Exploring rugby data

Jocelyn Mara

Overview

This document describes the exploratory analysis conducted on data related to try-scoring in the 2017 Super Rugby competition.

This data consists of tries that were scored during the 2017 Super Rugby competition (observations/rows). Here is a description of the variables:

1. `try_no`: a unique identification number given to each try
2. `round_no`: an identification number to distinguish the round number the try was scored in
3. `attacking_team`: the try-scoring team
4. `defending_team`: the opposition team who conceded the try
5. `attacking_rank`: the final league ranking at the end of the season of the try-scoring team
6. `defending_rank`: the final league ranking at the end of the season of the opposition team
7. `attacking_conference`: the conference group of the try-scoring team
8. `defending_conference`: the conference group of the opposition team
9. `game_time`: the game time in minutes when the try was scored
10. `try_source`: the initial source of possession for the attacking team preceding the try
11. `final_source`: the event that directly preceded the try and resulted in the try being scored
12. `phases`: the total number of phases between gaining possession, and the try being scored (a phase is from one ruck to the next ruck)
13. `time_from_source`: the time taken from gaining possession to scoring the try, in seconds
14. `possession_zone`: the zone on the field the attacking team gained possession of the ball before scoring the try (A = attacking 22m line to try-line, B = halfway to attacking 22m line, C = defensive 22m line to halfway, D =)
15. `offloads`: the number of offloads from gaining possession to the try being scored
16. `passes`: the number of passes from gaining possession to the try being scored
17. `total_passes`: the number of offloads plus passes

This data was collected by a former University of Canberra student, Molly Coughlan, as part of a project that identified playing patterns that led to tries in super rugby[1]

Packages

The following packages will be loaded and used in this analysis:

```
library(tidyverse)
library(naniar)
```

Reading in the data

The data can be read into RStudio and we can examine the structure using the following:

[1] Coughlan, Mountifield, Sharpe & Mara, 2019. How they scored the tries: applying cluster analysis to identify playing patterns that lead to tries in super rugby. IJPAS.

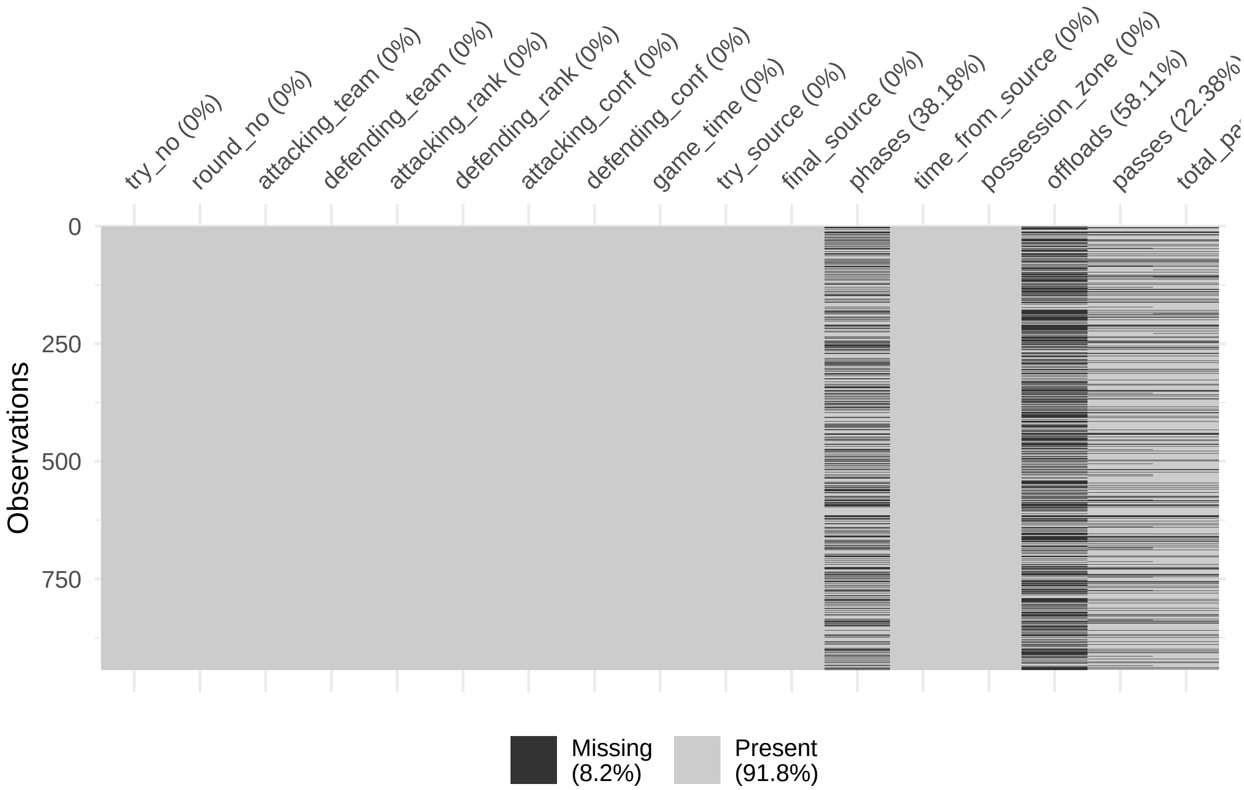
```r
df <- read_csv("data/2017_super-rugby_try-source-data.csv")
str(df)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 943 obs. of  17 variables:
##  $ try_no          : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ round_no        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ attacking_team  : chr  "Blues" "Blues" "Blues" "Blues" ...
##  $ defending_team  : chr  "Rebels" "Rebels" "Rebels" "Rebels" ...
##  $ attacking_rank  : num  9 9 9 9 9 9 9 4 15 15 ...
##  $ defending_rank  : num  18 18 18 18 18 18 18 2 3 3 ...
##  $ attacking_conf  : chr  "NZ" "NZ" "NZ" "NZ" ...
##  $ defending_conf  : chr  "AUS" "AUS" "AUS" "AUS" ...
##  $ game_time       : num  17 27 38 44 51 60 63 41 41 52 ...
##  $ try_source      : chr  "Scrum" "Ruck Turnover" "Intercept" "Lineout" ...
##  $ final_source    : chr  "Multiphase" "Ruck Turnover" "Intercept" "Lineout" ...
##  $ phases          : num  5 NA NA NA 1 5 NA 8 5 1 ...
##  $ time_from_source: num  41 24 9 10 13 54 4 81 48 15 ...
##  $ possession_zone : chr  "A" "C" "B" "B" ...
##  $ offloads        : num  NA 5 NA NA NA NA NA 3 2 1 ...
##  $ passes          : num  1 3 NA NA 2 2 1 19 7 2 ...
##  $ total_passes    : num  1 8 NA NA 2 2 1 22 9 3 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   try_no = col_double(),
##   ..   round_no = col_double(),
##   ..   attacking_team = col_character(),
##   ..   defending_team = col_character(),
##   ..   attacking_rank = col_double(),
##   ..   defending_rank = col_double(),
##   ..   attacking_conf = col_character(),
##   ..   defending_conf = col_character(),
##   ..   game_time = col_double(),
##   ..   try_source = col_character(),
##   ..   final_source = col_character(),
##   ..   phases = col_double(),
##   ..   time_from_source = col_double(),
##   ..   possession_zone = col_character(),
##   ..   offloads = col_double(),
##   ..   passes = col_double(),
##   ..   total_passes = col_double()
##   .. )
```

Checking for missing values

You can check for missing values by using a visualisation such as the `vis_miss()` function from the `naniar` package:

```r
vis_miss(df)
```



Alternatively, you can check how many missing values there are using the following:

```
sum(is.na(df))
```

```
## [1] 1315
```

You can also check which rows and columns the missing values are in using:

```
which(is.na(df), arr.ind = TRUE)
# output not printed here as too long
```

Dealing with missing values

The missing values in this data actually represent 0's. For example, if there were no passes for a try, that was left blank.

We can replace the missing values with 0's using the following:

```
# if there are multiple variables that contain missing values,
# you need to state what to replace NAs with for each one

df <- replace_na(data = df, replace = list(phases = 0,
                                           offloads = 0,
                                           passes = 0,
                                           total_passes = 0))
```

Let's double check that has done the job:

```
sum(is.na(df))
```

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

Exploratory visualisations

References