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 for modelling voting intentions and election results using polling 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) by introducing a long-short term event memory effect to the problem. This effect serves as a reasonable explanation for the observation that the vote share partially tends to reverse to the party's long-term trend after larger short-term movements. Any event influencing the vote share of a party is presumed to have a convex shape, and be 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 forgetfulness. 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 vote share, whereas the opposite effect is observed for opposition parties, mid- and long-term predictions of election outcomes can be considerably improved. The Stan-model is fitted and evaluated on polling data from seven pollsters of the German national elections ("Bundestagswahl") from 1994 to 2017, where low double-digit (out-of-sample) improvements in prediction performance can be seen between 3- and 18-months prior to the 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 created.

Data

We have data on more than 4,000 polls from seven different pollsters between November 1st, 1994, through the current date for the German federal election ("Bundestagswahl"). These data are scraped from the web page www.wahlrecht.de, which collects all available polling data and is frequently updated. Furthermore, we have election outcome data for all elections since 1998, and the respective parties forming the government and opposition. Data are available for the six large parties: CDU/CSU, SPD, Die Grünen, FDP, Die Linke, and AfD.

Get Poll and Election Data

First, we set a prediction date – March 25th, 2017 – exactly six months prior to the election.

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

The polling data for the German election, the "Bundestagswahl", is scraped from the web 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

```
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

The high levels of attention that Nate Silver's US Presidential election forecasts on fivethirtyeight.com have received demonstrate how data-based election forecasts are of great interest to the public, as well as in academia. There have been attempts to model elections 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 by Simon Jackman (https://www.tandfonline.com/doi/abs/10.1080/10361140500302472) and was used to predict 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 of removing bias from different pollsters, in our experience it is not superior to very simple poll-averaging methods for mid- or long-term forecasts. The question here is whether we can do better than just taking 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 those for the US Presidential election by Nate Silver (https://fivethirtyeight.com/features/a-users-guide-to-fivethirtyeights-2016-general-election-forecast/), multi-party forecasts for the UK election (http://www.electionforecast.co.uk), or the German federal election (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 in one way or another. For 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

FiveThirtyEight states that:

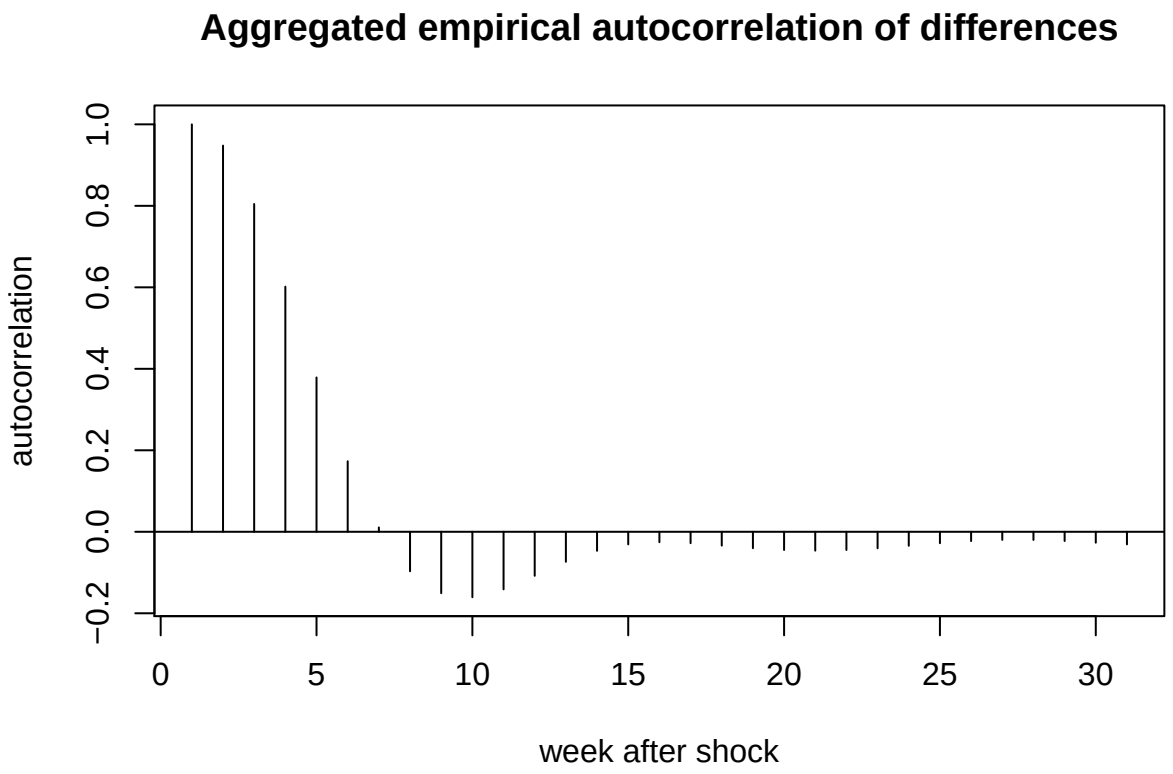
> Empirically, using more smoothing early in the race and less smoothing late in the race works best. In other words, the trend line starts out being quite conservative and becomes more aggressive as Election Day approaches.

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 by Drew Linzer (https://votamatic.org/wp-content/uploads/2013/07/Linzer-JASA13.pdf), a mean-reversion effect is incorporated for US Presidential elections by implicitely assuming that on the state level, the vote share is going to return to its long-term mean. However, such a form of mean stationarity is unrealistic, particularly in multi-party systems, as there are clear trends over time for different parties. For example, formerly successful parties can disappear entirely. Therefore, we propose a mixture between a non-stationary random walk and a mean-reversing stationary process for the vote 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 a few weeks the event or news might slowly disappear out of the public's mind, but won't entirely. The remaining effect accounts for the long-term effect of the initial event.

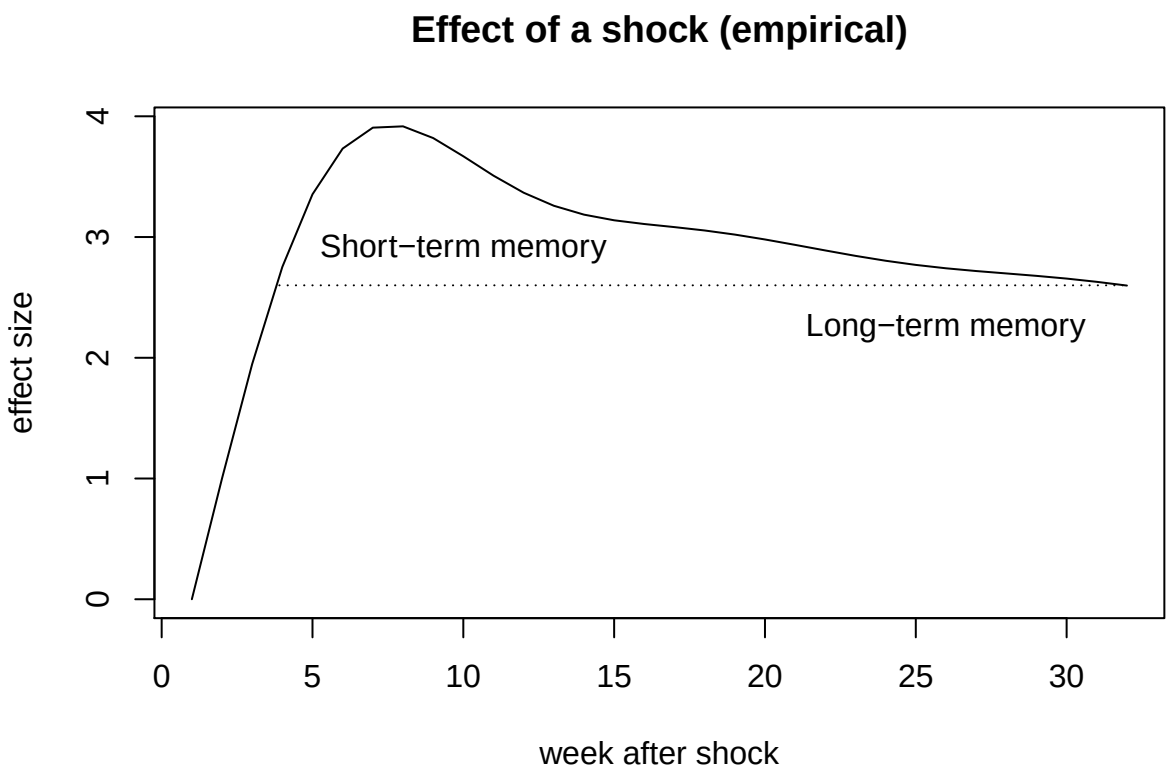
Is there evidence for this in our data on the German federal election? To investigate empirically, we smooth the polls on a weekly basis using a smoothing spline, compute the autocorrelation functions (ACFs) on the weekly differences, and average the ACFs on all parties, exluding AfD (as this right-wing party appeared only a few years ago):

```
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",
     main = "Aggregated empirical autocorrelation of differences")
abline(h = 0)
```



We can clearly see that there is a positive autocorrelation within the first weeks, turning negative between eight and 15 weeks after the initial shock. The latter interval can be seen as the period where the events get out of the short-term public memory. A clearer picture of the mean-reversing nature of polls can be seen when we integrate the autocorrelation function:

```
plot(c(0,cumsum(acfs)), type = "l", xlab = "week after shock", ylab = "effect size",
     main = "Effect of a shock (empirical)")
segments(x0 = 3.9, x1 = 32, y0 = 2.6, y1 = 2.6, lty = 3)
text(x = 26, y = 2.25, "Long-term memory")
text(x = 10, y = 2.9, "Short-term memory")
```



Another peculiarity of elections over time is the effect of government parties generally losing vote share, while opposition parties tend to gain voters. For the German federal elections between 1998 and 2017, the government party lost vote share in 10 of 12 cases in the following election, and the opposition parties gained vote share in 15 of 20 cases. The same observation can be seen in neighboring countries with similar multi-party election systems such as the Netherlands or Sweden. This may be influenced by the fact that government parties are more likely to face negative events, whereas opposition parties experience more positive events.

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

1. Uncertainty about future events, i.e. shocks to vote share
2. Uncertainty in polling

The second point can be further split in three sources:

1. The common bias of all pollsters 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

The first source in particular is often completely ignored, and is also challenging to estimate. To illustrate, we look at the final polls just before the 2005 election, and the outcome:

```
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

Institut	Datum	CDU/CSU	SPD	GRÜNE	FDP	LINKE	AfD	Sonstige	Befragte
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

```
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 give an estimate that is much too high, exceeding the real outcome by between five and seven percentage points. Such large changes virtually never happen within one week, such that in this and many other cases, one can strongly presume that there is a common bias for all pollsters. This kind of bias will be adjusted for in the pollsters' post-processing, such that it will be approximately zero just after the election.

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

Model

To model the vote share over time we implemented a "long-short term memory" state-space model in *STAN*. It incorporates a parameterized long- and short-term memory structure based on weekly events or shocks, incorporates the party status (government or opposition), accounts for different sources of uncertainty (future events, common pollster error, pollster-party-house bias, polling uncertainty), and heavy-tail errors.

Let $y_{party,t}$ be the true vote 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. We cannot measure the vote share directly except on election days, so have to rely on polling data. The polling 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. The polling data and the election vote share were transformed on the logit scale to guarantee estimates in the (0, 1) interval after re-transformation, and to provide variance regulation:

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

The pollsters' house bias and poll error come 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. 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 (a.k.a., cannot be learned from the data) and corresponds

to a half-life time of about nine months. Its value is set to zero just after an 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 vote share follow a t-distribution with five degrees of freedom. The expectation $\mu_{pt,t}$ depends on the current state, i.e. state of being in government or opposition for 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 vote share might be influenced by 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" that takes place. 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.

Data Preparation

Before we compile and sample from the model, the polling and election data are bound together into a single data frame. We also form a dummy matrix of the pollsters ("IMatrix"), a missing matrix ("Missing") and indicators on 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 (vote 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 polling 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)

#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 on 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.

```
#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,
#               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))
# samples <- extract(f)
load("model_results/Model_2017_03_25.RData")
```

Results

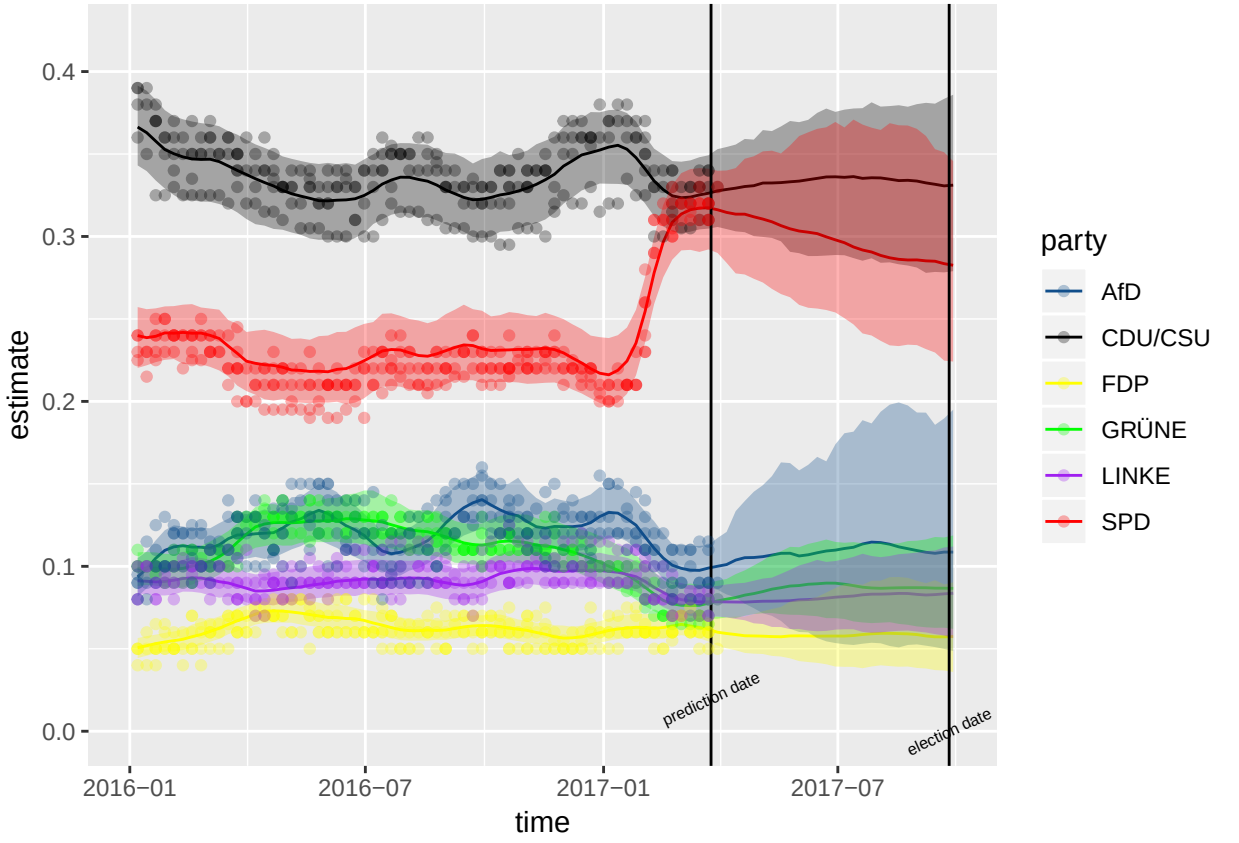
Interpretation and visualization of model results

Prediction six months prior to the election

Six months before the election, the Social Democratic Party (SPD) experienced a large increase (from slightly over 20% to more than 30% in the polls) in vote share due to hype from the recently-announced Chancellor candidate Martin Schulz. From the picture below, we can clearly see a mean-reversing trend until the election date for the SPD (in red). The model estimates of the vote share for each party are presented as solid lines, with additional 95% credible bands from January 1st, 2016, until the date of election on September 24th, 2017. The date of prediction (vertical line) and the poll results (points) are also added:

```
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=0.02,
           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), alpha = 0.3)
```



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 and the average of the most recent poll of each pollster:

```
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
```

```
#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
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

```
#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
```

```
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
```

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

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

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

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

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

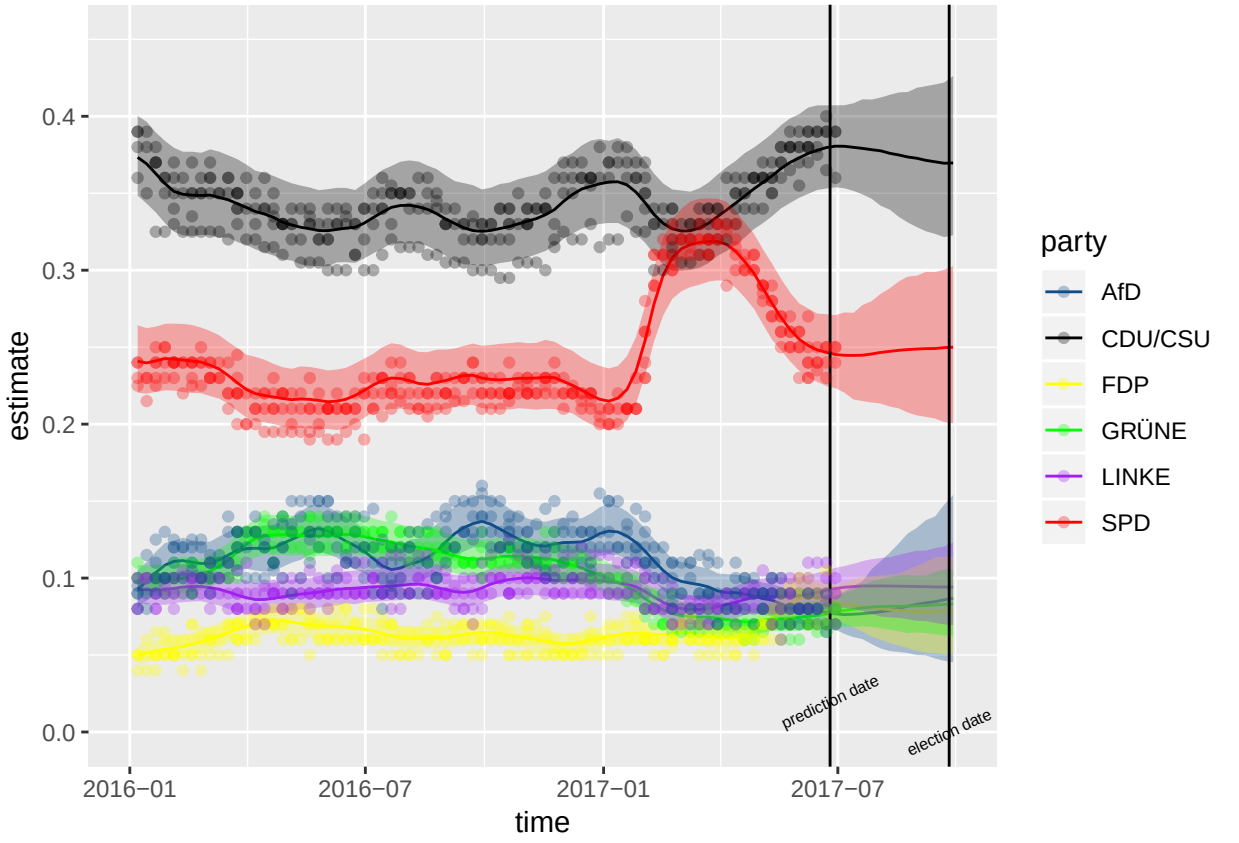
```
## [1] 0.0517
```

We can see that – at least at this point in time, six months before the 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 take a look at a second prediction date, June 25th, 2017 – three months before the election:

```
predDate <- "2017-06-25"
pollData <- getPollData(predDate) %>% arrange(desc(Datum))
```

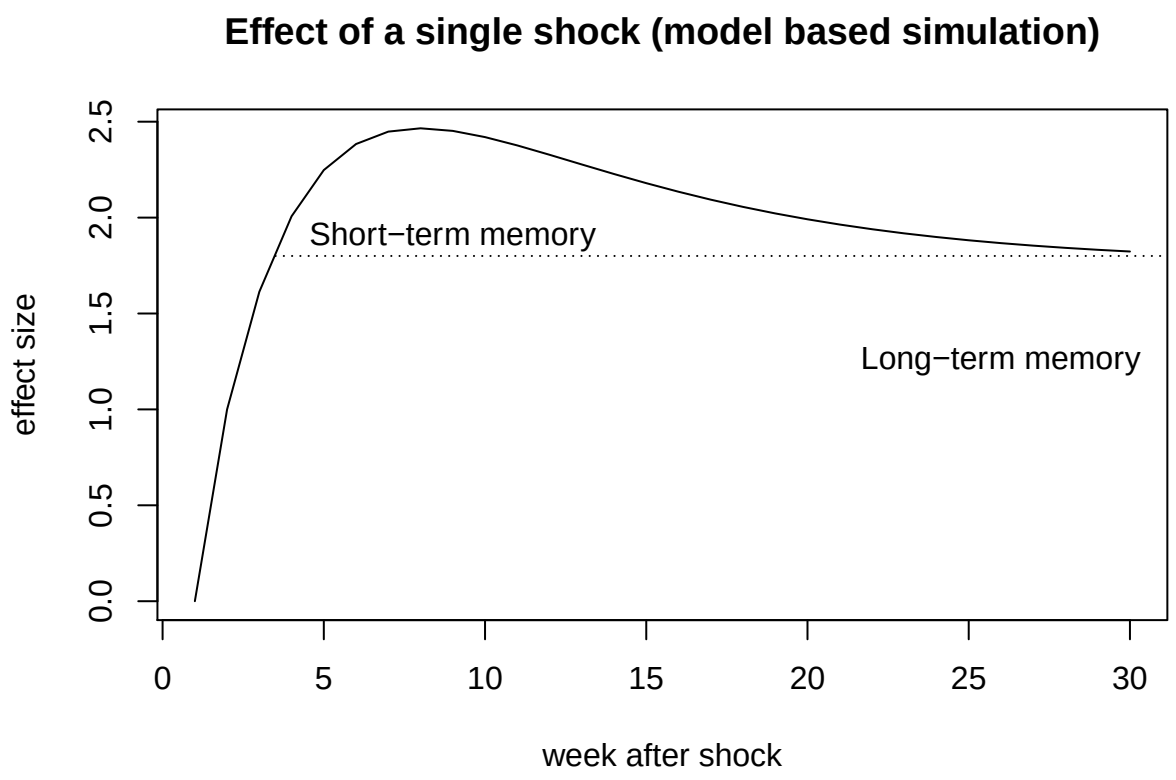
```
Elections <- read.csv2("data/Elections.csv", encoding = 'UTF-8')
Elections$Datum <- as.Date(Elections$Datum)
source('R/preparePollData.R', encoding = 'UTF-8')
data <- preparePollData(pollData, Elections, predDate)
# mpModel <- stan_model(file = "stan_models/lsModelUni.stan")
# f <- sampling(mpModel,
#               data = data,
#               iter= 700, warmup = 600, chains = 4, cores = 4, seed = 124567,
#               control = list(max_treedepth = 17, adapt_delta = 0.9))
# samples <- extract(f)
load("model_results/Model_2017_06_25.RData")
source('R/plotElectionData.R', encoding = 'UTF-8')
plotElectionData(samples, data, predDate)
```



In the Motivation section we simulated the effect of a single shock in time with the model and the estimated parameters. We instantly see that the graph is very similar to the empirical evidence on the model-free smoothed data in the Motivation section. We have a maximum effect around seven to eight weeks after the initial shock, which partially wears off in the following weeks.

```
### effect of single shock of size 1
adds <- 0
substract <- 0
weeks <- 30
y <- rep(0,weeks)
epsilon <- c(0, 1, rep(0,weeks-2))
theta =  mean(samples$theta)
```

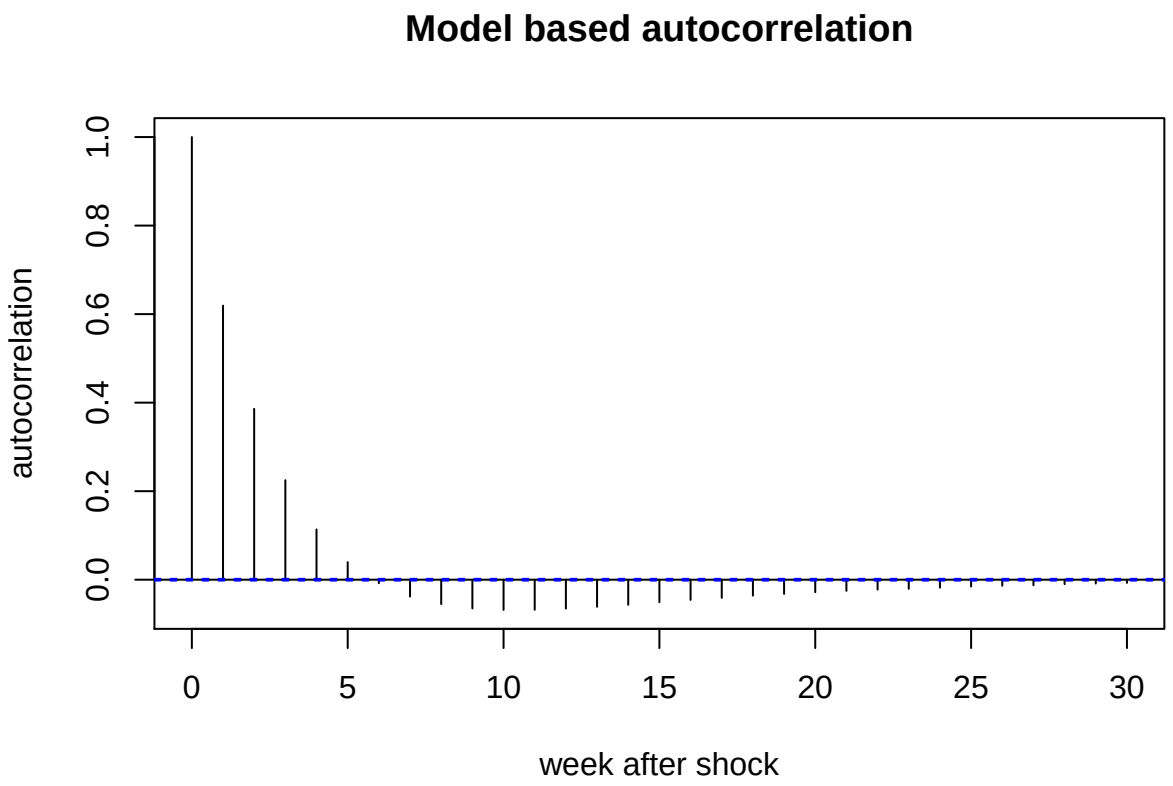
```r
theta2 = mean(samples$theta2)
alpha = mean(samples$alpha)

for(n in 2:weeks){
  y[n]         = y[n-1]  + epsilon[n] + adds - substract;
  adds = (adds + epsilon[n]) * theta2;
  substract = substract * theta + (alpha * (adds + epsilon[n])) * (1 - theta);
}

plot(y, type = "l", main = "Effect of a single shock (model based simulation)",
     xlab = "week after shock", ylab = "effect size")
segments(x0 = 3.5, x1 = 31, y0 = 1.8, y1 = 1.8, lty = 3)
text(x = 26, y = 1.25, "Long-term memory")
text(x = 9, y = 1.9, "Short-term memory")
```



We can also do a full simulation and compute the autocorrelation function. Again, the result is very similar to the results on the smoothed polling data in the Motivation section. This is a quite strong evidence.

```r
#Complete simulation; compute ACF
epsilon <- rt(1000000, df = 5)
y <- rep(0,1000000)
for(n in 2:1000000){
  y[n]         = y[n-1]  + epsilon[n] + adds - substract;
  adds = (adds + epsilon[n]) * theta2;
  substract = substract * theta + (alpha * (adds + epsilon[n])) * (1 - theta);
}
```

```r
acf(diff(y), 30, xlab = "week after shock", ylab = "autocorrelation",
    main = "Model based autocorrelation")
```



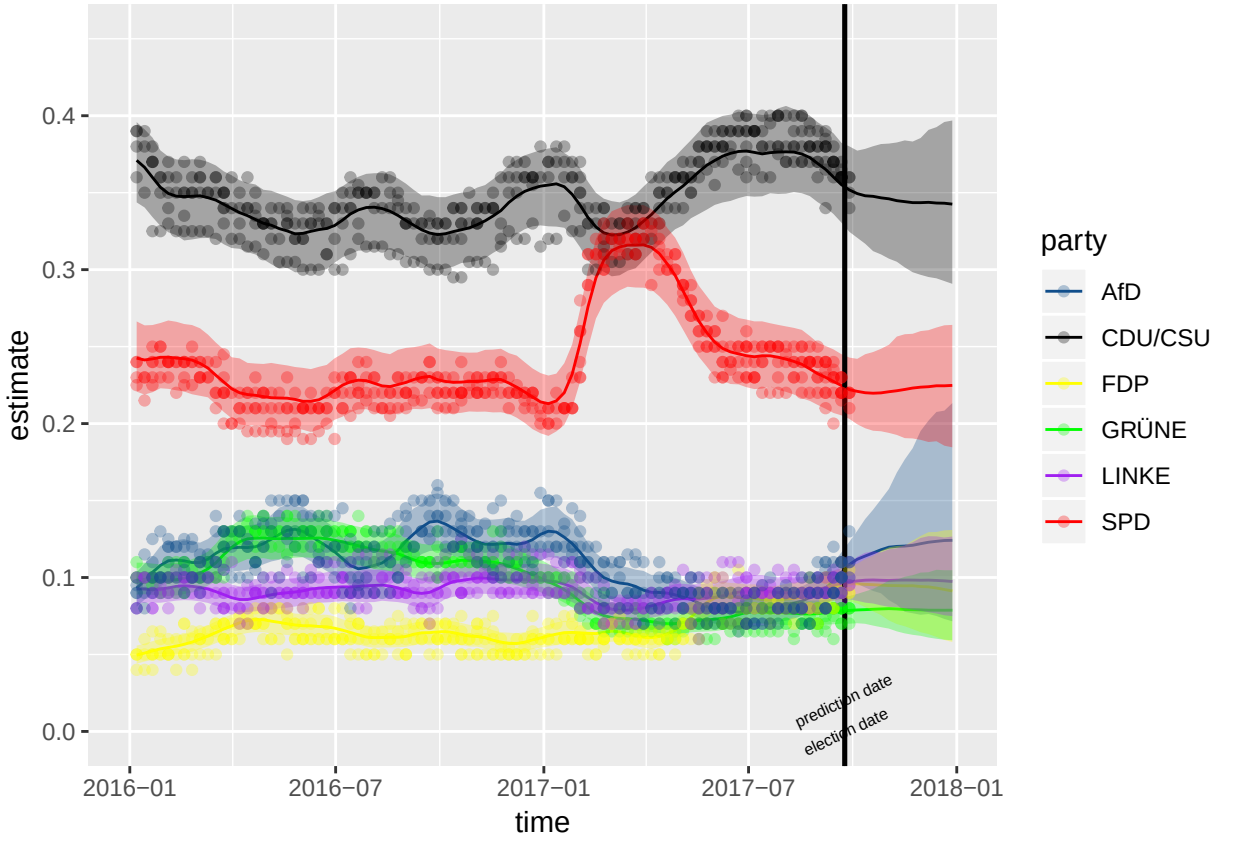
In the meantime, the uplift in vote share for the SPD has vanished almost completely, and the largest party, CDU/CSU (the party of Chancellor Angela Merkel) gained some share. Our model predicts a mean-reversing trend for the CDU/CSU until election (i.e. a decrease in vote share), but also for the AfD, where the model predicts an increase in vote share. It also becomes obvious that the AfD has much larger credible intervals compared to the other three smaller parties on election day. This might be due to the fact that this party was only founded a few years ago and is less stable, therefore experiencing larger swings in vote share.

Prediction the day prior to the election

Lastly, we take a closer look at the day before the election:

```r
predDate <- "2017-09-23"
pollData <- getPollData(predDate) %>% arrange(desc(Datum))
Elections <- read.csv2("data/Elections.csv", encoding = 'UTF-8')
Elections$Datum <- as.Date(Elections$Datum)
source('R/preparePollData.R', encoding = 'UTF-8')
data <- preparePollData(pollData, Elections, predDate)
# mpModel <- stan_model(file = "stan_models/lsModelUni.stan")
# f <- sampling(mpModel,
#               data = data,
#               iter= 700, warmup = 600, chains = 4, cores = 4, seed = 124567,
#               control = list(max_treedepth = 17, adapt_delta = 0.9))
# samples <- extract(f)
load("model_results/Model_2017_09_23.RData")
```

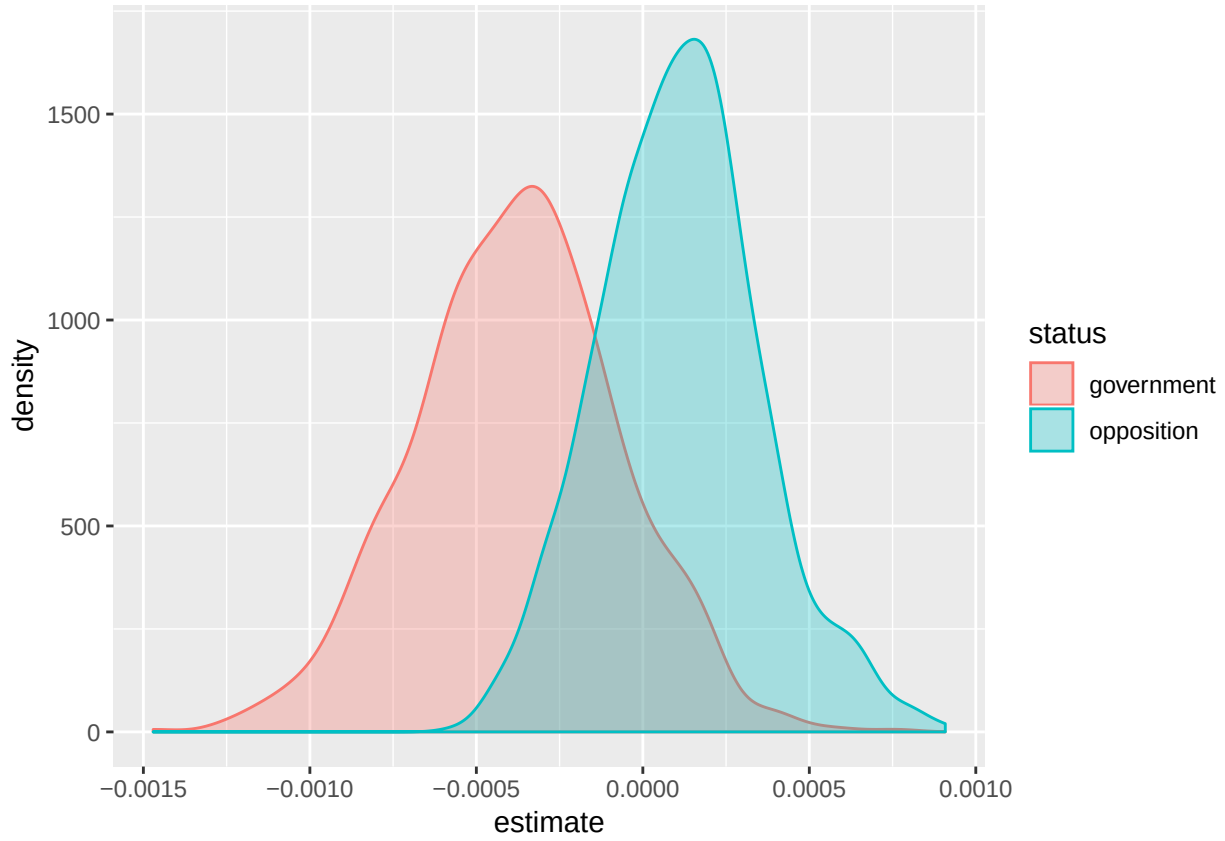
```
plotElectionData(samples, data, predDate, end = "2017-12-31")
```



The CDU/CSU lost some vote share, while the AfD gained quite a bit, confirming the estimated trend from the model above.

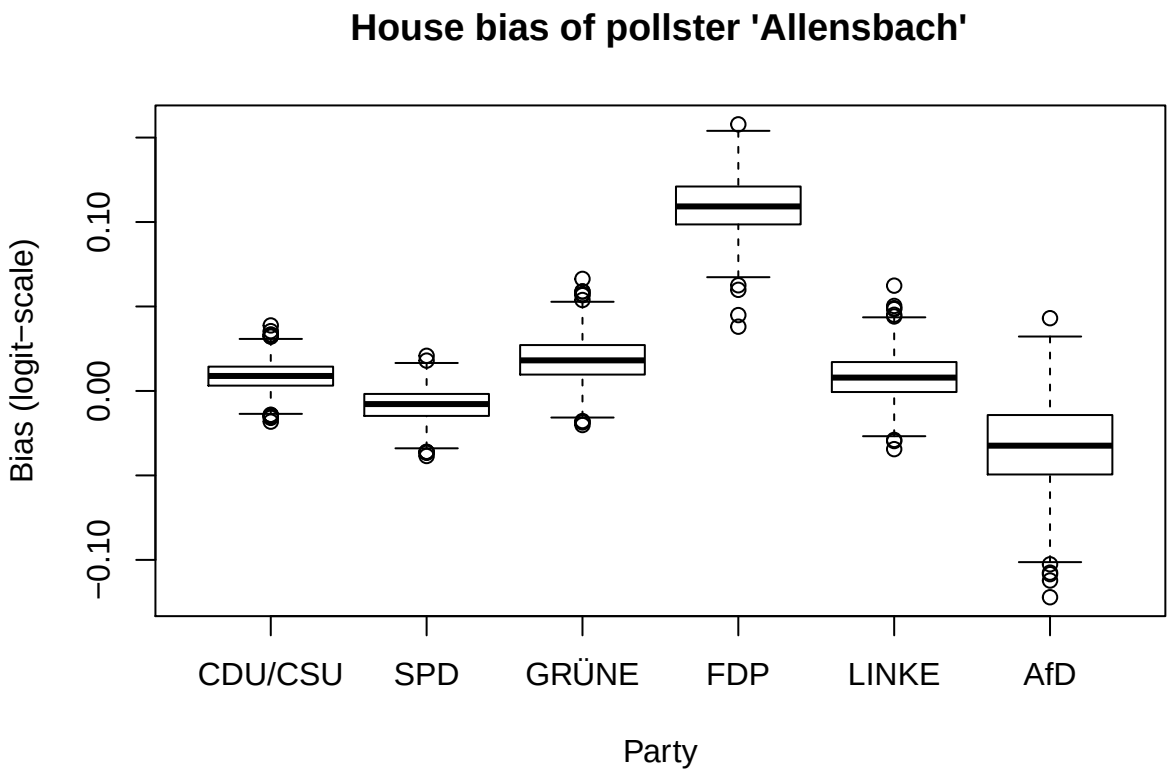
We can also take a look at the effect of government and opposition status on a party. As presumed, being in opposition has a positive influence (in terms of expected value) on the shock, while being in government has a negative effect:

```
goData <- data.frame(estimate = c(samples$opposition, samples$government),
           status = rep(c("opposition", "government"), each = length(samples$government)))

ggplot(goData, aes(estimate, colour= status, fill = status)) + geom_density(alpha = 0.3)
```



The model allows us to also look at the "house bias" , i.e. the bias of a pollster on a specific party or the polling uncertainty. In the example of the "Allensbach" pollster, we see that this pollster strongly favors the FDP and has a negative bias for the AfD party:

```
biasAllensbach <- samples$housebias[,,1]
colnames(biasAllensbach) <- partyNames
boxplot(biasAllensbach, xlab = "Party", ylab = "Bias (logit-scale)",
        main = "House bias of pollster 'Allensbach'")
```



Assessing predictive performance

To assess the performance of the proposed model, we have computed models on a monthly basis from 12 months before the election until the election in 2017. Now, we evaluate the (out-of-sample) performance. We have four competitors:

1. The average on party-level of the most recent poll of all seven pollsters ("Average")
2. The most recent poll ("Recent")
3. The best pollster for the 2017 election, which is the pollster "INSA" ("Best")
4. The best pollster for each time point ("Min")

Note that competitors 3 and 4 have an "unfair" hindsight advantage, as they are not available until we know the election results.

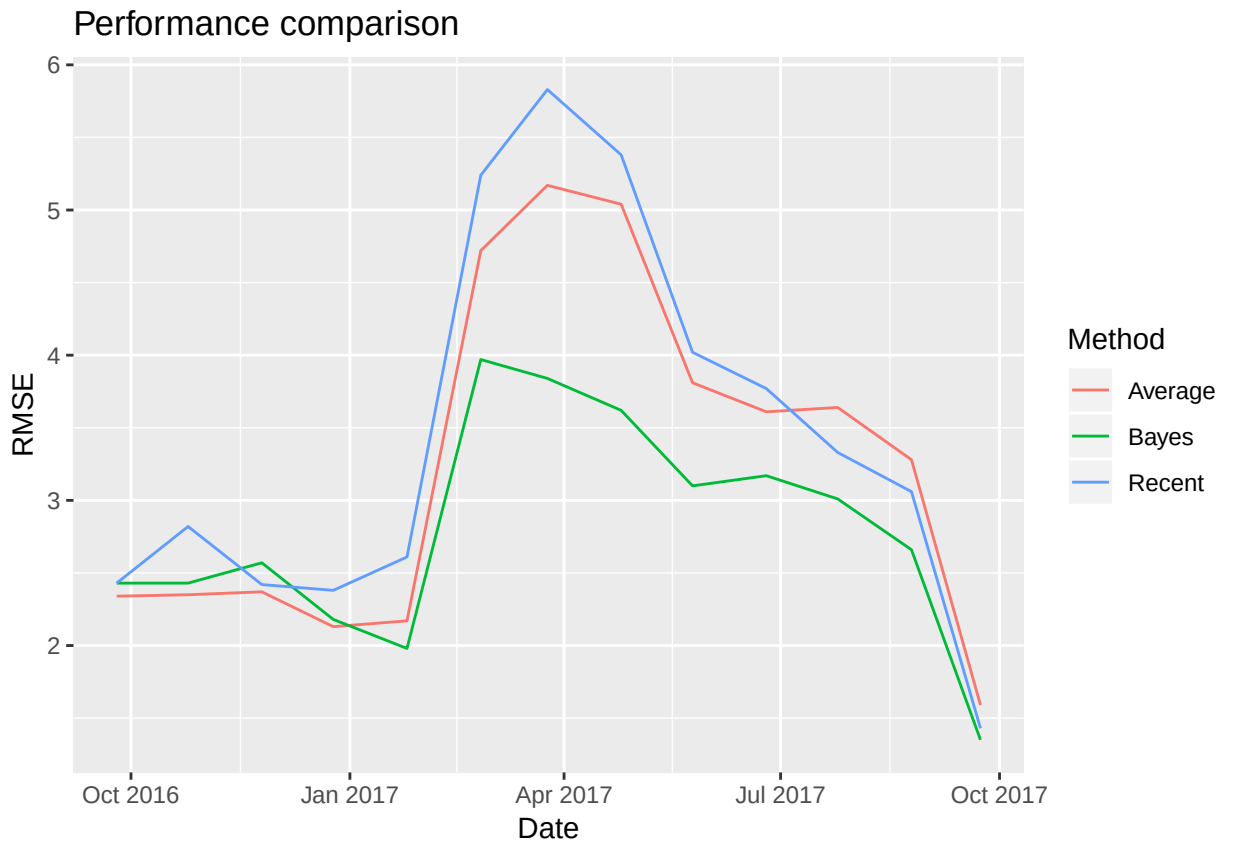
The Root Mean Square Errors averaged over all parties is stored in an excel file:

```
library("xlsx")
predictivePerformance <- read.xlsx("model_results/PredictivePerformance.xlsx",
                                   14, header=TRUE)
knitr::kable(predictivePerformance)
```

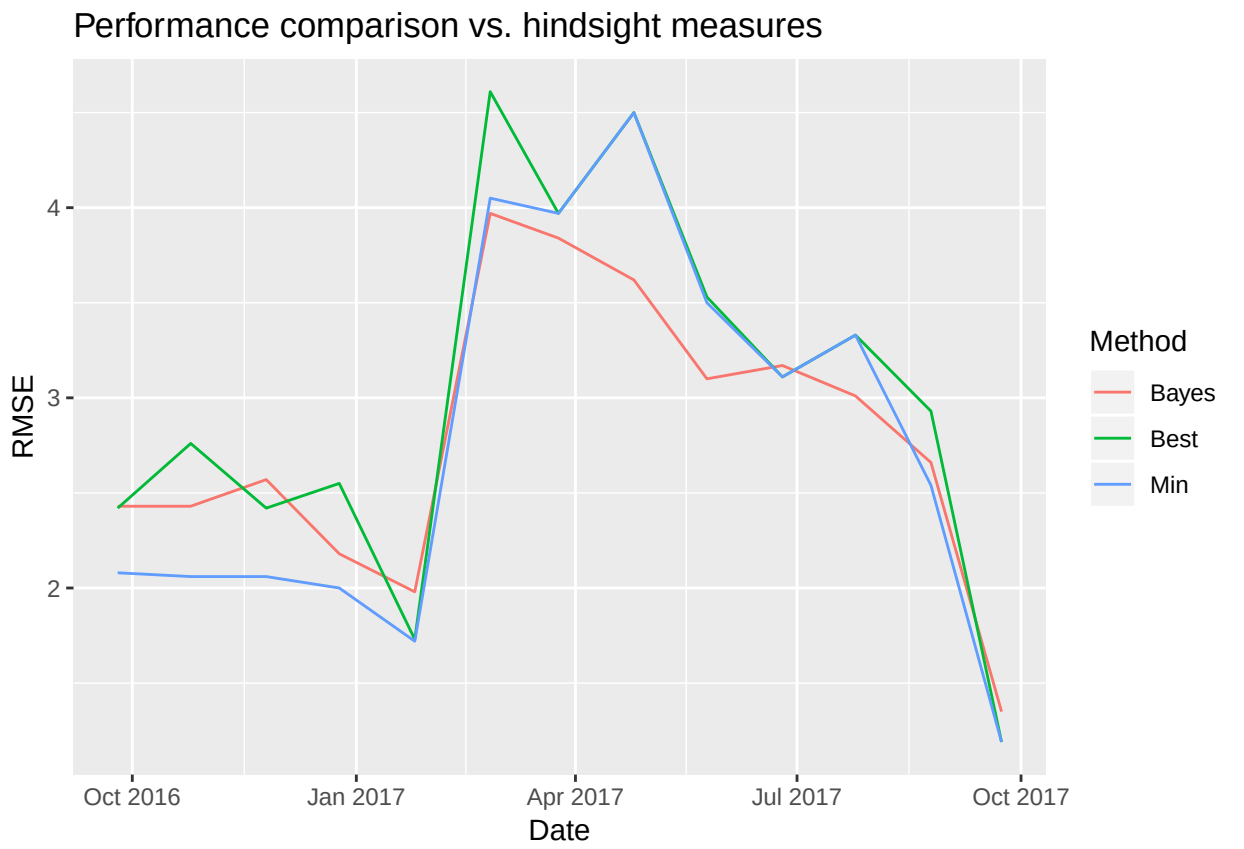
Date	Bayes	Average	Min	Best	Recent
2016-09-25	2.43	2.34	2.08	2.42	2.43
2016-10-25	2.43	2.35	2.06	2.76	2.82
2016-11-25	2.57	2.37	2.06	2.42	2.42
2016-12-25	2.18	2.13	2.00	2.55	2.38
2017-01-25	1.98	2.17	1.72	1.73	2.61

Date	Bayes	Average	Min	Best	Recent
2017-02-25	3.97	4.72	4.05	4.61	5.24
2017-03-25	3.84	5.17	3.97	3.97	5.83
2017-04-25	3.62	5.04	4.50	4.50	5.38
2017-05-25	3.10	3.81	3.50	3.53	4.02
2017-06-25	3.17	3.61	3.11	3.11	3.77
2017-07-25	3.01	3.64	3.33	3.33	3.33
2017-08-25	2.66	3.28	2.54	2.93	3.06
2017-09-23	1.35	1.59	1.19	1.19	1.43

```
longData <- gather(predictivePerformance, "Method", "RMSE", -Date)
ggplot(longData %>% filter(Method %in% c("Bayes", "Average", "Recent")),
       aes(y=RMSE, x= Date, colour = Method)) + geom_line() +
  ggtitle("Performance comparison")
```




```
ggplot(longData %>% filter(Method %in% c("Bayes", "Min", "Best")),
       aes(y=RMSE, x= Date, colour = Method)) + geom_line() +
  ggtitle("Performance comparison vs. hindsight measures")
```




```
#average over all time points
round(colMeans(predictivePerformance[,-1]), 3)
```

```
##   Bayes Average     Min    Best  Recent
##   2.793   3.248   2.778   3.004   3.440
```

The proposed Bayesian long-short term memory state-space model was able to beat a poll average by 15% on average over the year prior to election, and by almost 20% compared to the most recent election. It even turned out to beat the best pollster for this particular election – which is only available in hindsight – and is on par with best pollster for each point in time, a remarkable result. First test results show that these results are confirmed when applying the model on the previous 2013 election.

Troubleshooting

The model iterations exceed maximum tree size, for tree_depth < 18 (but results are alright for tree_depth larger than 15). As each additional tree_depth increases the computation time about two-fold, this leads to very slow computation. Fortunately, there are no divergent samples for adapt_stepsize value of 0.9. The models have been running with rather low sample size due to time constraints, and a bit longer sampling periods are desirable.

Outlook

Performance improvement

The model is quite slow, taking about one week to compute – the usage map_rect on party level might speed up computation considerably. Also, faster computation with future *STAN* versions is conceivable, as the

model could benefit from faster indexing in *STAN* 2.20, for example. We could also try to re-parameterize the model or find better priors to speed up computation, however, the current parametrization was the best we could find after trying some different ones.

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 has not been studied in detail.