Suplementary Material 3: Run Simulations on the Last Birth Omited Data

Richard Meitern

2023-04-12

Introduction

The aim of this document is to display the R code needed the reproduce the simulations for the PISH scenarios. The letters P, I, S & H are used to indicate which mechanisms were included in each simulation scenario:

- P indicates that a twinning event affects parity progression.
- I indicates that a twinning event influences the time between that birth and the next one.
- S indicates that a mother's reproductive schedule affects both her likelihood of having twins and her total number of births, creating a link between these two factors.
- H indicates that there is a connection between twinning propensity and intrinsic fertility due to maternal heterogeneity.

The scenario 0 does not include any of these mechanisms. This documents needs the twinR package to be installed to run.

NB! Some of the code e.g reading in the data is same as in S1.

```
#cleanup memory
gcstuff <- gc(verbose=FALSE); rm(gcstuff);
```

```
cat("The current directory is: ", getwd())
```

```
## The current directory is:  C:/Users/rix133/OneDriveUT/Peeter/papers/natCom_twinR_commentary
```

```
#get last birth adding function
source("./R/last_birth.R")

#simplified twinR summary tables
source("./R/twinR_summary.R")

#fix twinR compute predictions to do prediction with no lambda  as well
source("./R/twinR_predictions.R")

#simple convenience functions
source("./R/utils.R")
```

The memory (RAM) usage is quite high for simulations. The total number of CPUs used for running single simulation on Estonian data w/o last birth was 51. RAM usage varied between models but stayed around 18 GB per CPU core (dumped cores had a size range from 12 to 23GB).

```
## Identify number of CPU cores available for parallel computing,
## note: using a large number may lead RAM to max out, so you may have to adjust
## that according to your infrastructure:
nb_cores <- 40

#it seems that simulating the base model for Estonian not last birth data
#requires about 15GB of RAM per CPU core (tested with 61 CPU cores)


## Set option in spaMM:
spaMM::spaMM.options(nb_cores = nb_cores)
```

```
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

Data Import Estonia

The Estonian dataset has been formatted to include the same columns as *the data_births_all* dataset from the **twinR** package. The only difference is that the columns *pop* and *monthly* are excluded as these are constant.

```
#Import and preproccess Estonian Data

data_births_monthly_EE <- readRDS("./data/data_births_all_EE.rds")

#the twinR package expects population to be present
data_births_monthly_EE$pop <- "Estonia"

data_births_monthly_EE <- add_last_birth(data_births_monthly_EE)

data_births_monthly_EE_not_last <- data_births_monthly_EE[!data_births_monthly_EE$last,]
```

Running Simulations

```
# import the function to do model fit and predictions
source("./R/fit_models.R")
```

The following is copied directly from twinR documentation and only the saving directories and input data frames are changed.

```
#-----------------------------------------------------------------------------------------------
#-------------------------------- Goodness-of-fit tests -----------------------------------------
#-----------------------------------------------------------------------------------------------

### IMPORTANT: while all the steps above should work no matter the operating system and whether
### you are using R within a GUI (e.g. RStudio) or not, the following step is very
### computationally intensive and has been tailored to be used on a Unix system (e.g. Linux) and
### directly within a terminal. It may work in RStudio, but this is not guaranteed, nor
```

```r
### recommended. If you run this code using Windows, it should fallback to running the
### computation sequentially instead of in parallel across multiple CPU cores, which should work
### fine at the cost of requiring probably weeks of running time.

### Run scenarios one by one and save the output in a rda file:

library(twinR)
```

```
##
##  Welcome to twinR!
##  To start, simply type ?twinR and follow the examples

##
## Attaching package: 'twinR'

## The following objects are masked _by_ '.GlobalEnv':
##
##     build_data_summary.table, build_fit_summary.table,
##     compute_predictions
```

```r
library(doSNOW)
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: snow
```

```r
#it seems that simulating the base model for Estonian not last birth data
#requires about 15GB of RAM per CPU core (tested with 61 CPU cores)

timeout <- 1 * 60 * 60 # an Hour

baseSlopeDir <- "./exports/slopes_under_scenarios"
baseFitDir <- "./exports/fitted_models"

scenarios_to_do <- c("base_model", "P", "I", "S", "H",
                     "PI", "PS", "PH", "IS", "IH", "SH",
                     "PIS", "PIH", "PSH", "ISH", "PISH")
```

```r
#use models without populations predictor
source("./R/twinR_models.R")
```

TwinR data

```r
expName <- "_EE_nl"
slopeDir <- paste0(baseSlopeDir, expName)
fitDir <- paste0(baseFitDir, expName)

dir.create(slopeDir, recursive = T)
```

```
## Warning in dir.create(slopeDir, recursive = T): '.
## \exports\slopes_under_scenarios_EE_nl' already exists
```

```r
dir.create(fitDir, recursive = T)
```

```
## Warning in dir.create(fitDir, recursive = T): '.\exports\fitted_models_EE_nl'
## already exists
```

```r
data_births_monthly <- data_births_monthly_EE_not_last

#remove unnecessary data frames
rm(data_births_monthly_EE, data_births_monthly_EE_not_last)

cat("Fits are saved in: ", fitDir)
```

```
## Fits are saved in:  ./exports/fitted_models_EE_nl
```

```r
fit_models  <- FALSE
if(fit_models){
  ## fit all trios of models in parallel (for linux only, but easy to adjust for other OS):
  pbmcapply::pbmclapply(scenarios_to_do, function(scenario) {
    targetFname <-  paste0(fitDir,"/fits_", scenario, "_obs.rda")
    if(!file.exists(targetFname)){
      fits <- fit_life_histories(scenario = scenario,
                                 birth_level_data = data_births_monthly)
      save(fits, file =targetFname, compress = "xz")

      rm(fits)
    }

    }, mc.cores = min(c(length(scenarios_to_do), nb_cores)), mc.preschedule = FALSE)
}
```

Its suggested to use a linux computing cluster to run the simulation scenarios one by one (using *run_simulation_EE_not_last.R*) rather than running the for loop below. Example of a bash command > **Rscript run_simulation_EE_not_last.R –model H –cores 51**. The model indicates the scenario to be run.

```r
for (scenario in scenarios_to_do) {
   name_obj <- paste0("slopes_under_", scenario)
   targetFname <- paste0(slopeDir,"/", name_obj, ".rda")
   if(!file.exists(targetFname)){
     load(file = paste0(fitDir,"/fits_", scenario, "_obs.rda"))
     slopes_under_scenario <- simulate_slopes_for_GOF(N_replicates_level1 = 200L,
                                                       N_replicates_level2 = 20L,
                                                      birth_level_data = data_births_monthly,
                                                       life_history_fits = fits,
                                                       scenario = scenario,
                                                       nb_cores = nb_cores,
                                                       timeout = timeout,
                                                       .log = TRUE, lapply_pkg = "pbmcapply",
                                                       verbose = list(fit = TRUE, simu = FALSE))
```

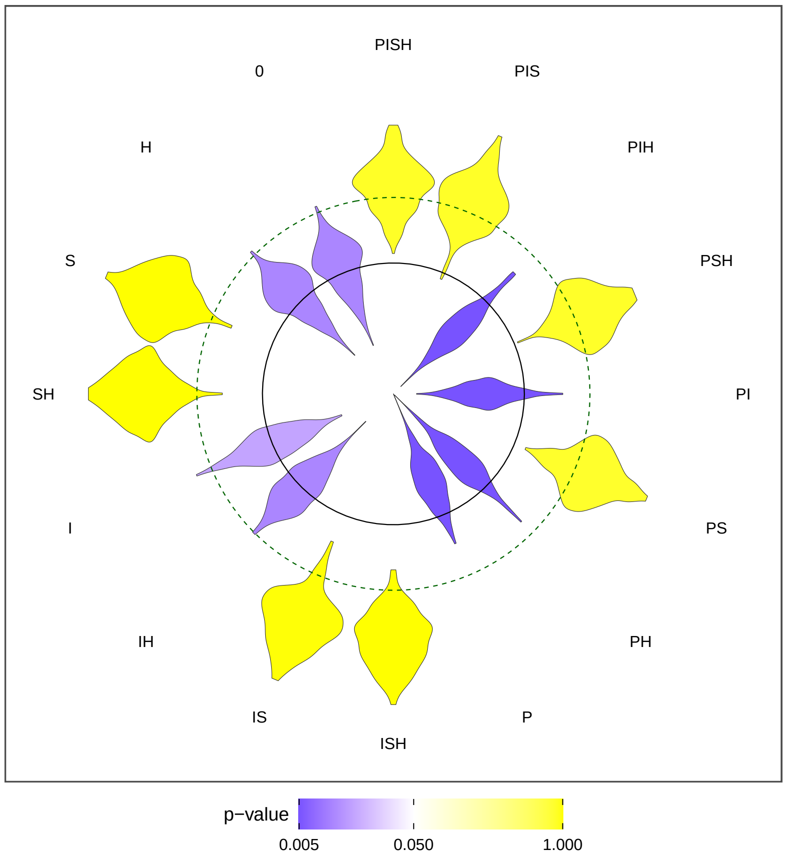
```
      rm(fits)
      assign(name_obj, value = slopes_under_scenario)
      save(list = name_obj, file = targetFname)
      rm(list = name_obj) # remove the object behind the name!
      rm(slopes_under_scenario, scenario, name_obj) # remove the object directly
      gc_stuff <- gc(verbose = FALSE)
    }

}
```

```
#change the p-val expected direction
source("./R/twinR_goodness_of_fit.R")

# unchanged from the original but needed to have positive slopes yellow.
source("./R/twinR_fig_5.R")
```

```
## Figure 5 with colours inverted as we look for positive slopes:
fig5 <- draw_fig_5(all_slopes, width = 1)
fig5Fname <- paste0("./exports/fig5_", expName, ".")
ggplot2::ggsave(filename = paste0(fig5Fname, "pdf"), fig5,
                width = 88, height = 70, units = "mm")
fig5
```



```r
## test scenarios (table S13):

tableS13 <- goodness_of_fit(all_slopes)

writexl::write_xlsx(tableS13, paste0(fig5Fname, "xlsx"))
knitr::kable(tableS13)
```

scenario	pv_gof	pv_raw
P	0.004975124	0.004975124
I	0.014925373	0.014925373
S	1.000000000	0.980099502
H	0.009950249	0.009950249
PI	0.004975124	0.004975124
PS	0.845771144	0.701492537
PH	0.004975124	0.004975124
IS	0.985074627	0.935323383
IH	0.009950249	0.009950249
SH	1.000000000	0.980099502
PIS	0.865671642	0.621890547
PIH	0.004975124	0.004975124
PSH	0.860696517	0.666666667
ISH	1.000000000	0.965174129
PISH	0.875621891	0.641791045
base_model	0.009950249	0.009950249

```r
## computing time (for gof analysis):
timeHours <- computing_time_analysis(all_slopes)
cat("Total CPU time to run simulations was ",
    unlist(round(timeHours / 24, 2)), "days. ",
    "On a high RAM desktop computer with 8 cores it would hence take around",
    unlist(round(timeHours / 24 / 8, 0)), "days to run. However, on a standard ",
    "desktop computer with 24 GB RAM only 1 core could be used.")
```

```
Total CPU time to run simulations was 264.36 days. On a high RAM desktop computer with 8 cores it would hence take around 33 days to run. However, on a standard desktop computer with 24 GB RAM only 1 core could be used.
```

```r
#END
```