How to easily build a prediction grid with the dshm R-package

Filippo Franchini

2019-01-13

Summary

In this tutorial you will learn how to use the `dshm` R-package to:

- Create a prediction grid.
- Fill the grid with relevant information such as covariate statisitcs.

Create a grid

You can create an empty grid using the function `dshm_make_grid` as follows:

```
empty_grid<-dshm_make_grid(extent = raster::extent(depth_crop),
            cell.size = 1000, projection = raster::crs(depth_crop))
```

You just created a grid with cell size of 1 km (i.e. 1000 m), the extent and projection of the raster `depth_crop`. You can plot the grid superimposed to the depth raster image and land shapefile using the following code:

```
dev.new()
raster::plot(depth_crop, main="Depth + empty grid")
raster::plot(land_crop,col="grey",add=TRUE)
raster::plot(empty_grid,add=TRUE)
```

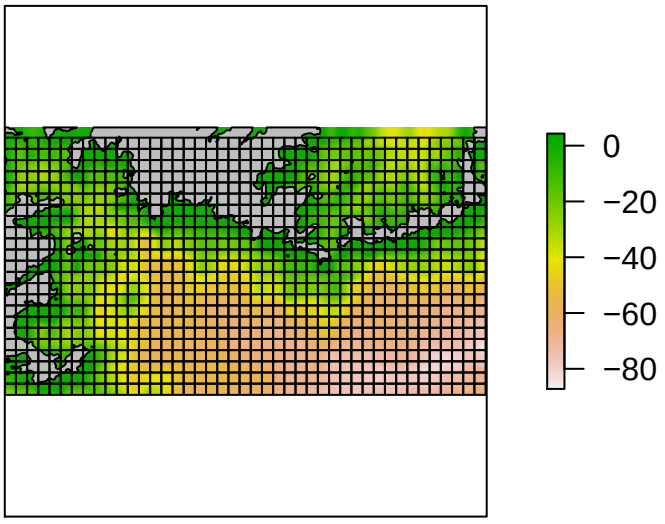

Figure 1: Prediction grid superimposed to covariate raster for depth and land shapefile.

As you may notice the upper part of the depth raster is not covered by grid cells. This is because, given the specified extent, there is not enough space for a 1x1 km grid cell. Here, we do not care about this problem but if you want to be picky you always have to design your grid bigger than the area in which you want to do your predictions.

Correcting and Filling the Grid

Now, we will correct the grid cells for land and we will add covariate statistics using the function `dshm_fill_grid` as follows:

```
grid_final<-dshm_fill_grid(empty.grid = empty_grid,
            land.data = land_crop,
            cov = list(depth=depth_crop,DR=DR_crop,DC=DC_crop),
            fun = mean,
            ncores = 7)
```

The function works only in parallel with at least 2 cores (i.e. almost all computers). This is because, depending on the raster resolution and the number of grid cells, the tasks requires a lot of time-consuming computations. The parallelization divides the grid into smaller grids that are given to different cores. In the function you can specify the arguments: `empty.grid` (i.e. the grid created with the function `dshm_make_grid`), `cov` (i.e. a list of covariate rasters), `fun` (i.e. the function to calculate covariate statisitcs within each grid cell), and `ncores` (i.e. the number of cores).

After running the function `dshm_fill_grid` you can visualize the final grid by typing:

```
raster::plot(grid_final)
```

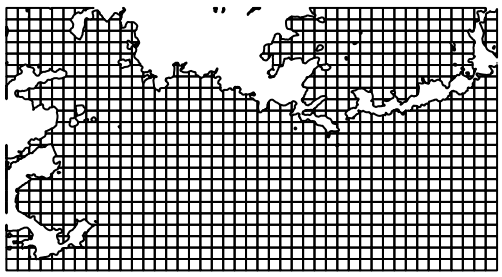

Figure 2: Finalized prediction grid.

As well as check the related data by typing:

```
head(round(grid_final@data,2), 6)
```

You should get something like this:

id	x.coord	y.coord	area	depth	DR	DC
1	45545.55	-2239999	1.00	-3.17	6737.31	716.57
2	46545.55	-2239999	0.97	-7.51	6541.56	408.30
3	47545.55	-2239999	1.00	-7.62	6288.32	563.12
4	48545.55	-2239999	0.99	-5.55	6545.47	631.91
5	49545.55	-2239999	1.00	-2.01	7148.85	735.36
6	50545.55	-2239999	0.92	-2.34	7903.22	483.67

For each grid cell you have the `id`, the centroids `x` and `y` coordinates, `area` in km^2, and all the covariates: in this case `depth`, distance to river (`DR`) and to coast (`DC`), in meters.

As for segments, some grid cell values for `depth` may by greater than 0. Moreover, we do not want to predict outside the covariate intervals of our segments, which will be used to calibrate the model. Predictions outside calibration intervals may generate unprobabilistic results. We have thus to exclude those grid cells whose covariate values are outside the covariate values of our segments. We can use the following code to create the segment covariate limits:

```
limits<-list(depth=min(cor_seg_final$depth),
        DC=max(cor_seg_final$DC),
        DR=max(cor_seg_final$DR))
```

And then we can use the specified limits to exclude the grid cells:

```
grid_final<-grid_final[grid_final$depth>=limits$depth&
            grid_final$depth<0&
            grid_final$DC<=limits$DC&
            grid_final$DR<=limits$DR,]
```

The new grid looks like this:

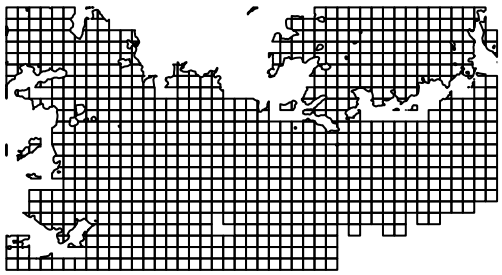

Figure 3: Finalized prediction grid including only grid cells constrained to segment covariate intervals.

Save Grid as Shapefile

You can easily save the grid as a shapefile (i.e. .shp) so that you can import them in other GIS software such as ArcGIS or QGIS. As a first step you have to abbreviate all column descripions:

```
names(grid_final)<-c("ID","coord.x","coord.y","A","D","DR","DC")
```

Then you can run the following code:

```
rgdal::writeOGR(obj=grid_final,dsn="/Users/User/Desktop/data",
layer="grid_final",driver="ESRI Shapefile",overwrite_layer=TRUE)
```

You have to specify your own `dsn`, i.e. the directory where the file will be saved. You can then upload the saved file into a GIS software or to reload it into R by typing:

```
rgdal::readOGR("grid_final.shp")
```

Note that you have to specify the directory where the file was saved.