Supplementary Information (SI) for article ‘Predicting Network Events to Assess Goodness of Fit of Relational Event Models’

Laurence Brandenberger

2018

Contents

R Specifications

```
rm(list = ls())
library(rem)
library(survival)
library(texreg)
library(ggplot2)
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
```

```
## [1] de_CH.UTF-8/de_CH.UTF-8/de_CH.UTF-8/C/de_CH.UTF-8/de_CH.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_2.2.1   texreg_1.36.23  survival_2.41-3 rem_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.16      knitr_1.17        magrittr_1.5
##  [4] splines_3.4.3     munsell_0.4.3     doParallel_1.0.10
##  [7] colorspace_1.3-2  lattice_0.20-35   rlang_0.1.2
## [10] foreach_1.4.3     plyr_1.8.4        stringr_1.3.0
## [13] tools_3.4.3       parallel_3.4.3    grid_3.4.3
## [16] gtable_0.2.0      htmltools_0.3.6   iterators_1.0.8
## [19] lazyeval_0.2.1    yaml_2.1.14       rprojroot_1.2
## [22] digest_0.6.15     tibble_1.3.4      Matrix_1.2-12
## [25] codetools_0.2-15  evaluate_0.10.1   rmarkdown_1.6
## [28] stringi_1.1.7     compiler_3.4.3    scales_0.5.0
## [31] backports_1.1.1
```

Additional Information on the Calculations in Figure 2 in the Article

First the artificial data is prepared:

```
sender <- c("i1", "i2", "i2", "i1", "i1")
target <- c("j1", "j1", "j2", "j1", "j2")
time <- c(1, 2, 2, 3, 4)
dt <- data.frame(sender, target, time)
dt
```

```
##   sender target time
## 1     i1     j1    1
## 2     i2     j1    2
## 3     i2     j2    2
## 4     i1     j1    3
## 5     i1     j2    4
```

And the risk set is defined:

```
# risk set: possible data
senderp <- c(rep("i1", 2), rep("i2", 2))
targetp <- c(rep(c("j1", "j2"),2))
start <- rep(1, 4)
end <- rep(4, 4)
att <- rep("1", 4)
dtp <- data.frame(senderp, targetp, start, end, att)
dtp
```

```
##   senderp targetp start end att
## 1      i1      j1     1   4   1
## 2      i1      j2     1   4   1
## 3      i2      j1     1   4   1
## 4      i2      j2     1   4   1
```

Next, the counting process data is set up and endogenous network statistics are calculated using the `rem`-package (Brandenberger 2018).

```
# load rem package and create counting process data
library(rem)
dtl <- createRemDataset(dt, dt$sender, dt$target, dt$time,
                        includeAllPossibleEvents = TRUE,
                        possibleEvents = dtp)
dtl
```

```
##    senderp targetp start end att eventTime eventdummy
## 1       i1      j1     1   4   1         1          1
## 2       i1      j2     1   4   1         1          0
## 3       i2      j1     1   4   1         1          0
## 4       i2      j2     1   4   1         1          0
## 5       i1      j1     1   4   1         2          0
## 6       i1      j2     1   4   1         2          0
## 7       i2      j1     1   4   1         2          1
## 8       i2      j2     1   4   1         2          1
## 9       i1      j1     1   4   1         3          1
## 10      i1      j2     1   4   1         3          0
## 11      i2      j1     1   4   1         3          0
## 12      i2      j2     1   4   1         3          0
## 13      i1      j1     1   4   1         4          0
## 14      i1      j2     1   4   1         4          1
## 15      i2      j1     1   4   1         4          0
## 16      i2      j2     1   4   1         4          0
```

```
dtl$activity <- degreeStat(dtl, dtl$eventTime, dtl$sender, 1,
                           eventvar = dtl$eventdummy)
dtl$popularity <- degreeStat(dtl, dtl$eventTime, dtl$target, 1,
                             eventvar = dtl$eventdummy)
tail(dtl)
```

```
##    senderp targetp start end att eventTime eventdummy  activity popularity
## 11      i2      j1     1   4   1         3          0 0.6931472  0.5198604
## 12      i2      j2     1   4   1         3          0 0.6931472  0.3465736
## 13      i1      j1     1   4   1         4          0 0.4332170  0.6065038
## 14      i1      j2     1   4   1         4          1 0.4332170  0.1732868
## 15      i2      j1     1   4   1         4          0 0.3465736  0.6065038
## 16      i2      j2     1   4   1         4          0 0.3465736  0.1732868
```

Artificial coefficients are set:

```
# set artificial coefficients
coef <- c(activity = 2.1, popularity = 1.4)
```

The baseline hazard is calculated for stratum $t = 4$

```
# calculate baseline hazard for strata t=5
strat4 <- subset(dtl, dtl$eventTime == 4)
nevents <- 1
baselhaz_newstrat_all <- exp(cbind(strat4$activity, strat4$popularity) %*% coef)
baselhaz_newstrat <- nevents/sum(baselhaz_newstrat_all)
# Illustration: calculate the baseline hazard by hand:
#i.e., activity * coefficient + popularity * coefficient for the first event
1/sum(exp(strat4$activity[1] * 2.1 + (strat4$popularity[1] * 1.4)),
      exp(strat4$activity[2] * 2.1 + (strat4$popularity[2] * 1.4)),
      exp(strat4$activity[3] * 2.1 + (strat4$popularity[3] * 1.4)),
      exp(strat4$activity[4] * 2.1 + (strat4$popularity[4] * 1.4)))
```

```
## [1] 0.06078858
```

```
# calculate survivor probabilty
strat4$survpr <- exp(-(baselhaz_newstrat))^(as.numeric(exp(cbind(strat4$activity,
                                                                 strat4$popularity) %*% coef)))
strat4
```

```
##    senderp targetp start end att eventTime eventdummy  activity popularity
## 13      i1      j1     1   4   1         4          0 0.4332170  0.6065038
## 14      i1      j2     1   4   1         4          1 0.4332170  0.1732868
## 15      i2      j1     1   4   1         4          0 0.3465736  0.6065038
## 16      i2      j2     1   4   1         4          0 0.3465736  0.1732868
##       survpr
## 13 0.7026281
## 14 0.8249478
## 15 0.7451161
## 16 0.8517842
```

Additional Information on the Simulation Test

The simulation test for the prediction procedure took the following steps:

1. Artificial data is prepared
2. Predictions are made
3. REMs are calculated over the newly predicted sequence and coefficients stored
4. Results are plotted

Step 1: Preparation of Artificial Data

```
# create senders and targets
nsenders <- 20
ntargets <- 30
allSenders <- paste0("IDSender", 1:nsenders)
allTargets <- paste0("IDTarget", 1:ntargets)

# create strata-data-set
for(i in 1:length(allSenders)){
  if(i == 1){
    dtstrata <- data.frame('sender' = rep(allSenders[i], length(allTargets)),
                           'target' = allTargets)
  }else{
    dtstrata <- rbind(dtstrata, data.frame('sender' = rep(allSenders[i],
                                                          length(allTargets)),
                                           'target' = allTargets))
  }
}

# create a data frame with first 50 events:
```

```r
for(i in 1:50){
  if(i == 1){
    dt <- cbind(dtstrata, data.frame('eventTime' = rep(i, nrow(dtstrata))))
  }else{
    dt <- rbind(dt, cbind(dtstrata,
                          data.frame('eventTime' = rep(i, nrow(dtstrata)))))
  }
}

# now create random 0/1 event-dummies
set.seed(2558)
dt$eventDummy <- sample(c(0,1), 30000, replace = TRUE,
                        prob = c((30000-200)/30000, 200/30000))

# create artificial model
coefficients <- c('inertia_hl20' = 2.9,
                  'senderActivity_hl20' = 2.4,
                  'targetPopularity_hl20' = 1.9,
                  'fourCycleStat_hl20' = 3.9)

# add endogenous nw stats (empty) to dt and to strata (so it can be filled
# later on)
dt$inertia_hl20 <- NA
dt$senderActivity_hl20 <- NA
dt$targetPopularity_hl20 <- NA
dt$fourCycleStat_hl20 <- NA
dtstrata$inertia_hl20 <- NA
dtstrata$senderActivity_hl20 <- NA
dtstrata$targetPopularity_hl20 <- NA
dtstrata$fourCycleStat_hl20 <- NA
dt$survpr <- NA
dtstrata$survpr <- NA

# Create a data set with only events:
dtEventOnly <- subset(dt, dt$eventDummy == 1)
```

Additionally a cpp-Function used in the `rem`-package is loaded as well via the `Rcpp`-package

```r
# generate a RCPP-function that speeds up calculation of weights
cppFunction('
  double weightTimesSummationCpp(
    NumericVector pastSenderTimes,
    double xlog,
    double currentTime,
    NumericVector weightvar) {

    double totalWeight = 0.0;
    double weight = 0.0;
    double result = 0.0;

    // for each bill that the current sender has cosponsored in past
    for (int j = 0; j < pastSenderTimes.size(); j++){
      weight = weightvar[j] * exp( - ( currentTime - pastSenderTimes[j] ) * xlog)  * xlog;
      totalWeight = totalWeight + weight ;
    }// close j-loop

    result = totalWeight;
    return result;
  }')
```

Step 2: Predicting from Random Events

A for-loop is used for the simulation process.

```
# specify the number of events as well as a partial calculation of the
# exponential decay function outside the loop:
nevents <- 10 # each strata will have 4 events
xlog50 <- log(2)/50
temp <- NULL
```

```
# Run the loop:
# for the next 49 events, predict new events based on coefficients provided
pb <- txtProgressBar(min = 51, max = 2000, style = 3)
for (i in 51:2000){
  # update progress bar
  setTxtProgressBar(pb, i)

  # 1) create new strata with new eventTime
  newstrat <- cbind(dtstrata, data.frame('eventTime' = i))

  # 2) calculate REM-statistics for new strata
  for(k in 1:nrow(newstrat)){
    # a) calculate inertia
    pastEventsInertia <-
      dtEventOnly$eventTime[dtEventOnly$sender == newstrat$sender[k] &
                            dtEventOnly$target == newstrat$target[k] &
                            dtEventOnly$eventTime < i]
    if(length(pastEventsInertia) == 0){
      newstrat$inertia_hl20[k] <- 0
    }else{
      newstrat$inertia_hl20[k] <-
        weightTimesSummationCpp(pastEventsInertia, xlog50, i,
                                rep(1, length(pastEventsInertia)))
    }
    # b) calculate sender activity
    pastEventsActivity <-
      dtEventOnly$eventTime[dtEventOnly$sender == newstrat$sender[k] &
                            dtEventOnly$target != newstrat$target[k] &
                            dtEventOnly$eventTime < i]
    if(length(pastEventsActivity) == 0){
      newstrat$senderActivity_hl20[k] <- 0
    }else{
      newstrat$senderActivity_hl20[k] <-
        weightTimesSummationCpp(pastEventsActivity, xlog50, i,
                                rep(1, length(pastEventsActivity)))
    }
```

```r
  # c) calculate target popularity
  pastEventsPopularity <-
    dtEventOnly$eventTime[dtEventOnly$target == newstrat$target[k] &
                            dtEventOnly$eventTime < i]
  if(length(pastEventsPopularity) == 0){
    newstrat$targetPopularity_hl20[k] <- 0
  }else{
    newstrat$targetPopularity_hl20[k] <-
      weightTimesSummationCpp(pastEventsPopularity, xlog50, i,
                              rep(1, length(pastEventsPopularity)))
  }
}
# d) calculate four cycles using the rem-package function "fourCycleStat()"
dtPastEventsDataset <- data.frame(time = dtEventOnly$eventTime,
                                  sender = as.character(dtEventOnly$sender),
                                  target = as.character(dtEventOnly$target),
                                  weight = rep(1, nrow(dtEventOnly)),
                                  stringsAsFactors = FALSE)
newstrat$fourCycleStat_hl20 <- fourCycleStat(data = newstrat,
                                             time = newstrat$eventTime,
                                             sender = as.character(newstrat$sender),
                                             target = as.character(newstrat$target),
                                             halflife = 50,
                                             dataPastEvents = dtPastEventsDataset,
                                             inParallel = TRUE, cluster = 7)

# 3) set event rate and decide on nevents
# set exogenously here, could be set dynamically allowing different nevents
# over the simulation process

# 4) calculate baseline hazard for new strata
x <- cbind(newstrat$inertia_hl20, newstrat$senderActivity_hl20,
           newstrat$targetPopularity_hl20, newstrat$fourCycleStat_hl20)
baselhaz_newstrat_all <- exp(x %*% coefficients)
baselhaz_newstrat <- nevents/sum(baselhaz_newstrat_all)

# 5) calculate survivor probability for new strata
newstrat$survpr <- exp(-(baselhaz_newstrat))^(as.numeric(exp(x %*% coefficients)))

# 6) select nevents based on survivor probability (lower = more likely to become an event)
# Pick from lowest 10%-quantile
newstrat$eventDummy <-
  as.numeric(lapply(newstrat$survpr, function(x) sample(c(0, 1), 1,
                                                        prob = c(x, 1-x))))
# make sure there are no more than 10 events selected:
temp <- c(temp, sum(newstrat$eventDummy))
while(sum(newstrat$eventDummy) > nevents){
  rowsWithEvents <- as.numeric(row.names(newstrat[newstrat$eventDummy == 1,]))
  newstrat$eventDummy[sample(rowsWithEvents, 1)] <- 0
}

# 7) bind new strata to dataset (dt + dtEventOnly) and start next iteration in the loop
dt <- rbind(dt, newstrat)
```

```
  dtEventOnly <- rbind(dtEventOnly, newstrat[newstrat$eventDummy == 1,])
}
```

Step 3: Calculating REMs Using Newly Predicted Sequences and Storing Coefficients

An other for-loop is used to estimate the REM-coefficients over different portions of the simulated sequence.

```
pb <- txtProgressBar(min = 0, max = 2000, style = 3)
for(i in seq(50, 2000, by = 50)){
  setTxtProgressBar(pb, i)
  if(i == 50){
    dttemp <- subset(dt, dt$eventTime  < 51)
    fitloop <- clogit(eventDummy ~
                        inertia_hl20 +
                        senderActivity_hl20 +
                        targetPopularity_hl20 +
                        fourCycleStat_hl20 +
                        strata(eventTime),
                      data = dttemp)
    dtloop <- data.frame('eventStrata' = 51,
                         'coef_inertia' = as.numeric(coefficients(fitloop)[1]),
                         'coef_activity' = as.numeric(coefficients(fitloop)[2]),
                         'coef_popularity' = as.numeric(coefficients(fitloop)[3]),
                         'coef_fourcycle' = as.numeric(coefficients(fitloop)[4]),
                         'sig_inertia' = coef(summary(fitloop))[1,5],
                         'sig_activity' = coef(summary(fitloop))[2,5],
                         'sig_popularity' = coef(summary(fitloop))[3,5],
                         'sig_fourcycle' = coef(summary(fitloop))[4,5],
                         'BIC' = BIC(fitloop),
                         'R2' = 1- (fitloop$loglik[2]/fitloop$loglik[1])) ## save Mc-Faddon's R^2
  }else{
    dttemp <- subset(dt, dt$eventTime  < i)
    fitloop <- clogit(eventDummy ~
                        inertia_hl20 +
                        senderActivity_hl20 +
                        targetPopularity_hl20 +
                        fourCycleStat_hl20 +
                        strata(eventTime),
                      data = dttemp)
    dtloop <- rbind(dtloop,
                    data.frame('eventStrata' = i,
                               'coef_inertia' = as.numeric(coefficients(fitloop)[1]),
                               'coef_activity' = as.numeric(coefficients(fitloop)[2]),
                               'coef_popularity' = as.numeric(coefficients(fitloop)[3]),
                               'coef_fourcycle' = as.numeric(coefficients(fitloop)[4]),
                               'sig_inertia' = coef(summary(fitloop))[1,5],
                               'sig_activity' = coef(summary(fitloop))[2,5],
                               'sig_popularity' = coef(summary(fitloop))[3,5],
                               'sig_fourcycle' = coef(summary(fitloop))[4,5],
                               'BIC' = BIC(fitloop),
                               'R2' = 1- (fitloop$loglik[2]/fitloop$loglik[1]))) ## save Mc-Faddon's R^
  }
}
```

Step 4: Plotting Results

Plotting Figure 3 in the article:

```
ggplot(dtloop, aes(x = eventStrata))+
  geom_line(aes(y = coef_inertia,
                color = 'inertia', linetype="simulated"), size = 1) +
  geom_line(aes(y = coef_activity, color = 'activity', linetype="simulated"),
            size = 1) +
  geom_line(aes(y = coef_popularity, color = 'popularity', linetype="simulated"),
            size = 1)+
  geom_line(aes(y = coef_fourcycle, color = 'four-cycle', linetype="simulated"),
            size = 1) +
  geom_hline(aes(yintercept = 2.9 , color = "inertia", linetype="fixed"),
             size = .8, show.legend = TRUE) +
  geom_hline(aes(yintercept = 2.4 , color = "activity", linetype="fixed"),
             size = .8) +
  geom_hline(aes(yintercept = 1.9 , color = "popularity", linetype="fixed"),
             size = .8) +
  geom_hline(aes(yintercept = 3.9 , color = "four-cycle", linetype="fixed"),
             size = .8) +
  scale_color_manual('Network statistics',
                     values = rep(RColorBrewer::brewer.pal(4, 'Spectral'), 2)) +
  scale_linetype_manual('Coefficients', values = c('dotted', 'solid'))+
  publicationtheme +
  scale_x_continuous(breaks = c(seq(0, 2000, 250))) +
  xlab("Number of simulated stratas (=event days)") +
  ylab("Coefficient sizes (conditional logit)")
```

Additional Information on the Correlation Table 2 in the Article

The correlation table was generated using the policy-debate data presented as an example in the paper (Leifeld 2016). Strata 989 was selected and the baseline hazard was calculated for different number of events:

```
# divide by nr of events for this strata
baselinehaz989_10 <- 10/sum(dtstrata989$baselhaz)
baselinehaz989_9 <- 9/sum(dtstrata989$baselhaz)
baselinehaz989_8 <- 8/sum(dtstrata989$baselhaz)
baselinehaz989_7 <- 7/sum(dtstrata989$baselhaz)
baselinehaz989_6 <- 6/sum(dtstrata989$baselhaz)
baselinehaz989_5 <- 5/sum(dtstrata989$baselhaz)
baselinehaz989_4 <- 4/sum(dtstrata989$baselhaz)
baselinehaz989_3 <- 3/sum(dtstrata989$baselhaz)
baselinehaz989_2 <- 2/sum(dtstrata989$baselhaz)
baselinehaz989_1 <- 1/sum(dtstrata989$baselhaz)
```

Afterwards, survivor probabilities were calculated for each event in the strata using the newly created baseline hazards.

```
# calculate survivor probability
dtstrata989$survpr_10 <- exp(-(baselinehaz989_10))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_9 <- exp(-(baselinehaz989_9))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_8 <- exp(-(baselinehaz989_8))^
```

```
  (exp(dtstrata989$inertia.hl20.sd *  modelred$coefficients[[1]]))
dtstrata989$survpr_7 <- exp(-(baselinehaz989_7))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_6 <- exp(-(baselinehaz989_6))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_5 <- exp(-(baselinehaz989_5))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_4 <- exp(-(baselinehaz989_4))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_3 <- exp(-(baselinehaz989_3))^
  (exp(dtstrata989$inertia.hl20.sd *  modelred$coefficients[[1]]))
dtstrata989$survpr_2 <- exp(-(baselinehaz989_2))^
  (exp(dtstrata989$inertia.hl20.sd * modelred$coefficients[[1]]))
dtstrata989$survpr_1 <- exp(-(baselinehaz989_1))^
  (exp(dtstrata989$inertia.hl20.sd *  modelred$coefficients[[1]]))
```

The `modelred`-object refers to the stratified Cox regression using only inertia as explanatory variable. The correlation table was generated by correlating all survivor probabilities with different specifications of number of events:

```
cor(cbind(dtstrata989$survpr_10, dtstrata989$survpr_9, dtstrata989$survpr_8,
          dtstrata989$survpr_7, dtstrata989$survpr_6, dtstrata989$survpr_5,
          dtstrata989$survpr_4, dtstrata989$survpr_3, dtstrata989$survpr_2,
          dtstrata989$survpr_1))
```

Additional Information on the Data used in the Illustrative Application

The data used in the illustrative application in Section 3 in the article is based on a discourse analysis of the German pension financing debate between 1993 and 2001. For additional information on the data, please refer to Leifeld (2016).

Figure 1 shows a barplot of the rate of the debate over the full course of the debate. The number of events refer to the number of recorded statements per day. Figure 2 shows the event rate for the period used in the simulation (between event day 700 and 800; measured in ordinal time). On average 7.78 statements were recorded per stratum (sd = 8.2; first quantil = 2, median = 6, thrid quantil = 10).

Additional Information on Comparing Network Characteristics

Figures 9 and 10 in the article showed box plots without outliers to improve visibility. Figures 3 and 4 here report the box plots with outliers marked as dots. Additionally Figure 5 holds the goodness of fit plots for the random sequences. They form a baseline that the model-based simulations can be compared to.

Additional Information and Tests on the Number of Simulations

The predictions presented in the article are based on 1,000 simulated event sequences each for the full and the reduced model as well as 1,000 random sequences. As mentioned in the Discussion in the article, these simulations can be computationally intense. The 3,000 simulated sequences took over two weeks (without breaks), running in parallel on multiple computers with a total of 16 cores. Most of the time was used to predict 1000 event sequences from the full model as these models contained the more complicated closing four-cycle statistics that strain computations. To better understand how the number of simulations affects the outcomes, several of the graphs are repeated with smaller number of simulations and reported here. Results do not change drastically with increased number of simulations and yet become more acturate which is desirable.

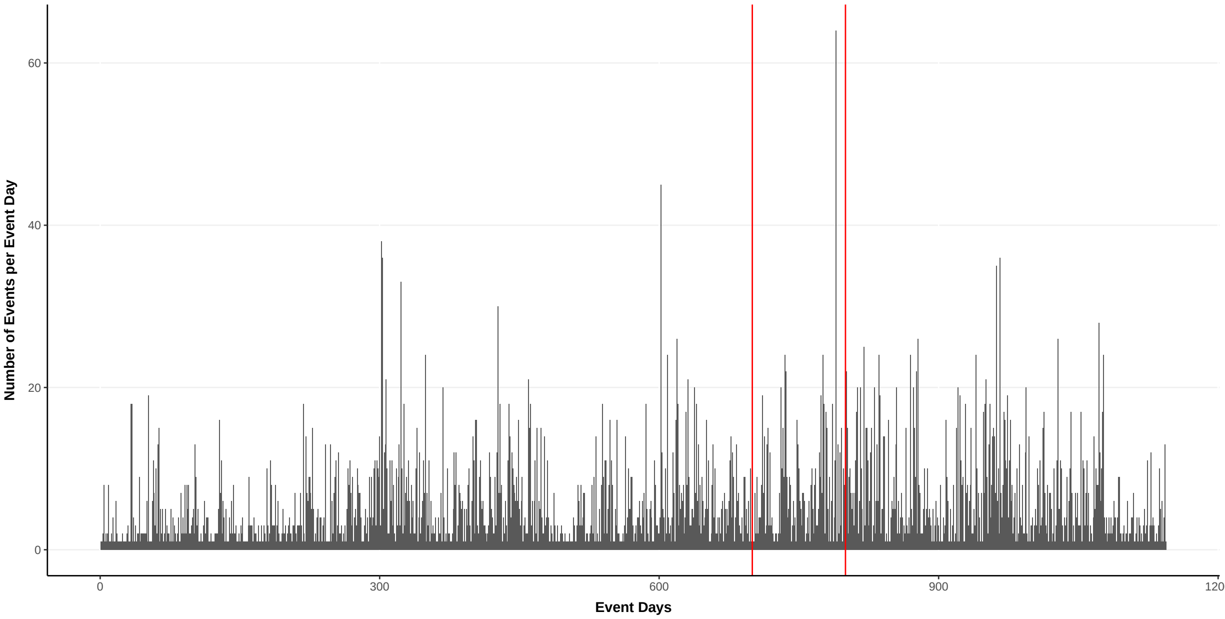

Figure 1: Number of statements per event day over the course of the debate. Red vertical lines indicate the period that was used to make the predictions.

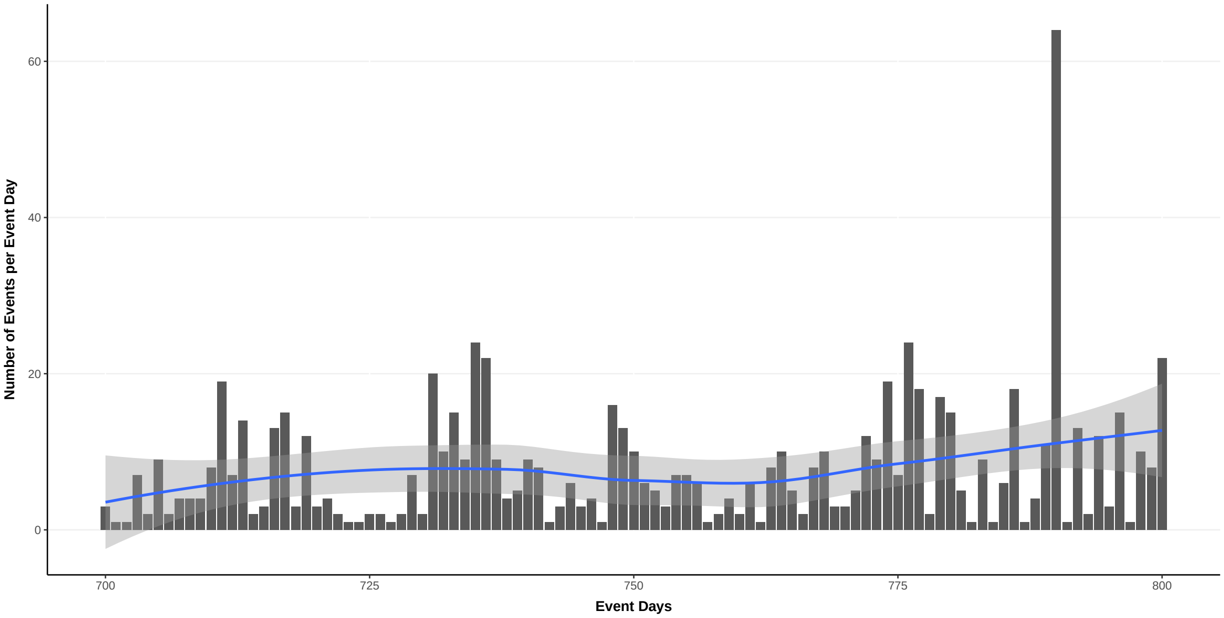

Figure 2: Number of statemetns per day during the period that was used to make the predictions. On average, 8 statements were coded per event day. The blue line represents a smoothed average.

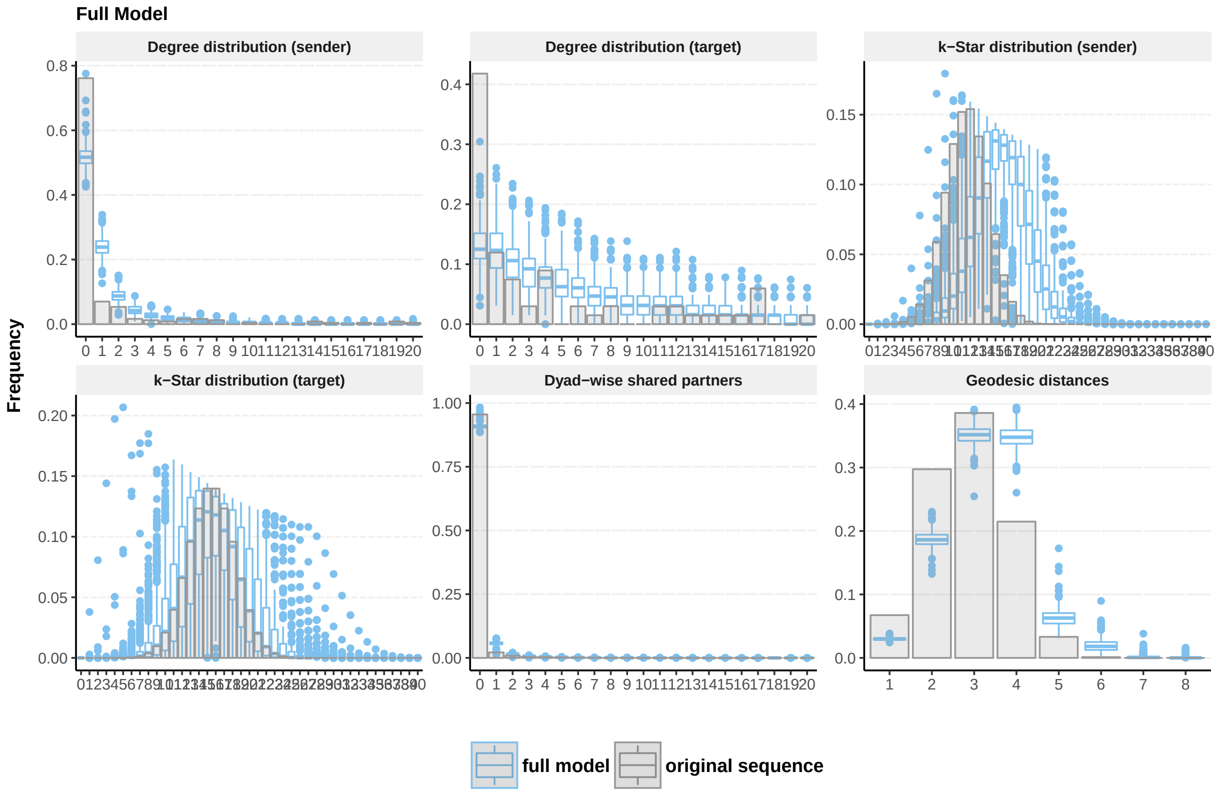

Figure 3: Goodness of fit assessment through comparison of network characteristics. The bar chart represents the distributions from the network snapshot generated by aggregating all events between event day 700 and 800 of the original event sequence. The boxplot corresponds to the distributions captured in the model-based simulations (full model). The full model captures degree distributions fairly well, slightly overestimates k-star distribution of the sender mode and yields good results for k-star distributions of the target mode, dyad-wise shared partner and geodesic distance distributions.

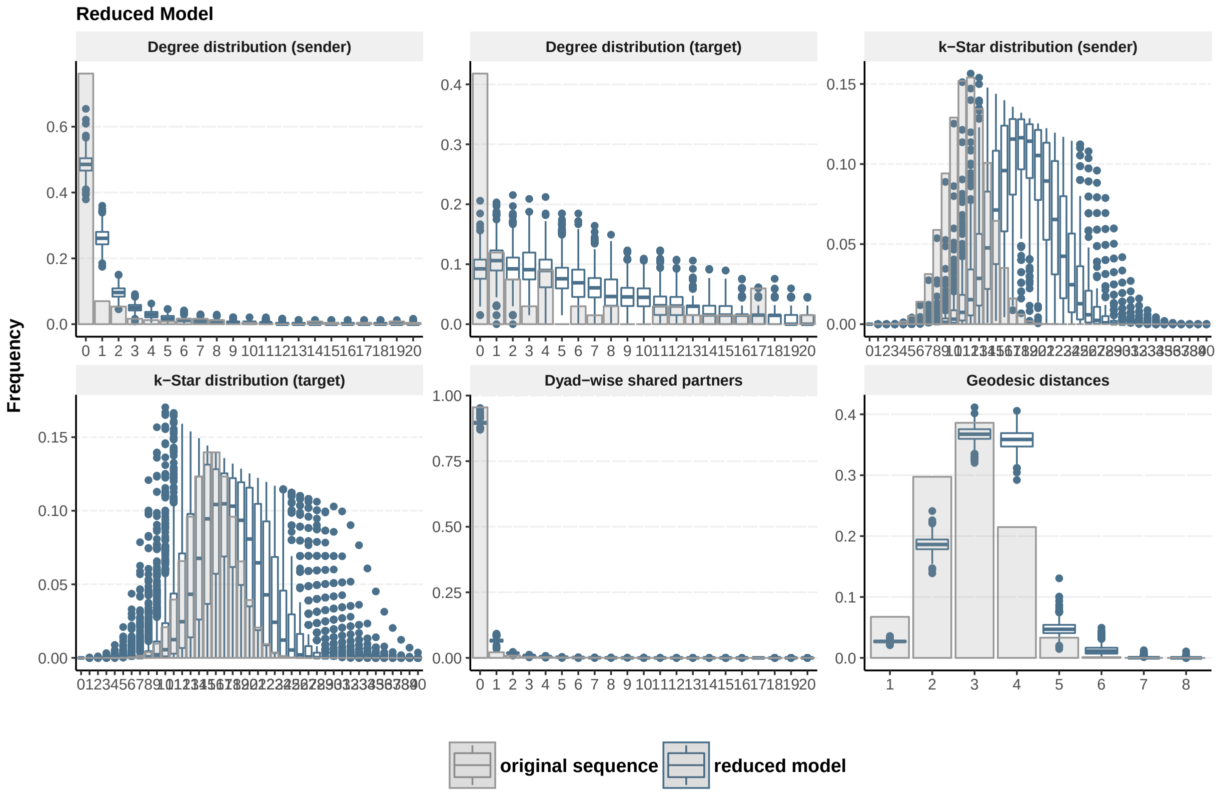

Figure 4: Goodness of fit assessment through comparison of network characteristics. The bar chart represents the distributions from the network snapshot generated by aggregating all events between event day 700 and 800 of the original event sequence. The boxplot corresponds to the distributions captured in the model-based simulations (reduced model). The reduced model captures degree distributions slightly skewed, overestimates k-star distribution of the sender mode and yields relatively good results for k-star distributions of the target mode, dyad-wise shared partner and geodesic distance distributions.

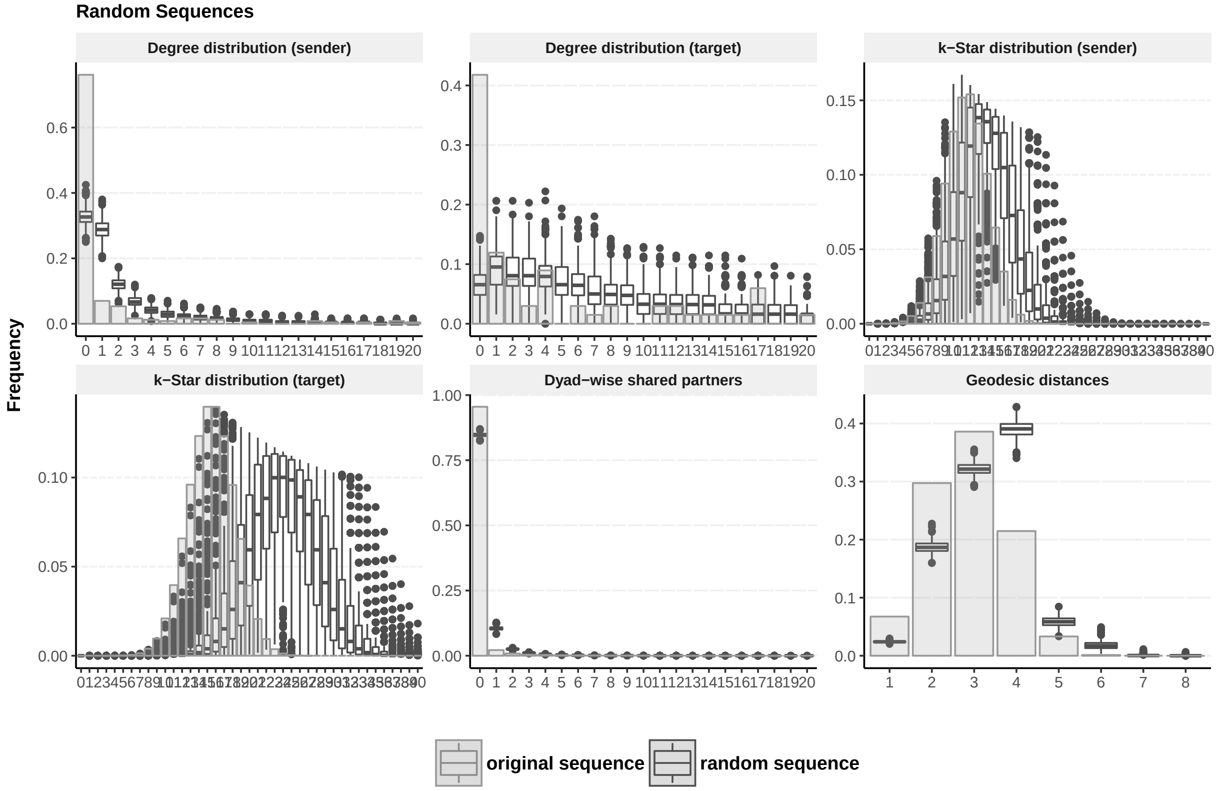

Figure 5: Goodness of fit assessment through comparison of network characteristics. The bar chart represents the distributions from the network snapshot generated by aggregating all events between event day 700 and 800 of the original event sequence. The boxplot corresponds to the distributions captured in the random sequences.

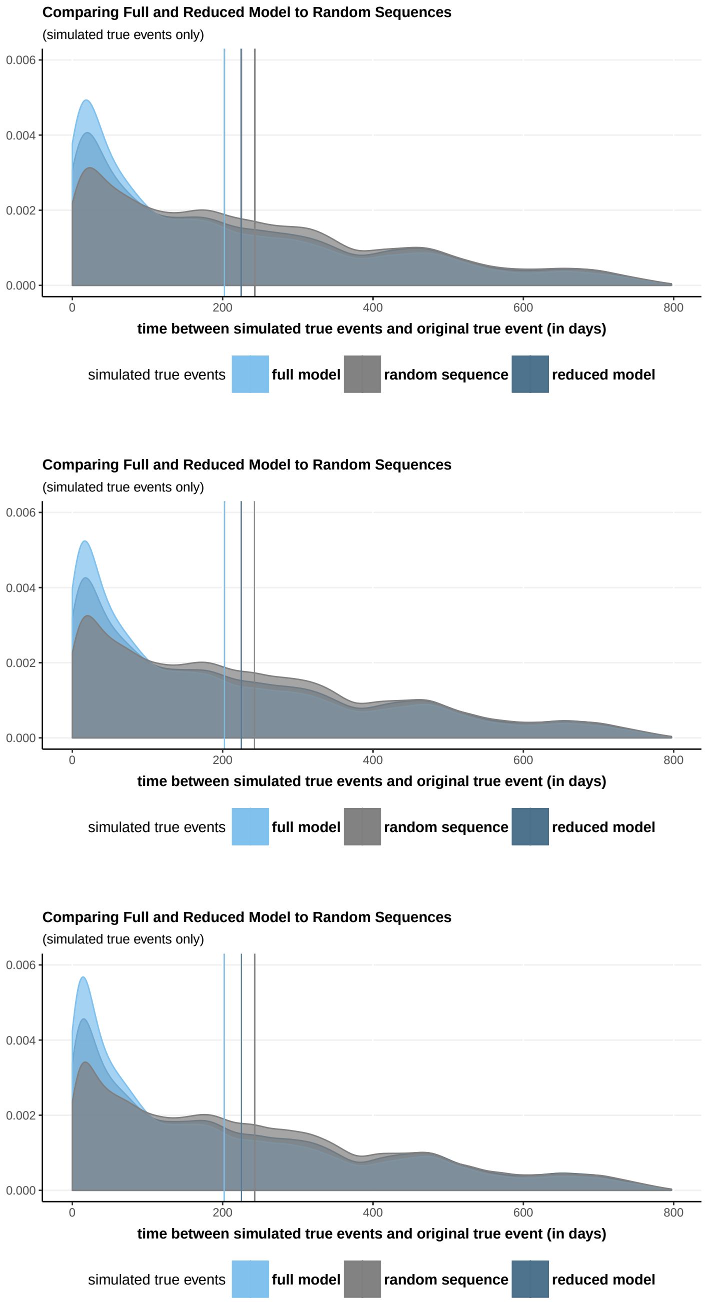

Figure 6: Temporal prediction error of simulated true events for the full and reduced model as well as for the random sequence. The top Figure contains 100 simulations, the middle Figure 200 simulations and the bottom Figure contains 500 simulations.

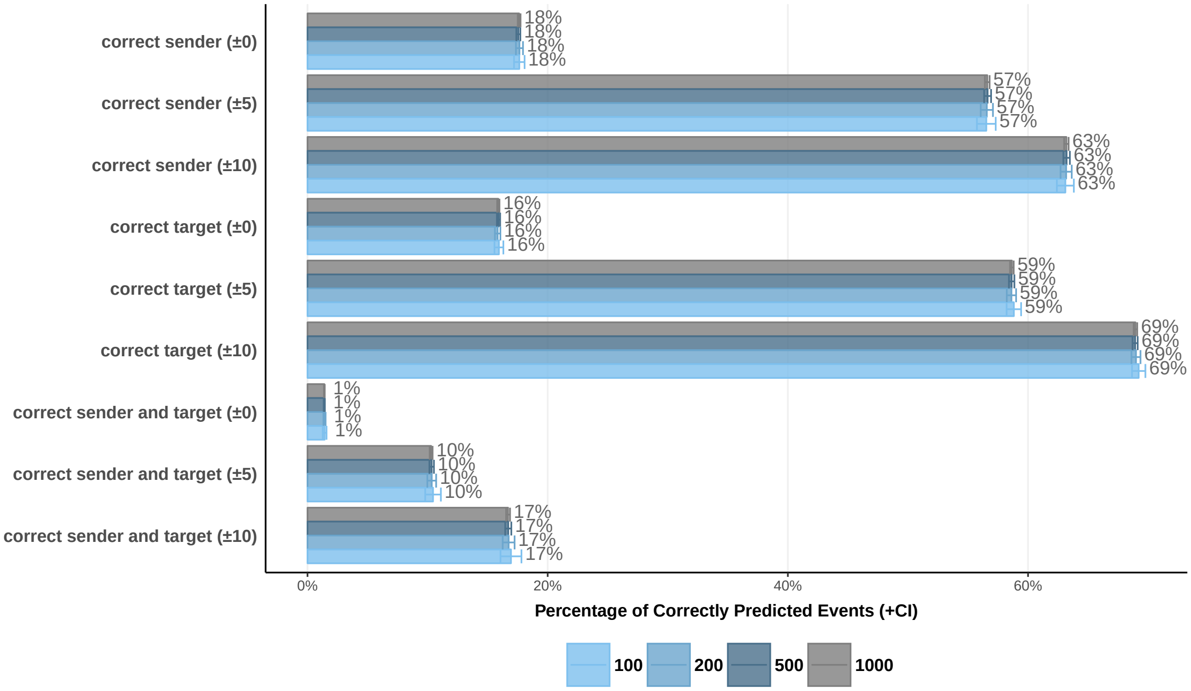

Figure 7: Goodness of fit assessment through event matching. Matches are counted for the full model only. Results are reported for 100, 200, 500 and 1,000 simulated sequences.

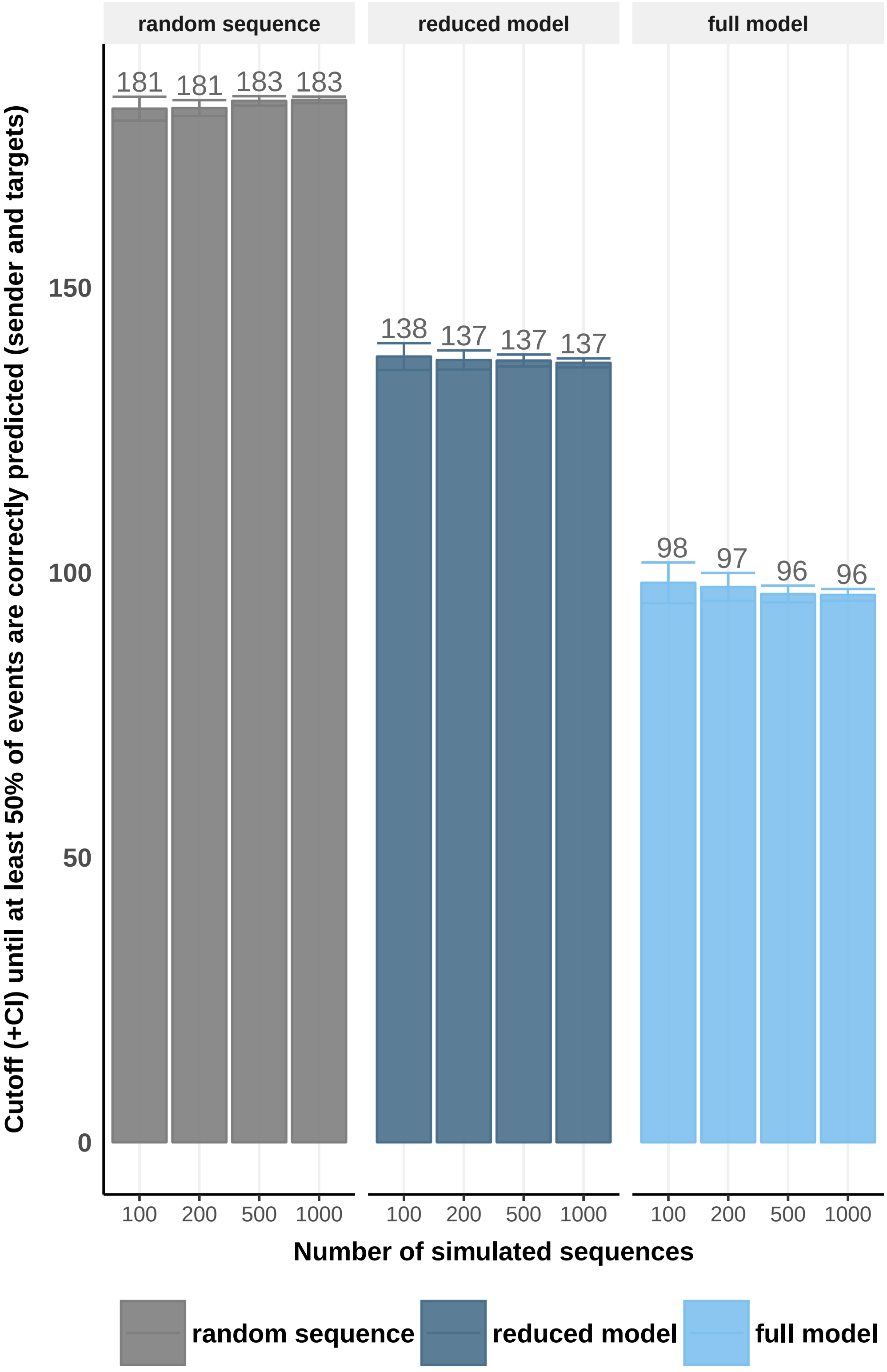

Figure 8: Goodness of fit assessment through event matching. The bar chart shows how large the tolerance has to get until the model reaches 50% correct event (sender-target combinations) predictions. Tolerance are reported for 100, 200, 500 and 1,000 simulations.

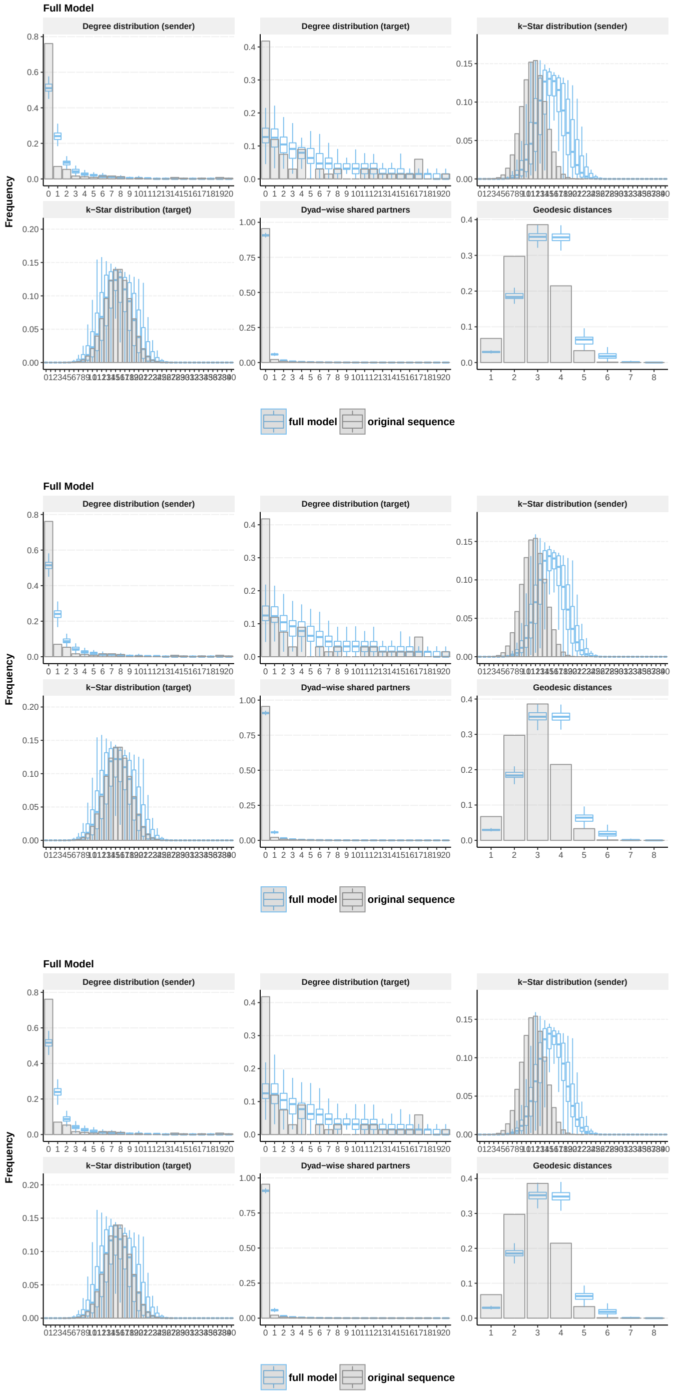

Figure 9: Goodness of fit assessment through comparison of network characteristics. Results for the full model are reported. The top Figure contains 100 simulations, the middle Figure 200 simulations and the bottom Figure contains 500 simulations.

Table 1: Exact numbers of the results of the effects of the number of simulations on the precision of the predictions presented in Figure 7

Nr. simulations	matching	mean correct matches	upper CI	lower CI
100	correct sender (±0)	0.176	0.181	0.172
100	correct sender (±10)	0.631	0.638	0.624
100	correct sender (±5)	0.565	0.573	0.557
100	correct sender and target (±0)	0.014	0.016	0.013
100	correct sender and target (±10)	0.169	0.178	0.161
100	correct sender and target (±5)	0.105	0.111	0.098
100	correct target (±0)	0.159	0.163	0.156
100	correct target (±10)	0.692	0.698	0.687
100	correct target (±5)	0.588	0.594	0.582
200	correct sender (±0)	0.176	0.179	0.174
200	correct sender (±10)	0.632	0.636	0.627
200	correct sender (±5)	0.566	0.571	0.561
200	correct sender and target (±0)	0.014	0.015	0.013
200	correct sender and target (±10)	0.167	0.172	0.163
200	correct sender and target (±5)	0.103	0.107	0.100
200	correct target (±0)	0.158	0.161	0.156
200	correct target (±10)	0.690	0.694	0.686
200	correct target (±5)	0.586	0.590	0.582
500	correct sender (±0)	0.176	0.177	0.174
500	correct sender (±10)	0.632	0.635	0.629
500	correct sender (±5)	0.566	0.569	0.563
500	correct sender and target (±0)	0.014	0.014	0.013
500	correct sender and target (±10)	0.167	0.170	0.165
500	correct sender and target (±5)	0.103	0.105	0.102
500	correct target (±0)	0.159	0.160	0.158
500	correct target (±10)	0.689	0.691	0.687
500	correct target (±5)	0.586	0.589	0.584
1000	correct sender (±0)	0.176	0.177	0.175
1000	correct sender (±10)	0.632	0.634	0.630
1000	correct sender (±5)	0.566	0.568	0.564
1000	correct sender and target (±0)	0.014	0.014	0.014
1000	correct sender and target (±10)	0.167	0.169	0.165
1000	correct sender and target (±5)	0.103	0.104	0.102
1000	correct target (±0)	0.159	0.160	0.158
1000	correct target (±10)	0.689	0.691	0.688
1000	correct target (±5)	0.587	0.588	0.585

Replication material

All replication materials are available at the *Political Analysis* Dataverse site: https://doi.org/10.7910/DVN/GM5SYQ.

References

Brandenberger, Laurence. 2018. “Rem: Relational Event Models.” R package version 1.3.1. https://github.com/brandenberger/rem (last access: August, 2018).

Leifeld, Philip. 2016. *Policy Debates as Dynamic Networks: German Pension Politics and Privatization Discourse.* Frankfurt am Main: Campus.