

Modelos Digitales del Terreno

Alexander Quevedo

Cargar paquetes en R

Una vez iniciado R, debemos cargar las librerías necesarias, en esta caso emplearemos `raster`,`spatialEco` y `rasterVis`, los siguientes tres fragmentos de código verificarán que los paquetes necesarios se encuentre instalados :

```
if (!require("raster")) install.packages("raster")

## Loading required package: raster
## Loading required package: sp
if (!require("spatialEco")) install.packages("spatialEco")

## Loading required package: spatialEco
## spatialEco 0.1-7
## Type se.news() to see new features/changes/bug fixes.
if (!require("rasterVis")) install.packages("rasterVis")

## Loading required package: rasterVis
## Loading required package: lattice
## Loading required package: latticeExtra
## Loading required package: RColorBrewer
```

El siguiente paso consiste en definir el directorio de trabajo usando el comando `setwd()`. Recuerde que en el caso del sistema operativo Windows debe de indicarse la ruta de la siguiente manera C:\Documentos\Modelos_DT; A continuación defina como directorio de trabajo Modelos_DT.

```
setwd("~/Google Drive/Curso_modelado_aq/Modelos_DT")
```

Trabajando con archivos raster

Cargamos el modelo digital de elevación y revisamos sus parámetros básicos:

```
MDE<-raster("MDE_15m.tif")
MDE

## class      : RasterLayer
## dimensions : 1800, 2400, 4320000 (nrow, ncol, ncell)
## resolution : 0.0001388887, 0.0001388883 (x, y)
## extent     : -102.3333, -102, 19.25, 19.5 (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : /Users/alexanderquevedo/Google Drive/Curso_modelado_aq/Modelos_DT/MDE_15m.tif
## names      : MDE_15m
## values     : 868, 3845 (min, max)
```

Es de crucial importancia conocer el sistema de referencia en el cual se encuentra el archivo que acabamos de cargar, para conocerlo usamos el comando `crs(Nombre del raster)`.

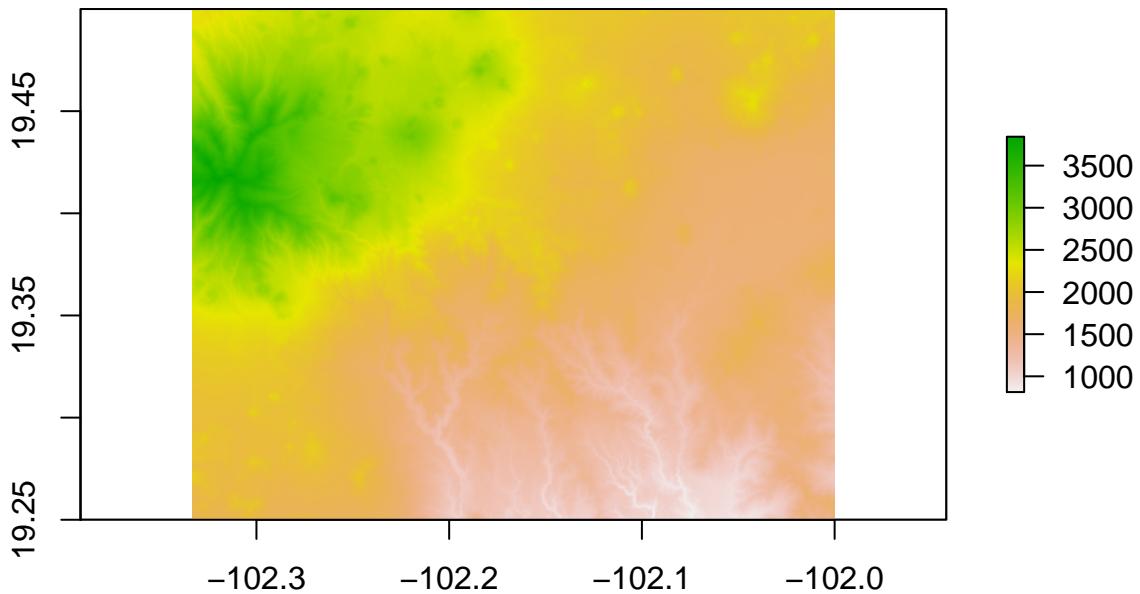
```
crs(MDE)
```

```
## CRS arguments:  
## +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
```

¿En que sistema de referencia esta? geograficas, planas.

visualicemos nuestro MDE

```
plot(MDE)
```



Proyectamos a un sistema de coordenadas planas

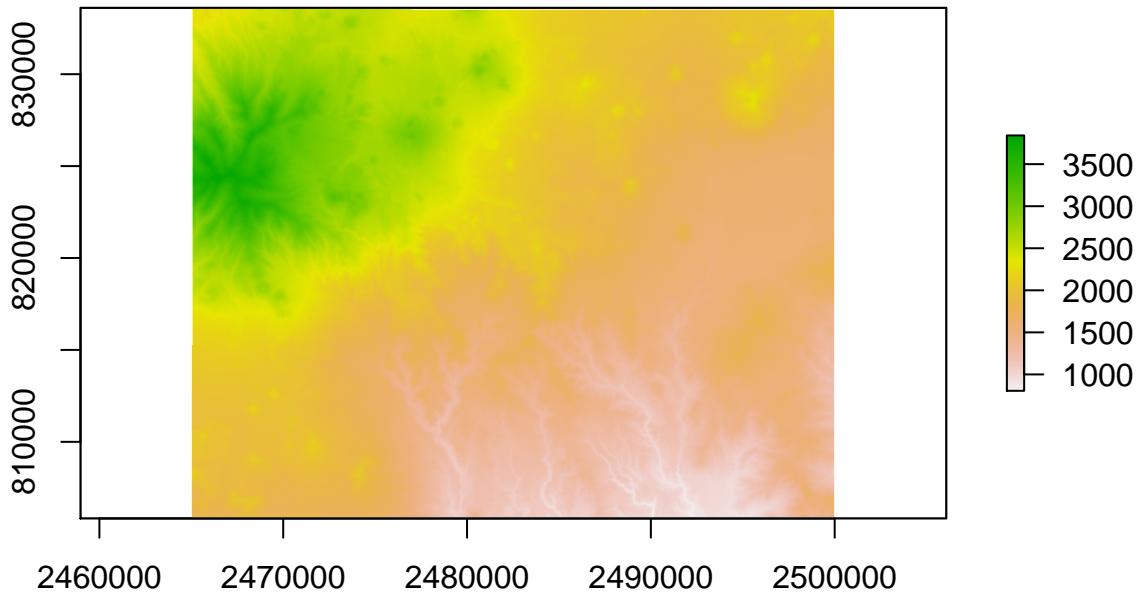
```
rs<-"+init=epsg:6362 +proj=lcc +lat_1=17.5 +lat_2=29.5 +lat_0=12 +lon_0=-102 +x_0=2500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

```
MDE_P<-projectRaster(MDE, crs = rs)
```

```
MDE_P
```

```
## class      : RasterLayer  
## dimensions : 1816, 2420, 4394720  (nrow, ncol, ncell)  
## resolution : 14.5, 15.3  (x, y)  
## extent     : 2464977, 2500067, 805828.5, 833613.3  (xmin, xmax, ymin, ymax)  
## coord. ref. : +init=epsg:6362 +proj=lcc +lat_1=17.5 +lat_2=29.5 +lat_0=12 +lon_0=-102 +x_0=2500000 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs  
## data source: in memory  
## names      : MDE_15m  
## values     : 804.7008, 3844.281  (min, max)
```

```
plot(MDE_P)
```

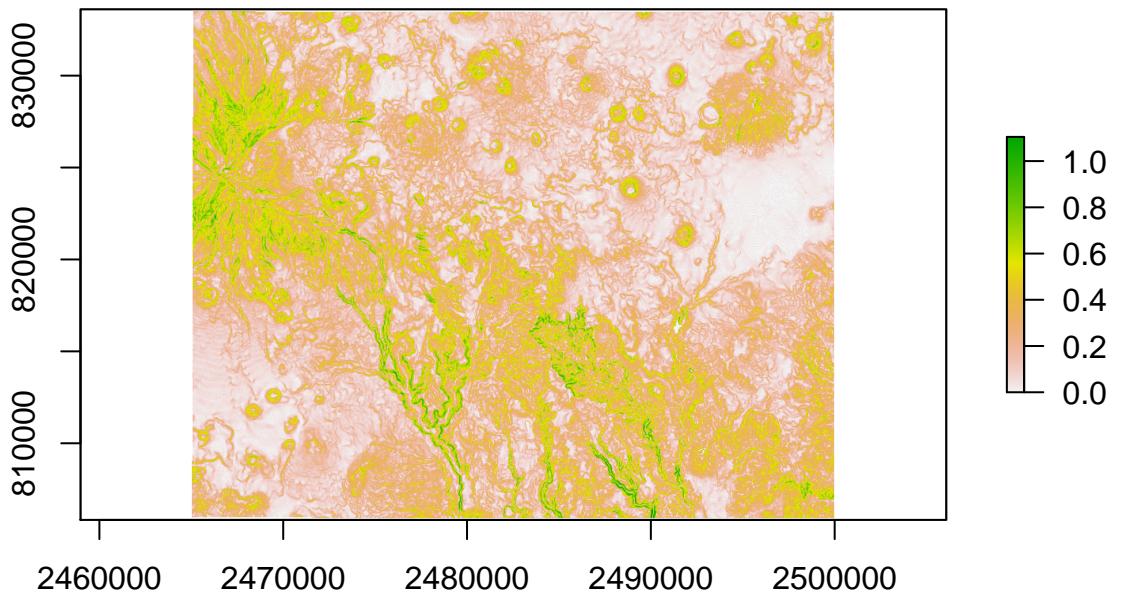


Primera Derivada.

Para el calculo de los modelos del terreno de la primera derivada, pendiente y orientación (aspect), se empleara el comando `terrain`. Los resultados obtenidos se expresan en radianes, para una mayor comprension los transformaremos en grados.

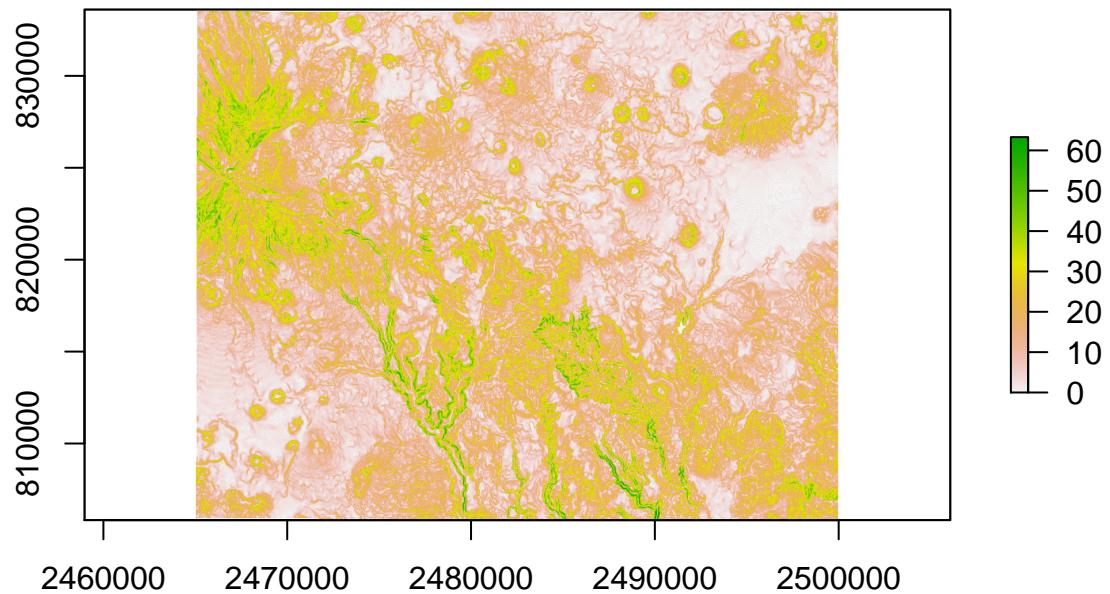
```
slope <- terrain(MDE_P, "slope")
plot(slope, main = "Pendiente en radianes")
```

Pendiente en radianes



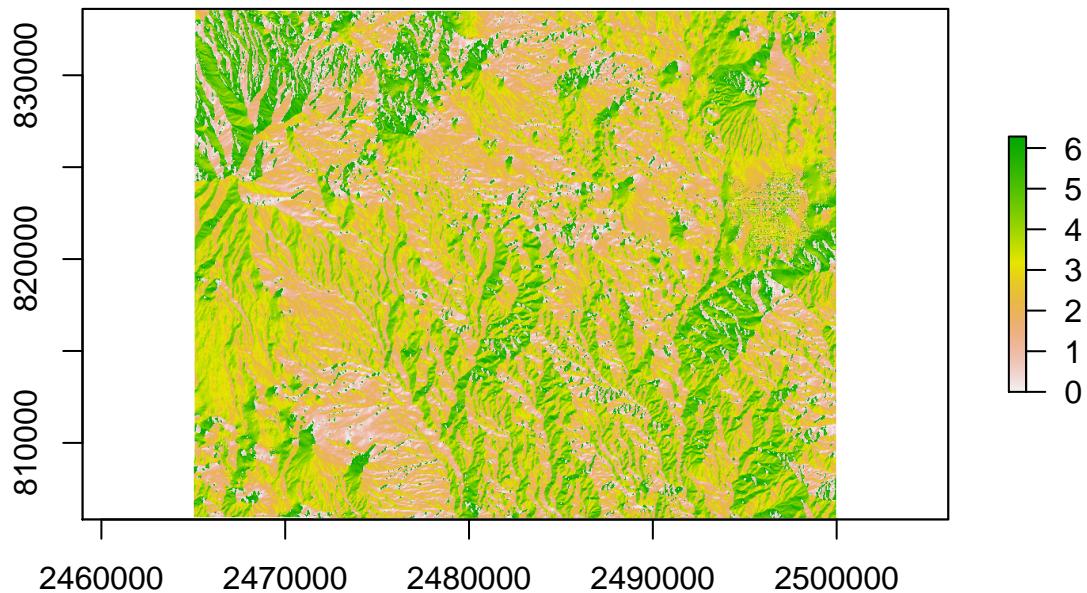
```
slope_d<-(slope*180)/pi
plot(slope_d, main="Pendiente en grados")
```

Pendiente en grados



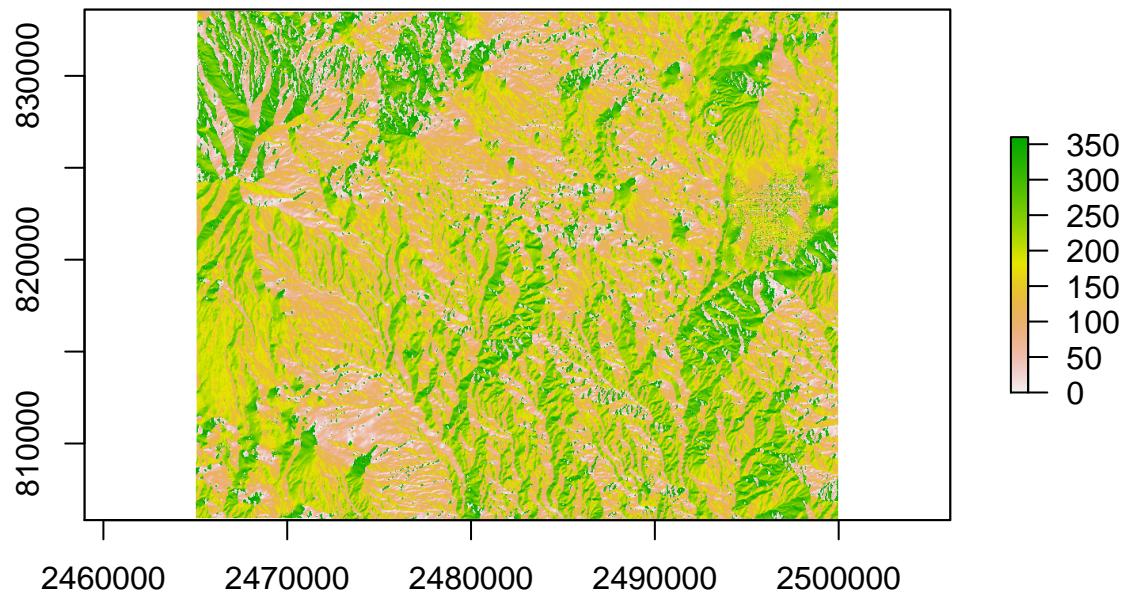
```
aspect<-terrain(MDE_P,opt = "aspect")
plot(aspect, main="Aspect en radianes")
```

Aspect en radianes



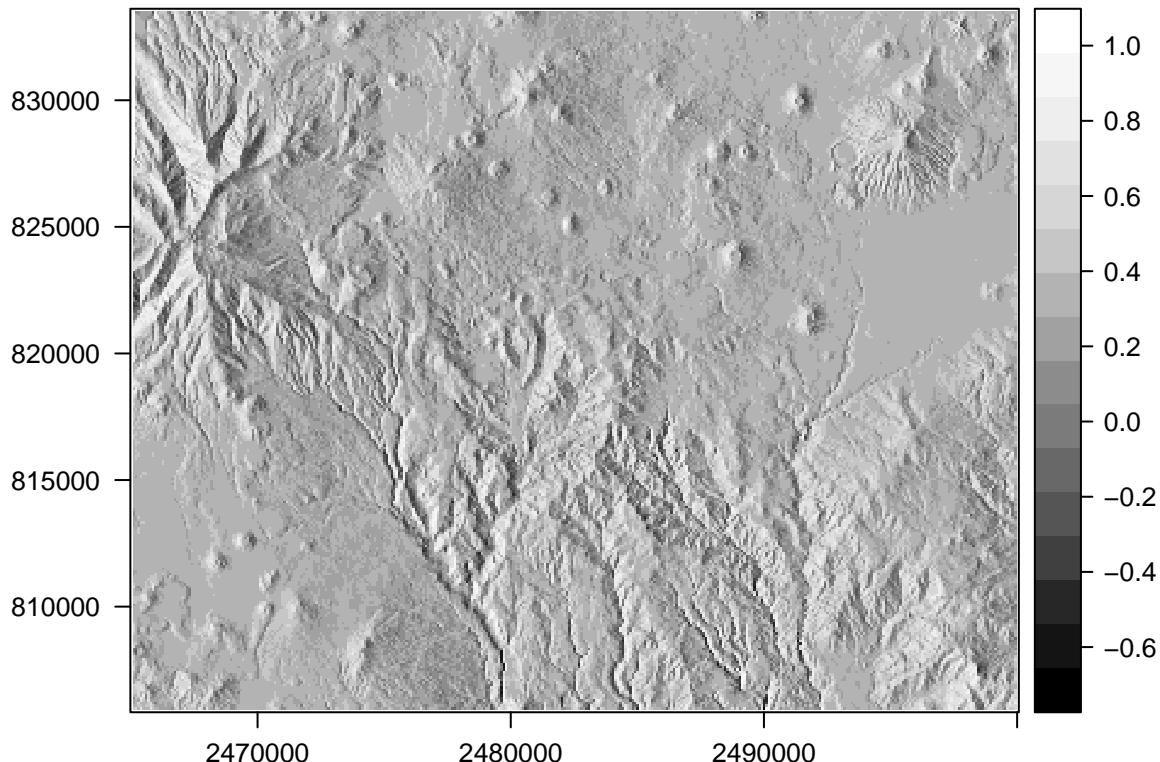
```
aspect_d<-(aspect*180)/pi
plot(aspect_d, main="Aspect en grados")
```

Aspect en grados



Para el calculo del modelo de sombras, se usa el comando `hillShade` , el cual necesita la pendiente (en radianes), `aspect(radianes)`, angulo de elevación del sol (grados) y el angulo de dirección del sol (grados).

```
hill <- hillShade(slope, aspect, 20, 270)
levelplot(hill, par.settings = GrTheme, margin = FALSE)
```

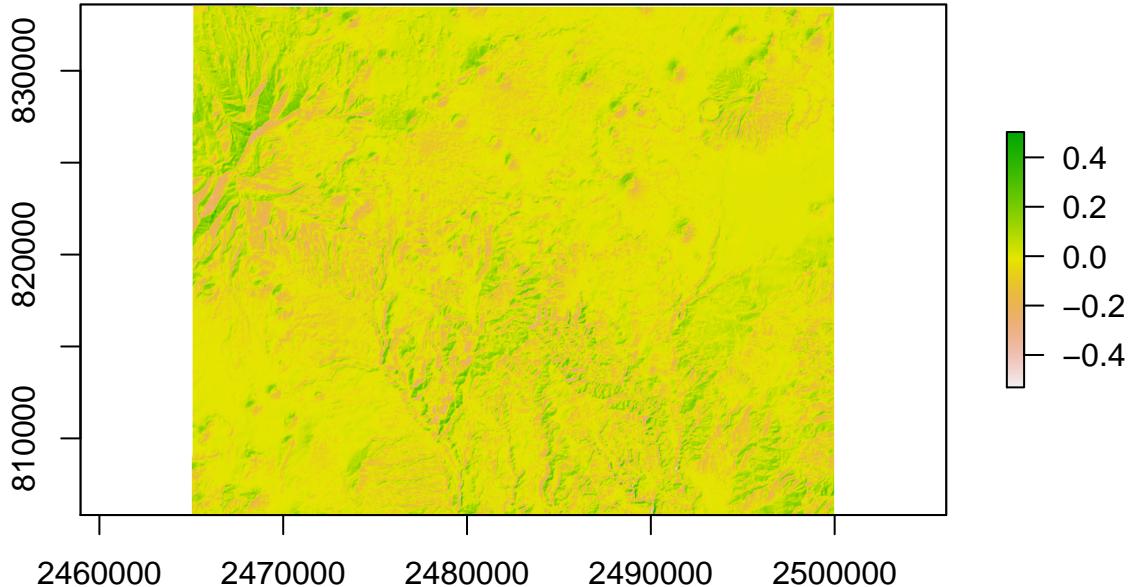


Segunda Derivada

La función de curvatura visualiza la forma o la curvatura de la pendiente. Una parte de la superficie puede ser concava o convexa. Es fácil de comprobar consultando el valor de la curvatura. La curvatura se obtiene calculando la segunda derivada de la superficie.

El comando que debemos utilizar es `curvature`, el cual requiere el modelo digital de elevación y el tamaño de la ventana(kernel) para la cual se calculará la curvatura.

```
mcrv <- curvature(MDE_P, s=3, type="mcnab")
plot(mcrv)
```

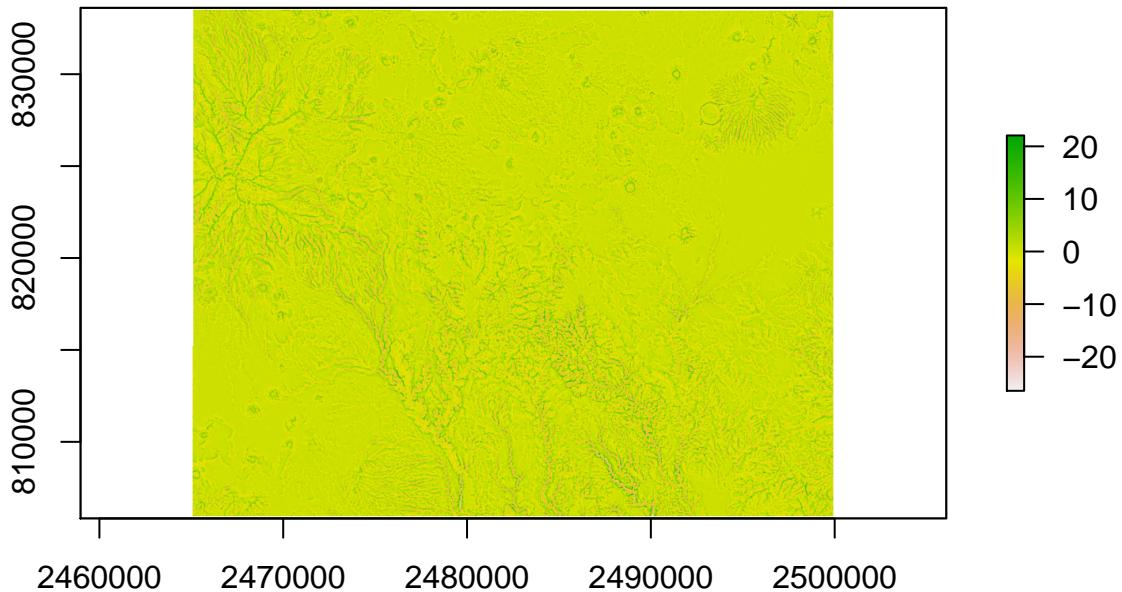


Índices de posición topográfica.

El índice de posición topográfica (TPI) es un algoritmo que se utiliza para medir las posiciones de la pendiente y para automatizar las clasificaciones de la topografía.

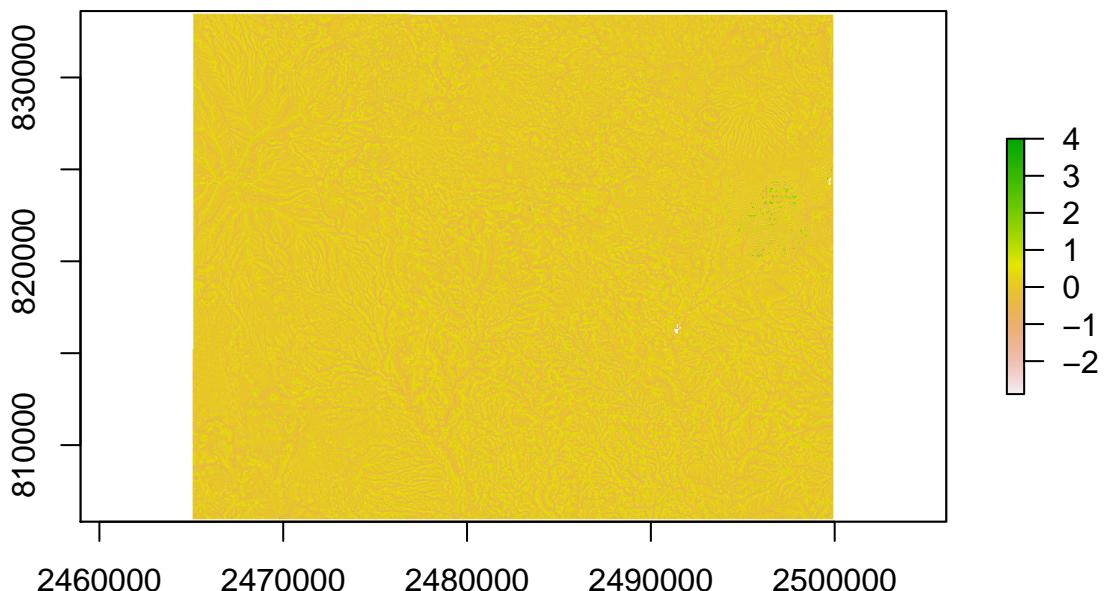
El comando `tpi` requiere que un modelo digital de elevación, el tamaño de la venta `scale`.

```
TPI_P<-tpi(MDE_P, scale = 7)
plot(TPI_P)
```



Es posible realizar una normalización del tpi usando la desviación estandar, especificando la opción `normalize = TRUE`.

```
TPI_PN<-tpi(MDE_P, scale = 7, win = "rectangle", normalize = TRUE)
plot(TPI_PN)
```



Clasificación de geoformas

A continuación procederemos a realizar una clasificación de geoformas, empleando modelos digitales del terreno calculados a partir del modelo de elevación.

En este caso se empleará la pendiente (en grados), el TPI y la desviación estandar del TPI. Para mayor información sobre esta clasificación puede consultar los siguientes link:

<http://scialert.net/fulltext/?doi=jas.2008.910.921>

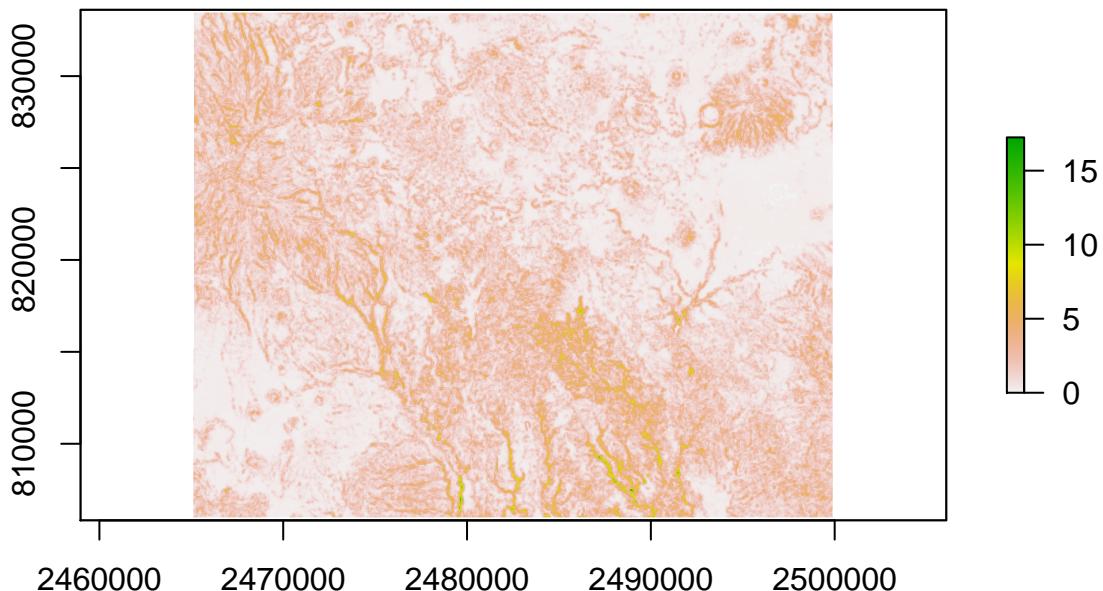
http://www.jennissent.com/downloads/tpi_documentation_online.pdf

Previamente se realizo el calculo de la pendiente y del TPI, ahora se calcularan la desviacion estandar, para lo cual usaremos el comando `focal`, el cual requiere de un archivo raster de entrada, una ventana de calculo la cual en este caso se define usando una matriz `w` y por ultimo la funcion que deseamos aplicar en este caso `fun = sd`.

```
matrix(1,7,7)

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]     1     1     1     1     1     1     1
## [2,]     1     1     1     1     1     1     1
## [3,]     1     1     1     1     1     1     1
## [4,]     1     1     1     1     1     1     1
## [5,]     1     1     1     1     1     1     1
## [6,]     1     1     1     1     1     1     1
## [7,]     1     1     1     1     1     1     1

tpi.sd <- focal(TPI_P, w = matrix(1,7,7), fun = sd)
plot(tpi.sd)
```



El siguiente paso es la clasificación las formas del relieve usando los criterios de TPI y pendiente como se formulan en la Figura 1.

Para aplicar esta clasificación a las capas raster se usara la función `overlay`, la cual requiere las capas raster y la función que se aplicara a las mismas

Construcción de la función.

```
rc <- function(x,x2,x3) {
  ifelse(x <= (-1*x2),1,
        ifelse(x>(-1*x2) & x<x2 & x3<6,2,
              ifelse( x>(-1*x2) & x<x2 & x3>=6,3,
                     ifelse(x>=x2,4,NA))))}
```

Aplicamos la clasificación

```
x1<-TPI_P
x2<-tpi.sd
```

	TPI and slope criteria			
Landform classification (Weiss, 2001)	Small neighborhood 50 m-neighborhood	Large neighborhood 450 m-neighborhood	Slope position classification (Weiss, 2001)	TPI and slope criteria
Canyons, deeply incised streams	$TPI \leq -1 SD$	$TPI \leq -1 SD$	Valley	$TPI \leq -1 SD$
Mid-slope drainages, Shallow valleys	$TPI \leq -1 SD$	$-1 SD < TPI < 1 SD$	Flat surface	$TPI > -1 SD$ and $TPI < 1 SD$ Slope $< 6^\circ$
Upland drainages, Headwaters	$TPI \leq -1 SD$	$TPI \geq 1 SD$	Mid slope	$TPI > -1 SD$ and $TPI < 1 SD$ Slope $\geq 6^\circ$
U-shaped valleys	$-1 SD < TPI < 1 SD$	$TPI \leq -1 SD$	Hilltop	$TPI = 1 SD$
Plains	$-1 SD < TPI < 1 SD$	$-1 SD < TPI < 1 SD$ Slope $\leq 5^\circ$		
Open slopes	$-1 SD < TPI < 1 SD$	$-1 SD < TPI < 1 SD$		
Slope $> 5^\circ$				
Upper slopes, mesas	$-1 SD < TPI < 1 SD$	$TPI \geq 1 SD$		
Local ridges, hills in valleys	$TPI \geq 1 SD$	$TPI \leq -1 SD$		
Mid-slope ridges, small hills in plains	$TPI \geq 1 SD$	$-1 SD < TPI < 1 SD$		
Mountain tops, high ridges	$TPI \geq 1 SD$	$TPI \geq 1 SD$		

Figura 1: Criterios de clasificación.

```
x3<-slope_d
landforms<-overlay(stack(x1,x2,x3),fun=rc)
plot(landforms, legend = FALSE, col = rev(terrain.colors(4)))
legend("topright", legend = c("Valley", "Flat surface", "Mid slope", "Hilltop"),
      fill = rev(terrain.colors(4)))
```

