DATA 607 Tidyverse Extend - Weyrich

Lu Beyer

04-17-2024

For my Tidyverse Extend, I am going to show some additional Tidyr functions that I have found to be helpful.

We start that same way Lucas has, by importing our libraries and soccer data

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidyr)

spi <-  read_csv('https://raw.githubusercontent.com/lucasweyrich958/DATA607/main/spi_matches.csv')
```

```
## Rows: 68913 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (3): league, team1, team2
## dbl  (19): season, league_id, spi1, spi2, prob1, prob2, probtie, proj_score1...
## date  (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We then filter our large dataset to something more manageable, games only in the year 2022.

```
cl_2022 <-  spi %>%
    filter(season == 2022)

head(cl_2022, 5)
```

```
## # A tibble: 5 x 23
##   season date       league_id league team1 team2  spi1  spi2 prob1 prob2 probtie
##    <dbl> <date>         <dbl> <chr>  <chr> <chr> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1   2022 2022-02-18      1947 Japan~ Kawa~ FC T~  58.2  35.3 0.634 0.166   0.2
## 2   2022 2022-02-18      1947 Japan~ Shim~ Cons~  28.0  31.8 0.354 0.394   0.252
## 3   2022 2022-02-18      1947 Japan~ Gamb~ Kash~  28.5  50.0 0.198 0.584   0.218
## 4   2022 2022-02-18      1947 Japan~ Avis~ Jubi~  31.0  24.0 0.479 0.262   0.259
## 5   2022 2022-02-18      1947 Japan~ Yoko~ Cere~  52.8  35.2 0.584 0.210   0.207
## # i 12 more variables: proj_score1 <dbl>, proj_score2 <dbl>, importance1 <dbl>,
## #   importance2 <dbl>, score1 <dbl>, score2 <dbl>, xg1 <dbl>, xg2 <dbl>,
## #   nsxg1 <dbl>, nsxg2 <dbl>, adj_score1 <dbl>, adj_score2 <dbl>
```

We then find a column of interest, importance1

```
cl_2022 <- cl_2022 %>%
    arrange(desc(importance1))
```

We note that all values aren't complete, and we have many NA values.

```
tail(cl_2022$importance1, 5)
```

```
## [1] NA NA NA NA NA
```

One way we may fix this is by ensuring the column is ordered by $importance1, descending. Then we use Tidyr to fill it downwards. This takes the last "real" value before the NA values, and applies it to everything until it reaches another valid entry. Because we arranged the column, it will fill all the NAs until the end of the dataset.

```
a <- cl_2022 %>%
    arrange(desc(importance1)) %>%
    fill(importance1, .direction = "down")

tail(a$importance1, 5)
```

```
## [1] 0 0 0 0 0
```

If we know what value we want to replace our data with, we can also use Tidyr to replace NA values specifically with replace_na(). In this case, we'll use the average value of $importance2

```
a <- a %>%
    arrange(desc(importance2)) %>%
    mutate(importance2 = replace_na(importance2, mean(importance2, na.rm = TRUE)))
```

Since we did not rearrange the data after filling, we can see that all of the former NA values were replaced with the average value, 32.122.

```
tail(a$importance2, 5)
```

```
## [1] 32.122 32.122 32.122 32.122 32.122
```

Another interesting Tidyr function involves nesting. First we will narrow down the relevant field for easier viewing.

```
b <- a %>%
    select(date, league, team1, team2, score1, score2)

head(b, 5)
```

```
## # A tibble: 5 x 6
##   date       league              team1             team2           score1 score2
##   <date>     <chr>               <chr>             <chr>            <dbl>  <dbl>
## 1 2022-05-02 Swedish Allsvenskan Hammarby          Malmo FF             0      0
## 2 2022-05-04 NWSL Challenge Cup  Seattle Reign FC  Washington Spi~      0      0
## 3 2022-05-04 NWSL Challenge Cup  Kansas City       North Carolina~      1      2
## 4 2022-05-07 NWSL Challenge Cup  North Carolina FC Washington Spi~      2      1
## 5 2022-07-24 Swedish Allsvenskan BK Hacken         Djurgardens IF       1      2
```

We can then group our data specifically by league, and the nest all of the data according to their respective league.

```
b <- b %>%
    group_by(league) %>%
    nest()

head(b, 5)
```

```
## # A tibble: 5 x 2
## # Groups:   league [5]
##   league                data
##   <chr>                 <list>
## 1 Swedish Allsvenskan   <tibble [240 x 5]>
## 2 NWSL Challenge Cup    <tibble [39 x 5]>
## 3 Brasileiro Série A    <tibble [380 x 5]>
## 4 Scottish Premiership  <tibble [228 x 5]>
## 5 Norwegian Tippeligaen <tibble [240 x 5]>
```

Nesting our data can allow us to analyze our individual nests separately or export our data to separate files a bit easier.