Variedade de *Landforms*

## Importando pacotes e inicializando *geemap*

import os  
import ee  
import geemap  
import matplotlib as mpl  
import matplotlib.pylab as plt  
from matplotlib.colors import ListedColormap, BoundaryNorm

geemap.ee\_initialize()

### Variedade de *landforms*

### *Landforms*

Nós exploramos três metodologias de classificação de *landforms* (Anderson *et al.* 2016; Theobald *et al.* 2015; Iwahashi & Yamazaki 2022) para definirmos quais delas seriam usadas no cálculo da variedade de *landforms*. Primeiro, nós classificamos as *landforms* como Anderson *et al*. (2016), exceto a *landform* ***Flat at the bottom of steep slope***. Assim, nossas *landforms* foram:

* **3 - Cool Steep Slope**
* **4 - Warms Steep Slope**
* **5 - Cliff**
* **11 - Summit/Ridgetop**
* **13 - Slope Crest**
* **21 - Flat Hilltop**
* **22 - Gentle Slope Hilltop**
* **23 - Cool Sideslope**
* **24 - Warm Sideslope**
* **30 - Dry Flats**
* **32 - Valley/Toeslope**
* **39 - Moist Flats**
* **43 - Cool Footslope**
* **44 - Warm Sideslope**

### Variáveis classificadoras das *landforms*

Nós classificamos as *landforms* pela **inclinação do relevo** (*slope*), **aspecto** (*aspect*), **Índice de Posição Topográfica** (*TPI*) e **Índice de Umidade** (*moisture index*). As variáveis foram discretizadas em classes e combinadas para comporem os tipos de *landforms*. As *landforms* são classificadas principalmente pelo *slope* e TPI (Figure 1). O *aspect* classifica as faces *quentes* ou *frias* do relevo e o *moisture index* classifica as áreas planas em secas ou úmidas.

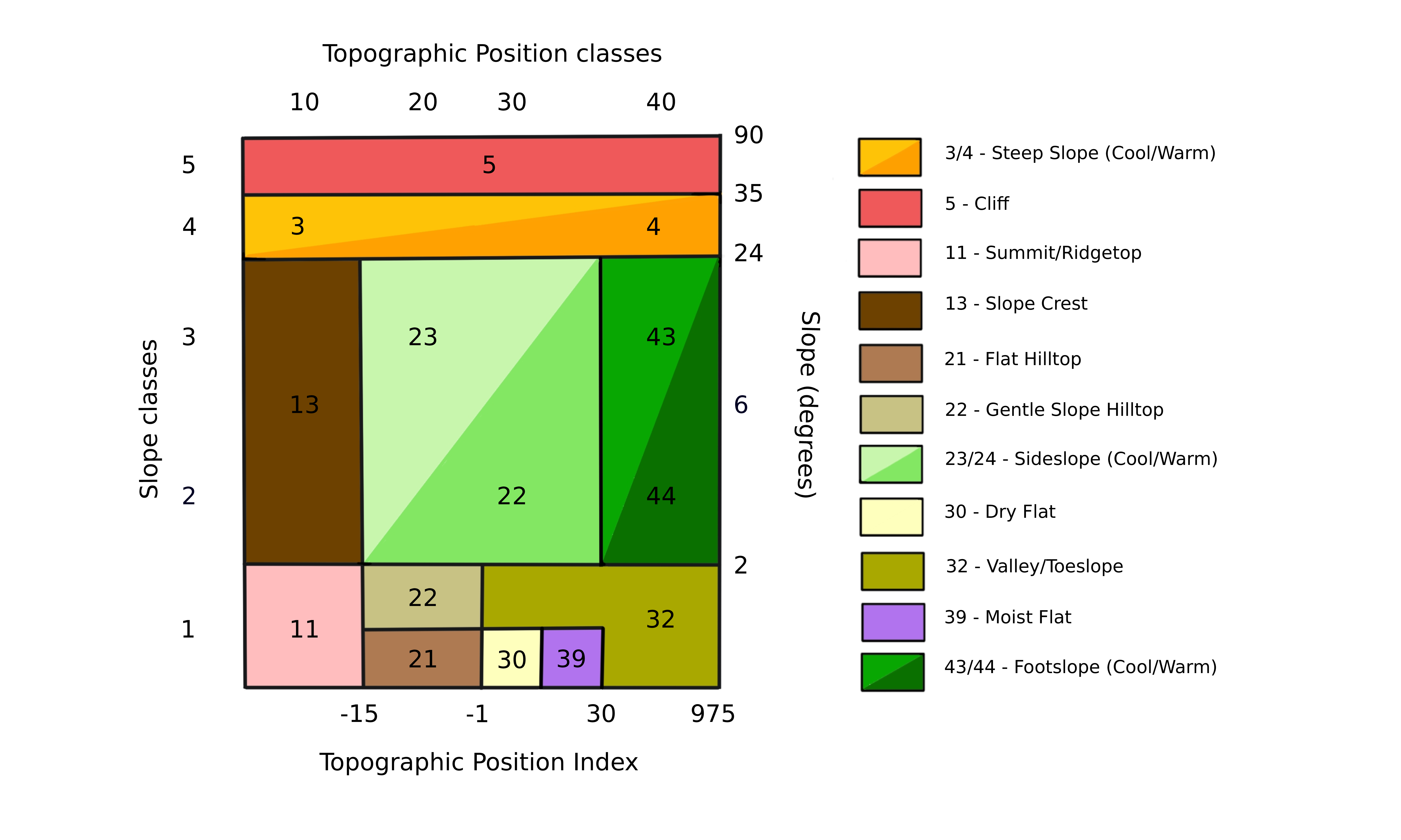


Figura 1. Classificação de landforms pela The Nature Conservancy para as paisagens na América do Norte.Adaptação de Anderson et al. (2016).

Cada variável foi discretizada em classes para comporem os tipos de *landforms*. Os limiares (Tabela 1) para a discretização foram definidos por ajustes visuais que melhor representavam as *landforms*.

Tabela 1. Descrição dos limiares de classificação de cada variável em classes.

| Variáveis | Classes | Limiar inferior | Limiar superior |
| --- | --- | --- | --- |
| Inclinação do relevo | 1 | -1 | 2 |
| Inclinação do relevo | 2 | 2 | 6 |
| Inclinação do relevo | 3 | 6 | 24 |
| Inclinação do relevo | 4 | 24 | 35 |
| Inclinação do relevo | 5 | 35 | 90 |
| TPI | 1 | -Inf | -15 |
| TPI | 2 | -15 | -1 |
| TPI | 3 | -1 | 30 |
| TPI | 4 | 30 | 975 |
| Aspecto | 2 | 0 | 90 |
| Aspecto | 1 | 90 | 270 |
| Aspecto | 2 | 270 | 360 |
| Índice de Umidade | 0 | -Inf | 30000 |
| Índice de Umidade | 1 | 3000 | Inf |

Em seguida, as classes foram combinadas pela soma de cada classe multiplicada por um peso. O *moisture index* foi multiplicado por 1000, *aspect* por 100, TPI por 10 e *slope* por 1. Desta forma, o número resultante representa um código descrevendo as classes de cada variável. Por exemplo, 1231 é a classe 1 de *moisture index*, 2 de *aspect*, 3 de TPI e 1 de *slope*. Posteriormente, os valores finais foram convertidos em tipos de *landforms*, seguindo a Tabela 2.

Tabela 2. Critério de conversão dos códigos da combinação de classes em tipos de landforms.

| Código | Tipos de *landforms* |
| --- | --- |
| 10 | 11 |
| 11 | 11 |
| 12 | 11 |
| 13 | 13 |
| 14 | 11 |
| 15 | 5 |
| 20 | 21 |
| 21 | 21 |
| 22 | 22 |
| 23 | 24 |
| 24 | 24 |
| 25 | 5 |
| 31 | 30 |
| 32 | 32 |
| 33 | 24 |
| 34 | 24 |
| 35 | 5 |
| 40 | 32 |
| 41 | 32 |
| 42 | 32 |
| 43 | 43 |
| 44 | 3 |
| 45 | 5 |
| 51 | 51 |
| 111 | 11 |
| 112 | 11 |
| 113 | 13 |
| 114 | 3 |
| 115 | 5 |
| 121 | 21 |
| 122 | 22 |
| 123 | 23 |
| 124 | 3 |
| 125 | 5 |
| 131 | 30 |
| 132 | 32 |
| 133 | 23 |
| 134 | 3 |
| 135 | 5 |
| 141 | 32 |
| 142 | 32 |
| 143 | 43 |
| 144 | 3 |
| 145 | 5 |
| 151 | 51 |
| 211 | 11 |
| 212 | 11 |
| 213 | 13 |
| 214 | 4 |
| 215 | 5 |
| 221 | 21 |
| 222 | 22 |
| 223 | 24 |
| 224 | 4 |
| 225 | 5 |
| 231 | 30 |
| 232 | 32 |
| 233 | 24 |
| 234 | 4 |
| 235 | 5 |
| 241 | 32 |
| 242 | 32 |
| 243 | 44 |
| 244 | 4 |
| 245 | 5 |
| 251 | 51 |
| 1000 | 39 |

## Bases de dados

Nós utilizamos para a classificação das *landforms* o Modelo Digital de Elevação (DEM) do Merit-DEM (Yamazaki *et al.* 2017), o acúmulo de fluxo do Merit-Hydro (Yamazaki *et al.* 2019) e a camada de uso do solo do MapBiomas (MapBiomas Project 2020). O Modelo Digital de Elevação possui uma resolução de 90 metros e foi escolhido por ser um produto global ao combinar dados dos satélites do *Shuttle Radar Topography Mission* (SRTM) (Farr *et al*. 2007) e *Advanced Land Observing Satellite* (ALOS) (Tadono *et al*. 2014), permitindo a replicabilidade da metodologia em outras regiões. O Merit-DEM corrige viéses de Modelo Digitais de Elevação gerados por imagens de satétite como *speckle noise*, *stripe noise*, *absolute bias* e *tree height bias* (Yamazaki *et al.* 2017). A correção de *tree height bias* é principalmente importante para a Floresta Amazônica devido à sua densidade de árvores altas. Além disso, há um produto derivado, o Merit-Hydro, que disponibiliza o acúmulo de fluxo global, que demandaria grande esforço computacional para ser calculado para todo o Brasil. O Merit-Hydro corrige os efeitos de densidade de árvores no cálculo da rede dendrítica, o que é importante para a Amazônia.

Nós incluímos as classes de água do MapBiomas para complementar a superfície gerada pelo acúmulo de fluxo na definição de áreas planas úmidas. O MapBiomas é um projeto nacional de mapeamento e classificação de mudanças do uso do solo dos últimos 30 anos a partir de dados de sensoriamento remoto.

## Códigos para a criação da variedade de *landforms*

Nossas análises foram rodadas no *Google Earth Engine* (Gorelick 2017), devido a demanda computacional do projeto, usando o pacote **geemap** (Wu 2020) em *Python* (Python Software Foundation 2023) como interface pela facilidade na documentação e reprodutividade das análises.

### Inclinação do relevo (*slope*)

Nós criamos a superfície de *slope* a partir do Merit-DEM.

# Importando Modelo Digital de Elevação  
  
DEM = ee.Image("MERIT/DEM/v1\_0\_3")   
  
# Calculando o slope  
  
slope = ee.Terrain.slope(DEM)

### Aspecto (*aspect*)

Nós calculamos o *aspect* do relevo utilizando o mesmo DEM.

aspect = ee.Terrain.aspect(DEM)

### Índice de Posição Topográfica (TPI)

Nós calculamos o Índice de Posição Topográfica (TPI)(Weiss 2001) para cada célula do *raster* dentro de um *kernel* circular com 7, 11 e 15 células de raio. O TPI é a diferença média de elevação entre a célula focal e um conjunto de células vizinhas.

a vizinhança *i* representa cada uma das *n* células dentro do *kernel* da célula focal. O índice final é composto pela média de TPI das três janelas, o que permite a consideração de diferentes níveis de resolução da paisagem, tanto local quanto regional (Theobald *et al*. 2015). Os tamanhos das janelas foram definidos visualmente para que melhor representassem as *landforms*, principalmente os *Summits*, *Valleys*, *Toeslopes* e *Hilltops* (*flat* e *gentle*). Os tamanhos das janelas também tinham que capturar os Platôs como *Summits*.

# Função para calcular o TPI  
  
def calculate\_TPI(pixel\_size):  
  
 # Calcule a média das células da vizinhança  
  
 focal\_mean = DEM.focalMean(\*\*{  
 'radius': pixel\_size,  
 'kernelType': "circle",  
 'units': "pixels"  
 })  
  
 # Calcule a diferença entre a ćelula focal e média da região   
  
 TPI = focal\_mean.subtract(DEM)  
  
 return TPI

# Tamanho das janelas   
  
window\_size = [7,11,15]  
  
# Calculo do TPI para cada janela e calculo do TPI médio das janelas  
  
TPI = ee.ImageCollection(list(map(calculate\_TPI, window\_size))).toBands().reduce("mean")

### Índice de Umidade (*Moisture index*)

Nós calculamos o *moisture index* (Anderson *et al*. 2016) baseado no acúmulo de fluxo presente no Merit-Hydro (Yamazaki *et al.* 2019), na camada **upg**, que é calculado sobre o Merit-DEM. O *moisture index* é calculado da seguinte forma:

onde fluxo é o acúmulo de fluxo e *slope* é o *slope* calculado anteriormente. O *moisture index* é a média do índice dentro de um *kernel* circular de uma célula de raio. O tamanho do raio foi escolhido visualmente para suavizar o índice, mas representando bem a distribuição dos cursos d’água.

# Importando o acúmulo de fluxo  
  
flow\_accumulation = ee.Image("MERIT/Hydro/v1\_0\_1").select("upg")  
  
# Calculando o moisture index  
  
moisture\_index = (  
 flow\_accumulation  
 .add(ee.Number(1))  
 .divide(slope.add(ee.Number(1)))  
 .log()  
 .multiply(1000)  
 .focalMean(\*\*{'radius': 1,  
 'kernelType': "circle",  
 'units': "pixels"  
 }  
 )  
)

### Convertendo variáveis em classes

### Inclinação do relevo (*slope*)

As variáveis foram convertidas em classes utilizando a Tabela 1, seguindo Anderson *et al*. (2016). As classes de slope foram criadas com o seguinte código:

slope\_classes = (  
 slope  
 .where(slope.gte(-1).And(slope.lte(2)), 1)  
 .where(slope.gt(2).And(slope.lte(6)), 2)  
 .where(slope.gt(6).And(slope.lte(24)), 3)  
 .where(slope.gt(24).And(slope.lte(35)), 4)  
 .where(slope.gt(35).And(slope.lte(90)), 5)  
)

### Aspecto (*aspect*)

O aspecto foi escolhido para definir as faces norte e sul do relevo no hemisfério sul.

aspect\_classes = (  
 aspect  
 .where(aspect.gte(0).And(aspect.lte(90)), 2)  
 .where(aspect.gt(90).And(aspect.lte(270)), 1)  
 .where(aspect.gt(270).And(aspect.lte(360)), 2)  
)

### Índice de Posição do Relevo (TPI)

As classes de TPI foram definidas para representarem bem os *Summits*, *Valleys*, *Toeslopes* e *Hilltops*, que foram as *landforms* mais difíceis de ajustar os parâmetros.

TPI\_classes = (  
 TPI  
 .where(TPI.lte(-15), 1)  
 .where(TPI.gt(-15).And(TPI.lt(-1)), 2)  
 .where(TPI.gte(-1).And(TPI.lte(30)), 3)  
 .where(TPI.gt(30).And(TPI.lte(975)), 4)  
)

### Índice de Umidade (*moisture index*)

O limiar do índice umidade para classificar as áreas como umidas ou secas foram definidos visualmente para capturarem a distribuição dos cursos d’água sem criar áreas planas secas com excesso de ramificações dendríticas. Grandes rios (ex. Rio Amazonas, represas e lagos) não foram bem representados pelo *moisture index*, pois classificava somente a partes mais profundas como áreas úmidas, mantendo o restante dos grandes corpos d’água como regiões planas secas. Nós corrigimos essa classificação combinando a área úmidade classificada pelo acúmulo de fluxo com a camada de águas produzida pelo MapBiomas.

# Classificando o índice de umidade em classes  
  
moisture\_classes = (  
 moisture\_index.where(moisture\_index.lte(3000), 0)  
 .where(moisture\_index.gt(3000), 1)  
)  
  
# Importando o dado de uso de solo do Mapbiomas e reprojetando para a escala do DEM  
  
mapbiomas = (  
 ee.Image("projects/mapbiomas-workspace/public/collection7/mapbiomas\_collection70\_integration\_v2")  
 .select("classification\_2020")  
 .reproject('EPSG:4326', None, 92.76624)  
)  
  
# Reclassificando o raster do MapBiomas em água (1) e outras classes (0)  
  
water = (  
 mapbiomas  
 .where(mapbiomas.eq(33), 1)  
 .where(mapbiomas.neq(33), 0)  
)  
  
# Combinado o índice de umidade com a camada de água e reclassificando  
  
moisture\_classes = moisture\_classes.add(water)  
  
moisture\_classes = (  
 moisture\_classes  
 .where(moisture\_classes.gte(1), 1)  
 .where(moisture\_classes.lt(1), 0)  
)

### Combinando as classes

Combinamos as classes para a geração de um código representativo de cada variável. O *moisture index* foi multiplicado por 1000, *aspect* por 100, TPI por 10 e *slope* por 1.

classes\_collection = ee.Image([moisture\_classes.multiply(ee.Number(1000)),  
 aspect\_classes.multiply(ee.Number(100)),  
 TPI\_classes.multiply(ee.Number(10)),  
 slope\_classes])  
  
landform\_combination = classes\_collection.reduce(ee.Reducer.sum())

### Classificando os tipos de *landforms*

Classificamos os tipos de *landforms* pelo código gerado anteriormente e ajustamos visualmente alguns códigos para representarem bem as *landforms*. Por exemplo, o código 11 representa áreas de baixa inclinação do relevo e uma posição do relevo mais alta que o entorno, sendo portanto um topo de montanha (*Summit*). No entanto, alguns códigos tiveram que ser bem inspecionados para separar alguns tipos de *landforms* como *Sideslopes* de *Valleys* e *Toeslopes*.

landform\_types = (  
 landform\_combination  
 .mask(landform\_combination.gt(0))  
 .where(landform\_combination.eq(10), 11)  
 .where(landform\_combination.eq(11), 11)  
 .where(landform\_combination.eq(12), 11)  
 .where(landform\_combination.eq(13), 13)  
 .where(landform\_combination.eq(14), 11)  
 .where(landform\_combination.eq(15), 5)  
 .where(landform\_combination.eq(20), 21)  
 .where(landform\_combination.eq(21), 21)  
 .where(landform\_combination.eq(22), 22)  
 .where(landform\_combination.eq(23), 24)  
 .where(landform\_combination.eq(24), 24)  
 .where(landform\_combination.eq(25), 5)  
 .where(landform\_combination.eq(31), 30)  
 .where(landform\_combination.eq(32), 32)  
 .where(landform\_combination.eq(33), 24)  
 .where(landform\_combination.eq(34), 24)  
 .where(landform\_combination.eq(35), 5)  
 .where(landform\_combination.eq(40), 32)  
 .where(landform\_combination.eq(41), 32)  
 .where(landform\_combination.eq(42), 32)  
 .where(landform\_combination.eq(43), 43)  
 .where(landform\_combination.eq(44), 3)  
 .where(landform\_combination.eq(45), 5)  
 .where(landform\_combination.eq(51), 51)  
 .where(landform\_combination.eq(111), 11)  
 .where(landform\_combination.eq(112), 11)  
 .where(landform\_combination.eq(113), 13)  
 .where(landform\_combination.eq(114), 3)  
 .where(landform\_combination.eq(115), 5)  
 .where(landform\_combination.eq(121), 21)  
 .where(landform\_combination.eq(122), 22)  
 .where(landform\_combination.eq(123), 23)  
 .where(landform\_combination.eq(124), 3)  
 .where(landform\_combination.eq(125), 5)  
 .where(landform\_combination.eq(131), 30)  
 .where(landform\_combination.eq(132), 32)  
 .where(landform\_combination.eq(133), 23)  
 .where(landform\_combination.eq(134), 3)  
 .where(landform\_combination.eq(135), 5)  
 .where(landform\_combination.eq(141), 32)  
 .where(landform\_combination.eq(142), 32)  
 .where(landform\_combination.eq(143), 43)  
 .where(landform\_combination.eq(144), 3)  
 .where(landform\_combination.eq(145), 5)  
 .where(landform\_combination.eq(151), 51)  
 .where(landform\_combination.eq(211), 11)  
 .where(landform\_combination.eq(212), 11)  
 .where(landform\_combination.eq(213), 13)  
 .where(landform\_combination.eq(214), 4)  
 .where(landform\_combination.eq(215), 5)  
 .where(landform\_combination.eq(221), 21)  
 .where(landform\_combination.eq(222), 22)  
 .where(landform\_combination.eq(223), 24)  
 .where(landform\_combination.eq(224), 4)  
 .where(landform\_combination.eq(225), 5)  
 .where(landform\_combination.eq(231), 30)  
 .where(landform\_combination.eq(232), 32)  
 .where(landform\_combination.eq(233), 24)  
 .where(landform\_combination.eq(234), 4)  
 .where(landform\_combination.eq(235), 5)  
 .where(landform\_combination.eq(241), 32)  
 .where(landform\_combination.eq(242), 32)  
 .where(landform\_combination.eq(243), 44)  
 .where(landform\_combination.eq(244), 4)  
 .where(landform\_combination.eq(245), 5)  
 .where(landform\_combination.eq(251), 51)  
 .where(landform\_combination.gte(1000), 39)  
)

### Exportando mapas para assets

# Nome do asset  
assetId = "projects/ee-lucasljardim9/assets/landform\_types"  
  
# Importando mapa de biomas do IBGE para extrair as coordenadas mínimas e máximas do Brasil  
bioma = ee.FeatureCollection("projects/ee-lucasljardim9/assets/Biome")  
  
def func\_cmp(feature):  
 return feature.bounds()   
  
# Extraindo as coordenadas mínimas e máximas do Brasil  
bioma\_box = bioma.map(func\_cmp).geometry().dissolve(\*\*{'maxError': 1}).bounds()  
  
# Extraindo a resolução do mapa  
escala = landform\_types.projection().nominalScale()

# Exportando para o gee  
geemap.ee\_export\_image\_to\_asset(  
 landform\_types, description='landform\_types', assetId=assetId, region=bioma\_box, scale=escala,maxPixels=1e13  
)

### Exemplo de *landforms*

Abaixo está uma representação das *landforms* na região de Alto Paraíso de Goiás-GO (Latitude:-14.11, Longitude:-47.26).

%%capture --no-display  
  
# Delimitando a região  
regiao = ee.Geometry.BBox(-47.4631, -13.9777, -47.1005, -14.1711)  
  
# Creando a pasta para exportar as figuras  
if not os.path.exists("figura"):  
   
 os.mkdir("figura")  
  
# Exportando a imagem da região  
geemap.ee\_export\_image(  
 landform\_types, filename="figura/landform\_types.tif", scale=escala, region=regiao, file\_per\_band=False  
)

# Paleta de cores das landforms  
palette = [  
 "#ffc408", # 3  
 "#ffa101", # 4  
 "#ef595a", # 5  
 "#ffbdbe", # 11  
 "#6e4100", # 13  
 "#af7b53", # 21  
 "#c8f6ad", # 23  
 "#c8c284", # 22  
 "#83e763", # 24  
 "#08a702", # 43  
 "#ffffbe", # 30  
 "#a9a800", # 32  
 "#b671f2", # 39  
 "#0a7000" ] # 44  
  
# Discretinzando a paleta de cores  
cmap = ListedColormap(  
 palette, 'Custom cmap')  
  
class\_bins = [3, 4, 5, 11, 13, 21, 22, 23, 24, 30, 32, 39, 43, 45]  
  
norm = BoundaryNorm(class\_bins,   
 13)  
  
# Plotando mapa   
geemap.plot\_raster("figura/landform\_types.tif", cmap = cmap, norm = norm, figsize = [20, 10])

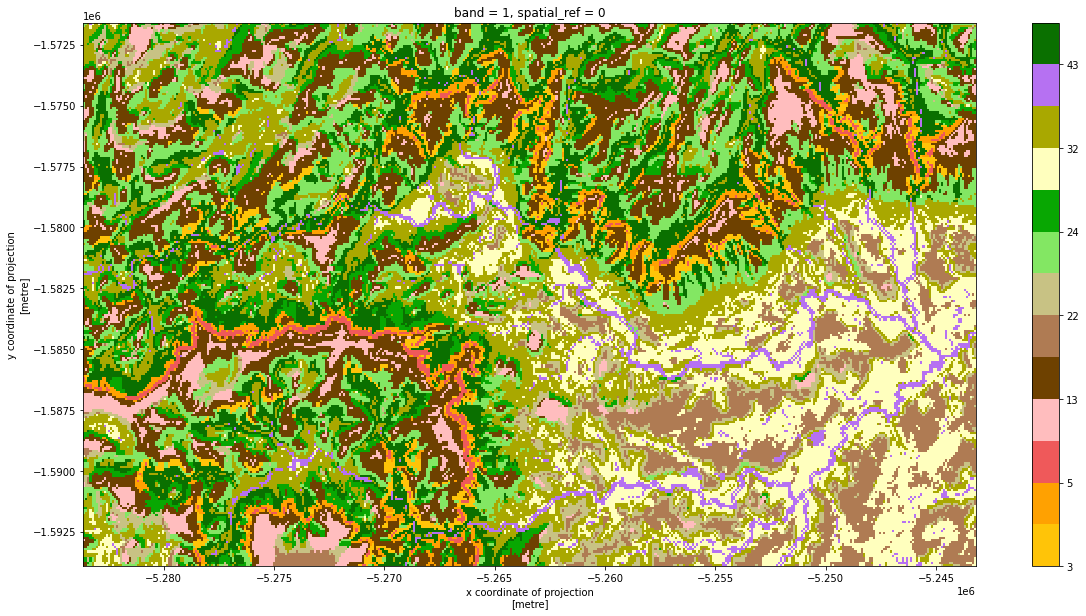


Figura 2. Landforms classificadas na região de Alto Paraíso de Goiás-GO, Brasil.

## Calculando a variedade de *landforms*

A variedade de *landforms* foi calculada como a soma de tipos diferentes de *landforms* dentro de um *kernel* circular de uma célula focal. O tamanho do raio do *kernel* foi definido calculando a variedade em diferentes raios (2, 5, 7, 10, 15, 20 células) e calculando o ganho de variedade a cada aumento de raio. O raio escolhido foi aquele que o subsequente não adicionou variedade. Desta forma, o raio representa o nível de resolução da paisagem que captura o máximo de variedade de *landforms*. Raios maiores podem aumentar a variedade, mas devido a mudança de paisagem. Assim, o raio escolhido foi de 5 células de raio (450 metros) para todo o Brasil. Abaixo está uma representação da variedade de *landforms* para a mesma região de Alto Paraíso de Goiás-GO.

radius\_pixels = 5  
  
landform\_variety = (  
 landform\_types  
 .neighborhoodToBands(ee.Kernel.circle(radius\_pixels))  
 .reduce(ee.Reducer.countDistinct())  
)

%%capture --no-display  
  
geemap.ee\_export\_image(  
 landform\_variety, filename="figura/landform\_variety.tif", scale=escala, region=regiao, file\_per\_band=False  
)

geemap.plot\_raster("figura/landform\_variety.tif", figsize = [20, 10])

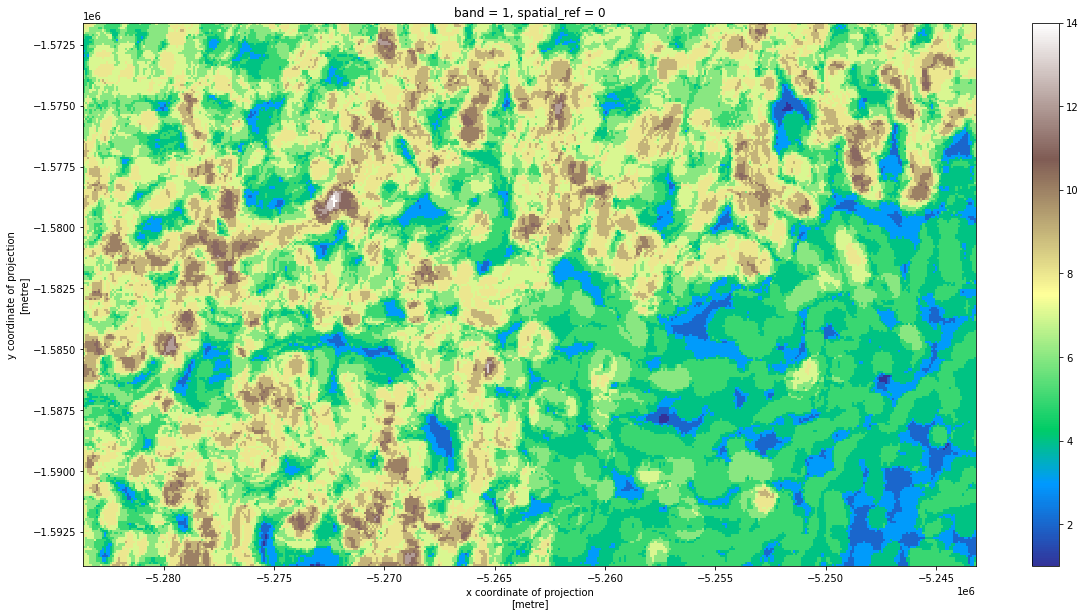


Figura 3. Variedade de landforms para região de Alto Paraíso de Goiás-GO, Brasil.

assetId = "projects/ee-lucasljardim9/assets/landform\_variety"  
  
geemap.ee\_export\_image\_to\_asset(  
 landform\_variety,   
 description='landform\_variety',   
 assetId=assetId,   
 region=bioma\_box,   
 scale=escala,maxPixels=1e13  
)

## Bibliografia

Anderson, M.G., Barnett, A., Clark, M., Ferree, C., Sheldon, A.O., Prince, J. 2016. Resilient Sites for Terrestrial Conservation in Eastern North America. The Nature Conservancy. http://easterndivision.s3.amazonaws.com/Resilient\_Sites\_for\_Terrestrial\_Conservation.pdf.

Farr, T.G., et al. 2007. The shuttle radar topography mission. Reviews of Geophysics, 45, 2, RG2004, https://doi.org/10.1029/2005RG000183.

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R. 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote Sensing of Environment, 202, 18-27, https://doi.org/10.1016/j.rse.2017.06.031.

MapBiomas Project. 2020. Collection 7 of the Annual Series of Land Use and Land Cover Maps of Brazil. Accessed on 2023 through the link: projects/mapbiomas-workspace/public/collection7/mapbiomas\_collection70\_integration\_v2.

Python Software Foundation. Python Language Reference. http://www.python.org.

Tadono, T., Ishida, H., Oda, F., Naito, S., Minakawa, K., Iwamoto, H. 2014. Precise Global DEM Generation by ALOS PRISM. Precise Global DEM Generation by ALOS PRISM. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, II-4, 71–76, https://doi.org/10.5194/isprsannals-ii-4-71-2014.

Theobald, D.M., Harrison-Atlas, D., Monahan, W.B., Albano, C.M. 2015. Ecologically-relevant maps of landforms and physiographic diversity for climate adaptation planning. Plos One, 12, 1-17, https://doi.org/10.1371/journal.pone.0143619.

Yamazaki, D., Ikeshima, D., Tawatari, R., Yamaguchi, T, O’Loughlin, F., Neal, J.C., Sampson, C.C., Kanae, S., Bates, P.D. 2017. A high-accuracy map of global terrain elevations. Geophysical Research Letters, 11, 5844-5853, doi:10.1002/2017GL072874.

Yamazaki, D., Ikeshima, D., Sosa, J., Bates, Paul, D., Allen, G.H., Pavelsky, T.M. 2019. MERIT Hydro: A High-Resolution Global Hydrography Map Based on Latest Topography Dataset. Water Resources Research, 6, 5053-5073, https://doi.org/10.1029/2019WR024873.

Iwahashi, J., Yamazaki, D. 2022. Global polygons for terrain classification divided into uniform slopes and basins. Progress in Earth and Planetary Science, 9, 33, https://doi.org/10.1186/s40645-022-00487-2.

Weiss, A.D., 2001. Topographic position and landforms analysis. Poster Presentation, ESRI Users Conference, San Diego, CA.

Wu, Q. 2020. geemap: A Python package for interactive mapping with Google. Journal of Open Source Software, 5, 51, 2305, https://doi.org/10.21105/joss.02305.