We’ve made our predictions about the past (estimating the relative strengths of teams based on past results), now we need to predict the future. I think it also nicely captures that even our predictions about the past are noisy- we can not ever truly know the exact strengths of football teams; the job of analytics is to estimate these are accurately as possible. But any noise in those past predictions will be carried forward and amplified when predicting the future.

Onward to the code, first as always, loading libraries and setting a seed for reproducibility:

library(tidyverse)

library(ggrepel)

set.seed(3459)

We’re then going to load all the stuff we prepped and predicted in the last post. Remember the α parameter below refers to a teams attacking strength (the relative number of goals they are expected to score), and the β parameter refers to the attacking strength (the inverse of the relative number of goals they are expected to concede). Finally, γ refers to the extra advantage of playing at home.

fixtures <- readRDS("../../static/files/dc\_2/dc\_fixtures.rds")

results <- readRDS("../../static/files/dc\_2/dc\_results.rds")

model <- readRDS("../../static/files/dc\_2/dc\_model.rds")

model

## $alpha

## Arsenal Blackburn\_Rovers Coventry\_City Dover\_Athletic

## 1.1106558 0.6370160 0.3023048 -0.2875353

## Enfield\_Town Frimley\_Green

## -0.3767038 -1.3857376

##

## $beta

## Arsenal Blackburn\_Rovers Coventry\_City Dover\_Athletic

## 0.6457175 0.4289270 0.3647815 -0.1362931

## Enfield\_Town Frimley\_Green

## -0.3852812 -0.9178517

##

## $gamma

## gamma

## 0.189462

We’ll define a quick function to do our prediction. For a quick explanation of exactly why it’s coded as presented, see [the previous post](https://www.robert-hickman.eu/post/dixon_coles_1/#Tinkering), under the title ‘Tinkering’.

For a given string of a home team and an away team, the function finds the relevant parameters from a third argument (param\_list) and calculates the expected goal for each team.

predict\_results <- function(home, away, param\_list) {

e\_goals\_home <- exp(param\_list$alpha[home] - param\_list$beta[away] + param\_list$gamma)

e\_goals\_away <- exp(param\_list$alpha[away] - param\_list$beta[home])

df <- data.frame(home = home, away = away,

e\_hgoal = as.numeric(e\_goals\_home),

e\_agoal = as.numeric(e\_goals\_away))

return(df)

}

If we run this for two example teams for example:

#two example teams

home <- "Blackburn\_Rovers"

away <- "Arsenal"

prediction <- predict\_results(home, away, model)

prediction

## home away e\_hgoal e\_agoal

## 1 Blackburn\_Rovers Arsenal 1.198128 1.977293

We can see that it gives Arsenal (the away team) a slightly more optimistic chance than Blackburn. The expected goals for each team of course can be rewritten as the mean, and in our Poisson model refers to λ (lambda)- the mean times an event (goal) happens per a time interval (match). We also set a maximum number of possible goals (7 in this case\*) to bound the area under the distribution so we aren’t sampling forever.

\*sharp readers might notice that this is actually *lower* than the lambda for our more extreme cases (e.g. Arsenal at home to Frimley Green), but for realistic matches (even between wildly different professional sides) this is a fair enough assumption.

We then use dpois() to calculate the probability of this Poisson function returning a value (0:7 goals) given it’s lambda value. So if we run this over the prediction we made for Blackburn Rovers vs. Arsenal we get:

#set a limit of where we'll calculate across

max\_goals <- 7

#calculate the probability of scoring x goals for either team

blackburn\_goal\_probs <- lapply(0:max\_goals, dpois, lambda = prediction$e\_hgoal)

arsenal\_goal\_probs <- lapply(0:max\_goals, dpois, lambda = prediction$e\_agoal)

#bind together in a df

df <- data.frame(goals = rep(0:max\_goals, 2),

team = rep(c(home, away), each = max\_goals+1),

p = c(unlist(blackburn\_goal\_probs), unlist(arsenal\_goal\_probs)))

#plot the p of scoring x goals for either team

p1 <- ggplot(df, aes(x = goals, y = p, fill = team)) +

geom\_density(stat = "identity", alpha = 0.5) +

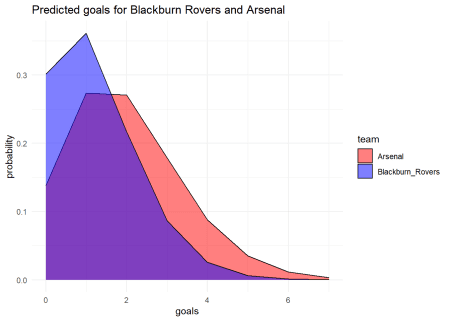
scale\_fill\_manual(values = c("red", "blue")) +

labs(title = "Predicted goals for Blackburn Rovers and Arsenal",

y = "probability") +

theme\_minimal()

p1



Because of how maths works, these curves are the same result we would get if we ran rpois() (sampling from the Poisson function) lots of times. We’ll do that quickly because it sets the stage nicely for what will come later.

#sample from the function lots of times for each team

n <- 100000

blackburn\_goals\_samples <- rpois(n, lambda = prediction$e\_hgoal)

arsenal\_goals\_samples <- rpois(n, lambda = prediction$e\_agoal)

df <- data.frame(team = rep(c(home, away), each = n),

sampled\_goals = c(blackburn\_goals\_samples, arsenal\_goals\_samples))

#look its the same plot!

p2 <- ggplot(df, aes(x = sampled\_goals, fill = team)) +

geom\_bar(stat = "count", position = "dodge", colour = "black", alpha = 0.5) +

geom\_line(aes(colour = team), stat = "count", size = 3) +

scale\_fill\_manual(values = c("red", "blue"), guide = FALSE) +

scale\_colour\_manual(values = c("red", "blue"), guide = FALSE) +

labs(title = "Predicted goals for Blackburn Rovers and Arsenal",

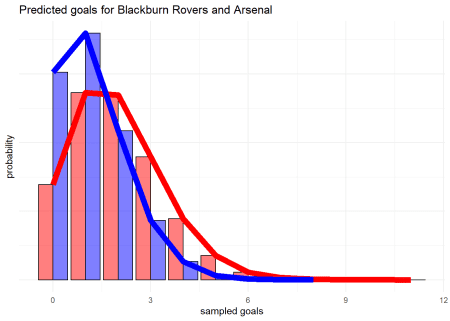
y = "probability",

x = "sampled goals") +

theme\_minimal() +

theme(axis.text.y = element\_blank())

p2



Ok great!, in terms of predicting the result, the rightwards shift of the red (Arsenal) curve here is the difference in the teams ability to generate a positive goal differential- it makes it more likely that if we sample event, Arsenal will have scored more goals than Blackburn Rovers at the end of the match. Of course, it’s also obvious that while Arsenal’s curve is right shifted, the bars for Arsenal scoring 0 goals and Blackburn scoring 6 are still sizable enough that it isn’t outside the realm of possibility.

This is a nice way of presenting the chance of each team scoring n goals, but doesn’t really help us in predicting the result of a match given that this relies on the interaction of both these distributions (we need to know how many goals BOTH Arsenal AND Blackburn will score).

To calculate this, we can do an outer product of the probabilities for both teams scoring n goals. We can then plot the probability of each *scoreline* as a tile plot:

#calculate matrix of possible results and probabilities of those

matrix <- outer(unlist(arsenal\_goal\_probs), unlist(blackburn\_goal\_probs)) %>%

as.data.frame() %>%

gather() %>%

#add in scorelines

mutate(hgoals = rep(0:max\_goals, max\_goals+1),

agoals = rep(0:max\_goals, each = max\_goals+1))

#make the tile plot

p3 <- ggplot(matrix, aes(x = hgoals, y = agoals, fill = value)) +

geom\_tile() +

geom\_text(aes(label = paste(hgoals, agoals, sep = "-"))) +

scale\_fill\_gradient2(low = "white", high = "red", guide = FALSE) +

theme\_minimal()

p3



Where we can see that the most common scorelines are low scoring (football is a low scoring game), and slightly biased towards away goals (i.e. Arsenal are more likely to win than lose). The darkest (most likely) tiles being 1-1 or a 2-1 Arsenal win seem very plausible given our calculated λs earlier.

We can then do this for every fixture and build a large graph of the expected results for each using a simple map2\_ apply. Because of the huge plot, I’ve restricted it here to a 3×3 matrix of the results for Arsenal, Coventry City, and Enfield Town, but if you click you should be linked to the full image.

#want to predict over the whole fixture space

all\_fixtures <- bind\_rows(fixtures, results) %>%

filter(!duplicated(paste(home, away), fromLast = TRUE))

#get the lambda for each team per game

predictions <- map2\_df(all\_fixtures$home, all\_fixtures$away,

predict\_results,

model)

#calc out probabilities and bind up

all\_predictions <- map2\_df(

predictions$e\_hgoal, predictions$e\_agoal,

function(lambda\_home, lambda\_away, max\_goals) {

hgoal\_prob <- dpois(0:max\_goals, lambda\_home) %>% `names<-`(0:max\_goals)

agoal\_prob <- dpois(0:max\_goals, lambda\_away) %>% `names<-`(0:max\_goals)

outer(hgoal\_prob, agoal\_prob) %>%

as.data.frame() %>%

gather() %>%

rownames\_to\_column("row") %>%

mutate(hgoal = as.numeric(row) %% (max\_goals+1)-1) %>%

mutate(hgoal = case\_when(hgoal < 0 ~ max\_goals, TRUE ~ hgoal),

agoal = as.numeric(key)) %>%

select(sample\_hgoal = hgoal, sample\_agoal = agoal, prob = value)

}, max\_goals) %>%

cbind(all\_fixtures[rep(seq\_len(nrow(all\_fixtures)), each=(max\_goals+1)^2),], .) %>%

group\_by(home, away) %>%

mutate(prob = prob / sum(prob)) %>%

ungroup()

#plot again

p3 <- all\_predictions %>%

#filter only a few out to scale plot

filter(home %in% c("Arsenal", "Coventry\_City", "Enfield\_Town"),

away %in% c("Arsenal", "Coventry\_City", "Enfield\_Town")) %>%

ggplot(aes(x = sample\_hgoal, y = sample\_agoal, fill = prob)) +

geom\_tile() +

geom\_point(aes(x = hgoal, y = agoal),

colour = "blue", size = 5, alpha = 0.5 / max\_goals^2) +

geom\_text(aes(label = paste(sample\_hgoal, sample\_agoal, sep = "-")), size = 2.3) +

scale\_fill\_gradient2(low = "white", high = "red", guide = FALSE) +

labs(

title = "predictions for final score across all fixtures",

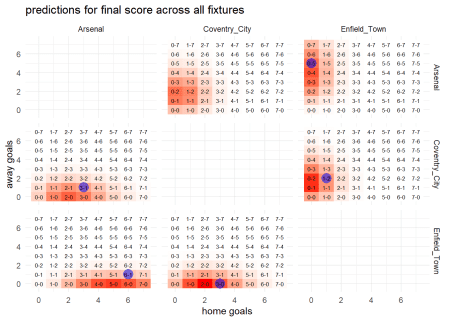
y = "away goals",

x = "home goals") +

theme\_minimal() +

facet\_grid(away~home, scales = "free")

p3



For the whole matrix, click [here](https://www.robert-hickman.eu/img/results_matrix.png)

**So what?**

These graphs are nice, but whats important is what they show: *we have a way to quantify how likely any result is in a match between two given teams*. Why is this useful

* Firstly, we can use the output of this to build betting models. Given the odds on final scores for any match, we can hedge effectively by betting on (e.g.) the five overwhelmingly most likely results.
* Secondly, we can simulate leagues. This is [perhaps especially of interest given the context of writing this post](https://www.bloomberg.com/graphics/2020-coronavirus-european-football/). I’m going to focus on this application because I don’t bet on football, and also because it’s hard to get a nice database of odds at the moment given the aforementioned situation.

We can do this using a technique called [Monte Carlo simulation](https://en.wikipedia.org/wiki/Monte_Carlo_method). There are lots of good explanation of the technique on the internet, but it basically boils down to this:

**“if events follow a known distribution\*, you can sample these events lots of times to get stochastic guesstimates, but over many samples you will reproduce exactly that distribution”**

\*a Poisson distribution for the expected number of goals scored in our case

For football, this means that while on an individual match level results are noisy (sometimes better teams lose!), if we simulate matches lots and lots of times, eventually they should converge to the ‘truth’\*

\*as defined by our Poisson distribution (which may or may not be a good/accurate ‘truth’ but go with it for now).

To work with this highly repetitive data, first we want to ‘nest’ the probabilities for each match. This basically means storing a df of all the possible results and their probabilities as a column inside a larger df so we can move between the data in those two structures easier.

For instance, the nest match results probability information for the next match to be played (Coventry City and home to Arsenal):

nested\_probabilities <- all\_predictions %>%

filter([is.na](http://is.na)(hgoal)) %>%

select(-hgoal, -agoal) %>%

nest(probabilities = c(sample\_hgoal, sample\_agoal, prob))

nested\_probabilities$probabilities[[1]] %>%

rename("Coventry City" = sample\_hgoal, "Arsenal" = sample\_agoal) %>%

arrange(-prob) %>%

#show first 15 rows

.[1:15,]

## # A tibble: 15 x 3

## `Coventry City` Arsenal prob

##

## 1 0 2 0.115

## 2 0 1 0.109

## 3 1 2 0.0983

## 4 1 1 0.0933

## 5 0 3 0.0806

## 6 1 3 0.0691

## 7 0 0 0.0516

## 8 1 0 0.0442

## 9 0 4 0.0425

## 10 2 2 0.0422

## 11 2 1 0.0400

## 12 1 4 0.0364

## 13 2 3 0.0296

## 14 2 0 0.0190

## 15 0 5 0.0179

The probability for any single result is small (otherwise match betting would be easy), but the probabilities for a 2-0 and 1-0 Arsenal wins are highest (as we found earlier). Indeed all of the most likely results are within a goal or two for either side of these.

To make sure these probabilities makes sense, we can sum them and see that the results space of 0:max\_goals for either side sums to 1

sum(nested\_probabilities$probabilities[[1]]$prob)

## [1] 1

Then we can easily use this data to simulate results. We sample a single row (a ‘result’ of the match) weighted by the probability of it occurring. For instance, when we sample from the Coventry City vs Arsenal match it picks a 3-1 Arsenal away win (not the likeliest result, but not the most unlikely either).

nested\_probabilities$probabilities[[1]] %>%

rename("Coventry\_City" = sample\_hgoal, "Arsenal" = sample\_agoal) %>%

sample\_n(1, weight = prob)

## # A tibble: 1 x 3

## Coventry\_City Arsenal prob

##

## 1 1 3 0.0691

We can of course repeat this across every match and see that the probabilities of the chosen results vary (because we’re randomly sampling we won’t always choose the most likely, or even a likely result), but all are within a reasonable range given the team playing:

nested\_probabilities %>%

mutate(sampled\_result = map(probabilities, sample\_n, 1, weight = prob)) %>%

select(-probabilities) %>%

unnest(cols = c(sampled\_result))

## # A tibble: 6 x 6

## home away gameweek sample\_hgoal sample\_agoal prob

##

## 1 Coventry\_City Arsenal 9 0 5 0.0179

## 2 Blackburn\_Rovers Dover\_Athletic 9 1 1 0.0575

## 3 Frimley\_Green Enfield\_Town 9 0 4 0.0418

## 4 Arsenal Blackburn\_Rovers 10 2 1 0.0966

## 5 Coventry\_City Frimley\_Green 10 3 0 0.170

## 6 Dover\_Athletic Enfield\_Town 10 2 1 0.0839

But when we are predicting what will happen, we want to find the *most likely* result. As mentioned earlier, if we sample enough, our average will converge towards this, so we can repeat this sampling technique n times (here I’ve done it 10 times), depending on how much time we want to wait for it to process.

You can see that as we do this many times, the results with the highest probability turn up more than others- as we would expect if we were to (e.g.) actually play Blackburn Rovers vs Arsenal many times.

rerun(10, nested\_probabilities %>%

filter(home == "Coventry\_City" & away == "Arsenal") %>%

mutate(sampled\_result = map(probabilities, sample\_n, 1, weight = prob)) %>%

select(-probabilities) %>%

unnest(cols = c(sampled\_result))

) %>%

bind\_rows() %>%

arrange(-prob)

## # A tibble: 10 x 6

## home away gameweek sample\_hgoal sample\_agoal prob

##

## 1 Coventry\_City Arsenal 9 0 2 0.115

## 2 Coventry\_City Arsenal 9 1 2 0.0983

## 3 Coventry\_City Arsenal 9 1 1 0.0933

## 4 Coventry\_City Arsenal 9 0 3 0.0806

## 5 Coventry\_City Arsenal 9 1 3 0.0691

## 6 Coventry\_City Arsenal 9 1 0 0.0442

## 7 Coventry\_City Arsenal 9 0 4 0.0425

## 8 Coventry\_City Arsenal 9 0 4 0.0425

## 9 Coventry\_City Arsenal 9 0 4 0.0425

## 10 Coventry\_City Arsenal 9 1 5 0.0154

If we do this a few more times per fixture (here 100, for a better estimate I’d advise at least 10000- it should only take a few minutes), we can then start assigning points and goal difference to each team based upon the result we’ve sampled. E.g. if one sample predicts Arsenal to beat Blackburn Rovers 4-0, we assign 3 points to Arsenal and 0 points to Blackburn Rovers for that simulation and +4 and -4 goal difference respectively.

n <- 100

fixture\_sims <- rerun(n, nested\_probabilities %>%

mutate(sampled\_result = map(probabilities, sample\_n, 1, weight = prob)) %>%

select(-probabilities) %>%

unnest(cols = c(sampled\_result)) %>%

select(-gameweek, -prob) %>%

pivot\_longer(c(home, away), names\_to = "location", values\_to = "team") %>%

mutate(points = case\_when(

location == "home" & sample\_hgoal > sample\_agoal ~ 3,

location == "away" & sample\_agoal > sample\_hgoal ~ 3,

sample\_hgoal == sample\_agoal ~ 1,

TRUE ~ 0

)) %>%

mutate(gd = case\_when(

location == "home" ~ sample\_hgoal - sample\_agoal,

location == "away" ~ sample\_agoal - sample\_hgoal

)))

fixture\_sims[1]

## [[1]]

## # A tibble: 12 x 6

## sample\_hgoal sample\_agoal location team points gd

##

## 1 0 0 home Coventry\_City 1 0

## 2 0 0 away Arsenal 1 0

## 3 4 0 home Blackburn\_Rovers 3 4

## 4 4 0 away Dover\_Athletic 0 -4

## 5 0 0 home Frimley\_Green 1 0

## 6 0 0 away Enfield\_Town 1 0

## 7 3 0 home Arsenal 3 3

## 8 3 0 away Blackburn\_Rovers 0 -3

## 9 6 1 home Coventry\_City 3 5

## 10 6 1 away Frimley\_Green 0 -5

## 11 1 1 home Dover\_Athletic 1 0

## 12 1 1 away Enfield\_Town 1 0

We can then average the points and goal difference won in these sims across each team and see what teams are predicted to win across their fixtures.

fixture\_sims %>%

bind\_rows() %>%

group\_by(team) %>%

summarise(av\_points = sum(points)/n,

av\_gd = sum(gd) / n)

## # A tibble: 6 x 3

## team av\_points av\_gd

##

## 1 Arsenal 4.19 2.44

## 2 Blackburn\_Rovers 3.16 0.7

## 3 Coventry\_City 3.61 2.42

## 4 Dover\_Athletic 2.26 -1.26

## 5 Enfield\_Town 2.95 0.23

## 6 Frimley\_Green 0.6 -4.53

Where we can see that we expect Arsenal to win 4.19 out of a possible 6 points (with games remaining against Coventry and Blackburn Rovers they are expected to drop points but win at least one and probably draw the other). Coventry City are expected to also do well- probably because their final game is at home to Frimley Green, whereas Blackburn have tougher fixtures away at Arsenal and home to Dover Athletic.

We can then add this to the calculated points teams have *already* accrued to get a prediction of where teams will end the season position wise:

table <- results %>%

pivot\_longer(c(home, away), names\_to = "location", values\_to = "team") %>%

mutate(points = case\_when(

location == "home" & hgoal > agoal ~ 3,

location == "away" & agoal > hgoal ~ 3,

hgoal == agoal ~ 1,

TRUE ~ 0

)) %>%

mutate(gd = case\_when(

location == "home" ~ hgoal - agoal,

location == "away" ~ agoal - hgoal

)) %>%

group\_by(team) %>%

summarise(points = sum(points),

gd = sum(gd))

predicted\_finishes <- map\_df(fixture\_sims, function(simulated\_fixtures, table) {

simulated\_fixtures %>%

select(team, points, gd) %>%

bind\_rows(., table) %>%

group\_by(team) %>%

summarise(points = sum(points),

gd = sum(gd)) %>%

arrange(-points, -gd) %>%

mutate(predicted\_finish = 1:n())

}, table) %>%

group\_by(team, predicted\_finish) %>%

summarise(perc = n() / n)

predicted\_finishes

## # A tibble: 10 x 3

## # Groups: team [6]

## team predicted\_finish perc

##

## 1 Arsenal 1 0.82

## 2 Arsenal 2 0.18

## 3 Blackburn\_Rovers 1 0.18

## 4 Blackburn\_Rovers 2 0.82

## 5 Coventry\_City 3 0.97

## 6 Coventry\_City 4 0.03

## 7 Dover\_Athletic 3 0.03

## 8 Dover\_Athletic 4 0.97

## 9 Enfield\_Town 5 1

## 10 Frimley\_Green 6 1

Which gives Arsenal an 82% chance of finishing champions, with only a 18% chance Blackburn manage to leapfrog them into 1st place. Given there are only 2 matches left with teams designed to have fairly large gulfs in ability, it’s not surprising most of the final positions are nailed on- e.g. Enfield Town finish 5th in every single simulation:

p4 <- ggplot(predicted\_finishes, aes(x = predicted\_finish, y = perc, fill = team)) +

geom\_bar(stat = "identity", colour = "black") +

scale\_fill\_manual(values = c("red", "blue", "skyblue", "white", "dodgerblue4", "blue")) +

labs(

title = "Predicted finish position of teams",

subtitle = "with two gameweeks left to play",

y = "fraction of finishes",

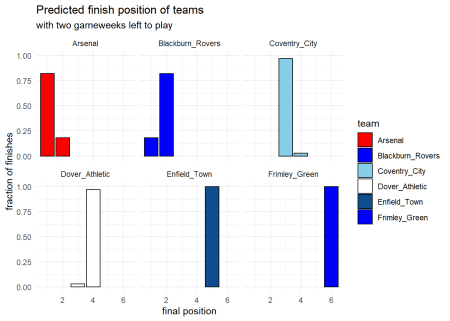
x = "final position"

) +

theme\_minimal() +

facet\_wrap(~team)

p4



**The Real Thing**

We’re now at the stage where we can start to look at real data. One of the motivating forces which drew me back to this putative blog series was the [current football situation](https://www.theguardian.com/football/blog/2020/apr/25/finishing-premier-league-season-pointless-football)– with season ending with games left to play.

We can use the knowledge we’ve built up over these last posts to see what we expect to happen in these unplayed games, if they cannot be completed.

The code following is also extremely similar to the final chunks of one of my previous [posts](https://www.robert-hickman.eu/post/five_min_trivia_invincibles/) in analysing the current Liverpool team’s achievements.

library(rvest)

library(regista)

First we need to download the data on the current English Premier League season. Once we have this we can split it into played matches (where we 100% know the result) and unplayed matches which we need to predict the result of. For the basis of the team strength estimates I’ve used the xg created and allowed per game, as I believe these give a better estimate of team strength

#download the match data from 2019/2020

fixtures\_2020 <- "<https://fbref.com/en/comps/9/schedule/Premier-League-Fixtures>" %>%

read\_html() %>%

html\_nodes("#sched\_ks\_3232\_1") %>%

html\_table() %>%

as.data.frame() %>%

separate(Score, into = c("hgoal", "agoal"), sep = "–") %>%

#only care about goals and expected goals

select(home = Home, away = Away, home\_xg = xG, away\_xg = xG.1, hgoal, agoal) %>%

filter(home != "") %>%

mutate(home = factor(home), away = factor(away)) %>%

#round expected goals to nearest integer

mutate\_at(c("home\_xg", "away\_xg", "hgoal", "agoal"), .funs = funs(round(as.numeric(.))))

#matches with a known result

#used for modelling

played\_matches <- fixtures\_2020 %>%

filter(![is.na](http://is.na)(home\_xg))

#matches with an unknown result

#used for simulation

unplayed\_matches <- fixtures\_2020 %>%

filter([is.na](http://is.na)(home\_xg)) %>%

select\_if(negate(is.numeric))

#fit the dixon coles model

#use xg per game, not 'actual' goals

fit\_2020 <- dixoncoles(home\_xg, away\_xg, home, away, data = played\_matches)

To get a look at what the team parameters in a real-life league look like we can extract them from the model and plot them:

#extract Dixon-Coles team strenth parameters

pars\_2020 <- fit\_2020$par %>%

.[grepl("def\_|off\_", names(.))] %>%

matrix(., ncol = 2) %>%

as.data.frame() %>%

rename(attack = V1, defence = V2)

pars\_2020$team <- unique(gsub("def\_\*|off\_\*", "", names(fit\_2020$par)))[1:20]

#plot as before

p5 <- pars\_2020 %>%

mutate(defence = 1 - defence) %>%

ggplot(aes(x = attack, y = defence, colour = attack + defence, label = team)) +

geom\_point(size = 3, alpha = 0.7) +

geom\_text\_repel() +

scale\_colour\_continuous(guide = FALSE) +

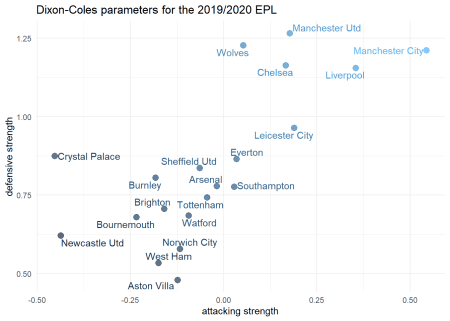
labs(title = "Dixon-Coles parameters for the 2019/2020 EPL",

x = "attacking strength",

y = "defensive strength") +

theme\_minimal()

p5

  
It might surprise some that Manchester City are predicted to be better than Liverpool by this model, but it shouldn’t given the underlying numbers for both teams. Liverpool have run very hot and Manchester City have run very cold this season.

Finally, we can then calculate the current Premier League table, and simulate remaining games to predict where teams will finish the season if the remainder of games were to be played. I’ve chosen 1000 sims just for sake of processing time, but you can scale up and down as desired.

#calculate the current EPL table

current\_epl\_table <- played\_matches %>%

select(home, away, hgoal, agoal) %>%

pivot\_longer(c(home, away), names\_to = "location", values\_to = "team") %>%

mutate(points = case\_when(

location == "home" & hgoal > agoal ~ 3,

location == "away" & agoal > hgoal ~ 3,

hgoal == agoal ~ 1,

TRUE ~ 0

)) %>%

mutate(gd = case\_when(

location == "home" ~ hgoal - agoal,

location == "away" ~ agoal - hgoal

)) %>%

group\_by(team) %>%

summarise(points = sum(points),

gd = sum(gd))

#the number of sims to run

n <- 10000

#simulate remaining matches

fixture\_sims\_2020 <- rerun(

n,

augment.dixoncoles(fit\_2020, unplayed\_matches, type.predict = "scorelines") %>%

mutate(sampled\_result = map(.scorelines, sample\_n, 1, weight = prob)) %>%

select(-.scorelines) %>%

unnest(cols = c(sampled\_result)) %>%

pivot\_longer(c(home, away), names\_to = "location", values\_to = "team") %>%

mutate(points = case\_when(

location == "home" & hgoal > agoal ~ 3,

location == "away" & agoal > hgoal ~ 3,

hgoal == agoal ~ 1,

TRUE ~ 0

)) %>%

mutate(gd = case\_when(

location == "home" ~ hgoal - agoal,

location == "away" ~ agoal - hgoal

)) %>%

select(team, points, gd))

#calculate final EPL tables

predicted\_finishes\_2020 <- map\_df(fixture\_sims\_2020, function(sim\_fixtures, table) {

sim\_fixtures %>%

select(team, points, gd) %>%

bind\_rows(., table) %>%

group\_by(team) %>%

summarise(points = sum(points),

gd = sum(gd)) %>%

arrange(-points, -gd) %>%

mutate(predicted\_finish = 1:n())

}, current\_epl\_table) %>%

group\_by(team, predicted\_finish) %>%

summarise(perc = n() / n) %>%

group\_by(team) %>%

mutate(mean\_finish = mean(predicted\_finish)) %>%

arrange(mean\_finish) %>%

ungroup() %>%

mutate(team = factor(team, levels = unique(team)))

If we then plot these predicted finishes (ordered by the chance of their highest finish position), we can get an idea of where we expect teams to end the season:

#list of team colours

team\_cols <- c("red", "skyblue", "darkblue", "darkblue", "darkred",

"orange", "red", "white", "red", "blue", "maroon",

"blue", "white", "red", "dodgerblue", "yellow",

"maroon", "red", "maroon", "yellow")

#plot the finishing position by chance based on these simualtions

p6 <- ggplot(predicted\_finishes\_2020,

aes(x = predicted\_finish, y = perc, fill = team)) +

geom\_bar(stat = "identity", colour = "black") +

scale\_fill\_manual(values = team\_cols, guide = FALSE) +

labs(

title = "Predicted finish position of teams",

subtitle = "for incomplete 2019/2020 EPL season",

y = "fraction of finishes",

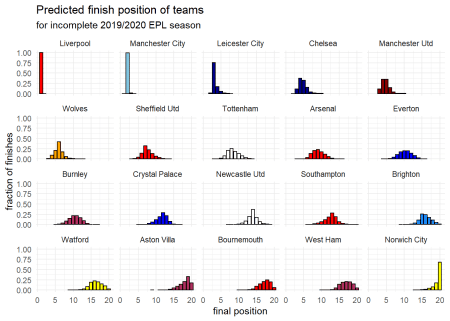
x = "final position"

) +

theme\_minimal() +

facet\_wrap(~team)

p6



So great news for Liverpool fans who the model believes have a 100% chance of finishing in first place. Leicester also might be happy with a nailed on 3rd place, with Chelsea or Manchester United probably rounding out the top four, and Wolves joining the loser of the two in the Europa League.

#get the predictions for the 2019/2020 champion

winner <- predicted\_finishes\_2020 %>%

filter(predicted\_finish < 2)%>%

mutate(prediction = "Champion chance")

winner

## # A tibble: 2 x 5

## team predicted\_finish perc mean\_finish prediction

##

## 1 Liverpool 1 1.00 1.5 Champion chance

## 2 Manchester City 1 0.0001 2.5 Champion chance

#get prediction for those who qualify for champions league

#and for europa league

champs\_league <- predicted\_finishes\_2020 %>%

filter(predicted\_finish < 5) %>%

group\_by(team) %>%

summarise(perc = sum(perc)) %>%

arrange(-perc) %>%

mutate(prediction = "Champions League chance")

champs\_league

## # A tibble: 10 x 3

## team perc prediction

##

## 1 Liverpool 1 Champions League chance

## 2 Manchester City 1 Champions League chance

## 3 Leicester City 0.933 Champions League chance

## 4 Chelsea 0.479 Champions League chance

## 5 Manchester Utd 0.46 Champions League chance

## 6 Wolves 0.106 Champions League chance

## 7 Sheffield Utd 0.0155 Champions League chance

## 8 Tottenham 0.004 Champions League chance

## 9 Arsenal 0.0018 Champions League chance

## 10 Everton 0.0005 Champions League chance

europa\_league <- predicted\_finishes\_2020 %>%

filter(predicted\_finish < 7) %>%

group\_by(team) %>%

summarise(perc = sum(perc)) %>%

arrange(-perc) %>%

mutate(prediction = "(at least) Europa League chance")

europa\_league

## # A tibble: 13 x 3

## team perc prediction

##

## 1 Liverpool 1 (at least) Europa League chance

## 2 Manchester City 1 (at least) Europa League chance

## 3 Leicester City 0.999 (at least) Europa League chance

## 4 Manchester Utd 0.954 (at least) Europa League chance

## 5 Chelsea 0.954 (at least) Europa League chance

## 6 Wolves 0.729 (at least) Europa League chance

## 7 Sheffield Utd 0.196 (at least) Europa League chance

## 8 Tottenham 0.096 (at least) Europa League chance

## 9 Arsenal 0.0479 (at least) Europa League chance

## 10 Everton 0.0139 (at least) Europa League chance

## 11 Burnley 0.0089 (at least) Europa League chance

## 12 Crystal Palace 0.0009 (at least) Europa League chance

## 13 Southampton 0.0008 (at least) Europa League chance

(obviously this model does not account for any ramifications of [Manchester City’s European ban](https://www.itv.com/news/2020-02-14/manchester-city-banned-from-champions-league-for-two-years-by-uefa/))

At the foot of the table, the model is fairly bullish on Norwich being relegated, with Aston Villa probably joining them, and then probably West Ham rounding out the relegation spots.

#get predictions for those who would be relegated

relegated <- predicted\_finishes\_2020 %>%

filter(predicted\_finish > 17) %>%

group\_by(team) %>%

summarise(perc = sum(perc)) %>%

arrange(-perc) %>%

mutate(prediction = "Relegation chance")

relegated

## # A tibble: 8 x 3

## team perc prediction

##

## 1 Norwich City 0.934 Relegation chance

## 2 Aston Villa 0.700 Relegation chance

## 3 Bournemouth 0.507 Relegation chance

## 4 West Ham 0.402 Relegation chance

## 5 Watford 0.270 Relegation chance

## 6 Brighton 0.171 Relegation chance

## 7 Newcastle Utd 0.0126 Relegation chance

## 8 Southampton 0.00270 Relegation chance

**Final Remarks**

I want to make it clear at the end of this post that this probably isn’t the most sophisticated model for predicting football matches (more to come in a part 3, maybe this time within less than a year), but does a pretty good job!