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

Marcus Groß

2019-30-07

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.

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

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

```r
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	GR.dc.NE	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 fivethirtyeight.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.

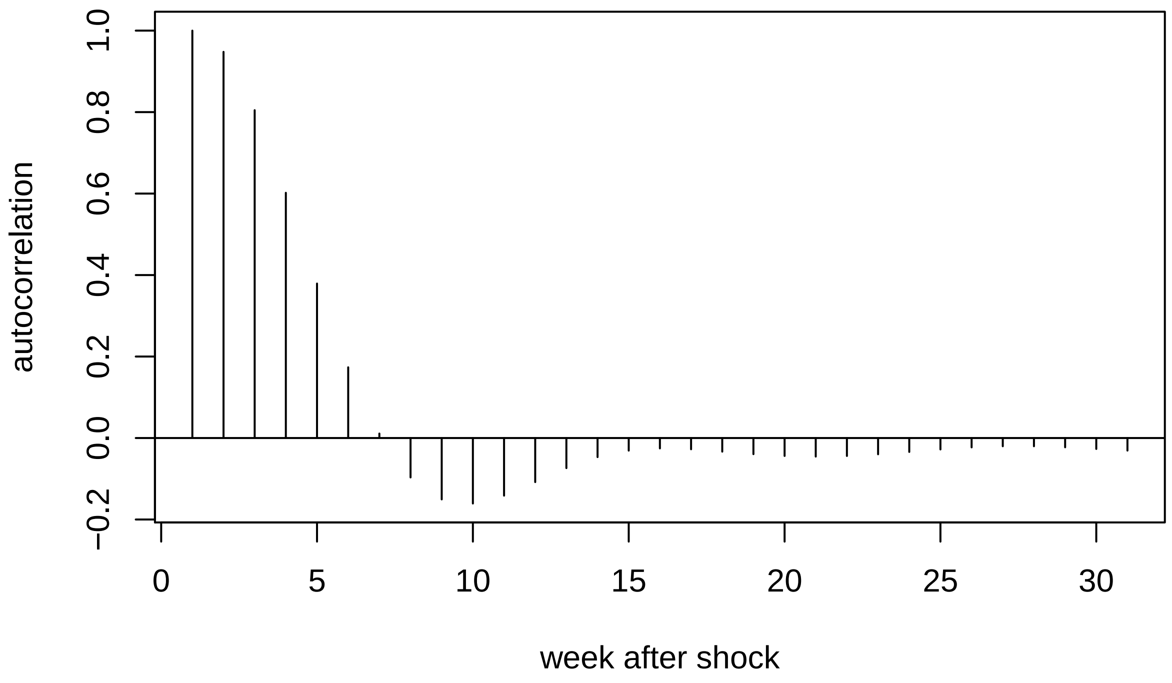
Other election forecasts, such as the US-presidential forecasts by Nate Silver (https://fivethirtyeight.com/features/a-users-guide-to-fivethirtyeights-2016-general-election-forecast/) and multi-party forecasts for the UK-election (http://www.electionforecast.co.uk) or for the german federal electin (http://zweitstimme.org) 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

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 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 completely. 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):

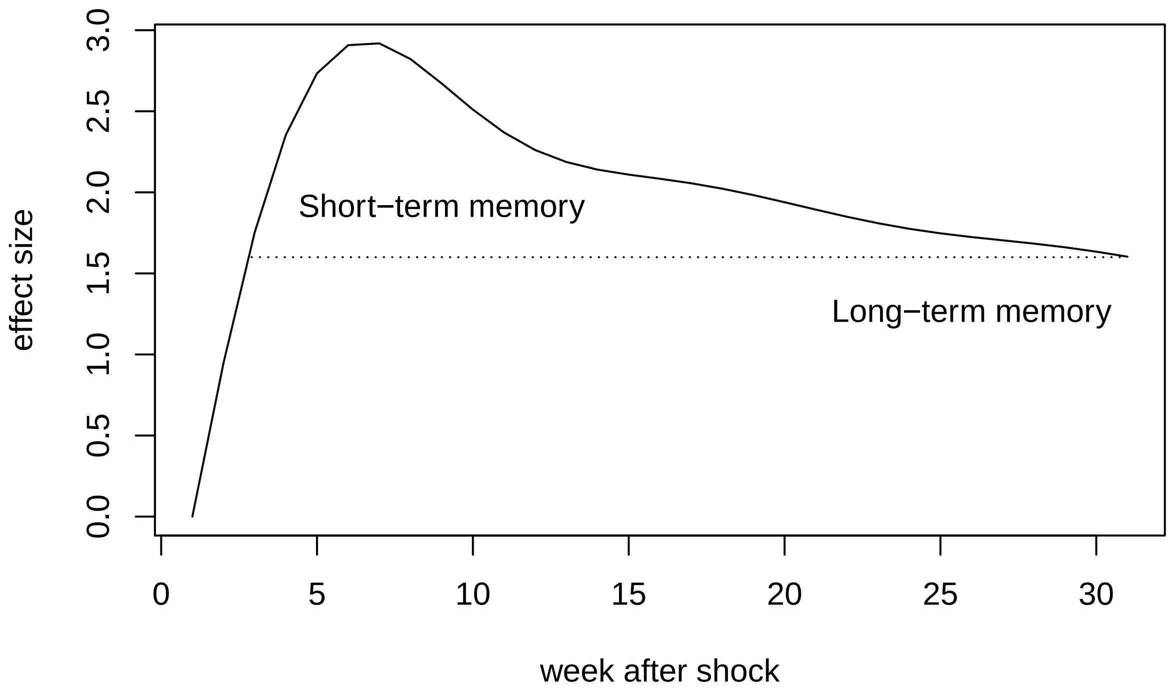
```r
pollDataShock <- pollData %>% arrange(desc(Datum)) %>% na.locf(na.rm = FALSE) %>%
  na.locf(fromLast = TRUE)
dateSeq <- seq(min(pollDataShock$Datum), max(pollDataShock$Datum) , by = "week")
smoothProportions <- sapply(colnames(pollDataShock)[3:7],
                            function(y){
      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

Model

Let $y_{party,t}$ be the true voter's share for a specific party 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. Poll data is published in irregular intervals by different institutes or pollsters.

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

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

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

$$\epsilon_{\text{poll}_{pt,t}} = 0.98 \cdot \epsilon_{\text{poll}_{pt,t-1}}$$

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

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

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

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

Prepare Data

```r
# 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)

#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
```

```r
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 parrallelization 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,
#                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")
```

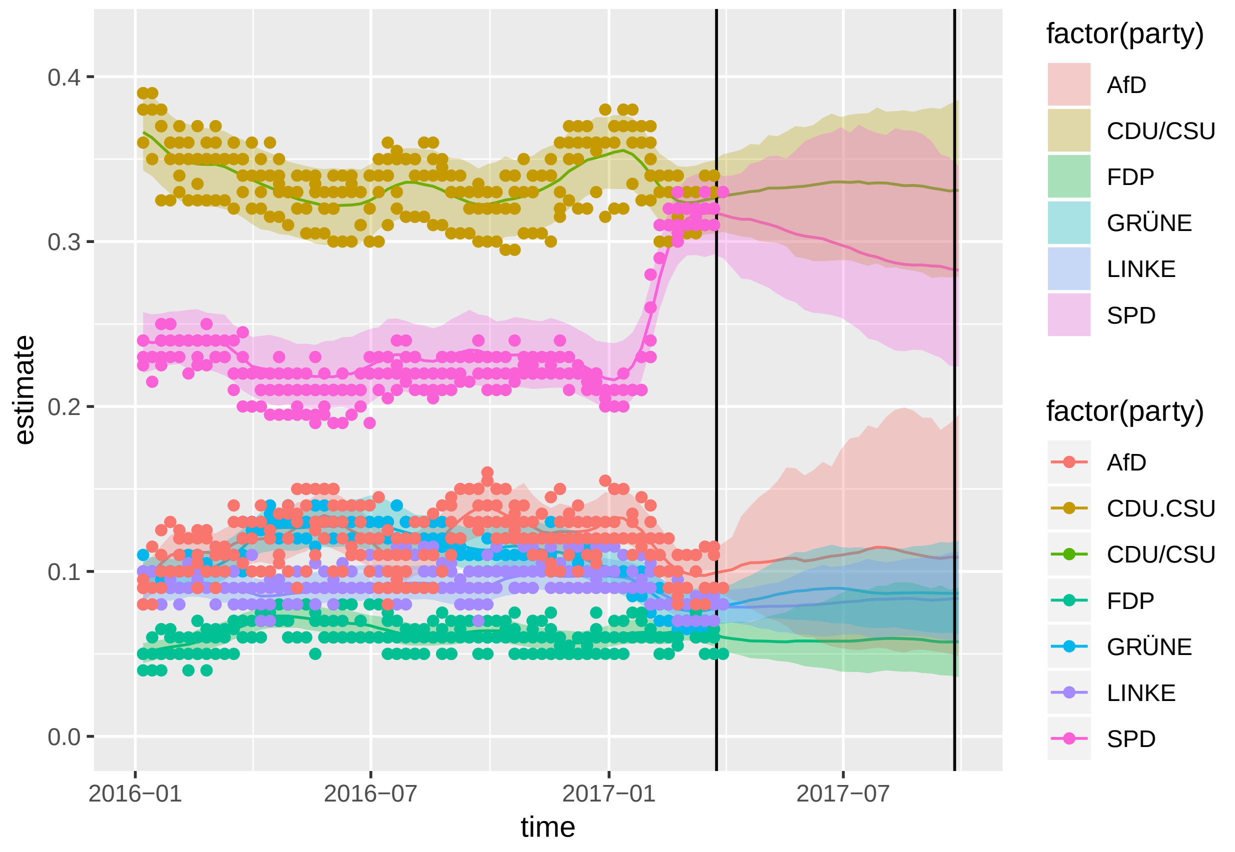
Results

```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 = rownames(pollData)[x])
}) %>% bind_rows()

plotPollData <- data.frame(t(pollData) %>% logistic,
                           time = as.POSIXct(timeSeq[matchedDates]*60*60*24*7,
                                             origin = "1970-01-01"))

plotPollData <- plotPollData %>% as_tibble %>% gather(key = "party",
                                                      value = "proportion", -time)

ggplot(data = plotData, aes(x = time, y = estimate, group = party,
                            colour = factor(party))) + geom_line() +
  geom_ribbon(aes(ymin = lower, ymax = upper,
                  fill = factor(party)),alpha = 0.3, colour = NA) +
  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)) +
  geom_point(data = plotPollData, aes(x = time, y = proportion, group = party))
```



##Troubleshooting

##Outlook