This document gives an overview of WHAM! beta version. It covers installation and the fundamentals of setting up and using the model.

## Installation

1. Download of code is via github. <https://github.com/OliPerkins1987/Wildfire_Human_Agency_Model>

Download from main branch. (If downloading using website, then Green code button on RHS -> download zip).

1. Conda virtual env – setup a virtual environment using conda or whatever using the ‘package-list.tx’ file

conda create --name <WHAM> --file <package-list.tx>

Alternatively, as agentpy is not available via conda forge, initialise a base conda environment, then install agentpy (pip install agentpy) and netCDF4 using conda install

1. Install the code by navigating to wherever you unzipped the code in conda prompt, and type: python setup.py install



1. Download data from sharepoint: <https://emckclac-my.sharepoint.com/:f:/r/personal/k1758409_kcl_ac_uk/Documents/wham_files/Model%20files?csf=1&web=1&e=ziscl6>
2. Linking the data and model
3. Open the ‘local\_load\_up.py’ script in the src/data\_import directory
4. Go to lines 25-26 & set the root directory locations for where the data files are stored, and the sub directory for where the map data is stored (by default …/wham\_dynamic)
5. save this version over the downloaded version
6. To check that has all worked, navigate to wham/tests and just type & enter ‘pytest’



This will take around 20 mins to run. There should be some warning messages about dividing by zero & about np.bool – that is fine. The latter is to do with under-the-hood operations of the netCDF4 package. Any test failures need to be explored.

## Running the model

### Set up instantiate.py

1. The instantiate.py script now has a parameter – write\_fp on lin 113 that specifies where you would like files to be written. This happens by default each model year. This needs to be written before the script will run correctly.
2. You can then also set the remaining parameters as relevant

### Run model (standalone)

1. You can either run the model from the command line or from an IDE such as spyder. To run from the command line, navigate to:

Enter python instantiate.py to run the model.

1. To run from an IDE, open the instantiate.py script, select all lines of the code and run the code.

### Overview of model parameters

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data format** | **Use** |
| Xlen, ylen | Integer | Set size of model grid |
| Start\_run, end\_run | Integer | Start and end of model run |
| AFTs | List of agents (AFTs) | Define model AFTs |
| LS | List of agents (land systems) | Define model land systems |
| Observers | Dictionary {str: agent (observer)} | Keys: Names of observer agents Values: Set observer agents |
| Fire seasonality | Dictionary {str: numpy array} | Keys: links seasonality to a managed fire type, must be in Fire\_types.keys() Values: Holds list of 12 grids, summing to 1, used to allocate annual fire outputs by month |
| theta | Scalar (0-1) | See free parameters doc |
| AFT\_pars | Dictionary {Complex} | Holds AFT, LS, Observer parameters, output of local\_load\_up |
| Maps | {str: 3-d masked array (time, ylen, xlen)} | Holds model forcing data sets |
| Fire\_types | Dictionary {str: str} | Keys: Set deliberate fire types Values: Assign fire type to land cover type |
| Constraint pars | Dictionary {str: numeric or Boolean} | Holds model constraints described in free parameters.txt – also below. |
| Defor\_pars | {str: numeric} | What fraction of deforestation uses fire; grouped by AFR |
| Seasonality | Boolean | True: monthly outputs  False: annual outputs |
| Escaped\_fire | Boolean | Should escaped fires be calculated? |
| reporters | List of strings | Sets what data should be recorded by the model as outputs |
| bootstrap | Boolean | Should classification trees be run with bootstrapped parameter distributions? |
| N\_cores | Integer | If bootstrap = True, how many cores should be used to run bootstrapped model parameters |
| Write\_annual | Boolean | Should model outputs be written to disk each timestep? |
| Write\_fp | Str (regex) | Filepath of where results should be written |

### Description of model free parameters

Theta

What : zeroing out for anthropogenic fire regimes, analogous to CRAFTY's “giving in”

Default: 0.1 (arbitrary - from testing)

Soil threshold

What : Value above which bare soil constraint on fire is applied

Default: 0.1325 (global mean)

Dominant afr (intensive) threshold

What : Value of intensive AFR above which fire exclusionary constraint kicks in

Default: 0.5 (arbitrary)

Rangeland stocking constraint

What : should rangeland fire be impacted by the degree of stocking rate?

Default: True

R\_s\_c\_Positive

What : should rangeland stocking also be able to increase fire (overstocking)?

Default: False

HG\_Market\_constraint

What : prevents hunter gatherers from burning in very wealth peri-urban areas

Default: Market influence > 7800 (empirical, 95th percentile)

Arson\_threshold

What : states at what level market access effects on arson should kick-in

Default: 0.5 (arbitrary)

## Model Analysis Tools

1. Accessing model results

Model outputs included in the reporters argument are stored as a list of dictionaries in WHAM.results. E.g. to access total Managed fire from year 1 –

WHAM.results[‘Managed\_fire’][0][‘Total’]

1. Visualising

The ‘basic visualise’ script has a useful ‘map\_output’ function that takes a list input of dimensions ylen\*xlen – so, e.g

map\_output([x[‘Total’] for x in WHAM.results[‘Managed\_fire’]])

1. Writing out

The ncdfwriter script can be used to write out files