ClimateSERV Documentation

Sarva Pulla

Jul 03, 2019

User Documentation

ClimateSERV is a SERVIR created web-based tool that provides three datasets together in one system to help decision-makers in SERVIR's data-sparse regions assess the evolving situation via holistic analysis of water and agriculture. Using ClimateSERV, development practitioners, scientists/researchers, and government decision-makers can readily analyze historical rainfall for the past 30 years and compare it with the best available forecasts for the next 180 days for their defined area of interest to improve understanding of, and make improved decisions for, issues related to agriculture and water availability.

Access ClimateSERV at this url

CHAPTER 1

Contents

1.1 User Guide

1.1.1 Getting Started!

Access ClimateSERV at this url

ClimateSERV allows development practitioners, scientists/researchers, and government decision-makers to visualize and download historical rainfall data, vegetation condition data, and 180-day forecasts of rainfall and temperature to improve understanding of, and make improved decisions for, issues related to agriculture and water availability.

In SERVIR regions, where long-term ground observations of rainfall are sparse, there is a critical need for satellite and model-derived rainfall data for predicting droughts, estimating crop yields, and more. Decision-makers need a way to accurately asses how severe a drought will be, how it compares to past droughts, and its potential effect on crop yields. Such assessments require accurate estimations of rainfall variations in space and time. It is important to place an evolving dryer-than-normal season into historical context in order to analyze the severity of rainfall deficits. Until now, such analyses used rainfall data from specific points on the Earth's surface. However, that data fails to show the region-wide variability that reveals comprehensive rainfall patterns.

Application Purpose

SERVIR has created a user-friendly, web-based tool – ClimateSERV – that provides several important datasets together in one system to help decision-makers in SERVIR's data-sparse regions assess the evolving situation via holistic analysis of water and agriculture. Using ClimateSERV, development practitioners, scientists/researchers, and government decision-makers can readily analyze historical rainfall for the past 30 years and compare it with the best available forecasts for the next 180 days for their defined area of interest to improve the understanding of, and make improved decisions for, issues related to agriculture and water availability.

Application Uses

With this tool, decision-makers can download, view, graph, and interpret the CHIRPS, eMODIS NDVI, and NMME seasonal forecast data in a web-based user interface. ClimateSERV can help decision-makers assess and monitor large-scale rainfall patterns, analyze how those patterns may be affected by climate change, determine likelihood of drought, and infer crop condition. Kenya Meteorological Service field offices are already using the data to provide climate resilience guidance to farmers. For example, KMS's Kericho office is using the CHIRPS dataset to downscale

seasonal climate outlooks for farmers' use in planning crop cultivars and planting times. SERVIR hubs plan to train end-users in their regions to use and analyze the CHIRPS, NMME, and NDVI data through ClimateSERV. Use of seasonal forecasts by end-users has been increasing. ClimateSERV will assist in the conversion of these large datasets into actionable information.

1.1.2 ClimateSERV Datasets

Climate Hazards group IR Precipitation with Stations (CHIRPS)

Scientists at Famine and Early Warning System (FEWS NET) who are members of the SERVIR Applied Sciences Team used 30 years' (1982- present) worth of multiple satellite data sources and ground observations to produce an unprecedented, global, spatially and temporally consistent and continuous 30-year record of satellite-derived rainfall data. This CHIRPS global dataset makes it possible to accurately assess and monitor large-scale rainfall patterns and analyze how they may be affected by climate change. The data are updated to the latest available rainfall estimates.

North American Multi-Model Ensemble (NMME) dataset

Forecasts of future precipitation are also critical to decision-makers. The NMME dataset, a compilation by National Oceanic and Atmospheric Administration (NOAA), reflects cutting edge work on seasonal forecasting. A SERVIR AST project has taken the NMME data and performed bias correction and spatial disaggregation using standard, well-accepted techniques to generate daily, 180-day temperature and precipitation forecasts for the entire globe. These seasonal forecasts, along with the CHIRPS historical rainfall data, provide an overall perspective to connect rainfall patterns from the past to future rainfall (up to 180 days out) as projected by NMME.

MODIS-derived Normalized Difference Vegetation Index (eMODIS NDVI)

SERVIR is also piloting the USGS pentadal eMODIS NDVI dataset at 250m spatial resolution over West Africa. NDVI, a measure of vegetation condition, provides a proxy for agricultural productivity by showing photosynthetic activity. By providing this 15 year dataset, SERVIR is enabling Ministries of Agriculture and the international donor community to explore how CHIRPS and NMME data link to vegetation growth and health. (The NDVI dataset is being expanded to other parts of Africa and beyond.) ClimateSERV enables decision-makers to link historical precipitation trends (CHIRPS) to past vegetation trends (NDVI) to gain insight into potential vegetation growth and health based on seasonal (up to 180 days) precipitation and temperature forecasts (NMME).

Evaporative Stress Index (ESI)

ESI reveals regions of drought where vegetation is stressed due to lack of water, enabling agriculture ministries to provide farmers with actionable advice about irrigation. The ESI can capture early signals of "flash drought," a condition brought on by extended periods of hot, dry, and windy conditions leading to rapid soil moisture depletion. Reduced rates of water loss can be observed through the use of land surface temperature before it can be observed through decreases in vegetation health or "greenness." The ESI describes soil moisture across the landscape without using observed rainfall data. This is critical in developing regions and other parts of the world lacking sufficient ground-based observations of rainfall. The ESI is based on satellite observations of land surface temperature, which are used to estimate water loss due to evapotranspiration (ET), the loss of water via evaporation from soil and plant surfaces and via transpiration through plant leaves. Generally, healthy green vegetation with access to an adequate supply of water warms at a much slower rate than does dry and/or stressed vegetation. Based on variations in land surface temperature, the ESI indicates how the current rate of ET compares to normal conditions. Negative ESI values show below normal ET rates, indicating vegetation that stressed due to inadequate soil moisture. (Plants' first response when stressed from lack of water is to reduce their transpiration to conserve water within the plant.)

Global Ensemble Forecast System (GEFS)

The Global Ensemble Forecast System (GEFS) is a weather forecast model made up of 21 separate forecasts, or ensemble members. GEFS was developed by the National Centers for Environmental Prediction (NCEP), a group within NOAA. NCEP started the GEFS to address the nature of uncertainty in weather forecasts – GEFS quantifies the amount of uncertainty in a forecast by generating an ensemble of multiple forecasts, each minutely different, or perturbed, from the original observations. GEFS consist of 21 different models that each produce a 16-day forecast every 6 hours. Each model has the spatial resolution that ranges from 25 to 100km, and often each model has a systematic bias from reality, one that is different from other model biases, making inter-comparison among models difficult. University of With SERVIR's guidance, FEWS NET/UCSB takes the GEFS data (the 21 GEFS model ensembles) and bias-adjusts each model using the historical CHIRPS data. Thus the merged CHIRPS-adjusted output dataset removes the inter-model biases, and results in a more consistent forecast dataset. University of California at Santa Barbara's Climate Hazards Group, produces the new CHIRPS-GEFS rainfall dataset and the forecasts go back to 1985. The CHIRPS-GEFS's forecast is updated every five days at a spatial resolution of 5 km across the globe. Figure 1 shows the forecast update cycle. The "first forecast" is the earliest time at which the 10-day rainfall can be forecasted. The "last forecast" shows the latest date on which that 10-day rainfall forecast can be produced. The data on ClimateSERV is the compilation of the latest available rainfall forecast for a given date per this production cycle.

Integrated Multi-satellitE Retrievals Precipication (IMERG)

Precipitation data from the Integrated Multi-satellitE Retrievals (IMERG) for Global Precipitation Mission (GPM).The IMERG algorithm intercalibrates, merges and interpolates "all" satellite passive microwave precipitation estimates, together with microwave-calibrated infrared (IR) satellite estimates, monthly precipitation gauge analyses, and potentially other precipitation estimators at fine time and space scales for the TRMM and GPM eras over the entire globe.

1.2 Developers API

A Developers API is provided for those who wish to incorporate the ClimateSERV data into their own separate application or script.

1.2.1 API Methods

All API methods must be called using the following pattern:

```
{{ base api url }}/[MethodName]/?param1=value1&param2=value2&...paramN=valueN
```

Base API URL

The API allows access to the ClimateSERV processing engine and resulting data so that developers can implement their own UI or extract needed data directly from the back-end.

The base URL for all API access is: `https://climateserv.servirglobal.net/chirps/`

Call-back Support

ALL API functions support passing in the parameter:

> "?callback=callBackFunctionName"

The return data/object resulting from the API function call will be wrapped into a javascript function as named in the line above. For instance, including "?callback=ProcessResults" as a parameter in the call to the API would require you to define a function "ProcessResults(returnObject)", where "returnObject" is the object passed back as output from the API.

getParameterTypes

`https://climateserv.servirglobal.net/chirps/getParameterTypes/`

Purpose: Get a list of the current supported statistical operation types and their code values.

Supported Methods: GET

Parameters(input): None

Returns(output): Array[] // list of items. Each item is a list with 3 elements. Each item represents a currently supported statistical operation type.

Output Details: currentListItem = returnList[n] currentListItem[0] is (int), operation type. This is the value that the server understands as the operation type. currentListItem[1] is (string), short name for operation type. currentListItem[2] is (string), more readable name for operation type (used as label on dropdown part of the UI).

Example Output:

```
[[0, "max", "Max"], [1, "min", "Min"], [2, "median", "Median"], [3, "range", "Range"],
↪ [4, "sum", "Sum"], [5, "avg", "Average"]]
```

getFeatureLayers

`http://climateserv.servirglobal.net/chirps/getFeatureLayers/`

Purpose: Get a list of the current feature layers included in the map for processing.

Supported Methods: GET

Params(input): None

Returns(output): Array[] //list of feature layer info. Each layer info includes visible, displayName, and id.

Output Details: currentLayer = returnList[n] currentLayer['visible'] is (string), true or false. currentLayer['displayName'] is (string), displayed layer name currentLayer['id'] is (string), id of layer

Example Output:

```
[{"visible": "true", "displayName": "Countries", "id": "country"},
{"visible": "false", "displayName": "Admin #1", "id": "admin_1_earth"},
{"visible": "false", "displayName": "Admin #2", "id": "admin_2_af"}]
```

getClimateScenarioInfo

`http://climateserv.servirglobal.net/chirps/getClimateScenarioInfo/`

Purpose: Get information about the data structure of currently supported climate scenario datatypes. At this time there are a total of 10 'Climate_Ensembles'. Each 'Climate_Ensemble' can have 1, 2, or n 'Climate_Variables'. The combination of 'Climate_Ensemble' and 'Climate_Variable' is unique and matches up to an individual dataset. (so 1 list of images per 'Climate_Ensemble' + 'Climate_Variable' combination.)

Supported Methods: GET

Params(input): None

Returns(output): Object{}

Output Details: Note on Climate Scenario Datatypes and how they relate to 'Data Type Numbers': To integrate this new data structure with the existing system of processing and serving out datasets, these combinations have been mapped to a unique datatype number.

Example Output:

```
// Return Object
returnObject.RequestName        // (string) Static, name of the request (currently:
↪'getClimateScenarioInfo')
returnObject.isError                                    // (bool)
returnObject.climate_DataTypeCapabilities               // (array[])
returnObject.climate_DatatypeMap                        // (array[])
// climate_DataTypeCapabilities
returnObject.climate_DataTypeCapabilities[n].dataTypeNumber     // (int) The server␣
↪uses the datatype to match up to a dataset using this number.
returnObject.climate_DataTypeCapabilities[n].current_Capabilities       // (JSON␣
↪encoded String) Contains various additional properties related to the dataset (such␣
↪as projection, forecast days, start/end date's for forecast range, fill value,␣
↪projection, grid info, etc.)

// List of properties in the current_Capabilities JSON String
capabilitiesItem = JSON.parse(returnObject.climate_DataTypeCapabilities[n].current_
↪Capabilities);
capabilitiesItem.data_category          // (string) Data Type Category (for all␣
↪climate datasets this should be the same)
capabilitiesItem.date_FormatString_For_ForecastRange    // (string) Format of the
↪'endDateTime' and 'startDateTime' props written as a python format string (i.e. "%Y_
↪%m_%d")
capabilitiesItem.description            // (string) Text description of this dataset
capabilitiesItem.endDateTime            // (string) Last Calendar date of the the␣
↪forecast range for this dataset
capabilitiesItem.ensemble               // (string) 'Climate_Ensemble' for the␣
↪current datatype
capabilitiesItem.fillValue              // (unsigned int) The 'no data' value of the␣
↪current dataset (usually set to -9999)
capabilitiesItem.grid                   // (array[6]) 6 elements to describe␣
↪geospatial raster resolution and positioning of this dataset
capabilitiesItem.name                   // (string) Shorter description of this␣
↪dataset
capabilitiesItem.number_Of_ForecastDays // (int) Number of days in the forecast range
capabilitiesItem.projection             // (string) GIS Projection value as a string (
↪"GEOGCS["WGS 84",DATUM......]etc")
capabilitiesItem.size                   // (array[2])  X,Y  ([0],[1]) pixel size of␣
↪original dataset image
capabilitiesItem.startDateTime          // (string) First calendar date of the␣
↪forecast range for this dataset
capabilitiesItem.variable                       // (string) 'Climate_Variable' code
capabilitiesItem.variable_Label         // (string) Human readable version of the
↪'capabilitiesItem.variable' property.


// climate_DatatypeMap
returnObject.climate_DatatypeMap[n].climate_Ensemble            // (string) Parent
↪'Climate_Ensemble'
returnObject.climate_DatatypeMap[n].climate_DataTypes           // (array[]) List of
↪'Climate_Variables' and their DataTypeNumbers for the current 'Climate_Ensemble'
```

(continues on next page)

(continued from previous page)

```
// climate_DataTypes
returnObject.climate_DatatypeMap[n].climate_DataTypes[m].climate_Ensemble       //␣
↪(string) Current Climate Ensemble value (should match the parent prop)
returnObject.climate_DatatypeMap[n].climate_DataTypes[m].climate_Ensemble_Label //␣
↪(string) Human readable version of 'returnObject.climate_DatatypeMap[n].climate_
↪DataTypes[m].climate_Ensemble'
returnObject.climate_DatatypeMap[n].climate_DataTypes[m].climate_Variable       //␣
↪(string) Current Climate Variable
returnObject.climate_DatatypeMap[n].climate_DataTypes[m].climate_Variable_Label //␣
↪(string) Human readable version of 'returnObject.climate_DatatypeMap[n].climate_
↪DataTypes[m].climate_Variable'
returnObject.climate_DatatypeMap[n].climate_DataTypes[m].dataType_Number                ␣
↪  // (int) The value the server uses to uniquely identify the current datatype (or
↪'climate_ensemble' + 'climate_variable' combination)
```

submitDataRequest

`https://climateserv.servirglobal.net/chirps/submitDataRequest/`

Purpose: Submit a new asynchronous processing request to the server.

Supported Methods: GET,POST

Parameters(input):

```
'datatype'      // (int), the unique datatype number for the dataset which this␣
↪request operates on
'begintime'     // (string), startDate for processing interval, format ("MM/DD/YYYY")
'endtime'       // (string), endDate for processing interval, format ("MM/DD/YYYY")
'intervaltype'  // (int), enumerated value that represents which type of time␣
↪interval to process (daily, monthly, etc) (This enumeration is currently hardcoded␣
↪in the mark up language of the current client).
'operationtype'         // (int), enumerated value that represents which type of␣
↪statistical operation to perform on the dataset, see api call 'getParameterTypes/'␣
↪for the list of currently available types.
// Either 'geometry' by itself or these other two params together, 'layerid' and
↪'featureids' are required
'geometry'(optional)// (object), the geometry that is defined by the user on the␣
↪current client
'layerid'(optional) // the layerid that is selected by the by the user on the current␣
↪client
'featureids'(optional)  // the featureids as selected by the user on the current␣
↪client
'isZip_CurrentDataType'(optional) // (string), Leaving this blank converts to 'False'␣
↪on the server.  Sending anything through equates to a 'True' value on the server. ␣
↪This lets the server know that this is a job to zip up and return a full dataset.
```

Returns(output): string // returns either the job ID ('uniqueid') as a UUID or an error message

Output Details: Submit the new datarequest and get the job ID as a response. The returned job ID can then be used to retrieve results (see getDataFromRequest/).

Warning: Be sure that the polygon and geojson coordinates are in the EPSG:4326 projection

Example request with a polygon:

If you are interested in retrieving the CHIRPS data for a certain polygon and a time period period. You will make the following request:

```
https://climateserv.servirglobal.net/chirps/submitDataRequest/?datatype=0&
↪begintime=04/01/2018&endtime=04/30/2018&intervaltype=0&operationtype=5&
↪callback=successCallback&dateType_Category=default&isZip_CurrentDataType=false&
↪geometry={"type":"Polygon","coordinates":[[[21.533203124999996,-3.1624555302378496],
↪[21.533203124999996,-6.489983332670647],[26.279296874999986,-5.441022303717986],[26.
↪10351562499999,-2.635788574166625],[21.533203124999996,-3.16245553023784961]]]}
```

Tip: The above request can be modified to include a geojson with multi-polygon. Simply replace the geometry parameter with the respective geojson geometry.

Example Output:

```
["7e917e63-600d-4a1e-a069-ab8f73c9fcaf"]
```

getDataRequestProgress

`https://climateserv.servirglobal.net/chirps/getDataRequestProgress/`

Purpose: Get the current progress the server has made on processing the given request job ID

Supported Methods: GET

Parameters(input): 'id' // (string/uuid), the unique job id (UUID format) of the job to check

Returns(output): float // returns the progress value as a float between 0.0 and 100.0. If error, a value of '-1' is returned instead

Output Details: Ask the server what the progress on processing the current jobID is. Get a number back, display/update the client progress bar, wait a few seconds, make the request again.

Example Output:

```
27.0
```

getDataFromRequest

`http://climateserv.servirglobal.net/chirps/getDataFromRequest/`

Purpose: Get the data from a job that has completed it's processing

Supported Methods: GET

Parameters(input): 'id' // (string/uuid), the unique job ID of the completed job

Returns (output): object{} // Returns the data generated from the request (usually a list of numbers and dates). See below.

Output Details: Ask the server for the data for a given completed Job, passing in the job ID (UUID string).

Example Output:

```
retObj.data                        // (Array[]) list of data granules that the␣
↪processing job output created.

granule = retObj.data[n]                  // (object), single data granule

granule.date                       // (string), readable date for current data granule.␣
↪Format "d/m/y" not fixed length
granule.workid                     // (string), unique id for that process item (this ID␣
↪is only used by the server internally.
granule.epochTime                  // (string), EpochTime (so we don't have to parse␣
↪readable date strings on the client side)
granule.value                      // (object), the key in this object matches the␣
↪statistical operation performed, and the value of that key is the value generated␣
↪for that particular data granule.
```

Example:

For a completed job where the initial submit data request was for: User defined polygon, 'Daily' time interval, 'Max' statistical value, and for the time range Jan 1, 2015 to Jan 31, 2015


```
{
        "data":
        [
                {
                        "date": "1/1/2015",
                        "workid": "01f4839f-7b9c-447f-b50f-0ca257c0a339",
                        "epochTime": "1420092000",
                        "value": {"max": 0.3055223822593689}
                },
                {
                        "date": "1/2/2015",
                        "workid": "58b6f7ea-5490-4ccd-a715-5e028407ad16",
                        "epochTime": "1420178400",
                        "value": {"max": 0.15552784502506256}
                },
                {
                        ....,
                        ....,
                },

                {
                "date": "1/31/2015",
                "workid": "e021a12c-7346-4b7b-a273-bd39c7fde99b",
                "epochTime": "1422684000",
                "value": {"max": 4.206714630126953}
                }
        ]
}
```


1.2.2 List of Datatypes

Regular Datasets

Dataset Name	Datatype Number	Availability
Global CHIRPS	0	Daily from 1981 to present
NDVI MODIS-West Africa	1	Every five days from 2001 to 2017
NDVI MODIS-East Africa	2	Every five days from 2001 to 2018
NDVI MODIS-Central Asia	28	Every five days from 2001 to 2017
Global ESI 4 Week	29	Every four weeks from 2001 to present
Global ESI 12 Week	33	Every twelve weeks from 2001 to present
IMGERG	26	Daily from 2015 to present
CHIRS-GEFS Anomalies	31	Decadal from 1985 to present
CHIRS-GEFS Precip	32	Decadal from 1985 to present

Seasonal Forecast Datasets

The seasonal forecasts are generated from a NMME model ensemble run.

Dataset Name	Datatype Number
Ensemble 1, Temperature	6
Ensemble 1, Precipitation	7
Ensemble 2, Temperature	8
Ensemble 2, Precipitation	9
Ensemble 3, Temperature	10
Ensemble 3, Precipitation	11
Ensemble 4, Temperature	12
Ensemble 4, Precipitation	13
Ensemble 5, Temperature	14
Ensemble 5, Precipitation	15
Ensemble 6, Temperature	16
Ensemble 6, Precipitation	17
Ensemble 7, Temperature	18
Ensemble 7, Precipitation	19
Ensemble 8, Temperature	20
Ensemble 8, Precipitation	21
Ensemble 9, Temperature	22
Ensemble 9, Precipitation	23
Ensemble 10, Temperature	24
Ensemble 10, Precipitation	25

CHAPTER 2

Indices and tables

- genindex
- search