Week 3: Introduction to the Tidyverse

01/28/2020

Jake Campbell

What is the Tidyverse?

- The tidyverse is a set of packages developed to make the data science process easier
  - They share a common philosophy and grammar
  - Intended to make code more understandable
- Several different people contributing code can lead to confusion and overlapping ideas
- Tidyverse packages have different purposes, but read easily into each other

Download the Packages

- The tidyverse consists of several different packages
- We can download them all at once, though
  - Just call `install.packages("tidyverse")`

Data Manipulation with `dplyr`

- `dplyr` has several functions for manipulating data frames
  - Note that they output tibbles rather than data frames
- We can chain functions together using `%>%`
  - Also known as the pipe operator
  - Note that if for some reason we needed to refer back to the tibble we are manipulating, we would specify it with a `.`

```
# Format would look like this

tibble_x %>%
  function_x() %>%
  function_y() %>%
  function_z()
```

Some Common `dplyr` Functions

- `select()`: pick certain variables
- `filter()`: filter your data on a given condition
- `arrange()`: order your data
- `rename()`: rename your columns
- `mutate()`: create a new column

select

- With `select`, we just specify the variables we want to choose
  - We can instead specify what columns we don't want by putting a `-` before the column

```
initial.starwars %>%
  # Select height and birth year of all characters
  select(height, birth_year) %>%
  head()
```

```
## # A tibble: 6 x 2
##   height birth_year
##    <int>      <dbl>
## 1    172       19
## 2    167      112
## 3     96       33
## 4    202       41.9
## 5    150       19
## 6    178       52
```

filter

- With `filter`, we subset the tibble on some condition
  - `==` means equal to and `!=` means not equal to

```
initial.starwars %>%
  # Filter on only males
  filter(gender == "male") %>%
  head()
```

```
## # A tibble: 6 x 10
##   name  height  mass hair_color skin_color eye_color birth_year gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
## 1 Luke~    172    77 blond      fair       blue            19   male
## 2 Dart~    202   136 none       white      yellow          41.9 male
## 3 Owen~    178   120 brown, gr~ light      blue            52   male
## 4 Bigg~    183    84 black      light      brown           24   male
## 5 Obi-~    182    77 auburn, w~ fair       blue-gray       57   male
## 6 Anak~    188    84 blond      fair       blue            41.9 male
## # ... with 2 more variables: homeworld <chr>, species <chr>
```

arrange

- With `arrange`, we order the tibble by some column
  - By default, order is ascending; to get descending order, we use `desc()`

```
initial.starwars %>%
  # Arrange by descending height
  arrange(desc(height)) %>%
  head()
```

```
## # A tibble: 6 x 10
##   name  height  mass hair_color skin_color eye_color birth_year gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr>
## 1 Yara~    264    NA none       white      yellow             NA male
## 2 Tarf~    234   136 brown      brown      blue               NA male
## 3 Lama~    229    88 none       grey       black              NA male
## 4 Chew~    228   112 brown      unknown    blue              200 male
## 5 Roos~    224    82 none       grey       orange             NA male
## 6 Grie~    216   159 none       brown, wh~ green, y~          NA male
## # ... with 2 more variables: homeworld <chr>, species <chr>
```

rename

- `rename` allows us to change the name of a column
  - First specify the new name, than specify what column is being renamed

```
initial.starwars %>%
  # Rename name to full_name
  rename(full_name = name) %>%
  head()
```

```
## # A tibble: 6 x 10
##   full_name height  mass hair_color skin_color eye_color birth_year gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr>
## 1 Luke Sky~    172    77 blond      fair       blue             19   male
## 2 C-3PO        167    75 <NA>       gold       yellow          112   <NA>
## 3 R2-D2         96    32 <NA>       white, bl~ red              33   <NA>
## 4 Darth Va~    202   136 none       white      yellow           41.9 male
## 5 Leia Org~    150    49 brown      light      brown            19   female
## 6 Owen Lars    178   120 brown, gr~ light      blue             52   male
## # ... with 2 more variables: homeworld <chr>, species <chr>
```

mutate

- `mutate` allows us to create new columns
    - Remember to save your output if you want to keep it!

```
new.starwars <- initial.starwars %>%
  # Creating a new column that is the square root of height
  mutate(sqrt_height = sqrt(height))

new.starwars %>%
  select(height, sqrt_height) %>%
  head(3)
```

```
## # A tibble: 3 x 2
##   height sqrt_height
##    <int>       <dbl>
## 1    172       13.1
## 2    167       12.9
## 3     96        9.80
```

summarize

- With `summarize`, we can perform certain summary statistics, like finding the average, or getting a count

```
initial.starwars %>%
  # Here we are finding the average height of all characters
  summarize(avg.height = mean(height, na.rm=T))
```

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

group_by

- group_by allows us to perform different operations by groups
  - If we want to end the grouping, use ungroup()

```
initial.starwars %>%
  # Grouping by gender
  group_by(gender) %>%
  # Finding average height
  summarize(avg.height = mean(height, na.rm=T)) %>%
  # Remember to ungroup if you want to perform non-grouped functions
  # after grouping
  ungroup()
```

```
## # A tibble: 5 x 2
##   gender        avg.height
##   <chr>              <dbl>
## 1 female              165.
## 2 hermaphrodite       175
## 3 male                179.
## 4 none                200
## 5 <NA>                120
```

Chaining Functions

- With the %>% operator, we can easily chain several functions together

```
initial.starwars %>%
  # Filter characters taller than 200 cm
  filter(height > 200) %>%
  # Select the mass column
  select(mass) %>%
  # Order mass in descending order
  arrange(desc(mass))
```

```
## # A tibble: 10 x 1
##     mass
##    <dbl>
##  1   159
##  2   136
##  3   136
##  4   112
##  5    88
##  6    82
##  7    80
##  8    NA
```

Joining Data

- We can use several functions that work similarly to `SQL` joins
- `inner_join`: return all rows from `x` where there are matching values in `y`
- `left_join`: return all rows from `x`
- `right_join`: return all rows from `y`
- `full_join`: return all rows from `x` and `y`

Reshaping Data

- Data generally comes in wide and long format
- Wide format makes every variable a column
- Long format puts multiple variables in one column (key) and their values in a second column
- Consider the goal of your analysis when choosing which format

Reshaping with `tidyr`

- `tidyr` allows us to easily go from long to wide or vice-versa
- The main functions we'll use are `gather()` and `spread()`
    - These change data sets into long and wide formats, respectively

Going from Wide to Long

- We'll use `gather()` to go from wide to long
- We need to first specify what columns we are gathering
  - The names of the columns will become our key, their values become our value
  - We need to also specify the new `key` and `value` column names

```
long.tb <- gather(Year1, Year2, data = initial.tb,
                  key = Year, value = Cases)
long.tb
```

```
## # A tibble: 6 x 3
##   country     Year   Cases
##   <chr>       <chr>  <int>
## 1 Afghanistan Year1    745
## 2 Brazil      Year1  37737
## 3 China       Year1 212258
## 4 Afghanistan Year2   2666
## 5 Brazil      Year2  80488
## 6 China       Year2 213766
```

Going from Long to Wide

- We'll use `spread()` to go from long to wide
- We need to first specify where the data is coming from, followed by the key and value columns

```
spread(data = long.tb, key = Year, value = Cases)
```

```
## # A tibble: 3 x 3
##   country      Year1  Year2
##   <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```