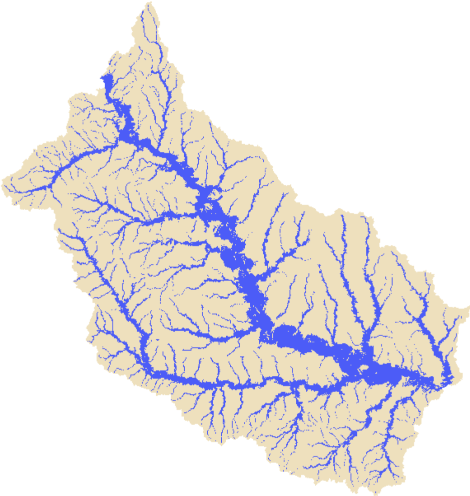
An Open-Source Software Solution for Repeatable and Interpretable Geospatial Evaluations

G. Petrochenkov (Lynker), F. Aristizabal (Earth Resources Technology),
F. Salas (NOAA OWP)

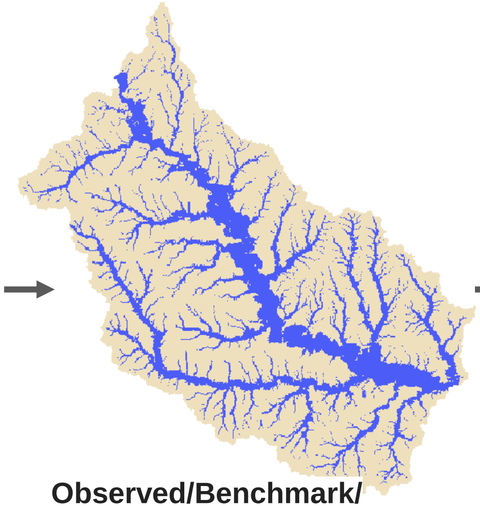
Problem

Evaluating Geospatial Datasets



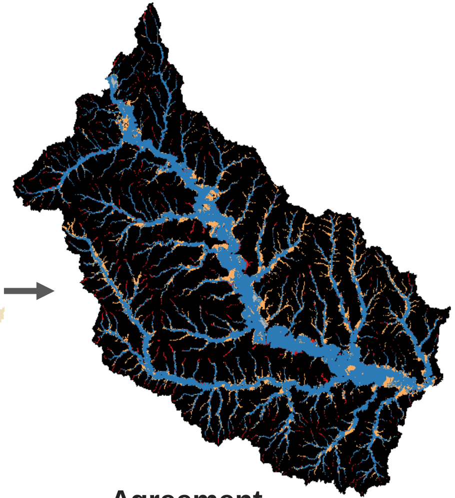
Modeled/Simulated

→



Observed/Benchmark/ Validation

→



Agreement

→

Critical Success Index

Root Mean Square Error

Equitable Threat Score

Mean Absolute Percentage Error

F1-Score

Many more …

Agreement Metrics

Problem:

<u>Geospatial ***Variable*** of Interest</u>
inundation extents, depth, velocity, precipitation, water quality, soil moisture, overland roughness, temperature, etc

<u>Data sources</u>
Models: physics-based, empirical, stochastic, coastal, fluvial, etc
Other geospatial data sources

Designing a Solution

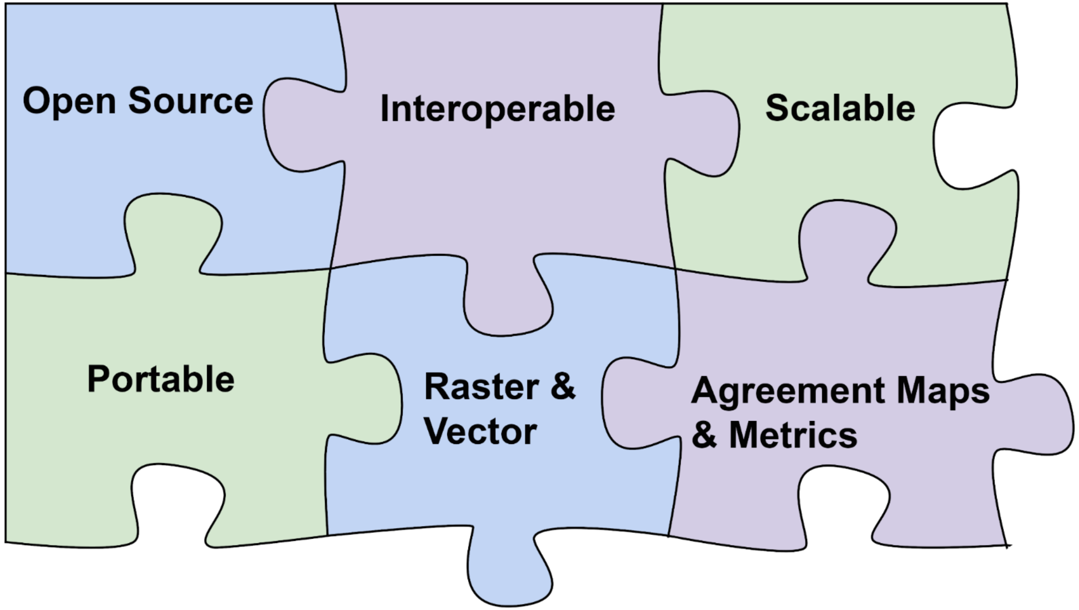
Existing GIS Software and Programming Frameworks

Proprietary	Open Source
ArcGIS, HEXAGON GEOSPATIAL, MapInfo Pro	GRASS GIS, QGIS

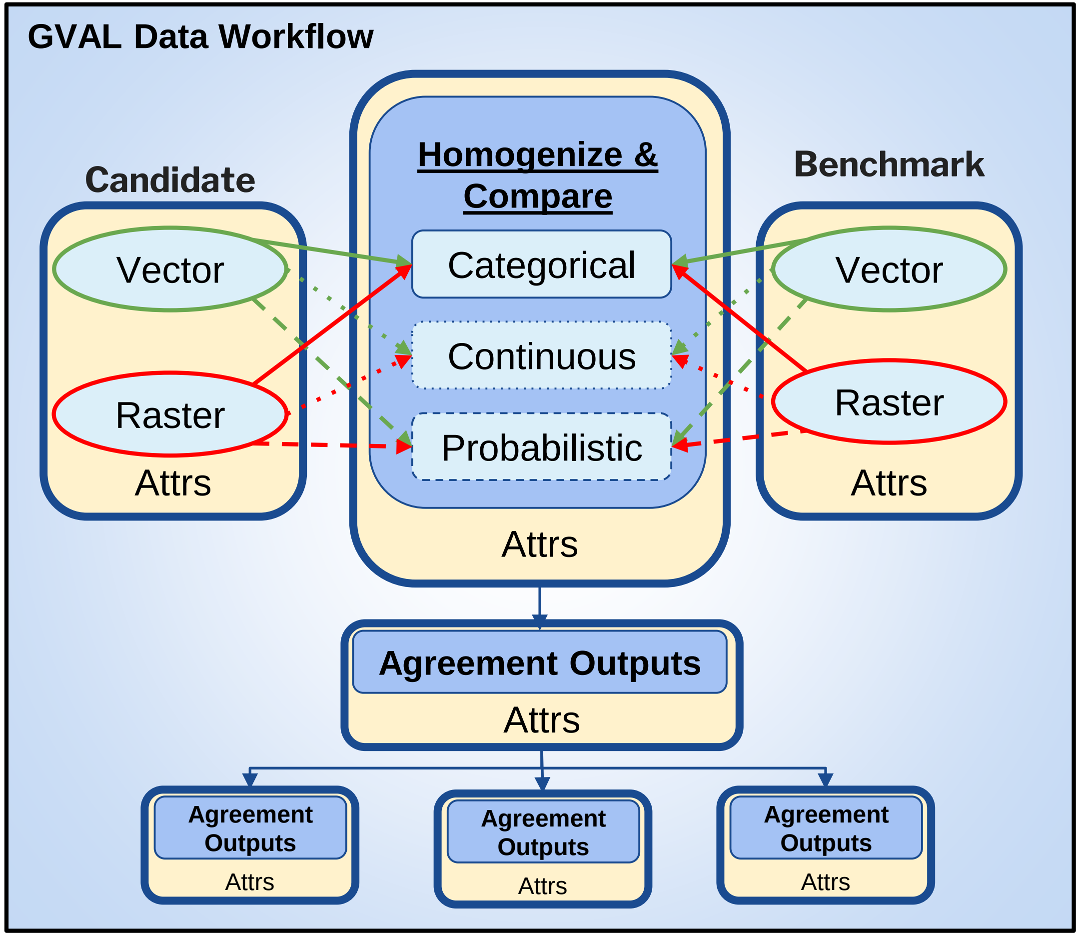
- Comparing data with different discretizations and/or resolutions?
- Large data and scalability?
- Batch processing?
- Capabilities for robust evaluations?

A Modern Approach

To address these problems, we developed a python software package that we call **GVAL**.



General Workflow



Software Design

GVAL
GEOSPATIAL EVALUATIONS

Core Libraries

GVAL uses accessors to run operations on commonly used libraries in the pangeo community.

- Raster operations extend Xarray
- Vector operations extend Pandas/GeoPandas

xarray

pandas

GeoPandas

Currently all processing is done in raster space

Data Sources

- Locally
- Direct AWS S3 Storage
- POSTGIS Service
- STAC Service

PostGIS
Spatial PostgreSQL

Amazon S3

STAC
SpatioTemporal
Asset Catalog

Supported Statistical Data Types

Type	Candidate	Benchmark	Agreement	Example Metrics
Categorical	1 0 0 1 1 0 1 1 1	1 1 1 1 1 0 1 0 0	TP FN FN TP TP TN TP FP FP *TP = True Positive* *TN = True Negative* *FP = False Positive* *FN = False Negative*	* True Positive Rate * False Discovery Rate * Accuracy * F-Score * Critical Success Index * Equitable Threat Score
Continuous	5.0 4.1 3.3 5.0 4.4 8.6 5.0 3.2 2.7	5.1 4.2 8.5 5.0 4.4 4.5 4.8 2.7 2.7	0.1 0.1 5.2 0.0 0.0 4.1 0.2 0.5 0.0	* Mean Absolute Error * Mean Squared Error * Mean Percentage Error * Root Mean Squared Error * Coefficient of Determination
Probabilistic	Ensemble 1 & 2 5.0 8.6 8.6 | 5.0 4.1 3.3 5.1 4.4 8.6 | 5.0 4.4 8.6 5.0 3.2 2.7 | 5.0 3.2 2.7	5.1 4.2 8.5 5.0 4.4 4.5 4.8 2.7 2.7	0.1 2.2 2.6 .05 0.0 4.1 0.2 0.5 0.0	* Receiver Operating Characteristic * Brier Score * Continuous Ranked Probability Score * Discrimination * Reliability

Using GVAL

Categorical Comparisons

GVAL
GEOSPATIAL EVALUATIONS

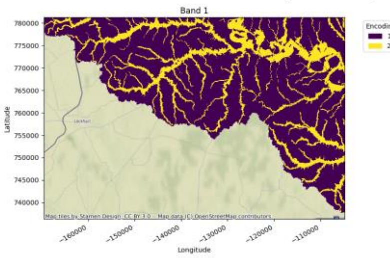
Two-class Categorical Comparisons

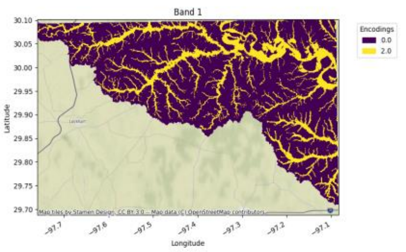
GVAL categorical evaluations can be run with minimal code:

```python
import rioxarray as rxr
import gval

candidate = rxr.open_rasterio(
    'candidate_map_two_class_categorical.tif', mask_and_scale=True
)
benchmark = rxr.open_rasterio(
    'benchmark_map_two_class_categorical.tif', mask_and_scale=True
)

agreement_map, crosstab_table, metric_table = candidate.gval.categorical_compare(
    benchmark,
    positive_categories=[2],
    negative_categories=[0, 1]
)
```

** Input*

Candidate and Benchmark Maps

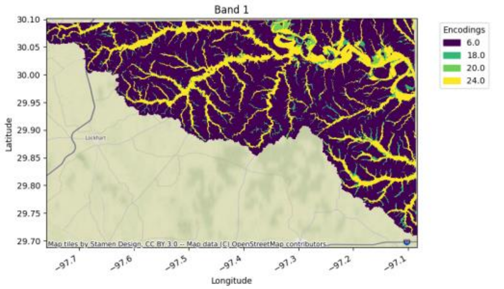




GVAL

** Output*

Agreement Map



Cross Tabulation Table

	band	candidate_values	benchmark_values	agreement_values	counts
0	1	1.0	0.0	6.0	10345720.0
1	1	2.0	0.0	20.0	512277.0
2	1	1.0	2.0	18.0	639227.0
3	1	2.0	2.0	24.0	2473405.0

Metric Table

	band	fn	fp	tn	tp	accuracy	critical_success_index	f_score	false_discovery_rate	false_negative_rate	...
0	1	639227.0	512277.0	10345720.0	2473405.0	0.917577	0.682336	0.811177	0.171578	0.205365	...

Continuous Comparisons

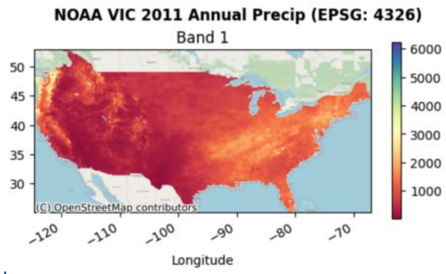
Continuous Comparisons

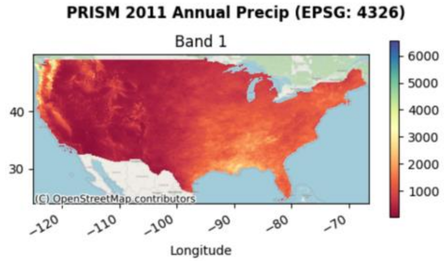
Similarly to categorical evaluations, the following runs continuous evaluations:

```python
import rioxarray as rxr
import gval

candidate = rxr.open_rasterio(
    './livneh_2011_precip.tif', mask_and_scale=True
) # VIC
benchmark = rxr.open_rasterio(
    './prism_2011_precip.tif', mask_and_scale=True
) # PRISM

agreement, metric_table = candidate.gval.continuous_compare(
    benchmark,
    metrics=[
        "coefficient_of_determination",
        "mean_absolute_error",
        "mean_absolute_percentage_error",
        "mean_normalized_mean_absolute_error"
    ]
)
```

** Input*

Candidate and Benchmark Maps

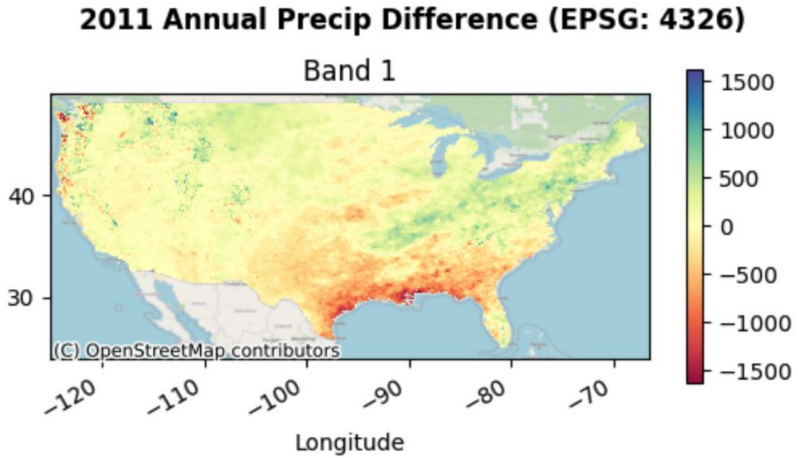




GVAL

** Output*

Agreement Map



Metric Table

	band	coefficient_of_determination	mean_absolute_error	mean_absolute_percentage_error	mean_normalized_mean_absolute_error
0	1	0.685261	216.089706	0.319234	0.267845

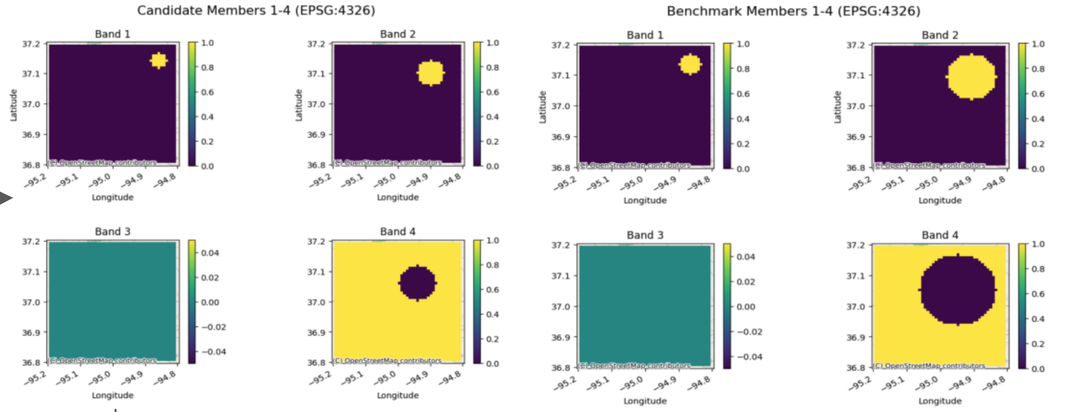
Probabilistic Comparisons

Probabilistic Comparisons

Like the previous two examples, probabilistic comparisons can also be run with minimal code:

```
candidate = rxr.open_rasterio(
    "./candidate_probabilistic.tif", mask_and_scale=True
)
benchmark = rxr.open_rasterio(
    "./benchmark_probabilistic.tif", mask_and_scale=True
)
candidate, benchmark = (
    candidate.rename({"band": "member"}),
    benchmark.rename({"band", "member"})
)

compute_kwargs = {
    "metric_kwargs": {
        "brier_score": {"member_dim": "member", "keep_attrs": True}
    },
    "return_on_error": "error",
}

_, metrics_df = candidate.gval.probabilistic_compare(
        benchmark, **compute_kwargs
)
```

** Input* **Candidate and Benchmark Maps**



GVAL

** Output* **Metric Table**

	band	brier_score
0	member	0.170926

Catalog Comparisons

Catalog Comparisons

GVAL can run evaluations on catalogs of maps:

- A catalog represents multiple maps and in GVAL is represented by a dataframe
- A candidate and benchmark can be compared using identifiers
- This will create metrics for each set of maps as well as agreement maps

** Input*

GVAL

** Output*

Candidate and Benchmark Catalogs

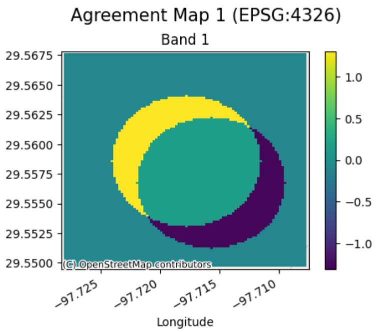
	map_id	compare_id	agreement_maps
0	./candidate_continuous_0.tif	compare1	./agreement_continuous_0.tif
1	./candidate_continuous_1.tif	compare2	./agreement_continuous_1.tif

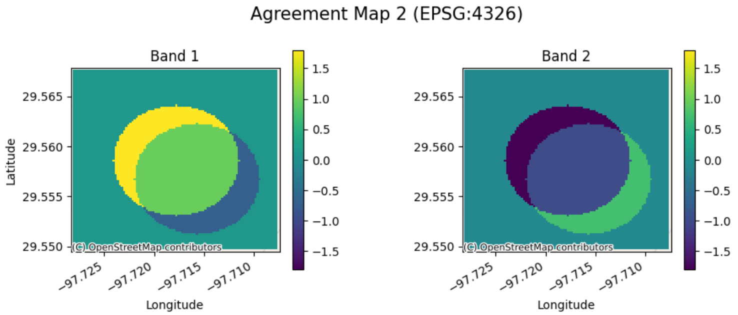
	map_id	compare_id
0	./benchmark_continuous_0.tif	compare1
1	./benchmark_continuous_1.tif	compare2

Catalog Result Table

	0	1	2
map_id_candidate	./candidate_continuous_0.tif	./candidate_continuous_1.tif	./candidate_continuous_1.tif
compare_id	compare1	compare2	compare2
agreement_maps	./agreement_continuous_0.tif	./agreement_continuous_1.tif	./agreement_continuous_1.tif
map_id_benchmark	./benchmark_continuous_0.tif	./benchmark_continuous_1.tif	./benchmark_continuous_1.tif
band	1	1	2
coefficient_of_determination	-0.06616	-2.829421	0.10903
mean_absolute_error	0.317389	0.485031	0.485031
mean_absolute_percentage_error	0.159568	0.202235	0.153235

Agreement Maps





Additional Functionality

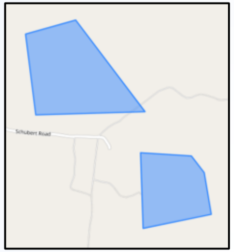
Additional Functionality

	Type	Regions	Metric Tables	Agreement Maps

Subsampling

Inclusionary

Exclusionary

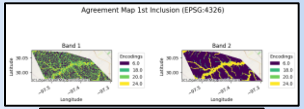


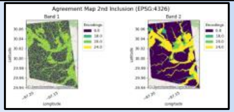
Metric Tables

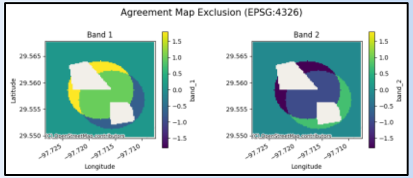
	0	1	2	3
band	1	1	2	2
subsample	1	2	1	2
fn	201953.0	182389.0	68239.0	58638.0
fp	761242.0	397531.0	43646.0	65967.0
tn	762262.0	398338.0	1479858.0	729902.0
tp	201936.0	181301.0	335650.0	305052.0
accuracy	0.50026	0.499879	0.94195	0.892541
balanced_accuracy	0.500157	0.499506	0.901198	0.877941
critical_success_index	0.173316	0.238171	0.749997	0.70999

	subsample	band	mean_percentage_error
0	1	1	0.125928
1	1	2	-0.111844
2	2	1	0.167116
3	2	2	-0.143187

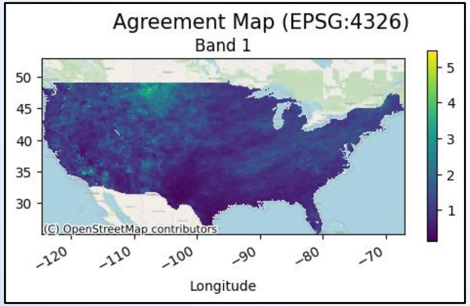
Agreement Maps

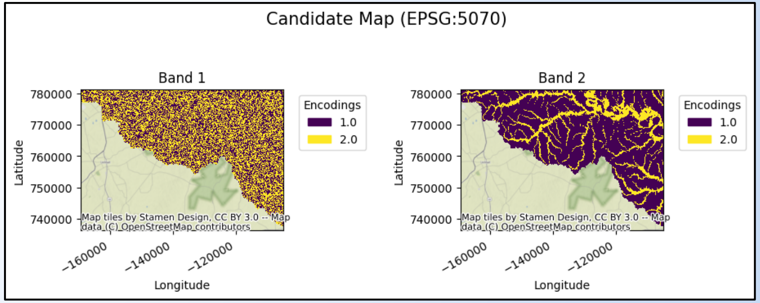






Visualization





Additional Functionality Continued

Registering Custom Statistical Functions

```python
from gval import CatStats

@CatStats.register_function(name="error_balance", vectorize_func=True)
def error_balance(fp: Number, fn: Number) -> float:
    return fp / fn
```

```python
metric_table_register = crosstab_table.gval.compute_categorical_metrics(
    negative_categories= None,
    positive_categories = [2],
    metrics=['error_balance']
)
```

	band	fn	fp	tn	tp	error_balance
0	1	639227.0	512277.0	NaN	2473405.0	0.801401

Stac API Client IO

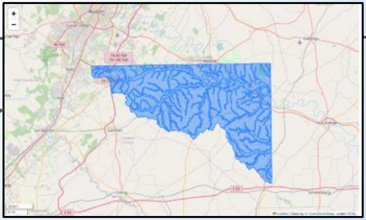
```python
from gval.utils.loading_datasets import dataset_from_stac

data = dataset_from_stac(
    url="https://earth-search.aws.element84.com/v1",
    collection="sentinel-2-l2a",
    time="2020-04-01/2020-05-01",
    bands=["aot"],
    time_aggregate="mean",
    bbox=[-105.78, 35.79, -105.66, 35.90]
)
```

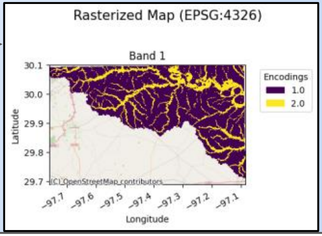
```
xarray.Dataset
▸ Dimensions:   (band: 1, x: 10980, y: 10980)
▾ Coordinates:
    band                 (band)       <U12     'aot'
    x                    (x)          float64  4e+05 4e+05 ... 5.097e+05 5.098e+05
    y                    (y)          float64  4e+06 4e+06 ... 3.89e+06 3.89e+06
    mgrs:grid_square     ()           <U2      'DV'
    constellation        ()           <U10     'sentinel-2'
    s2:saturated_de...   ()           int64    0
    instruments          ()           <U3      'msi'
    s2:datatake_type     ()           <U8      'INS-NOBS'
    mgrs:latitude_b...   ()           <U1      'S'
    mgrs:utm_zone        ()           int64    13
    s2:product_type      ()           <U7      'S2MSI2A'
    grid:code            ()           <U10     'MGRS-13SDV'
    proj:epsg            ()           int64    32613
    raster:bands         (band)       object   [{'nodata': 0, 'data_type': 'uin...
    title                (band)       <U31     'Aerosol optical thickness (AOT)'
    gsd                  (band)       object   None
    common_name          (band)       object   None
    center_wavelen...    (band)       object   None
    full_width_half_...  (band)       object   None
    epsg                 ()           int64    32613
▾ Data variables:
    band_1               (band, y, x) float64  dask.array<chunksize=(1, 1024, 1024), meta=np.ndarray>
▸ Indexes: (3)
▸ Attributes: (0)
```

Rasterization / Vectorization

```python
raster_candidate = vector_candidate.gval.rasterize_data(
    reference_map=benchmark, rasterize_attributes=["values"]
)
```

```python
vector_candidate = candidate.gval.vectorize_data()
```





Use and Contribution

- Documentation:
  https://github.com/NOAA-OWP/gval

- GitHub Issues:
  https://github.com/NOAA-OWP/gval/issues

- Main GitHub Page
  https://noaa-owp.github.io/gval/

GVAL
GEOSPATIAL EVALUATIONS

Acknowledgements

Gregory Petrochenkov [2]
Fernando Aristizabal [3]
Fernando Salas [1]

[1] NOAA/NWS Office of Water Prediction
[2] Lynker
[3] Earth Resources Technology

Thank You!

Greg Petrochenkov

greg.petrochenkov@noaa.gov

https://water.noaa.gov