INTRODUÇÃO AO GOOGLE EARTH ENGINE

DISCIPLINA OPTATIVA

CURSO DE ENGENHARIA HÍDRICA/CDTec

UNIVERSIDADE FEDERAL DE PELOTAS (UFPel)

JUN/2020

Prof. Felipe de Lucia Lobo

Monitores: Edgar Ramalho

Juliano Sinotti

EMENTA

Introdução à principal plataforma de processamento de dados geográficos com computação em nuvem: *Google Earth Engine (GEE)*. Trata-se de uma plataforma que com alta capacidade de armazenamento e processamento de imagens de satélites permite ao usuário realizar diversas análises espaço-temporais, a nível regional e global, com velocidade muito maior do que com às técnicas de SIG convencionais. Nesse curso serão introduzidas técnicas de programação em *JavaScript* (linguagem do GEE) para o aluno desenvolver a capacidade de aplicar essa ferramenta em diversos projetos e trabalhos.

OBJETIVOS

- Desenvolver aspectos conceituais e práticos relacionados ao uso do GEE em geoprocessamento com ênfase em recursos hídricos.
- Caracterizar a plataforma GEE e organização dos dados espaciais (imagens de satélites e dados vetoriais).
- Introduzir a estrutura da linguagem de programação em *JavaScript*
- Apresentar diferentes possibilidades de aquisição, manipulação, integração de dados, e geração de resultados/informações.

CONTEÚDO *

MÓDULO 1:

Tutorial 1. Introdução ao Sensoriamento Remoto e *Google Earth Engine.*
Tutorial 2. Imagens de satélites e composição RGB, e *JavaScript.*
Tutorial 3. Características das imagens.
Tutorial 4. Comportamento Espectral e índices espectrais
Tutorial 5. Importação e exportação de dados.

MÓDULO 2:

Tutorial 6. Mais comandos em *JavaScript* e Coleção de imagens.
Tutorial 7. Funções de coleções de imagens.
Tutorial 8. Análise temporal.
Tutorial 9. Criação de gráficos de séries temporais.

MÓDULO 3:

Tutorial 10. Dados vetoriais no *GEE*.
Tutorial 11. Coleções vetoriais (metadados e filtros).
Tutorial 12. Funções e reducers de dados vetoriais.

* Material ainda em construção e sujeito a modificações.

ORGANIZAÇÃO DO MATERIAL

O material do curso é uma adaptação do material originalmente em inglês disponível no próprio site do GEE: https://developers.google.com/earth-engine/edu, e de vários scripts disponíveis em https://www.csc.fi/web/training/-/introduction-to-using-google-earth-engine. Além da tradução, procuramos organizar de forma acessível aos estudantes com pouca, ou nenhuma, experiência em programação ou geoprocessamento alguns conceitos básicos nesses assuntos, mas com foco no desenvolvimento da habilidade em programar, nesse caso em JavaScript que é a linguagem do GEE. Para tornar a leitura mais agradável, dividimos os tutoriais em seções de informações, que incluem:

Conteúdo teórico – breve descrição dos conceitos teóricos necessários para acompanhar os tutoriais.

Estrutura e comandos *JavaScript* – alguns comandos em JavaScript estão destacados em tabelas e caixas.

Onde encontrar mais informações – nessa seção se pode encontrar mais informações sobre o tema.

Dicas – algumas dicas e sugestões estão destacadas em balões ou caixas ao longo do texto.

Exercícios – ao fim de cada tutorial, serão propostos exercícios práticos.

Desafios!! Problemas para você mergulhar no Engine!!

Módulo 1 – Trabalhando com imagens de Satélites no *GEE*

TUTORIAL 1 - Introdução ao Sensoriamento Remoto e *Google Earth Engine*

Resumo: Nesse tutorial são apresentados os principais conceitos em Sensoriamento Remoto, características da energia eletromagnética e os sensores remotos que formam as imagens de satélite. Em seguida, apresentamos o *Google Earth Engine*, sua estrutura e funcionalidades. E por fim, damos os primeiros passos para começar a programar na plataforma *GEE*.

Conceitos

- ### O que é sensoriamento remoto?

O sensoriamento remoto (SR) é a ciência de identificação, observação, coleta e medição de objetos sem entrar em contato direto com eles. Hoje em dia, quase todos têm um sensor remoto no bolso! A câmera do seu celular! Imagine agora uma câmera semelhante à de um celular colocada em órbita em torno da Terra. É isso que são os satélites de observação da Terra. Eles registram a energia eletromagnética refletida ou emitida de objetos na terra e guardam essa energia em forma de imagens digitais. As imagens digitais são matrizes de pixels com valores numéricos armazenados em cada pixel. Elas são diferentes das fotos analógicas que registram a cena toda de uma vez em um filme.

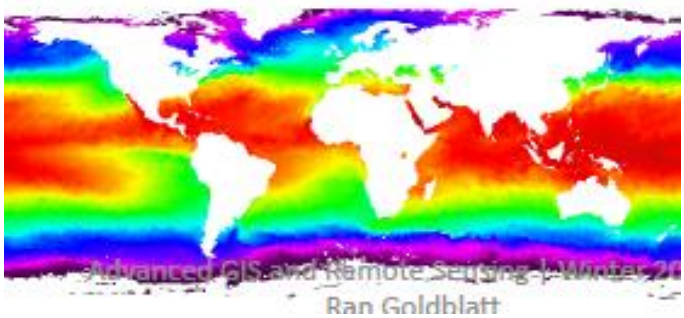
a)  b) 

Figura 1: a) Exemplo de satélite em órbita na Terra. b) Estimativa da temperatura da superfície dos oceanos a partir de sensoriamento remoto.

• Extraindo informações das imagens digitais

Existem diferentes tipos de satélites carregando a bordo diferentes tipos de câmeras. A partir desses diferentes tipos de satélites é possível extrair uma grande variedade de dados, que por sua vez, podem ser transformadas em informações. Por exemplo, a partir de dados de radares se pode extrair informações sobre a topografia de um terreno. Na agricultura as aplicações variam desde monitoramento de safra, como controle de pragas, irrigação etc. Com dados de calor da superfície da água é possível estimar sua temperatura superficial que são informações importantes para modelos de mudança climática.

As aplicações de SR em recursos hídricos também são variadas. Para modelagem hidrológica se pode obter informações sobre precipitação, limite da bacia hidrográfica, mapeamento do uso do solo na bacia; para o monitoramento da qualidade água, se pode obter informações sobre parâmetros de qualidade da água a partir da sua cor. Por exemplo, corpos de água muito verdes é sinal de alta concentração de clorofila-a presente na comunidade fitoplanctônica.

• Os sensores podem ser ativos ou passivos

Os sensores podem ser classificados em: **Passivos**, aqueles que captam e detectam a energia natural que é (naturalmente) refletida ou emitida a partir da superfície da Terra. A fonte mais comum de radiação para esses sensores é luz do sol refletida. E obviamente, eles só podem detectar energia quando a energia natural energia disponível está disponível! Ou seja, quando está nublado a captação dessa energia fica comprometida, pois grande parte dela é absorvida ou espalhada pelas nuvens.

Os sensores **Ativos** são aqueles que possuem uma fonte própria de energia que é emitida ao alvo (geralmente a superfície da Terra), essa energia retorna ao sensor onde é detectada. Exemplo: Um laser a bordo de um satélite é projetado na superfície da Terra e mede o tempo que leva para o laser refletir de volta ao sensor para se obter a distância e se estimar a atitude, assim funciona o LIDAR (da sigla inglesa Light Detection And Ranging).

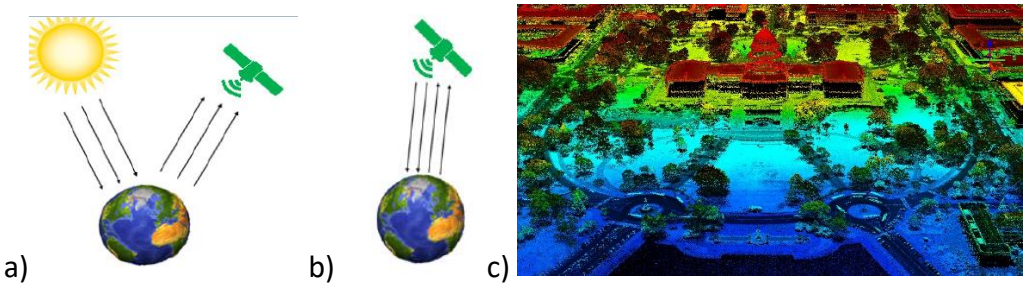


Figura 2: Representação de como funcionam os sensores (a) Passivos e (b) Ativos. Em (c) um exemplo de imagem LIDAR onde as cores indicam a altitude crescente do azul para o vermelho. Fonte: GIS Geography

- A energia do sol é composta de muitos tipos de radiação:

A 'luz', como nós habitualmente falamos, se refere a uma pequena faixa da radiação eletromagnética proveniente do sol, essa faixa é denominada de visível e é composta pelas cores vermelho, verde e azul. Na linguagem do sensoriamento remoto essas três faixas são conhecidas como RGB (do inglês Red, Green e Blue). Além da faixa do visível, existe um amplo espectro magnético variando das ondas **Ultra-Violeta (UV)** e depois a faixa do **Infravermelho** até a região termal do espectro (todo objeto emite calor que é passível de ser registrado por um sensor remoto. Como vimos na Figura 1 sobre a temperatura da superfície dos mares).

Grande parte da energia que é emitida pelo sol é absorvida ou dispersa através a atmosfera antes que ela atinja a terra. Os gases atmosféricos, tais como, vapor de água e NO_2 e também aerossóis participam do processo de **atenuação** da energia solar. Pergunta: Qual gás absorve a maior parte da radiação UV prejudicial do sol?

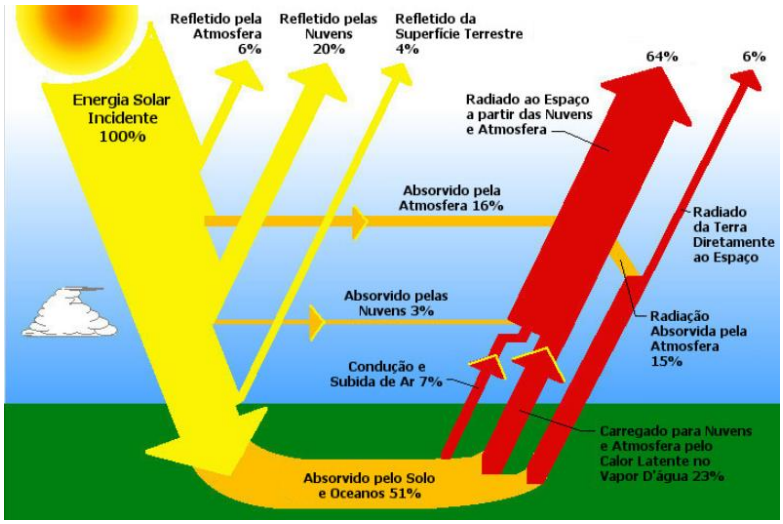


Figura 3: Balanço energético da Terra considerando as interações com a atmosfera e a superfície terrestre. Fonte: Geodesign.com.br.

No processo de interação da radiação com a atmosfera e a superfície da Terra, ela pode ser:

Transmitida: A radiação que atravessa o material ou o meio. Por exemplo, grande parte da energia do visível é transmitida ao atravessar a atmosfera.
Absorvida: Absorvido e convertido em outras formas de energia.
Espalhada: desviados em várias direções.
Refletida: retornado em um determinado ângulo. Ou seja, a superfície funciona como um espelho. Isso ocorre bastante com a superfície da água.
Emitida: geralmente em comprimentos de onda mais longos. É o que ocorre com superfícies escuras que absorvem a radiação no visível e emitem em forma de calor. Experimente usar uma camiseta preta no sol!?

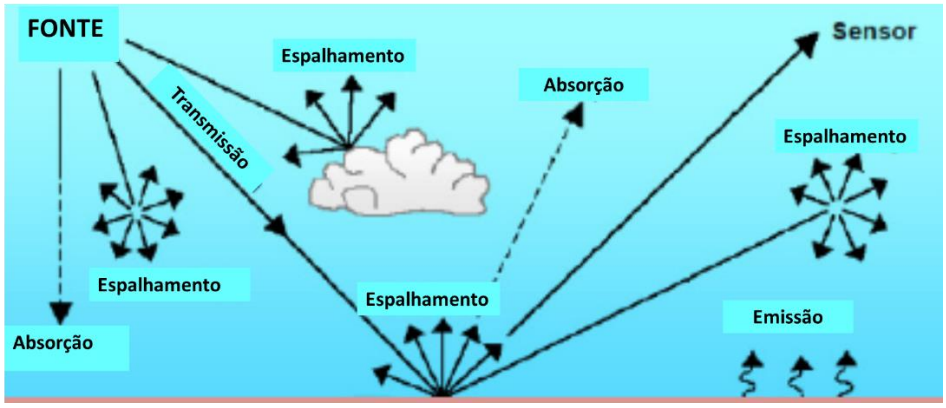


Figura 4: Processos de absorção, espalhamento, transmissão que a radiação sofre desde sua fonte (sol) até o sensor.

No caso do sensor colocado em órbita todo esse efeito da atmosfera indicado na Figura 4 ocorre também no caminho da radiação ao sensor que irá detectar essa energia.

- ## Características da radiação eletromagnética

A radiação tem propriedade de ondas eletromagnéticas. O **comprimento de onda** que é a distância de uma crista de onda à seguinte (medido em unidades de comprimento - metros, nanômetros etc.), é a principal referência sobre as faixas do espectro eletromagnético que vamos trabalhar em Sensoriamento remoto e nesse curso.

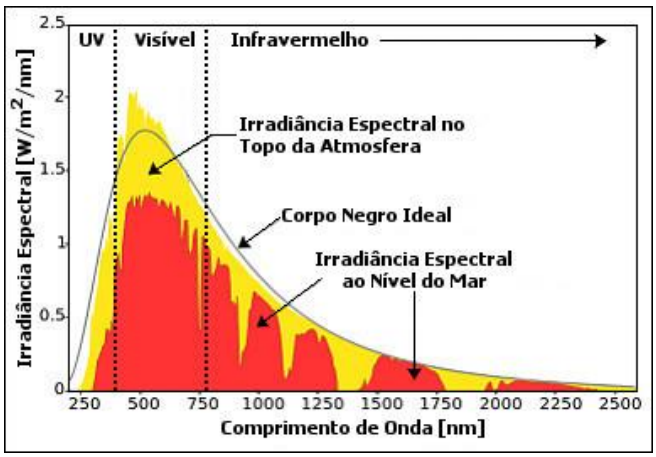


Figura 5: Exemplo da distribuição radiação proveniente do sol (Irradiância medida em Watts/m^2/nm) entre os comprimentos de onda 0 e 1500 nm (leia-se nanômetros), antes e depois de atravessar a atmosfera. Repare nos vales da curva abaixo, eles se referem às absorções dos gases e vapor de água. A faixa do visível compreende os comprimentos de onda entre 400 e ~700 nm. Mais informações aqui:

Frequência: o número de cristas que passam por um ponto fixo em um determinado período de tempo (geralmente medido em unidades hertz).

Amplitude: a altura de cada pico (medido em níveis de energia). Quanto maior a amplitude de uma onda, mais energia ela carrega.

- Sensoriamento Remoto: Investigando a radiação refletida e/ou emitida pelos objetos.

O grande objetivo do SR é obter informações sobre as características dos objetos a partir do sinal refletido por eles. Em linhas gerais, a unidade radiométrica principal é a **reflectância**, que nada mais é do que a porcentagem de energia refletida (e que o sensor detecta) pelo total de energia incidente naquele objeto. A reflectância pode ser calculada para cada comprimento de onda ou bandas dos satélites (faixas de comprimento de onda, por exemplo, a faixa entre 400 e 500 nm, define a banda espectral do azul). Com os diferentes valores de reflectância nos diferentes comprimentos de onda de objeto é possível, então, investigar sobre suas propriedades. Por exemplo, uma vegetação sadia, ou seja, bem verde aos nossos olhos, possui baixa reflectância na faixa espectral do vermelho e do azul, e mais alta na região do verde. Por isso, enxergamos as plantas na coloração esverdeada. Assim, o conjunto de valores reflectância ao longo do espectro magnético define a assinatura espectral ou comportamento espectral de um alvo (continentes, oceanos e atmosfera).

É a partir do conhecimento do comportamento espectral dos alvos que são desenvolvidos os sensores a bordo de satélites. Atualmente, existem vários satélites em órbita carregando diferentes tipos de sensores. A constelação de satélites mais conhecida, pelo fato de ter dados desde dos anos 70 é a família Landsat. Atualmente, na versão 8, esses satélites já coletaram milhões de cenas ao longo de quase cinco décadas. Mais informações sobre o Landsat aqui.

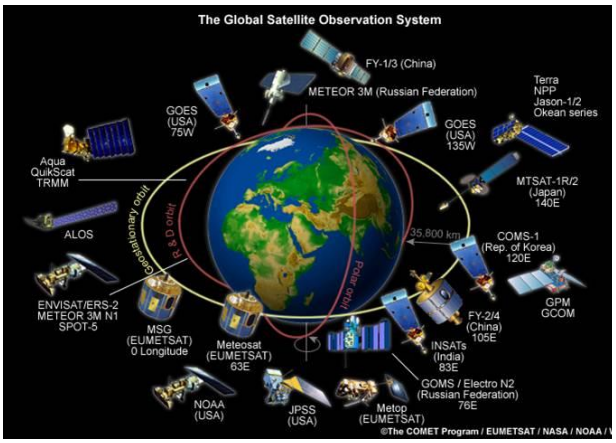


Figura 6: Exemplo de alguns satélites meteorológicos que transformam dados espectrais em informações sobre precipitação, temperatura, pressão entre outras.

• O Google Earth Engine (GEE)

É uma plataforma de computação baseada em nuvem hospedada pelo Google. GEE fornece acesso direto a um catálogo de vários petabytes de imagens de satélite e conjuntos de dados geo-espaciais, incluindo todo o catálogo Landsat do EROS (USGS / NASA), MODIS, e Sentinel-2 imagens, bem como, dados de precipitação, elevação, temperatura da superfície do mar e dados climáticos.

Além de simplesmente ser um arquivo de imagens, o GEE também fornece APIs para *JavaScript* e Python para permitir que os pesquisadores realizem análises em escala planetária da superfície da Terra. GEE é atualmente gratuito para pesquisa, educação e uso sem fins lucrativos. Por isso, o processamento de imagens de satélites com computação em nuvem está mudando o paradigma em Geoprocessamento, pois uma tarefa que se faz com as técnicas convencionais (por exemplo, baixar as imagens individualmente dos catálogos de imagens) e pode demorar horas ou dias, com o GEE se faz com algumas linhas de código e resultados imediatos!!

O GEE está implementado em uma página da web (https://code.earthengine.google.com/) e está estruturado da seguinte forma:

(1) **Code Editor**: janela de programação em *JavaScript* de capaz de acessar e processar petabytes de imagens de satélites e outros dados publicamente disponíveis. Aqui você cria o script;

(2) a janela à esquerda possui três abas: **Scripts**, onde se tem acesso a exemplos de scripts já prontos; **Docs**, documentação dos principais comandos e funções disponíveis no GEE; e **Assets**, onde podes armazenar seus arquivos geo-espaciais (imagens e vetores) no servidor GEE.

(3) a janela à direita também possui três abas: **Inspector**, onde pode se obter informações de um ponto (ao clicar) sobre as camadas que são visualizadas no mapa; **Console**, onde são impressas as informações que solicita através do comando print(), incluindo objetos não espaciais e gráficos; **Tasks**, nessa aba são geradas as tarefas de exportação dos dados do GEE para o Google Drive, por exemplo.

(4) um ambiente de desenvolvimento integrado (IDE) on-line para visualização de análises espaciais complexas usando a API Javascript (o **mapa**, propriamente dito). Também é importante notar que o mapa do Code Editor usa a projeção, Pseudo-Mercator e datum, WGS-84 (EPSG: 3857).

(5) Aqui você consegue manipular as camadas (**Layers**) alterando as configurações de visualização como a paleta de cores e transparência;

(6) nos botões indicados é possível criar geometrias (**Geometries**) dos tipos pontos, linhas e polígonos e editar as características de visualização. Essas geometrias podem ser usadas no seu script.

O que é uma API? API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "Application Programming Interface" que significa em tradução para o português "Interface de Programação de Aplicativos".

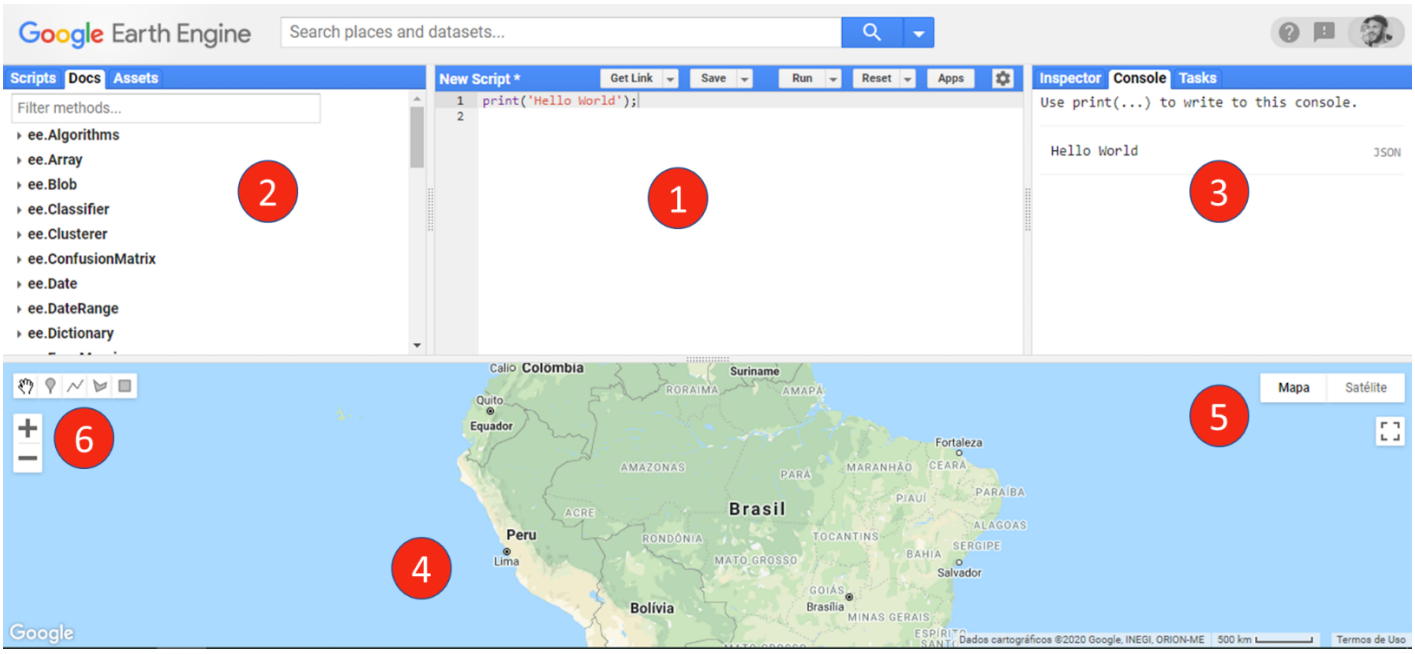


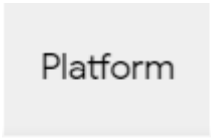
Figura 7: Página do GEE com indicação das janelas e interfaces descritos acima.

Agora vamos começar!!

- ## Acesso ao GEE!!

O primeiro passo é criar um conta de email da Google, caso ainda não tenha. Depois acesse o site (https://earthengine.google.com/) para criar um cadastro de novo usuário. A liberação do acesso demora de 1 a 3 dias, depois faça o LOGIN!

Dentro desse site você tem várias opções de uso do GEE. Vamos ver algumas aplicações de uso em diferentes interfaces. Na aba *Platform*, há 4 itens:



Overview - descreve o que é o GEE.

Code Editor - <u>interface da Fig. 7, é essa que vamos usar.</u>

Explorer - interface mais simples para visualização dos dados.

Documentation - acesso à toda documentação oficial com exemplos, tutoriais, descrição dos comandos e material educativo.

Dica:

De início, se familiarize com os dados no GEE através da aba EXPLORER.

- ## No CODE EDITOR

Nos tutoriais, todos os scripts, ou seja, texto que se insere na janela 1 (Figura 7), serão destacados com fundo cinza. Repare que as palavras podem ser destacadas em cores dependendo da sua função no script. Veja exemplo a seguir:

```
// 'Hello world!' em JavaScript

/*
O que é um script?
Um script é uma série de instruções que um computador segue para executar algum
processo. É como uma receita
- um conjunto de instruções que um cozinheiro segue um por um para criar um prato. Na
plataforma Code Editor, o
As instruções são escritas como instruções JavaScript. O computador usa essas
instruções para informar à Terra
Crie as informações que você está solicitando.
*/
// Este é um comentário. Começa com barra dupla.

/*
  Este é um comentário de várias linhas.
  Começa com asterisco com barra e termina com barra com asterisco.
*/

// Comentários são textos que você pode adicionar ao seu script para ajudar na //sua
interpretação,
// mas eles não são executados pelo computador.

// Em seguida, vem um comando (instrução / função) que será executado.
//Ele será impresso no console quando você clicar em Executar na parte superior //do
script.
```

Passo 1:

```
print('Hello World!');

// Para um exemplo relevante de sensoriamento remoto, a seguir,
//são impressos os metadados de uma imagem do Landsat 8:

print(ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_221082_20200207'));

// Você descobrirá em breve o que é ee.Image e a cadeia de caracteres entre
//parênteses.
```

Passo 2:

Tipos de variáveis em Javascript

```
Observe que as variáveis são definidas com a palavra-chave var e a marca da equação.
// Você mesmo pode definir os nomes das variáveis; A recomendação é mantê-los
compreensíveis.
// Os nomes das variáveis devem começar com uma letra.
// Variável numérica.
var my_num = 1;

// Variável string (caractere de texto): use aspas simples ou duplas, mas não misture
var my_str = 'Sou uma variável texto';

// Variável de lista: define listas com colchetes [].
// Colchetes também são usados para selecionar itens em uma lista.
// O índice zero refere-se ao primeiro item da lista.
var my_list = ['banana', 'maça', 'trigo'];
print(my_list[0]);
// Objeto / variável de dicionário. Objetos em JavaScript são dicionários de pares
chave: valor.
// Crie um objeto (ou dicionário) usando colchetes {}, por exemplo:
var my_dict = {
  'a': 'Hello',
  'num': 10,
  'floatNum': 0.1343,
  'L': my_list
};

// Os colchetes podem ser usados para acessar itens do dicionário por chave.
print(my_dict['a']);
// Ou você pode usar a notação de ponto para obter o mesmo resultado.
print(my_dict.a);

// Parênteses são usados para passar parâmetros para funções:
print(my_num, my_str, my_list, my_dict);

// Veja como cada seção executável termina em ponto e vírgula? As declarações devem
terminar em ponto-e-vírgula, ou o editor reclama.
var test = 'Está faltando algo..'

// Pressione Executar na parte superior e inspecione suas variáveis na janela do
console.
```

Passo 3:

```
// Exibindo uma única imagem do Landsat na visualização de mapa

/* A primeira novidade neste exemplo é o construtor de imagens ee.Image(). O
argumento fornecido ao construtor ee.Image() é o ID da string de uma imagem no
catálogo de dados do Earth Engine. */

/* Como obter o ID do Landsat conforme exigido pelo GEE? Você precisará saber o ID da
coleção específico no GEE, o ID do satélite Landsat,

Caminho e linha do WRS e data de aquisição. Então, nos casos em que você tem uma
imagem específica em mente e são informações do exemplo do USGS, você pode usar a
técnica neste script. Outras maneiras de encontrar imagens seguirão. */

// Defina uma variável para a imagem do Landsat que você deseja usar.

var ls_image = ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_221082_20200207');

// Defina o centro do mapa onde está a imagem, usando o comando Map.centerObject()
Map.centerObject(ls_image, 6);
// Adicione a imagem na visualização do mapa.
Map.addLayer(ls_image);

// Não se preocupe, agora que a imagem está mal visualizada.
```

- **EXERCÍCIOS**

1. Use a pesquisa de dados (‘search places and datasets...’) para descobrir qual coleção de imagens está sendo usada aqui.

2. Você pode descobrir o caminho da imagem, a linha e a data de aquisição usando o ID da imagem fornecido?

3. Encontre informações sobre as duas funções *Map.addLayer()* e *Map.centerObject()* usando a guia **Docs** à esquerda. Veja quais argumentos cada uma precisa.

4. Lembra como imprimir? Use o comando *print()* para inspecionar metadados sobre a imagem no console. Descubra quantas bandas a imagem possui e quais são os nomes das bandas.

5. Agora acesse o Moodle da disciplina (https://e-aula.ufpel.edu.br/) e responda às perguntas.

TUTORIAL 2 - Resolução dos sensores e composição RGB. Introdução ao JavaScript

Resumo: Nesse tutorial são apresentadas as quatro resoluções dos sensores e como são feitas as composições de cores utilizando os canais RGB do sistema digital de imagens. Em seguida, são apresentadas alguns comandos e funções básicas em JavaScript.

Conceitos

- ## As resoluções dos sensores

As imagens de satélites apresentam características que são definidas durante o seu desenvolvimento para observar fenômenos específicos na superfície da Terra, nos Oceanos ou na Atmosfera. Por exemplo, os satélites meteorológicos precisam fornecer informações sobre as massas de ar e nuvens em tempo quase real para as previsões de tempo. Já para o monitoramento de queimadas, são necessárias imagens com bastante detalhes do terreno. Assim os sensores podem ser caracterizados em **quatro resoluções**: Espacial, Temporal, Espectral e Radiométrica.

A **Resolução Espacial** está relacionada com a capacidade de cada sensor em detectar os objetos da superfície terrestre. Desta forma, quanto maior resolução espacial, menor o objeto distinguível pelo sensor. Em geral, se classificam os sensores em alta resolução espacial (pixels menores que 10 m), média (10 a 200 m) e baixa resolução espacial (de 200 a 5 km).

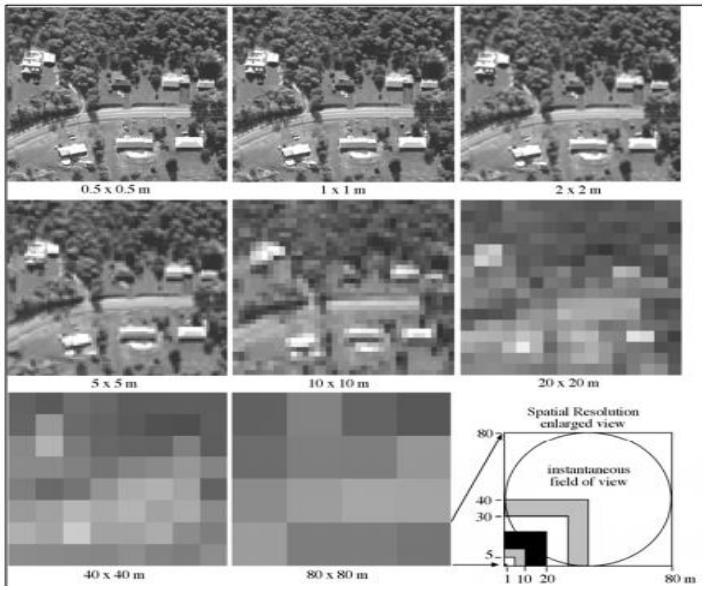


Figura 1: Exemplo de imagens com diferentes resoluções espaciais de 0,5 m a 80 m (largura do pixel).

A **Resolução Temporal** relaciona-se com o intervalo de tempo em que as informações são coletadas, ou seja, é o tempo de revisita do satélite ao mesmo local. Geralmente, é dada em dias ou para satélites geoestacionários (aqueles que ficam girando junto com a Terra e 'olhando' sempre para a mesma região do globo), o tempo de revisita pode ser em minutos. Por outro lado, há missões espaciais que fazem uma única visita à toda superfície terrestre, como foi o caso do sensor SRTM a bordo de uma nave espacial que fez o Modelo Digital de Elevação (DEM) de todo o globo durante poucos dias. Mais informações sobre essa missão **aqui.**

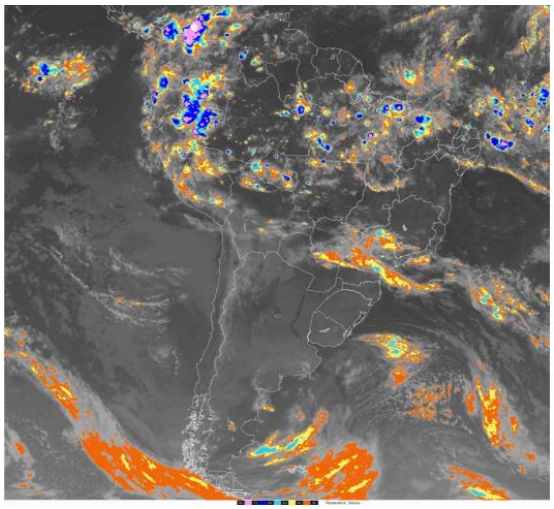


Figura 2: Imagem de sensor meteorológico que visualiza sempre a mesma região e fornece dados da atmosfera em períodos de horas ou minutos, ou seja, alta resolução temporal.

A **Resolução Espectral** define o número e a largura espectral das bandas do sensor. Quanto maior o número de bandas, maior é a resolução espectral. Em geral, os sensores são classificados em Multi-espectrais (algumas bandas, até 20, mais ou menos) e Hiper-espectrais, com centenas de bandas ao longo do espectro. Aqui nesse curso, só iremos trabalhar com sensores multi-espectrais (Landsat, Sentinel-2, e MODIS).

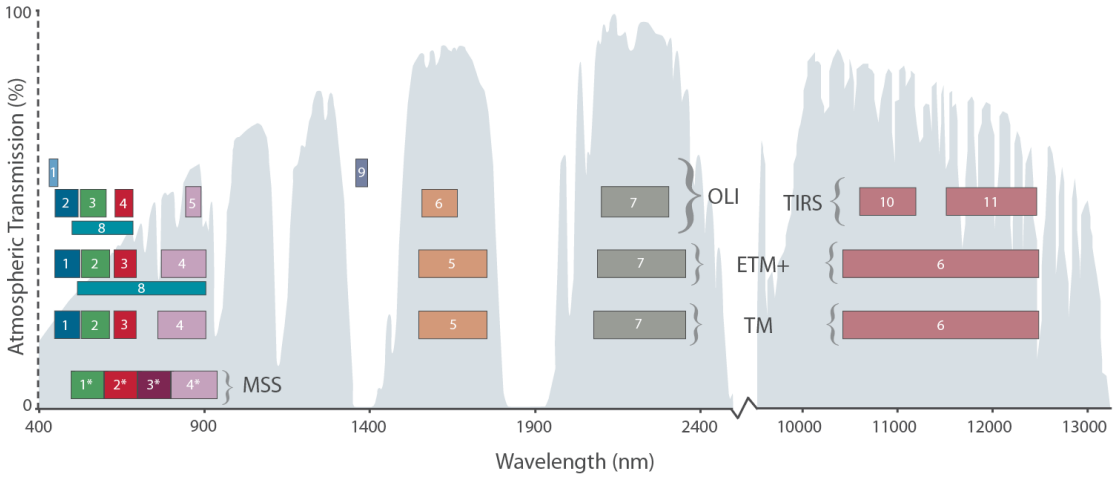


Figura 3: Indicação da posição e largura das bandas espectrais de diferentes satélites/sensores. Por exemplo, os sensores da família Landsat: L1-3 (MSS), L4-5 (TM), L7 (ETM+) e L8 (OLI).

- A **Resolução Radiométrica** caracteriza os sensores em sua capacidade de distinguir diferentes intensidades da radiação eletromagnética. Ou seja, quantos níveis de cinza o sensor é capaz de detectar. A resolução radiométrica é dada em 2^n, onde n é o número de bits definido no desenvolvimento do sensor. Por exemplo, uma imagem de resolução de 1 bit (2^1=2) só possui 2 níveis de cinza, preto e branco. Já um sensor de 8 bits, possui 256 níveis de cinza (2^8=256).

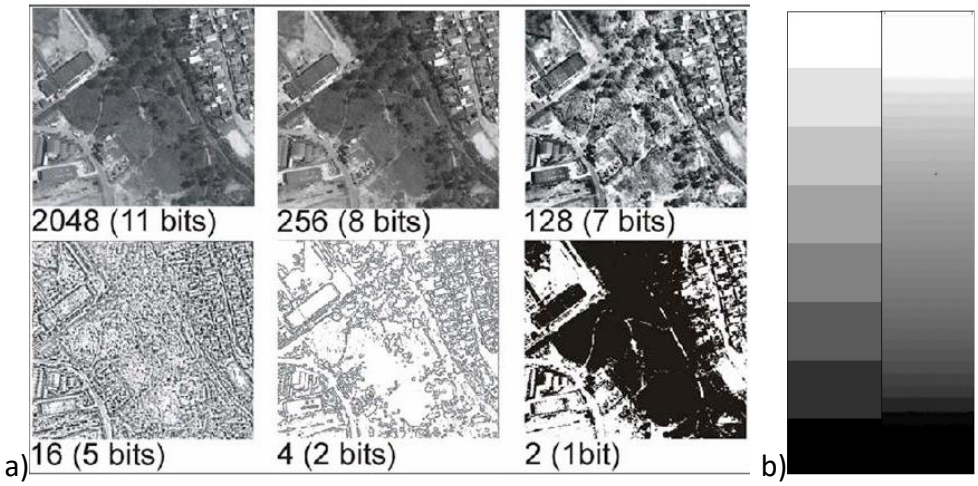


Figura 4: a) Imagens com diferentes resoluções radiométricas (de 1 a 11 bits); b) Duas paletas em níveis de cinza.

DESAFIO: Na figura 4a há um erro, encontre esse erro! Agora indique o número de bits das duas paletas na figura 4b.

• Visualizando as imagens digitais – Composição RGB

A composição RGB é a abreviatura de um sistema de cores aditivas em que o Vermelho (Red), o Verde (Green) e o Azul (Blue) são combinados de várias formas de modo a reproduzir um largo espectro cromático. O propósito principal do sistema RGB é a reprodução de cores em dispositivos eletrônicos como monitores de TV e computador, retroprojetores, scanners e câmeras digitais, assim como na fotografia tradicional.

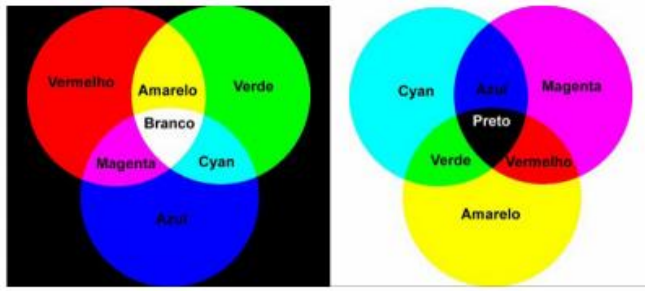


Figura 5: Sistema de cores aditivas (esquerda) usada em telas de computadores e em sensoriamento remoto. E o sistema de cores subtrativo usado em impressoras.

Nesse sistema aditivo de cores, as bandas dos sensores são colocadas nesses canais RGB do computador para formar uma composição colorida. Assim, passamos a ver as imagens em cores e não mais em níveis de cinza. Por exemplo, na figura

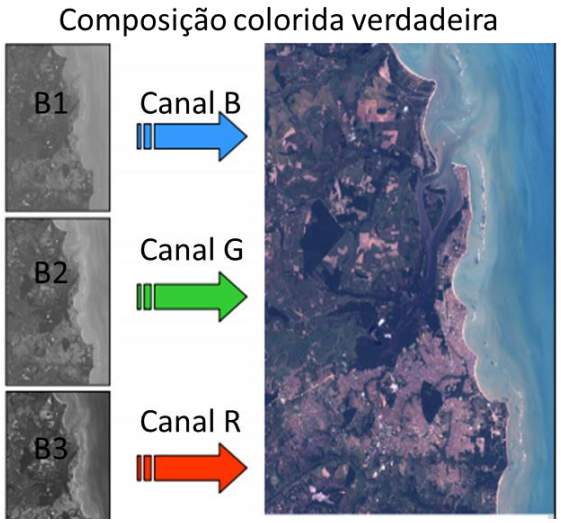


Figura 6: Criação de uma composição colorida (RGB-321) usando as bandas do Landsat-5/TM. Os comprimentos de onda centrais das bandas são B1 (490 nm, azul), B2 (550nm, verde), B3 (660 nm, vermelho). Como as bandas escolhidas correspondem ao canal em que foram inseridas, ou seja, banda do azul no canal azul e assim por diante, essa é uma composição colorida verdadeira. Qualquer outra composição de bandas é denominada de falsa-cor.

- JavaScript>

O que é?

JavaScript é uma linguagem de programação, muito utilizada para a criação de websites, que permite implementar itens complexos mostrando conteúdo que se atualiza em um intervalo de tempo, mapas interativos ou gráficos animados, etc. — você pode apostar que o JavaScript provavelmente está envolvido. JavaScript é uma **linguagem interpretada** — o código é executado de cima para baixo e o resultado da execução do código é imediatamente retornado.

Ao executar o código ele pode ser processado tanto do **lado do cliente** ou **do servidor**. Códigos do lado do cliente são executados no computador do usuário — quando uma página web é visualizada, o código do lado do cliente é baixado, executado e exibido pelo navegador. Códigos do lado do servidor, por outro lado, são executados no servidor e o resultado da execução é baixado e exibido no navegador. No GEE, nós podemos executar tanto comandos do lado do cliente como do servidor.

O núcleo da linguagem JavaScript consiste em alguns benefícios comuns da programação que permite a você fazer coisas como: Armazenar conteúdo útil em **variáveis**; Operações com pedaços de **texto** (conhecidos como "strings" em programação) e aplicar **funções**. Mais informações sobre o JS, aqui.

Para facilitar a nossa vida de programadores, existem ambientes de desenvolvimento integrados, da sigla **IDE** (em inglês, Integrated Development Environment). Uma das principais vantagens dos IDEs está

na capacidade de compilar **bibliotecas** completas de linguagem. Outra função bastante comum neste tipo de programa são os **debuggers**, que apontam os erros que ocasionalmente podem ocorrer ao escrever o código. Alguns IDEs também possuem o autocompletar. No caso do GEE, o Code Editor é um IDE implementado na Web para o API da Earth Engine em JavaScript.

Dica: Onde encontrar material sobre o GEE.

A principal fonte de informações é o site oficial do GEE (https://developers.google.com/earth-engine), onde você encontra desde guias dos principais comandos com exemplos, tutoriais, material educativo e informações sobre os catálogos de imagens disponíveis. Além disso, existem fórums (https://groups.google.com/forum/#!forum/google-earth-engine-developers) e uma página no Facebook de um grupo no Brasil (https://www.facebook.com/groups/googleearthenginebr/). Para quem gosta de vídeos, no canal do Sadeck Geotecnologias, tem explicações muito boas (https://www.youtube.com/playlist?list=PLNFvG6bTA4NReWtgC93Mh9Tw1RNG4EBMP). Se você encontrar outras fontes compartilhe com a turma!

- Agora vamos para o CODE EDITOR>

1. Comandos básicos para JavaScript

Strings (texto)

O uso de variáveis para armazenar objetos e primitivas ajuda na legibilidade do código. Por exemplo, uma variável que armazena um objeto de string é definida por aspas simples 'ou duplas "(mas não as misture), com aspas simples preferidas.

Crie uma nova string e armazene-a em uma variável chamada greetString:

```
// Use aspas simples (ou duplas) para criar uma string.
var greetString = 'Ahoy there!';

// Use parênteses para passar argumentos para funções.
print(greetString);
```

Números

Observe que as variáveis são definidas com a palavra-chave var. As variáveis também podem armazenar números:

```
// Armazenar um número em uma variável.
var number = 42;
print('The answer is:', number);
```

Neste exemplo, observe que, quando *print ()* recebe dois argumentos separados por vírgulas, cada argumento é impresso em uma linha diferente.

Listas

Defina listas com colchetes []. Uma lista de números, por exemplo:

```
// Use colchetes [] para fazer uma lista.
var listOfNumbers = [0, 1, 2, 3, 4, 5];
print('List of numbers:', listOfNumbers);
```

As listas também podem armazenar seqüências de caracteres ou outros objetos. Por exemplo:

```
// Faça uma lista de strings.
var listOfStrings = ['a', 'b', 'c', 'd'];
print('List of strings:', listOfStrings);
```

Objetos ou Dicionários

Objetos em JavaScript são dicionários de pares chave: valor.

Crie um objeto (ou dicionário) usando colchetes {}, por exemplo:

```
// Use colchetes {} para criar um dicionário de pares chave: valor.
var object = {
  foo: 'bar',
  baz: 13,
  stuff: ['this', 'that', 'the other thing']
};
print('Dictionary:', object);

// Acesse itens de dicionário usando colchetes.
print('Print foo:', object['foo']);

// Acesse itens de dicionário usando a notação de ponto.
print('Print stuff:', object.stuff);
```

Observe que você pode obter um valor de um dicionário fornecendo a chave. Este exemplo mostra como fazer isso para objetos JavaScript. Mais pra frente, você aprenderá como fazer isso para dicionários que estão no servidor Earth Engine.

Funções

As funções são outra maneira de melhorar a legibilidade e a reutilização do código, agrupando conjuntos de operações. Defina uma função com a palavra-chave *function*. Os nomes das funções começam com uma letra e têm um par de parênteses no final. As funções geralmente usam parâmetros que informam à função o que fazer. Esses parâmetros vão dentro dos parênteses ().

O conjunto de instruções que compõem a função entra entre colchetes. A palavra-chave return indica qual é a saída da função. Existem várias maneiras de declarar uma função, mas aqui usaremos algo como isto:

```
var myFunction = function(parameter1, parameter2, parameter3) {
  statement;
  statement;
  statement;
  return statement;
};
```

Vamos considerar as linhas uma a uma. A primeira linha cria uma nova função e a atribui à variável myFunction. Esta variável poderia ter sido nomeada qualquer coisa. Ele define como chamar a função posteriormente.

Os termos entre parênteses após o nome da função (ou seja, parâmetro1, parâmetro2, parâmetro3) são os nomes dos parâmetros e também poderiam ter sido nomeados, embora seja boa prática para fornecer nomes exclusivos diferentes do código fora da função. Qualquer que seja o nome, esses são os nomes que a função usará para se referir aos valores que são passados para a função quando ela é chamada.

O valor de um parâmetro depois que ele é passado para uma função é chamado de argumento. Embora as funções possam usar variáveis declaradas fora da função (variáveis globais), os argumentos da função não são visíveis fora da função. As funções podem assumir quantos parâmetros forem necessários, até zero. Aqui está um exemplo simples de uma função que apenas retorna seu argumento:

```
// A função refletir usa um único parâmetro: elemento.
var reflect = function(element) {
// Retornar o argumento.
  return element;
};
print('A good day to you!', reflect('Back at you!'));
```

Este é um exemplo de uma função definida pelo usuário. Também há muitas funções internas do Earth Engine. Explore a guia Documentos do editor de código para aprender sobre essas funções internas. Aqui está um exemplo muito simples de uma função do Earth Engine:

```
function myFunction() {
    return("Hello World!");
  }
//A função é chamada, o valor de retorno terminará em x
var x = myFunction(4, 3);
//A função retorna o produto de a e b
  function myFunction(a, b) {
    return a * b; }
  print (x)
```

```
// A função retorna um valor de temperatura
  function toCelsius(F) {
    return (5/9) * (F-32);
  }
  print ("The temperature is " + toCelsius(77) + " Celsius");
```

2. Visualizando uma imagem

Quando você adiciona uma imagem ao mapa usando *Map.addLayer ().* O Earth Engine precisa determinar como mapear os valores na (s) banda (s) da imagem para cores no visor. Se uma banda única é adicionada a um mapa, por padrão, o Earth Engine exibe a banda em escala de cinza, em que o valor mínimo é atribuído ao preto e o valor máximo é atribuído ao branco.

Se você não especificar quais devem ser os mínimos e os máximos, o Earth Engine usará os valores padrão. Para alterar a maneira como os dados são estendidos, você pode fornecer um parâmetro, **visParams**, para a chamada Map.addLayer. Este comando permite especificar os valores mínimo e máximo a serem exibidos.

Agora vamos definir uma variavel para a imagem Landsat que você deseja visualizar

Para isto, vamos utilizar os sequintes comandos no Code Editor do GEE:

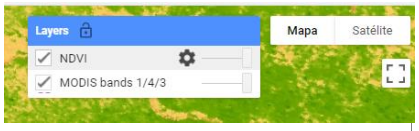
```
// Define a variable for Landsat image you wish to use.
var ls_image = ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_221082_20200207');


//Defina o centro do mapa onde está a imagem, utilize:
Map.centerObject(ls_image, 6);


//Adicione a imagem na visualização do mapa com alguns parâmetros de visualização.
Map.addLayer(ls_image, {'min': 0, 'max': 16000, 'bands': ['B4', 'B3', 'B2']});
```

Note que ao alterar o parametro de visualização ‘max’: 16000 para ‘max’: 1000 nesta composição de bandas a imagem fica mais nítida. Mais informações aqui.

Dica: a edição de algumas propriedades da composição pode ser feita diretamente no gerenciador de camadas (*Layers*), por exemplo >



3. Adicionando paletas de cores em uma imagem

As paletas permitem definir o esquema de cores para imagens de banda única. Uma paleta é uma lista delimitada por vírgula de cadeias de cores interpoladas linearmente entre os valores máximo e mínimo na visualização parâmetros (ou padrões de acordo com o tipo de banda, conforme descrito anteriormente).

Por exemplo, pixels menores que ou igual ao mínimo será exibido com a primeira cor na lista; pixels maiores ou iguais a o máximo será exibido com a última cor na lista. As cores intermediárias são esticadas linearmente para intermediar valores de pixel.

```
// Vamos trabalhar com dados de elevação SRTM.
var srtm_image = ee.Image('CGIAR/SRTM90_V4');

//Zoom para um local.
//Veja como você também pode definir o centro do mapa com o commando
// Map.setCenter em oposição a Map.centerObject. Qual é a diferença?
Map.setCenter(-52.2, -31.6, 9);

// Centro no Grand Canyon
// Exibir a imagem no mapa.
Map.addLayer(srtm_image, {}, 'no visualization');

// Colchetes vazios significa que não há parâmetros de visulização definidos. A
sequência entre aspas aparece como o nome da camada.
// Suponha que, com essa experimentação, você determine que os dados devem ser
estendidos para [0, 3000].
Map.addLayer(srtm_image, {min: 0, max: 3000}, 'custom visualization');

// Para exibir uma única banda usando uma paleta de cores, adicione uma propriedade de
paleta ao objeto visParams, como abaixo:
Map.addLayer(srtm_image, {min: 0, max: 3000, palette: ['blue', 'green', 'red']},
'custom palette');
```

Sobre código de cores na web

As cores são definidas usando o esquema HTML de valores, padrão da web. São cadeias hexadecimais que indicam a combinação de (vermelho, verde e azul). Valores RRGGBB, desta maneira: Paleta: ['FF0000', '00FFF0', '00FF00']: Mais informações aqui.

4. Carregando mapas categóricos

As paletas também são úteis para renderizar mapas com valores discretos, por exemplo, um mapa de cobertura do solo. No caso de várias classes, use a paleta para fornecer uma cor diferente para cada classe. O exemplo a seguir usa uma paleta para renderizar categorias de cobertura do solo:

```
// Carregue a cobertura do solo MODIS 2012 e selecione a classificação IGBP.
var cover = ee.Image('MODIS/051/MCD12Q1/2012_01_01')
  .select('Land_Cover_Type_1');


// Defina uma paleta para as 18 classes distintas de cobertura do solo.
var igbpPalette = [
  'aec3d4', // water
  '152106', '225129', '369b47', '30eb5b', '387242', // forest
  '6a2325', 'c3aa69', 'b76031', 'd9903d', '91af40',  // shrub, grass
  '111149', // wetlands
  'cdb33b', // croplands
  'cc0013', // urban
  '33280d', // crop mosaic
  'd7cdcc', // snow and ice
  'f7e084', // barren
  '6f6f6f'  // tundra
];
// Especifique os rótulos mínimo e máximo e a paleta de cores que corresponde aos
rótulos.
Map.setCenter(-52.229, -31.613, 5);
Map.addLayer(cover, {min: 0, max: 17, palette: igbpPalette},'IGBP classification');
```

- **EXERCÍCIOS**

1. Crie uma lista com valores de 0 a 100, a passos de 5, utilizando a função *ee.List.sequence(start, end, step).*
2. Na imagem, ls_image, tente outra visualização da imagem, por exemplo bandas 5-6-4. Em seguida, importe os parâmetros de visualização que você escolheu para o script. Depois, insira um nome para a imagem que vai aparecer no *Layers.*
3. Na imagem do SRTM, substitua os nomes da cores 'blue', 'green', e 'red', pelos seus respectivos códigos em HTML.

```
Map.addLayer(srtm_image, {min: 0, max: 3000, palette: ['blue', 'green', 'red']},
'custom palette');
```

4. Entenda a função a seguir e responda qual é o resultado da variável teste? É um número, lista, dicionário, função, variável??

```
function quad(x) {
  var a = Math.sqrt(x)
  var b = Math.pow(x,2)
  return [a,b] }
var teste = quad (4);
```

5. Agora acesse o Moodle da disciplina (https://e-aula.ufpel.edu.br/) e responda ao Questionário 2.

TUTORIAL 3 – Características das imagens

Resumo: O objetivo deste laboratório é demonstrar conceitos de resolução espacial, espectral, temporal e radiométrica. Você será apresentado aos dados de imagem de vários sensores em várias plataformas na prática.

No Tutorial 2 nós vimos os conceitos das quatro resoluções dos sensores: espacial, temporal, espectral e radiométrica. Agora, nós vamos investigar essas diferenças no GEE.

- Vamos para o CODE EDITOR>

1. Resolução Espacial

Para ver a diferença na resolução espacial resultante de diferentes sensores, visualize dados em diferentes escalas de diferentes sensores.

MODIS. Existem dois sensores MODIS (Moderate Resolution Imaging Spectro-Radiometers) a bordo de dois satélites: Terra e Aqua. Diferentes bandas MODIS produzem dados em diferentes resoluções espaciais. Para as bandas visíveis, a menor resolução comum é de 500 metros (vermelho e NIR são 250 metros).

Procure por 'MYD09GA' e importe 'MYD09GA.006 Aqua Reflectance Surface Daily Global 1km and 500m'. Nomeie a importação myd09. (Lista completa dos produtos terrestres MODIS. Observe que os conjuntos de dados Terra MODIS começam com 'MOD' e os conjuntos de dados MODIS Aqua começam com 'MYD').

Zoom no mapa para o aeroporto SFO:

```
// Defina um ponto de interesse no Aeroporto de SF (California)
var sfoPoint = ee.Geometry.Point(-122.3774, 37.6194);

// Centralize o mapa nesse ponto.
Map.centerObject(sfoPoint, 16);

// Obtenha uma imagem de refletância de superfície da coleção MODIS MYD09GA.
var modisImage = ee.Image(myd09.filterDate('2017-07-01').first());

// Crie uma lista de bandas para uma composição colorida verdadeira, RGB.
var modisBands = ['sur_refl_b01', 'sur_refl_b04', 'sur_refl_b03'];

// Defina os parametros de visualização.
var modisVis = {bands: modisBands, min: 0, max: 3000};
```

```
// Adicione ao mapa.
Map.addLayer(modisImage, modisVis, 'MODIS');
```

Observe o tamanho dos pixels em relação aos objetos no chão. (Use o mapa base do satélite para ver dados de alta resolução para comparação). Imprima o tamanho dos pixels (em metros) com o seguinte comando:

```
// Recupere a escala (tamanho do pixel) da primeira banda.
//Note que para rodar .nominalscale(), é preciso utilizar o comando .projection()
antes.
var modisScale = modisImage.select('sur_refl_b01')
    .projection().nominalScale();

print('MODIS scale:', modisScale);
```

LANDSAT/TM. O Thematic Mapper (TM) foi levado a bordo do Landsat 4-5. (Foi sucedido pelo ETM +), a bordo do Landsat 7 e pelos sensores Operational Land Imager (OLI) / Sensor Infravermelho Térmico (TIRS) a bordo do Landsat 8. Os dados da TM têm uma resolução espacial de 30 metros.

Pesquise 'landsat 5 toa' e importe o primeiro resultado (que deve ser 'USGS Landsat 5 TM Refletância TOA de nível 1 da coleção 1'. Nomeie para tm.

```
// Filtre a coleção TM5 TOA TIER1 por região e data.
// Depois, ordene pelo atributo 'ÇLOUD COVER'. E selecione a primeira imagem.
var tmImage = ee.Image(tm
    .filterBounds(Map.getCenter())
    .filterDate('2011-05-01', '2011-10-01')
    .sort('CLOUD_COVER')
    .first());

// Adicione a imagem no mapa com uma composição colorida falsa.
Map.addLayer(tmImage, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.4}, 'TM');

// Recupere a escala (tamanho do pixel) da primeira banda.
var tmScale = tmImage.select('B1')
    .projection().nominalScale();
print('TM scale:', tmScale);
```

NAIP. O Programa Nacional de Imagens da Agricultura (NAIP) é um esforço para obter imagens nos EUA em uma rotação de três anos usando sensores aéreos. As imagens têm uma resolução espacial de 1-2 metros. Pesquise 'naip' e importe o primeiro resultado (que deve ser 'NAIP: Programa Nacional de Imagens Agrícolas'. Nomeie para naip.

Como as imagens NAIP são distribuídas como quartos de Digital Ortho Quads (DOQs) em cadência irregular, carregue tudo do ano mais recente do ciclo de aquisição (2012) sobre a área de estudo e crie um mosaico com mosaic ():

```
// Filtre a coleção NAIP por região e data.
var naipImages = naip.filterDate('2012-01-01', '2012-12-31')
    .filterBounds(Map.getCenter());

// Crie um mosaico, use print(), para ver o resultado.
var naipImage = naipImages.mosaic();

// Adicione a imagem no mapa com uma composição colorida falsa.
Map.addLayer(naipImage, {bands: ['N', 'R', 'G']}, 'NAIP');

// Recupere a escala (tamanho do pixel) da primeira banda.
var naipScale = ee.Image(naipImages.first()).projection().nominalScale();
print('NAIP scale:', naipScale);
```

2. Resolução Espectral

A resolução espectral refere-se ao número e largura das bandas espectrais nas quais o sensor faz medições. Um sensor que mede o brilho em várias bandas é chamado sensor multiespectral, enquanto um sensor com muitas bandas (possivelmente centenas) é chamado sensor hiperespectral. Por exemplo, compare o **TM** multi-espectral a bordo do Landsat 5 com o **Hyperion**, um sensor hiperespectral a bordo do satélite EO-1. Importe a coleção Hyperion como hyperion e crie uma geometria (geometry) envolvente da ilha de SF.

Em seguida crie uma variável para filtrar a coleção, depois adicione ao mapa.

```
var HypImage = ee.Image(hyperion
    .filterBounds(geometry)
    .filterDate('2012-11-01', '2012-11-21');
```

Pergunta: Quantas bandas tem o Hyperion?

Para ver o número de faixas em uma imagem, use:

```
// Recupere o nome de todas bandas na imagem MODIS.
var modisBands = modisImage.bandNames();


print('MODIS bands:', modisBands);

// Recupere o número de bandas.
print('Length of the bands list:', modisBands.length());
```

3. Resolução Temporal

Resolução temporal refere-se ao tempo de revisita do sensor ao mesmo local. Pense nisso como a frequência de pixels em uma série temporal em um determinado local.

MODIS. MODIS (Terra ou Aqua) produz imagens com cadência aproximadamente diária. Para ver a série temporal de imagens em um local, você pode imprimir () o ImageCollection, filtrado para sua área e período de interesse. Por exemplo, para ver as imagens MODIS em 2011:

```
// Filtre a coleção MODIS para um intervalo de 1 ano.
var modisSeries = myd09.filterDate('2011-01-01', '2011-12-31');


print('MODIS series:', modisSeries);
```

Expanda a propriedade de recursos do ImageCollection impresso para ver uma Lista de todas as imagens da coleção. Observe que a data de cada imagem faz parte do nome do arquivo. Observe a cadência diária. Observe que cada imagem MODIS é um mosaico global, portanto, não há necessidade de filtrar por local.

Landsat. Os Landsats (5 e posteriores) produzem imagens a cada 16 dias. O TM e o MSS estão no mesmo satélite (Landsat 5), portanto, basta imprimir a série TM para ver a resolução temporal. Ao contrário do MODIS, os dados desses sensores são produzidos em uma cena, portanto, para ver uma série temporal, é necessário filtrar por local além do tempo:

Expanda novamente a propriedade do ImageCollection impresso. Observe que uma análise cuidadosa dos IDs da imagem da TM indica o dia do ano (DOY) no qual a imagem foi coletada. Um método um pouco mais complicado envolve expandir cada imagem na lista, expandir suas propriedades e procurar a propriedade 'DATE_ACQUIRED'.

```
// Filtre a coleção L5 para o mesmo intervalo de 1 ano.
```

```
var tmSeries = tm
    .filterBounds(Map.getCenter())
    .filterDate('2011-01-01', '2011-12-31');


print('TM series:', tmSeries);
```

Para transformar isso em uma lista mais agradável de datas, primeiro defina uma **função** para obter uma data dos metadados de cada imagem, usando as propriedades do sistema. Depois, aplique a função com o comando **map ()** sobre o **ImageCollection**:

```
//Função para recuperar uma lista de datas de todas as imagens em uma coleção.
var getDate = function(image) {
  // Note that you need to cast the argument
  var time = ee.Image(image).get('system:time_start');
  // Return the time (in milliseconds since Jan 1, 1970) as a Date
  return ee.Date(time);
};

// Aplicar a função na coleção L5.
var dates = tmSeries.toList(100).map(getDate);
print(dates);
```

Comandos no GEE

Comandos básicos	
Comando	**Descrição**
;	Cada linha de comando termina com um ponto-e-vírgula
var	Use a palavra-chave var para criar uma variável.
function()	Define uma função com argumentos.
[]	Os colchetes são usados para especificar itens em uma lista.
[0]	O índice zero é o primeiro item da lista
Map.getCenter	Retorna as coordenadas no centro do mapa.
Map.setCenter	Centraliza a visualização do mapa em uma determinada coordenada com o nível de zoom especificado.
Map.addLayer	Adiciona um determinado objeto EE ao mapa como uma camada.
addBands	Retorna uma imagem contendo todas as bandas copiadas da primeira entrada e as bandas selecionadas da segunda entrada, substituindo opcionalmente as bandas da primeira imagem com o mesmo nome. A nova imagem possui os metadados e a área de cobertura da primeira imagem de entrada.
Map.centerObject	Centraliza a visualização do mapa em um determinado objeto.
projection()	Retorna a projeção da geometria.
print()	Imprime os argumentos no console.
Select()	Seleciona bandas em uma imagem, retornando uma imagem apenas com as bandas escolhidas.
get()	Obtenha uma propriedade de metadados específica.

- **PRÁTICA**

Trabalhando com imagens:

Defina o centro do mapa e a localização do zoom.

```
Map.setCenter(102.42, 16.32, 6);
```

Adicione duas camadas no visor.

```
Map.addLayer(srtm, {min: 0, max: 3000}, 'srtm');
Map.addLayer(forest, {bands: 'treecover2000', max: 100}, 'treecover');
```

EXERCÍCIO 1:

1. Use a barra de pesquisa para encontrar os dois conjuntos de dados necessários:
    - Dados de elevação digital SRTM 30m
    - Hansen Global Forest Change v.1.5 (2000-2017)
2. Inspecione seus metadados e importe-os para o script.
3. Nomeie as importações para que correspondam às variáveis no script.
4. Agora tente executar o script.

EXERCÍCIO 2:

Adicione uma paleta de cores para cada uma das camadas e salve-a no script.

Como encontrar os metadados da imagem?

Carregue uma imagem.

```
var image = ee.Image('LANDSAT/LC08/C01/T1/LC08_044034_20140318');
```

Obtenha informações sobre as bandas como uma lista.

```
var bandNames = image.bandNames();
print('Band names: ', bandNames); // ee.List of band names
```

Obtenha informações de projeção da banda 1.

```
var b1proj = image.select('B1').projection();
print('Band 1 projection: ', b1proj); // ee.Projection object
```

Obtenha informações de escala (em metros) da banda 1.

```
var b1scale = image.select('B1').projection().nominalScale();
print('Band 1 scale: ', b1scale); // ee.Number
```

Observe que bandas diferentes podem ter projeções e escalas diferentes.

```
var b8scale = image.select('B8').projection().nominalScale();
print('Band 8 scale: ', b8scale); // ee.Number
```

Obtenha uma lista de todas as propriedades de metadados.

```
var properties = image.propertyNames();
print('Metadata properties: ', properties); // ee.List of metadata properties
```

Obtenha uma propriedade de metadados específica.

```
var cloudiness = image.get('CLOUD_COVER');
print('CLOUD_COVER: ', cloudiness); // ee.Number
```

Obtenha o carimbo de data e hora e converta-o em uma data.

```
var date = ee.Date(image.get('system:time_start'));
print('Timestamp: ', date); // ee.Date
```

```
Map.addLayer(image, {'min': 6000, 'max': 12500, 'bands': ['B4', 'B3', 'B2']}, "Image u
nder inspection");
Map.centerObject(image, 7);
```

Matemática nas imagens

O Earth Engine suporta muitos operadores matemáticos básicos e executa operações matemáticas por pixel. Quando um operador é aplicado a uma imagem, é aplicado a cada pixel não mascarado de cada banda. No caso de operações em duas imagens, a operação é aplicada apenas nos locais em que os pixels nas duas imagens não são mascarados.

Dica: Máscara é uma imagem binária (0 e 1), onde 1 se refere à região de interesse, e 0 é a área que você quer descartar da imagem.

Centralize o mapa no ponto de geometria predefinido.

```
Map.centerObject(geometry, 8)
```

Suponha que você queira transformar a elevação do SRTM de metros para pés:

```
var srtmFeet = srtm.multiply(3.2808399);
```

Você pode usar todas as operações aritméticas básicas, como adicionar, subtrair, multiplicar, dividir, pow (potência do segundo) etc.

Encontre todos os locais com mais de 500m.

```
var srtmgt500 = srtm.gt(500);
```

Produz uma saída binária 0/1, onde 1 = TRUE.

Você pode usar .gt () = maior que, .lt () = menor que, .gte () / .lte (), maior / menor que ou igual a.

Inspecione as saídas no mapa.

```
Map.addLayer(srtmFeet, {min:700, max:24000}, "SRTM in feet");
Map.addLayer(srtmgt500, {min:0, max:1, palette:["white", "red"]}, "Elev. > 500m");
```

Use um método estático para uma computação mais complexa.

```
var terrain = ee.Terrain.products(srtm);
```

Veja no console o que você obtém como saída.

```
print("Terrain", terrain);
```

Calcule NDVI (Índice de vegetação com diferença normalizada) usando a fórmula (NIR-RED) / (NIR + RED),

para mais informações, visite p. https://gisgeography.com/ndvi-normalized-difference-vegetation-index/

Carregue um composto Landsat 7 de 5 anos.

```
var landsat1999 = ee.Image('LANDSAT/LE7_TOA_5YEAR/1999_2003');
```

Faça o cálculo do NDVI

```
var ndvi1999 = landsat1999.select('B4').subtract(landsat1999.select('B3')).divide(landsat1999.select('B4').add(landsat1999.select('B3')));
Map.addLayer(ndvi1999, {min:0, max:1, palette:["white", "green"]}, 'NDVI 1999', false);
```

OBS: A operação de diferença normalizada é tão comum no sensoriamento remoto que o Earth Engine fornece um método de atalho, como será mostrado no próximo Tutorial.

TUTORIAL 4 – Comportamento espectral e índices espectrais

Resumo: O objetivo deste tutorial é apresentar o comportamento espectral dos principais alvos (ou objetos) na superfície da Terra e fornecer um tour pelos índices espectrais que podem ser usados para aprimorar fenômenos de interesse em imagens detectadas remotamente. Você será apresentado a métodos para criar índices de vegetação, água, solos e queimadas. No fim, você poderá programar índices espectrais e transformações para acentuar as informações de interesse em sua área de estudo.

1. Índices espectrais

Os índices espectrais são baseados no fato de que os espectros de refletância de diferentes coberturas de terra são diferentes. Os índices são projetados para explorar essas diferenças para acentuar determinados tipos de cobertura do solo. Considere o seguinte gráfico de espectros de refletância para vários alvos:

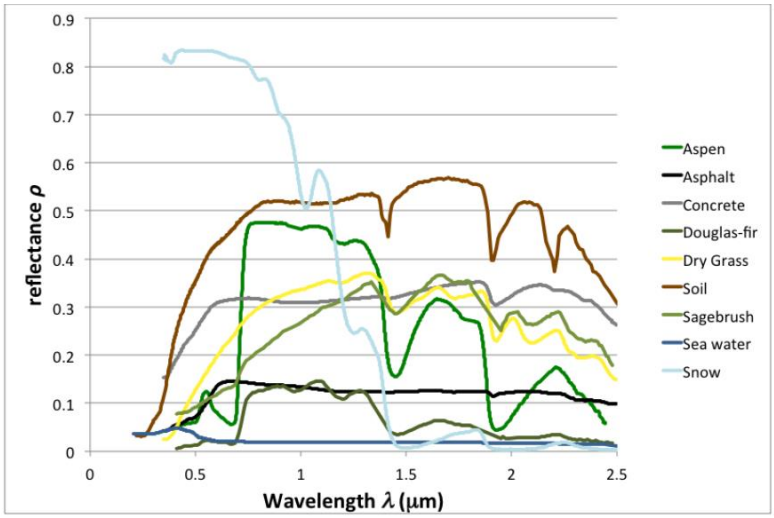


Figura 1: Comportamento espectral dos principais objetos na superfície terrestre: vegetação, solo, água, neve, concreto. No eixo Y, a reflectância de superfície (de 0 a 1) e em X, o comprimento de onda em micrometros (μm).

Observe que as coberturas da terra são separáveis em um ou mais comprimentos de onda. Observe, em particular, que as curvas da vegetação (verde) têm uma refletância relativamente alta na faixa do NIR, onde a energia radiante é dispersa pelas paredes das células. Observe também que a vegetação tem baixa refletância na faixa vermelha (0.6 μm), onde a energia radiante é absorvida pela clorofila. Essas observações motivam a formulação de índices de vegetação, por exemplo:

- Vamos para o CODE EDITOR>

a) NDVI.

O Índice de Vegetação por Diferenças Normalizadas (NDVI, do inglês Normalized Difference Vegetation Index) tem uma longa história em sensoriamento remoto. A formulação típica é:

$$NDVI = (NIR - red) / (NIR + red)$$

Onde NIR e vermelho se referem à refletância, brilho ou DN no respectivo comprimento de onda. Implemente índices deste formulário no Earth Engine com o método *normalizedDifference ()*. Primeiro, obtenha uma imagem de interesse desenhando um ponto chamado point, importando o TOA da coleção 1 do Landsat 8 como landsat8 e filtrando a coleção por localização (`-120.083, 37.850`) e metadados da cobertura de nuvens:

```
var image = ee.Image(landsat8
    .filterBounds(point)
    .filterDate('2015-06-01', '2015-09-01')
    .sort('CLOUD_COVER')
    .first());

var trueColor = {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3};
Map.addLayer(image, trueColor, 'image');
```

A computação NDVI é uma linha:

```
var ndvi = image.normalizedDifference(['B5', 'B4']);
```

Exiba a imagem NDVI com uma paleta de cores (fique à vontade para criar uma melhor):

```
var vegPalette = ['white', 'green'];
Map.addLayer(ndvi, {min: -1, max: 1, palette: vegPalette}, 'NDVI');
```

Use o *Inspector* para verificar os valores de pixel em áreas com vegetação e sem vegetação.

b) EVI.

O Índice de Vegetação Aprimorado (EVI, do inglês Enhanced Vegetation Index) é projetado para minimizar a saturação e os efeitos de fundo no NDVI. Como não é um índice de diferença normalizado, calcule-o com a expressão:

```
var evi = image.expression(
    '2.5 * ((NIR - RED) / (NIR + 6 * RED - 7.5 * BLUE + 1))', {
      'NIR': image.select('B5'),
      'RED': image.select('B4'),
      'BLUE': image.select('B2')
});
```

Observe que as bandas são referenciadas com a ajuda de um objeto que é passado como o segundo argumento para image.expression (). EVI de exibição:

```
Map.addLayer(evi, {min: -1, max: 1, palette: vegPalette}, 'EVI');
```

Compare EVI para NDVI. O que você observa?

c) NDWI.

O Índice de Água com Diferença Normalizada (NDWI, do inglês Normalized Difference Water Index) foi desenvolvido por Gao (1996) como um índice do conteúdo de água da vegetação:

$$NDWI = (NIR - SWIR) / (NIR + SWIR)$$

Computar NDWI no Earth Engine com:

```
var ndwi = image.normalizedDifference(['B5', 'B6']);
```

E exibir:

```
var waterPalette = ['white', 'blue'];
Map.addLayer(ndwi, {min: -0.5, max: 1, palette: waterPalette}, 'NDWI');
```

d) NDWBI.

É curioso que dois índices NDWI semelhantes tenham sido inventados independentemente em 1996. Para distinguir, defina o Índice Corporal de Água com Diferenças Normalizadas (NDWBI, do inglês Normalized Difference Water Body Index) como o índice descrito em McFeeters (1996):

$$NDWBI = (green - NIR) / (green + NIR)$$

Como anteriormente, implemente o NDWBI com normalizedDifference () e exiba o resultado:

```
var ndwi = image.normalizedDifference(['B3', 'B5']);

Map.addLayer(ndwi, {min: -1, max: 0.5, palette: waterPalette}, 'NDWBI');
```

Compare NDWI e NDWBI. O que você observa?

Pergunta: As quais comprimentos de onda as bandas 3 e 5 do Landsat 8 correspondem? Anote.

e) NDBI.

O Índice de Diferenças Normalizadas do Solo (NDBI, do inglês Normalized Difference Bare Index) foi desenvolvido por Zha et al. (2003) para auxiliar na diferenciação de áreas urbanas:

$$NDBI = (SWIR - NIR) / (SWIR + NIR)$$

Observe que o NDBI é o negativo do NDWI. Calcule NDBI e exiba com uma paleta adequada:

```
var ndbi = image.normalizedDifference(['B6', 'B5']);
var barePalette = waterPalette.slice().reverse();
Map.addLayer(ndbi, {min: -1, max: 0.5, palette: barePalette}, 'NDBI');
```

f) BAI

O Índice de Área Queimada (BAI, do inglês Burned Area Index) foi desenvolvido por Chuvieco et al. (2002) para auxiliar no delineamento de cicatrizes de queimaduras e avaliação da gravidade da queimadura. É baseado na distância espectral à refletância do carvão. Para examinar os índices de queimadura, carregue uma imagem de 2013 mostrando o efeito do fogo na Serra Nevadas:

```
var burnImage = ee.Image(landsat8
    .filterBounds(ee.Geometry.Point(-120.083, 37.850))
    .filterDate('2013-08-17', '2013-09-27')
    .sort('CLOUD_COVER')
    .first());


Map.addLayer(burnImage, trueColor, 'burn image');
```

Examine atentamente a exibição de cores reais desta imagem. Você consegue identificar o fogo? Caso contrário, o BAI pode ajudar. Como no EVI, use uma expressão para calcular o BAI no Earth Engine:

```
var bai = burnImage.expression(
    '1.0 / ((0.1 - RED)**2 + (0.06 - NIR)**2)', {
      'NIR': burnImage.select('B5'),
      'RED': burnImage.select('B4'),
});
```

Exibe o resultado. A área de queimada deve ficar mais nítida na visualização do BAI.

```
var burnPalette = ['green', 'blue', 'yellow', 'red'];
Map.addLayer(bai, {min: 0, max: 400, palette: burnPalette}, 'BAI');
```

- **PRÁTICA**

1. Defina uma geometria que envolva toda a represa de Santa Bárbara na cidade de Pelotas.
2. Adicione as duas imagens a seguir no mapa com Map.addLayer (Identifique a coleção de imagens, as bandas e data das imagens, edite os parâmetros de visualização):
   - COPERNICUS/S2_SR/20200601T133231_20200601T133231_T22JCL
   - COPERNICUS/S2_SR/20200626T133229_20200626T133227_T22JCL
3. Aplique o NDWBI nas duas imagens e crie uma máscara de água para cada uma delas. Adicione os resultados no mapa (certifique-se que está escolhendo as bandas corretas do índice, use as bandas do Landsat 8 do item d desse tutorial como referência).
4. Como você reparou a imagem MSI do quadrante T22JCL não cobre toda a represa. Para isso temos que fazer um mosaico de duas imagens da mesma data para mapear todo espelho d'água da represa. O código a seguir realiza um mosaico das imagens de um dia que interceptam a geometria definida:

```
var msi_filtro1 = msi.filterBounds(geometry)
.filterDate('2020-06-01','2020-06-02')
.mosaic();
```

   Para rodar o código acima é preciso definir a variável *msi*. Assim, adicione a coleção que você identificou no passo 2 e nomeie para *msi*.

5. Reaplique o NDWBI no mosaico das duas datas e compare os resultados (Crie um mosaico para cada uma das datas). Pronto, agora você tem mapeado todo espelho d'água da represa!!
6. Confira a precipitação no mês de junho em Pelotas, só de curiosidade!

Dica: 1 - Use o comando *.clip()* para cortar as imagens na área da geometria definida. 2- Use o comando *.updateMask()* na própria máscara de água para eliminar os valores zero da imagem. Por exemplo,

```
var mask_NDWBI1 = ndwbi1.updateMask(ndwbi1);
```

http://www.pelotas.com.br/noticia/nivel-da-barragem-santa-barbara-e-o-melhor-desde-2019

TUTORIAL 5 – Estatísticas, exportação e importação de imagens

Resumo: O objetivo deste tutorial é apresentar o modo que se faz para extrair informações estatísticas sobre regiões de interesse. No GEE, isso se faz através do uso de *Reducers.* Além disso, outras ferramentas como updateMask() serão apresentadas.

- Vamos para o CODE EDITOR>

Auto-Máscara

Eliminando os valores 0 ou nulos da imagem.

Para iniciar, vamos importar a coleção: Hansen Global Forest Change v1.5 (2000-2017)

Agora, vamos extrair uma única banda, ocultando valores nulos.

Selecione a banda chamada 'perda' fora do conjunto de dados Hansen

```
var loss = forest.select('loss');
```

Adicione apenas essa banda à visualização do mapa.

```
Map.addLayer(loss,{ min: 0, max:1}, "loss", false);
```

Como tornar transparentes os valores nulos?

Mascarar a banda remove os zeros.

```
Map.addLayer(loss.updateMask(loss), {palette: 'red'}, 'masked loss');
```

Inspecione o resultado na visualização do mapa e encontre informações sobre a chamada updateMask () na guia Documentos.

Redutores espaciais

Um redutor é um objeto do Earth Engine que representa uma maneira de agregar dados ou calcular uma estatística. Os redutores retornam um dicionário que pode ser impresso no console.

Vamos iniciar consultando: https://developers.google.com/earth-engine/reducers_intro

Um redutor é um objeto do Earth Engine que representa uma maneira de agregar dados ou calcular uma estatística. Os redutores retornam um dicionário que pode ser impresso no console.

Agora, crie variáveis de terreno a partir do modelo de elevação SRTM.

```
var terrain = ee.Terrain.products(srtm);
```

Calcule a média da inclinação na AOI predefinida (consulte as importações na parte superior).

O redutor reduzidoRegion () fornece estatísticas para uma região predefinida.

Definir geometria AOI.

```
var slopeMean = terrain.select('slope').reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: aoi,
  scale: 30});
```

Imprima o resultado.

```
print('Mean slope in AOI', slopeMean);
```

Altere a escala do reducer acima para 300 m.

Encontre todos os locais com mais de 200m de altitude.

```
var srtmgt200 = srtm.gt(200);
```

Gerar uma imagem na qual o valor de cada pixel é a área desse pixel em metros quadrados.

```
var area = srtmgt200.multiply(ee.Image.pixelArea())
      .reduceRegion({
       reducer: ee.Reducer.sum(),
       geometry: aoi,
       scale: 30});

print(ee.Image.pixelArea());
print('Area sqm', area);
```

Tente aplicar esse código:

```
var error = terrain.select('slope').reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: aoi,
  scale: 1}); //escala está definida para 1m

print('Bad attempt', error);
```

Corrigindo o erro:

Não faz sentido a escala ser menor que o dados de entrada que nesse caso é 30 m, mas você pode fazê-lo porque o Earth Engine fará uma nova amostra da entrada (vizinho mais próximo por padrão) na escala que você especificar. Então, estamos obtendo 900 pequenos pixels em cada pixel nativo de 30 metros.

Correção 1: altere a escala de volta para 30.

Correção 2: adicione maxPixels: 1e9 aos argumentos do reducer.

Correção 3: adicione bestEffort: true aos argumentos.

A definição de bestEffort calculará a escala de forma que maxPixels não sejam excedidos. Isso é útil para uma limpeza rápida e suja estatísticas, mas não muito mais, pois você não saberá a escala em que seu cálculo ocorreu.

A escala do cálculo é definida a partir da saída. As entradas são reamostradas conforme necessário para a escala definida na saída. Sempre especifique a escala!

O parâmetro de escala está sempre em metros.

Exercício: Descubra a elevação média da AOI.

Mais redutores espaciais

Como descobrir os valores mínimo e máximo de banda de uma única imagem?

Defina uma variável para a imagem do Landsat que você deseja usar.

```
var image = ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_189018_20181015');
```

Defina o centro do mapa onde está a imagem, usando o comando Map.centerObject ()

```
Map.centerObject(image, 6);
```

Adicione a imagem na visualização do mapa.

```
Map.addLayer(image, {'min': 80, 'max': 2800, 'bands': ['B4', 'B3', 'B2']});
```

Para obter min e max de cada banda:

```
var min = image.reduceRegion({
  reducer: ee.Reducer.min(),
  geometry: image.geometry(),
  crs: 'EPSG:4326',
  scale: 30,
  maxPixels: 1e9
  });
```

```
var max = image.reduceRegion({
  reducer: ee.Reducer.max(),
  geometry: image.geometry(),
  crs: 'EPSG:4326',
  scale: 30,
  maxPixels: 1e9
  });
```

```
print("Band min values", min);
print("Band max values", max);
```

Observe como uma única área da imagem pode ser recuperada com **image.geometry ().**

Lista de reducers disponível no GEE. Descubra para que servem algumas delas:

ee.Reducer.allNonZero()

ee.Reducer.anyNonZero()

ee.Reducer.autoHistogram(*maxBuckets*, *minBucketWidth*, *maxRaw*)

ee.Reducer.count()

ee.Reducer.countDistinct()

ee.Reducer.countEvery()

ee.Reducer.countRuns()

ee.Reducer.first()

ee.Reducer.firstNonNull()

ee.Reducer.fixedHistogram(min, max, steps)

ee.Reducer.frequencyHistogram()

ee.Reducer.histogram(*maxBuckets*, *minBucketWidth*, *maxRaw*)

ee.Reducer.intervalMean(minPercentile, maxPercentile, *maxBuckets*, *minBucketWidth*, *maxRaw*)

ee.Reducer.kendallsCorrelation(*numInputs*)

ee.Reducer.last()

ee.Reducer.lastNonNull()

ee.Reducer.linearFit()

ee.Reducer.linearRegression(numX, *numY*)

ee.Reducer.max(*numInputs*)

ee.Reducer.mean()

ee.Reducer.median(*maxBuckets*, *minBucketWidth*, *maxRaw*)

ee.Reducer.min(*numInputs*)

ee.Reducer.minMax()

ee.Reducer.mode(*maxBuckets*, *minBucketWidth*, *maxRaw*)

ee.Reducer.pearsonsCorrelation()

ee.Reducer.percentile(percentiles, *outputNames*, *maxBuckets*, *minBucketWidth*, *maxRaw*)

ee.Reducer.product()

ee.Reducer.ridgeRegression(numX, *numY*, *lambda*)

ee.Reducer.robustLinearRegression(numX, *numY*, *beta*)

ee.Reducer.sampleStdDev()

ee.Reducer.sampleVariance()

ee.Reducer.sensSlope()

ee.Reducer.skew()

ee.Reducer.spearmansCorrelation()

ee.Reducer.stdDev()

ee.Reducer.sum()

ee.Reducer.toCollection(propertyNames, *numOptional*)

ee.Reducer.toList(*tupleSize*, *numOptional*)

ee.Reducer.variance()

Exportação de dados

Carregue uma imagem landsat e selecione três bandas.

```
var landsat = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_123032_20140515')
  .select(['B4', 'B3', 'B2']);
```

Crie uma geometria representando uma região de exportação.

```
var geometry = ee.Geometry.Rectangle([116.2621, 39.8412, 116.4849,
40.01236]);
```

Observe como essa é uma maneira de definir uma geometria. Você também pode usar a ferramenta "Desenhar uma forma", Ou faça upload de um shapefile em seus ativos e importe-o para o script.

Exporta a imagem, especificando escala e região.

```
Export.image.toDrive({
  image: landsat,
  description: 'imageToDriveExample',
  scale: 30,
  region: geometry
});
```

Pressione run e vá para o painel Tasks à direita para iniciar o download.

Verifique os argumentos de Export.image.toDrive e Export.image.toAsset.

Importando Rasters

Você pode usar o Asset Manager para fazer upload de imagens ou outros conjuntos de dados raster georreferenciados no formato GeoTIFF ou TFRecord.

Você pode fazer upload de arquivos de imagem GeoTIFF de até 10 GB de tamanho na pasta de usuário do Earth Engine. Para carregar um GeoTIFF usando no Editor de código, clique no botão NOVO e selecione Upload de imagem.

Clique no botão SELECIONAR e navegue até um GeoTIFF no seu sistema de arquivos local.

Depois de iniciar o upload, uma tarefa 'Entrada de ativos' é exibida na guia Tarefas, no lado direito do Editor de código.

Passar o mouse sobre a tarefa no gerenciador de tarefas mostra a? ícone que você pode usar para verificar o status do upload.

Para cancelar um upload, clique no ícone giratório ao lado da tarefa.

Quando a ingestão estiver concluída, o ativo aparecerá na sua pasta de usuário com um ícone de imagem.

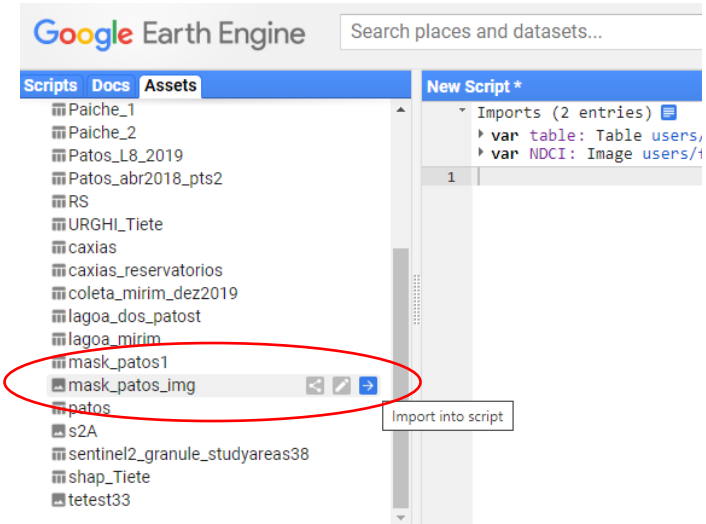
- **PRÁTICA**

1. Repetir o processo de geração das duas máscaras de água do reservatório de Sta. Bárbara nas duas datas indicadas (01 e 26 /junho/2020) na prática do Tutorial 4 (se tiver dúvidas, no começo da aula passada o exercício foi resolvido, veja aula gravada na página do curso).
2. Uma vez que tenha as duas máscaras geradas no item 1 (certifique-se que seja para o reservatório inteiro), calcule a área, **em km²**, do espelho d'água nas duas datas. Aqui você vai ter que usar o reducer de soma (*ee.Reducer.sum*) em cima da imagem (máscara), depois de aplicar o comando *ee.Image.pixelArea*, a exemplo do exercício desse Tutorial:

```
var area = srtmgt200.multiply(ee.Image.pixelArea())
      .reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: aoi,
  scale: 30});
```

* Você precisa editar esse código definindo a imagem e a geometria de interesse. Cheque a unidade de área.

3. Exporte as duas máscaras para o **Google Drive.** Uma vez na sua pasta do Drive você pode fazer o download e importar num software de SIG, como QGis, ArcGis, ou no próprio GEE.
4. Exporte as duas máscaras para o **Google Assets**. Depois importe os dois arquivos (rasters) para o seu código, a exemplo da figura a seguir:



Exemplo de como importar um arquivo do Assets para seu script. Clique naquela seta em azul.

Módulo 2 – Trabalhando com coleções de imagens no *GEE*

TUTORIAL 6 – Coleções de imagens e redutores

Resumo: Nesse tutorial vamos explorar ferramentas para processar imagens dentro de coleções ou pastas. Além de realizar a filtragem, serão explorados alguns redutores.

- Vamos para o CODE EDITOR>

Coleções de Imagens

No GEE, uma pasta com várias imagens é chamada de Coleção de imagens. Cada fonte de dados no GEE possui seu próprio ID de coleção de imagens, que você pode encontrar no catálogo de dados.

Defina o centro da visualização do mapa.

```
Map.setCenter(-119.84, 37.83, 8);
```

Adicione uma coleção de imagens na visualização do mapa.

ee.ImageCollection () chama uma coleção de imagens em vez de uma única imagem, em comparação com ee.Image ().

```
Map.addLayer(ee.ImageCollection('LANDSAT/LC08/C01/T1') // Landsat 8 Raw scenes
    .filterDate('2013-06-01', '2013-12-31'), // filtrado para um período de tempo.
    {'bands': ['B4', 'B3', 'B2'], 'min': 5000, 'max': 18000});
```

Observe como você pode adicionar parâmetros visuais, como fez antes com a imagem única.
Além disso, você está usando um filtro de tempo chamado filterDate. Veremos mais opções de filtragem nos próximos scripts.

- **PRÁTICA**

Usando o exemplo acima e o catálogo de dados, encontre e exiba a coleção do **Sentinel-2**.
Filtre-o para um período de sua escolha.

Filtragem espacial e temporal

Aplique alguns filtros nas coleções de imagens.

Abaixo um exemplo de filtragem de uma coleção de imagens para um período e local específicos por órbita (Path) e ponto (Row).

```
var collection = ee.ImageCollection('LANDSAT/LC08/C01/T1')
    .filterDate('2014-03-01', '2014-08-01')
    .filter(ee.Filter.eq('WRS_PATH', 44))
    .filter(ee.Filter.eq('WRS_ROW', 34));
```

Ou mesmo com um filtro de geometria de pontos (o ponto foi criado com a ferramenta "Adicionar marcador", veja o canto superior esquerdo da visualização do mapa.)

```
var collection2 = ee.ImageCollection('LANDSAT/LC08/C01/T1')
.filterDate('2014-03-01', '2014-08-01')
.filterBounds(point);
```

point é o nome da geometria do ponto, veja as variáveis no topo.

Também uma geometria de polígono pode ser usada (o polígono foi criado com a ferramenta "Desenhar uma forma"):

```
var collection3 = ee.ImageCollection('LANDSAT/LC08/C01/T1')
.filterDate('2014-03-01', '2014-08-01')
.filterBounds(area);    // area é o nome do polígono
```

Adicione as coleções acima à visualização do mapa: Centralize o mapa na coleção filtrada com map.CenterObject () (uma opção para Map.SetCenter)

```
Map.centerObject(collection2, 7);
```

Chame Map.addLayer para cada uma das coleções. Veja como você pode nomear uma camada no final de cada linha.

```
Map.addLayer(collection, {'bands': ['B4', 'B3', 'B2'], 'min': 5000, 'max': 18000}, 'wi
th WRS filter');
Map.addLayer(collection2, {'bands': ['B4', 'B3', 'B2'], 'min': 5000, 'max': 18000}, 'w
ith point filter');
Map.addLayer(collection3, {'bands': ['B4', 'B3', 'B2'], 'min': 5000, 'max': 18000}, 'w
ith polygon filter');
```

- **PRÁTICA**

Usando o exemplo de coleção de imagens do Sentinel-2 do script anterior, filtre-o com um ponto ou área/geometria de sua escolha (você pode modificar o script anterior ou adicionar a este.)

Metadados sobre coleções de imagens

Como em Imagens, existem várias maneiras de obter informações sobre uma coleção de imagens.
A coleção pode ser impressa diretamente no console, mas a impressão do console é limitada a 5000 elementos.
Coleções maiores que 5000 imagens precisarão ser filtradas antes da impressão.
Imprimir uma coleção grande será correspondentemente mais lento.

O exemplo a seguir mostra várias maneiras de obter informações sobre coleções de imagens:

Carregue uma ImageCollection do Landsat 8 (USGS Landsat 8 Collection 1 Tier 1 TOA Reflectance) para uma única linha de caminho.

```
var collection = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')
.filter(ee.Filter.eq('WRS_PATH', 44))
.filter(ee.Filter.eq('WRS_ROW', 34))
.filterDate('2014-03-01', '2014-08-01');
print('Collection: ', collection);
```

Obtenha o número de imagens.

```
var count = collection.size();
print('Count: ', count);
```

Obtenha o período das imagens na coleção.

```
var range = collection.reduceColumns(ee.Reducer.minMax(), ["system:time_start"]);
print('Date range: ', ee.Date(range.get('min')), ee.Date(range.get('max')));
```

NB: system: time_start é o carimbo de data / hora do Earth Engine em milissegundos desde a época do UNIX. Veja https://en.wikipedia.org/wiki/Unix_time. Pode ser alterado para uma data legível por humanos com ee.Date ()

Obtenha estatísticas para uma propriedade das imagens na coleção.

```
var sunStats = collection.aggregate_stats('SUN_ELEVATION');
print('Sun elevation statistics: ', sunStats);
```

Classifique por uma propriedade de cobertura de nuvens e obtenha a imagem menos nublada.

```
var image = ee.Image(collection.sort('CLOUD_COVER').first());
print('Least cloudy image: ', image);
```

Limite a coleção às 10 imagens mais recentes.

```
var recent = collection.sort('system:time_start', false).limit(10);
print('Recent images: ', recent);
```

Consulte as propriedades LANDSAT / LC08 / C01 / T1_TOA no catálogo de dados para entender de onde as diferentes propriedades são derivadas.

Isolando uma imagem: menor cobertura de nuvens a partir de um período de tempo específico.

A região de interesse (roi) predefinida (consulte importações) ou exclua-o e crie um novo ponto de interesse.
Observe que a coleção de imagens também já está nas importações.

Centralize o mapa no roi.

```
Map.centerObject(roi, 12);
```

Filtre a coleção de imagens com sua geometria e as datas fornecidas.
Observe que você precisa converter a saída para ee.Image.

```
var image = ee.Image(l8
    .filterBounds(roi)
    .aside(print) // Isso está OK * após * a filtragem por roi.
    .filterDate('2017-01-01', '2017-12-31'));
     //ordenar por cobertura de nuvens
     // retira o primeiro da pilha, ou seja, menos nublado
```

Defina uma visualização de cores reais para o Landsat 8.

```
var trueColorVis = {min: 0, max: 0.3, bands: ['B4', 'B3', 'B2']};

Map.addLayer(image, trueColorVis, 'Least cloudy image');
```

- **PRÁTICA**

Edite o script acima para classificar a coleção de imagens pela propriedade CLOUD_COVER e tire a primeira imagem menos nublada. Veja o script anterior para obter ajuda.
Imprima os metadados da imagem e adicione a imagem na visualização do mapa.

Reduzindo coleções de imagens

Ao chamar uma coleção de imagens, cada pixel no mapa é derivado de uma pilha de pixels.
O comportamento padrão do GEE é selecionar o pixel disponível mais recente (da cena mais recente na pilha).

Podemos alterar esse comportamento usando um redutor. Lembre-se de como os redutores são o caminho para agregar dados ao longo do tempo, espaço, faixas, matrizes e outras estruturas de dados no Earth Engine.
As reduções da coleção de imagens ocorrem com o tempo. Por exemplo: o redutor de mediana tem o benefício de remover nuvens (que têm um valor alto) e sombras (que têm um valor baixo).

- **EXERCÍCIO**

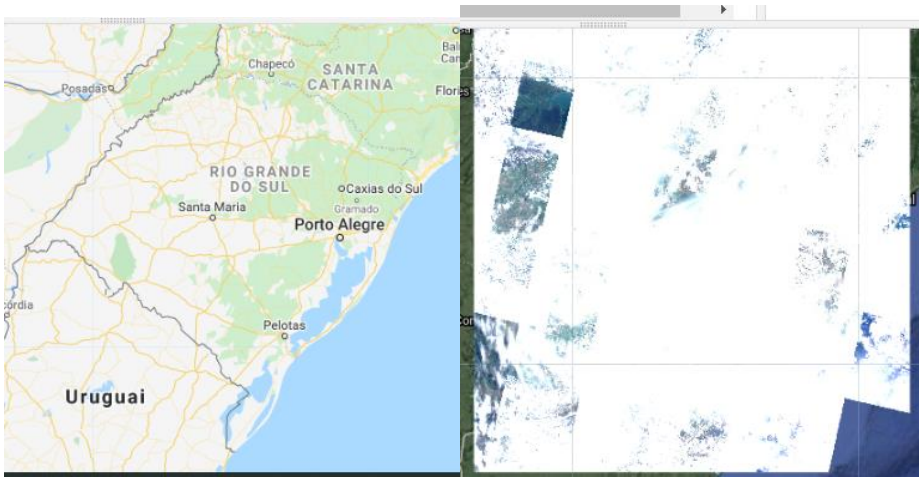
1. Defina o centro do mapa e o zoom.

```
//Edite as coordenadas para um ponto de interesse.
Map.setCenter(-119.84, 37.83, 10);
```

2. Crie uma geometria que envolva o estado do RS e nomeie para **RS**.
3. Importe uma coleção de imagens de cenas brutas (Raw) do Landsat-8 como **l8**, filtre-a e faça o redutor de máximo valor, a exemplo do comando a seguir:

```
var colecaoL8 = l8.filterDate('2020-01-01', '2020-06-01')
 .filterBounds(RS)
 .filterMetadata('CLOUD_COVER', 'greater_than', 10).aside(print)
 .max().aside(print)
```

4. Adicione a **colecaoL8** no mapa com Map.addLayer(), e defina os parâmetros de visualização para uma composição colorida verdadeira. Faça o print da coleção antes para saber o número de imagens filtradas. O resultado deve ser esse aqui: (Só nuvens!!).



Repare que a filtragem por metadados pode utilizar diferentes condicionais, por exemplo, 'greater_than'(maior que), ou 'less_than' (menor que), ou ainda, 'equals' (igual a).

5. Agora, edite as duas últimas linhas do script acima para obter a mediana da coleção de imagens com cobertura menor que 10%. Você deve obter o seguinte resultado:

6. Na página do curso, responda ao questionário.

TUTORIAL 7 – Coleções de imagens II

Resumo: Nesse tutorial vamos explorar mais ferramentas para processar imagens dentro de coleções ou pastas.

- Vamos para o CODE EDITOR>

Trabalhando com variáveis

A definição de variáveis limpa seu script e facilita a leitura.

Depois de armazenar algumas informações dentro de uma variável, você pode pular muita digitação e apenas reutilize o nome da variável em todas as próximas chamadas.

Este script faz o mesmo que o anterior, mas usando variáveis.

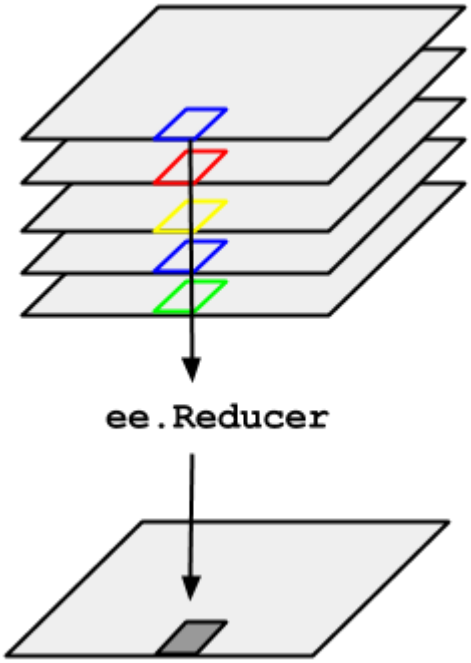
```
var landsat8 = ee.ImageCollection('LANDSAT/LC08/C01/T1');
var secondHalf2013 = landsat8.filterDate('2013-06-01', '2013-12-31');
var median = secondHalf2013.median();
var visParams = {'bands': ['B4', 'B3', 'B2'], 'min': 5000, 'max': 18000};
Map.addLayer(median, visParams);
Map.setCenter(-119.84, 37.83, 8);
```

Você acha isso mais auto-explicativo em comparação com o script anterior?

Sinta-se à vontade para adicionar comentários acima de cada linha de código, descrevendo o que ele faz.

Mais sobre como reduzir coleções de imagens

Considere a necessidade de se obter a mediana ao longo de uma série temporal de imagens representadas por uma ImageCollection. Para reduzir uma ImageCollection, use *imageCollection.reduce ()*. Isso reduz a coleção de imagens para uma imagem individual, como ilustrado na Figura 1.



Especificamente, a saída é uma imagem, de modo que cada pixel na saída seja composto pelo valor mediano de todas as imagens na coleção naquele local. Para obter outras estatísticas, como média, soma, variância, um percentil etc., o redutor apropriado deve ser selecionado e aplicado. Para estatísticas básicas como min, max, mean, etc., o ImageCollection possui métodos de atalho como *min (), max (), mean ()* etc. Eles funcionam exatamente da mesma maneira que a chamada de *reduce (),* exceto que os nomes das bandas resultantes não terão o nome do redutor anexado.

Os redutores pegam um conjunto de dados de entrada e produzem uma única saída. Alguns redutores, no entanto, produzem múltiplas saídas por exemplo : ee.Reducer.minMax (), ee.Reducer.histogram () ou ee.Reducer.toList ().

Carregar e filtrar a coleção de imagens do Sentinel-2.

```
var collection = ee.ImageCollection('COPERNICUS/S2')
    .filterDate('2016-01-01', '2016-12-31')
    .filterBounds(ee.Geometry.Point([-51.31, -31.90]));

//Reduza a coleção com o redutor minMax.
var extrema = collection.reduce(complete aqui)
print('Min max', extrema);
```

Inspecione a saída no console.

Em vez de uma saída de banda única, você tem os min e max para cada banda (originalmente 16, agora 32).

Combinando redutores

Se sua intenção é aplicar vários redutores nas mesmas entradas, é uma boa prática combinar () o redutores para eficiência. Especificamente, a chamada de combine () em um redutor com sharedInputs definido como **true** resultará em apenas uma única passagem sobre os dados. Por exemplo, para calcular a média e o desvio padrão de pixels em uma imagem, você pode usar algo como isto:

```
//Carregue uma imagem do Landsat 8.
var image = ee.Image('LANDSAT/LC08/C01/T1/LC08_044034_20140318');
//Combine os redutores de média e desvio padrão.
var reducers = ee.Reducer.mean().combine({
  reducer2: ee.Reducer.stdDev(),
  sharedInputs: true
});
//Use o redutor combinado para obter a média e o DP da imagem.
var stats = image.reduceRegion({
     reducer: reducers,
     bestEffort: true,
});
//Exiba o dicionário de médias de banda e SDs.
print('Combined reducers', stats);
```

Composição e mosaico

Em geral, composição refere-se ao processo de combinar imagens espacialmente sobrepostas em uma única imagem com base em uma função de agregação. Mosaico refere-se ao processo de montagem espacial de conjuntos de dados de imagem para produzir uma imagem espacialmente contínua. No Earth Engine, esses termos são usados de forma intercambiável, embora a composição e o mosaico sejam suportados. Mais informações: **https://developers.google.com/earth-engine/ic_composite_mosaic**

Para criar compostos Landsat simples e sem nuvens, o Earth Engine fornece o método ee.Algorithms.Landsat.simpleComposite(). Este método seleciona um subconjunto de cenas em cada local, converte em refletância TOA, aplica a pontuação simples na nuvem e mede a média dos pixels menos nublados.

//Carregue uma ImageCollection do Landsat 5 RAW por um único ano.

```
var collection = ee.ImageCollection('LANDSAT/LT05/C01/T1').filterDate('2010-01-01', '2010-12-31');
```

//Crie um composto livre de nuvem com parâmetros padrão.

```
var composite = ee.Algorithms.Landsat.simpleComposite(collection);
```

//Crie um composto livre de nuvem com parâmetros personalizados para limiar e percentil de //pontuação na nuvem.

```
var customComposite = ee.Algorithms.Landsat.simpleComposite({
      collection: collection,
      percentile: 75,
      cloudScoreRange: 5
});
```

//Exibir os compostos

```
Map.setCenter(-122.3578, 37.7726, 10);
Map.addLayer(composite, {bands: ['B4', 'B3', 'B2'], max: 128}, 'TOA
composite');
Map.addLayer(customComposite, {bands: ['B4', 'B3', 'B2'], max: 128},
    'Custom TOA composite');
```

Verifique os documentos sobre a função **simpleComposite**.

Crie um composto Landsat filtrando com a propriedade de qualidade da imagem.

Carregue os dados do Landsat 5, filtre por data e limites.

```
var collection = ee.ImageCollection('LANDSAT/LT05/C01/T2')
  .filterDate('1987-01-01', '1990-05-01')
  .filterBounds(ee.Geometry.Point(25.8544, -18.08874));
```

//Filtre também a coleção pela propriedade IMAGE_QUALITY.

```
var filtered = collection.filterMetadata('IMAGE_QUALITY', 'equals', 9);
```

//Crie dois compostos para verificar o efeito da filtragem por IMAGE_QUALITY.

```
var badComposite = ee.Algorithms.Landsat.simpleComposite(collection, 75,
3);
var goodComposite = ee.Algorithms.Landsat.simpleComposite(filtered, 75, 3);
```

//Exiba os compostos.

```
Map.setCenter(-53.8544, -31.08874, 10);
Map.addLayer(badComposite, {bands: ['B3', 'B5', 'B1'], min:20, max: 55},
'Without quality filtering');
Map.addLayer(goodComposite, {bands: ['B3', 'B5', 'B1'], min:20, max:
55},'Quality filtered');
```

//Inspecione uma diferença entre as camadas.

- **EXERCÍCIOS**

1. Compare a aplicação do mesmo reducer (ee.Reducer.max) em duas situações: a) em uma imagem e b) em uma coleção de imagens. Observe os resultados gerados.

```
// Carregue uma imagem.
var image = ee.Image('LANDSAT/LC08/C01/T1/LC08_044034_20140318')
    .select(['B4', 'B3', 'B2']);

// a) Aplique o redutor de valor máximo na imagem carregada.
var maxValue = image.reduce(ee.Reducer.max());

// Adicione o resultado no mapa.
Map.centerObject(image, 10);
Map.addLayer(maxValue, {max: 13000}, 'Maximum value image');

// b) Aplique o redutor de valor máximo na coleção da imagem.
var maxValue_col = l8.reduce(ee.Reducer.max()).aside(print);
Map.addLayer(maxValue_col,{}, 'Maximum value col');
```

2. Com base nas ferramentas de composição e mosaico que vimos até agora, faça um **mosaico** de imagens com a **menor cobertura de nuvens** possível de um **país da sua escolha** no ano de **2019** usando **qualquer coleção** de imagens (Landsat 8, Sentinel-2, MODIS ou outra coleção que ache

interessante). Lembre que as resoluções desses sensores são diferentes. Por exemplo, temos imagens diárias do MODIS, enquanto para o Landsat só a cada 16 dias.

3. Salve o script e tire um print screen do resultado.

   Dica: use o seguinte comando para obter o limite (polígono) do país que escolheu para depois filtrar a coleção de imagens e recortar o mosaico final usando o comando .clip(), por exemplo:

```
var pais = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017')
              .filterMetadata('country_na', 'equals' ,'Azerbaijan');

Map.addLayer (pais, {}, 'País');

//Observe que LSIB_SIMPLE são polígonos (limites dos países) organizados em uma coleção de
//vetores ou FeatureCollection, tópico que será abordado no Módulo 3 do curso.
```

4. Entre na página do curso e responda ao questionário. Bom trabalho!

TUTORIAL 8 – Funções em coleções de imagens

Resumo: Nesse tutorial vamos entender como as funções são aplicadas em coleções de imagens.

Funções já prontas - Normalized Difference

Como a diferença normalizada é uma coisa comum a se fazer com dados de sensoriamento remoto, existe uma função pronta para ele no GEE.

//Faça um composto Landsat-8 sem nuvens e adicione ao mapa.

```
var composite = ee.Algorithms.Landsat.simpleComposite({
  collection: l8raw.filterDate('2017-01-01', '2017-12-31'),
  asFloat: true
});
var trueColorVis = {min: 0, max: 0.3, bands: ['B4','B3','B2']};
Map.addLayer(composite, trueColorVis, 'composite');
```

//Calcule NDVI usando a função normalizedDifference ().

```
var ndvi = composite.normalizedDifference(['B5', 'B4']).rename('NDVI');
```

//Display NDVI.

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

Criando funções

As funções são uma maneira de aplicar comandos que exigem um retorno (ou seja, um objeto de saída) e uma maneira de melhorar a legibilidade e a reutilização do código, agrupando conjuntos de operações.

Defina uma função com a palavra-chave **function**. Os nomes das funções começam com uma letra e possuem um par de parênteses no final. As funções geralmente usam parâmetros que informam à função o que fazer.

Esses parâmetros vão dentro dos parênteses (). O conjunto de instruções que compõem a função entra colchetes. A palavra-chave **return** indica qual é a saída da função. Existem várias maneiras de declarar uma função, mas aqui usaremos algo como isto:

```
var myFunction = function(parameter1, parameter2, parameter3) {
  statement;
  statement;
  statement;
  return statement;
};
```

A função reflect abaixo pega um único parâmetro: elemento e o retorna

```
var reflect = function(element) {
  return element;
};
print('A good day to you!', reflect('Back at you!'));
```

Exemplo de uma função no GEE:

//Primeiro, faça um composto livre de nuvem

```
var composite = ee.Algorithms.Landsat.simpleComposite({
  collection: l8raw.filterDate('2017-01-01', '2017-12-31'),
  asFloat: true
});
```

//Defina uma função que adicionará uma banda NDVI a uma imagem do Landsat 8.

```
var addNDVI = function(image) {
  var ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
};
```

//Adicione uma banda NDVI ao composto.

```
var ndviAdded = addNDVI(composite);
```

//Exibir NDVI

```
var ndviVisParams = {bands: 'NDVI', min: 0, max: 1, palette: ['white',
'green']};
Map.addLayer(ndviAdded, ndviVisParams, 'ndviAdded');
```

Mapear uma função sobre uma coleção

Defina uma função que adicionará uma banda NDVI a uma imagem do Landsat 8.

```
var addNDVI = function(image) {
  var ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
};
```

Filtre a coleção de imagens e mapeie (usando .map) a função sobre a coleção, para que a banda NDVI seja adicionada em todas as imagens da coleção.

```
var withNDVI = l8.filterDate('2017-01-01', '2017-12-31')
    .map(addNDVI);
```

Nota: lembre-se de sempre usar map () em vez de loops de sua preferência.

Defina uma visualização de cores verdadeiras para o Landsat 8.

```
var trueColorVis = {min: 0, max: 0.3, bands: ['B4', 'B3', 'B2']};
```

Adicionar no mapa

```
Map.addLayer(withNDVI, trueColorVis, 'Collection with NDVI band');
```

Ative a guia Inspetor e clique em algum lugar no mapa.

Inspecione as imagens (e suas bandas).

Séries temporais em regiões de imagem

Para obter séries para várias regiões, use ui.Chart.image.seriesByRegion (). Por exemplo, o código a seguir plota séries temporais de temperatura da superfície da terra em três regiões que representam três tipos de cobertura da terra:

```
// Defina regiões de seu interesse
var regions = ee.FeatureCollection([
  ee.Feature(    // San Francisco.
    ee.Geometry.Rectangle(-122.45, 37.74, -122.4, 37.8), {label: 'City'}),
  ee.Feature(  // Tahoe National Forest.
    ee.Geometry.Rectangle(-121, 39.4, -120.8, 39.8), {label: 'Forest'}),
  ee.Feature(  // Black Rock Desert.
    ee.Geometry.Rectangle(-119.15, 40.8, -119, 41), {label: 'Desert'})
]);

// Carregue a coleção Landsat 8 temperatura por período de 1 ano
var temps2013 = ee.ImageCollection('LANDSAT/LC8_L1T_32DAY_TOA')
    .filterDate('2012-12-25', '2013-12-25')
    .select('B11');

// Crie um gráfico de série-temporal
var tempTimeSeries = ui.Chart.image.seriesByRegion(
    temps2013, regions, ee.Reducer.mean(), 'B11', 200, 'system:time_start', 'label')
        .setChartType('ScatterChart')
        .setOptions({
          title: 'Temperature over time in regions of the American West',
          vAxis: {title: 'Temperature (Kelvin)'},
          lineWidth: 1,
          pointSize: 4,
          series: {
            0: {color: 'FF0000'}, // urban
            1: {color: '00FF00'}, // forest
            2: {color: '0000FF'}  // desert
}});

// Display.
print(tempTimeSeries);
```

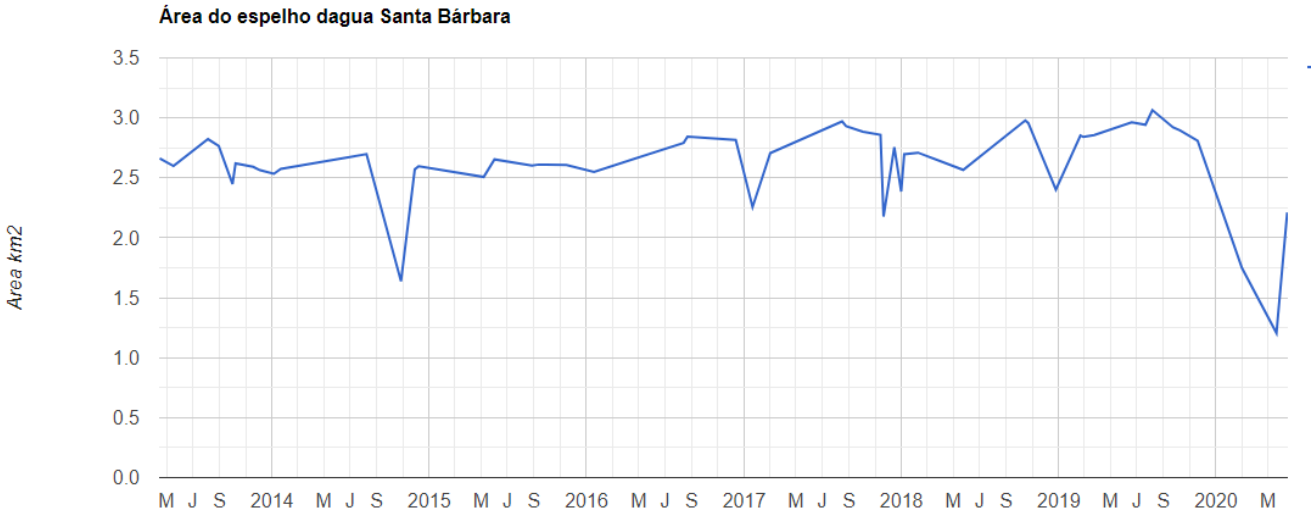
EXERCÍCIO

Consulte o Documentos para obter informações sobre a função normalizedDifference e, em seguida, crie uma função para adicionar um NDWBI (Índice de água com diferença normalizada).

No NDWBI, as bandas usadas são NIR e SWIR:

(Green-NIR) / (Green + NIR).

1. Crie uma função que adicione uma banda do NDWBI na coleção Landsat 8 Raw para o reservatório de Santa Bárbara entre janeiro de 2013 e agosto de 2020.
2. Edite a função acima para que se adicione a máscara de água (o e 1) e não o índice (valores contínuos) para cada imagem da coleção. Lembre como se faz a máscara de água no Tutorial 4.
3. Crie um gráfico que plote a soma dos pixels com valor 1 na máscara de água na região de Santa Bárbara.
4. Depois edite o seu script para que o gráfico represente área em km^2 entre 2013 e 2020. A exemplo da figura a seguir:



TUTORIAL 9 – Funções em coleções de imagens II

Resumo: Nesse tutorial vamos entender como as funções são aplicadas em coleções de imagens.

Mosaicos de qualidade

Para criar um composto que maximize uma banda arbitrária na entrada, use imageCollection.qualityMosaic ().

O método qualityMosaic () define cada pixel na composição com base em qual imagem da coleção tem um valor máximo para a banda especificada.

Nesse script, você também pode observar como as funções são mapeadas sobre as coleções de imagens.

//Esta função mascara nuvens nas imagens do Landsat 8.

```
var maskClouds = function(image) {
  var scored = ee.Algorithms.Landsat.simpleCloudScore(image);
  return image.updateMask(scored.select(['cloud']).lt(20));
};
```

//Essa função mascara nuvens e adiciona faixas de qualidade às imagens do Landsat 8.

```
var addQualityBands = function(image) {
  return maskClouds(image)
    // NDVI
    .addBands(image.normalizedDifference(['B5', 'B4']))
    // data
    .addBands(image.metadata('system:time_start'));
};
```

Carregue uma ImageCollection 2014 do Landsat 8.

Mapeie a função de máscara de nuvem e banda de qualidade sobre a coleção.

```
var collection = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')
  .filterDate('2014-06-01', '2014-12-31')
  .map(addQualityBands);
```

//Crie um composto de valor mais recente e sem nuvens.

```
var recentValueComposite = collection.qualityMosaic('system:time_start');
```

//Crie um composto de pixel mais verde.

```
var greenestPixelComposite = collection.qualityMosaic('nd');
```

//Quais são os valores de greenestPixelComposite que criamos, ou seja, o que eles representam?

//Exibir os resultados

```
Map.setCenter(-122.374, 37.8239, 12); // San Francisco Bay
var vizParams = {bands: ['B5', 'B4', 'B3'], min: 0, max: 0.4};
Map.addLayer(recentValueComposite, vizParams, 'recent value composite');
Map.addLayer(greenestPixelComposite, vizParams, 'greenest pixel
composite');
```

Use a guia Inspetor do editor de código para verificar os valores de pixel em diferentes locais no compósitos. Observe que a faixa system: time_start pode variar de acordo com o local, indicando que pixels diferentes vêm de épocas diferentes.

Funções para mascarar nuvens usando a banda QA

Sentinel-2 TOA

Este exemplo usa a banda de controle de qualidade do Sentinel-2 para mascarar a coleção.

Os sinalizadores de nuvem do Sentinel-2 são menos seletivos, portanto a coleção também é pré-filtrada pelo sinalizador CLOUDY_PIXEL_PERCENTAGE, para usar apenas grânulos relativamente sem nuvens.

//Crie a função para mascarar nuvens.

```
function maskS2clouds(image) {
  var qa = image.select('QA60');  //a banda da máscara de nuvens.

//Os bits 10 e 11 são nuvens e cirros, respectivamente

var cloudBitMask = 1 << 10;
var cirrusBitMask = 1 << 11;

//Ambos os sinalizadores devem ser definidos como zero, indicando condições claras.

var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
      .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

//Em seguida, devolva a imagem com a máscara de nuvem.

return image.updateMask(mask).divide(10000);

}

//Mapear a função em um ano de dados
//Carrega os dados de refletância do Sentinel-2 TOA.
var dataset = ee.ImageCollection('COPERNICUS/S2')
    .filterDate('2018-01-01', '2018-06-30')
// Pré-filtro para obter grânulos menos nublados.
     .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
     .map(maskS2clouds);
```

```
Map.setCenter(-9.1695, 38.6917, 12);
var rgbVis = {bands: ["B4", "B3", "B2"], min:0, max: 0.3};


//Pegue a mediana e adicione no mapa.
Map.addLayer(dataset.median(),rgbVis , 'Sentinel-2');
```

Refletância de superfície Landsat-8

Função para mascarar nuvens com base na banda pixel_qa dos dados do Landsat 8 SR.

```
Function maskL8sr(image) {
var cloudShadowBitMask = (1 << 3);
var cloudsBitMask = (1 << 5);
//Os bits 3 e 5 são sombra e nuvem, respectivamente.

Obtenha a banda de controle de qualidade de pixels.
var qa = image.select('pixel_qa');

Ambos os sinalizadores devem ser definidos como zero, indicando condições claras.
var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)
                 .and(qa.bitwiseAnd(cloudsBitMask).eq(0));
  return image.updateMask(mask);
}
var dataset = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
                  .filterDate('2016-01-01', '2016-12-31')
                  .map(maskL8sr);
var visParams = {
  bands: ['B4', 'B3', 'B2'],
  min: 0,
  max: 3000,
  gamma: 1.4,
};
Map.addLayer(dataset.median(), visParams, 'Landsat 8');
```

DESAFIO - Exportações de grandes áreas

Se a imagem exportada for grande, ela será automaticamente exportada como vários arquivos. Se você estiver exportando para GeoTIFF (s), a imagem é dividida em blocos.

O nome do arquivo de cada bloco estará no formato baseFilename-yMin-xMin onde xMin e yMin são as coordenadas de cada bloco na caixa delimitadora geral da imagem exportada.

//Adquira dados de refletância do Landsat Surface, crie um mosaico mediano.

```
var ls8_median2017 = ls8.filterDate('2017-06-01', '2017-10-31').median();
```

//Adicionar ao mapa

```
var rgb_vis = {min: 0, max: 3000, gamma: 1.5, bands: ['B4', 'B3', 'B2']};
Map.addLayer(ls8_median2017, rgb_vis, 'S2 RGB');
```

//Selecione algumas bandas para a exportação

```
var selected = ls8_median2017.select(['B4', 'B3', 'B2']);
```

//O parâmetro maxPixels visa impedir que exportações muito grandes sejam criadas inadvertidamente.

//Se o valor padrão for muito baixo para a imagem de saída pretendida, você poderá aumentar o maxPixels. Por exemplo:

```
Export.image.toDrive({
  image: selected,
  description: 'maxPixelsExample',
  folder: 'GEE',
  scale: 10,
  region: geometry,
  maxPixels: 1e11
});
```

Exportar em blocos para uma região especificada. Coordenadas delimitadoras para exportação

```
var lon_start = 30;
var lon_end = 35;
var lat_start = -8;
var lat_end = -4;
```

//Crie uma grade dentro das coordenadas delimitadoras:

//Tamanho da célula (borda de um quadrado de grade) em graus

```
var edge = 1;
var polys = [];
for (var lon = lon_start; lon < lon_end; lon += edge) {
  var x1 = lon - edge/2;
  var x2 = lon + edge/2;
  for (var lat = lat_start; lat < lat_end; lat += edge) {
    var y1 = lat - edge/2;
    var y2 = lat + edge/2;
    polys.push(ee.Geometry.Rectangle(x1, y1, x2, y2));
  }
}
var grid = ee.FeatureCollection(polys);

// Adicione no mapa
Map.centerObject(grid, 6);
Map.addLayer(grid, {palette: '000000'}, "Grid");
```

//Crie uma geometria representando uma região de exportação.

```
var features = grid.getInfo().features;
```

//Para loop chamando os recursos da grade e baixando-os:

```
for (var i = 0; i < features.length; ++i) {
var thisFeature = features[i].geometry;
```

//Exportando o mosaico em blocos para uma pasta do Google Drive chamada GEE no EPSG: 32736,
//resolução de 30 m:

```
Export.image.toDrive({
  image:ls8_median2017,
  description:'ls8_' + [i],
  folder: 'GEE',
  crs: 'EPSG:32736',
  scale: 30,
  region: thisFeature});
  }
```

Verifique o painel 'Tarefas' à direita

Módulo 3 – Trabalhando com coleções de vetores no *GEE*

TUTORIAL 10 – Coleções de vetores

Resumo: Nesse tutorial vamos explorar ferramentas para trabalhar com vetores no GEE. Incluindo criação de novos vetores, metadados, recorte de imagens usando vetores (clip) e importação/exportação.

Vetor geométrico

Criação de objetos de geometria você pode consultar por aqui:

https://developers.google.com/earth-engine/geometries

Você pode criar geometrias interativamente usando as ferramentas de geometria do Editor de código.

Para criar uma geometria programaticamente, forneça ao construtor a(s) lista (s) propriada(s) de coordenadas. Por exemplo:

```
var point = ee.Geometry.Point([1.5, 1.5]);

var lineString = ee.Geometry.LineString(
  [[-35, -10], [35, -10], [35, 10], [-35, 10]]);

var linearRing = ee.Geometry.LinearRing(
  [[-35, -10], [35, -10], [35, 10], [-35, 10], [-35, -10]]);

var rectangle = ee.Geometry.Rectangle([-40, -20, 40, 20]);

var polygon = ee.Geometry.Polygon([
  [[-5, 40], [65, 40], [65, 60], [-5, 60], [-5, 60]]
]);
```

Observe que a distinção entre LineString e LinearRing é que LinearRing é "fechado" por ter a mesma coordenada no início e no final da lista.

```
// Crie um vetor de várias partes.
var multiPoint = ee.Geometry.MultiPoint([[-121.68, 39.91], [-97.38,
40.34]]);
var geometries = multiPoint.geometries();

//Obtenha cada geometria individual da lista e imprima-a.
var pt1 = geometries.get(0);
var pt2 = geometries.get(1);
print('Point 1', pt1);
print('Point 2', pt2);
```

Vizualização vetorial

```
var point = ee.Geometry.Point([26.8, 67.1]);
var lineString = ee.Geometry.LineString(
  [[-18, 64], [15, 67], [40, 65], [52, 69]]);

var linearRing = ee.Geometry.LinearRing(
  [[-18, 64], [15, 67], [40, 65], [52, 69], [-18, 64]]);

var rectangle = ee.Geometry.Rectangle([63, 63, 73, 66]);

// Crie um polígono geodésico.
var polygon = ee.Geometry.Polygon([
  [[-5, 40], [65, 40], [65, 60], [-5, 60], [-5, 60]]]);

// Crie um polígono planar.
var planarPolygon = ee.Geometry(polygon, null, false);

// Exibir no mapa.
Map.centerObject(polygon);
Map.addLayer(point, {color: '#0000ff'}, 'point');
Map.addLayer(lineString, {color: '000000'}, 'lineString');
Map.addLayer(linearRing, {color: '#00ff00'}, 'linearRing');
Map.addLayer(rectangle, {color: '#ffff00'}, 'rectangle');
Map.addLayer(polygon, {color: 'FF0000'}, 'geodesic polygon');
Map.addLayer(planarPolygon, {color:'000000'}, 'planar polygon');
```

Verifique em:

https://developers.google.com/earth-engine/geometries_planar_geodesic para obter informações sobre Geodésicas x geometrias planas.

Metadados vetoriais

Para ver informações sobre uma geometria, use *print()*.
Para acessar as informações de maneira programática, O Earth Engine oferece vários métodos. Por exemplo:

```
// Crie um polígono geodésico.
var polygon = ee.Geometry.Polygon([
  [[-5, 40], [65, 40], [65, 60], [-5, 60], [-5, 60]]
]);

print('Polygon printout: ', polygon);

// Imprime a área do polígono em quilômetros quadrados
print('Polygon area: ', polygon.area().divide(1000 * 1000));

// Imprime o comprimento do perímetro do polígono em quilômetros.
print('Polygon perimeter: ', polygon.perimeter().divide(1000));

// Imprime a geometria como uma string GeoJSON.
print('Polygon GeoJSON: ', polygon.toGeoJSONString());

// Imprime o 'tipo' GeoJSON.
print('Geometry type: ', polygon.type());

// Imprime as coordenadas como listas.
print('Polygon coordinates: ', polygon.coordinates());

// Imprime se a geometria é geodésica.
print('Geodesic? ', polygon.geodesic());
```

Observe que o perímetro (ou comprimento) de uma geometria é retornado em metros e a área é retornado em metros quadrados, a menos que uma projeção seja especificada. Por padrão, o cálculo é executado no esferóide WGS84 e o resultado é calculado em metros ou metros quadrados.

Recorte com vetores

```
// Carregue uma imagem.
var image = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_044034_20140318');


//Dica: Use o método image.visualize () para converter uma imagem em uma imagem RGB
//de 8 bits para exibição ou exportação.
//Por exemplo, para criar uma imagem de exibição de 3 bandas em cores falsas, use:
// Crie camadas de visualização.
var imageRGB = image.visualize({bands: ['B5', 'B4', 'B3'], max: 0.5});


// Mosaico nas camadas de visualização e exibição.
Map.addLayer(imageRGB, {}, 'False color LS8', false);

// Crie um círculo desenhando um buffer de 20000 metros ao redor de um ponto.
var roi = ee.Geometry.Point([-122.4481, 37.7599]).buffer(20000);
Map.centerObject(roi, 10);

// Exibe uma versão recortada do mosaico.
Map.addLayer(imageRGB.clip(roi),{}, 'clipped');
```

Importando Vetores

Voce pode consultar por aqui: https://developers.google.com/earth-engine/importing

Tente fazer upload de um ativo de vetor. Use um arquivo vetorial de sua preferência.

Depois de ter o conjunto de dados, vá para a guia Ativos à direita -> Carregar New Table.

Selecione os arquivos de origem. Você precisará do conjunto de .shp, .dbf, .shx, .prj, arquivos etc. (uma mensagem de erro informará se arquivos de um tipo incorreto foram incluídos).

Clicar em OK inicia a ingestão de ativos na guia Tarefas à esquerda.

Exportando dados vetoriais

Para exportar um FeatureCollection para sua conta do Drive, use Export.table.toDrive ().

```
//Faça uma coleção de pontos.
var features = ee.FeatureCollection([
  ee.Feature(ee.Geometry.Point(30.41, 59.933), {name: 'Voronoi'}),
  ee.Feature(ee.Geometry.Point(-73.96, 40.781), {name: 'Thiessen'}),
  ee.Feature(ee.Geometry.Point(6.4806, 50.8012), {name: 'Dirichlet'})
]);
// Exporte o FeatureCollection para um arquivo KML.
Export.table.toDrive ({
  collection: features,
  description: 'VectorsToDriveExample',
  fileFormat: 'KML'
});
```

Verifique suas tarefas para localizar a exportação.

EXERCÍCIO:

TUTORIAL 11 – Coleções de vetores II

Resumo: Nesse tutorial vamos explorar ferramentas para trabalhar com vetores no GEE. Incluindo operações geométricas, atributos, coleções e outras formas de visualização no mapa.

Operações geométricas

```
// Crie um polígono geodésico.
var polygon = ee.Geometry.Polygon([
  [[-5, 40], [65, 40], [65, 60], [-5, 60], [-5, 60]]
]);

// Calcula um buffer do polígono.
var buffer = polygon.buffer(1000000);

// Calcula o centroide do polígono.
var centroid = polygon.centroid();

Map.centerObject(polygon, 3);
Map.addLayer(buffer, {color:'blue'}, 'buffer');
Map.addLayer(polygon, {}, 'polygon')
Map.addLayer(centroid, {color:'red'}, 'centroid');
```

Operações geométricas 2

O exemplo a seguir calcula e visualiza geometrias derivadas com base na relação entre dois polígonos:

```
// Crie duas geometrias circulares armazenando dois pontos em buffer.
var poly1 = ee.Geometry.Point([-50, 30]).buffer(1e6);
var poly2 = ee.Geometry.Point([-40, 30]).buffer(1e6);

// Exibe o polígono 1 em vermelho e o polígono 2 em azul.
Map.setCenter(-45, 30);
Map.addLayer(poly1, {color: 'FF0000'}, 'poly1');
Map.addLayer(poly2, {color: '0000FF'}, 'poly2');
```

```
// Calcule a interseção, exiba-a em azul.
var intersection = poly1.intersection(poly2, ee.ErrorMargin(1));
Map.addLayer(intersection, {color: '00FF00'}, 'intersection');

// Calcule a união, exiba-a em magenta.
var union = poly1.union(poly2, ee.ErrorMargin(1));
Map.addLayer(union, {color: 'FF00FF'}, 'union');

// Calcule a diferença, exiba em amarelo.
var diff1 = poly1.difference(poly2, ee.ErrorMargin(1));
Map.addLayer(diff1, {color: 'FFFF00'}, 'diff1');

//Computa a diferença simétrica, exibe em preto
var symDiff = poly1.symmetricDifference(poly2, ee.ErrorMargin(1));
Map.addLayer(symDiff, {color: '000000'}, 'symmetric difference');
```

Nesses exemplos, observe que o parâmetro maxError (ee.ErrorMargin) é definido como um metro para as operações geométricas.

O maxError é o erro máximo permitido, em metros, de transformações (como projeção ou reprojeção) isso pode alterar a geometria. Se uma das geometrias estiver em uma projeção diferente da outra.

O Earth Engine fará o cálculo em um sistema de coordenadas esféricas, com uma precisão de projeção fornecida por maxError. Você também pode especificar uma projeção específica para fazer o cálculo, se necessário.

Atributos

Um recurso no Earth Engine é definido como um recurso GeoJSON. Especificamente, um recurso é um objeto com uma propriedade geometry armazenando um objeto Geometry (ou nulo) e uma propriedade "properties" armazenando um dicionário de outras propriedades.

```
// Crie um ee.Geometry.
var polygon = ee.Geometry.Polygon([
  [[-35, -10], [35, -10], [35, 10], [-35, 10], [-35, -10]]
]);
```

```
// Crie um recurso a partir da geometria.
var polyFeature = ee.Feature(polygon, {id: 42, type: 'aoi'});

// Imprima o recurso no console.
print('Information about the Feature', polyFeature);

// Adicione no mapa.
Map.centerObject(polyFeature, 2)
Map.addLayer(polyFeature, {}, 'feature');

// Faça um recurso de ponto e defina algumas propriedades.
var feature = ee.Feature(ee.Geometry.Point([-122.22599, 37.17605]))
  .set('genus', 'Sequoia').set('species', 'sempervirens');

// Obtenha uma propriedade do recurso e imprima-a.
var species = feature.get('species');
print('Species', species);


// Defina uma nova propriedade.
feature = feature.set('presence', 1);

// Sobrescreve as propriedades antigas por um novo dicionário.
var newDict = {genus: 'Brachyramphus', species: 'marmoratus'};
var feature = feature.set(newDict);

// Verifique o resultado.
print('New Feature', feature);
```

Coleções de recursos

Criando FeatureCollections

Uma maneira de criar um FeatureCollection é fornecer ao construtor uma lista de recursos. Os recursos não precisam ter o mesmo tipo de geometria ou as mesmas propriedades. Por exemplo:

```
// Faça uma lista de recursos.
var features = [
  ee.Feature(ee.Geometry.Rectangle(30.01, 59.80, 30.59, 60.15), {name:
'Voronoi'}),
  ee.Feature(ee.Geometry.Point(-73.96, 40.781), {name: 'Thiessen'}),
  ee.Feature(ee.Geometry.Point(6.4806, 50.8012), {name: 'Dirichlet'})
];

// Crie um FeatureCollection fora da lista e imprima-o.
var fromList = ee.FeatureCollection(features);
print('Feature Collection 1', fromList);

//Geometrias individuais também podem ser transformadas em um FeatureCollection de
//apenas um recurso:

// Crie um FeatureCollection a partir de uma única geometria e imprima-o.
var fromGeom = ee.FeatureCollection(ee.Geometry.Point(16.37, 48.225));
print('Feature Collection 2', fromGeom);
```

Carregando FeatureCollections do catálogo de dados GEE

O Earth Engine hospeda uma variedade de conjuntos de dados de tabela.

Para carregar um conjunto de dados de tabela, forneça o ID da tabela ao construtor FeatureCollection.

Por exemplo, para carregar dados de estradas TIGER:

```
var fc = ee.FeatureCollection('TIGER/2016/Roads');
Map.setCenter(-73.9596, 40.7688, 12);
Map.addLayer(fc, {}, 'Census roads');
```

Há uma lista de conjuntos de dados vetoriais selecionados hospedados pelo Earth Engine nesta página:

https://developers.google.com/earth-engine/vector_datasets

Visualização de coleção de recursos

Voce pode consultar por aqui:

https://developers.google.com/earth-engine/feature_collections_visualizing

```
// Carrega um FeatureCollection de uma Fusion Table: 'global 200' ecorregions.
var g200 = ee.FeatureCollection('ft:11SfWB6oBS1iWGiQxEOqF_wUgBJL7Bux-pWU-mqd5');

// Exibe como padrão e com uma cor personalizada.
Map.addLayer(g200, {}, 'default display');
Map.addLayer(g200, {color: 'FF0000'}, 'colored');

// Para opções de exibição adicionais, use featureCollection.draw ():
Map.addLayer(g200.draw({color: '006600', strokeWidth: 5}), {}, 'drawn');
```

A saída de draw () é uma imagem com bandas vermelhas, verdes e azuis definidas de acordo com o parâmetro de cor especificado.

Para obter mais controle sobre como um FeatureCollection é exibido, use image.paint () que gera uma imagem com o número especificadovalor 'pintado' nele. Como alternativa, você pode fornecer o nome de uma propriedade no FeatureCollection que contém os números a pintar.

O parâmetro largura se comporta da mesma maneira: pode ser uma constante ou o nome de uma propriedade com um número para a largura da linha. Por exemplo:

```
//Crie uma imagem vazia para pintar os recursos, convertendo em byte.
var empty = ee.Image().byte();
//Pinte todas as bordas do polígono com o mesmo número e largura, exiba.
var outline = empty.paint({
  featureCollection: g200,
  color: 1,
  width: 3 });
Map.setCenter(-27, 13, 1);
Map.addLayer(outline, {palette: 'FF0000'}, 'edges');
```

Observe que a imagem vazia na qual você pinta os recursos precisa ser moldada antes da pintura.

EXERCÍCIO:

TUTORIAL 12 – Coleções de vetores III

Resumo: Nesse tutorial vamos explorar ferramentas para trabalhar com vetores no GEE. Incluindo filtros, funções, e redutores.

Visualização de coleção de recursos

Métodos para obter informações de coleções de imagens são diretamente aplicáveis às coleções de recursos.

```
// Carregar bacias hidrográficas de uma Fusion Table.
var sheds = ee.FeatureCollection('ft:1IXfrLpTHX4dtdj1LcNXjJADBB-
d93rkdJ9acSEWK');

// Exibe a tabela e imprime seu primeiro elemento.
Map.centerObject(sheds, 4);
Map.addLayer(sheds, {}, 'watersheds');
print('First', sheds.limit(1));

// Imprime o número de bacias hidrográficas.
print('Count: ', sheds.size());

// Imprimir estatísticas para uma propriedade da área.
print('Area stats:', sheds.aggregate_stats('AreaSqKm'));
```

Filtrar coleções de recursos

Filtrar um FeatureCollection é análogo a filtrar um ImageCollection.

```
// Carregar bacias hidrográficas de uma Fusion Table.
var sheds = ee.FeatureCollection('ft:1IXfrLpTHX4dtdj1LcNXjJADBB-
d93rkdJ9acSEWK');

// Defina uma região que cobre aproximadamente os EUA continentais.
var continentalUS = ee.Geometry.Rectangle(-127.18, 19.39, -62.75, 51.29);

// Filtre a tabela geograficamente com filterBounds: apenas bacias hidrográficas nos Estados
Unidos continentais.
```

```
var filtered = sheds.filterBounds(continentalUS);

// Verifique o número de bacias hidrográficas após filtrar por localização.
print('Count after filter:', filtered.size());

// Filtre para obter apenas bacias hidrográficas continentais maiores dos EUA.
var largeSheds = filtered.filter(ee.Filter.gt('AreaSqKm', 25000));

// Verifique o número de bacias hidrográficas após filtrar por tamanho e localização.
print('Count after filtering by size:', largeSheds.size());
```

Mapeamento sobre coleções de recursos

Para aplicar a mesma operação a todos os recursos em um FeatureCollection, use featureCollection.map ().

Por exemplo, para adicionar outro atributo de área a cada recurso em uma bacia hidrográfica FeatureCollection:

```
//Carregar bacias hidrográficas de uma Fusion Table.
var sheds = ee.FeatureCollection('ft:1IXfrLpTHX4dtdj1LcNXjJADBB-
d93rkdJ9acSEWK');

// Esta função calcula a área geométrica do elemento em hectares e a adiciona como uma
propriedade.
var addArea = function(feature) {
  return feature.set({areaHa: feature.geometry().area().divide(100 *
100)});
};

// Mapeie a função de adição de área em FeatureCollection.
var areaAdded = sheds.map(addArea);

// Imprime o primeiro recurso da coleção com a propriedade adicionada.
print('First feature: ', areaAdded.first());
```

Verifique as propriedades.

Reduzindo uma coleção de recursos

Para agregar dados nas propriedades de um FeatureCollection, use featureCollection.reduceColumns ().

Por exemplo, para verificar as propriedades da área nas bacias hidrográficas FeatureCollection, este código calcula o erro quadrático médio (RMSE) relativo à área computada do Earth Engine:

```
// Carregar bacias hidrográficas de uma Fusion Table e filtrar para os EUA continentais.
var sheds = ee.FeatureCollection('ft:1IXfrLpTHX4dtdj1LcNXjJADBB-d93rkdJ9acSEWK')
      .filterBounds(ee.Geometry.Rectangle(-127.18, 19.39, -62.75, 51.29));

// Esta função calcula a diferença quadrática entre uma propriedade de área já na tabela e a área calculada diretamente da geometria do recurso.
var areaDiff = function(feature) {

// Calcula a área em km2. diretamente da geometria.
var area = feature.geometry().area().divide(1000 * 1000);

// Calcule a diferença entre a área calculada e a propriedade da área na tabela.
var diff = area.subtract(feature.get('AreaSqKm'));

// Retorna o recurso com a diferença de quadrados definida para a propriedade 'diff'.
      return feature.set('diff', diff.pow(2));};

// Mapeie a função de diferença na coleção.
var rmse = ee.Number(sheds
    .map(areaDiff)

// Reduza para obter a diferença média quadrática.
.reduceColumns(ee.Reducer.mean(), ['diff'])
    .get('mean'))

// Calcule a raiz quadrada do quadrado médio para obter RMSE.
.sqrt();
// Imprime o resultado.
print('RMSE=', rmse);
```

Neste exemplo, observe que o valor de retorno de reduceColumns () é um dicionário com a chave 'mean'.

Para obter a média, converta o resultado de dictionary.get () em um número com **ee.Number** () antes de tentar

chame sqrt () nele.

Reduzindo com coleções de recursos

Para sobrepor recursos em imagens, use featureCollection.reduceRegions ().

Por exemplo, para calcular o volume de precipitação em bacias hidrográficas continentais dos EUA, use reduceRegions ()

seguido por um mapa ():

```
// Carrega uma imagem de precipitação diária em mm / dia.
var precip = ee.Image(ee.ImageCollection('NASA/ORNL/DAYMET_V3').first());

// Carregar bacias hidrográficas de uma Fusion Table e filtrar para os EUA continentais.
var sheds = ee.FeatureCollection('ft:1IXfrLpTHX4dtdj1LcNXjJADBB-
d93rkdJ9acSEWK')
  .filterBounds(ee.Geometry.Rectangle(-127.18, 19.39, -62.75, 51.29));

// Adicione a média de cada imagem como novas propriedades de cada recurso.
var withPrecip = precip.reduceRegions(sheds, ee.Reducer.mean());

// Esta função calcula a precipitação total em metros cúbicos.
var prcpVolume = function(feature) {

// Precipitação em mm / dia -> metros -> metros quadrados.
var volume = ee.Number(feature.get('prcp'))
    .divide(1000).multiply(feature.geometry().area());
  return feature.set('volume', volume);
};

var highVolume = withPrecip

// Mapeie a função na coleção.
```

```
.map(prcpVolume)

// Classifica em ordem decrescente.
.sort('volume', false)

// Obtenha apenas as 5 bacias hidrográficas de maior volume.
.limit(5);

// Imprima o FeatureCollection resultante
print(highVolume);
```

EXERCÍCIO:

Crie um composto Landsat, por exemplo com a função simpleLandsatComposite.

Crie um polígono da maneira que desejar ou use aquele que você carregou como um ativo e prenda o mosaico em seu AOI.

Crie uma seção no script que exporte a imagem RGB