Key tidyverse verbs to accelerate data analysis

Amy Willis

Friday 21 January 2021

Who am I?

I'm Amy Willis, PhD; I am a tenure-track Assistant Professor in Biostatistics

- Principal Investigator, Statistical Diversity Lab
  - Methods for microbiome data analysis
- PhD in Statistics, Cornell University
- I <3 statistics and data analysis; I've worked for Google, Australian Government, macroeconomic forecasting consulting, biotech...
- 10+ years of `R` experience
  - I'm a methods developer; most of my methods are coded in `R`
  - packages: `breakaway`, `DivNet`, `corncob`, `paramedic`...

Please call me *Amy*; I use she/her pronouns

My favourite toolbox for data analysis

```
> library(tidyverse)
```

My favourite tools

`tidyverse` functions that I use all the time:

- `summarize()` reduces multiple values to a single summary
- `group_by()` groups rows together (useful for `summarize`)
- `filter()` picks rows based on their valuess
- `select()` picks columns based on their names
- `arrange()` changes the ordering of the rows

`tidyverse` functions that I learnt *more recently* and now use **a lot**:

- `pivot_longer()` and `pivot_wider()`
- `across()`

Reading in the data

I prefer `read_csv` because it gives me a *tibble* - a clever data frame

```
> df <- read_csv("TESTIntegratedCaseIn_DATA_2021-01-21_for Amy.csv")

-- Column specification --------------------------------------------
cols(
  .default = col_character(),
  record_id = col_double(),
  translator_needed = col_double(),
  fever_any = col_double(),
  fever_temp_taken = col_double(),
  fever_temp_measure = col_double(),
  sx_headache = col_double(),
  sx_myalgia = col_double(),
  sx_congestion = col_double(),
  sx_pharyngitis = col_double(),
  sx_cough = col_double(),
  sx_dyspnea = col_double(),
  sx_pneumonia = col_double(),
  sx_nausea = col_double(),
  sx_vomiting = col_double(),
```

Looking at the data

Using *tibbles* helps me avoid endless scrolling up

```
> df
```

```
# A tibble: 924 x 226
   record_id remote_intervie~ specimen_collec~ translator_need~ languag
       <dbl> <chr>            <chr>                       <dbl> <chr>
 1 100924894 5/1/2020 14:11   5/1/2020                        1 ENGLISH
 2 100924893 5/1/2020 14:15   5/1/2020                       NA ENGLISH
 3 100924501 5/1/2020 14:44   4/20/2020                       1 ENGLISH
 4 100924671 5/1/2020 14:58   4/15/2020                       1 ENGLISH
 5 100924678 5/1/2020 14:58   4/23/2020                       1 ENGLISH
 6 100801703 5/1/2020 14:59   4/21/2020                       1 ENGLISH
 7 100683673 5/1/2020 14:59   4/28/2020                       1 ENGLISH
 8 100924499 5/1/2020 15:00   4/22/2020                       1 ENGLISH
 9 100924576 5/1/2020 15:01   5/24/2020                       1 ENGLISH
10 100924830 5/1/2020 15:04   4/23/2020                      NA ENGLISH
# ... with 914 more rows, and 221 more variables: startexposurenoill <c
#   endexposurenoill <chr>, contagiousnoill <chr>, asymptom_cleared <ch
#   fever_any <dbl>, fever_temp_taken <dbl>, fever_temp_measure <dbl>,
#   sx_headache <dbl>, sx_myalgia <dbl>, sx_congestion <dbl>,
#   sx_pharyngitis <dbl>, sx_cough <dbl>, sx_dyspnea <dbl>, sx_pneumoni
```

Looking at the data: global view

The pipe %>%: the glue that holds `tidyverse` functions together. Strong recommend for accelerating your data analysis!

```
> # names(df) # alternative
> df %>% names
```

```
  [1] "record_id"
  [2] "remote_interviewer_timestamp"
  [3] "specimen_collection"
  [4] "translator_needed"
  [5] "language"
  [6] "startexposurenoill"
  [7] "endexposurenoill"
  [8] "contagiousnoill"
  [9] "asymptom_cleared"
 [10] "fever_any"
 [11] "fever_temp_taken"
 [12] "fever_temp_measure"
 [13] "sx_headache"
 [14] "sx_myalgia"
 [15] "sx_congestion"
```

Looking at the data: global view

```
> # head(names(df), 15) # alternative
> df %>% names %>% head(15)
```

```
 [1] "record_id"                    "remote_interviewer_timestamp"
 [3] "specimen_collection"          "translator_needed"
 [5] "language"                     "startexposurenoill"
 [7] "endexposurenoill"             "contagiousnoill"
 [9] "asymptom_cleared"             "fever_any"
[11] "fever_temp_taken"             "fever_temp_measure"
[13] "sx_headache"                  "sx_myalgia"
[15] "sx_congestion"
```

Looking at my data: local view

```
> df %>%
+   group_by(fever_any) %>%
+   summarise(n())
```

```
# A tibble: 4 x 2
  fever_any `n()`
      <dbl> <int>
1         0   197
2         1   322
3       100     2
4        NA   403
```

Cleaning my data

```
> df %>%
+   filter(fever_any == 100) %>%
+   select(remote_interviewer_timestamp, investigator)
```

```
# A tibble: 2 x 2
  remote_interviewer_timestamp investigator
  <chr>                               <dbl>
1 5/4/2020 12:28                       8888
2 5/20/2020 16:49                       111
```

Let's not forget this going forward!

Looking at my data: larger view

Do you have columns of the same type named similarly? Awesome!

```
> df %>%
+   select(starts_with("sx"))
```

```
# A tibble: 924 x 22
   sx_headache sx_myalgia sx_congestion sx_pharyngitis sx_cough sx_dysp
         <dbl>      <dbl>         <dbl>           <dbl>    <dbl>    <d
 1          NA         NA            NA              NA       NA
 2          NA         NA            NA              NA       NA
 3           0          0             0               0        1
 4           0          0             0               0        0
 5           0          0             0               0        1
 6           0          0             0               0        1
 7           1          0             0               0        0
 8           0          1             0               0        1
 9           0          0             0               0        1
10          NA         NA            NA              NA       NA
# ... with 914 more rows, and 16 more variables: sx_pneumonia <dbl>,
#   sx_nausea <dbl>, sx_vomiting <dbl>, sx_diarrhea <dbl>, sx_abdominal
#   sx_dysgeusia <dbl>, sx_anosmia <dbl>, sx_other <dbl>, sx_other_det
```

Which symptoms were most commonly observed?

How could we answer this?

```
> df %>%
+   select(starts_with("sx"))
```

```
# A tibble: 924 x 22
   sx_headache sx_myalgia sx_congestion sx_pharyngitis sx_cough sx_dysp
         <dbl>      <dbl>         <dbl>           <dbl>    <dbl>    <d
 1          NA         NA            NA              NA       NA
 2          NA         NA            NA              NA       NA
 3           0          0             0               0        1
 4           0          0             0               0        0
 5           0          0             0               0        1
 6           0          0             0               0        1
 7           1          0             0               0        0
 8           0          1             0               0        1
 9           0          0             0               0        1
10          NA         NA            NA              NA       NA
# ... with 914 more rows, and 16 more variables: sx_pneumonia <dbl>,
#   sx_nausea <dbl>, sx_vomiting <dbl>, sx_diarrhea <dbl>, sx_abdominal
#   sx_dysgeusia <dbl>, sx_anosmia <dbl>, sx_other <dbl>, sx_other_det
```

Which symptoms were most commonly observed?

How could we answer this?

```
> df %>%
+   select(starts_with("sx")) %>%
+   pivot_longer(starts_with("sx"))
```

```
Error: Can't combine `sx_headache` <double> and `sx_other_det` <charact
```

Oh no!

What's going on?

```
> df$sx_other_det %>% unique
```

```
 [1] NA
 [2] "loose stools"
 [3] "overall weakness and body aches"
 [4] "Sneezing"
 [5] "weakness, low energy"
 [6] "Back ache & feet were tingling"
 [7] "very tired"
 [8] "Lungs felt on fire"
 [9] "Fatigue"
[10] "aching her body/and legs"
[11] "Loss of appetite; fatigue"
[12] "fatigue"
[13] "chills"
[14] "Nose bleed, belly distension, tears"
[15] "Trench Foot"
[16] "Not wanting to eat"
[17] "Seasonal Allergies"
[18] "Soreness in face"
[19] "Farts"
```

Pivoting: wide to long

```
> df %>%
+   select(starts_with("sx")) %>%
+   select(where(is.numeric)) %>%
+   pivot_longer(starts_with("sx"))
```

```
# A tibble: 15,708 x 2
   name            value
   <chr>           <dbl>
 1 sx_headache        NA
 2 sx_myalgia         NA
 3 sx_congestion      NA
 4 sx_pharyngitis     NA
 5 sx_cough           NA
 6 sx_dyspnea         NA
 7 sx_pneumonia       NA
 8 sx_nausea          NA
 9 sx_vomiting        NA
```

Which symptoms were most commonly observed?

Now, grab only entries that are neither zero nor NA

```
> df %>%
+   select(starts_with("sx")) %>%
+   select(where(is.numeric)) %>%
+   pivot_longer(starts_with("sx")) %>%
+   filter(!is.na(value) & value != 0)
```

```
# A tibble: 3,072 x 2
   name          value
   <chr>         <dbl>
 1 sx_cough          1
 2 sx_dysgeusia      1
 3 sx_anosmia        1
 4 sx_yesno          1
 5 sx_ongoing        1
 6 sx_nausea         1
 7 sx abdominal      1
```

Which symptoms were most commonly observed?

Then tabulate: group_by, summarise, arrange

```
> df %>%
+   select(starts_with("sx")) %>%
+   select(where(is.numeric)) %>%
+   pivot_longer(starts_with("sx")) %>%
+   filter(!is.na(value) & value != 0) %>%
+   group_by(name) %>%
+   summarise(n = n()) %>%
+   arrange(desc(n))
```

```
`summarise()` ungrouping output (override with `.groups` argument)

# A tibble: 17 x 2
   name                n
   <chr>           <int>
 1 sx_yesno          486
 2 sx_cough          347
 3 sx_headache       256
 4 sx_congestion     195
 5 sx_dysgeusia      189
 6 sx_pharyngitis    187
```

Clean up

How can we quickly change all those 100's to 1's?

Use `mutate` + `across`

```
> change_hundreds <- function(x) ifelse(x == 100, 1, x)
> change_hundreds(98:102)
```

```
[1]  98  99   1 101 102
```

```
> df %>%
+   select(starts_with("sx")) %>%
+   mutate(across(where(is.numeric), change_hundreds))
```

```
# A tibble: 924 x 22
   sx_headache sx_myalgia sx_congestion sx_pharyngitis sx_cough sx_dysp
         <dbl>      <dbl>          <dbl>           <dbl>    <dbl>    <d
 1          NA         NA             NA              NA       NA
 2          NA         NA             NA              NA       NA
 3           0          0              0               0        1
 4           0          0              0               0        0
 5           0          0              0               0        1
 6           0          0              0               0        1
```

How can we find which investigators set those 100's?

```
> df %>%
+   select(investigator,
+          starts_with("sx") & where(is.numeric)) %>%
+   pivot_longer(!investigator) %>%
+   filter(!is.na(value) & value != 0 & value != 1)
```

```
# A tibble: 76 x 3
   investigator name           value
          <dbl> <chr>          <dbl>
 1           32 sx_pneumonia     100
 2           96 sx_dysgeusia     100
 3           96 sx_anosmia       100
 4            9 sx_cough         100
 5           10 sx_anosmia       100
 6           85 sx_congestion    100
 7           68 sx_congestion    100
 8           30 sx_pharyngitis   100
 9         8888 sx_headache      100
10         8888 sx_myalgia       100
# ... with 66 more rows
```

How can we find which investigators set those 100's?

```
> df %>%
+   select(investigator, starts_with("sx") & where(is.numer
+   pivot_longer(!investigator) %>%
+   filter(!is.na(value) & value != 0 & value != 1) %>%
+   group_by(investigator) %>%
+   summarise(n = n()) %>%
+   arrange(desc(n))
```

```
# A tibble: 36 x 2
   investigator     n
          <dbl> <int>
 1           22     9
 2         8888     5
 3          104     4
 4          106     4
 5          131     4
 6           12     3
```

Scaling up

There is no need to memorise all of this!

Can't remember how to use `across`? Just type `?across`

My cheat list

- `mutate(across(everything(), fun_name))`
- `summarise(across(starts_with("sx"), fun_name))`
- `mutate(across(contains("Name"), fun_name))`
- `summarise(across(where(is.character), fun_name))`
- `mutate(across(FirstCol:LastCol, fun_name))`

Recap

`tidyverse` functions that I use all the time:

- `summarize()` reduces multiple values to a single summary
- `group_by()` groups rows together (useful for `summarize`)
- `filter()` picks rows based on their valuess
- `select()` picks columns based on their names
- `arrange()` changes the ordering of the rows

`tidyverse` functions that I learnt *more recently* and now use **a lot**:

- `pivot_longer()` and `pivot_wider()`
- `across()`

Wrap up

- I'm Amy Willis, an Assistant Professor in Biostatistics at the University of Washington in the School of Public Health
- I can do many things in `R` very quickly, and I want you to as well!
- This took years of practice... and a few key verbs!
- **I am available for consulting** – feel free to reach out at adwillis@uw.edu
- Slides available at github.com/adw96/presentations
- Keep in touch by e-mail (adwillis@uw.edu) or Twitter (@AmyDWillis)!

Key tidyverse verbs to accelerate data analysis

Amy Willis

Friday 21 January 2021