D607 - Assignment 3 - Normalization and Reg Exp

Marco Castro

2024-09-21

Overview

For assignment #3, we focused on normalization and regular expressions.

1. Normalization

This section focuses on normalizing dataframes. First I constucted the three dummy dataframes focusing on subset of Major League baseball teams, their respective stadiums and leagues, and the last 4 games (a series) played using the tribble functions.

```
# create division tibble
division <- tribble(
  ~division_id, ~league, ~division,
  "ALE", "American", "East",
  "ALC", "American", "Central",
  "ALW", "American", "West",
  "NLE", "National", "East",
  "NLC", "National", "Central",
  "NLW", "National", "West"
)

# create teams tibble
teams <- tribble(
  ~team_id, ~name, ~division,
  "PHI", "Phillies", "NLE",
  "NYM", "Mets", "NLE",
  "TOR", "Blue Jays", "ALE",
  "TB", "Rays", "ALE",
  "BAL", "Orioles", "ALE",
  "DET", "Tigers", "ALC",
  "WSH", "Nationals", "NLE",
  "CHC", "Cubs", "NLC",
  "CIN", "Reds",  "NLC",
  "PIT", "Pirates",  "NLC"
)

# create stadium tibble
stadium <- tribble(
  ~team_id, ~stadium, ~address, ~city, ~state,
  "PHI", "Citizens Bank Park","1 Citizens Bank Way","Philadelphia","PA",
```

```
  "NYM", "Citi Field", "41 Seaver Wy", "Flushing", "NY",
  "TOR", "Rogers Center","1 Blue Jays Way", "Toronto", "ON",
  "TB", "Tropicana Field","1 Tropicana Dr.","Tampa Bay", "FL",
  "BAL", "Camden Yards","333 W Camden St", "Baltimore", "MD",
  "DET", "Comerica Park","2100 Woodward Ave", "Detroit", "MI",
  "WSH", "Nationals Park", "1500 South Capitol St SE", "Washington", "DC",
  "CHC", "Wrigley Field","1060 W Addison St", "Chicago", "IL",
  "CIN", "Great American Ball Park", "100 Joe Nuxhall Wy", "Cincinnati", "OH",
  "PIT", "PNC Park", "115 Federal St", "Pittsburgh", "PA"
)

# create games tibble
games <- tribble(
  ~game_id, ~date, ~visitor, ~home, ~visitor_score, ~home_score,
  1, "2024-09-18", "PIT", "CIN", 3, 4,
  2, "2024-09-18", "WSH", "CHC", 7, 2,
  3, "2024-09-18", "DET", "BAL", 1, 0,
  4, "2024-09-18", "TOR", "TB", 2, 3,
  5, "2024-09-18", "PHI", "NYM", 4, 3,
  6, "2024-09-19", "PIT", "CIN", 1, 2,
  7, "2024-09-19", "WSH", "CHC", 2, 0,
  8, "2024-09-19", "DET", "BAL", 2, 3,
  9, "2024-09-19", "TOR", "TB", 1, 5,
  10, "2024-09-19", "PHI", "NYM", 8, 10,
  11, "2024-09-20", "PIT", "CIN", 2, 3,
  12, "2024-09-20", "WSH", "CHC", 2, 6,
  13, "2024-09-20", "DET", "BAL", 4, 1,
  14, "2024-09-20", "TOR", "TB", 5, 3,
  15, "2024-09-20", "PHI", "NYM", 2, 12,
  16, "2024-09-21", "PIT", "CIN", 1, 7,
  17, "2024-09-21", "WSH", "CHC", 5, 1,
  18, "2024-09-21", "DET", "BAL", 6, 4,
  19, "2024-09-21", "TOR", "TB", 2, 3,
  20, "2024-09-21", "PHI", "NYM", 3, 6,
)
```

Next I joined the dataframes using left joins. Repeating columns such as team "name" were renamed for clarity of wether a team was the home team hosting the game at their stadium or a visitor.

```
games_in_series <- games |>
  left_join(teams, join_by(home == team_id)) |>
  left_join(teams, join_by(visitor == team_id)) |>
  left_join(stadium, join_by(home == team_id)) |>
  left_join(division, join_by(division.x == division_id)) |>
  left_join(division, join_by(division.y == division_id)) |>
  select(date, home_score, visitor_score, name.x, league.x, division.x, name.y, league.y, division.y, s
  rename(
    home_team = name.x,
    home_league = league.x,
    home_division = division.x,
    visitor_team = name.y,
    visitor_league = league.y,
    visitor_division = division.y,
```

```
    played_at = stadium
  )
```

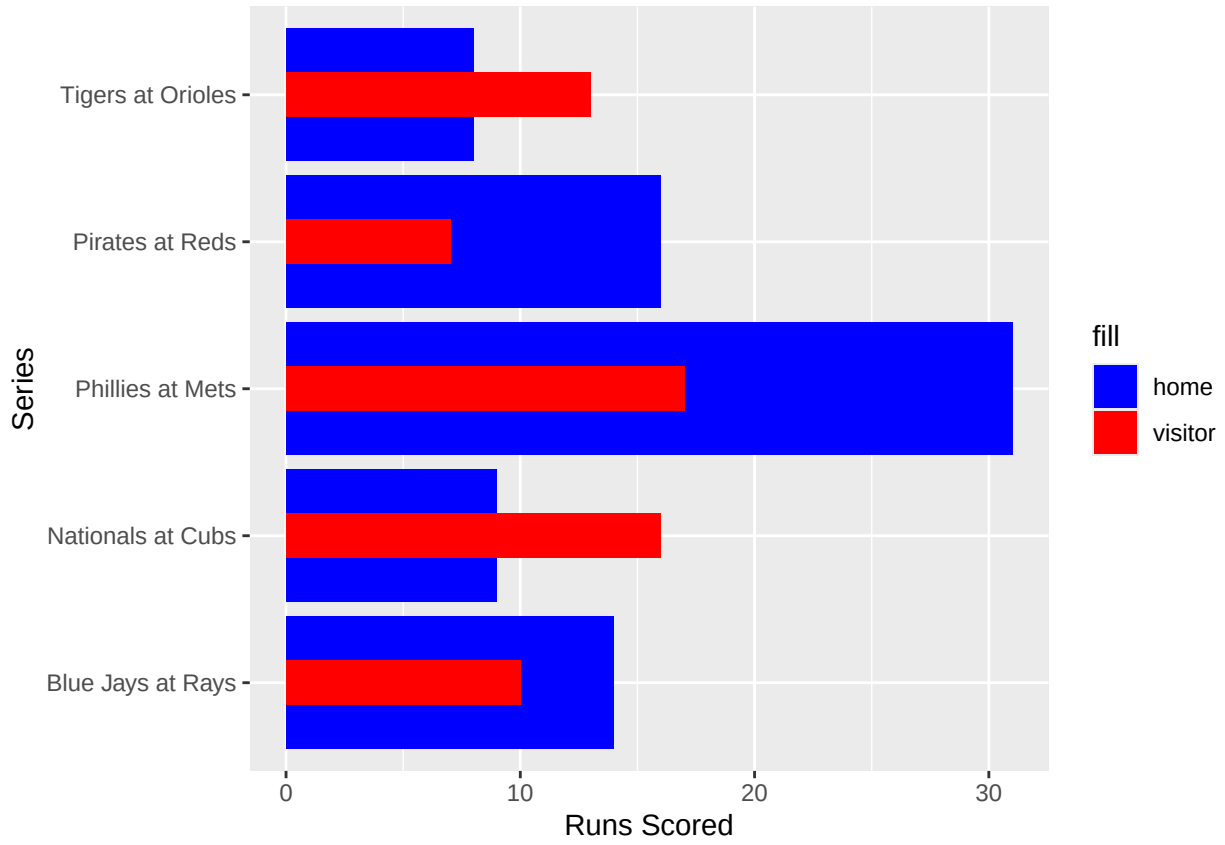
Next, I wanted to

```
runs_scored_in_series <- games_in_series |>
  mutate(teams = paste(visitor_team, home_team, sep=" at ")) |>
  group_by(teams) |>
  summarise(
    runs_home_team = sum(home_score),
    runs_visitor_team = sum(visitor_score)
  )

runs_scored_in_series
```

```
## # A tibble: 5 x 3
##   teams             runs_home_team runs_visitor_team
##   <chr>                      <dbl>             <dbl>
## 1 Blue Jays at Rays             14                10
## 2 Nationals at Cubs              9                16
## 3 Phillies at Mets              31                17
## 4 Pirates at Reds               16                 7
## 5 Tigers at Orioles              8                13
```

```
runs_scored_in_series |>
  ggplot(aes(x=teams)) +
    coord_flip() +
    geom_bar(stat='identity', aes(y=runs_home_team, fill=
                     "blue")) +
    geom_bar(stat='identity', aes(y=runs_visitor_team, width = 0.3, fill=
                     "red")) +
    labs(x="Series", y = 'Runs Scored') +
    scale_fill_manual(values = c("blue","red"),labels = c("home","visitor"))
```



Regular expression

This section focuses on using regular expressions.

2. Character Manipulation

We were asked to download a dataset showing 175 college majors from Five Thirty Eight (https://fivethirtyeight.com/features/the-economic-guide-to-picking-a-college-major/) and identify the majors containing "DATA" or "STATISTICS" in their names.

```
# load data from 538 github page
college_majors <- read_csv("https://raw.githubusercontent.com/fivethirtyeight/data/refs/heads/master/co

# filter df where majors contain "DATA" OR "STATISTICS" using str_detect
college_majors_data_statistics <- college_majors |>
  filter(str_detect(Major, "DATA|STATISTICS"))

# filter df where majors contain "DATA" OR "STATISTICS" using GREPL
college_majors_data_statistics_grepl <- college_majors |>
  filter(grepl("DATA|STATISTICS", Major))
```

Using the function *str_detect* within a filter, I was able to determine that the majors with those keywords are: c("6212", "2101", "3702"), c("MANAGEMENT INFORMATION SYSTEMS AND STATISTICS", "COMPUTER PROGRAMMING AND DATA PROCESSING", "STATISTICS AND DECISION SCIENCE"), c("Business", "Computers & Mathematics", "Computers & Mathematics")

For comparison, I also attempted to filter using grepl and achived the same results c("6212", "2101", "3702"), c("MANAGEMENT INFORMATION SYSTEMS AND STATISTICS", "COMPUTER PROGRAMMING AND DATA PROCESSING", "STATISTICS AND DECISION SCIENCE"), c("Business", "Computers & Mathematics", "Computers & Mathematics")

3. Description of expressions:

We were asked to define the following regular expressions:

```
p1 <- "(.)\1"
p2 <- "(.)(.)\\2\\1"
p3 <- "(..)\1"
p4 <- "(.).\\1.\\1"
p5 <- "(.)(.)(.).*\\3\\2\\1"
```

a: (.)\1\1 Returns true if string contains any characters followed by \1. For example "T\1" or "e\1" in the code block below:

```
test_str <- c('T\1sxded', 'apple\1', 'carrot', '\1brocoli', 'banana')
str_detect(test_str, p1)
```

```
## [1]  TRUE  TRUE FALSE FALSE FALSE
```

b: "(.)(.)\2\1" Returns true if string two characters followed by the same two characters reordered from last to first. For example: "xssx" or "anna" in the code block below:

```
test_str <- c("TsxssxT", 'apple\1', 'carrot', '\1brocoli', 'bananna')
str_detect(test_str, p2)
```

```
## [1]  TRUE FALSE FALSE FALSE  TRUE
```

c: (..)\1 Returns true if string contains two characters followed by \1. For example "Ts\1" or "le1" in the code block below:

```
test_str <- c('TsTs\1sxded', 'apple\1', 'carrot', '\1brocoli', 'banana')
str_detect(test_str, p3)
```

```
## [1]  TRUE  TRUE FALSE FALSE FALSE
```

d: "(.).\1.\1" Returns true if string contains a character followed by any character, repeats the first character followed by any character, then repeats the first character again. For example: "AbAcA" in the code block below:

```
test_str <- c( 'AbAcA', 'apple\1', 'AbAcA', '\1brocoli', 'banana')
str_detect(test_str, p4)
```

```
## [1]  TRUE FALSE  TRUE FALSE  TRUE
```

_e: "(.)(.)(.).*\3\2\1"_ Returns true if string contains three characters (ABC in my example), followed by 0 or more characters, then repeats the first three characters in the matching expression in reverse order. For example: "ABCsomethingCBA" in the code block below or "ABCsCBA, or"ABCCBA."

```
test_str <- c('ABCsomethingCBA', 'apple\1', 'carrot', '\1brocoli', 'banana')
str_detect(test_str, p5)
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE
```

4. Constructing regular expressions

We were asked to construct regular expressions for the following conditions:

a: Start and end with the same character (.).+\1

```
str_detect("xHellox", "(.).+\\1")
```

```
## [1] TRUE
```

b: Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.) (..).+\1

```
str_detect("church", "(..).+\\1")
```

```
## [1] TRUE
```

c: Contain one letter repeated in at least three places (e.g. "eleven" contains three "e"s.)

(.).\1.\1

```
str_detect(c("eleven","bananas"), "(.).\\1.\\1")
```

```
## [1] TRUE TRUE
```

Conclusion

Regular expressions are a powerful tool that will come in handy.