# Guide to running and installing WHAM!

This document gives an overview of WHAM! v1.0. It covers installation and the fundamentals of setting up and using the model.

# Contents:

# 1.Installation & setup

# 2.Running the model

# 3.Model output and analysis tools

# 4. Codebase overview

# Installation & setup

## Download code and data

Code can be downloaded from the Zenodo reposistory:

<https://doi.org/10.5281/zenodo.8319431>

Similarly, data can be downloaded from:

<https://zenodo.org/records/8363979>

Alternatively, code can be downloaded from Github at:

<https://github.com/OliPerkins1987/Wildfire_Human_Agency_Model>

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

## Set up virtual environment

Instructions for setting up a virtual environment are provided, with specific commands given for anaconda. This has been tested for Windows, MacOS and Linux. *Ubuntu users can…*

Package requirements are provided as a **wham38.yml** file for windows users, and a **requirements.txt** otherwise. These files are stored in the base directory of where you downloaded & unzipped the wham code.

* + 1. **Initialise environment**

Code is written in Python version 3.8. Please initialise and activate an empty python environment.

**Windows**

Windows users can either use the .yml file for convenience, or follow Mac / Linux instructions below. If using the .yml file, skip through package installation below, and move to installation of WHAM.

Option 1: .yml file

conda env create -f wham38.yml

conda activate wham38

Option 2: requirements.txt file

conda create --name wham38 python=3.8

conda activate wham38

**MacOS & Linux**

conda create --name wham38 python=3.8

conda activate wham38

* + 1. **Install packages**

If using windows and installing with requirements .txt, or using an alternative operating system, please install the requirements.txt file as below. This file is stored in the base directory of where WHAM was unzipped. Windows users who installed using wham38.yml should skip this stage.

pip install -r requirements.txt

## 1.3 Install wham code

Installation of wham code is done using python install. If using anaconda, install the code by navigating to the filepath where you unzipped the code, and type:

python setup.py install



## 1.4 Data download and set up

Download data from zenodo:

<https://zenodo.org/records/8363979>

Linking the data and model

1. Open the ‘local\_load\_up.py’ script in the src/data\_import directory in the source code
2. 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/)

* NB ensure the file paths both end in a trailing forward slash - e.g. - ‘…/mypath**/**’

1. save this version over the downloaded version

## 1.5 Testing

To check that has all worked, navigate to /tests in the code files type & enter ‘pytest’



This takes around 15 mins to run on a medium performance desktop. There should be many warning messages about dividing by zero – this is where the land fraction equals 0 and is fine – & warnings about np.bool being deprecated – it is used by the nectCDF4 package & that is fine. Any test failures need to be explored. Please report them to [oliver.perkins@kcl.ac.uk](mailto:oliver.perkins@kcl.ac.uk)!

# Running the model

### Set up instantiate.py

The model is run by the instantiate.py script in src/model\_interface/

The instantiate.py script has a parameter – write\_fp on line 113 that specifies where you would like output files to be written. This happens by default each model year (see 2.3 below). This needs to be set before the script will run correctly. Additional model parameters are described in section 2.3.

### Run model (standalone)

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 /src/model\_interface and enter:

python instantiate.py

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

*(Continues below)*

## 2.3 Overview of model run options

Model run options are set in the parameters dictionary in the instantiate.py script

**Table 1: Description of model parameters**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data format** | **Use** |
| Xlen, ylen | Integer | Set size of model grid |
| Start\_run, end\_run | Integer | Set start and end of model run (Year1 = 0) |
| AFTs | List of agents (AFTs) | Declare model AFTs |
| LS | List of agents (land systems) | Declare model land systems |
| Observers | Dictionary {str: agent (observer)} | Keys: Set names of observer agents Values: Declare 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: Declare managed fire use types Values: Assign fire type to land cover type |
| Constraint pars | Dictionary {str: numeric or Boolean} | Holds model constraints on managed fire use – 2.4 below |
| Defor\_pars | {str: numeric} | Declares 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

As primarily an empirical model, WHAM! has few free parameters. Those that do exist primarily relate to constraints on managed fire, which are set in the ‘constraints’ key of the parameters dictionary.

2.4.1 Theta

What: Defines the point at which the fraction of a cell occupied by an anthropogenic fire regime is pushed to 0; the area of the cell < Theta is then reassigned to proportionally to other AFRs.

Default: 0.1 (From trial and error)

2.4.2 Bare soil threshold

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

Default: 0.1325 (global mean of bare soil fraction)

2.4.3 Dominant afr (intensive) threshold

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

Default: 0.5 (Half or more of the cell occupied by intensive AFR)

2.4.4 Rangeland stocking constraint

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

Default: True

2.4.5 R\_s\_c\_Positive

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

Default: False

2.4.6 HG\_Market\_constraint

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

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

2.4.7 Arson\_threshold

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

Default: 0.5 (Trial and error)

# Model outputs and analysis tools

If Write\_annual was set to True, the model outputs are saved as netcdf files of same dimension as specified by the xlen and ylen model parameters.

### 3.1 Accessing model results

Model outputs included in the reporters argument are stored as a list of dictionaries in WHAM.results. Each level of the list relates to a model year, and the keys of each dictionary to a model output. E.g. to access total Managed fire from year 1 (zero indexed) –

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

The utilities folder also has several useful functions for gathering model results from the reporters dictionary in the WHAM object.

### 3.2 Visualising

The ‘basic visualise’ script in the /visualisation folder has a useful ‘map\_output’ function that takes a list input comprising numpy arrays output stored in the .results method of the WHAM model object.

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

### 3.3 Writing out

The ncdfwriter script can be used to write out files if the ‘write\_annual’ argument in model parameters is set to False

# Codebase overview

This section gives an overview of the structure of the codebase within the /src folder.

## Core\_functionality

As the title suggests, this is where the majority of the architecture of WHAM is stored.

The **/AFTs** folder contains two central classes: the AFT (agent\_class) which is used as the basis of all agents in the model, and the land system class, which is the container for cropland, pasture, rangeland, forestry, etc. Other files here then give the AFTs & land systems, with associated parameters defined in WHAM v1.0.

**/prediction\_tools** contains utilities for running the sub-models for either AFT distribution or fire uses

**/top\_down\_processes** includes all landscape level processes. These include arson, background ignitions and all fire constraints (e.g. vegetation and fire suppression constraints).

**/Trees** contains functions for predicting from tree model frames. These are a simple way of encoding a classification or regression tree in a table storable as a .csv. They are produced by the tree package in R.

## Data\_import

This contains the data load up script. A sharepoint api has been developed, but is not included in version 1.0 (see development branch on Github).

## Experiments

This contains experiments run in the initial model description paper (Perkins et al., submitted). All\_parameter\_run.py is used to store separate outputs from each of the samples of the model’s numerical distributions for its parameters. LC\_90.py and LU\_90.py run the model with either landcover, or land use intensity, held constant at 1990 levels. Sensitivity\_analysis.py can be used to run the model with multiple values for its free parameters.

## Model\_interface

This contains the core model class (**wham.py**). This is structured as per a Net Logo agent-based model. IE: it has a setup and go method that define the initial conditions and duration of a model run. The model scheduler (.step method) orders the code to be executed at each timestep.

Instantiate.py (described above) is the script used to execute the model.

## Output\_analysis

Contains some utilities for gathering model results from the WHAM.results container, and for writing out as netcdf files.

## Visualisation

Contains a utility for visualising model map outputs.