Fit models, extract results

GS Verhoeven

Summary

This Notebook fits the models, and extracts the results.

Load packages

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

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

```
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")

toggle_static <- 0
```

Read & prep Eredivisie dataset

`NL_ALL` contains all data for 2000-2018.

```
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)]

saveRDS(NL_ALL, "output/NL_ALL.rds")
```

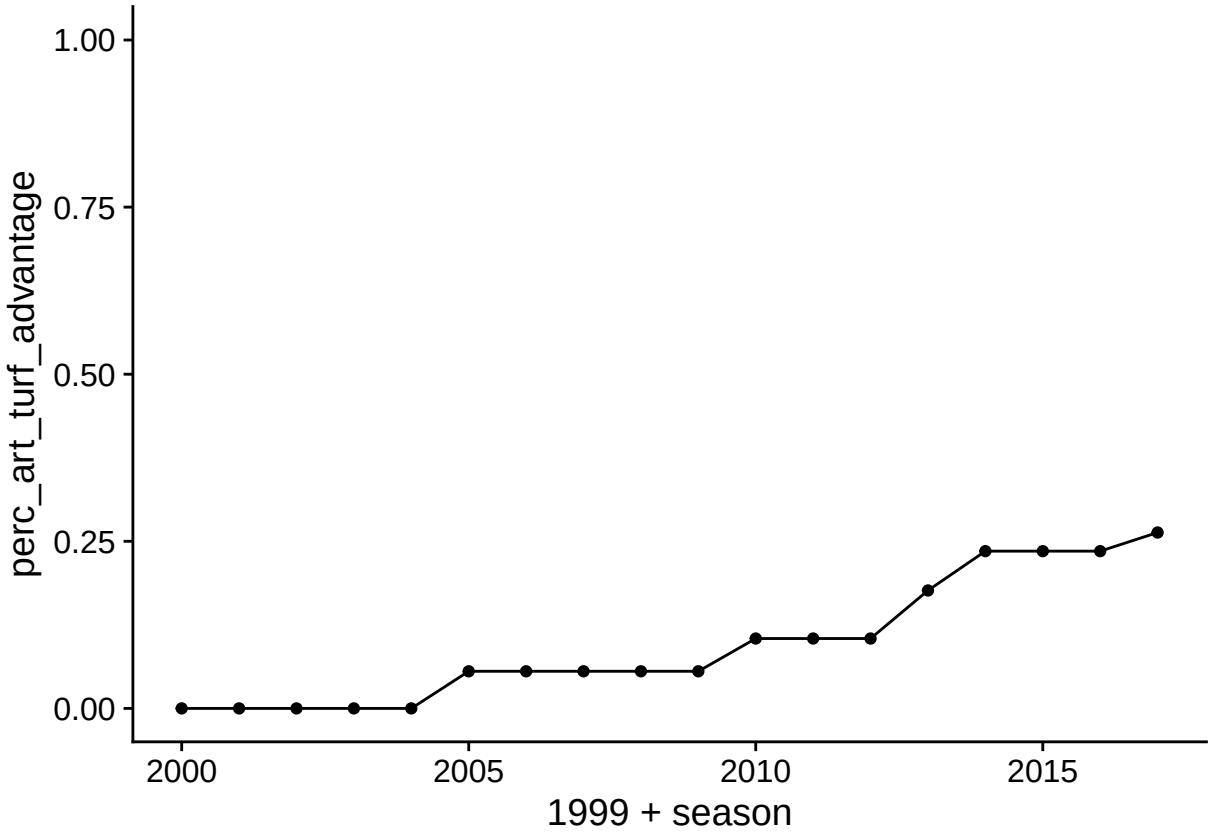
NL_ALL does not contain the betting odds of Bet365. We need these for the out-of-sample seasons 2015/2016 and 2016/2017 to compare our forecasts.

```
# read in raw data with betting odds
NL15 <- data.table(read.table(unz("data\\data15.zip", "N1.csv"), header=T, quote="\"", sep=","))
NL16 <- data.table(read.table(unz("data\\data16.zip", "N1.csv"), header=T, quote="\"", sep=","))

NLodds <- rbind(NL15, NL16)

NLodds <- NLodds[, Date := as.Date(Date, "%d/%m/%y")]
```

Plot percentages of artificial turf matches vs season

Since we're interested in the effect of artificial turf, lets plot the number of matches where the artificial turf home advantage could be "at work".

```
res <- NL_ALL[, .(perc_art_turf_advantage = mean(art_turf_advantage)), .(season)]
ggplot(res, aes(x = 1999+season, y = perc_art_turf_advantage)) +
  geom_point() + ylim(0, 1) + geom_line()
```



Forecasting approach: train / test choices

We pick two seasons as training set up to the first out-of-sample predictions, we then use an expanding window into the next season.

For the first two out-of-sample weeks (home and away) we miss a prediction for the new team, which is not present in the in-sample dataset). This gives 304 predictions (17 * 9 -2).

Need to think carefully about new teams in t+1. They don't have parameters yet. Problem: team_ids in sample fitting don't match the prediction. We solve this by generating matching id's during creation of model_Data.

Out of sample predictions: results for seasons 2015/2016 and 2016/2017

This is part where all the MCMC work gets done. Fit all models in `model_list` with fitting parameter settings fit_pars. For each model we specify which parameters should be saved, this is to limit the amount of disk space used by the model fits.

`fit_models` sets whether fitting should run or whether only saved fits are read and processed. `fullrun` sets whether saved fits should be processed or the saved result table should be read instead.

Need to include the main predictors as well (b_home, artificial_turf_effect, etc).

```
source("code/Create_model_data_for_TS2.R")
source("code/fitOneStepAhead.R")
source("code/ReadfitsandCalculateRPS.R")

model_list <- data.table(read.csv2("models/model_list.csv"))

fit_pars <- list(nsamples = 1000, # samples PER chain, including warmup
                 chains = 6,
                 warmup = 500,
                 nshifts_start = 0,
                 nshifts_end = 67,
                 init_r_val = 0.1,
                 start_round = 443,
                 end_round = 511,
                 prev_perf_season = 13,
                 target_folder = "c:/testversleutel/FITS/"
                 )
#saveRDS(fit_pars, "output/20180101 fitpars.rds")

fit_models <- 0
fullrun <- 0

if(fullrun){
  if(fit_models == 1) model_list <- model_list[active == 1,]
  # else extract predictions only
  osa_res <- list()
  for(i in 1:nrow(model_list)){

    split_string <- str_split(as.character(model_list[i, ]$include_pars), ", ")

    osa_res[[i]] <- fitOneStepAhead(toggle_dynamic = fit_models,
                                sm_string = paste("models/", model_list$stan_file[i], sep = ''),
                                model_short_name = model_list$short_name[i],
                                model_long_name = model_list$name[i],
                                stanfit_include_pars = unlist(split_string),
                                fitpars = fit_pars)
  }
  saveRDS(osa_res, "output/20180405 osa_res.rds")
} else {osa_res <- readRDS("output/20180405 osa_res.rds")}

# re-read model_list to include non-active models as well
model_list <- data.table(read.csv2("models/model_list.csv"))
```

The no pooling model needs an increased adapt_delta to silence all warnings during sampling. All models are now "warning" free from Stan.

extract coefficients from model fits

We're interested in all non-team parameters. So constant_mu, home_advantage, artificial_turf_advantage etc. Want this for all 68 weeks available for each model.

This allows us to see change over the two year period that we ADD to the years we start with (2013/2014 and 2014/2015). So the final model (last week of 2016/2017 season) is based on four seasons of match data.

```
read_fit_coefs <- 0

model_list_tmp <- model_list

coef_res <- c()

if(read_fit_coefs){
  for(i in 1:nrow(model_list_tmp)){

    split_string <- str_split(as.character(model_list_tmp[i, ]$extract_pars), ", ")

    coef_res[[i]] <- ReadFitsWrapper(sm_string = paste("models/", model_list_tmp$stan_file[i], sep = ''
                                     model_short_name = model_list_tmp$short_name[i],
                                     model_long_name = model_list_tmp$name[i],
                                     stanfit_extract_pars = unlist(split_string),
                                     fitpars = fit_pars)
  }
  #saveRDS(coef_res, "output/20180202 coef_res.rds")
  tidy_coef_res <- tidyCoefs(coef_res, model_list_tmp)
  # use this table in the manuscript
  saveRDS(tidy_coef_res, "output/tidy_coef_res.rds")

} else {tidy_coef_res <- readRDS("output/tidy_coef_res.rds")}
```

Extract Rhat and n_eff from fits

```
#source("code/ReadfitsandExtractCoefs.R")
read_rhat_neff <- 0

model_list_tmp <- model_list

rhat_neff_res <- c()

if(read_rhat_neff){
  for(i in 1:nrow(model_list_tmp)){

    split_string <- str_split(as.character(model_list_tmp[i, ]$extract_pars), ", ")

    rhat_neff_res[[i]] <- ReadRhatNeffWrapper(sm_string = paste("models/", model_list_tmp$stan_file[i],
                                     model_short_name = model_list_tmp$short_name[i],
                                     model_long_name = model_list_tmp$name[i],
                                     stanfit_extract_pars = unlist(split_string),
                                     fitpars = fit_pars)
  }
  #saveRDS(rhat_neff_res, "output/20180323 rhat_neff_res.rds")
  tidy_rhat_neff_res <- tidyCoefs(rhat_neff_res, model_list_tmp)
  # use this table in the manuscript
  saveRDS(tidy_rhat_neff_res, "output/tidy_rhat_neff_res.rds")

} else {tidy_rhat_neff_res <- readRDS("output/tidy_rhat_neff_res.rds")}
```

Create dataset with all matches, and all predictions from each model

We use this dataset to identify strengths and weaknesses from the models.

We also add the bookmakers probabilities to calculate the average RPS.

Lets pick B365 and WHH. Bet365 because it appears to be a popular choice. William Hill because it is also present for matches in 2000. We use basic normalization to convert betting odds to probabilities.

```
source("code/CombineDataWithPredictions.R")
fullrun <- 0

if(fullrun){
  for(i in 1:nrow(model_list)){
    if(i == 1){NL_ALL_PRED <- CombineDataWithPredictions(NL_ALL,
                                                         fit_pars,
                                                         osa_res,
                                                         model_nr = i)
    } else{NL_ALL_PRED <- rbind(NL_ALL_PRED,CombineDataWithPredictions(NL_ALL,
                                                         fit_pars,
                                                         osa_res,
                                                         model_nr = i))
    }
  }

  # abuse WH odds as sort key (also present in NL_ALL_PRED)
  setkey(NLodds, Date, WHH, WHD, WHA)

  # convert odds to probabilistic forecasts using basic normalization method
  bet365_probs <- odds_to_probability(NLodds[, .(B365H, B365D, B365A)])
  WH_probs <- odds_to_probability(NLodds[, .(WHH, WHD, WHA)])

  # Add equal probabilities (aka Equal_prob on typewriter) model
  Equal_prob_probz <- data.table(game_id = 1:nrow(NLodds),
                            prob_win = 1/3,
                            prob_draw = 1/3,
                            prob_loss = 1/3)

  # function to add the odds as model to NL_ALL_PRED table
  addOddsAsModel <- function(prediction_table, converted_odds){
    # we need a template
    prediction_table_new <- prediction_table[model_nr == 1,]
    # sort matches as in converted_odds
    setkey(prediction_table_new, Date,  WHH, WHD, WHA)

    new_model_nr <- max(prediction_table$model_nr) + 1
    prediction_table_new <- prediction_table_new[, model_nr := new_model_nr]
    dropcols <-  c("rps_vec", "p_win", "p_draw", "p_loss")
    prediction_table_new <- prediction_table_new[, !(colnames(prediction_table_new) %in% dropcols), wit

    prediction_table_new <- cbind(converted_odds[,.( prob_win, prob_draw, prob_loss)], prediction_table_

    actual_scorez <- Convert_actual_to_win_draw_loss_vector(NLodds$FTHG - NLodds$FTAG)
```

```
    # calculate RPS per game
    rps_vec <- calculate_rps(converted_odds[,.( prob_win, prob_draw, prob_loss)],
                             actual_scorez[,.(act_win, act_draw, act_loss)])

    prediction_table_new <- data.table(rps_vec, prediction_table_new)
    prediction_table_new <- prediction_table_new[is.na(act_win), rps_vec := NA]
    setnames(prediction_table_new, "prob_win", "p_win")
    setnames(prediction_table_new, "prob_draw", "p_draw")
    setnames(prediction_table_new, "prob_loss", "p_loss")
    prediction_table <- rbind(prediction_table_new, prediction_table)
    prediction_table
  }

  # add WH and Bet365 as models to NL_ALL_PRED
  NL_ALL_PRED <- addOddsAsModel(NL_ALL_PRED, WH_probs)
  NL_ALL_PRED <- addOddsAsModel(NL_ALL_PRED, bet365_probs)
  NL_ALL_PRED <- addOddsAsModel(NL_ALL_PRED, Equal_prob_probz)

  # remove the four matches without prediction also from betting odds "models"
  matches_wo_preds <- NL_ALL_PRED[is.na(rps_vec), .N, .(matchKey)]$matchKey

  NL_ALL_PRED <- NL_ALL_PRED[, has_pred := 1]
  NL_ALL_PRED <- NL_ALL_PRED[matchKey %in% matches_wo_preds, has_pred := 0]
saveRDS(NL_ALL_PRED, file = "Output/20180406 NL_ALL_PRED_w_odds.rds")
} else {NL_ALL_PRED <- readRDS("Output/20180406 NL_ALL_PRED_w_odds.rds")}
```

At this point, NL_ALL_PRED has rps scores for all 608 predicted matches for all models, including the betting odds.

Create main result by model table

```
result <- NL_ALL_PRED[!(model_nr %in% c(11:13)), .(aRPS = mean(rps_vec, na.rm = T),
                       N_pred = nrow(.SD[!is.na(rps_vec)])), .(model_nr)][order(aRPS)]

setkey(result, model_nr)
setkey(model_list, model_nr)
result <- model_list[result]
```

Add betting odds result to main result table

Use this table to present results in the manuscript.

```
actual_scorez <- Convert_actual_to_win_draw_loss_vector(NLodds$FTHG - NLodds$FTAG)

result_WH <- data.table(model_nr = 11, name = "William_hill odds", stan_file = NA,
                             short_name = "WH_odds", include_pars = NA, dist = "Benchmark", ata = NA,
                             ha = NA, part_pool = NA, n_ability = NA, zif = NA, extract_pars = NA,
                             active = 0, aRPS = NA, N_pred = NA)

result_WH$aRPS <- mean(NL_ALL_PRED[model_nr == 11 & !is.na(rps_vec),]$rps_vec)
```

```
result_WH$N_pred <- nrow(NL_ALL_PRED[model_nr == 11 & !is.na(rps_vec),])

result_Bet365 <- data.table(model_nr = 12, name = "Bet365 odds", stan_file = NA,
                            short_name = "B365_odds", include_pars = NA, dist = "Benchmark", ata = NA,
                            ha = NA, part_pool = NA, n_ability = NA, zif = NA, extract_pars = NA,
                            active = 0, aRPS = NA, N_pred = NA)

result_Bet365$aRPS <- mean(NL_ALL_PRED[model_nr == 12 & !is.na(rps_vec),]$rps_vec)
result_Bet365$N_pred <- nrow(NL_ALL_PRED[model_nr == 12 & !is.na(rps_vec),])

result_Equal_prob <- data.table(model_nr = 13, name = "Equal probability odds", stan_file = NA,
                            short_name = "Equal_prob_odds", include_pars = NA, dist = "Benchmark", ata
                            ha = NA, part_pool = NA, n_ability = NA, zif = NA, extract_pars = NA,
                            active = 0, aRPS = NA, N_pred = NA)

result_Equal_prob$aRPS <- mean(NL_ALL_PRED[model_nr == 13 & !is.na(rps_vec),]$rps_vec)
result_Equal_prob$N_pred <- nrow(NL_ALL_PRED[model_nr == 13 & !is.na(rps_vec),])


# unit test for calculate_arps(): expect rps of zero when using actual scores as predictions for acutal
#calculate_arps(actual_scorez[,.(act_win, act_draw, act_loss)], actual_scorez[,.(act_win, act_draw, act

result <- rbind(result, result_WH)
result <- rbind(result, result_Bet365)
result <- rbind(result, result_Equal_prob)
```

Add Diebold-Mariano test statistic

We add the DM statistic using various reference models.

```
source("code/AddDMTest.R")
# bet365, beste odds
result <- AddDMtest(result, NL_ALL_PRED, ref_model_nr = 12)
# t-dist base model
result <- AddDMtest(result, NL_ALL_PRED, ref_model_nr = 1)
# Skellam no zif is base model
result <- AddDMtest(result, NL_ALL_PRED, ref_model_nr = 7)

saveRDS(result, file = "output/20180406 average RPS by model.rds")
```

Check if we understand DM.

```
osa_res[[1]]$name
```

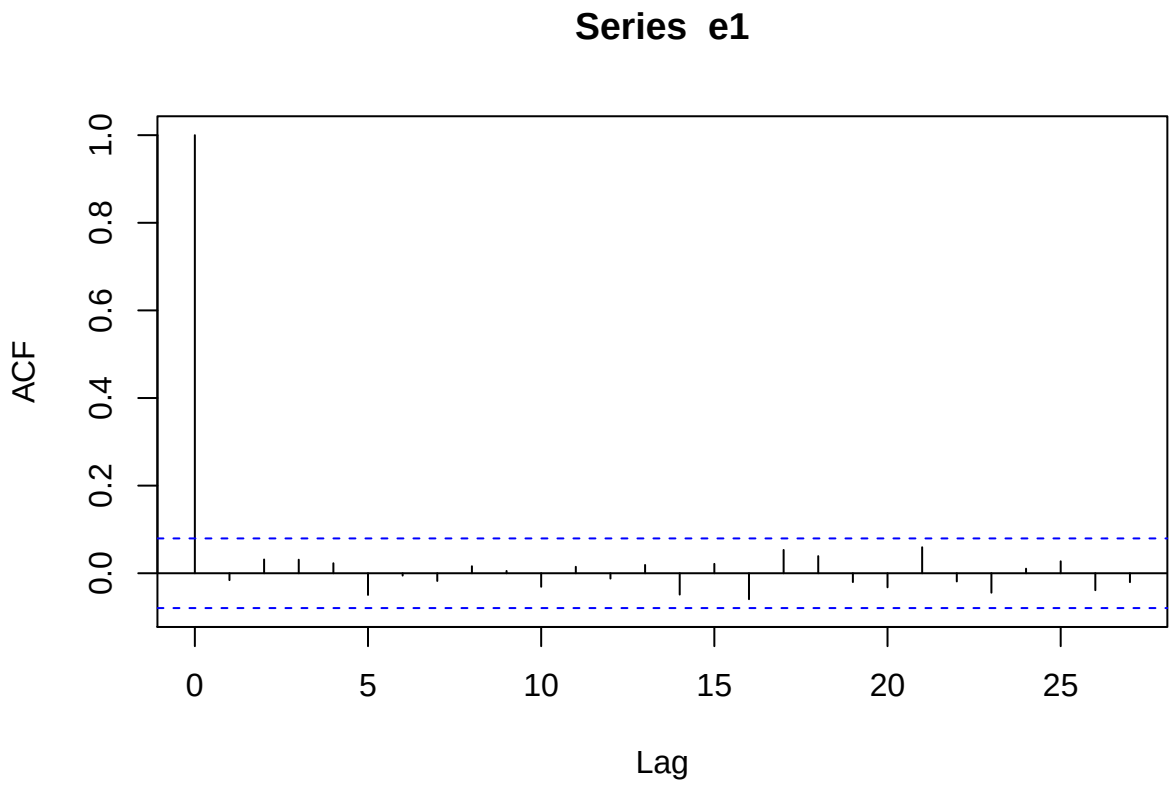
```
## [1] T-dist original
## 10 Levels: Skellam offense/defense ... T-dist with AT advantage run2
```

```
osa_res[[6]]$name
```

```
## [1] T-dist no HA
## 10 Levels: Skellam offense/defense ... T-dist with AT advantage run2
```

```
# check autocorrelation of time series (expect none since different teams)
e1 <- NL_ALL_PRED[model_nr == 6 & !is.na(rps_vec), ]$rps_vec
```

```r
w <- acf(e1)
```




```r
# Zero autocorrelation indeed

# partial pooling yes/no
delta <- osa_res[[1]]$rps_vec - osa_res[[6]]$rps_vec
summary(lm(y ~ 1, data = data.frame(y = delta)))
```

```
##
## Call:
## lm(formula = y ~ 1, data = data.frame(y = delta))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13166 -0.05587 -0.01216  0.05283  0.15557
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.006036   0.002628  -2.297    0.022 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06479 on 607 degrees of freedom
```

```r
result[model_nr == 6,]$DMstat_1
```

```
## [1] -2.297
```

```
result[model_nr == 6,]$DMpval_1
```

```
## [1] 0.022
```

```
# yes we do
```

Display main result by model table

```
# exclude reruns
knitr::kable(result[!(model_nr %in% c(8, 9)), .(name, N_pred, aRPS)][order(aRPS)])
```

name	N_pred	aRPS
Bet365 odds	608	0.1892859
William_hill odds	608	0.1901592
Skellam, no zif, offense/defense	608	0.1913812
Skellam offense/defense with AT	608	0.1916885
Skellam offense/defense	608	0.1917053
Skellam single ability	608	0.1920124
T-dist original	608	0.1920899
T-dist with AT advantage	608	0.1922611
T-dist no pooling	608	0.1957098
T-dist no HA	608	0.1981257
Equal probability odds	608	0.2374819