Check fit diagnostics, analyse results

GS Verhoeven

Summary

This Notebook contains the following analyses:

- MCMC diagnostics
  - screening for high Rhat's / low effective sample sizes.
  - MCMC error and run consistency
- Dot plots of fitted model parameters with credible (HPDI) intervals
- Analysis of the DM-statistic
- Bookmaker odds
  - Predictive quality (using RPS) over time
  - Longshot bias: Model calibration for low probability events
- Model checking (comparing average RPS between models by various strata, including over time)
- Appendix: Ranked probability score

Load packages

```
library(data.table)
library(ggplot2)
library(rstan)
library(cowplot)

rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

```
source("code/rankProbScore.R")
source("code/ppc_coverage_plot.R")
source("code/MakeTimeSeriesPlot.R")
source("code/Create_model_data_for_TS2.R")
source("code/addTeamIds.R")
source("code/create_league_table.R")
source("code/MakeEPLPlotAbility.R")
source("code/games_predicted_vs_actual_intervals.R")
source("code/ppc_coverage_plot.R")
source("code/calc_rps_scores.R")
source("code/odds_to_probability.R")
source("code/ReadfitsandCalculateRPS.R")
source("code/FitOneStepAhead.R")
source("code/ReadfitsandExtractCoefs.R")
```

```
NL_ALL <- readRDS("data/NL Eredivisie 2000-2018.rds")

# set 2017/2018 season apart
NL_17 <- NL_ALL[Date > as.Date("2017-07-01")]

NL_ALL <- NL_ALL[Date < as.Date("2017-07-01")]
setkey(NL_ALL, Date)
```

```
# add round and season
nrounds <- nrow(NL_ALL)/9
nseasons <- nrow(NL_ALL)/(9*34)
NL_ALL <- NL_ALL[, round := rep(1:nrounds, each = 9)]
NL_ALL <- NL_ALL[, season := rep(1:nseasons, each = 34*9)]

# prep 2017/2018 separately
setkey(NL_17, Date)
nrounds <- ceiling(nrow(NL_17)/9)
start_nr_round <- max(NL_ALL$round)+1

round_vec <- rep(start_nr_round:(start_nr_round + nrounds), each = 9)
NL_17 <- NL_17[, round := round_vec[1:nrow(NL_17)]]
NL_17 <- NL_17[, season := 18]

# add to NL_ALL
NL_ALL <- rbind(NL_ALL, NL_17)

setkey(NL_ALL, Date)

NL_ALL <- NL_ALL[, row_id := 1:nrow(NL_ALL)]
```

```
# model checking
# res <- readRDS("output/res_match_type.rds")
# res_upper_lower <- readRDS("output/res_upper_lower.RDS")
# res_upper_lower <- res_upper_lower[, ul_type_name := "Upper / Upper"]
# res_upper_lower <- res_upper_lower[upper_lower_type == 1, ul_type_name := "Upper / Lower"]
# res_upper_lower <- res_upper_lower[upper_lower_type == 0, ul_type_name := "Lower / Lower"]
#
# res_upper_lower_at <- readRDS("output/res_upper_lower_at.RDS")
# res_time <- readRDS("output/res_time.RDS")
```

MCMC Diagnostics

Analyse rhat values

We screen all model fits for low n_eff of high Rhat values. Each model has about 150.000 to 300.000 parameters sampled in total.

```
# # check n_eff for T-dist and skellam base models.
# fit <- readRDS("c:/testversleutel/FITS/fitX_0.rds")
# #print(fit, pars = c("a"))
# fit <- readRDS("c:/testversleutel/FITS/fit_sd_0.rds")
# options(max.print=1000000)
#print(fit)
# Check Rhat
#tidy_rhat_neff_res[, .(mean(Rhat), sd(Rhat)), .(varnames)][order(-V1)][1:10]

table(tidy_rhat_neff_res$Rhat > 1.07)
```

```
##
##   FALSE
```

```
## 2364413
```

```
table(tidy_rhat_neff_res$Rhat < 1.04 & tidy_rhat_neff_res$Rhat > 1.02)
```

```
##
##   FALSE    TRUE
## 2364376      37
```

```
table(tidy_rhat_neff_res$Rhat < 0.99)
```

```
##
##   FALSE
## 2364413
```

```
#tidy_rhat_neff_res[, .(sum((ifelse(Rhat > 1.01, 1, 0)))), .(model_nr, shift_nr)][order(-V1)]
tidy_rhat_neff_res[!(varnames %in%  c("sigma_a0", "tau_a")) & Rhat > 1.02, .(model_nr, shift_nr, varname
```

```
##     model_nr shift_nr    varnames
##  1:        4       27  sigma_a[5]
##  2:        4       48 sigma_a[16]
##  3:        4       54 sigma_a[16]
##  4:        5        6  sigma_a[5]
##  5:        6       55      a[1,7]
##  6:        6       55     a[1,20]
##  7:        6       55      a[2,7]
##  8:        6       55      a[3,7]
##  9:        6       55      a[4,7]
## 10:        6       55      a[5,7]
## 11:        6       55 sigma_a[20]
## 12:        8       44  sigma_a[3]
## 13:        8       44  sigma_a[5]
```

```
tidy_rhat_neff_res[(varnames %in%  c("sigma_a0", "tau_a")) & Rhat > 1.02, .N,]
```

```
## [1] 25
```

```
# model 6, shift 55 might have some problems. This is the model without home advantage.
```

Analyse N_eff values

```
N_total <- 3000
tidy_rhat_neff_res <- tidy_rhat_neff_res[, neff_ratio := n_eff/N_total]

tidy_rhat_neff_res[neff_ratio < 0.1 , .N, .(varnames, model_nr)]
```

```
##        varnames model_nr N
##  1:    sigma_a0        1 3
##  2:       tau_a        1 1
##  3:       tau_a        2 1
##  4:    sigma_a0        3 1
##  5: sigma_a[16]        4 1
##  6:      a[1,7]        6 2
##  7:    sigma_a0        6 5
##  8:      a[2,7]        6 1
##  9:      a[3,7]        6 1
## 10:      a[4,7]        6 1
```

```
## 11:    sigma_a0        7 1
## 12:       tau_a        8 3
## 13:    sigma_a0        8 2
## 14:    sigma_a0        9 2
## 15:    sigma_a0       14 3
```

```
tidy_rhat_neff_res[neff_ratio < 0.1 , .N,]
```

```
## [1] 28
```

```
tidy_rhat_neff_res[neff_ratio < 0.1 & Rhat >= 1.02,]
```

```
##     model_nr                                    name
##  1:        1                        T-dist original
##  2:        1                        T-dist original
##  3:        2               T-dist with AT advantage
##  4:        3                Skellam offense/defense
##  5:        6                           T-dist no HA
##  6:        6                           T-dist no HA
##  7:        6                           T-dist no HA
##  8:        6                           T-dist no HA
##  9:        6                           T-dist no HA
## 10:        6                           T-dist no HA
## 11:        6                           T-dist no HA
## 12:        8          T-dist with AT advantage run2
## 13:        8          T-dist with AT advantage run2
## 14:        8          T-dist with AT advantage run2
## 15:        8          T-dist with AT advantage run2
## 16:       14 Skellam, no zif, offense/defense, no mu
##                           stan_file         short_name
##  1:                  epl_model.stan               fitX
##  2:                  epl_model.stan               fitX
##  3:          epl_model_art_turf.stan         fit_epl_at
##  4:             skellam_dynamic.stan             fit_sd
##  5:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  6:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  7:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  8:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  9:             epl_model_no_ha.stan fit_epl_xtra_no_ha
## 10:             epl_model_no_ha.stan fit_epl_xtra_no_ha
## 11:             epl_model_no_ha.stan fit_epl_xtra_no_ha
## 12:          epl_model_art_turf.stan    fit_epl_at_xtra
## 13:          epl_model_art_turf.stan    fit_epl_at_xtra
## 14:          epl_model_art_turf.stan    fit_epl_at_xtra
## 15:          epl_model_art_turf.stan    fit_epl_at_xtra
## 16: skellam_dynamic_no_zif_no_mu.stan  fit_sd_nozif_nomu
##
##  1:                                                                                goal_difference_pred_rep, a
##  2:                                                                                goal_difference_pred_rep, a
##  3:                                                        goal_difference_pred_rep, a, b_home, b_prev
##  4: goal_difference_pred_rep, a_offense, a_defense, mixing_proportion, constant_mu, home_advantage, l
##  5:                                                                                   goal_difference_pre
##  6:                                                                                   goal_difference_pre
##  7:                                                                                   goal_difference_pre
##  8:                                                                                   goal_difference_pre
```

```
##  9:                                                                      goal_difference_pre
## 10:                                                                      goal_difference_pre
## 11:                                                                      goal_difference_pre
## 12:                                                  goal_difference_pred_rep, a, b_home, b_prev
## 13:                                                  goal_difference_pred_rep, a, b_home, b_prev
## 14:                                                  goal_difference_pred_rep, a, b_home, b_prev
## 15:                                                  goal_difference_pred_rep, a, b_home, b_prev
## 16:                                  goal_difference_pred_rep, a_offense, a_defense, home_advantage, l
##        dist ata ha part_pool n_ability zif mu
##  1:  T-dist   0  1         1         1   0  1
##  2:  T-dist   0  1         1         1   0  1
##  3:  T-dist   1  1         1         1   0  1
##  4: Skellam   0  1         1         2   1  1
##  5:  T-dist   0  0         1         1   0  1
##  6:  T-dist   0  0         1         1   0  1
##  7:  T-dist   0  0         1         1   0  1
##  8:  T-dist   0  0         1         1   0  1
##  9:  T-dist   0  0         1         1   0  1
## 10:  T-dist   0  0         1         1   0  1
## 11:  T-dist   0  0         1         1   0  1
## 12:  T-dist   1  1         1         1   0  1
## 13:  T-dist   1  1         1         1   0  1
## 14:  T-dist   1  1         1         1   0  1
## 15:  T-dist   1  1         1         1   0  1
## 16: Skellam   0  1         1         2   0  0
##                                    extract_pars active varnames
##  1:                       b_home, b_prev, sigma_y      0    tau_a
##  2:                       b_home, b_prev, sigma_y      0 sigma_a0
##  3:      b_home, b_prev, sigma_y, art_turf_effect      0    tau_a
##  4: constant_mu, home_advantage, mixing_proportion      0 sigma_a0
##  5:                               b_prev, sigma_y      0 sigma_a0
##  6:                               b_prev, sigma_y      0   a[1,7]
##  7:                               b_prev, sigma_y      0   a[2,7]
##  8:                               b_prev, sigma_y      0   a[3,7]
##  9:                               b_prev, sigma_y      0   a[4,7]
## 10:                               b_prev, sigma_y      0 sigma_a0
## 11:                               b_prev, sigma_y      0 sigma_a0
## 12:      b_home, b_prev, sigma_y, art_turf_effect      0    tau_a
## 13:      b_home, b_prev, sigma_y, art_turf_effect      0    tau_a
## 14:      b_home, b_prev, sigma_y, art_turf_effect      0 sigma_a0
## 15:      b_home, b_prev, sigma_y, art_turf_effect      0    tau_a
## 16:                                home_advantage      1 sigma_a0
##         n_eff     Rhat shift_nr neff_ratio
##  1: 228.21284 1.025167       48 0.07607095
##  2: 290.54254 1.029672       54 0.09684751
##  3: 297.02778 1.024664       31 0.09900926
##  4: 207.34276 1.034209       12 0.06911425
##  5: 244.35226 1.020664       24 0.08145075
##  6: 212.14751 1.029125       55 0.07071584
##  7: 229.13956 1.027080       55 0.07637985
##  8: 234.09124 1.026741       55 0.07803041
##  9: 270.58315 1.022938       55 0.09019438
## 10:  93.80422 1.066476       55 0.03126807
## 11: 283.71571 1.020514       61 0.09457190
```

```
## 12: 265.28116 1.023351       31 0.08842705
## 13: 287.32870 1.031538       44 0.09577623
## 14: 252.04925 1.037166       53 0.08401642
## 15: 209.44873 1.028745       54 0.06981624
## 16: 249.52097 1.030105       31 0.08317366
```

```
tidy_rhat_neff_res[neff_ratio < 0.1 & Rhat < 1.02,]
```

```
##     model_nr                                      name
##  1:        1                     T-dist original
##  2:        1                     T-dist original
##  3:        4                   T-dist no pooling
##  4:        6                       T-dist no HA
##  5:        6                       T-dist no HA
##  6:        6                       T-dist no HA
##  7:        7       Skellam, no zif, offense/defense
##  8:        8          T-dist with AT advantage run2
##  9:        9                          T-dist run2
## 10:        9                          T-dist run2
## 11:       14 Skellam, no zif, offense/defense, no mu
## 12:       14 Skellam, no zif, offense/defense, no mu
##                          stan_file         short_name
##  1:                  epl_model.stan               fitX
##  2:                  epl_model.stan               fitX
##  3:        epl_model_no_pooling.stan         fit_epl_np
##  4:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  5:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  6:             epl_model_no_ha.stan fit_epl_xtra_no_ha
##  7:      skellam_dynamic_no_zif.stan       fit_sd_nozif
##  8:          epl_model_art_turf.stan    fit_epl_at_xtra
##  9:                  epl_model.stan       fit_epl_xtra
## 10:                  epl_model.stan       fit_epl_xtra
## 11: skellam_dynamic_no_zif_no_mu.stan  fit_sd_nozif_nomu
## 12: skellam_dynamic_no_zif_no_mu.stan  fit_sd_nozif_nomu
##
##  1:                                                    goal_difference_pred_rep, a, b_home, b_prev, s
##  2:                                                    goal_difference_pred_rep, a, b_home, b_prev, s
##  3:                                                           goal_difference_pred_rep, a, b_home, b
##  4:                                                            goal_difference_pred_rep, a, b_prev, s
##  5:                                                            goal_difference_pred_rep, a, b_prev, s
##  6:                                                            goal_difference_pred_rep, a, b_prev, s
##  7: goal_difference_pred_rep, a_offense, a_defense, constant_mu, home_advantage, b_prev_offense, b_pr
##  8:                                      goal_difference_pred_rep, a, b_home, b_prev, sigma_a0, tau_a, r
##  9:                                                    goal_difference_pred_rep, a, b_home, b_prev, s
## 10:                                                    goal_difference_pred_rep, a, b_home, b_prev, s
## 11:             goal_difference_pred_rep, a_offense, a_defense, home_advantage, b_prev_offense, b_pr
## 12:             goal_difference_pred_rep, a_offense, a_defense, home_advantage, b_prev_offense, b_pr
##        dist ata ha part_pool n_ability zif mu
##  1:  T-dist   0  1         1         1   0  1
##  2:  T-dist   0  1         1         1   0  1
##  3:  T-dist   0  1         0         1   0  1
##  4:  T-dist   0  0         1         1   0  1
##  5:  T-dist   0  0         1         1   0  1
##  6:  T-dist   0  0         1         1   0  1
##  7: Skellam   0  1         1         2   0  1
```

```
##  8:  T-dist   1  1         1         1   0  1
##  9:  T-dist   0  1         1         1   0  1
## 10:  T-dist   0  1         1         1   0  1
## 11: Skellam   0  1         1         2   0  0
## 12: Skellam   0  1         1         2   0  0
##                                extract_pars active    varnames    n_eff
##  1:                  b_home, b_prev, sigma_y      0    sigma_a0 271.5525
##  2:                  b_home, b_prev, sigma_y      0    sigma_a0 288.2400
##  3:                  b_home, b_prev, sigma_y      0 sigma_a[16] 279.0365
##  4:                          b_prev, sigma_y      0      a[1,7] 279.1058
##  5:                          b_prev, sigma_y      0    sigma_a0 234.7308
##  6:                          b_prev, sigma_y      0    sigma_a0 264.7700
##  7:              constant_mu, home_advantage      0    sigma_a0 261.9756
##  8: b_home, b_prev, sigma_y, art_turf_effect      0    sigma_a0 246.2839
##  9:                  b_home, b_prev, sigma_y      0    sigma_a0 252.6708
## 10:                  b_home, b_prev, sigma_y      0    sigma_a0 285.2170
## 11:                           home_advantage      1    sigma_a0 284.1839
## 12:                           home_advantage      1    sigma_a0 299.4209
##         Rhat shift_nr neff_ratio
##  1: 1.012340       28 0.09051749
##  2: 1.017984       29 0.09608001
##  3: 1.015263       50 0.09301215
##  4: 1.013086       18 0.09303526
##  5: 1.015346       18 0.07824359
##  6: 1.014111       20 0.08825667
##  7: 1.010614       53 0.08732518
##  8: 1.019565       39 0.08209462
##  9: 1.019989        8 0.08422358
## 10: 1.017254       31 0.09507234
## 11: 1.010815        5 0.09472796
## 12: 1.019721       23 0.09980695
```

```
tidy_rhat_neff_res[, .(min = min(n_eff), mean(n_eff), sd(n_eff)), .(model_nr, varnames, name)][order(mi
```

```
##     model_nr    varnames                                     name       min
##  1:        6    sigma_a0                           T-dist no HA  93.80422
##  2:        3    sigma_a0                 Skellam offense/defense 207.34276
##  3:        8       tau_a          T-dist with AT advantage run2 209.44873
##  4:        6      a[1,7]                           T-dist no HA 212.14751
##  5:        1       tau_a                       T-dist original 228.21284
##  6:        6      a[2,7]                           T-dist no HA 229.13956
##  7:        6      a[3,7]                           T-dist no HA 234.09124
##  8:        8    sigma_a0          T-dist with AT advantage run2 246.28385
##  9:       14    sigma_a0 Skellam, no zif, offense/defense, no mu 249.52097
## 10:        9    sigma_a0                            T-dist run2 252.67075
## 11:        7    sigma_a0       Skellam, no zif, offense/defense 261.97555
## 12:        6      a[4,7]                           T-dist no HA 270.58315
## 13:        1    sigma_a0                       T-dist original 271.55248
## 14:        4 sigma_a[16]                     T-dist no pooling 279.03645
## 15:        2       tau_a               T-dist with AT advantage 297.02778
## 16:        6       tau_a                           T-dist no HA 302.20478
## 17:        2    sigma_a0               T-dist with AT advantage 305.61716
## 18:       10       tau_a        Skellam offense/defense with AT 323.57311
## 19:       10    sigma_a0        Skellam offense/defense with AT 336.19197
## 20:        9       tau_a                            T-dist run2 336.26167
```

```
##            V2       V3
##  1:  628.2825 199.4755
##  2:  675.1131 144.4530
##  3:  728.1778 223.5935
##  4:  948.3791 314.2630
##  5:  726.7944 186.0739
##  6: 1036.3374 334.7256
##  7: 1133.6828 344.7675
##  8:  661.7828 192.2468
##  9:  708.8033 246.2338
## 10:  634.7264 177.9515
## 11:  665.2367 181.0253
## 12: 1218.3793 346.4045
## 13:  661.5899 197.4209
## 14:  755.3142 241.0250
## 15:  732.0124 204.8314
## 16:  708.7668 198.9739
## 17:  700.4531 208.2168
## 18:  748.2083 171.1749
## 19:  676.4549 198.8015
## 20:  703.9500 172.1950
```

```
tidy_rhat_neff_res[varnames != "sigma_a0", .(min = min(n_eff), mean(n_eff), sd(n_eff)), .(model_nr, varr
```

```
##     model_nr    varnames                                      name      min
##  1:        8       tau_a          T-dist with AT advantage run2 209.4487
##  2:        6      a[1,7]                           T-dist no HA 212.1475
##  3:        1       tau_a                       T-dist original 228.2128
##  4:        6      a[2,7]                           T-dist no HA 229.1396
##  5:        6      a[3,7]                           T-dist no HA 234.0912
##  6:        6      a[4,7]                           T-dist no HA 270.5831
##  7:        4 sigma_a[16]                     T-dist no pooling 279.0365
##  8:        2       tau_a               T-dist with AT advantage 297.0278
##  9:        6       tau_a                           T-dist no HA 302.2048
## 10:       10       tau_a        Skellam offense/defense with AT 323.5731
## 11:        9       tau_a                            T-dist run2 336.2617
## 12:        3  sigma_a[7]                Skellam offense/defense 341.7934
## 13:        1  sigma_a[5]                       T-dist original 346.8952
## 14:        6      a[5,7]                           T-dist no HA 361.2007
## 15:        4 sigma_a[12]                     T-dist no pooling 375.0560
## 16:        8  sigma_a[3]          T-dist with AT advantage run2 376.5165
## 17:        1      a[1,7]                       T-dist original 389.3744
## 18:        8  sigma_a[5]          T-dist with AT advantage run2 393.0529
## 19:        8 sigma_a[14]          T-dist with AT advantage run2 407.4574
## 20:       14       tau_a Skellam, no zif, offense/defense, no mu 413.3849
##            V2       V3
##  1:  728.1778 223.5935
##  2:  948.3791 314.2630
##  3:  726.7944 186.0739
##  4: 1036.3374 334.7256
##  5: 1133.6828 344.7675
##  6: 1218.3793 346.4045
##  7:  755.3142 241.0250
##  8:  732.0124 204.8314
##  9:  708.7668 198.9739
```

```
## 10:  748.2083 171.1749
## 11:  703.9500 172.1950
## 12: 1037.9490 202.6038
## 13:  978.9988 254.7394
## 14: 1372.6232 442.2931
## 15:  766.1300 176.9277
## 16: 1087.5988 252.2231
## 17: 1051.3474 328.1904
## 18:  982.5901 262.7767
## 19: 1040.2268 277.4977
## 20:  780.5369 204.5053
```

```
#summary(tidy_rhat_neff_res[varnames == "tau_a" & model_nr == 10, ]$n_eff)
```

The parameters with the lowest n_eff are the sigma_a0, that we use to initialize the time series. And tau, the hyper prior for the hierarchical model.

MCMC error and run consistency

To check MCMC error / consistency of MCMC runs, we ran for two models the whole forecasting procedure (including fitting all out of sample weeks) twice and compared our main quantity of interest, the RPS score per match.

```
# check correlation between runs
rps_tdist <- NL_ALL_PRED[model_nr %in% c(1, 9) & has_pred == 1,.(matchKey, round, rps_vec, model_nr)]
rps_tdist_cast <- dcast(rps_tdist, matchKey + round ~ model_nr, value.var = "rps_vec")
cor(rps_tdist_cast[,"1", with = F], rps_tdist_cast[,"9", with = F])
```

```
##           9
## 1 0.9976148
```

```
# 0.998

# check correlation between runs
rps_tdist <- NL_ALL_PRED[model_nr %in% c(2, 8) & has_pred == 1,.(matchKey, round, rps_vec, model_nr)]
rps_tdist_cast <- dcast(rps_tdist, matchKey + round ~ model_nr, value.var = "rps_vec")
cor(rps_tdist_cast[,"2", with = F], rps_tdist_cast[,"8", with = F])
```

```
##           8
## 2 0.9979439
```

```
# 0.998
```

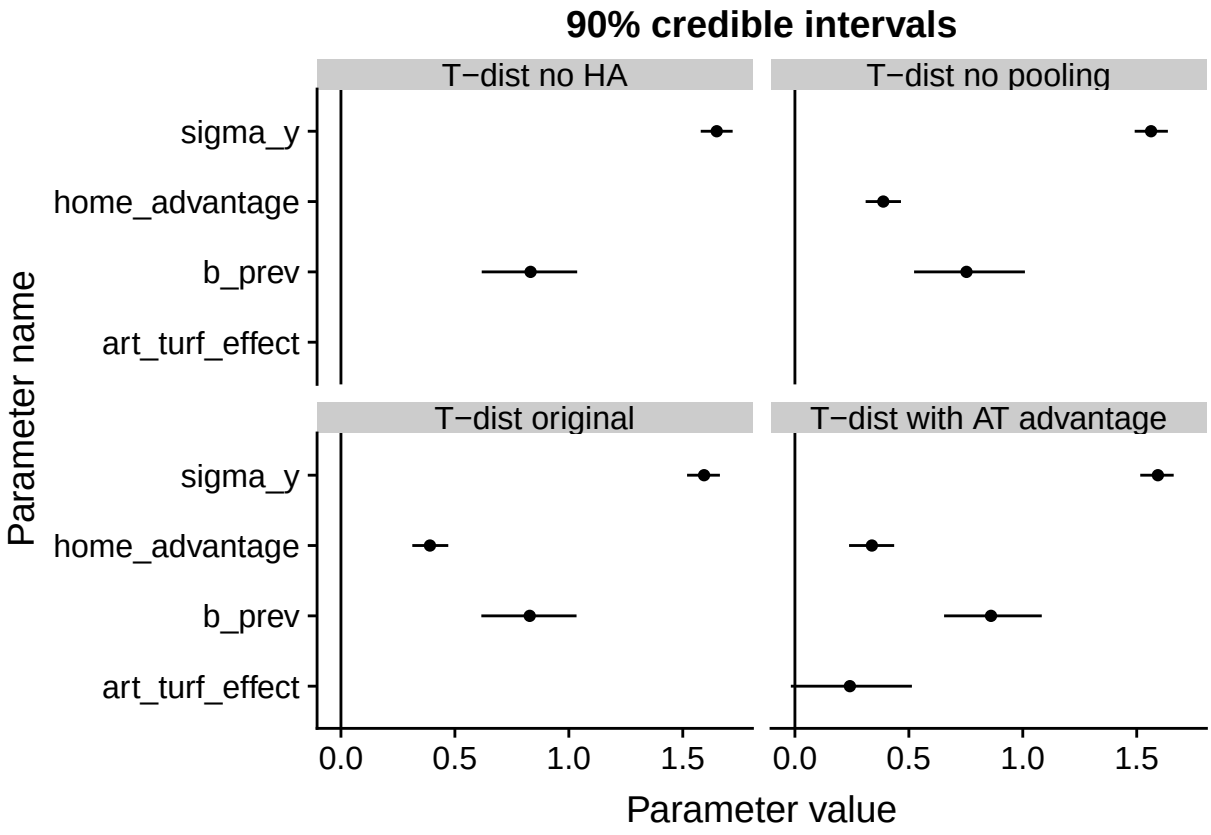
We judge that the correlation is sufficiently high for our purposes.

Panel plot of in-sample coefficients using four seasons of data

T-distibution models

Plot all coefs for all models in one big panel plot. Do this for the final model (ie.. shift_nr = 67)

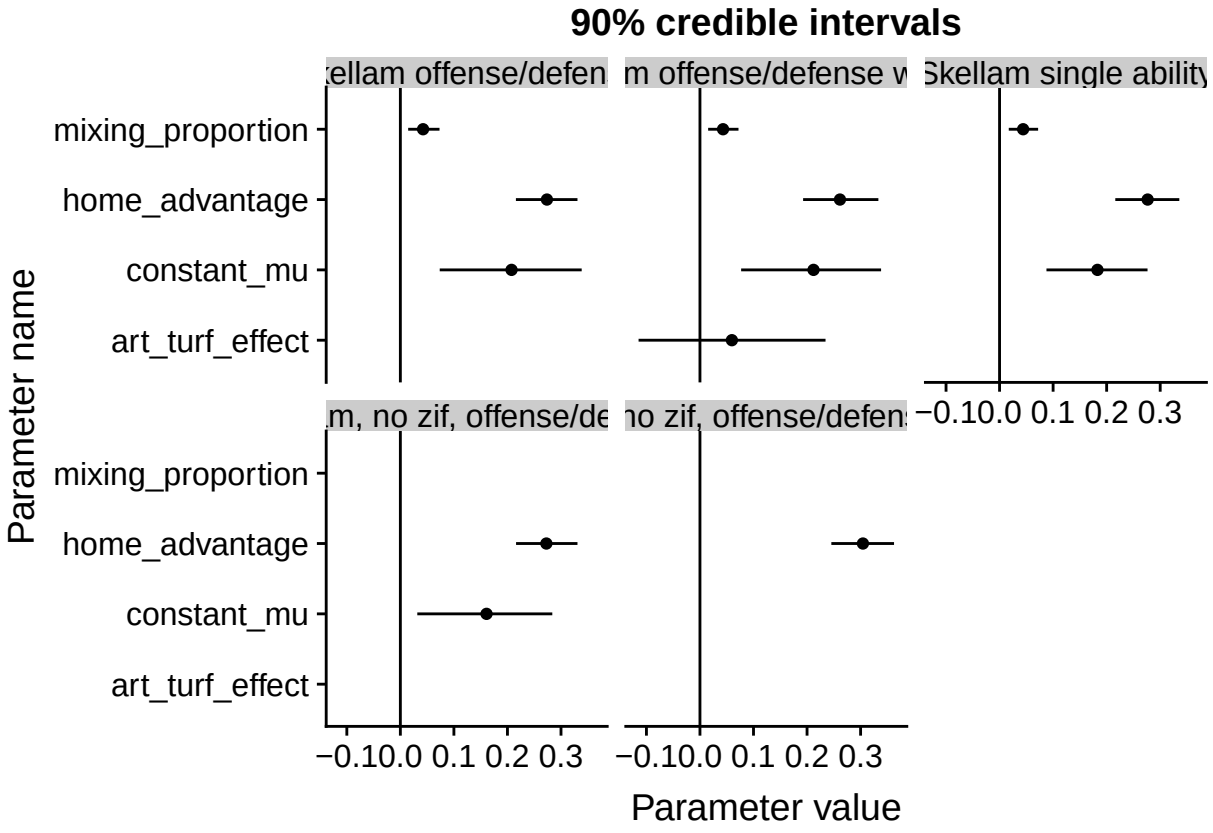
```
ggplot(tidy_coef_res[shift_nr == 67 & !(model_nr %in% c(8,9)) & dist == "T-dist",] , aes(x=L1, y = Q50)
  geom_linerange(aes(ymin = Q5, ymax = Q95)) +
  geom_line() + geom_point() + facet_wrap(~ name) + coord_flip() +
  geom_hline(yintercept = 0) + ggtitle("90% credible intervals") + ylab("Parameter value") + xlab("Para
```



Skellam models

Plot all coefs for all models in one big panel plot. Do this for the final model (ie.. shift_nr = 67)

```
ggplot(tidy_coef_res[shift_nr == 67 & !(model_nr %in% c(8,9)) & dist == "Skellam",] , aes(x=L1, y = Q50)
  geom_linerange(aes(ymin = Q5, ymax = Q95)) +
  geom_line() + geom_point() + facet_wrap(~ name) + coord_flip() +
  geom_hline(yintercept = 0) + ggtitle("90% credible intervals") + ylab("Parameter value") + xlab("Para
```



DM-like test for differences in predictions

Can we perform statistical testing to compare the RPS values from two models? (Constantinou et al plot the cumulative RPS difference.) Use the best bookmaker odds as reference.

```
knitr::kable(result_by_model[!(model_nr %in% c(8,9)), .(model_nr, name, DMstat_12, DMpval_12)][order(DM
```

model_nr	name	DMstat_12	DMpval_12
13	Equal probability odds	-8.443	0.000
4	T-dist no pooling	-2.986	0.003
6	T-dist no HA	-2.915	0.004
1	T-dist original	-1.727	0.085
2	T-dist with AT advantage	-1.715	0.087
5	Skellam single ability	-1.674	0.095
11	William_hill odds	-1.469	0.142
3	Skellam offense/defense	-1.440	0.150
10	Skellam offense/defense with AT	-1.427	0.154
14	Skellam, no zif, offense/defense, no mu	-1.405	0.161
7	Skellam, no zif, offense/defense	-1.283	0.200
12	Bet365 odds	NA	NA

So it appears that for both the partial pooling and the regular home advantage we have sufficient statistical power to conclude that those models perform worse in OOS predictions. As well as the equal probability

reference predictions.

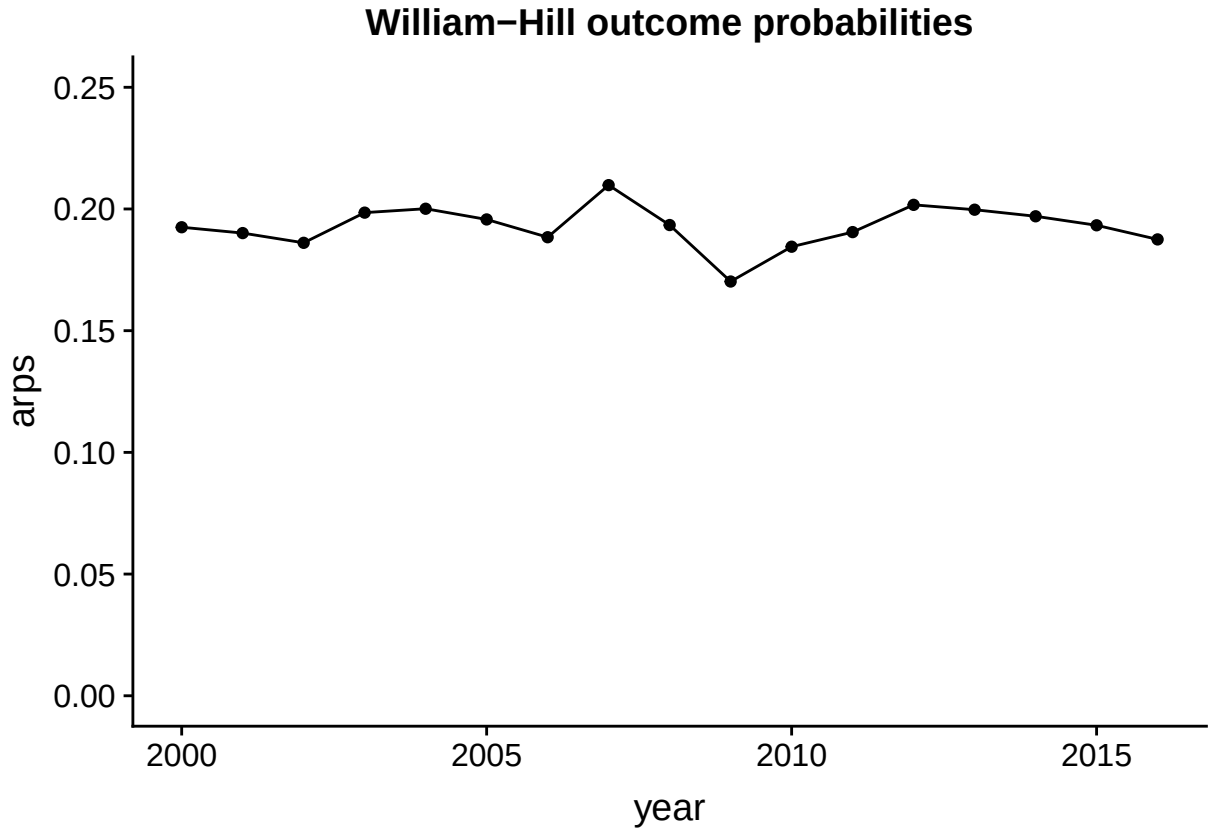
The other models can not reliably be distinguished based on 608 match predictions.

Analyse Bookmaker odds over time 2000-2017

Check year to year variability in the average RPS of William Hill. Apparently sometimes there are no odds for a match.

```
arps_season <-c()
for(i in 1:17){
  WH_probs <- odds_to_probability(NL_ALL[season %in% c(i), .(WHH, WHD, WHA)])
  actual_scorez <- Convert_actual_to_win_draw_loss_vector(NL_ALL[season %in% c(i),]$goal_difference)
  # select non-NA records
  selection <- !is.na(WH_probs$prob_win)
  arps_season[i] <- calculate_arps(WH_probs[selection,.( prob_win, prob_draw, prob_loss)], actual_scorez
}

ggplot(data.frame(year = 2000:2016, arps = arps_season), aes(x= year, y = arps)) +
  geom_point() + ylim(0, 0.25) + geom_line() + ggtitle("William-Hill outcome probabilities")
```



We conclude that there is a large variability from season to season ranging between 0.17 and 0.2.

Calculate Bookmaker benchmark for Lit et al study

Check what the Bookmaker average RPS benchmark is, ** exactly **, for the Lit study. They use the period 2009/2010 to 2015/2016, i.e. the average of seven seasons as out-of-sample. By comparing this with the performance of their best models, we can learn if they "beat the bookies".

```
# calculate aprs bookmakers for oos period Lit:
WH_probs <- odds_to_probability(NL_ALL[season %in% c(10:16), .(WHH, WHD, WHA)])
actual_scorez <- Convert_actual_to_win_draw_loss_vector(NL_ALL[season %in% c(10:16),]$goal_difference)
# select non-NA records
selection <- !is.na(WH_probs$prob_win)
print(calculate_arps(WH_probs[selection,.( prob_win, prob_draw, prob_loss)], actual_scorez[selection,.(a
```

```
## [1] 0.191
```

So Lit et al also do not "beat the bookies" with their best dynamic models (0.193) for the NL Eredivisie.

analyse long shot bias

There is some literature on "long shot bias" in bookmakers odds, where models could outperform the bookmakers for low-probability events, where the bettors would overvalue low-probability events. We interpret this as checking calibration for low-probability events. To get sufficient statistics, we choose the probability bin between 5% and 15%. Check all three outcomes.

```
NL_ALL_PRED[p_win >= 0.05 & p_win < 0.15, .(mean(act_win), .N), .(model_nr)][order(-N)]
```

```
##     model_nr         V1  N
##  1:        6 0.09803922 51
##  2:       12 0.08571429 35
##  3:        5 0.09677419 31
##  4:        8 0.09677419 31
##  5:       14 0.09677419 31
##  6:        2 0.10000000 30
##  7:       10 0.10000000 30
##  8:        3 0.10714286 28
##  9:       11 0.11538462 26
## 10:        1 0.11538462 26
## 11:        4 0.12000000 25
## 12:        9 0.12000000 25
## 13:        7 0.13043478 23
```

```
NL_ALL_PRED[p_draw >= 0.05 & p_draw < 0.15, .(mean(act_draw), .N), .(model_nr)][order(-N)]
```

```
##     model_nr        V1   N
##  1:        4 0.2158273 139
##  2:        2 0.2000000  75
##  3:        8 0.1830986  71
##  4:        1 0.1884058  69
##  5:        9 0.1940299  67
##  6:        6 0.2419355  62
##  7:       12 0.1707317  41
##  8:       14 0.2564103  39
##  9:        7 0.2432432  37
## 10:       11 0.2307692  26
## 11:        5 0.1538462  13
```

```
## 12:       10 0.2000000  10
## 13:        3 0.1250000   8
```

```
NL_ALL_PRED[p_loss >= 0.05 & p_loss < 0.15, .(mean(act_loss), .N), .(model_nr)][order(-N)]
```

```
##     model_nr         V1   N
##  1:       12 0.06306306 111
##  2:       10 0.09000000 100
##  3:       14 0.07070707  99
##  4:        5 0.08163265  98
##  5:        3 0.07368421  95
##  6:       11 0.05376344  93
##  7:        7 0.06818182  88
##  8:        9 0.06818182  88
##  9:        2 0.08045977  87
## 10:        4 0.05813953  86
## 11:        1 0.04878049  82
## 12:        8 0.07317073  82
## 13:        6 0.02000000  50
```

What sampling variation do we expect for these sample sizes? For 100 events with a probability of 0.1 of happening, we expect a range of values between 5 and 15, ie.. percentages of 0.05 to 0.15.

```
summary(rbinom(1000, 100, 0.1))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    8.00   10.00   10.09   12.00   20.00
```

```
summary(rbinom(1000, 50, 0.1))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   4.929   6.000  12.000
```

The actual frequency for low probability wins and losses are within acceptable range of sampling error if we assume that the predicted probabilities are correct. The frequencies for the low probability draws however are structurally too high (around 20% instead of 10%). I.e. they occur more often than predicted.

Model checking

The Rank Probability score quantifies the quality of our model-based forecasts. We can use this metric to learn about the performance of our models. The purpose is not to select the best model, but to identify strengths and weaknesses for each model, and how they compare with one another.

We stratify the matches along four dimensions and calculating average RPS for each stratum:

- Team type (Artficial turf or natural grass)
- playing surface type (Artificial turf or natural grass)
- Relative team strength
- time (Round number)

```
res <- NL_ALL_PRED[, .(arps = mean(rps_vec, na.rm = T),
                       N = length(na.omit(rps_vec))), .(match_type,model_nr)]
setkey(res, model_nr)
setkey(result_by_model, model_nr)
res <- result_by_model[res]
#res
```

```
# use in manuscript
saveRDS(res, "output/20180828 res_match_type.rds")
```

As a benchmark, we also plot the odds-derived predictions, as well as the before mentioned equal-probability (win, draw, loss) uninformed model.

T-distribution models stratified by relative team strength

We start with comparing the T-distribution models. We expect that matches where the team abilities are similar are more difficult to predict. Based on a two-season league table, we label each team as relatively "strong" when it ranks in the upper halve of the league table, and as relatively "weak" when it ends up in the lower halve of the ranking. With these labels, we then classify each match as a match between two strong teams, a match between a weak and a strong team, or a match between two weak team. For each of the three match types we calculate the average RPS for each model. The results are plotted in Figure **??**fig:figTdist). It should be noted that the "Lower / Lower" segment contains less matches (N = 110) than the "Upper / Upper" segment (N= 180) because of relegated and promoted teams between the two seasons.

```
ltab <- create_league_table(NL_ALL_PRED)
lower_half <- ltab[order(total_points)][1:10,]$HomeTeam
upper_half <- ltab[order(total_points)][11:20,]$HomeTeam

NL_ALL_PRED[, upper_lower_type := 0]
NL_ALL_PRED[HomeTeam %in% upper_half, upper_lower_type := upper_lower_type+1]
NL_ALL_PRED[AwayTeam %in% upper_half, upper_lower_type := upper_lower_type+1]

res_upper_lower <- NL_ALL_PRED[, .(arps = mean(rps_vec, na.rm = T),
                      N = length(na.omit(rps_vec))), .(upper_lower_type, model_nr)]
setkey(res_upper_lower, model_nr)
setkey(result_by_model, model_nr)
res_upper_lower <- result_by_model[res_upper_lower]

res_upper_lower <- res_upper_lower[, ul_type_name := "Upper / Upper"]
res_upper_lower <- res_upper_lower[upper_lower_type == 1, ul_type_name := "Upper / Lower"]
res_upper_lower <- res_upper_lower[upper_lower_type == 0, ul_type_name := "Lower / Lower"]

saveRDS(res_upper_lower, "output/20180828 res_upper_lower.RDS")
```

We first compare the base model (with home advantage and partial pooling) with odds-based predictions, as well as models that leave out these two elements.

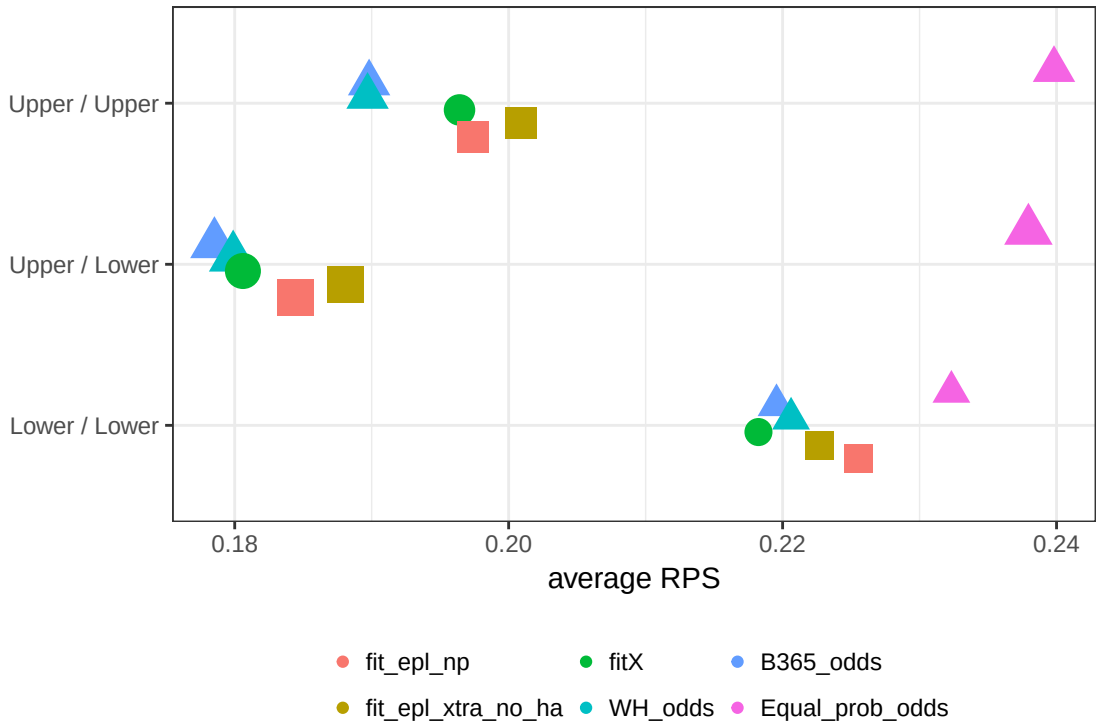


Figure 1: T-distribution models compared by match type.

Not including the regular home advantage, or leaving out the partial pooling caused by the hierarchical model results in worse predictions for all three match types (squared symbols). From here on we consider these two elements as "must haves" for optimal predictive performance, and no longer include them in the graphs. Compared to the odds-based predictions (triangle symbols), we find that only for matches between two lower ranking teams do the model predictions perform (slightly) better. The largest room for model improvement seems to be for matches between two "upper half" ranking teams.

Next we compare the base model with an expanded model that includes the artificial turf advantage (ATA) predictor. Figure @ref(fig:figTdistATA) shows both models.

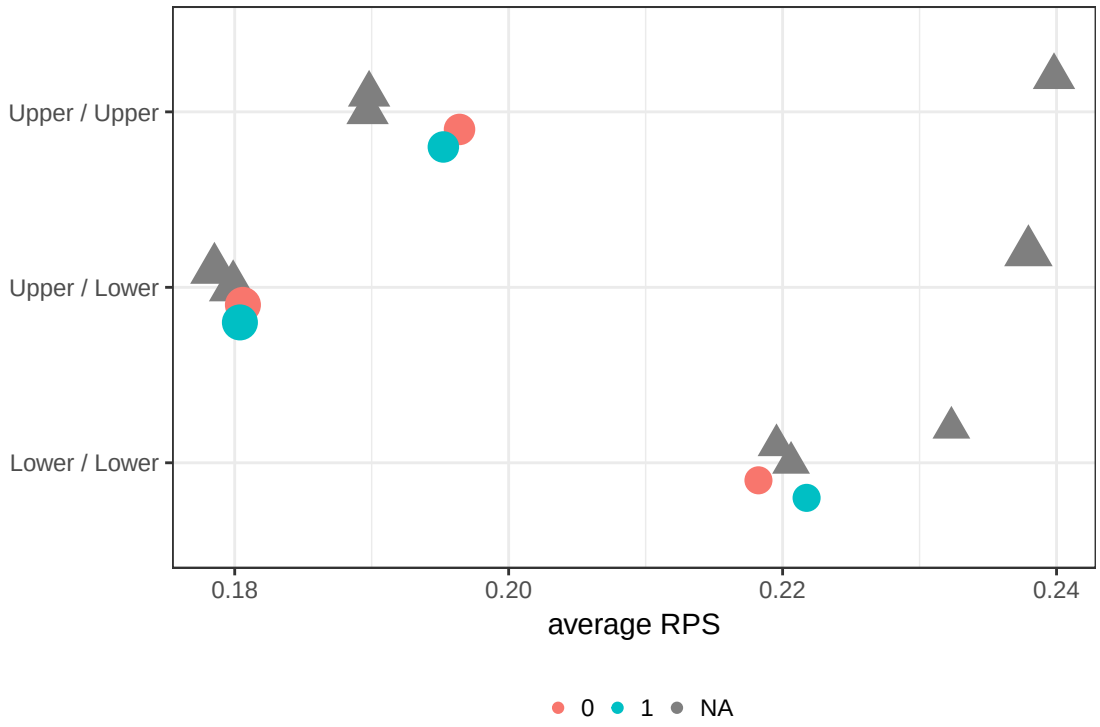


Figure 2: T-distribution model with and without Artificial turf advantage.

Including the ATA predictor slightly improves predictions for matches between upper half ranking teams (i.e. Heracles), but results in worse predictions for matches between lower half ranking teams. For matches between a strong and a weak team, the quality of the predictions is similar for both models.

T-distribution models by Team playing surface

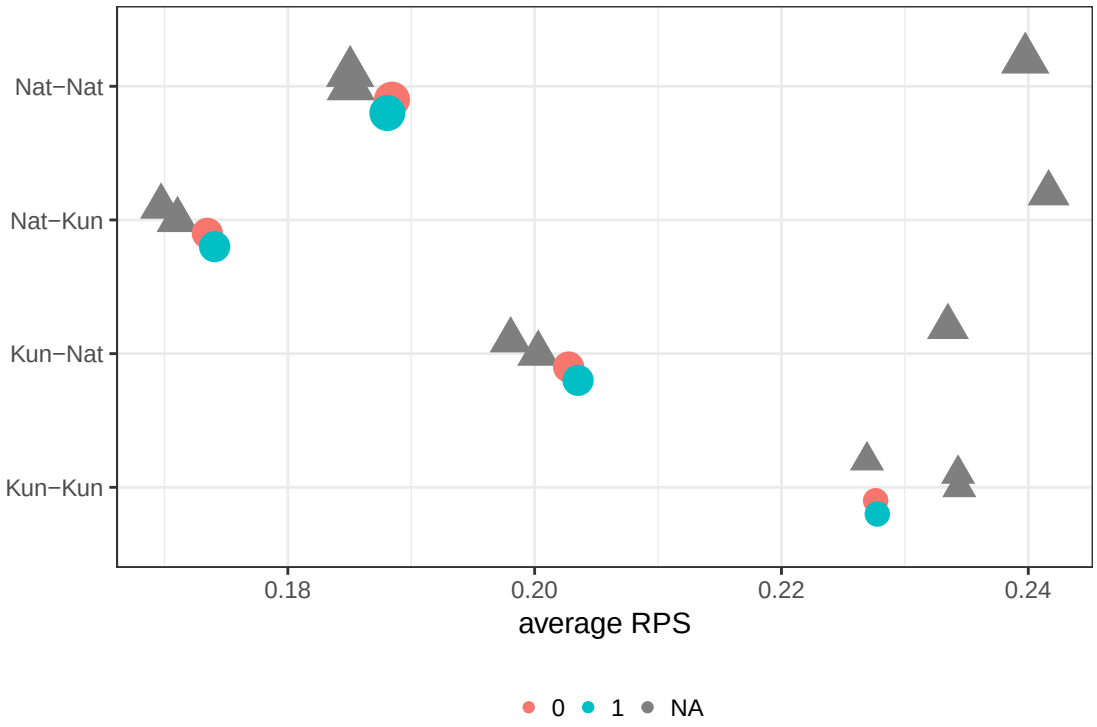


Figure 3: T-distribution model with and without Artificial turf advantage by playing surface.

We see that there is no stratum for which the artificial turf model clearly outperforms the base T-distribution model.

Skellam models stratified by relative team strength

We also compare several Skellam models. Here we focus on the choice between one or two team ability parameters, as well as the zero inflation component.

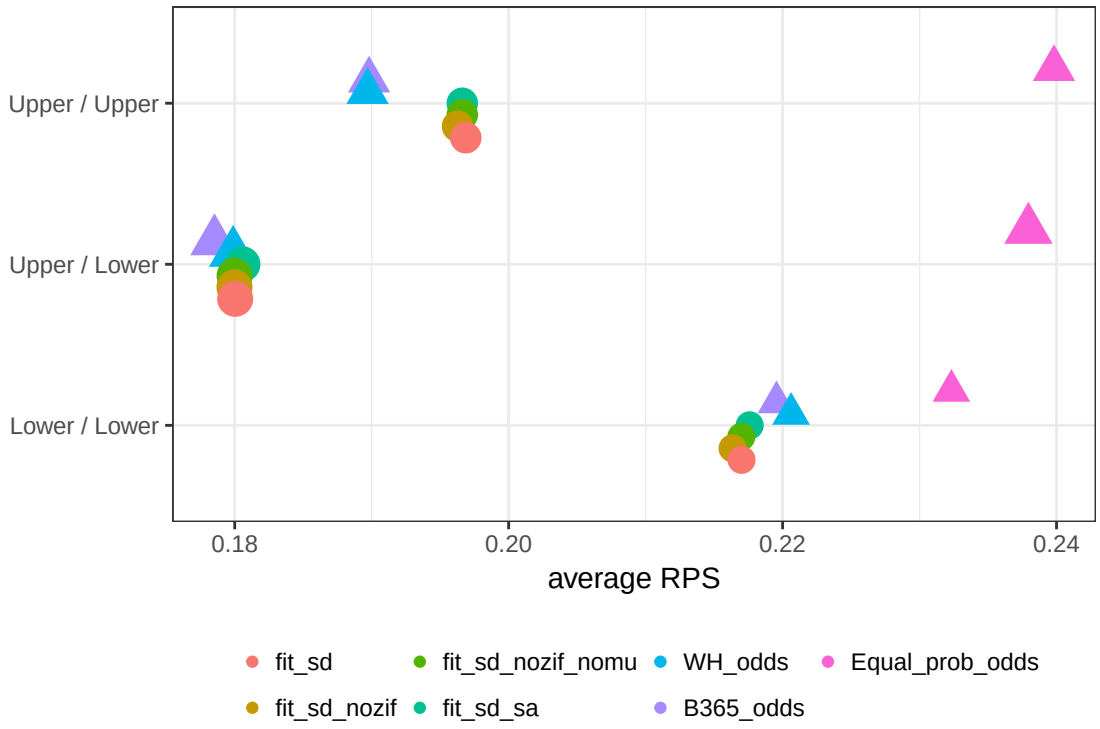


Figure 4: Skellam models compared by Relative team strength match type.

Interestingly, when two strong teams play each other, the odd-based predictions clearly outperforms the Skellam models. For matches between a lower half ranking team and a upper half ranking team, the models perform similar to the betting odds derived predictions. And finally, when two lower half ranking teams play each other, the models outperform the betting odds.

The differences between the three Skellam variants are very small. The model with a single team strength parameter appears slightly worse. Leaving out the zero inflation slighly improves the predictions. That adding a zero inflation element does not contribute to model fit was also found by [@karlis_bayesian_2009-1] as well as [@koopman_dynamic_2014-1].

Next we compare the base Skellam model with an expanded model that includes the artificial turf advantage (ATA) predictor. Figure @ref(fig:figSkellamATA) shows both models.

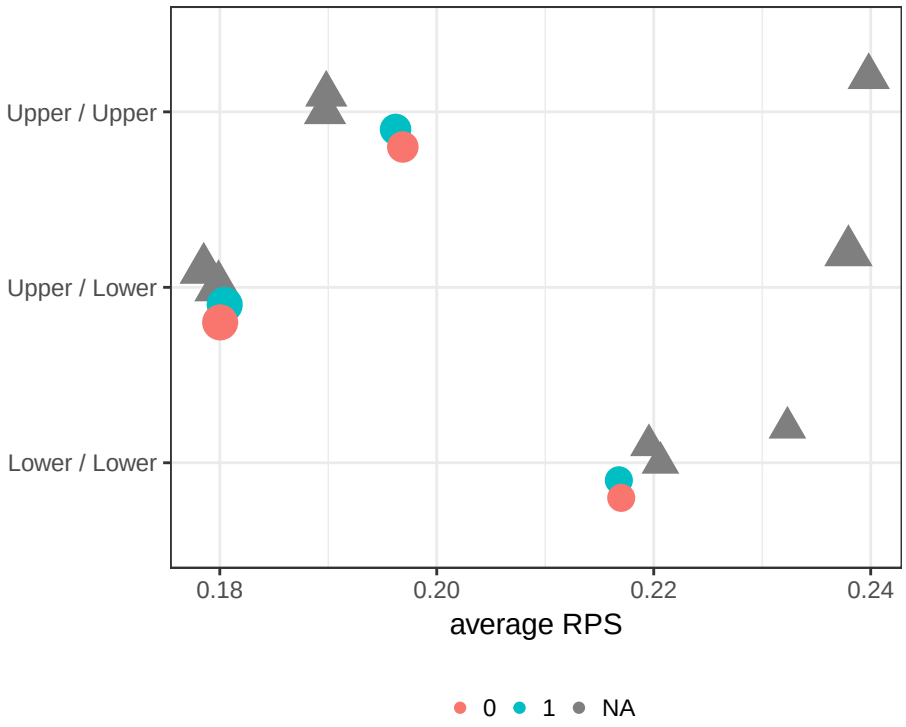


Figure 5: Skellam model with and without Artficial turf advantage.

For all three categories, the average RPS is very similar comparing with and without the artificial turf advantage parameter. Including the AT effect slightly improves predictions between two upper half ranking teams (as we see later because match outcome prediction for Heracles improve), but at the cost of slightly worse predictions for matches between an upper half and a lower half ranking team.

Skellam models by Team playing surface

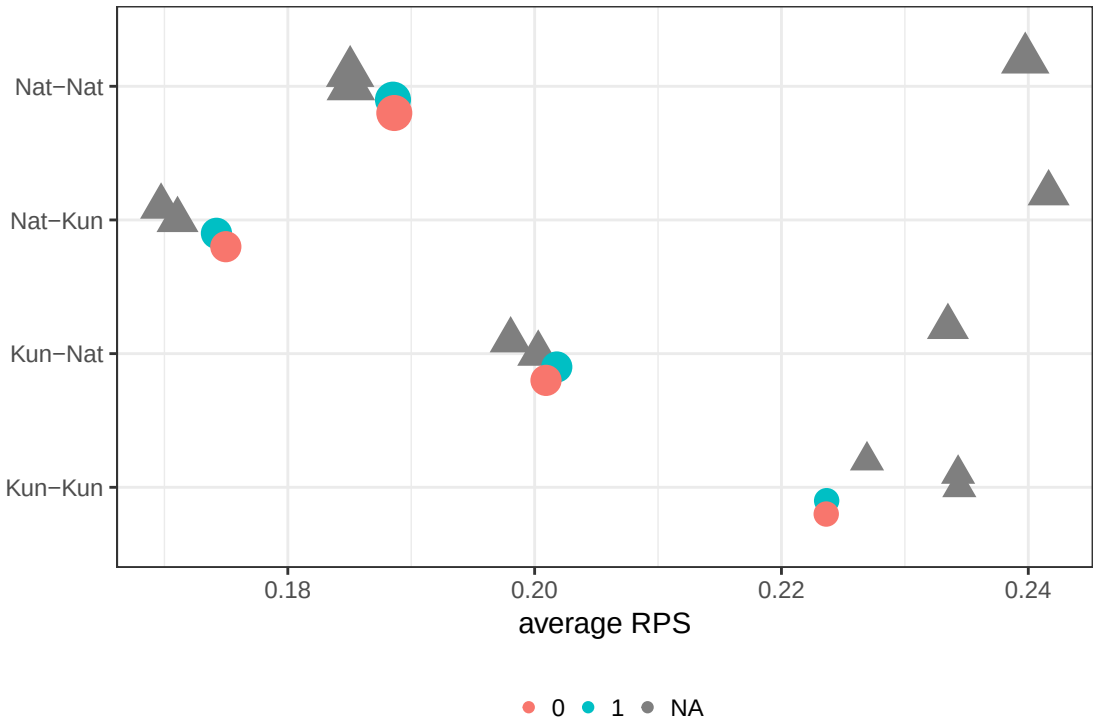


Figure 6: Skellam model with and without Artificial turf advantage by playing surface.

We see that there is no stratum for which the artificial turf model clearly outperforms the base Skellam model.

Stratifying by Match surface and relative team strength

```
res_upper_lower_at <- NL_ALL_PRED[, .(arps = mean(rps_vec, na.rm = T),
                      N = length(na.omit(rps_vec))), .(upper_lower_type, art_turf, model_nr)]

res_upper_lower_at[, upper_lower_art_turf_type := paste(upper_lower_type, "at=", art_turf, sep = '')]
setkey(res_upper_lower_at, model_nr)
setkey(result_by_model, model_nr)
res_upper_lower_at <- result_by_model[res_upper_lower_at]
saveRDS(res_upper_lower_at, "output/20180828 res_upper_lower_at.RDS")
```

In Figure x, we stratify the matches by playing surface type and relative team strength. There are clear differences in predictability in this section, with matches played on natural grass between a relatively strong and a relatively weak team having the highest predictability.

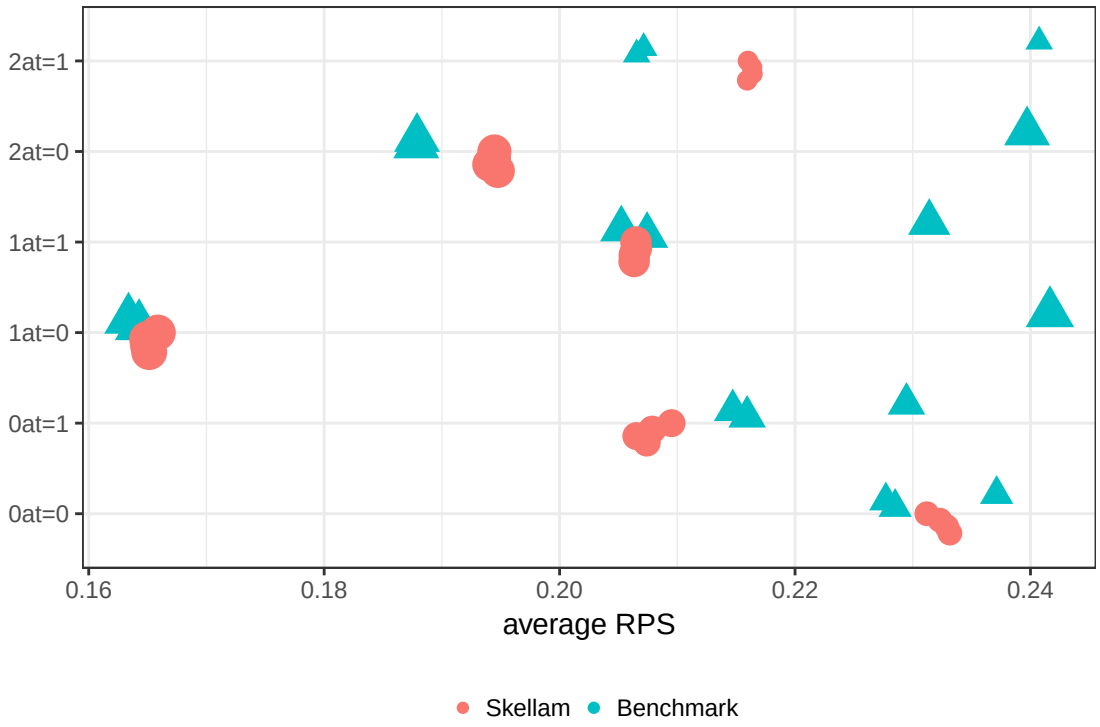


Figure 7: Skellam models compared by playing surface and relative team strength.

Here we find that there are no large differences between the various "flavors" for the Skellam. Now compare base model with and without AT:

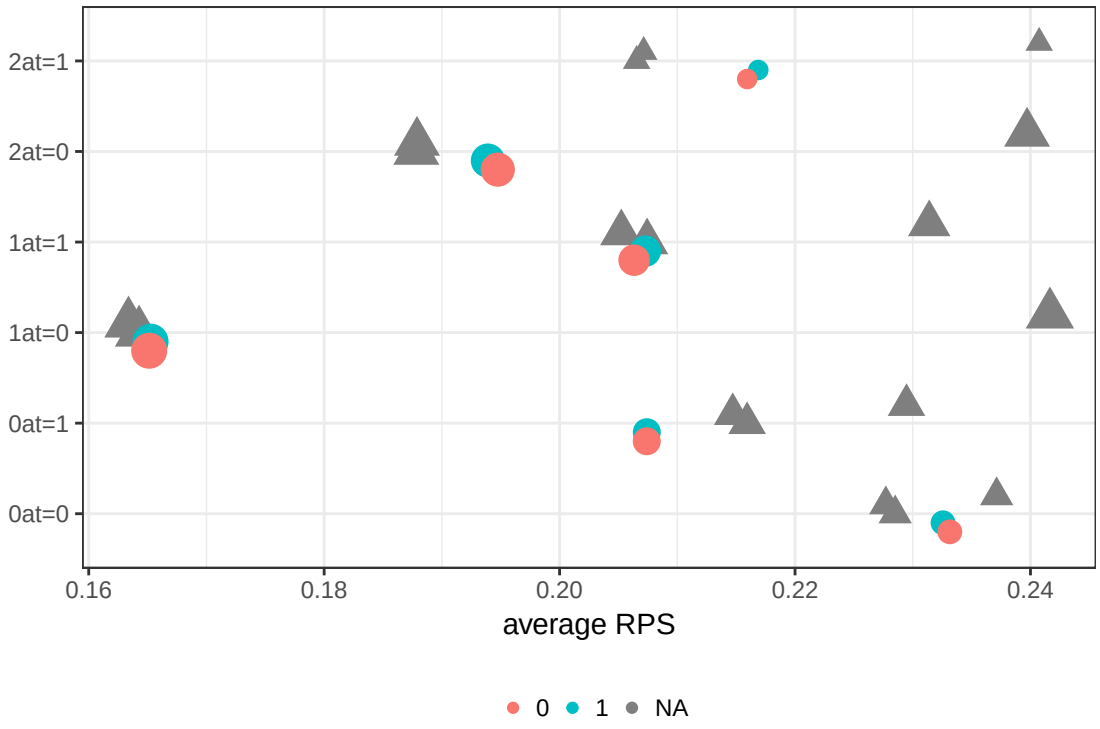


Figure 8: Skellam models compared by playing surface and relative team strength.

Including the AT parameter does not do much, but since the coefficient is small this is no surprise.

Comparing Skellam and t-distribution models by match surface x relative team strength

Compare the best Skellam and T-dist models with this slicing:

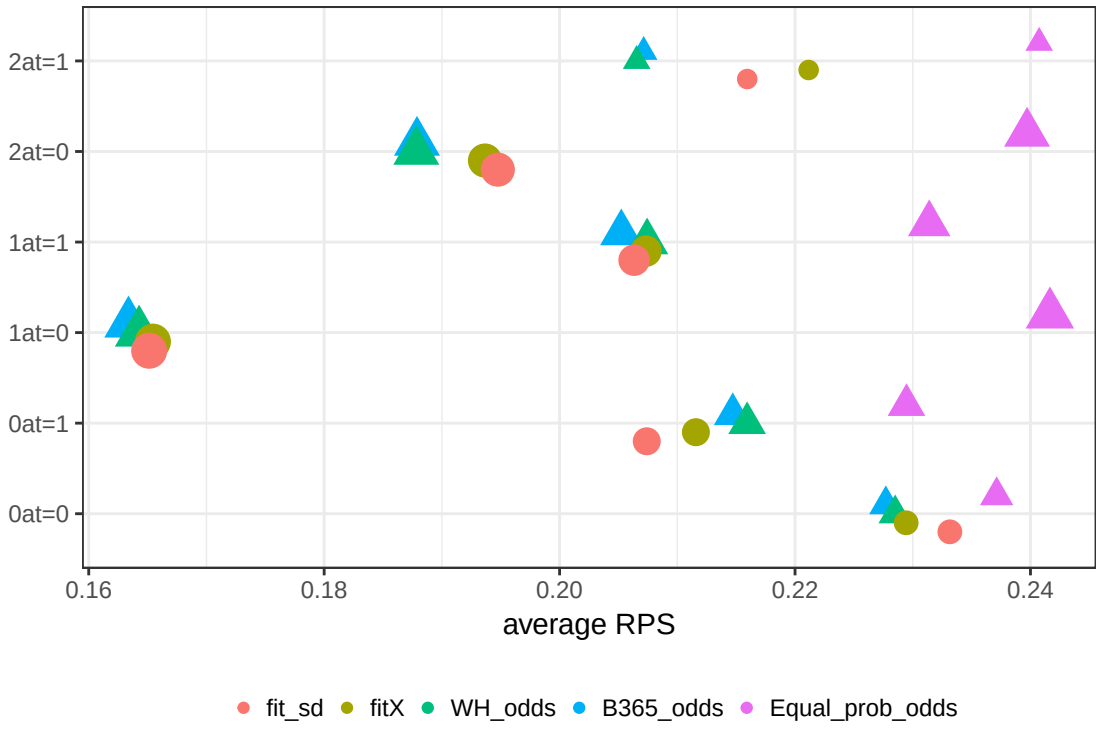


Figure 9: Skellam models compared by match surface type and relative team strength.

We see that for some subgroups, the Skellam model outperforms the t-distribution model and vice versa.

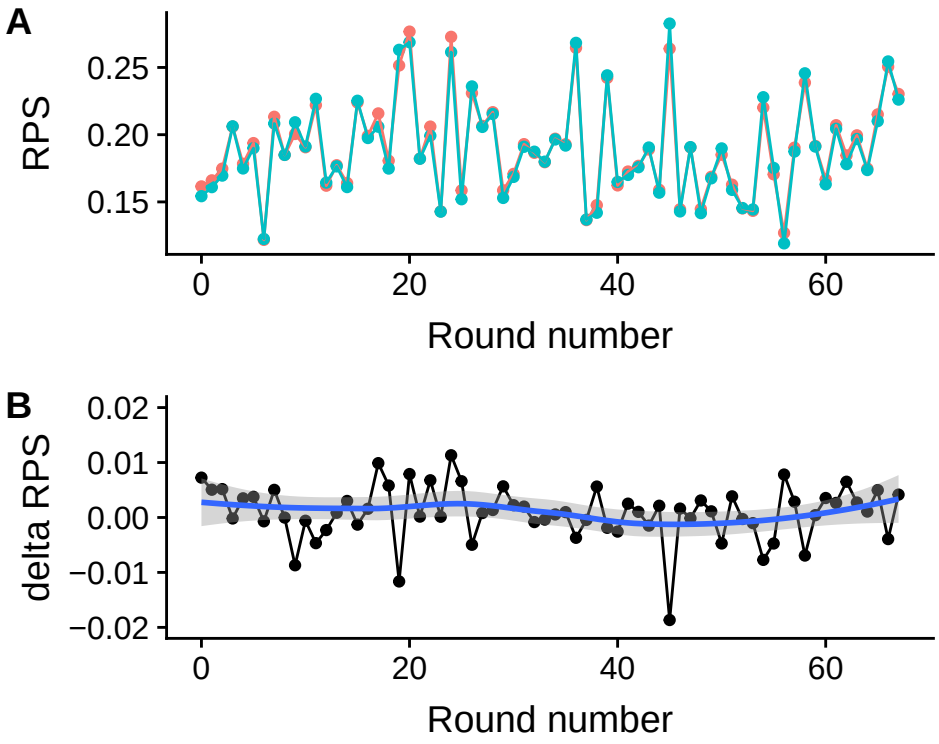
Stratifying by round (time)

Given the suggestion that the artificial turf effect is strongest during the second season of forecasts, we check how the predictions of the various models change over time.

```
res_time <- NL_ALL_PRED[, .(arps = mean(rps_vec, na.rm = T),
                        N = length(na.omit(rps_vec))), .(model_nr, round)]
setkey(res_time, model_nr)
setkey(result_by_model, model_nr)
res_time <- result_by_model[res_time]
saveRDS(res_time, "output/20180828 res_time.rds")
```

We first compare the two bookmakers, to see how they line up.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



There are small differences in the forecast accuracy between the two bookmakers, but no particular trend over time.

Next, we are curious if the artificial turf advantage model starts outperforming the base model in the second season, when the in-sample coefficient value for the artificial turf advantage starts to increase.

We first compare the t-distribution model with and without AT effect:

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
## Warning: Removed 1 rows containing missing values (geom_point).
```

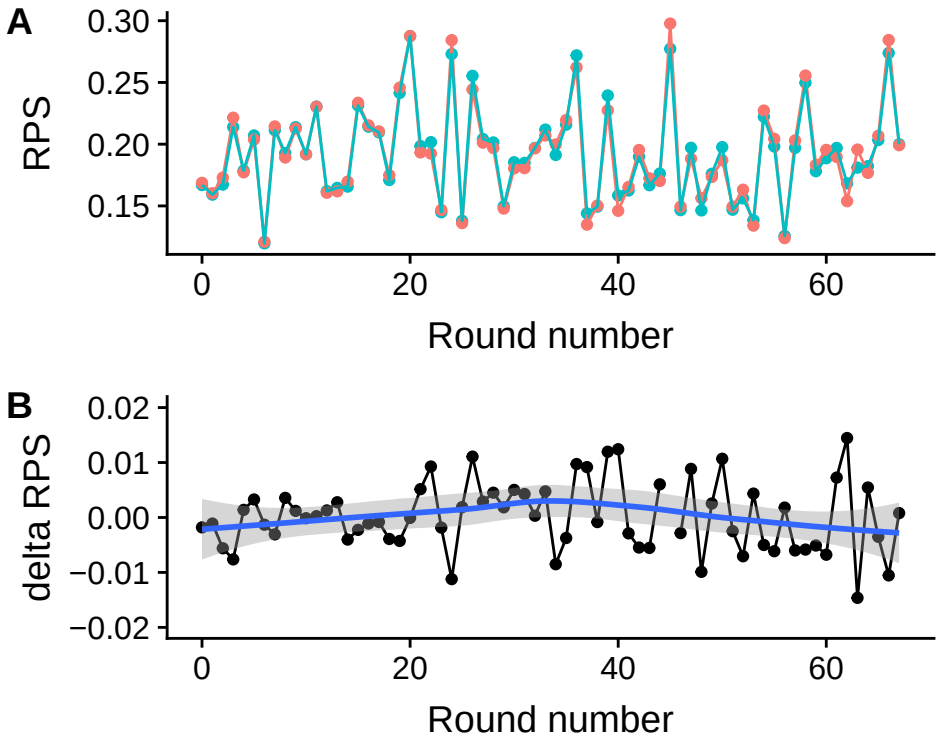


Figure 10: T-distribution model with and with artificial turf advantage.

This turns out not to be the case. There appears no pattern over time in differences between the parametric models.

The same check for the Skellam model:

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

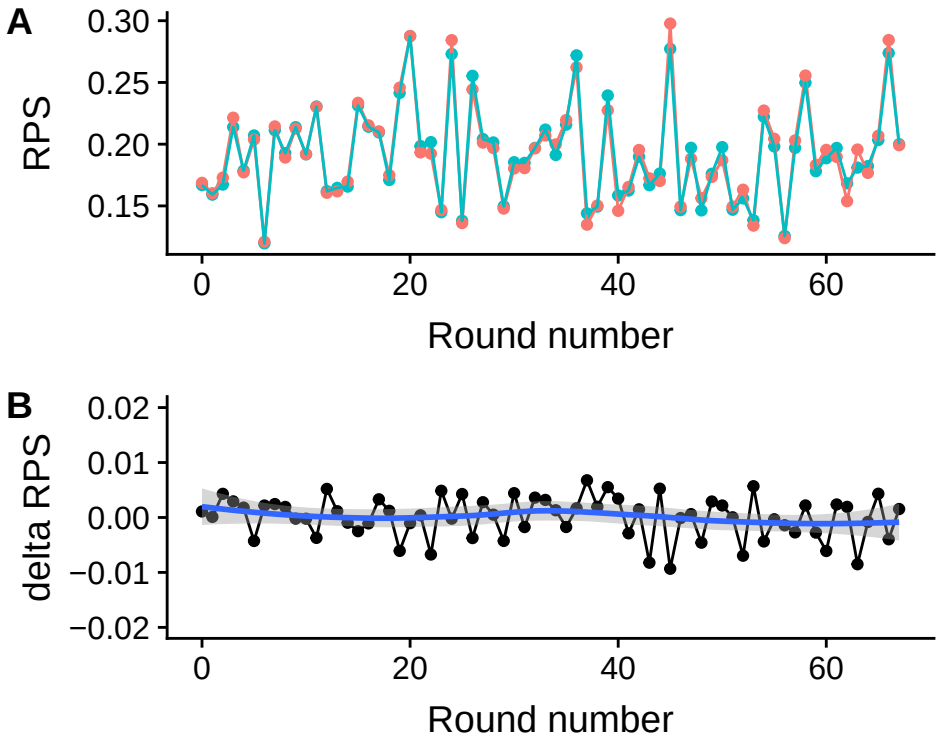


Figure 11: Skellam model with and with artificial turf advantage.

For the Skellam model with and without AT effect, the difference in weekly averaged RPS over time are very small, consistent with the small estimated value of the parameter.

Finally, we compare the best bookmaker odds (Bet365) to our best parametric model, the Skellam model without zero inflation. This might suggests directions for improvement.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

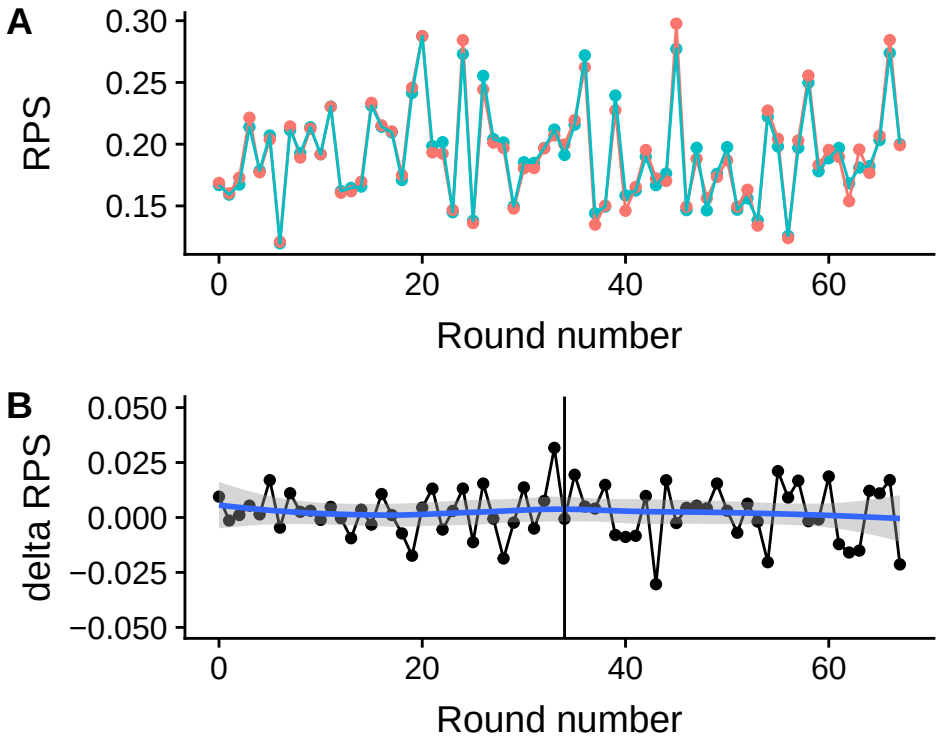


Figure 12: Bet365 vs best Skellam model forecasts compared over time.

In particular the last round of the first season shows the models performing worse. However, this is reversed for the last round of the second season.

Appendix: Ranked Probability Score

To compare model predictive performance we use the Rank Probability Score (RPS), which is the mean-squared error for a multi-category forecast. Here we consider three outcomes, win, draw of loss.

We first test if we have implemented the algorithm properly. For this, we reproduce the predictions and outcomes from the Constantinou & Fenton 2012 paper, and compare the RPS values for these examples.

```
# Unit test: calculate Constantinou & Fenton 2012 examples
predictions <- rbind(c(1, 0, 0),
                     c(0.9, 0.1, 0),
                     c(0.8, 0.1, 0.1),
                     c(0.5, 0.25, 0.25),

                     c(0.35 ,0.3 ,0.35), # this one is better
                     c( 0.6, 0.3, 0.1), # or not?

                     c(0.6 ,0.25 ,0.15),
                     c( 0.6, 0.15, 0.25),
                     c( 0.57,0.33,0.1),
                     c(0.6 ,0.2 ,0.2),
                     c(0.4, 0.2, 0.4), # not in C&F 2012 paper
                     c(1/3, 1/3, 1/3), # not in C&F 2012 paper
```

```
                    c(1/3, 1/3, 1/3) # not in C&F 2012 paper

)

observed <- rbind(c(1, 0, 0),
                  c(1, 0, 0),
                  c(1, 0, 0),
                  c(1, 0, 0),

                  c(0,1,0),
                  c(0,1,0),

                  c(1,0,0),
                  c(1,0,0),
                  c(1,0,0),
                  c(1,0,0) ,
                  c(1, 0, 0),
                  c(0, 1, 0),
                  c(1, 0, 0)
)


vec <- calculate_rps(predictions, observed)

# According their paper, the calculations should give:
Constantinou_Fenton <- c(0, 0.005, 0.025, 0.1562, 0.1225, 0.1850, 0.09125, 0.11125, 0.09745, 0.1)

signif(vec[1:10],5) == Constantinou_Fenton
```

```
##  [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
Constantinou_Fenton[4]
```

```
## [1] 0.1562
```

```
vec[4]
```

```
## [1] 0.15625
```

Apart from what appears a rounding error in C&F, we conclude we have implemented the Rank probability score properly.

To get some intuition on what "good" RPS values are for soccer matches, we try a few simple prediction strategies.

- All wins
- all draws
- equal probs
- long run average probability of a win, draw of loss

This last strategy automatically incorporates the home advantage, but does not take into account team strength.

```
model_data <- Create_model_data_for_TS2(NL_ALL[season %in% c(17)],
                                        NL_ALL[season == 16],
                                        NL_ALL[season %in% c(17)][1:2,])

actual_scorez <- Convert_actual_to_win_draw_loss_vector(model_data$goal_difference)
```

```r
# always draw
pred_probz <- data.table(game_id = 1:model_data$n_games,
                         p_win = 0,
                         p_draw = 1,
                         p_loss = 0)
calculate_arps(pred_probz[,.( p_win, p_draw, p_loss)], actual_scorez[,.(act_win, act_draw, act_loss)])
```

```
## [1] 0.3807
```

```r
# always win
pred_probz <- data.table(game_id = 1:model_data$n_games,
                         p_win = 1,
                         p_draw = 0,
                         p_loss = 0)
calculate_arps(pred_probz[,.( p_win, p_draw, p_loss)], actual_scorez[,.(act_win, act_draw, act_loss)])
```

```
## [1] 0.4232
```

```r
# equal probs
pred_probz <- data.table(game_id = 1:model_data$n_games,
                         p_win = 1/3,
                         p_draw = 1/3,
                         p_loss = 1/3)
calculate_arps(pred_probz[,.( p_win, p_draw, p_loss)], actual_scorez[,.(act_win, act_draw, act_loss)])
```

```
## [1] 0.238
```

```r
# average over all matches (this takes into account the home advantage)
pred_probz <- data.table(game_id = 1:model_data$n_games,
                         p_win = mean(actual_scorez$act_win),
                         p_draw = mean(actual_scorez$act_draw),
                         p_loss = mean(actual_scorez$act_loss))
calculate_arps(pred_probz[,.( p_win, p_draw, p_loss)], actual_scorez[,.(act_win, act_draw, act_loss)])
```

```
## [1] 0.2299
```

So our models must score at least an average RPS of 0.23 to be any good.