Computer Practical 2

Relational (Hyper) Event Model: Inference

Wit, EC., Lomi, A., Lerner, J., Boschi, M. & Lembo, M.

2025-06-24

Remark: Each CP begins by loading data from the previous CP, which is located in the corresponding `_OUTPUT_CP_x_` folder. To ensure this file runs smoothly, the contents of that folder should be copied to the `_INPUT_CP_y_` folder of the current CP.

Session 1: Preparatory Steps

1.1. Installing libraries

```
if (!require("mgcv", quietly = TRUE)) {
  install.packages("mgcv")
  library("mgcv")
} else {
  if (!require("mgcViz", quietly = TRUE)) {
    install.packages("mgcViz")
    library("mgcViz")
  } else {
    if (!require("ggplot2", quietly = TRUE)) {
      install.packages("ggplot2")
      library("ggplot2")
    } else {
      if (!require("survival", quietly = TRUE)) {
        install.packages("survival")
        library("survival")
      } else {
        if (!require("RColorBrewer", quietly = TRUE)){
          install.packages("RColorBrewer")
        } else {
          if (!require("dplyr", quietly = TRUE)){
          install.packages("dplyr")
          } else {
            library("mgcv")
            library("mgcViz")
            library("ggplot2")
            library("survival")
            library("RColorBrewer")
            library(dplyr)
          }
        }
      }
    }
```

```
  }
}
```

1.2. Loading Data

During Computer Practical 1, you computed the necessary statistics to support the inference techniques that will be explored in this second practical. Upon inspecting the dataset, you may notice the presence of missing values (NA) in some columns.

In particular, the `TARGET` column contains only missing values, while the `SOURCE` column includes multiple entries. This indicates that we are dealing with **undirected relational hyper-events**, where the sender set is a subset of the vertices in a relational hypergraph V, and the receiver set is empty.

Additionally, the `EVENT_INTERVAL_ID` column does not hold interpretable values, as in this application we just know the order of events. The `IS_OBSERVED` column is used to distinguish between actual observed events and non-events (i.e., events that could have occurred but did not).

```
data_original <- read.csv("_INPUT_CP_2_/jean_events_EVENTS.csv")
```

```
head(data_original)
```

```
##   IS_OBSERVED     SOURCE TARGET    TYPE EVENT_INTERVAL_ID EVENT INTEGER_TIME
## 1           1 |MY|NP|MB|     NA chapter             1.1.1   109          108
## 2           0 |CL|GR|MT|     NA                             111          110
## 3           0 |MA|BB|CV|     NA                             111          110
## 4           0 |PG|GT|CM|     NA                             111          110
## 5           0 |TH|MM|HL|     NA                             111          110
## 6           0 |TH|LL|SS|     NA                             111          110
##   TIME_POINT TIME_UNIT EVENT_INTERVAL num.actors individual.activity
## 1        109       108              2          3                   0
## 2        111       110              2          3                   0
## 3        111       110              2          3                   0
## 4        111       110              2          3                   0
## 5        111       110              2          3                   0
## 6        111       110              2          3                   0
##   dyadic.activity closure female diff.female
## 1               0       0      1           2
## 2               0       0      2           2
## 3               0       0      0           0
## 4               0       0      0           0
## 5               0       0      1           2
## 6               0       0      2           2
```

In this application, the `num.actors` column is also unnecessary. This is because non-events have been sampled to match the cardinality of the observed events, meaning that the effect of event size cannot be meaningfully estimated.

To simplify visualization, we proceed by removing all unnecessary columns from the dataset.

```
data <- data_original[,setdiff(colnames(data_original),
                               c("TARGET", "TYPE", "EVENT_INTERVAL_ID", "EVENT",
                                 "INTEGER_TIME", "TIME_POINT",
                                 "TIME_UNIT", "num.actors"))]
rm(data_original)
head(data)
```

```
##   IS_OBSERVED     SOURCE EVENT_INTERVAL individual.activity dyadic.activity
```

```
## 1           1 |MY|NP|MB|              2                   0               0
## 2           0 |CL|GR|MT|              2                   0               0
## 3           0 |MA|BB|CV|              2                   0               0
## 4           0 |PG|GT|CM|              2                   0               0
## 5           0 |TH|MM|HL|              2                   0               0
## 6           0 |TH|LL|SS|              2                   0               0
##   closure female diff.female
## 1       0      1           2
## 2       0      2           2
## 3       0      0           0
## 4       0      0           0
## 5       0      1           2
## 6       0      2           2
```

1. Which inference technique can we apply?

During the theoretical session, we explored several inference techniques. However, **not all of them are applicable in this context** due to limitations that often arise in the available data.

Firstly, full-likelihood-based methods cannot be applied because **we lack precise timestamp information**. The dataset only provides the ordinal ordering of events, not their exact timings.

Secondly, consider a situation where you are not able to control the sampling of non-events—as was possible using `eventnet` in previous exercises — and are instead provided with a **pre-sampled dataset**. In this case, you cannot construct a full risk set, and must instead rely on case-control sampling techniques.

1.1. Inspect the Data and the Risk Set

If you remember the computations you did in the Computer Practical 1, you can already guess the number of lines of the following datasets.

Just for the sake of clarity — since some confusion may arise — in this CP we use m to denote the number of sampled non-events, but theoretically it used to name size of the sampled risk set, including the event as well.

```
m = 20
```

```
data_ev <- data %>% filter(IS_OBSERVED == 1)
data_nv <- data %>% filter(IS_OBSERVED == 0)
```

```
nrow(data_ev)
```

```
## [1] 288
```

```
nrow(data_nv)
```

```
## [1] 5760
```

```
nrow(data_nv) == nrow(data_ev) * m
```

```
## [1] TRUE
```

1.2. Data looks ready for... coxph or clogit!

Even though we do not have access to the full risk set, we can still use the `coxph` function in R to fit our model. While this does not exactly replicate the coding that would be performed with full timing information and a complete risk set, it remains quite similar.

```
coxph_fit <- coxph(Surv(time = rep(1,nrow(data)),
                        event = data$IS_OBSERVED) ~
                         + diff.female
```

```
                        # + female
                        + individual.activity
                        + dyadic.activity
                        + closure
                        + strata(EVENT_INTERVAL)
                        , data = data)
summary(coxph_fit)
```

```
## Call:
## coxph(formula = Surv(time = rep(1, nrow(data)), event = data$IS_OBSERVED) ~
##     +diff.female + individual.activity + dyadic.activity + closure +
##         strata(EVENT_INTERVAL), data = data)
##
##   n= 6048, number of events= 288
##
##                          coef exp(coef)  se(coef)      z Pr(>|z|)
## diff.female         -0.404858  0.667071  0.103253 -3.921 8.82e-05 ***
## individual.activity  0.044119  1.045107  0.003909 11.288  < 2e-16 ***
## dyadic.activity      0.437302  1.548524  0.068854  6.351 2.14e-10 ***
## closure              0.002510  1.002513  0.010117  0.248    0.804
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## diff.female            0.6671     1.4991    0.5449    0.8167
## individual.activity    1.0451     0.9568    1.0371    1.0531
## dyadic.activity        1.5485     0.6458    1.3530    1.7723
## closure                1.0025     0.9975    0.9828    1.0226
##
## Concordance= 0.89  (se = 0.014 )
## Likelihood ratio test= 815.5  on 4 df,   p=<2e-16
## Wald test            = 245.2  on 4 df,   p=<2e-16
## Score (logrank) test = 1503  on 4 df,   p=<2e-16
```

When applied in this context, the `clogit` function internally calls the `coxph` routine.

```
clogit_fit <- clogit(IS_OBSERVED ~
                        + diff.female
                        # + female
                        +  individual.activity
                        + dyadic.activity
                        + closure
                        + strata(EVENT_INTERVAL)
                        , data = data)
summary(clogit_fit)
```

```
## Call:
## coxph(formula = Surv(rep(1, 6048L), IS_OBSERVED) ~ +diff.female +
##     individual.activity + dyadic.activity + closure + strata(EVENT_INTERVAL),
##     data = data, method = "exact")
##
##   n= 6048, number of events= 288
##
##                          coef exp(coef)  se(coef)      z Pr(>|z|)
## diff.female         -0.404858  0.667071  0.103253 -3.921 8.82e-05 ***
```

```
## individual.activity  0.044119  1.045107  0.003909 11.288  < 2e-16 ***
## dyadic.activity      0.437302  1.548524  0.068854  6.351 2.14e-10 ***
## closure              0.002510  1.002513  0.010117  0.248    0.804
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## diff.female            0.6671     1.4991    0.5449    0.8167
## individual.activity    1.0451     0.9568    1.0371    1.0531
## dyadic.activity        1.5485     0.6458    1.3530    1.7723
## closure                1.0025     0.9975    0.9828    1.0226
##
## Concordance= 0.89  (se = 0.014 )
## Likelihood ratio test= 815.5  on 4 df,   p=<2e-16
## Wald test            = 245.2  on 4 df,   p=<2e-16
## Score (logrank) test = 1503  on 4 df,   p=<2e-16
```

```
coefficients(coxph_fit)
```

```
##         diff.female individual.activity     dyadic.activity             closure
##        -0.404858184         0.044119046         0.437302413         0.002510065
```

```
coefficients(clogit_fit)
```

```
##         diff.female individual.activity     dyadic.activity             closure
##        -0.404858184         0.044119046         0.437302413         0.002510065
```

MODEL INTERPRETATION

We find a positive effect of individual, dyadic. This suggests that prior (co-)presence in previous chapters — whether alone or in pairs — increases the rate of participating in events. A negative effect for diff.female suggests a positive effect for gender homophily, favouring co-appearence with other actors of the same gender. Closure is not significant, and, thus, should not be interpreted.

2. Let's play with... m!

The goal is to investigate how the parameter estimates behave as a function of m. To this end, we consider four different values of m: 1, 5, 10, and 15. These values correspond to the number of non-events sampled for each observed event. Note that we already have a dataset where $m = 20$, which will serve as a reference point for comparison.

```
merge_id_cols <- c("EVENT_INTERVAL")
# Consider 4 different values of m
ms <- c(1, 5, 10, 15)
```

2.1. First, we need to create the data...

To evaluate the effect of m, we can create multiple datasets by varying the number of non-events included per event. We first tag the `data_ev` and `data_nv` datasets with a `.row_type` column to distinguish events from non-events when merging.

```
# Add a new column to label each row in 'data_ev' and 'data_nv'
# as event data and non-event data respectively
data_ev_tagged <- data_ev %>%
  mutate(.row_type = "ev")
data_nv_tagged <- data_nv %>%
  mutate(.row_type = "nv")
head(data_ev_tagged)
```

```
##   IS_OBSERVED              SOURCE EVENT_INTERVAL individual.activity
## 1           1          |MY|NP|MB|              2                    0
## 2           1          |MY|ME|MB|              3                    2
## 3           1               |MY|              4                    2
## 4           1 |MY|ME|CL|GE|MC|MB|              5                    6
## 5           1          |MY|MB|ME|              6                    9
## 6           1             |ME|MY|              7                    8
##   dyadic.activity closure female diff.female .row_type
## 1               0       0      1           2        ev
## 2               1       2      2           2        ev
## 3               0       0      0           0        ev
## 4               4       8      3           9        ev
## 5               7      32      2           2        ev
## 6               3      12      1           1        ev
```

```
# head(data_nv_tagged)
rm(data_ev)
rm(data_nv)
```

For each value of `m` in `ms`, we group the non-event data (`data_nv`) by a set of identifier columns (`EVENT_INTERVAL`) and sample `m` rows per group. These sampled non-events are then combined with the corresponding events and stored in a list `dat_sampled`.

Each entry in the list corresponds to a different sampling size and is labeled accordingly.

```
# List to store the sampled datasets for each sampling size
dat_sampled <- list()

# Loop through each sampling size
for (mm in ms){

  set.seed(mm)

  # Sample 'm' rows (if available) from each group in 'data_nv'
  data_nv_m_sampled <- data_nv_tagged %>%
    group_by(across(all_of(merge_id_cols))) %>%
    slice_sample(n = mm, replace = FALSE) %>%
    ungroup()

  # Combine the event data and the sampled non-event data
  dat_sampled_m <- bind_rows(data_ev_tagged, data_nv_m_sampled) %>%

    # Sort the combined data by the grouping columns and by row type
    # (ev first, then nv)
    arrange(across(all_of(merge_id_cols)), .row_type)

  # Store the resulting data frame in the list
  dat_sampled[[match(mm, ms)]] <- dat_sampled_m
}

# Assign names to each list element
names(dat_sampled) <- ms

# Remove intermediate variables to clean up the workspace
```

```
rm(data_nv_m_sampled, dat_sampled_m)
```

```
# let's check
nrow(dat_sampled[[1]]) == nrow(data_ev_tagged)*(ms[1]+1)
```

```
## [1] TRUE
```

```
nrow(dat_sampled[[2]]) == nrow(data_ev_tagged)*(ms[2]+1)
```

```
## [1] TRUE
```

```
nrow(dat_sampled[[3]]) == nrow(data_ev_tagged)*(ms[3]+1)
```

```
## [1] TRUE
```

```
nrow(dat_sampled[[4]]) == nrow(data_ev_tagged)*(ms[4]+1)
```

```
## [1] TRUE
```

To better understand the structure of the sampled datasets, we can inspect one of them. As shown below, each grouped instance includes exactly one observed event along with a set of randomly sampled non-events, based on the selected value of m. The structure of the data remains consistent with the original format, but is now restricted to a smaller, case-control-like risk set per event.

```
head(dat_sampled[[2]], 10)
```

```
##    IS_OBSERVED     SOURCE EVENT_INTERVAL individual.activity dyadic.activity
## 1            1 |MY|NP|MB|              2                   0                0
## 2            0 |MA|BB|CV|              2                   0                0
## 3            0 |BM|MT|BL|              2                   0                0
## 4            0 |NP|LL|FT|              2                   0                0
## 5            0 |GT|BZ|HL|              2                   0                0
## 6            0 |MT|JV|GA|              2                   0                0
## 7            1 |MY|ME|MB|              3                   2                1
## 8            0 |JU|EN|TM|              3                   0                0
## 9            0 |CR|FF|BL|              3                   0                0
## 10           0 |JO|AZ|BR|              3                   0                0
##    closure female diff.female .row_type
## 1        0      1           2        ev
## 2        0      0           0        nv
## 3        0      1           2        nv
## 4        0      1           2        nv
## 5        0      1           2        nv
## 6        0      1           2        nv
## 7        2      2           2        ev
## 8        0      1           2        nv
## 9        0      0           0        nv
## 10       0      1           2        nv
```

2.2. . . . and then, we can fit the models!

```
clogit_fits <- list()
```

```
for (mm in ms){
  print(mm)
  clogit_fits[[match(mm, ms)]] <- clogit(IS_OBSERVED ~
                          + diff.female
```

```
                        # + female
                        + individual.activity
                        + dyadic.activity
                        + closure
                        + strata(EVENT_INTERVAL)
                        , data = dat_sampled[[match(mm, ms)]])
}
```

```
## [1] 1
## [1] 5
## [1] 10
## [1] 15
```

```
# summary(clogit_fits[[1]])
```

```
summary(clogit_fits[[2]])
```

```
## Call:
## coxph(formula = Surv(rep(1, 1728L), IS_OBSERVED) ~ +diff.female +
##     individual.activity + dyadic.activity + closure + strata(EVENT_INTERVAL),
##     data = dat_sampled[[match(mm, ms)]], method = "exact")
##
##   n= 1728, number of events= 288
##
##                         coef exp(coef)  se(coef)      z Pr(>|z|)
## diff.female         -0.604575  0.546306  0.120661 -5.011 5.43e-07 ***
## individual.activity  0.052845  1.054266  0.006607  7.998 1.26e-15 ***
## dyadic.activity      0.662640  1.939906  0.142777  4.641 3.47e-06 ***
## closure             -0.020830  0.979386  0.013946 -1.494    0.135
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## diff.female            0.5463     1.8305    0.4313    0.6921
## individual.activity    1.0543     0.9485    1.0407    1.0680
## dyadic.activity        1.9399     0.5155    1.4664    2.5663
## closure                0.9794     1.0210    0.9530    1.0065
##
## Concordance= 0.891  (se = 0.017 )
## Likelihood ratio test= 548.5  on 4 df,   p=<2e-16
## Wald test            = 107.8  on 4 df,   p=<2e-16
## Score (logrank) test = 478.1  on 4 df,   p=<2e-16
```

```
summary(clogit_fits[[3]])
```

```
## Call:
## coxph(formula = Surv(rep(1, 3168L), IS_OBSERVED) ~ +diff.female +
##     individual.activity + dyadic.activity + closure + strata(EVENT_INTERVAL),
##     data = dat_sampled[[match(mm, ms)]], method = "exact")
##
##   n= 3168, number of events= 288
##
##                         coef exp(coef)  se(coef)      z Pr(>|z|)
## diff.female         -0.343828  0.709051  0.087017 -3.951 7.77e-05 ***
## individual.activity  0.045224  1.046262  0.004686  9.650  < 2e-16 ***
## dyadic.activity      0.468795  1.598067  0.087932  5.331 9.75e-08 ***
```

```
## closure              0.012236  1.012311  0.011431  1.070    0.284
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## diff.female            0.7091     1.4103    0.5979    0.8409
## individual.activity    1.0463     0.9558    1.0367    1.0559
## dyadic.activity        1.5981     0.6258    1.3451    1.8986
## closure                1.0123     0.9878    0.9899    1.0352
##
## Concordance= 0.89  (se = 0.015 )
## Likelihood ratio test= 686.6  on 4 df,   p=<2e-16
## Wald test            = 166  on 4 df,   p=<2e-16
## Score (logrank) test = 882.5  on 4 df,   p=<2e-16
```

```
summary(clogit_fits[[4]])
```

```
## Call:
## coxph(formula = Surv(rep(1, 4608L), IS_OBSERVED) ~ +diff.female +
##     individual.activity + dyadic.activity + closure + strata(EVENT_INTERVAL),
##     data = dat_sampled[[match(mm, ms)]], method = "exact")
##
##   n= 4608, number of events= 288
##
##                          coef exp(coef)  se(coef)      z Pr(>|z|)
## diff.female         -0.393556  0.674653  0.103287 -3.810 0.000139 ***
## individual.activity  0.043605  1.044569  0.004140 10.533  < 2e-16 ***
## dyadic.activity      0.456604  1.578704  0.075043  6.085 1.17e-09 ***
## closure              0.008066  1.008099  0.010591  0.762 0.446317
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## diff.female            0.6747     1.4822    0.5510     0.826
## individual.activity    1.0446     0.9573    1.0361     1.053
## dyadic.activity        1.5787     0.6334    1.3628     1.829
## closure                1.0081     0.9920    0.9874     1.029
##
## Concordance= 0.891  (se = 0.014 )
## Likelihood ratio test= 770.2  on 4 df,   p=<2e-16
## Wald test            = 211.4  on 4 df,   p=<2e-16
## Score (logrank) test = 1196  on 4 df,   p=<2e-16
```

3. It's now time for m=1 using GAMs!

3.1. Again, data first...

As discussed in the theoretical session, when only one non-event is available per observed event ($m = 1$), we can perform inference using a degenerate logistic regression. In this setup, all responses are set to 1, and the covariates are defined as the difference between the covariate values for the event and the corresponding non-event.

We now proceed to construct a dataset formatted in this way, which will allow us to apply this degenerate logistic regression approach.

```r
set.seed(10)

# For each group defined by the merge_id_cols, take one random non-event
data_nv_1_sampled <- data_nv_tagged %>%
  group_by(across(all_of(merge_id_cols))) %>%
  slice_sample(n = 1) %>%
  ungroup()

# Perform a left join:
# for each row in data_ev, attach the corresponding non-event row (if available)
# based on matching values in the merge_id_cols
# The suffixes "_ev" and "_nv" will be added to columns from data_ev and data_nv
dat_gam_1 <- data_ev_tagged %>%
  left_join(data_nv_1_sampled,
            by = merge_id_cols, suffix = c("_ev", "_nv"))

# All responses are set equal to 1
dat_gam_1$y <- 1

rm(data_nv_1_sampled)
```

```r
# covariates defined as difference between
# covariate values for event and corresponding non-event
dat_gam_1$female <-
  dat_gam_1$female_ev - dat_gam_1$female_nv
dat_gam_1$diff_female <-
  dat_gam_1$diff.female_ev - dat_gam_1$diff.female_nv
dat_gam_1$individual_activity <-
  dat_gam_1$individual.activity_ev - dat_gam_1$individual.activity_nv
dat_gam_1$dyadic_activity <-
  dat_gam_1$dyadic.activity_ev - dat_gam_1$dyadic.activity_nv
dat_gam_1$closure <-
  dat_gam_1$closure_ev - dat_gam_1$closure_nv
```

3.2. . . . and fitting then!

```r
gam_fit <- glm(y ~
                + diff_female
                # + female
                + individual_activity
                + dyadic_activity
                + closure
                - 1 # no intercept
               , data = dat_gam_1,
               family="binomial")
```

```
## Warning: glm.fit: si sono verificate probabilità stimate numericamente pari a 0
## o 1
```

```r
summary(gam_fit)
```

```
##
## Call:
## glm(formula = y ~ +diff_female + individual_activity + dyadic_activity +
```

```
##     closure - 1, family = "binomial", data = dat_gam_1)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## diff_female         -0.40997    0.14455  -2.836  0.00457 **
## individual_activity  0.06587    0.01370   4.807 1.54e-06 ***
## dyadic_activity      1.69840    0.47198   3.598  0.00032 ***
## closure             -0.09284    0.04165  -2.229  0.02582 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 399.25  on 288  degrees of freedom
## Residual deviance: 149.14  on 284  degrees of freedom
## AIC: 157.14
##
## Number of Fisher Scoring iterations: 10
```

MODEL INTERPRETATION

Conclusion on the effect of individual and dyadic activity and gender homophily are as in the previously fitted model. Closure, on the other hand, appears significant and has a negative effect.This means that hyperevents with actors that have appeared with a common third-party in the past, are less likely.

SAVING MATERIALS FOR CP3!:

```
save(dat_gam_1, file="_OUTPUT_CP_2/dat_gam_1.RData")
save(gam_fit, file="_OUTPUT_CP_2/gam_fit.RData")
```

3.3. We can still play with m...

```
# Store the case-control datasets for different m
dat_gam <- list()

# Loop through each sampling size defined in 'ms'
for (mm in ms){

  set.seed(mm)

  # Sample randomly 'm' rows from each group in 'data_nv'
  data_nv_m_sampled <- data_nv_tagged %>%
    group_by(across(all_of(merge_id_cols))) %>%
    slice_sample(n = mm, replace = FALSE) %>%
    ungroup()

  # Left join the sampled non-event data to the event data using the merge keys
  # this combines each row of event data with 'm' non-event rows (if available)
  dat_gam_m <- data_ev_tagged %>%
    left_join(data_nv_m_sampled, by = merge_id_cols, suffix = c("_ev", "_nv"))

  # Compute covariate difference
  dat_gam_m$female <-
    dat_gam_1$female_ev - dat_gam_1$female_nv
  dat_gam_m$diff_female <-
```

```
    dat_gam_1$diff.female_ev - dat_gam_1$diff.female_nv
  dat_gam_m$individual_activity <-
    dat_gam_m$individual.activity_ev - dat_gam_m$individual.activity_nv
  dat_gam_m$dyadic_activity <-
    dat_gam_m$dyadic.activity_ev - dat_gam_m$dyadic.activity_nv
  dat_gam_m$closure <-
    dat_gam_m$closure_ev - dat_gam_m$closure_nv

  # All responses are set equal to 1
  dat_gam_m$y <- 1

  # Store the resulting data frame in the list
  dat_gam[[match(mm, ms)]] <- dat_gam_m
}

# Assign names to each list element
names(dat_gam) <- ms

# Remove intermediate variables to clean up the workspace
rm(data_nv_m_sampled, dat_gam_m)
```

3.3.1. Let's create the data we need

```
glm_fits <- list()
```

```
for (mm in ms){
  glm_fits[[match(mm, ms)]] <- glm(y ~
                                     + diff_female
                                     # + female
                                     + individual_activity
                                     + dyadic_activity
                                     + closure
                                     - 1 ,# no intercept
                           family = "binomial",
                           data = dat_gam[[match(mm, ms)]])
}
```

```
summary(glm_fits[[1]])
```

3.3.3. ... and fit the gam!

```
##
## Call:
## glm(formula = y ~ +diff_female + individual_activity + dyadic_activity +
##     closure - 1, family = "binomial", data = dat_gam[[match(mm,
##     ms)]])
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## diff_female         -0.1654268  0.1087905  -1.521  0.12836
## individual_activity  0.0710698  0.0152223   4.669 3.03e-06 ***
## dyadic_activity      0.8171296  0.3143267   2.600  0.00933 **
```

```
## closure             -0.0001117  0.0460743  -0.002  0.99807
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 399.25  on 288  degrees of freedom
## Residual deviance: 151.20  on 284  degrees of freedom
## AIC: 159.2
##
## Number of Fisher Scoring iterations: 11
```

```
summary(glm_fits[[2]])
```

```
##
## Call:
## glm(formula = y ~ +diff_female + individual_activity + dyadic_activity +
##     closure - 1, family = "binomial", data = dat_gam[[match(mm,
##     ms)]])
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## diff_female         -0.142786   0.041502  -3.440 0.000581 ***
## individual_activity  0.051057   0.005113   9.986  < 2e-16 ***
## dyadic_activity      0.518905   0.086545   5.996 2.02e-09 ***
## closure              0.001309   0.006866   0.191 0.848817
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1996.26  on 1440  degrees of freedom
## Residual deviance:  852.61  on 1436  degrees of freedom
## AIC: 860.61
##
## Number of Fisher Scoring iterations: 9
```

```
summary(glm_fits[[3]])
```

```
##
## Call:
## glm(formula = y ~ +diff_female + individual_activity + dyadic_activity +
##     closure - 1, family = "binomial", data = dat_gam[[match(mm,
##     ms)]])
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## diff_female         -0.114251   0.026214  -4.358 1.31e-05 ***
## individual_activity  0.055889   0.003731  14.979  < 2e-16 ***
## dyadic_activity      0.856406   0.091342   9.376  < 2e-16 ***
## closure             -0.040617   0.008909  -4.559 5.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 3992.5  on 2880  degrees of freedom
## Residual deviance: 1654.7  on 2876  degrees of freedom
## AIC: 1662.7
##
## Number of Fisher Scoring iterations: 9
```

```
summary(glm_fits[[4]])
```

```
##
## Call:
## glm(formula = y ~ +diff_female + individual_activity + dyadic_activity +
##     closure - 1, family = "binomial", data = dat_gam[[match(mm,
##     ms)]])
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## diff_female         -0.133330   0.021222  -6.283 3.33e-10 ***
## individual_activity  0.054709   0.003038  18.010  < 2e-16 ***
## dyadic_activity      0.813777   0.075224  10.818  < 2e-16 ***
## closure             -0.026653   0.007812  -3.412 0.000645 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5988.8  on 4320  degrees of freedom
## Residual deviance: 2446.2  on 4316  degrees of freedom
## AIC: 2454.2
##
## Number of Fisher Scoring iterations: 10
```