A long-short term event memory state-space model for multi-party elections

Marcus Groß

2019-31-07

Abstract

State-space models are a popular choice in modelling voting intentions and election results by using poll data. The presented multivariate state-space model attempts to go beyond random-walk or Kalman-filter approaches (with comparable performance to simple weighted survey averages) to the problem by introducing a long-short term event memory effect. This effect serves as reasonable explanation to the observation that the voter's share partially tends to reverse to the party's long-term trend after larger short term movements. Any event influencing the voter's share of a party is presumed to have a convex shape decomposable into a short term effect due to e.g. media spreading and a smaller long term effect remaining despite overlay effects of new events and forgetting. This effect is modelled by a mixture of a random walk and two contrasting autoregressive processes. By also taking advantage of the widely observed effect that government parties tend to fall in voter's share, whereas the opposite effect is observed for opposition parties, mid- and long-term predictions of election outcomes can be considerably be improved. The Stan-model is fitted and evaluated on poll data from seven pollsters for the German national elections ("Bundestagswahl") from 1994 to 2017, where low double digits (out-of-sample) improvements in prediction performance can be seen between 3- and 18-months prior elections. By taking into account the pollsters house effects, their poll errors and even more importantly their correlations in poll errors, an appropriate and realistic estimation error can be propagated.

Data

We have data on more than 4000 polls from 7 different pollsters between the November 1st, 1994 until the current date for the german federal election ("Bundestagswahl"). These data are scraped from the web page www.wahlrecht.de, which collects all available poll data and is updated very often. Furthermore, we have the election outcome data for all elections since 1998 and the respective partys forming the government and oppositions. Data is available for the six large parties "CDU/CSU", "SPD", "GRÜNE", "FDP", "Linke" and "AfD".

Get Poll and Election Data

First, we set a prediction date, which is the 25th of march 2017, exactly six months prior election.

```
predDate <- "2017-03-25"
```

The poll data for the german election, the "Bundestagswahl" is scraped from the wep-page wahlrecht.de.

```
require('dplyr')
require('tidyr')
require('xml2')
require('rvest')
require('XML')
require('magrittr')
require('stringr')
require('zoo')
require('rstan')
source('R/getPollData.R', encoding = 'UTF-8')
pollData <- getPollData(predDate) %>% arrange(desc(Datum))
knitr::kable(head(pollData))
```

Institut	Datum	CDU/CSU	SPD	GRÜNE	FDP	LINKE	AfD	Sonstige	Befragte
Emnid	2017-03-25	0.33	0.33	0.080	0.050	0.080	0.090	0	2450
GMS	2017-03-23	0.34	0.31	0.080	0.060	0.080	0.090	0	1008
Infratestdimap	2017-03-23	0.32	0.32	0.080	0.060	0.070	0.110	0	1023
Forsa	2017-03-22	0.34	0.31	0.070	0.060	0.070	0.090	0	2504
INSA	2017-03-20	0.31	0.32	0.065	0.065	0.085	0.115	0	1933
Emnid	2017-03-18	0.33	0.32	0.080	0.050	0.080	0.090	0	1832

```r
Elections <- read.csv2("data/Elections.csv", encoding = 'UTF-8')
Elections$Datum <- as.Date(Elections$Datum)
knitr::kable(Elections)
```

CDU.CSU	SPD	GRÜNE	FDP	LINKE	AfD	Sonstige	Year	Datum	Institut
0.351	0.409	0.067	0.062	0.051	NA	5.9	1998	1998-09-27	Election
0.385	0.385	0.086	0.074	0.040	NA	3.0	2002	2002-09-22	Election
0.352	0.342	0.081	0.098	0.087	NA	4.0	2005	2005-09-18	Election
0.338	0.230	0.107	0.146	0.119	NA	6.0	2009	2009-09-27	Election
0.415	0.257	0.084	0.048	0.086	0.047	6.4	2013	2013-09-22	Election
0.329	0.205	0.089	0.107	0.092	0.126	5.0	2017	2017-09-24	Election
NA	NA	NA	NA	NA	NA	NA	2021	2021-09-24	Election

Motivation

At the latest since the great attention, the election forecasts of the US presidential elections on fivethir-tyeight.com by Nate Silver have received, data based election forecast are of great interest in the public as well as in academia. There were already some attempts using *STAN*, which rely on state-space models, where the state (voter's intention) is modeled by a random walk. This approach follows the 2005 paper of Simon Jackman (https://www.tandfonline.com/doi/abs/10.1080/10361140500302472) and was in predicting the australian election (cf. http://freerangestats.info/blog/2017/06/24/oz-polls-statespace), for example. While this method gives valuable insights and does a good job in removing bias from different pollsters, to our experience it does not really help in doing mid- or long term forecasts compared to very simple poll-averaging methods. The question here is, if we can do better than just take the current latent state or voter's intention as our forecast for the actual election that may be six months or one year in the future.

Other election forecasts, such as the US-presidential forecasts by Nate Silver (https://fivethirtyeight.com/features/a-users-guide-to-fivethirtyeights-2016-general-election-forecast/) and multi-party forecasts for the UK-election (http://www.electionforecast.co.uk) or for the german federal electin (http://zweitstimme.org) state that there is some form of mean-reversion in polls or election results in the long term and incorporate this into their election forecast one way or another. In example http://www.electionforecast.co.uk states that:

> The basic principle is that polling has some systematic biases, in particular a tendency to overstate changes from the previous election

There is some discussion in a 1993 paper by Andrew Gelman and Gary King on this issue (https://gking.harvard.edu/files/abs/variable-abs.shtml). In a 2013 paper of Drew Linzer (https://votamatic.org/wp-content/uploads/2013/07/Linzer-JASA13.pdf) a mean reversion effect is incorporated for the US presidential elections, by implicitly assuming that on state level, the voter's share is going to return to it's long term mean. However, such a form of mean stationarity, even more in multi-party systems is unrealistic, as there are clear trends over time for different parties. Formerly successful parties can disappear entirely. Therefore, we propose some kind of mixture between a non-stationary random walk and a mean-reversing stationary process for the voter's share over time.

How can we interpret this kind mean reversion process? Our idea is to model this as a long-short term

memory process. Assuming weekly data and weekly changes, we state that each shock or change in voter's intention for a party is attributable to events, e.g. political scandals, controversial statements or candidate selection. The initial shock, which can be positive or negative, is covered by the media in the following weeks, increasing the initial effect. After some weeks the event or news might slowly disappear out of the people's minds, but not entirely. The remaining effect accounts for the long-term effect of the initial event. Is there, however, some evidence in our data of the german federal election? To investigate this empirically, we smooth the polls on a weekly basis by a smoothing spline, compute the autocorrelation functions (acf's) on the weekly differences and average the acfs on all parties exluding AfD (as this right-winged party appeared only some years ago):

```r
pollDataShock <- pollData %>% arrange(desc(Datum)) %>% na.locf(na.rm = FALSE) %>%
  na.locf(fromLast = TRUE)
dateSeq <- seq(min(pollDataShock$Datum), max(pollDataShock$Datum) , by = "week")
smoothProportions <- sapply(colnames(pollDataShock)[3:7],
                            function(y){
                              sSpline <- smooth.spline(x = pollDataShock$Datum,
                                                       y = unlist(pollDataShock[,y]))
                              predict(sSpline, x = as.numeric(dateSeq))$y
                            })

acfs <- rowMeans(apply(apply(smoothProportions, 2, diff), 2,
                       function(x) acf(x, 30, plot = FALSE)$acf))
plot(acfs, type = "h", xlab = "week after shock", ylab = "autocorrelation")
abline(h = 0)
```
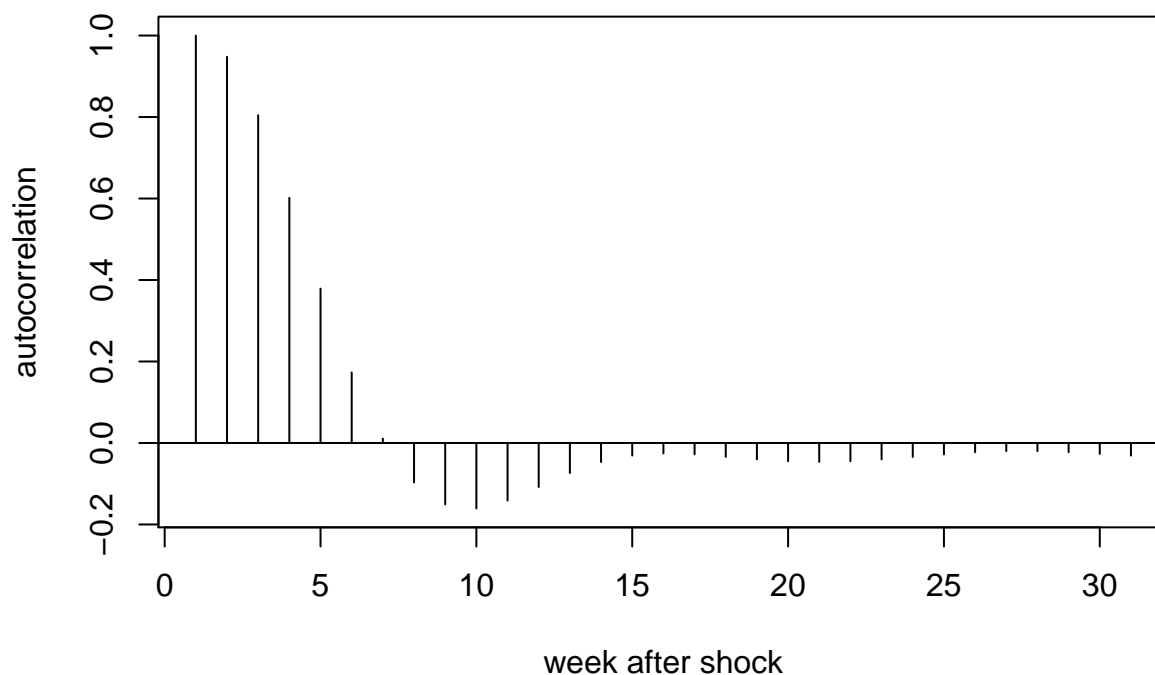

We can clearly see that there is a positive autocorrelation within the first weeks, that turns negative between 8 and 15 weeks after the initial shock. The latter interval can be seen as the period where the events gets out of the short-term memory of the people. A more clear picture of the mean-reversing nature of polls can be

seen, when we integrate the autocorrelation function:

```r
plot(cumsum(acfs) - 1, type = "l", xlab = "week after shock", ylab = "effect size")
segments(x0 = 2.9, x1 = 31, y0 = 1.6, y1 = 1.6, lty = 3)
text(x = 26, y = 1.25, "Long-term memory")
text(x = 9, y = 1.9, "Short-term memory")
```
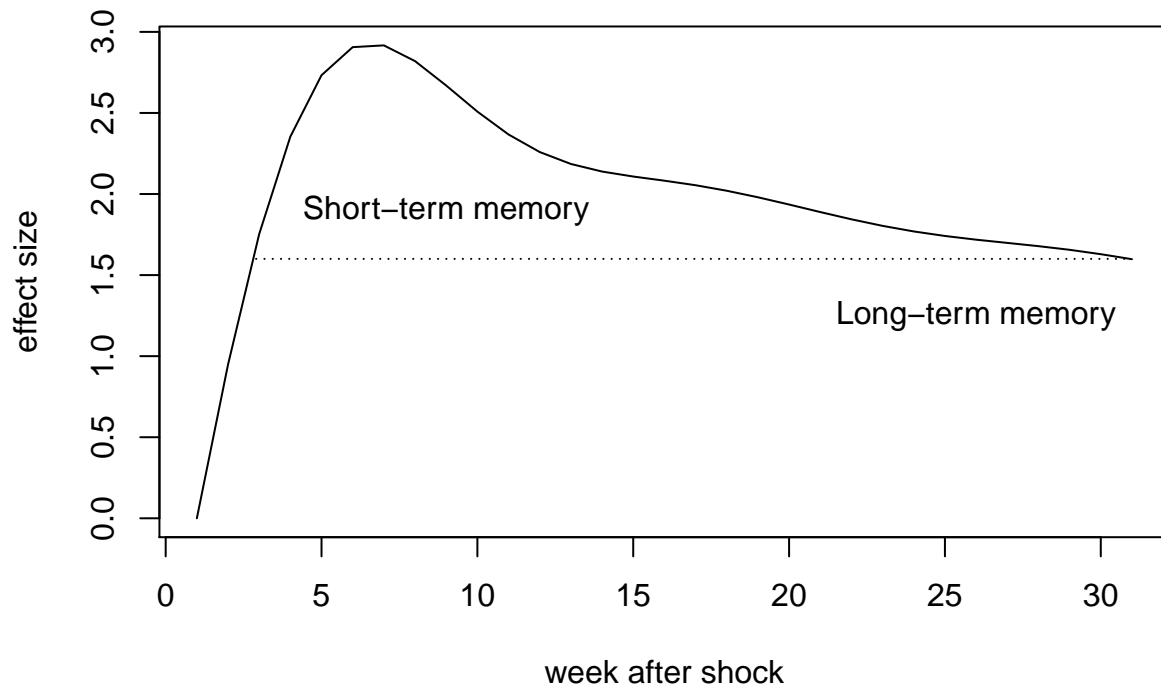
Another peculiarity in elections over time is the effect that government party lose voter's share, whereas opposition partys tend to gain share. For the german federal elections between 1998 and 2017, the government party lost share in 10 of 12 cases in the next election and in 15 of 20 cases the opposition parties were gaining share. The same observation can be made in neighbouring countries with a similar multi-party election system such as the Netherlands or Sweden. That be due to the effect that government parties rather face negative events, whereas opposition parties face positive events.

Central to any election forecast is the realistic assessment of the forecast error. We would see two different sources of uncertainty in the forecast:

1. Uncertainty in future events, i.e. shocks to voter's share
2. Uncertainty in polling

The second one can again be split in three sources:

1. The common bias of all pollster's for a specific party
2. The (house) bias of a specific pollster for a specific party
3. The polling uncertainty of a specific pollster for a specific party

Especially the first source is often completely ignored, but is also hard to estimate. For illustration we look at the finale polls just before the 2005 election and the outcome:

```r
knitr::kable(head(pollData %>% filter(Datum < "2005-09-18")))
```

Institut	Datum	CDU/CSU	SPD	GRÜNE	FDP	LINKE	AfD	Sonstige	Befragte
Allensbach	2005-09-16	0.415	0.325	0.070	0.080	0.085	NA	0.085	1682
Allensbach	2005-09-13	0.417	0.329	0.072	0.070	0.085	NA	0.085	2000
Emnid	2005-09-13	0.420	0.335	0.070	0.065	0.080	NA	0.080	2000
Forsa	2005-09-12	0.420	0.350	0.070	0.060	0.070	NA	0.070	2504
GMS	2005-09-12	0.420	0.330	0.080	0.070	0.070	NA	0.000	1008
Emnid	2005-09-10	0.405	0.345	0.070	0.070	0.080	NA	0.080	NA

```r
knitr::kable(Elections %>% filter(Year == 2005))
```

CDU.CSU	SPD	GRÜNE	FDP	LINKE	AfD	Sonstige	Year	Datum	Institut
0.352	0.342	0.081	0.098	0.087	NA		4	2005	2005-09-18	Election

For the largest party, the CDU/CSU, all polls are much too high, exceeding the real outcome between 5 and 7 percentage points. Such large changes virtually never happen within one week, such that in this and many other case, one can strongly presume that there is a common bias for all pollsters. This kind of bias will be adjusted for in the poll's post-processing, such that it will be approx. zero just after the election.

For the third source, one may presume that the sample size ("Befragte" in the poll data) is an important or even the only factor for determining uncertainty of a poll, but due to our experience the effect is negligable, as the pollsters have their own correction methods such that the actual poll outcome and the published polls differ substantially.

Model

Let $y_{party,t}$ be the true voter's share for a specific party, pt for a certain time point t, if an election would be held, whereby t is indexed on a weekly basis. Except for election dates, we cannot measure the voter's share directly, but have to rely on poll data. The poll data $poll_{pt,t,pll}$ for a party pt, time point t and pollster pll is published in irregular intervals by different institutes or pollsters. Beforehand, the poll data and the election voter's share were transformed on the logit scale:

$$logit(poll_{pt,t,pll}) \sim N(y_{pt,t} + bias_{pt,pll} + \epsilon_{\text{pollError}_{pt,t}}, \sigma_{pt,pll} + 1E-4)$$

The pollsters housebias and poll error are coming from a common distribution:

$$bias_{pt,pll} \sim N(0, \tau)$$

$$\sigma_{pt,pll} \sim N(0, \tau_2)$$

For an election this the bias and additional variance gets omitted and the upper expression gets simplified to:

$$logit(election_{pt,t}) \sim N(y_{pt,t}, 1E-4)$$

The common pollster bias follows an AR1-process and is slowly hovering around zero. The AR1 parameter was fixed to a value of 0.98 as it is not identifiable (aka cannot be learned from data) and corresponds to a half-life time of about 9 months. Its value is set to zero just after election as we assume that the pollsters adjust for the bias afterwards.

$$\epsilon_{\text{pollError}_{pt,t}} = 0.98 \cdot \epsilon_{\text{pollError}_{pt,t-1}} + \epsilon_{\text{polls}_{pt,t}}$$

The shifts in the common bias of the pollsters for a specific party follow a t-distribution with five degrees of freedom and a standard deviation σ_{pollbias} that is different for each party pt:

$$\epsilon_{\text{polls}_{pt,t}} \sim t(0, \sigma_{\text{pollbias}_{pt}}, df = 5)$$

$$y_{pt,t} = y_{pt,t-1} + \epsilon_{pt,t} + \nu_{pt,t} - \eta_{pt,t}$$

The shocks in voter's share follow a t-distribution with five degrees of freedom. The expectation $\mu_{pt,t}$ depends on the current state, i.e. state of government or opposition of the party, whereas the standard deviation $\sigma_{\text{shift}_{pt}}$ is different for each party pt:

$$\epsilon_{pt,t} \sim t(\mu_{pt,t}, \sigma_{\text{shift}_{pt}}, df = 5)$$

As explained in the previous section, the expectation of the weekly shift in voter's share might be influenced, whether a party is part of the government or the opposition.

$$\mu_{pt,t} = opposition_{pt,t} + government_{pt,t}$$

The positive short term memory effect $\nu_{pt,t}$ follows a process resembling AR1 :

$$\nu_{pt,t} = \theta_2 \cdot (\nu_{pt,t-1} + \epsilon_{pt,t-1})$$

The diminishing short term effect parameter $\eta_{pt,t}$ also follows a process resembling AR1. The additional parameter α governs the amount of "forgetting". For $\alpha = 1$, the long term effect is zero, while for $\alpha = 0$, the long-term effect equals the short term effect.

$$\eta_{pt,t} = \theta \cdot \eta_{pt,t-1} + (1 - \theta) \cdot \alpha(\nu_{pt,t} + \epsilon_{pt,t-1})$$

Within *STAN*, the model, transformed parameters and priors (mostly weakly informative, matched on the right scale) are the following:

```
transformed parameters{
  vector[YTOTAL] y[NParties];
  vector[YTOTAL] pollError[NParties];
  vector[YTOTAL] eps[NParties];
  vector[YTOTAL] w[NParties];
  real eta;
  real nu;

  for(i in 1:NParties){
    y[i,1]        =  y_start[i];
    pollError[i,1] =  pe_start[i];
    eta = 0;
    nu = 0;
    eps[i]  = epsilon[i] * sqrt(WT) * sigma_shift[i] + opposition +
      govMatrix[i] * government;
    for(n in 2:YTOTAL){
      y[i,n]          = y[i,n-1]  + eps[i,n] + nu - eta;
      pollError[i,n] =  0.98 * pollError[i,n-1] * electionIndikator[n] +
        epsilonPolls[i,n] * sqrt(WT2) * sigma_pollbias;
      nu = (nu + eps[i,n]) * theta2;
      eta = eta * theta + (alpha * (nu + eps[i,n])) * (1 - theta);
    }
    w[i] = y[i] + electionIndikator2 .* pollError[i];
```

```
  }
}
model {
  sigma_shift ~ normal(0, tau3);
  sigma_pollbias ~ normal(0.025, 0.0125);
  pe_start ~ normal(0, tau4);
  y_start ~ normal(0, 2);
  government ~ normal(0, 0.0005);
  opposition ~ normal(0, 0.0005);
  tau2 ~ normal(0, 0.05);
  theta ~ beta(10, 3);
  theta2 ~ beta(3, 3);
  alpha ~ beta(5, 5);
  tau ~ normal(0, 0.05);
  tau3 ~ normal(0, 0.03);
  tau4 ~ normal(0, 0.05);
  WT ~ scaled_inv_chi_square(5,1);
  WT2 ~ scaled_inv_chi_square(5,1);

  for(i in 1:NParties){
    housebias[i] ~ normal(0, tau);
    epsilon[i] ~ normal(0, 1);
    epsilonPolls[i] ~ normal(0, 1);
    sigma_sd[i] ~ normal(0.05, tau2);
    pollData[i] ~ normal(w[i,matchedDates] + IMatrix * housebias[i],
                         IMatrix * sigma_sd[i] + 1E-4 + Missing[i] * 1E4);
  }
}
```

STAN doesn't support ragged arrays. Thus, for missing data, especially for the "AfD" party which didn't appear before the last third of the observation date interval, an extremely high standard deviation was chosen.

Prepare Data

Before we compile and sample from the model, the poll and election data is binded together into a single data frame. We also form a dummy matrix of the pollsters ("IMatrix"), a missing matrix ("Missing") and indicators whether a certain observation of the combined data is an election ("electionIndikator") or the week after election ("electionIndikator2"). Additionally we have to match the weekly time sequence of the underlying state (voter's share) with the time of the actual polls ("matchedDates") and a matrix that assigns the government/opposition state for each party and point in time ("govMatrix").

```
# combine election and poll data
partyNames <- c("CDU/CSU", "SPD", "GRÜNE", "FDP", "LINKE", "AfD")
colnames(Elections)[1:length(partyNames)] <- partyNames
electionsTemp <- Elections[Elections$Datum < predDate, c("Institut", "Datum", partyNames)]
pollsTemp <- pollData[,c("Institut", "Datum", partyNames)]
names(electionsTemp) <- names(pollsTemp)

electionsTemp$Election = TRUE
pollsTemp$Election = FALSE

allData <- rbind(pollsTemp, electionsTemp)
allData <- allData %>% filter(!is.na(Datum)) %>% arrange(Datum)
```

```r
#save missing positions and replace missings
Missing <- t((is.na(allData[,c(partyNames)]))) * 1
for(i in partyNames){
  allData[, i] <- na.locf(na.locf(allData[, i], fromLast = FALSE, na.rm = FALSE),
                          fromLast = TRUE, na.rm = FALSE)}


#create pollster dummy matrix
IMatrix <- model.matrix(~ Institut - 1, data = allData)
IMatrix <- IMatrix[, - which(colnames(IMatrix) == "InstitutElection")]

#Remove pollster variable (institute), create numeric date (weeks since 1970)
allData <- allData %>% select(-Institut)
allData[,1] <- ceiling(as.numeric(difftime(allData[, "Datum"],
                                            as.Date("1970-01-01"), units = "weeks")))
allData <- as.matrix(allData)


pollData <- allData[, partyNames]

#Logit-transformation
pollData <- log(pollData / (1 - pollData))

#create weekly sequence for state-space
timeSeq <- seq(min(allData[,"Datum"]), max(allData[,"Datum"]) + 52, by = 1)
matchedDates = match(allData[,"Datum"], timeSeq)

#get constants
NParties <- ncol(pollData)
NTOTAL = nrow(pollData)
YTOTAL = length(timeSeq)
NPollsters = ncol(IMatrix)

#create matrix of government parties
source('R/createGovMatrix.R', encoding = 'UTF-8')
govMatrix <- createGovMatrix(partyNames, YTOTAL, Elections, timeSeq)

#indicator of weeks of state-space time sequence with election and week after election
electionIndikator <- rep(1, YTOTAL)
electionIndikator[match(allData[rowSums(IMatrix) == 0, "Datum"], timeSeq) + 1] <- 0
electionIndikator2 <- rep(1, YTOTAL)
electionIndikator2[match(allData[rowSums(IMatrix) == 0, "Datum"], timeSeq)] <- 0
```

Sampling with RStan

Now we can compile the model and sample from it. The quite complex nature of the model requires a high tree depth of 17 or more. Together with the large number of parameters (several thousands) it takes several days to complete it, depending in the machine. A parallelization using map_rect is planned, however. For this report, we pre-computed several models at different points in time and saved the samples.

```r
#transpose data for stan script (due to indices)
pollData <- t(pollData)
govMatrix <- t(govMatrix)

# mpModel <- stan_model(file = "stan_models/lsModelUni.stan")
# f <- sampling(mpModel,
```

8

```r
#              data = list(NTOTAL = NTOTAL,
#                          YTOTAL = YTOTAL,
#                          NPollsters = NPollsters,
#                          NParties = NParties,
#                          matchedDates = matchedDates,
#                          pollData = pollData,
#                          IMatrix = IMatrix,
#                          govMatrix = govMatrix,
#                          Missing = Missing,
#                          electionIndikator = electionIndikator,
#                          electionIndikator2 = electionIndikator2),
#              iter= 700, warmup = 600, chains = 4, cores = 4, seed = 124567,
#              control = list(max_treedepth = 17, adapt_delta = 0.9))

load("model_results/Model_2017_03_25.RData")
```

Results

Visualization

Six months before election, the social democratic party ("SPD") experienced a large increase (from slightly over 20% to more than 30% in the polls) in voter's sharedue to a hype of the shortly announced chancellor candidate Martin Schulz. From the picture below, we can clearly see a mean-reversing trend until election date for the "SPD" (in red):

```r
plotData <- lapply(1:NParties, function(x){
  data.frame(estimate = samples$y[,x,] %>% logistic %>% colMeans,
             lower = apply(samples$y[,x,] %>% logistic, 2, quantile, 0.025),
             upper = apply(samples$y[,x,] %>% logistic, 2, quantile, 0.975),
             time = as.POSIXct(timeSeq*60*60*24*7, origin = "1970-01-01"),
             party = factor(rownames(pollData)[x]))
}) %>% bind_rows()
plotPollData <- as.data.frame(logistic(t(pollData)))
plotPollData <- cbind(plotPollData,
                      data.frame(time = as.POSIXct(timeSeq[matchedDates]*60*60*24*7,
                                                   origin = "1970-01-01")))

plotPollData <- plotPollData %>% as_tibble %>% gather(key = "party",
                                                      value = "proportion", -time)
partyColors <- c("dodgerblue4", "black", "yellow", "green", "purple", "red")
ggplot(data = plotData, aes(x = time, y = estimate, group = party,
                            colour = party)) + geom_line() +
  geom_ribbon(aes(ymin = lower, ymax = upper,
                  fill = party),alpha = 0.3, colour = NA, show.legend = FALSE) +
    scale_color_manual(values=partyColors) +   scale_fill_manual(values=partyColors) +
  xlim(as.POSIXct(c("2016-01-01", "2017-09-30"))) + ylim(0,0.42) +
  geom_vline(xintercept = as.POSIXct("2017-09-25")) +
  geom_vline(xintercept = as.POSIXct(predDate)) +
  annotate(geom = "text", x=as.POSIXct(predDate), y=0,
           label="prediction date", angle = 25, size = 2) +
  annotate(geom = "text", x=as.POSIXct("2017-09-25"),
           y=0, label="election date", angle = 25, size = 2)  +
  geom_point(data = plotPollData, aes(x = time, y = proportion, group = party))
```
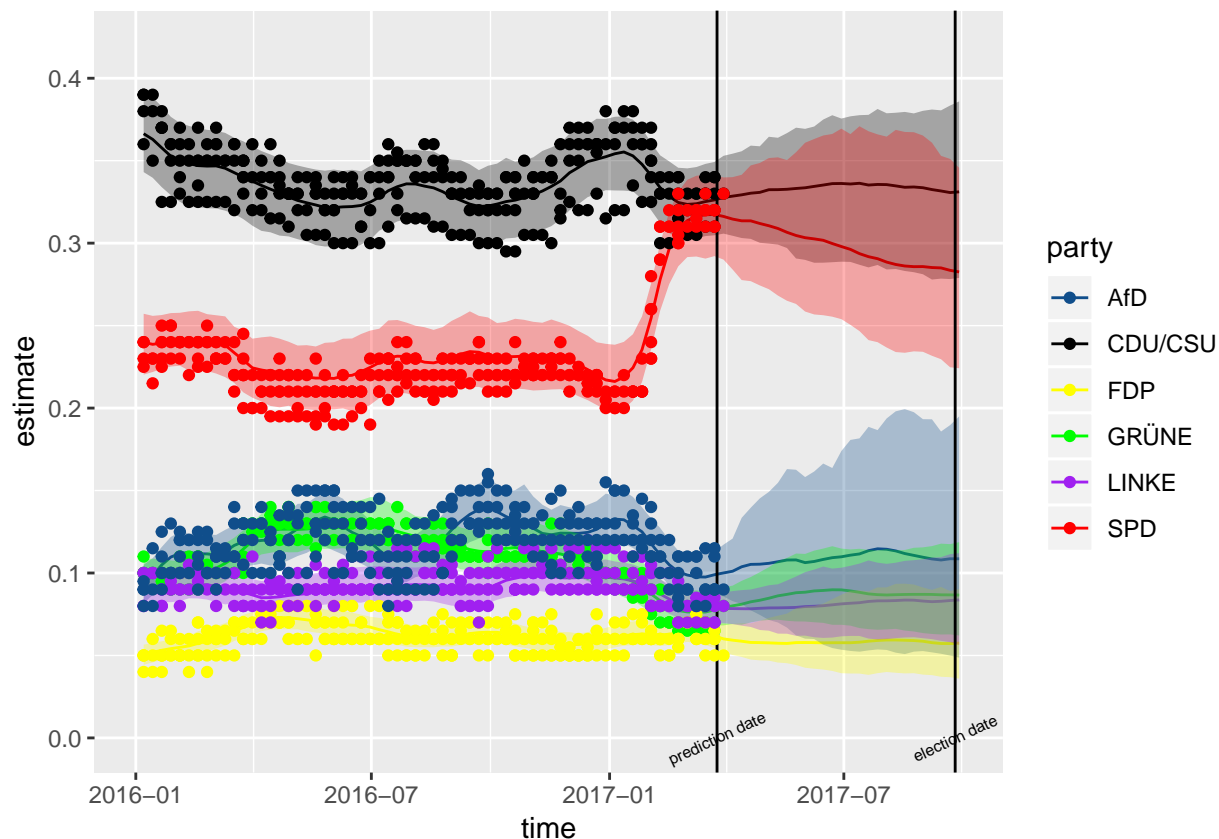
Assessing predictive performance

To assess the predictive performance, we evaluate the prediction with the RMSE ("Root Mean Square Error"). As competitors or comparison we use the most recent poll as well as the average of the most recent poll of each pollster:

```r
predElection <- samples$y[,,which(timeSeq ==
                          ceiling(as.numeric(difftime(as.Date("2017-09-25"),
                                  as.Date("1970-01-01"),
                                  units = "weeks"))))] %>%
  logistic %>% colMeans
#Model predictions:
predElection
```

```
## [1] 0.33112080 0.28255260 0.08664112 0.05745475 0.08366454 0.10860608
```

```r
#Recent polls:
knitr::kable(pollsTemp %>% filter(Datum <= predDate) %>%
  arrange(desc(Datum)) %>%
  group_by(Institut) %>% slice(1))
```

Institut	Datum	CDU/CSU	SPD	GRÜNE	FDP	LINKE	AfD	Election
Allensbach	2017-02-22	0.33	0.305	0.080	0.070	0.080	0.085	FALSE
Emnid	2017-03-25	0.33	0.330	0.080	0.050	0.080	0.090	FALSE
Forsa	2017-03-22	0.34	0.310	0.070	0.060	0.070	0.090	FALSE
Forsch'gr.Wahlen	2017-03-10	0.34	0.320	0.070	0.050	0.080	0.090	FALSE
GMS	2017-03-23	0.34	0.310	0.080	0.060	0.080	0.090	FALSE

Institut	Datum	CDU/CSU	SPD	GRÜNE	FDP	LINKE	AfD	Election
Infratestdimap	2017-03-23	0.32	0.320	0.080	0.060	0.070	0.110	FALSE
INSA	2017-03-20	0.31	0.320	0.065	0.065	0.085	0.115	FALSE

```r
#Compute competitors
predRecent <- pollsTemp %>% filter(Datum <= predDate) %>%
  arrange(desc(Datum)) %>% select(partyNames) %>%
  slice(1) %>% unlist
predRecent
```

```
## CDU/CSU     SPD   GRÜNE     FDP   LINKE     AfD
##    0.33    0.33    0.08    0.05    0.08    0.09
```

```r
predAvgRecent <- pollsTemp %>% filter(Datum <= predDate) %>%
  arrange(desc(Datum)) %>%
  group_by(Institut) %>% slice(1) %>% ungroup %>%
  select(partyNames) %>% colMeans %>% unlist
predAvgRecent
```

```
##    CDU/CSU        SPD      GRÜNE        FDP      LINKE        AfD
## 0.33000000 0.31642857 0.07500000 0.05928571 0.07785714 0.09571429
```

```r
#RMSE model and competitors
sqrt(mean((predElection - unlist(Elections %>% filter(Year == 2017) %>%
                        select(partyNames)))^2)) %>% round(4)
```

```
## [1] 0.0384
```

```r
sqrt(mean((predRecent - unlist(Elections %>% filter(Year == 2017) %>%
                        select(partyNames)))^2)) %>% round(4)
```

```
## [1] 0.0583
```

```r
sqrt(mean((predAvgRecent - unlist(Elections %>% filter(Year == 2017) %>%
                        select(partyNames)))^2)) %>% round(4)
```

```
## [1] 0.0517
```

We can see, that –at least at this point in time, six months before election– our model estimates beat the pollsters by a fairly large amount, a 26% lower RMSE than the average of the recent predictions.

Now, we evaluate the performance on a monthly basis from 12 months before election until the election week:

Troubleshooting

The iterations exceeds maximum tree size, for tree_depth <18 (but results are ok for tree_depth larger than 15). Fortunately, there are no divergent samples for adapt_stepsize value of 0.9. The models have been running with rather low sample size and

Outlook

Model speed up

The model is quite slow and about 1 week to compute - Use map_rect on party level to speed up computation - Faster computation with future *STAN* versions, model could benefit from faster indexin in *STAN* 2.20 - Potentially re-parameterize model

Model extensions

Instead of independent shifts and common pollster errors, i.e. independent t-distributions, multivariate t-distributions accounting for correlated shocks and poll errors could be employed. Also further improvements by introducing a mid-term memory parameter are possible. Both extensions can be found in the stan_models directory. However, these are even slower and the predictive performance as not been studied in detail.