Climate Change Adaptation and Visualization Tool Documentation

Release 0.0.9.dev

James Mills

June 17, 2014

Contents

Release 0.0.9.dev

Date June 17, 2014

CHAPTER 1

About

ccav is a Climate Change and Adaptation Visualization tool and web application written in the Python Programming Language to help visualize, analyze and track climate change using varying climate models and data.

- Visit the Project Website
- Read the Docs
- Download it from the Downloads Page

1.1 Requirements

1.1.1 ccav backend

- Python
- MapServer
- Fiona
- GDAL
- procname
- ujson
- six
- py
- circuits
- funcy
- argparse (*For Python <2.7*)
- jsonselect

1.1.2 ccav webui

- jquery
- jqueryui
- leaflet

- slickgrid
- d3

1.1.3 ccav data processing and command line tools

- progress
- requests
- fabric
- numpy
- py
- Fiona
- GDAL
- Shapely
- matplotlib
- geojson
- Pillow
- ujson
- plumbum
- funcy

1.2 Supported Platforms

- Linux, FreeBSD, Mac OS X
- Python 2.6, 2.7, 3.2, 3.3
- PyPy: 2.0

Windows: We acknowledge that Windows exists and make reasonable efforts to maintain compatibility. Unfortunately we cannot guarantee support at this time.

Note: Although the above environments are supported we currently only test for Python 2.7 on a CentOS 6.3 x86_64 (Linux) system.

For further information see the ccav documentation.

CHAPTER 2

Documentation

2.1 Development

2.1.1 Getting Started

The simplest way to get started with development is to grab the ccav ansible playbooks and run them using the ansible tool against a blank Centos 6.4 x86_64 virtual machine using your preferred virtualization platform. The playbooks will install everything required to build, develop and deploy the application and ia always kept up-to-date for dependency testing and deployment purposes.

You may also follow these *rough* instructions to get up and running on Mac OS X with the Homebrew package management tool.

1. Install Homebrew:

```
ruby -e "$(curl -fsSL https://raw.github.com/mxcl/homebrew/go)"
```

1. Install python:

```
brew install python
```

1. Install numpy:

```
pip install numpy
```

1. Install gdal:

```
brew install gdal
```

1. Install freetype:

```
brew install freetype
```

1. Install mapserver:

```
brew install mapserver
```

1. Install virtualenvwrapper:

```
pip install virtualenvwrapper.sh
```

1. Make sure you have something like this setup in your bash profile:

```
#!/bin/bash

# Check if virtualenvwrapper.sh is available and source it
if $(which virtualenvwrapper.sh &> /dev/null); then
    source $(which virtualenvwrapper.sh)
fi
```

1. Create a new virtual environment:

```
mkvirtualenv --system-site-packages ccav
```

1. Clone the development repository:

```
hg clone https://bitbucket.org/ccaih/ccav
```

1. Bootstrap, build and run:

```
cd /path/to/ccav
./bootstrap.sh
fab develop
dev webui:dev=yes
ccav
```

There is a nice ASCIINema based screencast of how to setup a full working development environment based on the above steps:

- https://asciinema.org/a/8300

2.2 Data

The following list of data sources are used by this sytem:

- Wallace Initiative

The data will be stored in the following directory layout (*sample only*):

```
.
-- models
|   -- RCP3PD
|   |   -- cccma-cgcm31
|   |   |   -- 2015 -> ../../../.sources/models/RCP3PD_cccma-cgcm31_2015
|   |   |   -- 2025 -> ../../../.sources/models/RCP3PD_cccma-cgcm31_2025
|   |   -- csiro-mk30
|   |       -- 2015 -> ../../../.sources/models/RCP3PD_csiro-mk30_2015
|   |       -- 2025 -> ../../../.sources/models/RCP3PD_csiro-mk30_2025
|   -- RCP45
|       -- cccma-cgcm31
|       |   -- 2015 -> ../../../.sources/models/RCP45_cccma-cgcm31_2015
|       |   -- 2025 -> ../../../.sources/models/RCP45_cccma-cgcm31_2025
|       -- csiro-mk30
|           -- 2015 -> ../../../.sources/models/RCP45_csiro-mk30_2015
|           -- 2025 -> ../../../.sources/models/RCP45_csiro-mk30_2025
-- regions
|   -- IBRA
|       -- IBRA7_regions.dbf
|       -- IBRA7_regions.prj
|       -- IBRA7_regions.sbn
|       -- IBRA7_regions.sbx
|       -- IBRA7_regions.shp
```

```
|       -- IBRA7_regions.shp.xml
|       -- IBRA7_regions.shx
-- summaries
```

This structure will support the addition of new data and ensure future expansion of the system to support global data visualization.

2.3 Web API

The following document outlines an overall design of the RESTful Web Service implemented and provided by this system for access to various underlying vector, raster and meta data.

Features of the API include:

- Versioned Access
- RESTful Architecture
- Multiple Output Formats
- Self Documenting and Traversing

2.3.1 Global Parameters

All endpoints accept the following optional parameter(s):

- `format` representation format. – **Default:** `json`

2.3.2 Global Help

All endpoints accept and implement a `help` resource that returns a document that describes the API endpoint being requested, what it does and what data is provided. Example:

```
/api/v1/help
```

2.3.3 Responses

There are two types of responses. Normal responses are responses to any API endpoint that results in a valid response with data, a Help response is a response to an API endpoint's help document describing what the API endpoint does and what data is provided by it.

Normal Responses

```
{
    data: [ ... ]
}
```

Where:

- `data` is a list (*possibly empty*) of data returned

Help Response

A response describing the API endpoint being requested, what it does and the data provided.

Note: This can be requested in any supported format. e.g: text, html, json

2.3.4 Summaries

This data can be used to explore region similarities and differences in tabular or graph form.

A single "resource" is a table of data for a given intersection of region type, year, future model, and emission scenario. The resource includes summary data for every region of the given region type.

Endpoint: `/summaries`

Structure:

```
/summaries
/summaries/projections
/summaries/projections/scenario
/summaries/projections/scenario/model
/summaries/projections/scenario/model/year
/summaries/projections/scenario/model/year/region
/summaries/projections/scenario/model/year/var
/summaries/projections/scenario/model/year/var/region
```

Where:

- `model` a climate model
- `scenario` an emissions scenario
- `year` a future year
- `var` a specific bio-climatic variable
- `region` a collection of regions to intersect against

Examples(s):

```
GET /api/v1/summaries/projections/scenario/model/year/region
GET /api/v1/summaries/projections/scenario/model/year/var/region
```

2.3.5 Variables

This data describes various aspects of the bio-climatic variables that are given values in the region summaries, and/or displayed as WMS layers. The data includes the variable's description, measurement units, and maximum and minimum (across all available summaries).

A single "resource" is a set of metadata for a particular variable.

Endpoint: `/variables`

Structure:

```
/variables
/variables/var
```

Where:

- `var` the name of a specific bio-climatic variable

Example 1:

```
GET /api/v1/variables
```

will return a list of all variables, similar to:

```
{
    data: [
        {
            id: "bio1",
            shortname: "Annual Precip",
            longname: "Mean Annual Precipitation",
            units: "mm",
            min: 3,
            max: 2456
        },
        {
            id: "bio2",
            shortname: "Annual Max Temp",
            longname: "Mean Annual Maximum Temperature",
            units: "C",
            min: 15,
            max: 48
        },
        ...
        { ... }
    ]
}
```

Example 2:

```
GET /api/v1/variables/bio2
```

Will return a similar list but containing just the specified variable:

```
{
    data: [
        {
            id: "bio2",
            shortname: "Annual Max Temp",
            longname: "Mean Annual Maximum Temperature",
            units: "mm",
            min: 15,
            max: 48
        }
    ]
}
```

2.3.6 Regions

This data could be used by a client to describe a particular region in detail, including a geographic rendering of the borders of the region. A single "resource" is a list of data for a given region.

Endpoint: `/regions`

Structure:

```
/regions/
/regions/collection/
/regions/collection/sub-collection
/regions/collection/sub-collection/id
```

Where:

- `collection` is a collection.
- `subcollection` is a sub-collection of regions belonging to a parent collection.
- `id` is a unique region belonging to a collection/sub-collection.

Note:

- Region hierarchy can be arbitrarily nested.

Examples:

```
GET /api/v1/regions/
GET /api/v1/regions/collection/
GET /api/v1/regions/collection/subcollection/
GET /api/v1/regions/collection/subcollection/id
```

2.3.7 Features

Used to request and filter GeoJSON features from a vector data source such as a *Shapefile <http://en.wikipedia.org/wiki/Shapefile>*).

Endpoint: `/features`

2.3.8 Geometry

Used for server-side geometric operations (//buffer, intersection, union, etc//).

Endpoint: `/geometry`

2.3.9 WMS

Standard WMS service for tile rendering of Raster and Vector layers.

Endpoint: `/wms`

2.4 API Documentation

2.4.1 ccav

ccav package

Subpackages

ccav.api package

Submodules

ccav.api.data module

ccav.api.features module

ccav.api.hello module

ccav.api.map module

ccav.api.vars module

Module contents

Submodules

ccav.config module

ccav.main module

ccav.reprconf module

ccav.resource module

ccav.root module

ccav.static module

ccav.tools module

ccav.unrepr module

ccav.utils module

ccav.version module

Module contents

2.5 Changes

2.5.1 ccav 0.0.9.dev

-

2.5.2 ccav 0.0.8 (*2014-04-07*)

- Issue #77: Scale bar and legend not rendering
- Issue #74: Added support for tile indexes to support `time` parameter. Simple test: http://localhost:9000/api/Map?service=WMS&request=GetMap&version=1.1.1&layers=indexes:RCP3PD/csiro-mk30/bioclim_11.shp&styles=&format=image/png&transparent=true&height=256&width=256&time=2015-01-01&zIndex=500&srs=EPSG:3857&&bbox=16906647.66422842,-3130860.678560819,17532819.799940586,-2504688.542848654
- Added support for SLD (*which MapServer already has*). Simple test: http://127.0.0.1:9000/api/Map?service=WMS&request=GetMap&version=1.1.1&layers=models:RCP3PD/cccma-cgcm31/2015/bioclim_11.tif&styles=&format=image/png&transparent=true&height=256&width=256&zIndex=500&srs=EPSG:2504688.542848654,17532819.799940586,0&SLD=http://127.0.0.1:9000/styles/heatmap.xml
- Move the OWS Proessing out to a Worker in process mode to handle concurrent Map API requests.
- Added Year selection to the UI
- Updated Map UI to use local Map API to render raster models of bioclim layers.
- Updated Map API so that just a region's name can be passed rather than a hard coded path.

  Example: http://localhost:9000/api/Map?service=WMS&request=GetMap&version=1.1.1&layers=regions:States&styles=&form 5009377.085697311,15028131.257091932,-2504688.542848655&filter=([code]%20=%205)
- Added heatmap styling xml spectrums for temperature (3 scales) and rainfall (4 scales).
- Added hash set/check functionality to support copy/paste of links to exchange selections between users.

  Example: http://localhost:9000/climate-projection-geo-browser#scenario=RCP3PD&model=csiro-mk30&bioclim=bioclim_12&year=2065
- Added leaflet legends to match XML SLD spectrums.

2.5.3 ccav 0.0.7 (*2014-03-06*)

- Use a separate virtual environment when building and deplyoing the documentation.
- Updated documented requirements
- Integrated use of sphinxcontrib-issuetracker for linking to bitbucket issues in documentatin. See Issue #29
- Implemented support for using *MapServer Expressions <http://mapserver.org/mapfile/expressions.html>*.

Example request:

```
http://localhost:9000/api/Map
    ?service=WMS
    &request=GetMap
    &version=1.1.1
    &layers=regions:States/States
    &styles=
    &format=image/png
    &transparent=true
    &height=256
    &width=256
    &zIndex=500
    &srs=EPSG:3857
    &bbox=12523442.714243276,-5009377.085697311,15028131.257091932,-2504688.542848655
    &filter=([code]%20=%205)
```

This url (*clickable*) is: http://localhost:9000/api/Map?service=WMS&request=GetMap&version=1.1.1&layers=regions:States/States&st
5009377.085697311,15028131.257091932,-2504688.542848655&filter=([code]%20=%205)

- Added infrastructure for performing integration and user interface testing via pytest and selenium.
- Switch to using seleniumwrapper to make writing tests a little easier.
- Fixed Issue #66

2.5.4 ccav 0.0.6 (*2013-12-09*)

- Fixed packaging. ccav **is NOT** `zip_safe`.

2.5.5 ccav 0.0.5 (*2013-12-09*)

- Fixed the port that the test instance (http://testccav.terranova.org.au/) runs on. Was conflicting with prod (http://ccav.terranova.org.au/)
- Added links to the Changes and Docs and the currently running version to the landing page.
- Deploy test instnace (http://testccav.terranova.org.au/) in `dev` mode for easier debugging of JS/CSS.
- Improved the style/theme of the internally hosted documentation.
- Added link to API and API Docs to landing page.
- Added support for querying the Features API utilizing JSON Select via the nicily written jsonselect library by Matthew Hooker.
- Added new Data API `/api/Data` for accessing data about the models.
- Improved overall API and allowed querying on any endpoint. Any API endpoint can accept a `q=<json select>` query-string paremter.

e.g:

```
GET /api/Data/Features/States?q=.STATE_NAME
```

- Added new Vars API `/api/Vars` for accessing bioclim descriptions and min/max values across regions.
- Added basic Map API providing WMS/WFS services. This is available at `/api/Map`
- Beginning of the **Map Browser** tool.

2.5.6 ccav 0.0.4 (*2013-11-28*)

- This version starts to document changes in a *Change Log* style manner.

2.6 Road Map

Here's a list of upcoming releases of ccav in order of "next release first".

Each bullet point states a high level goal we're trying to achieve for the release whilst the "Issues List" (*linked to our Issue Tracker*) lists specific issues we've tagged with the respective milestone.

Note: At this stage we don't have any good estimates for our milestones but we hope we can improve this with future releases and start adding estimates here.

2.6.1 ccav 0.0.9

See also:

ccav 0.0.9 milestone

2.6.2 ccav 1.0

See also:

ccav 1.0 milestone

2.7 TODO

- Build filtering support for the Map API. e.g: `filter=id%20=%201` as a query-string parameter.
- Create a tool `remapshapefiles` to remap the properties within Shapefile(s).
- Use pystat for building up collected stats over a large collection. Replaces `streamio.minmax`.

See also:

https://bitbucket.org/ccaih/ccav/issues

2.8 Glossary

VCS Version Control System, what you use for versioning your source code

2.9 Documentation TODO

2.10 PyPi README Page

ccav is a Climate Change and Adaptation Visualization tool and web application written in the Python Programming Language to help visualize, analyze and track climate change using varying climate models and data.

- Visit the Project Website
- Read the Docs
- Download it from the Downloads Page

2.10.1 Requirements

ccav backend

- Python
- MapServer
- Fiona
- GDAL

- procname
- ujson
- six
- py
- circuits
- funcy
- argparse (*For Python <2.7*)
- jsonselect

ccav webui

- jquery
- jqueryui
- leaflet
- slickgrid
- d3

ccav data processing and command line tools

- progress
- requests
- fabric
- numpy
- py
- Fiona
- GDAL
- Shapely
- matplotlib
- geojson
- Pillow
- ujson
- plumbum
- funcy

2.10.2 Supported Platforms

- Linux, FreeBSD, Mac OS X
- Python 2.6, 2.7, 3.2, 3.3
- PyPy: 2.0

Windows: We acknowledge that Windows exists and make reasonable efforts to maintain compatibility. Unfortunately we cannot guarantee support at this time.

Note: Although the above environments are supported we currently only test for Python 2.7 on a CentOS 6.3 x86_64 (Linux) system.

For further information see the ccav documentation.

CHAPTER 3

Indices and tables

- *Index*
- *modindex*
- *search*
- *Glossary*
- *Documentation TODO*
- *PyPi README Page*