Table of Contents

Raster Recipes 1
- Raster Properties 1
- Raster Bands 6
- Raster Values 8
- Raster Processing 14
- World File 41
- Map Algebra 42

Raster Recipes

The Raster classes are in the **geoscript.layer** package.

Raster Properties

Read a Raster from a File

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")
```

Get the Raster's Bounds.

```
Bounds bounds = raster.bounds
println "Bounds: ${bounds}"
```

```
Bounds: (-179.99999999999997,-89.99999999998205,179.99999999996405,90.0,EPSG:4326)
```

Get the Raster's Projection.

```
Projection projection = raster.proj
println "Projection: ${projection}"
```

```
Projection: EPSG:4326
```

Get the Raster's Size.

```
List size = raster.size
println "Size: ${size[0]}x${size[1]}"
```

```
Size: 800x400
```

Get the Raster's number of columns and rows.

```
int cols = raster.cols
int rows = raster.rows
println "Columns: ${cols} Rows: ${rows}"
```

```
Columns: 800 Rows: 400
```

Get the Raster's Bands.

```
List<Band> bands = raster.bands
println "Bands:"
bands.each { Band band ->
    println "  ${band}"
}
```

```
Band:
  RED_BAND
  GREEN_BAND
  BLUE_BAND
```

Get the Raster's block size.

```
List blockSize = raster.blockSize
println "Block size: ${blockSize[0]}x${blockSize[1]}"
```

```
Block size: 800x8
```

Get the Raster's pixel size.

```
List pixelSize = raster.pixelSize
println "Pixel size: ${pixelSize[0]}x${pixelSize[1]}"
```

```
Pixel size: 0.44999999999995505x0.4499999999999551
```

Get more information about a Raster's Bounds.

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")
List<Band> bands = raster.bands
bands.each { Band band ->
    println "${band}"
    println "  Min = ${band.min}"
    println "  Max = ${band.max}"
    println "  No Data = ${band.noData}"
    println "  Is No Data = ${band.isNoData(12.45)}"
    println "  Unit = ${band.unit}"
    println "  Scale = ${band.scale}"
    println "  Offset = ${band.offset}"
    println "  Type = ${band.type}"
}
```

```
RED_BAND
  Min = 0.0
  Max = 255.0
  No Data = [0.0]
  Is No Data = false
  Unit = null
  Scale = 1.0
  Offset = 0.0
  Type = byte

GREEN_BAND
  Min = 0.0
  Max = 255.0
  No Data = [0.0]
  Is No Data = false
  Unit = null
  Scale = 1.0
  Offset = 0.0
  Type = byte

BLUE_BAND
  Min = 0.0
  Max = 255.0
  No Data = [0.0]
  Is No Data = false
  Unit = null
  Scale = 1.0
  Offset = 0.0
  Type = byte
```

Get the minimum and maximum values from a Raster for a band

```
double minValue = raster.getMinValue(0)
double maxValue = raster.getMaxValue(0)
println "Min values: ${minValue} Max values: ${maxValue}"
```

```
Min values: 56.0 Max values: 255.0
```

Get the minimum and maximum values from a Raster for each band

```
Map extrema = raster.extrema
println "Min values: ${extrema.min} Max values: ${extrema.max}"
```

```
Min value: [56.0, 84.0, 91.0] Max value: [255.0, 255.0, 255.0]
```

Get a Point at the given pixel location.

```
Point point = raster.getPoint(7,9)
println "Geographic location at pixel 7,9 is ${point}"
```

```
Geographic location at pixel 7,9 is POINT (-176.625 85.7249984741211)
```

Get a Pixel location at the given Point.

```
List pixel = raster.getPixel(new Point(-176.625, 85.72499))
println "Pixel coordinates at POINT (-176.625 85.7249984741211) is ${pixel[0]},
${pixel[1]}"
```

```
Pixel coordinates at POINT (-176.625 85.7249984741211) is 7.0, 9.0
```

Determine whether the Raster covers the given Point.

```
boolean containsPoint = raster.contains(new Point(-180, -90))
println "Does raster cover point? ${containsPoint}"
```

```
Does raster cover point? true
```

Determine whether the Raster covers the given Pixel.

```
boolean containsPixel = raster.contains(500,600)
println "Does raster cover pixel? ${containsPixel}"
```

```
Does raster cover pixel? false
```

Get a RenderedImage from the Raster

```
RenderedImage image = raster.image
```

Dispose of the Raster when you are done

```
raster.dispose()
```

Raster Bands

Create a Band

```
Band band = new Band(
    "red", ①
    0,     ②
    255    ③
)
println "Band = ${band.toString()} Min = ${band.min} Max = ${band.max}"
```

① Description

② Minimum value

③ Maximum value

```
Band = red Min = 0.0 Max = 255.0
```

Create a Band with a no data value

```
Band band = new Band(
        "red", ①
        0,     ②
        255,   ③
        255    ④
)
println "Band = ${band.toString()} Min = ${band.min} Max = ${band.max} No Data =
${band.noData[0]}"
```

① Description

② Minimum value

③ Maximum value

④ No data value

```
Band = red Min = 0.0 Max = 255.0 No Data = 255.0
```

Create a new Raster from Bands and set values to a random color.

```
Raster raster = new Raster(
        new Bounds(-180,-90,180,90,"EPSG:4326"),
        400,300,
        [
                new Band("red", 0, 255, 256),
                new Band("green", 0, 255, 256),
                new Band("blue", 0, 255, 256)
        ]
)

// Set values of each pixel
raster.eachCell { double value, double x, double y ->
    Color color = Color.randomPastel
    raster.setValue([x,y], color.rgb[0], 0)
    raster.setValue([x,y], color.rgb[1], 1)
    raster.setValue([x,y], color.rgb[2], 2)
}
```

Raster Values

Get values from a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
```

Get values from a Raster with a Point.

```
double elevation = raster.getValue(new Point(-121.799927,46.867703))
println elevation
```

```
3069.0
```

Get values from a Raster with a Pixel Location.

```
List pixel = [100,200]
elevation = raster.getValue(pixel)
println elevation
```

```
288.0
```

Get neighboring values from a Raster with a Point Location.

```
Map neighborsOfPoint = raster.getNeighbors(new Point(-176.625, 85.72499), 0)
println "Values neighboring POINT (-176.625 85.7249984741211) = ${neighborsOfPoint}"
```

```
Values neighboring POINT (-176.625 85.7249984741211) = [nw:103.0, n:104.0, ne:109.0,
e:109.0, se:111.0, s:110.0, sw:110.0, w:108.0]
```

Get neighboring values from a Raster with a Pixel Location.

```
Map neighborsOfPixel = raster.getNeighbors([7,9], 0)
println "Values neighboring pixel 7,9 = ${neighborsOfPixel}"
```

```
Values neighboring pixel 7,9 = [nw:103.0, n:104.0, ne:109.0, e:109.0, se:111.0,
s:110.0, sw:110.0, w:108.0]
```

Get values from a Raster for a range of pixels in a list of lists.

```
int x = 10
int y = 8
int w = 5
int h = 6
int band = 0
List values = raster.getValues(x, y, w, h, band, false)
println values
```

```
[[1032, 1186, 1340, 1435, 1301], [1143, 1143, 1193, 1224, 1313], [942, 938, 966, 982,
1129], [746, 835, 912, 949, 1028], [723, 948, 1130, 1244, 1211], [673, 890, 1100,
1133, 1024]]
```

Get values from a Raster for a range of pixels in a flat list.

```
List flatValues = raster.getValues(x, y, w, h, band, true)
println flatValues
```

```
[1032, 1186, 1340, 1435, 1301, 1143, 1143, 1193, 1224, 1313, 942, 938, 966, 982, 1129,
746, 835, 912, 949, 1028, 723, 948, 1130, 1244, 1211, 673, 890, 1100, 1133, 1024]
```

Get values from a Raster for a range of pixels as a pretty printed string.

```
String valuesAsString = raster.getValuesAsString(x, y, w, h, band, prettyPrint: true)
println valuesAsString
```

```
-----------------------------------------------------
| 1032.00 | 1186.00 | 1340.00 | 1435.00 | 1301.00 |
-----------------------------------------------------
| 1143.00 | 1143.00 | 1193.00 | 1224.00 | 1313.00 |
-----------------------------------------------------
|  942.00 |  938.00 |  966.00 |  982.00 | 1129.00 |
-----------------------------------------------------
|  746.00 |  835.00 |  912.00 |  949.00 | 1028.00 |
-----------------------------------------------------
|  723.00 |  948.00 | 1130.00 | 1244.00 | 1211.00 |
-----------------------------------------------------
|  673.00 |  890.00 | 1100.00 | 1133.00 | 1024.00 |
-----------------------------------------------------
```

Iterate over the cells in a Raster.

```
raster.eachCell(bounds: [0,0,5,5]) { double value, double pixelX, double pixelY ->
    println "${pixelX},${pixelY} = ${value}"
}
```

```
0.0,0.0 = 1061.0
1.0,0.0 = 996.0
2.0,0.0 = 945.0
3.0,0.0 = 960.0
4.0,0.0 = 904.0
0.0,1.0 = 1167.0
1.0,1.0 = 1149.0
2.0,1.0 = 1085.0
3.0,1.0 = 966.0
4.0,1.0 = 862.0
0.0,2.0 = 1112.0
1.0,2.0 = 998.0
2.0,2.0 = 882.0
3.0,2.0 = 775.0
4.0,2.0 = 700.0
0.0,3.0 = 990.0
1.0,3.0 = 850.0
2.0,3.0 = 715.0
3.0,3.0 = 638.0
4.0,3.0 = 654.0
0.0,4.0 = 833.0
1.0,4.0 = 706.0
2.0,4.0 = 611.0
3.0,4.0 = 681.0
4.0,4.0 = 841.0
```

Iterate over a window of cells in a Raster.

```
raster.eachWindow (bounds: [0,0,8,8], window: [2,2]) { Number[][] windowsValues,
double pixelX, double pixelY ->
    println "${pixelX},${pixelY} = ${windowsValues}"
}
```

```
0.0,0.0 = [[1061, 996], [1167, 1149]]
1.0,0.0 = [[996, 945], [1149, 1085]]
2.0,0.0 = [[945, 960], [1085, 966]]
3.0,0.0 = [[960, 904], [966, 862]]
4.0,0.0 = [[904, 727], [862, 696]]
5.0,0.0 = [[727, 744], [696, 748]]
6.0,0.0 = [[744, 934], [748, 900]]
7.0,0.0 = [[934, 1099], [900, 1042]]
0.0,1.0 = [[1167, 1149], [1112, 998]]
1.0,1.0 = [[1149, 1085], [998, 882]]
2.0,1.0 = [[1085, 966], [882, 775]]
3.0,1.0 = [[966, 862], [775, 700]]
4.0,1.0 = [[862, 696], [700, 661]]
5.0,1.0 = [[696, 748], [661, 818]]
6.0,1.0 = [[748, 900], [818, 995]]
```

```
7.0,1.0 = [[900, 1042], [995, 1125]]
0.0,2.0 = [[1112, 998], [990, 850]]
1.0,2.0 = [[998, 882], [850, 715]]
2.0,2.0 = [[882, 775], [715, 638]]
3.0,2.0 = [[775, 700], [638, 654]]
4.0,2.0 = [[700, 661], [654, 826]]
5.0,2.0 = [[661, 818], [826, 945]]
6.0,2.0 = [[818, 995], [945, 922]]
7.0,2.0 = [[995, 1125], [922, 1078]]
0.0,3.0 = [[990, 850], [833, 706]]
1.0,3.0 = [[850, 715], [706, 611]]
2.0,3.0 = [[715, 638], [611, 681]]
3.0,3.0 = [[638, 654], [681, 841]]
4.0,3.0 = [[654, 826], [841, 949]]
5.0,3.0 = [[826, 945], [949, 1084]]
6.0,3.0 = [[945, 922], [1084, 1054]]
7.0,3.0 = [[922, 1078], [1054, 1093]]
0.0,4.0 = [[833, 706], [652, 618]]
1.0,4.0 = [[706, 611], [618, 548]]
2.0,4.0 = [[611, 681], [548, 631]]
3.0,4.0 = [[681, 841], [631, 694]]
4.0,4.0 = [[841, 949], [694, 877]]
5.0,4.0 = [[949, 1084], [877, 1018]]
6.0,4.0 = [[1084, 1054], [1018, 1142]]
7.0,4.0 = [[1054, 1093], [1142, 1172]]
0.0,5.0 = [[652, 618], [631, 579]]
1.0,5.0 = [[618, 548], [579, 506]]
2.0,5.0 = [[548, 631], [506, 483]]
3.0,5.0 = [[631, 694], [483, 556]]
4.0,5.0 = [[694, 877], [556, 686]]
5.0,5.0 = [[877, 1018], [686, 825]]
6.0,5.0 = [[1018, 1142], [825, 1004]]
7.0,5.0 = [[1142, 1172], [1004, 1053]]
0.0,6.0 = [[631, 579], [772, 766]]
1.0,6.0 = [[579, 506], [766, 627]]
2.0,6.0 = [[506, 483], [627, 529]]
3.0,6.0 = [[483, 556], [529, 473]]
4.0,6.0 = [[556, 686], [473, 556]]
5.0,6.0 = [[686, 825], [556, 669]]
6.0,6.0 = [[825, 1004], [669, 810]]
7.0,6.0 = [[1004, 1053], [810, 917]]
0.0,7.0 = [[772, 766], [900, 880]]
1.0,7.0 = [[766, 627], [880, 778]]
2.0,7.0 = [[627, 529], [778, 671]]
3.0,7.0 = [[529, 473], [671, 540]]
4.0,7.0 = [[473, 556], [540, 483]]
5.0,7.0 = [[556, 669], [483, 536]]
6.0,7.0 = [[669, 810], [536, 641]]
```

Set values on a Raster

```
File file = new File("src/main/resources/earth.tif")
GeoTIFF geotiff = new GeoTIFF(file)
Raster raster = geotiff.read("earth")

File arcGridFile = new File("target/earth.asc")
ArcGrid arcGrid = new ArcGrid(arcGridFile)
arcGrid.write(raster)
Raster arcGridRaster = arcGrid.read("earth")

arcGridRaster.eachCell {double value, double x, double y ->
    double newValue = value + 100
    arcGridRaster.setValue([x as int, y as int], newValue)
}

File arcGridAddFile = new File("target/earth_100.asc")
ArcGrid arcGridAdd = new ArcGrid(arcGridAddFile)
arcGridAdd.write(arcGridRaster)
Raster arcGridRasterAdd = arcGridAdd.read("earth_100")

List pixels = [
    [92, 298],
    [393.0, 343.0],
    [795.0, 399.0]
]
pixels.each { List pixel ->
    println "Original: ${raster.getValue(pixel)} New:
${arcGridRasterAdd.getValue(pixel)}"
}
```

```
Original: 97.0 New: 197.0
Original: 96.0 New: 196.0
Original: 237.0 New: 337.0
```

Raster Processing

Crop

Crop a Raster with a Bounds

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")
Raster croppedRaster = raster.crop(new Bounds(-160.927734,6.751896,-34.716797
,57.279043, "EPSG:4326"))
```

Project

Reproject a Raster to another Projection

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")
Projection projection = new Projection("EPSG:3857")
Raster projectedRaster = raster.crop(projection.geoBounds).reproject(projection)
```

Transform

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
```

Scale a Raster

```
Raster scaledRaster = raster.transform(scalex: 10, scaley: 10)
```

Shear a Raster

```
Raster shearedRaster = raster.transform(shearx: 10, sheary: 10)
```

Translate a Raster

```
Raster translatedRaster = raster.transform(translatex: 10.1, translatey: 12.6)
```

Transform a Raster with a combination of parameters

```
Raster transformedRaster = raster.transform(
    scalex: 1.1, scaley: 2.1,
    shearx: 0.4, sheary: 0.3,
    translatex: 10.1, translatey: 12.6,
    nodata: [-255],
    interpolation: "NEAREST"
)
```

Select Bands

Extract a band from a Raster to create a new Raster

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")

Raster band1 = raster.selectBands([0])
Raster band2 = raster.selectBands([1])
Raster band3 = raster.selectBands([2])
```

Band 1

[raster selectband r] | *raster_selectband_r.png*

Band 2

[raster selectband g] | *raster_selectband_g.png*

Band 3

[raster selectband b] | *raster_selectband_b.png*

Merge

Merge a List of Rasters representing different bands together to create a single Raster

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")

Raster band1 = raster.selectBands([0])
Raster band2 = raster.selectBands([1])
Raster band3 = raster.selectBands([2])

Raster mergedRaster = Raster.merge([band1,band2,band3], transform: "FIRST")
```

Mosaic

Mosaic a List of Rasters together to create a single Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")

Bounds bounds = raster.bounds
List<Raster> rasters = bounds.tile(0.5).collect { Bounds b ->
    raster.crop(b)
}

Raster mosaicedRaster = Raster.mosaic(rasters)
```

Contours

Create vector contours from a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
int band = 0
int interval = 300
boolean simplify = true
boolean smooth = true
Layer contours = raster.contours(band, interval, simplify, smooth)
```

Stylize

Stylize a Raster by baking in a style to create a new Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster stylizedRaster = raster.stylize(new ColorMap([
        [color: "#9fd182", quantity:25],
        [color: "#3e7f3c", quantity:470],
        [color: "#133912", quantity:920],
        [color: "#08306b", quantity:1370],
        [color: "#fffff5", quantity:1820],
]))
```

Shaded Relief

Create a shaded relief Raster from another Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster shadedReliefRaster = raster.createShadedRelief(
    1.0, ①
    25,  ②
    260  ③
)
```

① scale

② altitude

③ azimuth

Reclassify

Reclassify a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster reclassifiedRaster = raster.reclassify([
        [min:0,    max:0,    value: 1],
        [min:0,    max:50,   value: 2],
        [min:50,   max:200,  value: 3],
        [min:200,  max:1000, value: 4],
        [min:1000, max:1500, value: 5],
        [min:1500, max:4000, value: 6]
])
```

Scale

Scale a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
println "Original Raster Size = ${raster.size[0]}x${raster.size[1]}"

Raster scaledRaster = raster.scale(0.5, 0.5)
println "Scaled Raster Size = ${scaledRaster.size[0]}x${scaledRaster.size[1]}"
```

```
Original Raster Size = 800x400
Scaled Raster Size = 400x200
```

Resample

Resample a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
println "Original Raster Bounds = ${raster.bounds}"
println "Original Raster Size = ${raster.size[0]}x${raster.size[1]}"

Raster resampledRaster = raster.resample(size: [400, 400], bbox: raster.bounds.scale(-
2))
println "Resampled Raster Bounds = ${resampledRaster.bounds}"
println "Resampled Raster Size =
${resampledRaster.size[0]}x${resampledRaster.size[1]}"
```

```
Original Raster Bounds = (-123.55291606131708,46.25375026634816,-
120.73958272798374,47.522916933014834,EPSG:4326)
Original Raster Size = 800x400
Resampled Raster Bounds = (-124.95958272798374,45.619166933014824,-
119.33291606131708,48.157500266348165,EPSG:4326)
Resampled Raster Size = 400x400
```

Normalize

Normalize a Raster by diving all values by the maximum value.

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
println "Original Raster Min Max values = ${raster.extrema.min[0]} -
${raster.extrema.max[0]}"

Raster normalizedRaster = raster.normalize()
println "Normalized Raster Min Max values = ${normalizedRaster.extrema.min[0]} -
${normalizedRaster.extrema.max[0]}"
```

```
Original Raster Min Max values = -23.0 - 4370.0
Normalized Raster Min Max values = -0.005263158120214939 - 1.0
```

Convolve

Convolve a Raster with a radius.

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
println "Original Raster Min Max values = ${raster.extrema.min[0]} -
${raster.extrema.max[0]}"

Raster convolvedRaster = raster.convolve(2)
println "Convolved Raster Min Max values = ${convolvedRaster.extrema.min[0]} -
${convolvedRaster.extrema.max[0]}"
```

```
Original Raster Min Max values = -23.0 - 4370.0
Convolved Raster Min Max values = -32767.0 - 32767.0
```

Convolve a Raster with a width and height.

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
println "Original Raster Min Max values = ${raster.extrema.min[0]} -
${raster.extrema.max[0]}"

Raster convolvedRaster = raster.convolve(1,2)
println "Convolved Raster Min Max values = ${convolvedRaster.extrema.min[0]} -
${convolvedRaster.extrema.max[0]}"
```

```
Original Raster Min Max values = -23.0 - 4370.0
Convolved Raster Min Max values = -32767.0 - 8675.0
```

Invert

Invert the values of a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster invertedRaster = raster.invert()
```

Exponent

Calculate the exponent of the values of a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster expRaster = raster.exp()
```

Absolute

Calculate the absolute value of the values of a Raster

```
File file = new File("src/main/resources/absolute.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("absolute")
Raster absolute = raster.absolute()
```

Log

Calculate the log of the values of a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster logRaster = raster.log()
```

Vectorize

Create a Polygon Layer from a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
Raster reclassifiedRaster = raster.reclassify([
        [min:0,    max:0,    value: 1],
        [min:0,    max:50,   value: 2],
        [min:50,   max:200,  value: 3],
        [min:200,  max:1000, value: 4],
        [min:1000, max:1500, value: 5],
        [min:1500, max:4000, value: 6]
])
Layer layer = reclassifiedRaster.polygonLayer
```

Create a Point Layer from a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc").crop(new Bounds(-121.878548,46.808402,-121.636505
,46.896097, "EPSG:4326"))
Layer layer = raster.pointLayer
```

Extract a foot print from a Raster

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")
Layer layer = raster.extractFootPrint()
```

Calculate zonal statistics

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")

Layer zones = new Memory().create("zones", [new Field("geom","Geometry","EPSG:4326")])
Bounds bounds = raster.bounds
bounds.tile(0.5).each{b -> zones.add([b.geometry])}

Layer stats = raster.zonalStatistics(0, zones)
```

count	min	max	sum	avg	stddev
79950	-3.0	1718.0	2.6618944E7	332.94489055659943	262.73459374414483
80000	254.0	4370.0	9.2963902E7	1162.0487749999913	439.0841907966851
71728	-23.0	1755.0	1.1585759E7	161.52351940664724	179.29389227723505
80000	24.0	2728.0	7.9051464E7	988.1433000000056	465.8407188458327

Histogram

Get histogram of the Raster

```
File file = new File("src/main/resources/earth.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("earth")

Histogram histogram = raster.getHistogram()
println "# of bands = ${histogram.numberOfBands}"
println "# Counts = ${histogram.counts().size()}"
println "# Bins = ${histogram.bins().size()}"
println "Count 25 = ${histogram.count(25)}"
println "Bin 45 = ${histogram.bin(45)}"

Chart chart = Bar.xy(histogram.counts(0).withIndex().collect {int count, int index ->
[index, count]})
```

```
# of bands = 3
# Counts = 256
# Bins = 256
Count 25 = 0
Bin 45 = [45.0, 46.0]
```

Raster Algebra

Add

Add a constant value to a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
double elevation1 = raster.getValue(new Point(-121.799927,46.867703))
println elevation1

Raster higherRaster = raster.add(100.00)
double elevation2 = higherRaster.getValue(new Point(-121.799927,46.867703))
println elevation2
```

```
3069.0
3169.0
```

Add two Raster together

```
Raster lowRaster = Format.getFormat(new File("src/main/resources/low.tif")).read(
"low")
Raster highRaster = Format.getFormat(new File("src/main/resources/high.tif")).read
("high")
Raster lowPlusHighRaster = lowRaster.add(highRaster)
```

Low

13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0
1.0	2.0	3.0	4.0

High

17.0	18.0	19.0	20.0
13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0

Low + High

30.0	32.0	34.0	36.0
22.0	24.0	26.0	28.0
14.0	16.0	18.0	20.0
6.0	8.0	10.0	12.0

Subtract

Subtract a constant value from a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
double elevation1 = raster.getValue(new Point(-121.799927,46.867703))
println elevation1

Raster lowerRaster = raster.minus(50.00)
double elevation2 = lowerRaster.getValue(new Point(-121.799927,46.867703))
println elevation2
```

```
3069.0
3019.0
```

Subtract the Raster from a constant value

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
double elevation1 = raster.getValue(new Point(-121.799927,46.867703))
println elevation1

Raster lowerRaster = raster.minusFrom(2000.0)
double elevation2 = lowerRaster.getValue(new Point(-121.799927,46.867703))
println elevation2
```

```
3069.0
-1069.0
```

Subtract a Raster from another Raster

```
Raster lowRaster = Format.getFormat(new File("src/main/resources/low.tif")).read(
"low")
Raster highRaster = Format.getFormat(new File("src/main/resources/high.tif")).read
("high")
Raster highMinusLowRaster = highRaster.minus(lowRaster)
```

Low

13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0
1.0	2.0	3.0	4.0

High

17.0	18.0	19.0	20.0
13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0

High - Low

4.0

Multiply

Multiply a constant value against a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
double elevation1 = raster.getValue(new Point(-121.799927,46.867703))
println elevation1

Raster higherRaster = raster.multiply(2.0)
double elevation2 = higherRaster.getValue(new Point(-121.799927,46.867703))
println elevation2
```

```
3069.0
6138.0
```

Multiply a Raster with another Raster

```
Raster lowRaster = Format.getFormat(new File("src/main/resources/low.tif")).read(
"low")
Raster highRaster = Format.getFormat(new File("src/main/resources/high.tif")).read
("high")
Raster multiplyRaster = highRaster.multiply(lowRaster)
```

Low

13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0
1.0	2.0	3.0	4.0

High

17.0	18.0	19.0	20.0
13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0

High * Low

221.0	252.0	285.0	320.0
117.0	140.0	165.0	192.0
45.0	60.0	77.0	96.0
5.0	12.0	21.0	32.0

Divide

Divide a constant value against a Raster

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
double elevation1 = raster.getValue(new Point(-121.799927,46.867703))
println elevation1

Raster lowerRaster = raster.divide(2.0)
double elevation2 = lowerRaster.getValue(new Point(-121.799927,46.867703))
println elevation2
```

```
3069.0
1534.5
```

Divide a Raster by another Raster

```
Raster lowRaster = Format.getFormat(new File("src/main/resources/low.tif")).read(
"low")
Raster highRaster = Format.getFormat(new File("src/main/resources/high.tif")).read
("high")
Raster divideRaster = highRaster.divide(lowRaster)
```

Low

13.00	14.00	15.00	16.00
9.00	10.00	11.00	12.00
5.00	6.00	7.00	8.00
1.00	2.00	3.00	4.00

High

17.00	18.00	19.00	20.00
13.00	14.00	15.00	16.00
9.00	10.00	11.00	12.00
5.00	6.00	7.00	8.00

High / Low

1.31	1.29	1.27	1.25
1.44	1.40	1.36	1.33
1.80	1.67	1.57	1.50
5.00	3.00	2.33	2.00

World File

Create a world file from a Bounds and size.

```
File file = new File("target/worldfile.txt")
WorldFile worldFile = new WorldFile(new Bounds(-123.06, 46.66, -121.15, 47.48), [500
,500], file)
println "Pixel Size = ${worldFile.pixelSize[0]} x ${worldFile.pixelSize[1]}"
println "Rotation = ${worldFile.rotation[0]} x ${worldFile.rotation[1]}"
println "Upper Left Coordinate = ${worldFile.ulc.x}, ${worldFile.ulc.y}"
println "File = ${file.text}"
```

```
Pixel Size = 0.003819999999999993 x -0.0016400000000000006
Rotation = 0.0 x 0.0
Upper Left Coordinate = -123.05809, 47.47918
File = 0.003819999999999993
0.0
0.0
-0.0016400000000000006
-123.05809
47.47918
```

Create a world file from an existing file.

```
File file = new File("src/main/resources/worldfile.txt")
WorldFile worldFile = new WorldFile(file)
println "Pixel Size = ${worldFile.pixelSize[0]} x ${worldFile.pixelSize[1]}"
println "Rotation = ${worldFile.rotation[0]} x ${worldFile.rotation[1]}"
println "Upper Left Coordinate = ${worldFile.ulc.x}, ${worldFile.ulc.y}"
```

```
Pixel Size = 0.003819999999999993 x -0.0016400000000000006
Rotation = 0.0 x 0.0
Upper Left Coordinate = -123.05809, 47.47918
```

Map Algebra

GeoScript uses Jiffle to perform map or raster algebra.

Add two Rasters together

```
Raster lowRaster = Format.getFormat(new File("src/main/resources/low.tif")).read(
"low")
Raster highRaster = Format.getFormat(new File("src/main/resources/high.tif")).read
("high")

MapAlgebra mapAlgebra = new MapAlgebra()
Raster output = mapAlgebra.calculate("dest = raster1 + raster2;", [raster1: lowRaster,
raster2: highRaster], size: [300, 200])
```

Low

13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0
1.0	2.0	3.0	4.0

High

17.0	18.0	19.0	20.0
13.0	14.0	15.0	16.0
9.0	10.0	11.0	12.0
5.0	6.0	7.0	8.0

High + Low

30.0	32.0	34.0	36.0
22.0	24.0	26.0	28.0
14.0	16.0	18.0	20.0
6.0	8.0	10.0	12.0

Generate a wave Raster

```
MapAlgebra algebra = new MapAlgebra()

String script = """
    init {
      // image centre coordinates
      xc = width() / 2;
      yc = height() / 2;

      // constant term
      C = M_PI * 8;
    }

    dx = (x() - xc) / xc;
    dy = (y() - yc) / yc;
    d = sqrt(dx*dx + dy*dy);

    destImg = sin(C * d);
"""

Raster output = algebra.calculate(script, [:], outputName: "destImg")
```

Create a Raster of all cells greater than a given value

```
File file = new File("src/main/resources/pc.tif")
Format format = Format.getFormat(file)
Raster raster = format.read("pc")
println raster.size
MapAlgebra mapAlgebra = new MapAlgebra()
Raster output = mapAlgebra.calculate("dest = src > 1100;", [src: raster], size: [800,
400])
println output.extrema
```