

---

# **Telluric Documentation**

***Release v0.13.0***

**Juan Luis Cano, Slava Kerner, Lucio Torre**

**Nov 25, 2020**



**CONTENTS:**

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	User Guide . . . . .	3
1.2	API Reference . . . . .	9
1.3	Changelog . . . . .	9
<b>2</b>	<b>Indices and tables</b>	<b>19</b>



telluric is a Python library to manage vector and raster geospatial data in an interactive and easy way.

The [source code](#) and [issue tracker](#) are hosted on GitHub, and all contributions and feedback are more than welcome. There is a [public chat](#) for users and developers too.



## INSTALLATION

You can install telluric using pip:

```
pip install telluric[vis]
```

telluric is a pure Python library, and therefore should work on Linux, OS X and Windows provided that you can install its dependencies. If you find any problem, [please open an issue](#) and we will take care of it.

**Warning:** It is recommended that you **never ever use sudo** with pip because you might seriously break your system. Use [venv](#), [Pipenv](#), [pyenv](#) or [conda](#) to create an isolated development environment instead.

## 1.1 User Guide

### 1.1.1 Geometries on a map: GeoVector

```
[1]: import telluric as tl
from telluric.constants import WGS84_CRS, WEB_MERCATOR_CRS
```

The simplest geometrical element in telluric is the **GeoVector**: it represents a shape in some coordinate reference system (CRS). The easiest way to create one is to use the `GeoVector.from_bounds` method:

```
[2]: gv1 = tl.GeoVector.from_bounds(
    xmin=0, ymin=40, xmax=1, ymax=41, crs=WGS84_CRS
)
print(gv1)

GeoVector(shape=POLYGON ((0 40, 0 41, 1 41, 1 40, 0 40)), crs=CRS({'init': 'epsg:4326
↪'}))
```

If we print the object, we see its two defining elements: a shape (actually a shapely `BaseGeometry` object) and a CRS (in this case WGS84 or <http://epsg.io/4326>). Rather than reading a dull representation, we can directly visualize it in the notebook:

```
[3]: gv1

/home/juanlu/SatelloLogic/telluric/telluric/plotting.py:141: UserWarning: Plotting a_
↪limited representation of the data, use the .plot() method for further customization
    "Plotting a limited representation of the data, use the .plot() method for further_
↪customization")
```

[3]:

You can ignore the warning for the moment. Advanced plotting techniques are not yet covered in this User Guide.

---

As you can see, we have an interactive Web Mercator map where we can display our shape. We can create more complex objects using the [Shapely](#) library:

[4]: `from shapely.geometry import Polygon`

```
gv2 = tl.GeoVector(
    Polygon([(0, 40), (1, 40.1), (1, 41), (-0.5, 40.5), (0, 40)]),
    WGS84_CRS
)
```

```
print(gv2)
GeoVector(shape=POLYGON ((0 40, 1 40.1, 1 41, -0.5 40.5, 0 40)), crs=CRS({'init':
↪ 'epsg:4326'}))
```

And we can access any property of the underlying geometry using the same attribute name:

[5]: `print(gv1.centroid)`

```
GeoVector(shape=POINT (0.5 40.5), crs=CRS({'init': 'epsg:4326'}))
```

[6]: `gv1.area` *# Real area in square meters*

[6]: 9422706289.175217

[7]: `gv1.is_valid`

[7]: True

[8]: `gv1.within(gv2)`

[8]: False

[9]: `gv1.difference(gv2)`

```
/home/juanlu/Satellogic/telluric/telluric/plotting.py:141: UserWarning: Plotting a
↪ limited representation of the data, use the .plot() method for further customization
    "Plotting a limited representation of the data, use the .plot() method for further
↪ customization")
```

[9]:

## 1.1.2 Geometries with attributes: `GeoFeature` and `FeatureCollection`

The next object in the telluric hierarchy is the `GeoFeature`: a combination of a `GeoVector` + some attributes. These attributes can represent land use, types of buildings, and so forth.

```
[10]: gf1 = tl.GeoFeature(
    gv1,
    {'name': 'One feature'}
)
gf2 = tl.GeoFeature(
    gv2,
```

(continues on next page)



(continued from previous page)

```

    {'name': 'Another feature'}
)
print(gf1)
print(gf2)

GeoFeature(Polygon, {'name': 'One feature'})
GeoFeature(Polygon, {'name': 'Another feature'})

```

But the most interesting thing is to combine these features into a [FeatureCollection](#). A `FeatureCollection` is essentially a sequence of features, with some additional methods:

```

[11]: fc = tl.FeatureCollection([gf1, gf2])
      fc

/home/juanlu/Satellogic/telluric/telluric/plotting.py:141: UserWarning: Plotting a
↳ limited representation of the data, use the .plot() method for further customization
    "Plotting a limited representation of the data, use the .plot() method for further
↳ customization")
[11]: <telluric.collections.FeatureCollection at 0x7f283ea41f60>

```

```

[12]: print(fc.convex_hull)

GeoVector(shape=POLYGON ((0 40, -0.5 40.5, 0 41, 1 41, 1 40, 0 40)), crs=CRS({'init':
↳ 'epsg:4326'}))

```

```

[13]: print(fc.envelope)

GeoVector(shape=POLYGON ((-0.5 40, 1 40, 1 41, -0.5 41, -0.5 40)), crs=CRS({'init':
↳ 'epsg:4326'}))

```

### 1.1.3 Input and Output

Apart from all the previous geospatial operations, we can also save these `FeatureCollection` objects to disk, for example using the GeoJSON or ESRI Shapefile formats:

```

[14]: fc.save("test_fc.shp")

[15]: !ls test_fc*

test_fc.cpg  test_fc.dbf  test_fc.json          test_fc.prj  test_fc.shp  test_fc.shx

[16]: fc.save("test_fc.json")

[17]: !python -m json.tool < test_fc.json | head -n28
{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
    }
  },
  "features": [
    {

```

(continues on next page)

(continued from previous page)

```

    "type": "Feature",
    "properties": {
        "name": "One feature",
        "highlight": {},
        "style": {}
    },
    "geometry": {
        "type": "Polygon",
        "coordinates": [
            [
                [
                    0.0,
                    40.0
                ],
                [
                    0.0,
                    41.0
                ],
            ]
        ]
    }
}

```

To retrieve this data from disk again, we can use another object, `FileCollection`, which behaves in the same way as a `FeatureCollection` but does some smart optimizations so the files are not read completely into memory:

```

[18]: print(list(tl.FileCollection.open("test_fc.shp")))

[GeoFeature(Polygon, {'name': 'One feature', 'highlight': '{}', 'style': '{}'}),
↪ GeoFeature(Polygon, {'name': 'Another feature', 'highlight': '{}', 'style': '{}'})]

```

### 1.1.4 Raster data: GeoRaster2

After reviewing how to read, manipulate and write vector data, we can use `GeoRaster2` to do the same thing with raster data. `GeoRaster2` will read the raster lazily so we only retrieve the information that we need.

```

[19]: # This will only save the URL in memory
rs = tl.GeoRaster2.open(
    "https://github.com/mapbox/rasterio/raw/master/tests/data/rgb_deflate.tif"
)

# These calls will fetch some GeoTIFF metadata
# without reading the whole image
print(rs.crs)
print(rs.footprint())
print(rs.band_names)

CRS({'init': 'epsg:32618'})
GeoVector(shape=POLYGON ((101984.9999999127 2826915, 339314.9999997905 2826915,
↪ 339314.9999998778 2611485, 101985.0000002096 2611485, 101984.9999999127 2826915)),
↪ crs=CRS({'init': 'epsg:32618'}))
[0, 1, 2]

```

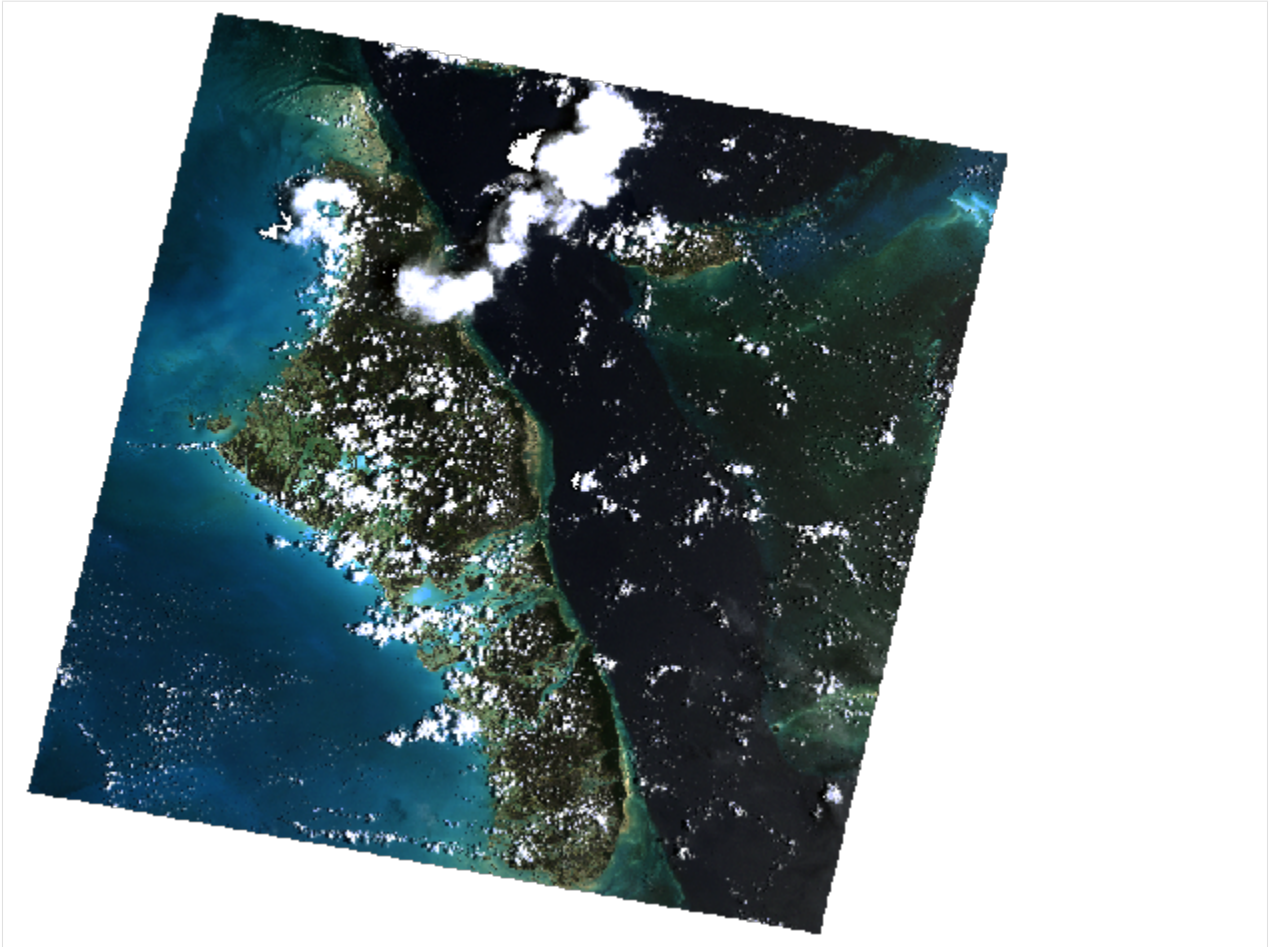
`GeoRaster2` also displays itself automatically:

```

[20]: rs

```

[20]:



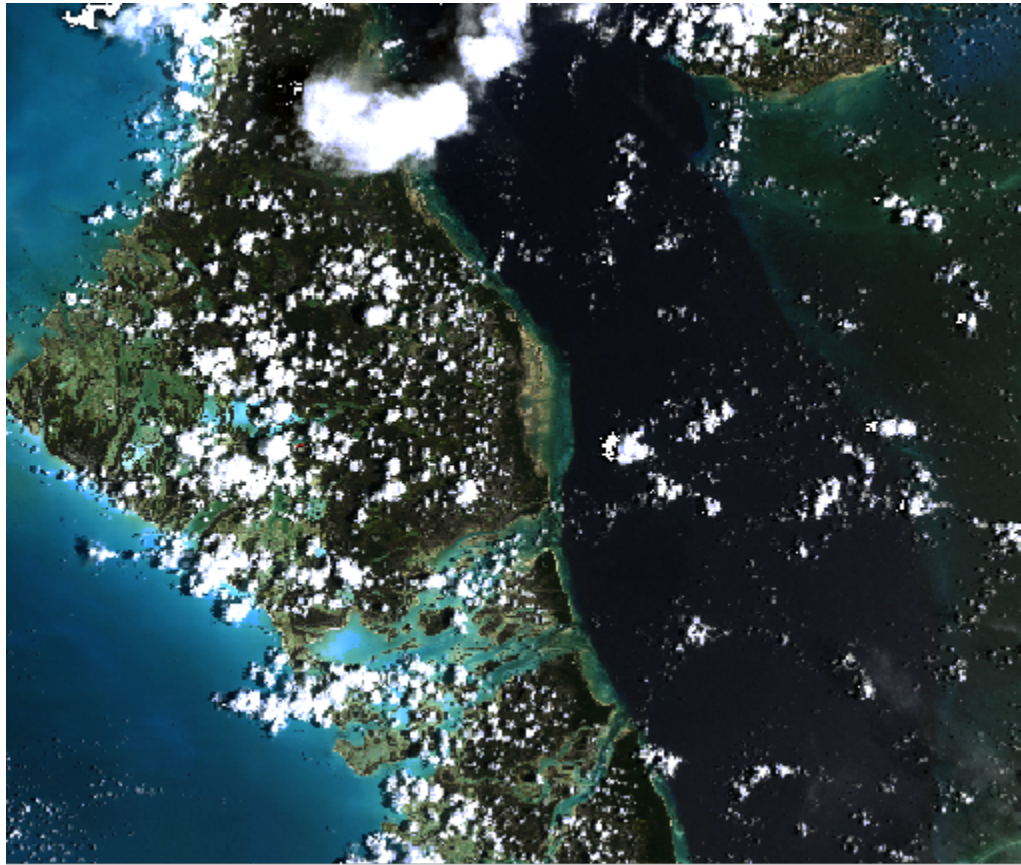
We can slice it like an array, or cropping some parts to discard others:

```
[21]: rs.shape
```

```
[21]: (3, 718, 791)
```

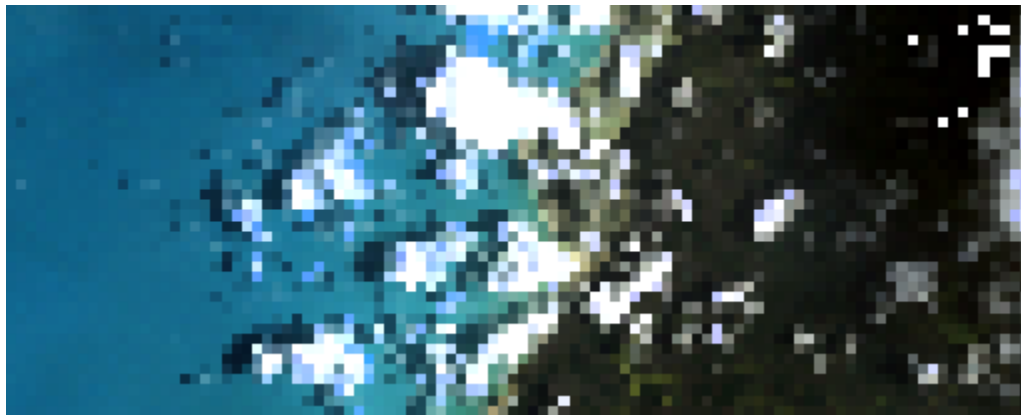
```
[22]: rs.crop(rs.footprint().buffer(-50000))
```

[22]:



```
[23]: rs[200:300, 200:240]
```

[23]:



And save again to GeoTIFF format using a variety of options:

```
[24]: rs[200:300, 200:240].save("test_raster.tif")
```

## 1.1.5 Conclusion

There are many things not covered in this User Guide. The documentation of telluric is a work in progress, so we encourage you to [read the full API reference](#) and even [contribute to the package](#)!

## 1.2 API Reference

### 1.2.1 telluric.constants module

### 1.2.2 telluric.vectors module

### 1.2.3 telluric.features module

### 1.2.4 telluric.collections module

### 1.2.5 telluric.georaster module

### 1.2.6 telluric.plotting module

### 1.2.7 telluric.util package

## 1.3 Changelog

### 1.3.1 telluric 0.13.0 (2020-11-25)

#### Changes

- Make visualization dependencies optional (#260)

### 1.3.2 telluric 0.12.1 (2020-08-10)

#### Bug fixes

- Check if the raster's footprint intersects the tile's footprint in `telluric.georaster.GeoRaster2.get_tile()` (#273)

### 1.3.3 telluric 0.12.0 (2020-08-02)

#### New features

- Preserve nodata value while saving rasters (#271)
- FileCollection created out of file-like object can be iterated (#272)

### 1.3.4 telluric 0.11.1 (2020-06-27)

#### Bug fixes

- Fix `telluric.collections.FileCollection.sort()` (#259)
- Fix potential bug in `ThreadContext` when it is uninitialized (#259)
- Disable transformation if source CRS equals to destination (#270)

### 1.3.5 telluric 0.11.0 (2019-12-02)

#### New features

- Now `MutableGeoRaster` inherits `nodata_value`

### 1.3.6 telluric 0.10.8 (2019-08-30)

#### Bug fixes

- Now reprojection retains nodata values

### 1.3.7 telluric 0.10.7 (2019-06-06)

#### New features

- Adding support of resources accessed through HTTP and HTTPS to VRT (#248)

#### Big fixes

- Remove unnecessary call of `fiona.Env` (#247)

### 1.3.8 telluric 0.10.6 (2019-05-02)

#### New features

- Creating COG with internal mask (#244)
- Removed pinning for `pyproj` (#245)

### 1.3.9 telluric 0.10.5 (2019-04-08)

#### Bug fixes

- Workaround to overcome impossible transformations (#241)

### 1.3.10 telluric 0.10.4 (2019-03-17)

#### Bug fixes

- Prevent image loading while copying (#235)

#### New features

- Refactored raster join implementation (#230)
- Changed default value of “nodata” in `GeoRaster2` constructor, now it is `None` (#231)
- Accelerate tests (#232)
- Added new method `telluric.georaster.GeoRaster2.mask_by_value()` (#233)
- Added new method `telluric.vectors.GeoVector.from_record()` (#238)
- Rasterio 1.0.21 compatibility (#239)
- Adding support to lazy resize that can use overviews if exist (#240)

### 1.3.11 telluric 0.10.3 (2019-01-10)

#### Bug fixes

- Fix `FeatureCollection` plotting (#229)

### 1.3.12 telluric 0.10.2 (2019-01-10)

#### New features

- SpatioTemporal Asset Catalog (STAC) compatibility (#223)
- Support custom schema in `telluric.collections.BaseCollection.save()` (#224)

#### Bug fixes

- Preserve the original schema while using `telluric.collections.BaseCollection.apply()` and `telluric.collections.BaseCollection.groupby()` (#225)
- Better handling of an empty collections (#226)
- Remove the reference to the raster object in the asset entry (#227)
- Retrieve mask in a safer way to avoid shrunk masks (#228)

### 1.3.13 telluric 0.10.1 (2018-12-27)

#### Bug fixes

- Fix masking by `GeoFeature` (#216)
- Fix issue in `GeoRaster.from_asset()` (#217, #220)
- `telluric.features.GeoFeature.envelope()` returns instance of `GeoVector` (#218)
- Use local tile server for visualization of `GeoFeatureWithRaster` (#221)
- `telluric.georaster.GeoRaster2.mask()` uses crop internally to reduce memory footprint (#219)
- `telluric.georaster.GeoRaster2.limit_to_bands()` is lazy (#222)

### 1.3.14 telluric 0.10.0 (2018-12-21)

#### New features

- Fiona 1.8.4 and Rasterio 1.0.13 compatibility (#207, #208)
- Support multiple rasters in a single `GeoFeatureWithRaster` (#209)
- Added new method `telluric.vectors.GeoVector.get_bounding_box()` (#213)

#### Bug fixes

- Remove hardcoded tile server port (#205)
- The internal state of the raster is not changed while saving (#210)
- Fix `telluric.georaster.GeoRaster2.save()` (#211)
- Fix bug in reproject (#212)
- Better handling of `telluric.features.GeoFeature.from_record()` (#214)

### 1.3.15 telluric 0.9.1 (2018-12-14)

#### New features

- LZW compression is used by default for creating COG rasters (#200)
- Added way to change port for local tile server (#202)

#### Bug fixes

- Fix iterating over `FileCollection` (#203)
- Fix fiona's GDAL environment issue (#204)



### 1.3.16 telluric 0.9.0 (2018-12-12)

#### New features

- Added new method `telluric.collections.FeatureCollection.from_georasters()` to create collections of rasters (#184)
- Visualization feature collection with rasters in Jupyter Notebook (#186)
- Added new method `telluric.collections.BaseCollection.apply()` (#188)
- Added new method `telluric.georaster.GeoRaster2.from_wms()` for creating rasters out of web services (#190, #192)
- Generalizing the process of making VRT files (#191, #193)
- Rasterio 1.0.11 compatibility (#194)
- Added new method `telluric.georaster.GeoRaster2.from_rasters()` to create raster out of a list of rasters (#195)
- Added support of several domains in a single VRT file (#196)

#### Bug fixes

- Reproject features before polygonization (#182)
- Fix `matplotlib.cm` call (#187)
- Fix `telluric.georaster.GeoRaster2.save()` (#197)
- Pin minimal version of Folium (#198)
- Fix rasterio's GDAL environment issue (#201)

### 1.3.17 telluric 0.8.0 (2018-11-18)

#### New features

- Interactive representation of rasters in Jupyter Notebook (#178)
- Fiona 1.8.1 and Rasterio 1.0.10 compatibility (#179, #180)

### 1.3.18 telluric 0.7.1 (2018-11-12)

#### Bug fixes

- Removed `pyplot` import from the module level to overcome issues at headless environments (#177)

### 1.3.19 telluric 0.7.0 (2018-11-06)

#### New features

- Added new method `telluric.georaster.GeoRaster2.chunks()` for iterating over the chunks of the raster (#169)

#### Bug fixes

- Workaround to overcome fiona's GDAL environment issue (#175)

### 1.3.20 telluric 0.6.0 (2018-11-05)

#### New features

- Added resampling parameter to `telluric.georaster.merge_all()` function (#166)
- New `telluric.vectors.GeoVector.tiles()` method for iterating over the tiles intersecting the bounding box of the vector (#167)
- Fiona 1.8.0 compatibility (#171)

#### Bug fixes

- Workaround to overcome rasterio's GDAL environment issue (#174)

### 1.3.21 telluric 0.5.0 (2018-10-26)

#### New features

- A new class `MutableGeoRaster` was added (#165)

### 1.3.22 telluric 0.4.1 (2018-10-23)

#### Bug fixes

- The right way to calculate `dest_resolution` in `telluric.georaster.merge_all()` if one is not provided (#163)
- Read mask only if it exists (#164)

### 1.3.23 telluric 0.4.0 (2018-10-19)

#### New features

- Rasterio 1.0.3 and higher compatibility (#152)
- Non-georeferenced images may be opened by providing affine and crs parameters to `telluric.georaster.GeoRaster2.open()` (#153)
- A new argument `crs` was added to `telluric.collections.FileCollection.open()` for opening vector files that don't contain information about CRS (#156)
- A new `telluric.util.raster_utils.build_overviews()` utility was added (#158)

#### Bug fixes

- Treat 0 as legitimate value in `telluric.georaster.GeoRaster2.colorize()` (#160)
- Fix rasterization of an empty collection with callable `fill_value` (#161)

### 1.3.24 telluric 0.3.0 (2018-09-20)

#### New features

- New class `GeoFeatureWithRaster` that extends `GeoFeature`.

### 1.3.25 telluric 0.2.1 (2018-09-12)

#### Bug fixes

- Retrieve mask in a safer way in `telluric.georaster.GeoRaster2.save()` (#136)
- Fix affine calculation in `telluric.georaster.GeoRaster2.get_tile()` (#137)
- Convert dimensions to ints (#140)
- Masking areas outside the window in `telluric.georaster.GeoRaster2.get_window()` (#141)
- `telluric.georaster.merge_all()` does not crash for resolution in ROI units (#143, #146)
- Limit rasterio version to <1.0.3
- Add LICENSE into the MANIFEST (#147)

### 1.3.26 telluric 0.2.0 (2018-08-22)

#### New features

- Slicing a `FeatureCollection` now returns a `FeatureCollection` (#29, #32)
- Rasterization methods can now accept multiple fill values to produce nonbinary images (#34)
- `telluric.collections.FileCollection.save()` now saves types better (#20, #36)
- Merging functions and `telluric.georaster.GeoRaster2.empty_from_roi()` now support more ways to define the raster extent (#39, #57)

- Added utilities to convert to Cloud Optimized GeoTIFF (COG) and reproject files on disk (#45, #87)
- Raster data can be converted from/to different floating point formats thanks to enhancements in `telluric.georaster.GeoRaster2.astype()` (#33, #66)
- Added new method `telluric.georaster.GeoRaster2.colorize()` to colorize a band of a raster for visualization purposes (#81)
- Collections now have experimental “groupby/dissolve” functionality inspired by pandas and GeoPandas (#77, #98)
- Add a `telluric.georaster.PixelStrategy` enum with a new mode that allows the user to produce the “metadata” of a merge process (#68, #91)
- `telluric.vectors.GeoVector.rasterize()` can now accept a custom output CRS (#125)
- A new argument was added to the `GeoVector` constructor for disabling arguments validity checking (#126)
- Unnecessary CRS equality checking in `telluric.vectors.GeoVector.get_shape()` was removed for performance reasons (#127)

### Deprecations and removals

- Rasterization methods no longer support specifying a “nodata” value, and an appropriate nodata value will be generated depending on the fill value(s) (#28, #34)
- Properties in the sense of the GeoJSON standard are now called “properties” instead of “attributes” for consistency (#84)
- Non georeferenced raster data is no longer supported (although we are considering re adding it under some restrictions) (#64, #74)
- It is not required for collections to be reprojected to output CRS for rasterization with *fill\_value* (#125)

### Bug fixes

- `telluric.vectors.GeoVector.from_record()` now treats `None` values properly (#37, #38)
- `GeoRaster2` methods and functions work with non isotropic resolution (#39)
- Cropping now behaves correctly with rasterio 1.0.0 (#44, #46)
- Crop size is now correctly computed for rasters in WGS84 (#61, #62)
- Fix rasterio 1.0.0 warnings regarding CRS comparison (#64, #74)
- `telluric.georaster.merge_all()` now is order independent and produces consistent results in all situations (#65, #62)
- `GeoRaster2` methods and functions work with rasters with positive y scale (#76, #78)
- `telluric.georaster.GeoRaster2.save()` with default arguments does not crash for small rasters anymore (#16, #53)
- `telluric.collections.FileCollection.save()` does not have side effects on heterogeneous collections anymore (#19, #24)
- Fix rasterization of points with default arguments (#9)

### 1.3.27 telluric 0.1.0 (2018-04-21)

Initial release



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`