Final Exam, Question 1b

Stats 506, Fall 2019

12/16/2019

```
# Final Exam, Question 1b
# Stats 506, Fall 2019
#
# Updated: December 15, 2019
# Author: James Henderson
# 80: --------------------------------------------------------------------------

# libraries: -------------------------------------------------------------------
library(tidyverse)

# CIs using bootstrap methods: -------------------------------------------------
confint_boot_mc = function(x, y, R = 1e2, level = 0.95, debug = FALSE) {
  # Function to compute
  # Inputs:
  #   x, y - numeric matrices, with each *column* representing a single Monte
  #          Carlo sample. They need not have the same number of columns.
  #  level - the desired confidence level, defaults to 95%
  #      R - the number of bootstrap replicates to use
  #  debug - if true, tests are performed to check intermediate output

  # Returns:
  #   estimated 95% CI ofs the estimators mean(x[i, ]) / mean(y[i, ])
  #   for all rows i.
  #
  # Details: this function returns three types of bootstrap CIs constructed
  #  using (1) the precentile method, (2) the basic bootstrap, and (3) a normal
  #  approximation using the bootstrap estimator for the standard devation.

  # form the bootstrap samples
  xind = sample( 1:nrow(x), size = R * ncol(x) * nrow(x), replace = TRUE )
  yind = sample( 1:nrow(y), size = R * ncol(y) * nrow(y), replace = TRUE )

  # to index column j, we add nrow(x) * {j - 1} to each column "j" of indices
  # which occurs every R * n samples
  xind = xind + rep( nrow(x) * {1:ncol(x) - 1L}, each = R * nrow(x))
  yind = yind + rep( nrow(y) * {1:ncol(y) - 1L}, each = R * nrow(y))

  # the results are n x R x mcrep
  xb = x[xind]
  yb = y[yind]
  return( sapply(ls(), function(x) pryr::object_size(get(x))) )
  rm(xind, yind)

  dim(xb) = c(nrow(x), R, ncol(x))
  dim(yb) = c(nrow(y), R, ncol(y))
```

```r
  # Check that we have retained the structure
  if ( debug ) {
    stopifnot( all( xb[, , 1]  %in% x[, 1]) )
    stopifnot( all( yb[, , ncol(y)] %in% y[, ncol(y)] ) )
  }

  # compute the boostrap statistics for each MC sample
  xbar =  colMeans(xb, dims = 1)
  ybar =  colMeans(yb, dims = 1)
  rm(xb, yb)

  # check the lengths here
  if ( debug ) {
    stopifnot( dim(xbar)[2] == ncol(x) && dim(ybar)[1] == R)
  }

  # compute the ratios
  theta = xbar / ybar
  # compute the quantiles, ok to use a loop here
  m = c( {1 - level} / 2, 1 - {1 - level} / 2)
  theta_q = apply(theta, 2, quantile, probs = m)

  # compute the bootstrap std errors
  se_boot = sqrt( R * colMeans( {theta - colMeans(theta)}^2 ) / { R - 1} )

  # point estimates
  est = colMeans(x) / colMeans(y)

  # final bootstrap CI's
  tibble(
    method = rep(
      c('percentile bootstrap', 'basic bootstrap', 'normal bootstrap'),
      each = ncol(x)),
    estimate = rep(est, 3),
    lower = c(theta_q[1, ], 2 * est - theta_q[2, ], est + qnorm(m[1])*se_boot),
    upper = c(theta_q[2, ], 2 * est - theta_q[1, ], est + qnorm(m[2])*se_boot)
  )
}

nx = 30; mux = 1
ny = 20; muy = 2
theta =  muy / mux # target ratio, mux and muy are actually rate not mean (1/r)

# Monte Carlo sampling in blocks, to avoid allocating too long vectors
mcrep = 1e3 # previously per block * 10 blocks, but used once here.

x = matrix(rexp( mcrep * nx, mux), mcrep, nx)
y = matrix(rexp( mcrep * ny, muy), mcrep, ny)

result_boot = confint_boot_mc(t(x), t(y), R = 1e4)
```