svisits: finding HIV transmission and serosorting pairs using shared clinic visit dates

Alex Marzel

This package helps to find some stable HIV-infected partnerships in cohort studies basing on the assumption that some patients frequently attend the clinical visits together. These pairs are useful for biological and epidemiological research.

Package installation

```
install.packages("https://github.com/alexmarzel/svisits/raw/master/svisits_1.1.tar.gz"
                 ,repos = NULL, type = "source")
library(svisits)
set.seed(14051948)
```

First load the database with visit dates in a long format (each row is a visit of a particular person). This simulated dataset contains 20 transmission pseudo pairs. The data were simulated based on visits distribution in the SHCS.

```
data("simulated_data")
# prepare the database
db_dates<-prepare_db(your_database = simulated_data,
                     ids_column = "subject",dates_column = "sim_dates")
head(db_dates)
```

```
##   subject      dates
## 1       1 1993-10-02
## 2       1 1993-12-24
## 3       1 1994-03-16
## 4       1 1994-05-27
## 5       1 1994-08-21
## 6       1 1994-11-06
```

Deconstruct the dates into 3 months periods

```
db_dates<-cbind(db_dates,month.day.year(db_dates$dates));periodMonths<-3
db_dates<-cbind(db_dates,fupdatePeriod=db_dates$year
                +round(floor((db_dates$month-1)/periodMonths) *periodMonths/12,2))
```

Get unadjusted, observed shared visits. Can take some time.

```
unadjusted_observed_pairs<-get_observed_pairs(db_dates)
head(unadjusted_observed_pairs)
```

```
## allPairs
##   1_10 1_1019 1_1023 1_1027  1_103 1_1031
##      1      1      1      1      1      4
```

Descriptive statistics of shared visits for pairs that shared at least a single visit

```r
summary(as.numeric(unadjusted_observed_pairs))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   1.000   1.000   1.321   1.000  24.000
```

Prepare the ids

```r
ids<- data.table(ids=as.character(names(table(db_dates$subject))),
                 N_visits=as.character(as.numeric(table(db_dates$subject))))
setkey(ids, "ids")
```

The chances of two unrelated cohort members to share a visit are increasing with the overall number of visits of each member of the pair. To account for this, we have to adjust the number of shared visits within each pair. This will return the adjusted visits for the observed, unshuffled data, the adjusted number is under the column "Corrected"

```r
adjusted_observed<-adjust_visits(unadjusted_pairs=unadjusted_observed_pairs,ids=ids
                                 ,prob = 1/75)
head(adjusted_observed)
```

```
##   allPairs Freq id_1 id_2 N_visits.x N_visits.y Corrected
## 1     1_10    1    1   10         46         68 0.1132248
## 2     4_10    1    4   10         39         68 0.1514599
## 3     8_10    1    8   10         10         68 0.4666841
## 4     9_10    1    9   10         12         68 0.4244555
## 5 161_1000    1  161 1000         17         11 0.4446087
## 6 176_1000    1  176 1000         11         11 0.4446087
```

Due to the multiple comparisons problem and a strongly simplifying assumptions about the uniform distribution of visits, the corrected number of shared visits is not an optimal test statistic. Instead, the visits are first shuffled within each quarter (such that the original distribution of the number of visits per individual was preserved) and the number of randomly collided shared visits per pair is counted and penalized using "adjust_visits". In other words, the observed visit dates from a given quarter were randomly re-assigned between the patients that attended during this quarter.

Define the number of desired shuffling simulations (ideally >100, but it will take some time) and run. Will take some time to finish.

```r
n_shuffl<-10
Shuffling_simulation_output<- replicate(n_shuffl,
                                        Shuffling_simulation(db_dates,ids=ids,adjusted_observed),
                                        simplify=F)

# inspect the false-positive thresholds
head(Shuffling_simulation_output)
```

```
## [[1]]
##  [1] 0.9394822 0.7810526 0.3095238 0.0000000 0.0000000 0.0000000 0.0000000
##  [8] 0.0000000 0.0000000 0.0000000
## 
## [[2]]
##  [1] 0.9434465 0.7515789 0.2857143 0.0000000 0.0000000 0.0000000 0.0000000
##  [8] 0.0000000 0.0000000 0.0000000
##
## [[3]]
##  [1] 0.9418979 0.6905263 0.1904762 0.0000000 0.0000000 0.0000000 0.0000000
##  [8] 0.0000000 0.0000000 0.0000000
##
## [[4]]
##  [1] 0.9389866 0.6673684 0.2619048 0.0000000 0.0000000 0.0000000 0.0000000
##  [8] 0.0000000 0.0000000 0.0000000
##
## [[5]]
##  [1] 0.9485258 0.6547368 0.1428571 0.0000000 0.0000000 0.0000000 0.0000000
##  [8] 0.0000000 0.0000000 0.0000000
##
## [[6]]
##  [1] 0.92758920 0.65263158 0.30952381 0.04761905 0.00000000 0.00000000
##  [7] 0.00000000 0.00000000 0.00000000 0.00000000
```

Extract False-Positive thresholds

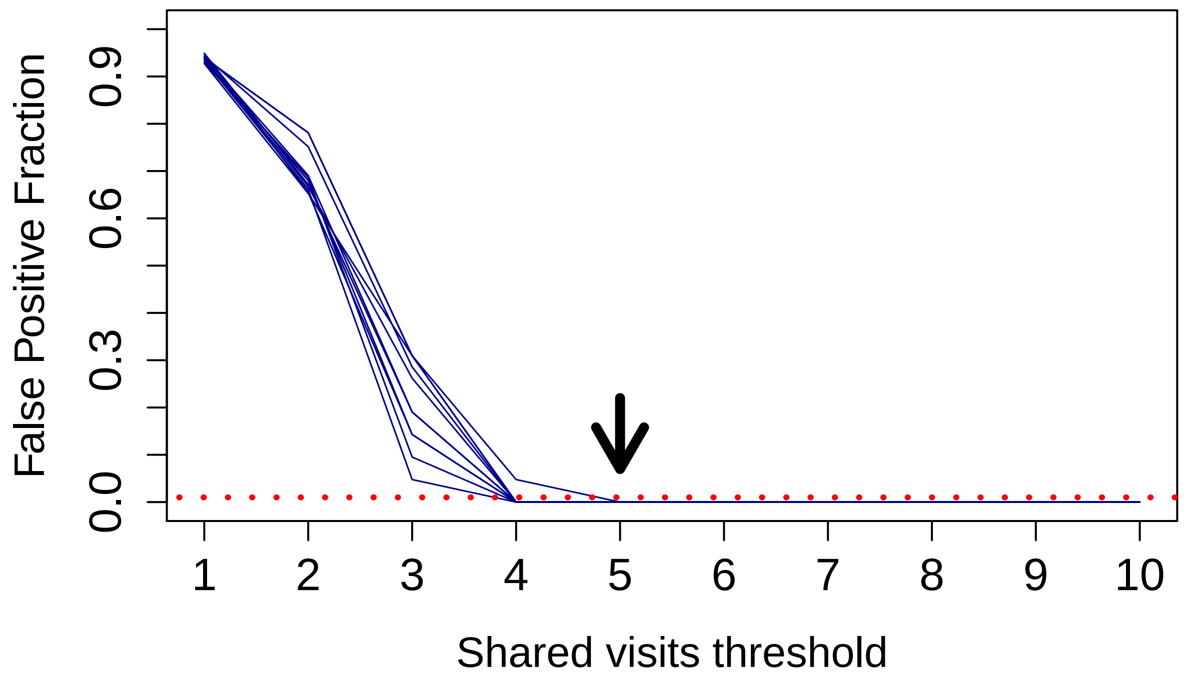
```
# Thresholds
for_thresholds<-do.call("rbind",Shuffling_simulation_output)

# Take max FP value for each position from all the simulations, one can take mean/median
for_thresholds_max<-apply(X = for_thresholds,2,max,na.rm=T)

# find lowest threshold below alpha 1%
Thresholds_lowest<-match(for_thresholds_max[for_thresholds_max<0.01][1],for_thresholds_max)
```

Plot the simulations and the the False-Positive threshold (indicated by an arrow)

```
plot(x=1:10,
     y=unlist(Shuffling_simulation_output[[1]]),xlab="Shared visits threshold",
     ylab="False Positive Fraction",
     type="l",col="darkblue",yaxp=c(0.0,1,10),lwd=0.8,main="",
     xaxp=c(1,12,11), cex.lab=1.3,cex.axis=1.4, cex.main=1.3, ylim=c(0,1))
arrows(x0 = Thresholds_lowest,x1=Thresholds_lowest,
       y0=0.22,y1=0.07, cex=3,lwd=5)
invisible(lapply(X=1:n_shuffl,
                 function(X){lines(x=1:10,y=unlist(Shuffling_simulation_output[[X]]),
                                   col="darkblue", lwd=0.8 )}))
lines(c(-100,100),0.01*c(1,1),lty=3,col="red",lwd=3)
```



```
Thresholds_lowest
```

```
## [1] 5
```

```
# This is the cutoff for the number of adjusted visits
```

Extract the pairs above the threshold

```
select_shuffled<-adjusted_observed[which(adjusted_observed$Corrected>Thresholds_lowest),]

# We found the 20 pairs
length(select_shuffled[,1])
```

```
## [1] 20
```

Alternative (and much faster approach) Predict the probabilities and than adjust for multiple testing using Bonferroni

```
# alpha 0.01, one false-positive pair
Bonferroni_m_output<-Bonferroni_m(unadjusted_observed_pairs,ids=ids,prob = 1/75,alpha =0.01)
length(Bonferroni_m_output[,1])
```

```
## [1] 21
```

```
# alpha 0.001, only 20 true pairs
Bonferroni_m_output<-Bonferroni_m(unadjusted_observed_pairs,ids=ids,prob = 1/75,alpha =0.001)
length(Bonferroni_m_output[,1])
```

```
## [1] 20
```