GCAM landuse downscaling

Readme file: general documentation of the code and support to run it.

Contact: Yannick Le Page -- [Yannick.LePage@pnnl.gov](mailto:Yannick.LePage@pnnl.gov)

A. Overview 2

B. Folders/files structure 4

C. Running the code 4

C.1. Software requirements 4

C.2. Running 5

D. Overall code structure 6

E. User inputs 6

E.1. General parameters 6

E.2. PFT harmonization and LUC transition rules file 9

E.2.1. Aggregation of the spatial data to the common PFT class scheme 9

E.2.2. Aggregation of the GCAM data to the common PFT class scheme 10

E.2.3. Transition priority rules 11

E.2.4. Treatment order 12

F. Downscaled land use files 12

G. Examples of outputs and diagnostic 13

G.1. Outputs/RunName/DiagLevel1/areacoef.csv 13

G.2. Outputs/RunName/DiagLevel2/ 14

G.3. Outputs/RunName/KernelDensity/ 15

G.4. Outputs/RunName/Maps/LUC/ 17

H. Version 2 20

H.1. Downscaling all 12 GCAM crops separately 20

H.2. Spatial constrains to land allocation 22

H.3. NETCDF outputs 23

H.4. Stochasticity of expansion 24

H.5. Sensitivity of performance to the parameterization 24

# Overview

The downscaling system spatially allocates GCAM Region/AEZ land use to a grid (Figure 1).

|  |
| --- |
| Macintosh HD:Users:lepa724:Documents:Programs:GlobalDownscaling:README:Figures:AEZ_level_crops.png  GCAM Region/AEZ-level crops 2005 |
|  |
| Macintosh HD:Users:lepa724:Documents:Programs:GlobalDownscaling:Outputs:test2KateData:Maps:LUC:Total_LUC_crops2005.png  GCAM gridded crops 2005 |

Figure . Illustration of the downscaling: GCAM crop area at AEZ/Region scale (top) and it distribution on a 0.05degree grid.

In this first version, the downscaling is based on:

1. Base year information on land use distribution from satellite observation (MODIS)
2. User-defined allocation rules:
   1. Allocation of a given increasing PFT on grid-cells where it already exists (e.g. crop area increases in current agricultural regions). This is referred to as **intensification**.
   2. Allocation of a given increasing PFT on grid-cells where it doesn’t yet exist, by proximity (e.g. crop area increases close to current agricultural regions). This is referred to as **expansion**.
   3. Land use transition priorities: a given PFT preferentially replaces specific PFTs, in the case these are projected to decrease (e.g. crops expand preferentially into grasslands, then into shrublands, then into forests).
   4. Treatment order: PFTs are downscaled one after the other, so there must be an order, which influences the results. For example, we can start downscaling urban, then croplands, then natural PFT classes.

All of these rules are flexible, and users can also decide the ratio of intensification versus expansion they ideally want. Most of the parameters can be defined separately per PFT and per Region/AEZ (e.g. you might want to expand crops preferentially into grasslands in the US, but into forests in tropical regions).

The method of this first global version has similarities with what was developed for the US-only downscaling algorithm, which is described in the following publication:

*West, T. O., Le Page, Y., Huang, M., Wolf, J. and Thomson, A. M. (2014) Downscaling global land cover projections from an integrated assessment model for use in regional analyses: results and evaluation for the US from 2005 to 2095, Environmental Research Letters, 9(6), 064004.*

The way the code works, both **landuse patterns** and **landuse transition matrices** can be obtained. Landuse patterns simply describe PFT distribution at a given timestep, while landuse transition matrices describe all transitions, i.e. which PFT expands into which other PFT, between 2 timesteps. Transitions are important for many aspects, including the carbon cycle, and some Earth system models can use them.

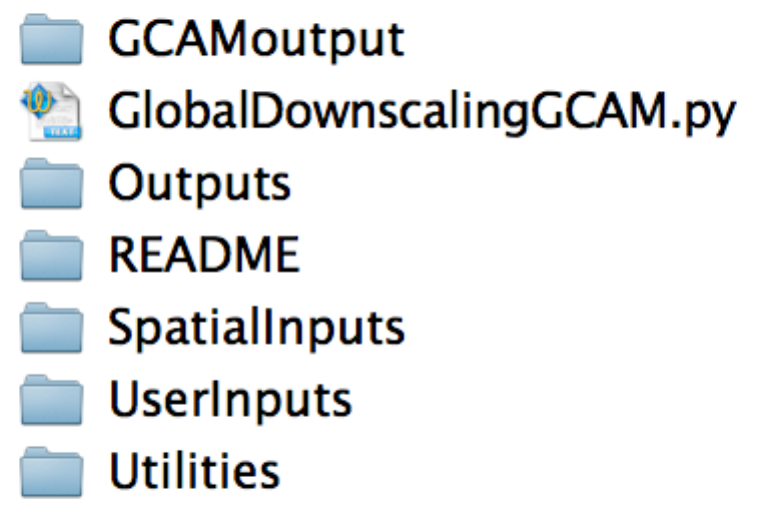
Note that both spatial patterns and transitions are projected based on **NET** land use change, as opposed to **GROSS** land use change. That is, say a region sees a net increase of 100km2 in crops over a 5-years timestep. Crops may have actually increased by 150km2 in the first 3 years, then decreased by 50km2 in the last 2 years. Similarly, crops might have increased by 200km2 in the western half of the region, while decreasing by 100km2 in the eastern half. All these scenarios have a NET change of 100km2, but different land swaps and different implications for ecosystems, carbon and spatial PFT distribution. This is a very complex aspect to model though and we do not attempt to do so for now.

Ongoing/expected developments include:

* Integration of agricultural suitability to the crop allocation algorithm. (**done in version2**).
* Integration of protected areas as a constrain to landuse expansion.
* Disaggregation of crop types and practices (e.g. corn, rice, irrigated, rainfed). (**crops types done in version2**).
* Anything that might be needed, suggestions are welcome (e.g. shifting cultivation).
* In terms of code and format: let me know if you’d like specific features (e.g. colorbars, MODIS inputs at lower resolution to run the code more efficiently but still be of use for ESMs, other diagnostic statistics) or exporting in different formats (e.g. netcdf). **Netcdf done in version2**.

# Folders/files structure

The downscaling system has 6 main folders:



The file *GlobalDownscalingGCAM.py* is the code, the folders contain all the necessary information to run it (e.g. input data, parameters).

* GCAMoutput: contains the GCAM landuse file. Typically exported to excel through the GCAM model interface, then saved into a .csv file. **!!! scenario names seem to have commas in them (before the date/time). If so, they should be removed (easy with find and replace all) before saving as .csv (=comma separated file…), otherwise scenario name is split.**
* Outputs: This folder will contain a sub-folder with the name you give to the run. That sub-folder will contain the downscaled landuse outputs, a copy of the code and of the parameter file used for the run, and a number of diagnostic outputs.
* README: support files.
* SpatialInputs: contains the spatial data (e.g. MODIS landcover, agricultural suitability) used to drive the downscaling.
* UserInputs: The *Downscaling\_params.xls* file contains general user-defined parameters (e.g. resolution, timespan, diagnostic outputs and maps desired, run name, etc). It also contains the name of a second parameter file, by default named *Landclass\_harmonization.xls,* where the user defines PFT aggregation rules (e.g. all crop types into a single crop PFT), transition priority rules (e.g. crops expanding preferentially into grasslands rather than forests), etc.
* Utilities: contains stuff used in the code and for outputs/diagnostic (e.g. country borders for mapping).

# Getting the downscaling system

The code and most associated files are located on github for development and version control: <https://github.com/JGCRI/GCAMLU>.

However, files in the “*SpatialInputs*” folder are located elsewhere because of their size. **To run the downscaling, you need to download the zip archive to the same location as the other folders (see section B), and unzip it.** The zip archive is publicly available here: <https://drive.google.com/file/d/0B0eVUyq8yOojblpZQ2V2ZnU3R2M/view?usp=sharing>

# Running the code

## Software requirements

You will need a python installation (tested on Python 2.7) with the numpy and scipy modules. These modules generally come with the default python installation that every MAC computer has. You also need the xlrd module to read the excel parameter files.

If you turn on mapping diagnostic options, you will need to install another python module, called matplotlib. Pretty easy to do, here are 2 python guides with step-by-step directions on such things: <https://wiki.python.org/moin/BeginnersGuide> and <http://docs.python-guide.org/en/latest/>

## Running

Open a terminal window, cd to the GlobalDownscaling folder and type:

Python –i GlobalDownscalingGCAM.py

The downscaling system works as is, with default user-defined parameters, but these might not be adapted to your needs.

**Next sections below describe the code and all these parameters.**

# Overall code structure

Here is a “pythonized” summary of the code:

**1. Initialization**

# Reading in data/parameters

# Harmonizing GCAM and MODIS for total land area, for each Region/AEZ

# Aggregation to final PFT classes

**2. Main code**

**For each timestep:**

**For each PFT:**

# Computes kernel density maps

**For each Region**:

**For each AEZ**:

**For each PFT**:

**If PFT increasing:**

# **Intensification** (increase PFT area on grid-cells where it exists), based on transition priority rules, and on the intensification/expansion ratio (80% intensification by default).

**For each Region**:

**For each AEZ**:

**For each PFT**:

**If PFT increasing:**

# **Expansion** (increase PFT area on grid-cells where it does not exist), based on transition priority rules, and on the intensification/expansion ratio (20% expansion by default).

**For each Region**:

**For each AEZ**:

**For each PFT**:

**If PFT increasing:**

# **Second round of intensification**. Sometimes, you cannot find enough area through the 2 first rounds. That is, the code might not be able to fulfill the 20% expansion, because it runs out of land to spare on such grid-cells. So whatever is left has to be done through intensification again.

# User inputs

## General parameters

E.g. *Downscaling\_params.xls*

Here are the parameters as they appear in the file *Downscaling\_params.xls*:

|  |  |
| --- | --- |
| **Parameter Name** | **Parameter Value** |
| outpath | ./Outputs/RunName/ |
| resin | 0.05 |
| errortol | 0.01 |
| printlevel | 2 |
| RootPath | ./ |
| LUfile | GCAMoutput/Landuse.csv |
| firstMODfile | SpatialInputs/Landcover/Gridcode\_Land\_AEZreg\_coord\_all.txt |
| PFTharmonization | UserInputs/AllocationRules/Landclass\_harmonization.xls |
| yearB | 2005 |
| yearE | 2005 |
| timestep | 5 |
| scenario | SPA4\_26date=2014-9-11T16:17:55-05:00 |
| GCAMregnamefile | ./Utilities/GCAM\_region\_names\_32reg.csv |
| diagnosticsavelevel | 2 |
| intensification\_ratio | 0.8 |
| map\_kernels | 0 |
| map\_LUC\_steps | 1 |
| map\_LUC | 1 |
| kerneldistance | 5 |
| map\_tot\_LUC | 1 |
| save\_netcdf | 1 |

And their description:

**Outpath**: the new folder were all outputs/diagnostics will be written, which will be located in the *Outputs* main folder. In the example, “RunName” is the new folder. The code will either create that sub-folder, or over-write any files that are in there from previous runs with the same run name.

**resin**: the resolution of the spatial downscaling. This is flexible in the code (the downscaling will work at any resolution), but you need base-year spatial data with the corresponding resolution. As of now there is the 0.05 and 0.25degree data.

**errotol**: error tolerance, in km2. When downscaling, the code will stop once the PFT area change that hasn’t yet been done is equal to that errortol value in any Region/AEZ. Default is 0.01km2. The lower, the longer it takes to run, but not that much actually, there are other things that have a bigger impact (e.g. generating diagnostic maps, kerneldistance).

**printlevel**: hierarchical levels of on-screen printing of some information while the code runs. Basically levels <=2 will let you see progress (initialization, data reading, years, regions, etc). Levels >=3 are more for debugging, and will print quite a lot in your terminal window (AEZ, each PFT area, etc.).

**RootPath:** the root path to print your outpath (e.g. to which is added the outpath folder). Typically “./”, that is where the downscaling code is, but you can change it.

**LUfile:** the GCAM landuse file as exported from the model interface to an excel table, and saved as a .csv file. Remember to delete any comma in the model run name (see folders/files structure section).

**firstMODfile**: spatial landuse data file. MOD is for MODIS, but could be anything. The file contains landuse areas (in squared degrees) for each **spatial** PFT in the *PFTharmonization.csv* file, not for each **final** PFT: aggregation of satellite PFTs to final PFTs occurs in the downscaling code (e.g. all forest types of MODIS, evergreen, broadleaved, etc. are grouped into a single forest class for downscaling).

**PFTharmonization**: this is the file where users indicate a lot of stuff: PFT harmonization scheme, transition priority rules, etc. (see description of that file in Sect. E.2).

**yearB** and **yearE**: the first and last year of the downscaling. These can be equal to the first and last year of the GCAM output, or less (e.g. you might want to downscale only the first 20 years).

**timestep**: the timestep of the GCAM outputs (that will be automatized soon).

**scenario**: the name of the scenario you want to downscale. The model interface allows you to extract landuse for several GCAM runs (scenarios) at the same time. You can have several in the same .csv file input for downscaling (LUfile), but you can downscale only one scenario per code run. Scenario names are in the first column of LUfile.

**GCAMregnamefile**: the name of the .csv file with the correspondence Region code – Region name. This is used because GCAM outputs contain the region name, while the spatial data have a Region/AEZ code for each grid-cell. You probably don’t need to change it, unless you downscale the 14 region version of GCAM.

**diagnosticsavelevel**: not really necessary for running, I use it for development. Will save a table of the difference between total land area in MODIS (summing all grid-cells minus their water content) and in GCAM (summing all GCAM land categories) within each Region/AEZ. The discrepancies are a problem, because we do not expand on water (by default). So when MODIS has let total land than GCAM, we cannot spatially allocate all of GCAM land. Instead we scale it down. If MODIS has 98% of the GCAM area in a Region/AEZ, we multiply all GCAM land categories by 0.98. We do the same when MODIS has more land than GCAM (GCAM land is inflated).

**Intensification**\_**ratio**: this is a number from 0 to 1. It is the fraction of land use change that you would ideally want to happen on grid-cells where the considered increasing PFT already exists (intensification). 0.8 means the code will try to allocate 80% of the landuse change by intensification, and the rest, 20%, will occur through expansion. It’s not always possible to do as much as you want by intensification, so it’s a target, not a forced percentage.

**map\_kernels**: 1 or 0. If one, maps of the proximity value (kernel density) for each final PFT will be saved in the outpath folder. Has some impact on total run time. You need the matplotlib library for that (see software requirements section).

**map\_LUC\_steps:** 1 or 0. 1 to save global landuse change maps for each PFT, each timestep, and each allocation step within that timestep (intensification, expansion and final intensification, see Section D). You need the matplotlib library for that (see software requirements section). Has quite some impact on runtime and output folder size. Turn it on if you want to explore what the code those through the 3 allocation steps. If you just want to see the final allocation and change for that timestep, use the map\_LUC parameter instead (below).

**map\_LUC:** 1 or 0. 1 to save global landuse change maps for each PFT and each timestep. Has some impact on total run time but really worth it to check that everything went well, and for a first exploration of the outputs. You need the matplotlib library for that (see software requirements section).

**kerneldistance**: the radius (in grid-cells) you want the code to consider when computing the proximity value (kernel density). The bigger the radius, the smoother the kernel density values. Increasing radius has a big impact on total run time !

**map**\_**tot**\_**LUC**: 1 or 0. 1 to map the total landuse change for each PFT, globally. That is from the first timestep to the last, e.g. total landuse change from 2010 to 2100. It creates a map of the PFTs distribution in the first year, in the last year, and a third map of the change.

**save\_netcdf:** 1 or 0. 1 to save landuse in netcdf format. For each PFT, a single netcdf file is created, with annual gridded fraction. Annual is obtained by linear interpolation from one timestep to the other (e.g. linear interpolation from 2010 to 2015 to get 2011, 2012, 2013, 2014 landuse). **Files can get really big at high resolution !**

## PFT harmonization and LUC transition rules file

E.g. *Landclass\_harmonization.xls*

This is the file containing the aggregation and landuse change rules.

The file contains 4 sections, all in one spreadsheet.

### Aggregation of the spatial data to the common PFT class scheme

This is the part where you re-classify your spatial data (e.g. MODIS) into your final downscaled PFT classes. In the default scheme (Figure 2), there’s 10 MODIS PFT classes (the rows), and we re-class them to the final 7 classes (the columns, forest, shrub, grass, crops, urban, snow, sparse). For example, all 4 MODIS forest types are reclassified into the unique forest class that we want downscaled. Values are typically zero or one, meaning one class gets entirely attributed to one class. But you could have fractional value. If you have a mixed forest/crops class in the spatial data (some landcover products do), you could have half of its area attributed to forests, half to crops), by simply putting 0.5 in the forest column, and 0.5 in the crops column. Cells with “SPATIAL” and “SPATIALFIN” are needed at the top/bottom for fire reading in the code.

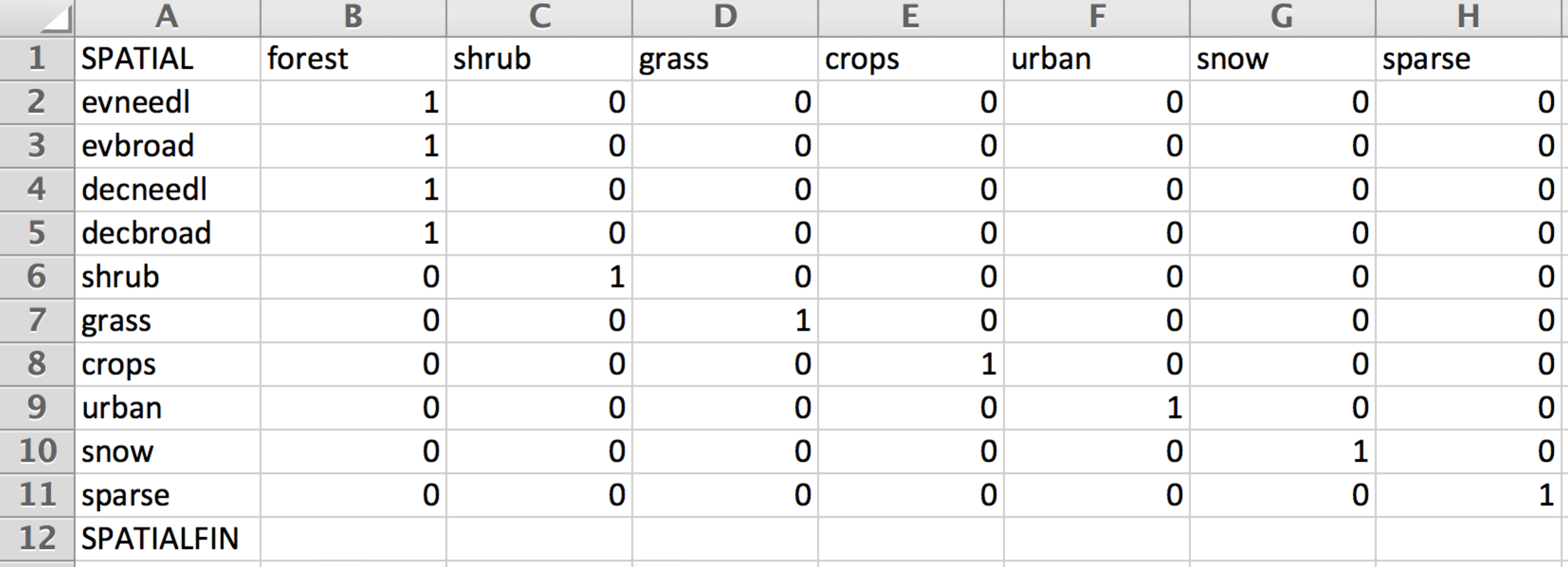


Figure . Aggregation of the spatial data PFT (rows) to the final PFT classes (columns).

### Aggregation of the GCAM data to the common PFT class scheme

This is the part where you re-classify your GCAM data into your final downscaled PFT classes. In the default scheme, there are 40 GCAM PFT classes, and we re-class them to the final 7 classes. For example, all the different crop types, whether irrigated or not, get aggregated to one single crop PFT. In contrast to aggregation of the spatial data, values are zero or one, not fractional. **BUT**, you can have 1 in several columns for a given GCAM class, meaning they will be attributed to 2 classes, but not with user-defined split shares. For example, RockIceDesert has a 1 in the final Snow PFT, and another 1 in the final sparse PFT(~desert in MODIS). So the GCAM IceRockDesert contains both, but we don’t know how much of each in a given Region/AEZ. The code will do the split equal to the Snow/Sparse share **observed** in the spatial data: some of the GCAM IceRockDesert class will go to snow in mountainous AEZs, but all of it goes to sparse in the Sahara.

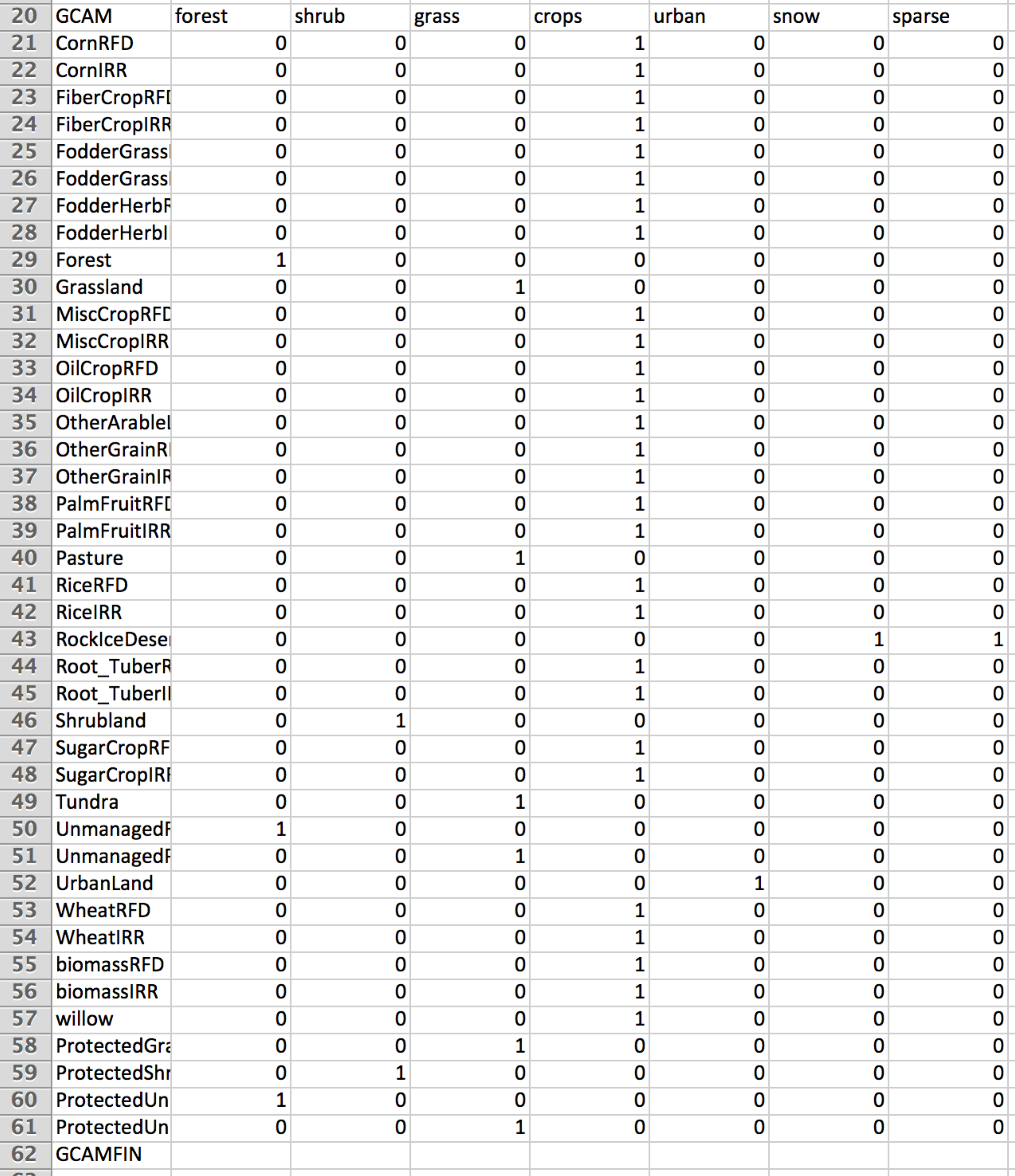
****

Figure . Aggregation of the GCAM PFT (rows) to the final PFT classes (columns).

### Transition priority rules

This is the part where you define land use transition preferences. For each PFT in the rows, you indicate what other PFT it would preferentially replace when its area is increasing. For example, when crop area increases (row 71), the code will first try to do that in urban areas. That’s just there for harmonization with the spatial data in the base year, urban area doesn’t change in GCAM projections. So because urban is fixed in GCAM, the code won’t find any available urban areas to convert to crops for future timesteps, so it will then go to number 2, which is grasslands. Say we are in a situation where grasslands are also projected to increase in the Region/AEZ, then we cannot transition any to crops (we do NET landuse change, not GROSS). So the code moves on to number 3, shrubs. Say shrubs are decreasing in that Region/AEZ, the code will transition as much as it can to crops, until it reaches the point when it reached the crop change, or when no more shrub area can be spared, in which case it moves to number 4 (forests), etc.

These rules apply both for the base year downscaling and for the future projection downscaling. They also apply to both the intensification and expansion phases.

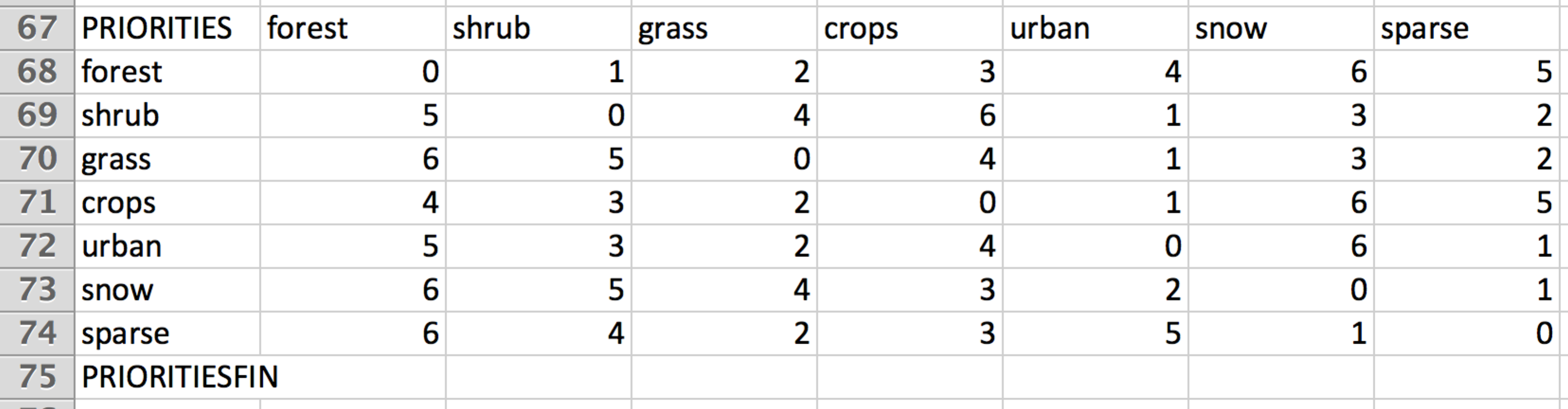


Figure . Land use transition priority rules.

### Treatment order

This is the part where you define the order in which you want to downscale the PFTs. The code downscales one by one (through each intensification and expansion steps), and that order will influence the results. If you downscale crops first, and it happens into grasslands, then that converted grassland area wont be available for say shrub expansion. By default, we start with urban, snow and sparse. For these PFT, that’s only valid for the base year again (when we allocate GCAM PFT areas in space), but not for projections since they are fixed categories. They are done first to avoid having to place them in whatever is left after downscaling all other classes. E.g. snow could have ended up in some low-lying areas instead of on top of mountains. I haven’t played too much with the order, don’t know how sensitive the outputs are.

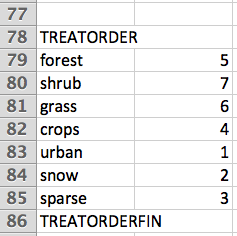


Figure . Treatment order

# Downscaled land use files

The code saves .csv files of the landuse for each timestep, in the *Outputs/RunName/Spatial\_LU folder*.

These files can be pretty big (more than 5 million grid-cells at 0.05degree resolution). Let me know if you’d like other formats (e.g. netcdf now available in version 2).

It includes the header of each column: GRIDCODE, water, evneedl, evbroad, decneedl, decbroad, shrub, grass, crops, urban, snow, sparse, regAEZ, Latcoord, Loncoord

|  |  |
| --- | --- |
| GRIDCODE | Code of the grid-cell, for re-integration into Arcmap and vizualization. Gridcode is the same as the original gridcode in the spatial data (e.g. MODIS). |
| water | Area of water, for each grid cell (squared degrees) |
| forest | Area of forest |
| shrub | Area of shrublands |
| grass | Area of grasslands |
| crops | Area of crops |
| urban | Area of urban land |
| snow | Area of snow |
| sparse | Area of sparse vegetation / deserts |
| regAEZ | Region/AEZ code (region\*100 + AEZ) |
| Latcoord | Latitude at the center of the grid-cell |
| Loncoord | Longitude at the center of the grid-cell |

# Examples of outputs and diagnostic

The results below are what you should obtain running the downscaling with the default parameters and with the original input files. I describe them in the order in which they are computed in the code (i.e. first the PFT aggregation and harmonization part, then the kernel density, then the downscaled outputs).

## Outputs/RunName/DiagLevel1/areacoef.csv

This file indicates the difference between the total land available in the spatial data and in GCAM. For example, summing up all terrestrial land (non-water) grid-cells in MODIS for AEZ 7 of the US, we get to 98% of the land in GCAM (Figure 6). This is a problem because we do not expand on water (although we could implement that). So in such a case, we harmonize GCAM total land area to match the spatial data. It’s thus not fully consistent with GCAM outputs, but we have to. All GCAM PFTs are thus multiplied by 0.98 in this example. The day we get a <1degree AEZ map should improve that quite a bit.

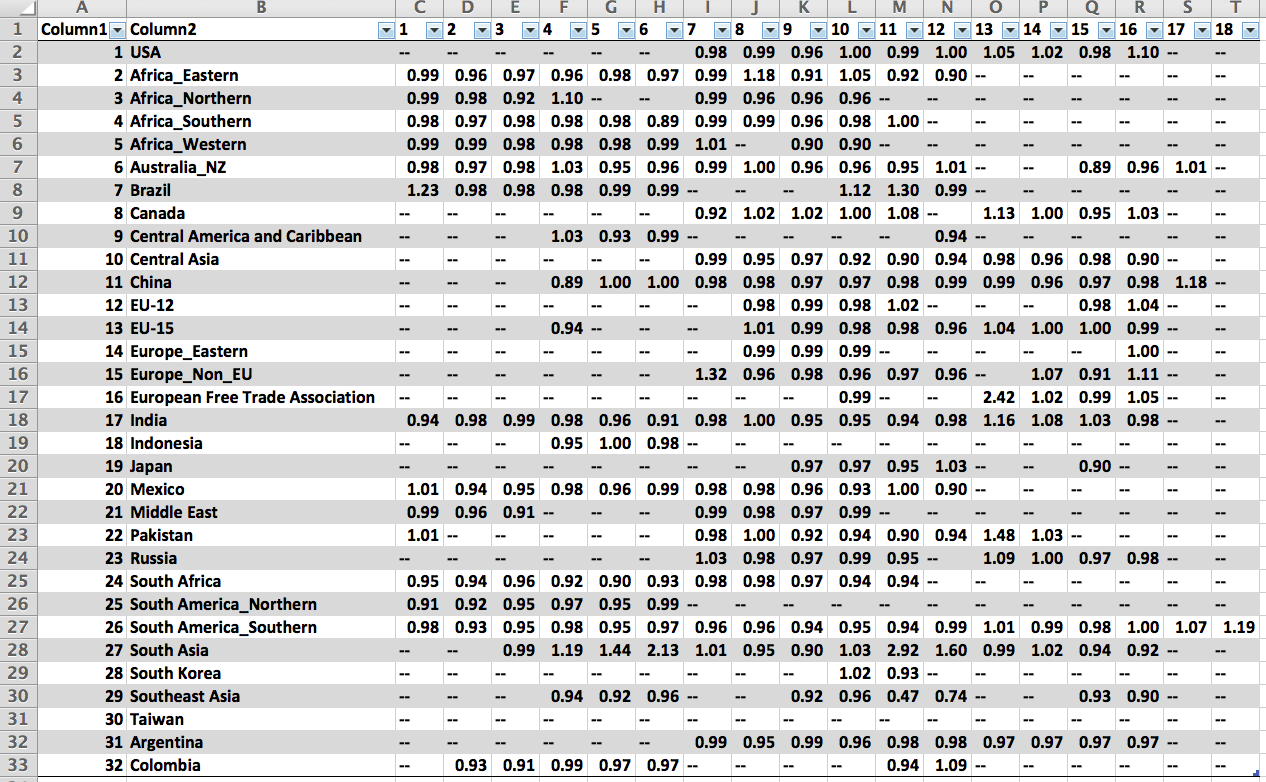
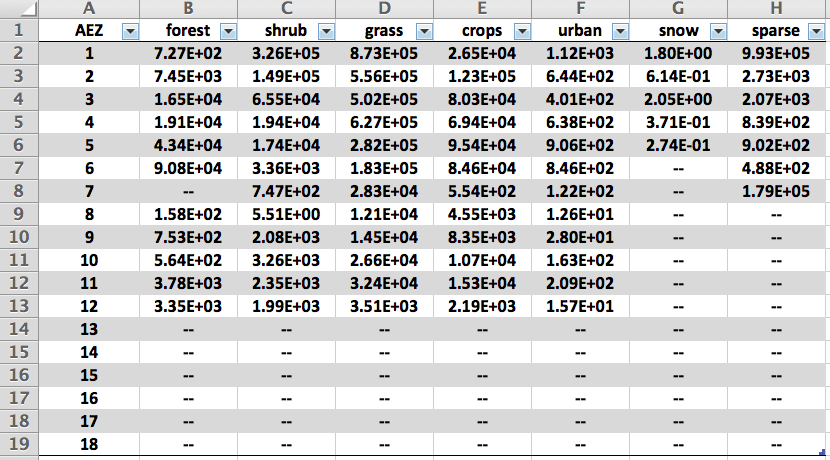
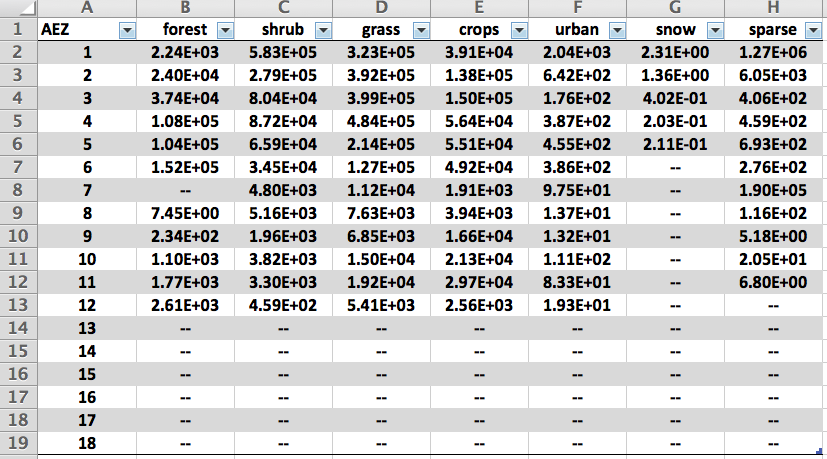


Figure . For each Region (rows) and AEZ (columns), areacoef.csv indicates the difference between the total land (non-water) area in the spatial data and in GCAM.

## Outputs/RunName/DiagLevel2/

That folder contains 2 files per region and per timestep. One shows the landuse area for each AEZ and each PFT class in GCAM in the timestep considered (Figure 7 top left), the other one does the same with GCAM PFT area from the previous timestep, or from the spatial data (MODIS) if it’s the first year (Figure 7 top left). From there you can easily compute the difference (the landuse change, Figure 7 bottom), which the code ultimately downscales following all the user-defined rules.

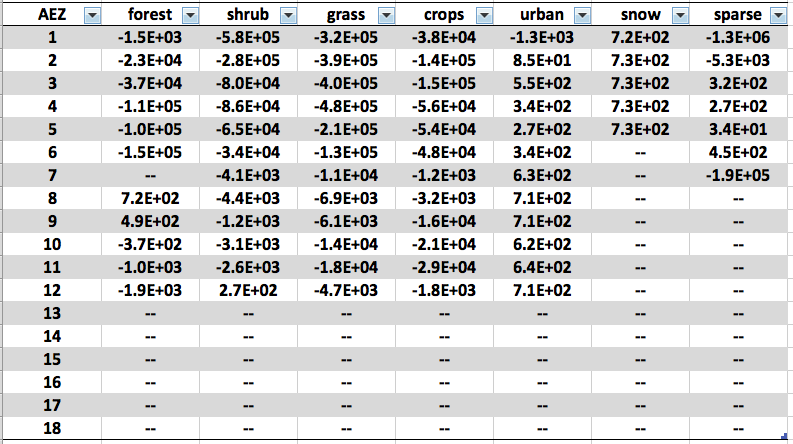


Figure . DiagLevel2 diagnostic file, example for the Africa\_Eastern region. Top left shows the area (km2) in the GCAM data. Top right shows the area in the spatial data (MODIS). Bottom shows the difference, which is what the downscaling algorithm allocates spatially. In future projection timesteps, it’s the same, but instead of showing the difference between GCAM and the spatial data, it shows the difference between the two timesteps, i.e. the landuse change from say 2015 to 2020.

## Outputs/RunName/KernelDensity/

If you turn on the parameter map\_kernels, this folder will be created and contain the kernel density maps, which are used for **expansion** (into grid-cells that didn’t have that PFT previously), to determine which cells will receive the increasing PFT. This is illustrated in Figure 8 and Figure 9.

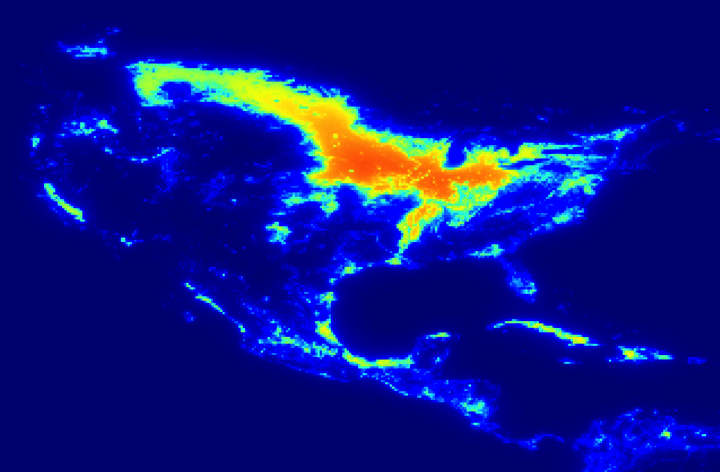
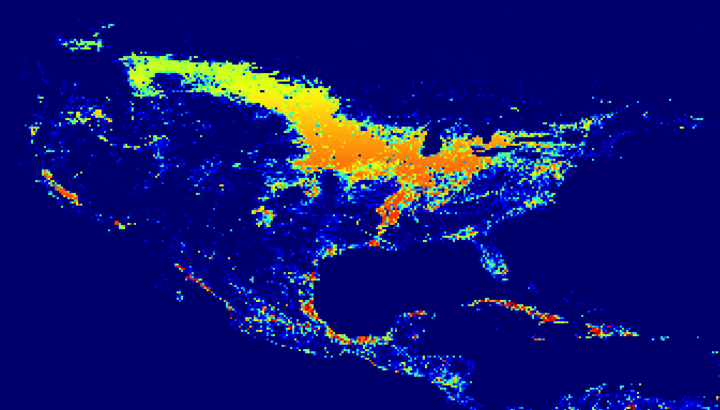


Figure . Top, cropland distribution in the US at 0.05 degree resolution (although picture quality is lower). Bottom, distribution of the crop kernel density values. Crops undergoing expansion (after intensification) will go to cells that are deep blue in the top figure (don’t have crops), and have the highest values in the bottom map. Basically, with more probability around existing agricultural regions. The final choice of what grid-cells receive expansion is done stochastically, with a probability dependent on the kernel density (the higher, the higher the probability that grid-cell is picked).

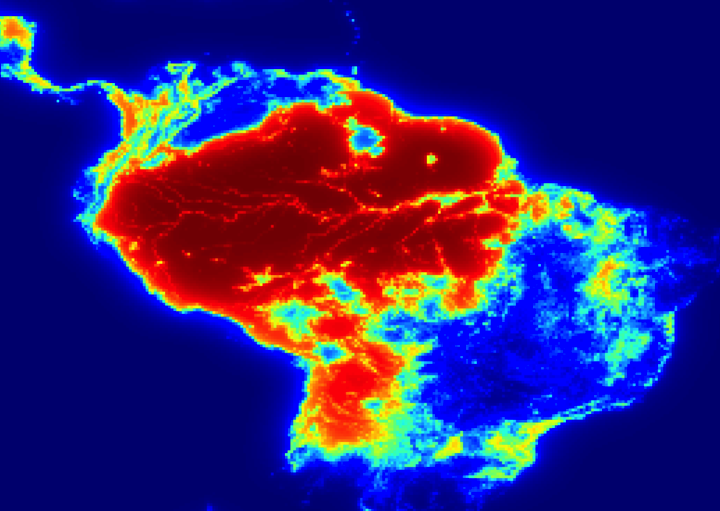


Figure . Same as Figure 8, kernel density for forests in the Amazon basin.

## Outputs/RunName/Maps/LUC/

This folder contains maps of the downscaled land use and land use change, if you turned on the parameter *map\_LUC*.

For each timestep, the previous timestep land use, the current timestep land use, and the difference between both (Figure 10).

That same folder also can contain maps of the total landuse change over the scenario if you turned on the parameter *map\_tot\_LUC*. If you’re looking at 2005-2100, it will be maps of say crops in 2100 minus crops in 2005. This doesn’t include the change from the base-year harmonization to MODIS, so you’re really looking at the GCAM land use decisions Figure 11.

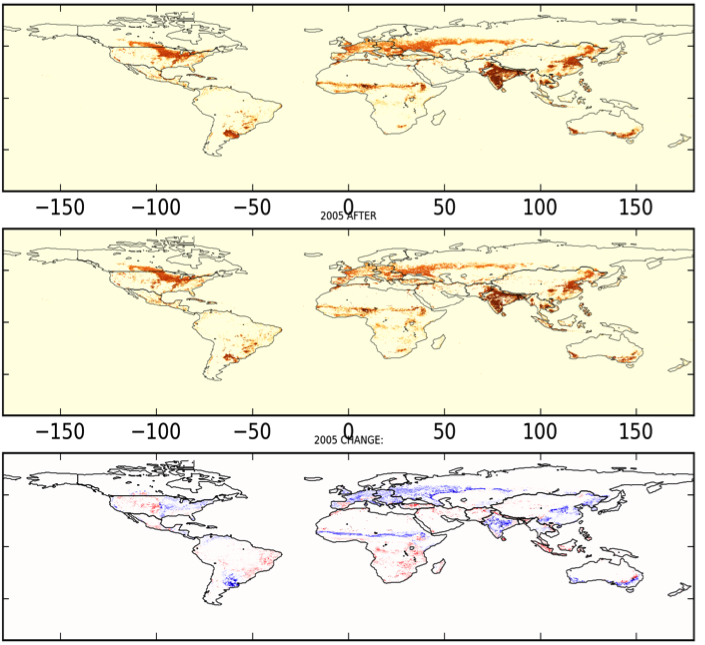


Figure . 0.05 degree resolution land use of croplands in 2005 from the observation data (top) and from the downscaled GCAM data (middle). The difference between both is highlighted in the 3rd figure. For example, GCAM has less crops than MODIS in Eastern Europe and Russia. For projection years, the top figure would be the downscaled land use from the previous timestep. Typically, the change in the base year (changing MODIS patterns with GCAM total areas) is larger than a 5-year landuse change from GCAM.

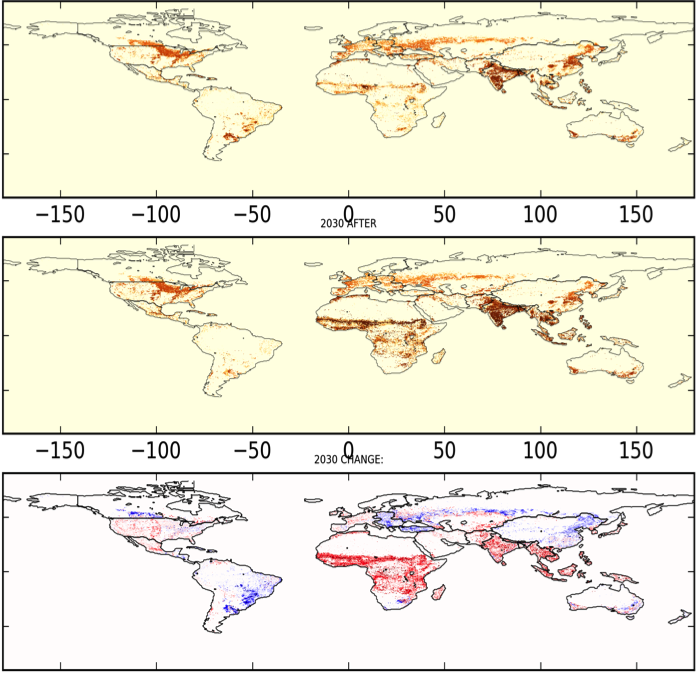


Figure . Same as Figure 10 but from 2005 (post-harmonization with MODIS) to 2030. The difference map thus shows the cumulative landuse change as projected by GCAM over that period. In this case, cropland expansion in Africa, India and southeast Asia, some in the western US too.

# Version 2

Version 2 of the downscaling includes the following developments:

* All 12 GCAM crop categories are now considered separately (e.g. Rice in Figure 12). Version 1 had a single crop category (all crops), now we downscale **Corn, FiberCrop, FodderGrass, FodderHerb, MiscCrop, OilCrop, OtherArableLand, OtherGrain, PalmFruit, Rice, Root\_Tuber, Wheat**.



Figure . Downscaled GCAM rice crops in 2005.

* Spatial constrains to land allocation for any PFT. Two constrains have been implemented in version 2: the role of soil workability and nutrient availability in driving cropland expansion/intensification. The better the land for agriculture, the more it attracts agriculture. Kernel density, which was a standalone factor in spatial allocation in version 1, is now considered as a constrain too. Any kind of other constrains can be considered, given the input data are made available to the code. For example, implementing the constrain of protected status for deforestation would be straightforward (no code development).
* Results can be saved as netcdf files if the user needs.
* Stochasticity of landuse expansion is now an option, turned off by default.

The following sections describe these developments in more details.

## Downscaling all 12 GCAM crops separately

This feature came from the need to couple GCAM with vegetation/crop/earth-system models with various crop types, and with it’s water module (irrigation/rainfed) which might need such agricultural details. The base-year distribution of the 12 GCAM crop categories is not readily available in any dataset, but we reconstruct it starting from the 175 crops global distribution (Monfreda et al., 2008) produced. Each of these 175 crops is attributed to one of the 12 GCAM crop categories, summing their land area to infer the share of croplands that belongs to each 12 categories in any given grid-cell. These shares are then applied to the MODIS **area** of croplands, ending up with a consistent distribution of all natural ecosystems (forests, shrublands, grasslands, deserts, etc) and all 12 crop categories, from which we can apply the downscaling code, given that we’ve edited the input parameter files to reflect that change from 7 to 18 final PFTs (crops is replaced by Corn, FiberCrop, FodderGrass, FodderHerb, MiscCrop, OilCrop, OtherArableLand, OtherGrain, PalmFruit, Rice, Root\_Tuber).

The Monfreda data and the code to extract the data and combine them with MODIS croplands into the GCAM crop categories are located in the folder: *./SpatialInputs/Monfreda*

The code is called *Monfreda\_to\_MODIS.py* and relies on the Monfreda-crops-to-GCAM-crops mapping from the excel file *CropMapping.xls* (Figure 13). The data has already been extracted so you will only need to run it if you want to re-compute the data or for development.

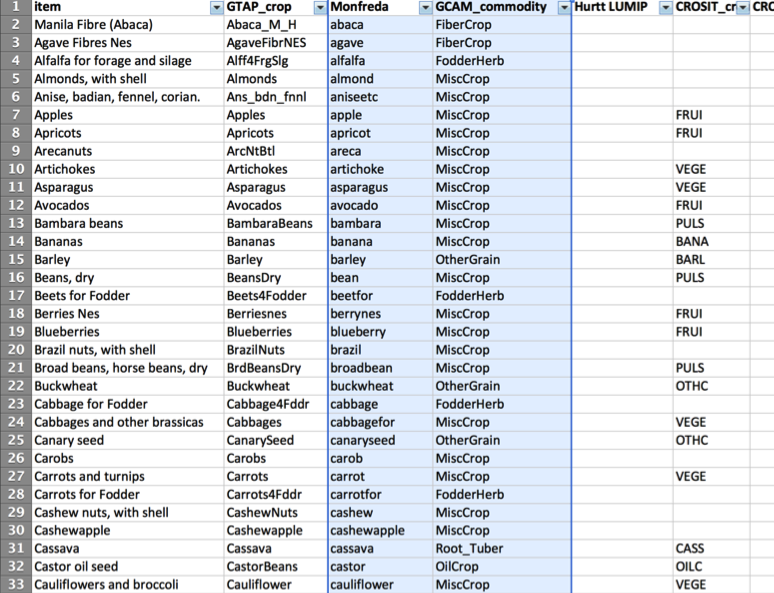


Figure . Subset of the mapping table from Monfreda’s 175 crops to GCAM’s 12 crop categories. The two columns relevant for the code are highlighted in blue.

The input parameter files needed to be modified to provide the aggregation, transition and priority rules. It is now the default version of the downscaling, and looks like the subset in Figure 14.

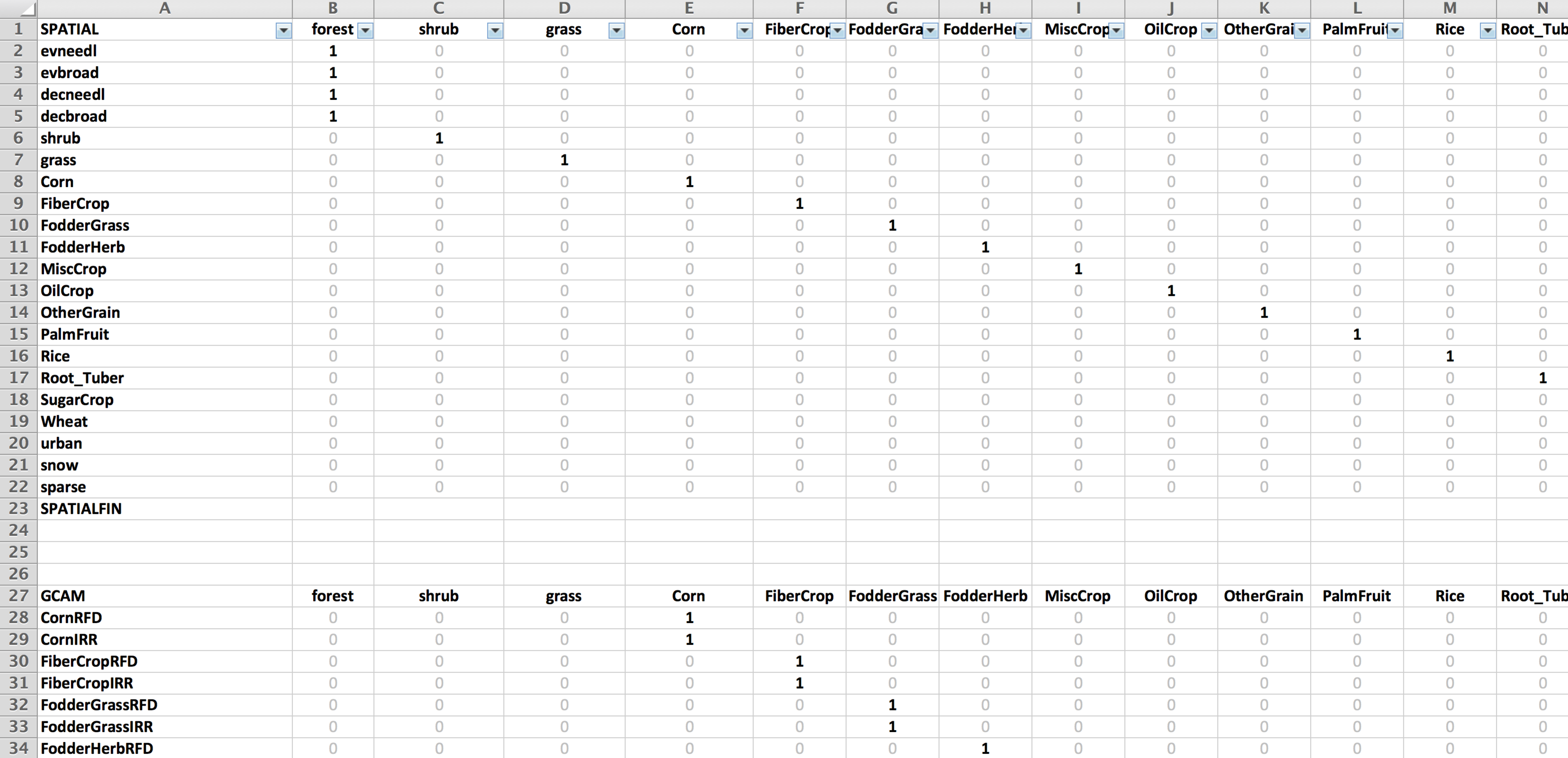


Figure . Subset of the downscaling rules file (*Landclass\_Harmonization\_Monfreda.xls*) with the 12 crop categories. The first table - from SPATIAL to SPATIALFIN - aggregates the MODIS/Monfreda data to the final PFTs (columns). The second table aggregates GCAM land PFTs to the final PFTs. Then, not shown here, there is the transition priorities table, the treatment order table, and a new one, for constrains (see Section H.2).

If you want to revert to a single crop category, as in version 1, you will need to:

* Change the downscaling rules in the file (*Landclass\_Harmonization\_Monfreda.xls* by default) back to the final 7 PFTs of version 1: aggregate all crop types from the new spatial data file (Rice, Corn, Oilcrops, etc) to a single crop category; and update the final PFT categories (columns) in each column, as well as the rules themselves.

## Spatial constrains to land allocation

This feature was developped to better represent a key driver of where landuse change occurs: the suitability of land for agriculture. Most of the work involved developing the capability, and the data to apply soil workability and nutrient availability are provided. However, you can add any type of constrains (e.g. climate, terrain, proximity to infrastructures), and constrains can be applied to any PFT, not just crops (e.g. you can add protected areas to avoid deforesting them in the downscaling). Kernel density is now applied in the same way as constrains.

Constrains are provided as additional .csv files (except for kernel density which is computed on the fly in the code), with the value of the constrain for each grid-cell. The first column is the grid-cell ID, as found in the MODIS/Monfreda baseyear datafile. The second column is the value of the constrain, from zero (full constrain) to 1 (not constraining, e.g. lots of soil nutrients). Be careful when you use zero, grid-cells with that value cannot receive new areas of the PFT for which the constrain applies, under any circumstances. So you might run out of land and may not match GCAM projections. Instead, using a very small value (e.g. 0.001) ensures that the downscaled landuse will be consistent with GCAM, by using these grid-cells as a very last resort if needed.

To include one or multiple constrains in the downscaling, you must indicate so in the parameter files:

* In the downscaling rules file (*Landclass\_Harmonization\_Monfreda.xls* by default), there is a table (new to version 2) that starts with CONSTRAINS and ends with CONSTRAINSFIN. This table must always be there, even if you are not applying any constrains. You add one row per constrain, and for each final PFT you indicate whether that constrain should apply, and how much. To do so, you add any value between -1 and 1. Only zero indicates the constrain does not apply. 1 indicates the constrain applies “fully” (see later). -1 indicates the constrain applies fully, but in the opposite way. For example, in Figure 16, I’ve applied both soil quality and nutrient availability to all crops, each with the same weight (0.5). Both constrains do not apply to forests and shrubs though (0). For grasslands, soil quality does not apply (0), but nutrient availability does, with the opposite effect (-1). While croplands will be attracted by high nutrient availability, grasslands allocation will tend to avoid good soils. This was just as an example to show that constrains can be applied as the opposite. It could make sense for example to have desertification avoiding great soils.
* In *downscaling\_param.xls*, for each constrain, add a row with the exact same name in the first cell (parameter name), the file containing the constrain data in the second cell, and str in the 3rd cell (“string” in python), as shown in Figure 15. For kernel density, no need to provide the file, as we compute kernel density on the fly while running the code.
* If you want to map your constrains as an output, you can do so by turning on the parameter *map\_constrains* in *Downscaling\_params.xls* (1 is on, 0 is off).

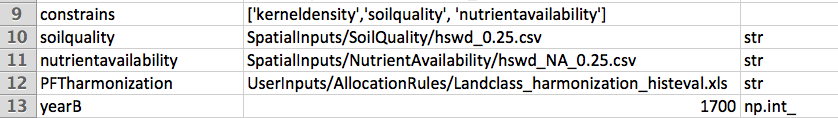


Figure . Downscaling\_params.xls with 2 constrains. The constrain names in brackets must match the constrain parameter names in pointing to the file location (first cell of second and third row). For kernel density, no file is provided because it is computed on the fly in the code.

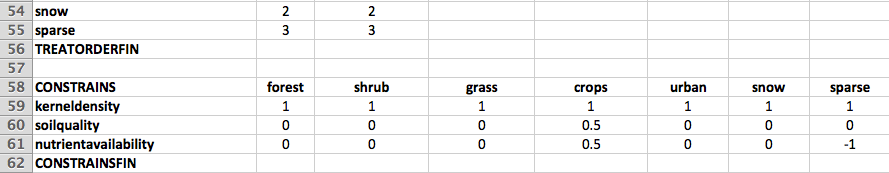


Figure . The new downscaling rules table in version 2, to apply constrains.

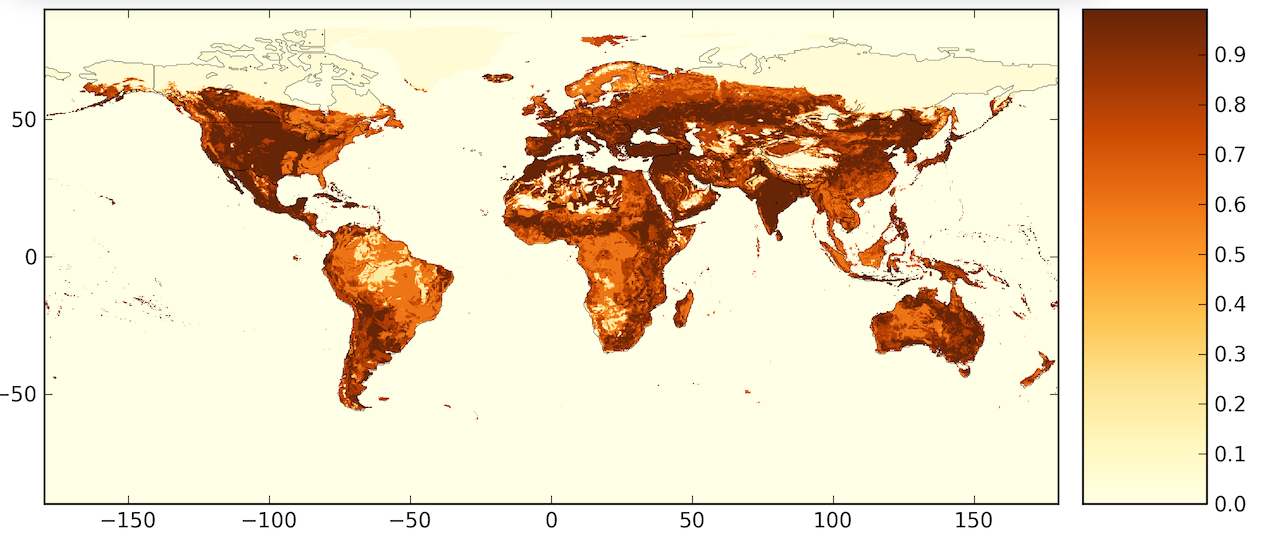


Figure . Example of constrains: nutrient availability, as saved in the output folder when the parameter map\_constrains is turned on. The constrain must be prepared with values from 0 (full constrain) to 1 (no constrain). Data are from the Harmonized world soil database (HWSD).

## NETCDF outputs

Netcdf is a classic format widely used for spatial data. The downscaling now includes the option to save the distribution of the final PFTs in netcdf. The files will be saved in the folder *./Outputs/run\_name/Spatial\_LU/netcdf* as one file per PFT for the whole temporal extent of the run. Additionally, because it has been asked (IPCC, ESM runs), the gridded distribution is produced for all years within a timestep. For example, downscaled forests from 2005 to 2010 are saved in .csv as 2 files, one for each timestep. In the netcdf file (*LU\_forest.nc*), each year (2005, 2006, 2007, 2008, 2009, 2010) has a distribution, which is obtained by linearly interpolating between the forest area of 2005 and 2010 in a grid-cell.

Files are saved in netcdf version 4 (netcdf4). The files contains a lot of metadata, including the PFT distribution as 3-D maps, the coordinates (latitude, longitude and time), a description of the data, references to the GCAM model, the projection (geographic lat/lon), etc.

Important: netcdf files can get huge (see performance section), because they are stored as maps (ocean grid-cells are not skipped), contain every year instead of every timestep, and do not have data-compression capabilities (netcdf version does, but was not the format asked for the IPCC). You can reduce file size to about a 7th of the uncompressed size by zipping them after the downcaling is over, however. Free softwares such as NASA’s Panoply let you readily visualize the nectdf data through maps.

By default, the netcdf option is off, to turn it on simply change the *save\_netcdf* parameter in the *Downscaling\_params.xls* from 0 to 1.

## Stochasticity of expansion

Expansion is the increasing area of a given PFT which occurs on grid-cells which did not previously contain that PFT. In version 1, potential grid-cells for expansion were weighted according to their kernel density (index of proximity to grid-cells with that PFT). The grid-cells actually receiving the expanding PFT were then chosen following a draw where the normalized kernel density (from 0 to 1), represents the probability of that grid-cell being among the chosen ones.

For the sake of reproducibility of the results, stochasticity is now an option, turned off by default. In this configuration, potential grid-cells for expansion are weighted according to their kernel density as before (in combination with the constrains (see Section H.2.). But then, the grid-cells actually receiving the expanding PFT are simply chosen as the 10% grid-cells with the highest weight. There’s no scientific basis for the 10%, I experimented with a few values and it does matter, but I don’t think there’s any research to support a global number. It can be changed in the code (look for the string “*drawcells = expansion\_likelihood*” in the code, the 90% (0.9) comes just after).

## Sensitivity of performance to the parameterization

I did a few test runs to get a rough idea of performance and the influence of user-defined parameters, as shown in Figure 17. Note that producing maps for each timestep is pretty time-consuming, as is going for higher resolution. Run-time sort of changes proportionnally to the number of final PFTs being downscaled. Output folder size depends very much on whether or not you want netcdf files.



Figure . Performance sensitivity. Run time and total output folder size for different configuration of the downscaling. Note that producing maps for each timestep is pretty time-consuming, as is going for higher resolution. Run-time sort of changes proportionnally to the number of final PFTs being downscaled. Output folder size depends very much on whether or not you want netcdf files.