The **rasterList** package has been developed to create a S4 class object that can manage complex operations and objects in a spatially gridded map, i.e. a raster map. The *rasterList-Class* (Cordano 2017b) object differs from a traditional raster map by the fact that each cell can contain a generic object instead of numeric values on one or more bands or layers. The raster of objects, i.e. the *RasterList-class* object, consists on a *rasterLayer-Class* (Hijmans 2016) connected to a generic list: one list element for each raster cells. It allows to write few lines of R code for complex map algebra. Furthermore, in accordance with the existing classes defined in the R environment, each cell of the raster is “occupied” by an an istanced object. This may be of utitily plenty in most applications, e.g. environmental modellig.

**Analysing Raster Data: an introduction**

Nowadays the availability of spatially gridded coverages , often obtained by distributed biophysical/environmental models or remote sensed observations, e. g. CHIRPS rainfall (Funk et al. 2015), lead to the fact that normal operations on time series analysis and statistics must be replicated in each pixel or cell of the spatially gridded domain (raster). Based on **raster** package (Hijmans 2016), a S4 class has been created such that results of complex operations or speficfic R objects (e.g, S3 or S4) can be executed on each cells of a raster map. The raster of objects contains the traditional raster map with the addition of a list of generic objects: one object for each raster cells. In such a way, different kinds of objects, e.g. the probability distrubution of the mapped random variable or an *htest*, *lm* ,*glm* or *function* class objects (Chambers 1992)(Hastie and Pregibon 1992), can be saved for each cell, in order that the user can save intermediate results without loosing any geographic information. This new oblect class is implemented in a new package **rasterList** (Cordano 2017b). Two applications are presented:

* Creation of a 2D/3D modelled map of soil water retention curve based on existing soil properties map and state-of-art empirical formulation (VanGenuchten 1980);
* Processing spatial gridded datasets of daily or momthly precipitation (Funk et al. 2015) for a trend or frequancy analysis (estimation of the return period of critical events).

**Analysing Raster Data: Application 1**

The application 1 is the estimation of the mathematical relationships between soil water content \thetaand soil water pressure \psi, i.e. soil water retention curves, in the cells of a raster map. Soil water retention curve is widely used in subsurface water/groundwater hydrological modeling because it allows to close the water balance equation. Generally some theoretical formulas are used to model this curve in function of soil properties. In this examples, given a raster map of soil properties, a map of soil water retention curves is produced. First of all, in absence of detailed soil information, as often used in hydrological modeling (Kollet et al. 2017), soil water curve is defined with the following empirical formula (VanGenuchten 1980):

\theta = \theta_{res}+(\theta_{sat}-\theta_{res})*(1+ |\alpha \psi|^n)^{-m} \qquad \psi data-recalc-dims=0″ />

\theta = \theta_{sat} \qquad \psi data-recalc-dims=0″ />

where \thetais the volumetric water content (e.g. water volume per soil volume) ranging between residual water content \theta_{res}and saturated water content \theta_{sat}. \alpha,n and mare parameters assuming to depend on soil type. An estimation of the parameter \theta_{sat},\theta_{res},\alpha,m,n values in function of soil type is given through the following table (Maidment 1992):

library(soilwater)

library(stringr)

soilparcsv <- system.file("external/soil\_data.csv",package="soilwater")

soilpar <- read.table(soilparcsv,stringsAsFactors=FALSE,header=TRUE,sep=",")

knitr::kable(soilpar,caption="Average value of Va Genuchten's parameter per each soil type")

| Average value of Va Genuchten’s parameter per each soil type | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **type** | **swc** | **rwc** | **alpha** | **n** | **m** | **Ks\_m\_per\_hour** | **Ks\_m\_per\_sec** |
| sand | 0.44 | 0.02 | 13.8 | 2.09 | 0.52153 | 0.15000 | 4.17e-05 |
| loamy sand | 0.44 | 0.04 | 11.5 | 1.87 | 0.46524 | 0.10000 | 2.78e-05 |
| sandy loam | 0.45 | 0.05 | 6.8 | 1.62 | 0.38272 | 0.03900 | 1.08e-05 |
| loam | 0.51 | 0.04 | 9.0 | 1.42 | 0.29577 | 0.03300 | 9.20e-06 |
| silt loam | 0.50 | 0.03 | 4.8 | 1.36 | 0.26471 | 0.00900 | 2.50e-06 |
| sandy clay loam | 0.40 | 0.07 | 3.6 | 1.56 | 0.35897 | 0.00440 | 1.20e-06 |
| clay loam | 0.46 | 0.09 | 3.9 | 1.41 | 0.29078 | 0.00480 | 1.30e-06 |
| silty clay loam | 0.46 | 0.06 | 3.1 | 1.32 | 0.24242 | 0.00130 | 4.00e-07 |
| sandy clay | 0.43 | 0.11 | 3.4 | 1.40 | 0.28571 | 0.00073 | 2.00e-07 |
| silty clay | 0.48 | 0.07 | 2.9 | 1.26 | 0.20635 | 0.00076 | 2.00e-07 |
| clay | 0.48 | 0.10 | 2.7 | 1.29 | 0.22481 | 0.00041 | 1.00e-07 |

soilpar$color <- str\_sub(rainbow(nrow(soilpar)),end=7) ## Only first 7 characters of HTML code are considered.

It it is assumed that each cell of a raster maps has its own soil water retention curve. Therefore to map the soil water retion curve and not only its parameters, **rasterList** package is loaded:

library(rasterList)

## Loading required package: raster

## Loading required package: sp

The study area is is the area of the *meuse* dataset (n.d.):

library(sp)

data(meuse)

help(meuse)

data(meuse.grid)

help(meuse.grid)

A soil map is avaialbe from soil type according to the 1:50 000 soil map of the Netherlands. In the Meuse site, the following soil type are present

* 1 = Rd10A (Calcareous weakly-developed meadow soils, light sandy clay);
* 2 = Rd90C/VII (Non-calcareous weakly-developed meadow soils, heavy sandy clay to light clay);
* 3 = Bkd26/VII (Red Brick soil, fine-sandy, silty light clay) which can be respectively assimilated as:

soiltype\_id <- c(1,2,3)

soiltype\_name <- c("sandy clay","clay","silty clay loam")

soilpar\_s <- soilpar[soilpar$type %in% soiltype\_name,]

is <- order(soilpar\_s[,1],by=soiltype\_name)

soilpar\_s <- soilpar\_s[is,]

soilpar\_s$id <- soiltype\_id

A geografic view of Meuse test case is available though **leaflet** package (Cheng, Karambelkar, and Xie 2018):

library(rasterList)

library(soilwater)

library(leaflet)

set.seed(1234)

data(meuse.grid)

data(meuse)

coordinates(meuse.grid) <- ~x+y

coordinates(meuse) <- ~x+y

gridded(meuse.grid) <- TRUE

proj4string(meuse) <- CRS("+init=epsg:28992")

proj4string(meuse.grid) <- CRS("+init=epsg:28992")

soilmap <- as.factor(stack(meuse.grid)[['soil']])

ref\_url <- "https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png"

ref\_attr <- 'Map data: © [OpenStreetMap](http://www.openstreetmap.org/copyright), [SRTM](http://viewfinderpanoramas.org) | Map style: © [OpenTopoMap](https://opentopomap.org) ([CC-BY-SA](https://creativecommons.org/licenses/by-sa/3.0/))'

opacity <- 0.6

color <- colorFactor(soilpar\_s$color,level=soilpar\_s$id)

labFormat <- function(x,values=soilpar\_s$type){values[as.integer(x)]}

leaf1 <- leaflet() %>% addTiles(urlTemplate=ref\_url,attribution=ref\_attr)

leaf2 <- leaf1 %>% addRasterImage(soilmap,opacity=opacity,color=color) %>%

addLegend(position="bottomright",colors=soilpar\_s$color,labels=soilpar\_s$type,title="Soil Type",opacity=opacity)

leaf2

Therefore, a map of each VanGenuchten (1980)’s formula parameter can be obtained as follow (The names of saturated and residual water contents *swc* and *rwc* mean \theta_{sat}and \theta_{res}according to the the arguments of *swc* function of **soilwater** package (Cordano, Andreis, and Zottele 2017))

soil\_parameters\_f <- function (soiltype,sp=soilpar\_s) {

o <- sp[soiltype,c("swc","rwc","alpha","n","m")]

names(o) <- c("theta\_sat","theta\_res","alpha","n","m")

return(o)

}

soil\_parameters\_rl <- rasterList(soilmap,FUN=soil\_parameters\_f)

A *RasterList-class* object has been created and each cell contains a vector of soil water parameters:

soil\_parameters <- stack(soil\_parameters\_rl)

soil\_parameters

## class : RasterStack

## dimensions : 104, 78, 8112, 5 (nrow, ncol, ncell, nlayers)

## resolution : 40, 40 (x, y)

## extent : 178440, 181560, 329600, 333760 (xmin, xmax, ymin, ymax)

## coord. ref. : +init=epsg:28992 +proj=sterea +lat\_0=52.15616055555555 +lon\_0=5.38763888888889 +k=0.9999079 +x\_0=155000 +y\_0=463000 +ellps=bessel +towgs84=565.4171,50.3319,465.5524,-0.398957,0.343988,-1.87740,4.0725 +units=m +no\_defs

## names : theta\_sat, theta\_res, alpha, n, m

## min values : 0.43000, 0.06000, 2.70000, 1.29000, 0.22481

## max values : 0.48000, 0.11000, 3.40000, 1.40000, 0.28571

A map of saturated water content (porosity) can be visualized as follows:

theta\_sat <- soil\_parameters[["theta\_sat"]]

color <- colorNumeric("Greens",domain=theta\_sat[])

leaf3 <- leaf1 %>% addRasterImage(theta\_sat,color=color,opacity=opacity) %>%

addLegend(position="bottomright",pal=color,values=theta\_sat[],opacity=opacity,title="Porosity")

Let consider 3 points of interest in the Meuse located (latitude and logitude) as follows:

lat <- 50.961532+c(0,0,0.02)

lon <- 5.724514+c(0,0.01,0.0323)

name <- c("A","B","C")

points <- data.frame(lon=lon,lat=lat,name=name)

knitr::kable(points,caption="Points of interest: geographical coordinates (latitude andlongitude) and name")

| Points of interest: geographical coordinates (latitude andlongitude) and name | | |
| --- | --- | --- |
| **lon** | **lat** | **name** |
| 5.724514 | 50.96153 | A |
| 5.734514 | 50.96153 | B |
| 5.756814 | 50.98153 | C |

coordinates(points) <- ~lon+lat

proj4string(points) <- CRS("+proj=longlat +ellps=WGS84")

leaf3 %>% addMarkers(lng=points$lon,lat=points$lat,label=points$name)

<

They are indexed as follows in the raster map:

points$icell <- cellFromXY(soil\_parameters,spTransform(points,projection(soil\_parameters)))

where *icell* is a vector of integer numbers corresponding to the raster cells containing the points A,B and C. Information on these points can be extracted as follows:

soil\_parameters[points$icell]

## theta\_sat theta\_res alpha n m

## [1,] 0.48 0.10 2.7 1.29 0.22481

## [2,] 0.48 0.10 2.7 1.29 0.22481

## [3,] 0.43 0.11 3.4 1.40 0.28571

Using *swc* , a function that given the soil parameters returns a soil water function is defined:

library(soilwater)

swc\_func <- function(x,...) {

o <- function(psi,y=x,...) {

args <- c(list(psi,...),as.list(y))

oo <- do.call(args=args,what=get("swc"))

}

return(o)

}

And in case that it is applied to point A (i.e. setting the soil parameters of point A), it is:

soilparA <- soil\_parameters[points$icell[1]][1,]

swcA <- swc\_func(soilparA)

where *scwA* is a function corresponding to the Soil Water Retention Curve in point A that can be plotted as follows:

library(ggplot2)

psi <- seq(from=-5,to=1,by=0.25)

title <- "Soil Water Retention Curve at Point A"

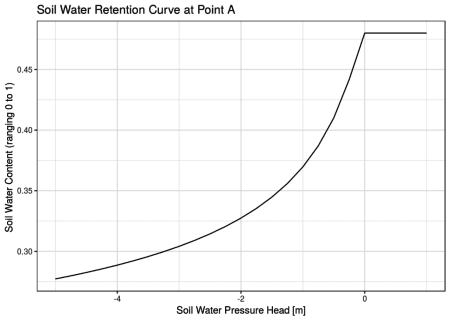
xlab <- "Soil Water Pressure Head [m]"

ylab <- "Soil Water Content (ranging 0 to 1) "

gswA <- ggplot()+geom\_line(aes(x=psi,y=swcA(psi)))+theme\_bw()

gswA <- gswA+ggtitle(title)+xlab(xlab)+ylab(ylab)

gswA

  
The procedure is replicated for all cells of the raster map, saving the soil water retention curves as an R object for all the cells, then making use of *rasterList* function:

swc\_rl <- rasterList(soil\_parameters,FUN=swc\_func)

The *RasterList-class* object contains each soil water retention curve per each cell. Since the list element referred to each raster cell are *function* class, *swc\_rl* is coerced to a single function defined all raster extension through *rasterListFun* function:

swc\_rlf <- rasterListFun(swc\_rl)

In such a way, the function *swc\_rlf* can be used to estimate to estimate soil water function given a spatially unimorm value of soil water pressure head \psion all Meuse region.

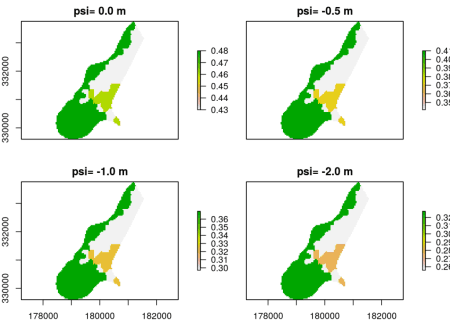
psi <- c(0,-0.5,-1,-2)

names(psi) <- sprintf("psi= %1.1f m",psi)

soil\_water\_content <- stack(swc\_rlf(psi))

names(soil\_water\_content) <- names(psi)

plot(soil\_water\_content,main=names(psi))



Finally, assuming that \psimay vary with space ranging between -0.5 m and -1.5 m with a spatial exponetial dacay profile from point A, a map for \psiis calculated as follows:

region <- raster(swc\_rlf(0))

mask <- !is.na(region)

region[] <- NA

region[points$icell[1]] <- 1

dist <- distance(region)

dist[mask==0] <- NA

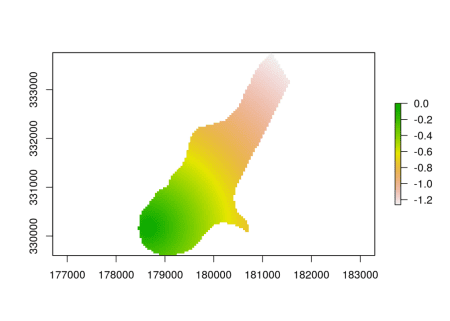
mdist <- max(dist[],na.rm=TRUE)

psi\_max <- 0

psi\_min <- -2

psi <- psi\_max+(psi\_min-psi\_max)\*(1-exp(-dist/mdist))

plot(psi)



color <- colorNumeric("Blues",domain=psi[])

leaf\_psi <- leaf1 %>% addRasterImage(psi,color=color,opacity=opacity) %>%

addLegend(position="bottomright",pal=color,values=psi[],opacity=opacity,title="Psi") %>% addMarkers(lng=points$lon,lat=points$lat,label=points$name)

## Warning in colors(.): Some values were outside the color scale and will be

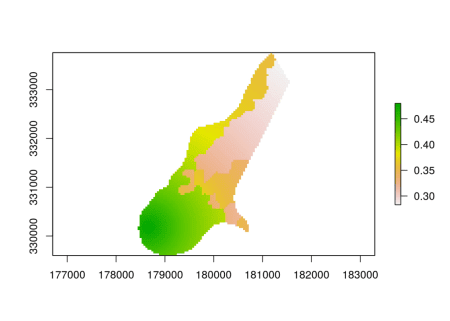
## treated as NA

leaf\_psi

The obtained map of \thetais then visualized through **leaflet**:

theta <- raster(swc\_rlf(psi))

plot(theta)



color <- colorNumeric("Blues",domain=theta[])

leaf\_psi <- leaf1 %>% addRasterImage(theta,color=color,opacity=opacity) %>%

addLegend(position="bottomright",pal=color,values=theta[],opacity=opacity,title="Psi") %>% addMarkers(lng=points$lon,lat=points$lat,label=points$name)

leaf\_psi

**Conclusions**

The above applications illustrate how statistical analysis and other processing may be replicated at every cell of a raster map. The **rasterList** package has been created to answer the need to make complex operation on spatially gridded region. The shown examples go beyond their own task but they have the task of presenting the most important functions of the package and how thay can be used to solve practical problems. **rasterList** is able to save different type of objects within a cell of the raster map and allows that each object of a raster cell may be transformed and/or coerced to another object. Two or more *RasterList-class* objects can interact through operations with one another. This is of great relevance because it allows to save intermediate steps during a processing workflow (e.g. statistical analysis).