Programando mapas con R

Guillermo Luijk
overfitting.net
Madrid, mayo 2024

Programando mapas con R

1. INTRODUCCIÓN
2. FUNCIONES AUXILIARES
3. MAPAS MINIMALISTAS
4. MAPAS PSEUDO 3D
5. MAPAS CREATIVOS
6. VIDEO

```
arrayresample()
arrayblur()
solid()
contour()
```

```
linemap()
circlemap()
joymap()
horizonmap()
```

```
aerialpersp()
slicemap()
hillshademap()
shadowmap()
```

```
legomap()
foldmap()
karesansuimap()
```

Programando mapas con R

1. INTRODUCCIÓN

2. FUNCIONES AUXILIARES

3. MAPAS MINIMALISTAS

4. MAPAS PSEUDO 3D

5. MAPAS CREATIVOS

6. VIDEO

1. INTRODUCCIÓN

- Plantearemos **algoritmos** para resolver cada problema
- Programaremos los mapas en **R base** (paquete 'terra')
- Código genérico **portable** a otros lenguajes
- Trataremos de **vectorizar** el código
- Salida bitmap en **escala de grises** y sin antialiasing
- El **color** se aplica como paso final (R, curvas Photoshop,…)

Programando mapas con R

1. INTRODUCCIÓN

```
arrayresample()
arrayblur()
solid()
contour()
```

2. FUNCIONES AUXILIARES

```
linemap()
circlemap()
joymap()
horizonmap()
```

3. MAPAS MINIMALISTAS

4. MAPAS PSEUDO 3D

```
aerialpersp()
slicemap()
hillshademap()
shadowmap()
```

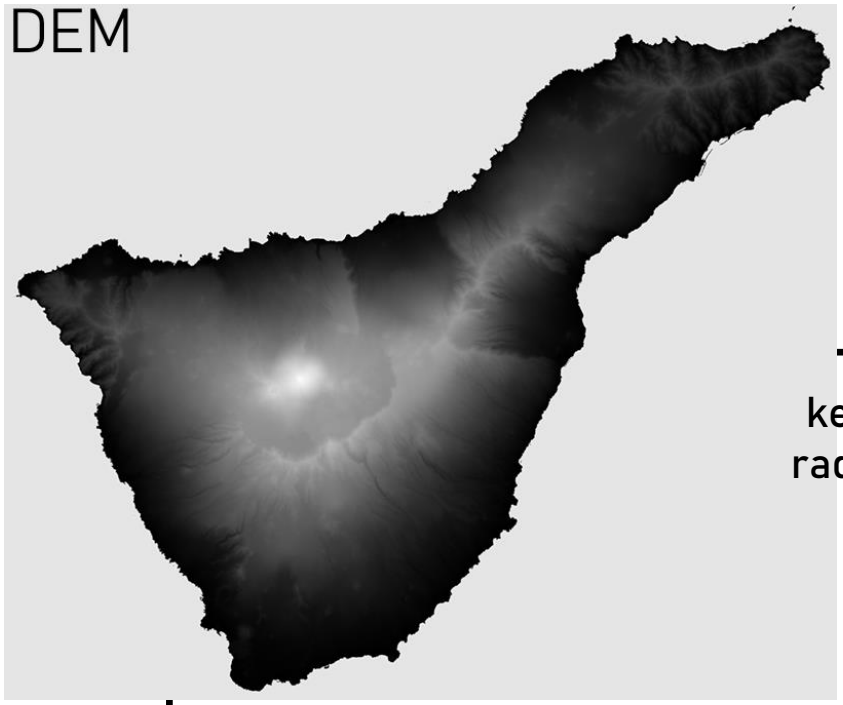
5. MAPAS CREATIVOS

```
legomap()
foldmap()
karesansuimap()
```

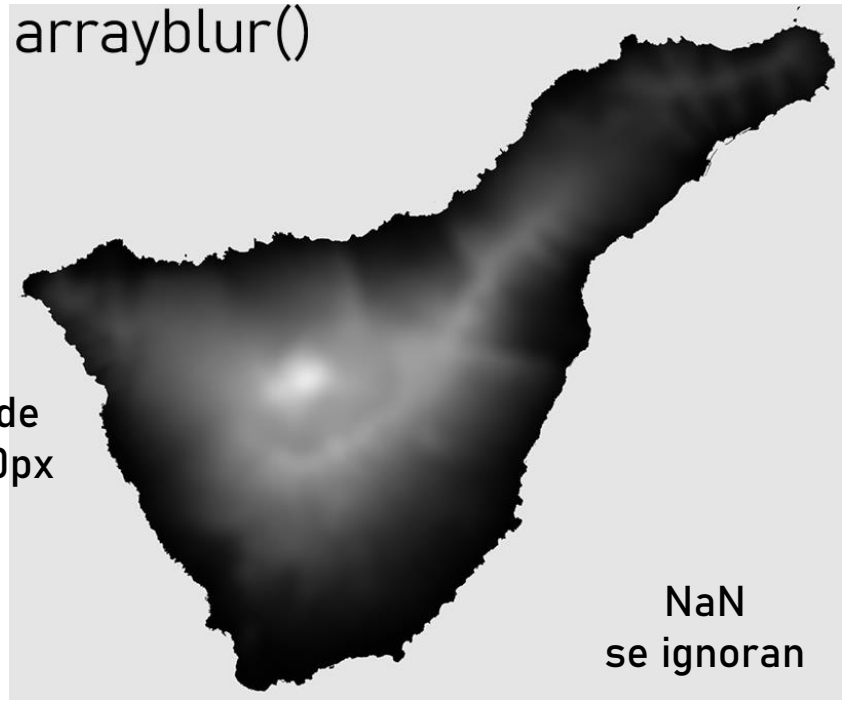
6. VIDEO

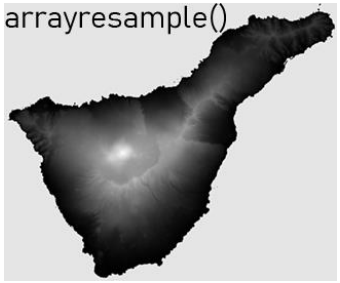
2. FUNCIONES AUXILIARES

arrayresample() **arrayblur()** solid() contour()



kernel de
radio=40px



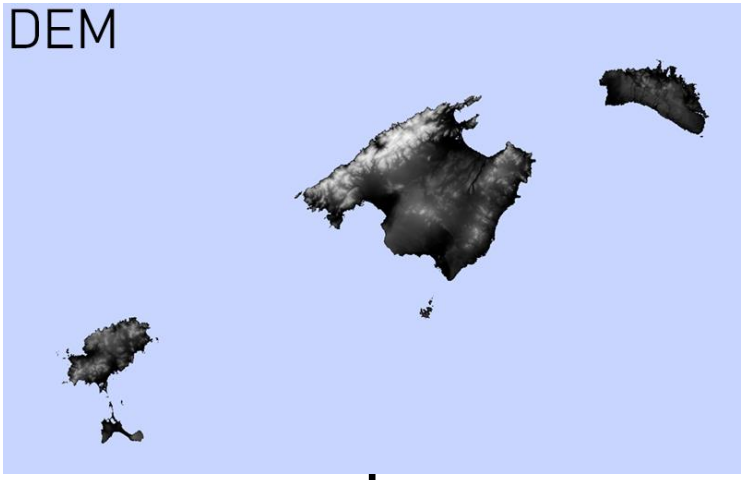



```
arrayresample=function(img, DIMX, DIMY, method='bilinear')
    # method=c('near', 'bilinear', 'cubic', 'cubicspline', 'lanczos')

arrayblur=function(img, radius=10)
    # radius: radius of the circular averaging window
```

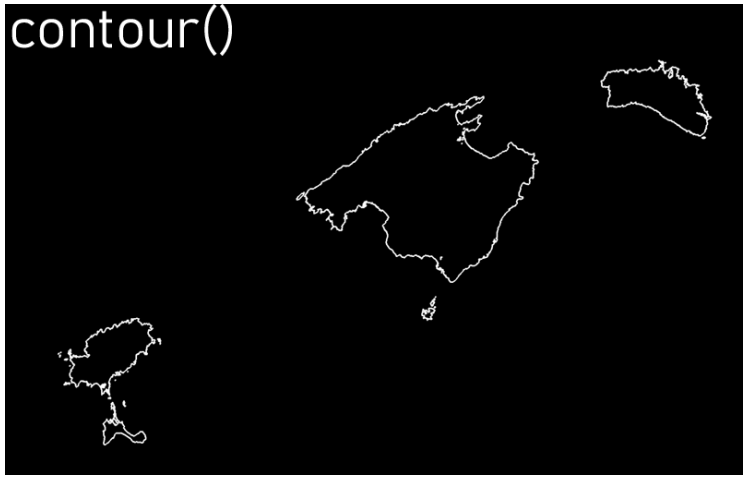
2. FUNCIONES AUXILIARES

arrayresample() arrayblur() **solid()** **contour()**




```
solid=function(DEM, altitude=0, isnan=0)
    # solid() calculates a solid version of a DEM
    #
    # altitude: DEM altitude level contour
    # isnan: value assigned to NaN data

contour=function(DEM, stroke=1)
    # contour() calculates the contours of any colour change
    #
    # stroke: line width (pixels)
```



Programamando mapas con R

1. INTRODUCCIÓN

arrayresample()
arrayblur()
solid()
contour()

2. FUNCIONES AUXILIARES

linemap()
circlemap()
joymap()
horizonmap()

3. MAPAS MINIMALISTAS

4. MAPAS PSEUDO 3D

aerialpersp()
slicemap()
hillshademap()
shadowmap()

5. MAPAS CREATIVOS

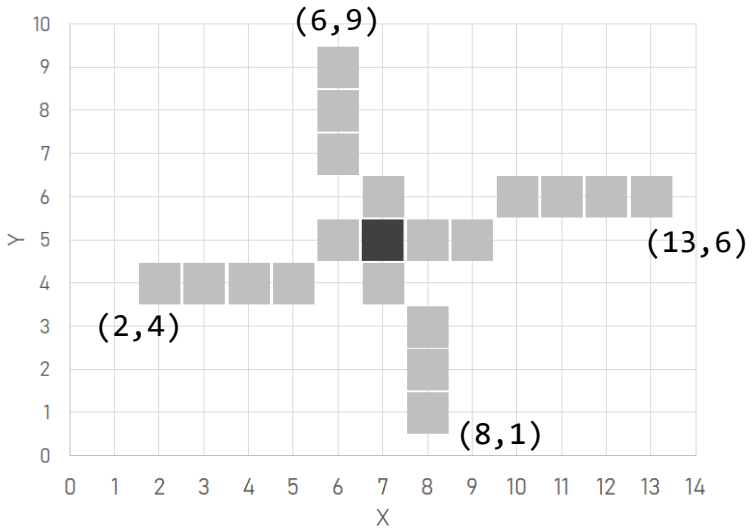
legomap()
foldmap()
karesansuimap()

6. VIDEO

3. MAPAS MINIMALISTAS

linemap() circlemap() joymap() horizonmap()

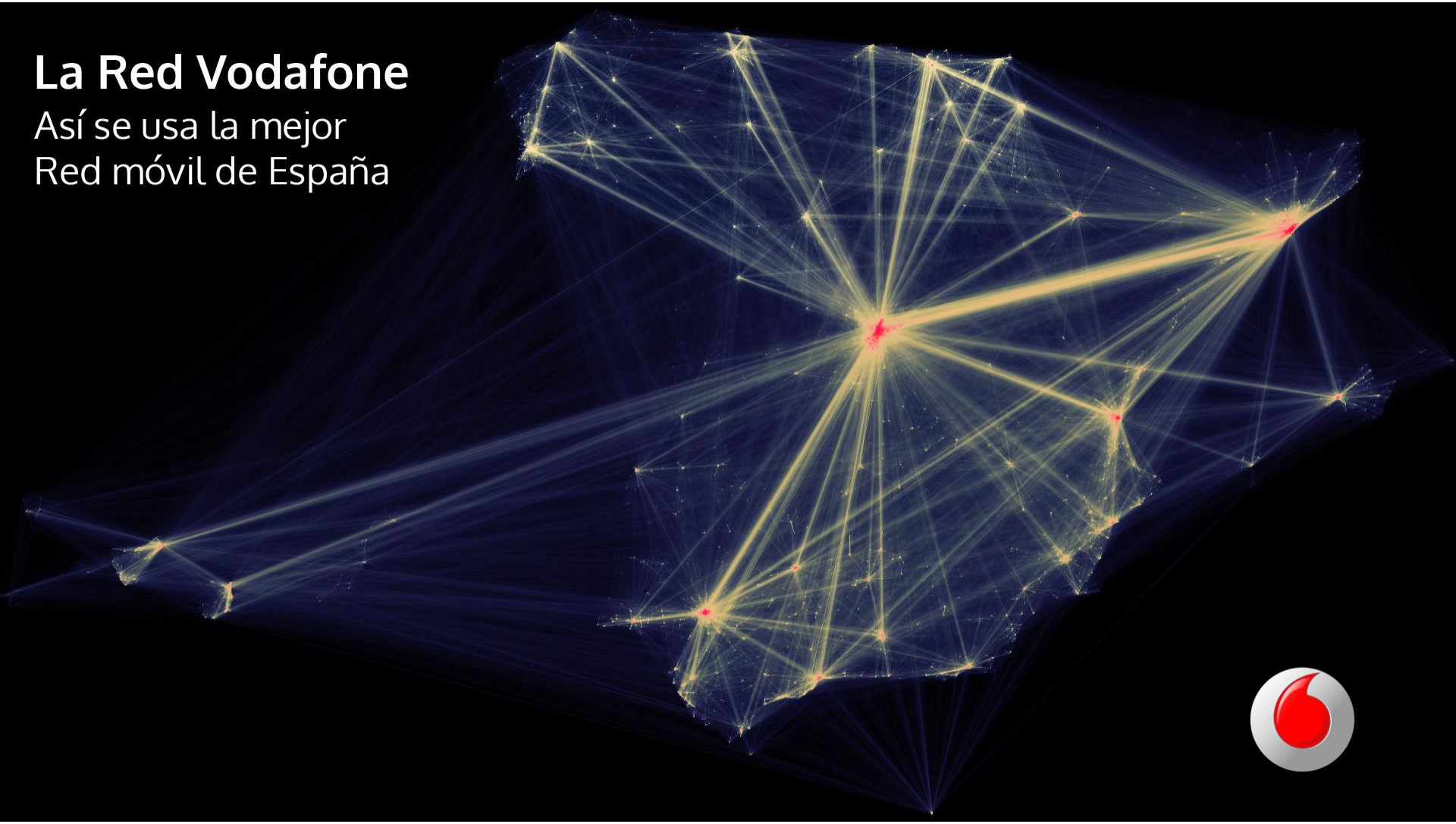
- Leemos coordenadas **origen/destino** desde un dataframe
- Acumulamos líneas sobre una matriz (algoritmo de **Bresenham**)
- El mapa “aparece” por **densidad** de datos




```r
# Por Carlos Gil Bellosta
indices.drawline=function(x0, y0, x1, y1) {
    x0=round(x0); y0=round(y0)
    x1=round(x1); y1=round(y1)

    if (y0 == y1) return(cbind(x0:x1, y0)) # Recta de m=0 o un punto
    if (abs(x1 - x0) >= abs(y1 - y0)) { # Recta de 0 < |m| <= 1
        m = (y1 - y0) / (x1 - x0)
        cbind(x0:x1, round(y0 + m * ((x0:x1) - x0)))
    } else indices.drawline(y0, x0, y1, x1)[, 2:1]  # Recta de |m| > 1
    # Llamada traspuesta recursiva y traspuesta
}
```

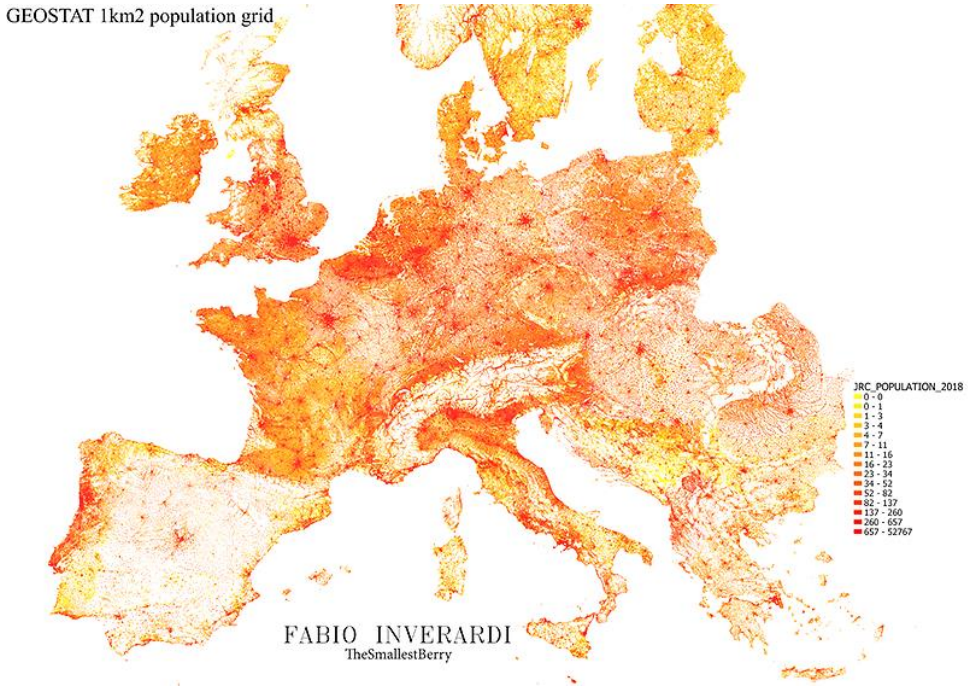
150
million calls
make a map

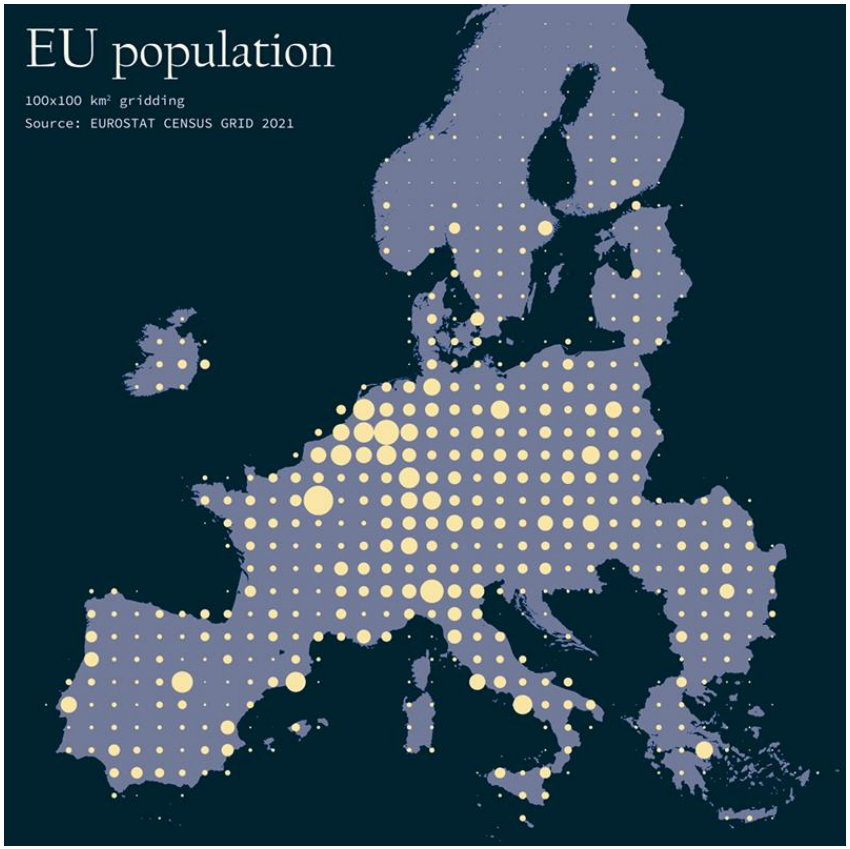

La Red Vodafone
Así se usa la mejor
Red móvil de España

3. MAPAS MINIMALISTAS

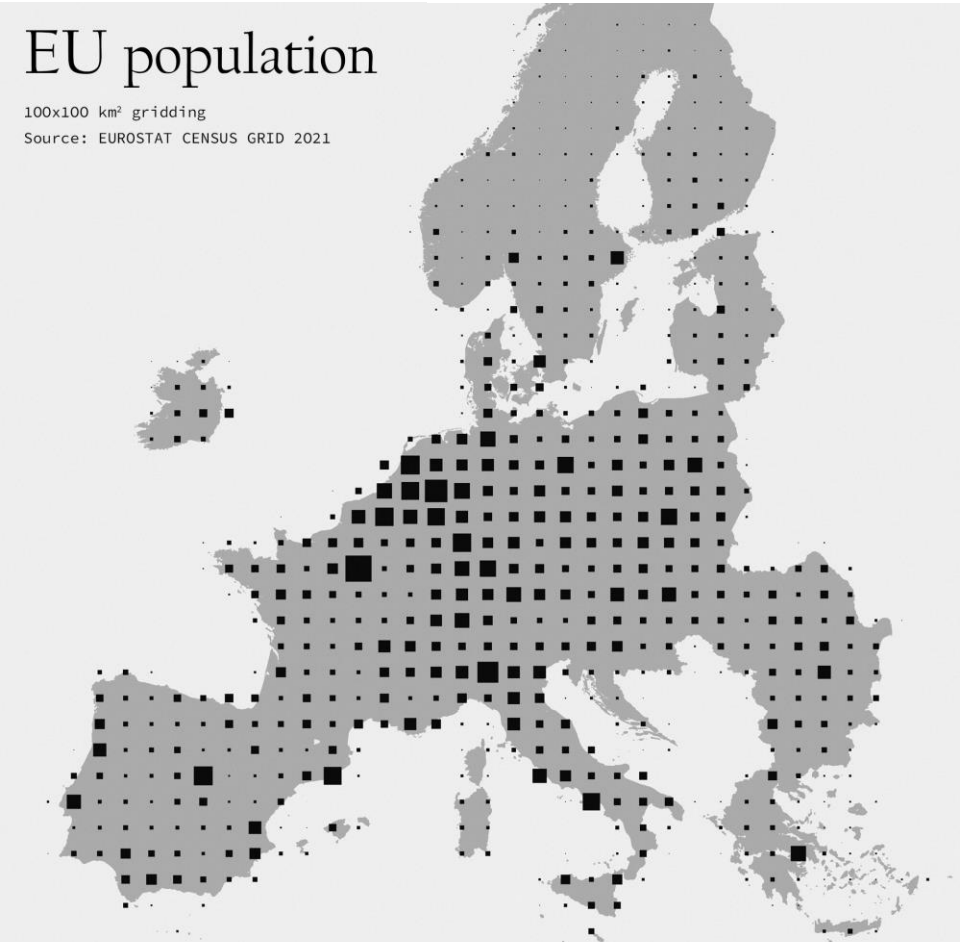
`linemap()` **`circlemap()`** `joymap()` `horizonmap()`

- El **área** del círculo es **proporcional** a la magnitud representada
- Mejor **interpretación** de las relaciones numéricas espaciales de una variable



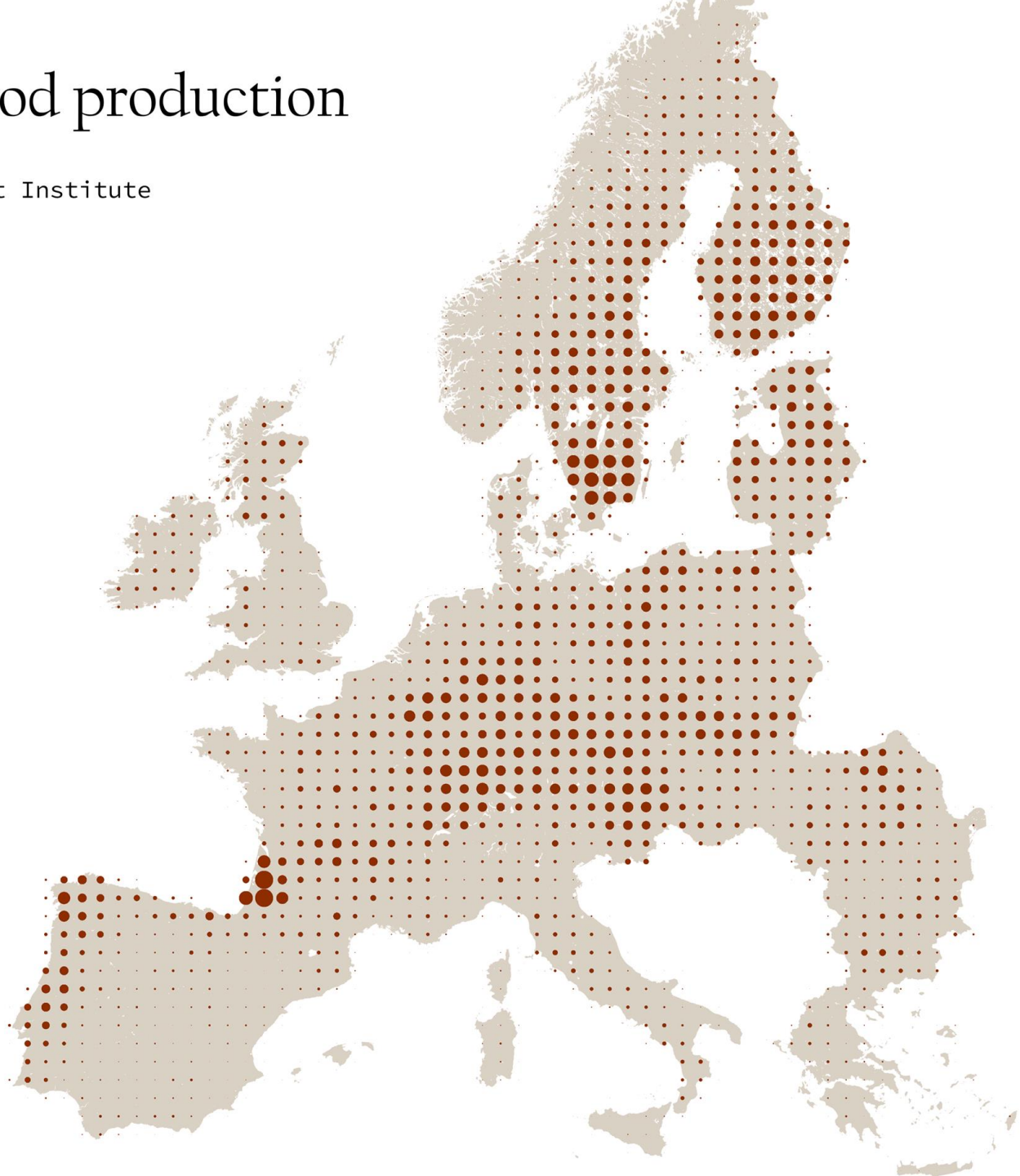


```r
circlemap=function(map,
          shape='circle', shapestyle='solid', mapstyle='solid', grid='none',
          inwidth=100, outwidth=100, overlap=1, allownegative=FALSE, gamma=1) {
    # map must be a matrix where...
    #   <0 values will be previosly set to NA if not allowed
    #   >0 and <0 values will be summed
    #   NA and <0 values will define the geographical limits of the map
    # shape=c('circle', 'square', 'none')
    # shapestyle=c('solid', 'outline', 'none')
    # mapstyle=c('solid', 'outline', 'none')
    # grid=c('none', 'centre', 'wrap')
    # inwidth: input grid size in pixels
    # outwidth: output grid size in pixels (inwidth=outwidth avoids resampling)
    # overlap: how much a square/circle can overlap its neighbours
    # allownegative=TRUE: <0 values allowed and plotted in different colour
    # gamma: output gamma lift curve
```



European wood production

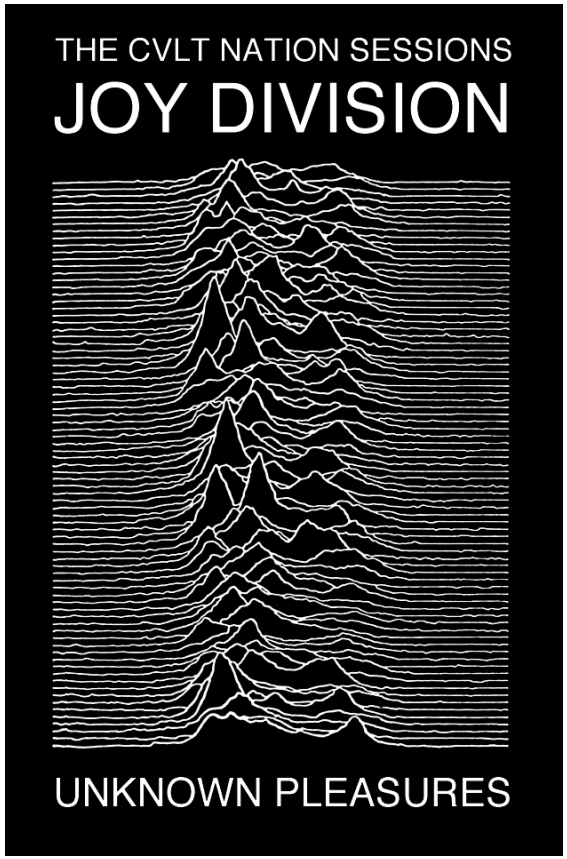
60x60 km² gridding
Source: European Forest Institute

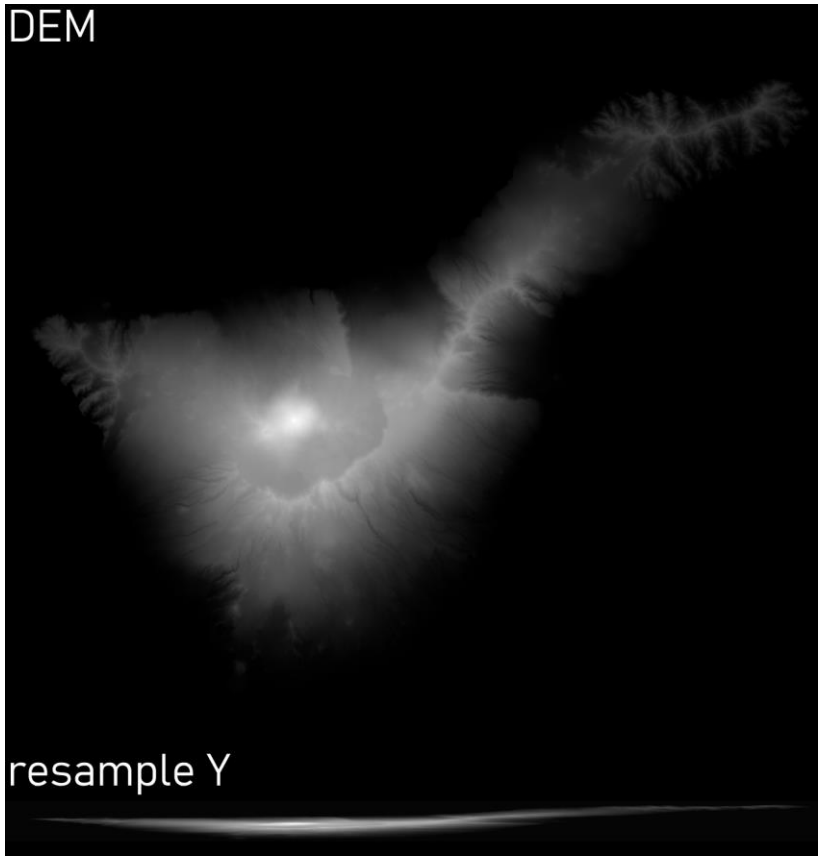


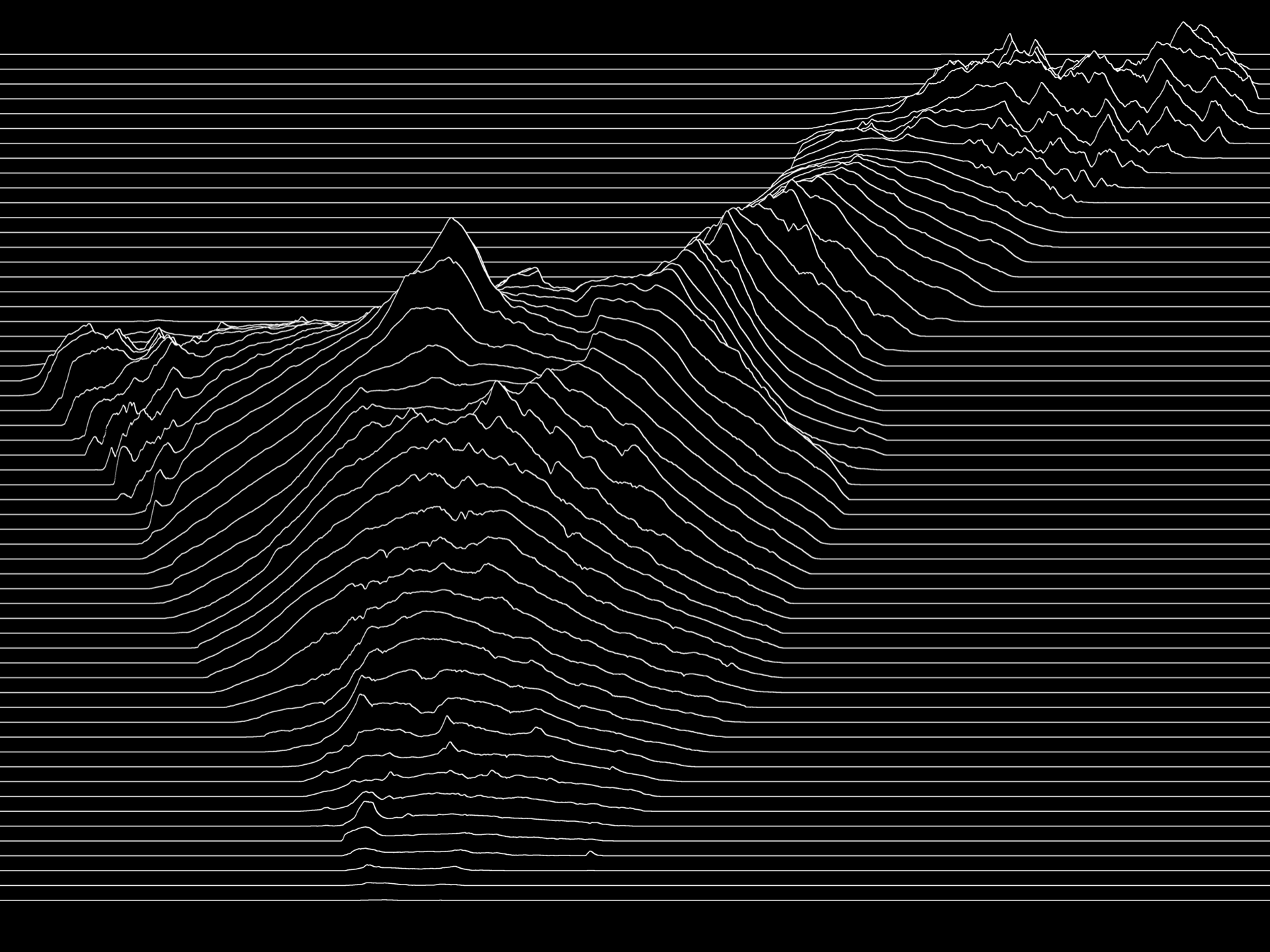
3. MAPAS MINIMALISTAS

`linemap()` `circlemap()` **`joymap()`** `horizonmap()`

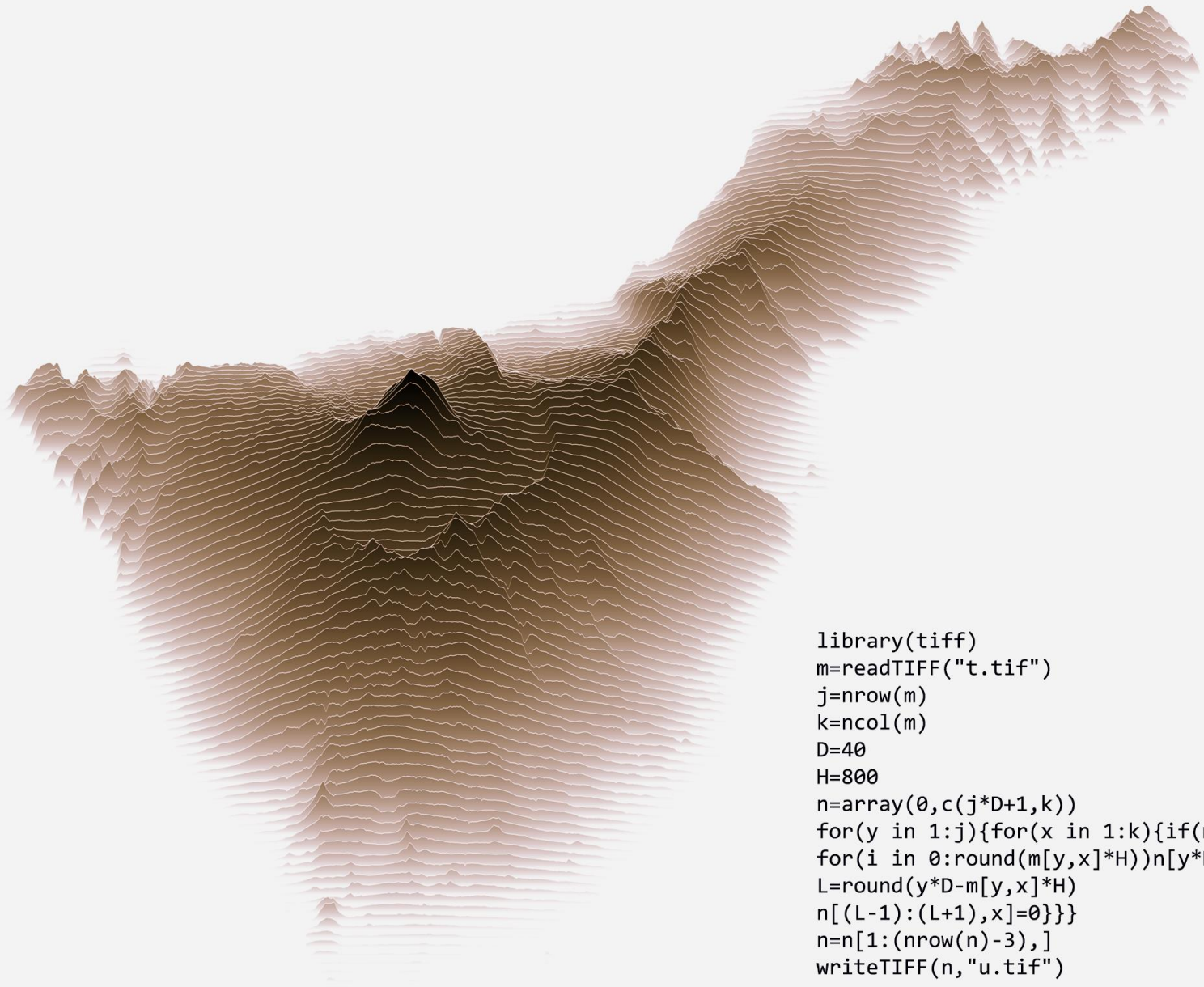
- Mapa inspirado en la portada de **Joy Division**
- **Reducción de resolución** espacial en un eje







Coffee Solid Tenerife

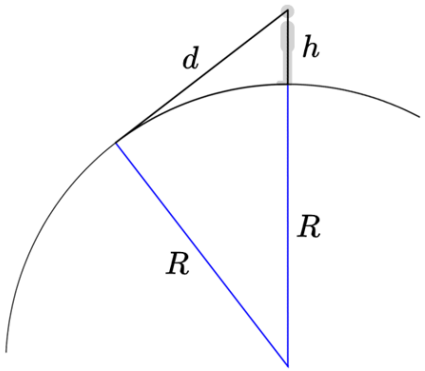


```
library(tiff)
m=readTIFF("t.tif")
j=nrow(m)
k=ncol(m)
D=40
H=800
n=array(0,c(j*D+1,k))
for(y in 1:j){for(x in 1:k){if(m[y,x]){
for(i in 0:round(m[y,x]*H))n[y*D-i,x]=i/H
L=round(y*D-m[y,x]*H)
n[(L-1):(L+1),x]=0}}}
n=n[1:(nrow(n)-3),]
writeTIFF(n,"u.tif")
```

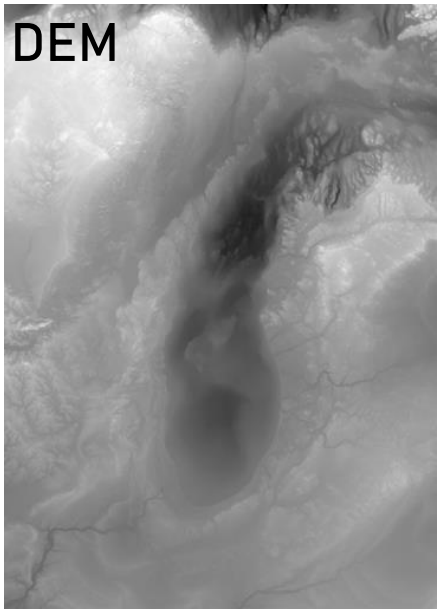
3. MAPAS MINIMALISTAS

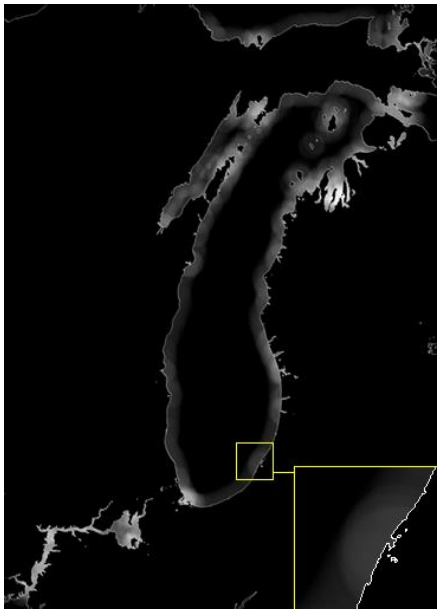
linemap() circlemap() joymap() **horizonmap()**

$$d = \sqrt{(R+h)^2 - R^2}$$

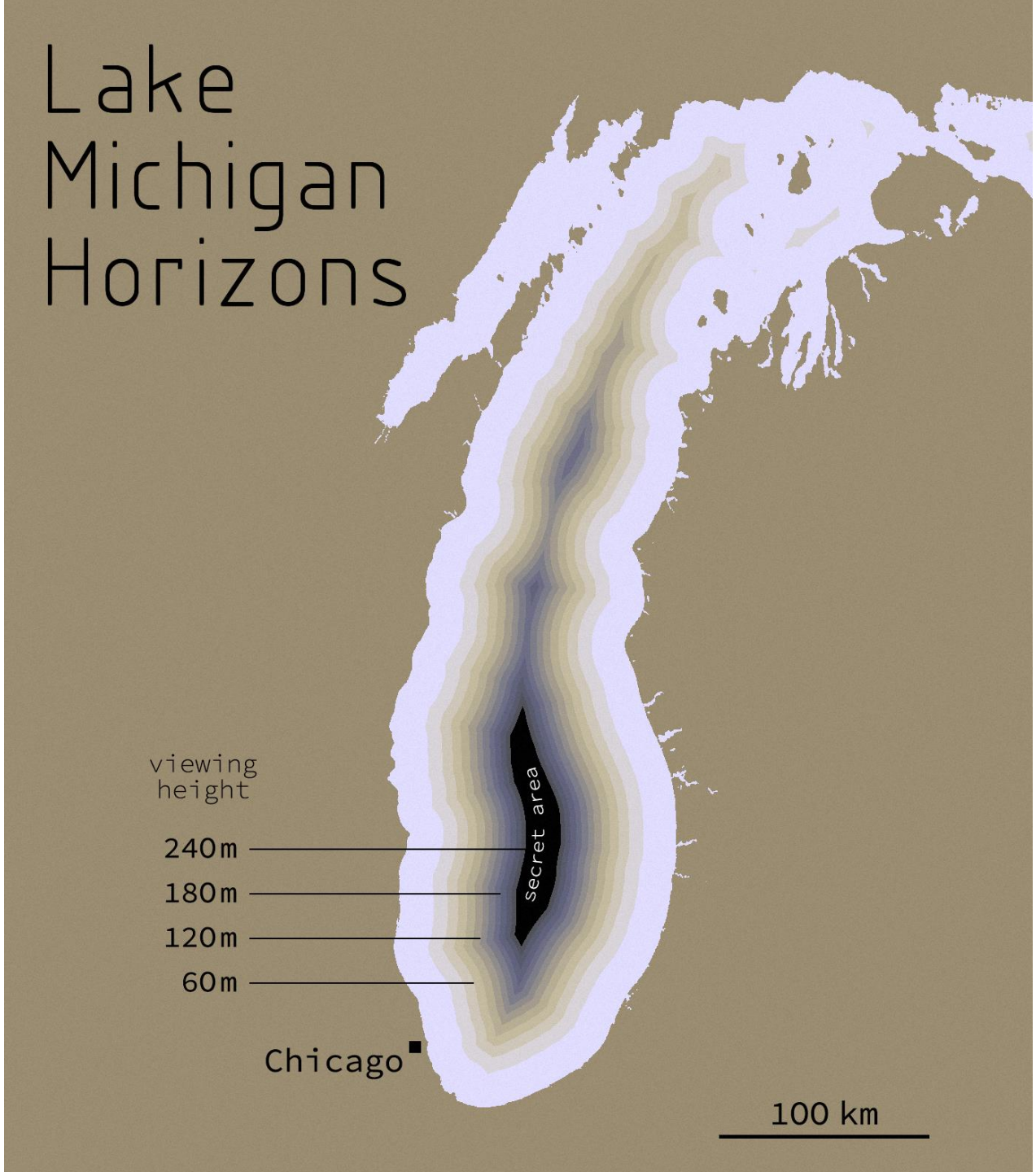


Superponemos por fuerza bruta todos los posibles círculos de radio **d** con centro sobre los píxeles que componen el contorno del lago → envolvente del horizonte





Lake Michigan Horizons



Programando mapas con R

1. INTRODUCCIÓN

arrayresample()
arrayblur()
solid()
contour()

2. FUNCIONES AUXILIARES

linemap()
circlemap()
joymap()
horizonmap()

3. MAPAS MINIMALISTAS

4. MAPAS PSEUDO 3D

aerialpersp()
slicemap()
hillshademap()
shadowmap()

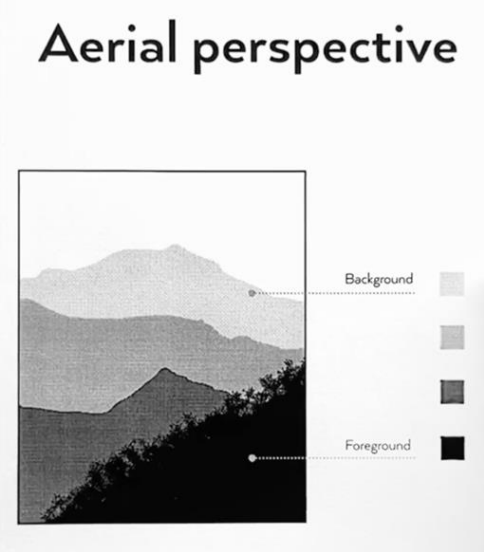
5. MAPAS CREATIVOS

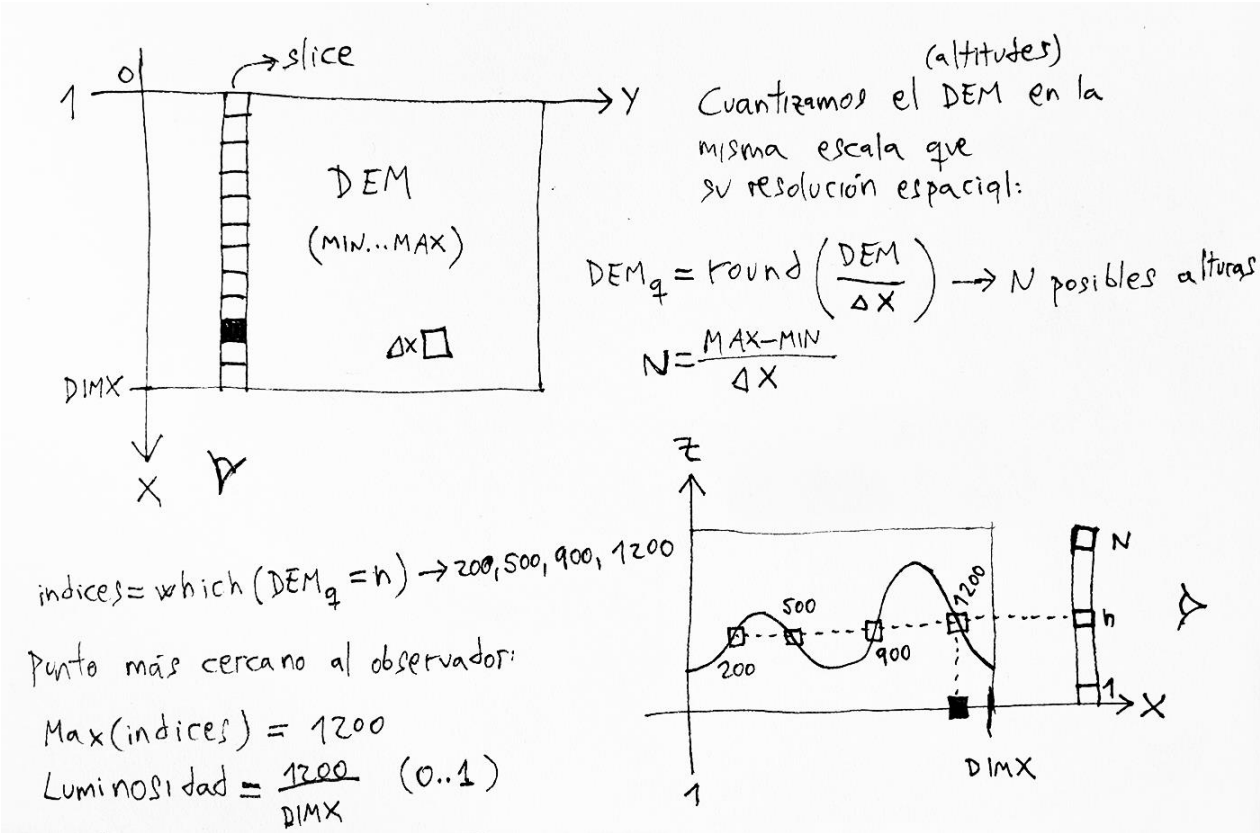
legomap()
foldmap()
karesansuimap()

6. VIDEO

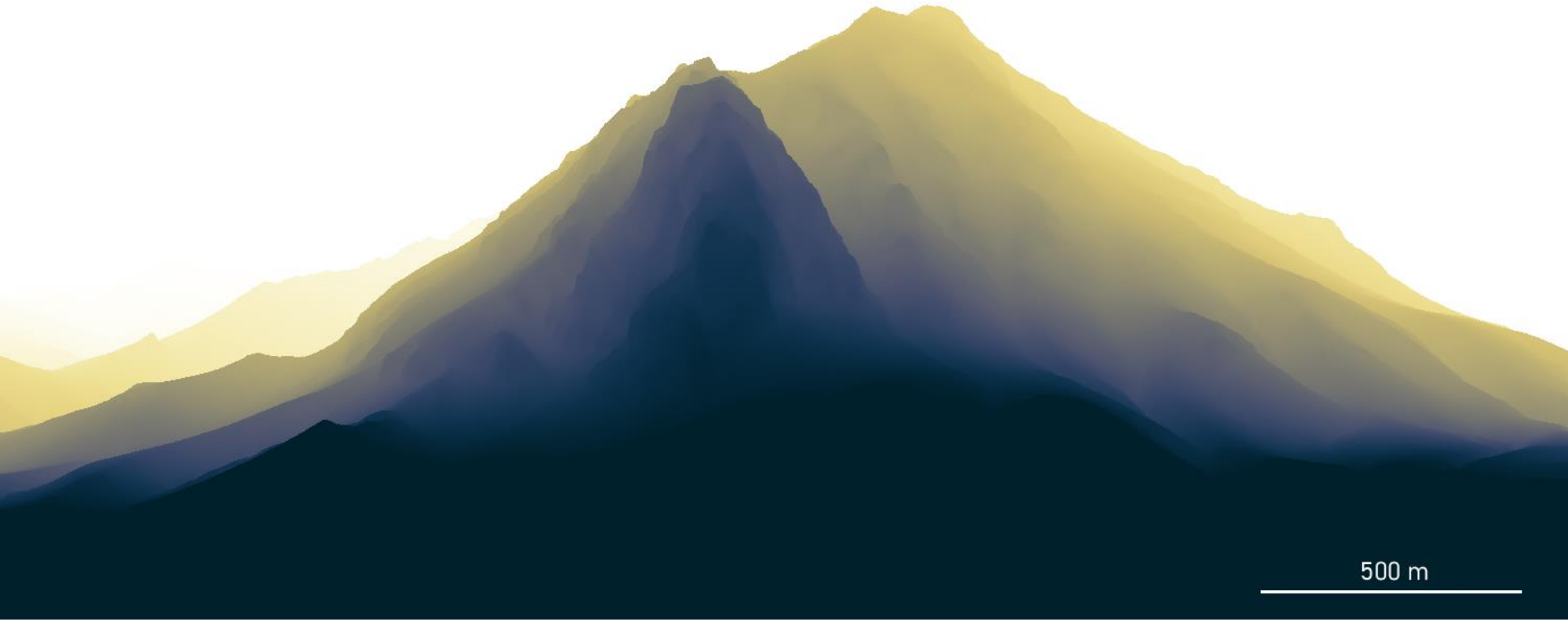
4. MAPAS PSEUDO 3D

aerialpersp() slicemap() hillshademap() shadowmap()

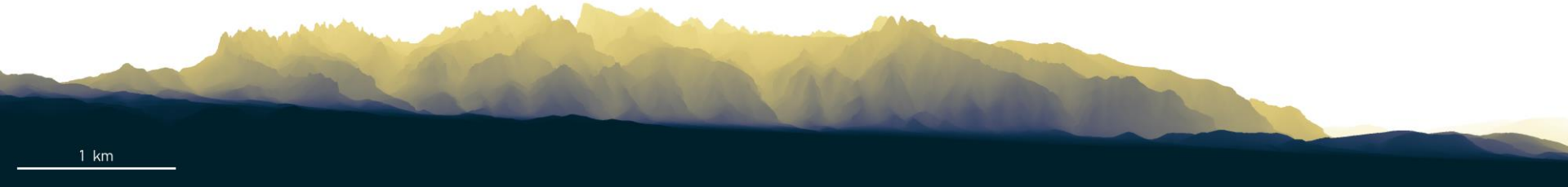




Puig Campana
(Alicante)

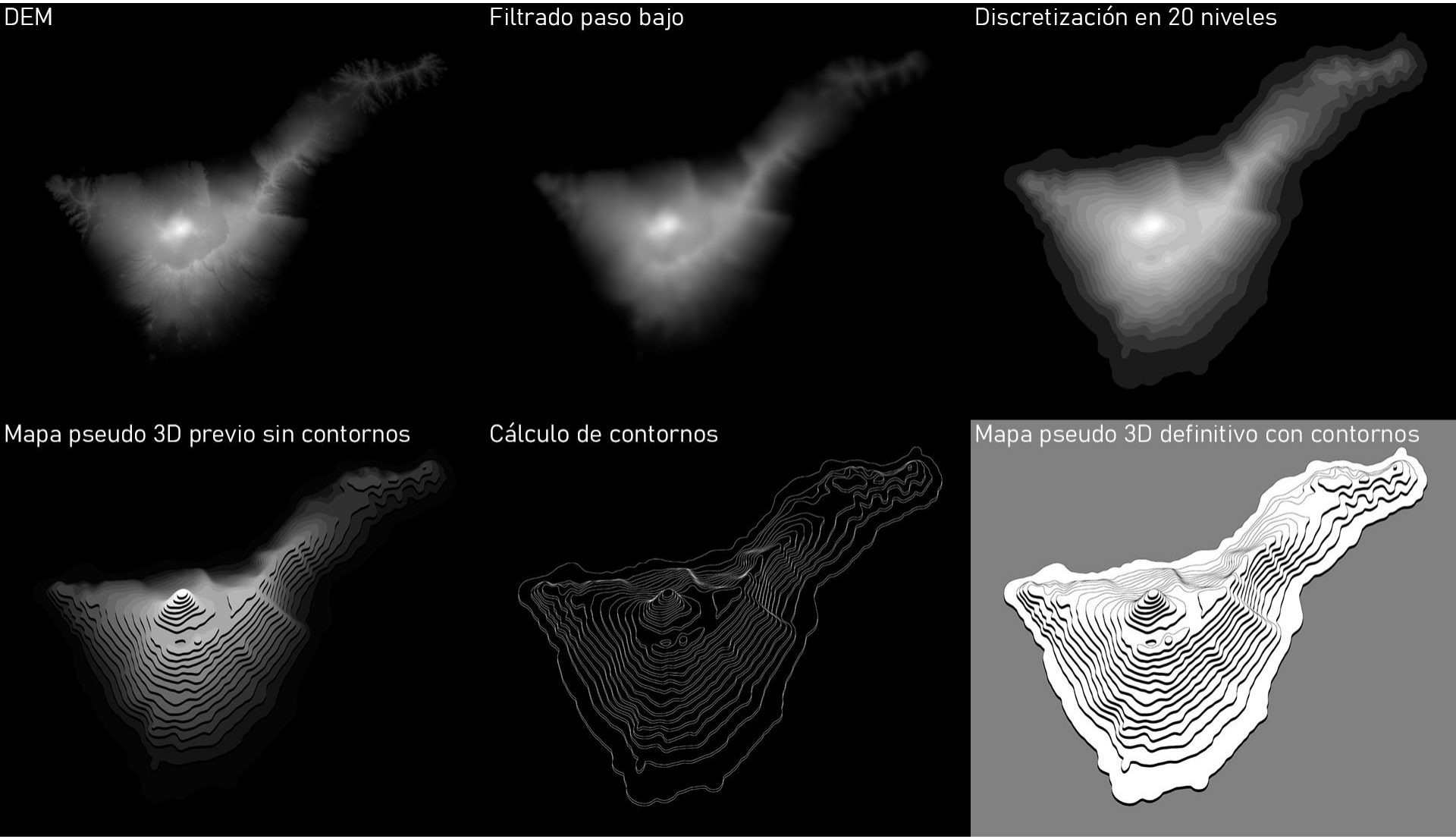


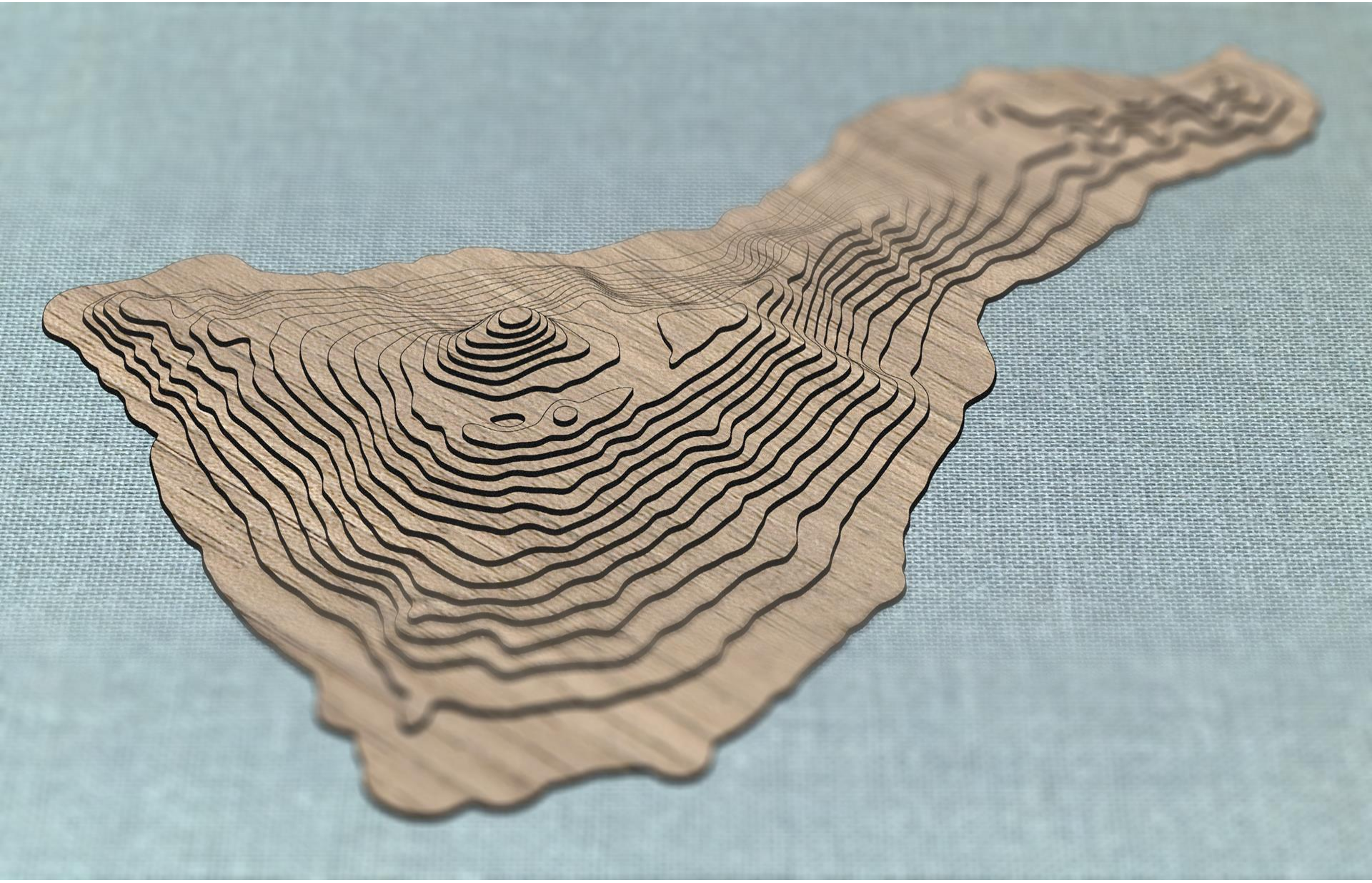
Muntanya de Montserrat
(cara sud)

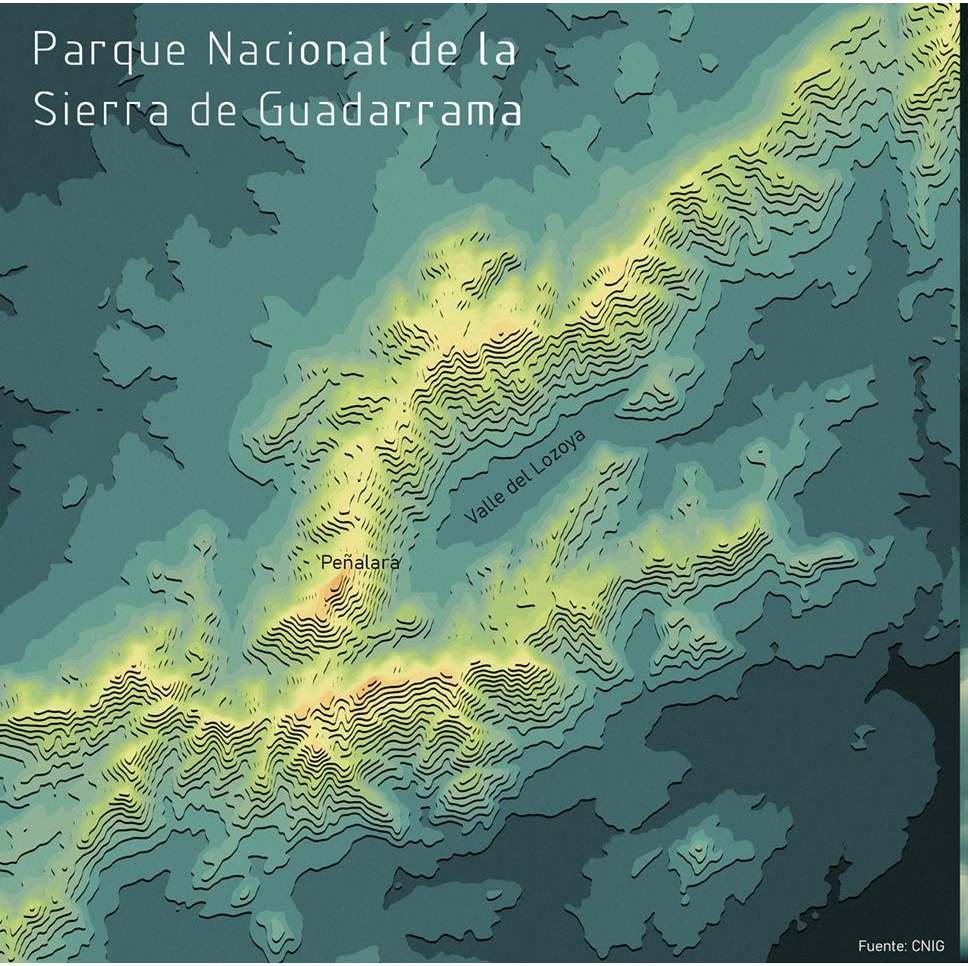


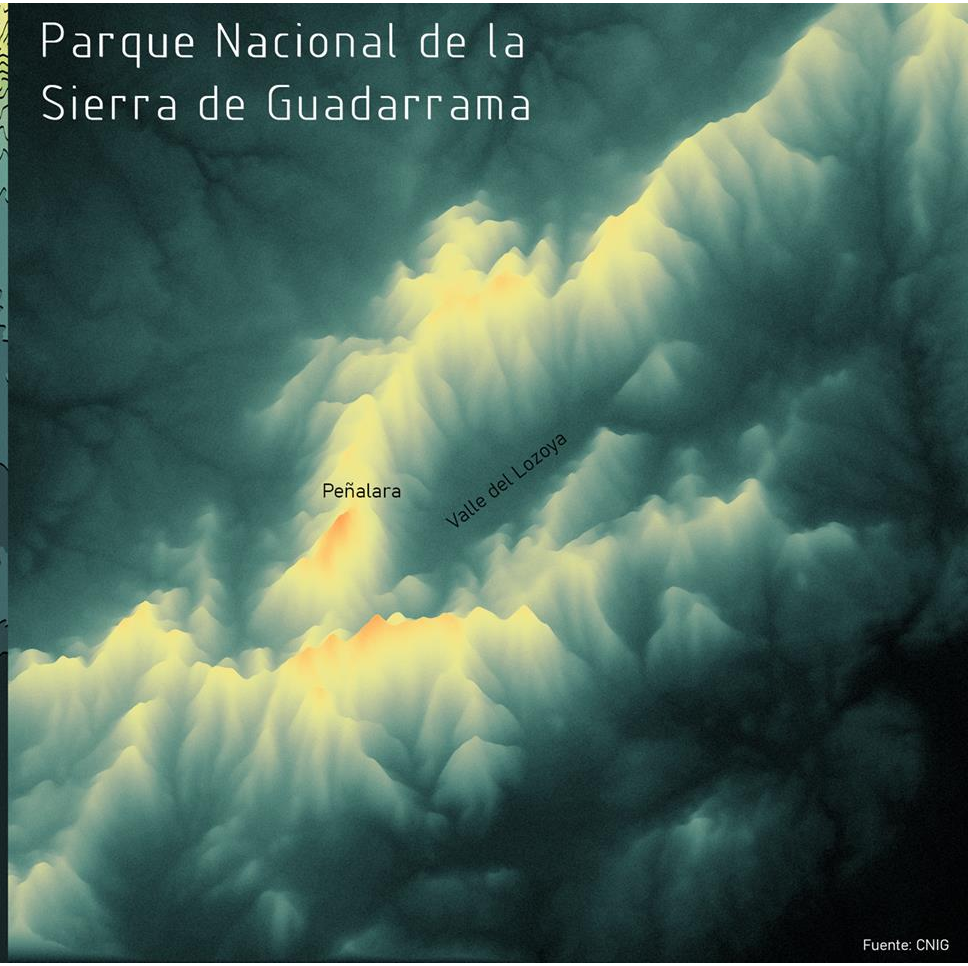
4. MAPAS PSEUDO 3D

`aerialpersp()` **`slicemap()`** `hillshademap()` `shadowmap()`



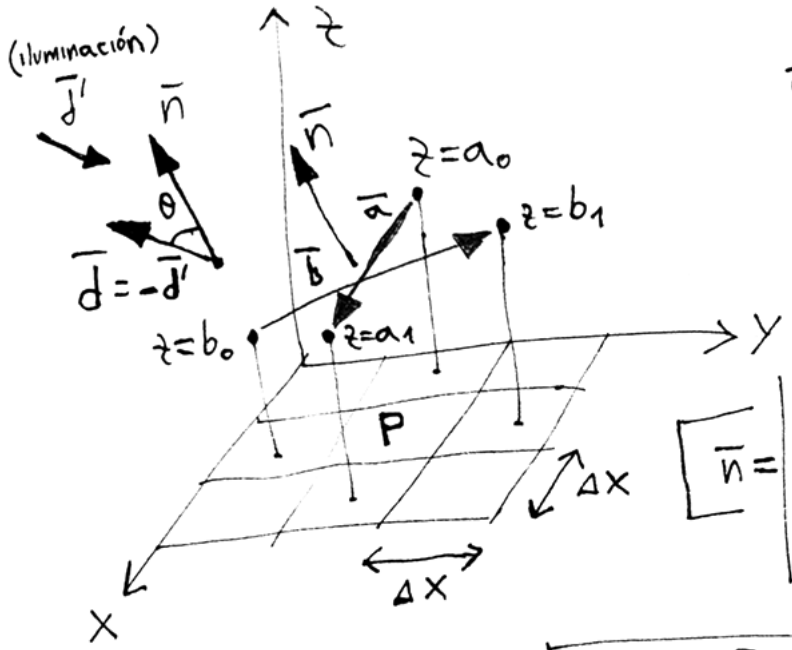



Parque Nacional de la
Sierra de Guadarrama
Valle del Lozoya
Peñalara
Fuente: CNIG


Parque Nacional de la
Sierra de Guadarrama
Peñalara
Valle del Lozoya
Fuente: CNIG

4. MAPAS PSEUDO 3D

aerialpersp() slicemap() **hillshademap()** shadowmap()



$\bar{n} = \bar{a} \wedge \bar{b}$ → Vector normal al terreno en P

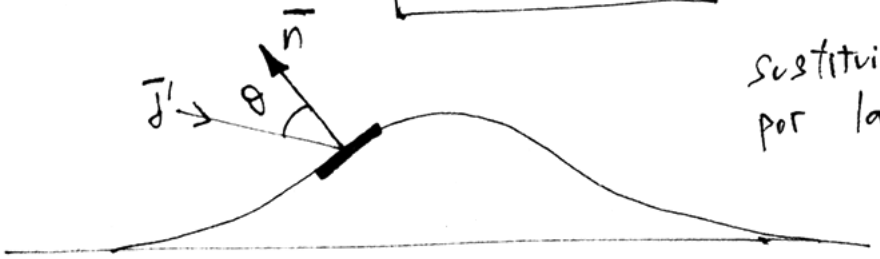
$\bar{a} = 2 \cdot \Delta x \cdot \hat{x} + (a_1 - a_0) \cdot \hat{z}$

$\bar{b} = 2 \cdot \Delta x \cdot \hat{y} + (b_1 - b_0) \cdot \hat{z}$

$$\bar{n} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ 2\cdot\Delta x & 0 & a_1 - a_0 \\ 0 & 2\cdot\Delta x & b_1 - b_0 \end{vmatrix} = 4\cdot\Delta x^2 \cdot \hat{z} + \left(2\cdot\Delta x \cdot (a_1 - a_0)\hat{x} + 2\cdot\Delta x \cdot (b_1 - b_0)\hat{y}\right) =$$

$$= 2\cdot\Delta x \cdot [a_0 - a_1,\ b_0 - b_1,\ 2\cdot\Delta x]$$

$$\bar{d} \cdot \bar{n} = |\bar{d}| \cdot |\bar{n}| \cdot \cos\theta \ // \ \boxed{\cos\theta = \frac{\bar{d} \cdot \bar{n}}{|\bar{d}| \cdot |\bar{n}|}}$$



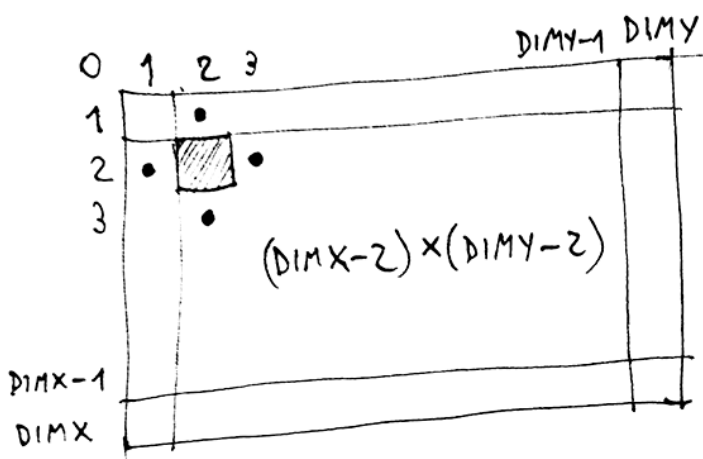
Sustituimos los datos de elevación por la luminosidad reflejada ($= \cos\theta$)

↓

$[0, 1]$

4. MAPAS PSEUDO 3D

aerialpersp() slicemap() **hillshademap()** shadowmap()



$\bar{d} = d_x \cdot \hat{x} + d_y \cdot \hat{y} \cdot d_z \cdot \hat{z} = cte.\ en\ XY \rightarrow |\bar{d}|$

1 matriz

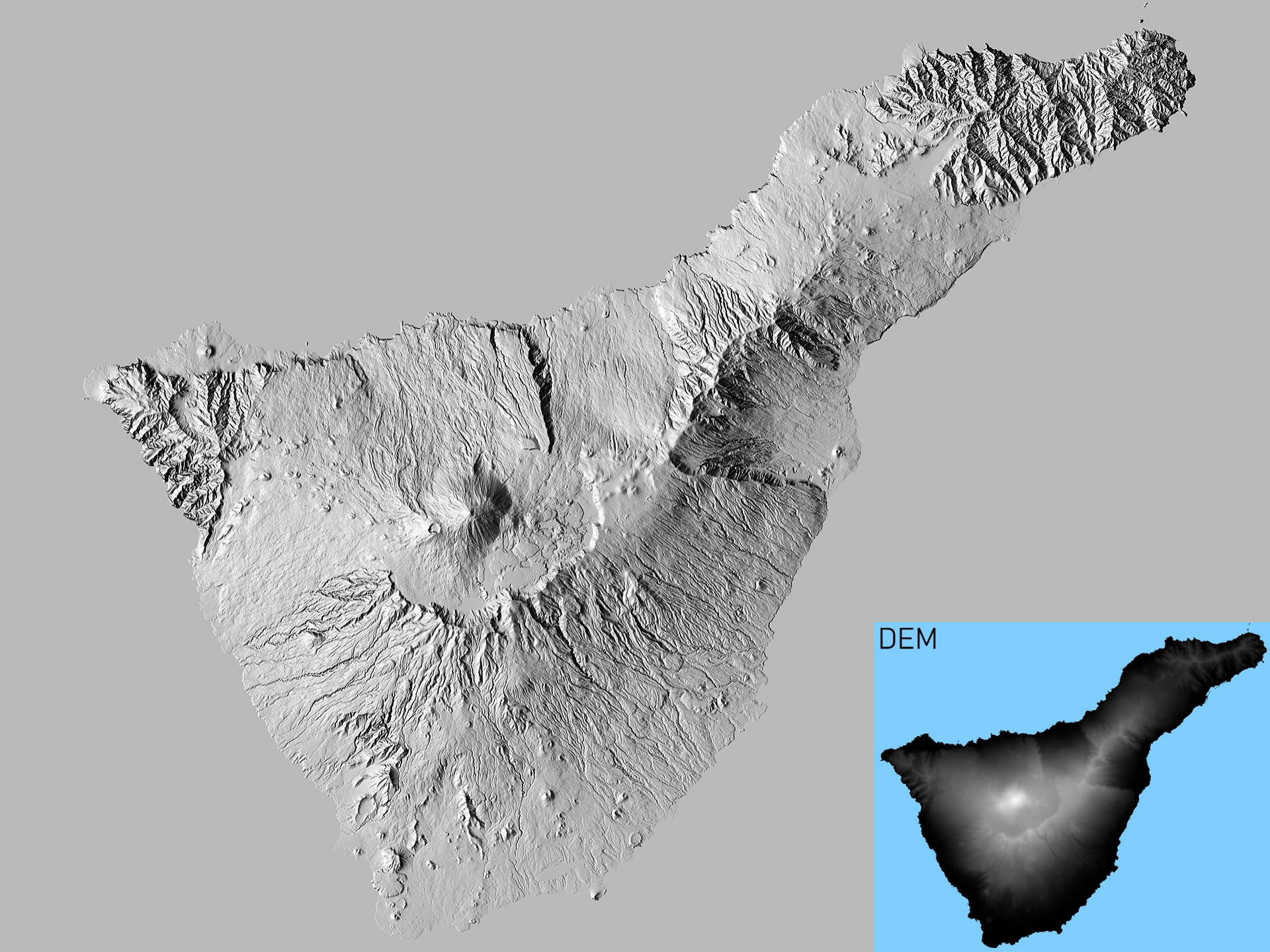
$dim(DEM) = DIMX \times DIMY$

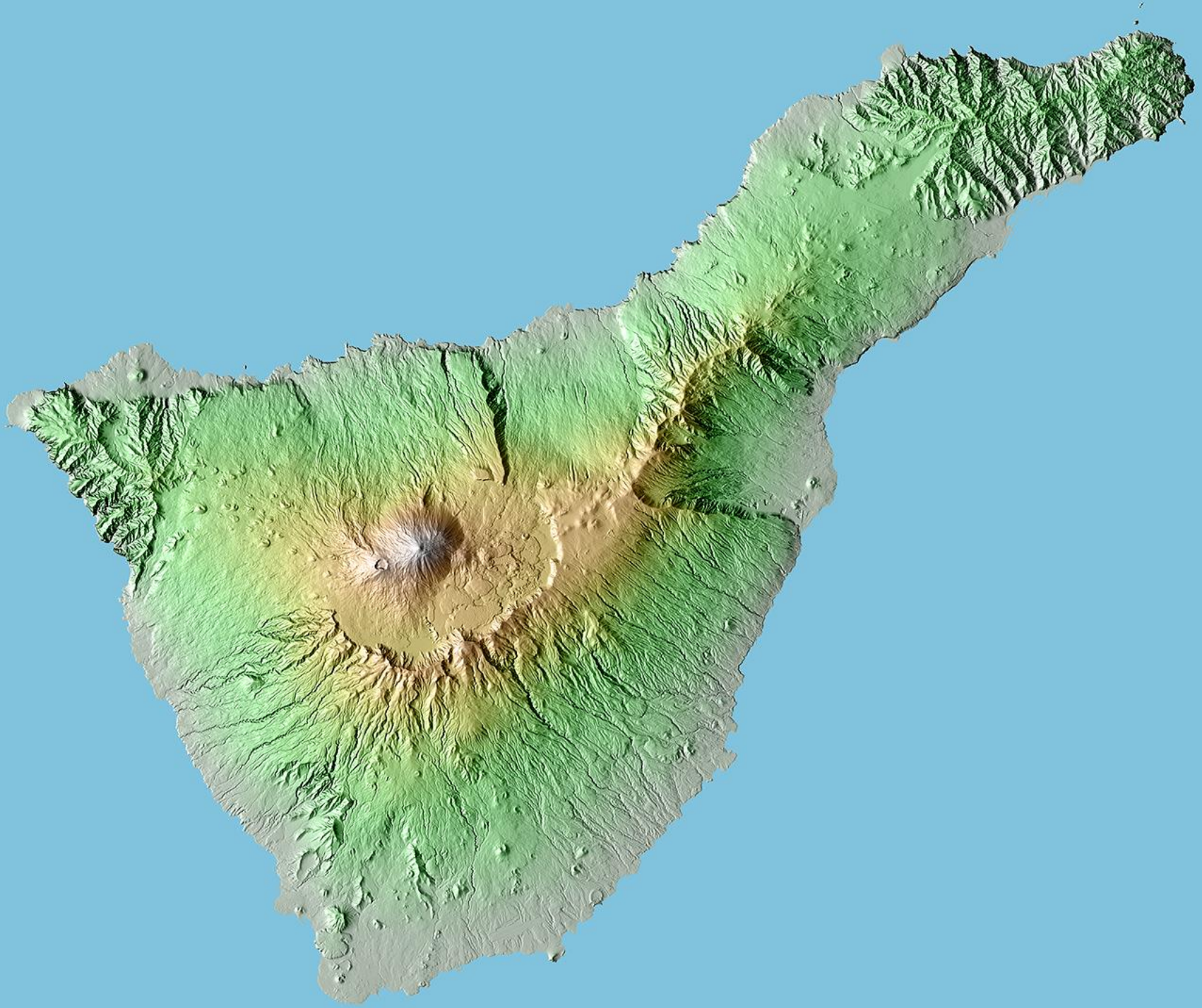
$dim(n_{x,y,z}) = (DIMX-2) \times (DIMY-2)$

↳ 3 matrices

$$\bar{n} \begin{cases} \bullet\ n_x = 2 \cdot \Delta x \cdot \left(DEM[1:(DIMX-2),\ 2:(DIMY-1)] - DEM[3:DIMX,\ 2:(DIMY-1)] \right) \\ \bullet\ n_y = 2 \cdot \Delta x \cdot \left(DEM[2:(DIMX-1),\ 1:(DIMY-2)] - DEM[2:(DIMX-1),\ 3:DIMY] \right) \\ \bullet\ n_z = 4 \cdot \Delta x^2 \big|_{(DIMX-2)\times(DIMY-2)} = cte.\ en\ XY \end{cases} \rightarrow |\bar{n}| \Big|_{(DIMX-2)\times(DIMY-2)}$$

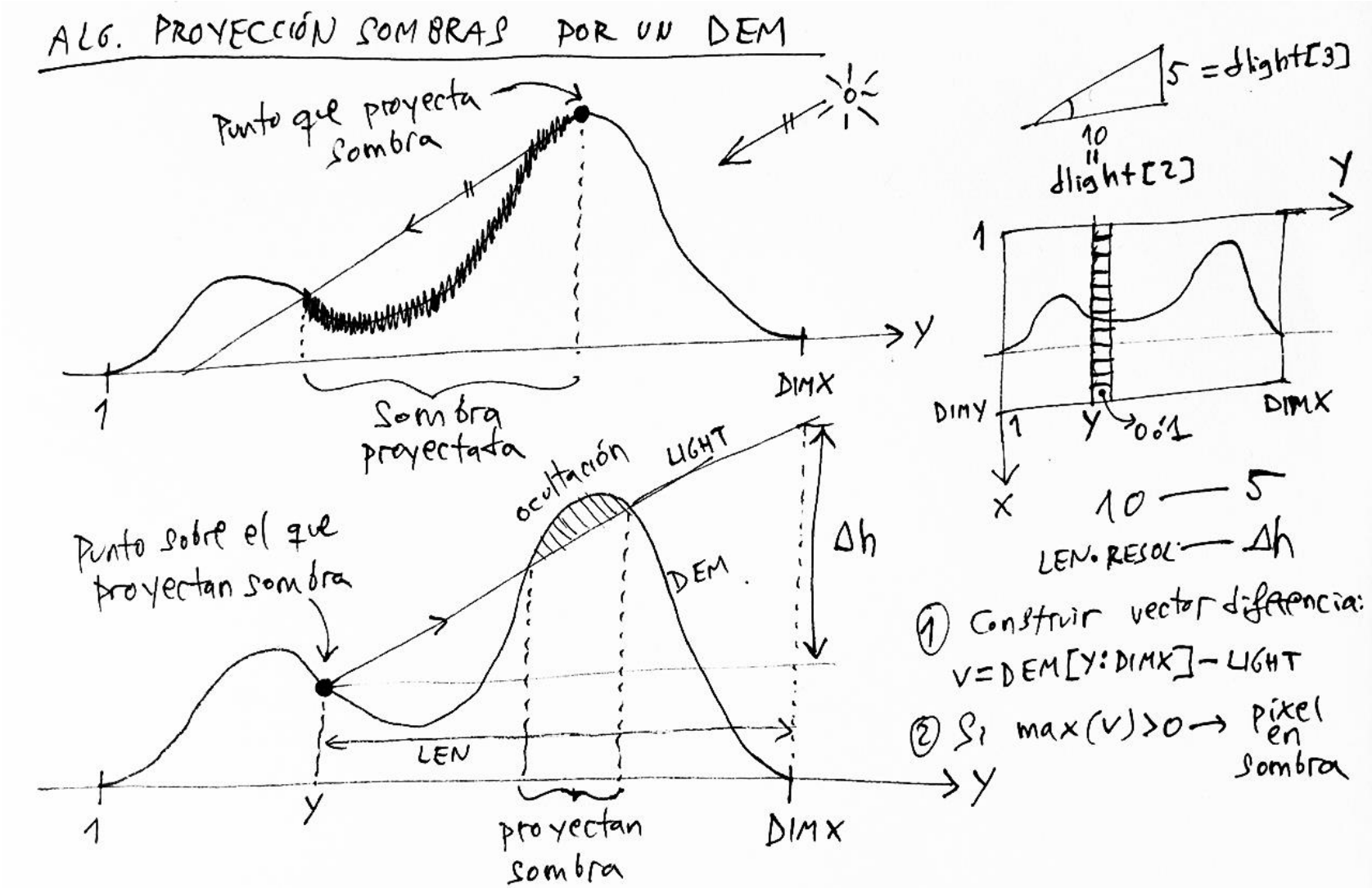
$$\bar{d} \cdot \bar{n} = d_x \cdot n_x + d_y \cdot n_y + d_z \cdot n_z \Big|_{(DIMX-2)\times(DIMY-2)} \qquad |\bar{n}| = \left(n_x^2 + n_y^2 + n_z^2 \right)^{1/2} \Big|_{(DIMX-2)\times(DIMY-2)}$$


DEM



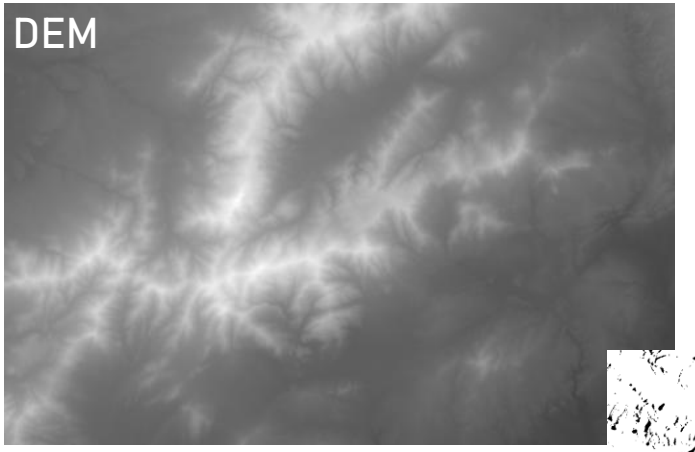
4. MAPAS PSEUDO 3D

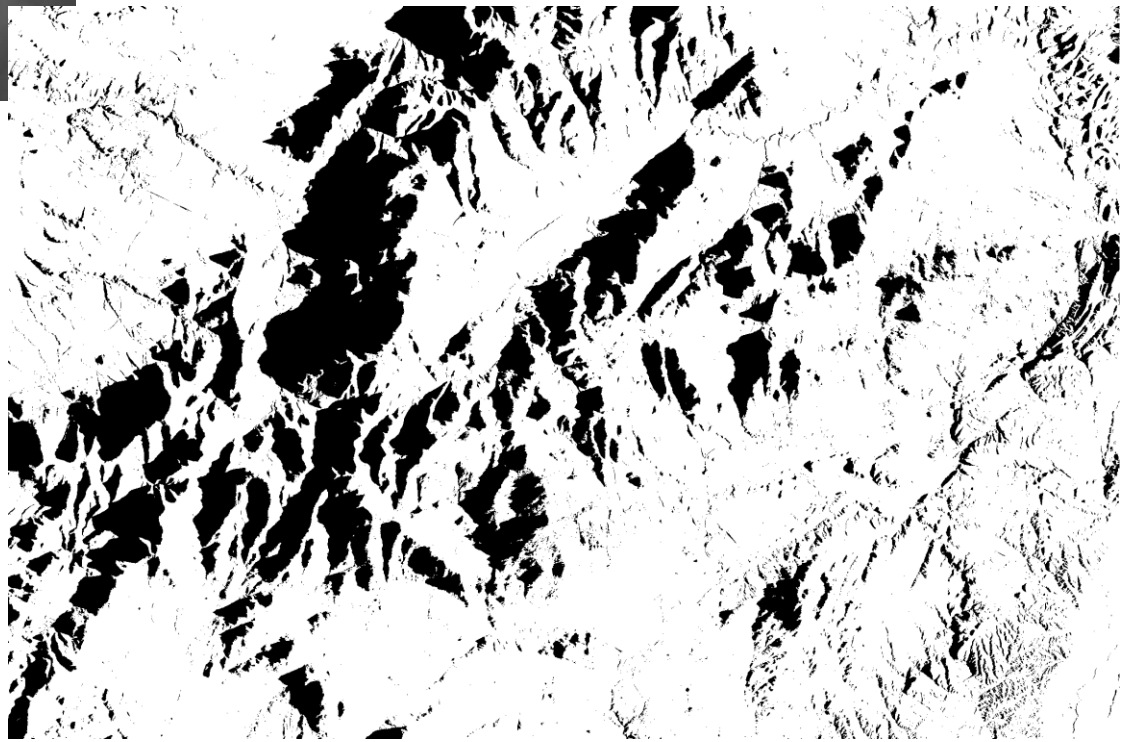
aerialpersp() slicemap() hillshademap() **shadowmap()**

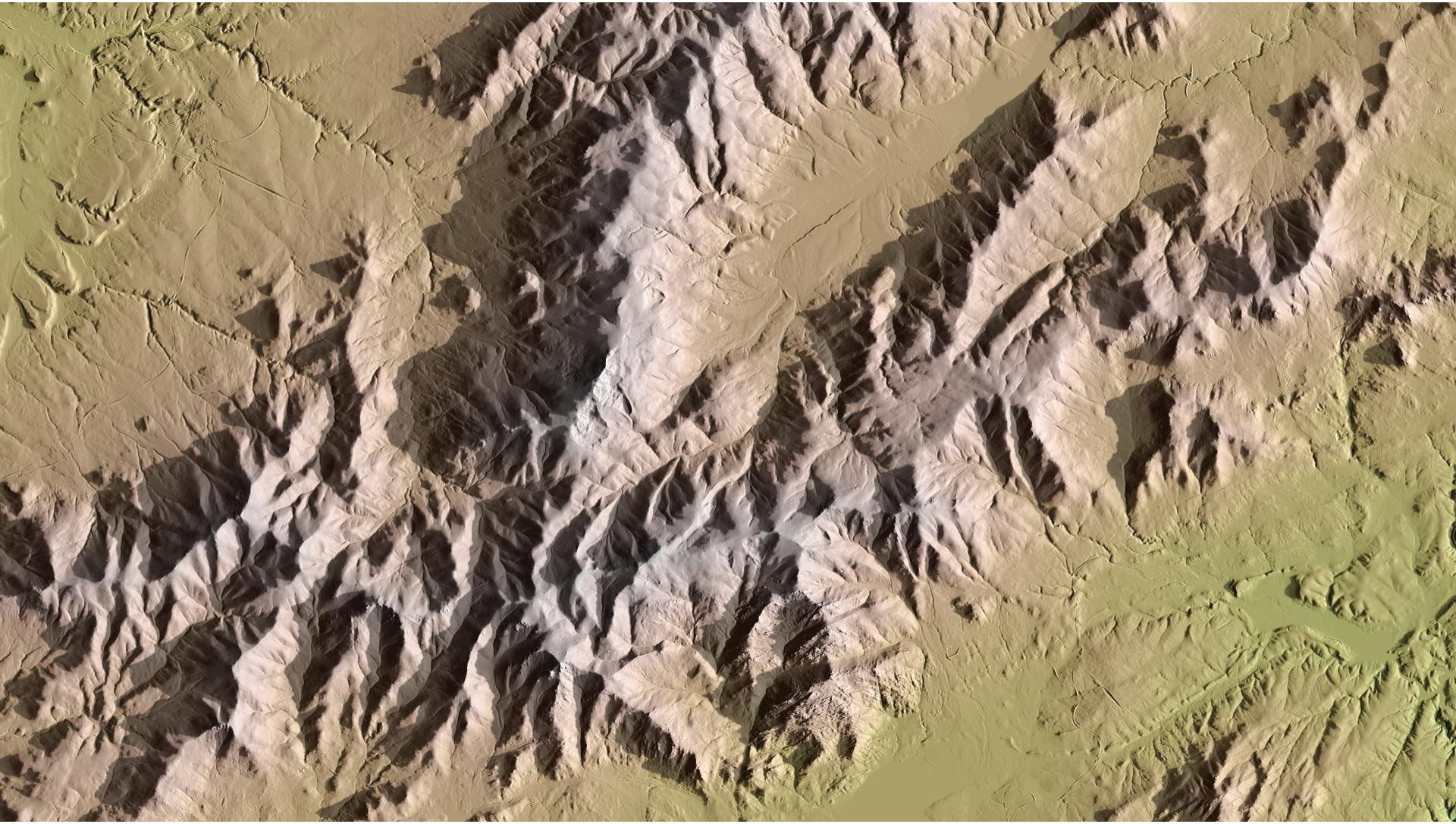


4. MAPAS PSEUDO 3D

`aerialpersp()` `slicemap()` `hillshademap()` **`shadowmap()`**

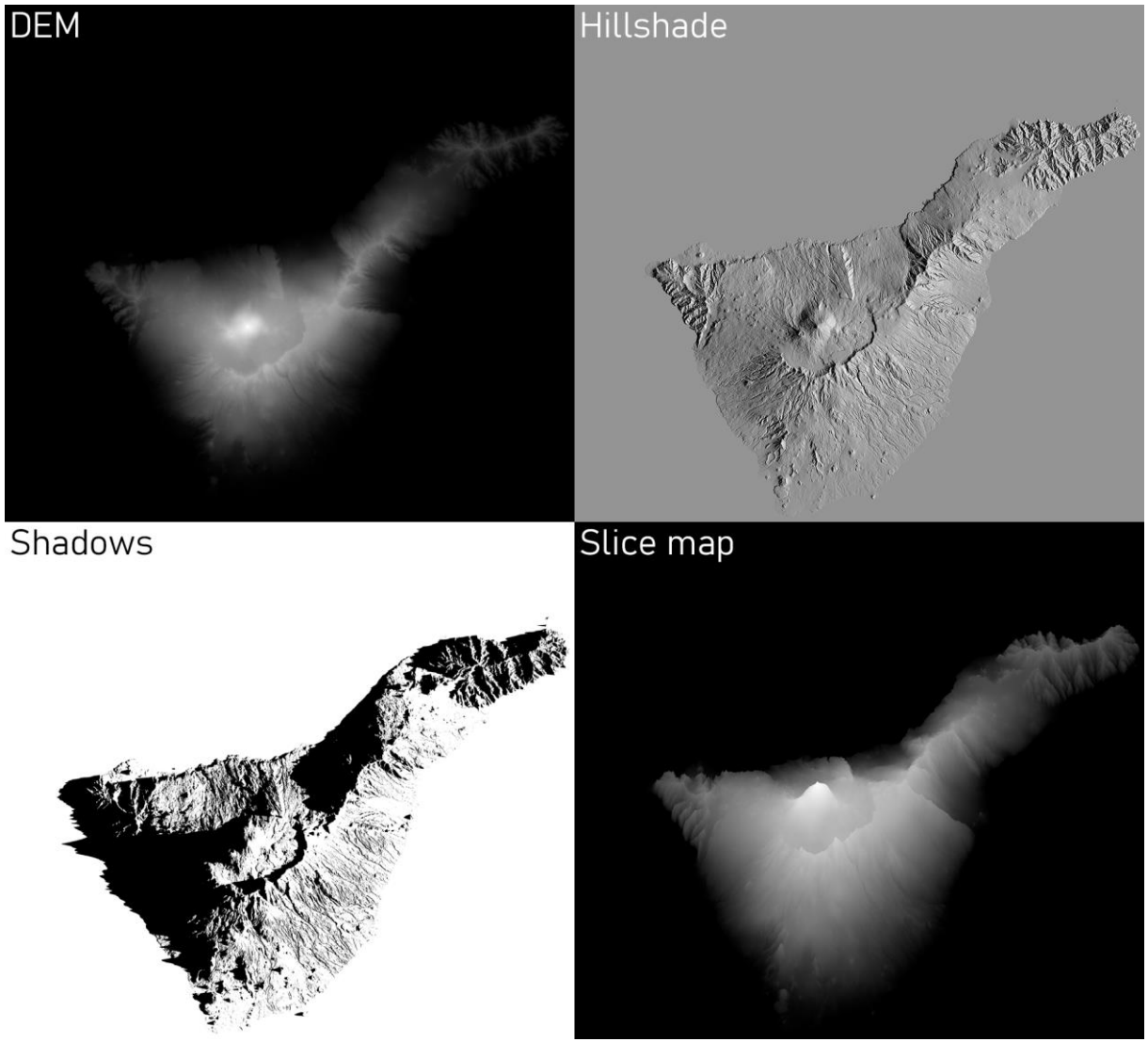


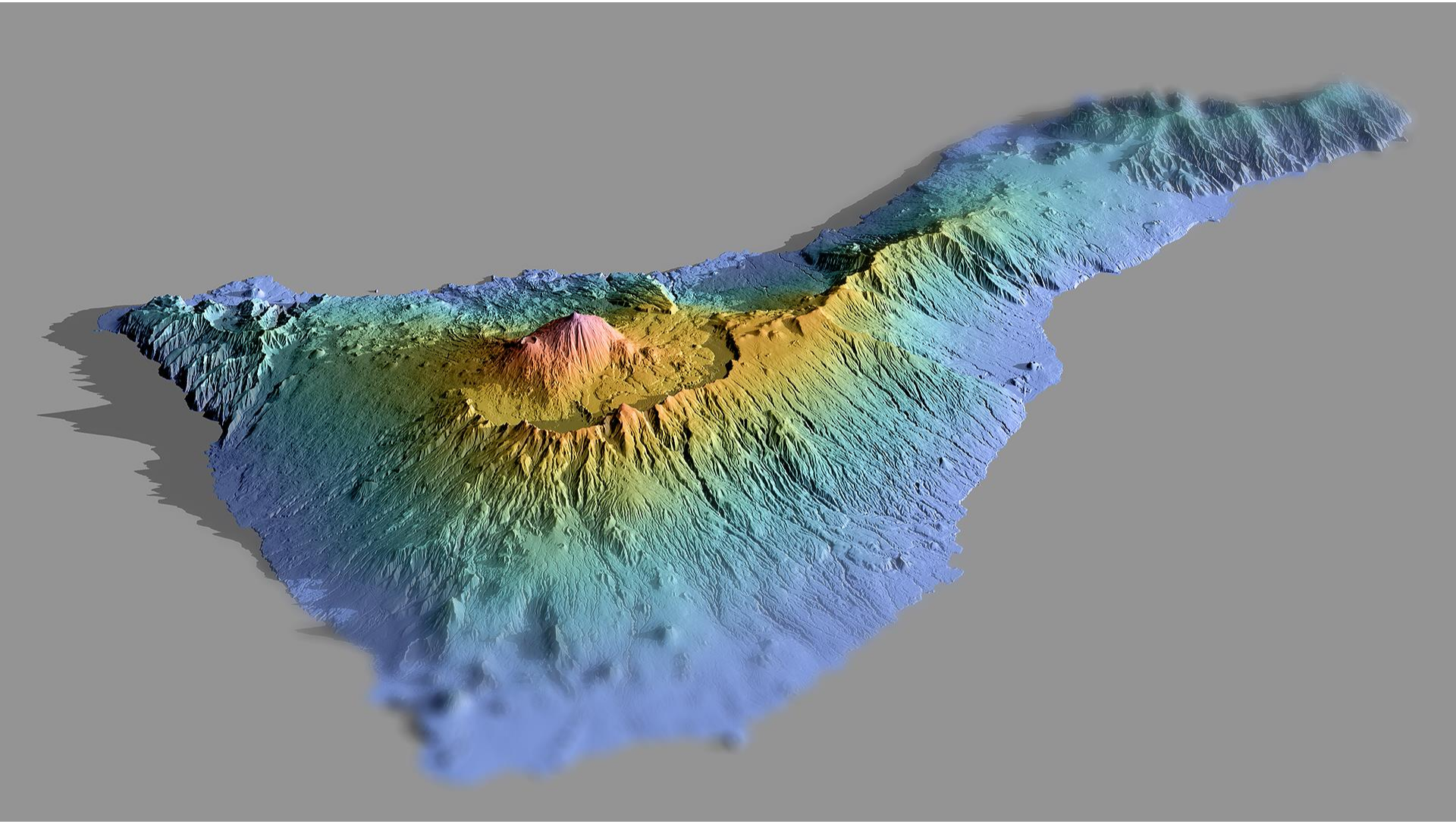




4. MAPAS PSEUDO 3D

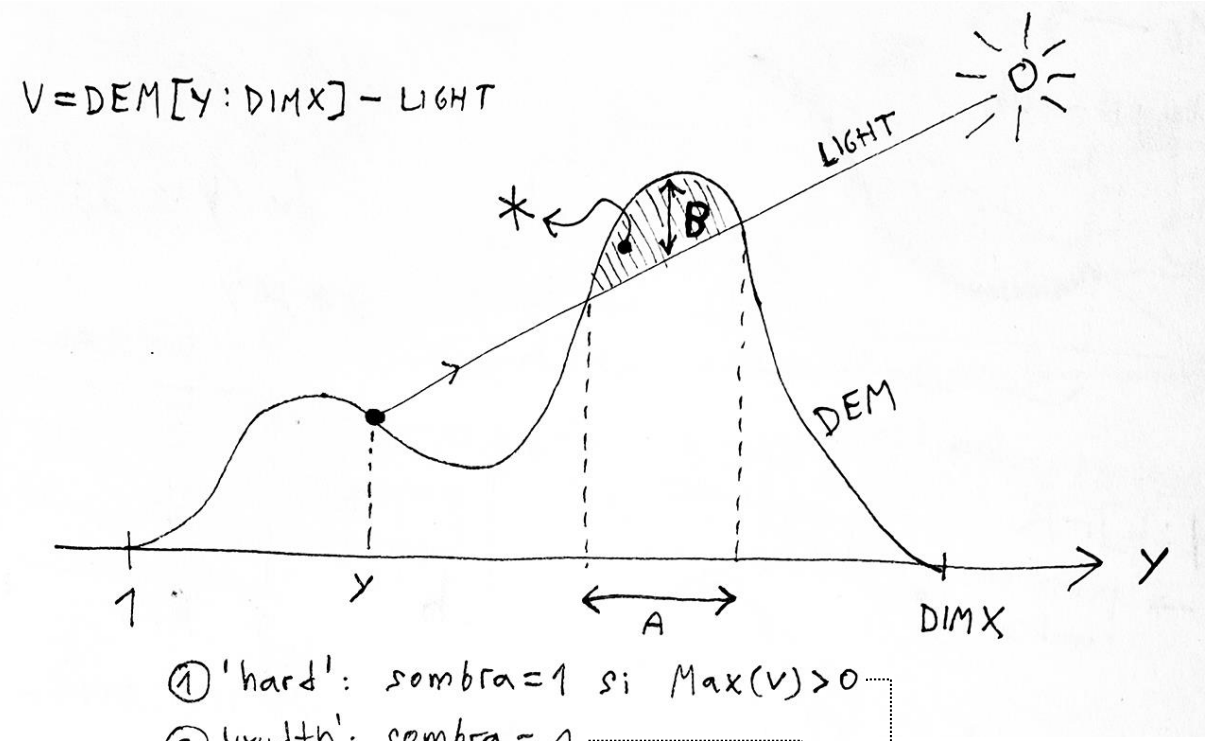
`aerialpersp()` **`slicemap()`** **`hillshademap()`** **`shadowmap()`**





4. MAPAS PSEUDO 3D

aerialpersp() slicemap() hillshademap() **shadowmap()**



① 'hard': sombra=1 si Max(v)>0

② 'width': sombra = A

③ 'max': sombra = B > 0

④ 'mixed': sombra = Área *

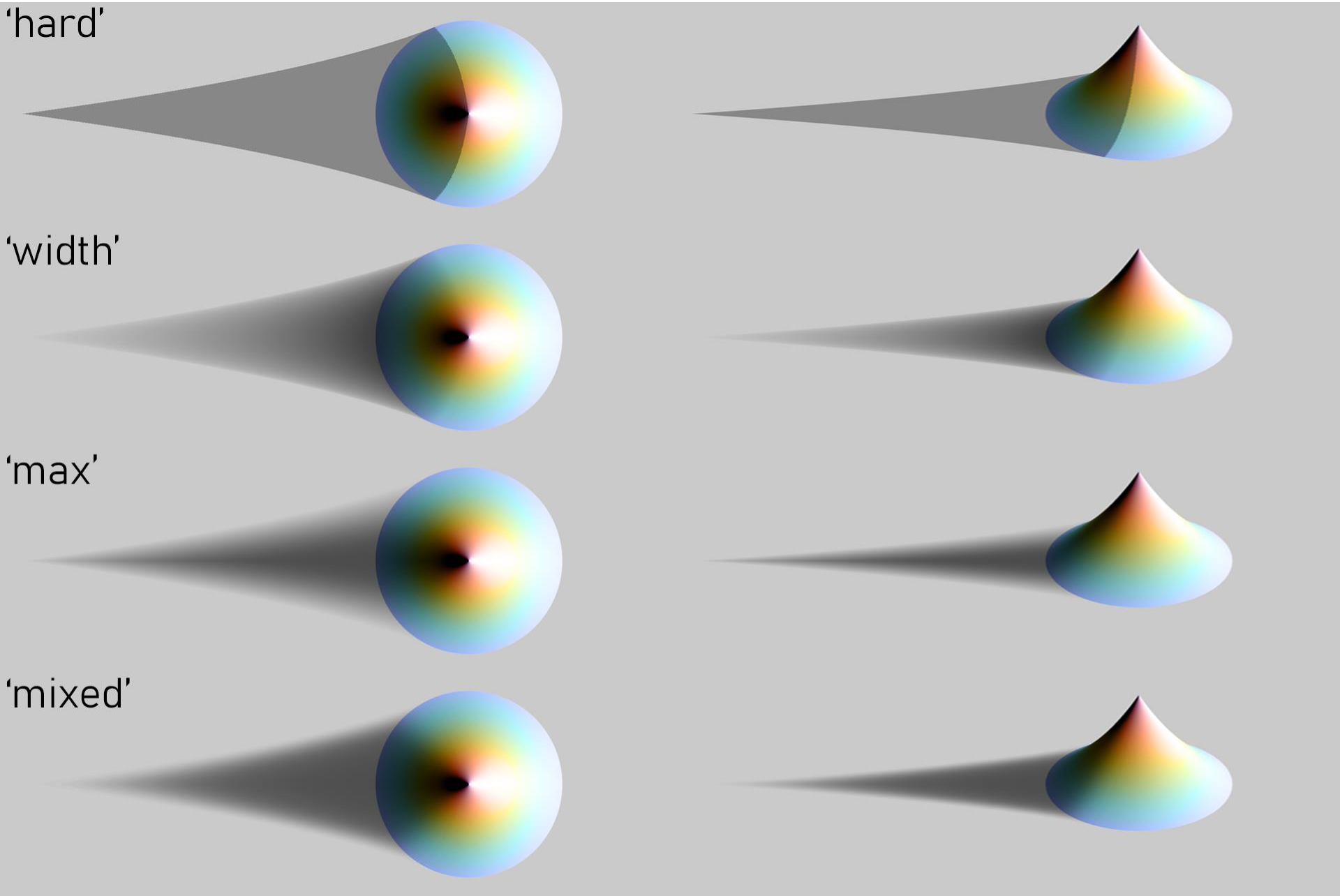
```
# 1. style='hard': shadow=1 if some elevation protrudes above light path
if (max(DELTA)>0) shadows[x,y]=1

# 2. style='width': thickness of protrusion
shadows[x,y]=length(DELTA[DELTA>0])

# 3. style='max': highest protrusion
shadows[x,y]=max(0,DELTA[DELTA>0])

# 4. style='mixed': width + height of protrusion
shadows[x,y]=sum(DELTA[DELTA>0])
```

Estilos de proyección de sombras basados en cuantificar el nivel de obstrucción



Programando mapas con R

1. INTRODUCCIÓN

arrayresample()
arrayblur()
solid()
contour()

2. FUNCIONES AUXILIARES

linemap()
circlemap()
joymap()
horizonmap()

3. MAPAS MINIMALISTAS

4. MAPAS PSEUDO 3D

aerialpersp()
slicemap()
hillshademap()
shadowmap()

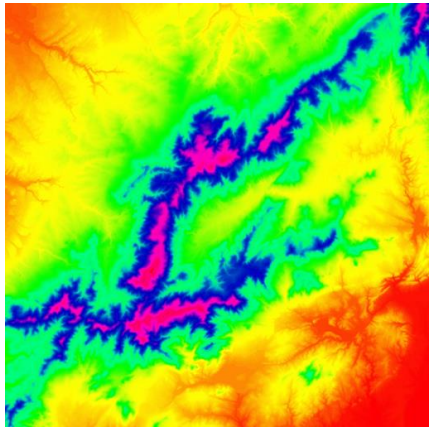
5. MAPAS CREATIVOS

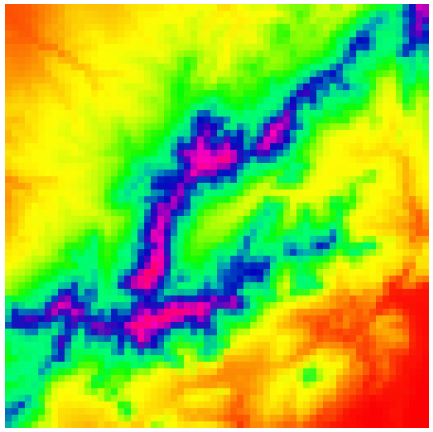
legomap()
foldmap()
karesansuimap()

6. VIDEO

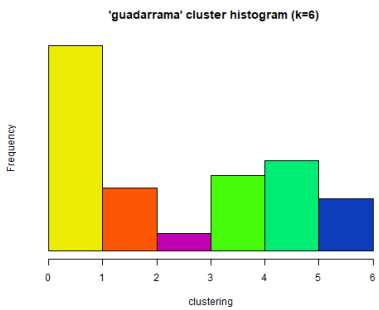
5. MAPAS CREATIVOS

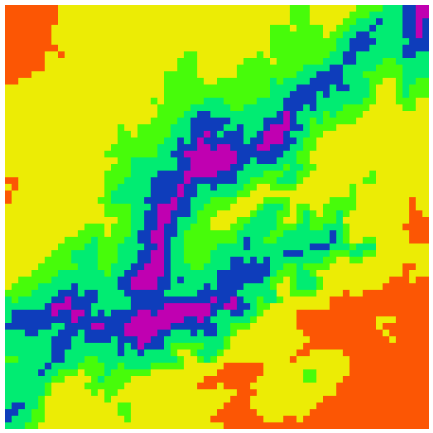
`legomap()` `foldmap()` `karesansuimap()`

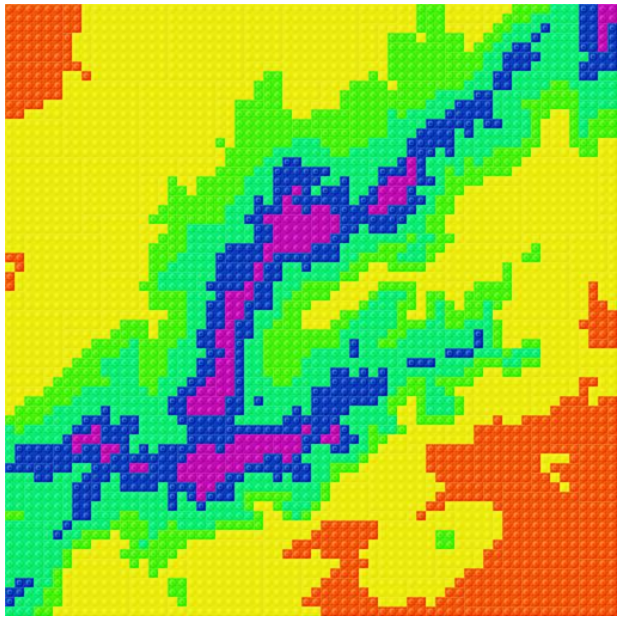




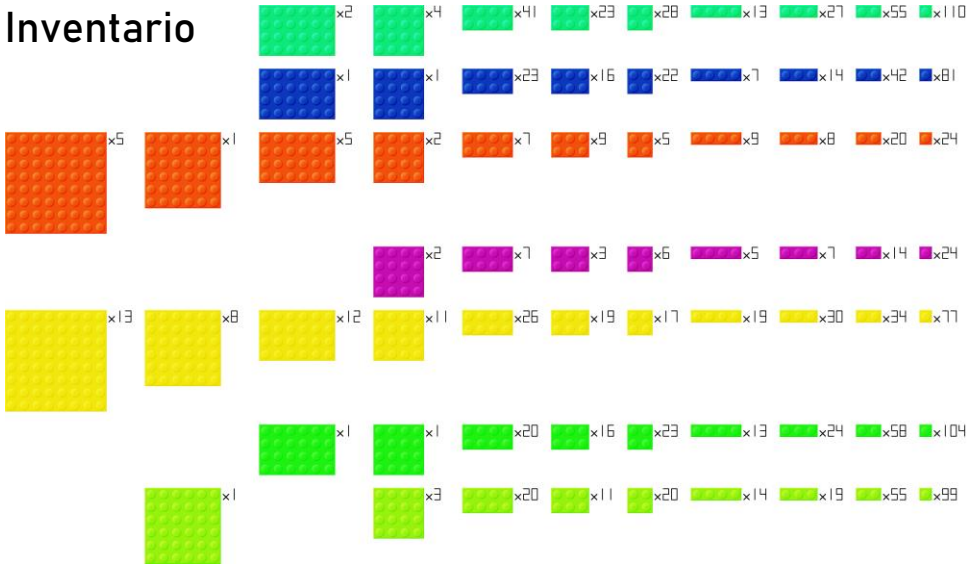
k-means clustering







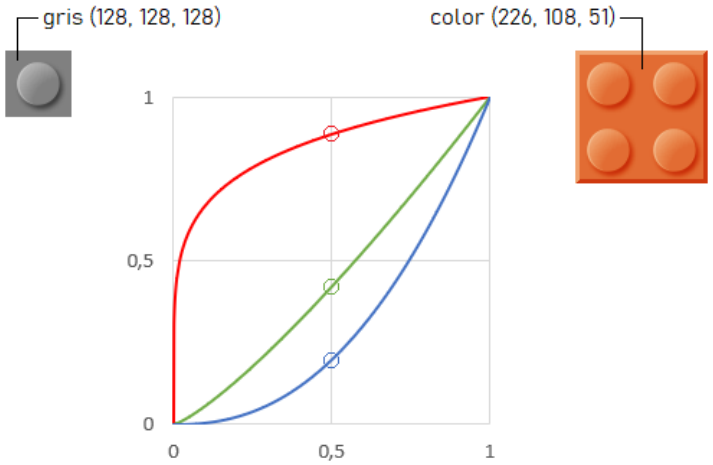
Inventario

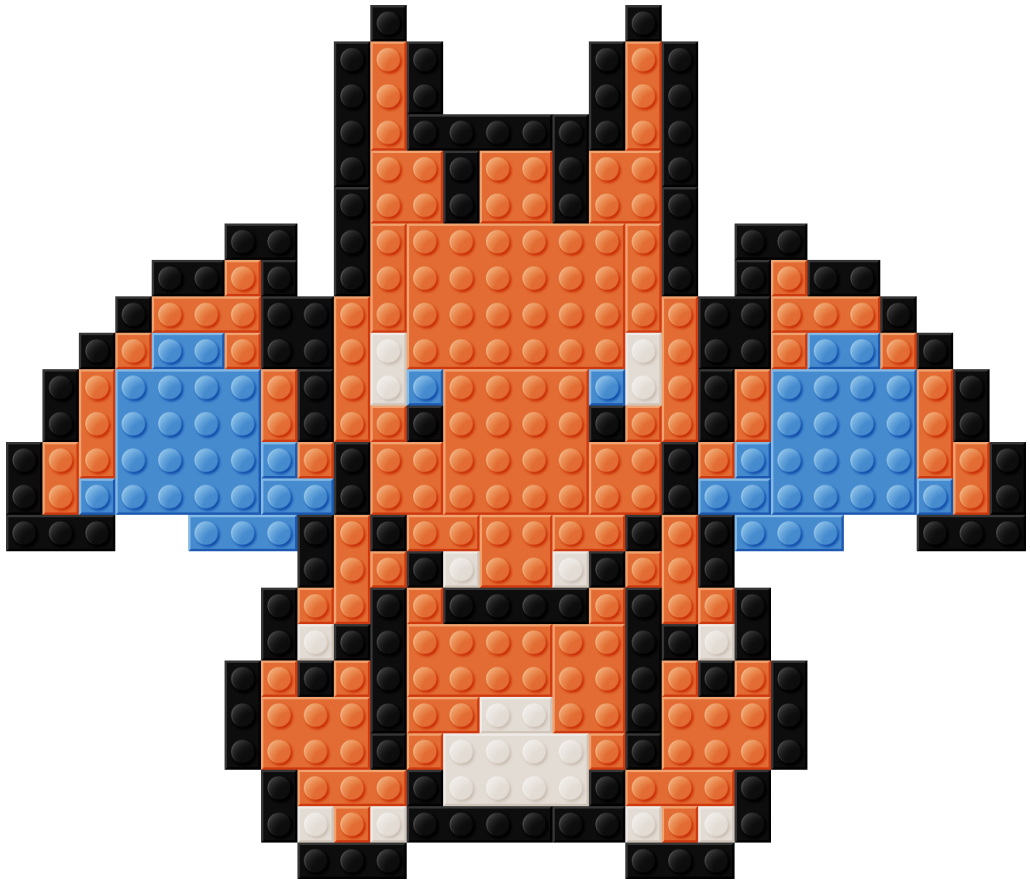


5. MAPAS CREATIVOS

`legomap()` `foldmap()` `karesansuimap()`

Coloreado con curvas gamma

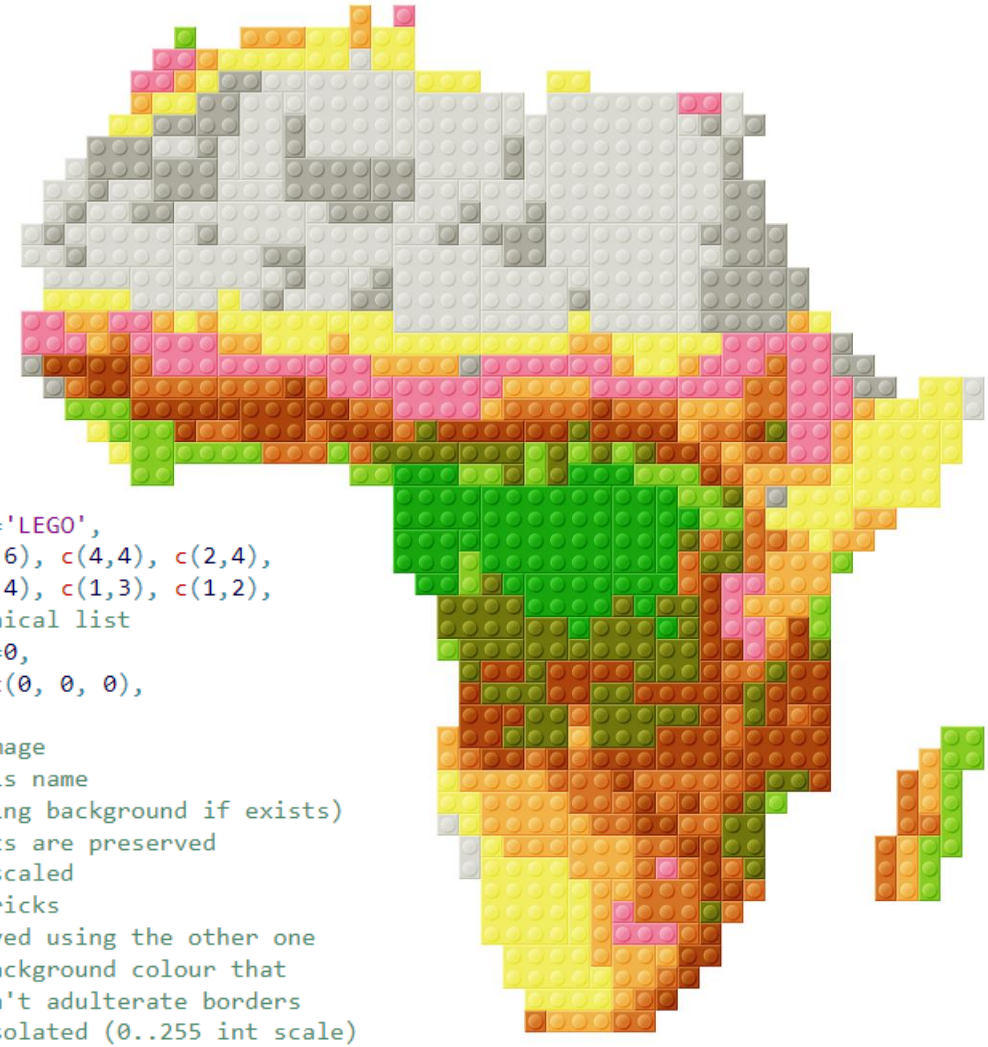




5. MAPAS CREATIVOS

legomap() foldmap() karesansuimap()

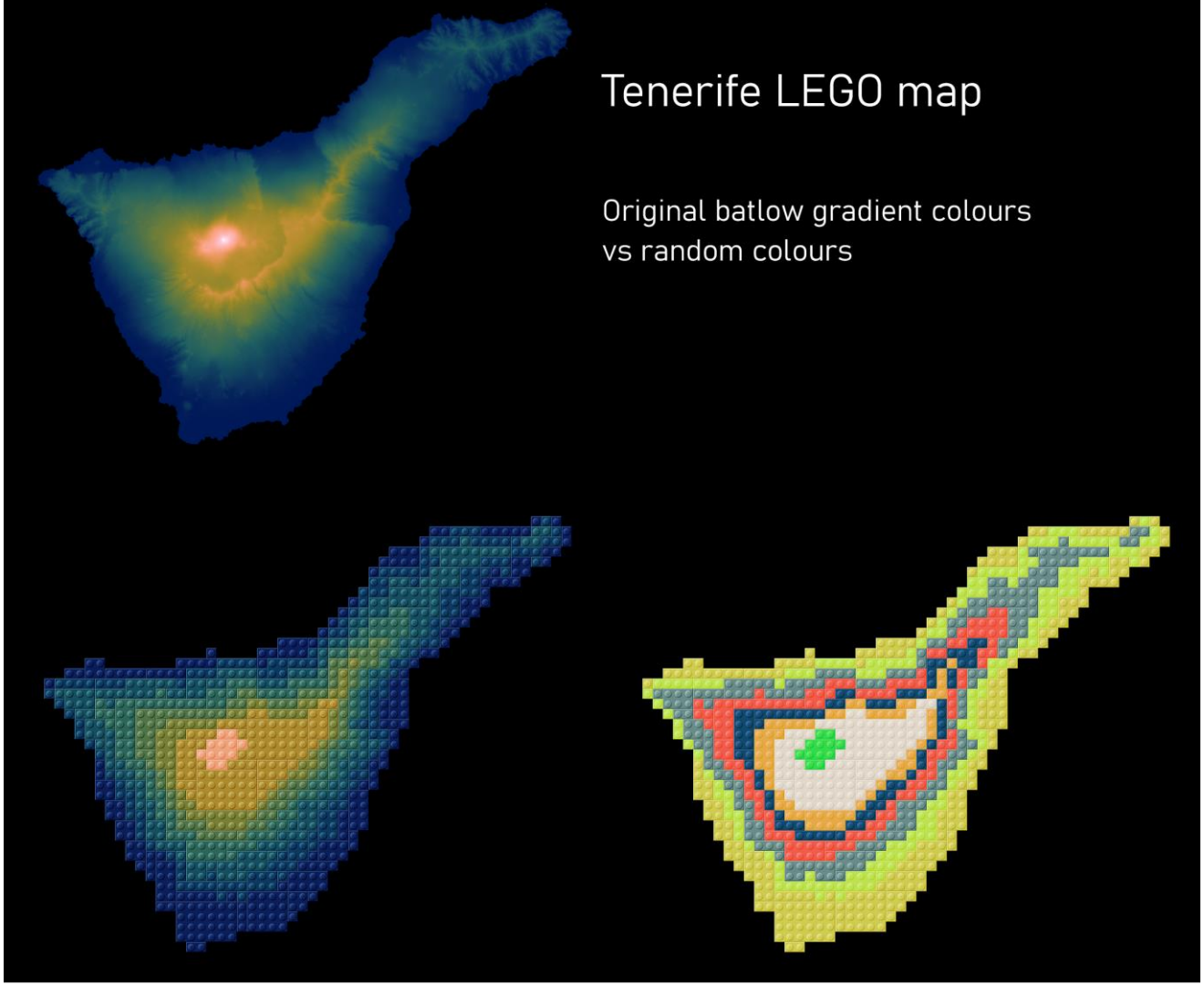
Los píxeles de fondo se pueden ignorar y no participarán en el remuestreo ni en el *clustering* de color

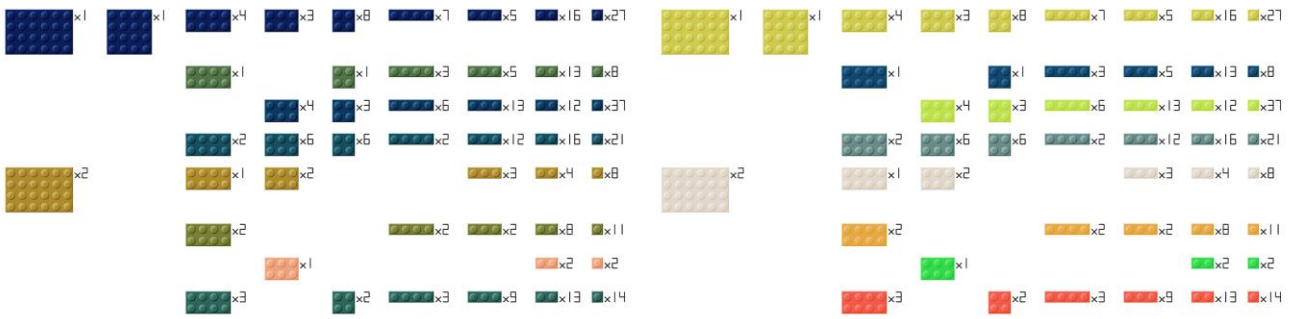


```
legomap=function(img, name, k=8, method='mean', type='LEGO',
                 LEGOBRICKS=list(c(8,8), c(6,6), c(4,6), c(4,4), c(2,4),
                                 c(2,3), c(2,2), c(1,4), c(1,3), c(1,2),
                                 c(1,1)),  # hierarchical list
                 resize=TRUE, LEGOSIZEX=0, LEGOSIZEY=0,
                 background=FALSE, backgroundcolour=c(0, 0, 0),
                 randomcolours=FALSE) {
    # img: DIMY x DIMX x 3 array containing an RGB image
    # name: output PNG files will be created with this name
    # k: number of colours in the clustering (including background if exists)
    #    if k=0 no clustering is applied and gradients are preserved
    # resize: bool to indicate that image must be rescaled
    # LEGOSIZEX/LEGOSIZEY: output size in 1x1 LEGO bricks
    #    if any of them is 0, aspect ratio is preserved using the other one
    # background: bool to indicate that there is a background colour that
    #    must be ignored in the rescaling so it doesn't adulterate borders
    # backgroundcolour: the EXACT RGB colour to be isolated (0..255 int scale)
    # randomcolours: randomize clustering colours instead of using centroids
```

Tenerife LEGO map

Original batlow gradient colours
vs random colours



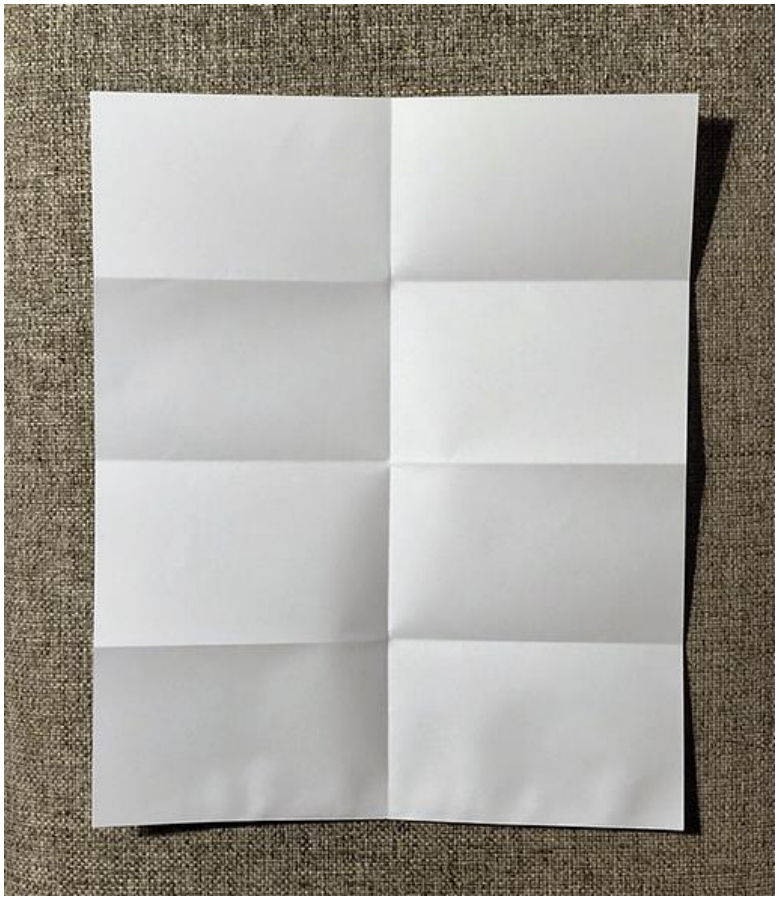


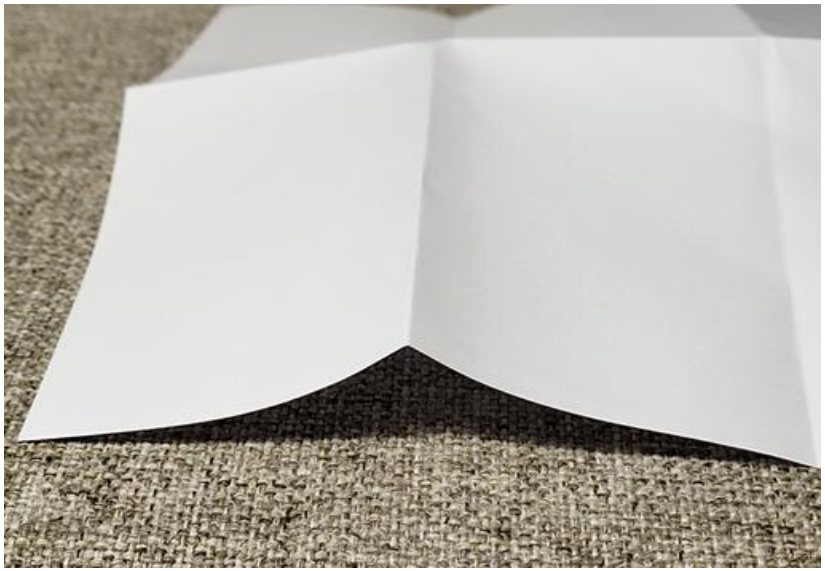

LEGO
TECHNIC
5+
42151
Pokémon
145
pcs/pzs
Building Toy
Jouet de construction
Juguete para Construir
IP LABEL

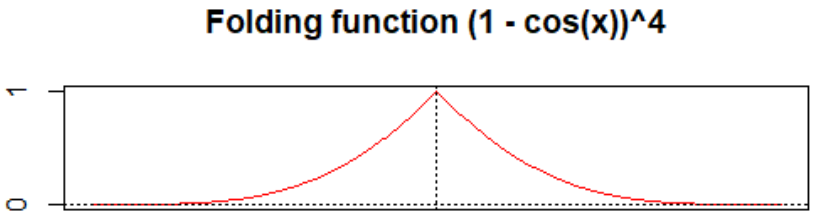
5. MAPAS CREATIVOS

`legomap()` **`foldmap()`** `karesansuimap()`

Modelamos la forma de las **elevaciones** de los pliegues de una hoja real

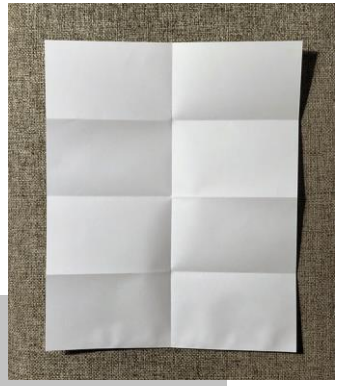




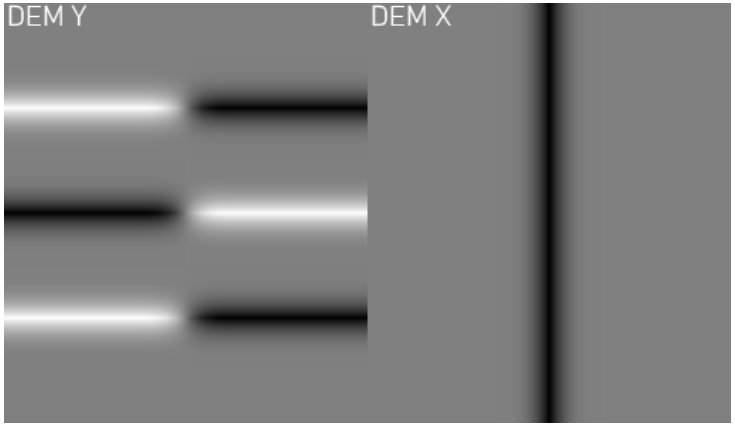


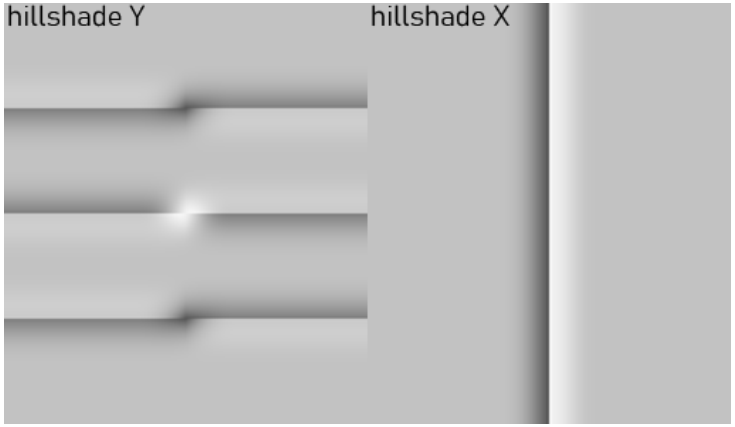
5. MAPAS CREATIVOS

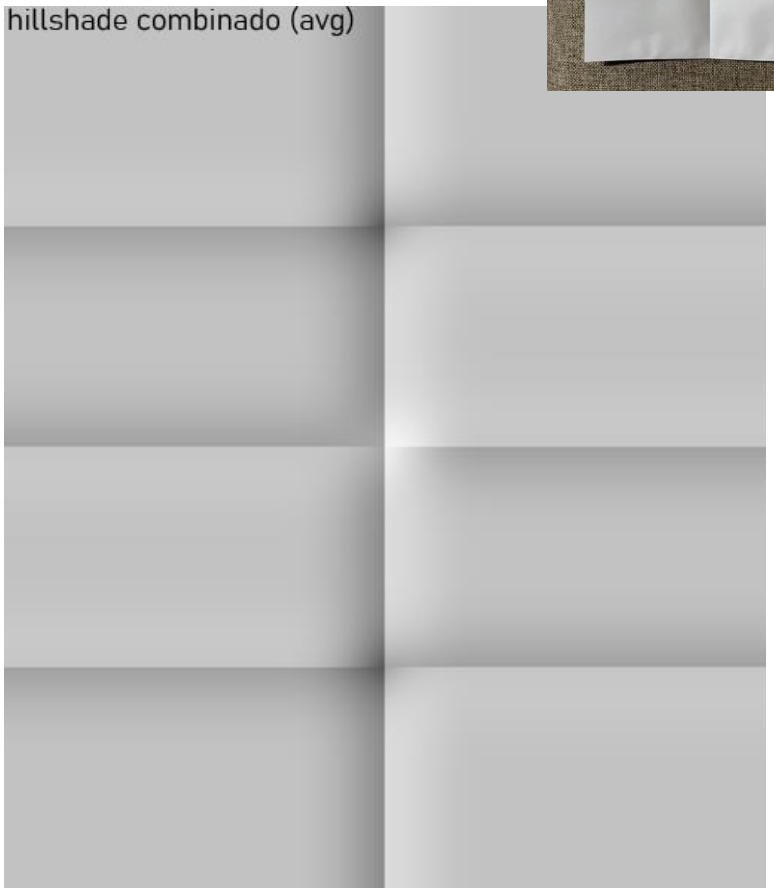
legomap() **foldmap()** karesansuimap()

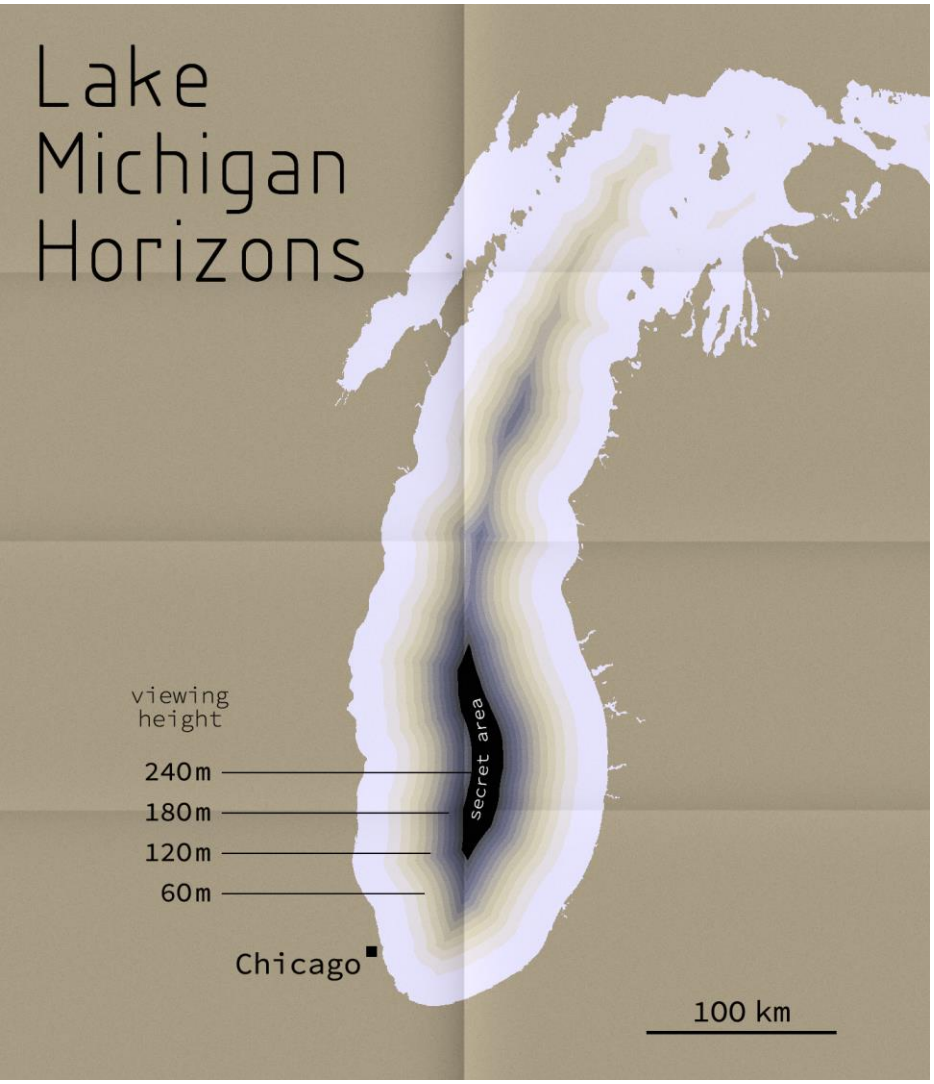


Obtenemos los ***hillshade*** de los DEM en cada eje y los promediamos





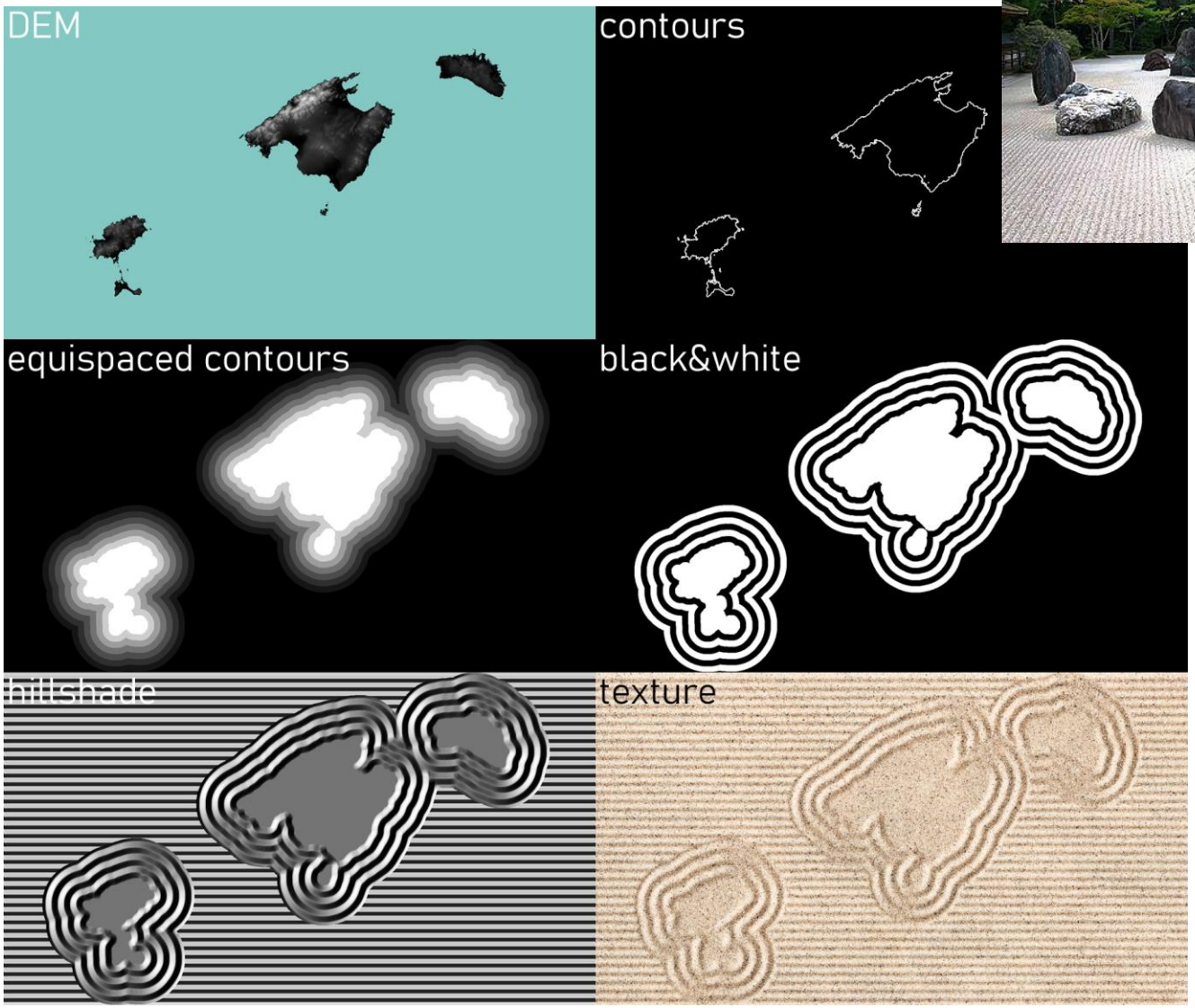



Lake
Michigan
Horizons
viewing
height
240m
180m
120m
60m
secret area
Chicago
100 km


Lake
Michigan
Horizons
viewing
height
240m
180m
120m
60m
secret area
Chicago
100 km

5. MAPAS CREATIVOS

`legomap()` `foldmap()` **`karesansuimap()`**



Programando mapas con R

1. INTRODUCCIÓN

2. FUNCIONES AUXILIARES

3. MAPAS MINIMALISTAS

4. MAPAS PSEUDO 3D

5. MAPAS CREATIVOS

6. VIDEO

gracias

overfitting.net