WIP - Seabed Analysis

Marco Giovanni Ferrari

2024-11-26

Introduction

The aim of this code is to define an algorithm to extract the Riley Ruggedness of the seabed under subsea cables. The ideal algorithm should perform the following tasks:

- Takes the **cohordinates** of a subseacable and define the neighbourhood around it. It should allow to work dynamically by defining a **radius** (in terms of kilometers) around the **cable**.
- Once the area is defined it is necessary to compute the **Riley Ruggedness** in **every cell** in the **area**. Note that for every cell in a **spatial grid**, the Riley ruggedness is defined as square root of the summation of the squared difference in the elevation between the cell itself and its eight neighboring cells. In mathematical terms, a cell in row r and column c and elevation e:

$$RileyRuggedness_{rc} = \sqrt{\left(\sum_{i=r-1}^{r+1} \sum_{j=c-1}^{c+1} (e_{ij} - e_{rc})^2 \right)}.$$

  When considering a spatial grids, cells and rows are **not indexed** in integer units, but have geographical coordinates, which must then be **ordered**.
- Finally, following Juhász and Steinwender (2018) one can compute **different statistics** on the ruggedness of the cells in the area around the cable. In their paper, they simply considered the **mean** ruggedness.

In performing this analysis, the following libraries will be necessary:

```
library(terra) # To import and visualize geographic data
library(dplyr) # For data manipulation
library(knitr) # For better visualization
library(ggplot2) # For data visualization and publication
library(ggthemes)
library(ggpubr)
library(purrr)
library(sf) # To manipulate geographic data and maps
```

How bathymetry data work

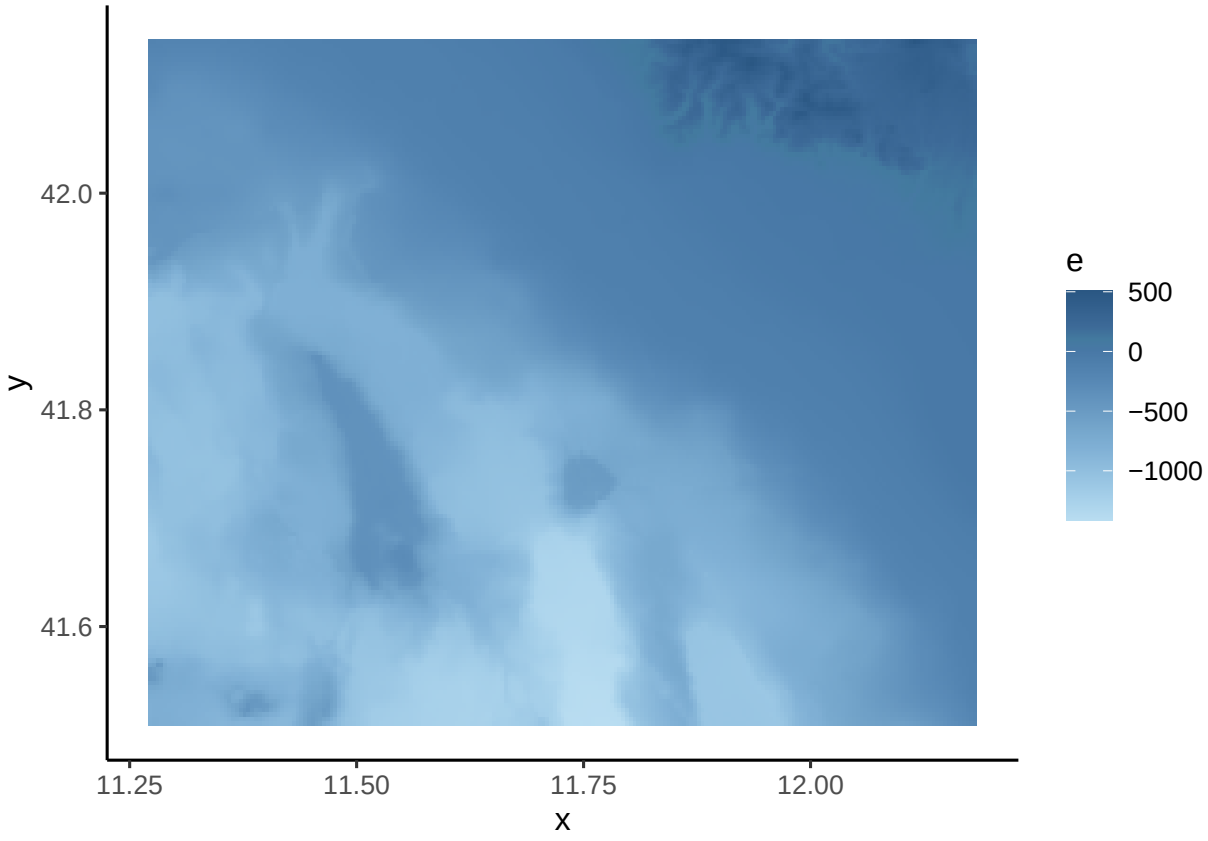
First, consider the following example of bathymetry data, which can be visualized in a $n \times 3$ dataset, with columns x (the longitude), y (the latitude) and e (the elevation).

```
geodata_example_raw <- rast("raw_data/ocean_bathymetry_data/example_bathymetry_data.tif")
geodata_example <- as.data.frame(geodata_example_raw, xy = TRUE)

geodata_example <- geodata_example %>% rename(e = example_bathymetry_data)

kable(head(geodata_example), format = "markdown")
```

x	y	e
11.27292	42.13958	-173
11.27708	42.13958	-176
11.28125	42.13958	-179
11.28542	42.13958	-181
11.28958	42.13958	-182
11.29375	42.13958	-182

Which visually appear as:

```
ggplot() + geom_raster(data = geodata_example, aes(x=x, y=y, fill=e))+
  scale_fill_continuous_tableau() + theme_classic2()
```

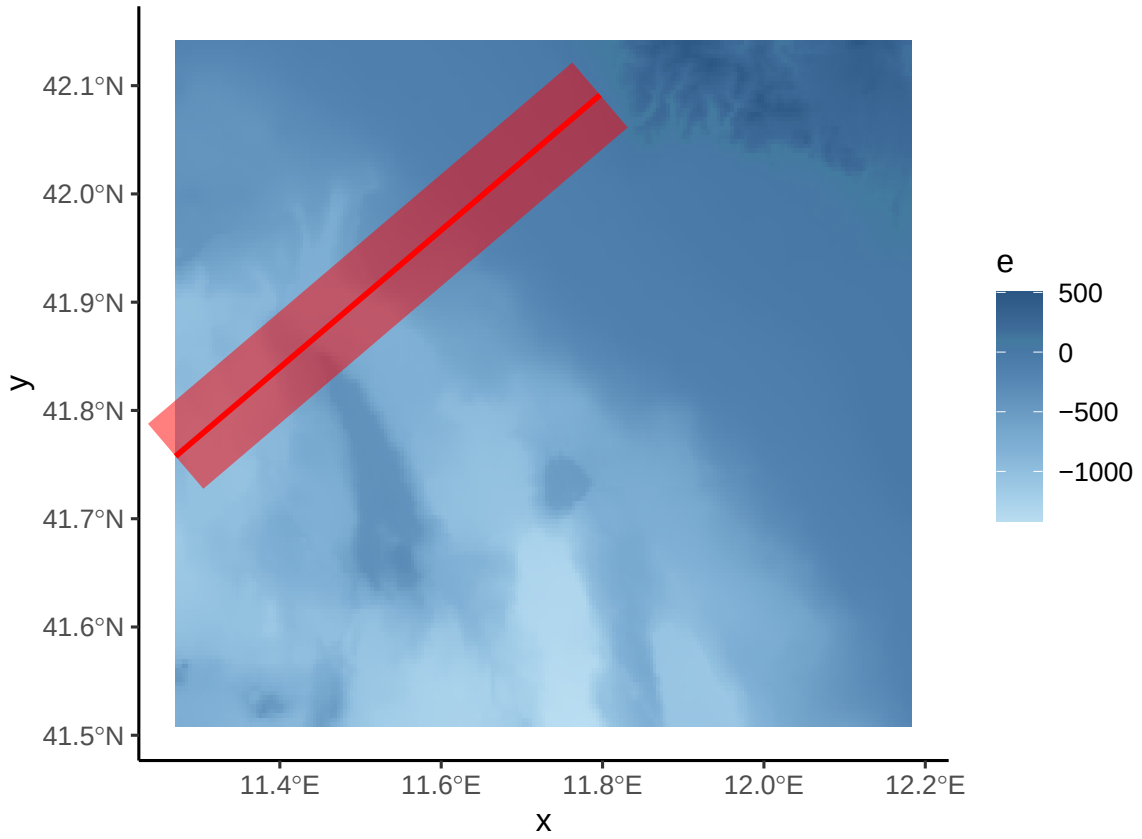



```
  coord_quickmap()
```

Extracting the area around the path

Note that, to avoid excessive calculation, we will limit this example to the area in the figure.

```
cable_data <- st_read("raw_data/subsea_cables_location/sub_cables.geojson")

janna_raw <- cable_data %>% filter(name == "Janna")

# Limiting the analysis (and the dataset) to the area in the figure
geodata_example_border <- st_as_sfc(st_bbox(geodata_example_raw))
janna <- st_intersection(janna_raw, geodata_example_border)

ggplot() + geom_raster(data = geodata_example, aes(x=x, y=y, fill=e)) +
  geom_sf(data = janna, aes(geometry = geometry), color = "red", linewidth = 15, alpha= 0.5) +
  geom_sf(data = janna, aes(geometry = geometry), color = "red", linewidth = 1, alpha= 1) +
  scale_fill_continuous_tableau() + theme_classic2()
```

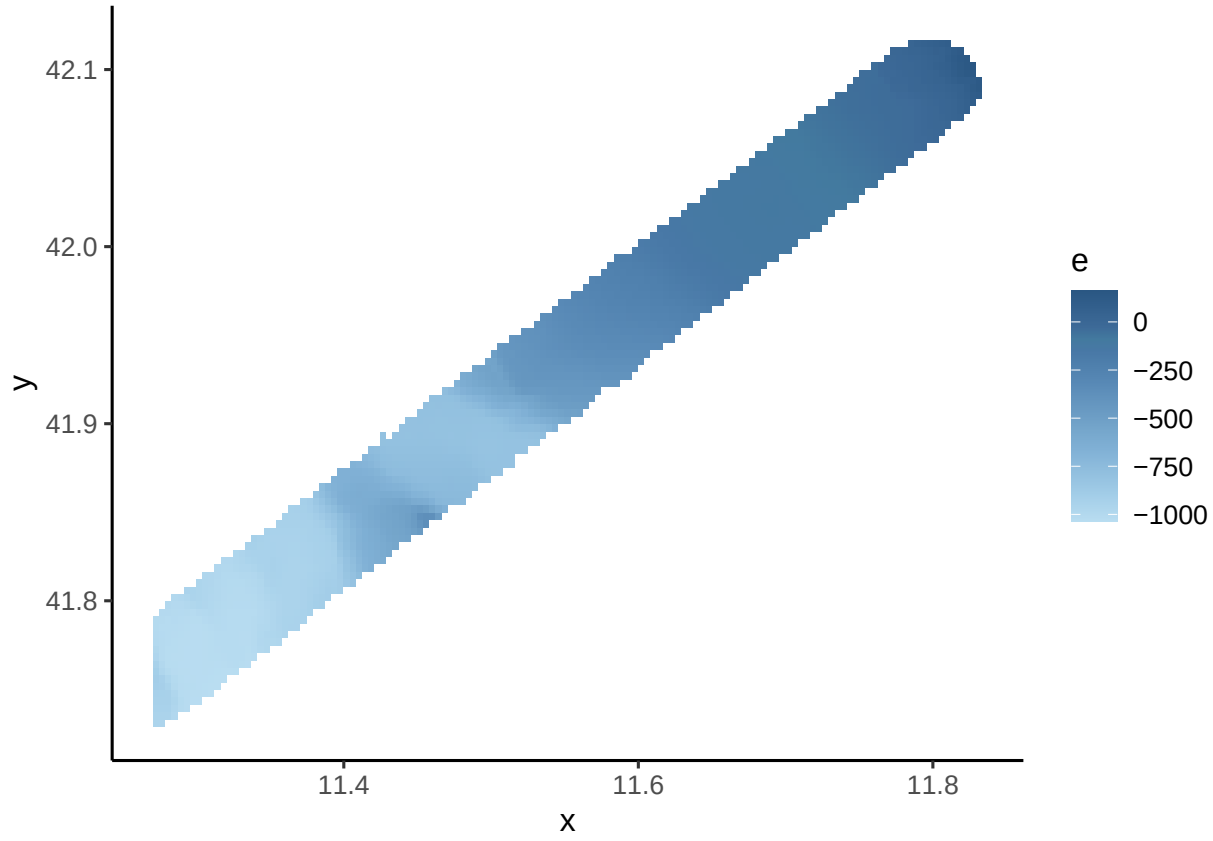



```
  coord_quickmap()
```

To compute the neighbouring area, we can leverage on the function st_buffer (package sf) and mask (package terra):

```
# Defining the area around the cable in meters
radius = 2500
janna_area <- st_buffer(janna, dist = radius)
```

```r
# Using the area around the cable to mask the dataset
janna_area <- mask(geodata_example_raw, janna_area)
janna_area <- as.data.frame(janna_area, xy = TRUE) %>% rename(e = example_bathymetry_data)

# Plotting results
ggplot() + geom_raster(data = janna_area, aes(x=x, y=y, fill=e ))+
  scale_fill_continuous_tableau() + theme_classic2()
```




```r
  coord_quickmap()
```

Computing Riley Ruggedness in each cell

First, we are going to rank each longitude and latitude, so to find the nearest cells

```r
janna_area <- janna_area %>% mutate(
  x_order = dense_rank(x),
  y_order = dense_rank(y)
)
```

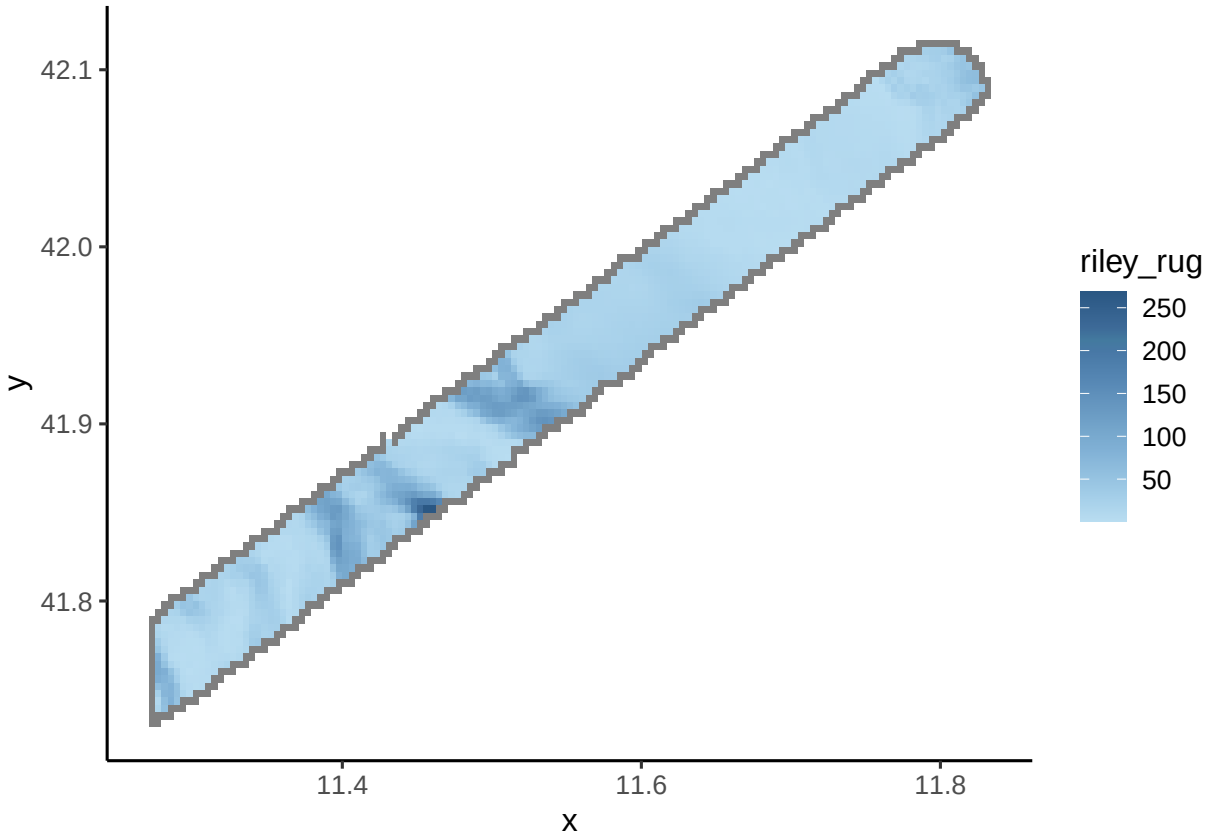
```r
calculate_riley_rug <- function(x_loc, y_loc, depth_loc, df) {
  # Filter neighboring points within the specified range
  neighboring_area <- df %>%
```

```
    filter(
      x_order == (x_loc + 1) | x_order == (x_loc - 1),
      y_order == (y_loc + 1) | y_order == (y_loc - 1)
    )

  # Check if neighboring_area has enough rows
  if (nrow(neighboring_area) < 4) {
    return(NA)  # Return NA if not enough neighbors
  }

  # Calculate the Riley Rug value
  riley_rug_value <- neighboring_area %>%
    mutate(
      sqhdist = (depth_loc - e)^2
    ) %>%
    summarize(riley_rug = sqrt(sum(sqhdist))) %>%
    pull(riley_rug)  # Extract riley_rug value directly

  return(riley_rug_value)
}
```

Applying the function:

```
janna_area <- janna_area %>%
  mutate(
    riley_rug = pmap_dbl(
      list(x_order, y_order, e),
      ~ calculate_riley_rug(..1, ..2, ..3, janna_area)
    )
  )
```

Which appears as:


42.1
42.0
41.9
41.8
y
11.4
11.6
11.8
x
riley_rug
250
200
150
100
50