_clean)
```

```
## # A tibble: 6 × 13
##   record_id month   day  year plot_id species_id sex   hindfoot_length weight genus   species  taxa   plot_type
##       <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl> <chr>   <chr>    <chr>  <chr>
## 1       845     5     6  1978       2 NL         M                  32    204 Neotoma albigula Rodent Control
## 2      1164     8     5  1978       2 NL         M                  34    199 Neotoma albigula Rodent Control
## 3      1261     9     4  1978       2 NL         M                  32    197 Neotoma albigula Rodent Control
## 4      1756     4    29  1979       2 NL         M                  33    166 Neotoma albigula Rodent Control
## 5      1818     5    30  1979       2 NL         M                  32    184 Neotoma albigula Rodent Control
## 6      1882     7     4  1979       2 NL         M                  32    206 Neotoma albigula Rodent Control
```

write_csv()

And save it to a file called `dat_eco_cleaned.csv`

```
write_csv(dat_eco_clean, file = "dat_eco_cleaned.csv")
```

Done!

Outline

- Describe the structure of tidy data, and understand how to load and save it in R
- Understand the function and operation of key `dplyr` verbs
- Flexibly and elegantly manipulate real-world data using piped commands (in R)

Thanks!