a RayShader Tutorial

Gaspard Palay

December 29, 2020

I. Présentation

rayshader est un logiciel libre pour la production de visualisations de données 2D et 3D en R. rayshader utilise les données d'élévation dans une matrice de base R et une combinaison de raytracing, de cartographie de texture sphérique, de superpositions et d'occlusion ambiante pour générer de magnifiques cartes topographiques 2D et 3D. En plus des cartes, rayshader permet également à l'utilisateur de traduire les objets ggplot2 en de magnifiques visualisations de données 3D.

Les modèles peuvent être tournés et examinés de manière interactive ou le mouvement de la caméra peut être scripté pour créer des animations. L'utilisateur peut également créer un effet de post-traitement de la profondeur de champ cinématographique pour diriger le regard de l'utilisateur vers les régions importantes de la figure. Les modèles 3D peuvent également être exportés dans un format imprimable en 3D avec une fonction d'exportation STL intégrée.

II. Installations

2.2. Installations élémentaires

Vous devez d'abord installer le package devtools qui vous permettra par la suite d'installer le package Rayshader depuis le GitHub de son développeur **Tyler Morgan**

```
install.packages("devtools")
```

Vous aurez ensuite besoin des packages Rayshader et Rayrender.

```
devtools::install_github("tylermorganwall/rayrender")
install.packages("rayshader")
install.packages("rayrender")
```

Vous devrez impérativement avoir Rtools sur Windows OS pour lire les plot en 3D Si vous êtes sous MacOs je vous conseille d'intaller **XQuartz** https://www.xquartz.org/ depuis votre navigateur.
Conseil : lorsque vous plottez votre image en 3D sur Rtools (rgl wondow) ou sur Xquartz, n'oubliez pas de fermer cette fenêtre avant de générer une nouvelle image, si vous ne faites pas cette manipulation, vous aurez des problèmes pour générer la novuelle parcelle.

Installez aussi le package Raster

```
install.packages("raster")
```

2.3. Installations secondaires

Ensuite, selon vos besoins, vous aurez besoin de différents packages annexes :
* rgdal : permet de lire des images dans R
* Rlang
* Png : permet de lire des images au format png dans R

* Gifski : si vous souhaitez tranformer votre plot en gif animé
* Av

III. Fonctions

Rayshader a sept fonctions liées à la cartographie :

ray_shade utilise les directions de lumière spécifiées par l'utilisateur pour calculer une carte d'ombre globale pour une matrice d'élévation. Par défaut, il met également à l'échelle l'intensité lumineuse en chaque point par le produit de la direction moyenne du rayon et de la normale à la surface (également implémentée dans la fonction **lamb_shade**, celle-ci peut être désactivée en définissant lambert=FALSE.

sphere_shade fait correspondre une texture RVB à une ombre de colline par un mappage sphérique. Une texture peut être générée avec la fonction create_texture, ou chargée à partir d'une image. sphere_shade comprend également 7 palettes intégrées : “imhof1”, “imhof2”, “imhof3”, “imhof4”, “desert”, “bw”, “unicorn”.

Le programme **create_texture** permet de créer des cartes de texture en cinq couleurs : une surbrillance, une ombre, une lumière de remplissage à gauche, une lumière de remplissage à droite et une couleur centrale pour les zones planes. L'utilisateur peut également spécifier les couleurs des coins, mais create_texture interpole celles qui ne sont pas données.

ambient_shade crée une couche d'ombre d'occlusion ambiante, assombrissant les zones qui ont moins de lumière diffusée par l'atmosphère. Les vallées sont donc plus sombres que les zones plates et les crêtes.

lamb_shade utilise une direction de lumière spécifiée par un utilisateur unique pour calculer une carte d'ombre locale basée sur le produit des points entre la normale à la surface et la direction de la lumière pour une matrice d'élévation.

add_shadow prend deux des cartes d'ombres ci-dessus et les combine, en mettant à l'échelle la seconde (ou, si la seconde est un tableau RGB, la matrice) comme spécifié par l'utilisateur.

add_overlay prend un tableau RVB/RGBA à 3 ou 4 couches et le superpose à la carte actuelle. Si la carte comporte une transparence, celle-ci est prise en compte lors de la superposition de l'image. Sinon, l'utilisateur peut spécifier une seule couleur qui sera marquée comme complètement transparente, ou définir la superposition complète comme partiellement transparente.

Rayshader dispose également de trois fonctions pour détecter et ajouter de l'eau aux cartes :
detect_water utilise un algorithme de remplissage pour détecter les masses d'eau d'une zone minimale spécifiée par l'utilisateur.

add_water utilise la sortie de detect_water pour ajouter une couleur d'eau à la carte. L'utilisateur peut entrer sa propre couleur, ou passer le nom d'une des palettes prédéfinies de sphere_shade pour obtenir une teinte correspondante.

render_water ajoute une couche d'eau transparente en 3D aux cartes 3D, après que le dispositif rgl ait déjà été créé. Cela peut soit ajouter à une carte qui n'a pas encore de couche d'eau, soit remplacer une couche d'eau existante sur la carte.

Deux fonctions sont également incluses pour ajouter des effets et des informations supplémentaires à vos visualisations 3D : **render_highquality** rend dans la scène avec un pathtracer intégré, alimenté par le paquet rayrender. Utilisez-le pour des cartes de haute qualité avec un transport de lumière réaliste.

render_depth génère un effet de profondeur de champ pour la carte 3D. L'utilisateur peut spécifier la distance focale, la longueur focale et le diaphragme de la caméra, ainsi que la forme de l'ouverture et l'intensité du bokeh. Cela permet soit de tracer l'image sur l'appareil local, soit de l'enregistrer dans un fichier si on lui donne un nom de fichier.

render_label ajoute une étiquette de texte aux coordonnées x et y de la carte à une altitude z spécifiée (en unités de la matrice). L'altitude peut être spécifiée soit par rapport à l'altitude à ce point (par défaut),

soit de manière absolue. Et quatre fonctions pour afficher et enregistrer vos visualisations :

plot_map Trace la carte actuelle. Accepte soit une matrice, soit un tableau.

write_png Ecrit la carte actuelle sur le disque avec un nom de fichier spécifié par l'utilisateur.

plot_3d Crée une carte en 3D, en lui donnant une texture et une matrice d'élévation. Vous pouvez personnaliser l'apparence de la carte, ainsi qu'ajouter un niveau d'eau défini par l'utilisateur.
render_snapshot Enregistre une image de la vue 3D actuelle sur le disque (si un nom de fichier lui a été attribué), ou trace la vue 3D sur le périphérique actuel (utile pour inclure des images dans les fichiers R Markdown).

render_movie Crée et enregistre un fichier mp4 de la caméra tournant autour de la scène 3D en utilisant soit une orbite intégrée, soit une orbite fournie par l'utilisateur.

Enfin, rayshader a une fonction unique pour générer des tracés 3D en utilisant des objets ggplot2 :
plot_gg Prend un objet ggplot2 (ou une liste de deux objets ggplot2) et utilise l'esthétique de remplissage ou de couleur pour transformer le tracé en une surface 3D. Vous pouvez passer tous les arguments utilisés pour spécifier la caméra et les couleurs d'arrière-plan/ombre dans plot_3d(), et manipuler le tracé 3D affiché en utilisant render_camera() et render_depth().

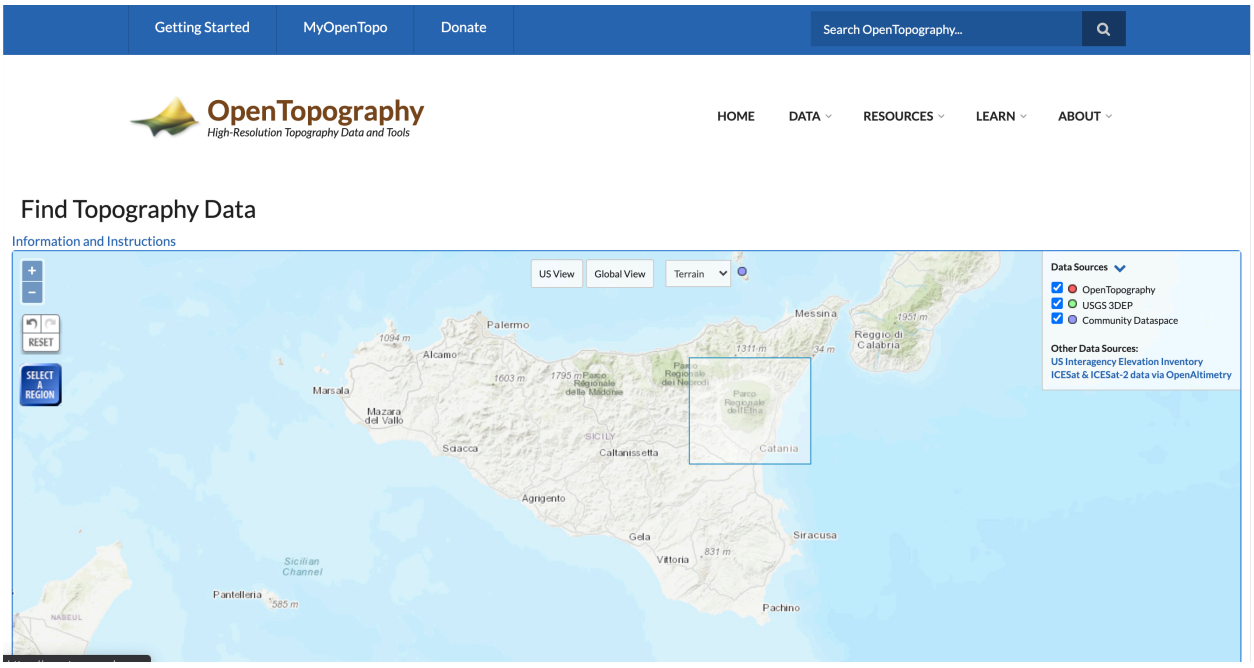
Toutes ces fonctions sont conçues pour être utilisées avec le tube magrittr %>%

IV. RayShader appliqué

4.1.Téléchargement d'un fichier de manipulation

Pous vous montrer l'étendue des pouvoirs de RayShader, j'utilise un cas pratique qui a pour objectif d'élever en 3D une image.tif topographique du volcan l'Etna. Pour se faire :
1. Rendez vous sur le site https://opentopography.org/

2. Selectionnez la région que vous souhaitez élever en 3D



3. Téléchargez l'image en :
   selectionnant l'output : geotiff
   Choisissant de préférence un fichier Global Multi-Resolution Topography (GMRT) Data Synthesis
   Selectionnant la résolution maximale
   générez, si vous le souhaitez "additional color-relief and colored hillshades"
4. Soumettez votre choix et attendez le lancement du téléchargement du fichier. Vous le recevrez par courriel.

Vous obtiendrez une image topgraphique composée d'une multitude de données à traiter.

Une fois l'image stockéé dans votre ordinateur, vous allez pouvoir commencer à la manipuler sur R.

4.2. Paramétrage du fichier

Pour commencer, n'oubliez pas d'appeler sur votre script tous les packages que vous avez préalablement téléchargés.

Vous devrez dans un premier temps créer une variable d'élévation appelant votre image .tiff

```
elevation <- "/Users/gaspardpalay/Documents/PSB - Paris School of Business/Programmation R/Projet Rayre
```

Vous devrez ensuite "ratseriser" votre image. Cela consiste à créer un objet raster contenant les informations nécessaires pour la transformation en matrix (matrice d'élévation)

```
elevation.raster <- raster::raster(elevation)
```

Vous pourrez alors transformer votre image en matrice d'élévation. Cette étape transforme l'image d'élévation en matrix. Vous pouvez observer l'inversion au niveau nrow() et ncol(). La fonction extract extrait les valeurs de l'objet, quant à la fonction extent, elle, définit la taille de la matrice.

```
elevation.matrix <- matrix(extract(elevation.raster, extent(elevation.raster),
                                   buffer = 100),
nrow = ncol(elevation.raster), ncol = nrow(elevation.raster))
```

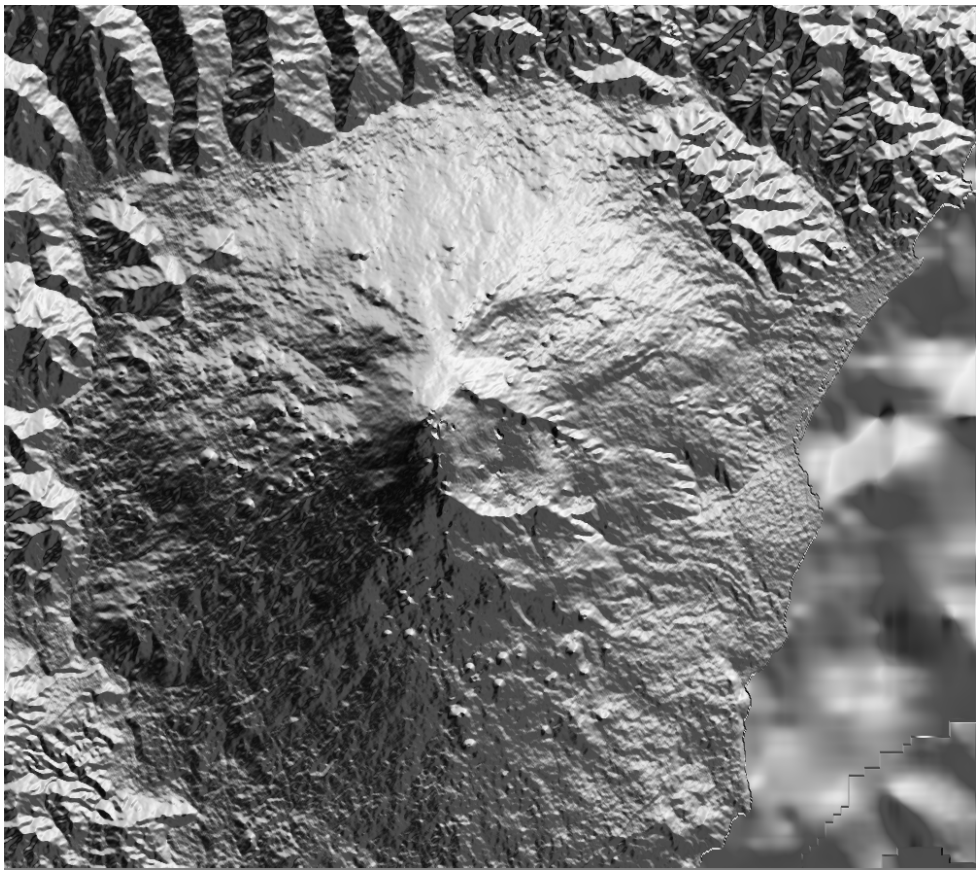
Affichons la dimension de notre variable elevation.matrix.

```
dim(elevation.matrix)
```

```
## [1] 1001  890
```

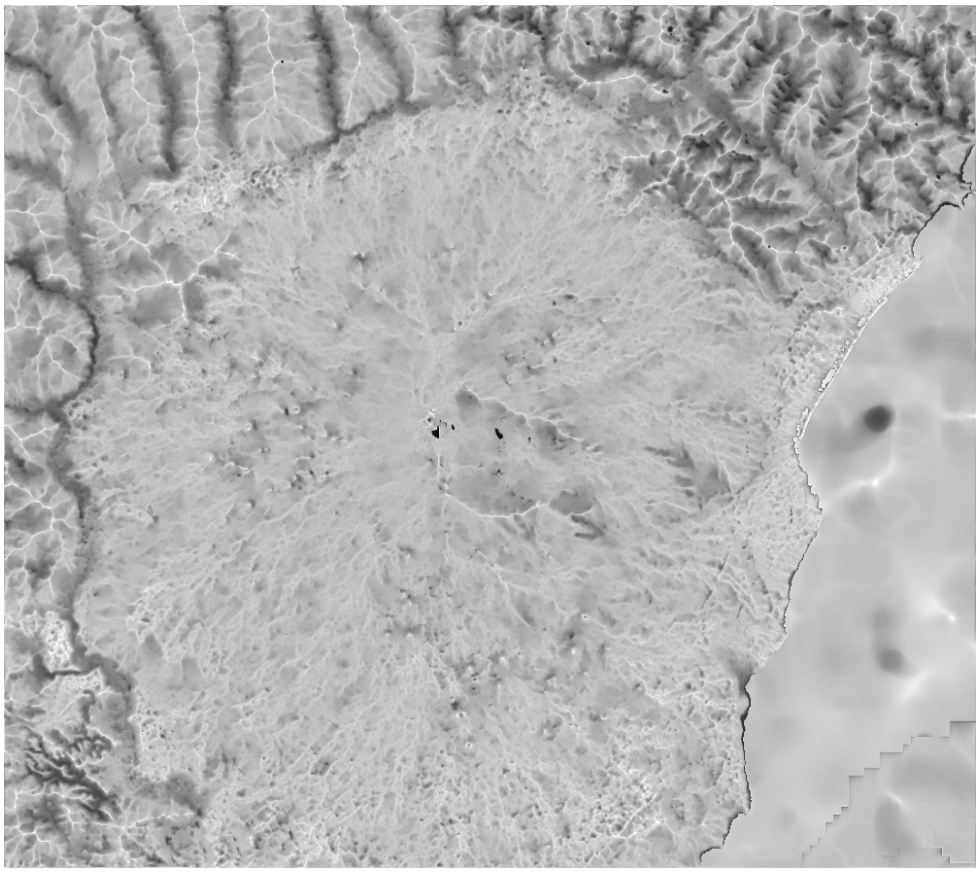
Vous allez ensuite pouvoir ajouter une texture à votre matrix. La fonction sphere_shade le permet. L'argument sunangle simule la lumiere sur la matrix, ici, elle est à 45°.
L'argument texture ajoute une texture à la matrix. il en existe 6 pré programmées dans le package RayShader. Sinon, vous pouvez aussi faire votre propre texture avec la fonction create_texture().

```
elevation.matrix %>%
  sphere_shade(sunangle = 45, texture = "bw") %>%
  plot_map()
```



Ajoutez ensuite “l’ambiant occulsion mapping”. Celui ci va assombrir les zones non accessibles à la lumière.

```
elevation.matrix %>%
  ambient_shade() %>%
  plot_map()
```



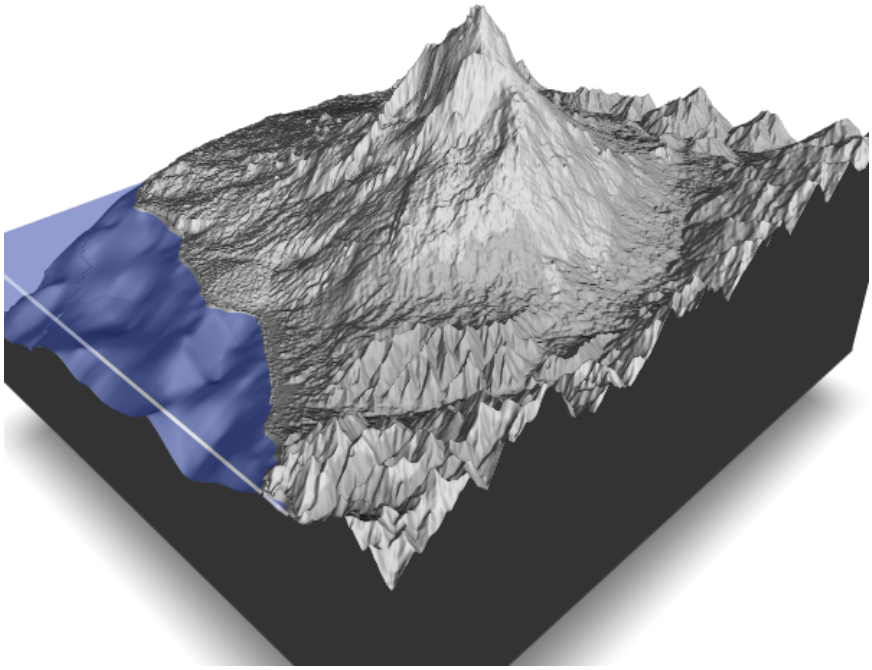
Calculez les ombres par “retraycing”. Le paramètre anglebreaks va positionner la source de lumière en degré à partir de l’horizon.

4.3. Elevation finale en 3D

Pour élever la fonction en 3D nous répétons tous les étapes précédentes et finissons par utiliser la fonction **plot_3D**. Elle possède plusieurs arguments :
zscale permet de régler la hauteur de l’élévation **theta** ajuste la rotation de la camera
phi l’angle à partir de l’azimut **water** ajoute ou enlève l’eau, définit sa profondeur (waterdepth) et sa couleur (watercolor) **waterlinecolor** ajoute un contour plus ou moins épais à la ligne d’horizon de l’eau

```
elevation.matrix %>%
  sphere_shade(sunangle = 45, texture = "bw") %>%
  add_water(detect_water(elevation.matrix), color = "turquoise3") %>%
  add_shadow(ambient_shade(elevation.matrix), 0.5) %>%
  add_shadow(ray_shade(heightmap = elevation.matrix, anglebreaks = seq(0,45, 5),
                       sunangle = 45, zscale = 0.05)) %>%
  plot_3d(heightmap = elevation.matrix,
          zscale = 10,
          fov = 70,
          lineantialias = TRUE,
          theta = 135,
          phi = 30,
          zoom = 0.6,
          water = TRUE, waterdepth = 0, wateralpha = 0.5, watercolor = "#233aa1",
          waterlinecolor = "white", waterlinealpha = 0.5)
```

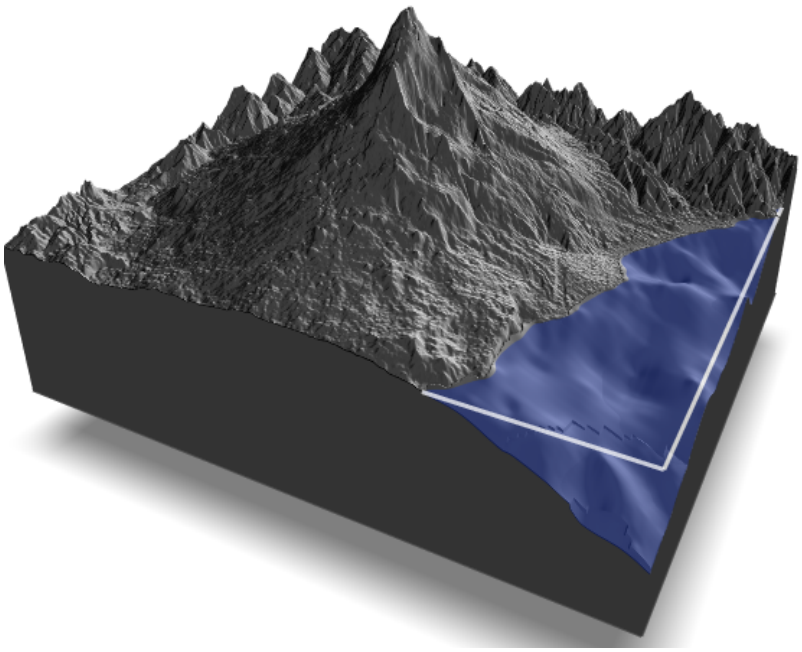
```
render_snapshot()
```



4.3.1. Modifier l'orientation

Vous pouvez modifier l'orientation de l'image 3D en jouant sur les arguments **theta**, **phi**, et **zoom**.

```
elevation.matrix %>%
  sphere_shade(sunangle = 45, texture = "bw") %>%
  add_water(detect_water(elevation.matrix), color = "turquoise3") %>%
  add_shadow(ambient_shade(elevation.matrix), 0.5) %>%
  add_shadow(ray_shade(heightmap = elevation.matrix, anglebreaks = seq(0,45, 5),
                       sunangle = 45, zscale = 0.05)) %>%
  plot_3d(heightmap = elevation.matrix,
          zscale = 10,
          fov = 70,
          lineantialias = TRUE,
          theta = 28,
          phi = 25,
          zoom = 0.68,
          water = TRUE, wateralpha = 0.5, watercolor = "#233aa1",
          waterlinecolor = "white", waterlinealpha = 0.5,
          solid = TRUE,
          shadow = TRUE)
render_snapshot()
```



4.3.2. Ajouter une texture supplémentaire externe au package

Vous pouvez ensuite ajouter une texture supplémentaire récupérée ailleurs. Pour ce cas pratique, j'utilise une image satelitte de mon fichier de base prise sur Google Eearth. Pour se faire, je l'enregistre sur ma machine et lui définit une variable. Vous aurez besoin du package "png" pour appeler l'image au format .png.

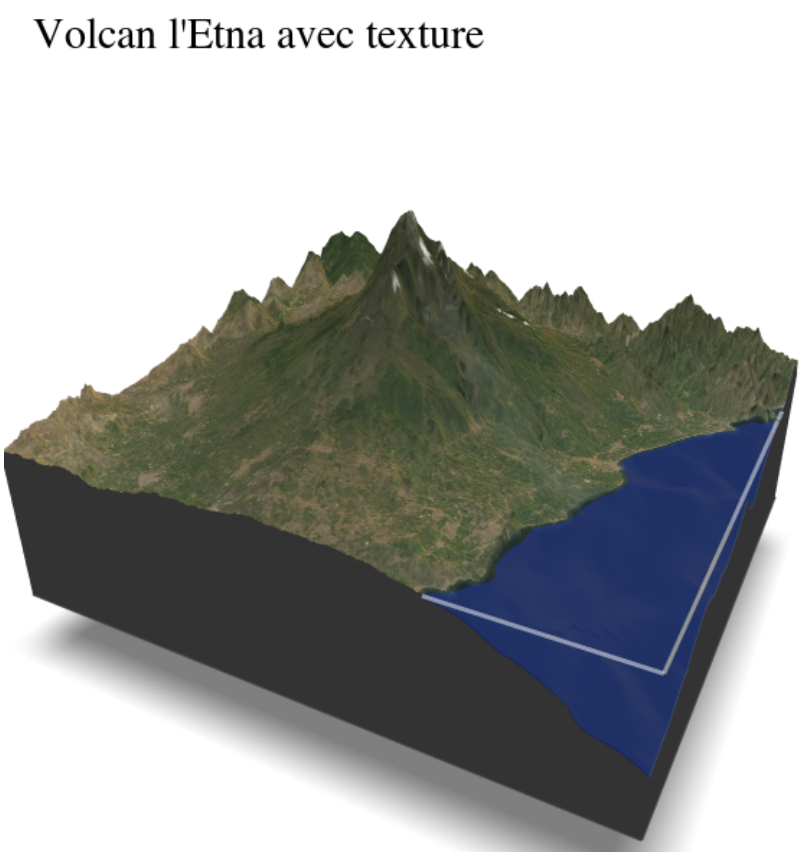
```
elevation.texture.map <- readPNG("/Users/gaspardpalay/Documents/PSB - Paris School of Business/Programma
```

Vous pourrez ensuite ajouter cette texture en utilisant la fonction intégrée de Rayshader **add_overlay**

```
rgl::rgl.clear()
```

```
elevation.matrix %>%
  sphere_shade(sunangle = 45) %>%
  add_shadow(ambient_shade(elevation.matrix), 0.5) %>%
  add_shadow(ray_shade(heightmap = elevation.matrix, anglebreaks = seq(0,45, 5),
                       sunangle = 45, zscale = 0.05)) %>%
  add_water(detect_water(elevation.matrix), color = "turquoise3") %>%
  add_overlay(elevation.texture.map, alphacolor = NULL, alphalayer = 0.9) %>%
  plot_3d(heightmap = elevation.matrix,
          zscale = 10,
          fov = 70,
          lineantialias = TRUE,
          theta = 28,
          phi = 25,
          zoom = 0.68,
```

```
          water = TRUE, wateralpha = 0.5, watercolor = "#233aa1",
          waterlinecolor = "white", waterlinealpha = 0.5,
          solid = TRUE,
          shadow = TRUE)
render_snapshot(clear = TRUE, title_text = "Volcan l'Etna avec texture",
                title_font = "Times", title_position = "northwest")
```



V. Autres utilisations de RayShader

5.1. Render Movie

La fonction **render_movie**, permet, selon certains paramètres, de configurer une vidéo animée de votre plot 3D en définissant son angle de rotation, zoom… etc. Vous pourrez alors, sur la base de cette video, créer un giff à intégrer sur un site web par exemple, à l'aide du package **gifski**.

5.2. RayShader avec GGplot

RayShader permet aussi de transformer un banal graphique Ggplot en 2D en une matrice graphique 3D et animée. Pour se faire, vous devrez procéder ainsi :

1. Récupérer un DataSet sur lequel vous souhaitez extraire un graphique
2. Générer un tracé cartographique en 2D avec ggplot2
3. Transformer le tracé en 2D en un tracé en 3D avec rayshader (notamment avec la fonction **plot_gg**)

VI. Sources et liens utiles

https://www.rayshader.com/
https://cran.r-project.org/bin/windows/Rtools/
https://medium.com/@nathanaelsheehan/an-easy-way-to-make-impressive-topological-maps-with-rayshader-66b257dc1e02
https://www.xquartz.org/
https://www.davidsolito.com/post/a-rayshader-base-tutortial-bonus-hawaii/
https://wcmbishop.github.io/rayshader-demo/
https://towardsdatascience.com/introducing-3d-ggplots-with-rayshader-r-c61e27c6f0e9
https://www.google.com/intl/fr/earth/
https://www.rayshader.com/reference/height_shade.html
https://www.rayshader.com/reference/add_overlay.html
https://www.tylermw.com/3d-maps-with-rayshader/
https://github.com/wcmbishop/rayshader-demo/blob/master/R/rayshader-gif.R
https://nbviewer.jupyter.org/github/patrickcgray/rayshader_experiments/blob/master/rayshader_nc_inundation.nb.html
https://rdrr.io/github/tylermorganwall/rayshader/man/render_movie.html
https://www.xquartz.org/
https://github.com/tylermorganwall/rayshader
https://github.com/tylermorganwall/rayrender