# Calculando *Z-scores* das variáveis

```
import os
import ee
import geemap
```

```
geemap.ee_initialize()
```

Antes de calcularmos a diversidade da paisagem e a resiliência, nós transformamos as variáveis, variedade de *landforms*, amplitude de elevação, densidade e quantidade de áreas úmidas e diversidade de solo, em valores de Z. O cálculo de Z é realizado dentro de regiões similares ecologicamente e geologicamente (unidade eco-geológicas), criadas anteriormente. Além disso, os valores de densidade e quantidade de áreas úmidas são combinados em *wetland score*.

Os valores de Z de cada variável, em cada célula (i), é o desvio do valor da célula da média da unidade eco-geológica (u), divido pelo seu desvio padrão:

$$Z - score_{iu} = \frac{i - media_u}{desviopadrao_u}$$

O *wetland score* é a média ponderada das densidades locais e regionais de áreas úmidas. Em locais onde o valor de Z da densidade média é menor que o Z da quantidade de áreas úmidas, o *wetland score* assume o segundo valor Z.

$$wetland.score = \begin{cases} \frac{2 \times densidade_{local}(Z) + densidade_{regional}(Z)}{3} & \text{se densidade média é maior ou igual à quantidade regional} \\ \frac{2 \times densidade_{local}(Z) + densidade_{regional}(Z) + quantidade_{regional}(Z)}{4} & \text{se a quantidade regional for maior a densidade média} \end{cases}$$

## Base de Dados

Nós utilizamos as variáveis criadas anteriormente e guardadas como *assets* no *Google Earth Engine*. As variáveis são:

- variedade de *landforms*

- amplitude de elevação
- diversidade de solo
- unidades eco-geológicas
- conectividade da paisagem
- densidade local de áreas úmidas
- densidade regional de áreas úmidas
- quantidade regional de áreas úmidas

## Códigos para o cálculo dos *Z-scores*

As 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.

Primeiro, nós criamos uma função para calcular o *Z-score* dentro de cada unidade eco-geológica (**code**) para cada variável (**image**). A função cria uma máscara da variável pelo código da unidade eco-geológica e calcula as médias e desvio padrão de cada unidade, depois as variáveis originais são subtraídas da média de sua unidade e divida pelo desvio padrão.

```
def calculate_Z(code, image):
  #Crie uma máscara de unidade eco-geologica para cada codigo da unidade (code)
  mask = (geophysical_setting
          .where(geophysical_setting.eq(ee.Number.parse(code)), 1)
          .where(geophysical_setting.neq(ee.Number.parse(code)), 0))

  # Corte a imagem pela máscara
  map1 = image.mask(mask).rename("b1")
  # Calcule a média de cada unidade
  mean = (map1.reduceRegion(**{'reducer':ee.Reducer.mean(),
                               'geometry':bioma_box,
                               'scale':escala,
                               'maxPixels':1e13
                              })
              .get("b1"))
  # Calcule o desvio padrão de cada unidade
  sd = (map1.reduceRegion(**{'reducer':ee.Reducer.stdDev(),
                             'geometry':bioma_box,
                             'scale':escala,
                             'maxPixels':1e13
                            })
            .get("b1"))
```

```
  # calcule o Z
  Z = map1.subtract(ee.Number(mean)).divide(ee.Number(sd))

  return Z
```

Em seguida, criamos também um função para aplicar o **calculate_Z** para uma imagem definida pelo usuário transforme em uma image de banda única.

```
def wrap_calculate_Z(image):
    # Aplique calculate_Z para cada unidade (classe), guarde numa ImageCollection,
    # transform em bandas e resuma numa imagem de banda única
    Z_map = (ee.ImageCollection(classes.map(lambda i: calculate_Z(i, image)))
             .toBands()
             .reduce("sum"))

    return Z_map
```

Aqui nós importamos as variáveis de interesse para o cálculo de Z.

```
# Importando raster das unidades eco-geologicas

geophysical_setting = (ee.Image("projects/ee-lucasljardim9/assets/ecoregions_geology")
                       .unmask())

# Importando as variáveis

landform_variety = ee.Image("projects/ee-lucasljardim9/assets/landform_variety")

elevation_range = ee.Image("projects/ee-lucasljardim9/assets/elevation_range_residual")

soil_diversity = ee.Image('projects/ee-resilience/assets/New_window_size/soil_diversity')

connectedness = ee.Image("projects/ee-lucasljardim9/assets/Biomas_resistencia_kernel")

# Importando os dados de wetlands para o wetland score

wetlands_count = ee.Image("projects/ee-lucasljardim9/assets/wetlands_count")

wetlands_density = ee.Image("projects/ee-lucasljardim9/assets/wetlands_density")

wetlands_density_1000 = ee.Image("projects/ee-lucasljardim9/assets/wetlands_density_1000")
```

```
# Importando o polígono de bioma para definir as
# coordenadas máximas e mínimas do Brasil

bioma = ee.FeatureCollection("projects/ee-lucasljardim9/assets/Biome")

# ModeloDigital de Elevação para extrair a resolução
DEM = ee.Image("MERIT/DEM/v1_0_3")

# função para extrair as bordas dos polígonos
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()

# Resolução das análises
escala = DEM.projection().nominalScale()
```

Nós criamos uma lista com os nomes das unidades eco-geológicas (classes).

```
# extraindo os valores do raster de unidades eco-geológicas em histograma
histogram = geophysical_setting.reduceRegion(**{'reducer':ee.Reducer.frequencyHistogram(),
                                                'geometry': bioma_box,
                                                'scale': escala,
                                                'maxPixels':1e13
                                            })
# Criando uma lista com os nomes das unidades eco-geológicas
classes = ee.Dictionary(histogram.get("b1")).keys()
```

## Calculando Z-scores

Nós aplicamos a função **wrap_calculate_Z** para cada variável e guardamos o valores de Z calculados.

```
# Calculando os valores de Z para cada variável
Z_landform_variety = wrap_calculate_Z(landform_variety)

Z_elevation_range = wrap_calculate_Z(elevation_range)

Z_soil_diversity = wrap_calculate_Z(soil_diversity)
```

```
Z_wetlands_count = wrap_calculate_Z(wetlands_count)

Z_wetlands_density = wrap_calculate_Z(wetlands_density)

Z_wetlands_density_1000 = wrap_calculate_Z(wetlands_density_1000)

Z_connectedness = wrap_calculate_Z(connectedness).multiply(-1)
```

### Calculando wetland score

Aqui, nós calculamos o *wetland score*, aplicando a fórmula descrita anteriormente.

```
# Calculando a densidade de areas úmidas como a média do local e regional
wetlands_density = (Z_wetlands_density
                    .multiply(2)
                    .add(Z_wetlands_density_1000)
                    .divide(3))

# Testando se o Z da quantidade de áreas úmidas é maior que a densidade média

wet_test = wetlands_density.lt(Z_wetlands_count)

# Média de densidade local, regional e quantidade de áreas úmidas
wet_average = (wetlands_density
               .multiply(3)
               .add(Z_wetlands_count)
               .divide(4))

# Substituindo os locais com densidade menor que a quantidade pelos valores de quantidade
Z_wetlands_score = Z_wetlands_density.where(wet_test, wet_average)
```

### Exportando os Z-scores

Por último, exportamos todas as imagens de Z como *asset* no *Google Earth Engine*.

```
# Criando os links dos assets
landform_assetId = "projects/ee-lucasljardim9/assets/Z_landform_variety_byregion"

elevation_assetId = "projects/ee-lucasljardim9/assets/Z_elevation_range_region"
```

```
wetland_assetId = "projects/ee-lucasljardim9/assets/Z_wetlands_score_byregion"

soil_assetId = "projects/ee-lucasljardim9/assets/Z_soil_diversity_byregion"

connectedness_assetId = "projects/ee-lucasljardim9/assets/Z_connectedness_byregion"

# Exportando as imagens
geemap.ee_export_image_to_asset(
    Z_landform_variety,
    description='Z_landform_variety',
    assetId=landform_assetId,
    region=bioma_box,
    scale=escala,
    maxPixels=1e13
)

geemap.ee_export_image_to_asset(
    Z_elevation_range,
    description='Z_elevation_range',
    assetId=elevation_assetId,
    region=bioma_box,
    scale=escala,
    maxPixels=1e13
)

geemap.ee_export_image_to_asset(
    Z_wetlands_score,
    description='Z_wetlands_score',
    assetId=wetland_assetId,
    region=bioma_box,
    scale=escala,
    maxPixels=1e13
)

geemap.ee_export_image_to_asset(
    Z_soil_diversity,
    description='Z_soil_diversity',
    assetId=soil_assetId,
    region=bioma_box,
    scale=escala,
    maxPixels=1e13
```

```
)

geemap.ee_export_image_to_asset(
    Z_connectedness,
    description='Z_connectedness',
    assetId=connectedness_assetId,
    region=bioma_box,
    scale=escala,
    maxPixels=1e13
)
```