Habitable Exo-Planets Documentation

- **Created at:** 2017-10-17 16:43:55.638844
- **Corral Version:** 0.3

This is an example pipeline using a custom version of the Exoplanets dataset (http://exoplanets.org/).

Models

Model HabitableZoneStats

- **Table:** `HabitableZoneStats`

Resume of data about the capability of the planet to have life

Fields

- `planet`: Planet of the statistics
- `luminosity`: Stellar luminosity [solar luminosity]
- `radio_inner`: Inner boundary of habitable zone [AU]
- `radio_outer`: Outer boundary of habitable zone [AU]
- `in_habitable_zone`: [boolean]

Model Planet

- **Table:** `Planet`

Represent a single exoplanet.

Fields

- `name`: Name.
- `per` : Period [days]
- `mass`: Planet mass [solar masses]
- `sep`: Star-planet Separation [AU]
- `dist`: Distance to the star [pc]
- `mstar`: Stellar mass [solar masses]
- `rstar`: Stellar radius [solar radii]
- `teff`: Effective temperature [K]
- `fe`: Metallicity

Loader:

- **Python Path** `exo.load.Loader`

Extract data from the `exoplanets.csv` and feed the stream of the pipeline.

Steps

Step HabitableZone

- **Python Path** `exo.steps.HabitableZone`

Calculate some statistics of the star of a given planet and then determines if is in their habitable zone.

Alerts

Alert InHabitableZoneAlert

- **Python Path** `exo.alerts.InHabitableZoneAlert`

Store a list of planets in habitable zone in a log file and also generate a period vs mass plot of this planets

Command Line Interface

run 'python in_corral.py –help'

Available subcommands

CORRAL

- `alembic`: Execute all the Alembic migration tool commands under Corral enviroment
- `check-alerts`: Run the alerts and announce to the endpoint if something is found
- `create`: Create a new corral pipeline
- `create-doc`: Generate a Markdown documentation for your pipeline
- `create-models-diagram`: Generates a class diagram in 'dot' format of the models classes
- `createdb`: Create all the database structure
- `dbshell`: Run an SQL shell throught sqlalchemy
- `exec`: Execute file inside corral environment
- `groups`: List all existent groups for Steps and Alerts
- `load`: Excecute the loader class
- `lsalerts`: List all available alert classes
- `lssteps`: List all available step classes
- `makemigrations`: Generate a database migration script for your current pipeline
- `migrate`: Synchronizes the database state with the current set of models and migrations

- `notebook`: Run the Jupyter notebook inside Corral enviroment
- `profile`: Run a CPU profile (with cProfile) and then open the report with your default browser
- `qareport`: Run the QA test for your pipeline and make a reports of errors, maintanability, coverage and a full QA index.
- `run`: Excecute the steps in order or one step in particular
- `run-all`: Shortcut command to run the loader, steps and alerts asynchronous. For more control check the commands 'load', 'run' and 'check-alerts'.
- `shell`: Run the Python shell inside Corral enviroment
- `test`: Run all unittests for your pipeline