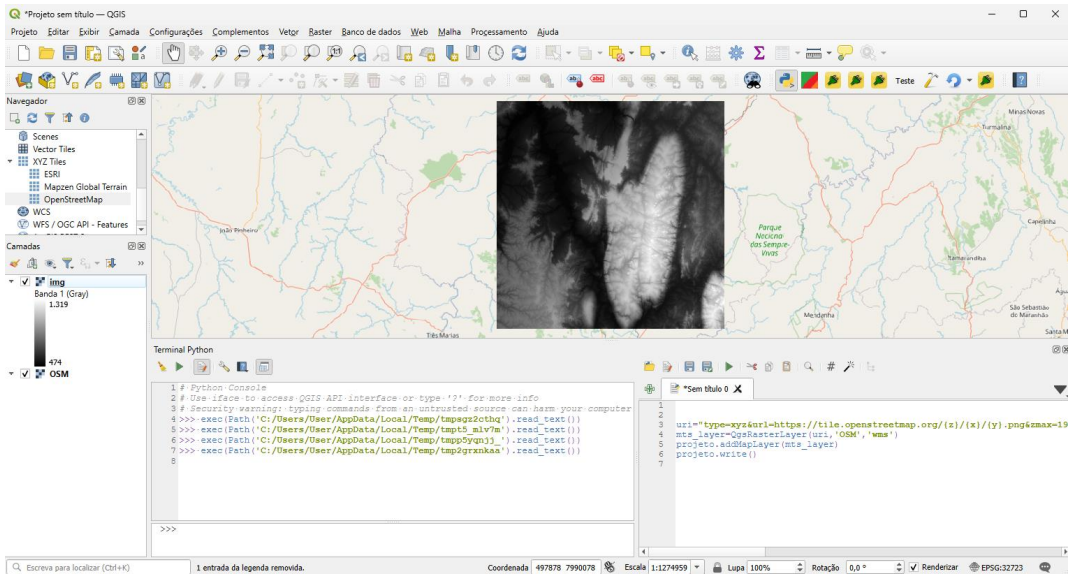
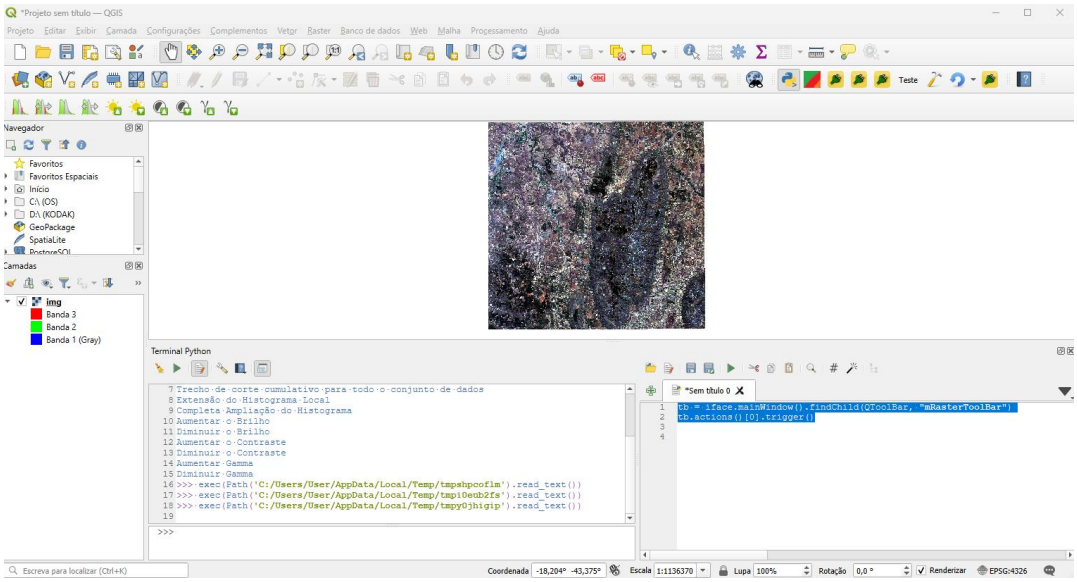




Usando pyQGIS

Classe Raster - *QgsRasterLayer*

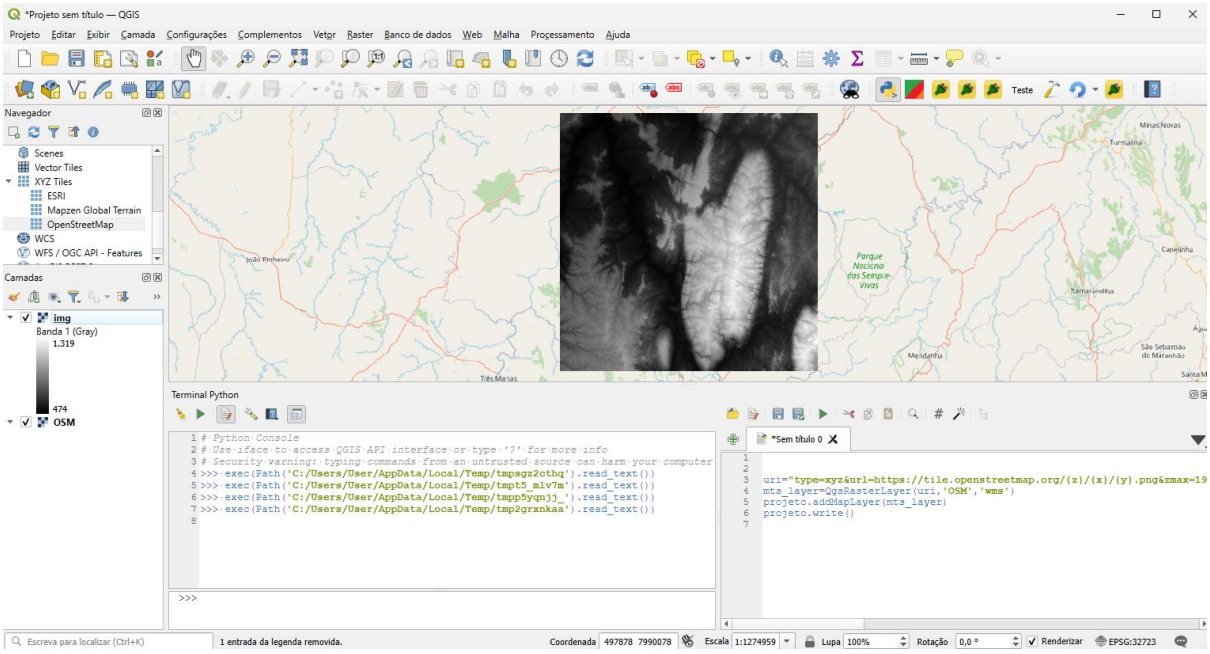
Classe Raster - *QgsRasterLayer*

Similar à forma que adicionamos camadas vetoriais, podemos adicionar imagens raster no nosso projeto usando objeto da a classe raster. Vamos criar um projeto e adicionar uma imagem raster nele.

```
projeto=QgsProject.instance()
camadaR =QgsRasterLayer("C:/Users/user/desktop/dash/dem2.tif","img")
projeto. addMapLayer(camadaR)
projeto.write('C:/Users/user/desktop/dash/meu_projeto3.qgs')
```

Podemos também adicionar dados do tipo raster usando provedores do tipo TMS (TileMapService) ou WMS (WebMapaService).

```
uri="type=xyz&url=https://tile.openstreetmap.org/{z}/{x}/{y}.png&zmax=19&zmin=0"
mts_layer=QgsRasterLayer(uri,'OSM','wms')
projeto.addMapLayer(mts_layer)
projeto.write()
```

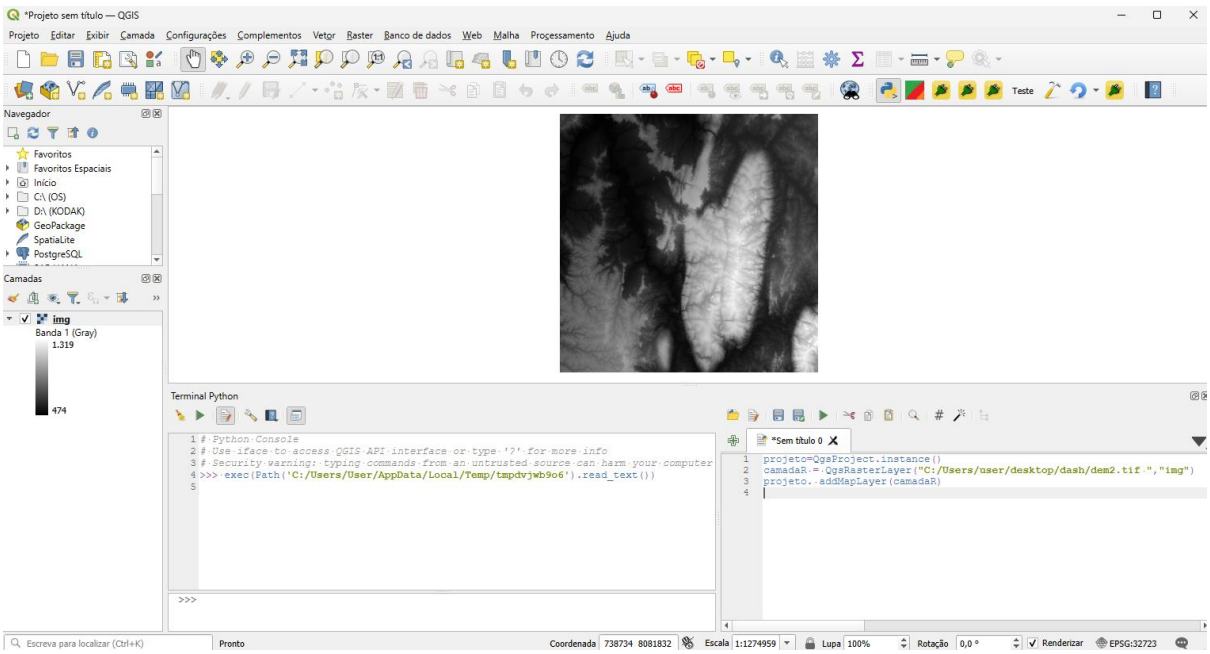


Interagindo com informações de objetos da classe Raster

Podemos extrair informações relevantes de um objeto raster também tais como dimensões, resoluções, número de bandas, valor de um pixel, etc. Vamos carregar uma imagem inicialmente.

```
projeto=QgsProject.instance()
camadaR = QgsRasterLayer("C:/Users/user/desktop/dash/dem2.tif ","img")
projeto. addMapLayer(camadaR)
```

Isso abrirá a imagem como mostrado abaixo:



Informações de dimensão do objeto Raster

Podemos acessar informações de parâmetros dimensionais de uma imagem raster usando métodos específicos para o tal.

```
camadaR.width(), camadaR.height() #largura e altura
```

(3649, 3650)

```
camadaR.extent() #extensão da imagem na unidade da coordenada
```

<QgsRectangle: 500000 7990240, 609780 8100040>

```
camadaR.crs().description() #sistema de referência
```

'WGS 84 / UTM zone 23S'

```
camadaR.rasterUnitsPerPixelX() #resolução em X
```

30.08495478213209

```
camadaR.rasterUnitsPerPixelY() #resolução em Y
```

30.08219178082192

Essas importantes informações sobre o raster poderão ser usadas para análises espacias futuras. Existem outras formas de usar os métodos para obtermos a mesma informação.

Podemos calcular a resolução em X usando o código abaixo em vez de usar o método rasterUnitsPerPixelX():

```
(camadaR.extent().xMaximum()-camadaR.extent().xMinimum())/camadaR.width()
```

30.08495478213209

Podemos também visualizar informações usando o método htmlMetadata().

```
camadaR.htmlMetadata()
```

Texto HTML gerado acimo visualizado no navegador.

Informação do provedor

Extensão	500000.0000000000000000,7990240.0000000000000000 : 609780.0000000000000000,8100040.0000000000000000
Largura	3649
Altura	3650
tipo de dado	Int16 - Inteiro de 16 bits com sinal
Descrição do driver GDAL	GTiff
Metadados do driver GDAL	GeoTIFF
Descrição do registro	C:/Users/user/desktop/dash/dem2.tif
Compressão	
Banda 1	• STATISTICS_APPROXIMATE=YES • STATISTICS_MAXIMUM=1319 • STATISTICS_MEAN=739.82844614963 • STATISTICS_MINIMUM=474 • STATISTICS_STDDEV=201.15310812442 • STATISTICS_VALID_PERCENT=100 • Escala: 1 • Deslocamento: 0
Mais informação	• AREA_OR_POINT=Area • TIFFTAG_XRESOLUTION=1 • TIFFTAG_YRESOLUTION=1
Dimensões	X: 3649 Y: 3650 Bandas: 1
Origem	500000.0000000000000000,8100040.0000000000000000
Tamanho do Pixel	30.0849547821320904,-30.0821917808219102

Sistema de referência de coordenadas (SRC)

Nome	EPSG:32723 - WGS 84 / UTM zone 23S
Unidades	metros
Type	Projetado
Método	Universal Transverse Mercator (UTM)
Celestial Body	Earth
Precisão	Com base na *World Geodetic System 1984 ensemble* (EPSG:6326), que tem uma precisão limitada de **no máximo 2 metros**.
Referência	Dinâmico (depende de um dado que não está fixado na placa)

Identificação

Identifier
Parent Identifier
Title
Type dataset
Language
Abstract
Categories
Keywords

Extensão

CRS EPSG:32723 - WGS 84 / UTM zone 23S - Projected
Spatial Extent
Temporal Extent

Acesso

Fees
Licenses
Rights
Constraints

Bandas

Contagem de bandas 1

Número	Banda	Sem Dados	Mín	Máx
1	Banda 1	n/a	474.0000000000	1319.0000000000

Contatos

No contact yet.

Referências

No links yet.

Histórico

No history yet.

Raster com uma banda de valores

Vamos ver agora métodos para raster de uma banda. Os métodos abaixo informam o número de bandas e o tipo da imagem raster. 0 para cinza ou não definido de banda única, 1 para paletado de banda única e 2 para multibanda.

```
camadaR.bandCount()
```

1

```
camadaR.rasterType()
```

<RasterLayerType.GrayOrUndefined: 0>

A função dataProvider() funciona como uma interface entre o objeto raster os seus dados individuais, seu método sample() toma dois valores, um objeto ponto (coordenadas XZ) e o número da banda. Se a coordenada for dentro da imagem e a banda existir o resultado será um tuple com o valor do pixel e se o dado é verdadeiro ou não.

```
valor=camadaR.dataProvider().sample(QgsPointXY(687567, 7460876),1)
valor
(nan, False)

valor2=camadaR.dataProvider().sample(QgsPointXY(547802,8043049), 1)
valor2
(980.0, True)
```

A rampa de cor assinalada ao objeto raster pode ser checada usando o método type() do método renderer(). O tipo singlebandgray é o padrão inicial.

```
>>> camadaR.renderer().type()

'singlebandgray'
```

Podemos alterar via python a rampa de cores, o processo é mostrado abaixo. O processo envolve na criação de um objeto do tipo ColorRampShader e definimos a rampa de cor de preenchimento como sendo do tipo interpolado.

```
fcn = QgsColorRampShader()
fcn.setColorRampType(QgsColorRampShader.Interpolated)
```

Criamos agora uma lista com as cores representando os dois valores extremos do raster (0 e 2046 que serão interpolados entre azul e amarelo. Em seguida adicionamos esta lista como item do ColorRampShader criado acima.

```
lista = [ QgsColorRampShader.ColorRampItem(0, QColor(0,0,255)),
QgsColorRampShader.ColorRampItem(2046, QColor(255,255,0))]
fcn.setColorRampItemList(lista)
```

O próximo passo é criarmos o RasterShader (preenchedor de cor) e associarmos o RampShader a ele.

```
shader = QgsRasterShader()
shader.setRasterShaderFunction(fcn)
```

Finalmente criamos o objeto renderizador de cor com: dados do objeto raster, banda 1 e shader acima. Em seguida aplicamos este ao objeto raster e chamamos a repintura do objeto.

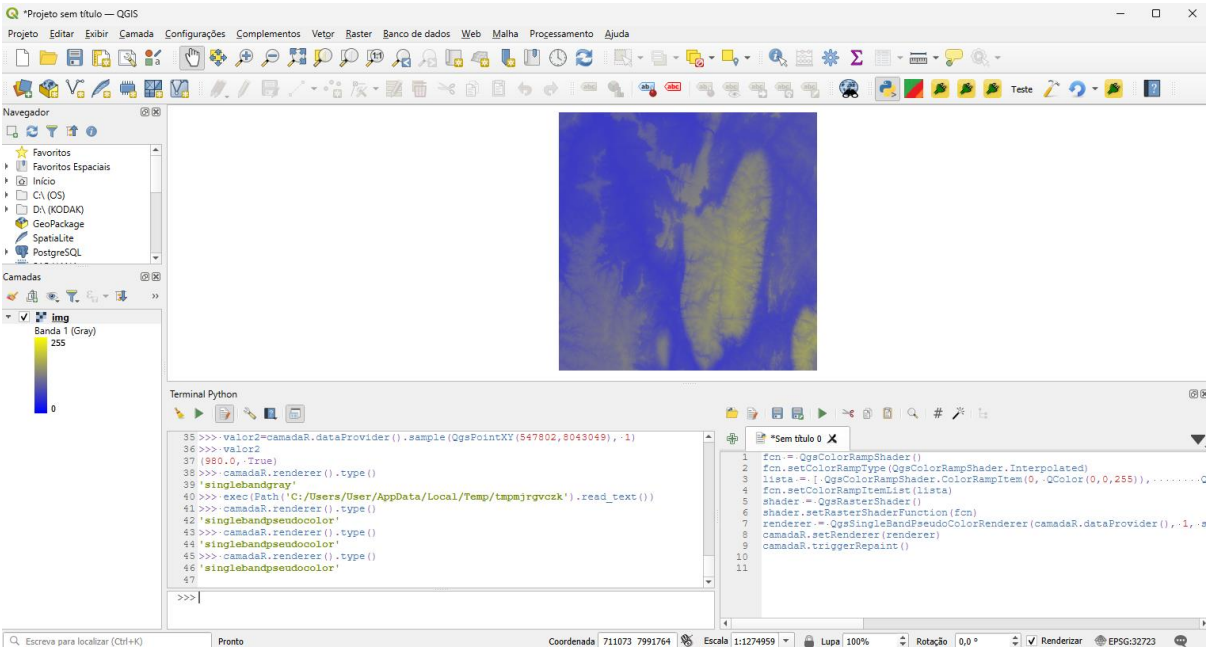
```
renderer = QgsSingleBandPseudoColorRenderer(camadaR.dataProvider(), 1, shader)
camadaR.setRenderer(renderer)
camadaR.triggerRepaint()
```

Se chamarmos o tipo novamente podemos ver a mudança.

```
camadaR.renderer().type()

'singlebandpseudocolor'
```

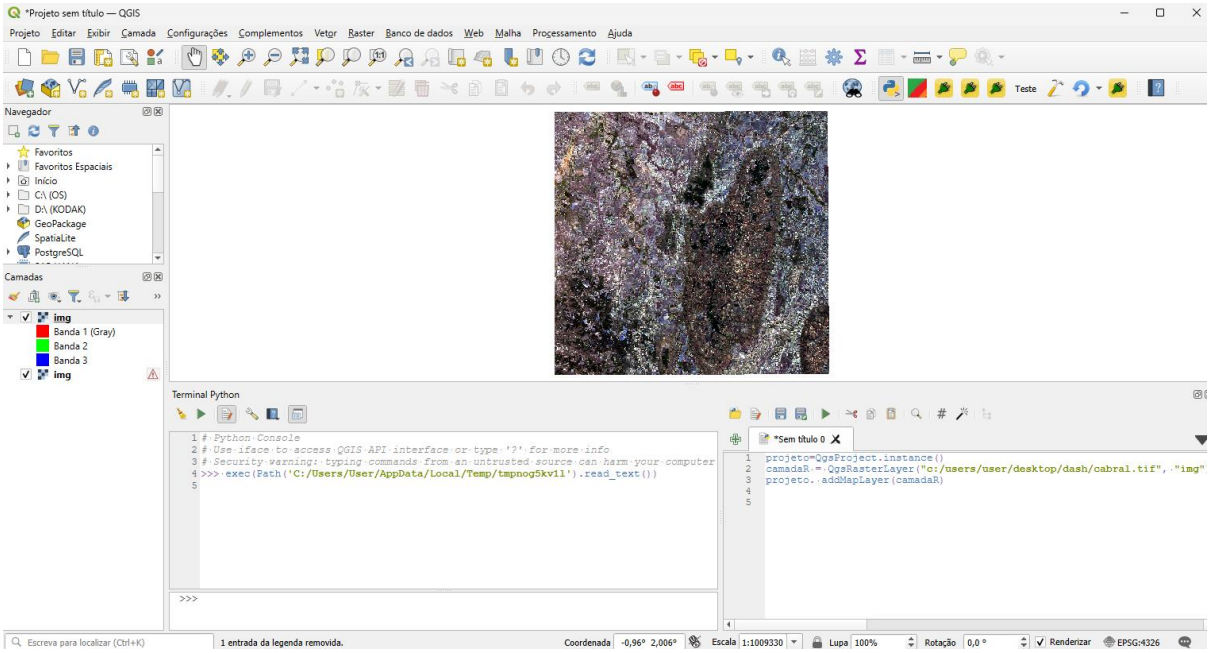
O resultado é mostrado na imagem abaixo.



<u>Raster com mais de uma banda de valores</u>

Vamos ver agora métodos para raster de mais de uma banda. Vamos trabalhar um pouco agora com imagem raster de 3 bandas. Carregamos o raster de forma similar e vamos extrair algumas de suas informações.

```
projeto=QgsProject.instance()
camadaR = QgsRasterLayer("c:/users/user/desktop/dash/cabral.tif",
"img")
projeto. addMapLayer(camadaR)
```



Podemos ver algumas das informações usando:

```
camadaR.bandCount()# número de bandas
```

3

```
camadaR.rasterType() # 2 para multi banda
```

<RasterLayerType.MultiBand: 2>

```
# valor do pixel na banda 1
valor=camadaR.dataProvider().sample(QgsPointXY(547802,8043049),1)
>>> valor
```

(893.0, True)

```
# valor do pixel na banda 2
valor=camadaR.dataProvider().sample(QgsPointXY(547802,8043049),2)
valor
```

(781.0, True)

```
# valor do pixel na banda 3
valor=camadaR.dataProvider().sample(QgsPointXY(547802,8043049),3)
valor
```

(719.0, True)

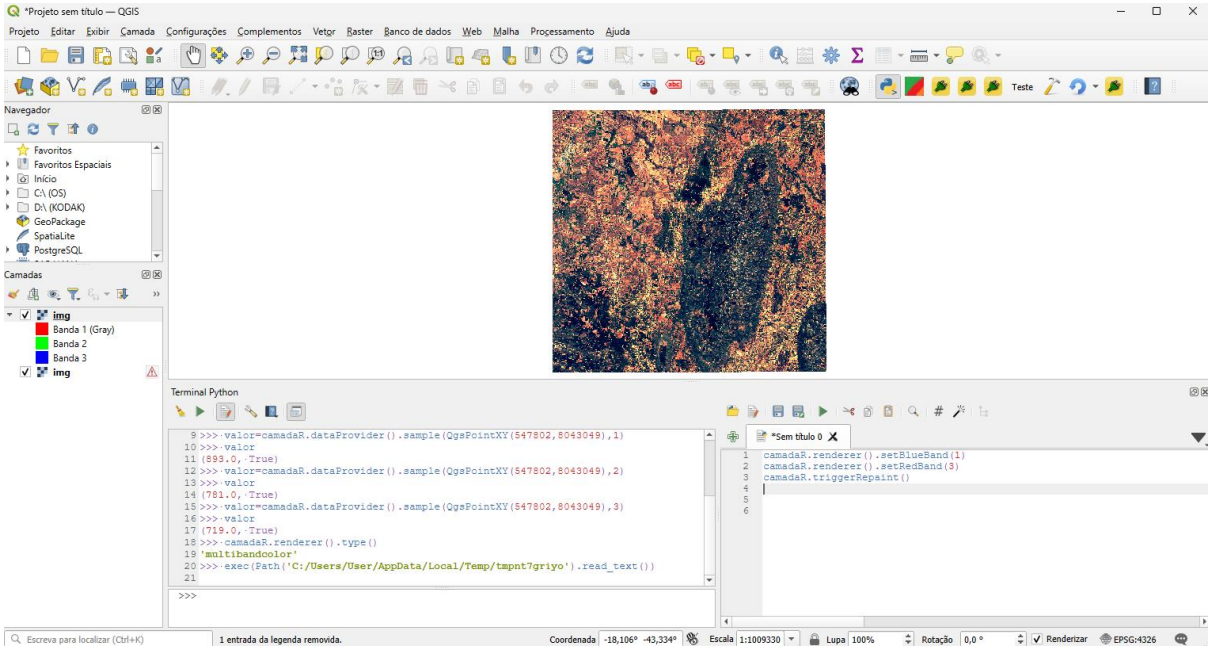
```
camadaR.renderer().type()
```

'multibandcolor'

Vamos ver abaixo como modificar a imagem para que a banda 1 fique no canal azul (B) e a banda 3 fique no canal Vermelho (R).

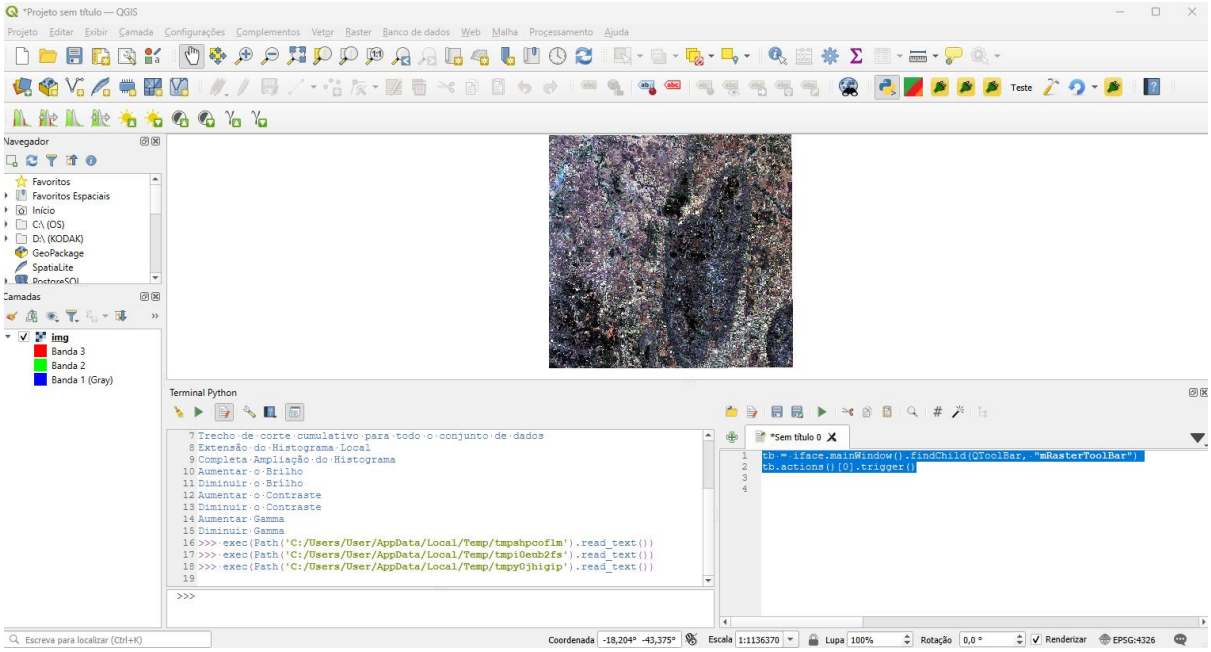
```
camadaR.renderer().setBlueBand(1)
camadaR.renderer().setRedBand(3)
camadaR.triggerRepaint()
```

Note que o histograma da imagem não foi apropriadamente ajustado porque ainda usa os valores de máximo e mínimo das bandas anteriores.



Para ajustar execute

```
tb = iface.mainWindow().findChild(QToolBar, "mRasterToolBar")
tb.actions()[0].trigger()
```



Criando objeto raster

Imagens raster também podem ser criadas via script de forma bem eficiente usando uma lista de dados pontuais com um determinado valor. Vamos aqui criar um raster mostrando a temperatura média de uma área com base em informações pontuais de vários locais. O arquivo CSV tfinal.csv tem os dados com coordenadas, e respectivos valores. Vamos carregar a informação em um objeto do tipo QsgInterpolator camada de dados (layerData).

```
uri="file:///c:/users/user/desktop/dash/tfinal.csv?
type=csv&xField=LONGITUDE&yField=LATITUDE&crs=epsg:4326"
camada = QgsVectorLayer(uri, 'Converte', "delimitedtext")
c_data = QgsInterpolator.LayerData()
c_data.source = camada
c_data.zCoordInterpolation = False
c_data.interpolationAttribute = 6
c_data.sourceType = QgsInterpolator.SourcePoints
```

Executaremos a interpolação usando o inverso da distância ponderada (IDW) ao quadrado (coeficiente 2).

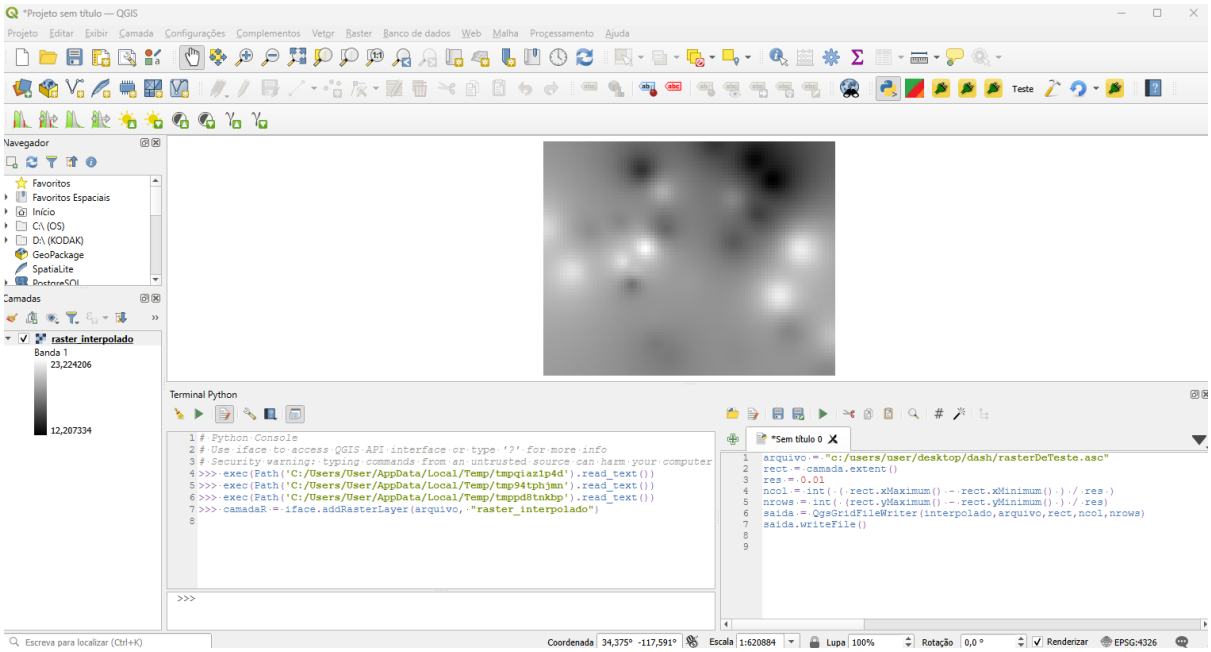
```
interpolado = QgsIDWInterpolator([c_data])
interpolado.setDistanceCoefficient(2)
```

Agora definimos qual arquivo será criado e os parâmetros do grid a ser usado.

```
arquivo = "c:/users/user/desktop/dash/rasterDeTeste.asc"
rect = camada.extent()
res = 0.01
ncol = int( ( rect.xMaximum() - rect.xMinimum() ) / res )
nrows = int( (rect.yMaximum() - rect.yMinimum() ) / res)
saida = QgsGridFileWriter(interpolado,arquivo,rect,ncol,nrows)
saida.writeFile()
```

Carregamos o arquivo do grid usando.

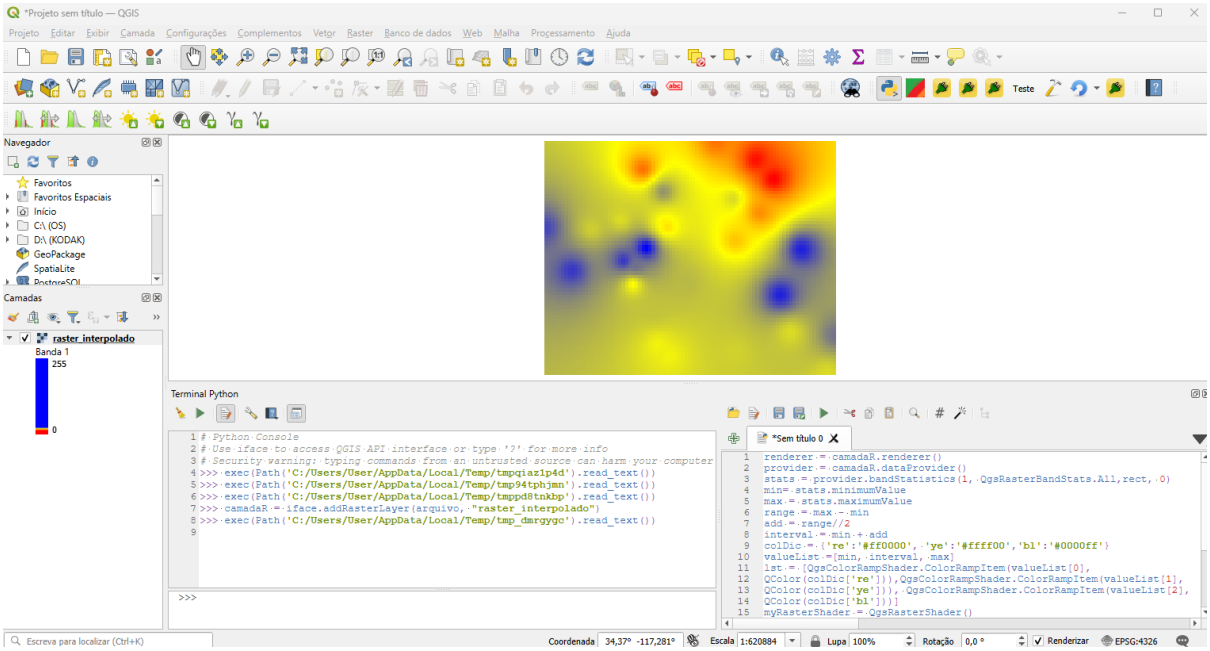
```
camadaR = iface.addRasterLayer(arquivo, "raster_interpolado")
```



Podemos alterar a aparência do raster que criamos usando o código seguinte.

```
renderer = camadaR.renderer()
provider = camadaR.dataProvider()
stats = provider.bandStatistics(1, QgsRasterBandStats.All,rect, 0)
min= stats.minimumValue
max = stats.maximumValue
range = max - min
add = range//2
interval = min + add
colDic = {'re':'#ff0000', 'ye':'#ffff00','bl':'#0000ff'}
valueList =[min, interval, max]
lst = [QgsColorRampShader.ColorRampItem(valueList[0],
QColor(colDic['re'])),QgsColorRampShader.ColorRampItem(valueList[1],
QColor(colDic['ye'])),
QgsColorRampShader.ColorRampItem(valueList[2],
QColor(colDic['bl']))]
myRasterShader = QgsRasterShader()
myColorRamp = QgsColorRampShader()
myColorRamp.setColorRampItemList(lst)
myRasterShader.setRasterShaderFunction(myColorRamp)
myPseudoRenderer
=QgsSingleBandPseudoColorRenderer(camadaR.dataProvider(),
camadaR.type(), myRasterShader)
camadaR.setRenderer(myPseudoRenderer)
camadaR.triggerRepaint()
```

O resultado da rampa de cores criada aplicado no raster será.



No próximo módulo veremos outras classes do pyQGIS e suas utilidades. Até lá!