# Heat Stress Code Documentation

Code files (.R) are kept in the project’s root directory and are numbered sequentially e.g. 01\_wbgt\_create\_climate\_rasters.R, 02\_wbgt\_create\_climate\_dataframes.R…

Scripts are numbered in the order that they should be run if you wanted to *completely reproduce this analysis from scratch* using the original ISIMIP3b climate data. In reality, this would be a long and tedious process due to the amount of time it takes to download the raw data files and convert them (using 01\_wbgt\_create\_climate\_rasters.R) into the format required by the HeatStress package. Therefore, I recommend that new users instead download the *processed* ISIMIP3b data from the Github repo and proceed with script ‘02\_’ onwards. ***Only use script 01\_wbgt\_create\_climate\_rasters.R if you wish to fully recreate the climate data!***

All geospatial code is in WGS84 (EPSG: 4326) unless otherwise stated.

Data downloaded from ISIMIP3b are all bias-corrected, near-surface variables and include: Mean Air Temperature, Maximum Air Temperature, Relative Humidity, Wind Speed, Downwelling Shortwave Radiation.

This analysis uses the [HeatStress](https://github.com/anacv/HeatStress) package, developed in the framework of the [Horizon2020 HEAT-SHIELD](https://www.heat-shield.eu/) project. Various WBGT methods are included in the package and require varying combinations of the above climate variables in order to run.

## 00\_wbgt\_create\_coordinates.R

This script is a data preparation script which sets the area of extent for the whole analysis. You must run this analysis the first time you run the code on your machine and *whenever you wish to change the study area’s extent.*

In short, this script takes an input shapefile which defines your study area and creates output dataframes of cell centroids and kiln locations. Without running this script, the rest of the scripts in the repository won’t ‘know’ where your study area is.

## 01\_wbgt\_create\_climate\_rasters.R

This script is used to process the *original* ISIMIP3b climate data. These data are NCDF4 files and are global in extent. Script 01 therefore crops and masks the global extent rasters such that they only extend across India and only cover terrestrial areas. At ISIMIP’s 0.5x0.5 resolution, India is represented by 1,302 cells across the landscape.

Original ISIMIP3b data is supplied in (mostly) decadal blocks with the time span taking the form of 2021-01-01 to 2030-12-31 (note ISO 8601 date format). In the interests of cleaner code and input files, I made the decision to stack 3 decade’s worth of ISIMIP .nc files into a single rasterstack which is 10,957 layers ‘deep’. This may become impractical if the Lab wishes to extent the analysis to, say, 2100 (the stack will be very deep at that point!) For now, however, it means that each climate variable’s output raster covers the extent of India and has a layer for each day between 2021 and 2050 (inclusive.)

I have written a small function which creates the cropped, masked and stacked raster files. This function must be run for each climate layer, GCM and emissions scenario. Therefore, for each GCM (n = 5, e.g. GFDL-ESM4M) the function will need to be run 12 times; for each climate variable (n=4) it needs to be run for the three different emissions scenarios (SSP126, SSP370, SSP585). The function outputs the climate data into the project’s root directory and it is up to the user to put them into a sensible directory structure. I recommend something like "output/climate\_data/gfdl-esm4/spp126/hursAdjust\_gfdl\_2021\_2050\_ssp126.nc"

This code could be refactored into a single function to reduce verbosity. However, I felt that for the present analysis, it would be useful to provide overly verbose code for the sake of clarity.

## 02\_wbgt\_create\_climate\_dataframes.R

This script takes the cropped ISIMIP3b climate data and returns tibbles for each point across the landscape using the function create\_wbgt\_df(). Running this function creates a list of length 1302 (one element for every point across India at 0.5x0.5 resolution ISIMIP3b climate data). Each element in the list is a tibble which has 10,957 rows – each row represents 1 day across the 2021 – 2050 time period. Each element contains the following data: date, mean temperature, maximum temperature, dewpoint (calculated from frost::calcDewPoint(), relative humidity, windspeed and shortwave solar radiation.

The outputs from create\_wbgt\_df() are used in 03\_wbgt\_run\_models.R to run the various WBGT methods.

## 03\_wbgt\_run\_models.R

This script is primarily responsible for running WBGT models from the climate dataframes generated from 02\_wbgt\_create\_climate\_dataframes.R. All logic is wrapped in the function run\_wbgt(). When running the Bernard and Liljegren models, the function executes the model runs in parallel to reduce computation time. Note that the Liljegren methodology is complex and thus takes longer to run. Whereas the Stull (2011) model will run across the entire study extent in a few seconds, Liljegren (2008) takes approximately 45 minutes (on a 2020 MacBook Air M1 with 16GB RAM.)

run\_wbgt() must be run for every combination of GCM, climate scenario and WBGT model (on the assumption you’re interested in all combinations.) There are 5 GCMs, 3 climate scenarios and 3 WBGT models leading to 5\*3\*3 = 45 calls to run\_wbgt()

This script is designed to generate two different ‘kinds’ of dataframe. The first (e.g. gfdl\_ssp126\_stull) is a list of length 1302; an element for every point across the landscape. Within each element there’s a tibble of length 10,957; a row for every day in the year from 2021 – 2050.

The second dataframe (e.g. gfdl\_ssp126\_stull\_trs) is a *transformed* dataframe which binds together all of the rows within each element from e.g. gfdl\_ssp126\_stull. It is 14,266,014 rows long (i.e. 1302\*10957=14,266,014). This “mega dataframe” has been created so that it’s easier to group by days/months/years when subsequently analysing WBGT the data. It would of course have been possible to loop over every element in gfdl\_ssp126\_stull and work from that. But the study is small enough that it is computationally practical to work on 14-million-row dataframes in memory. Given the choice, I opted for the ‘easier’ data structure!

Whether looking at gfdl\_ssp126\_stull or gfdl\_ssp126\_stull\_trs, you’re simply looking at a tibble with information on the wet bulb temperature. When running the Stull (2011) implementation, this is the only data that’s returned (aside from data which is present in the input data i.e. the date and coordinates of each cell across the landscape.)

When looking at Bernard (1999) or Liljegren (2008) you get a lot more output. In the case of the former, you will see wet bulb temperature and phychrometric wet bulb temperature. In the latter you will see wet bulb temperature, natural wet bulb temperature and globe temperature.