parallel_foreach_tutorial.R

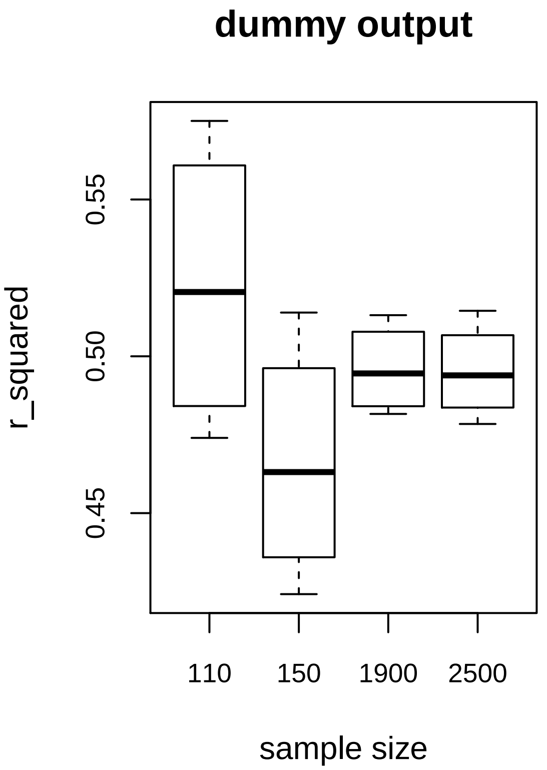
lwillem

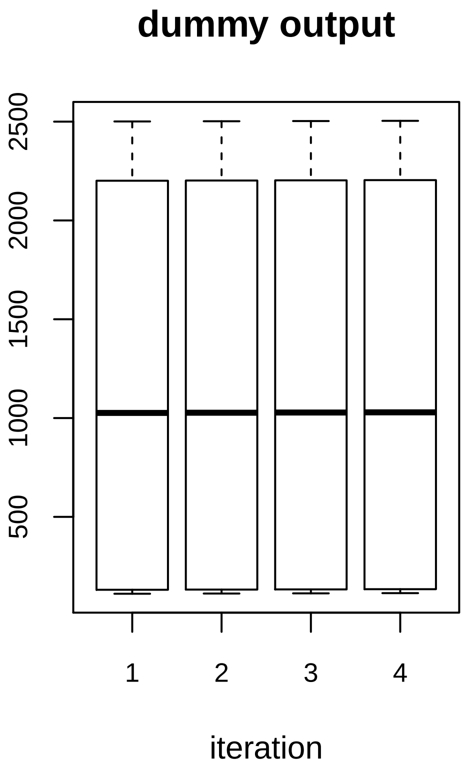
Fri Mar 8 15:00:55 2019

```
#############################################################################
# SIMID TUTORIAL: PARALLEL FOREACH
#
# Default functions:
#   - expand.grid         to get all combinations of given parameters
#   - foreach             to loop over a set (similar to 'for')
#
# New functions form the simid.rtools package
#   - smd_load_packages   to load (and install) packages
#   - smd_start_cluster   to start mutiple local processing nodes
#   - smd_stop_cluster    to end local work nodes
#   - smd_progress        to print the progress during the parallel loop
#
# 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())

# uncomment the following lines to install the simid.rtools package from github
#install.packages('devtools')
#library(devtools)
#devtools::install_github("lwillem/simid_rtools",force=F,quiet=T)

# load the package
library('simid.rtools')

# option: load packages (and install them if required)
smd_load_packages('scales')

# create a user defined (dummy) function
smd_sum <- function(x){
  return(sum(x))
}

# set up an experimental design
exp_design <- expand.grid( num_iter    = seq(4),                 # number of iterations
                           sample_size = c(110,150,1900,2500))   # sample size
# inspect dimensions
dim(exp_design)
```

```
## [1] 16  2
```

```r
# start parallel working nodes
# note: make sure you loaded all required user defined functions at this point
par_nodes_info <- smd_start_cluster()

# print message to user, and store the current time (for the progress report)
smd_print('GRIDSEARCH...'); time_stamp <- Sys.time()

# run all experiments from the design, and store results as 'exp_results'
exp_results <- foreach(i_exp     = 1:nrow(exp_design),  # all experiments
                       .combine  ='rbind') %dopar% {    # combine the output by row

   # print progress (only the first work-node)
   # note: the parallel environment requires to specify that we need the
   # 'smd_progress' function from the 'simid.rtools' package
   simid.rtools::smd_progress(i_exp,nrow(exp_design),time_stamp,par_nodes_info)

   # some dummy operations...
   x_sample <- sample(seq(1,1000),exp_design$sample_size[i_exp],replace = T)
   y_sample <- sample(seq(50,75),exp_design$sample_size[i_exp],replace = T)
   z_sample <- x_sample + y_sample + sample(x_sample) # add random effect by shuffling 'x'

   # some fitting
   lm_model   <- lm(z_sample ~ x_sample + y_sample)
   lm_summary <- summary(lm_model)

   # user defined function
   user_sum <- smd_sum(exp_design[i_exp,])

   # return loop index, parameters and results
   data.frame(id_iter      = i_exp,                 # run id
              exp_design[i_exp,],                   # input param
              r_squared    = lm_summary$r.squared,  # output...
              user_sum     = user_sum)              # output...
}

# terminate parallel cluster
smd_stop_cluster()

# explore output
par(mfrow=1:2)
boxplot(r_squared ~ sample_size,data=exp_results,
        xlab='sample size',ylab='r_squared',
        main='dummy output',cex.axis=0.8)

boxplot(user_sum ~ num_iter, data=exp_results,
        xlab='iteration',main='dummy output',cex.axis=0.8)
```


dummy output
r_squared
0.45
0.50
0.55
110
150
1900
2500
sample size


dummy output
500
1000
1500
2000
2500
1
2
3
4
iteration