Appendix S2 - The parametric Rao's Q variation over the planet

From zero to infinity: minimum to maximum diversity of the planet by spatio-parametric Rao's quadratic entropy

February 7, 2021

We applied the algorithm to a Copernicus Proba-V NDVI (Normalized Difference Vegetation Index) long term average image (June 21st 1999-2017) at 5 km grain, also provided in the `rasterdiv` package as a free `Rasterlayer` dataset which can be loaded by the function `data()` (Figure S1). The parametric Rao's Q algorithm can also be applied to multispectral data; in such a case distances are calculated in the multisystem created by the values of the pixels in each axis/band. The moving window passing throughout the whole image will return M_{Q_α} matrices/layers where α is the value chosen in the R function `paRao`.

With $\alpha \to 0$ the $\prod$ in Equation 6 (in the main text of the manuscript) leads to zeroes throughout the whole map (Figure S2). Increasing α will increase the weight of higher distances among different values until reaching the maximum distance value for $\alpha \to \infty$. In this case the maximum turnover is reached and areas with maximum β-diversity will be apparent. In this case, a multitemporal set is used (long term average NDVI from June 21st 1999-2017). Hence, areas with the highest spatial and temporal turnover are enhanced, namely major mountain ridges. We expect that using single frame images would lead to the enhancement of the spatial component of diversity.

Since the whole process is based on distances in a spectral space between pairs of pixels in terms of their "spectral characters" or in the "spectral space", it is important to notice some cornerstone aspects on the use of

distances from satellite images, especially when comparing different images or the same image in different times. In satellite images, the measure of distances could be impacted by: i) the use of different sensors with different radiometric resolutions, as an example an 8-bit ($2^8 = 256$ values) with respect to a 16-bit ($2^{16} = 65536$ values) image, or ii) the radiometric calibration which has been performed, e.g. with a non-linear transform. Therefore, care should be taken when making use of distances in remote sensing data, explicitly taking into account how the vector of proportions between pixels belonging to some defined classes (e.g., digital numbers, DNs) was obtained.

The complete code of the function can be directly seen in R by typing the `paRao` function name. Moreover, a complete R coding session, to perform the above described analysis is provided at the end of this Appendix.

1 Figures

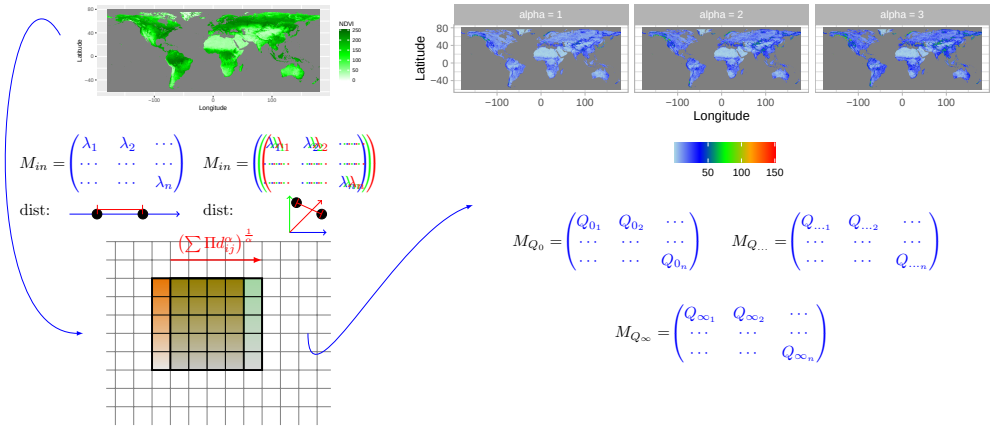

Figure S1. Starting from Copernicus Proba-V NDVI (Normalized Difference Vegetation Index) long term average image (June 21st 1999-2017) at 5km grain, parametric Rao's Q is calculated in a moving window. In this paper NDVI was used as a single layer to calculate distances on one axis, but several layers can be used as well. In this example, three layers (blue, green and red matrices) are shown to calculate distances. The algorithm is based on a moving window passing throughout the whole image, calculating the Rao's Q_α and saving the output in the central pixel. In this example a moving window of 5x5 pixels is passing (red arrow) from one position (orange) to the other (green). The output is a stack of layers each of which represents

a different mean of the whole generalized mean spectrum of Equation 5 (in the main text of the paper).

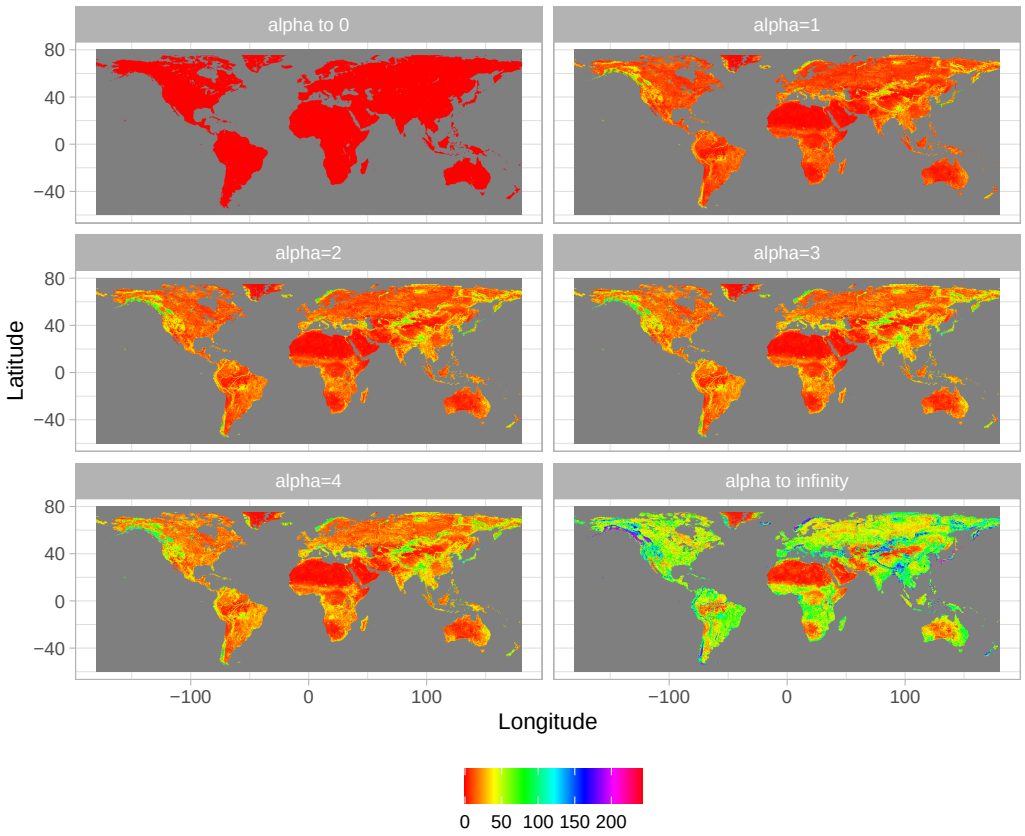

Figure S2. Output of the application of the algorithm shown in Figure S1, achieved by applying different α values: from 0 to 4 until $\alpha \to \infty$. The higher the value of the parameter α, the higher the weight of highest distances among pixel values, until reaching the maximum potential β-diversity (maximum distance) at $\alpha \to \infty$.

2 Code

2.1 paRao function

```
function (x, dist_m = "euclidean", window = 9, alpha = 1,
    method = "classic",
     rasterOut = TRUE, lambda = 0, na.tolerance = 0, rescale =
     FALSE,
     diag = TRUE, simplify = 3, np = 1, cluster.type = "SOCK",
     debugging = FALSE)
{
     is.wholenumber <- function(x, tol = .Machine$double.eps
    ^0.5) abs(x -
         round(x)) < tol
     if (!(is(x, "matrix") | is(x, "SpatialGridDataFrame") |
    is(x,
         "RasterLayer") | is(x, "list"))) {
         stop("\nNot a valid x object.")
     }
     if (is(x, "SpatialGridDataFrame")) {
         x <- raster(x)
     }
     else if (is(x, "matrix") | is(x, "RasterLayer")) {
         rasterm <- x
     }
     else if (is(x, "list")) {
         rasterm <- x[[1]]
     }
     if (na.tolerance > 1 | na.tolerance < 0) {
         stop("na.tolerance must be in the [0-1] interval.
    Exiting...")
     }
     if (any(!is.numeric(alpha))) {
         stop("alpha must be a numeric vector. Exiting...")
     }
     if (any(alpha < 0)) {
         stop("alphas must be only positive numbers. Exiting
    ...")
     }
     if (method == "classic" & is(x, "RasterLayer")) {
         isfloat <- FALSE
         if (!is.wholenumber(rasterm@data@min) | !is.
    wholenumber(rasterm@data@max) |
             is.infinite(rasterm@data@min) | !is.wholenumber(
    median(getValues(rasterm),
             na.rm = T))) {
```

```
            message("Input data are float numbers. Converting
 x data in an integer matrix...")
            isfloat <- TRUE
            mfactor <- 100^simplify
            rasterm <- getValues(rasterm) * mfactor
            rasterm <- as.integer(rasterm)
            rasterm <- matrix(rasterm, nrow(x), ncol(x),
byrow = TRUE)
            gc()
        }
        else {
            rasterm <- matrix(getValues(rasterm), ncol = ncol
(x),
                nrow = nrow(x), byrow = TRUE)
        }
        message("Matrix check OK: \nParametric Rao output
matrix will be returned")
    }
    else if (method == "classic" & (is(x, "matrix") | is(x, "
list"))) {
        isfloat <- FALSE
        if (!is.integer(rasterm)) {
            message("Input data are float numbers. Converting
 x in an integer matrix...")
            isfloat <- TRUE
            mfactor <- 100^simplify
            rasterm <- as.integer(rasterm * mfactor)
            rasterm <- matrix(rasterm, nrow(x), ncol(x),
byrow = TRUE)
            gc()
        }
        else {
            rasterm <- as.matrix(rasterm)
        }
        message("Matrix check OK: \nParametric Rao output
matrix will be returned")
    }
    else ("The class of x is not recognized. Exiting...")
    if (window%%2 == 1) {
        w <- (window - 1)/2
    }
    else {
        stop("The size of the moving window must be an odd
number. Exiting...")
    }
    if (np == 1) {
        if (method == "classic") {
            out <- lapply(alpha, paRaoS, rasterm = rasterm, w
 = w,
```

```
                dist_m = dist_m, na.tolerance = na.tolerance,
                diag = diag, debugging = debugging, isfloat =
 isfloat,
                mfactor = mfactor)
        }
        else if (method == "multidimension") {
            out <- lapply(alpha, mpaRaoS, x = x, rasterm =
rasterm,
                w = w, dist_m = dist_m, na.tolerance = na.
tolerance,
                rescale = rescale, lambda = lambda, diag =
diag,
                debugging = debugging)
        }
        if (rasterOut == T & class(x) == "RasterLayer") {
            outR <- lapply(out, raster, template = x)
            return(outR)
        }
        else {
            return(out)
        }
    }
    else if (np > 1) {
        if (method == "multidimension") {
            stop("Multidimensional paRao not yet implemented,
 set 'np=1'. Exiting...")
        }
        else {
            message("\n###################### Starting
parallel calculation ########################")
            if (debugging) {
                cat("#check: Before parallel function.")
            }
            if (cluster.type == "SOCK" || cluster.type == "
FORK") {
                cls <- makeCluster(np, type = cluster.type,
outfile = "",
                  useXDR = FALSE, methods = FALSE, output = "
")
            }
            else if (cluster.type == "MPI") {
                cls <- makeCluster(np, outfile = "", useXDR =
 FALSE,
                  methods = FALSE, output = "")
            }
            else {
                message("Wrong definition for 'cluster.type'.
 Exiting...")
            }
```

```
            doParallel::registerDoParallel(cls)
            on.exit(stopCluster(cls))
            gc()
            out <- lapply(alpha, paRaoP, rasterm = rasterm, w = w,
                dist_m = dist_m, na.tolerance = na.tolerance,
                diag = diag, debugging = debugging, isfloat = isfloat,
                mfactor = mfactor)
            if (rasterOut == T & class(x) == "RasterLayer") {
                outR <- lapply(out, raster, template = x)
                return(outR)
            }
            else {
                return(out)
            }
        }
    }
}
```

2.2 Application of the paRao function to a synthetic set

```
# install rastediv
install.packages("rasterdiv")

library(raster)
library(rasterdiv)

# generate matrix
synth <- raster(ncol = 8, nrow = 8, xmn = 1, xmx = 6, ymn = 1, ymx = 6)
values(synth) <- rpois(ncell(synth), lambda=3)

# paRao function, using the code in the manuscript
synth.parao <- paRao(synth, alpha = c(0:4,30^9), dist_m = "euclidean", window = 9, na.tolerance = 0.5, simplify = 3, diag = T, rasterOut = T)
```

2.3 Application of the paRao function to the 8bit copNDVI dataset

```
library(rasterdiv)

st <- paRao(copNDVI, alpha = c(0:4,Inf), dist_m = "euclidean", window = 9, na.tolerance = 0.5, simplify = 3, diag = TRUE, rasterOut = TRUE)
```

2.4 Output plot

```
library(raster)
library(ggplot2)
library(rasterVis)
library(RColorBrewer)

var.labs=c("layer.1" = "alpha to 0", "layer.2" = "alpha=1", "
   layer.3" = "alpha=2", "layer.4" = "alpha=3", "layer.5" = "
   alpha=4", "layer.6" = "alpha to infinity")

gplot(st, maxpixels=500000) +
  geom_raster(aes(fill = value),  color = "black") +
  labs(x="Longitude",y="Latitude", fill="")+
  scale_fill_gradientn(colors=rainbow(100)) +
  coord_equal()+
  theme_light()+
  facet_wrap(~ variable, ncol = 2,  labeller = labeller(
   variable = var.labs))+
  theme(legend.position = "bottom") +
  NULL
```