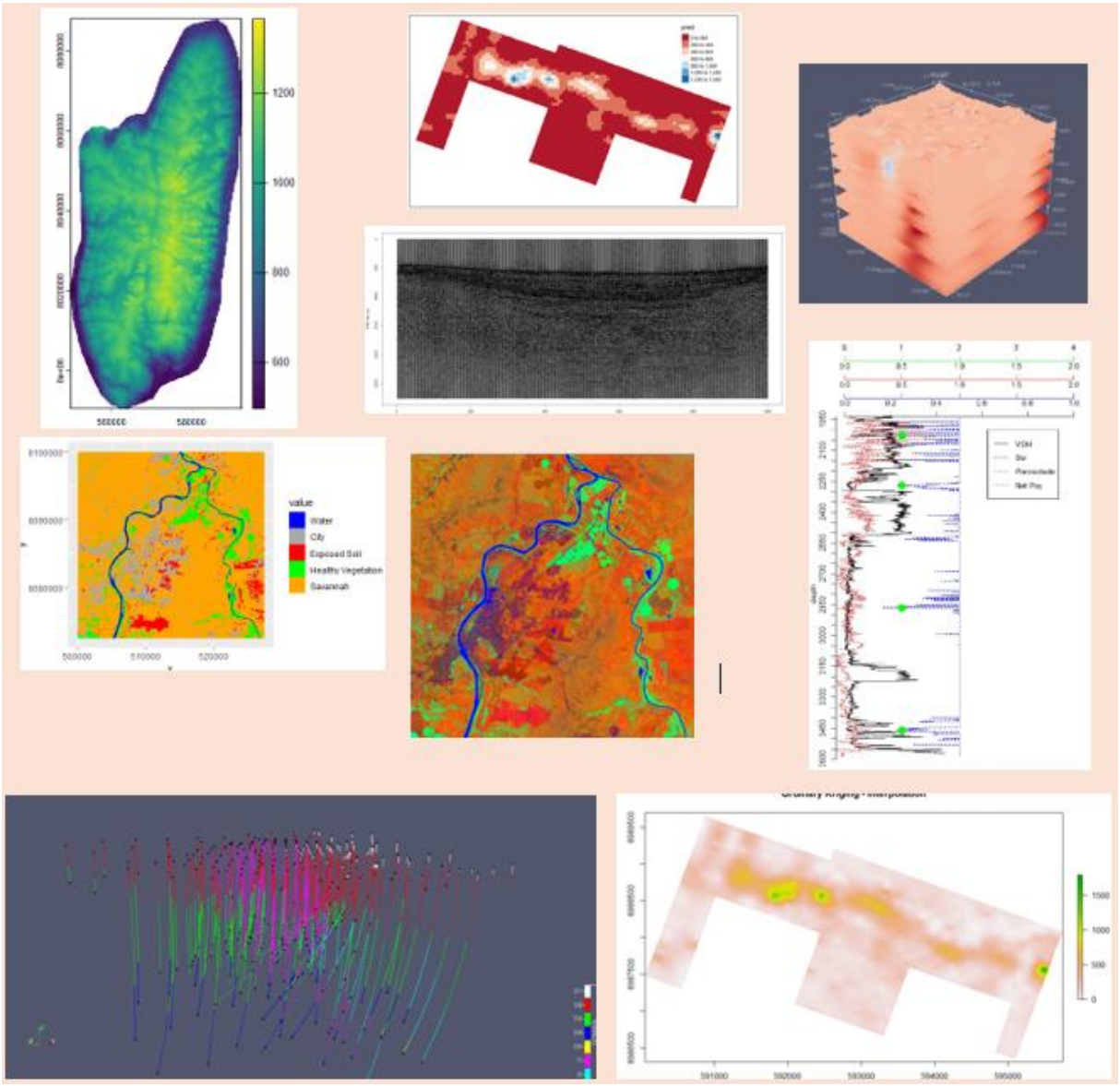
Cookbook R:

Processamento e Manipulação de Dados Espaciais Geológicos com R



André Luiz Lima Costa, M.Sc., P.Geo., FAIG

Edição Revisada - Setembror 2024

Introdução

Nesta nova edição do Cookbook vamos focar nas tarefas de carregamento, importação, manipulação, processamento e exportação de dados geoespaciais e outros usando R e suas bibliotecas.

Avançaremos progressivamente na medida que novos processamentos e manipulações são introduzidos.

Bibliotecas usadas:

terra	*rpostgis*	*raster*	*RColorBrewer*	
dismo	*gstat*	*gdalUtils*	*spdep*	*spgwr*
cluster	*float*	*DT*	*data.table*	*tmap*
RStollbox	*ggplot2*			

Baixe o Cookbook e os dados em:

https://gdatasystems.com/apostilaRgeo/CookbookR_geo_v01_pt.pdf
https://gdatasystems.com.br/apostilaRgeo/index.php

Todas as receitas devem iniciar com:

```
#Diretório de trabalhe onde os dados estão
wd<-'C:/Users/User/cookbooksitefiles'#modifique de acordo com o seu sistema
setwd(wd)
```

VISITE:

https://gdatasystems.com

1 Carregando dados geoespaciais

Receita - 1.1 – Carregando Dado Vetorial

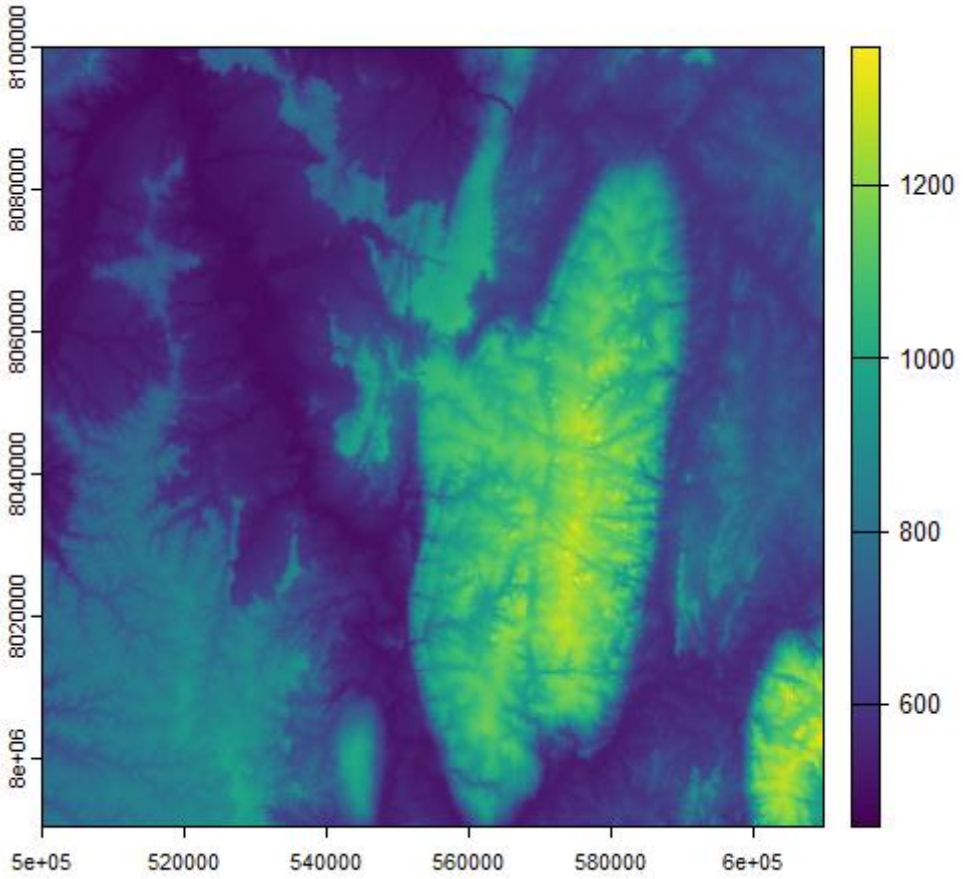
Carregar, inspecionar e visualizar um dado vetorial a partir de um arquivo shapefile.

```
library(terra) #install terra using install.packages("terra")
estado<-vect("estado.shp") #carrega o shapefile estado.shp
estado  #inspecionando
 class       : SpatVector
 geometry    : polygons
 dimensions  : 27, 5  (geometries, attributes)
 extent      : -73.99024, -32.39088, -33.75136, 5.270972  (xmin,xmax,ymin,ymax)
 source      : estado.shp
 coord. ref. : lon/lat GCS_unknown
 names       :    id     nome sigla regiao_id codigo_ibg
 type        : <chr>    <chr> <chr>     <chr>      <chr>
 values      :     1     Acre    AC         3         12
                   2  Alagoas    AL         4         27
                   3 Amazonas    AM         3         13
plot(estado)
```

Receita - 1.2 – Carregando Dado Raster

Carregar, inspecionar e visualizar um dado raster a partir de um arquivo geoTiff.

```
library(terra)
dem<-rast("dem.tif")
dem
class       : SpatRaster
dimensions  : 3650, 3649, 1  (nrow, ncol, nlyr)
resolution  : 30.08495, 30.08219  (x, y)
extent      : 5e+05, 609780, 7990240, 8100040  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
source      : dem.tif
name        : dem
plot(dem)
```



2 Criando dados geoespaciais

Receita - 2.1 – Criando Dado Vetorial

Criar um dado vetorial a partir de pontos com coordenadas X-Y, georreferenciar, adicionar dados ao ponto e salvar como shapefile.

2.1.1 Criando o dado

```
longitude <- c(-116.7, -120.4, -116.7, -113.5, -115.5,-120.8, -119.5, -113.7, -
113.7, -110.7)
latitude <- c(45.3, 42.6, 38.9, 42.1, 35.7, 38.9,36.2, 39, 41.6, 36.9)
pontos<-cbind(longitude, latitude) #une long-lat em uma matrix
library(terra)
pts<-vect(pontos)
pts
 class       : SpatVector
 geometry    : points
 dimensions  : 10, 0  (geometries, attributes)
 extent      : -120.8, -110.7, 35.7, 45.3  (xmin, xmax, ymin, ymax)
 coord. ref. :
```

2.1.2 Georreferenciando

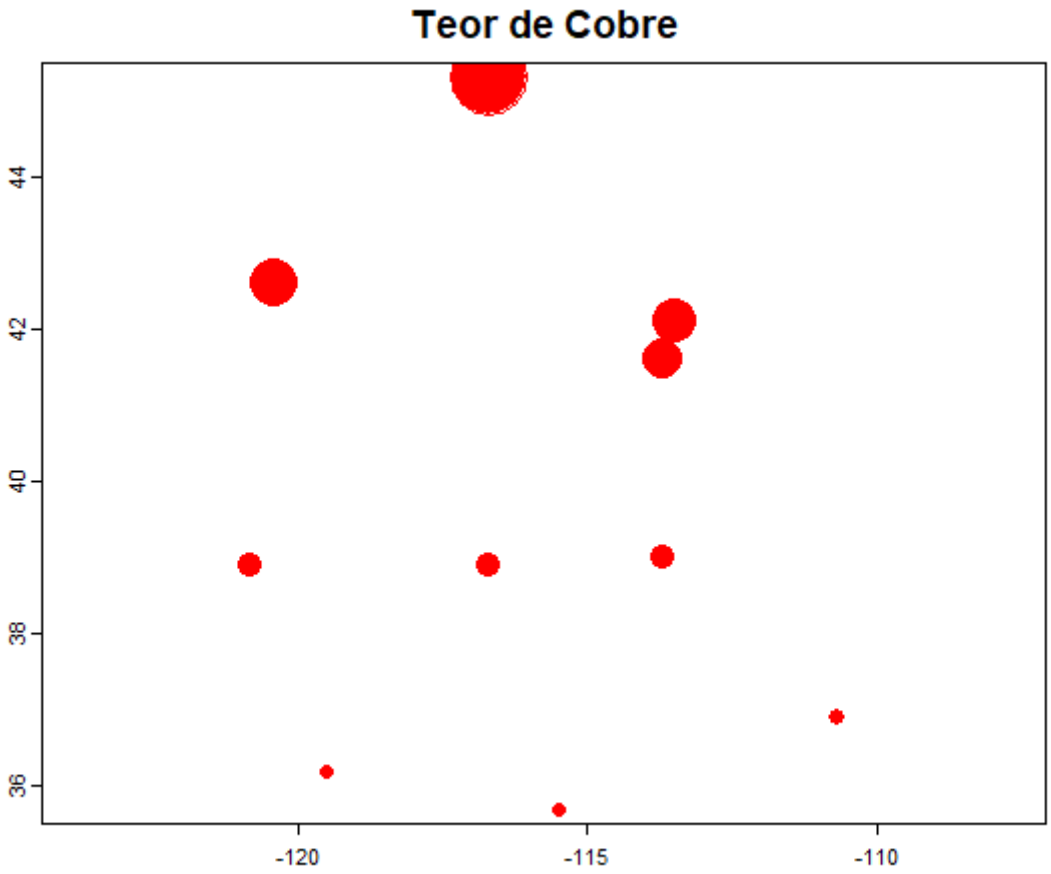
```
pts<-vect(pontos,crs="WGS84")
pts
 class       : SpatVector
 geometry    : points
 dimensions  : 10, 0  (geometries, attributes)
 extent      : -120.8, -110.7, 35.7, 45.3  (xmin, xmax, ymin, ymax)
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
```

2.1.3 Adicionando dados

```
df <- data.frame(ID=1:nrow(pontos), teorCu=(latitude-30)^3)
df
   ID   teorCu
1   1 3581.577
2   2 2000.376
3   3  704.969
4   4 1771.561
5   5  185.193
6   6  704.969
7   7  238.328
8   8  729.000
9   9 1560.896
10 10  328.509
ptsdf<-vect(pontos,crs="WGS84", atts=df)
ptsdf
 class       : SpatVector
 geometry    : points
 dimensions  : 10, 2  (geometries, attributes)
 extent      : -120.8, -110.7, 35.7, 45.3  (xmin, xmax, ymin, ymax)
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
 names       :    ID teorCu
 type        : <int>  <num>
 values      :     1   3582
                   2   2000
                   3    705
```

2.1.4 Visualizando

```
plot(ptsdf, cex=1+ptsdf$teorCu/500, pch=20, col='red', main='Teor de Cobre')
```



2.1.5 Gravando como shapefile

```
writeVector(ptsdf, 'pontocobre.shp', overwrite=TRUE)
```

Receita - 2.2 – Criando Dado Raster

Criar um dado raster georreferenciado, carregar com valores aleatórios, visualizar e salvar como arquivo geoTiff.

2.2.1 Criando raster georreferenciado em 32723 (UTM WGS84 zona 23S)

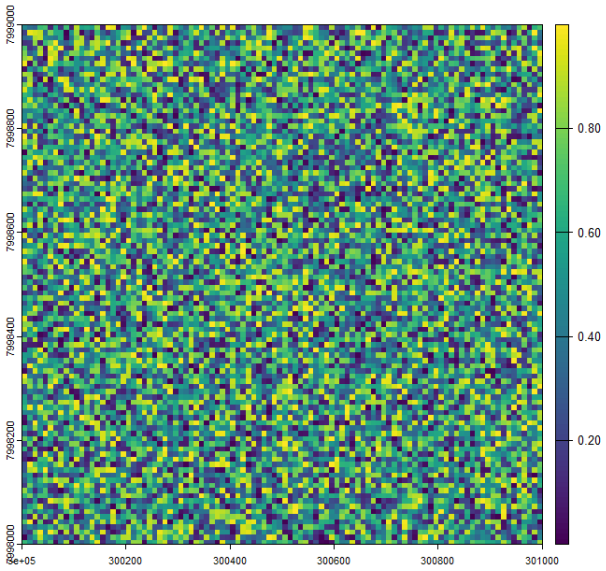
```
library(terra)
rl<-rast(ncol=100,nrow=100,xmax=301000,xmin=300000,ymin=7998000, ymax=7999000,
         crs="+init=epsg:32723")
rl
class       : SpatRaster
dimensions  : 100, 100, 1  (nrow, ncol, nlyr)
resolution  : 10, 10  (x, y)
extent      : 3e+05, 301000, 7998000, 7999000  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 23S
```

2.2.2 Adicionando dados com valores aleatórios

```
values(rl)<-runif(ncell(rl))
rl
class       : SpatRaster
dimensions  : 100, 100, 1  (nrow, ncol, nlyr)
resolution  : 10, 10  (x, y)
extent      : 3e+05, 301000, 7998000, 7999000  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 23S
source(s)   : memory
name        :      lyr.1
min value   : 7.768977e-05
max value   : 9.999792e-01
```

2.2.3 Visualizando o raster

```
plot(rl)
```



2.2.4 Gravando em arquivo geoTiff

```
writeRaster(rl, 'rastersaida.tif',overwrite=TRUE)
```

3 Atributos/valores de dados geoespaciais

Receita - 3.1 – Extraindo atributos de Dado Vetorial

Ler um dado vetorial, extrair uma coluna de seus atributos e exportar como CSV.

```
library(terra)
arq<-vect("minasgerais_recmin.shp")
arq
 class       : SpatVector
 geometry    : points
 dimensions  : 6818, 24  (geometries, attributes)
 extent      : -50.94278, -39.92139, -22.86944, -14.29722  (xmin,xmax, ymin, ymax)
 source      : minasgerais_recmin.shp
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
 names       : data_cadas subst_prin abreviacao      associacao associac_1
 type        :      <chr>      <chr>      <chr>           <chr>      <chr>
 values      : 2003/11/26       OURO         Au Ouro, Pirita, ~         Au
               2004/02/06   MANGANÊS         Mn                NA    Mn - Fe
               2003/11/26       OURO         Au Ouro, Pirita, ~         Au
 extrmin_x grau_de_in      metodo_geo  textura_mi        origem (and 14 more)
     <chr>      <chr>           <chr>       <chr>         <chr>
 Filoneana         NA Levantamento e~ Disseminada São Francisco
    Maciça         NA Levantamento e~    Granular São Francisco
df<-cbind(crds(arq),arq$subst_prin)
head(df)
     x                   y
[1,] "-43.7547222199999" "-20.5644444399999" "OURO"
[2,] "-43.8311111099999" "-20.56416667"      "MANGANÊS"
[3,] "-43.7616666699999" "-20.56305556"      "OURO"
[4,] "-43.77138889"      "-20.56138889"      "OURO"
[5,] "-43.76944444"      "-20.56138889"      "OURO"
[6,] "-43.77333333"      "-20.55861111"      "OURO"
colnames(df)<-c("Longitude","Latitude","Ocorrência")
head(df)
     Longitude           Latitude            Ocorrência
[1,] "-43.7547222199999" "-20.5644444399999" "OURO"
[2,] "-43.8311111099999" "-20.56416667"      "MANGANÊS"
[3,] "-43.7616666699999" "-20.56305556"      "OURO"
[4,] "-43.77138889"      "-20.56138889"      "OURO"
[5,] "-43.76944444"      "-20.56138889"      "OURO"
[6,] "-43.77333333"      "-20.55861111"      "OURO"
write.csv(df, "ocorrencias.csv", row.names=FALSE)
```

Receita - 3.2 – Extraindo Valores de Dado Raster

Carregar um dado raster, extrair o valor e exportar como CSV.

```
library(terra)
arq<-rast("dem.tif")
arq
class       : SpatRaster
dimensions  : 3650, 3649, 1  (nrow, ncol, nlyr)
resolution  : 30.08495, 30.08219  (x, y)
extent      : 5e+05, 609780, 7990240, 8100040  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
source      : dem.tif
name        : dem
elev<-values(arq)
df<-cbind(crds(arq),elev)
colnames(df)<-c("utme","utmn","elev")
head(df)
         utme    utmn elev
[1,] 500015.0 8100025  704
[2,] 500045.1 8100025  704
[3,] 500075.2 8100025  703
[4,] 500105.3 8100025  703
[5,] 500135.4 8100025  702
[6,] 500165.5 8100025  702
write.csv(df, "elevacao.csv", row.names=FALSE)
```

4 Reprojetando dados Geoespaciais

Receita - 4.1 – Reprojetando dados Vector

Carregar um dado vector, reprojetar e exportar como um novo shapefile.

```
library(terra)
arq<-vect("minasgerais_recmin.shp")
arq
 class       : SpatVector
 geometry    : points
 dimensions  : 6818, 24  (geometries, attributes)
 extent      : -50.94278, -39.92139, -22.86944, -14.29722  (xmin,xmax, ymin, ymax)
 source      : minasgerais_recmin.shp
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
 names       : data_cadas subst_prin abreviacao      associacao associac_1
 type        :      <chr>      <chr>      <chr>           <chr>      <chr>
 values      : 2003/11/26       OURO         Au Ouro, Pirita, ~         Au
               2004/02/06   MANGANÊS         Mn                NA    Mn - Fe
               2003/11/26       OURO         Au Ouro, Pirita, ~         Au
 extrmin_x grau_de_in      metodo_geo  textura_mi        origem (and 14 more)
     <chr>      <chr>           <chr>       <chr>         <chr>
 Filoneana         NA Levantamento e~ Disseminada São Francisco
    Maciça         NA Levantamento e~    Granular São Francisco
 Filoneana         NA Levantamento e~ Disseminada São Francisco
arq2<-project(arq, "EPSG:32723")
arq2
class       : SpatVector
 geometry    : points
 dimensions  : 6818, 24  (geometries, attributes)
 extent      : -122394.4, 1043414, 7470310, 8419329  (xmin, xmax, ymin, ymax)
 coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
 names       : data_cadas subst_prin abreviacao      associacao associac_1
 type        :      <chr>      <chr>      <chr>           <chr>      <chr>
 values      : 2003/11/26       OURO         Au Ouro, Pirita, ~         Au
               2004/02/06   MANGANÊS         Mn                NA    Mn - Fe
               2003/11/26       OURO         Au Ouro, Pirita, ~         Au
 extrmin_x grau_de_in      metodo_geo  textura_mi        origem (and 14 more)
     <chr>      <chr>           <chr>       <chr>         <chr>
 Filoneana         NA Levantamento e~ Disseminada São Francisco
    Maciça         NA Levantamento e~    Granular São Francisco
 Filoneana         NA Levantamento e~ Disseminada São Francisco
writeVector(arq2, 'recmin_reprjt.shp', overwrite=TRUE)
```

Receita - 4.2 – Reprojetando dados Raster

Carregar um dado raster, reprojetar e exportar como um novo tif.

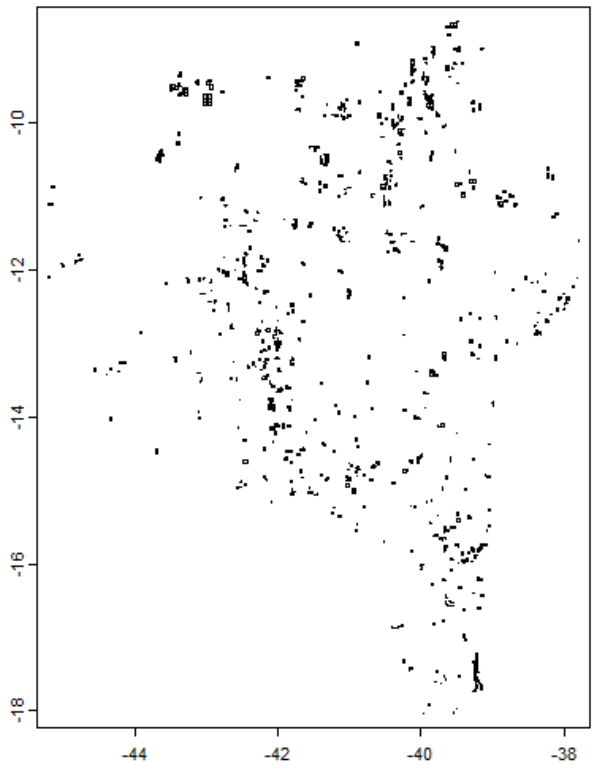
```
library(terra) #install terra using install.packages("terra")
arq<-rast("dem.tif")
arq
class       : SpatRaster
dimensions  : 3650, 3649, 1  (nrow, ncol, nlyr)
resolution  : 30.08495, 30.08219  (x, y)
extent      : 5e+05, 609780, 7990240, 8100040  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
source      : dem.tif
name        : dem
arq2<-project(arq, "EPSG:4326")
arq2
class       : SpatRaster
dimensions  : 3581, 3735, 1  (nrow, ncol, nlyr)
resolution  : 0.0002778776, 0.0002778776  (x, y)
extent      : -45, -43.96213, -18.17689, -17.18181  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source(s)   : memory
name        :       dem
min value   :  439.9265
max value   : 1369.7501
writeRaster(arq2, 'dem_reprjt.tif', overwrite=TRUE)
```

5 Acessando Banco de Dados Geoespacial

Receita - 5.1 – Carregando objeto geoespacial vector de banco de dados remoto

Carregar uma camada de áreas que foram para edital na ANM localizado em um banco de dados remoto e salvar como shapefile localmente.

```
library(terra)
cstr<-"PG:dbname=dnpmba host=pg.gdatasystems.com user=droid password=devcor
       port=5432"
lyr <- vect(cstr, layer="edital") # name of vector table
lyr
 class       : SpatVector
 geometry    : polygons
 dimensions  : 708, 13  (geometries, attributes)
 extent      : -45.21476, -37.78562, -18.03977, -8.610073  (xmin, xmax, ymin,
ymax)
 source              :  PG:dbname=dnpmba  host=pg.gdatasystems.com  user=droid
password=devcor port=5432  (edital)
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
 names       :   gid    processo numero   ano area_ha                  id
 type        : <int>        <chr>  <int> <int>   <num>              <chr>
 values      :   575 870509/2002 870509  2002   785.5 {23B096F9-D7C2~
                 801 872126/2006 872126  2006   996.6 {E9BA669A-1C5A~
                1212 871762/2008 871762  2008   575.6 {70DE0C14-15C8~
            fase      ult_evento             nome            subs (and 3 more)
           <chr>           <chr>            <chr>           <chr>
 DISPONIBILIDADE 2468 - DISPONI~ MARCIA OSIAS D~          ARGILA
 DISPONIBILIDADE 2468 - DISPONI~ CARLANE CLÉA R~         QUARTZO
 DISPONIBILIDADE 2468 - DISPONI~ JOSE FARIAS DE~ MINÉRIO DE VAN~
plot(lyr)
```

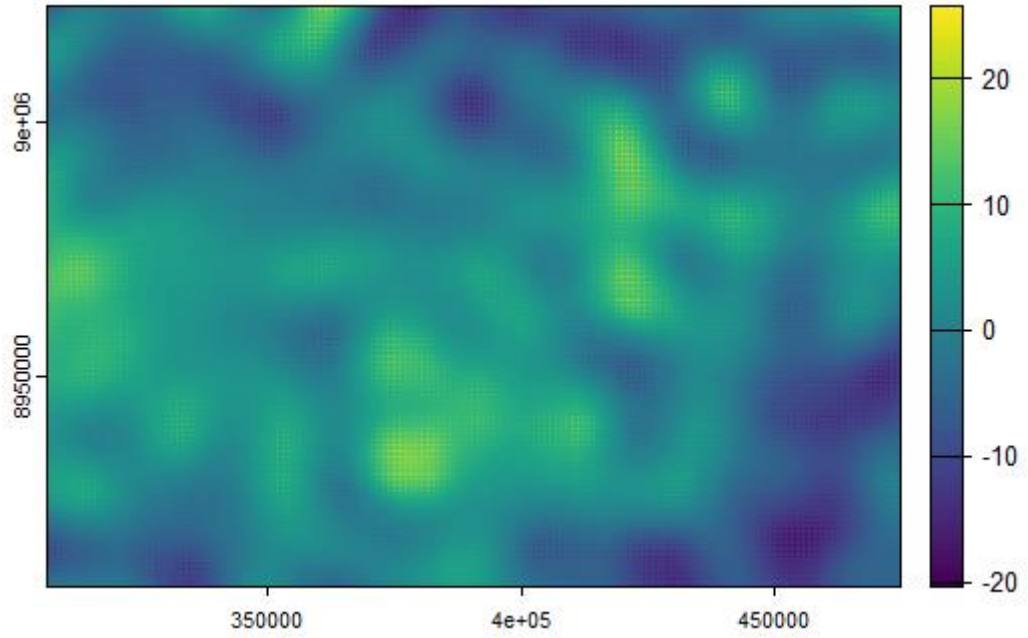



```
writeVector(lyr, 'ucusBA.shp', overwrite=TRUE)
```

Receita - 5.2 – Carregando objeto geoespacial raster de banco de dados remoto

Carregar uma imagem raster de um banco de dados remoto e salvar como geoTiff localmente.

```
library(terra)
cstr2<-"PG:dbname=dnpmmt host=pg.gdatasystems.com user=droid password=devcor
          port=5432 schema='public' table='sai'"
camada <- rast(cstr2)
camada
class       : SpatRaster
dimensions  : 382, 562, 1  (nrow, ncol, nlyr)
resolution  : 299.5464, 299.4152  (x, y)
extent      : 306437.9, 474783, 8908453, 9022829  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 21S (EPSG:32721)
source      : PG:dbname=dnpmmt host=pg.gdatasystems.com user=droid password=devcor
port=5432 schema='public' table='sai'
varname     : PG:dbname=dnpmmt host=pg.gdatasystems.com
name        : PG:dbname=dnpmmt host=pg.gdatasystems.com
plot(camada)
```



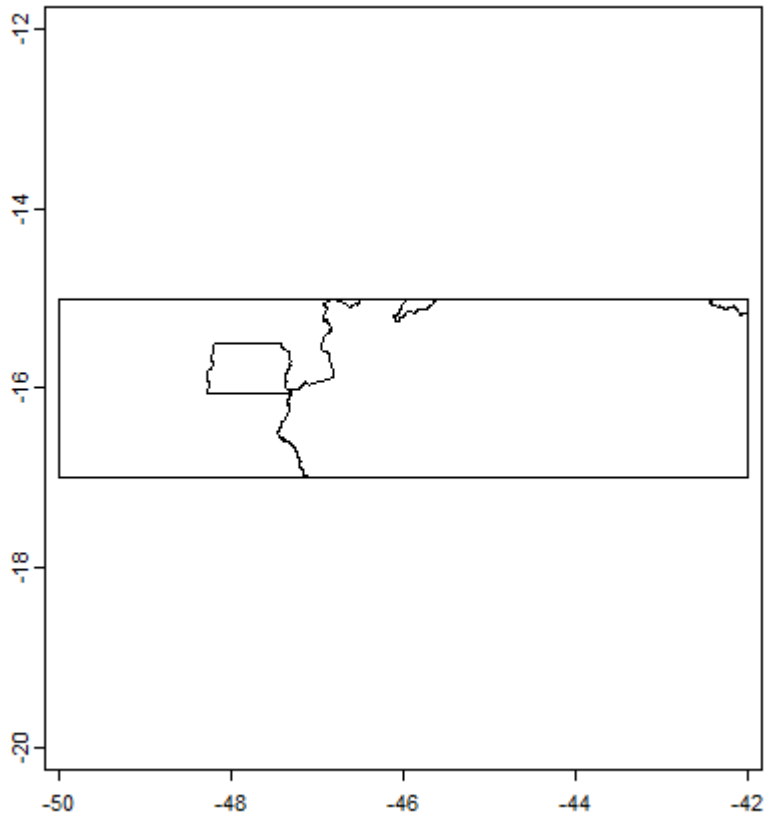

```
writeRaster(camada, 'output.tif', overwrite=TRUE)
```

6 Recortando dados Geoespaciais

Receita - 6.1 – Recortando dado vector com retângulo

Recortar o dado vector carregado usando um retângulo definido pela função ext(x mínimo, x máximo, y mínimo, y máximo).

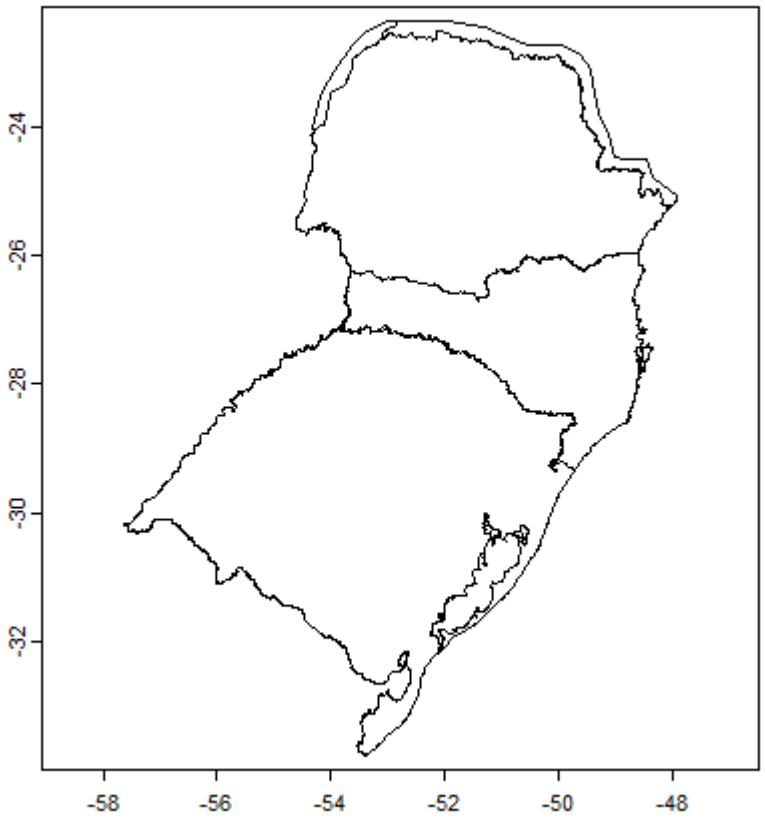
```
library(terra)
estado<-vect("estado.shp") #carrega o shapefile estado.shp
ext<-ext(-50,-42,-17,-15)
extraido<-crop(estado,ext)
extraido
 class       : SpatVector
 geometry    : polygons
 dimensions  : 4, 5  (geometries, attributes)
 extent      : -50, -42, -17, -15  (xmin, xmax, ymin, ymax)
 coord. ref. : lon/lat GCS_unknown
 names       :    id              nome sigla regiao_id codigo_ibg
 type        : <chr>             <chr> <chr>     <chr>      <chr>
 values      :     5             Bahia    BA         4         29
                   7 Distrito Federal    DF         5         53
                   9             Goiás    GO         5         52
plot(extraido)
```



Receita - 6.2 – Recortando dado vector com shapefile

Recortar o dado vector carregado usando um outro arquivo vector como máscara.

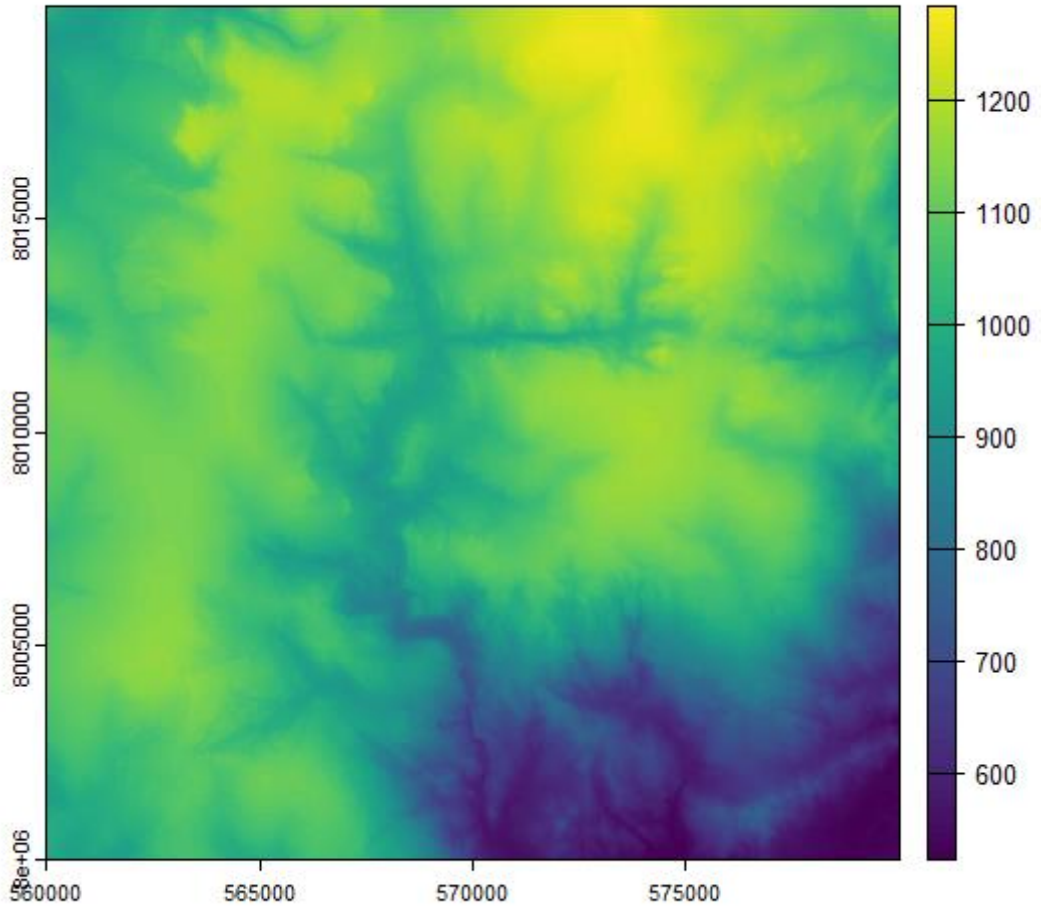
```
library(terra)
estado<-vect("estado.shp") #carrega o shapefile estado.shp
sul<-vect('sul.shp')
sul
 class       : SpatVector
 geometry    : polygons
 dimensions  : 1, 1  (geometries, attributes)
 extent      : -57.83859, -47.86047, -34.01961, -22.37426  (xmin,xmax,ymin,ymax)
 source      : sul.shp
 coord. ref. : lon/lat WGS 84 (EPSG:4326)
 names       :    id
 type        : <int>
 values      :    NA
extraido<-crop(estado,sul)
extraido
 class       : SpatVector
 geometry    : polygons
 dimensions  : 5, 5  (geometries, attributes)
 extent      : -57.64272, -47.90666, -33.75136, -22.37426  (xmin,xmax,ymin,ymax)
 coord. ref. : lon/lat GCS_unknown
 names       :    id                nome sigla regiao_id codigo_ibg
 type        : <chr>               <chr> <chr>     <chr>      <chr>
 values      :    12 Mato Grosso do Sul    MS         5         50
                  18             Paraná    PR         1         41
                  23  Rio Grande do Sul    RS         1         43
plot(extraido)
```



Receita - 6.3 – Recortando dado raster com retângulo

Recortar o dado raster carregado usando um retângulo definido pela função ext(x mínimo, x máximo, y mínimo, y máximo).

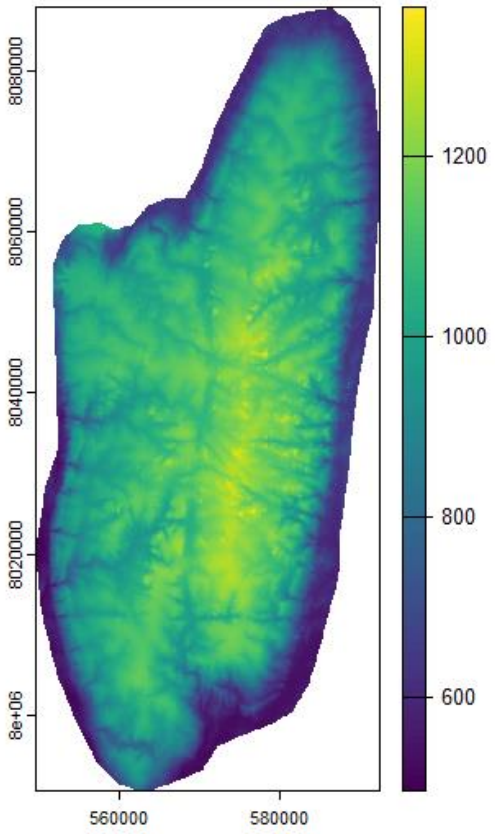
```
library(terra)
dem<-rast("dem.tif")
ext<-ext(560000,580000,8000000,8020000)
extraido<-crop(dem,ext)
extraido
 class       : SpatRaster
 dimensions  : 665, 665, 1  (nrow, ncol, nlyr)
 resolution  : 30.08495, 30.08219  (x, y)
 extent      : 559989.4, 579995.9, 7999987, 8019991  (xmin, xmax, ymin, ymax)
 coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
 source(s)   : memory
 name        :  dem
 min value   :  524
 max value   : 1284
plot(extraido)
```



Receita - 6.4 – Recortando dado raster com shapefile

Recortar o dado raster carregado usando um arquivo vector como máscara.

```
library(terra)
dem<-rast("dem.tif")
serra<-vect('serra.shp')
serra
 class       : SpatVector
 geometry    : polygons
 dimensions  : 1, 1  (geometries, attributes)
 extent      : 549704.2, 592405.6, 7990569, 8087955  (xmin, xmax, ymin, ymax)
 source      : serra.shp
 coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
 names       :    id
 type        : <int>
 values      :    NA
extraido<-crop(dem,serra,mask=T)
extraido
 class       : SpatRaster
 dimensions  : 3237, 1419, 1  (nrow, ncol, nlyr)
 resolution  : 30.08495, 30.08219  (x, y)
 extent      : 549700.3, 592390.9, 7990571, 8087947  (xmin, xmax, ymin, ymax)
 coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
 source(s)   : memory
 name        :  dem
 min value   :  492
 max value   : 1373
plot(extraido)
```



7 Carregando Dados de Arquivos Externos

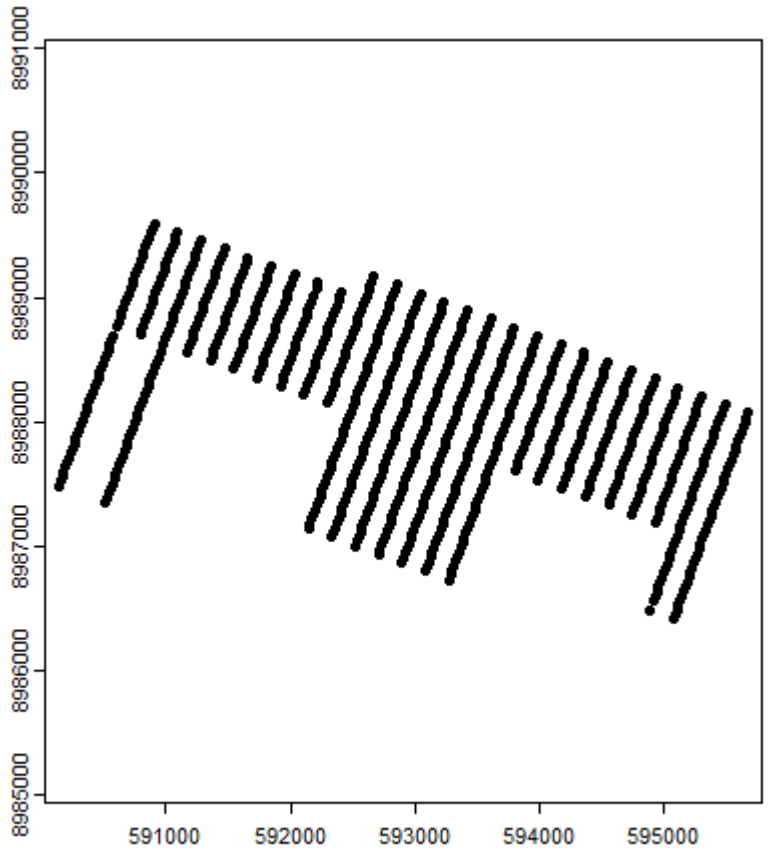
Receita - 7.1 – Dados no Formato CSV

Dados estruturados no formato texto delimitado por vírgulas (ou qualquer outro delimitador) podem ser carregados facilmente como um dataframe que posteriormente, usando as colunas de coordenadas, podem ser transformados em dado vector e um CRS pode ser assinalado a este. Podemos agora gravar este vector como shapefile para ser usado posteriormente.

```
geoq<-read.csv('dado.csv')
colnames(geoq)[1]<-"amostra"
library(terra)
geoqui<-vect(geoq, geom=c("UTME", "UTMN"), crs="epsg:32717")
geoqui
```

```
 class       : SpatVector
 geometry    : points
 dimensions  : 956, 16  (geometries, attributes)
 extent      : 590142, 595675, 8986418, 8989594  (xmin, xmax, ymin, ymax)
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       : ï..amostra Au_gpt    Ba    Cd    Co    Cr    Cu    Mo    Ni   Pb (and 6 more)
 type        :      <int>  <num> <num> <num> <num> <num> <num> <num> <num> <num>
 values      :          1   0.01   5.1   6.8     3    58    46     1   5.2    12
                        2   0.01    10   3.9     3    38    35     1   4.6    11
                        3   0.01   7.1   3.7     3    43    29     1     4    12
```

```
plot(geoqui)
```




```
writeVector(geoqui, 'geoqui.shp', overwrite=TRUE)
```

O arquivo geoqui.shp será usado no capítulo 8 – Interpolação de Pontos.

Receita - 7.2 – Dados no Formato LAS

Carregando um arquivo LAS (Log ASCII Standard), visualizando os dados e executando uma análise petrofísica.

7.2.1 Lendo Arquivo

```
las<-readLines('1046531020.las')
data<-''
for (i in 1:length(las)){
 if(substr(las[i], 1, 2) == "~A"){
  data<- las[(i+1):length(las)]
  break
 }
}
hd<-''
a<-1
go<-0
for (i in 1:length(las)){
  if(substr(las[i], 1, 2) == "~C"){go<-1}
  if(substr(las[i+1], 1, 2) == "~P" || substr(las[i+1], 1, 2) == "~A" ||
  substr(las[i+1], 1, 2) == "~O"){break}
  if(go==1){
   value<-strsplit(trimws(las[i+1],"l"),'\\s{1,}')[[1]]
   hd[a]<-value[1]
   a<-a+1
  }
}
las.data<-read.table(header=TRUE, text=data)
names(las.data)<-hd
las.data[las.data==-999.2500]<-NA
las.df<-las.data[which(las.data$DEPT>250 & las.data$DEPT<3590),]
head(las.df)
```

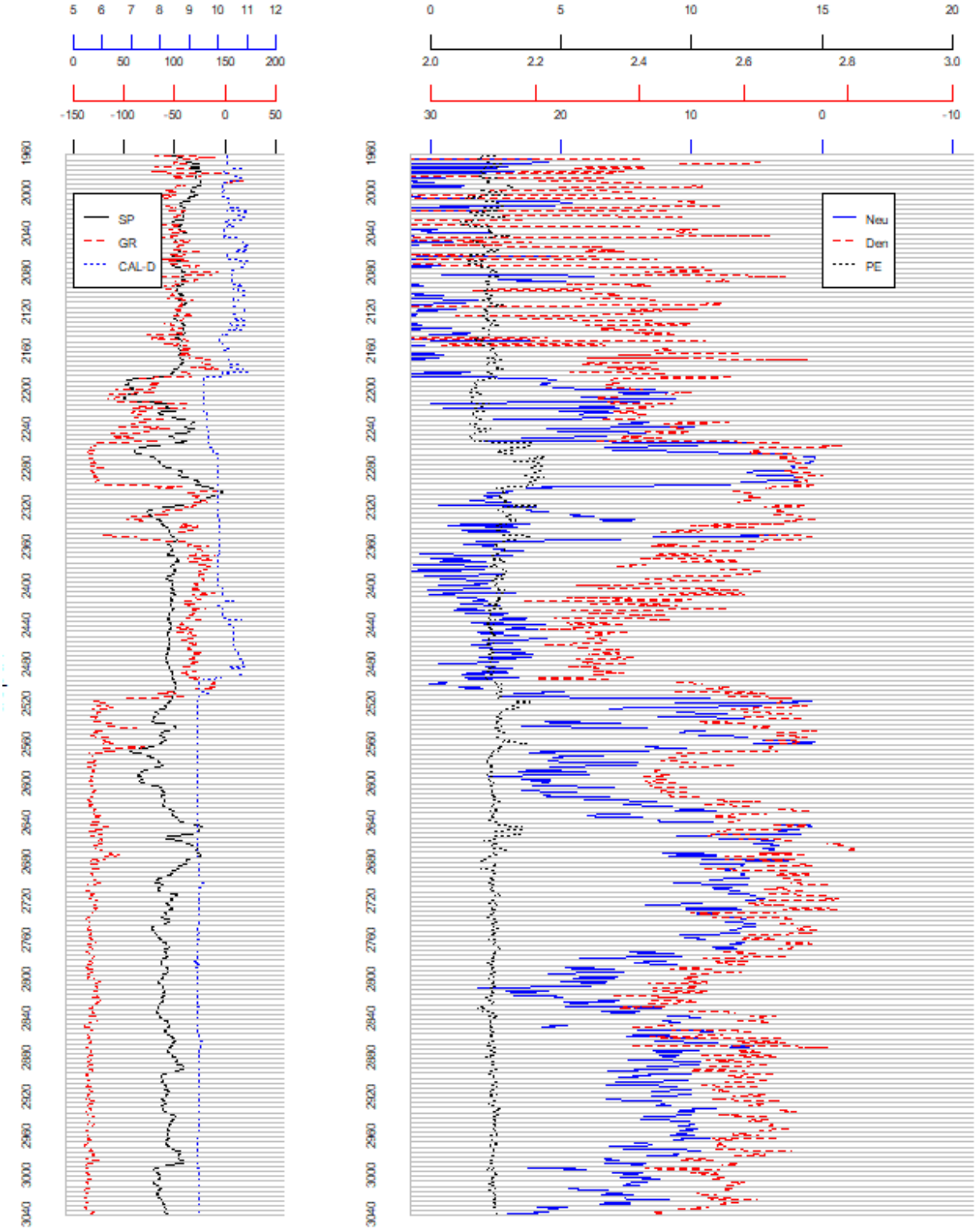
```
     DEPT  CILD CALN CALD    GR   ILD  ILM MINV    LWTLB CALM MNOR  NPLS  DPLS RXRT  SFL      SP  DRHO RHOB   PE
INV_RF NOR_RF
113 250.5 27.64 2.25 10.3 62.98 36.18 4.12   NA 1543.53   NA   NA 51.34 67.41   NA 0.05 114.94 -0.97 1.56 8.62
NA     NA
114 251.0 28.26 2.25 10.3 59.10 35.38 4.29   NA 1547.25   NA   NA 46.84 69.38   NA 0.05 115.51 -1.01 1.52 8.66
NA     NA
115 251.5 28.94 2.25 10.3 58.81 34.55 4.51   NA 1552.30   NA   NA 43.84 70.23   NA 0.05 116.17 -1.02 1.51 8.66
NA     NA
116 252.0 29.58 2.25 10.3 60.60 33.80 4.73   NA 1556.61   NA   NA 42.89 69.14   NA 0.05 116.52 -1.01 1.53 8.72
NA     NA
117 252.5 30.10 2.25 10.3 59.48 33.22 4.95   NA 1562.14   NA   NA 41.38 66.90   NA 0.05 116.60 -0.97 1.57 8.57
NA     NA
118 253.0 30.57 2.25 10.3 59.71 32.71 5.13   NA 1572.72   NA   NA 41.41 66.57   NA 0.05 116.58 -0.96 1.57 8.31
NA     NA
```

7.2.2 Visualizando os dados

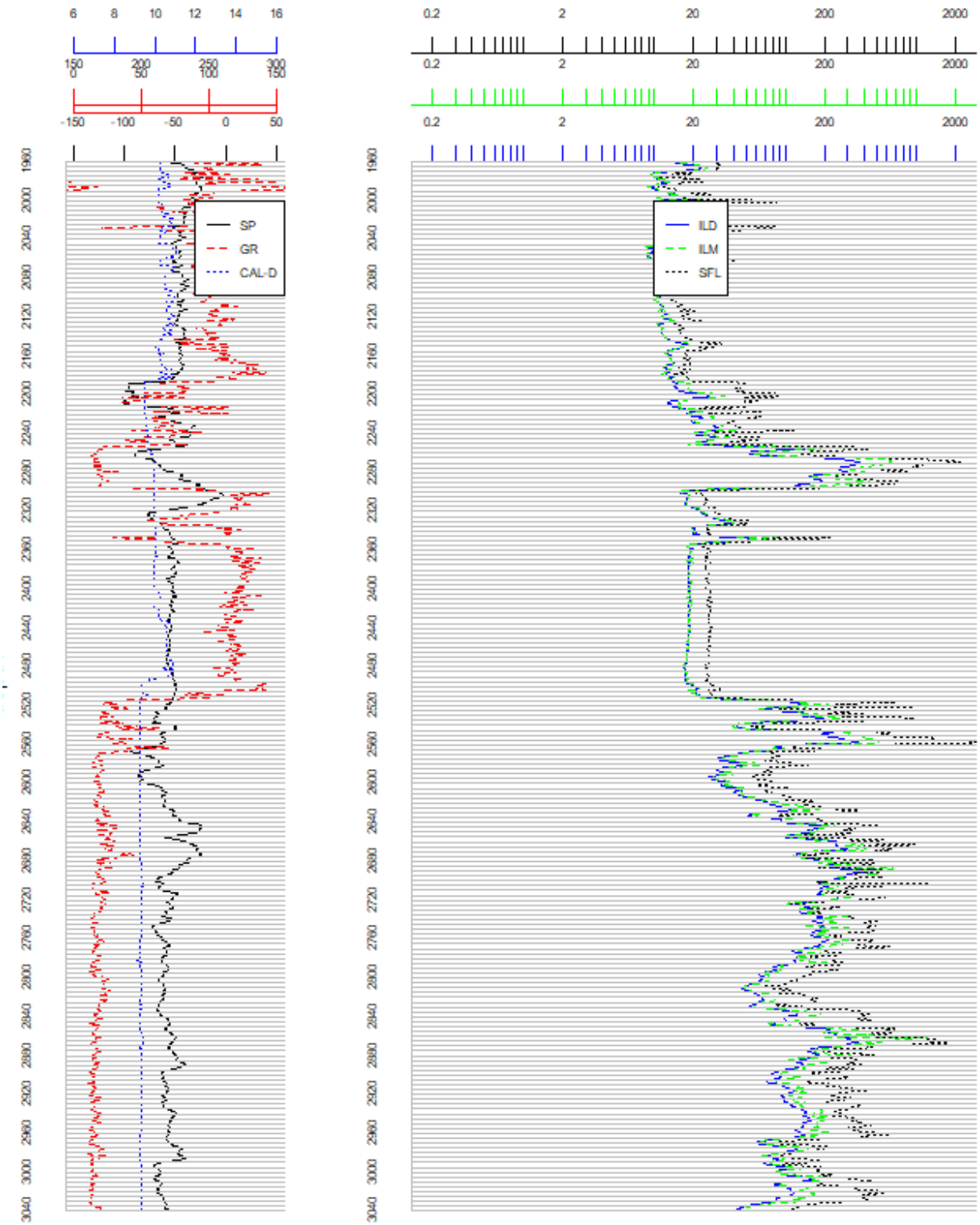
```
graf1<-function(base,topo){ #curvas SP, GR e CALD associadas com NPLS, RHOB e PE
 par()
 dev.off()
 par(mar=c(1,2,5,2) + 0.1)
 layout(matrix(c(1,2),nrow=1), widths=c(1,2))
 plot(las.df$SP,las.df$DEPT,axes=FALSE,xlim=c(-
 150,50),type='l',xlab='', ylab='',col='black',ylim=c(topo,base))
 axis(3, xlim=c(-150,50),col='black',lwd=1,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 plot(las.df$GR,las.df$DEPT, axes=FALSE, xlim=c(0,200),
 type='l',xlab='', ylab='',col='red',lty=2, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(0,200),col='red',lwd=1,line=1.8,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 plot(las.df$CALD,las.df$DEPT, axes=FALSE, xlim=c(5,12),
 type='l',xlab='', ylab='',col='blue',lty=3, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(5,12),col='blue',lwd=1,line=3.5,cex.lab=1,cex.axis=0.5)
 axis(2,pretty(range(las.df$DEPT),500),cex.lab=1,tck=1,cex.axis=0.5,
 col='gray')
 mtext('depth',side=2,col="black",line=2)
 legend(x=5,y=base,legend=c('SP','GR','CAL-D'),lty=c(1,2,3),
 col=c('black','red','blue'), cex=0.5)
```

```
 plot(las.df$NPLS,las.df$DEPT,axes=FALSE,xlim=c(30,-
 10),type='l',xlab='', ylab='',col='blue',ylim=c(topo,base))
 axis(3, xlim=c(30,-10),col='blue',lwd=1,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 plot(las.df$RHOB,las.df$DEPT, axes=FALSE, xlim=c(2,3), type='l',
 xlab='', ylab='' ,col='red',lty=2, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(2.3),col='red',lwd=1,line=1.8,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 par(new=TRUE)
 plot(las.df$PE,las.df$DEPT, axes=FALSE, xlim=c(0,20), type='l',xlab='',
 ylab='',col='black',lty=3, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(0,20),col='black',lwd=1,line=3.5,cex.lab=1,cex.axis=0.5)
 axis(2,pretty(range(las.df$DEPT),500),cex.lab=1,tck=1,cex.axis=0.5,
 col='gray')
 legend(x=15,y=base,legend=c('Neu','Den','PE'),lty=c(1,2,3),col=c(       'blue',
'red','black'),cex=0.5)
}

graf2<-function(base,topo){ #curvas SP, GR e CALD associadas com ILD, ILM e SFL
 par()
 dev.off()
 par(mar=c(1,2,5.5,2) + 0.1)
 layout(matrix(c(1,2),nrow=1), widths=c(1,2))
 plot(las.df$SP,las.df$DEPT,axes=FALSE,xlim=c(-
 150,50),type='l',xlab='', ylab='',col='black',ylim=c(topo,base))
 axis(3, xlim=c(-150,50),col='black',lwd=1,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 plot(las.df$GR,las.df$DEPT, axes=FALSE, xlim=c(0,150),
 type='l',xlab='', ylab='',col='red',lty=2, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(0,150),col='red',lwd=1,line=1.6,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 plot(las.df$GR,las.df$DEPT, axes=FALSE, xlim=c(150,300),
 type='l',xlab='', ylab='',col='red',lty=2, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(0,150),col='red',lwd=1,line=1.9,cex.lab=1,cex.axis=0.5)
 par(new=TRUE)
 plot(las.df$CALD,las.df$DEPT, axes=FALSE, xlim=c(6,16),
 type='l',xlab='', ylab='',col='blue',lty=3, ylim=c(topo,base),lwd=1)
 axis(3, xlim=c(6,16),col='blue',lwd=1,line=3.6,cex.lab=1,cex.axis=0.5)
 axis(2,pretty(range(las.df$DEPT),500),cex.lab=1,tck=1,cex.axis=0.5,
 col='gray')
 mtext('depth',side=2,col="black",line=2)
 legend(x=12,y=base,legend=c('SP','GR','CAL-D'),lty=c(1,2,3),
 col=c('black', 'red','blue'),cex=0.5)
 plot(las.df$ILD,las.df$DEPT,axes=FALSE,xlim=c(10^-
 1,10^3),type='l',xlab='', ylab='',col='blue',ylim=c(topo,base),log='x')
 at.y <- outer(1:9, (.5*10^(-1:3)))
 lab.y <- ifelse(log10(at.y) %% 1 == 0, at.y*2, NA)
 axis(3, xlim=c(10^-1,10^3),col='blue',lwd=1,cex.lab=1,cex.axis=0.5,
 at=at.y, labels=lab.y, las=1)
 par(new=TRUE)
 plot(las.df$ILM,las.df$DEPT, axes=FALSE, xlim=c(10^-1,10^3),
 type='l',xlab='', ylab='',col='green',lty=2,
 ylim=c(topo,base),lwd=1,log='x')
 axis(3, xlim=c(10^-
 1,10^3),col='green',lwd=1,line=1.9,cex.lab=1,cex.axis=0.5, at=at.y,
 labels=lab.y, las=1)
 par(new=TRUE)
 plot(las.df$SFL,las.df$DEPT, axes=FALSE, xlim=c(10^-1,10^3),
 type='l',xlab='', ylab='',col='black',lty=3, ylim=c(topo,base),
 lwd=1,log='x')
 axis(3, xlim=c(10^-1,10^3),col='black',lwd=1,line=3.6,cex.lab=1,
 cex.axis=0.5, at=at.y, labels=lab.y, las=1)
 axis(2,pretty(range(las.df$DEPT),500),cex.lab=1,tck=1,cex.axis=0.5,
 col='gray')
 legend(x=5,y=base,legend=c('ILD','ILM','SFL'),lty=c(1,2,3),col=c('blue',
'green','black'),cex=0.5)
}
```

```
graf1(2000,3000)
```



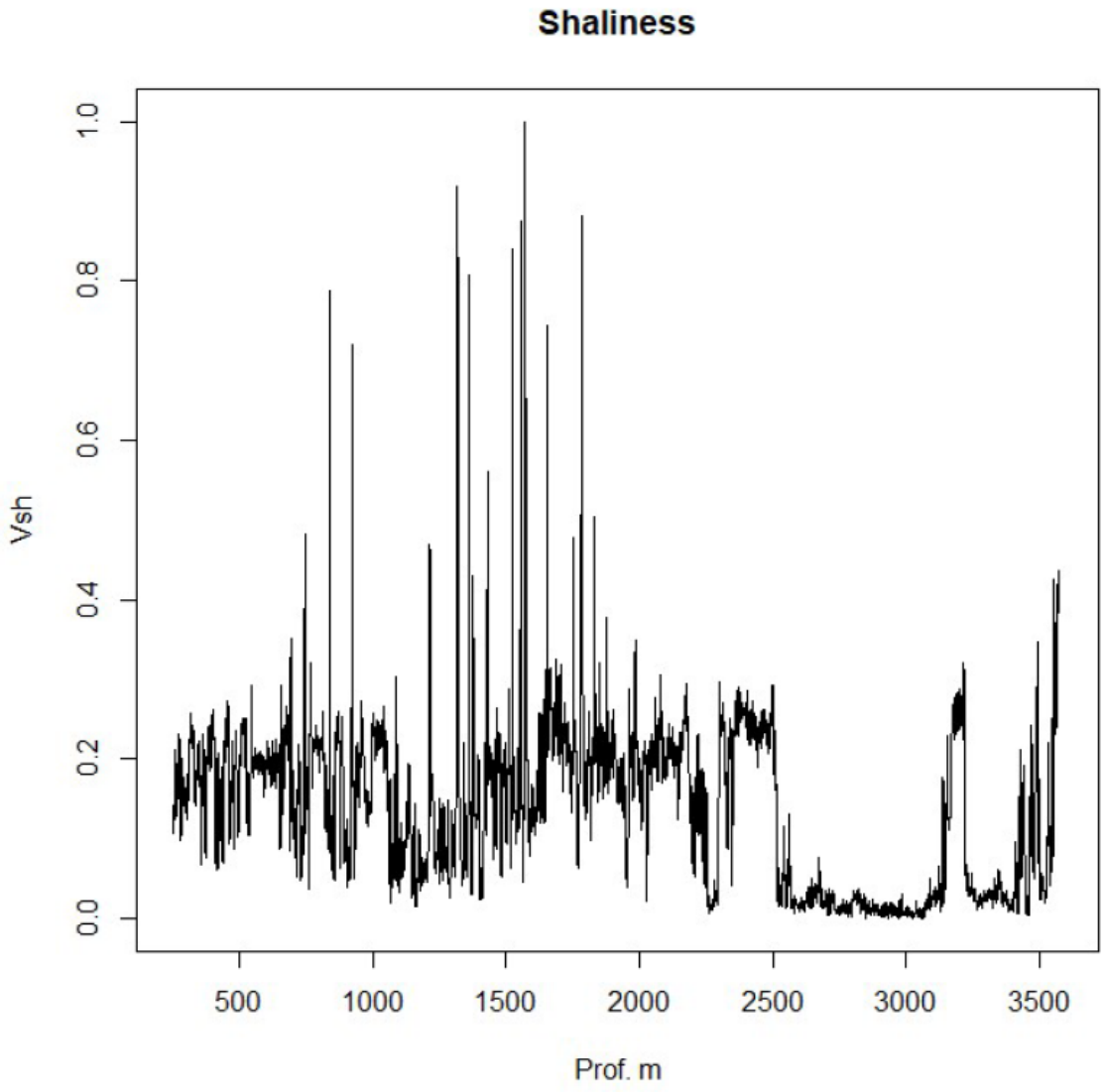
```
graf2(2000,3000)
```



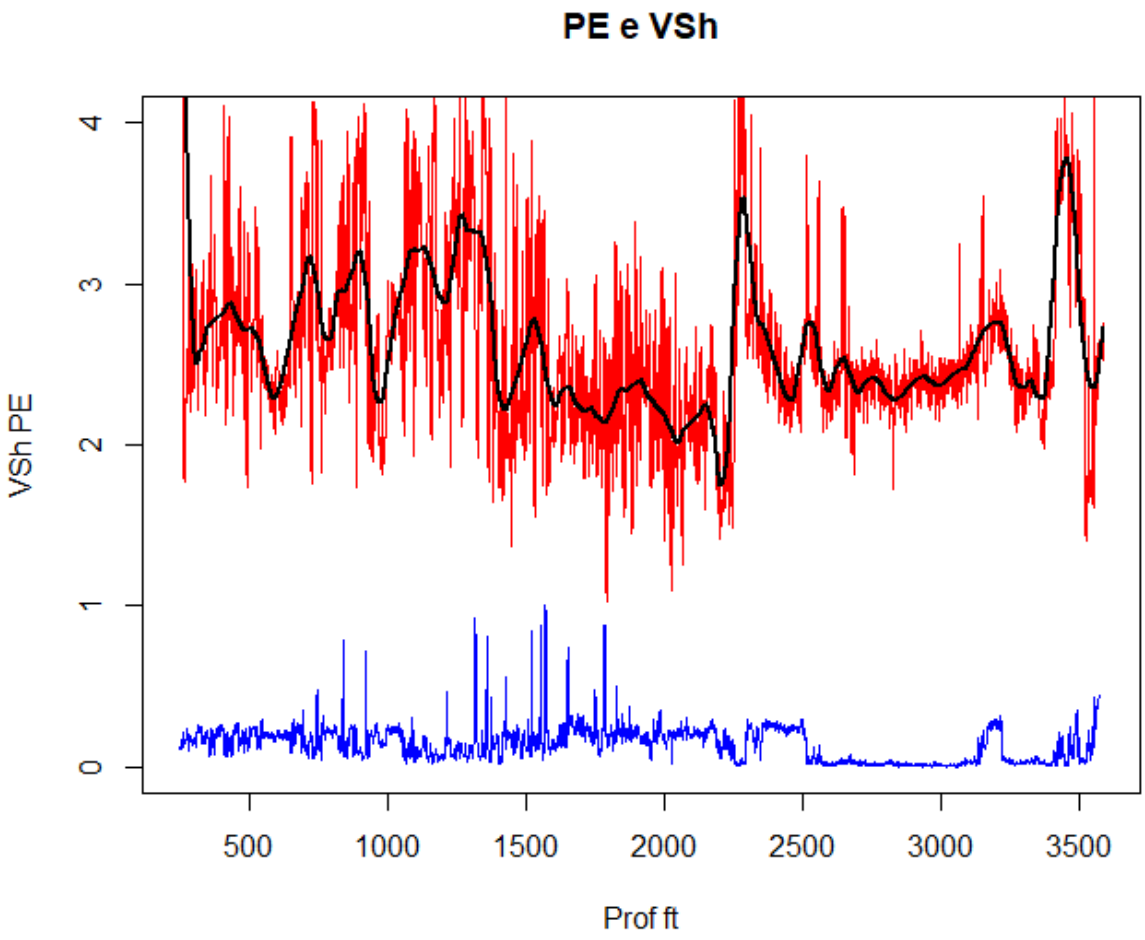
7.2.3 Análise Petrofísica

Executando um processo de análise petrofísica com base nos dados carregados acima. A intenção é mostrar como pode ser feito e muito foi assumido durante o processo. A intenção é só ilustrar como pode ser feita usando somente R (e o arquivo original, claro).

```
# SELECIONANDO AS CURVAS A SEREM USADAS
petro.df<-las.df[,c(1,5,6,7,12,18,19)]
names(petro.df)<-c('depth','GR','ILD','ILM','NPLS','RHOB','PE')

#  CALCULANDO O SHALINESS VSh
GRmax<-max(petro.df$GR,na.rm=TRUE)
GRmin<-min(petro.df$GR,na.rm=TRUE)
petro.df$VSH<-(petro.df$GR-GRmin)/(GRmax-GRmin)
plot(petro.df$depth,petro.df$VSH,type='l',main='Shaliness',xlab='Prof.m',
ylab='Vsh')
```



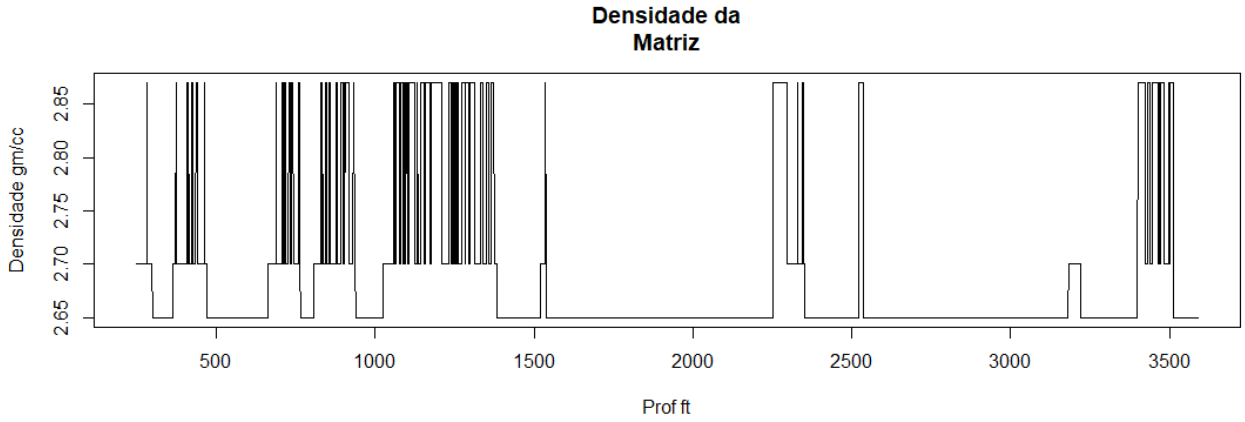

```
# CALCULANDO A POROSIDADE
dev.off()
lowpass.PE <- loess(PE~depth,data=petro.df, span = 0.05)
PE.lp<-predict(lowpass.PE, petro.df$depth)
plot(petro.df$depth,petro.df$PE,type='l',col='red',main='PE e VSh',
xlab='Prof ft', ylab='VSh PE',ylim=c(0,4))
lines(petro.df$depth,PE.lp,type='l',lwd=2)
lines(petro.df$depth,petro.df$VSH,type='l',lwd=.5,col='blue')
petro.df<-cbind(petro.df,PE.lp)
```



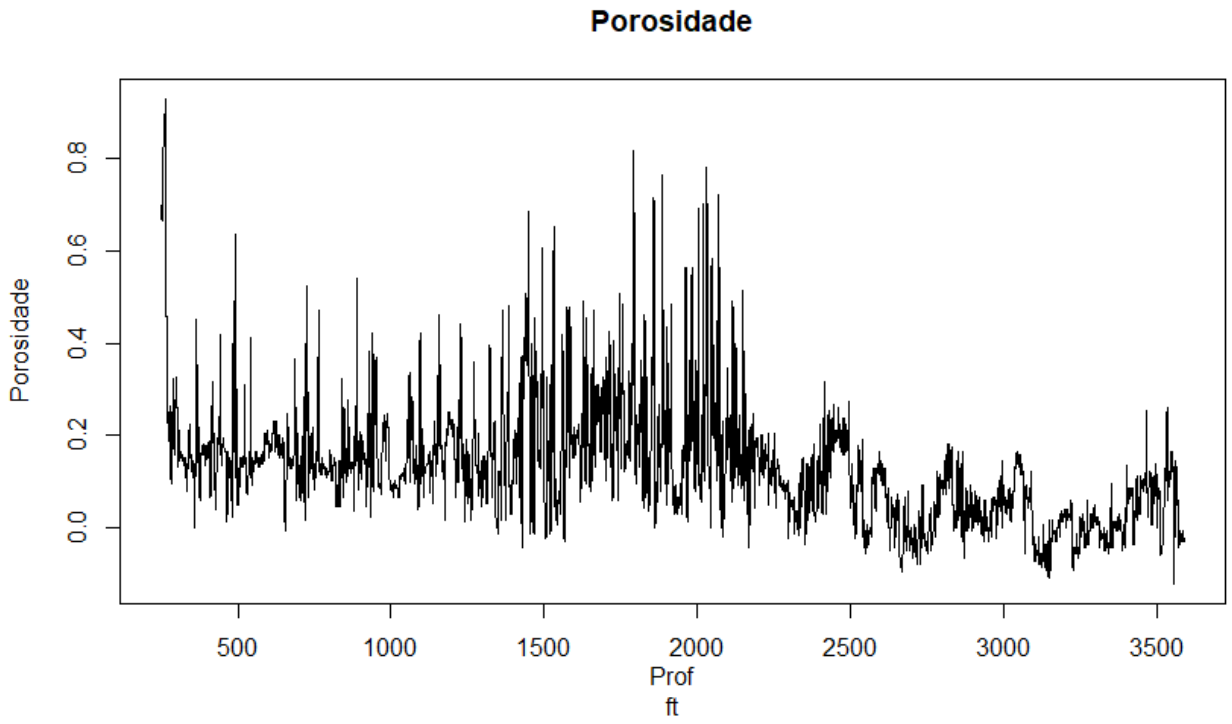
Usaremos a porosidade da matriz de 2.87 gm/cc para o dolomito, 2.7 gm/cc para o folhelho e 2.65 gm/cc para o arenito. Pe < 2.5 será interpretado como arenito, Pe > 2.5 será folhelho quando o Vsh > 0.1 caso contrário será interpretado como dolomito.

Abaixo ajustaremos a densidade da matriz como o descrito acima. De uma forma geral executamos um processo de identificação litológica do poço como bônus nesse processo.

```
petro.df$ro.matrix <- ifelse(petro.df$PE.lp >=2.75, ifelse(petro.df$VSH
<0.1,2.87,2.7), 2.65)
plot(petro.df$depth,petro.df$ro.matrix,type='l',main='Densidade da
Matriz',xlab='Prof ft',ylab='Densidade gm/cc')
petro.df$phi<-(petro.df$ro.matrix-petro.df$RHOB)/(petro.df$ro.matrix-1)
```



```
plot(petro.df$depth,petro.df$phi,type='l',main='Porosidade',xlab='Prof
ft', ylab='Porosidade')
```



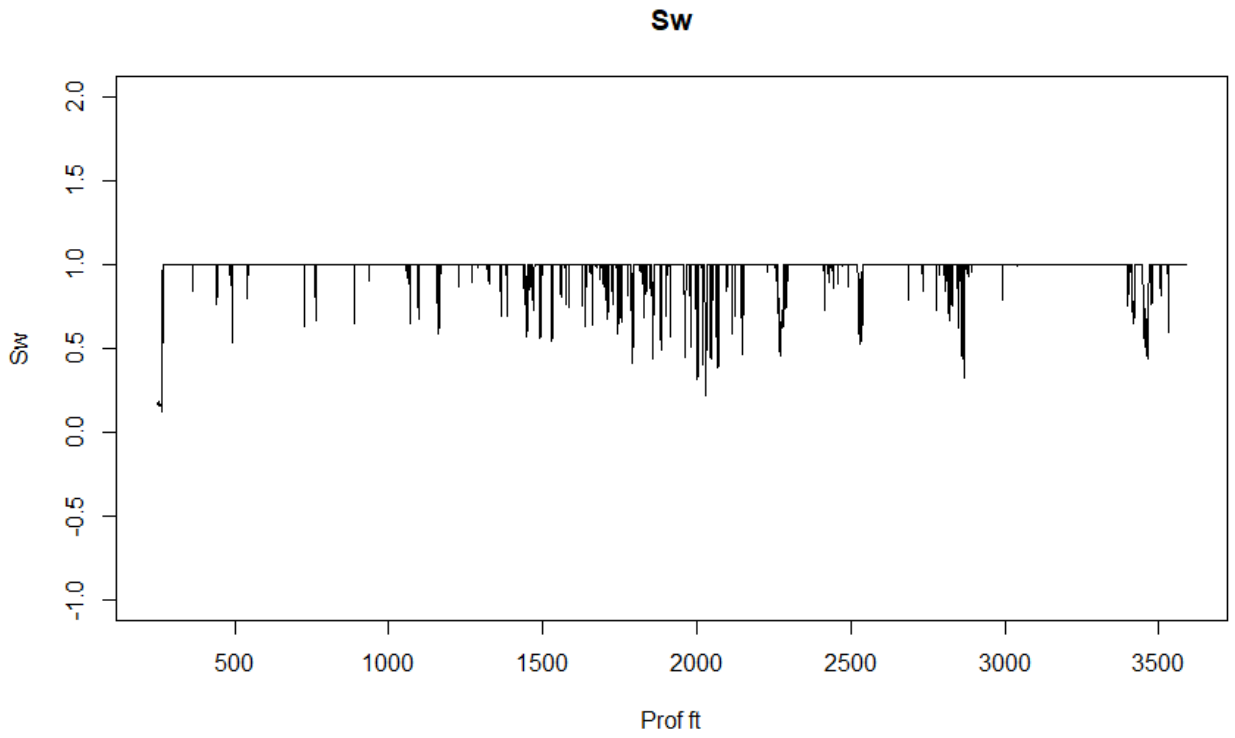
O primeiro passo é calcular o Rw a partir de uma "zona de água" (water zone) selecionada. Novamente, este é um exemplo didático e estamos aplicando para o poço inteiro. Geralmente este processo é aplicado para intervalos de furo selecionados. Então, vamos usar um valor único de Rw que pode não ser apropriado para o poço inteiro pois a resistividade da zona da água tende a mudar com a profundidade.

Selecione um intervalo onde a resistividade é baixa e que seja bem poroso (arenito) para obter o valor de Rw. A 3042 pés de profundidade temos essa condição. Então calculamos o Rw usando:

```
# CALCULANDO O SW
petro.df[petro.df$depth == 3042,]
```

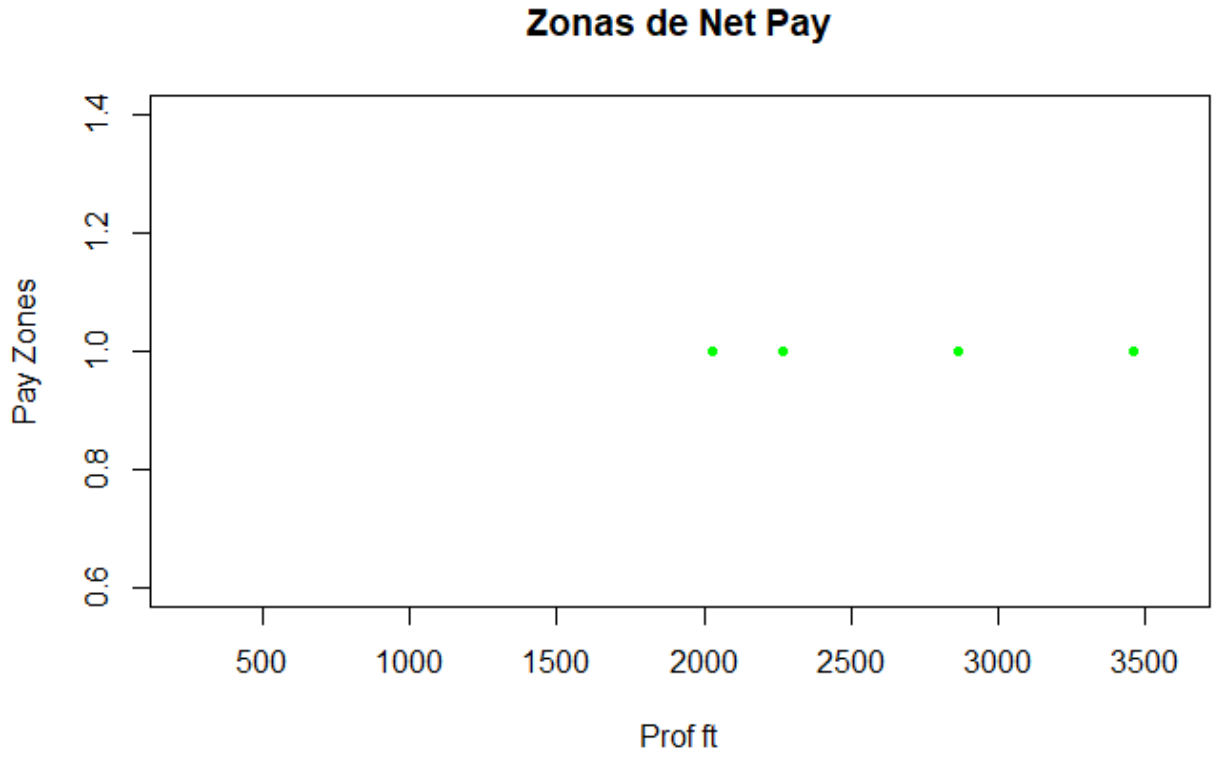
```
     depth    GR   ILD   ILM  NPLS RHOB   PE        VSH    PE.lp ro.matrix       phi
5696  3042 14.91 17.93 19.78 25.97 2.38 2.35 0.01037538 2.428599      2.65 0.1636364
```

```
Rt<-petro.df[petro.df$depth == 3042,]$ILD
phiT<-petro.df[petro.df$depth == 3042,]$phi
Rw<-phiT*phiT*Rt
Rw
[1] 0.4801091

petro.df$Sw<-1/petro.df$phi*sqrt(Rw/petro.df$ILD)
index<- petro.df$Sw > 1
petro.df$Sw[index]<- 1
index<- petro.df$Sw < 0
petro.df$Sw[index]<- 1
plot(petro.df$depth,petro.df$Sw,type='l',main='Sw',xlab='Prof ft',
ylab='Sw', ylim=c(-1,2))
```

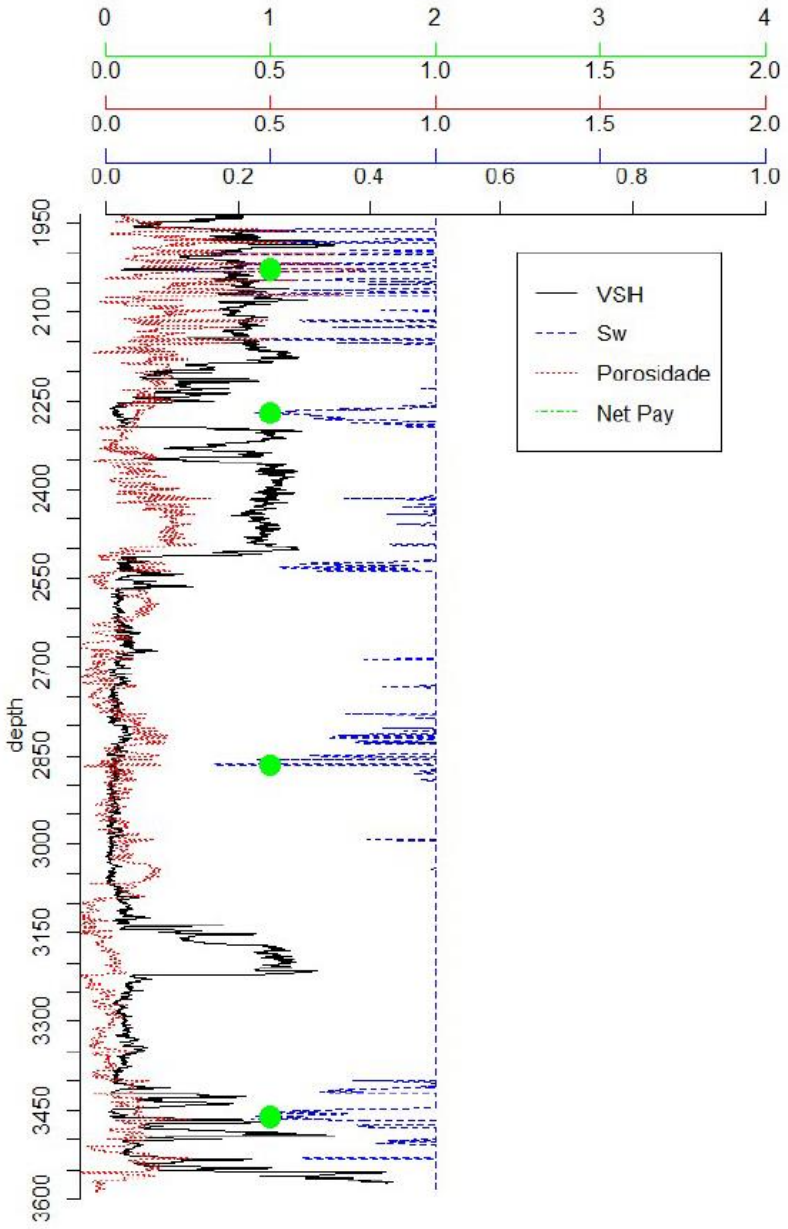


Net Reservoir é a zona onde o VSh é baixo e a porosidade é alta. O Net Pay é a zona dentro do Net Reservoir onde a saturação de água e pequena. Usaremos os valores de corte de 0.05 para o Vsh, de 0.1 para a Porosidade e 0.5 para a Sw para definirmos nosso Net Pay.

```
# CALCULANDO O NETPAY
petro.df$net.pay<-ifelse(petro.df$phi>0.1,ifelse(petro.df$VSH<0.05,
ifelse(petro.df$Sw<0.5,1,NA), NA),NA)
plot(petro.df$depth,petro.df$net.pay,type='l',main='Zonas de Net Pay',
xlab='Prof ft',ylab='Pay Zones',lwd=5,col='green')
```



```
# PLOTANDO TUDO JUNTO
dev.off()
par(mar=c(1, 4, 8, 4) + 0.1)
plot(petro.df$VSH,petro.df$depth,axes=FALSE,xlim=c(0,1),type='l',xlab=
'', ylab='',col='black',ylim=c(3600,2000))
axis(3, xlim=c(0,1),col='black',lwd=1)
par(new=TRUE)
plot(petro.df$Sw,petro.df$depth, axes=FALSE, xlim=c(0,2),
type='l',xlab='', ylab='',col='blue',lty=2, ylim=c(3600,2000),lwd=1)
axis(3, xlim=c(0,2),col='blue',lwd=1,line=2)
par(new=TRUE)
plot(petro.df$phi,petro.df$depth, axes=FALSE, xlim=c(0,2),
type='l',xlab='', ylab='',col='red',lty=3, ylim=c(3600,2000),lwd=1)
axis(3, xlim=c(0,2),col='red',lwd=1,line=4)
par(new=TRUE)
plot(petro.df$net.pay,petro.df$depth, axes=FALSE, xlim=c(0,4),
type='l',xlab='', ylab='',col='green',lty=4,ylim=c(3600,2000),lwd=15)
axis(3, xlim=c(0,4),col='green',lwd=1,line=6)
axis(2,pretty(range(petro.df$depth),100))
mtext('depth',side=2,col="black",line=2)
legend(x=2.5,y=2000,legend=c('VSH','Sw','Porosidade','Net
Pay'),lty=c(1,2,3,4),col=c('black','blue','red','green'))
```



Receita - 7.3 – Dados no Formato SEG-Y

Ler e interagir com informações oriundas de um arquivo de dados sísmico e GPR. Este formato é um pouquinho mais complexo e apresenta muitas variações de formato. O exemplo é ilustrativo e funciona com boa parte dos arquivos seg-Y.

Arquivos seg-Y são divididos em três partes principais (genericamente falando); Cabeçalho geral do arquivo (3200 bytes do cabeçalho texto mais 400 bytes do cabeçalho binário), cabeçalho binário do traço (240 bytes) e dado binário do traço.

Carregando um arquivo seg-Y por etapas.

7.3.1 Lendo Cabeçalho texto do arquivo

```
hdbn<-c('JOB_ID_NO','LINE_NUMBER','REEL_NUMBER','TRACES_PER_RECORD',
'AUXS_PER_RECORD','SAMPLE_RATE','SAMPLE_RATE_FIELD','NSAMPLES','NSAMPL
ES_FIELD','FORMAT_CODE','ENSEMBLE_FOLD','TRACE_SORT','VERTICAL_SUM','S
WEEP_FREQ_START','SWEEP_FREQ_END','SWEEP_FREQ_LENGTH','SWEEP_TYPE','SW
EEP_TRACE_NO','SWEEP_TAPER_LENGTH_START','SWEEP_TAPER_LENGTH_END','SWE
EP_TAPER_TYPE','CORRELATED_TRACES','BINARY_GAIN','AMP_RECOVERY_METHOD'
,'UNITS','SIGNAL_POLARITY','VIBRATOR_POL_CODE','UNUSED','SEGY_FORMAT_R
EVISION_NO','SEGY_FIXEDLEN_FLAG','SEGY_NO_TEXTFHEADERS','UNUSED')
con <- file("1104-30A.segy","rb")
library(float) #install float using install.packages("float")
cabe.texto<-c('bogus','data')
for(i in 1:40){
dec<-readBin(con,"raw",size=1,n=80)
cabe.texto[i]<-
paste0(iconv(rawToChar(dec,multiple=T),'cp500','utf8'), collapse='')
}
cabe.texto
```

```
[1] "C 1 CLIENT MMS                    COMPANY GECO                    CREW NO         "
 [2] "C 2 LINE 1104-30A   AREA PHASE-30A              MAP ID                           "
 [3] "C 3 REEL NO 17784MIG0 DAY-START OF REEL     YEAR      OBSERVER                   "
 [4] "C 4 INSTRUMENT: MFG            MODEL            SERIAL NO                       "
 [5] "C 5 DATA TRACES/RECORD        AUXILIARY TRACES/RECORD        CDP FOLD          "
 [6] "C 6 SAMPLE INTERVAL       4 SAMPLES/TRACE  2750 BITS/IN    0 BYTES/SAMPL       0"
 [7] "C 7 RECORDING FORMAT        FORMAT THIS REEL        MEASUREMENT SYSTEM         "
 [8] "C 8 SAMPLE CODE: FLOATING PT 032 FIXED PT     FIXED PT-GAIN     CORRELAT       "
 [9] "C 9 GAIN  TYPE: FIXED     BINARY      FLOATING POINT     OTHER                  "
[10] "C10 FILTERS: ALIAS     HZ  NOTCH     HZ  BAND     -     HZ  SLOPE    -         "
[11] "C11 SOURCE: TYPE AIRGUN    NUMBER/POINT          POINT INTERVAL                  "
[12] "C12     PATTERN:                          LENGTH        WIDTH                   "
[13] "C13 SWEEP: START     HZ  END     HZ  LENGTH      MS  CHANNEL NO     TYPE       "
[14] "C14 TAPER: START LENGTH       MS  END LENGTH       MS  TYPE                    "
[15] "C15 SPREAD: OFFSET        MAX DISTANCE         GROUP INTERVAL                  "
[16] "C16 GEOPHONES: PER GROUP     SPACING     FREQUENCY     MFG          MODE        "
[17] "C17     PATTERN:                          LENGTH        WIDTH                   "
[18] "C18 TRACES SORTED BY: RECORD     CDP     OTHER                                  "
[19] "C19 AMPLITUDE RECOVERY: NONE      SPHERICAL DIV       AGC    OTHER               "
[20] "C20 PROJECTION : SPHEROID :                  "
[21] "C21 FAMILY 1 DEFAULTS: DATA TRACES/RECORD        AUXILIARY TRACES/RECORD        "
[22] "C22                    SAMPLE INTERVAL          SAMPLES/TRACE                  "
[23] "C23 FAMILY 3 DEFAULTS: SAMPLE INTERVAL          SAMPLES/TRACE                  "
[24] "C24 CENTRAL MERIDIAN :  0  0 0.0 E    ORIGIN PARALLEL :  0  0 0.0 N             "
[25] "C25 FALSE NORTHING :      0.0 FALSE EASTING :      0.0 UNIT TO METER :0.000000  "
[26] ""
[27] "C27 MXYMRGOV 15.06.22 12- 8-88 A6602IM4 0526     1104-30                        "
[28] "C28                                                                             "
[29] "C29                                                                             "
[30] "C30                                                                             "
[31] "C31                                                                             "
[32] "C32                                                                             "
[33] "C33                                                                             "
[34] "C34                                                                             "
[35] "C35                                                                             "
[36] "C36                                                                             "
[37] "C37                                                                             "
[38] "C38                                                                             "
[39] "C39                                                                             "
[40] "C40 END EBCDIC                                                                  "
```

7.3.2 Lendo Cabeçario binário do arquivo

```
seek(con,3200) #posicionando arquivo no byte 3201 (começa com 0)
cabe.bin<-c(1.0,1.7)
passo<-1
for (i in passo:(passo+2)){
cabe.bin[passo]<-readBin(con,'int',size=4,endian='big')
passo<-passo+1
}
for (i in passo:(passo+23)){
cabe.bin[passo]<-readBin(con,'int',size=2,endian='big')
passo<-passo+1
}
cabe.bin[passo]<-0
seek(con,3500)
passo<-passo+1
for (i in passo:(passo+2)){
cabe.bin[passo]<-readBin(con,'int',size=2,endian='big')
passo<-passo+1
}
cabe.bin[passo]<-0
seek(con,3600)
cabecalho.bin<-data.frame(campo=hdbn,valor=cabe.bin)
cabecalho.bin
```

	campo	valor
1	JOB_ID_NO	526
2	LINE_NUMBER	1104
3	REEL_NUMBER	30
4	TRACES_PER_RECORD	1
5	AUXS_PER_RECORD	0
6	SAMPLE_RATE	4000
7	SAMPLE_RATE_FIELD	0
8	NSAMPLES	2751
9	NSAMPL\nES_FIELD	2751
10	FORMAT_CODE	1
11	ENSEMBLE_FOLD	1
12	TRACE_SORT	4
13	VERTICAL_SUM	2
14	S\nWEEP_FREQ_START	0
15	SWEEP_FREQ_END	0
16	SWEEP_FREQ_LENGTH	0
17	SWEEP_TYPE	0
18	SW\nEEP_TRACE_NO	0
19	SWEEP_TAPER_LENGTH_START	0
20	SWEEP_TAPER_LENGTH_END	0
21	SWE\nEP_TAPER_TYPE	0
22	CORRELATED_TRACES	0
23	BINARY_GAIN	0
24	AMP_RECOVERY_METHOD	0
25	UNITS	1
26	SIGNAL_POLARITY	0
27	VIBRATOR_POL_CODE	0
28	UNUSED	0
29	SEGY_FORMAT_R\nEVISION_NO	0
30	SEGY_FIXEDLEN_FLAG	0
31	SEGY_NO_TEXTFHEADERS	0
32	UNUSED	0

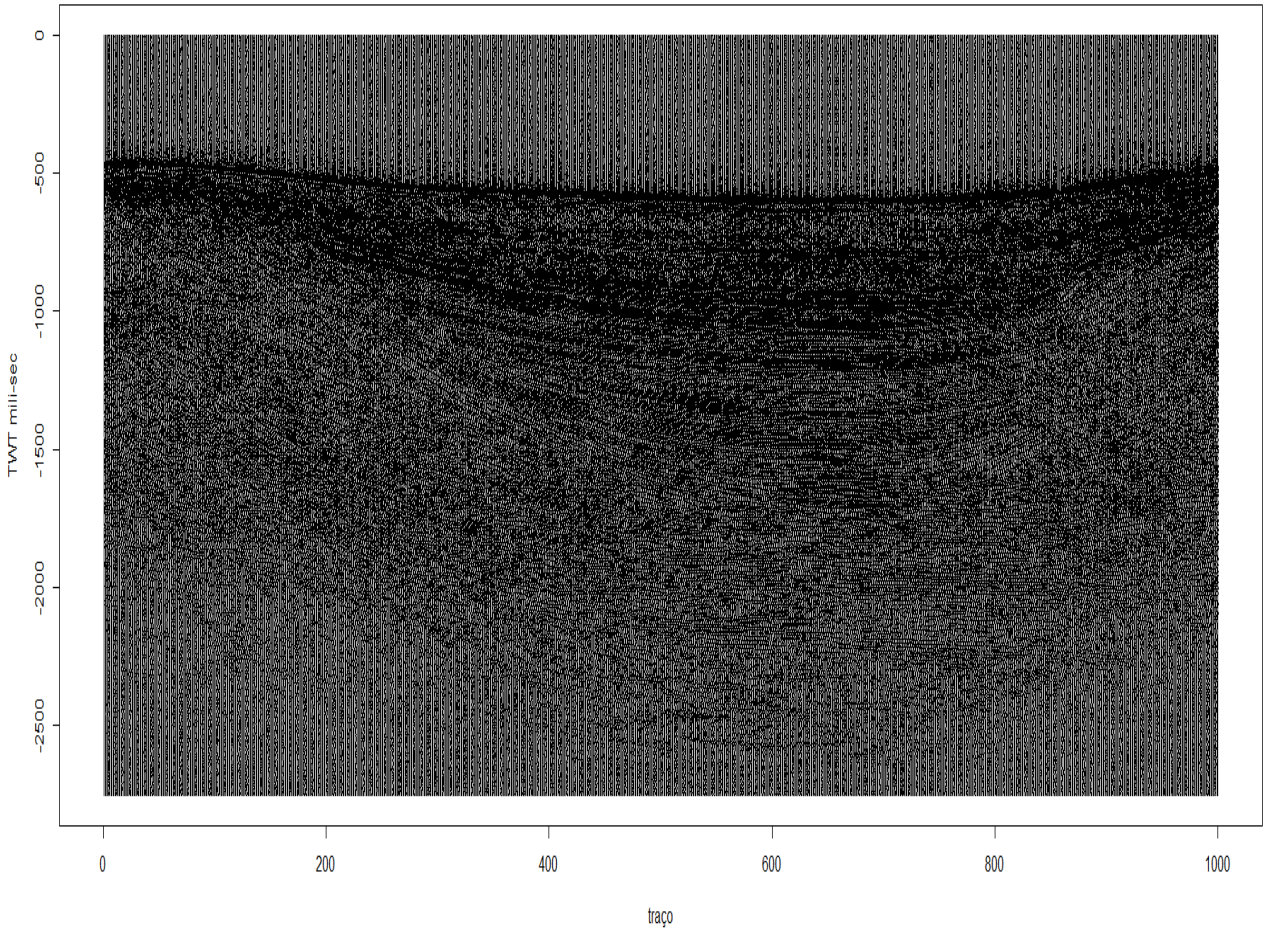
7.3.3 Criando o dataframe cabeçalho do traço e o vetor traço

```
tam.arq<-file.info('1104-30A.segy')$size
tam.arq
[1] 50995140
rep.cabecalho<-cabecalho.bin[31,2]
rep.cabecalho
numero.samples<-cabecalho.bin[8,2]
numero.samples
[1] 2751
num.tr<-(tam.arq-3600-3600*rep.cabecalho)/(240+numero.samples*4)
num.tr
[1] 4535
conta<-c(7,4,8,2,4,46,5,2,3,4,6,1,2)
byte.me<-c(4,2,4,2,4,2,4,2,2,2,2,2,4)
tr.bin<-c(1,2)
l.cabe.tr.bin<-list(data.frame(campo='oi',valor=1))
l.tr<-list(c(1,2,3))
traco<-c(1,2,3)
```

```
for (i in 1:num.tr){
tbc<-0
for (a in 1:13){
for (c in 1:conta[a]){
tr.bin[tbc]<-readBin(con,'int',size=byte.me[a],endian='big')
tbc<-tbc+1
}
}
frame<-data.frame(valor=tr.bin)
l.cabe.tr.bin[[i]]<-frame
for(s in 1:numero.samples){
traco[s]<-readBin(con,'double',size=4,endian='big')
}
l.tr[[i]]<- as.single (traco)
}
close(con)
segY<-list(cabe.texto,cabecalho.bin,l.cabe.tr.bin,l.tr)
```

7.3.4 Plotando Arquivo

```
plot(((segY[[4]][[1]])/40)+1,c(-1:-numero.samples),type='l',
xlim=c(1,1000), xlab='traço', ylab='TWT mili-sec',lwd=0.02)
for(i in 2:1000){
lines(((segY[[4]][[i]])/40)+i,c(-1:-numero.samples),type='l',lwd=0.02)
}
```



Receita - 7.4 – Dados no Formato OASIS-MONTAJ XYZ

Dados de geofísica Oasis Montaj normalmente estão nos seguintes formatos:

```
/ ------------------------------------------------------------------------------
/ XYZ EXPORT [05/11/10]
/ DATABASE [F:\1100\DVD_01\GDB\1100_MAGLINE.gdb : SUPER]
/ ------------------------------------------------------------------------------
/
/ X Y FIDUCIAL GPSALT BARO ALTURA MDT MAGBRU MAGCOM MAGCOR...
/========== ========== ========== ========== ========== ========== ==========
=======...
/
Line 10010
170583.35 8895511.73 26618.0 423.09 426.67 109.01 322.03 24287.7188 24286.1147
24272.3286...
```

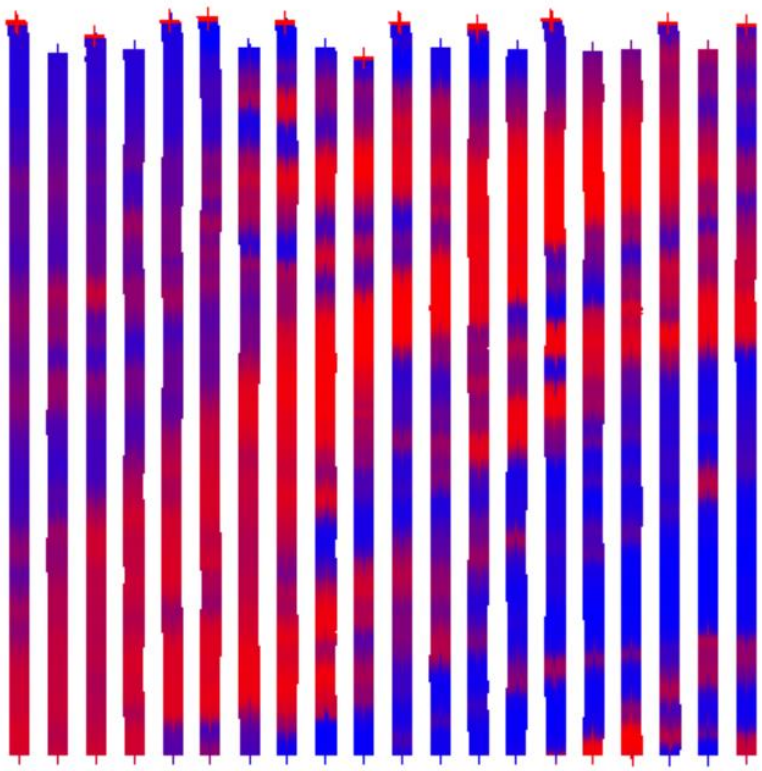
Ou

```
/ ------------------------------------------------------------------------------
/ XYZ EXPORT [05/11/10]
/ DATABASE [F:\1100\DVD_01\GDB\1100_GAMALINE.gdb : SUPER]
/ ------------------------------------------------------------------------------
/
/ X Y FIDUCIAL GPSALT BARO ALTURA MDT CTB KB UB ....
/========== ========== ========== ========== ========== ========== ==========...
/
Line 10010
170583.28 8895543.28 1752.0 422.61 426.67 107.69 314.92 1644 151 52 ...
170582.68 8895622.11 1753.0 422.09 424.89 106.13 315.96 1616 132 55 ...
170581.96 8895701.04 1754.0 422.19 425.78 106.93 315.26 1655 132 56 ...
```

Criaremos um dataframe com os dados do aquivo XYZ. Se o arquivo que você tem é do tipo GDB este pode ser facilmente exportado/gravado no formato XYZ usando o Oasis Montaj Viewer.

```
xmi<-180000
xma<-190000
ymi<-8940684
yma<-8950684
flcon<-file('1099_MAGLINE.XYZ',open='r')
# lendo as primeiras linhas
tmp <- readLines(flcon, n=5)
# obtendo o nome das colunas
coluna <-c(strsplit(substring(readLines(flcon, n=1),11),"\\s+")[[1]],'line/tie')
n.co<-length(coluna)+1
# lendo as linhas adicionais abaixo da linha com nome das colunas
tmp <- readLines(flcon, n=2)
lin<-c('a')
o<-1 #define o índice
lineOrTie<-'vazio'
while (length(line <- readLines(flcon, n = 1, warn = FALSE)) > 0) {
  if(substring(line,2,3)=='ie' || substring(line,2,3)=='in'){
     if(substring(line,2,3)=='ie')lineOrTie<-'Tie'
     if(substring(line,2,3)=='in')lineOrTie<-'Line'
    }else{
     v<-as.numeric(strsplit(line, "\\s+")[[1]][2])
     w<-as.numeric(strsplit(line, "\\s+")[[1]][3])
     if(v>xmi & v<xma & w>ymi & w<yma){
      lin[o]<-paste(line,lineOrTie)
      o<-o+1
     }
   }
}
close(flcon)
r <- strsplit(sub("^\\s+","",lin), "\\s+")
s <-as.data.frame(do.call(rbind, r))
names(s)<- coluna
cols = c(1:(ncol(s)-3))
s[,cols] = apply(s[,cols], 2, function(x) as.numeric(as.character(x)))
```

```
# convertendo em SpatVector e plotando
library(terra) #install terra using install.packages("terra")
sgeo<-vect(s, geom=c("X", "Y"), crs="epsg:32721")
library(raster) #install raster using install.packages("raster")
mag<-as(sgeo, "Spatial")
plot(mag,col=colorRampPalette(c('blue','red'))(length(mag$MAGIGRF))[rank(mag$MA
GIGRF)])
```



Receita - 7.5 – Dados no Formato UBC

Muitos dados de inversão geofísica são armazenados em formato UBC mesh-model 3D que normalmente estão nos seguintes formatos:

Arquivo .msh:

NE	NN	NZ	
E_o	N_o	Z_o	
ΔE_1	ΔE_2	$\cdots$	ΔE_{NE}
ΔN_1	ΔN_2	$\cdots$	ΔN_{NN}
ΔZ_1	ΔZ_2	$\cdots$	ΔZ_{NZ}

Onde:
- NE NN NZ representa o número de pontos em cada direção
- Eo No Zo são as coordenadas na origem
- ΔE(1 a N) é o espaçamento entre os dados direção E de W para E
- ΔN(1 a N) é o espaçamento entre os dados direção N de S para N
- ΔZ(1 a N) é o espaçamento entre os dados direção Z do Topo para Base

Arquivo .mod:

$$
\begin{array}{l}
m_{1,1,1} \\
m_{1,1,2} \\
\vdots \\
m_{1,1,NZ} \\
m_{1,2,1} \\
\vdots \\
m_{1,j,k} \\
\vdots \\
m_{NN,NE,NZ}
\end{array}
$$

Onde:
- Cada {i,j,k} é a propriedade medida na [i,j,k]^{éssima} célula do modelo.
- A célula [i, j, k]=[1, 1, 1] é por definição no topo sudoeste do modelo.
- O número total de linhas do arquivo deve ser igual NN x NE x NZ.
- A ordem no arquivo vem primeiro na direção Z (topo para base) depois na direção leste (de W para E) e por último na direção norte (de S para N).

Carregando os dados nos arquivos mesh e model:

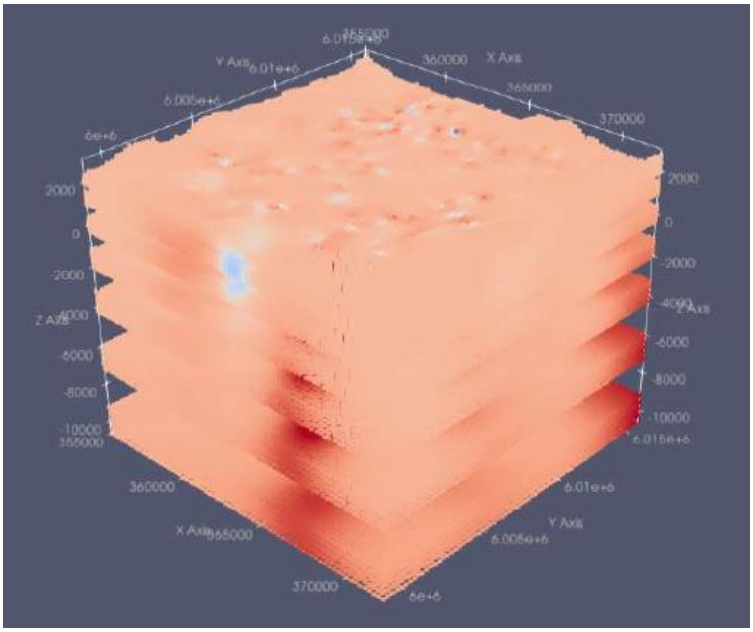
```
file<-file('arquivo_mesh.msh',open='rt')
GV<-list()
GV[[1]]<-scan(file,what=1,nlines=1,sep=" ")
GV[[2]]<-scan(file,what=1,nlines=1,sep=" ")
GV[[3]]<-scan(file,what=1,nlines=1,sep=" ")
GV[[4]]<-scan(file,what=1,nlines=1,sep=" ")
GV[[5]]<-scan(file,what=1,nlines=1,sep=" ")
close(file)
```

```
GV[[6]]<-data.frame(z=NA,x=NA,y=NA,v=read.table('arquivo_model.mod')$V1)
names(GV)<-c('dimensão','origem','deltaX','deltaY','deltaZ','valores')
GV$deltaX<-na.omit(GV$deltaX)
GV$deltaY<-na.omit(GV$deltaY)
GV$deltaZ<-na.omit(GV$deltaZ)
summary(GV)
          Length Class      Mode
dimensão   3     -none-     numeric
origem     3     -none-     numeric
deltaX    69     -none-     numeric
deltaY    69     -none-     numeric
deltaZ    40     -none-     numeric
valores    4     data.frame list
```

Carregando os dados das coordenadas no dataframe ‘valores’ e exportando como ubc.csv:

```
coordz<-GV$origem[3]+GV$deltaZ-GV$deltaZ*seq(1:GV$dimensão[3])
GV$valores[1]<-coordz
coordx<-rep(GV$origem[1],GV$dimensão[1])
coordxx<-coordx-GV$deltaX+GV$deltaX*seq(1:GV$dimensão[1])
coordxxx<-rep(coordxx, each=GV$dimensão[3])
GV$valores[2]<-coordxxx
coordy<-rep(GV$origem[2],GV$dimensão[2])
coordyy<-coordy-GV$deltaY+GV$deltaY*seq(1:GV$dimensão[2])
coordyyy<-rep(coordyy, each=GV$dimensão[1]*GV$dimensão[3])
GV$valores[3]<-coordyyy
GV$valores<-na.omit(GV$valores)
write.csv(GV$valores,'ubc.csv',row.names=F)
```

Abrindo o arquivo ubc.csv em Paraview veremos:



Receita - 7.6 – De Dados de Sondagem para Banco de Dados

O conjunto de dados Marvin está carregado em um servidor PostgreSQL. Para acessar este banco de dados use as seguintes credenciais:

Servidor: **pg.gdatasystems.com**
Usuário: **droid**
Senha: **devcor**

Para colocar dados em um servidor PostgreSQL próprio siga as instruções seguintes:

Criando o banco de dados com o nome geofoss:

$ *createdb geofoss --encoding=utf-8*

Crie um arquivo create.sql com o seguinte conteúdo:

```
CREATE TABLE collar(
holeid      varchar(50) NOT NULL,
x           numeric(6,2) NOT NULL,
y           numeric(6,2) NOT NULL,
z           numeric(6,2) NOT NULL,
enddepth    numeric(6,2)NOT NULL,
UNIQUE(holeid));

CREATE TABLE survey(
holeid      varchar(50) NOT NULL,
_at         numeric(4,1) NOT NULL,
azm         numeric(5,2) NOT NULL,
dip         numeric(4,2)NOT NULL,
UNIQUE(holeid,_at));

CREATE TABLE assay(
holeid      varchar(50) NOT NULL,
_from       numeric(4,1) NOT NULL,
_to         numeric(4,1) NOT NULL,
au          numeric(4,2));

CREATE TABLE litho(
holeid      varchar(50) NOT NULL,
_from       numeric(4,1) NOT NULL,
_to         numeric(4,1) NOT NULL,
domain      varchar(10),
rocktype    varchar(10),
weath       varchar(10));
```

Grave o arquivo e execute usando:
$ *psql -d geofoss -f create.sql*

Carregue o dado dos arquivo CSV collar.csv, assay.csv, survey.csv e litho.csv usando:
$ *psql geofoss*
geofoss=# \copy collar (holeid,x,y,z,enddepth) FROM 'collar.csv' DELIMITER ',' HEADER CSV
geofoss=# \copy assay FROM 'assay.csv' DELIMITER ',' HEADER CSV
geofoss=# \copy survey FROM 'survey.csv' DELIMITER ',' HEADER CSV
geofoss=# \copy litho FROM 'litho.csv' DELIMITER ',' HEADER CSV
geofoss=# GRANT SELECT ON ALL TABLES IN SCHEMA public to user;

Não esqueça de substituir user pelo usuário do banco de dados.

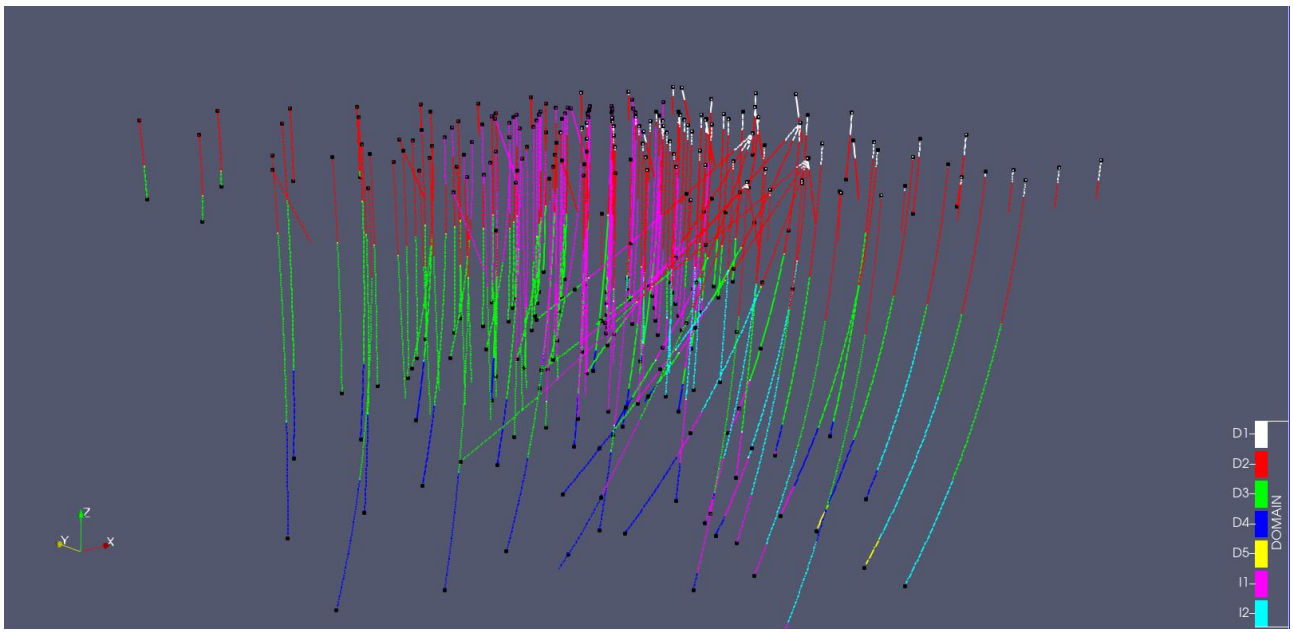
Receita - 7.7 – Desurvey de Dados de Sondagem

A partir de dados de sondagem já armazenados em um banco de dados execute o desurvey usando.

```
library(RPostgreSQL) #install RPostgreSQL using install.packages("RPostgreSQL")

#
con<-dbConnect(PostgreSQL(),host='pg.gdatasystems.com',user='droid',
password='devcor', dbname='geofoss')
#
dat <- dbGetQuery(con, "select c.holeid as hole, c.x as x, c.y as
y, c.z as z, a.au as au, a._from as prof, l._from as profl, a._to as
toa, l._to as tol, (a._to-a._from) as lena, (l._to-l._from) as lenl,
l.domain, l.rocktype, l.weath from collar as c, assay as a, litho as
l where c.holeid = a.holeid and l.holeid=c.holeid and l._from between
a._from and (a._to-0.02) order by hole,prof")
#
collar <- dbGetQuery(con, "select c.holeid as hole,s._at as prof, c.x
as x, c.y as y, c.z as z, s.dip as dip, s.azm as az from collar as c,
survey as s where c.holeid = s.holeid and s._at=0")
# if dip is positive uncoment this line------------------------------
collar$dip <- collar$dip*-1
#
station <- dbGetQuery(con, "SELECT holeid as hole, _at as prof, null
as x, null as y, null as z, dip as dip, azm FROM survey order by
holeid,prof")
#
station$dip <- station$dip*-1
#
d2r <- function(x) { x * pi/180}
#
tresD<-function(co,es){
 linha <- data.frame()
 for(i in 1:nrow(co)){
 x <- co[i,3]
 y <- co[i,4]
 z <- co[i,5]
 di <- 0
 fur <- es[es[[1]]==co$hole[[i]],]
 fur <- fur[order(fur$prof),]
 ro <- nrow(fur)
 for(j in 1:ro){
 ang <- fur[j,7]
 d <- fur[j,2]-di
 di <- d+di
 dip <- fur[j,6]
 if(j>1)dip <- fur[(j),6]-(fur[j,6]-fur[(j-1),6])/2
 deltaz <- d*sin(d2r(dip))
 r <- d*cos(d2r(dip))
 x <- round(x+r*sin(d2r(ang)))
 y <- round(y+r*cos(d2r(ang)))
 z <- round(z+deltaz)
 linha<-
rbind(linha,data.frame(hole=fur[j,1] ,prof=fur[j,2],x=x ,y=y
,z=z,dip=fur[j,6],az=fur[j,7]))
 }
 }
 lin <- linha
 lin <- lin[order(lin$hole,lin$prof),]
 return(lin)
}
#
Spts <- tresD(collar,station)
#
calculo <- function(azt,azb,dt,db,dp){
 dt <- (90-dt)
 db <- (90-db)
 dbt <- db-dt
 abt <- azb-azt
 d <- acos(cos(d2r(dbt))-sin(d2r(dt))*sin(d2r(db))*(1-cos(d2r(abt))))
```

```
 r <- 1
 if(d==0){r <- 1}
 else{r <- 2*tan(d/2)/d}
 x = 0.5*dp*(sin(d2r(dt))*sin(d2r(azt))+ sin(d2r(db))*sin(d2r(azb)))*r
 y = 0.5*dp*(sin(d2r(dt))*cos(d2r(azt))+ sin(d2r(db))*cos(d2r(azb)))*r
 z = 0.5*dp*(cos(d2r(dt))+cos(d2r(db)))*r
 return(c(x,y,z))
}
#
calcXYZ<-function(hole,depth){
 dadoxyz <- c(0.0,0.0,0.0)
 t <- Spts[which(Spts$hole==hole & Spts$prof <=depth),]
 t <- t[order(-t$prof),]
 b <- Spts[which(Spts$hole==hole & Spts$prof > depth),]
 if(dim(t)[1] == 0){return (c(0,0,0))}
 di <- b[1,2]- t[1,2]
 rga <- t[1,6] - b[1,6]
 stp <- rga/di
 dpt <- depth-t[1,2]
 pang <-stp*dpt
 if(dim(b)[1] >= 1){
 res <- calculo(t[1,7], b[1,7],t[1,6], t[1,6]-pang, dpt)
 return(c((t[1,3]+res[1]), (t[1,4]+res[2]), (t[1,5]+res[3])))
 }
 else{
 res <- calculo(t[1,7], t[1,7], t[1,6], t[1,6], dpt)
 return(c((t[1,3]+res[1]), (t[1,4]+res[2]), (t[1,5]+res[3])))
 }
}
#
for(i in 1 : dim(dat)[1]){
 dado <- calcXYZ(dat$hole[i],dat$prof[i])
 dat$x[i] <- dado[1]
 dat$y[i] <- dado[2]
 dat$z[i] <- dado[3]
}
#
colnames(dat)<-c("BHID","X","Y","Z","AU","FROM","FROML","TO",    "TOL","LENGTH",
"LENHTL","DOMAIN","ROCKTYPE","WEATH")
#
write.csv(dat,"RDESURVEYED.CSV",row.names = FALSE)
write.csv(Spts,"SURVEY_PTS.CSV",row.names = FALSE)
```

Visualizando os arquivos gerados com Paraview:



. . .

8 Interpolação de Pontos 2D

Neste capítulo vamos trabalhar com bibliotecas que não conseguem usar dados do pacote terra ainda. Vamos usar o pacote raster e o arquivo geoqui.shp criado anteriormente nestas receitas.

Receita - 8.1 – Distância

Obtermos a distância entre pontos usando a função spDists que toma como parâmetros dois objetos Spatial* (no nosso caso usaremos o mesmo objeto duas vezes) e um terceiro parâmetro que informa se o dado é Euclidiano (ex. UTM) ou Longitude / Latitude.

Usaremos arrays nessa função como os dois primeiros argumentos onde a primeira e segunda colunas denotando Longitude e Latitude e cada linha representa um ponto.

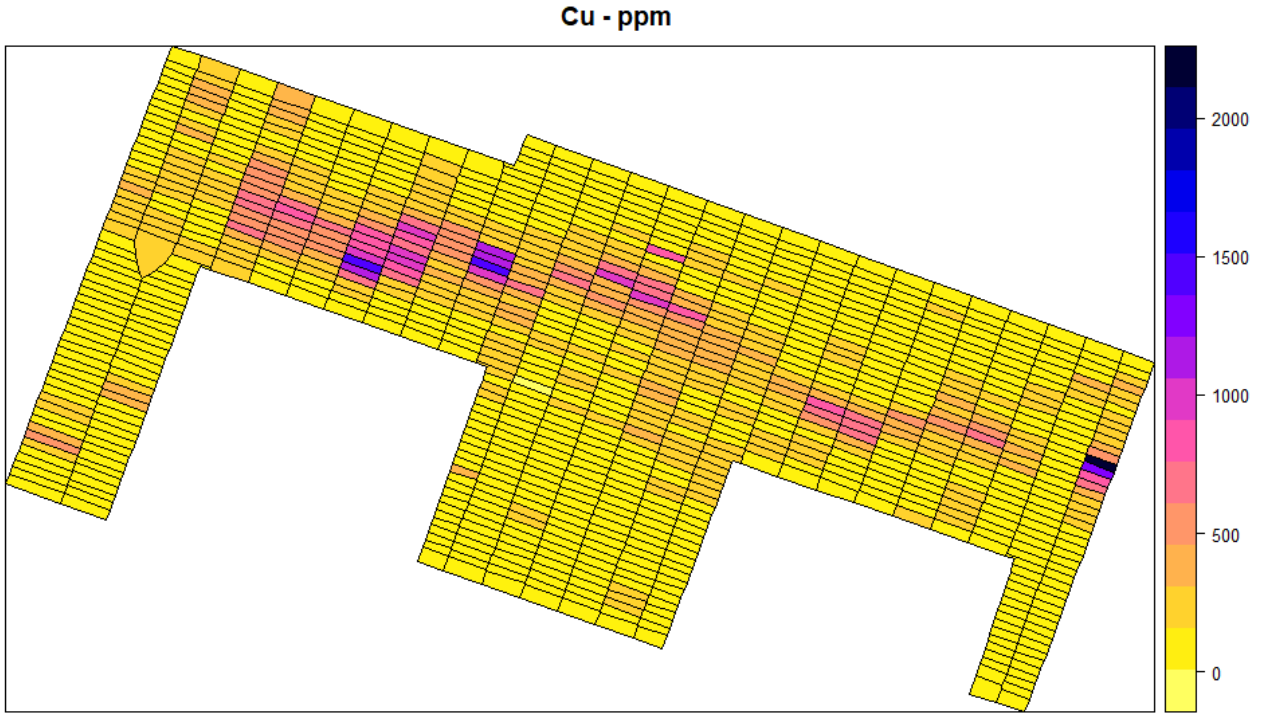
Criando a nossa matriz de distâncias e em seguida criar uma matriz de inverso da distância normalizado a 1:

```
library(terra) #install terra using install.packages("terra")
dado<-vect('geoqui.shp')
dado
 class       : SpatVector
 geometry    : points
 dimensions  : 956, 16  (geometries, attributes)
 extent      : 590142, 595675, 8986418, 8989594  (xmin, xmax, ymin, ymax)
 source      : geoqui.shp
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       : amostra Au_gpt    Ba    Cd    Co    Cr    Cu    Mo    Ni    Pb
 type        :   <int>  <num> <num> <num> <num> <num> <num> <num> <num> <num>
 values      :       1   0.01   5.1   6.8     3    58    46     1   5.2    12
                     2   0.01    10   3.9     3    38    35     1   4.6    11
                     3   0.01   7.1   3.7     3    43    29     1     4    12
 (and 6 more)
distancias<-distance(dado, y = dado)
rownames(distancias)<-dado$amostra
colnames(distancias)<-dado$amostra
distancias[1:6,1:6] #visualizando as 6 primeiras linhas e colunas
          1         2         3         4         5         6
1   0.00000  40.16217  79.71198 120.20815 159.42396 199.91998
2  40.16217   0.00000  39.56008  80.05623 119.26860 159.76545
3  79.71198  39.56008   0.00000  40.49691  79.71198 120.20815
4 120.20815  80.05623  40.49691   0.00000  39.21734  79.71198
5 159.42396 119.26860  79.71198  39.21734   0.00000  40.49691
6 199.91998 159.76545 120.20815  79.71198  40.49691   0.00000
w<-1/distancias
w[!is.finite(w)] <- NA
rtot<-apply(w,1,sum,na.rm=TRUE)
wnorm<-round(w/rtot,8)
wnorm[1:6,1:6]
           1          2          3          4          5          6
1         NA 0.04343730 0.02188549 0.01451263 0.01094275 0.00872617
2 0.04105078         NA 0.04167556 0.02059413 0.01382332 0.01031943
3 0.02005059 0.04040114         NA 0.03946652 0.02005059 0.01329587
4 0.01298204 0.01949313 0.03853496         NA 0.03979226 0.01957732
5 0.00960954 0.01284488 0.01921907 0.03906411         NA 0.03782981
6 0.00755341 0.00945184 0.01256219 0.01894417 0.03728870         NA
```

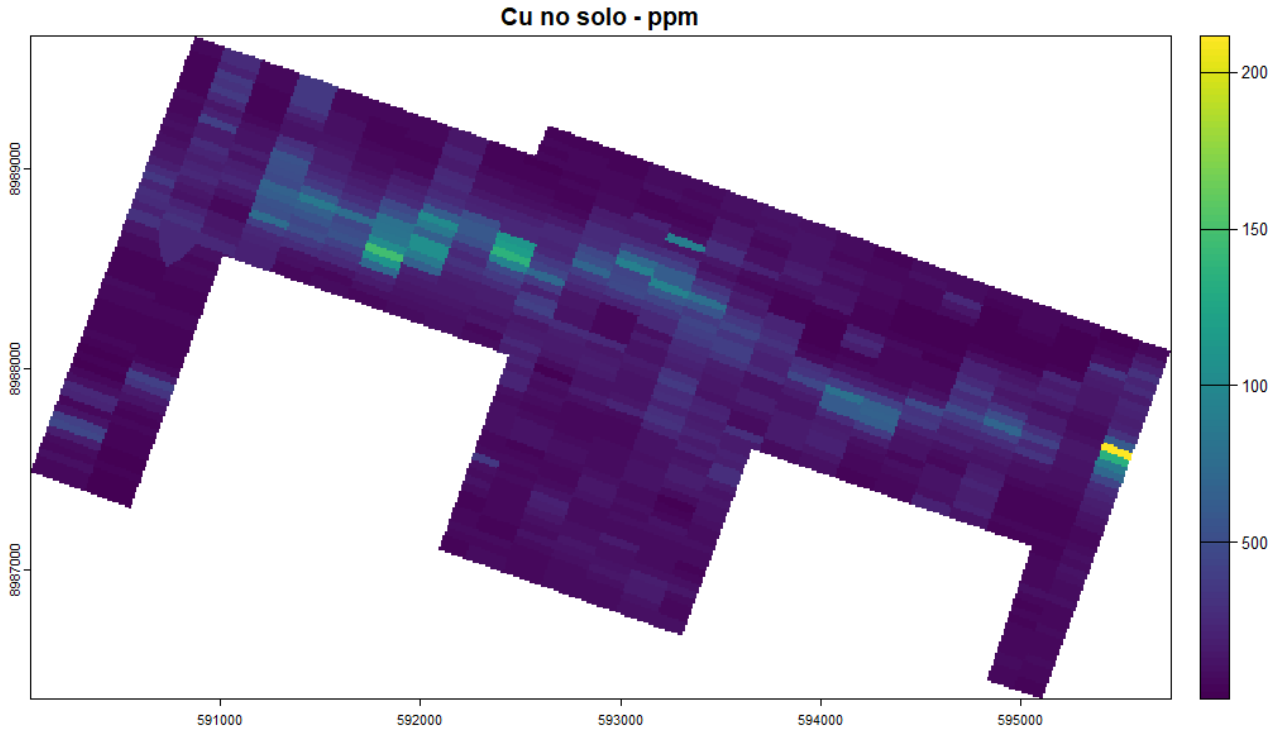
Receita - 8.2 – Voronoi

Gerando polígonos Voronoi para armazenar valores de cada ponto individual e usando uma máscara (ot.shp) para cortar dados fora do domínio dos pontos. Posteriormente executamos a rasterização do resultado obtido.

```
library(dismo) #install dismo using install.packages("dismo")
library(terra)
dado<-vect('geoqui.shp')
dado
 class       : SpatVector
 geometry    : points
 dimensions  : 956, 16  (geometries, attributes)
 extent      : 590142, 595675, 8986418, 8989594  (xmin, xmax, ymin, ymax)
 source      : geoqui.shp
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       : amostra Au_gpt    Ba    Cd    Co    Cr    Cu    Mo    Ni    Pb
 type        :   <int>  <num> <num> <num> <num> <num> <num> <num> <num> <num>
 values      :       1   0.01   5.1   6.8     3    58    46     1   5.2    12
                     2   0.01    10   3.9     3    38    35     1   4.6    11
                     3   0.01   7.1   3.7     3    43    29     1     4    12
 (and 6 more)
mascara<- vect('ot.shp')
 OGR data source with driver: ESRI Shapefile
 Source: "C:\Users\andre.costa\Documents", layer: "ot"
 with 1 features
 It has 1 fields
 Integer64 fields read as strings:  id
mascara
 class       : SpatVector
 geometry    : polygons
 dimensions  : 1, 1  (geometries, attributes)
 extent      : 590050, 595746.7, 8986354, 8989662  (xmin, xmax, ymin, ymax)
 source      : ot.shp
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       :    id
 type        : <int>
 values      :    NA
v<-voronoi(dado)
vor<- crop(v,mascara)
spplot(vor,'Cu',col.regions=rev(get_col_regions()), main="Cu - ppm")
```



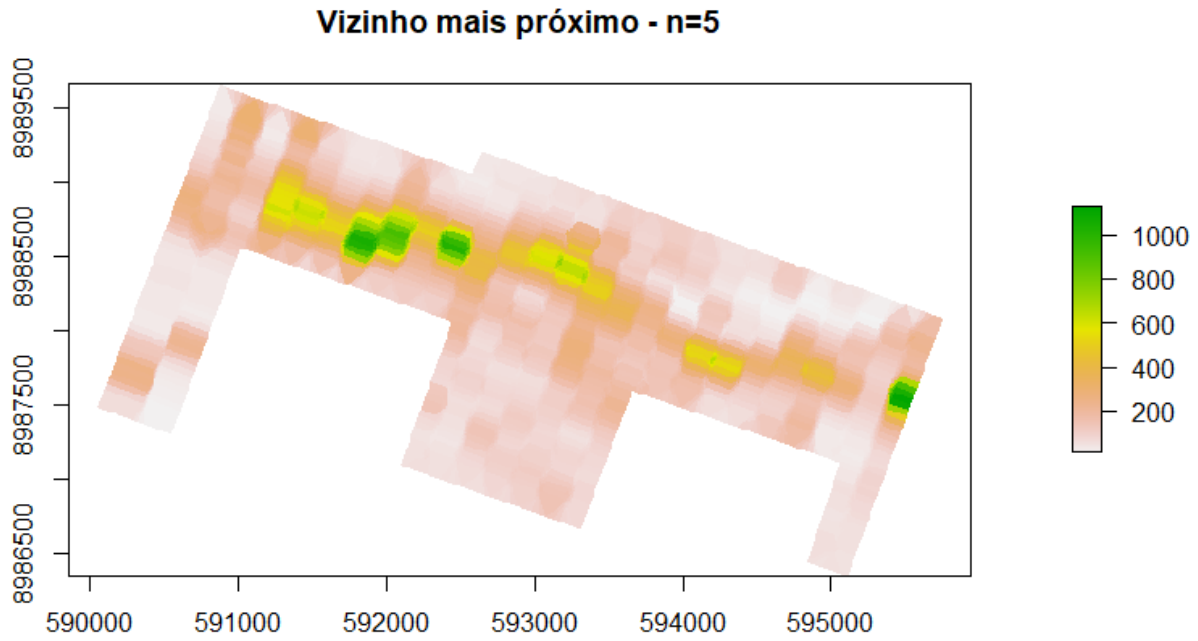

```
rast<-rast(mascara,res=10)
v.rast<-rasterize(vor,rast,'Cu')
plot(v.rast,main='Cu no solo - ppm')
```



Receita - 8.3 – Vizinho mais próximo

Interpolação pelo método vizinho mais próximo.

```
library(raster)
library(terra)
dado<-as(vect('geoqui.shp') , "Spatial")
dado
class       : SpatialPointsDataFrame
features    : 956
extent      : 590142, 595675, 8986418, 8989594  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=17 +south +datum=WGS84 +units=m +no_defs
variables   : 16
names       : amostra, Au_gpt,  Ba, Cd,  Co,   Cr,   Cu, Mo,  Ni, Pb, Sb, Sc,   V,  Y, Zn, ...
min values  :       1,   0.01,   1,  1,   3,    1,    1,  1,   1,  3,  5,  3,   3,  1,  1, ...
max values  :    1008,   1.06, 272, 25, 123, 1219, 2115, 49, 143, 83, 19, 41, 386, 79, 66, ...
3   0.01   7.1   3.7     3    43    29     1     4    12
(and 6 more)
mascara<- vect('ot.shp')
mascara
 class       : SpatVector
 geometry    : polygons
 dimensions  : 1, 1  (geometries, attributes)
 extent      : 590050, 595746.7, 8986354, 8989662  (xmin, xmax, ymin, ymax)
 source      : ot.shp
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       :    id
 type        : <int>
 values      :    NA
rast<-raster(as(mascara, "Spatial"),res=10)
library(gstat)
metodo<-gstat(formula=Cu~1,locations=dado,nmax=5,set=list(idp=0))
vmp<-interpolate(rast,metodo,debug.level=0,index=1)
vmpmsc<-mask(vmp,as(mascara, "Spatial"))
plot(vmpmsc,main='Vizinho mais próximo - n=5')
```



Receita - 8.4 – Inverso da Distância

Interpolação pelo Método IQD.

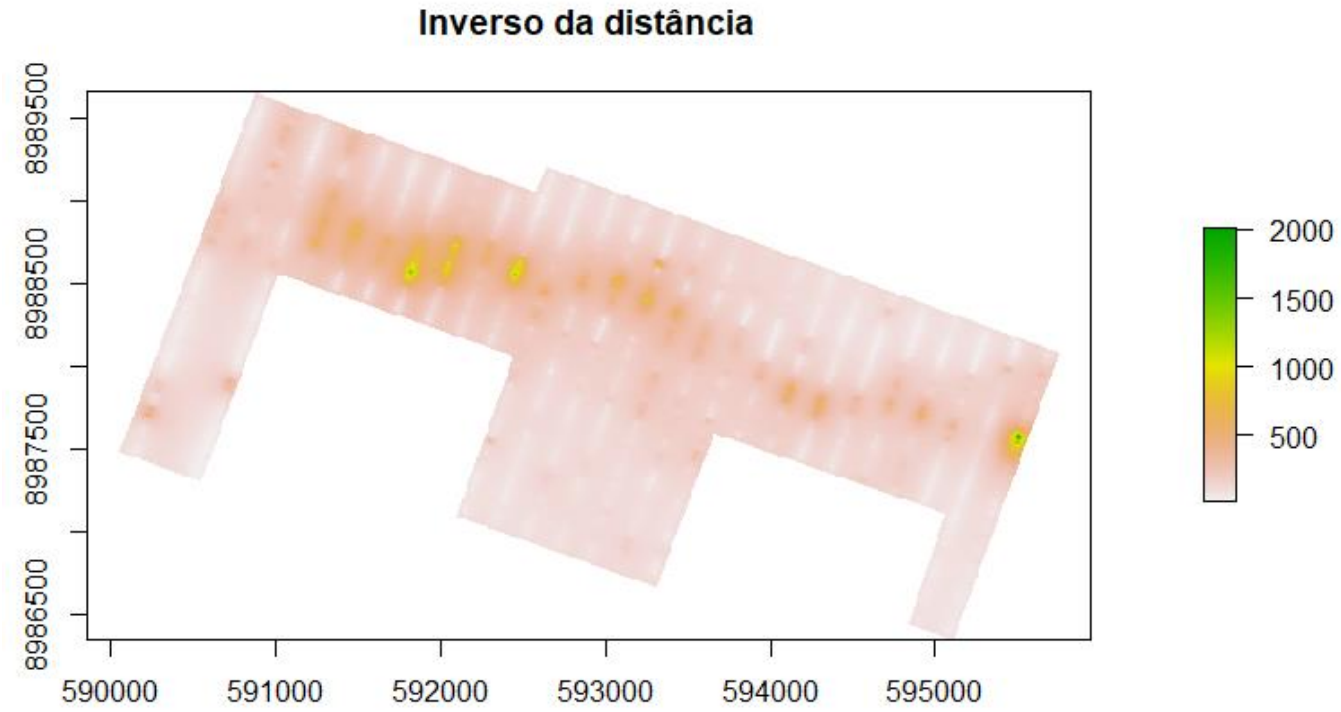
```
library(raster)
library(terra)
dado<-as(vect('geoqui.shp') , "Spatial")
dado
```

```
class       : SpatialPointsDataFrame
features    : 956
extent      : 590142, 595675, 8986418, 8989594  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=17 +south +datum=WGS84 +units=m +no_defs
variables   : 16
names       : amostra, Au_gpt,  Ba, Cd,  Co,   Cr,   Cu, Mo,  Ni, Pb, Sb, Sc,   V,  Y, Zn, ...
min values  :       1,   0.01,   1,  1,   3,    1,    1,  1,   1,  3,  5,  3,   3,  1,  1, ...
max values  :    1008,   1.06, 272, 25, 123, 1219, 2115, 49, 143, 83, 19, 41, 386, 79, 66, ...
(and 6 more)
```

```
mascara<- vect('ot.shp')
mascara
```

```
 class       : SpatVector
 geometry    : polygons
 dimensions  : 1, 1  (geometries, attributes)
 extent      : 590050, 595746.7, 8986354, 8989662  (xmin, xmax, ymin, ymax)
 source      : ot.shp
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       :    id
 type        : <int>
 values      :    NA
```

```
rast<-raster(as(mascara, "Spatial"),res=10)
library(gstat)

metodo <- gstat(formula=Cu~1, locations=dado)
idw<-interpolate(rast,metodo)
idwmsc <-mask(idw,as(mascara, "Spatial"))
plot(idwmsc,main='Inverso da distância')
```



Receita - 8.5 – Krigagem

Interpolação por krigagem Ordinária.

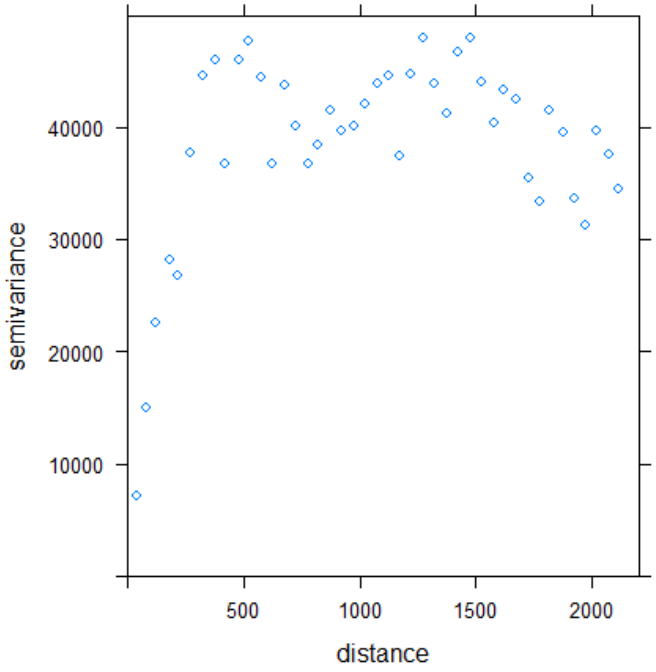
```
library(raster)
library(terra)
dado<-as(vect('geoqui.shp') , "Spatial")
dado
```

```
class       : SpatialPointsDataFrame
features    : 956
extent      : 590142, 595675, 8986418, 8989594  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=17 +south +datum=WGS84 +units=m +no_defs
variables   : 16
names       : amostra, Au_gpt,  Ba, Cd,  Co,   Cr,   Cu, Mo,  Ni, Pb, Sb, Sc,   V,  Y, Zn, ...
min values  :       1,   0.01,   1,  1,   3,    1,    1,  1,   1,  3,  5,  3,   3,  1,  1, ...
max values  :    1008,   1.06, 272, 25, 123, 1219, 2115, 49, 143, 83, 19, 41, 386, 79, 66, ...
(and 6 more)
```

```
mascara<- vect('ot.shp')
mascara
```

```
 class       : SpatVector
 geometry    : polygons
 dimensions  : 1, 1  (geometries, attributes)
 extent      : 590050, 595746.7, 8986354, 8989662  (xmin, xmax, ymin, ymax)
 source      : ot.shp
 coord. ref. : WGS 84 / UTM zone 17S (EPSG:32717)
 names       :    id
 type        : <int>
 values      :    NA
```

```
library(gstat)
krig.met <- gstat(formula=Cu~1, locations=dado)
vario <- variogram(krig.met, width=50)
f.v <- fit.variogram(vario, vgm(c("Sph","Gau","Exp")))
plot(vario)
```

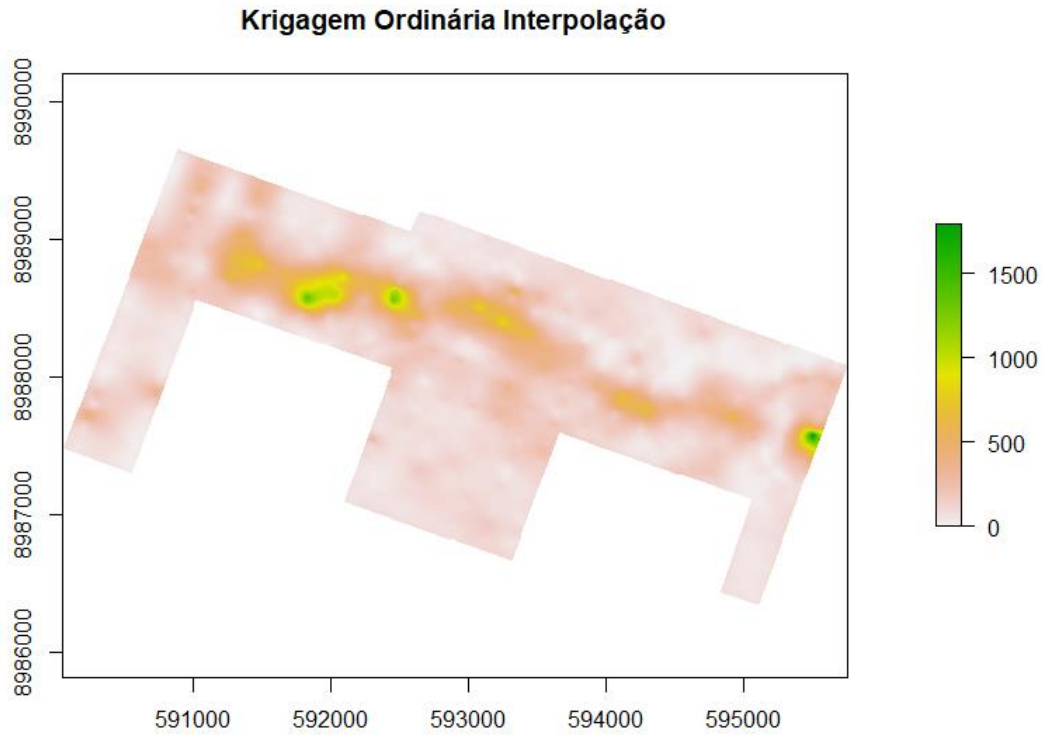



```
f.v  #parâmetros do semivariograma
```

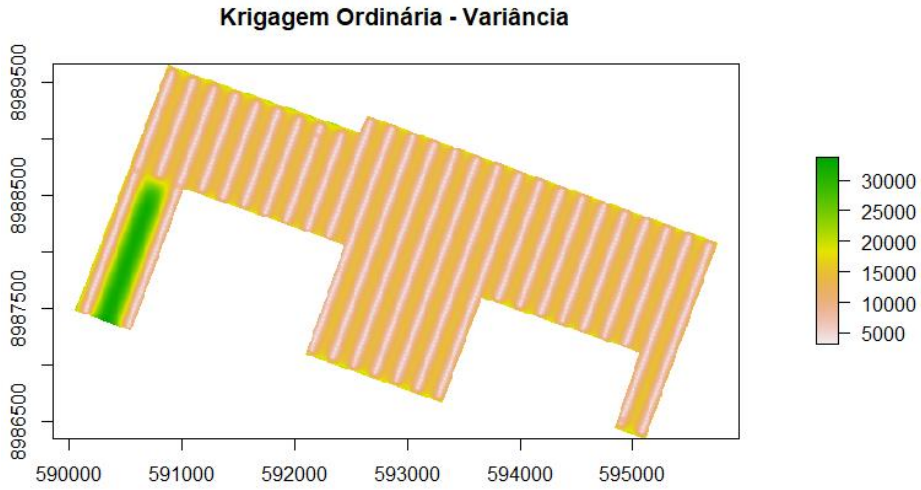
```
  model     psill    range
1   Nug  1679.221   0.0000
2   Sph 39454.770 398.5911
```

```
krig<- gstat(formula=Cu~1, locations=dado, model=f.v)
ur<- raster(as(mascara, "Spatial"),res=10)
ugri <- as(ur, 'SpatialGrid')
kri.pred<-predict(krig,ugri)

ko <- brick(kri.pred)
ko <- mask(ko, as(mascara, "Spatial"))
```

```
names(ko) <- c('Interpolação', 'Variância')
plot(ko$Interpolação, main='Krigagem Ordinária Interpolação')
```




```
plot(ko$Variância, main= 'Krigagem Ordinária - Variância')
```



Sendo este o melhor resultado vamos exportar o raster como CSV para usar nos testes de correlações seguintes.

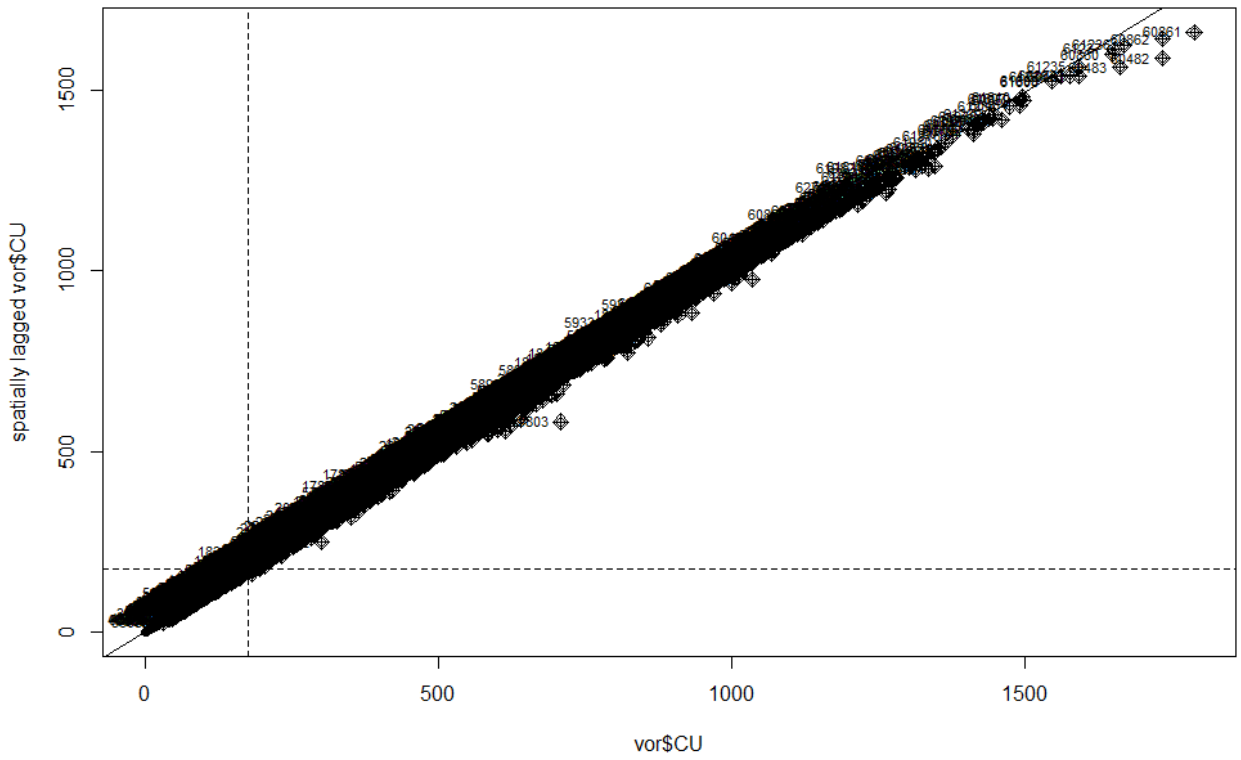
```
xyz <- rasterToPoints(ko$Interpolação)
colnames(xyz)<-c("UTME","UTMN","CU")
write.csv(xyz, "melhor.csv", row.names=FALSE)
```

Receita - 8.6 – Autocorrelação Espacial (Teste de Moran)

Este modelo é conhecido como teste de Moran. Ele criará uma graduação de correlação entre -1 e 1. Tal como um coeficiente de correlação, 1 determina autocorrelação espacial positiva perfeita, 0 indica que os dados são aleatoriamente distribuídos e -1 representa autocorrelação espacial negativa.

Calculamos inicialmente os vizinhos mais próximos sobre o arquivo com melhor resultado de interpolação (melhor.csv) nas etapas acima (no caso a krigagem) criando polígonos Voronoi de cada ponto com valor associado criando uma lista de vizinhos e aplicamos o teste de Moran.

```
library(terra)
library(raster)
geoq<-read.csv('melhor.csv')
geoqui<-vect(geoq, geom=c("UTME", "UTMN"),crs="+init=epsg:32717")
mascara<- vect('ot.shp')
v<-voronoi(geoqui)
vor<-crop(v,mascara)
library(spdep)
vizin<-poly2nb(as(vor,'Spatial'))
vizin
 Neighbour list object:
 Number of regions: 78521
 Number of nonzero links: 622292
 Percentage nonzero weights: 0.01009305
 Average number of links: 7.925167
lista.vz<-nb2listw(vizin)
lista.vz
 Characteristics of weights list object:
 Neighbour list object:
 Number of regions: 78521
 Number of nonzero links: 622292
 Percentage nonzero weights: 0.01009305
 Average number of links: 7.925167

 Weights style: W
 Weights constants summary:
   n          nn    S0       S1       S2
 W 78521 6165547441 78521 19901.57 314248.9
mo<-moran.test(vor$CU,lista.vz)
mo
      Moran I test under randomisation
 data:  vor$CU
 weights: lista.vz

 Moran I statistic standard deviate = 554.44, p-value < 2.2e-16
 alternative hypothesis: greater
 sample estimates:
 Moran I statistic       Expectation          Variance
      9.960117e-01     -1.273561e-05      3.227192e-06
moran <- moran.plot(vor$CU, listw = nb2listw(vizin, style = "W"))
```



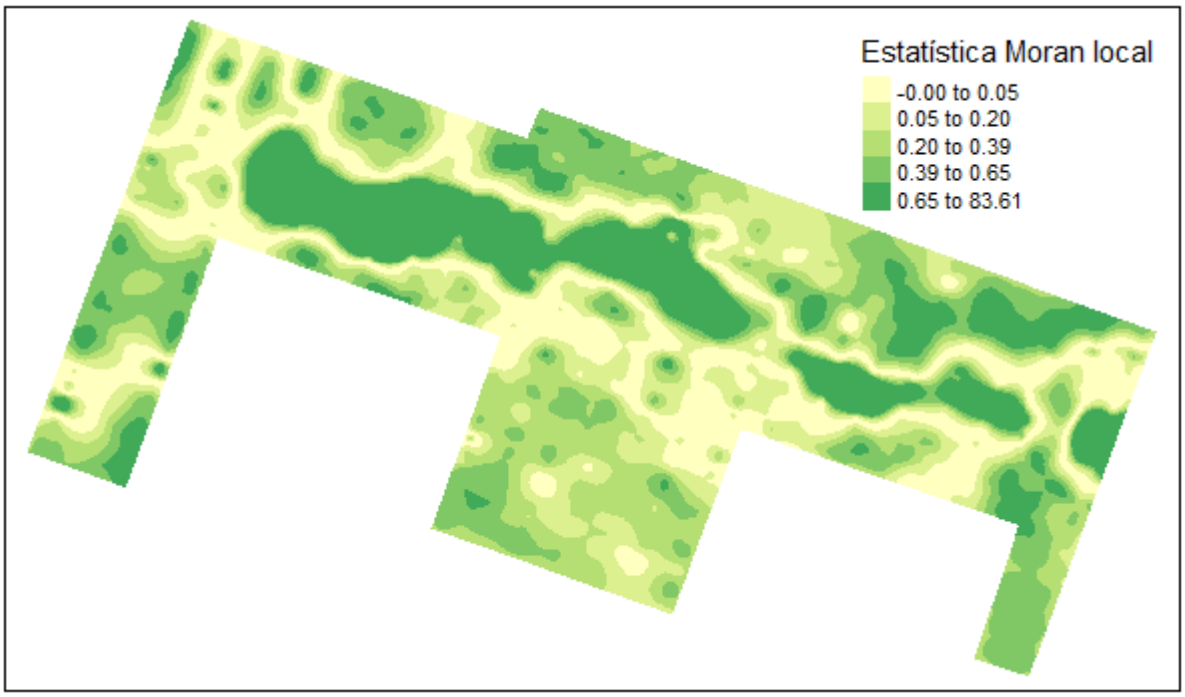

```
mo.local <- localmoran(x=vor$CU, listw = nb2listw(vizin, style = "W"))
head(mo.local)
```

```
         Ii           E.Ii     Var.Ii     Z.Ii Pr(z != E(Ii))
1 0.4750409 -6.075008e-06 0.15900022 1.191345     0.23351802
2 0.4558659 -5.627332e-06 0.11046110 1.371633     0.17017784
3 0.5046008 -6.470802e-06 0.08467637 1.734093     0.08290158
4 0.4748128 -5.991551e-06 0.06720340 1.831607     0.06700997
5 0.4350343 -5.444376e-06 0.07124471 1.629869     0.10312927
6 0.3950668 -4.841281e-06 0.09503146 1.281570     0.19999369
```

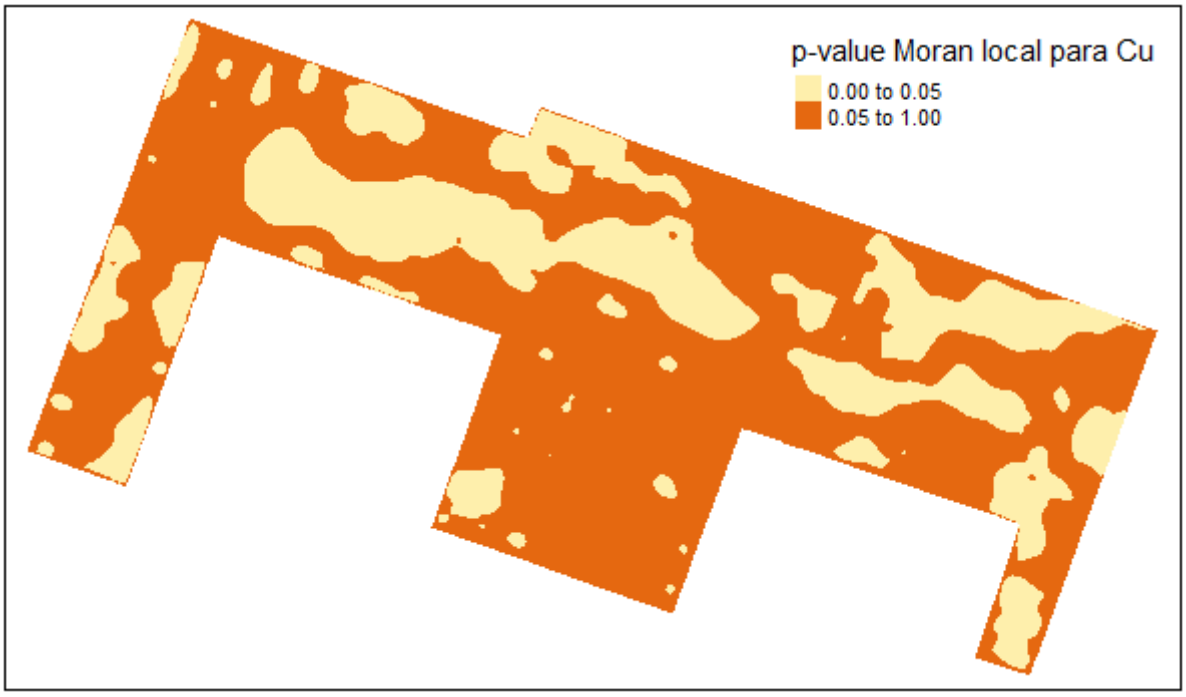
Onde:

Ii	Estatística Moran do ponto
E.li	Expectativa da estatística Moran local
Var.li	Variância da estatística Moran local
Z.li	Desvio padrão da estatística Moran local
Pr()	p-value de estatística Moran local

```
mapa.moran<-cbind(vor,mo.local)
library(tmap)
mm<-as(mapa.moran,'Spatial')
tm_shape(mm) + tm_fill('Ii',style='quantile',title='Estatística Moran local')
```



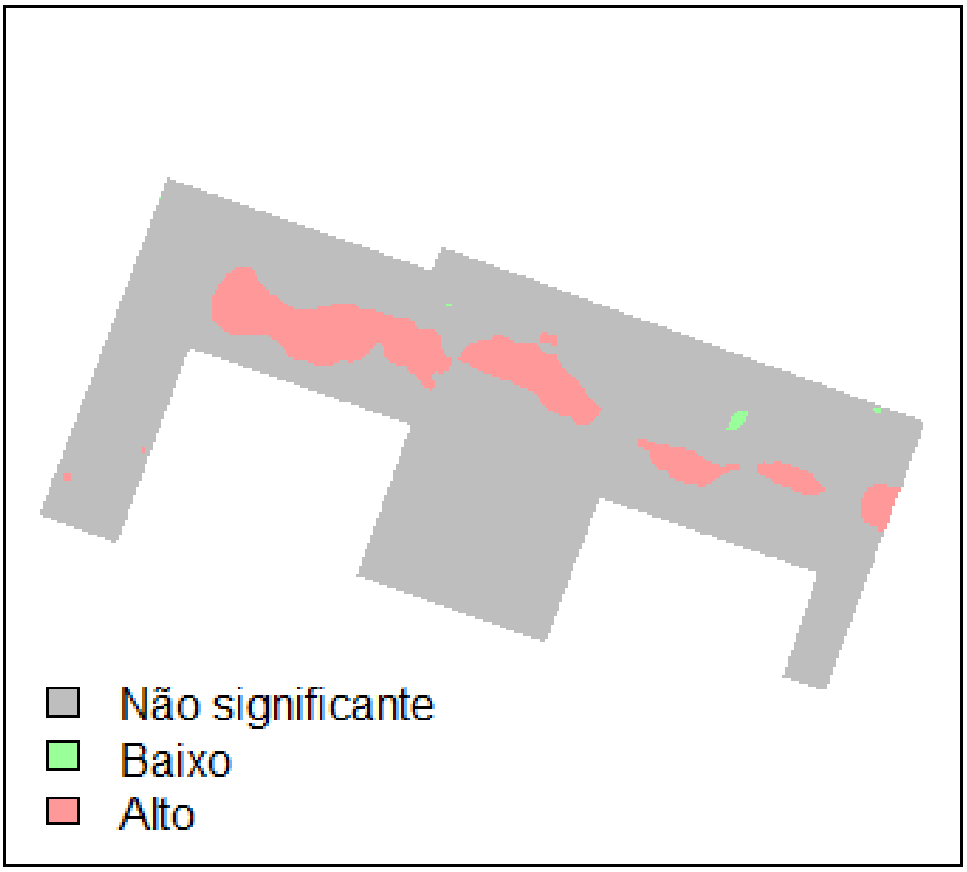

```
tm_shape(mm)+tm_fill('Pr(z != E(Ii))',n=10,style='fixed',breaks= c(0,0.05,1),
title = 'p-value Moran local para Cu')
```



Receita - 8.7 – LISA (Local Indicators of Spatial Association)

Efetuando um mapa que categoriza o tipo de relação cada elemento possui com seu vizinho ou LISA .

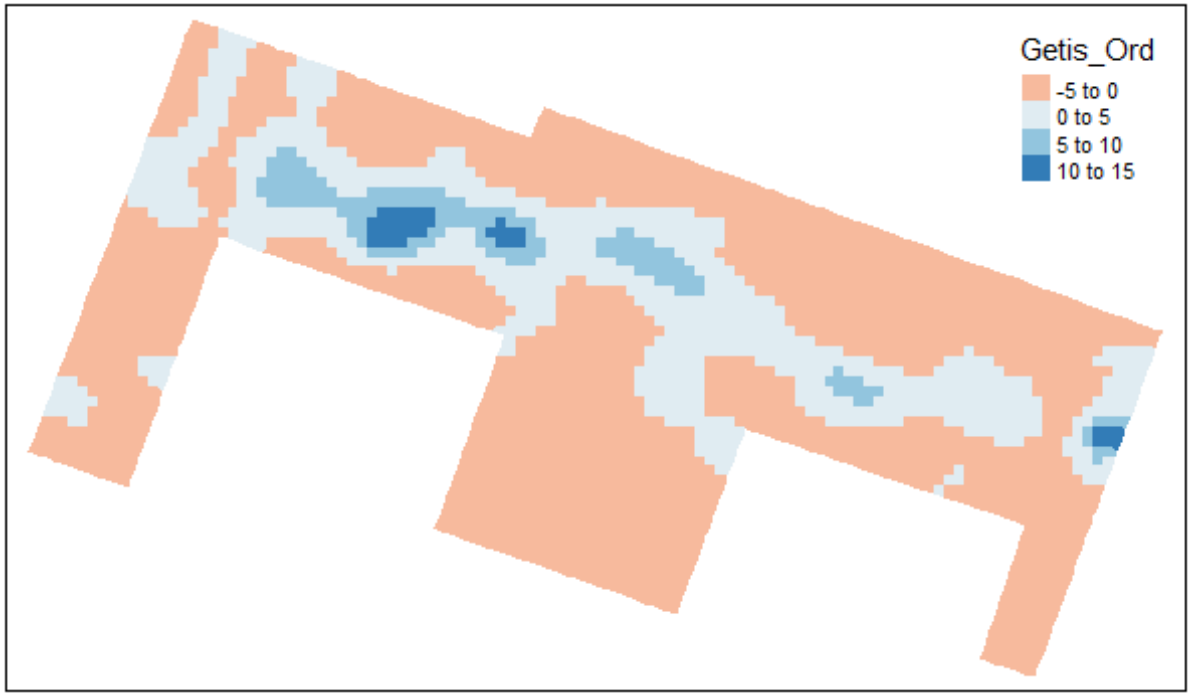
```
library(terra)
library(raster)
geoq<-read.csv('melhor.csv')
geoqui<-vect(geoq, geom=c("UTME", "UTMN"),crs="+init=epsg:32717")
mascara<- vect('ot.shp')
v<-voronoi(geoqui)
vor<-crop(v,mascara)
library(spdep)
vizin<-poly2nb(as(vor,'Spatial'))
lista.vz<-nb2listw(vizin)
mo<-moran.test(vor$CU,lista.vz)
mo.local <- localmoran(x=vor$CU, listw = nb2listw(vizin, style = "W"))

qua<-vector(mode='numeric',length=nrow(mo.local))
media.Cu<-vor$CU-mean(vor$CU)
media.mo<-mo.local[,1]-mean(mo.local[,1])
significancia<-0.1
qua[media.Cu >0 & media.mo>0]<-2
qua[media.Cu <0 & media.mo>0]<-1
qua[mo.local[,5] >significancia]<-0
quebras <- c(0,1,2)
cores <- c('grey',rgb(0,1,0,alpha=0.4),rgb(1,0,0,alpha=0.4))
plot(vor,border=NA,col=cores[findInterval(qua,quebras,all.inside=FALSE)])
box(lty='blank')
legend('bottomleft',legend=c('Não      significante','Baixo','Alto'),fill=cores,
bty="n")
```



Receita - 8.8 – Getis-Ord Gi*

Usando o método Getis-Ord Gi*, também conhecido como análise de hot-spot. Com esse método achamos zonas quentes no meio de zonas relativamente altas também.
Definimos um novo grupo de vizinhos baseados somente na proximidade, não necessariamente dividindo bordas.

```
library(terra)
library(raster)
mascara<- vect('ot.shp')
dado<-as(vect('geoqui.shp') , "Spatial")
library(gstat)
krig.met <- gstat(formula=Cu~1, locations=dado)
vario <- variogram(krig.met, width=50)
f.v <- fit.variogram(vario, vgm(c("Sph","Gau","Exp")))
krig<- gstat(formula=Cu~1, locations=dado, model=f.v)
ur<- raster(as(mascara, "Spatial"),res=50)
ugri <- as(ur, 'SpatialGrid')
kri.pred<-predict(krig,ugri)
ko <- brick(kri.pred)
ko <- mask(ko, as(mascara, "Spatial"))
names(ko) <- c('Interpolação', 'Variância')
xyz <- rasterToPoints(ko$Interpolação)
colnames(xyz)<-c("UTME","UTMN","CU")
write.csv(xyz, "melhorLR.csv", row.names=FALSE)
geoq<-read.csv('melhorLR.csv')
geoqui<-vect(geoq, geom=c("UTME", "UTMN"),crs="+init=epsg:32717")
v<-voronoi(geoqui)
vor<-crop(v,mascara)
library(spdep)
vizin2<-dnearneigh(coordinates(as(vor,"Spatial")),0,80)
lista.vz2<-nb2listw(vizin2,style='B')
go.local <- localG(vor$CU, lista.vz2)
go.local <- cbind(vor,as.matrix(go.local))
names(go.local)[2]<-'Getis_Ord'
library(tmap)
tm_shape(as(go.local, "Spatial")) + tm_fill("Getis_Ord", palette = "RdBu", style
= "pretty")
```

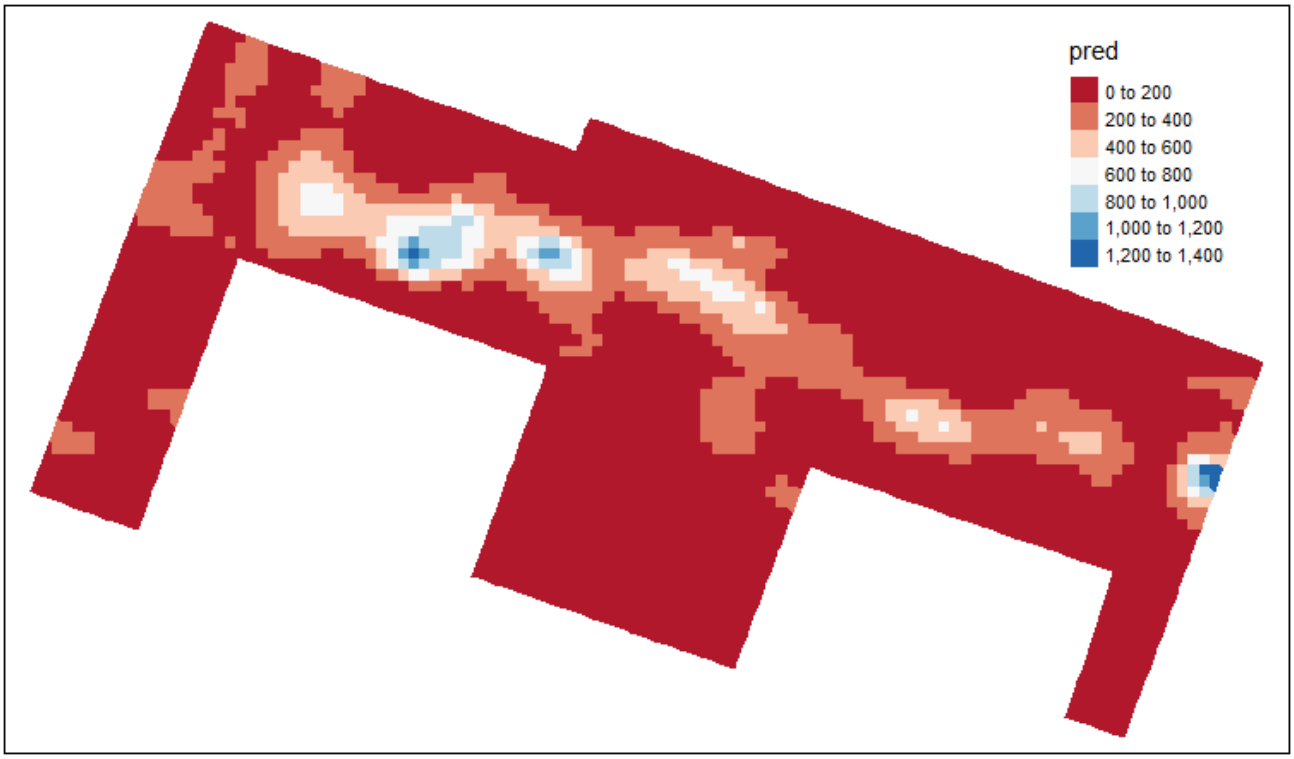


Receita - 8.9 – GWR (Geographically Weighted Regression)

Efetuaremos um teste usando um modelo GWR (Geographically Weighted Regression). Usaremos o pacote spgwr neste teste. Usaremos ouro para prever o cobre e para isso precisarmos criar uma base usando krigagem para os teores de ouro como fizemos anteriormente para Cobre.

```
library(terra)
library(raster)
mascara<- vect('ot.shp')
dado<-as(vect('geoqui.shp') , "Spatial")
library(gstat)

krig.met <- gstat(formula=Au_gpt~1, locations=dado)
vario <- variogram(krig.met, width=50)
f.v <- fit.variogram(vario, vgm(c("Sph","Gau","Exp")))
krig<- gstat(formula=Au_gpt~1, locations=dado, model=f.v)
ur<- raster(as(mascara, "Spatial"),res=50)
ugri <- as(ur, 'SpatialGrid')
kri.pred<-predict(krig,ugri)
ko <- brick(kri.pred)
ko <- mask(ko, as(mascara, "Spatial"))
names(ko) <- c('Interpolação', 'Variância')
xyz <- rasterToPoints(ko$Interpolação)
colnames(xyz)<-c("UTME","UTMN","AU")
write.csv(xyz, "melhor_au.csv", row.names=FALSE)

krig.met <- gstat(formula=Cu~1, locations=dado)
vario <- variogram(krig.met, width=50)
f.v <- fit.variogram(vario, vgm(c("Sph","Gau","Exp")))
krig<- gstat(formula=Cu~1, locations=dado, model=f.v)
ur<- raster(as(mascara, "Spatial"),res=50)
ugri <- as(ur, 'SpatialGrid')
kri.pred<-predict(krig,ugri)
ko <- brick(kri.pred)
ko <- mask(ko, as(mascara, "Spatial"))
names(ko) <- c('Interpolação', 'Variância')
xyz <- rasterToPoints(ko$Interpolação)
colnames(xyz)<-c("UTME","UTMN","CU")
write.csv(xyz, "melhor_cu.csv", row.names=FALSE)

geoqui1<-read.csv('melhor_cu.csv')
geoqui2<-read.csv('melhor_au.csv')
geoqu<-data.frame(geoqui1,geoqui2$AU)
colnames(geoqu)<-c("UTME","UTMN","CU","AU")
geoqui<-vect(geoqu, geom=c("UTME", "UTMN"),crs="+init=epsg:32717")
v<-voronoi(geoqui)
vor<-crop(v,mascara)
library(spgwr)
faixa.gwr<-gwr.sel(vor$CU~vor$AU,data=as(vor, "Spatial"), adapt=TRUE)
Adaptive q: 0.381966 CV score: 55716070
Adaptive q: 0.618034 CV score: 57387225
Adaptive q: 0.236068 CV score: 52874806
Adaptive q: 0.145898 CV score: 48211707
Adaptive q: 0.09016994 CV score: 42392348
Adaptive q: 0.05572809 CV score: 36204448
Adaptive q: 0.03444185 CV score: 31505155
Adaptive q: 0.02128624 CV score: 26530922
Adaptive q: 0.01315562 CV score: 22098388
Adaptive q: 0.008130619 CV score: 18115633
Adaptive q: 0.005024999 CV score: 12528538
Adaptive q: 0.00310562 CV score: 10384401
Adaptive q: 0.001919379 CV score: 6209050
Adaptive q: 0.001186241 CV score: 3825456
Adaptive q: 0.0007331374 CV score: 3679102
Adaptive q: 0.000894264 CV score: 3700954
Adaptive q: 0.0006924473 CV score: 3673672
Adaptive q: 0.000427956 CV score: 3618510
Adaptive q: 0.0002644913 CV score: 2934807
Adaptive q: 0.0001634646 CV score: 2351761
Adaptive q: 0.0001010267 CV score: 2772414
Adaptive q: 0.0002041547 CV score: 2481491
Adaptive q: 0.0001634646 CV score: 2351761
```

```
modelo.gwr<-gwr(vor$CU ~ vor$AU, data= as(vor, "Spatial"),adapt=faixa.gwr,
hatmatrix=TRUE, se.fit=TRUE)
modelo.gwr
Call:
gwr(formula = vor$CU ~ vor$AU, data = vor, adapt = faixa.gwr,
    hatmatrix = TRUE, se.fit = TRUE)
Kernel function: gwr.Gauss
Adaptive quantile: 0.0001634646 (about 0 of 3141 data points)
Summary of GWR coefficient estimates at data points:
                   Min.    1st Qu.     Median    3rd Qu.       Max.  Global
X.Intercept.   -782.815     24.123     80.045    157.451   1707.361  115.51
vor.AU       -58106.717   -118.601   1360.545   4167.845  65895.934 1489.71
Number of data points: 3141
Effective number of parameters (residual: 2traceS - traceS'S): 2675.557
Effective degrees of freedom (residual: 2traceS - traceS'S): 465.4425
Sigma (residual: 2traceS - traceS'S): 21.73402
Effective number of parameters (model: traceS): 2092.351
Effective degrees of freedom (model: traceS): 1048.649
Sigma (model: traceS): 14.47965
Sigma (ML): 8.366411
AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 34822.47
AIC (GWR p. 96, eq. 4.22): 24350.5
Residual sum of squares: 219860.1
Quasi-global R2: 0.9975564
gwr.res<-as.data.frame(modelo.gwr$SDF)
names(gwr.res)
[1] "sum.w"  "X.Intercept."    "vor.AU"    "X.Intercept._se"     "vor.AU_se"         "gwr.e"
[7] "pred"   "pred.se"         "localR2"   "X.Intercept._se_EDF" "vor.AU_se_EDF"     "pred.se.1"
gwr.final<-cbind(as(vor, "Spatial"),as.matrix(gwr.res))
library(tmap)
tm_shape(gwr.final) + tm_fill("pred", palette = "RdBu", style = "pretty")
```



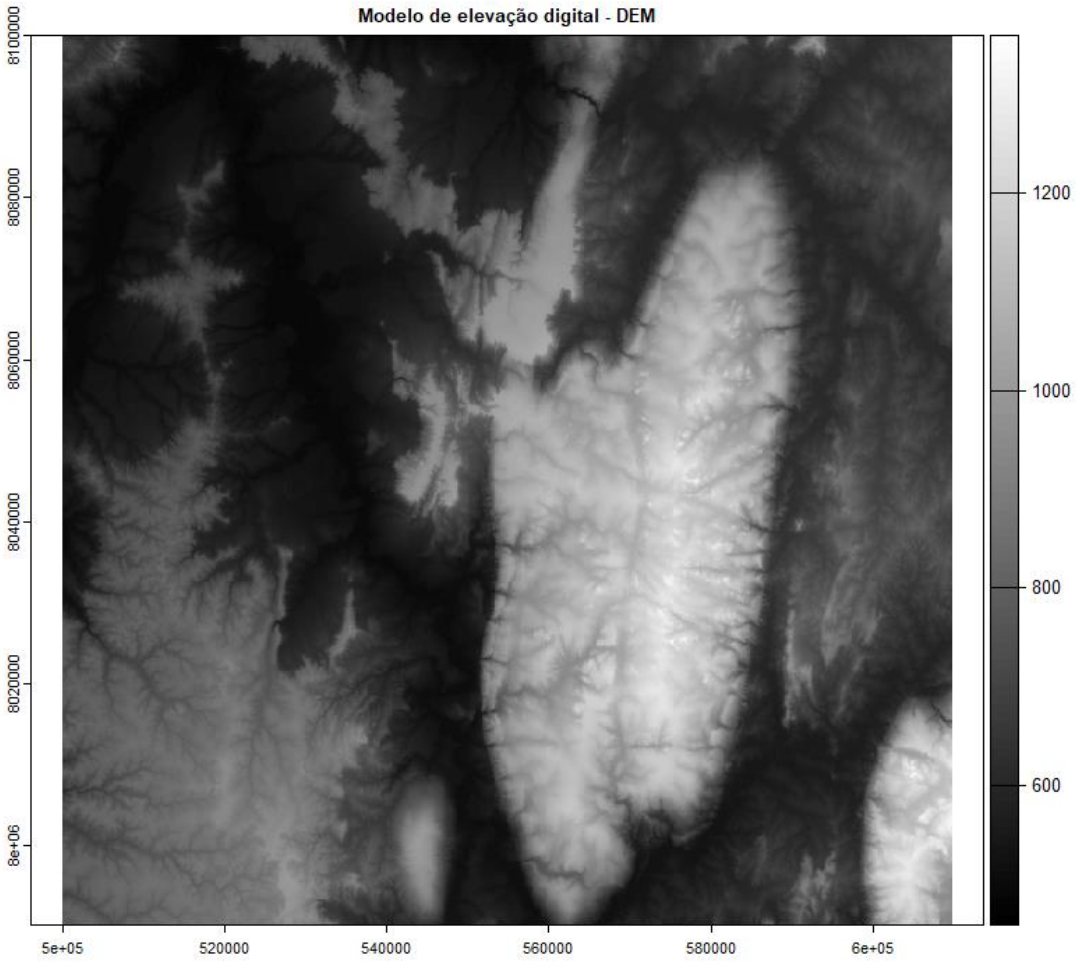
Observamos uma alta correlação de ouro para predizer o Cobre usando GWR.

9 Manipulando Modelos de Elevação

Receita - 9.1 – DEM

Carregar uma Imagem contendo um modelo digital de elevação.

```
library(terra)
dem<-rast('dem.tif')
plot(dem,col=grey(0:100/100),main='Modelo de elevação digital - DEM')
```

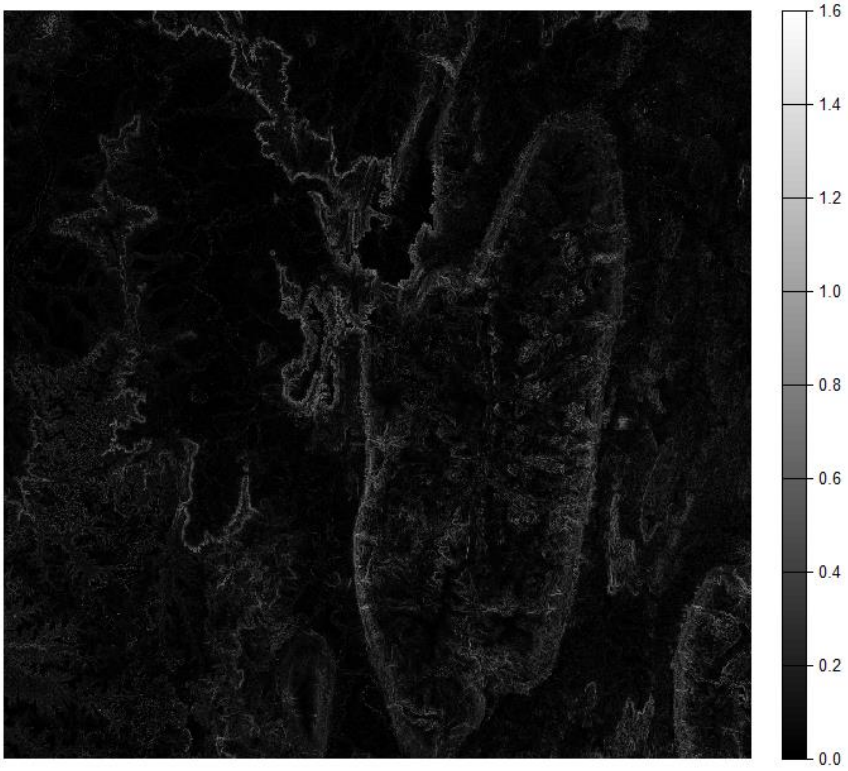


Receita - 9.2 – Extraindo imagens derivadas do DEM

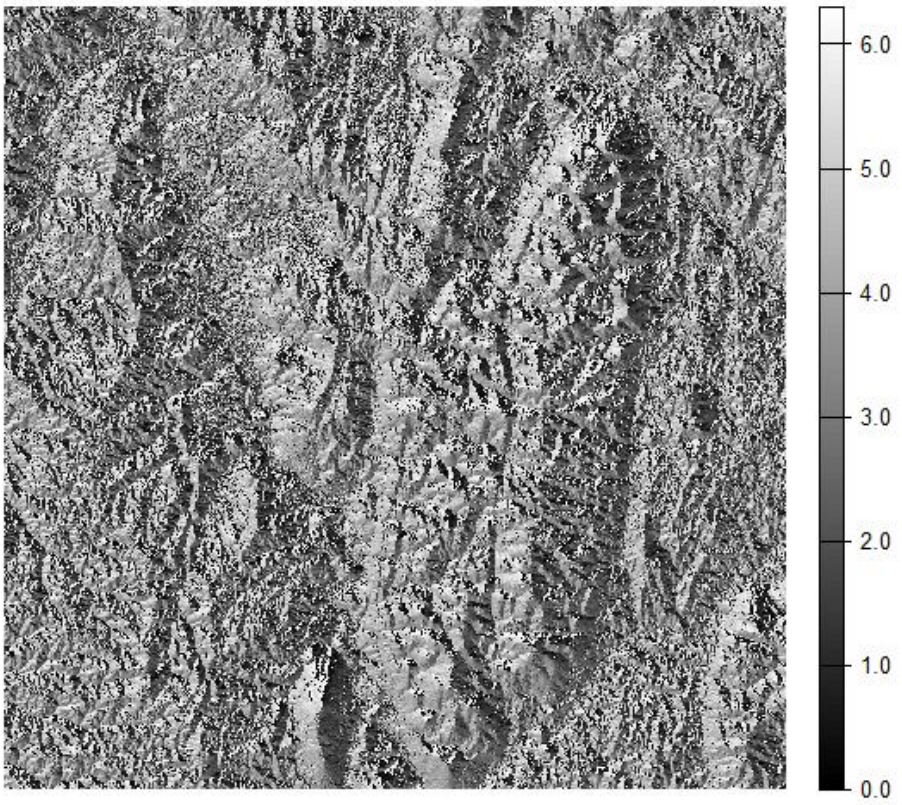
De um modelo digital de elevação é possível derivar várias outras imagens que podem ser usadas em estudos da morfologia da área, entre elas temos:

- Gradiente - ângulo que o plano inclinação faz com o plano horizontal
- Aspecto - a direção do mergulho máximo do plano inclinação com relação ao Norte
- TPI - índice de posição topográfica, é a diferença da altitude de cada célula com a elevação média de sua vizinhança
- TRI - índice de rugosidade topográfica expressa a média da diferença absoluta de elevação entre as células adjacentes
- Rugosidade - diferença entre o valor máximo e o mínimo de uma célula e suas oito células vizinhas
- Fluxo - direção de maior queda em elevação ou de menor aclive

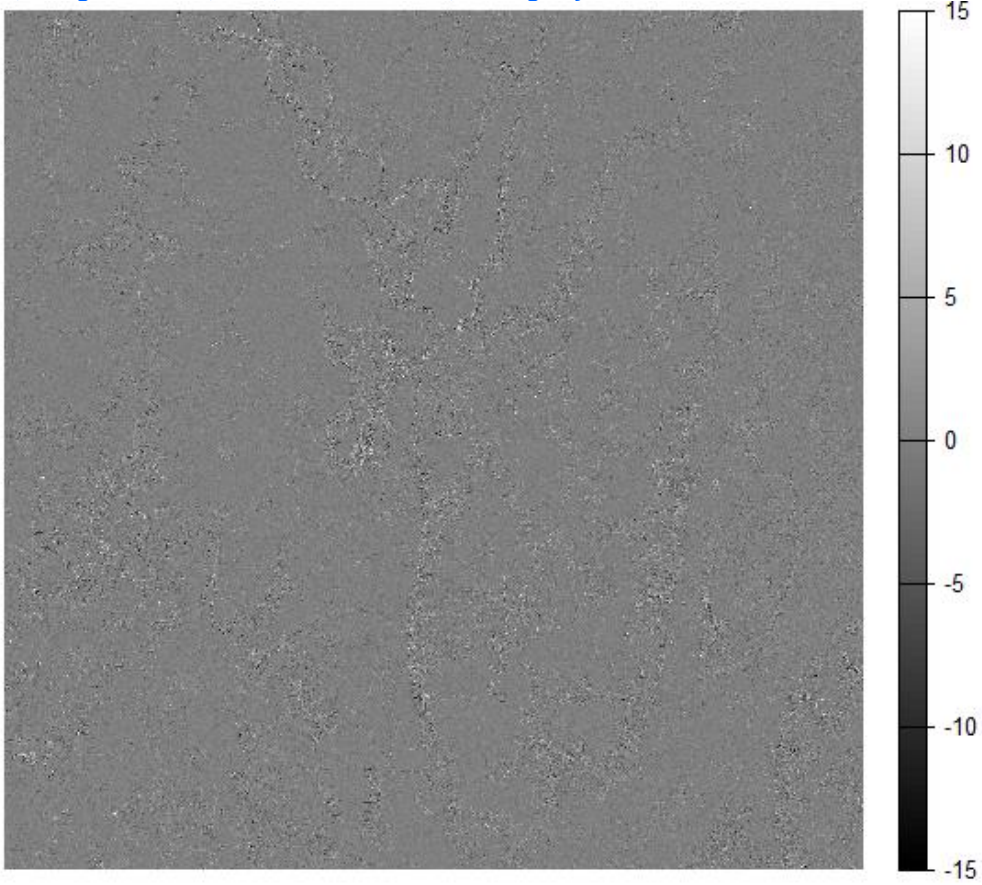
```
library(terra)
dem<-rast('dem.tif')
gradiente<-terrain(dem,v='slope',unit='radians')
aspecto<-terrain(dem,v='aspect',unit='radians')
tpi<-terrain(dem,v='TPI')
tri<-terrain(dem,v='TRI')
rugosidade<-terrain(dem,v='roughness')
fluxo<-terrain(dem,v='flowdir',unit='radians')
plot(gradiente,range=c(0,1.6),axes=FALSE,col=grey(0:100/100))
```



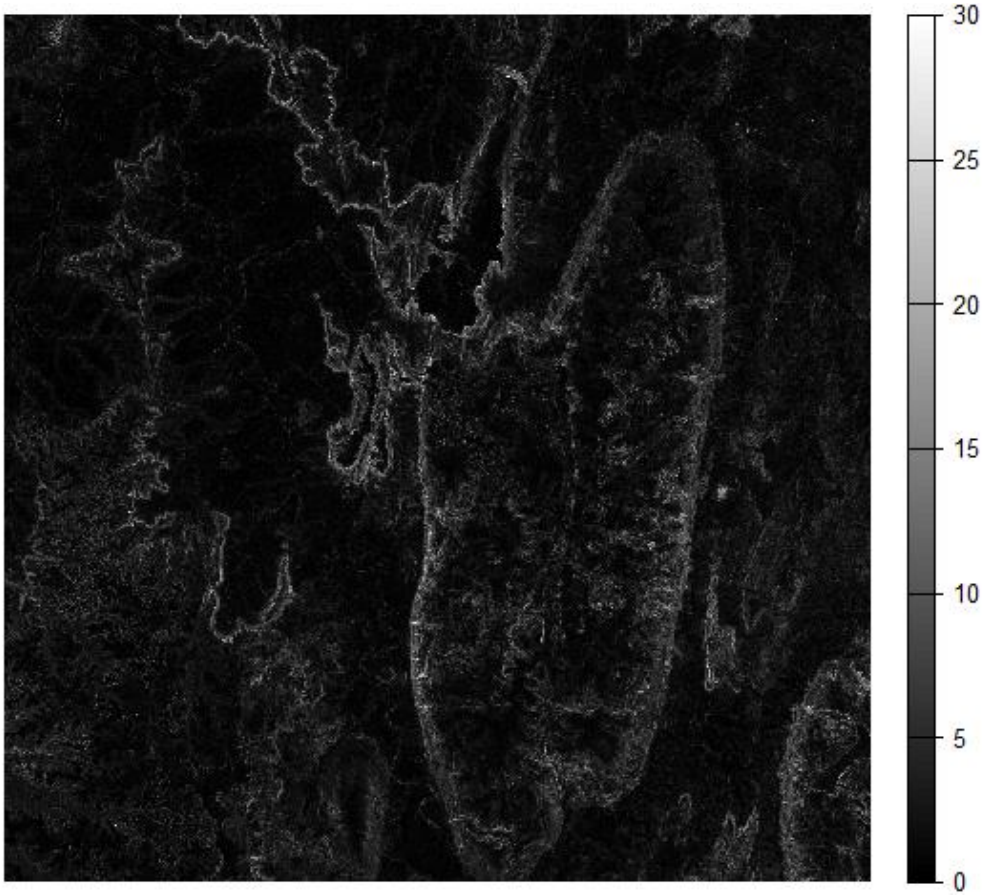
```
plot(aspecto,range=c(0,6.3),axes=FALSE,col=grey(0:100/100))
```



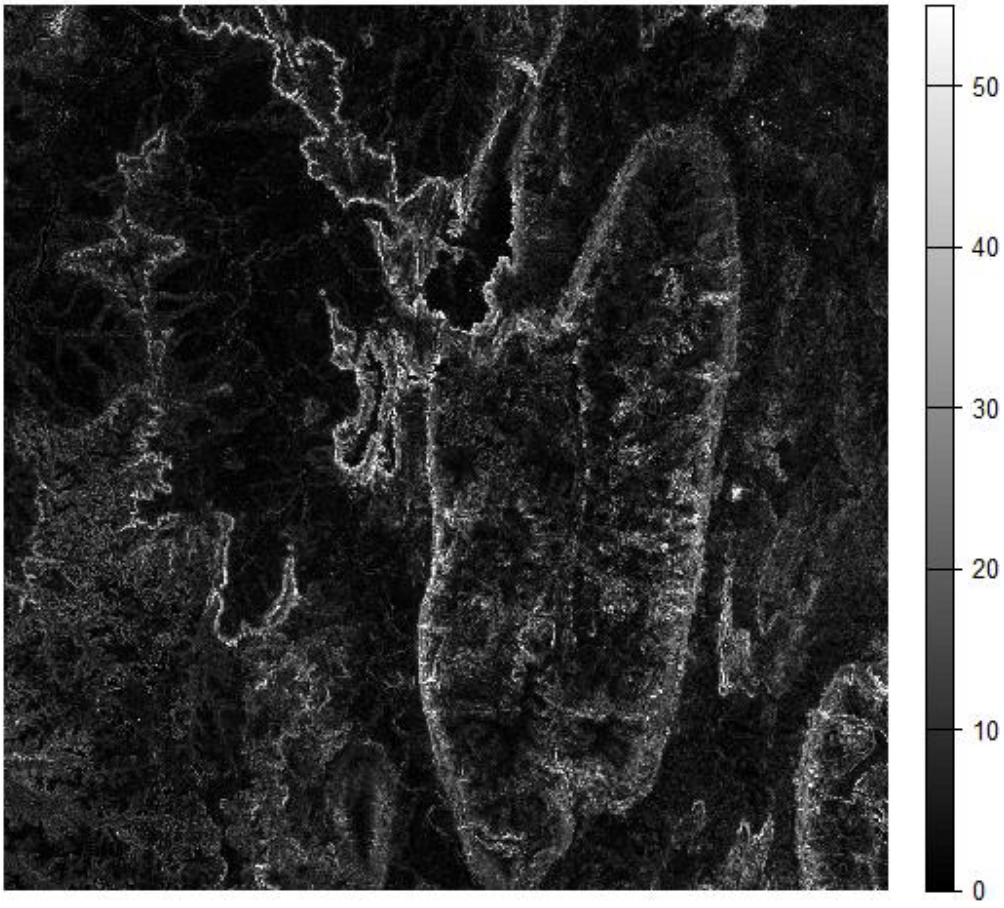

```
plot(tpi,range=c(-15,15),axes=FALSE,col=grey(0:100/100))
```



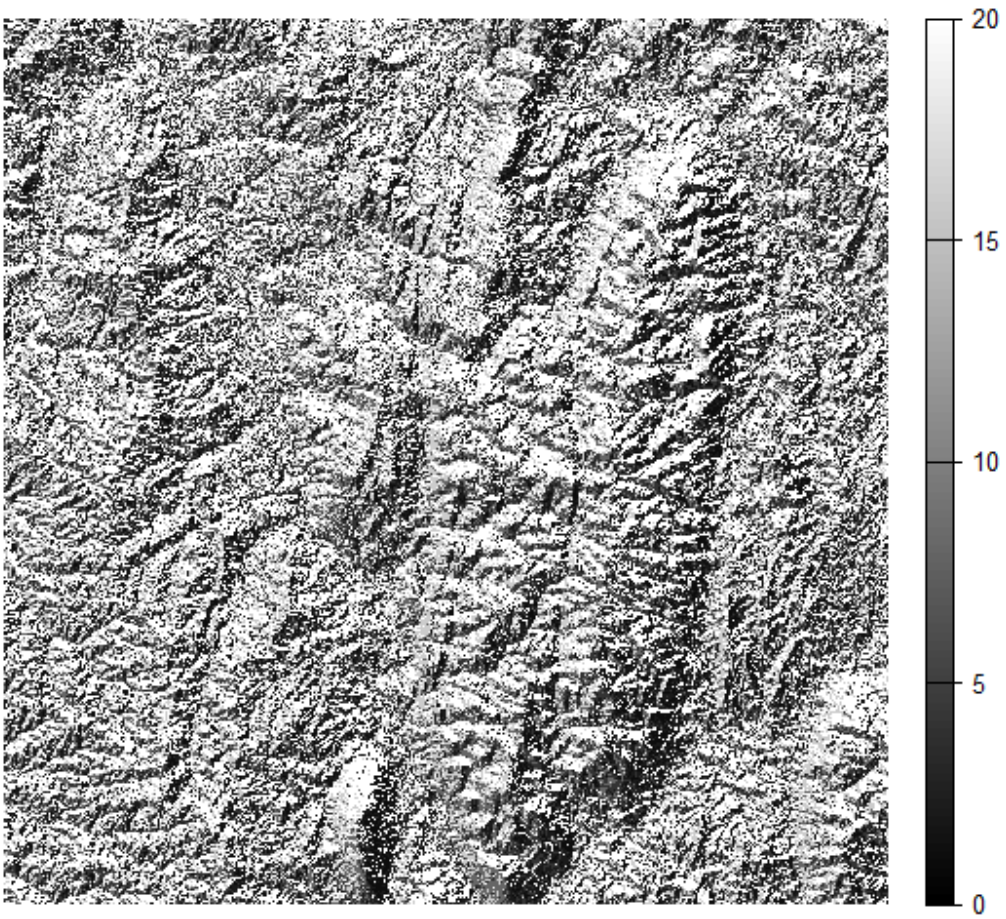
```
plot(tri,range=c(0,30),axes=FALSE,col=grey(0:100/100))
```




```
plot(rugosidade,range=c(0,55),axes=FALSE,col=grey(0:100/100))
```



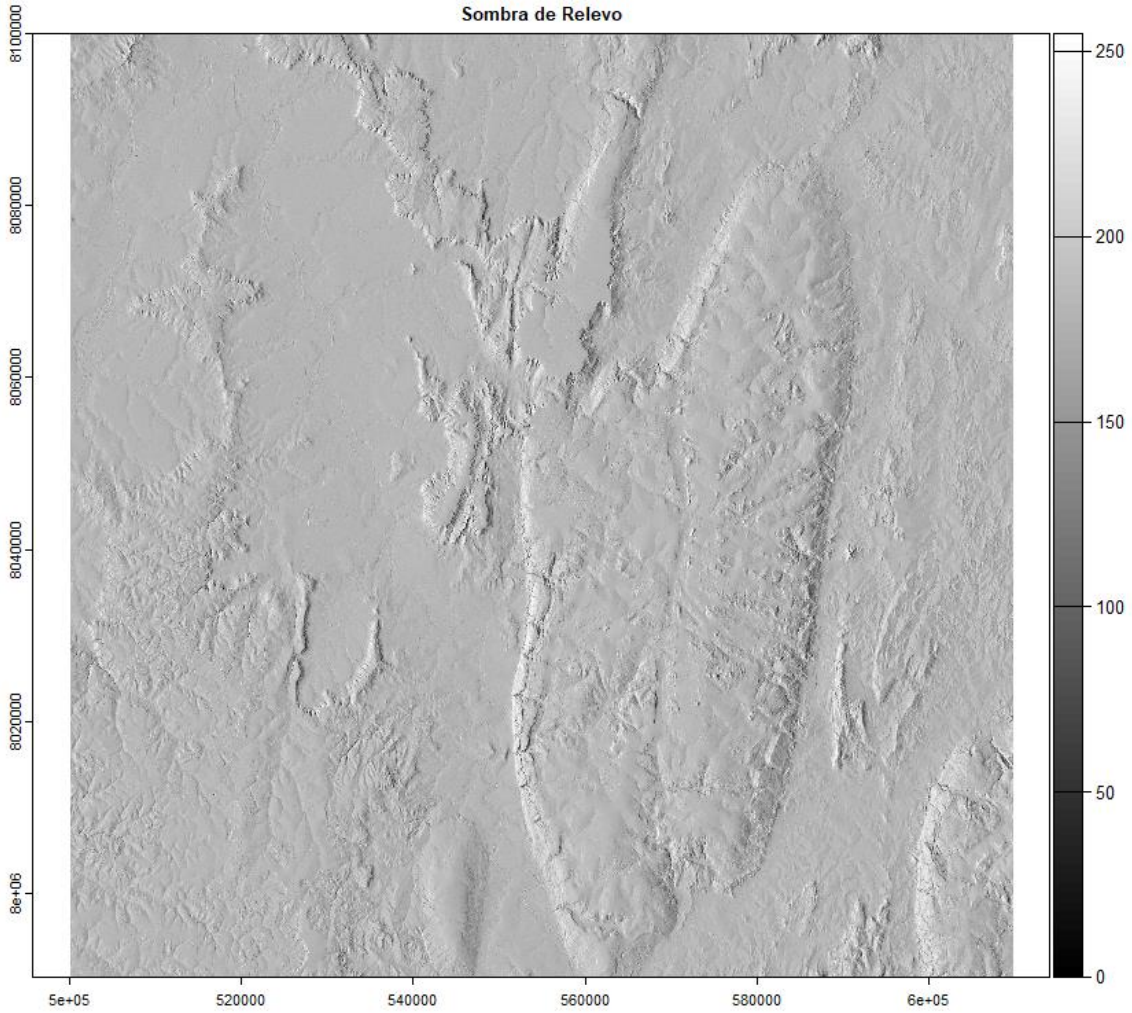
```
plot(fluxo,range=c(0,20),axes=FALSE,col=grey(0:100/100))
```



Receita - 9.3 – Sombra de Relevo (Hillshade)

Gerando sobra de relevo de um DEM usando Aspecto e gradiente e gravando como GeoTiff.

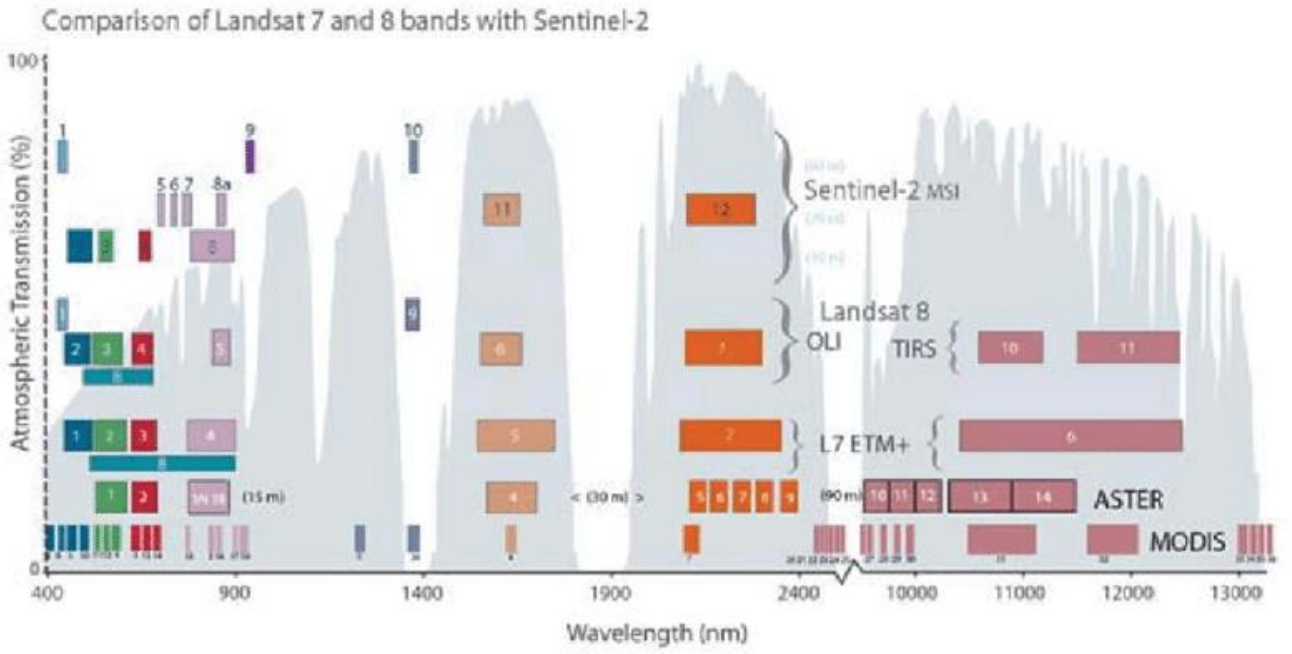
```
library(terra) #install terra using install.packages("terra")
dem<-rast('dem.tif')
gradiente<-terrain(dem,v='slope',unit='radians')
aspecto<-terrain(dem,v='aspect',unit='radians')
relevo<-shade(gradiente,aspecto,angle=45,direction=270,normalize= TRUE)
plot(relevo,col=grey(0:100/100),zlim=c(0,250),main='Sombra de Relevo')
```




```
writeRaster(relevo,'hillshade.tif',overwrite=TRUE)
```

10 Imagens de Satélites e Índices Normalizados

As faixas espectrais variam com o tipo de satélite que adquiriu a imagem. Elas se dividem basicamente em visível (azul, verde e vermelho), infravermelho muito próximo (VNIR), infravermelho próximo (NIR), infravermelho de ondas curtas (SWIR) e termal.



Vamos aqui trabalhar com imagens do tipo sentinel2 (cortesia ESA) com as bandas 2, 3, 4, 5, 6, 7, 8A, 11 e 12. Geraremos Composições coloridas:

- Visível (VIS)
- Infravermelho próximo (VNIR ou RedEdge)
- infravermelho de ondas curtas e infravermelho próximo (SWIR+NIR)

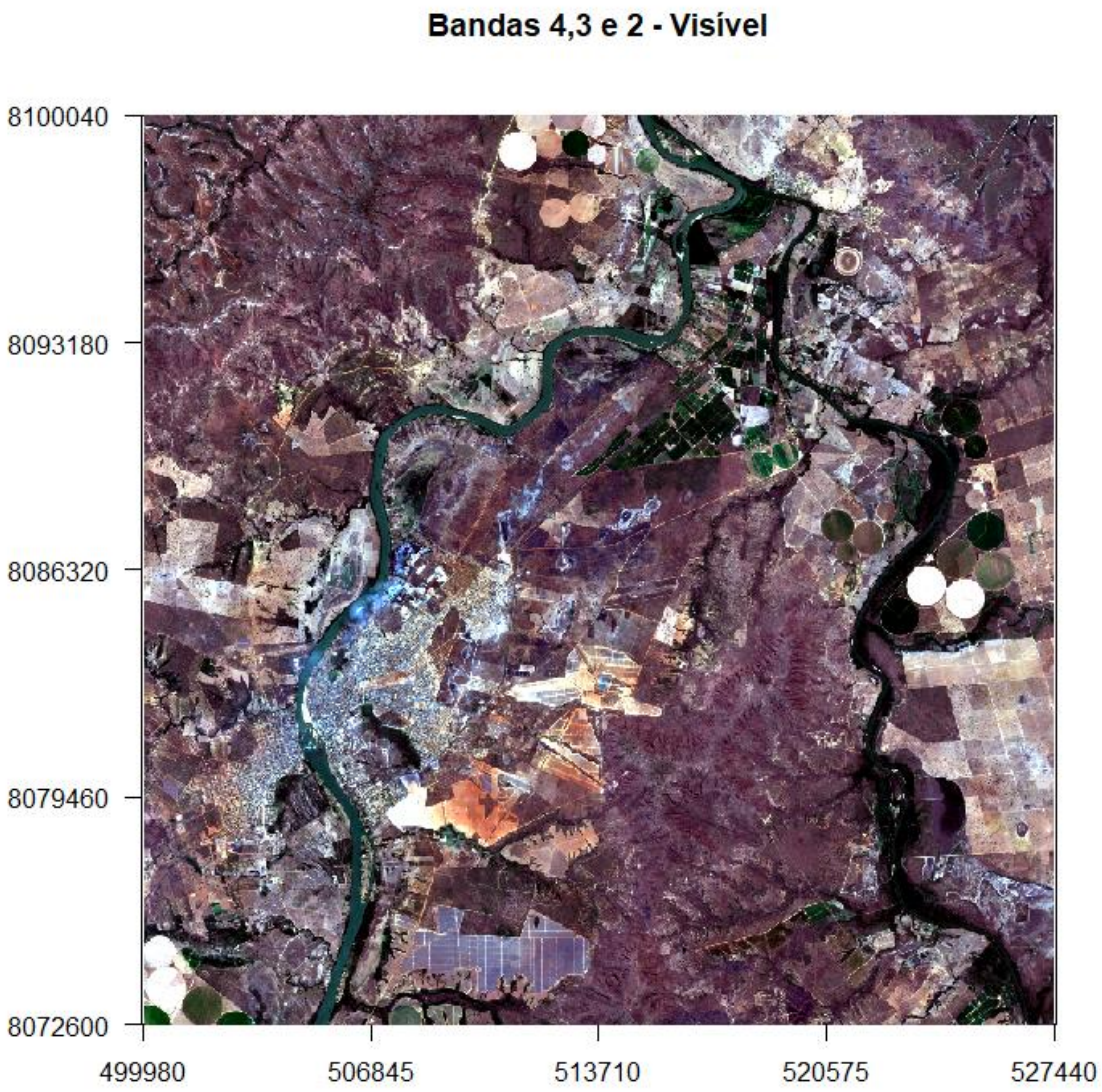
Os seguintes Índices são aqui apresentados:

- NDVI
- NDVIre1
- SAVI
- NDWI
- MNDWI
- NDMI
- NDTI
- NDBI

Receita - 10.1 – TCC Visível

Composição Visível ou TCC (True Color Composite)

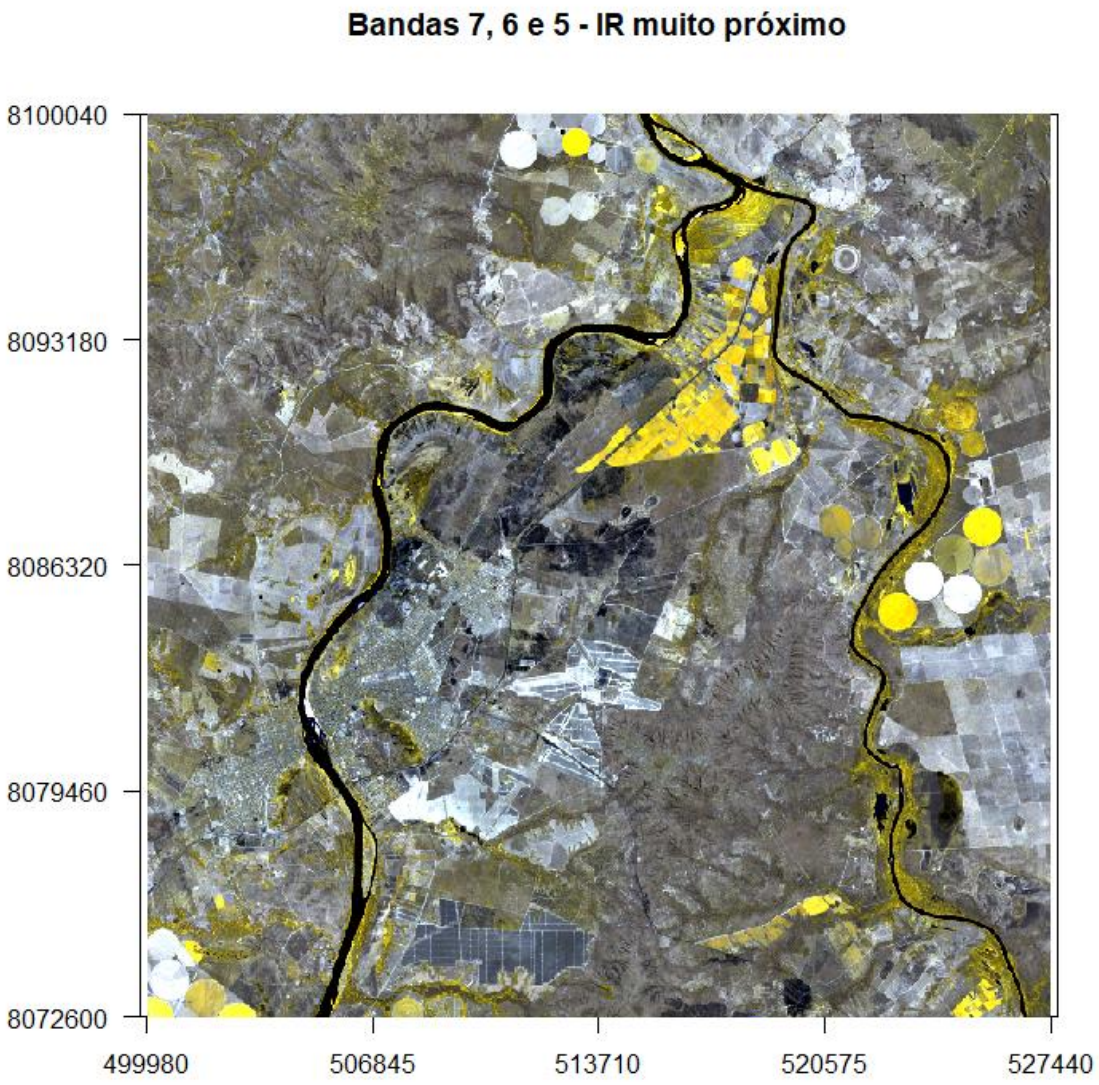
```
library(terra) #install terra using install.packages("terra")
filenames <- paste0('B0', 4:2, "_20m.jp2")
tcc4_3_2<-rast(filenames)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(tcc4_3_2,e)
plotRGB(rc,stretch='lin',axes=TRUE,main='Bandas 4,3 e 2 - Visível',
mar=c(6,6,6,6))
```



Receita - 10.2 – FCC VNIR

Composição VNIR ou Red-Edge.

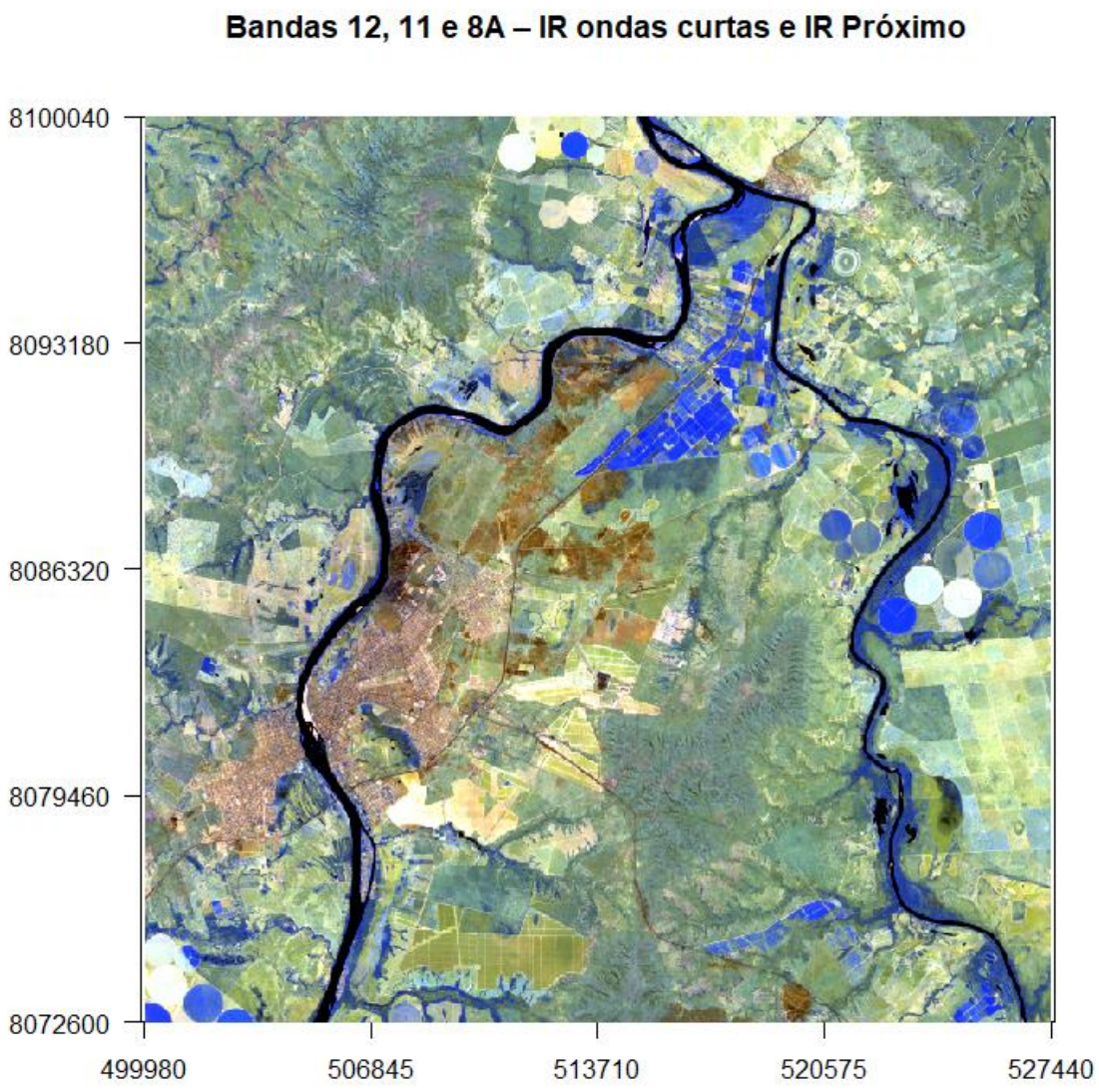
```
library(terra) #install terra using install.packages("terra")
filenames <- paste0('B0', 7:5, "_20m.jp2")
fcc7_6_5<- rast(filenames)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(fcc7_6_5,e)
plotRGB(rc,stretch='lin',axes=TRUE,main='Bandas 7, 6 e 5 - IR muito próximo' ,
mar=c(6,6,6,6))
```



Receita - 10.3 – FCC SWIR+NIR

Composição SWIR+NIR.

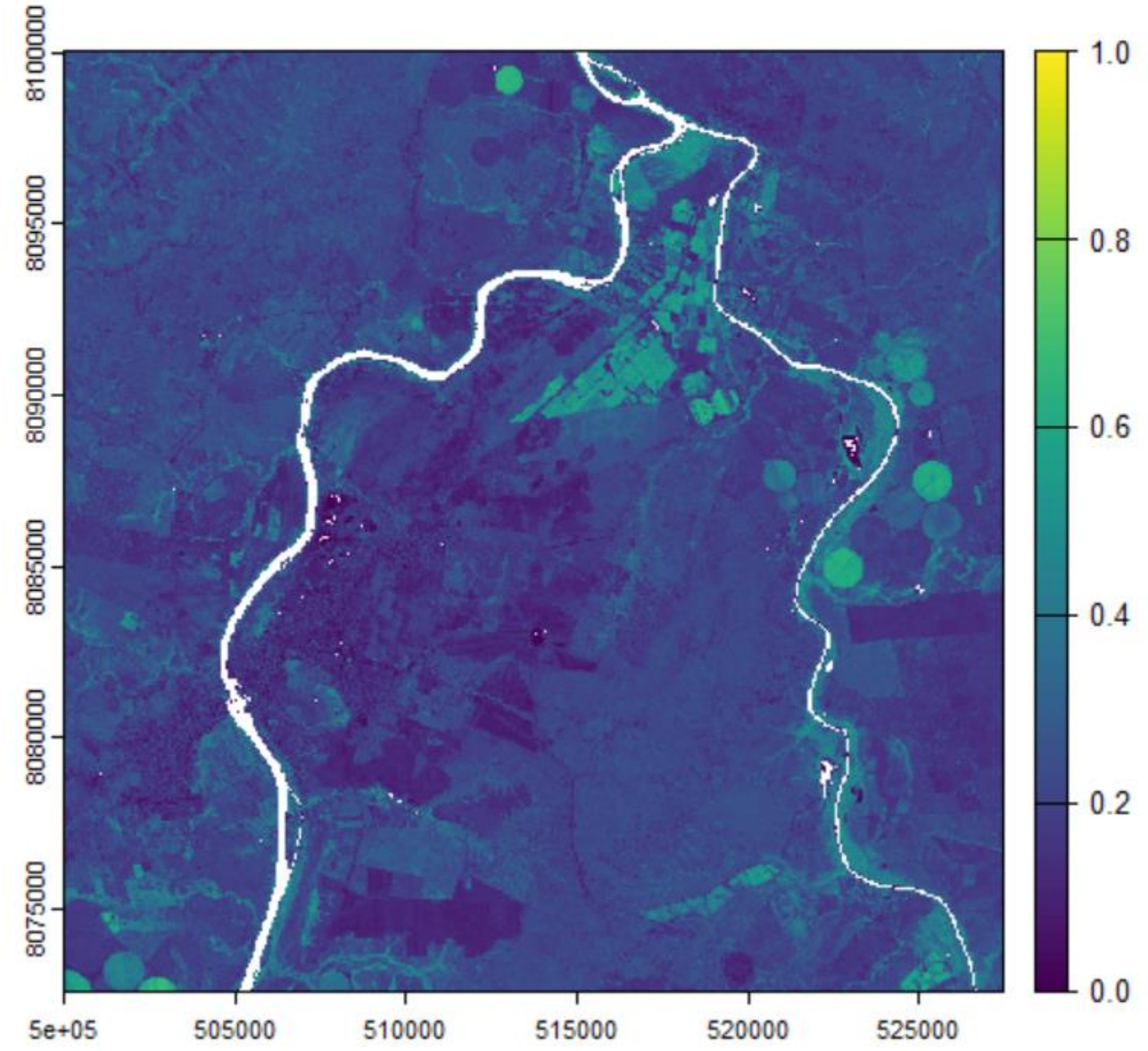
```
library(terra) #install terra using install.packages("terra")
filenames <- paste0('B',c('12','11','8A'), "_20m.jp2")
fcc12_11_8a<- rast(filenames)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(fcc12_11_8a,e)
plotRGB(rc,stretch='lin',axes=TRUE,main='Bandas 12, 11 e 8A – IR ondas curtas e
IR Próximo' , mar=c(6,6,6,6))
```



Receita - 10.4 – NDVI

O NDVI (Normalized Difference Vegetation Index).

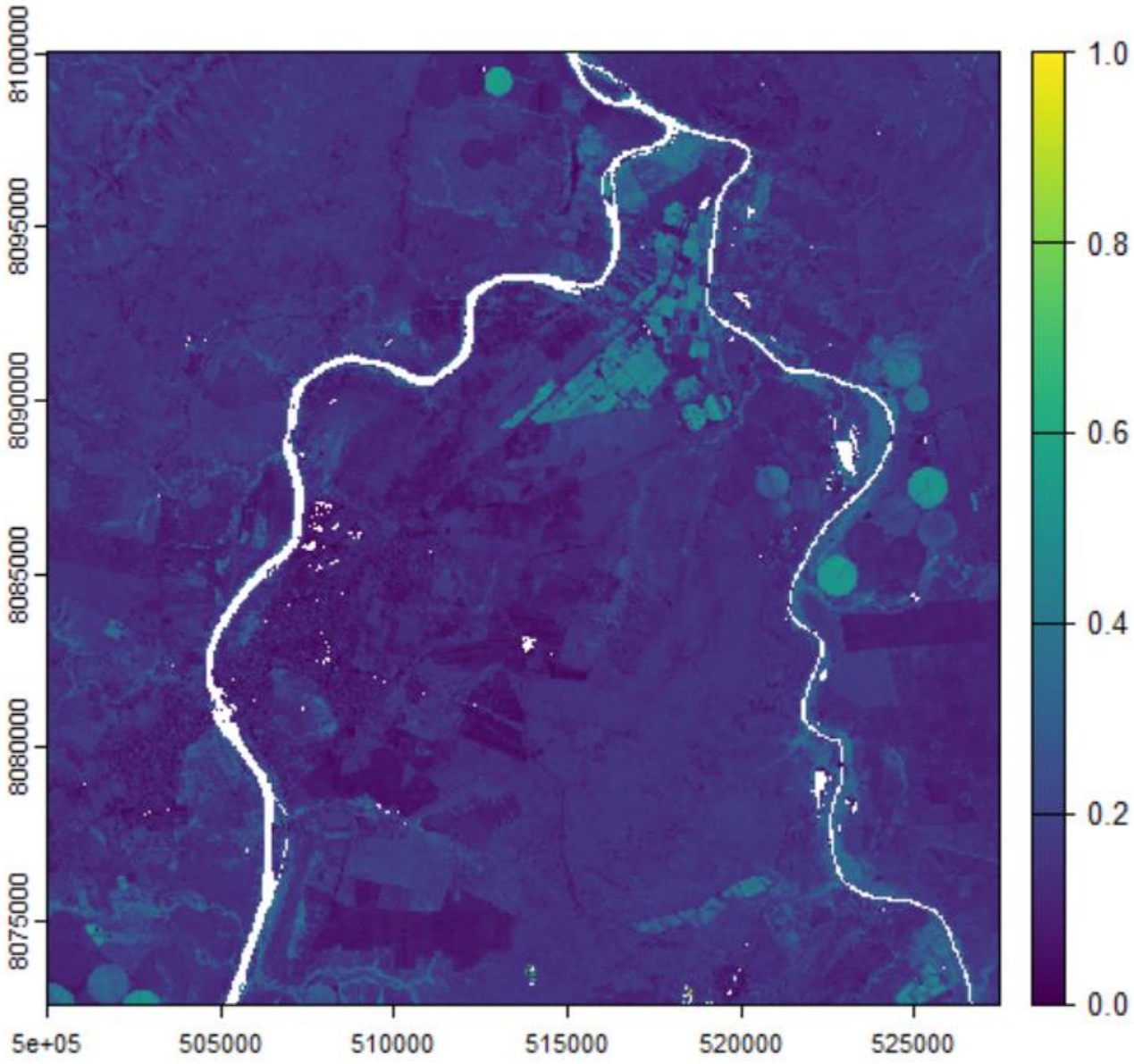
```
library(terra) #install terra using install.packages("terra")
nir<-rast('B8A_20m.jp2')
vermelho<-rast('B04_20m.jp2')
ndvi<-(nir-vermelho)/(nir+vermelho)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(ndvi,e)
plot(rc,axes=TRUE ,range=c(0,1))
```



Receita - 10.5 – NDVIre1

O NDVIre1 (red-edge-based Normalized Difference Vegetation Index).

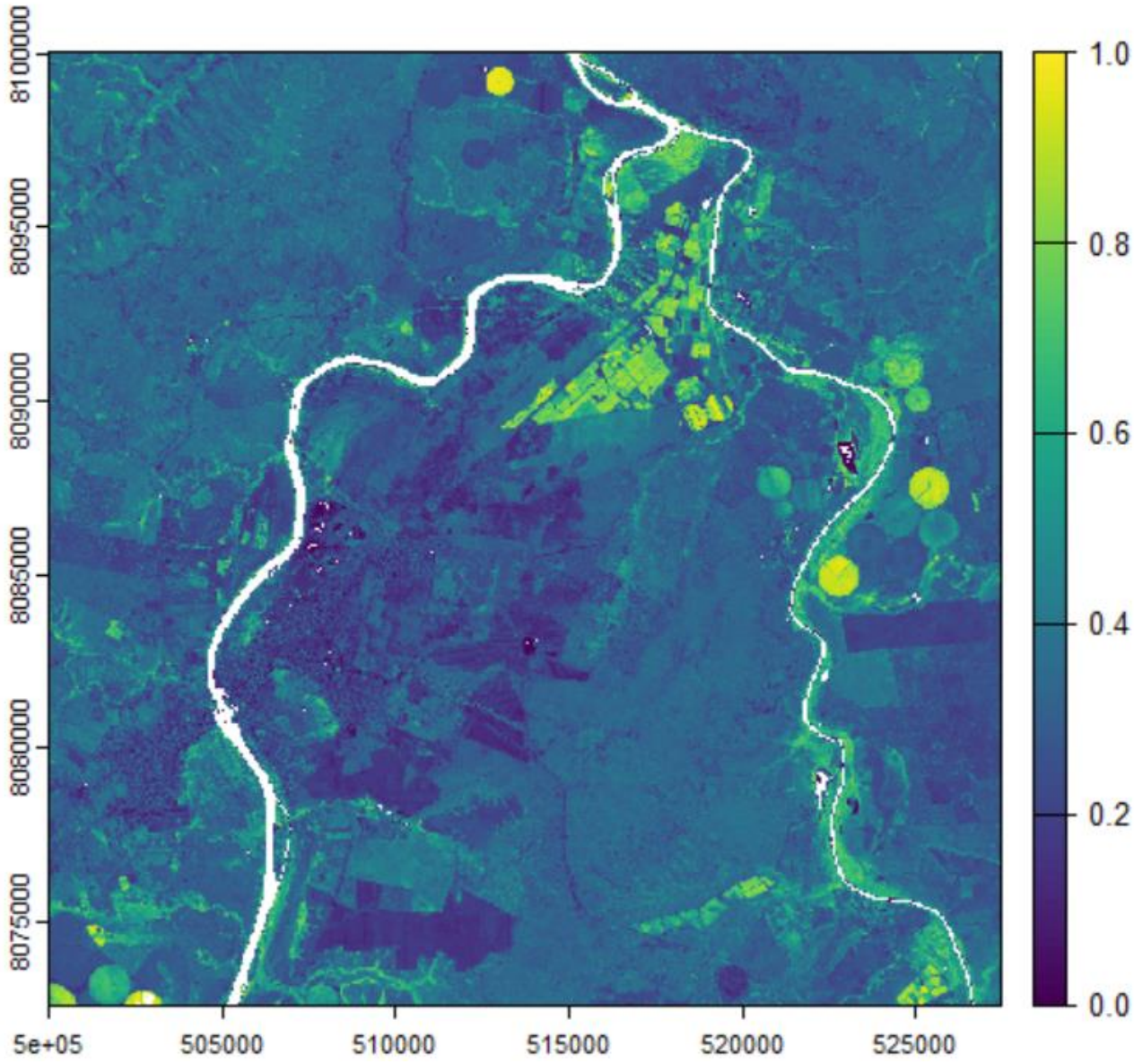
```
library(terra) #install terra using install.packages("terra")
vnir<-rast('B05_20m.jp2')
nir<-rast('B8A_20m.jp2')
ndvire<-(nir-vnir)/(nir+vnir)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(ndvire,e)
plot(rc,axes=TRUE,range=c(0,1))
```



Receita - 10.6 – SAVI

O SAVI (Soil Adjusted Vegetation Index).

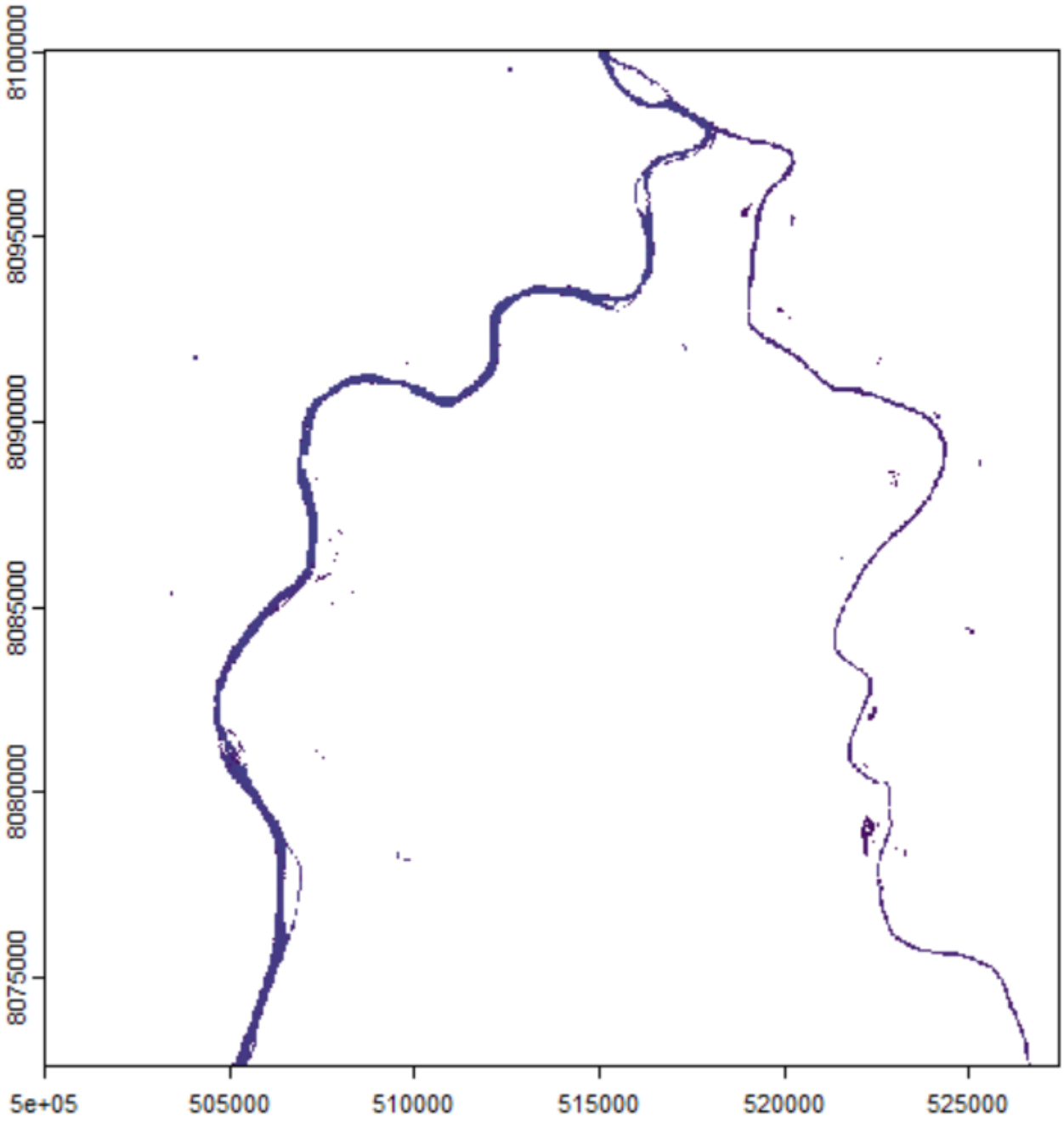
```
library(terra) #install terra using install.packages("terra")
nir<-rast('B8A_20m.jp2')
vermelho<-rast('B04_20m.jp2')
savi<-(nir-vermelho)/(nir+vermelho+0.5)*1.5
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(savi,e)
plot(rc,axes=TRUE,range=c(0,1))
```



Receita - 10.7 – NDWI

O NDWI (Normalized Difference Water Index).

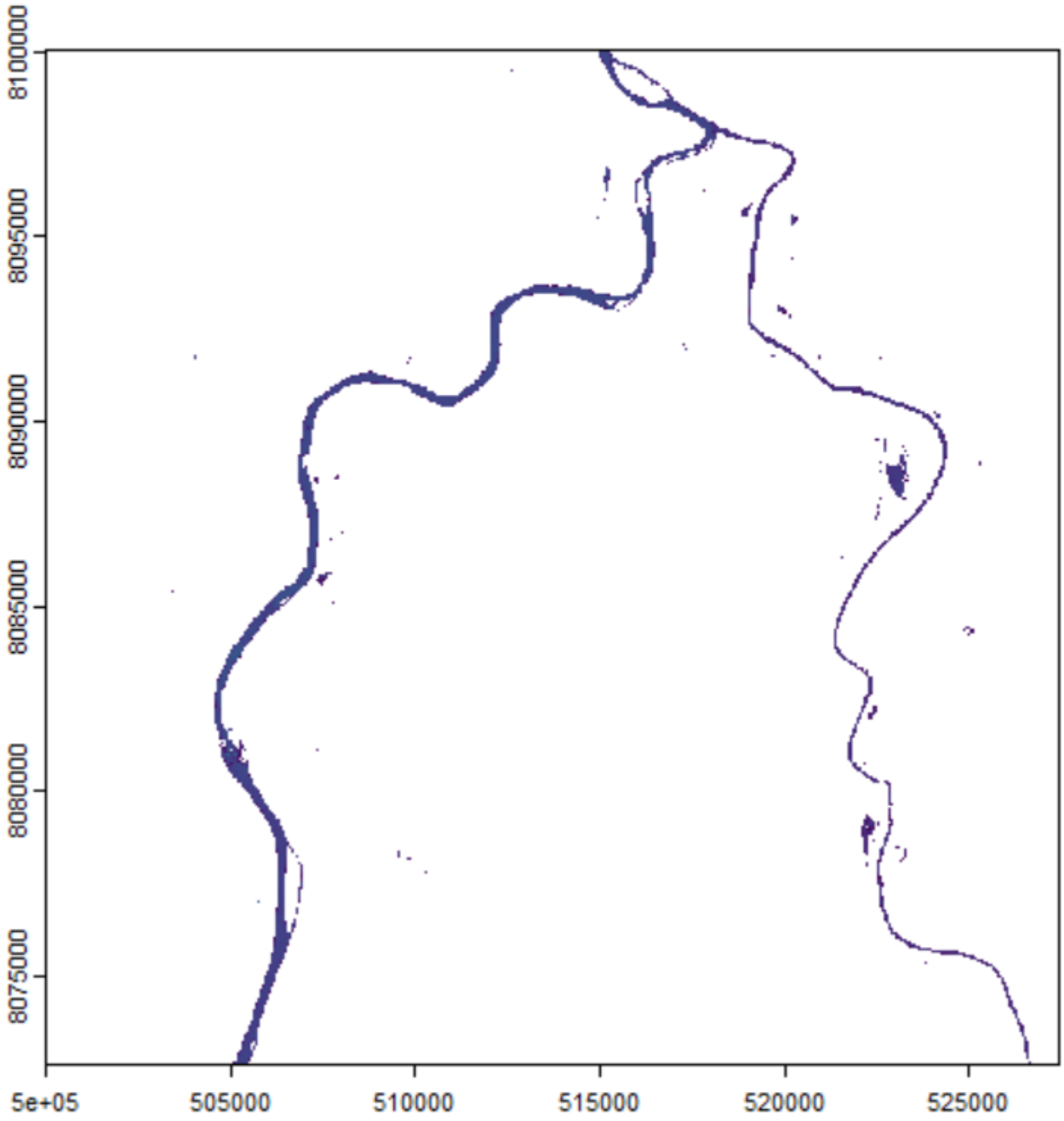
```
library(terra) #install terra using install.packages("terra")
green<-rast('B03_20m.jp2')
nir<-rast('B8A_20m.jp2')
ndwi<-(green-nir)/(green+nir)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(ndwi,e)
plot(rc,axes=TRUE,legend=FALSE,range=c(0,1))
```



Receita - 10.8 – MNDWI

O MNDWI (Modified Normalized Difference Water Index).

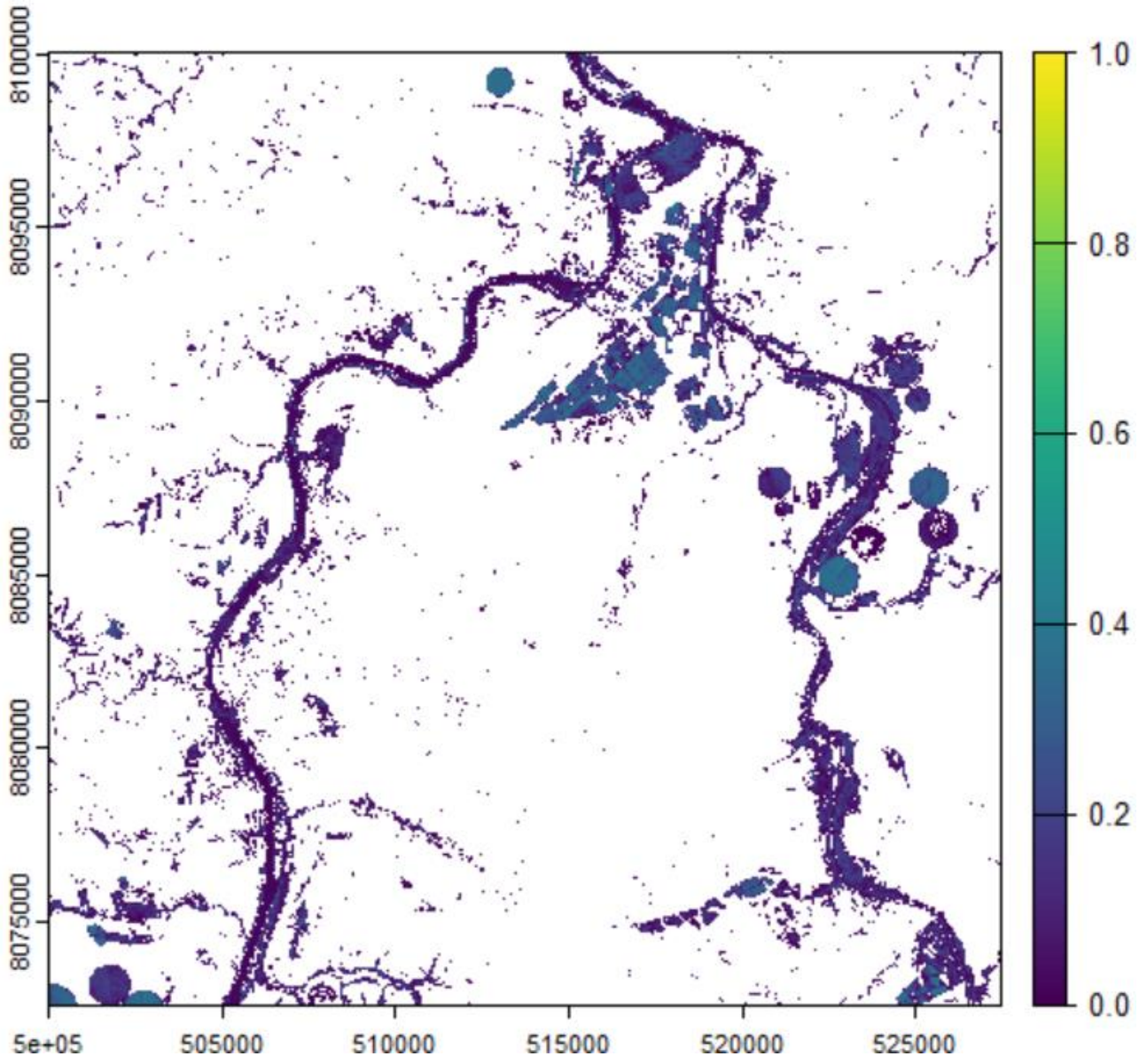
```
library(terra) #install terra using install.packages("terra")
green<-rast('B03_20m.jp2')
swir<-rast('B11_20m.jp2')
mndwi<-(green-swir)/(green+swir)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(mndwi,e)
plot(rc,axes=TRUE,legend=FALSE,range=c(0,1))
```



Receita - 10.9 – NDMI

O NDMI (Normalized Difference Moisture Index).

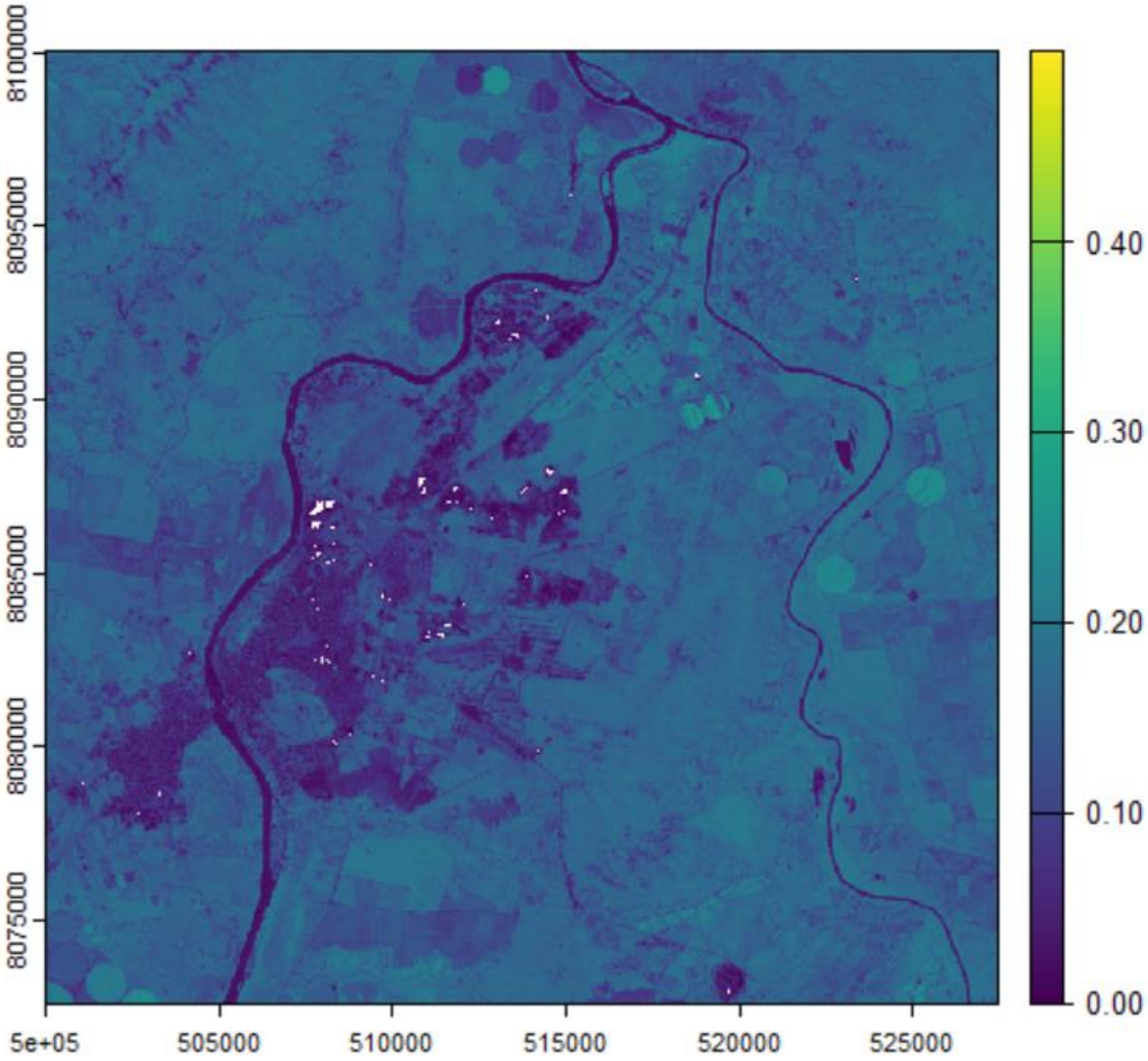
```
library(terra) #install terra using install.packages("terra")
nir<-rast('B8A_20m.jp2')
swir<-rast('B11_20m.jp2')
ndmi<-(nir-swir)/(nir+swir)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(ndmi,e)
plot(rc,axes=TRUE,range=c(0,1))
```



Receita - 10.10 – NDTI

O NDTI (Normalized Difference Tillage Index) .

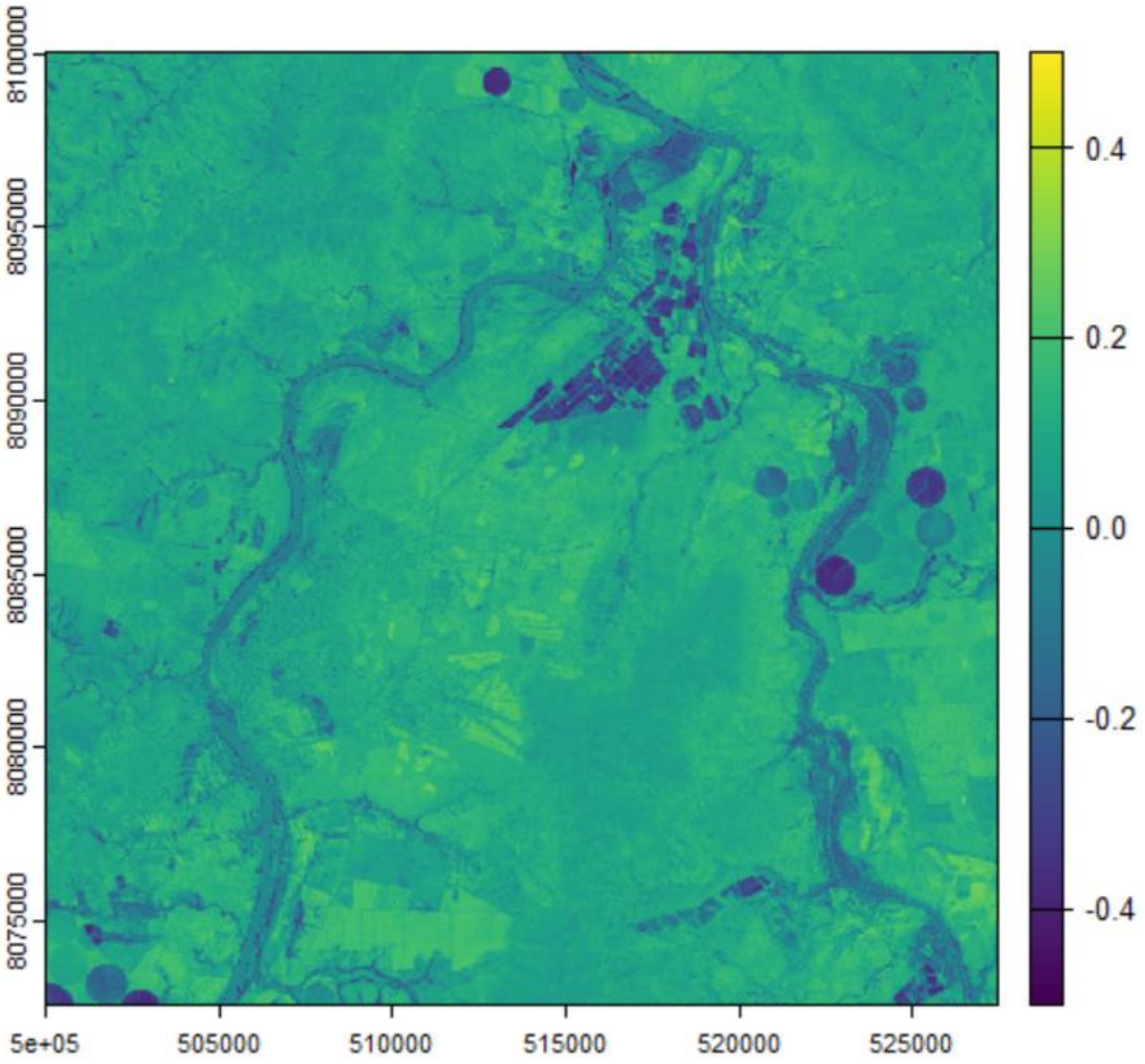
```
library(terra) #install terra using install.packages("terra")
swir1<-rast('B11_20m.jp2')
swir2<-rast('B12_20m.jp2')
ndti<-(swir1-swir2)/(swir1+swir2)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(ndti,e)
plot(rc,axes=TRUE,range=c(0,0.5))
```



Receita - 10.11 – NDBI

O NDBI (Normalized Difference Built-up Index) .

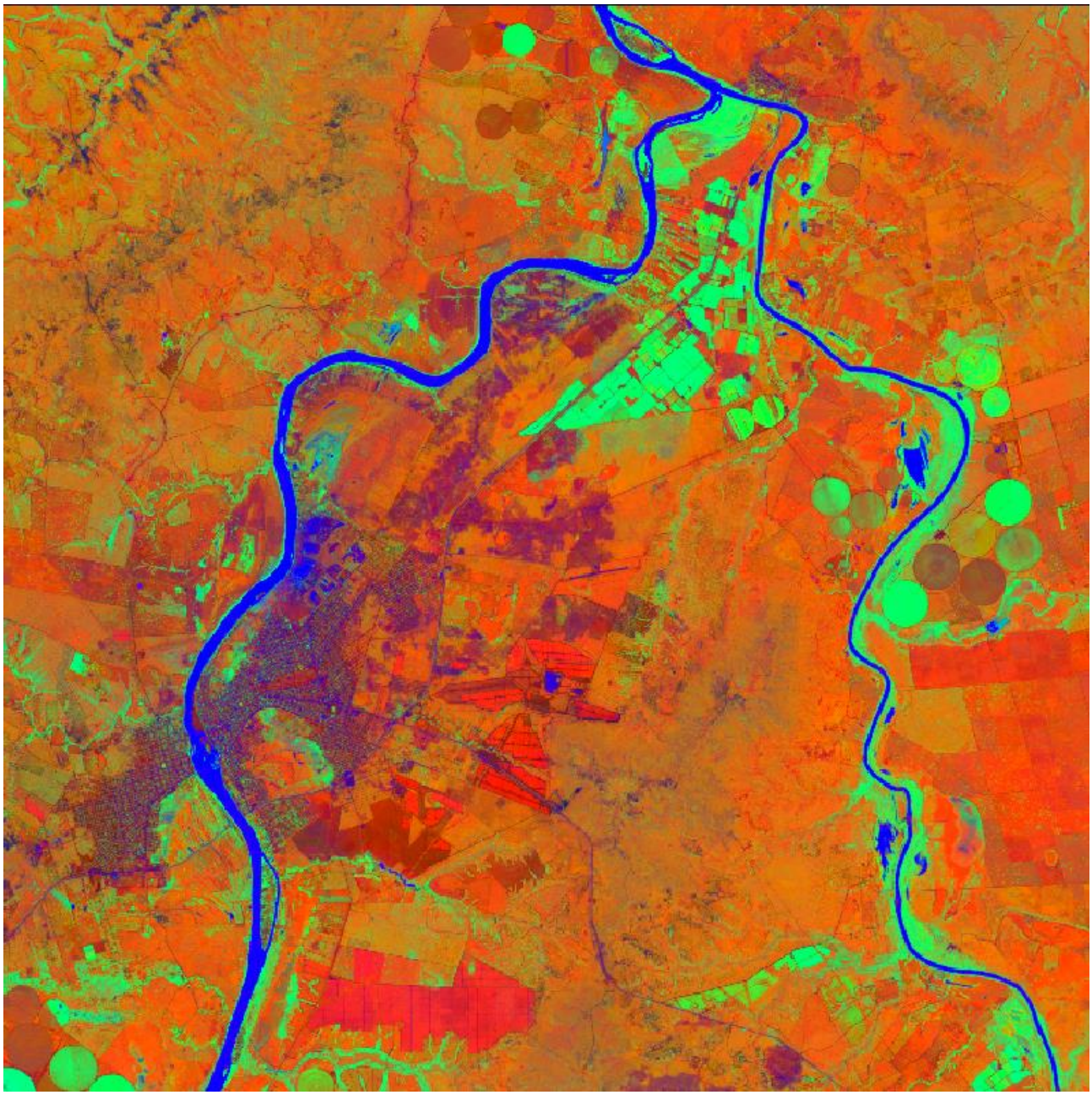
```
library(terra) #install terra using install.packages("terra")
nir<-rast('B8A_20m.jp2')
swir<-rast('B11_20m.jp2')
ndbi<-(swir-nir)/(swir+nir)
e <- ext(499980, 527430, 8072590,8100040)
rc<-crop(ndbi,e)
plot(rc,axes=TRUE,range=c(-0.5,0.5))
```



Receita - 10.12 – Composição RGB com Índices

Criando imagem RGB FCC de classificação com os índices MNDWI (Blue), SAVI (Green) e NDBI+NDTI (Red).

```
library(terra) #install terra using install.packages("terra")
nir<-rast('B8A_20m.jp2')
vermelho<-rast('B04_20m.jp2')
savi<-(nir-vermelho)/(nir+vermelho+0.5)*1.5
green<-rast('B03_20m.jp2')
swir<-rast('B11_20m.jp2')
mndwi<-(green-swir)/(green+swir)
swir1<-rast('B11_20m.jp2')
swir2<-rast('B12_20m.jp2')
ndti<-(swir1-swir2)/(swir1+swir2)
nir<-rast('B8A_20m.jp2')
swir<-rast('B11_20m.jp2')
ndbi<-(swir-nir)/(swir+nir)
e <- ext(499980, 527430, 8072590,8100040)
rgb<-rast(list(ndbi+ndti,savi,mndwi))
rcrgb<-crop(rgb,e)
plotRGB(rcrgb,stretch='lin')
writeRaster(rcrgb, indexes.tif', overwrite=TRUE)
```



11 Processamentos Avançados de Imagens de Satélite

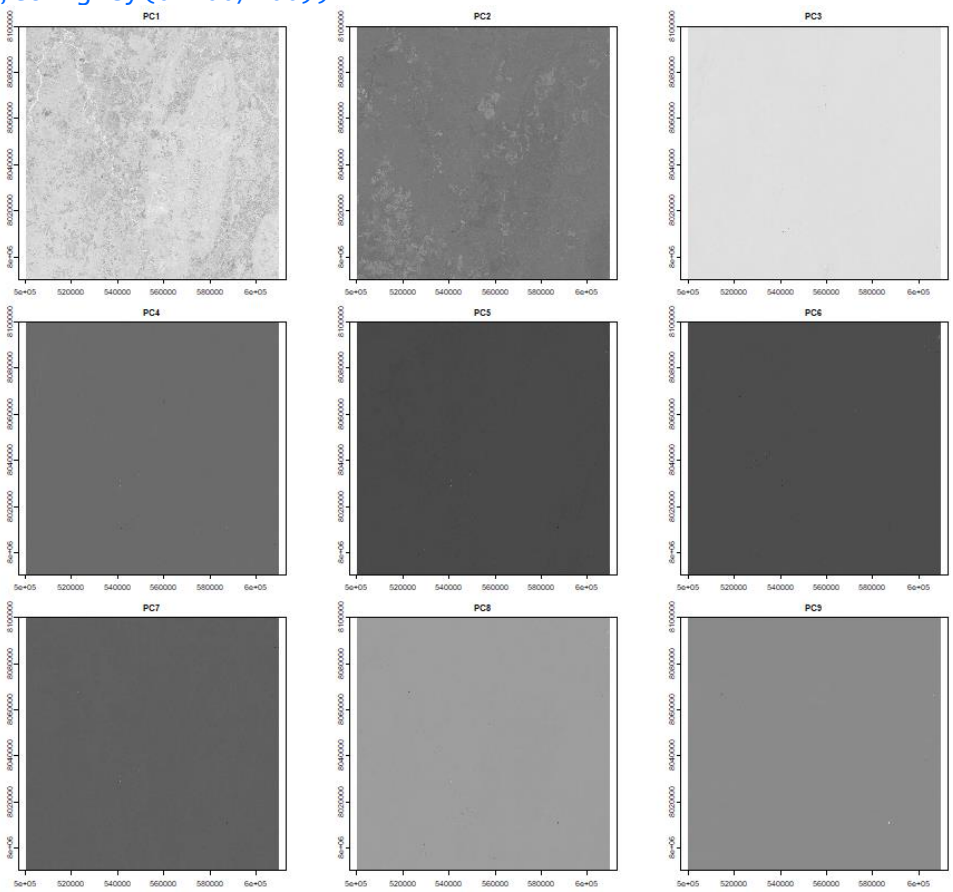
Receita - 11.1 – PCA

Gerando uma Análise de Componentes Principais (PCA). Primeiro carregamos todas a bandas (com mesma resolução) que trabalharemos na análise. Carregar as bandas 2,3,4 em baixa resolução e as bandas 5, 6, 7, 8A, 11 e 12 dentro de um objeto SpatRaster e executar o PCA.

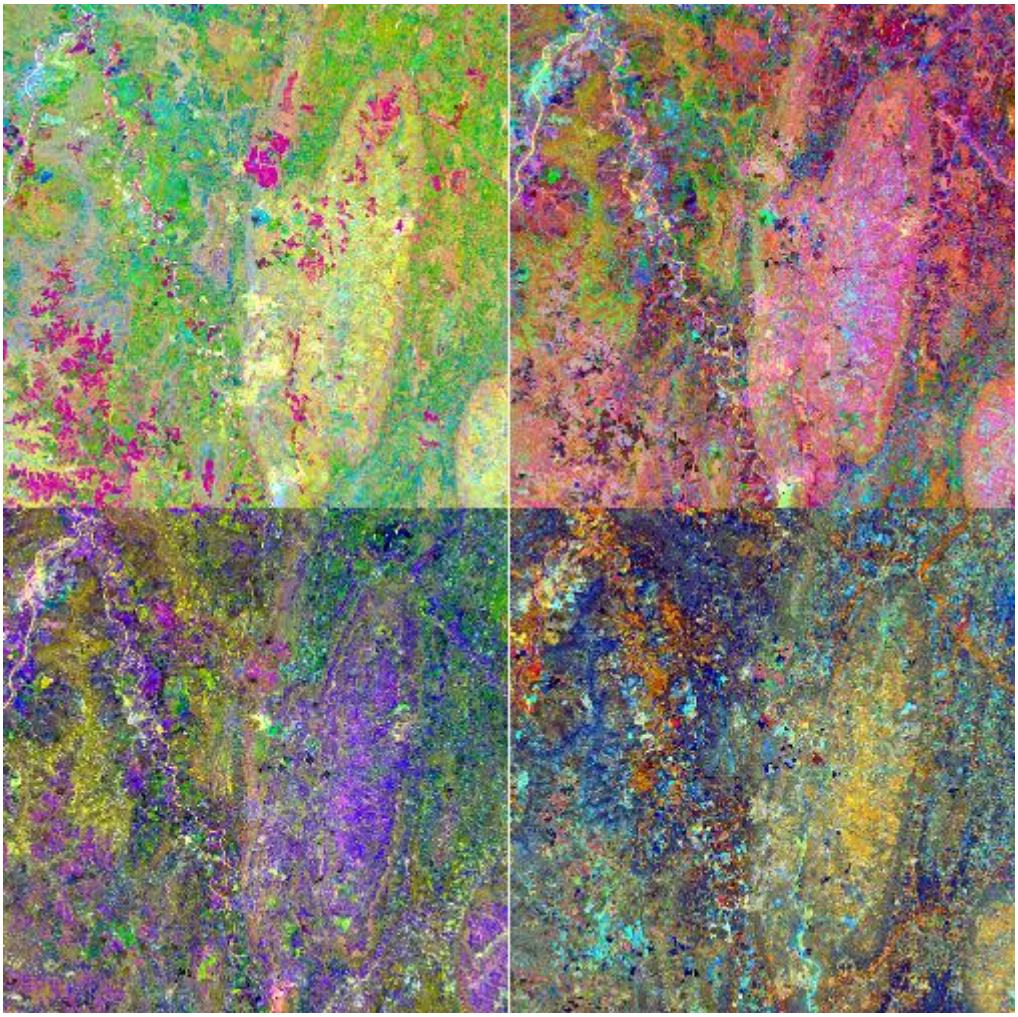
```
library(terra)
rlis<-c("B02_20m.jp2","B03_20m.jp2","B04_20m.jp2","B05_20m.jp2","B06_20m.jp2",
"B07_20m.jp2","B8A_20m.jp2","B11_20m.jp2","B12_20m.jp2")
rlis2<-c("b2","b3","b4","b5","b6","b7", "b8a","b11","b12")
sentinel<-rast(rlis)
names(sentinel)<-rlis2
set.seed(1)
amostra <- spatSample(sentinel, 300000,method="random")
pca <- prcomp(amostra, scale = TRUE)
rm(amostra)
pca
```

```
Standard deviations (1, .., p=9):
[1] 2.5013165 1.3679497 0.6039902 0.4215913 0.3048038 0.2968446 0.2802938 0.2076735 0.1639242

Rotation (n x k) = (9 x 9):
           PC1         PC2          PC3         PC4          PC5          PC6          PC7          PC8          PC9
b2  -0.3162025 -0.2320622 -0.815739281 -0.4095477 -0.08747453  0.04109089 -0.05010989  0.031012661 -0.01379107
b3  -0.3687928 -0.1370622 -0.193331645  0.5862211  0.45532814 -0.18406493  0.45459507  0.126958259 -0.01395845
b4  -0.3770770 -0.1772968  0.074984493  0.2998913 -0.30954832 -0.07939164 -0.34219088 -0.263856469  0.66481823
b5  -0.3875970 -0.0605321  0.105833744  0.2603263 -0.43205877 -0.10680666 -0.25781674  0.007790049 -0.70888736
b6  -0.3045409  0.4373218 -0.009146478  0.1081630 -0.12972038  0.78848093  0.13684802  0.201314573  0.08023160
b7  -0.2661544  0.5173791 -0.015049521 -0.1031059  0.48539072 -0.24427509 -0.56828982  0.179660082  0.01085614
b8a -0.2684434  0.5145304  0.043052373 -0.2538508 -0.20887038 -0.35783075  0.47315466 -0.448271831  0.02025474
b11 -0.3540186 -0.2316913  0.415203567 -0.4004747 -0.10396148 -0.17407037  0.20126868  0.619399635  0.15146809
b12 -0.3325883 -0.3389823  0.325286320 -0.2894144  0.44233833  0.32711634 -0.03468634 -0.505796451 -0.15875327
```

```
pcis<-predict(sentinel,pca,filename='pcis.tif',overwrite=TRUE)
plot(pcis,col=grey(0:100/100))
```



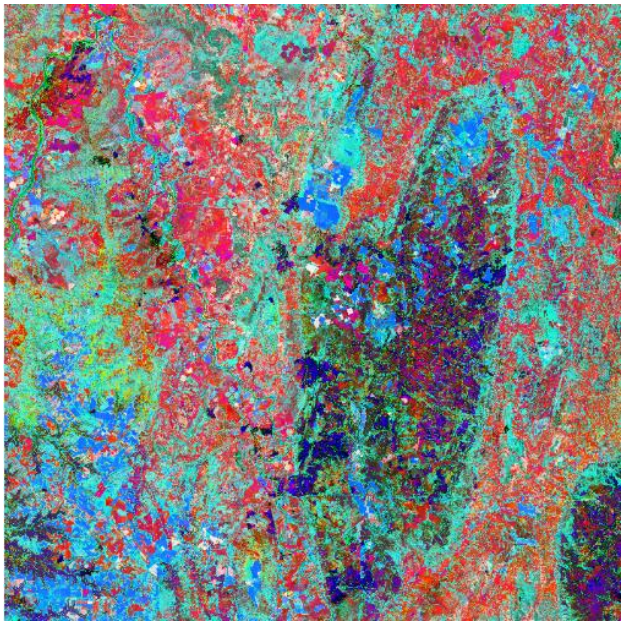
```
dev.off()
par(mfrow=c(2,2))
plotRGB(pcis,r=1,g=2,b=3,stretch='lin')
plotRGB(pcis,r=1,g=3,b=5,stretch='lin')
plotRGB(pcis,r=3,g=4,b=5,stretch='lin')
plotRGB(pcis,r=5,g=6,b=7,stretch='lin')
```



Receita - 11.2 – IHS

Decomposição IHS a partir de um conjunto RGB. Bandas 12, 11 e 8A .

```
library(terra)
fcc.rgb<-rast(c('B12_20m.jp2','B11_20m.jp2','B8A_20m.jp2'))
Nor<-fcc.rgb/32767
Nor
class       : SpatRaster
dimensions  : 5490, 5490, 3  (nrow, ncol, nlyr)
resolution  : 20, 20  (x, y)
extent      : 499980, 609780, 7990240, 8100040  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 23S (EPSG:32723)
source      : spat_s9YbPQxviLyJU8U_3384.tif
names       : B12_20m, B11_20m, B8A_20m
min values  :       0,       0,       0
max values  :       1,       1,       1
I<-(Nor[[1]]+Nor[[2]]+Nor[[3]])/3
S<-1-(3/(Nor[[1]]+Nor[[2]]+Nor[[3]]))*min(Nor[[1]],Nor[[2]],Nor[[3]])
H<- acos((0.5*((Nor[[1]]-Nor[[2]])+(Nor[[1]]-Nor[[3]])))/(sqrt((Nor[[1]]-
Nor[[2]])^2+((Nor[[1]]-Nor[[3]])*(Nor[[2]]-Nor[[3]])))))
ihs<-c(I,H,S)
plotRGB(ihs,stretch='hist')
```



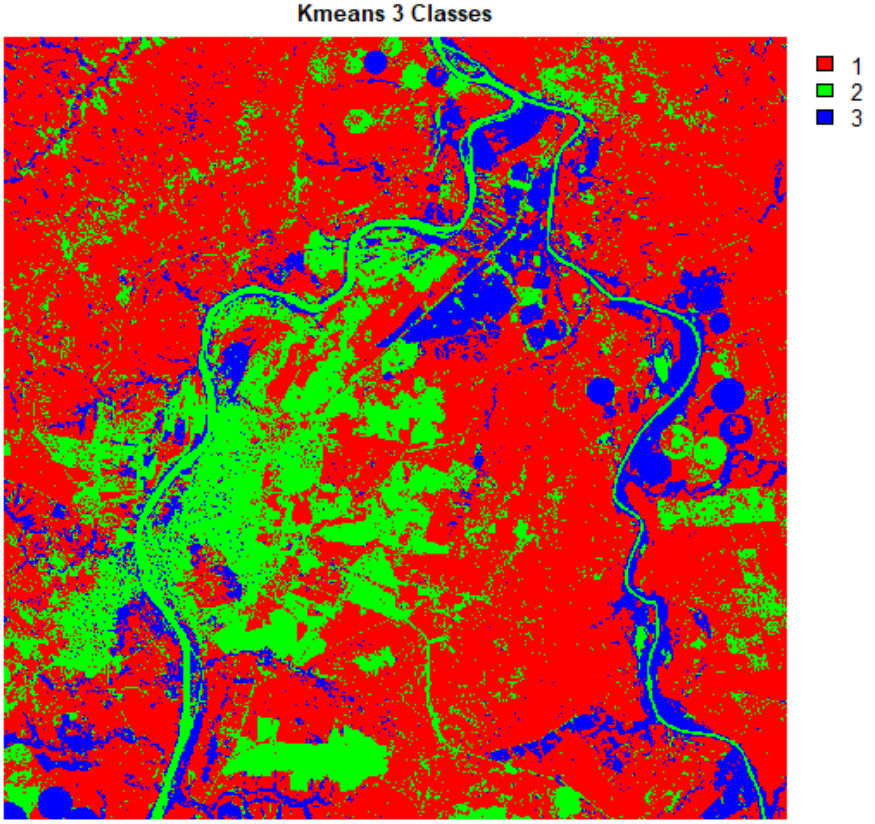
```
dev.off()
par(mfrow=c(1,3))
plot(I,axes=FALSE,legend=FALSE)
plot(H,axes=FALSE,legend=FALSE)
plot(S,axes=FALSE,legend=FALSE)
```

12 Classificação de Imagens

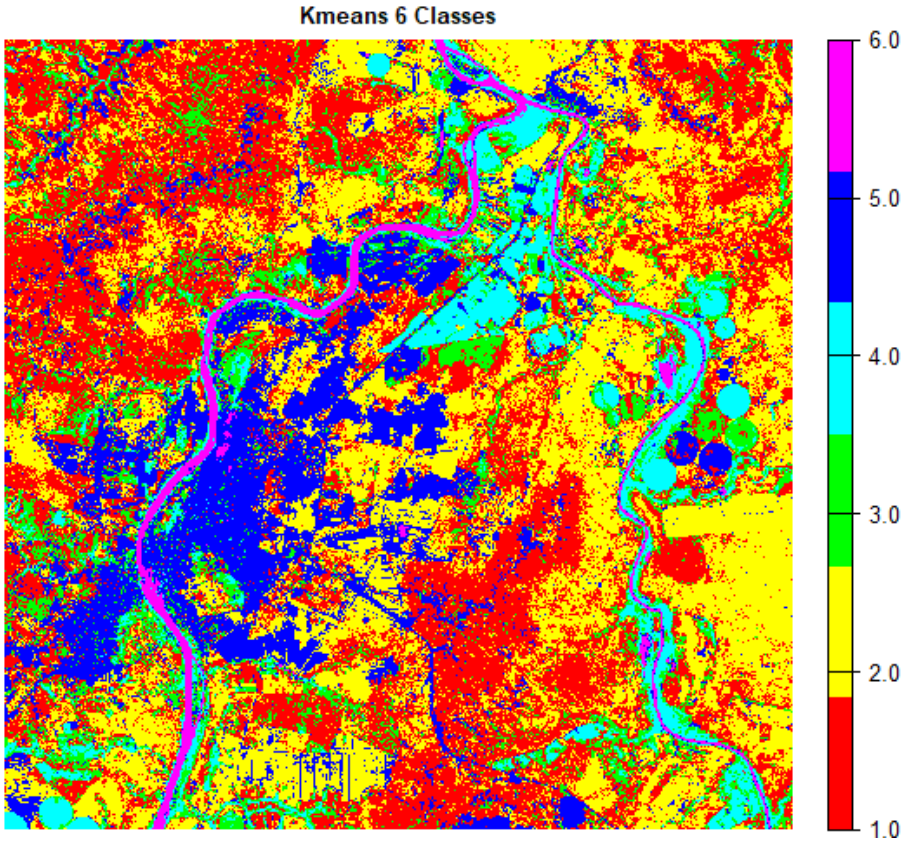
Receita - 12.1 – Kmeans

Executando classificação KMeans não supervisionada na Composição de Índices MNDWI, SAVI e NDBI+NDTI.

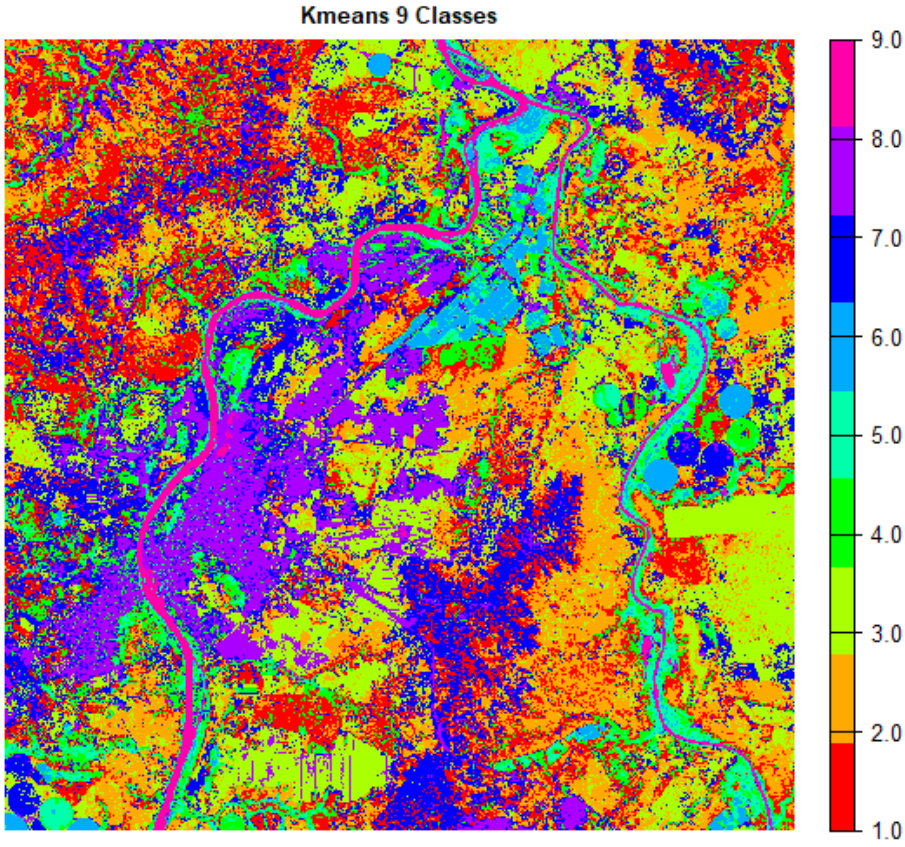
```
library(terra)
library(cluster) #instale cluster using install.packages("cluster")
nir<-rast('B8A_20m.jp2')
vermelho<-rast('B04_20m.jp2')
savi<-(nir-vermelho)/(nir+vermelho+0.5)*1.5
green<-rast('B03_20m.jp2')
swir<-rast('B11_20m.jp2')
mndwi<-(green-swir)/(green+swir)
swir1<-rast('B11_20m.jp2')
swir2<-rast('B12_20m.jp2')
ndti<-(swir1-swir2)/(swir1+swir2)
nir<-rast('B8A_20m.jp2')
swir<-rast('B11_20m.jp2')
ndbi<-(swir-nir)/(swir+nir)
e <- ext(499980, 527430, 8072590,8100040)
rgb<-rast(list(ndbi+ndti,savi,mndwi))
rgbi<-crop(rgb,e)
img<-c(rgbi[[1]],rgbi[[2]],rgbi[[3]])
ar <- values(img) #carregando o raster em um vetor
values <- which(!is.na(ar)) #índice de valores válidos (valores não NA)
ar <- na.omit(ar) #removendo NA do vetor
Result <- clara(ar,3,samples=500,metric="manhattan",pamLike=T)
kmraster3 <- rast (rgbi[[1]])
values(kmraster3) <- Result$clustering #carregando valores da classe
Result <- clara(ar,6,samples=500,metric="manhattan",pamLike=T)
kmraster6 <- rast(rgbi[[1]])
values(kmraster6) <- Result$clustering #carregando valores da classe
Result <- clara(ar,9,samples=500,metric="manhattan",pamLike=T)
kmraster9 <- rast(rgbi[[1]])
values(kmraster9) <- Result$clustering #carregando valores da classe
plot(kmraster3, legend=T,axes=F, col = rainbow(3),main='Kmeans 3 Classes')
```



```
plot(kmraster6, legend=T,axes=F, col = rainbow(6),main='Kmeans 6 Classes')
```



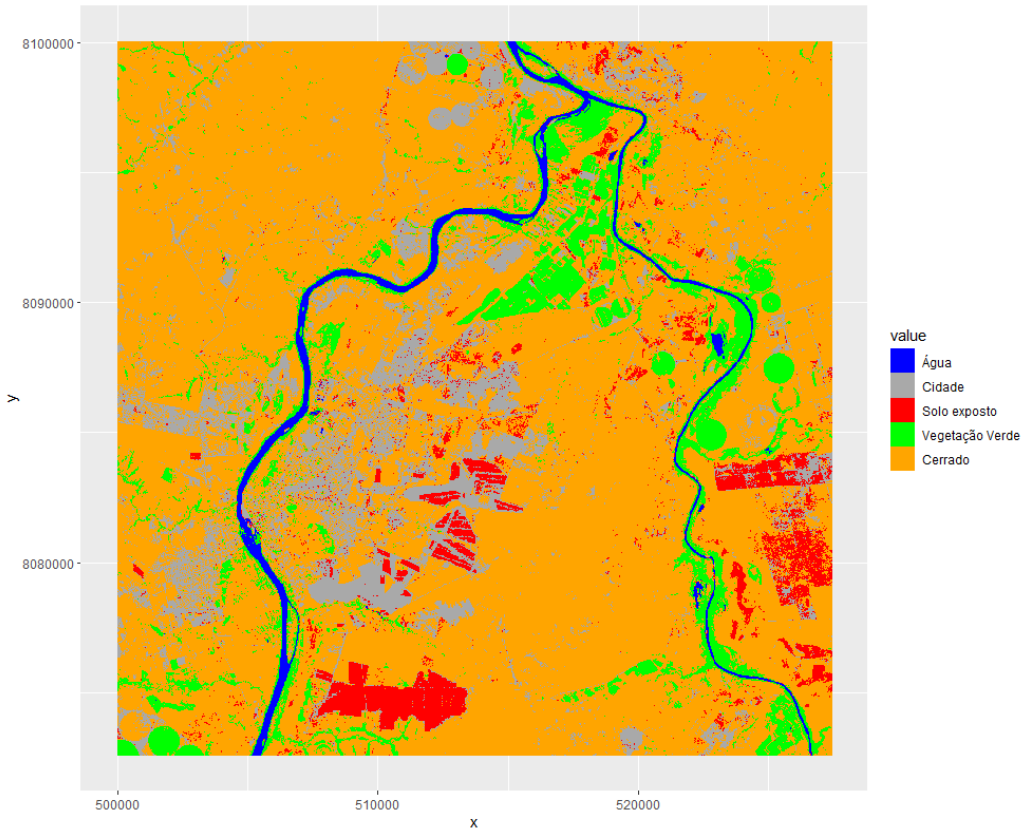

```
plot(kmraster9, legend=T,axes=F, col = rainbow(9),main='Kmeans 9 Classes')
```



Receita - 12.2 – SAM

Aplicando Spectral Angle Mapper (SAM) supervisionado para classificar Composição de Índices MNDWI, SAVI e NDBI+NDTI.

```
library(terra)
library(RStoolbox) #install RStoolbox using install.packages("RStoolbox")
nir<-rast('B8A_20m.jp2')
vermelho<-rast('B04_20m.jp2')
savi<-(nir-vermelho)/(nir+vermelho+0.5)*1.5
green<-rast('B03_20m.jp2')
swir<-rast('B11_20m.jp2')
mndwi<-(green-swir)/(green+swir)
swir1<-rast('B11_20m.jp2')
swir2<-rast('B12_20m.jp2')
ndti<-(swir1-swir2)/(swir1+swir2)
nir<-rast('B8A_20m.jp2')
swir<-rast('B11_20m.jp2')
ndbi<-(swir-nir)/(swir+nir)
e <- ext(499980, 527430, 8072590,8100040)
rgb<-rast(list(ndbi+ndti,savi,mndwi))
rgbi<-crop(rgb,e)
library(raster) #instale raster usando install.packages("raster ")
## endmember de treino
pontos <- data.frame(x = c(506009, 507004,512952,504937,518565), y = c(8084856,
8084148,8083483,8085263,8081092))
endmembers <- extract(rgbi, pontos)
rownames(endmembers) <- c("água", "asfalto","solo","vegetação verde","cerrado
seco")
## Classificando com base no ângulo Mínimo
samCl <- sam(rgbi, endmembers, angles = FALSE)
library(ggplot2) #instale ggplot2 usando install.packages("ggplot2 ")
ggR(samCl, forceCat = TRUE, geom_raster=TRUE) + scale_fill_manual(values =
c("blue", "darkgray","red","green","orange"), labels = c("Água", "Cidade","Solo
exposto","Vegetação Verde","Cerrado"))
```



Goiânia - GO, 20 de setembro de 2024