If you plan to do anything with the raster package you should definitely consider parallelize all your processes, especially if you are working with very large image files. I couldn’t find any blog post describing how to parallelize with the [raster package](https://cran.r-project.org/web/packages/raster/) (it is well documented in the package documentation, though). So here my notes.

**Load some example data**

Let’s first get some raster data from [here](http://www.francescobailo.net/2019/01/are-you-parallelizing-your-raster-operations-you-should/), any file will do but I’m using the Cambodian population data for 2015 (KHM\_ppp\_v2b\_2015\_UNadj).

library(raster)

khm\_pop.r <-

raster("~/Downloads/KHM\_ppp\_v2b\_2015\_UNadj/KHM\_ppp\_v2b\_2015\_UNadj.tif")

We can plot it with

library(rasterVis)

library(viridis)

library(ggplot2)

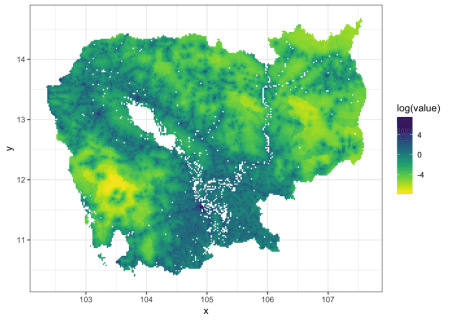
rasterVis::gplot(khm\_pop.r) +

geom\_tile(aes(fill = log(value))) +

viridis::scale\_fill\_viridis(direction = -1,

na.value='#FFFFFF00') +

theme\_bw()

[](https://i0.wp.com/www.francescobailo.net/wordpress/wp-content/uploads/2019/01/unnamed-chunk-2-1.png)

**Projection**

Now, let’s first project the raster without any parallelization.

start\_time <- Sys.time()

res1 <-

projectRaster(khm\_pop.r,

crs = '+proj=utm +zone=48 +datum=WGS84 +units=m +no\_defs')

end\_time <- Sys.time()

end\_time - start\_time

## Time difference of 1.088329 mins

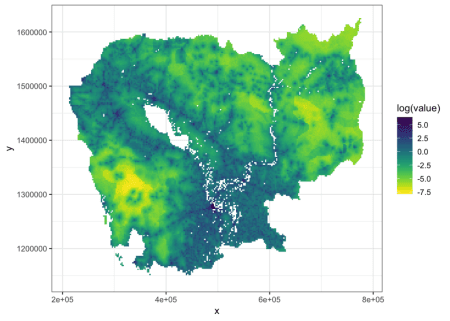
rasterVis::gplot(res1) +

geom\_tile(aes(fill = log(value))) +

viridis::scale\_fill\_viridis(direction = -1,

na.value='#FFFFFF00') +

theme\_bw()

[](https://i1.wp.com/www.francescobailo.net/wordpress/wp-content/uploads/2019/01/unnamed-chunk-4-1.png)

And now let’s parallelize the process. There are two approaches to parallelization with raster objects (do ?clusterR for the documentation of the package mantainers):

1. By including the raster function between a beginCluster() and an endCluster().
2. By using clusterR() like in clusterR(x, fun, args=NULL, cl=mycluster), where mycluster is a cluster object generated for example by getCluster().

Yet clusterR() doesn’t work with merge, crop, mosaic, disaggregate, aggregate, resample, projectRaster, focal, distance, buffer and direction.

Let’s try the first approach first.

start\_time <- Sys.time()

beginCluster()

## 4 cores detected, using 3

res2 <-

projectRaster(khm\_pop.r,

crs = '+proj=utm +zone=48 +datum=WGS84 +units=m +no\_defs')

## Using cluster with 3 nodes

endCluster()

end\_time <- Sys.time()

end\_time - start\_time

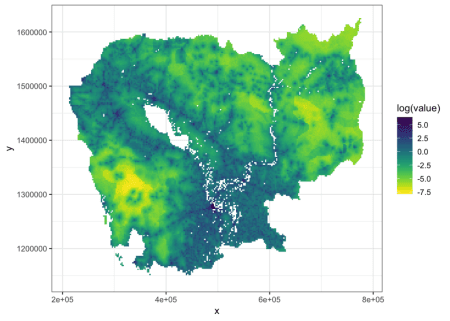
## Time difference of 1.548856 mins

rasterVis::gplot(res2) +

geom\_tile(aes(fill = log(value))) +

viridis::scale\_fill\_viridis(direction = -1, na.value='#FFFFFF00') +

theme\_bw()

[](https://i2.wp.com/www.francescobailo.net/wordpress/wp-content/uploads/2019/01/unnamed-chunk-6-1.png)

**Maths**

To test the second approach, let’s use the calc() and sqrt() functions, first without parallelization:

start\_time <- Sys.time()

calc(khm\_pop.r, sqrt)

## class : RasterLayer

## dimensions : 5205, 6354, 33072570 (nrow, ncol, ncell)

## resolution : 0.0008333, 0.0008333 (x, y)

## extent : 102.3375, 107.6323, 10.35008, 14.6874 (xmin, xmax, ymin, ymax)

## coord. ref. : +proj=longlat +datum=WGS84 +no\_defs +ellps=WGS84 +towgs84=0,0,0

## data source : in memory

## names : layer

## values : 0.02269337, 42.87014 (min, max)

end\_time <- Sys.time()

end\_time - start\_time

## Time difference of 3.316296 secs

and then with parallelization, this time with clusterR():

start\_time <- Sys.time()

beginCluster()

## 4 cores detected, using 3

clusterR(khm\_pop.r, sqrt)

## class : RasterLayer

## dimensions : 5205, 6354, 33072570 (nrow, ncol, ncell)

## resolution : 0.0008333, 0.0008333 (x, y)

## extent : 102.3375, 107.6323, 10.35008, 14.6874 (xmin, xmax, ymin, ymax)

## coord. ref. : +proj=longlat +datum=WGS84 +no\_defs +ellps=WGS84 +towgs84=0,0,0

## data source : in memory

## names : layer

## values : 0.02269337, 42.87014 (min, max)

endCluster()

end\_time <- Sys.time()

end\_time - start\_time

## Time difference of 16.49228 secs