OPERAÇÕES DE ÁLGEBRA DE MAPAS NO CONTEXTO DA BIBLIOTECA GEOTRELLIS

CAP 349 – 3 – Banco de Dados Geográficos

JULIANA MARINO BALERA

Agenda...

- Motivação
- GeoTrellis
- Metodologia
- Estudos de Caso
- Resultados

Motivação...

- Existem uma série de ferramentas para a realização de operações de álgebra de mapas
  - Spring
  - Qgis
  - ArqGis
- Diferentes ferramentas → diferentes funcionalidades
  - Lidar com grande quantidade de dados

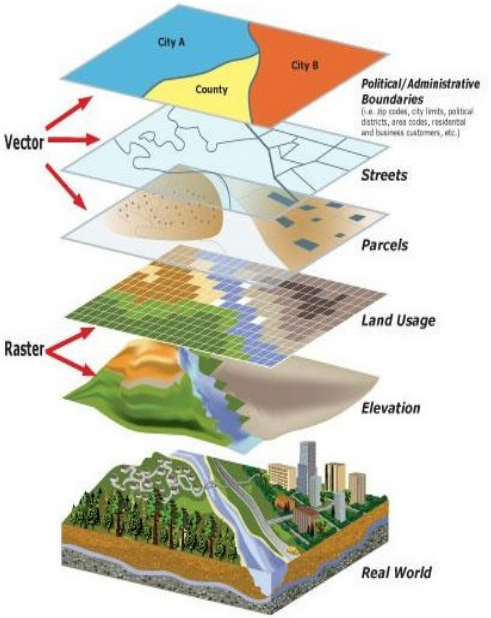
Motivação...

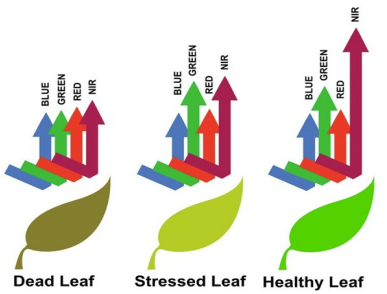
- Necessidade de processar um conjunto muito grande de dados
- Processamento de forma distribuída
- GeoTrellis → possibilita a manipulação de dados GEO em grande escala

Objetivo...

- Explorar o uso da biblioteca GeoTrellis aplicada a dois estudos de caso:
  - Cálculo do risco de fogo (com base no script do TerraMa2)
  - Determinação de índices de auxílio a identificação de anomalias de vegetação

GeoTrellis





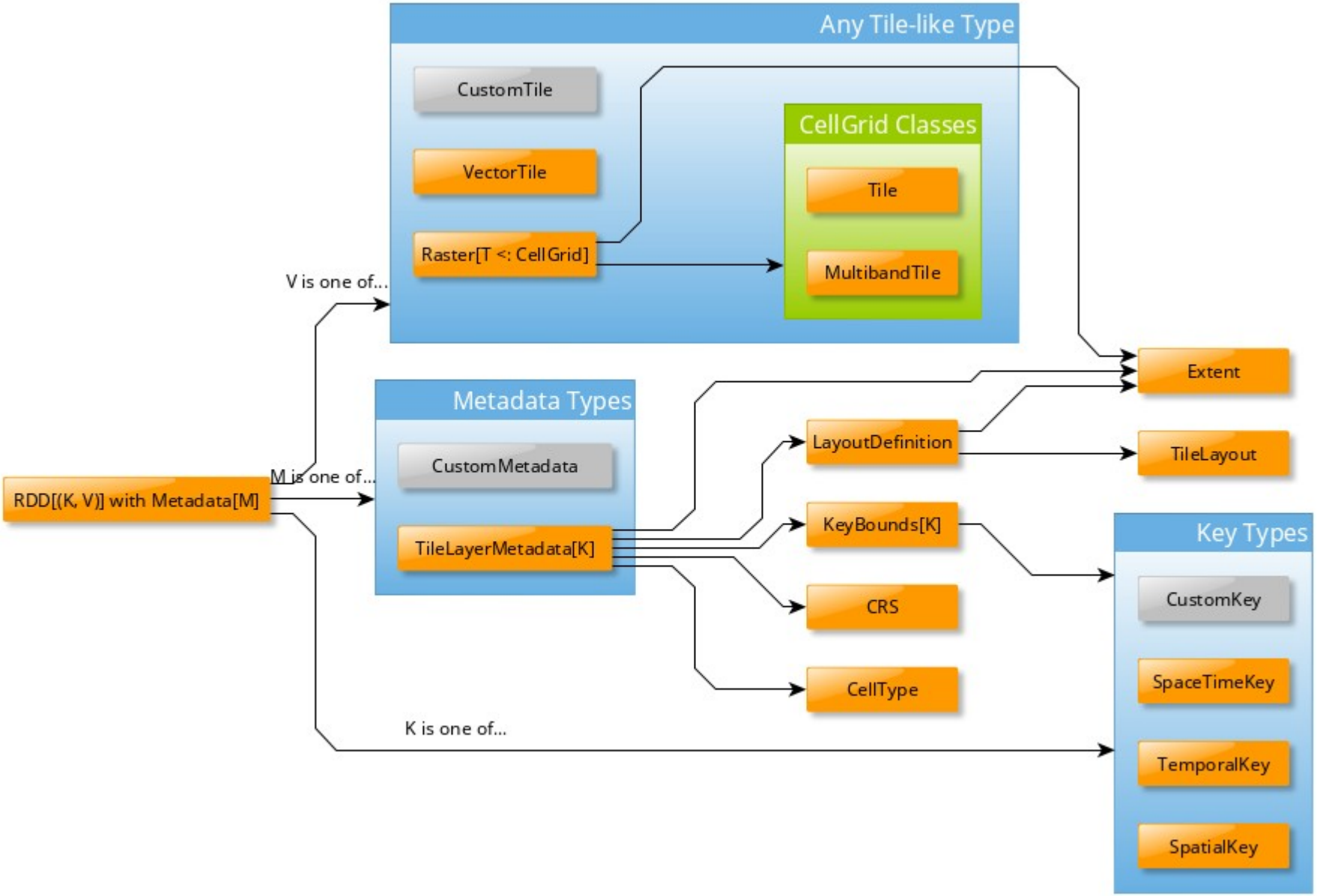
GeoTrellis

- Biblioteca que possibilita a realização de operações de álgebra de mapas de forma distribuída

- Utiliza o ApacheSpark para tornar possível a execução de operações espaciais em sistemas distribuídos.
  - Resilient Distributed Datasets

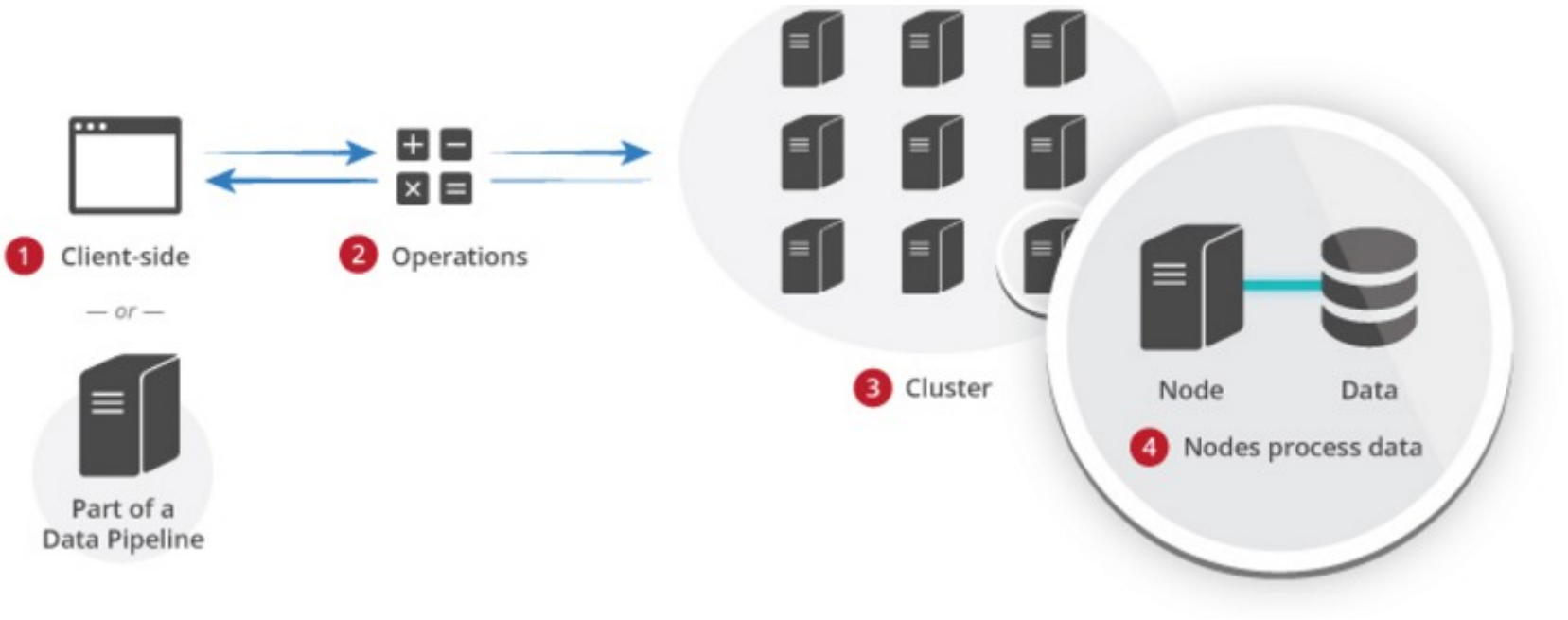
- Linguagem Scala
  - Java Virtual Machine
  - Multiparadigma

Resilient Distributed Datasets (RDD)



GeoTrellis

GeoTrellis

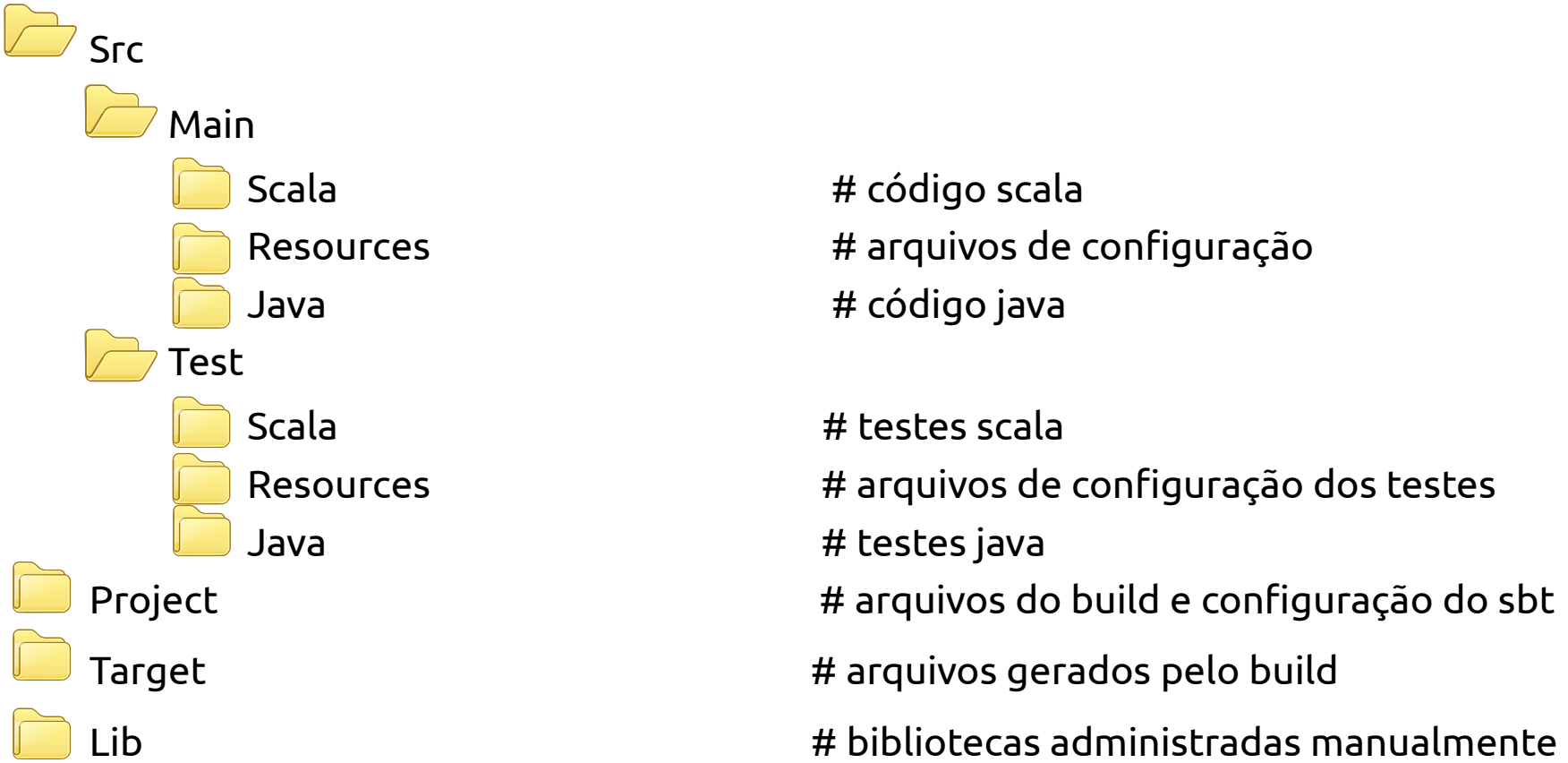


GeoTrellis - Simple Build Tool (SBT)

- Ferramenta para a automação da compilação utilizada em projetos *Scala*
- Configurações
- Gerenciamento de dependências
  - Acesso a dependências por meio de repositórios Maven

GeoTrellis - Simple Build Tool (SBT)

- Estrutura típica de diretórios



```
name := "geotrellis-sbt-template"

version := "0.1.0"

scalaVersion := "2.11.8"

organization := "com.azavea"

licenses := Seq("Apache-2.0" -> url("http://www.apache.org/licenses/LICENSE-2.0.html"))

scalacOptions ++= Seq(
  "-deprecation",
  "-unchecked",
  "-Yinline-warnings",
  "-language:implicitConversions",
  "-language:reflectiveCalls",
  "-language:higherKinds",
  "-language:postfixOps",
  "-language:existentials",
  "-feature")

publishMavenStyle := true
publishArtifact in Test := false
pomIncludeRepository := { _ => false }

resolvers += Resolver.bintrayRepo("azavea", "geotrellis")

libraryDependencies ++= Seq(
  "com.azavea.geotrellis" %% "geotrellis-spark" % "1.0.0-d9f051d",
  "org.apache.spark"      %% "spark-core"       % "2.0.1" % "provided",
  "org.scalatest"         %%  "scalatest"       % "2.2.0" % "test"
)

assemblyMergeStrategy in assembly := {
  case "reference.conf" => MergeStrategy.concat
  case "application.conf" => MergeStrategy.concat
  case "META-INF/MANIFEST.MF" => MergeStrategy.discard
  case "META-INF\\MANIFEST.MF" => MergeStrategy.discard
  case "META-INF/ECLIPSEF.RSA" => MergeStrategy.discard
  case "META-INF/ECLIPSEF.SF" => MergeStrategy.discard
  case _ => MergeStrategy.first
}
```

Álgebra de Mapas

Algumas operações

```
val Tile: tile3 = tile1.tile + tile2.tile

val Tile: tile2 = tile1.tile.mapDouble{(x:Double) => funcao(x)}

val Tile: tile2 = tile1.tile.foreach{(x: Double) => math.pow(x,2)}

val Tile: tile2 = tile1.tile.localSin()

val Tile: tile3 = tile1.combine(tile2) { (z1, z2) => z1 * z2 }

val nd = NODATA
val input = Array[Int](
        nd, 7, 1, 1, 3, 5, 9, 8, 2,
        9, 1, 1, 2, 2, 2, 4, 3, 5,
        3, 8, 1, 3, 3, 3, 1, 2, 2,
        2, 4, 7, 1, nd, 1, 8, 4, 3)
val iat = IntArrayTile(input, 9, 4)
val focalNeighborhood = Square(1)
val meanTile = iat.focalMean(focalNeighborhood)

val hexColored: IntArrayTile = IntArrayTile.fill(0xFF0000FF, 100, 100)

val (zoom, reprojected): (Int, RDD[(SpatialKey, MultibandTile)] with Metadata[TileLayerMetadata[SpatialKey]]) =
        MultibandTileLayerRDD(tiled, rasterMetaData)
        .reproject(WebMercator, layoutScheme, Bilinear)

Polygon((10.0, 10.0), (10.0, 20.0), (30.0, 30.0), (10.0, 10.0)).toGeoJson
```

RDD

#ABCDEFGHIJKLMNOPQRSTUVWXYZ – deprecated

display packages only

geotrellis.doc.examples hide focus

- LandsatMultibandRDDExample

geotrellis.spark hide focus

- CellGridLayoutRDDMethods
- ContextRDD
- ContextRDDMethods
- withRDDCostDistanceMethods
- withLayerRDDCropMethods
- withRDDDoubleKernelDensityMethods
- withRDDIntKernelDensityMethods
- withEuclideanDistanceMultiPointRDDMethods
- withEuclideanDistancePointRDDMethods
- withEuclideanDistanceRDDMethods
- withRDDMultibandEqualizationMethods
- withRDDSinglebandEqualizationMethods
- withTileLayerRDDFilterMethods
- withElevationTileLayerRDDMethods
- withFocalTileRDDMethods
- withLocalTileRDDMethods
- withLocalTileRDDSeqMethods
- withLocalTemporalTileRDDMethods
- withZonalTileRDDMethods
- withTileRDDMaskMethods
- withRDDMultibandMatchingMethods
- withRDDSinglebandMatchingMethods
- withRDDLayoutMergeMethods
- withTileRDDMergeMethods
- MultibandTileLayerRDD
- MultibandTileLayerRDDMethods
- withCellValueFeatureRDDRasterizeMethods
- withFeatureRDDRasterizeMethods
- withGeometryRDDRasterizeMethods
- withTileRDDReprojectMethods
- withRDDMultibandSigmoidalMethods
- withRDDSinglebandSigmoidalMethods
- withProjectedExtentRDDSplitMethods
- withSpatialTileLayoutRDDMethods
- withSpatialTileRDDMethods
- withStatsMultibandTileRDDMethods
- withStatsTileRDDMethods
- withZonalSummaryFeatureRDDMethods
- withZonalSummaryKeyedFeatureRDDMethods
- withZonalSummaryTileLayerRDDMethods
- TileLayerRDD
- TileLayerRDDMethods
- withRDDTimeSeriesMethods
- withRDDViewshedMethods
- withCellGridLayoutRDDMethods
- withContextRDDMethods

geotrellis.spark

ContextRDD

Related Docs: object ContextRDD | package spark

```
class ContextRDD[K, V, M] extends RDD[(K, V)] with Metadata[M]
```

▸ Linear Supertypes

Ordering: Alphabetic | By Inheritance

Inherited: ContextRDD | Metadata | RDD | Logging | Serializable | Serializable | AnyRef | Any

Hide All | Show All

Visibility: Public | All

Instance Constructors

```
new ContextRDD(rdd: RDD[(K, V)], metadata: M)
```

Value Members

```
def ++(other: RDD[(K, V)]): RDD[(K, V)]
def aggregate[U](zeroValue: U)(seqOp: (U, (K, V)) ⇒ U, combOp: (U, U) ⇒ U)(implicit arg0: ClassTag[U]): U
def cache(): ContextRDD.this.type
def cartesian[U](other: RDD[U])(implicit arg0: ClassTag[U]): RDD[((K, V), U)]
def checkpoint(): Unit
def coalesce(numPartitions: Int, shuffle: Boolean, partitionCoalescer: Option[PartitionCoalescer])(implicit ord: Ordering[(K, V)]): RDD[(K, V)]
def collect[U](f: PartialFunction[(K, V), U])(implicit arg0: ClassTag[U]): RDD[U]
def collect(): Array[(K, V)]
def compute(split: Partition, context: TaskContext): Iterator[(K, V)]
def context: SparkContext
def count(): Long
def countApprox(timeout: Long, confidence: Double): PartialResult[BoundedDouble]
def countApproxDistinct(relativeSD: Double): Long
def countApproxDistinct(p: Int, sp: Int): Long
def countByValue()(implicit ord: Ordering[(K, V)]): Map[(K, V), Long]
def countByValueApprox(timeout: Long, confidence: Double)(implicit ord: Ordering[(K, V)]): PartialResult[Map[(K, V), BoundedDouble]]
final def dependencies: Seq[Dependency[_]]
def distinct(): RDD[(K, V)]
def distinct(numPartitions: Int)(implicit ord: Ordering[(K, V)]): RDD[(K, V)]
```

Primeiro Estudo de Caso

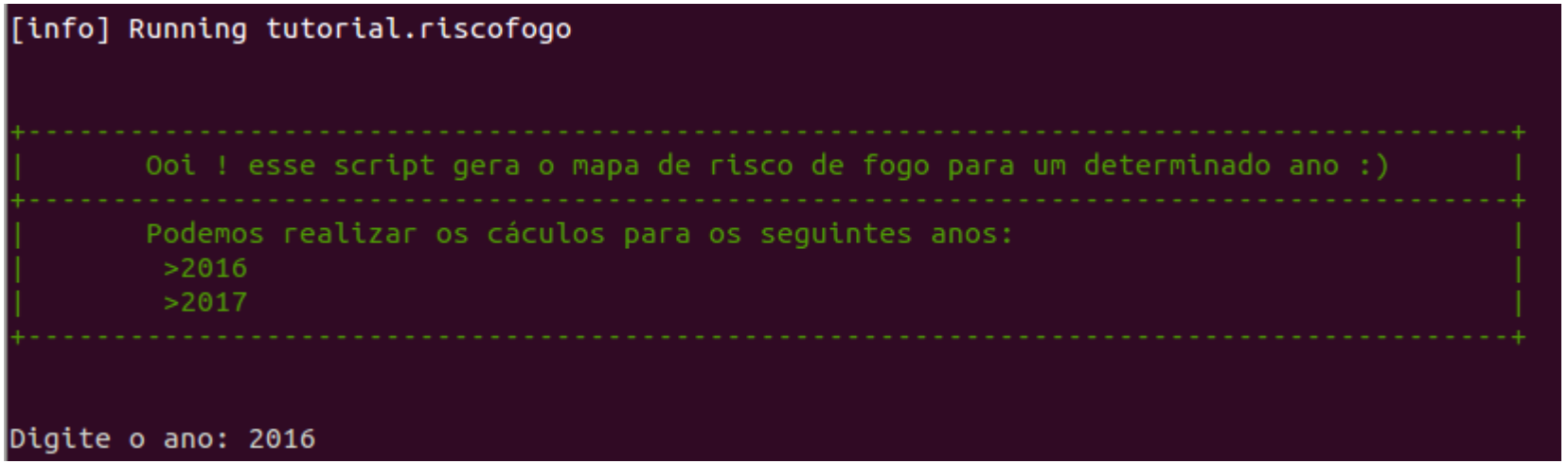
Cálculo do Risco de Fogo

Risco de Fogo...

- Queimadas, quando não controladas, geram prejuízos de diversas naturezas
- Projeto Queimadas (INPE)
- Seu potencial depende de uma série de fatores
  - Bioma
  - Histórico de precipitação
  - Temperatura
  - Umidade

Cálculo do Risco de Fogo...

- Cálculo do Risco de Fogo (serial) para os anos de 2016 e 2017



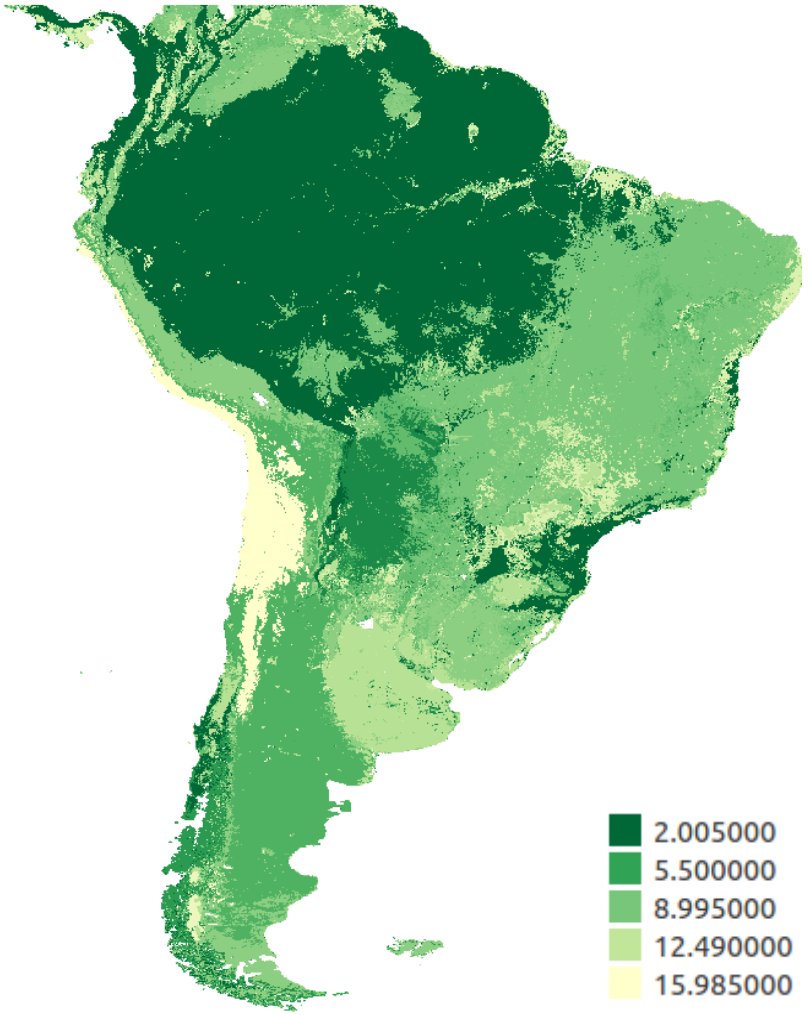
Cálculo do Risco de Fogo...

- Conjunto de dados
  - 120 imagens de precipitação
  - 1 mapa de vegetação
  - 1 imagem umidade
  - 1 imagem temperatura

Cálculo do Risco de Fogo...

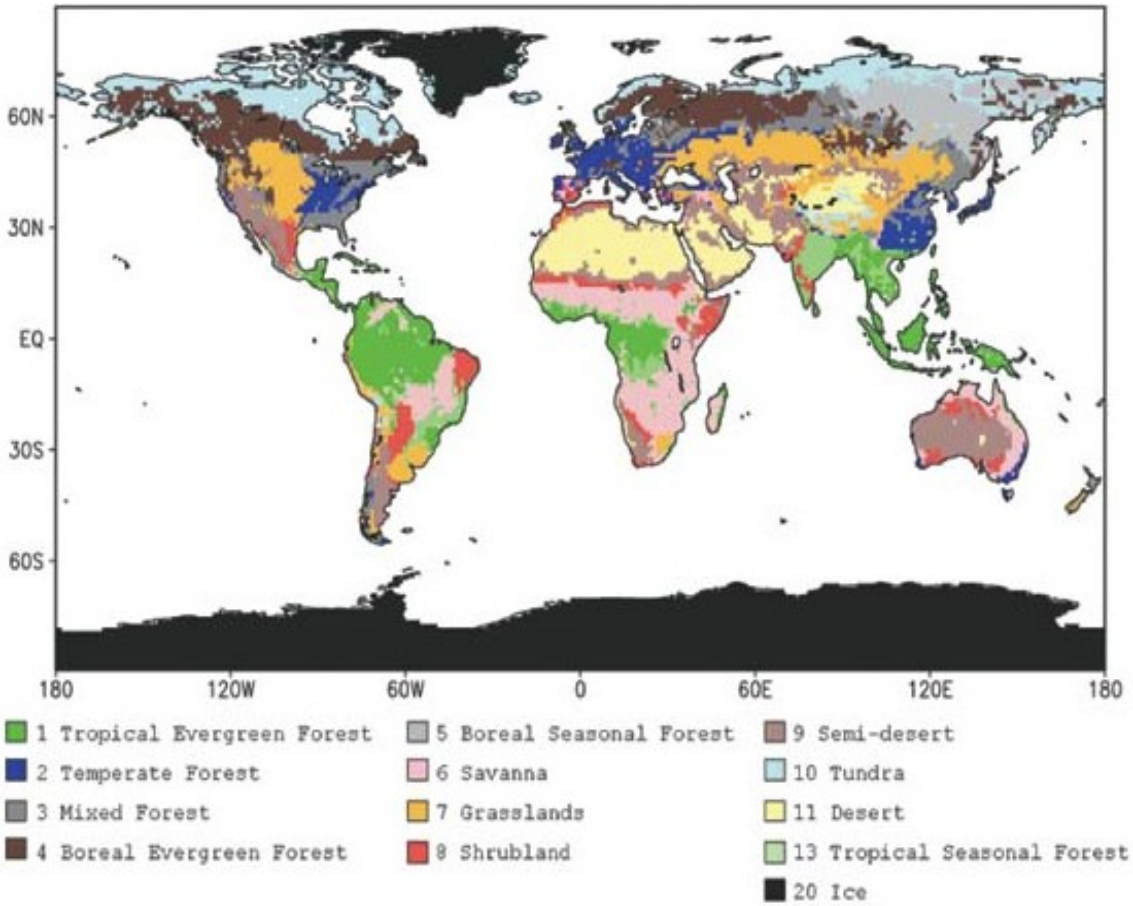
- Mapa de vegetação
  - Cada bioma teve um peso diferente no cálculo do risco do fogo
    - Biomas que com vegetação mais densa → pesos menores

```
def logica_ComputeVegetationFactor(vegetation_class: Double) : Double = {

        if (vegetation_class == 0 || vegetation_class == 13 || vegetation_class==15 || vegetation_class == 16) {
            return 0
        }
        if (vegetation_class == 2 || vegetation_class==11) {
            return 1.5
        }
        if (vegetation_class==4) {
            return 1.72
        }
        if (vegetation_class==1 || vegetation_class==3 || vegetation_class==5) {
            return 2
        }
        if (vegetation_class==6 || vegetation_class==8) {
            return 2.4
        }
        if (vegetation_class==7 || vegetation_class==9) {
            return 3
        }
        if (vegetation_class==12 || vegetation_class==14) {
            return 4
        }
        if (vegetation_class==10) {
            return 6
        }
        return 0
}
```



Tipos de vegetação - IGBP – Landcover/2012

```
def logica_ComputeVegetationFactor(vegetation_class: Double) : Double = {

            if (vegetation_class == 0 || vegetation_class == 13 || vegetation_class==15 || vegetation_class == 16) {
                return 0
            }
            if (vegetation_class == 2 || vegetation_class==11) {
                return 1.5
            }
            if (vegetation_class==4) {
                return 1.72
            }
            if (vegetation_class==1 || vegetation_class==3 || vegetation_class==5) {
                return 2
            }
            if (vegetation_class==6 || vegetation_class==8) {
                return 2.4
            }
            if (vegetation_class==7 || vegetation_class==9) {
                return 3
            }
            if (vegetation_class==12 || vegetation_class==14) {
                return 4
            }
            if (vegetation_class==10) {
                return 6
            }
            return 0
  }
```

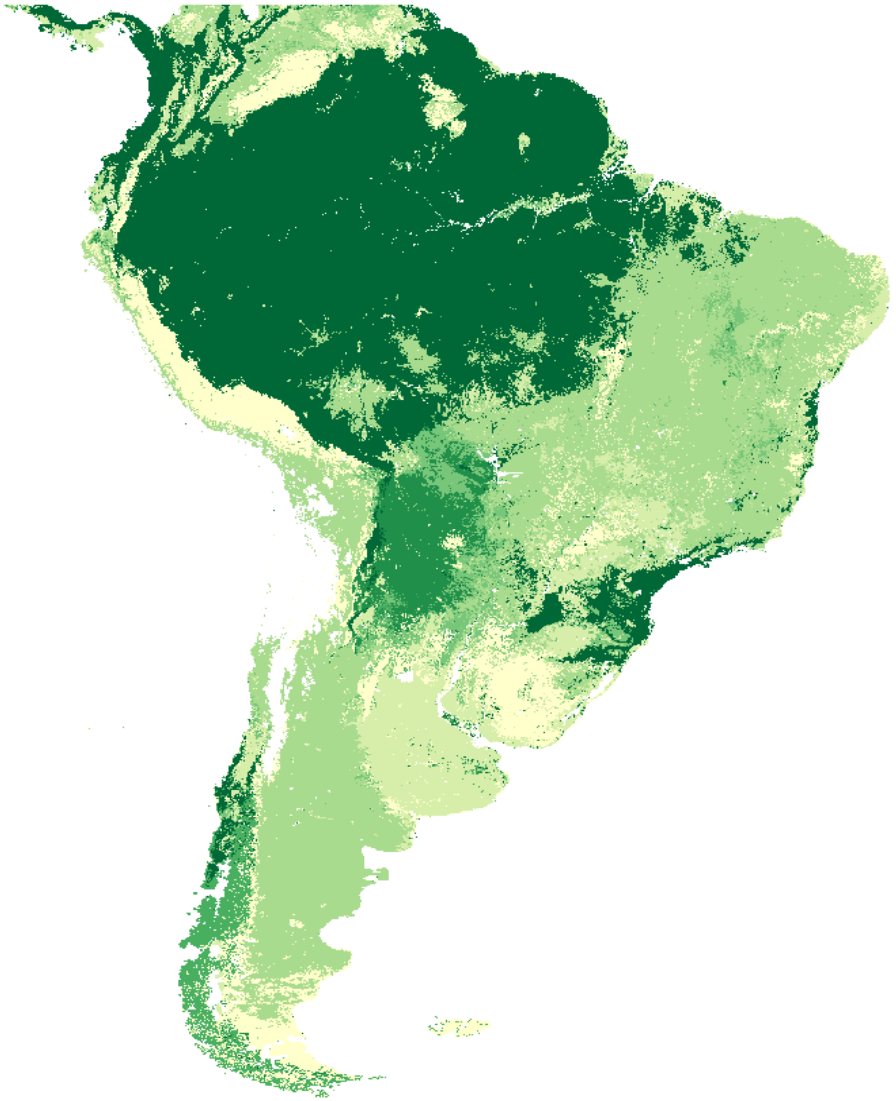


Cálculo do Risco de Fogo...

- Conversão dos dados: IGBP → IBGE

```
        def ConvertFromIGBP_To_IBGE(igbp_class: Double) : Double = {

                if (igbp_class==0 || igbp_class==11 || igbp_class==13 || igbp_class==15 ||
igbp_class==16) {
                        return 0
                }
                if (igbp_class==10) {
                        return 1
                }
                if (igbp_class==12 || igbp_class==14) {
                        return 2
                }
                if (igbp_class==7 || igbp_class==9) {
                        return 3
                }
                if (igbp_class==6 || igbp_class==8) {
                        return 4
                }
                if (igbp_class==1 || igbp_class==3 || igbp_class==5) {
                        return 5
                }
                if (igbp_class==4) {
                        return 6
                }
                if (igbp_class==2) {
                        return 7
                }

                return -9999

        }
```

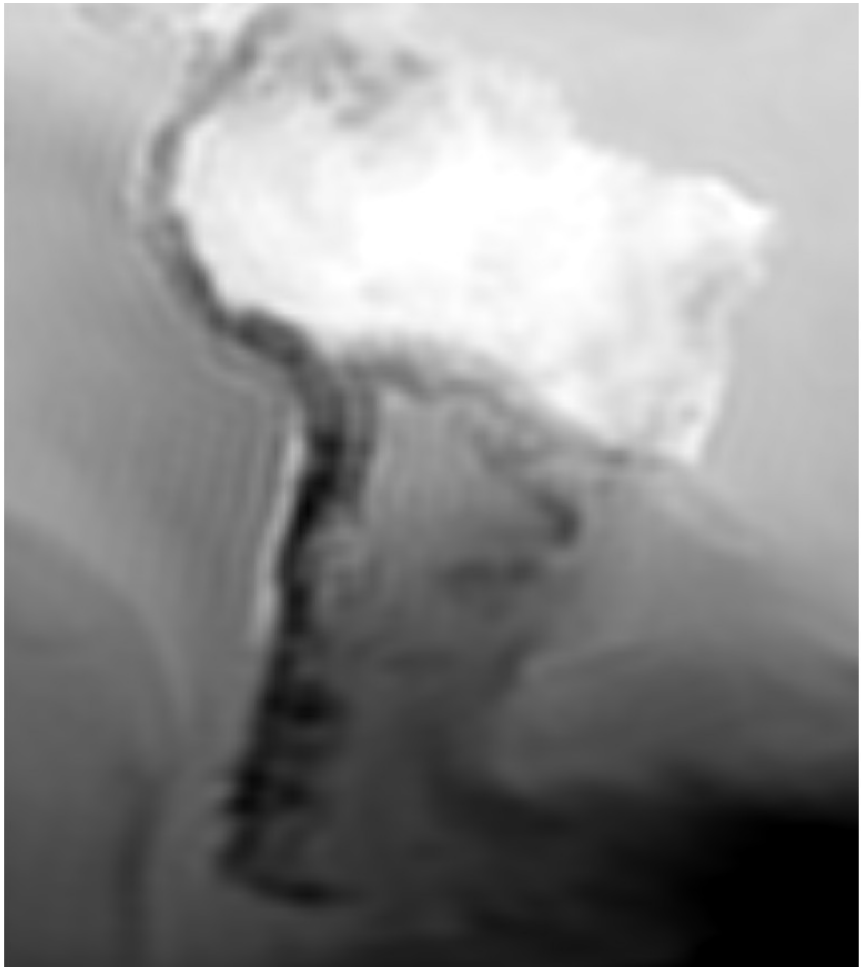
Tipos de vegetação – Landcover/2012



Tipos de vegetação IBGE – Landcover/2012

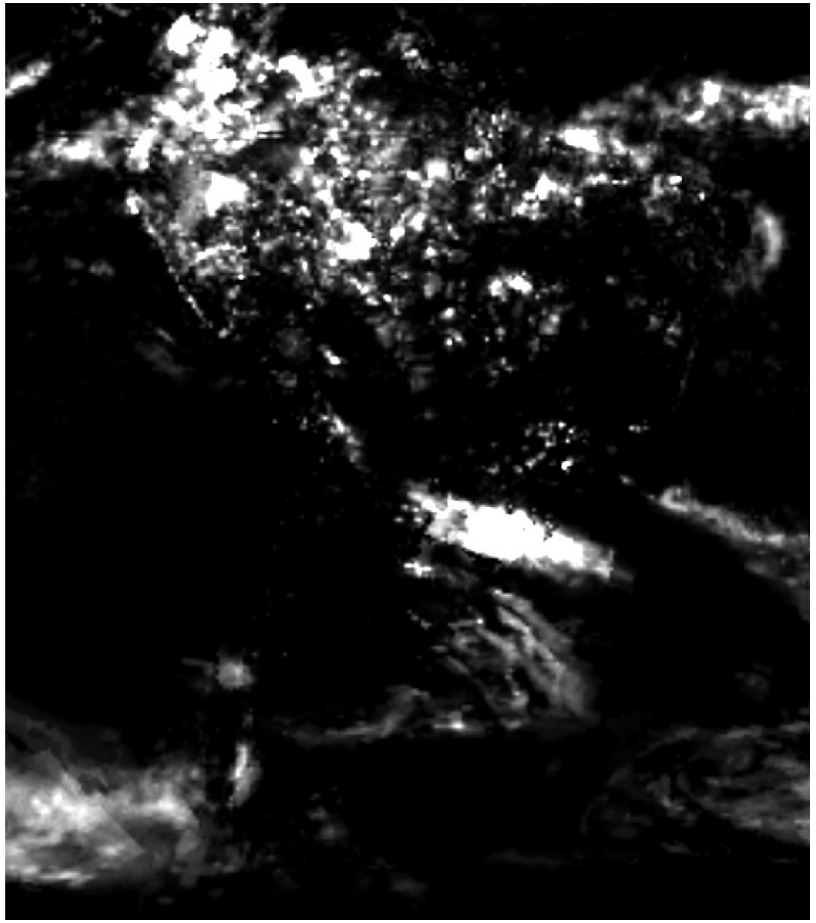
Cálculo do Risco de Fogo...

- Dados de temperatura e umidade



Cálculo do Risco de Fogo...

- Precipitação → 120 dias



```
val p10_total = p10_6.tile +
                p10_7.tile +
                p10_8.tile +
                p10_9.tile +
                p10_10.tile
println("Realizando a soma dos tiles no intervalo de tempo 15d-11d...")
val p15_total = p15_11.tile +
                p15_12.tile +
                p15_13.tile +
                p15_14.tile +
                p15_15.tile
println("Realizando a soma dos tiles no intervalo de tempo 30d-16d...")
val p30_total = p30_16.tile +
                p30_17.tile +
                p30_18.tile +
```

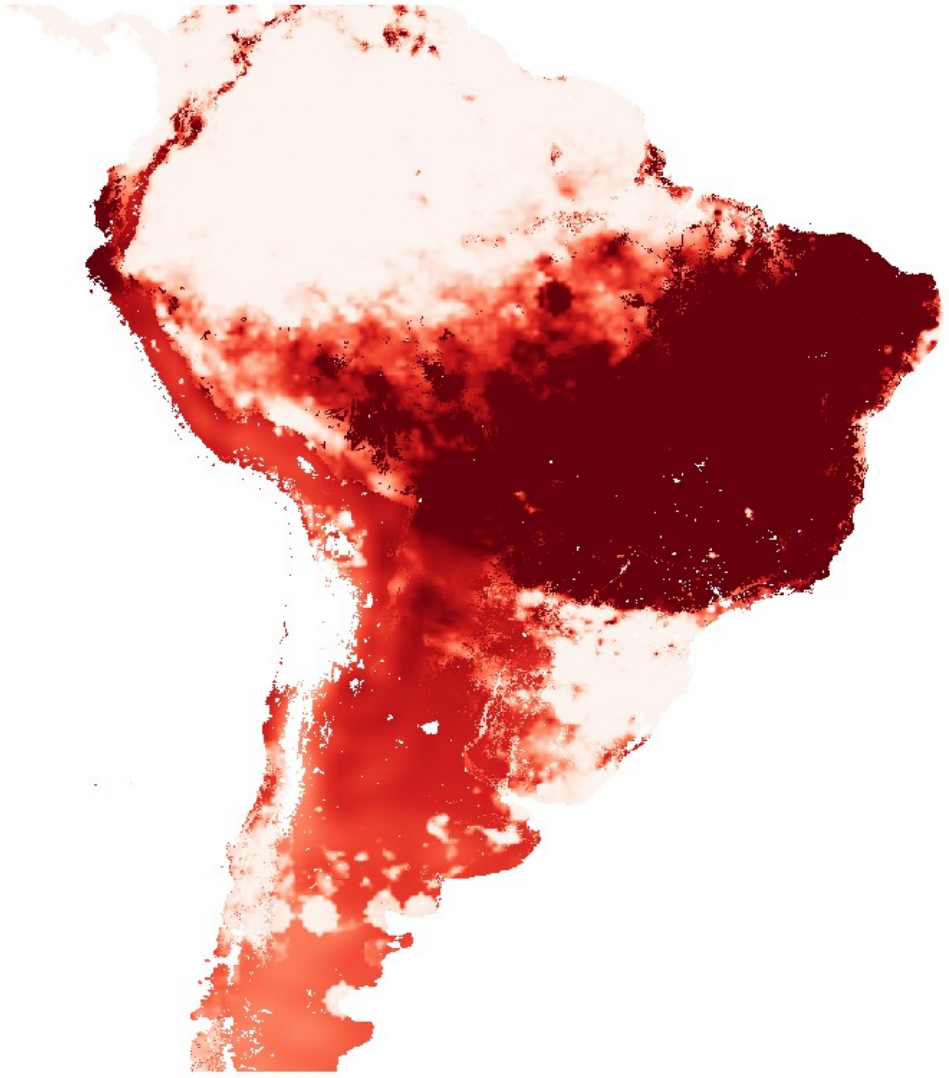
Sobreposição de camadas

Cálculo do Risco de Fogo...

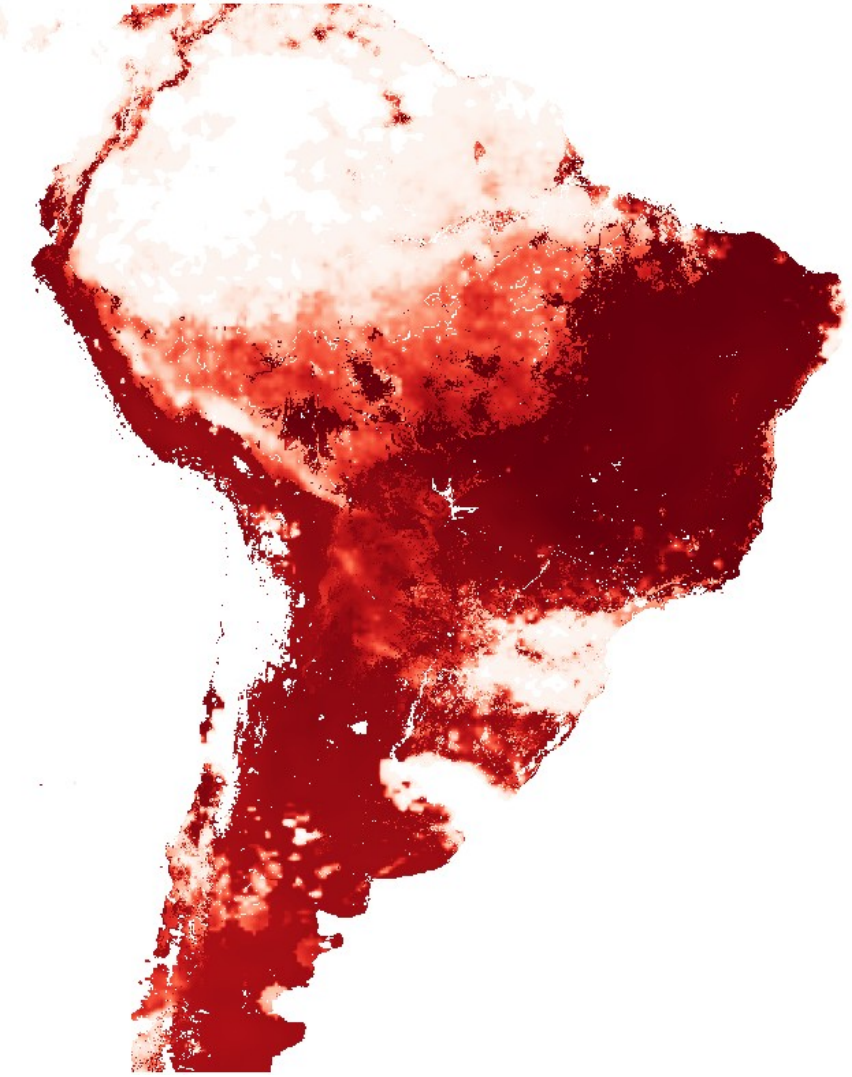
```
println(Console.CYAN+"Realizando o cálculo do fp..."+Console.WHITE+"")
println("Realizando o cálculo do fp1...")
var fp1 = p1_total.tile.mapDouble{x => x*(-0.14)}
fp1 = fp1.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp2...")
var fp2 = p2_total.tile.mapDouble{x => x*(-0.07)}
fp2 = fp2.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp3...")
var fp3 = p3_total.tile.mapDouble{x => x*(-0.04)}
fp3 = fp3.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp4...")
var fp4 = p4_total.tile.mapDouble{x => x*(-0.03)}
fp4 = fp4.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp5...")
var fp5 = p5_total.tile.mapDouble{x => x*(-0.02)}
fp5 = fp5.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp10...")
var fp10 = p10_total.tile.mapDouble{x => x*(-0.01)}
fp10 = fp10.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp15...")
var fp15 = p15_total.tile.mapDouble{x => x*(-0.008)}
fp15 = fp15.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp30...")
var fp30 = p30_total.tile.mapDouble{x => x*(-0.004)}
fp30 = fp30.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp60...")
var fp60 = p60_total.tile.mapDouble{x => x*(-0.002)}
fp60 = fp60.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp90...")
var fp90 = p90_total.tile.mapDouble{x => x*(-0.001)}
fp90 = fp90.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println("Realizando o cálculo do fp120...")
var fp120 = p120_total.tile.mapDouble{x => x*(-0.0007)}
fp120 = fp120.tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x)}
println(Console.YELLOW+"Realizou os cálculos direitinho :) ...")
println(" ")
```

Intervalos com pesos distintos

```
                println(Console.CYAN+"Obtendo tipo da vegetação..."+Console.WHITE+"")
                val vegetation_type = SinglebandGeoTiff("data/RiscoFogo/vegetacao/
vegetacao_landcover_2012.tif")

                val A = Calculations.ComputeVegetationFactor(vegetation_type.tile)

                val vegetation_type_ibge = SinglebandGeoTiff("data/RiscoFogo/vegetacao/
vegetacao_landcover_2012_ibge.tif")
                println(Console.CYAN+"Ajuste dos dias de secura pelo máximo..."+Console.WHITE+"")
                var _PSE = Calculations.CapByVegetation(vegetation_type_ibge.tile,PSE)

                var A_mult_PSE =  Calculations.multiplicar(A, _PSE)
                var A_mult_PSE_subtract_90 = A_mult_PSE.tile.mapDouble{x => (x-90)*3.1416/180}
                var sin_A_mult_PSE = A_mult_PSE_subtract_90.tile.localSin()
                var som_1 = sin_A_mult_PSE.tile.mapDouble{ x => x+1.0}
                var RB_1 = som_1.mapDouble{x => 0.9*x*0.5}

                println(Console.CYAN+"Cálculo de risco de fogo..."+Console.WHITE+"")
                var ufac = SinglebandGeoTiff(s"data/RiscoFogo/umidade/UMRS${ano}071618${ano}
071618.tif")
                val _ufac = ufac.tile.mapDouble{x => x*(-0.006)+1.3}
                println(Console.CYAN+"Fator de umidade..."+Console.WHITE+"")

                var tfac = SinglebandGeoTiff(s"data/RiscoFogo/temperatura/TEMP${ano}071618${ano}
071618.tif")
                val _tfac = tfac.tile.mapDouble{x => x*(0.02)+0.4}

                println(Console.CYAN+"Cálculo de temperatura..."+Console.WHITE+"")
                val ufac_tfac = Calculations.multiplicar(_ufac, _tfac)
                val valor_total = Calculations.multiplicar(RB_1,ufac_tfac)
                val valor_final = valor_total.tile.normalize(0,8,0,1)
                // gerando arquivo de saída
                val mb = ArrayMultibandTile(valor_final).convert(DoubleConstantNoDataCellType)
                //mb.foreach{x => println(x.getDouble())}
                println(Console.CYAN+"Gerando mapa de risco de fogo..."+Console.WHITE+"")
                MultibandGeoTiff(mb, p1.extent, p1.crs).write(s"data/saida/mapa_risco_fogo_
${ano}.tif")
```



Mapa do Risco de Fogo
15/07/2016
TerraMA



Mapa do Risco de Fogo
15/07/2016
GeoTrellis

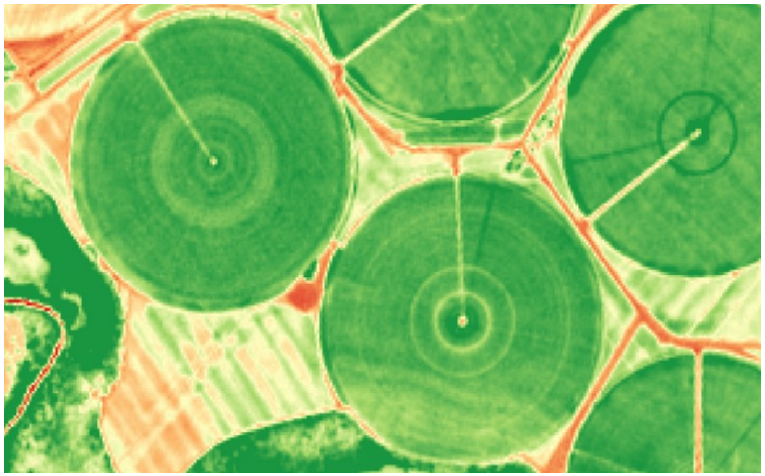
Segundo Estudo de Caso

Cálculo do NDVI – Anomalias de Vegetação

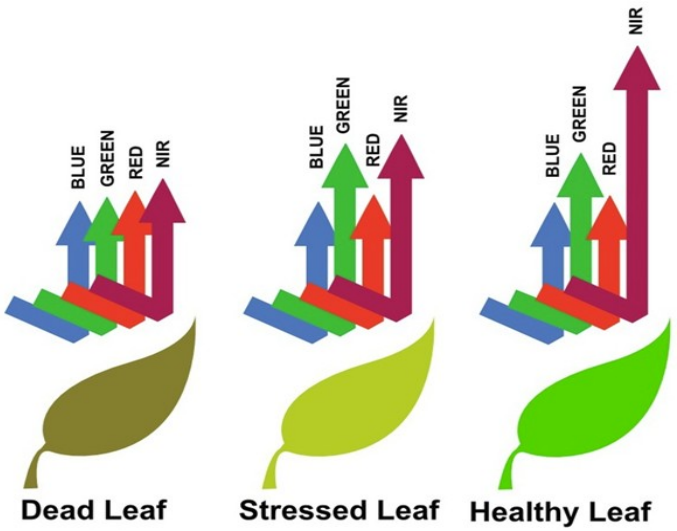
Anomalias de vegetação...

- Imagens de satélite podem auxiliar na identificação de anomalias em plantações, solo, corpos d'água e florestas
  - Nematóides
  - Estresse hídrico
  - Erosões
  - Falhas no sistema de irrigação
- NDVI auxilia na identificação de anomalias de vegetação

Fonte: agrosmart

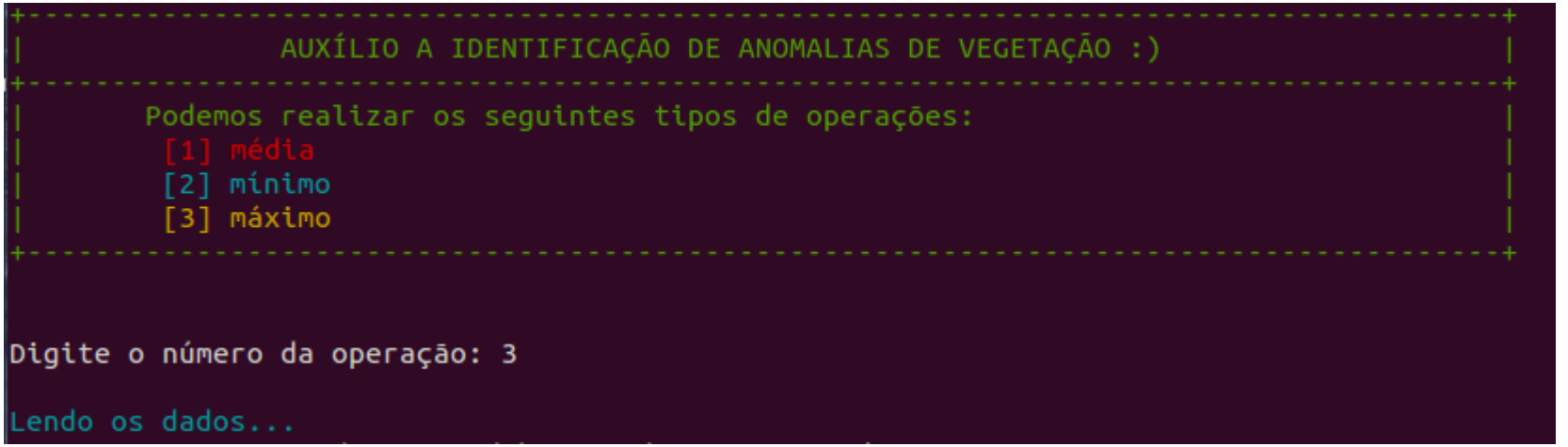


Anomalia por falhas no sistema de irrigação
Consequência: comprometimento do rendimento da lavoura em determinadas regiões



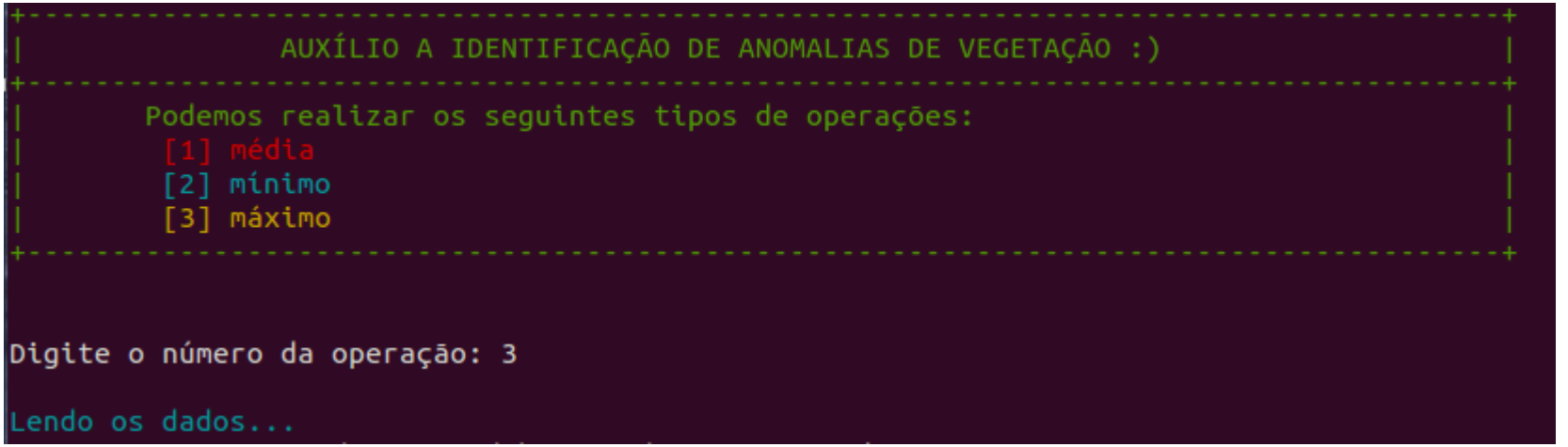
Anomalias de vegetação

- Com base no NDVI → média, máxima, mínima

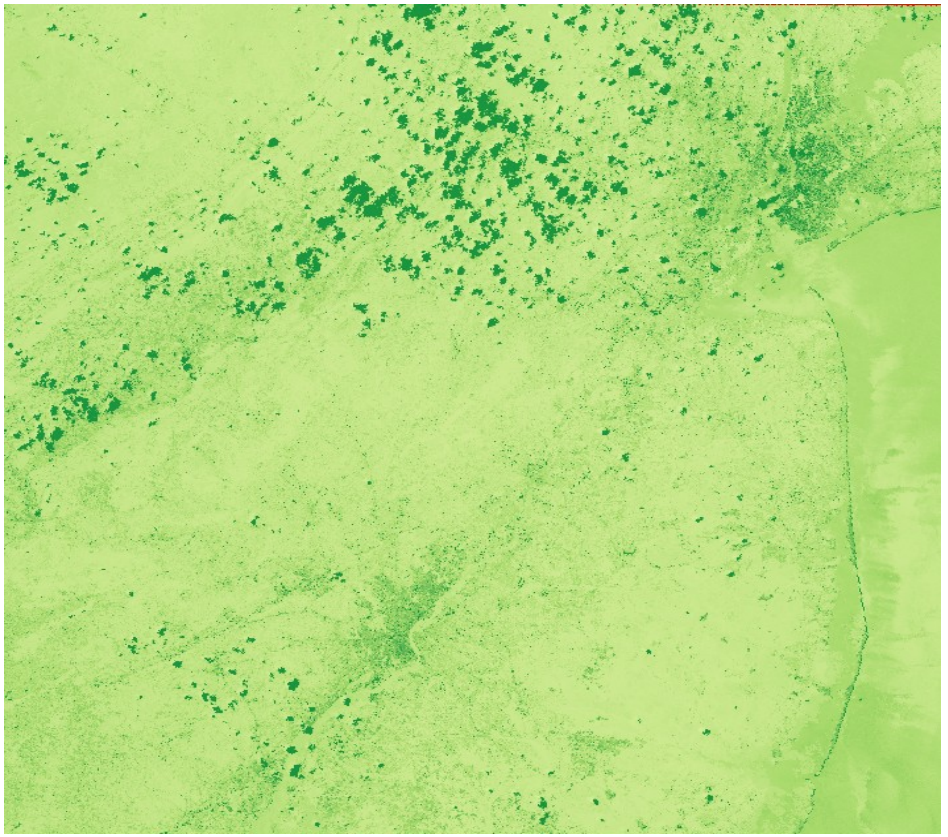


Anomalias de vegetação

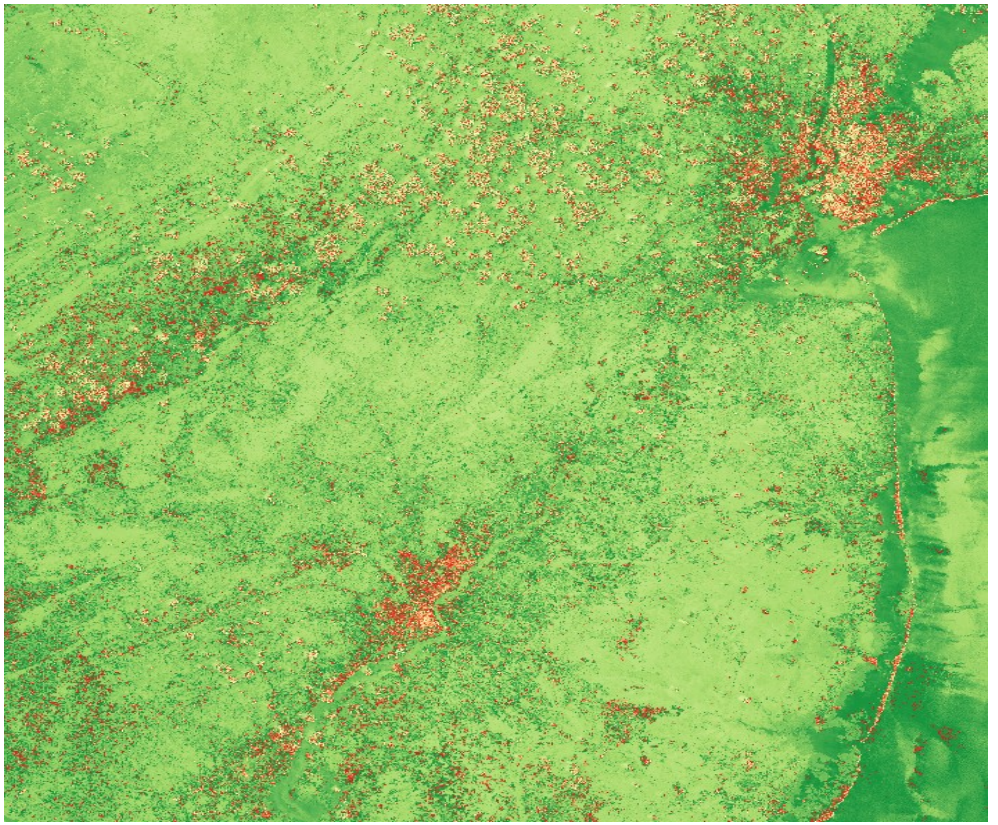
- Com base no NDVI → média, máxima, mínima



Cálculo do NDVI...



NDVI máximo - GeoTrellis



NDVI média - GeoTrellis

Álgebra de Mapas

Algumas operações de forma distribuída

```
//p1
val p1: RDD[(ProjectedExtent, MultibandTile)] =
        sc.hadoopMultibandGeoTiffRDD(s"data/RiscoFogo/precipitacao_${ano}/S10648241_${ano}07151200${nc}.tif")

val (_, rasterMetaData_p1) =
        TileLayerMetadata.fromRdd(p1, FloatingLayoutScheme(512))

val tiled_p1: RDD[(SpatialKey, MultibandTile)] =
        p1
                .tileToLayout(rasterMetaData_p1.cellType, rasterMetaData_p1.layout, Bilinear)
                .repartition(100)
//Converter para Seq (pq o parallelize só le Seq)
val p1_geotiff_seq = Seq.apply(p1)
//paralelizando em 10 partes
val p1_distribuido = sc.parallelize(p1_geotiff_seq, 10)


val p11 = SinglebandGeoTiff(s"data/RiscoFogo/precipitacao_2017/S10648241_201707151200.tif")
val cols = p11.tile.cols
val rows = p11.tile.rows
reprojected
        .foreach{case(key,iterator) =>

                val tile = DoubleArrayTile.empty(cols, rows)
                iterator.foreach{case(i: Int,j: Int,value: Double) =>
                                tile.setDouble(i,j,value)
                }

        SinglebandGeoTiff(tile, p11.extent, p11.crs).write("data/lala.tif")
}
```

```
    val p1 = SinglebandGeoTiff(s"data/RiscoFogo/precipitacao_${ano}/S10648241_${ano}07151200${nc}.tif")
```

```
println("Realizando a soma dos tiles no intervalo de tempo 10d-6d...")
val p10_total = p10_6_distribuido.++(p10_7_distribuido)
                p10_total.++(p10_8_distribuido)
                p10_total.++(p10_9_distribuido)
                p10_total.++(p10_10_distribuido)
```

```
println("Realizando a soma dos tiles no intervalo de tempo 15d-11d...")
val p15_total = p15_11.tile +
                p15_12.tile +
                p15_13.tile +
                p15_14.tile +
                p15_15.tile
```

```
println(Console.CYAN+"Realizando o cálculo do fp..."+Console.WHITE+"")
def funcao_1(tile: Tile, const: Double) = {
        tile.mapDouble{x => math.pow(2.71828182845904523536028 7,x*(const))}
}



println("Realizando o cálculo do fp1...")
var fp1 = p1_total.map{case(x:Tile) => funcao_1(x,-0.014)}
println("Realizando o cálculo do fp2...")
var fp2 = p2_total.map{case(x:Tile) => funcao_1(x,-0.07)}
println("Realizando o cálculo do fp3...")
var fp3 = p3_total.map{case(x:Tile) => funcao_1(x,-0.04)}
println("Realizando o cálculo do fp4...")
var fp4 = p4_total.map{case(x:Tile) => funcao_1(x,-0.03)}
println("Realizando o cálculo do fp5...")
var fp5 = p5_total.map{case(x:Tile) => funcao_1(x,-0.02)}
println("Realizando o cálculo do fp10...")
var fp10 = p10_total.map{case(x:Tile) => funcao_1(x,-0.01)}
println("Realizando o cálculo do fp15...")
var fp15 = p15_total.map{case(x:Tile) => funcao_1(x,-0.008)}
println("Realizando o cálculo do fp30...")
var fp30 = p30_total.map{case(x:Tile) => funcao_1(x,-0.004)}
println("Realizando o cálculo do fp60...")
var fp60 = p60_total.map{case(x:Tile) => funcao_1(x,-0.002)}
println("Realizando o cálculo do fp90...")
var fp90 = p90_total.map{case(x:Tile) => funcao_1(x,-0.001)}
println("Realizando o cálculo do fp120...")
var fp120 = p120_total.map{case(x:Tile) => funcao_1(x,-0.0007)}
println(Console.YELLOW+"Realizou os cálculos direitinho :) ...")
println(" ")
```

Conclusões

- Documentação da biblioteca GeoTrellis ainda é muito deficiente

collect

```
public Object collect()
```

Return an array that contains all of the elements in this RDD.

Returns:

(undocumented)

toArray

```
public Object toArray()
```

Return an array that contains all of the elements in this RDD.

Returns:

(undocumented)

toLocalIterator

```
public scala.collection.Iterator<T> toLocalIterator()
```

Return an iterator that contains all of the elements in this RDD.

The iterator will consume as much memory as the largest partition in this RDD.

Returns:

(undocumented)

Conclusões

- Muitas operações de interesse para o tratamento de dados em grande escala ainda não foram implementadas
  - Multiplicação de matrizes

- No entanto, há muita informação em relação a linguagem *scala*