Fantasy Football

Andrew Little

25/01/2021

As a markov chain ——————————————————————————-

At each gameweek, we have 11 players that will play. Let's assume that these are the only players we have control of (i.e. no subs).

There are N total players on the game (N~650), and given some conditions, are available to be bought.

We will start assuming that we must make one transfer or less each gameweek, unlike in Fantasy football where you can pay 4 points to do an extra.

On gameweek one, the state of our team is:

- $\underline{s_1} = \{s_{1j}\}_{j=1}^{j=11}$

i.e. $\underline{s_1}$ is an eleven-vector

Here, we let s_{ij} represent the player in the j^{th} player slot on gameweek i.

Let's fix $\underline{s_1}$:

$$\underline{s_1} \equiv \{s_{1_1}, s_{1_2}, ..., s_{1_{11}}\} \in \{1, 2, ..., N\}^{11}$$

- Note that we cannot select a single player more than once, and so we have for fixed i:

- $\forall (j,k) \in \{1, 2, ..., 11\}^2$, $s_{i_j} \neq s_{i_k}$ for $j \neq k$.

For gameweek 2, we can now transition to another state. Essentially, we have the following two choices:

- We could swap out one $s_{1_j} \in \{1, 2, ..., N\}$ for s_{2_j}.
- We could make no transfers so that $\underline{s_1} = \underline{s_2}$
- We have $11(N - 11) \sim 7029$ choices for transfer as we can swap any of the 11 players out, and for each player that we swap out, we have $N - 11$ choices, as we cannnot swap for the same people.

We must filter out: - The players that are out of pay range, given who they are substituted for. - Therefore we will need to apply the filter for each player, and then bind the tables together.

Finding the possible transfers

We will first give a toy example, assuming we are in gameweek one. To do this we will use the {fplr} package.

First we give ourselves a random team of 11 to begin with.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(fplr)
# Create a dataframe of players and teams simple information
team_lookup <- fpl_get_teams() %>% select(name, id)

player_lookup <- fpl_get_player_all() %>%
  left_join(team_lookup, c("team" = "id")) %>%
  select(-team) %>%
  rename("team_name" = name) %>%
  select(first_name, second_name, team_name, id)
```

```r
set.seed(1871)
s_1 <- fpl_get_player_all() %>%
  sample_n(11, replace = FALSE) %>%
  left_join(team_lookup, c("team" = "id")) %>%
  select(-team) %>%
  rename("team_name" = name) %>%
  select(first_name, second_name, team_name, id, everything())
s_1
```

```
## # A tibble: 11 x 67
##    first_name second_name team_name    id chance_of_playi~ chance_of_playi~
##    <chr>      <chr>       <chr>     <int>            <int>            <int>
##  1 Shane      Long        Southamp~   361               NA               NA
##  2 Matt       Targett     Aston Vi~    42               NA               NA
##  3 Michael    Obafemi     Southamp~   381               25               25
##  4 Matteo     Guendouzi   Arsenal      25                0                0
##  5 Joël       Veltman     Brighton     67               NA               NA
##  6 Erik       Pieters     Burnley      83              100              100
##  7 Ellis      Simms       Everton     596                0                0
##  8 Danny      Ings        Southamp~   366              100              100
##  9 James      Trafford    Man City    595               NA               NA
## 10 Shane      Duffy       Brighton     58                0                0
## 11 John       Ruddy       Wolves      453               NA               NA
## # ... with 61 more variables: code <int>, cost_change_event <dbl>,
## #   cost_change_event_fall <int>, cost_change_start <dbl>,
## #   cost_change_start_fall <int>, dreamteam_count <int>, element_type <int>,
## #   ep_next <dbl>, ep_this <dbl>, event_points <int>, form <dbl>,
## #   in_dreamteam <lgl>, news <chr>, news_added <chr>, now_cost <dbl>,
```

```
## #   photo <chr>, points_per_game <dbl>, selected_by_percent <dbl>,
## #   special <lgl>, squad_number <lgl>, status <chr>, team_code <int>,
## #   total_points <int>, transfers_in <int>, transfers_in_event <int>,
## #   transfers_out <int>, transfers_out_event <int>, value_form <dbl>,
## #   value_season <dbl>, web_name <chr>, minutes <int>, goals_scored <int>,
## #   assists <int>, clean_sheets <int>, goals_conceded <int>, own_goals <int>,
## #   penalties_saved <int>, penalties_missed <int>, yellow_cards <int>,
## #   red_cards <int>, saves <int>, bonus <int>, bps <int>, influence <dbl>,
## #   creativity <dbl>, threat <dbl>, ict_index <dbl>, influence_rank <int>,
## #   influence_rank_type <int>, creativity_rank <int>,
## #   creativity_rank_type <int>, threat_rank <int>, threat_rank_type <int>,
## #   ict_index_rank <int>, ict_index_rank_type <int>,
## #   corners_and_indirect_freekicks_order <int>,
## #   corners_and_indirect_freekicks_text <chr>, direct_freekicks_order <int>,
## #   direct_freekicks_text <chr>, penalties_order <int>, penalties_text <chr>
```

Okay so we now have 11 players. Therefore the following transfers are:

```
library(purrr)

# Find players that do not have an id in s1 and then give current id as id_out. Then bind rows.
transfers <- s_1$id %>%
  map(function(id) {
    transf <- player_lookup %>%
      rename("id_in" = id) %>% # here id is not the argument to map, but the column in player_lookup
      filter(!(id_in %in% s_1$id)) %>%
      mutate("id_out" = id)
  }) %>%
  bind_rows()
transfers
```

```
## # A tibble: 6,941 x 5
##    first_name     second_name          team_name id_in id_out
##    <chr>          <chr>                <chr>     <int>  <int>
##  1 Mesut          Özil                 Arsenal       1    361
##  2 Sokratis       Papastathopoulos     Arsenal       2    361
##  3 David          Luiz Moreira Marinho Arsenal       3    361
##  4 Pierre-Emerick Aubameyang           Arsenal       4    361
##  5 Cédric         Soares               Arsenal       5    361
##  6 Alexandre      Lacazette            Arsenal       6    361
##  7 Shkodran       Mustafi              Arsenal       7    361
##  8 Bernd          Leno                 Arsenal       8    361
##  9 Granit         Xhaka                Arsenal       9    361
## 10 Pablo          Marí                 Arsenal      10    361
## # ... with 6,931 more rows
```

Note how we have 642 players to begin with, and thus should expect $11 * (642 - 11)$ total available transfer choices. This is what we have:

```
nrow(transfers)
```

```
## [1] 6941
```

Now let's give a cost to each transfer. Clearly, this is 'player in cost' - 'player out cost':

```
player_costs <- select(fpl_get_player_all(), id, now_cost)

transfers <- transfers %>%
  left_join(player_costs,
            by = c("id_in" = "id")) %>%
  rename("now_cost_in" = now_cost) %>%
  left_join(player_costs,
            by = c("id_out" = "id")) %>%
  rename("now_cost_out" = now_cost) %>%
  mutate("transfer_cost" = now_cost_in - now_cost_out)
transfers %>%
  select(transfer_cost, everything())
```

```
## # A tibble: 6,941 x 8
##    transfer_cost first_name second_name team_name id_in id_out now_cost_in
##            <dbl> <chr>      <chr>       <chr>     <int>  <int>       <dbl>
##  1         1.6   Mesut      Özil        Arsenal       1    361         6.7
##  2        -0.300 Sokratis   Papastatho~ Arsenal       2    361         4.8
##  3         0.3   David      Luiz Morei~ Arsenal       3    361         5.4
##  4         6.2   Pierre-Em~ Aubameyang  Arsenal       4    361        11.3
##  5        -0.5   Cédric     Soares      Arsenal       5    361         4.6
##  6         3.2   Alexandre  Lacazette   Arsenal       6    361         8.3
##  7        -0.100 Shkodran   Mustafi     Arsenal       7    361         5
##  8        -0.100 Bernd      Leno        Arsenal       8    361         5
##  9         0.1   Granit     Xhaka       Arsenal       9    361         5.2
## 10        -0.700 Pablo      Marí        Arsenal      10    361         4.4
## # ... with 6,931 more rows, and 1 more variable: now_cost_out <dbl>
```

Assume we already have a model for player's points

We now assume that we have modelled, elsewhere, the expected number of points a player will score in a particular gameweek. Number of points in gameweek i is given by: $*p_i$

We give this expected number of points as