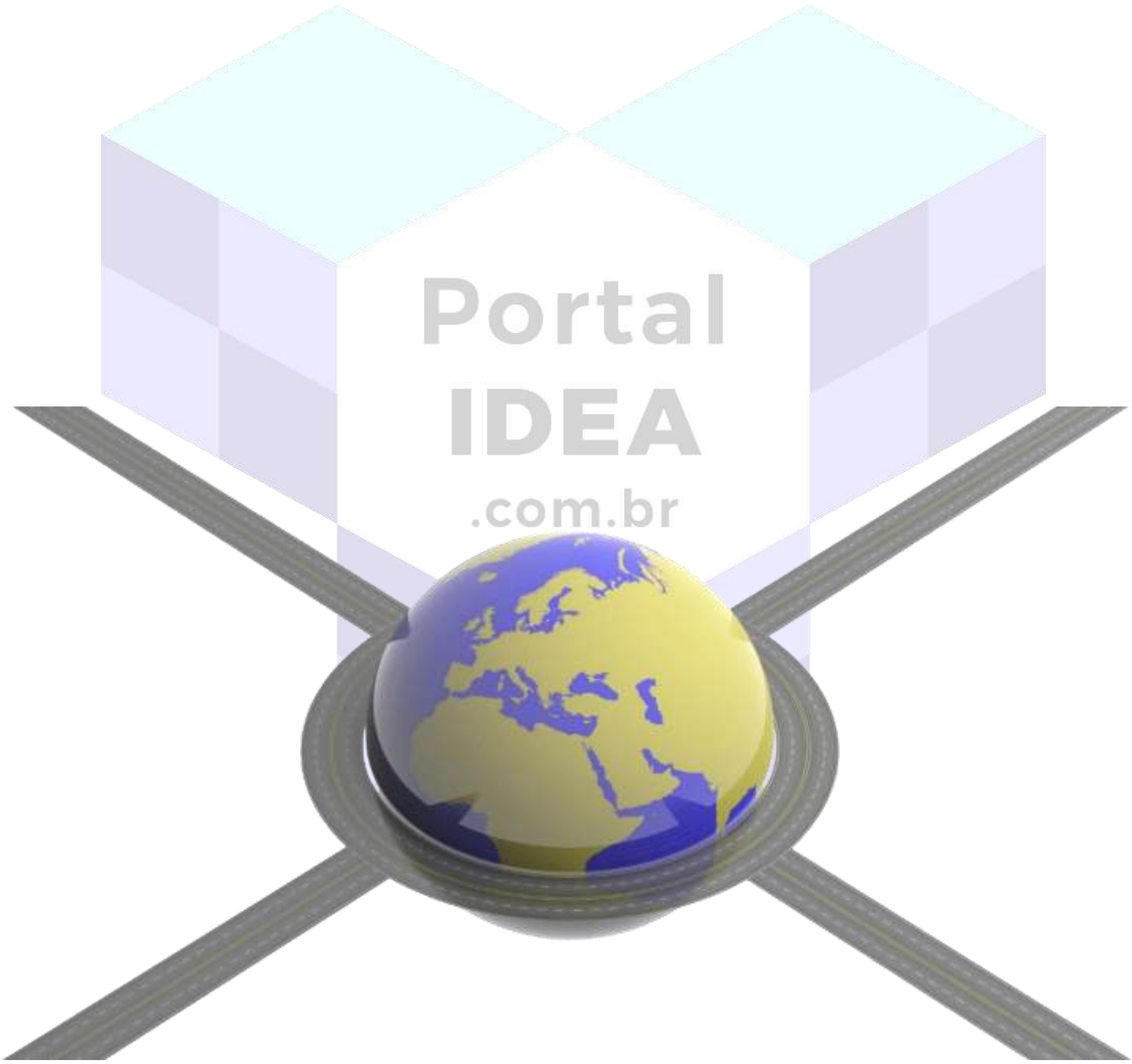
INTRODUÇÃO AO GOOGLE EARTH ENGINE



Programação no Google Earth Engine

Linguagem JavaScript no GEE

A linguagem de programação JavaScript é a base para o desenvolvimento de scripts no Google Earth Engine (GEE). Com ela, é possível manipular dados geoespaciais, realizar análises e visualizar resultados diretamente na plataforma. O uso do JavaScript no GEE é adaptado para lidar com objetos e métodos específicos da biblioteca Earth Engine.

Fundamentos do JavaScript Aplicados ao GEE

O JavaScript no GEE segue a sintaxe tradicional da linguagem, com algumas adaptações para interagir com os dados geoespaciais. Aqui estão os conceitos principais:

1. **Objetos do Earth Engine:**
    - Todos os dados no GEE são representados como objetos específicos, como ee.Image, ee.Feature, ee.Geometry, e ee.ImageCollection.
    - Métodos associados a esses objetos são utilizados para realizar operações, como filtros, cálculos e exportações.
2. **Funções:**
    - Funções são amplamente usadas no GEE para definir operações personalizadas.

- É comum usar **funções anônimas** (funções inline) ao aplicar métodos, como map().

3. **Variáveis:**
   - Variáveis são declaradas para armazenar referências a dados ou resultados de operações.
   - A palavra-chave var é usada frequentemente, mas let e const também são suportados.
4. **Coleções:**
   - O GEE utiliza coleções para manipular conjuntos de dados, como imagens (ee.ImageCollection) ou feições (ee.FeatureCollection).

Exemplo Básico:

```
// Definindo uma variável com uma coleção de imagens do Landsat 8
var landsat = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR');

// Filtrando por data e local
var filtered = landsat.filterDate('2023-01-01', '2023-12-31')
                      .filterBounds(ee.Geometry.Point([-43.1729, -22.9068]));
```

Estrutura Básica de Scripts no GEE

Um script típico no GEE segue uma estrutura lógica que inclui:

1. **Importação de Dados:**
   - Definir as coleções ou dados necessários para análise.

2. **Processamento:**
    - Aplicar filtros, cálculos e operações nos dados.
3. **Visualização:**
    - Exibir os resultados no mapa interativo com Map.addLayer().
4. **Exportação (opcional):**
    - Salvar os resultados para uso externo com métodos de exportação.

Exemplo Completo:

```
// Importando uma coleção de imagens do Sentinel-2
var sentinel = ee.ImageCollection('COPERNICUS/S2');

// Filtrando por data e área de interesse
var image = sentinel.filterDate('2023-01-01', '2023-03-31')
                    .filterBounds(ee.Geometry.Point([-43.1729, -22.9068]))
                    .median();

// Visualizando os dados no mapa
Map.centerObject(ee.Geometry.Point([-43.1729, -22.9068]), 10);
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Sentinel-2 RGB');
```

Execução de Scripts e Depuração

Uma vez que o script é criado, ele pode ser executado diretamente no **Code Editor**. O GEE oferece ferramentas integradas para verificar erros e analisar a saída dos scripts.

1. **Execução de Scripts:**
    - Clique no botão "Run" no topo do Code Editor para executar o script.
    - Resultados são exibidos no mapa interativo ou no console, dependendo do comando.
2. **Depuração:**
    - O console no GEE é essencial para depurar scripts.
    - Use o comando print() para inspecionar variáveis ou visualizar informações detalhadas sobre objetos.

Exemplo:

```
// Verificando informações da imagem filtrada
print('Informações da Imagem:', image);
```

2. **Mensagens de Erro:**
    - Erros comuns, como métodos mal utilizados ou parâmetros ausentes, são destacados automaticamente.
    - Leia a mensagem de erro no console para corrigir problemas.
3. **Logs e Saídas:**
    - Use print() para verificar o progresso do script e os valores intermediários.
    - O comando Map.addLayer() também pode ser usado para verificar visualmente a lógica do script.

Dica de Depuração:

Adicione etapas ao script gradualmente e teste cada parte antes de prosseguir. Isso facilita a identificação de problemas.

A linguagem JavaScript no Google Earth Engine é um componente essencial para criar análises geoespaciais eficientes. Com um bom entendimento da estrutura dos scripts e do uso das ferramentas de depuração, os usuários podem explorar todo o potencial do GEE para suas pesquisas e projetos.

Manipulação de Imagens Satelitais no Google Earth Engine

A manipulação de imagens satelitais é uma das principais funcionalidades do Google Earth Engine (GEE). A plataforma permite trabalhar com vastas coleções de imagens de satélites, como Landsat, Sentinel e MODIS, possibilitando análises detalhadas e monitoramento ambiental em larga escala.

Seleção de Coleções de Imagens

No GEE, as imagens satelitais são organizadas em coleções, que agrupam dados de um mesmo satélite ou sensor. A seleção de coleções é o primeiro passo para manipular esses dados.

1. **Acesso ao Catálogo de Dados:**
    - Utilize o catálogo de dados do GEE para encontrar coleções de imagens relevantes.
    - Exemplos de coleções populares:
        - **Landsat 8 (LANDSAT/LC08/C01/T1_SR)**: Imagens ópticas de alta qualidade.
        - **Sentinel-2 (COPERNICUS/S2)**: Imagens multiespectrais com alta resolução espacial.
        - **MODIS (MODIS/006/MOD13A2)**: Dados de vegetação e clima em escala global.

2. **Importação de Coleções:**
    - As coleções são importadas no script utilizando a função ee.ImageCollection().
    - É possível manipular e processar as imagens da coleção diretamente.

Exemplo:

```
// Importando a coleção Sentinel-2
var sentinel = ee.ImageCollection('COPERNICUS/S2');
print('Coleção Sentinel-2:', sentinel);
```


Aplicação de Filtros Espaciais e Temporais

Após selecionar uma coleção, os filtros espaciais e temporais permitem refinar os dados para atender às necessidades do projeto.

1. **Filtros Temporais:**
    - Use o método filterDate(start, end) para selecionar imagens dentro de um intervalo de tempo.
    - Exemplo:

```
var filteredByDate = sentinel.filterDate('2023-01-01', '2023-12-31');
```

2. **Filtros Espaciais:**

- Utilize filterBounds(geometry) para selecionar imagens que cobrem uma área específica.

```
var area = ee.Geometry.Point([-43.1729, -22.9068]);
var filteredByArea = filteredByDate.filterBounds(area);
```

3. **Combinação de Filtros:**

- Combine filtros temporais e espaciais para obter imagens mais relevantes.

Exemplo Completo:

```
var filteredImages = sentinel.filterDate('2023-01-01', '2023-12-31')
                             .filterBounds(ee.Geometry.Point([-43.1729, -22.9068]));
print('Imagens filtradas:', filteredImages);
```

Visualização e Interpretação de Imagens Satelitais

O GEE facilita a visualização e interpretação de imagens, permitindo ajustes de parâmetros para melhorar a compreensão dos dados.

1. **Adicionando Imagens ao Mapa:**
    - Use Map.addLayer() para exibir imagens no mapa interativo.
    - Especifique as bandas de interesse e configure parâmetros como intervalo de valores (min, max) e paletas de cores.

Exemplo:

```
var image = filteredImages.median(); // Obtendo a média das imagens filtradas
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Sentinel-2 RGB');
Map.centerObject(image, 10);
```

2. **Interpretando Bandas Espectrais:**

- Cada banda representa uma faixa do espectro eletromagnético, como vermelho (B4), verde (B3) e azul (B2).
- Combinações de bandas ajudam a destacar características específicas:

- **RGB Natural:** B4 (Vermelho), B3 (Verde), B2 (Azul).
- **Infravermelho Próximo:** B8 (Infravermelho), B4 (Vermelho), B3 (Verde).

3. **Extração de Informações:**

- Inspecione pixels no mapa interativo para obter valores específicos.
- Use o console para calcular índices, como NDVI (Índice de Vegetação por Diferença Normalizada):

```
var ndvi = image.normalizedDifference(['B8', 'B4']);

Map.addLayer(ndvi, {min: 0, max: 1, palette: ['blue', 'green']}, 'NDVI');
```

4. **Ajustes de Visualização:**
   - Experimente diferentes intervalos de valores (min, max) e paletas para destacar aspectos específicos da imagem.

A manipulação de imagens satelitais no GEE é uma tarefa poderosa que permite explorar e analisar dados geoespaciais de forma eficiente. Desde a seleção de coleções até a aplicação de filtros e a visualização no mapa, o GEE fornece ferramentas abrangentes para atender a uma ampla gama de necessidades analíticas e científicas.

Manipulação de Dados Vetoriais no Google Earth Engine

O Google Earth Engine (GEE) não se limita apenas ao trabalho com dados raster, como imagens de satélite; ele também oferece suporte robusto para dados vetoriais. Esses dados, compostos por pontos, linhas e polígonos, são fundamentais para representar objetos e realizar análises detalhadas sobre áreas específicas.

Importação de Shapefiles e Criação de Feições

1. **Importação de Shapefiles:**
    - O GEE permite importar shapefiles ou arquivos vetoriais diretamente para o ambiente. Para isso, siga os passos:
        - Faça login no Code Editor do GEE.
        - Clique no botão **"Assets"** no painel lateral esquerdo.
        - Faça o upload do shapefile compactado em formato .zip, que deve conter os arquivos .shp, .shx e .dbf.
        - Após o upload, o shapefile estará disponível como um ee.FeatureCollection.

Exemplo:

```
// Importando um shapefile carregado como asset

var regioes = ee.FeatureCollection('users/usuario/regioes_brasil');

Map.addLayer(regioes, {}, 'Regiões do Brasil');
```

2. **Criação de Feições:**

- Além de importar arquivos, é possível criar feições diretamente no GEE usando o editor de código.
- Utilize ee.Geometry para definir pontos, linhas ou polígonos, e ee.Feature para associar atributos às geometrias.

Exemplo:

```
// Criando um polígono e associando atributos
var poligono = ee.Geometry.Polygon([
  [[-43.2, -22.8], [-43.2, -22.9], [-43.1, -22.9], [-43.1, -22.8]]
]);
var regiao = ee.Feature(poligono, {nome: 'Região de Estudo', tipo: 'Área Urbana'});
var colecao = ee.FeatureCollection([regiao]);
Map.addLayer(colecao, {}, 'Polígono Criado');
```

Processamento e Análise de Dados Vetoriais

1. **Filtros em Coleções Vetoriais:**
    - Use filter() para selecionar feições específicas com base em atributos.
    - Exemplo:

```
var urbanas = regioes.filter(ee.Filter.eq('tipo', 'Urbana'));
Map.addLayer(urbanas, {}, 'Áreas Urbanas');
```

2. **Interseção e Clipping:**

- O GEE permite realizar operações espaciais como interseção e recorte (clipping) entre dados vetoriais e raster.
- Exemplo:

```
var intersecao = poligono.intersection(ee.Geometry.Point([-43.15, -22.85]));

Map.addLayer(intersecao, {color: 'red'}, 'Interseção');
```

3. **Cálculo de Estatísticas Espaciais:**

- Extraia informações de imagens raster para áreas definidas por dados vetoriais.
- Exemplo:

```
var imagem = ee.Image('COPERNICUS/S2').filterDate('2023-01-01', '2023-01-31').median();

var estatisticas = imagem.reduceRegions({
  collection: colecao,
  reducer: ee.Reducer.mean(),
  scale: 10
});

print('Estatísticas:', estatisticas);
```

4. **Transformações Geométricas:**

- O GEE suporta operações como buffer, diferença e união em dados vetoriais.
- Exemplo:

```
var buffer = poligono.buffer(1000); // Buffer de 1 km
Map.addLayer(buffer, {color: 'blue'}, 'Buffer');
```

Exportação de Resultados

1. **Exportação de Dados Vetoriais:**

Os resultados processados podem ser exportados para uso externo em formatos como .shp ou .csv.

Exemplo de exportação para Google Drive:

```
Export.table.toDrive({
  collection: estatisticas,
  description: 'Exportacao_Vetorial',
  fileFormat: 'CSV'
});
```

2. **Exportação para Assets:**

- Alternativamente, salve os resultados como assets para reutilização no GEE.
- Exemplo:

```
Export.table.toAsset({
  collection: estatisticas,
  description: 'Estatisticas_Asset',
  assetId: 'users/usuario/estatisticas_area'
});
```

3. **Configuração de Parâmetros de Exportação:**

- Ao exportar shapefiles, defina parâmetros como escala e região de interesse.
- Exemplo:

```
Export.table.toDrive({
  collection: colecao,
  description: 'Shapefile_Regioes',
  fileFormat: 'SHP'
});
```

A manipulação de dados vetoriais no Google Earth Engine oferece um conjunto robusto de ferramentas para análise geoespacial, desde a importação e criação de feições até o processamento avançado e a exportação de resultados. Combinando esses recursos, é possível realizar análises detalhadas e integrar dados raster e vetoriais em projetos de alta complexidade.