simid_workshop_tutorial_bootstrap.R

lwillem

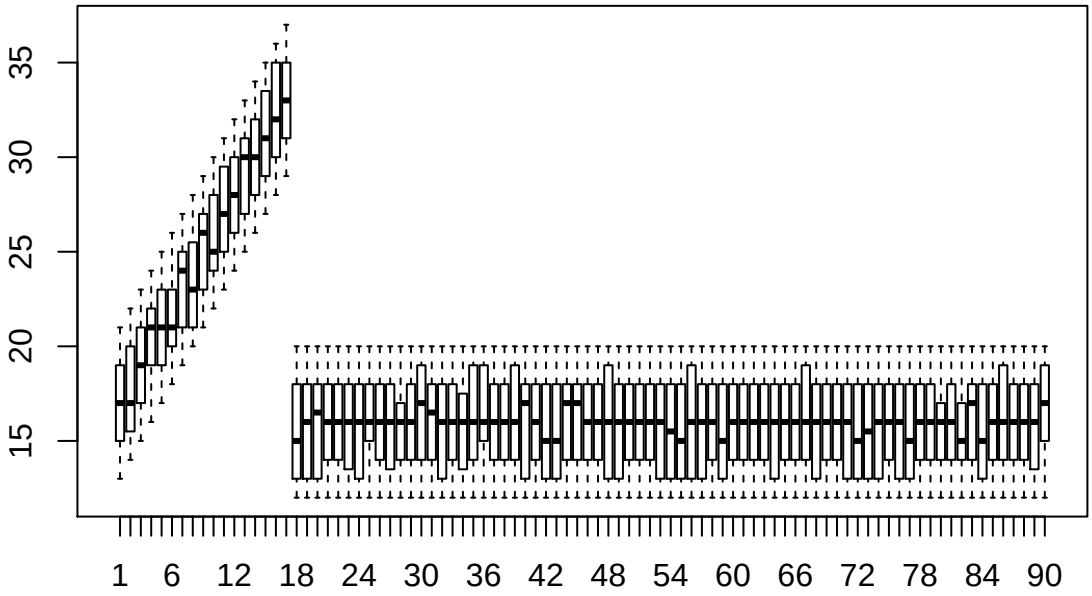
Tue Jun 11 14:09:08 2019

```
##############################################################################
# SIMID TUTORIAL: BOOTSTRAP & PARALLEL PROGRAMMING
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# Copyright (C) 2019 lwillem, SIMID, UNIVERSITY OF ANTWERP, BELGIUM
##############################################################################

# clear workspace
rm(list=ls())

# # set working directory (or open RStudio with this script)
# setwd("C:\User\path\to\the\rcode\folder") ## WINDOWS
# setwd("/Users/path/to/the/rcode/folder") ## MAC

# ####################
#  SETUP
# ####################
print('SETUP PARAMETERS: START')
```

```
## [1] "SETUP PARAMETERS: START"
```

```
# set random number generator
rng_seed <- 20190524
set.seed(rng_seed)

# e.g. population details
pop_size      <- 1e4
max_age       <- 90
adult_age     <- 18
gender_male   <- 'M'
gender_female <- 'F'
gender_opt    <- c(gender_male, gender_female)
zipcode_opt   <- c(1000,2160,2640,2018,2110,3000,3500,3520,3590,9000)
community_opt <- 1:30

# e.g. social contact details
mean_num_cnt  <- 16          # population average
dev_num_cnt   <- -4:4        # deviance
```

```r
# e.g. create population data
pop_data <- data.frame(id        = 1:pop_size,
                       age       = sample(1:max_age,pop_size,replace = T),
                       gender    = sample(gender_opt,pop_size,replace = T),
                       zipcode   = sample(zipcode_opt,pop_size,replace = T),
                       community = sample(community_opt,pop_size,replace = T))

# e.g. sample number of contacts, based on population average
pop_data$num_contacts <- mean_num_cnt + sample(dev_num_cnt,pop_size,replace=T)

# e.g. children make more contacts, relative to their age
is_child                          <- pop_data$age<adult_age
pop_data$num_contacts[is_child] <- pop_data$num_contacts[is_child] +
  pop_data$age[is_child]

# explore "number of contacts"
boxplot(num_contacts ~ age, data = pop_data)
```




```r
print('SETUP PARAMETERS: COMPLETE')
```

```
## [1] "SETUP PARAMETERS: COMPLETE"
```

```r
# ###################
#  BOOTSTRAP: FOR-LOOP
# ###################
print('BOOTSTRAPS: START')
```

```
## [1] "BOOTSTRAPS: START"
```

```r
num_bootstraps <- 500

# for-loop 1
for(i_bootstrap in 1:num_bootstraps){

  # set RGN seed per bootstrap, so the processing sequence does not affect the results
  set.seed(rng_seed + i_bootstrap)

  # bootstrap population
  pop_bootstrap        <- pop_data[sample(pop_size,replace=T),]

  # get mean age by community and gender
  summary_num_contacts <- aggregate(num_contacts ~ age + gender, data = pop_bootstrap, mean)

  # aggregte results: data.frame
  out_data_frame       <- data.frame(i_bootstrap,
                                     summary_num_contacts)

  # # plot results
  # pdf(file=paste0('output/bootstrap',i_bootstrap,'.pdf'))
  # plot(out_data_frame$num_contacts)
  # dev.off()
  #
  # # save results to txt file
  # write.table(out_data_frame,file=paste0('output/bootstrap',i_bootstrap,'.txt'),row.names=F,sep=',')
  #
  # # save results to RData file
  # save(out_data_frame,file=paste0('output/bootstrap',i_bootstrap,'.RData'))
}

# retrieve data etc
# ....

print('BOOTSTRAPS: COMPLETE')
```

```
## [1] "BOOTSTRAPS: COMPLETE"
```

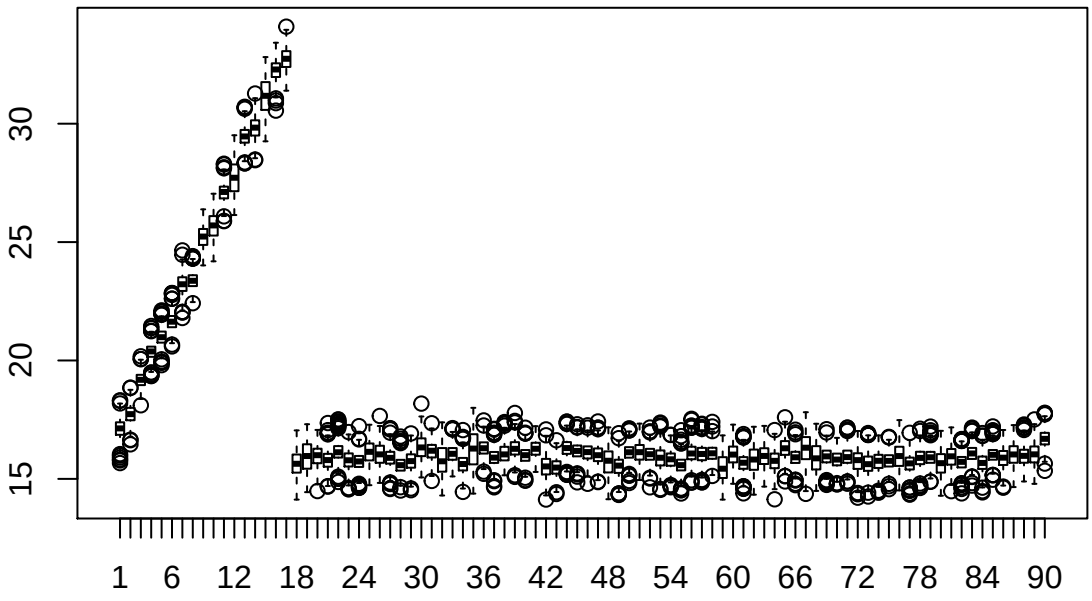
```
# ###################
#  SEQUENTIAL FOREACH
# ###################
print('SEQUENTIAL FOREACH: START')
```

```
## [1] "SEQUENTIAL FOREACH: START"
```

```
library('foreach')

print(system.time( # start manual profiling
# for-loop 2: return values
bootstrap_out <- foreach(i_bootstrap = 1:num_bootstraps,
                         .combine      = 'rbind') %do%{

  # set RGN seed per bootstrap, so the processing sequence does not affect the results
  set.seed(rng_seed + i_bootstrap)

  # bootstrap population
  pop_bootstrap        <- pop_data[sample(pop_size,replace=T),]

  # get mean age by community and gender
  summary_num_contacts <- aggregate(num_contacts ~ age + gender, data = pop_bootstrap, mean)

  # aggregte results: data.frame
  out_data_frame       <- data.frame(i_bootstrap,
                                     summary_num_contacts)

  # return results
 return(out_data_frame)
                         }
)) # end system.time()
```

```
##    user  system elapsed
##   5.999   1.061   7.067
```

```
# plot bootstrap results
boxplot(num_contacts ~  age, data=bootstrap_out)
```




```
print('BOOTSTRAPS: COMPLETE')
```

```
## [1] "BOOTSTRAPS: COMPLETE"
```

```r
# ###################
#  PARALLEL FOREACH
# ###################
print('PARALLEL FOREACH')
```

```
## [1] "PARALLEL FOREACH"
```

```r
# load package 'devtools'
library(devtools)

# install (and load) package "simid_rtools" from https://github.com/lwillem/simid_rtools
devtools::install_github("lwillem/simid_rtools",force=F,quiet=T)

# start parallel working nodes
# note: make sure you loaded all required user defined functions at this point
simid.rtools::smd_start_cluster()
```

```
## $par_cluster
## socket cluster with 8 nodes on host 'localhost'
##
## $pid_master
## [1] 71818
##
## $pid_slave1
## [1] 71869
```

```r
print(system.time( # manual profiling
# run loop in parallel
bootstrap_out <- foreach(i_bootstrap = 1:num_bootstraps,
                         .combine      = 'rbind') %dopar%{
   # set RGN seed per bootstrap, so the processing sequence does not affect the results
   set.seed(rng_seed + i_bootstrap)

   # bootstrap population
   pop_bootstrap        <- pop_data[sample(pop_size,replace=T),]

   # get mean age by community and gender
   summary_num_contacts <- aggregate(num_contacts ~ age + gender, data = pop_bootstrap, mean)

   # aggregte results: data.frame
   out_data_frame       <- data.frame(i_bootstrap,
                                      summary_num_contacts)
   # return results
   return(out_data_frame)
  }
)) # end system.time()
```

```
##    user  system elapsed
##   0.388   0.139   1.881
```

```r
# terminate parallel cluster
simid.rtools::smd_stop_cluster()

print('PARALLEL FOREACH: COMPLETE')
```

```
## [1] "PARALLEL FOREACH: COMPLETE"
```