So now that I live in Melbourne, to blend in with the locals I need to at least vaguely follow the AFL (Australian Football League). For instance, my work like many others has an AFL footy-tipping competition. I was initially going to choose my tips based on wisdom of the crowds (ie choose the favourite) but decided that this was a good occasion to try something a bit more scientific.

As is the case for most organised sports these days, there is rich data available on AFL results and other metrics. After wasting 20 minutes trying to locate and scrape various websites I remembered “someone must have already done this”, and sure enough found Jimmy Day’s highly effective fitzRoy R package.

**Elo ratings over the long term**

An easy way to generate predictions of the likely winner in a head to head game is by comparing Elo ratings based on past performance.

Here’s the Elo ratings of present and past AFL (and its predecessor, the Victorian Football League) teams, treating them continuously from 1897 to the present:

Amongst other things, this provides at least *an* answer to the vexed question of which AFL team is the best overall – apparently Geelong (prepares to duck). Interesting to see that sustained period of Collinwood dominance in the 1930s too. Also suprising (to me, showing how little attention I’ve paid) is Sydney sitting on the second highest Elo rating today based on the full history of the game. When I last paid attention to the footy in the late 1990s, the Sydney Swans were literally the punchline of an evening comedy show on TV, which adopted them out of the pure humour of supporting such a perpetually losing team. Obviously they’ve recovered, no doubt in part due to the support of their fans through the tough patch.

I can’t remember the name of the TV show.

Anyway, before we get on to thinking about predictions, here’s the R code to bring in those results and draw the chart of Elo ratings.

*Post continues after code extract*

# devtools::install\_github("jimmyday12/fitzRoy")

# devtools::install\_github("ellisp/frs-r-package/pkg")

library(tidyverse)

library(scales)

library(fitzRoy) # for reading AFL scores

library(frs) # for Elo functions

library(Cairo)

library(ggrepel)

library(lubridate)

# some parameters for graphics:

update\_geom\_defaults("text\_repel", list(colour = "brown"))

the\_caption <- "Source: afltables via fitZroy; analysis by freerangestats.info"

# download AFL results:

results <- get\_match\_results()

# set up a data frame in the right shape to sort through the games one at a time

r <- results %>%

rename\_all(tolower) %>%

rename\_all(function(x){gsub("\\.", "\_", x)}) %>%

# limit to 2018 season and earlier:

filter(date < as.Date("2019-01-01")) %>%

select(home\_team, away\_team, date, game, margin) %>%

gather(location, team, -date, -game, -margin) %>%

mutate(location = gsub("\_team", "", location),

margin = ifelse(location == "home", margin, -margin)) %>%

mutate(winner = margin > 0) %>%

arrange(date) %>%

mutate(starting\_elo = 1500,

new\_elo = 1500) %>%

group\_by(game) %>%

# sequence so the winner is the first listed each time for each game:

arrange(game, desc(winner))

#' Calculate Elo ratings for a data frame of AFL results

#'

#' @param r a data frame with columns including date, team, game, starting\_elo, new\_elo

afl\_elos <- function(r){

r <- r %>%

group\_by(game) %>%

arrange(date, game, desc(winner))

all\_games <- unique(r$game)

# This loop will give some tidyverse advocates conniptions but it works for me; and

# it seems inherently iterative so perhaps a loop is the logical way to do it:

for(g in all\_games){

this\_game <- r[r$game == g, ]

er <- elo\_rating(a = this\_game[1, "starting\_elo"],

b = this\_game[2, "starting\_elo"],

winner = "a",

ml = round(this\_game[1, "margin"] / 6))

r[r$game == g, "new\_elo"] <- unlist(c(er$a, er$b))

r <- r %>%

group\_by(team) %>%

mutate(starting\_elo = lag(new\_elo, default = 1500)) %>%

ungroup()

}

r <- r %>%

mutate(season = lubridate::year(date)) %>%

group\_by(game) %>%

arrange(game, desc(starting\_elo)) %>%

mutate(predicted\_winner = c(TRUE, FALSE),

successful\_prediction = predicted\_winner == winner) %>%

ungroup()

return(r)

}

# Elo ratings if we started back in 1897 (takes a couple of minutes):

elos\_all <- afl\_elos(r)

# Plot of Elo ratings over all of history

elos\_all %>%

mutate(team = fct\_reorder(team, -new\_elo, .fun = last)) %>%

ggplot(aes(x = date, y = new\_elo, colour = team, group = paste(season, team))) +

geom\_line(size = 0.2) +

theme(legend.position = "right") +

labs(x = "", y = "Elo rating", colour = "",

caption = the\_caption) +

ggtitle("Elo ratings of AFL / VFL teams, 1897 to present",

"Elo rating calculated by modified FIBS method. Gaps in lines are the inter-season breaks.")

**Ratings change depending on when you start from**

What happens if we base our ratings only on recent performance? As an example, the chart below shows team ratings for the 2018 season if all teams were reset to 1500 at the beginning of the year:

The rankings differ quite noticeably from those based on the full history.

2018’s top ranked team, Richmond, was only seventh when the full history was used. They didn’t make the grand final – such is the luck inherent in this sort of tournament – but the eventual premiers West Coast (my own team, for what it’s worth, as a result of growing up in Perth) were ranked a solid second. The four top teams were those in the semi-finals, so that system works to a degree.

So that’s ratings based on one year, what if we choose a dozen years? A very interesting story of Geelong’s complete dominance up to about 2014, caught up in the past five years or so by Hawthorn and Sydney. Richmond is much less prominent in this view, reflecting how suprisingly well it went in 2018 (despite missing out on the final) compared to form in the previous ten years:

The ratings based on 2007 and onwards end up in a very similar position to those based on 1897 and onwards; it looks like there is about 10 years of momentum stored up in a rolling Elo rating.

*Post continues after code extract*

# Elo ratings if we did them only for last year:

elos\_2018 <- r %>%

filter(date > as.Date("2018-01-01")) %>%

afl\_elos()

# Elo ratings if we did them only from 2007 onwards

elos\_2007 <- r %>%

filter(date > as.Date("2007-01-01")) %>%

afl\_elos()

# Plot of Elo ratings if done just from 2018

elos\_2018 %>%

mutate(team = fct\_reorder(team, -new\_elo, .fun = last)) %>%

ggplot(aes(x = date, y = new\_elo, colour = team)) +

geom\_line() +

theme(legend.position = "right") +

labs(x = "", y = "Elo rating", colour = "",

caption = the\_caption) +

ggtitle("Elo ratings of AFL / VFL teams, 2018 season",

"Elo rating calculated by modified FIBS method, ignoring pre-2018 performance.")

# Plot of Elo ratings if done from 2007

elos\_2007 %>%

mutate(team = fct\_reorder(team, -new\_elo, .fun = last)) %>%

ggplot(aes(x = date, y = new\_elo, colour = team, group = paste(team, season))) +

geom\_line() +

theme(legend.position = "right") +

labs(x = "", y = "Elo rating", colour = "",

caption = the\_caption) +

ggtitle("Elo ratings of AFL / VFL teams, 2007 to present",

"Elo rating calculated by modified FIBS method, ignoring pre-2007 performance.")

**Recent performance is a better guide for predictions than the full history**

Let’s turn to the question of using Elo ratings, whether based on 12 months of performance or 120 years, to predict winners in the coming season. This next chart compares the teams’ ratings at the end of 2018 (having been reset to 1500 at the beginning of the year) with two candidate explanatory variables with predictive power – Elo rating based just on the 2017 season, and Elo rating based on 120 years of performance to end of 2017. Both sets of ratings have predictive power, but the ratings based on only 12 months are slightly better.

Finally, here’s a chart of how well we would have gone in a 120 year footy-tipping competition if we simply tipped the team with the higher Elo rating, based on performance to date, to win. Our success rate in the red line (using full history of performance) hovers around 65%, which isn’t stellar but is clearly better than chance.

The short blue line is success rate when using just performance from 2017 onwards to predict. We see its slightly better than the predictions that used full history. Ideally, I would calculate Elo ratings based on the past 12, 24 and 36 months only for every season to find the best level of history to include, but that’s more rigour than I’m interested in at the moment. I’m going to use Elo ratings based on the 2018 and 2019 seasons for my footy tips.

*Post continues after code extract*

#----------------is it best just to use recent Elo, or all history---------

# Elo ratings if we did them only for 2017

elos\_2017\_final <- r %>%

filter(year(date) == 2017) %>%

afl\_elos() %>%

group\_by(team) %>%

arrange(desc(date)) %>%

slice(1) %>%

select(team, elo\_2017 = new\_elo) %>%

ungroup()

# based on all history up to 2017 (already calculated this so can just filter to get it)

elos\_2017\_and\_earlier <- elos\_all %>%

filter(year(date) == 2017) %>%

group\_by(team) %>%

arrange(desc(date)) %>%

slice(1) %>%

select(team, elo\_up\_to\_2017 = new\_elo) %>%

ungroup()

elos\_2018\_final <- elos\_2018 %>%

group\_by(team) %>%

arrange(desc(date)) %>%

slice(1) %>%

select(team, elo\_2018 = new\_elo) %>%

ungroup()

elos\_two\_types <- elos\_2017\_final %>%

left\_join(elos\_2017\_and\_earlier, by = "team") %>%

inner\_join(elos\_2018\_final, by = "team")

elos\_two\_types %>%

gather(variable, value, -team, -elo\_2018) %>%

ggplot(aes(x = value, y = elo\_2018, label = team)) +

geom\_smooth(method = "gam") +

geom\_point() +

facet\_wrap(~variable, scales = "free\_x") +

geom\_text\_repel() +

labs(x = "Elo rating before the 2018 season",

y = "Elo rating at end of 2018",

caption = the\_caption) +

ggtitle("Elo ratings at end of 2018 compared to two alternative starting points",

"Horizontal axis for panel on left shows rating based on just 2017; on right is based on 1897 to 2017")

mod <- lm(elo\_2018 ~ elo\_up\_to\_2017 + elo\_2017, data = elos\_two\_types)

summary(mod)

anova(mod)

# this suggests that Elo ratings have too long a memory; it is better

# to shrink them towards 1500 in between each season

#---------------success over time-----------------

elos\_2017\_onwards <- r %>%

filter(year(date) >= 2017) %>%

afl\_elos() %>%

group\_by(season) %>%

summarise(successful\_predictions = mean(predicted\_winner == winner)) %>%

mutate(method = "2017")

CairoSVG("..http://freerangestats.info/img/0147-success-rates.svg", 8, 4)

elos\_all %>%

group\_by(season) %>%

summarise(successful\_predictions = mean(successful\_prediction)) %>%

mutate(method = "1897") %>%

rbind(elos\_2017\_onwards) %>%

ggplot(aes(x = season, y = successful\_predictions, colour = method)) +

geom\_line() +

labs(x = "Season", colour = "Prediction based on performance since:") +

scale\_y\_continuous("Prediction success", label = percent\_format(accuracy = 1)) +

ggtitle("Success rate of predicted AFL outcomes from Elo rating based on full history")

**And the answer is…**

So here’s my rating table I used for tips for round 1 of 2019 (other than the first game, which being on a Thursday snuck up on me before I realised I had to get tips in – newbie mistake, annoying because I would have picked the winner correctly).

| **team** | **new\_elo** |
| --- | --- |
| Richmond | 1560.921 |
| West Coast | 1553.291 |
| Melbourne | 1553.150 |
| Collingwood | 1543.454 |
| Geelong | 1533.995 |
| Hawthorn | 1523.387 |
| GWS | 1522.148 |
| Port Adelaide | 1515.349 |
| Sydney | 1514.495 |
| Essendon | 1514.178 |
| North Melbourne | 1509.164 |
| Adelaide | 1507.768 |
| Brisbane Lions | 1465.610 |
| Footscray | 1461.265 |
| Fremantle | 1457.496 |
| St Kilda | 1433.498 |
| Gold Coast | 1423.489 |
| Carlton | 1407.342 |

This led to tips that were mostly consistent with the crowd favourites and bookies’ odds. The main exception was that I tip Hawthorn to beat the Adelaide Crows, against the strong expectations of everyone else. Possibly they know something additional to what’s in my data; did Hawthorn lose some key players, or have a bad off-season? We’ll only know at the end of the season when we can see if my method gets better results than the average punter.

*Post continues after code extract*

#---------------current ratings--------------------------

# This provides round 1 tips for 2019

# The differ from the popular choice on two:

# Hawthorn predicted by me to beat Adelaide Crows, although tips are 88:12 other direction

# North Melbourne predicted by me to beat Fremantle, although tips are 62:38 in other direction

elos\_2018 %>%

group\_by(team) %>%

arrange(desc(date)) %>%

slice(1) %>%

ungroup() %>%

arrange(desc(new\_elo)) %>%

select(team, new\_elo) %>%

knitr::kable() %>%

clipr::write\_clip()

That’s all.

Here’s the R packages used in producing this post:

thankr::shoulders() %>%

mutate(maintainer = str\_squish(gsub("<.+>", "", maintainer))) %>%

knitr::kable() %>%

clipr::write\_clip()

| **maintainer** | **no\_packages** | **packages** |
| --- | --- | --- |
| Hadley Wickham | 16 | assertthat, dplyr, ellipsis, forcats, ggplot2, gtable, haven, httr, lazyeval, modelr, plyr, rvest, scales, stringr, tidyr, tidyverse |
| R Core Team | 11 | base, compiler, datasets, graphics, grDevices, grid, methods, splines, stats, tools, utils |
| Winston Chang | 4 | extrafont, extrafontdb, R6, Rttf2pt1 |
| Yihui Xie | 4 | evaluate, knitr, rmarkdown, xfun |
| Kirill Müller | 4 | DBI, hms, pillar, tibble |
| Yixuan Qiu | 3 | showtext, showtextdb, sysfonts |
| Lionel Henry | 3 | purrr, rlang, tidyselect |
| Gábor Csárdi | 3 | cli, crayon, pkgconfig |
| Dirk Eddelbuettel | 2 | digest, Rcpp |
| Jeroen Ooms | 2 | curl, jsonlite |
| Jim Hester | 2 | glue, withr |
| Kamil Slowikowski | 1 | ggrepel |
| Vitalie Spinu | 1 | lubridate |
| Deepayan Sarkar | 1 | lattice |
| Patrick O. Perry | 1 | utf8 |
| Jennifer Bryan | 1 | cellranger |
| Michel Lang | 1 | backports |
| Jennifer Bryan | 1 | readxl |
| Kevin Ushey | 1 | rstudioapi |
| Martin Maechler | 1 | Matrix |
| Justin Talbot | 1 | labeling |
| Jim Hester | 1 | readr |
| Charlotte Wickham | 1 | munsell |
| Alex Hayes | 1 | broom |
| Simon Wood | 1 | mgcv |
| Joe Cheng | 1 | htmltools |
| Simon Urbanek | 1 | audio |
| Peter Ellis | 1 | frs |
| Brodie Gaslam | 1 | fansi |
| R-core | 1 | nlme |
| Stefan Milton Bache | 1 | magrittr |
| Marek Gagolewski | 1 | stringi |
| James Hester | 1 | xml2 |
| Max Kuhn | 1 | generics |
| Simon Urbanek | 1 | Cairo |
| Jeremy Stephens | 1 | yaml |
| James Day | 1 | fitzRoy |
| Achim Zeileis | 1 | colorspace |
| Rasmus Bååth | 1 | beepr |