```
##################################################################################
## Function simulatoR                                                 ##
##                                                                    ##
## Author: Hannes Feilhauer with contributions by Frank Thonfeld             ##
## Contact: hannes.feilhauer@geographie.uni-erlangen.de                    ##
## Version: 2012/07/25                                                ##
## License: GNU General Public License                                      ##
## Requires: raster, rgdal                                            ##
##                                                                    ##
## This function simulates signals of multispectral sensors based            ##
## on hyperspectral data (image or spectral library) and the spectral         ##
## response curves of the target sensor.                                   ##
##                                                                    ##
## SimulatoR() calls simspec() for simulations based on spectral libraries    ##
## (matrix) or simrast() for imagery (raster data).                         ##
##                                                                    ##
## Arguments:                                                         ##
## x     hyperspectral dataset, either a matrix (rows=spectra,                ##
##       columns=bands) or a raster stack/brick                            ##
## x.res vector with the wavelengths (nm) of the band centers of x            ##
## x.rad radiometric resolution of x (bit), 0=reflectance scaled 0 to 1       ##
## y     spectral response curves of the output sensor                       ##
## y.res vector with the wavelengths corresponding to the values of y         ##
## y.rad radiometric resolution of y (bit)                                  ##
## tile  logical, should raster x be processed in tiles? Tile processing      ##
##       increases the computation time considerably but enables processing   ##
##       of large datasets.                                            ##
## size  tile-size in pixels, required if tile=T                            ##
##                                                                    ##
##################################################################################

simulatoR <- function (x, x.res, x.rad, y, y.res, y.rad, tile=F, size=200) {

 require (raster)

 ## check input data
 if (is.matrix (x) == FALSE & class (x) != "RasterStack" & class (x) !=
 "RasterBrick")
 stop ("invalid input dataset")

 nx <- dim (x)[length (dim (x))]

 if (is.vector (x.res) == FALSE | length (x.res) != nx)
        stop ("invalid x.res")

 if (is.matrix (y) == FALSE)
        stop ("invalid spectral response curves")

 ny <- dim (y)[1]

 if (is.vector (y.res) == FALSE | length (y.res) != ny)
        stop ("invalid y.res")

 ## check spectral matching

 if (min (x.res) > min (y.res) | max (x.res) < max (y.res))
        print (
"Warning: Input data do not meet spectral requirements. Results may be invalid")
```

```
######################## prepare spectral response curves ########################

 ## interpolate the spectral response curves to 1 nm

 interpol <- function (y, y.res, z) {
        approx (y.res, y, xout=z)$y
 }

 z <- c (floor (min (y.res)):ceiling (max (y.res)))
 y <- apply (y, 2, interpol, y.res, z)

 ## convert to weight matrix

 y <- t (t (y) / apply (y, 2, sum))

####################################################################################
##                          Function simspecs                              ##
####################################################################################

simspecs <- function (x, x.res, y) {

 specsim <- function (s) {
        intspec <- approx (x.res, s, xout=z)$y
        apply (intspec * y, 2, sum, na.rm=TRUE)
        }
 t (apply (x, 1, specsim))
 }

########################## end of function simspecs ##############################

######################### simulate individual spectra ############################

 if (is.matrix (x) == T) {
        output <- simspecs (x, x.res, y)
        rownames (output) <- rownames (x)
        if (x.rad != 0)
        output <- output / 2^x.rad
        output <- round (output * 2^y.rad, 0)
        }

####################################################################################
##                          Function simrast                              ##
####################################################################################

simrast <- function (r, x.res, y) {

 ## prepare z-profiles

 pixsim <- function (r) {
        pixspec <- approx (x.res, r, xout=z)$y
        apply (pixspec * y, 2, sum, na.rm=TRUE)
        }

 ## apply pixsim onto the raster

 result <- aperm (apply (r, c (1,2), pixsim), c (2, 3, 1))

 if (x.rad !=0)
 result <- result / 2^x.rad
```

```
 round (result * 2^y.rad, 0)
 }

############################# end of function simrast ############################

############################## simulate raster ####################################

 if (class (x) == "RasterStack" | class (x) == "RasterBrick") {

        ######################### without tile processing ########################

                if (tile==FALSE) {
                 r <- as.array (x)
                 result <- simrast (r, x.res, y)

                 output <- raster (as.matrix (result [,,1]))
                        extent (output) <- extent (x)
                 for (i in 2:dim (result)[3]) {
                        b <- raster (as.matrix (result [,,i]))
                        extent (b) <- extent (x)
                        output <- addLayer (output, b)
                        }}

        ######################### with tile processing ###########################

                if (tile==TRUE) {
                 xcoords <- floor (seq (xmin (x), xmax (x), length.out=
                         ceiling (dim (x)[2] / size))) ## tile corners
                 ycoords <- floor (seq (ymin (x), ymax (x), length.out=
                         ceiling (dim (x)[1] / size))) ## tile corners

                        ## progress bar

                        ntiles <- (length (xcoords)-1) * (length (ycoords)-1)
                        pbval <- matrix (1:ntiles, length (ycoords)-1, length
                                (xcoords)-1)
                 pb <- txtProgressBar (min = 0, max = ntiles, style=3)
                        setTxtProgressBar (pb, value=0)

                        output <- NULL ## tile storage

                 ########################### process tiles ######################

                 for (j in 1:(length (xcoords)-1)) {
                         for (k in 1:(length (ycoords)-1)) {
                         xt <- crop (x, extent (xcoords[j], xcoords[j+1],
                                ycoords[k], ycoords[k+1])) ## crop tile
                         r <- as.array (xt) ## convert tile to array
                         result <- simrast (r, x.res, y) ## simulate tile

                         ## convert to raster stack

                         ot <- raster (as.matrix (result [,,1]))
                                extent (ot) <- extent (xt)
                         for (i in 2:dim (result)[3]) {
                                b <- raster (as.matrix (result [,,i]))
                                extent (b) <- extent (xt)
                                ot <- addLayer (ot, b)
                                }
```

```
                output <- c (output, ot)

                setTxtProgressBar(pb, value=pbval[k,j])

                }
               } ## end process tiles
          output <- do.call(merge, output) ## merge tiles
          }

         ## end simulate raster with tile processing

         }

 output

}

########################### end of function simulatoR ########################



## -------------------------------- Example -------------------------------- ##

## Exemplary simulation of Landsat 5 TM spectral signals based on spectra from
## the USGS spectral library 06 (http://speclab.cr.usgs.gov/spectral-lib.html).
## The LS5 spectral response curves are taken from http://calvalportal.ceos.org.
## Both online-sources were accessed on July 25th, 2012. HF and FT are not
## responsible for the content of these third-party pages.

## To execute this example, copy the code below without hashes into the R
## console. URLs to the online repositories must be assembled without spaces.

## ------------------------------------------------------------------------- ##
## download spectral response curves of the Landsat sensors
## ------------------------------------------------------------------------- ##

## path <- getwd ()

## download.file ("http://calvalportal.ceos.org/cvp/c/document_library/
## get_file?uuid=0a3eb756-f142-43a9-af31-ebf215f4f37e&groupId=10136",
## paste (path, "/b1.txt", sep=""))

## download.file ("http://calvalportal.ceos.org/cvp/c/document_library/
## get_file?uuid=97b30449-344e-4820-b4d3-e47796368129&groupId=10136",
## paste (path, "/b2.txt", sep=""))

## download.file ("http://calvalportal.ceos.org/cvp/c/document_library/
## get_file?uuid=35b45116-fe3b-4135-a25f-5f2e9c32c0a2&groupId=10136",
## paste (path, "/b3.txt", sep=""))

## download.file ("http://calvalportal.ceos.org/cvp/c/document_library/
## get_file?uuid=9ccbaf5e-e400-48fe-911e-a6c4d6871344&groupId=10136",
## paste (path, "/b4.txt", sep=""))

## download.file ("http://calvalportal.ceos.org/cvp/c/document_library/
## get_file?uuid=4f6c8877-10db-499c-b7e3-ec95b66f53a5&groupId=10136",
## paste (path, "/b5.txt", sep=""))

## download.file ("http://calvalportal.ceos.org/cvp/c/document_library/
```

```
## get_file?uuid=cc0b6e5d-d0cb-4dfb-8f91-2b2825dfda5b&groupId=10136",
## paste (path, "/b7.txt", sep=""))

## ------------------------------------------------------------------------ ##
## load the spectral response curves into the workspace, delete downloaded files
## ------------------------------------------------------------------------ ##

## b1 <- read.table ("b1.txt", header=T, skip=2)
## b2 <- read.table ("b2.txt", header=T, skip=2)
## b3 <- read.table ("b3.txt", header=T, skip=2)
## b4 <- read.table ("b4.txt", header=T, skip=2)
## b5 <- read.table ("b5.txt", header=T, skip=2)
## b7 <- read.table ("b7.txt", header=T, skip=2)

## file.remove (c ("b1.txt","b2.txt", "b3.txt", "b4.txt", "b5.txt", "b7.txt"))

## ------------------------------------------------------------------------ ##
## prepare the LS5 srcs for the simulatoR
## ------------------------------------------------------------------------ ##
## wavelength <- c (410:2400)

## b1 <- approx (b1[,1], b1[,4], wavelength)$y
## b2 <- approx (b2[,1], b2[,4], wavelength)$y
## b3 <- approx (b3[,1], b3[,4], wavelength)$y
## b4 <- approx (b4[,1], b4[,4], wavelength)$y
## b5 <- approx (b5[,1], b5[,4], wavelength)$y
## b7 <- approx (b7[,1], b7[,4], wavelength)$y

## src <- cbind (b1, b2, b3, b4, b5, b7)
##  src [is.na (src)] <- 0

## ------------------------------------------------------------------------ ##
## download 3 exemplary vegetation spectra from the USGS spectral library 06
## ------------------------------------------------------------------------ ##

## download.file ("http://speclab.cr.usgs.gov/spectral.lib06/ds231/ASCII/V/
## aspen_aspen-1.29568.asc", "aspen.txt")

## download.file ("http://speclab.cr.usgs.gov/spectral.lib06/ds231/ASCII/V/
## cactus_opuntia-1.29715.asc", "opuntia.txt")

## download.file ("http://speclab.cr.usgs.gov/spectral.lib06/ds231/ASCII/V/
## golden_dry_grass.gds480.30093.asc", "drygrass.txt")

## ------------------------------------------------------------------------ ##
## load the spectra into the workspace and delete downloaded files
## ------------------------------------------------------------------------ ##

## spec.res <- read.table ("aspen.txt", header=F, skip=16)[,1] * 1000

## aspen <- read.table ("aspen.txt", header=F, skip=16)[,2]
## opuntia <- read.table ("opuntia.txt", header=F, skip=16)[,2]
## drygrass <- read.table ("drygrass.txt", header=F, skip=16)[,2]

## file.remove (c ("aspen.txt","opuntia.txt", "drygrass.txt"))

## spectra <- rbind (aspen, opuntia, drygrass)
##   spectra [spectra < 0] <- 0
```

```
## -------------------------------------------------------------------------- ##
## apply the simulatoR
## -------------------------------------------------------------------------- ##

## pseudoreflectance <- simulatoR (spectra, x.res=spec.res, x.rad=0, y=src,
## y.res=wavelength, y.rad=8)

## print (pseudoreflectance)

## ---------------------------------- End ---------------------------------- ##
```