

# EGGS-WG: Operations Manual V1.2.0

Alex Schuddeboom  
alex.schuddeboom@canterbury.ac.nz

November 2025

## 1 Overview

The EGGS-WG program is a stochastic weather generator that operates using the ERA5-Land data. This program is written in python, with a detailed description given in the published paper related to this project. This document exists to guide the operation of the program without providing an extensive background on the design of the model. This includes the process that is needed to follow to install the model, how to run the model, and how to download the underlying ERA5-Land data. For any issues that are unclear or unresolvable from this document, feel free to email me directly, and I will try my best to resolve the issue.

## 2 Program Structure

The program is set up so that no modification of the code is required for operation. Instead, the code is controlled through the usage of a YAML file. The parameters that control the simulation are set in this file, then the model loads in the file and alters the simulation based on the provided parameters. This file is discussed in detail later in the report. The model construction process is written to be as restartable as possible, meaning that if errors occur or computing resources need to be used for something else, the program can be exited and then resumed from the same place at a future date by just rerunning the program. The setup that is required to run the model is as follows:

- The model consists of several independent python and YAML files. In particular, the files needed are `SWG_Build_Functions.py`, `Main.py`, `SWG_Model.py`, `DependancyCheck.py`, `E5_DL.py`, `Parameters.yaml` and `environment.yaml`.
- The corresponding ERA5 land data and the associated geopotential file. These are available on the project shared drive.
- A valid python install with the supported packages. This will be installed by following the directions in the installation process section below.

## 3 EGGS-WG Installation

The installation process is relatively straightforward and summarized below.

1. Download the program files from the project Github.
2. Download and install anaconda (This is a widely used and freely available python management system from <https://www.anaconda.com/download> or any Linux system's native package manager). This step can be skipped if proficient with python and have a good grasp of the package handling process. Even for those familiar with python, I would strongly recommend using anaconda to simplify the handling of the python environment.
3. Open the anaconda program and launch the command line window through anaconda (normally either CMD.exe or PowerShell). This can also normally be done directly from the start menu (either anaconda prompt or anaconda PowerShell under the anaconda folder). On non-windows operating systems, this process will normally not work. Instead, if you are using a Linux system or an HPC, the regular terminal can likely be used in place of the launching a specific anaconda terminal.
4. run the command `conda env create -f "filename"` where "filename" is the full file name including the directory of the `environment.yaml` file included with the program. This should install all the requisite python packages for the operation of the simulation. Specifically, this creates

an anaconda environment named SWG, which contains all the packages necessary for the operation of the model. An example terminal output is given in figure 1 below. Note, this may take some time to complete (it is not uncommon for the solving environment step to take around 20 minutes).

```
(base) C:\Users\Alex\Desktop-C7QL2QR>conda env create -f "D:\Model V3\environment.yml"
Retrieving notices: ...working... done
Collecting package metadata (repodata.json): done
Solving environment: done
```

Figure 1: Example of the code installing the anaconda environment. Note, there is more text output in the terminal, but I have cut it short for brevity.

## 4 EGGS-WG Operation

With the python environment established and all the prerequisite packages installed, running the program is a relatively simple process. This can be done by following the process below, with an example figure from a simulation performed on my pc included:

1. Open the command prompt through the anaconda program as described in the installation section. Again, if you are using a Linux system or an HPC system, it should be possible to do through the regular terminal.
2. Activate the anaconda environment using the command ”conda activate SWG”. This loads in all the needed python packages for the simulation process.
3. Change the directory to the directory which contains the python model code. This is done with the command “cd directory” or “cd /d directory” if you need to change to a different hard drive. Commands may need to be changed for non-windows systems but, it should be simple to find the alternative commands.
4. Run the command “python Main.py”. Note this command will take a long time to run, but the progress is gradually saved, so the program can be exited without losing all the progress. Rerunning the same commands will allow the model creation process to pick up from where it was last time it was running.

Alternatively, the program can be run by specifically designating the location of the yaml file. This is done by adding an extra argument to the terminal command, indicating the location of the yaml file. For example, “python Main.py” is replaced with ”python Main.py /path/to/yaml/parameters.yaml”.

```
(base) C:\Users\Alex.DESKTOP-C7QL2QR>conda activate SWG
(SWG) C:\Users\Alex.DESKTOP-C7QL2QR>cd /d "D:Model V1"
(SWG) D:\Model V1>python Main.py
```

Figure 2: Example of code execution that runs the simulation.

Generally, this process is relatively simple and will take a different amount of time depending on the size of the domain and the computer the code is executed on. For small domains, I have seen model development times that are below 3 hours, whereas for longer domains it may take days for this process to complete. Statements are output continuously through the simulation building process that can be used to pinpoint the locations of code if issues arise in the simulation process.

### YAML file

The YAML file sets the parameters that are used by the model to generate the simulation. A detailed description of this file and each of its variables is included below. Figure 3 shows the basic structure of the YAML file and the 11 different model parameters included within. All the 11 parameters identified in the figure should be set for any model run.

```
Simulation name: 'Canterbury Test Region'
Simulation Start Year: 1990
Simulation End Year: 2100
Number of Simulations: 1
ERA5 Directory: 'C:\Users\asc182\Full NZ DL\
Regenerate Data: False
Data Output Directory: 'C:\Users\asc182\SWG Runs\
Lat Bounds: [-43.2, -44.01]
Lon Bounds: [171.9, 172.8]
Variable Adjust: True
Non-stationary Mode: False
```

Figure 3: Example of the yaml file which guides the simulation.

The different variables that need to be included in any given simulation are:

- Simulation name: The name given to the simulation. This determines the name of the folder which the simulation will be saved under. This provides an easy way to separate out each of the different simulations.

- Simulation Start year: Set the start year for the simulation.
- Simulation End year: Set the end year for the simulation.
- Number of Simulations: The number of times the simulation should be repeated.
- ERA5 Directory: The local directory of the ERA5 land data, which is used to create the model. In most operating systems, this will need to either to be locally downloaded or directly available over the network. This data is available on the shared project drive online.
- Regenerate Data: This is a boolean flag which indicates if all existing data needs to be over-ridden. This **does not** impact the data generated by the simulation (this is always regenerated), but instead is for the data used to run the simulation.
- Data Output Directory: The location for the data to be saved. Specifically, this is the folder where a folder will be created which has all the output associated with the simulation stored within.
- Lat Bounds: Identify the latitude boundaries of the simulation domain.
- Lon Bounds: Identify the longitude boundaries of the simulation domain.
- Variable Adjust: This is a boolean flag that determines if the post simulation adjustments are applied to the precipitation and temperature values. In general, I would recommend this variable be set to True unless there is a very clear reason to do otherwise.
- Non-stationary Mode: This is a boolean flag that tells the model if it should evolve with time or not. The default is that this variable is set to False which makes each simulation year set to equivalent conditions. When set to True the model will replicate the historic temperature trends seen in the ERA5-land data with a flat temperature until 1950, then rising temperatures and remaining steady after 2020.

## 5 ERA5-Land Downloads

A local or network version of the ERA5-Land data needs to be available for the model to work. Included with the files is a python script, `ER_DL.py`, which can be used to download the ERA5-Land data directory. This can be done with the following process:

1. Create a new Conda environment and install the numpy and cdsapi packages (`conda install conda-forge::cdsapi`). This could be done in an existing anaconda environment, it is just considered best practice to isolate the python packages to different environments.

2. Follow the standard instructions for operation of the api on the CDS website <https://cds.climate.copernicus.eu/how-to-api> for registering an account, saving a local copy of the api key and agreeing to dataset terms and conditions.
3. Alter the `ER_DL.py` script as required for the download process. Note, this is initially configured with the naming structure and file structure expected by the model, with the only variable that will likely need to be changed being the area over which the ERA5-Land data is requested.
4. Call the `ER_DL.py` through the terminal with the command "Python `ER_DL.py`"

The model has been built with a very particular expectation of the formatting and naming conventions of the ERA5-Land data. These are relatively simple with all the required files just being placed in the single overarching directory as shown in figure 4. Each of these grib files follows a specific naming convention, specifically each file needs to be named `ERA_Land_Multi_var_YYYY_MM` where Y values are replaced with a four-digit year value and M is a two-digit month value.

		1,754 items	22/08/23 at 10:48 AM
└──	Full NZ DL		
└──	? ERA_Land_Multi_var_1950_01.grib	54.4 MiB	29/05/23 at 3:23 PM
└──	? ERA_Land_Multi_var_1950_02.grib	49.2 MiB	29/05/23 at 3:38 PM
└──	? ERA_Land_Multi_var_1950_03.grib	54.5 MiB	29/05/23 at 3:55 PM
└──	? ERA_Land_Multi_var_1950_04.grib	52.7 MiB	29/05/23 at 4:10 PM
└──	? ERA_Land_Multi_var_1950_05.grib	54.5 MiB	29/05/23 at 4:25 PM

Figure 4: Example of the folder structure for the ERA5-Land data.

There also needs to be a geopotential file included with the other files named `ERA_Geopotential.nc`. This should just have the invariant ERA5 geopotential variable included, and can be downloaded directly from the cds website.