Replication of Figures, Tables and Numbers. The Zweitstimme Model: A Dynamic Forecast of the 2021 German Federal Election

Marcel Neunhoeffer, Thomas Gschwend, Klara Müller,
Simon Munzert & Lukas F. Stoetzer

21 June 2021

Introduction

This file replicates all the figures, tables and numbers in "The Zweitstimme Model: A Dynamic Forecast of the 2021 German Federal Election" (Gschwend et al. 2021). As re-running all the models can be time consuming, this file takes the MCMC draws (as of 100 days prior to the election) in the dataverse and reproduces the Figures and Numbers in the main text.

If you want to re-run our models you find all necessary data and code in our github repository. You first will have to re-run the pre-training of the structural model by running `code/R/01_ger_structural_pre_train_stan.R`. Then you can run the combined model by running `code/R/02_ger_structural_pre_train_stan.R`.

The Stan code for the models can be found in `code/model_code`. All the data we used to estimate our models is stored in `data/ger` for Germany, or will be scraped from the polling website wahlrecht.de along the way.

R Environment

This code was last tested and run on 21 June 2021.

platform	x86_64-pc-linux-gnu
arch	x86_64
os	linux-gnu
system	x86_64, linux-gnu
status	
major	4
minor	0.4
year	2021
month	02
day	15
svn rev	80002
language	R
version.string	R version 4.0.4 (2021-02-15)
nickname	Lost Library Book
RStudio	1.4.1103

Dataverse Structure

Dataverse

- district_prediction - Contains all necessary code and data to replicate the district predictions
  - figures
  - processed-data
  - raw-data
  - scripts
- ger_2017 - Contains forecasts for the 2017 election for several cutoffs to calculate the rmse
- zweitstimme_output_100.RDS - Output from our model 100 days prior to the election
- 2021_structural_pre_train_stan.RDS - Contains draws from the structural dirichlet regression model
- pre_train_data_21.RDS - Contains all data for the structural model
- zweitstimme_replication_PS.Rmd - This file, reproduces everything in the article

Questions

If you have any further questions or encounter issues while replicating the results please let us know via email to Marcel Neunhoeffer (marcel.neunhoeffer@stat.uni-muenchen.de).

Reproduction of Table 1

In table 1 we present an evaluation of the zweitstimme model applied to the 2017 election in Germany.

To that end we first load the forecasts at various cutoffs before the election 2017 (starting 148 days prior to the election) and calculate the root mean squared error (RMSE) across all parties at each cutoff.

```
# Set election year
Election <- 2017

# Set cutoffs (only those with existing files will work)
cutoffs <- c(2, 8, 36, 64, 92, 116, 148)

# Create an empty object to collect processed results
df_forecast <- NULL

plot_names <- c("CDU_CSU", "SPD", "Linke", "Grüne", "FDP", "AfD", "Andere")

for (cutoff in cutoffs) {
  # Look up file
  file <- list.files(
    path = "ger_2017",
    pattern = paste0("draws*.*_", Election, "_", cutoff, ".RDS"),
    full.names = T
  )

  # Read file for cutoff
  df <- readRDS(file)

  # Adjust order (just a convenience)
  adjustOrder <-
    match(c("cdu", "spd", "lin", "gru", "fdp", "afd", "oth"),
          df$party_names)

  forecast <- df$forecast[, adjustOrder]

  # Calculate means and quantiles
  tmp_forecast <- data.frame(
    y = apply(forecast, 2, mean),
```

```
    ci = t(apply(forecast, 2, function(x)
      quantile(x, c(
        1 / 12, 11 / 12
      )))),
    ci95 = t(apply(forecast, 2, function(x)
      quantile(x, c(
        0.025, 0.975
      ))))
  )


  # Add some names and auxiliary values for plotting
  rownames(tmp_forecast) <- plot_names
  colnames(tmp_forecast) <-
    c("value", "low", "high", "low95", "high95")

  tmp_forecast <- round(tmp_forecast * 100, 1)

  tmp_forecast$name <- c("CDU/CSU", plot_names[2:7])
  tmp_forecast$name_eng <-
    c("CDU/CSU", "SPD", "Left", "Greens", "FDP", "AfD", "Others")
  tmp_forecast$t <- cutoff
  tmp_forecast$y <- tmp_forecast$value
  tmp_forecast$x <- seq(0, 6 , 1)

  df_forecast <- rbind(df_forecast, tmp_forecast)
}

# In addition we need the election results
election_res <- c(32.9,20.5, 9.2, 8.9, 10.7, 12.6, 5)

# Convenience function to calculate the RMSE
rmse <- function(error) {
  sqrt(mean(error ^ 2))
}

# Calculate the errors per party at each cutoff
errors <- df_forecast$value - rep(election_res, 7)

# Calculate RMSE and collect it in a table
rmse_res <-
  sapply(unique(df_forecast$t), function(x)
    round(rmse(errors[df_forecast$t == x]), 1))

table_1 <- cbind(unique(df_forecast$t), rmse_res)

colnames(table_1) <- c("Lead time in days", "RMSE")

knitr::kable(table_1)
```

Lead time in days	RMSE
2	1.9
8	2.1

Lead time in days	RMSE
36	3.3
64	3.4
92	3.3
116	3.4
148	4.3

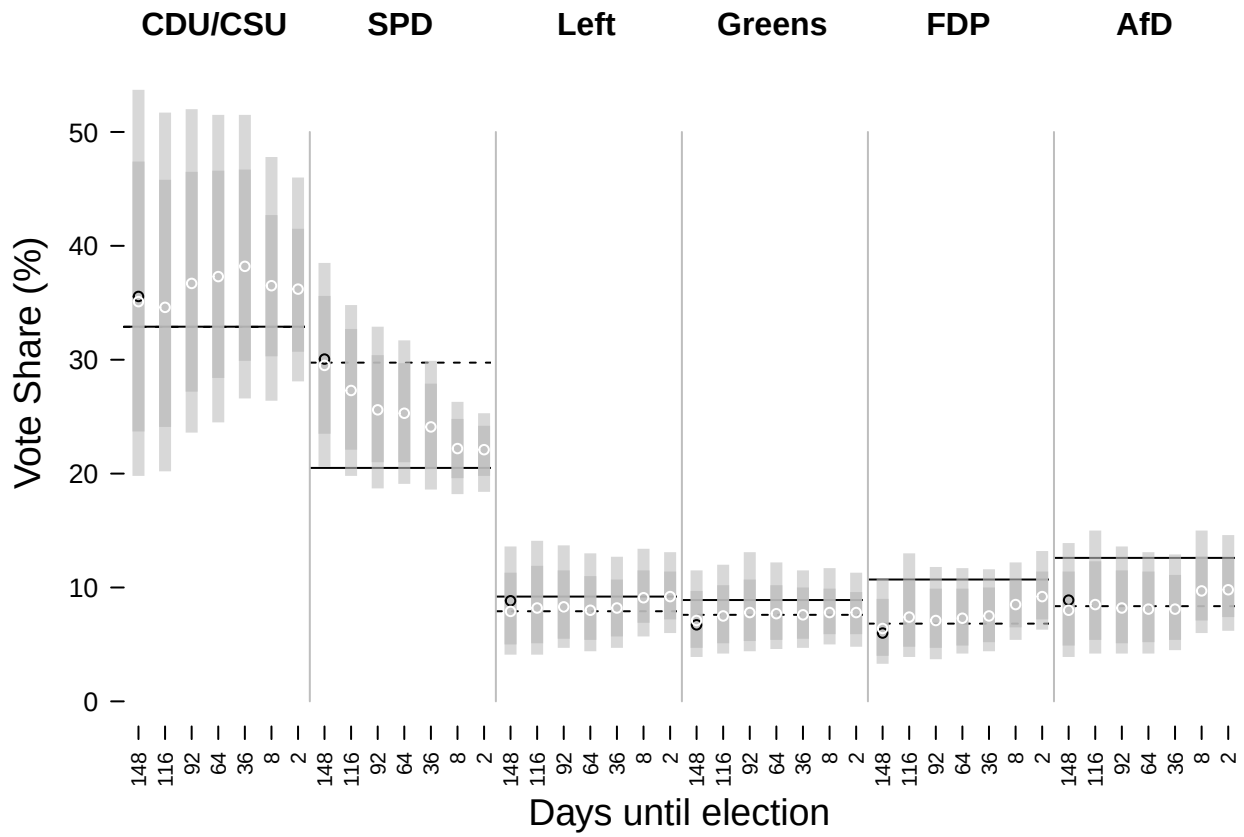
Reproduction of Figure A2

In addition to table 1 we present Figure A2 in the Supplementary Material.

```
avg_polls <- matrix(NA, nrow = length(cutoffs), ncol = 7)
df_polls <- df$polls
for (i in 1:length(cutoffs)) {
  avg_polls[i, ] <-
    apply(df_polls[(df_polls$days_to_election >= cutoffs[i] &
                      df_polls$days_to_election < cutoffs[i] + 14), ][, 6:ncol(df_polls)], 2, mean)
}

colnames(avg_polls) <- names(df_polls[6:ncol(df_polls)])
rownames(avg_polls) <- cutoffs

adjustOrder <-
  match(c("cdu", "spd", "lin", "gru", "fdp", "afd", "oth"),
        colnames(avg_polls))

avg_polls <- avg_polls[, adjustOrder]

struct_forecast <-
  readRDS(paste0(
    "ger_2017/",
    Election,
    "_structural_forecast.RDS"
  ))


adjustOrder <-
  match(c("cdu", "spd", "lin", "gru", "fdp", "afd", "oth"),
        colnames(struct_forecast))


struct_forecast <- struct_forecast[, adjustOrder]

parties <- unique(df_forecast$x)


par(mar = c(0, 0, 2, 0) + .1)
par(oma = c(5, 5, 0, 0) + .1)
layout(matrix(c(1, 2, 3, 4, 5, 6), 1, 6, byrow = TRUE))

for (i in 1:6) {
  sel <- parties[i]
```

```
plot(
  x = 7:1,
  y = df_forecast$value[df_forecast$x == sel],
  ylim = c(0, 55),
  type = "n",
  xlim = c(0.7, 7),
  yaxt = "n",
  xaxt = "n",
  ylab = "",
  xlab = "",
  bty = "n",
  las = 1
)
abline(h = election_res[i])
abline(h = apply(struct_forecast, 2, mean)[i] * 100, lty = "dashed")
segments(
  x0 = 7:1,
  y0 = df_forecast$low[df_forecast$x == sel],
  y1 = df_forecast$high[df_forecast$x == sel],
  col = adjustcolor("grey", alpha = 0.8),
  lwd = 6,
  lend = 1
)
segments(
  x0 = 7:1,
  y0 = df_forecast$low95[df_forecast$x == sel],
  y1 = df_forecast$high95[df_forecast$x == sel],
  col = adjustcolor("grey", alpha = 0.6),
  lwd = 6,
  lend = 1
)

points(x = 7:1, y = avg_polls[, i], col = "black")
points(x = 7:1,
       y = df_forecast$value[df_forecast$x == sel],
       col = "white")
title(paste(df_forecast$name_eng[df_forecast$x == sel][1]), cex.main = 1.5)
#if(i > 3)
axis(
  1,
  at = 7:1 ,
  labels = paste(cutoffs),
  col = NA,
  col.ticks = 1,
  las = 2
)
#if(i == 1 | i == 4)
if (i == 1)
  axis(
    2,
    col = NA,
    col.ticks = 1,
    las = 1,
```

```
      cex.axis = 1.2
    )
  if (i > 1)
    axis(
      2,
      col = "grey",
      col.ticks = NA,
      las = 1,
      labels = NA
    )
}

mtext("Days until election", 1, 3, outer = TRUE, cex = 1.2)
mtext("Vote Share (%)", 2, 3, outer = TRUE, cex = 1.2)
```



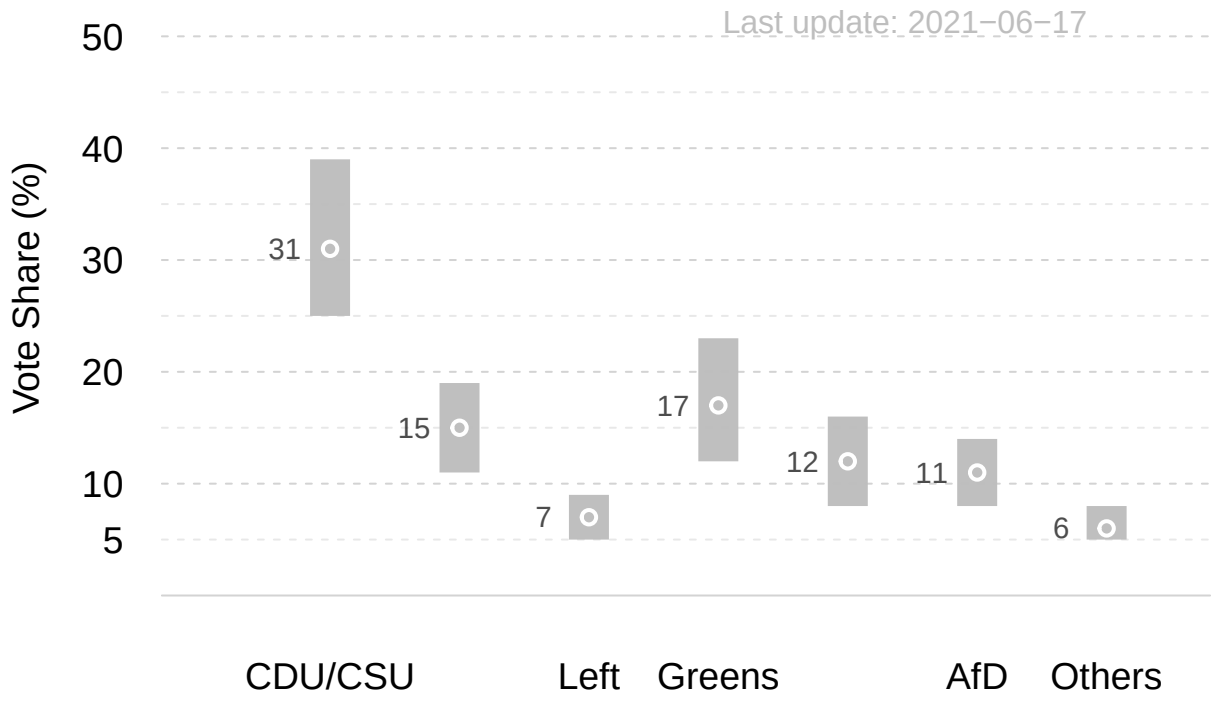
Reproduction of Figure 1

```
# Load forecast as of 100 days prior to the 2021 election
zweitstimme_output <- readRDS("zweitstimme_output_100.RDS")

# Pre-process the draws to get the mean, 5/6 CI and 95% CI
df_forecast <-
  data.frame(
    y = apply(zweitstimme_output$forecast, 2,  mean),
    ci = t(apply(zweitstimme_output$forecast, 2, function(x)
```

```
      quantile(x, c(1 / 12, 11 / 12)))),
    ci95 = t(apply(zweitstimme_output$forecast, 2, function(x)
      quantile(x, c(0.025, 0.975))))
  )
rownames(df_forecast) <-
  c("CDU/CSU", "SPD", "Left", "Greens", "FDP", "AfD", "Others")
colnames(df_forecast) <-
  c("value", "low", "high", "low95", "high95")

# Multiply with 100 to get percent
df_forecast$value <- round(df_forecast$value * 100, 0)

df_forecast[, c(2, 4)] <- floor(df_forecast[, c(2, 4)] * 100)
df_forecast[, c(3, 5)] <- ceiling(df_forecast[, c(3, 5)] * 100)

df_forecast$y <- df_forecast$value
df_forecast$x <- seq(0, 6, 1)




# Define plot function for Figure 1

plot_zs <- function(df_forecast, means = T) {
  # Set up empty plot
  par(mar = c(5, 5, 0, 0) + .1)
  plot(
    x = c(1, 2, 3, 4, 5, 6, 7),
    y = df_forecast$value,
    col = "white",
    type = "n",
    bty = "n",
    ylim = c(0, 55),
    xlim = c(0, 7.5),
    xlab = "",
    ylab = "",
    yaxt = "n",
    xaxt = "n",
    cex.axis = 1.2
  )
  abline(h = c(10, 20, 30, 40, 50),
         lty = "dashed",
         col = "lightgrey")
  abline(h = c(0), lty = "solid", col = "lightgrey")
  abline(
    h = c(5, 15, 25, 35, 45),
    lty = "dashed",
    col = adjustcolor("lightgrey", alpha = 0.5)
  )



  # Now the 5/6 CI
```

```r
segments(
  x0 = c(1, 2, 3, 4, 5, 6, 7),
  y0 = df_forecast$low,
  y1 = df_forecast$high,
  lwd = 20,
  col = adjustcolor("grey", alpha = 0.99),
  lend = 1
)

if (means) {
  # Add the means
  points(
    x = c(1, 2, 3, 4, 5, 6, 7),
    y = df_forecast$value,
    col = "white",
    lwd = 2
  )
  # Add text labels for the mean forecast
  text(
    y = df_forecast$value,
    x = c(1, 2, 3, 4, 5, 6, 7) - 0.35,
    labels = df_forecast$value,
    cex = 0.9,
    col = adjustcolor("black", alpha = 0.7)
  )

} else {
  # Add text labels for the mean forecast
  text(
    y = df_forecast$value,
    x = c(1, 2, 3, 4, 5, 6, 7) - 0.35,
    labels = paste0(df_forecast$low, "-", df_forecast$high),
    cex = 0.9,
    col = adjustcolor("black", alpha = 0.7)
  )
}
# Add axis labels
axis(
  1,
  at = c(1, 2, 3, 4, 5, 6, 7),
  labels = rownames(df_forecast),
  las = 1,
  tick = 0,
  cex.axis = 1.2
)
axis(
  2,
  at = c(5, 10, 20, 30, 40, 50),
  labels = c(5, 10, 20, 30, 40, 50),
  las = 1,
  tick = 0,
  cex.axis = 1.2
)
```

```
  mtext("Vote Share (%)",
        side = 2,
        line = 3,
        cex = 1.2)
}


# Plot forecast
plot_zs(df_forecast)
text(7,
     51,
     paste0("Last update: ", as.Date(zweitstimme_output$timestamp) + 1),
     pos = 2,
     col = "gray")
```



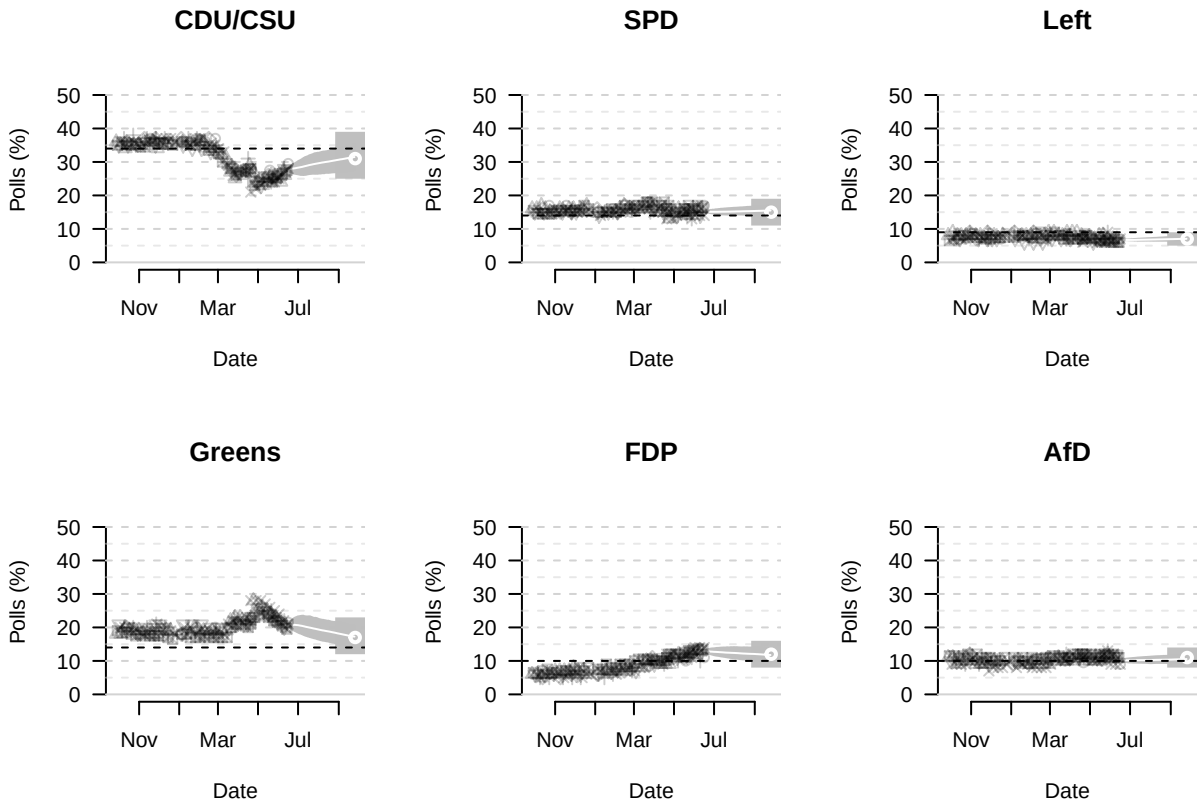
Reproduction of Figure 2

```
# Load Structural Data and structural forecast
data_structural <-
  readRDS("pre_train_data_21.RDS")

results <-
  readRDS(
    "2021_structural_pre_train_stan.RDS"
  )
```

```r
res <- as.matrix(results)


jags_matrix <- as.matrix(res)


structural_forecast <-
  jags_matrix[, grepl("y_mis\\[", colnames(jags_matrix))]
colnames(structural_forecast) <-
  data_structural$party[!complete.cases(data_structural$voteshare)]

plot_evo_pred <-
  function(zweitstimme_output,
           structural_forecast,
           party = "cdu",
           legend = F) {
    date_df <-
      lapply(zweitstimme_output$poll_aggregator, function(x)
        rbind(mean = colMeans(x), apply(x, 2, quantile, c(1 / 12, 11 / 12))))

    # Pre-process the draws to get the mean, 5/6 CI and 95% CI
    df_forecast <-
      data.frame(
        y = apply(zweitstimme_output$forecast, 2,  mean),
        ci = t(apply(zweitstimme_output$forecast, 2, function(x)
          quantile(x, c(1 / 12, 11 / 12)))),
        ci95 = t(apply(zweitstimme_output$forecast, 2, function(x)
          quantile(x, c(0.025, 0.975))))
      )
    rownames(df_forecast) <-
      c("CDU/CSU", "SPD", "Left", "Greens", "FDP", "AfD", "Others")
    colnames(df_forecast) <-
      c("value", "low", "high", "low95", "high95")

    # Multiply with 100 to get percent
    df_forecast$value <- round(df_forecast$value * 100, 0)

    df_forecast[, c(2, 4)] <- floor(df_forecast[, c(2, 4)] * 100)
    df_forecast[, c(3, 5)] <- ceiling(df_forecast[, c(3, 5)] * 100)

    df_forecast$y <- df_forecast$value
    df_forecast$x <- seq(0, 6, 1)


    party_labs <-
      cbind(
        c("cdu", "spd", "lin", "gru", "fdp", "afd"),
        c("CDU/CSU", "SPD", "Left", "Greens", "FDP", "AfD")
      )

    party_name <- party_labs[party_labs[, 1] == party, 2]
```

```r
plot(
  as.Date(names(zweitstimme_output$poll_aggregator)),
  sapply(zweitstimme_output$poll_aggregator, function(x)
    x[1, party]),
  type = "n",
  ylim = c(0, 55),
  bty = "n",
  las = 1,
  xlab = "Date",
  ylab = "Polls (%)",
  main = party_name
)

abline(h = c(10, 20, 30, 40, 50),
       lty = "dashed",
       col = "lightgrey")
abline(h = c(0), lty = "solid", col = "lightgrey")
abline(
  h = c(5, 15, 25, 35, 45),
  lty = "dashed",
  col = adjustcolor("lightgrey", alpha = 0.5)
)

polygon(
  c(as.Date(names(date_df)), rev(as.Date(names(
    date_df
  )))),
  c(sapply(date_df, function(x)
    x[2, party]) * 100, rev(sapply(date_df, function(x)
      x[3, party]) * 100)),
  col = adjustcolor("grey", 0.99),
  border = NA
)


segments(
  x0 = as.Date(names(date_df)[length(date_df)]),
  y0 = df_forecast$low[rownames(df_forecast) == party_name],
  y1 = df_forecast$high[rownames(df_forecast) == party_name],
  lwd = 20,
  col = adjustcolor("grey", alpha = 0.99),
  lend = 1
)



lines(as.Date(names(date_df)[-length(names(date_df))]), sapply(date_df, function(x)
  x[1, party])[-length(names(date_df))] * 100, col = "white")

points(
  x = as.Date(names(date_df)[length(date_df)]),
  y = df_forecast$value[rownames(df_forecast) == party_name],
  col = "white",
```

```
      lwd = 2
    )


    # Add the actual polls
    points(
      as.Date(zweitstimme_output$polls$date),
      zweitstimme_output$polls[, party],
      pch = zweitstimme_output$polls$iid,
      col = adjustcolor("black", 0.2)
    )

    ### Fundamental prediction
    fundamental_predictions <-
      round(apply(structural_forecast, 2, mean) * 100, 0)

    abline(h = fundamental_predictions[party], lty = "dashed")


    if (legend) {
      legend(
        "bottomleft",
        legend = c(
          "Allensbach",
          "Emnid",
          "Forschungsgruppe Wahlen",
          "Forsa",
          "GMS",
          "Infratest Dimap",
          "Insa"
        ),
        pch = 1:7,
        bty = "n",
        cex = 1.2
      )
    }
  }

par(mfrow = c(2, 3))
plot_evo_pred(zweitstimme_output,
              structural_forecast,
              party = "cdu",
              legend = F)
plot_evo_pred(zweitstimme_output, structural_forecast, party = "spd")
plot_evo_pred(zweitstimme_output, structural_forecast, party = "lin")
plot_evo_pred(zweitstimme_output, structural_forecast, party = "gru")
plot_evo_pred(zweitstimme_output, structural_forecast, party = "fdp")
plot_evo_pred(zweitstimme_output, structural_forecast, party = "afd")
```



Reproduce Forecast of party vote shares in text

The numbers in the text are based on the same data frame as Figure 1.

```
knitr::kable(df_forecast[,1:3])
```

	value	low	high
CDU/CSU	31	25	39
SPD	15	11	19
Left	7	5	9
Greens	17	12	23
FDP	12	8	16
AfD	11	8	14
Others	6	5	8

Coalition Probabilities in text

```
# Get majority threshold for each simulation
maj <- 0.5-zweitstimme_output$forecast[,7]/2

# Define interesting coalitions
coalitions <- list(1, # absolute majority CDU/CSU
                   c(1, 2), # CDU/CSU-SPD
                   c(1, 4), # CDU/CSU-Greens
                   c(1, 5), # CDU/CSU-FDP
```

```
                c(2, 4), # SPD-Greens
                c(2, 3, 4), # SPD-Left-Greens
                c(2, 4,  5), #SPD-Greens-FDP
                c(1, 4, 5), # CDU/CSU-Greens-FDP
                c(1, 2, 5) # CDU/CSU-SPD-FDP
                )


# Calculate whether a coalition has a majority in parliament
coa_maj <-
  sapply(
    coalitions,
    FUN = function(x)
      rowSums(zweitstimme_output$forecast[, x, drop = F]) > maj
  )

coa_probs <- colMeans(coa_maj)

names(coa_probs) <-
  c("CDU/CSU",
    "CDU/CSU-SPD",
    "CDU/CSU-Greens",
    "CDU/CSU-FDP",
    "SPD-Greens",
    "SPD-Left-Greens",
    "SPD-Greens-FDP",
    "CDU/CSU-Greens-FDP",
    "CDU/CSU-SPD-FDP")

round(coa_probs * 100, 0)
```

```
##            CDU/CSU        CDU/CSU-SPD     CDU/CSU-Greens        CDU/CSU-FDP
##                  1                 47                 67                 20
##         SPD-Greens    SPD-Left-Greens     SPD-Greens-FDP CDU/CSU-Greens-FDP
##                  0                  6                 29                100
##    CDU/CSU-SPD-FDP
##                100
```

Chancellor Probability in text

We calculate the probability of the three main candidates to become chancellor based on these coalition probabilities. However, we exclude oversized coalitions (e.g. if the CDU/CSU-SPD coalition achieves a majority the probability for CDU/CSU-SPD-FDP is set to 0). We further assume that the biggest party in a coalition will appoint the chancellor and that all possible coalitions in one run of the simulation have equal formation probabilities (this is of course debatable).

```
# Set oversized coalitions to zero
for (i in 1:9000) {
  if (coa_maj[i, 1]) {
    coa_maj[i, c(2, 3, 4, 8, 9)] <- F
    next
  }
  if (coa_maj[i, 2] | coa_maj[i, 4]) {
    coa_maj[i, 9] <- F
```

```
  }
  if (coa_maj[i, 3] | coa_maj[i, 4]) {
    coa_maj[i, 8] <- F
  }
  if (coa_maj[i, 5]) {
    coa_maj[i, c(6, 7)] <- F
  }
}

# Check for the biggest party in each coalition
ranks <- t(apply(-zweitstimme_output$forecast, 1, rank))
coa_chancellor <- array(NA, dim = dim(coa_maj))

for (j in 1:9) {
  coa <- coalitions[[j]]

  chanc <- NULL
  for (i in 1:9000) {
    chanc <- c(chanc, names(which.min(ranks[i, coa])))
  }
  coa_chancellor[, j] <- chanc
}


chanc_probs <- list()

# Tabulate probabilities
for (i in 1:9000) {
  chanc_probs[[i]] <-
    table(coa_chancellor[i, coa_maj[i, ]]) / sum(table(coa_chancellor[i, coa_maj[i, ]]))
}



knitr::kable(data.frame(
  candidate = c("Laschet", "Baerbock", "Scholz"),
  prob = c(round(sum(unlist(
    lapply(chanc_probs, function(x)
      x[names(x) == "cdu"])
  )) / 90, 0),
  round(sum(unlist(
    lapply(chanc_probs, function(x)
      x[names(x) == "gru"])
  )) / 90, 0),

  round(sum(unlist(
    lapply(chanc_probs, function(x)
      x[names(x) == "spd"])
  )) / 90, 0))
))
```

candidate	prob
Laschet	88

candidate	prob
Baerbock	8
Scholz	3

Reproduction of district forecasts and seat distribution

More information on the neural net and the assumptions for our district predictions can be found in the Supplementary Material. Please also directly refer to the code in the folder `district_prediction`.

Please note that if you want to re-run the models, the necessary file with all data on all candidates since the 1983 election (`btw_candidates_1983-2017.csv`) is not available in the dataverse repository due to privacy considerations. If you want to replicate the model, please contact us and we can share the data with you.

Even without re-running the models this might take some time...

```
# Set true if you want to run all models.
# If false it will use our stored models.
replicate <- F
source("district_prediction/scripts/district_predictions.R")
```