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.

Introduction

Data

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.

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

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

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

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

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

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

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

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

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

```
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

Prepare Data

```
# combine election and poll data
partyNames <- c("CDU/CSU", "SPD", "GRÜNE", "FDP", "LINKE", "AfD")
```

```r
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 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")
```

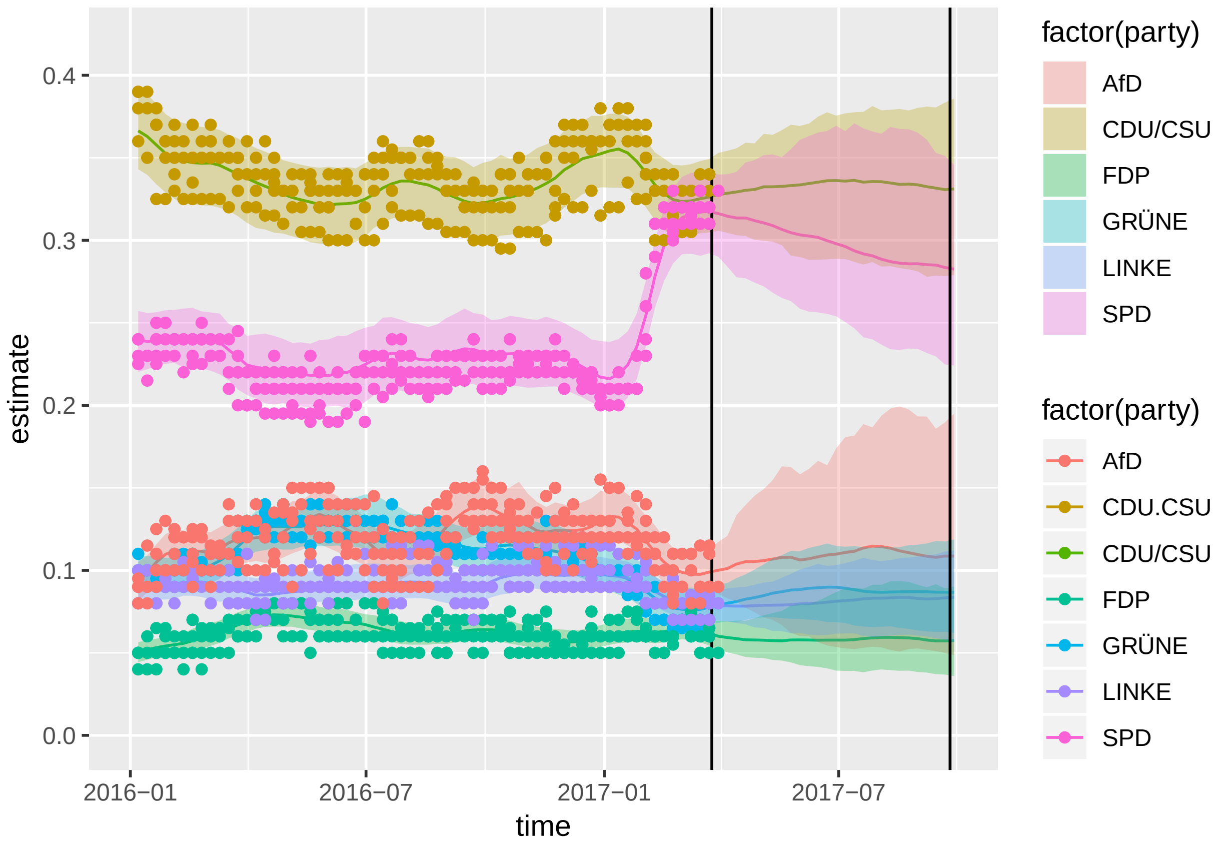
Results

```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