"2024 League of Ireland in Review"

Learning new skills through the 2024 League of Ireland Premier Division Season

Lorcán Mason

Table of contents

More Resources Available My Website

1 Introduction

The League of Ireland, established in 1921, is the top-level professional football league in the Republic of Ireland, and includes one club from Northern Ireland, Derry City. The 2024 season saw **Shelbourne F.C. win their 14th title**, topping the Premier Division. The Premier Division is the top tier of the League of Ireland, and the season

is structured so that each team plays the other nine teams four times, resulting in 36 games per season. Teams are awarded three points for a win, one for a draw, and zero for a loss. At the end of the season, the bottom-placed team in the Premier Division is relegated, and the winner of the First Division is promoted. The second to fourth-placed teams in the First Division then compete in a play-off series to determine the final team that will be promoted to the Premier Division.

The League of Ireland has a rich history, with the inaugural season in 1921 featuring eight teams from County Dublin. St James's Gate were the first champions. The league then expanded to include clubs from outside of County Dublin. Shamrock Rovers are the team with the most titles overall, having won 21 championships. League of Ireland matches are broadcast by RTÉ and Virgin Media Television. The 2024 season saw an average attendance of 3,490 with the highest attendance being 10,094. The league also has a presence in European competitions with teams participating in the UEFA Champions League, UEFA Europa League, and UEFA Conference League.

This tutorial will provide a basic tutorial on how to create data visualisations of the 2024 League of Ireland season using R. We will also go through the process of creating team profiles from the season. The data used in this tutorial is from the 2024 season and is for ***illustrative purposes only***. There are people much smarter than me who can do much better work, I just like to learn and share new skills - enjoy!

2 Data Preparation

The first step in creating a data visualisation is to prepare the data. The data used in this tutorial is from the 2024 League of Ireland season and is available in a Excel file. The data includes information on the teams statistics from the season. The data is read into R using the `readxl` package, which allows you to read Excel files into R. I will also use the `tidyverse` package, which is a collection of packages designed for data manipulation and visualisation. The `janitor` package is also used to clean the data.

```
# Load the required packages
library(readxl)
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   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     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(janitor)
```

```
Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

First we will read in the data from the Excel file. The data includes information on the teams statistics from the 2024 League of Ireland season. The data is read into R using the `read_excel()` function from the `readxl` package.

```
# Read in the data
all_team <- read_excel("all_team copy.xlsx")
```

Next we will clean the data using the `clean_names()` function from the `janitor` package. This function converts the column names to lower-case and replaces spaces with underscores. We will also check the structure of the data using the `str()` function. This will give us an overview of the data and the variables it contains.

```
# Clean the data
all_team <- all_team %>% clean_names()
```

```
# Check the structure of the data
str(all_team)
```

```
tibble [10 x 24] (S3: tbl_df/tbl/data.frame)
 $ participant_name                    : chr [1:10] "Derry City" "Shamrock Rovers" "Bohemian FC" "St
 $ accurate_cross_per_match_team       : num [1:10] 5.9 5.6 5.1 4.7 4.7 4.3 3.6 3.4 3.4 3.3
 $ accurate_long_balls_per_match_team  : num [1:10] 39.2 32.4 32.3 31.6 30.3 28.4 30.3 26 26.1 32.2
 $ accurate_pass_per_match_team        : num [1:10] 328 409 283 305 164 ...
 $ avg_possession_percentage_team      : num [1:10] 55.9 62.3 52.1 52.6 39.2 48 49.4 38.8 47.7 53.9
 $ big_chance_missed_team              : num [1:10] 40 59 26 23 25 27 19 27 24 22
 $ big_chance_team                     : num [1:10] 65 87 46 49 45 35 43 47 43 45
 $ clean_sheet_team                    : num [1:10] 13 13 8 12 16 9 7 10 12 16
 $ clearance_per_match_team            : num [1:10] 16.7 15.6 21.4 19.8 20.9 19.4 20.9 22.6 23.6 19.
 $ corners_team                        : num [1:10] 211 223 181 183 184 157 169 138 151 157
 $ fotmob_rating_team                  : num [1:10] 6.93 6.87 6.75 6.87 6.82 6.6 6.79 6.64 6.8 6.85
 $ fouls_per_match_team                : num [1:10] 11 12.4 13.6 11.8 12.7 12.6 11.6 11.6 14.1 12.1
 $ goals_conceded_per_match_team       : num [1:10] 0.9 1 1.2 1 0.8 1.4 1.3 1.6 1.4 0.8
 $ goals_per_match_team                : num [1:10] 1.3 1.4 1.1 1.4 0.9 0.6 1.2 1.1 1.1 1.1
```

```
 $ interception_per_match_team          : num [1:10] 7.5 6.9 8.2 7.8 9.8 7.3 8.8 8 9.9 7.5
 $ penalty_conceded_team                : num [1:10] 4 6 3 3 3 5 4 3 7 4
 $ penalty_won_team                     : num [1:10] 7 2 8 5 2 2 3 5 2 6
 $ possession_won_att_3rd_per_match_team: num [1:10] 4.3 4.7 3.6 4.1 4.5 3.2 2.9 4.1 4.4 2.6
 $ red_cards_team                       : num [1:10] 1 2 2 NA 3 3 2 5 5 5
 $ saves_per_match_team                 : num [1:10] 2 2.1 3.2 3 2.3 2.3 3.2 2.4 4 2.5
 $ shots_on_target_per_match_team       : num [1:10] 4.1 5.1 3.9 4.5 3.4 3.3 3.8 3.6 3.6 3.8
 $ tackles_won_per_match_team           : num [1:10] 9.5 10.3 11.4 10 9.2 11.9 11.6 10.8 10.9 9.4
 $ touches_in_opp_box_per_match_team    : num [1:10] 767 871 726 708 764 638 612 661 556 612
 $ yellow_cards_team                    : num [1:10] 72 96 101 78 71 89 85 90 91 106
```

Now that we have read in the data, cleaned the names and checked the structure, we can move on to creating the data visualisation.

3 Data Visualisation

First thing we need to do is to load the required packages for the data visualisation. We will be using the `ggplot2` package, which is a powerful and flexible package for creating data visualisations in R. We will also use the `scales` package, which provides functions for formatting scales and labels. The `ggtext` package is also used to format the text in the plot. `ggrepel` is used to add labels to the plot and differs from `geom_text()` in that it will automatically adjust the position of the labels to avoid overlap. `geomtextpath` is also used to add text to the plot along a specific path or direction. The `gt` and `gtsummary` packages are used to create customisable tables. We can also use the `ggstatsplot` package to create plots with statistical details for more "scientific" analysis. `plotly` is also a great package for creating interactive visualisations.

```
# Load the required packages
library(ggplot2)
library(scales)
```

```
Attaching package: 'scales'
```

```
The following object is masked from 'package:purrr':

    discard
```

```
The following object is masked from 'package:readr':

    col_factor
```

```
library(ggtext)
library(ggrepel)
library(geomtextpath)
library(gt)
library(gtsummary)
library(ggstatsplot)
```

```
You can cite this package as:
     Patil, I. (2021). Visualizations with statistical details: The 'ggstatsplot' approach.
     Journal of Open Source Software, 6(61), 3167, doi:10.21105/joss.03167
```

```
library(plotly)
```

```
Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

Now that we have loaded the required packages, we can create the data visualisation. First, lets have a look at the column names to see what data we have available to visualise.

```
# View the column names
colnames(all_team)
```

```
 [1] "participant_name"
 [2] "accurate_cross_per_match_team"
 [3] "accurate_long_balls_per_match_team"
 [4] "accurate_pass_per_match_team"
 [5] "avg_possession_percentage_team"
 [6] "big_chance_missed_team"
```

```
 [7] "big_chance_team"
 [8] "clean_sheet_team"
 [9] "clearance_per_match_team"
[10] "corners_team"
[11] "fotmob_rating_team"
[12] "fouls_per_match_team"
[13] "goals_conceded_per_match_team"
[14] "goals_per_match_team"
[15] "interception_per_match_team"
[16] "penalty_conceded_team"
[17] "penalty_won_team"
[18] "possession_won_att_3rd_per_match_team"
[19] "red_cards_team"
[20] "saves_per_match_team"
[21] "shots_on_target_per_match_team"
[22] "tackles_won_per_match_team"
[23] "touches_in_opp_box_per_match_team"
[24] "yellow_cards_team"
```

3.1 Splitting the Data

To make the data easier to work with and define, we will split the data into three separate data frames. The first data frame will contain the attacking statistics, the second data frame will contain the defensive statistics, and the third data frame will contain the disciplinary statistics. This will allow us to create separate visualisations for each set of statistics. To do this we will need to use the `select()` function from the `dplyr` package to select the relevant columns for each data frame. Then, we will use the `gather()` function from the `tidyr` package to convert the data from wide to long format. This will make it easier to create the visualisations.

```
# Split the data into attacking, defensive and disciplinary statistics
attacking <- all_team %>%
  select(
    participant_name,
    accurate_cross_per_match_team,
    accurate_long_balls_per_match_team,
    accurate_pass_per_match_team,
    avg_possession_percentage_team,
    big_chance_missed_team,
    big_chance_team,
    # clean_sheet_team,
    # clearance_per_match_team,
    corners_team,
    # fotmob_rating_team,
```

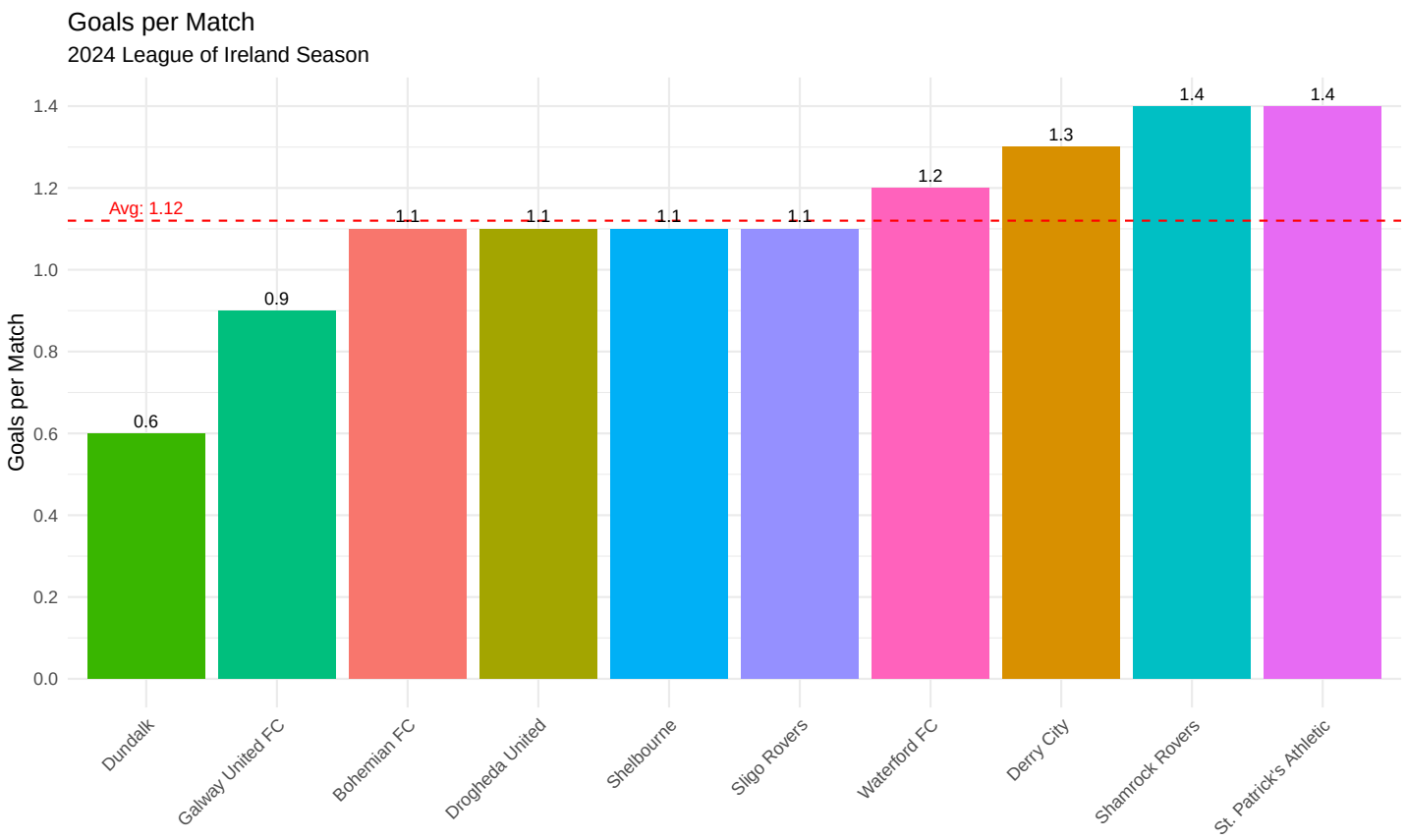
```
    # fouls_per_match_team,
    # goals_conceded_per_match_team,
    goals_per_match_team,
    # interception_per_match_team,
    # penalty_conceded_team,
    # penalty_won_team,
    possession_won_att_3rd_per_match_team,
    # red_cards_team,
    # saves_per_match_team,
    shots_on_target_per_match_team,
    # tackles_won_per_match_team,
    touches_in_opp_box_per_match_team,
    # yellow_cards_team
  ) %>%
  gather(key = "statistic", value = "value", -participant_name)

defensive <- all_team %>%
  select(
    participant_name,
    # accurate_cross_per_match_team,
    # accurate_long_balls_per_match_team,
    # accurate_pass_per_match_team,
    # avg_possession_percentage_team,
    # big_chance_missed_team,
    # big_chance_team,
    clean_sheet_team,
    clearance_per_match_team,
    # corners_team,
    # fotmob_rating_team,
    fouls_per_match_team,
    goals_conceded_per_match_team,
    # goals_per_match_team,
    interception_per_match_team,
    # penalty_conceded_team,
    # penalty_won_team,
    # possession_won_att_3rd_per_match_team,
    # red_cards_team,
    saves_per_match_team,
    # shots_on_target_per_match_team,
    tackles_won_per_match_team,
    # touches_in_opp_box_per_match_team,
    # yellow_cards_team
```

```
  ) %>%
  gather(key = "statistic", value = "value", -participant_name)

disciplinary <- all_team %>%
  select(
    participant_name,
    # accurate_cross_per_match_team,
    # accurate_long_balls_per_match_team,
    # accurate_pass_per_match_team,
    # avg_possession_percentage_team,
    # big_chance_missed_team,
    # big_chance_team,
    # clean_sheet_team,
    # clearance_per_match_team,
    # corners_team,
    # fotmob_rating_team,
    fouls_per_match_team,
    # goals_conceded_per_match_team,
    # goals_per_match_team,
    # interception_per_match_team,
    penalty_conceded_team,
    penalty_won_team,
    # possession_won_att_3rd_per_match_team,
    red_cards_team,
    # saves_per_match_team,
    # shots_on_target_per_match_team,
    # tackles_won_per_match_team,
    # touches_in_opp_box_per_match_team,
    yellow_cards_team
  ) %>%
  gather(key = "statistic", value = "value", -participant_name)
```

3.2 Basic Plotting

Now that we have split the data into three separate data frames, we can create the data visualisations. We will start by creating a bar chart of the attacking statistics. This will show the average number of goals scored per match by each team in the Premier Division. We will use the `ggplot()` function to create the plot, and the `geom_bar()` function to add the bars. We will also use the `geom_text()` function to add labels to the bars. The `theme_minimal()` function is used to set the theme of the plot.

```r
# Create a bar chart of the attacking statistics
attacking %>%
  filter(statistic == "goals_per_match_team") %>%
  ggplot(aes(x = reorder(participant_name, value), y = value, fill = participant_name)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(breaks = breaks_width(0.2)) +
  geom_hline(yintercept = mean(all_team$goals_per_match_team), linetype = "dashed", color =
  ↪  "red") +
  annotate("text", x = 1, y = mean(all_team$goals_per_match_team), label = "Avg: 1.12", vjust
  ↪  = -0.5, size = 3, color = "red") +
  geom_text(aes(label = round(value, 2)), vjust = -0.5, size = 3) +
  labs(
    title = "Goals per Match",
    subtitle = "2024 League of Ireland Season",
    x = "Team",
    y = "Goals per Match"
  ) +
  theme_minimal() +
  theme(
    # Remove legend
    legend.position = "none",
    # Remove x-axis title
    axis.title.x = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```



As we can see from the bar chart, *St. Pat's* and *Shamrock Rovers* scored the most goals per match in the 2024 season, with an average of 1.4 goals per match. The average number of goals scored per match by all teams in the Premier Division was 1.12. The lowest number of goals scored per match was by *Dundalk*, with an average of 0.6 goals per match.

3.3 Faceted Plotting

Now this graph is great for looking at an individual statistics, however, we can also create a faceted plot to show multiple statistics at once. This will allow us to compare the attacking statistics of each team in the Premier Division. We will use the `facet_wrap()` function to create a faceted plot, and the `scales = "free_x"` argument to allow each facet to have its own x-axis scale. We will also use the `dir = "h"` argument to arrange the facets horizontally.

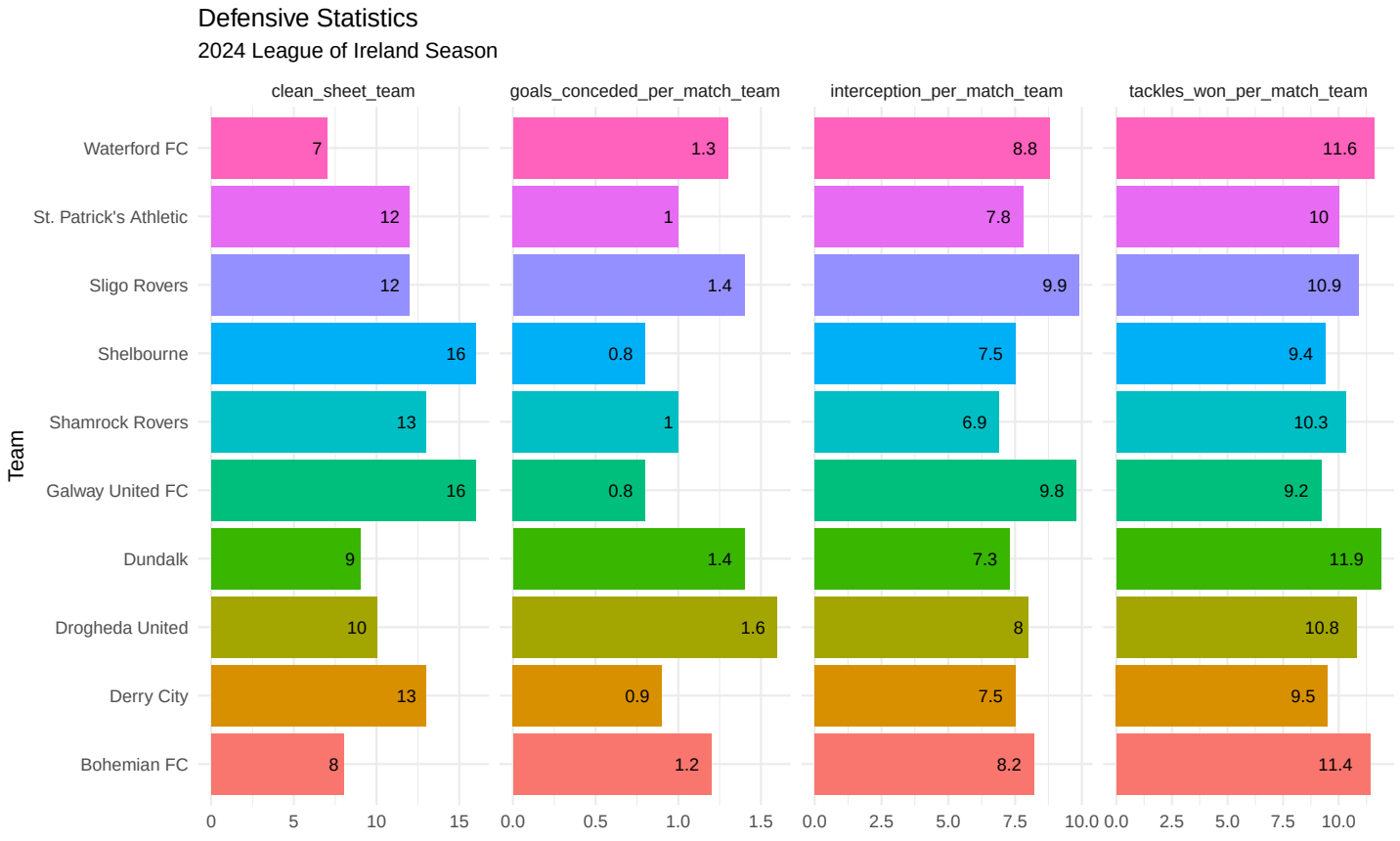
```
attacking %>%
 filter(statistic %in% c("accurate_cross_per_match_team", "accurate_pass_per_match_team",
"avg_possession_percentage_team", "shots_on_target_per_match_team")) %>%
 ggplot() +
  aes(x = reorder(participant_name, value), y = value, fill = participant_name) +
  geom_col() +
  geom_text(aes(label = round(value, 2)), hjust = 1.5, size = 3) +
  coord_flip() +
  labs(
```

```
    title = "Attacking Statistics",
    subtitle = "2024 League of Ireland Season",
    x = "Team"
  ) +
theme_minimal() +
  theme(
    # Remove legend
    legend.position = "none",
    # Remove x-axis title
    axis.title.x = element_blank()
    ) +
  facet_wrap(vars(statistic), scales = "free_x", ncol = 4L, dir = "h")
```

This plot shows a selection of attacking statistics for each team in the Premier Division. As we can see from this plot *Shamrock Rovers* out performed all other teams in the league in terms of accurate crosses, accurate passes, average possession and shots on target. *Shelbourne* had the lowest number of accurate crosses per match, while *Drogheda United* had the lowest number of accurate passes per match. *Drogheda United* also had the lowest average possession percentage, and *Dundalk* had the lowest number of shots on target per match.

Although not the prettiest of plots (my lack of skills!), it does give a good overview of multiple attacking statistics for each team in the Premier Division. We can also create similar plots for the defensive and disciplinary statistics.

```r
defensive %>%
 filter(!(statistic %in% c("clearance_per_match_team", "fouls_per_match_team",
  ↪  "saves_per_match_team"
))) %>%
  ggplot() +
  aes(x = participant_name, y = value, fill = participant_name) +
  geom_col() +
  geom_text(aes(label = round(value, 2)), hjust = 1.5, size = 3) +
  coord_flip() +
  labs(
    title = "Defensive Statistics",
    subtitle = "2024 League of Ireland Season",
    x = "Team"
  ) +
theme_minimal() +
  theme(
    # Remove legend
    legend.position = "none",
    # Remove x-axis title
    axis.title.x = element_blank()
    ) +
  facet_wrap(vars(statistic), scales = "free_x", ncol = 4L, dir = "h")
```



This plot shows a selection of defensive statistics for each team in the Premier Division. As we can see from this plot *Galway United* and *Shelbourne* had the most clean sheets, while *Waterford* had the fewest. *Sligo Rovers* also had the most interceptions per match, while *Shamrock Rovers* had the fewest. *Dundalk* had the most tackles won per match, while *Galway United* had the fewest. *Drogheda United* also had the most goals conceded per match, while *Galway United* and *Shelbourne* conceded the fewest.

3.3.1 Filtering Conditions

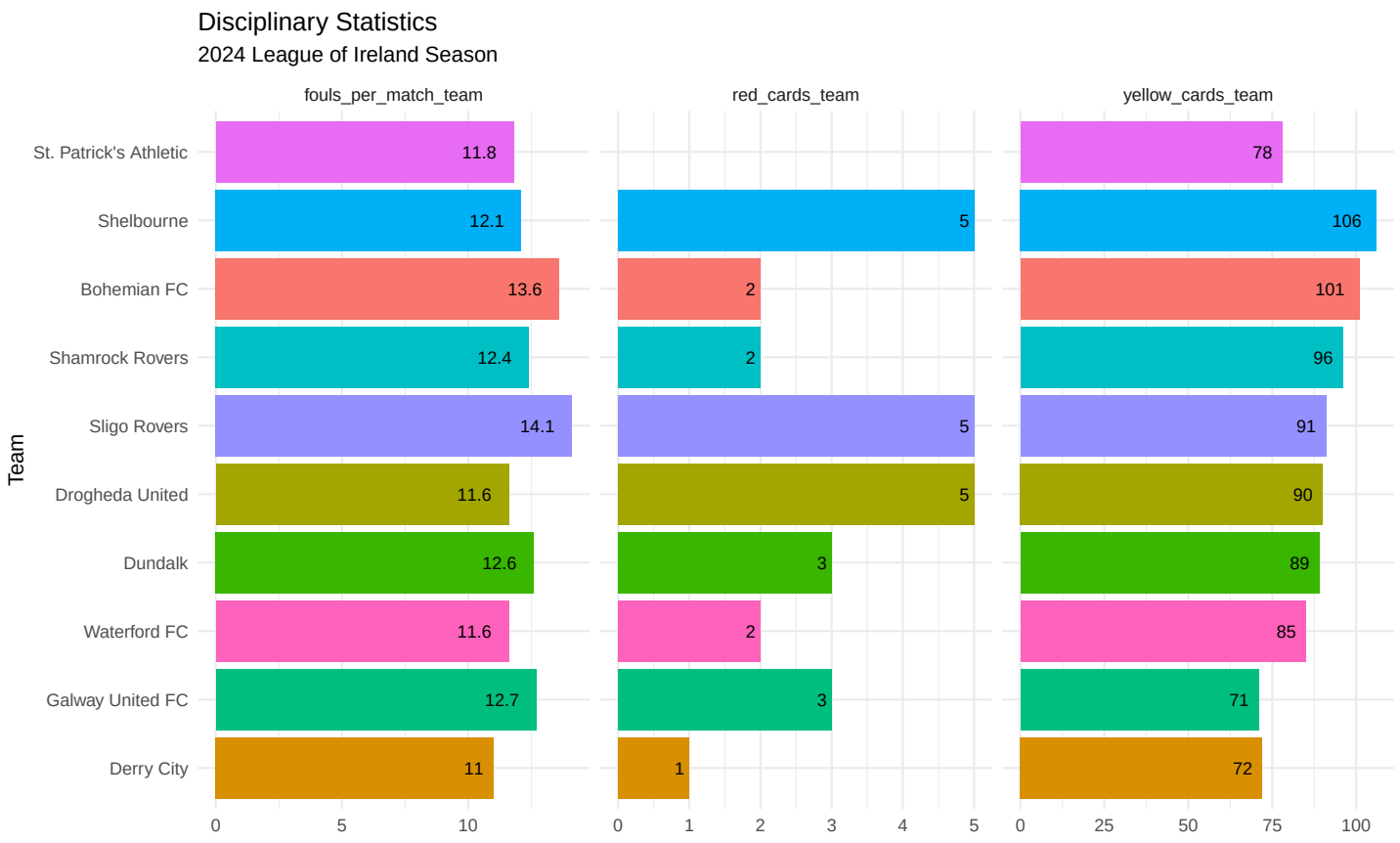
It's important to not the the `filter()` function is different to that in the attacking statistics plot. For the plot above `filter(!(statistic %in% c("clearance_per_match_team", "fouls_per_match_team", "saves_per_match_team")))` is used to remove the statistics that are not relevant to the plot, while `filter(statistic %in% c("accurate_cross_per_match_team", "accurate_pass_per_match_team", "avg_possession_percentage_team", "shots_on_target_per_match_team"))` is used to select the statistics that are relevant to the plot.

Lets also now create a plot for the disciplinary statistics.

```
disciplinary %>%
 filter(!(statistic %in% c("penalty_conceded_team", "penalty_won_team"))) %>%
  ggplot() +
  aes(x = reorder(participant_name, value), y = value, fill = participant_name) +
  geom_col() +
  geom_text(aes(label = round(value, 2)), hjust = 1.5, size = 3) +
  coord_flip() +
  labs(
    title = "Disciplinary Statistics",
    subtitle = "2024 League of Ireland Season",
    x = "Team"
  ) +
theme_minimal() +
  theme(
    # Remove legend
    legend.position = "none",
    # Remove x-axis title
    axis.title.x = element_blank()
    ) +
  facet_wrap(vars(statistic), scales = "free_x", ncol = 3L, dir = "h")
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_col()`).
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_text()`).
```



3.4 `gt` and `gtsummary` Tables

Lets also make a table from this graph using the `gt` and `gtExtras` package. This will allow us to create a customisable table from the data. We can also use the `gtsummary` package to create a summary table of the data. This will provide a summary of the data which will summarise the descriptive statistics of the data.

```
disciplinary %>%
  pivot_wider(names_from = statistic, values_from = value) %>%
  select(participant_name, yellow_cards_team, red_cards_team, fouls_per_match_team) %>%
  arrange(desc(fouls_per_match_team)) %>%
  gt() %>%
  tab_header(
    title = md("<img
    ↪  src='https://www.sligorovers.com/wp-content/uploads/2018/03/221009_SSE-LOI-MENS-PREM_LOGO-1_H
    ↪  style='height:30px;'>     **Disciplinary Statistics**"),
    subtitle = md("2024 League of Ireland Season"))%>%
  cols_align(
    align = "left",
    columns = 1
  ) %>%
  cols_label(
```

Disciplinary Statistics

2024 League of Ireland Season

	Statistics		
Team	Yellow Cards	Red Cards	Fouls per Match
Sligo Rovers	91	5	14.1
Bohemian FC	101	2	13.6
Galway United FC	71	3	12.7
Dundalk	89	3	12.6
Shamrock Rovers	96	2	12.4
Shelbourne	106	5	12.1
St. Patrick's Athletic	78	NA	11.8
Waterford FC	85	2	11.6
Drogheda United	90	5	11.6
Derry City	72	1	11.0

```
    participant_name = "Team",
    yellow_cards_team = "Yellow Cards",
    red_cards_team = "Red Cards",
    fouls_per_match_team = "Fouls per Match"
  ) %>%
  tab_spanner(
    label = "Statistics",
    columns = c("yellow_cards_team", "red_cards_team", "fouls_per_match_team")
  )
```

```
Warning: HTML tags found, and they will be removed.
* Set `options(gt.html_tag_check = FALSE)` to disable this check.
```

Both methods are effective and the use of either the graph or the table depends on the context in which you are presenting the data. The table is great for a more detailed look at the data, while the graph is great for a quick overview. The great thing about the `gt` package is that you can customise the table to suit your needs. For this example I was able to add the League of Ireland Mens Premier Division logo to the table header using the `md()` function from the `gtExtras` package. The `md()` function allows you to add markdown to the table header, which makes it possible to add an image from a URL to the title of the table.

Now, lets create a summary table of the data using the `gtsummary` package. This will provide a summary of the data which will summarise the descriptive statistics of the data. For this example, lets go back to the defensive statistics and create a summary table of the data.

```
defensive %>%
  pivot_wider(names_from = statistic, values_from = value) %>%
  select(-participant_name) %>%
```

```
  gtsummary::tbl_summary(
    #by = participant_name,
    statistic = list(all_continuous() ~ "{mean} ± {sd}",
                     all_categorical() ~ "{n} / {N} ({p}%)"),
    type = c(
      clean_sheet_team~ "continuous",
      clearance_per_match_team~ "continuous",
      fouls_per_match_team~ "continuous",
      goals_conceded_per_match_team~ "continuous",
      interception_per_match_team~ "continuous",
      saves_per_match_team~ "continuous",
      tackles_won_per_match_team~ "continuous"
    ),
    # Show 2 decimal places
    digits = list(all_continuous() ~ c(2, 2))
  ) %>%
  modify_header(label = "**Statistic**") |> # update the column header
  as_gt() %>%
  tab_header(
    title = md("<img
     ↪  src='https://www.sligorovers.com/wp-content/uploads/2018/03/221009_SSE-LOI-MENS-PREM_LOGO-1_H
     ↪  style='height:30px;'>     **Disciplinary Statistics**"),
    subtitle = md("2024 League of Ireland Season"))%>%
  cols_align(
    align = "left",
    columns = 1
  )
```

```
Warning: HTML tags found, and they will be removed.
* Set `options(gt.html_tag_check = FALSE)` to disable this check.
```

The table above provides a summary of the defensive statistics for the Premier Division. The table shows the mean and standard deviation for the number of clean sheets, clearances per match, fouls per match, goals conceded per match, interceptions per match, saves per match, and tackles won per match during the 2024 Premier Division season.

3.5 Team Profiling

The final part of this tutorial will be to create team profiles. This can also be done for individual players. This will allow us to create a visual representation of the team's performance in the 2024 season. We will create a radar

Disciplinary Statistics
2024 League of Ireland Season

Statistic	N = 10[1]
clean_sheet_team	11.60 ± 3.10
clearance_per_match_team	20.05 ± 2.45
fouls_per_match_team	12.35 ± 0.95
goals_conceded_per_match_team	1.14 ± 0.28
interception_per_match_team	8.17 ± 1.03
saves_per_match_team	2.70 ± 0.63
tackles_won_per_match_team	10.50 ± 0.97

[1]Mean ± SD

plot to show the attacking, defensive, and disciplinary statistics for each team in the Premier Division. This will allow us to compare the performance of each team in the different areas of the game.

The first thing we need to do is to create two functions `z_score` and `t_score`. The `z_score` function will calculate the z-score for a given variable, and the `t_score` function will calculate the t-score for a given variable. These functions will be used to standardise the data and convert it to a scale of 0 to 100. This will make it easier to compare the performance of each team in the different areas of performance.

3.5.1 Z Score

```
z_score <- function(x){
  z = (x - mean(x, na.rm = T)) / sd(x, na.rm = T)
  return(z)
}
```

3.5.2 T Score

```
t_score <- function(x){
  t = (x * 10) + 50
  t = ifelse(t > 100, 100,
             ifelse(t < 0, 0, t))
  return(t)
}
```

Now that we have created the functions, we can use them to standardise the data and convert it to a scale of 0 to 100. We will use the `mutate()` function from the `dplyr` package to create new columns for the z-score and

t-score of each variable. This will allow us to create the radar plot. Below is the code to standardise the data and convert it to a scale of 0 to 100, please see the comments for more information on the code.

```
# Standardise the data
attacking_std <- attacking %>%
 pivot_wider(names_from = statistic, values_from = value) %>% # Pivot the data from long to
  ↪  wide format to create columns for each statistic
  mutate_at(vars(-participant_name), z_score) %>% # Calculate the z-score for each variable
   ↪  column
  mutate_at(vars(-participant_name), t_score) # Calculate the t-score for each variable column

defensive_std <- defensive %>%
 pivot_wider(names_from = statistic, values_from = value) %>%
  mutate_at(vars(-participant_name), z_score) %>%
  mutate_at(vars(-participant_name), t_score)

disciplinary_std <- disciplinary %>%
 pivot_wider(names_from = statistic, values_from = value) %>%
  mutate_at(vars(-participant_name), z_score) %>%
  mutate_at(vars(-participant_name), t_score)
```

Now that we have standardised the data and converted it to a scale of 0 to 100, we need to rename each column using the `rename()` function from the `dplyr` package. Then, we will convert the data from wide to long format using the `pivot_longer()` function from the `tidyr` package. This will make it easier to create the radar plot.

```
# Rename the columns
attacking_std <- attacking_std %>%
  rename(
    team = participant_name,
    crosses = accurate_cross_per_match_team,
    long_balls = accurate_long_balls_per_match_team,
    passes = accurate_pass_per_match_team,
    possession = avg_possession_percentage_team,
    big_chance_missed = big_chance_missed_team,
    big_chance = big_chance_team,
    corners = corners_team,
    goals = goals_per_match_team,
    possession_won_att_3rd = possession_won_att_3rd_per_match_team,
    shots_on_target = shots_on_target_per_match_team,
    touches_in_opp_box = touches_in_opp_box_per_match_team
  ) %>%
  select(team, big_chance_missed, big_chance, goals, possession_won_att_3rd, shots_on_target,
   ↪  touches_in_opp_box) %>%
```

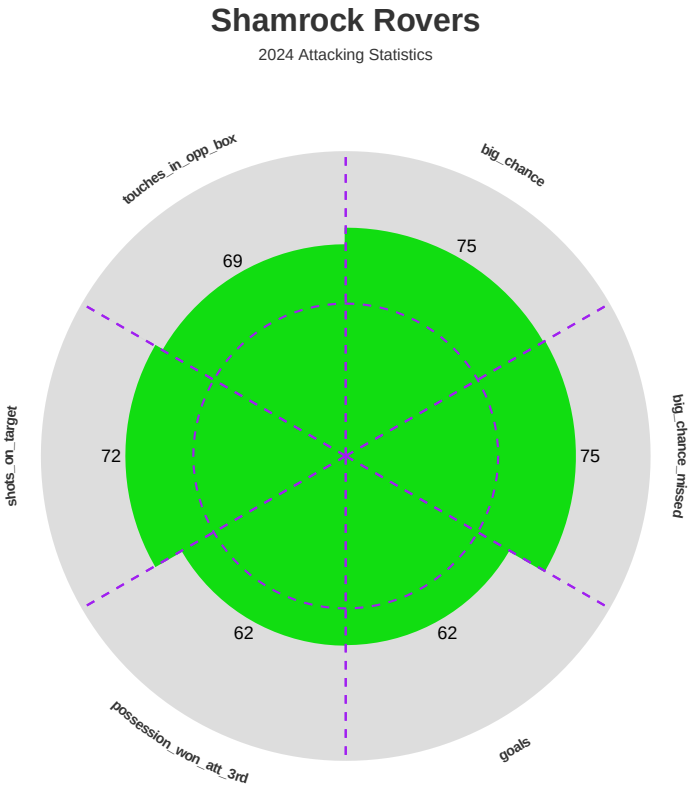
```
  pivot_longer(cols = -team, names_to = "statistic", values_to = "value")

defensive_std <- defensive_std %>%
  rename(
    team = participant_name,
    clean_sheets = clean_sheet_team,
    clearances = clearance_per_match_team,
    fouls = fouls_per_match_team,
    goals_conceded = goals_conceded_per_match_team,
    interceptions = interception_per_match_team,
    saves = saves_per_match_team,
    tackles_won = tackles_won_per_match_team
  ) %>%
  select(team, clean_sheets, clearances, goals_conceded, interceptions, saves, tackles_won)
  ↪  %>%
  pivot_longer(cols = -team, names_to = "statistic", values_to = "value")

disciplinary_std <- disciplinary_std %>%
  rename(
    team = participant_name,
    fouls = fouls_per_match_team,
    penalty_conceded = penalty_conceded_team,
    penalty_won = penalty_won_team,
    red_cards = red_cards_team,
    yellow_cards = yellow_cards_team
  ) %>%
  select(team, fouls, penalty_conceded, penalty_won, red_cards, yellow_cards) %>%
  pivot_longer(cols = -team, names_to = "statistic", values_to = "value")
```

3.5.3 Radar Plot

Now that we have standardised the data and converted it to a scale of 0 to 100, we can create the radar plot. We will use the `geom_bar()` function from the `ggplot2` package to create the bars for the radar plot. We will also use the `coord_curvedpolar()` function to create the radar plot. This will allow us to compare the performance of each team in the different areas of performance. Below is the code to create the radar plot, please see the comments for more information on the code.

```
att <- attacking_std %>%
  filter(team == "Shamrock Rovers") %>%
  ggplot(aes(x = statistic, y = value, label = round(value, 0))) +
  geom_bar(width = 1,
```

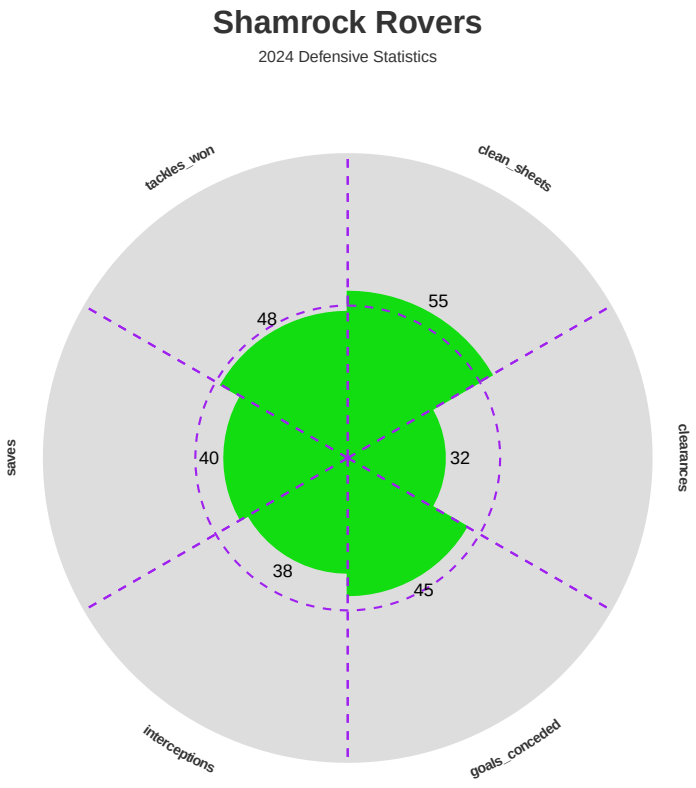
```
           fill = "green",
           color = "green",
           stat = "identity") +
  coord_curvedpolar() +
  geom_bar(aes(y=100), stat="identity", width=1, alpha=.2) +
  # add & customize line that border whole polar
  geom_hline(yintercept = seq(50, 50, by = 1),
             color = "purple",
             linetype = "dashed") +
  # add & customize lines between each polar segment
  geom_vline(xintercept = seq(.5, 12, by = 1),
             color = "purple",
             linetype = "dashed") +
  geom_text(aes(y = value + 5), size = 3) +
  labs(
    title = "Shamrock Rovers",
    subtitle = "2024 Attacking Statistics") +
theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = .5, colour = "gray20", face = "bold", size = 16),
        plot.subtitle = element_text(hjust = .5, colour = "gray20", size = 8),
        panel.grid = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        axis.text = element_text(face = "bold", size = 6.8, colour = "gray20"),
        axis.title = element_blank(),
        axis.text.x = element_text(face = "bold", size = 7))

att
```



Now lets also create a radar plot for their defensive statistics.

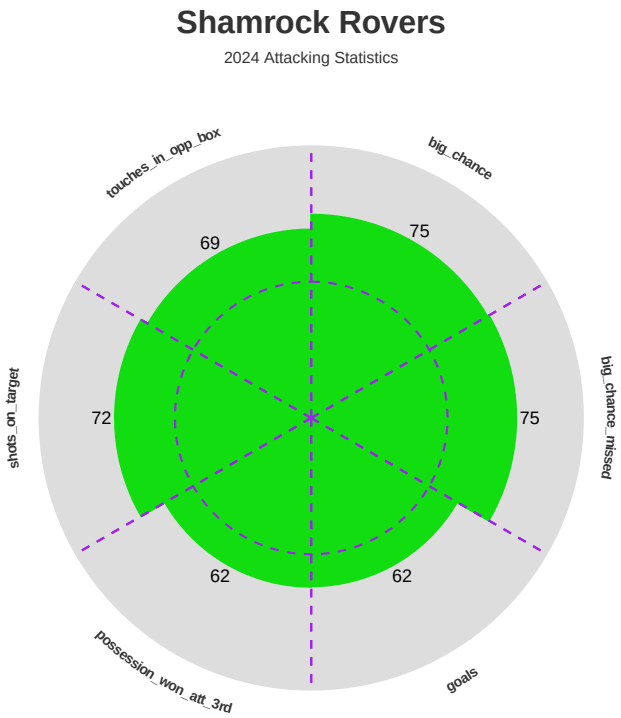
```
def <- defensive_std %>%
  filter(team == "Shamrock Rovers") %>%
  ggplot(aes(x = statistic, y = value, label = round(value, 0))) +
  geom_bar(width = 1,
           fill = "green",
           color = "green",
           stat = "identity") +
  coord_curvedpolar() +
  geom_bar(aes(y=100), stat="identity", width=1, alpha=.2) +
  # add & customize line that border whole polar
  geom_hline(yintercept = seq(50, 50, by = 1),
             color = "purple",
             linetype = "dashed") +
  # add & customize lines between each polar segment
  geom_vline(xintercept = seq(.5, 12, by = 1),
             color = "purple",
             linetype = "dashed") +
  geom_text(aes(y = value + 5), size = 3) +
  labs(
    title = "Shamrock Rovers",
```

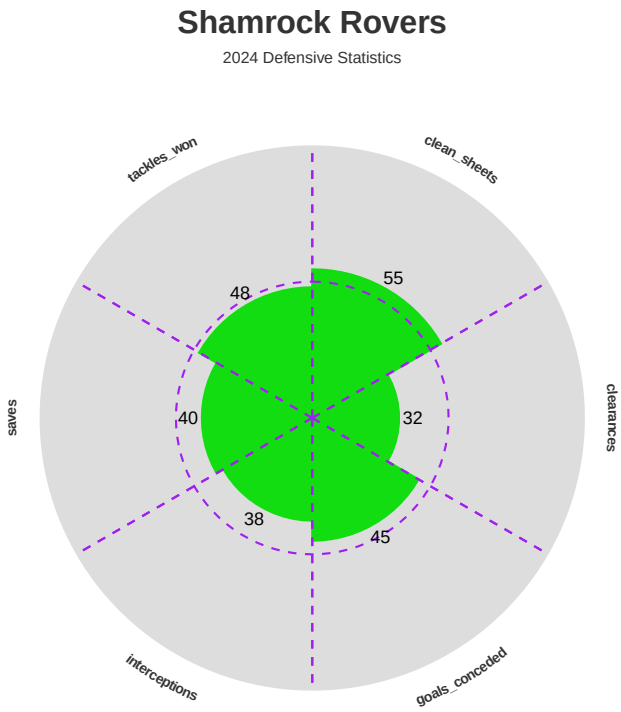
```
    subtitle = "2024 Defensive Statistics") +
theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = .5, colour = "gray20", face = "bold", size = 16),
        plot.subtitle = element_text(hjust = .5, colour = "gray20", size = 8),
        panel.grid = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        axis.text = element_text(face = "bold", size = 6.8, colour = "gray20"),
        axis.title = element_blank(),
        axis.text.x = element_text(face = "bold", size = 7))

def
```



Finally, lets use a package called `patchwork` to combine the two plots into one. This will allow us to compare the attacking and defensive statistics for each team in the Premier Division. The documentation for the `patchwork` package can be found here. Patchwork is a great package for combining multiple plots into one, and allows you to customise the layout of the plots.

```
library(patchwork)
att + def
```





4 Conclusion

In this tutorial, we have created data visualisations of the 2024 League of Ireland season using R. We have created bar charts of the attacking, defensive, and disciplinary statistics for each team in the Premier Division. We have also created a radar plot to compare the performance of each team in the different areas of performance. The data used in this tutorial is for ***illustrative purposes only***, there are people far smarter than myself who can do this much better - I just document what I learn. The data visualisations created in this tutorial provide a snapshot of the performance of each team in the 2024 season, and can be used to compare the performance of each team in the different areas of performance. I hope you have enjoyed this tutorial and have learned something new about creating data visualisations in R. Thank you for reading!

More Resources Available My Website