GOANIM

Model documentation and code

Barbieri. P., Pellerin S., Seufert V., Smith L., Ramankutty N., Nesme T.

Contents

1 Aim of document

The aim of this document is to provide the Supplementary Information for Barbieri et al. 2019 [1], describing the GOANIM model into details and providing the model code in R language. This supplementary Document was created with knitr [2], a software that combines the typessetting system LaTeX and the statistical computing language R.

2 Introduction

The GOANIM (Global Organic Agriculture NutrIent Molator) model is a spatially explicit, biophysical and linear optimisation model simulating cropland soil nitrogen (N) cycling in organic farming systems and its feedback effects on food production at the global scale. It is based on the overall assumption that the global harvested cropland area remains constant at current levels [3]. GOANIM includes different compartments (organically managed croplands, livestock animals and permanent grasslands) and accounts the biomass and N flows among these compartments (as feedstuff, grazed biomass and animal manure) as well as the N flows between cropland soils and the environment as represented in Figure 1. GOANIM uses high resolution (5 arc-min) spatially explicit databases for the year 2000, providing consistent data on agricultural biomass flows (see Table S1 for the full list of the variables used), encompassing the 61 most important crop species (see the Suppelementary Dataset) by global acreage and 5 livestock animal species (cattle, sheep , goats, pigs, and poultry). The model respects the thermodynamic law of conservation of mass and energy it embraces all countries and geographic territories (164) covered by FAOSTAT. GOANIM simulates the supply side of the global food system -i.e. food production-, by maximizing organic production from both cropland and livestock food commodities. Cropland production is maximized in each grid-cell as a function of N supply to cropland soils from different sources, including organic manure from livestock animals, biological N fixation, atmospheric depositions, and additional external inputs. The floating variables in the optimisation model correspond to the local livestock animal population and the allocation of animal manure to the different crop species; both floating variables are optimised at the grid-cell scale (Figure 2). Any use of cropland resources other than food and feed is not directly considered here, apart for crops like rubber and cotton, whose production is be used for industrial purposes. All model compartments are assumed to be at steady-state -i.e. all state variables are constant in spite of ongoing processes that strive to change them- and all flows are simulated considering a timeframe of one year. The model assumes that each raster's grid-cells is independent, i.e. resources are not transportable or exchangeable outside grid-cell boundaries. This choice is in line with organic farming principles aiming at a local sourcing for both feed and fertilizers [**IFOAM2005**, 4]. Additionally, the resolution of our datasets (˜10 x 10 km at the equator) represents a feasible distance for livestock manure sustainable transportability [5].

First, a pre-modelling module (*conventional add-in module*) calculates the total nitrogen input and outputs to soils in conventional farming systems, computes N budgets in each grid-cell and calculates the N in livestock animal manure and in wastewater that is not needed to balance the N crop demand. This N surplus embed in conventional manure and wastewater resources, can then be made available by the model user as additional N inputs to organic systems for ay given scenario where organic farming production has a global share inferior to 100% - i.e. between 0 and 100% -. Then, the GOANIM optimization model maximizes organic food production in every single grid-cell as a function to N available resources. For each chosen

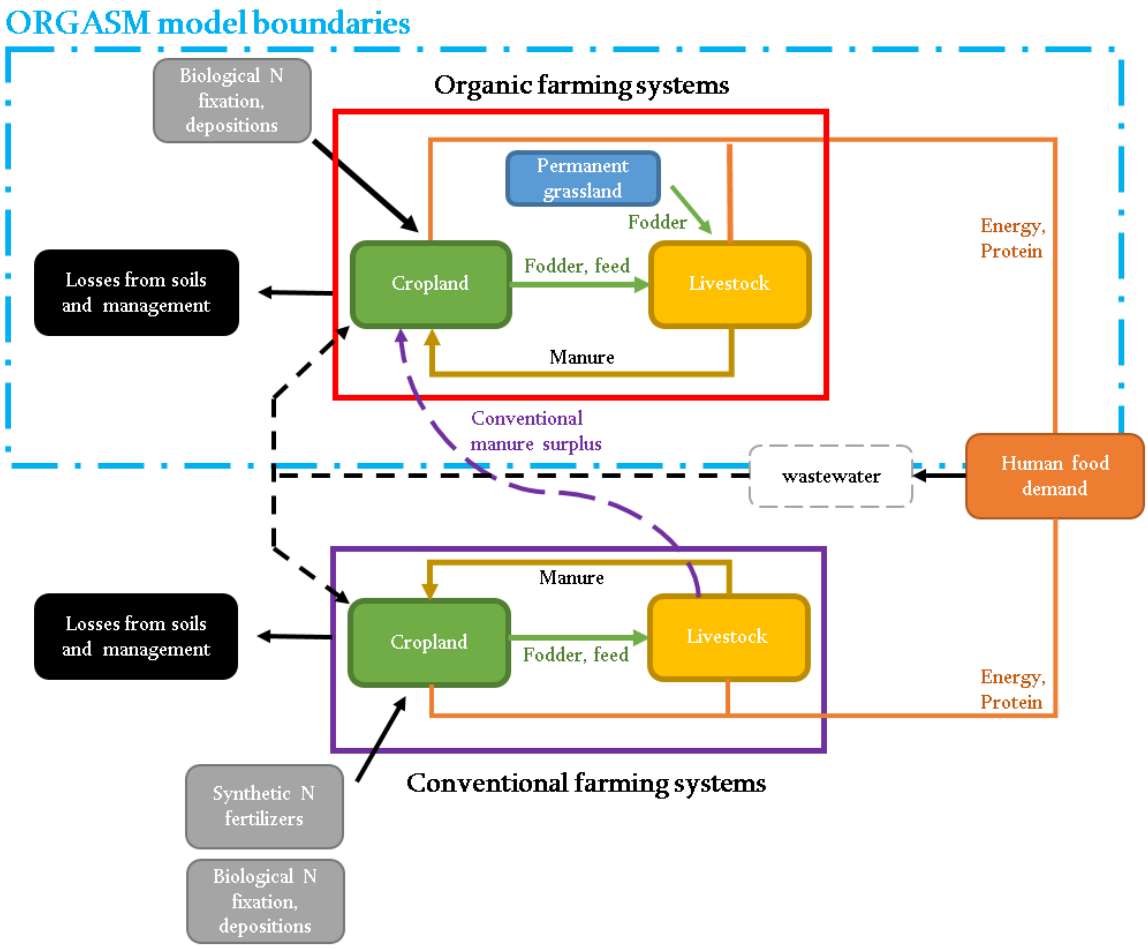

Figure 1: Model boundaries

global organic farming production share, GOANIM assumes the conventional farming production share to be equal to $(1 - organic_{globalshare})$. The conventional add-in module and GOANIM-model are interlinked by allowing, or not, the import of conventional N surplus resources (manure and wastewater) into organic systems.

The model is coded using the R software (R Open 3.3.2 - MRAN 2016, https://mran.microsoft.com/) and the optimisation problem is solved using the COIN-OR clp solver.

2.1 Main model hypothesis

GOANIM is grounded on a few hypotheses:

1. Global hypothesis valid for all model add-ins:

   (a) Nitrogen inputs include: (i) synthetic and (ii) organic fertilisers, N fixation by (iii) symbiotic biological nitrogen fixation (BNF) and (iv) non-symbiotic free-living cyanobacteria, (v) N dry and wet atmospheric depositions, and (vi) N from crop residues recycled to soils;

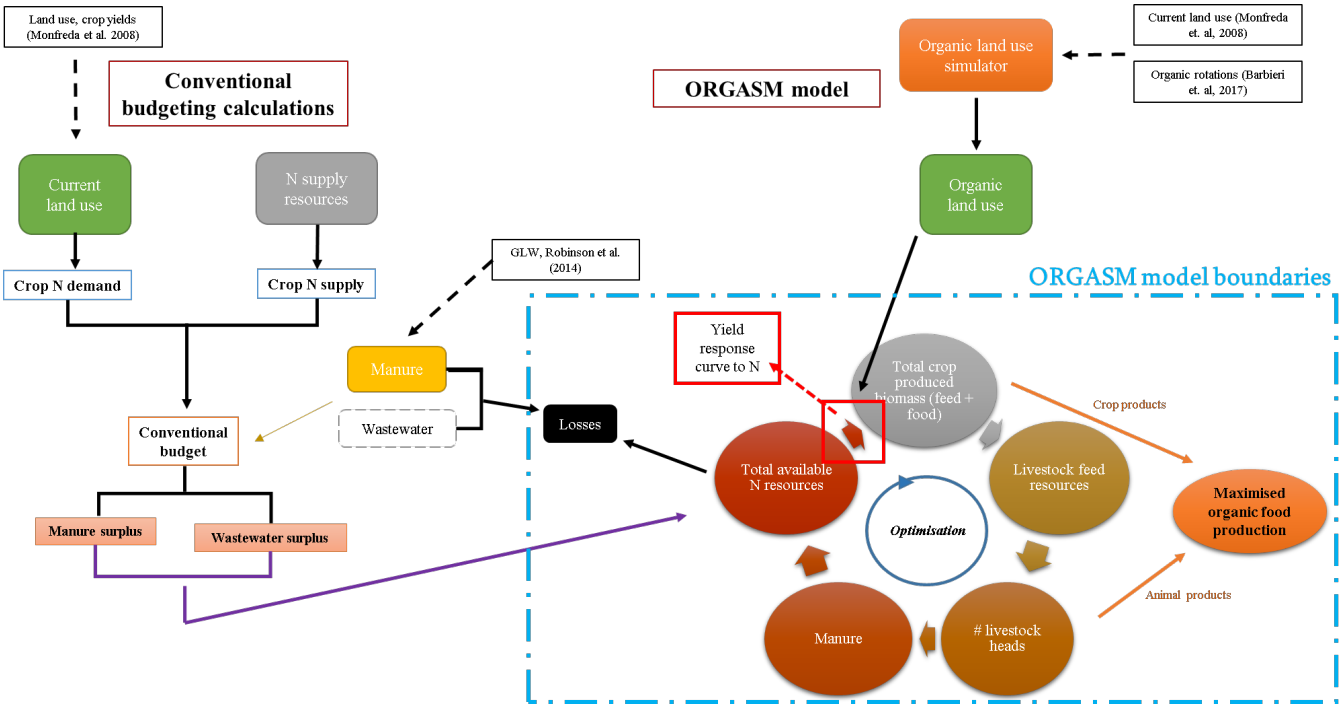

Figure 2: Model overview

(b) Nitrogen outputs include: (i) crops N demand (harvested products and crop residues), (ii) losses -i.e. N leaching (NH_3), volatilisation (N_2O, N_2, and NO_x) due to manure management/storage and after the application to the soil of any organic or inorganic input;

(c) Grid-cells are independent, i.e. resources are not transportable outside grid-cells, and cannot be exchanged between grid-cells. Grid-cells have a size of about 100 km^2 at the equator, which well represent the maximum distance to which manure can be transported in an economic rentable way;

(d) The nitrogen density of crops and livestock products does not differ between farming systems;

(e) Livestock diets - calculate in terms of calories and proteins - do not differ between farming systems;

(f) Livestock yields do not differ between farming systems.

2. Hypothesis applicable only to organic farming systems:

(a) Organic maximum attainable yields are estimated by multiplying the current - i.e. conventional - by an organic-to-conventional yield gap (Ponisio et al. 2015). We corrected this organic-to-conventional yield gap in order to account for a yield reduction due only to pest and diseases, i.e. not accounting for the yield reduction due to N deficiency;

(b) Organic yields are calculates as a linear function of nitrogen supply, up to the maximum attainable yield (representing the plateau of the N response curve);

(c) Conversion to organic farming is considered to take place equally in every single grid-cell. This share directly affect the share of cropland and pasture area farmed organically or conventionally - i.e. for a 20% conversion to organic farming, 20% of total cropland and pasture areas in each

grid-cell is considered to be converted to organic farming -, as well as the share of livestock animal populations managed organically.

3 Conventional add-in module

3.1 Nitrogen flows

Nitrogen flows from and to cropland soils are estimated by calculating different nitrogen inputs (IN) and outputs (OUT) elements [6]. In this study, IN is divided into six elements, whereas OUT is divided into five elements (Equations 1 and 2):

$$IN = IN_{fert} + IN_{man} + IN_{dep} + IN_{fix} + IN_{res} + IN_{ww} \tag{1}$$

$$OUT = OUT_{crop} + OUT_{res} + OUT_{lea} + OUT_{gas} \tag{2}$$

where IN and OUT are the total nitrogen input and output, respectively; IN_{fert} is the N in synthetic fertilizers; IN_{man} is the N in manures; IN_{dep} is the N input from wet and dry atmospheric deposition; IN_{fix} is N fixation (symbiotic and non-symbiotic); IN_{res} is the N from the fraction of crop residue recycled to soils; IN_{ww} is the N from wastewater; OUT_{crop}, OUT_{res}, OUT_{lea}, OUT_{gas} are the N output of the harvested crops, crop residues, leaching, and gaseous losses, respectively. All flows variables are calculated in [$tons\ Nha^{-1}y^{-1}$]. A full reference to the input data is reported in Table 1.

IN_{fert} is calculated using a global spatial explicit database on synthetic N fertilisers application rates [$t\ N\ ha^{-1}$] for every of the 61 crop species considered in this study (Mueller et al. [7]).

IN_{man} is calculated by multiplying livestock populations with animal species-specific N excretion rates. Livestock populations density data for the year 2007 were obtained from the Gridded Livestock of the World (GLW) from the FAO [8]. The GLW describes the spatial distribution of cattle, sheep , goats, pigs, and poultry at a resolution of 1 arc-min. Data were aggregated to a 5 arc-mi resolution. Others livestock species were not considered in this study since they have a minor contribution to global livestock production and a little influence in the manure estimates [6].

The GLW maps report the livestock populations density as harmonised with FAOSTAT data and include animal raised for both draft and production purposes. More in details, these maps report the number of animals standing at the moment of enumeration. Since the GOANIM-model only simulates producing animals (section 4.1.8), we corrected the GLW density maps to account only for producing animals (no draft-raised animals). To do so, we (i) retrieved from FAOSTAT the number of both standing and producing animals in one year for every of the 164 considered countries (average of the years 2005-2009), (ii) corrected, when appropriate, the number of producing animals by the respective number of batches per year (personal estimation), and (iii) we used the ratio *producing animals/standing animals*, for every given country, to correct the GLW maps and estimate the fraction of producing animals only.

We calculate the N excretion rates of each livestock animal species for every individual country following Sheldrick et al. [9] and Liu et al. [6] - i.e. by assuming that the excretion rate of each livestock species is proportional to its slaughter weight -. Sheldrick et al. [9] provides reference data on livestock live body

weight and N excretion rates for cattle, pigs, sheep, goats, and poultry (Table 2). We calculated the live weights of each livestock species in every of the 162 individual countries multiplying the carcass weight (from FAOSTAT [10]) by species-specific factors converting the carcass yield into live weight (Table 2). Finally, N excreta was estimated proportionally to the reference body weight reported in Table 2.

In order to estimate the collectable fraction of the excreta (i.e .the fraction of excreta produced in stables or in confined management systems), we retrieved and applied housing system statistics, for different global regions, following the IPCC Tier 1 procedure [11]. Excreta produced in meadows were not accounted as available for cropland application. We estimate all N losses (volatilisation, leaching and run off) happening during manure collection and storage and management following the Chapter 10 of the IPCC procedure [11]. No is considered to be used for permanent grasslands fertilisation in addition to the share of excreta directly deposited on permanent grasslands. Instead, fertilisation of temporary fodder crops (< 5 years) is accounted; since these crops are part of the 61 arable crop species covered in this study[1].

IN_{dep} is calculated from Dentener's spatially explicit modelled dry and wet ($NO_y + NH_x$) N atmospheric depositions [12]. Here we applied the estimation for the year 1993, after conversion to a 5 arc-min resolution.

IN_{fix} is the estimate N input from the fixation of atmospheric N, including both symbiotic N fixation via legumes crop species (BNF) and non-symbiotic N fixation by free-living cyanobacteria. BNF was estimating applying the model developed by Hogh-Jensen et al. [13] reported in equation (3). Free-living cyanobacteria was estimated using data from Liu et. al [6].

$$BNF_i = Y_i \times NiC_i \times \frac{1}{NHI_i} \times (1 + PRoot_i) \times Ndfa_i \tag{3}$$

where: BNF_i is the total N fixed [$tons\ N\ ha^{-1}$], Y_i is the crop yield [$tons\ DM\ ha^{-1}$], NiC_i is the N content of the above-ground biomass [$tons\ tons^{-1}$], NHI_i is the N Harvest Index [$tons\ tons^{-1}$], i.e. the ratio between the N content in the harvested biomass and the total N amount in the above-ground biomass, $PRoot_i$ is the amount of N fixed in the roots as proportion of the N fixed in the shoots [$tons\ tons^{-1}$], and $Ndfa_i$ is the ratio between the amount if symbiotically fixed N and the total N content in the crop biomass [$tons\ tons^{-1}$], for each given leguminous crop species i. BNF from sugarcane was accounted using the average coefficient of 100 $kg\ N\ ha^{-1}$. Free-living cyanobacteria N fixation was estimated following Liu et al. [6]. Cyanobacteria fixation coefficients were taken from Liu et al. [6] and Smil et al. [14]. More in details we considered a fixation of 20 $kg\ N\ ha^{-1}$ for rice and 12 $kg\ N\ ha^{-1}$ for cereal (except rice), tuber, and oil crop species.

IN_{res} is calculated by multiplying OUT_{res} with a removal factor (β) that accounts for the share of the crop residues removed from the soil, as in equation (4). The removal factor (β) was retrieved from different sources [6, 9, 14]. According to these sources, we considered a removal factor of 35 % and 50 % in developed and developing countries, respectively.

$$IN_{res} = (1 - \beta) \times OUT_{res} \tag{4}$$

IN_{ww} is estimated as described in section 3.4.

[1]Note that the N excreta coefficient were further adjusted as explained in paragraph 4.1.12 to match the same coefficients used in the GOANIM-model

OUT_{crop} is estimated by multiplying the crop production ($yield \times area$) by the N density of each harvested product as in equation (5):

$$OUT_{crop} = \sum_{i=1}^{61} Y_i \times A_i \times [N_i] \tag{5}$$

where Y_i is the current conventional yield [$tons\ DM\ ha^{-1}$] for each give crop i, A_i is the current area [ha] under cultivated under each crop i, and $[N]_i$ is the nitrogen density [$tons\ tons^{-1}$] of each crop species i harvested product (see Supplementary Dataset). The spatial explicit crop fresh yield for the 61 considered crop species was obtained from Monfreda et al. [15]. Moisture and nitrogen content of various crops was obtained from various sources [15–17] (see Supplementary Dataset).

OUT_{res} is calculated multiplying the crop residues dry yield by their nitrogen content. The crop residues dry yield was calculated as a function of the harvested yield, as in equation (6):

$$Y_{res_i} = Y_i \times RPR_i \tag{6}$$

Where Y_{res_i} is the crop residue dry yield, Y_i is the dry matter yield, and RPR_i is the residue-to-product ratio, respectively for each crop species i. The RPR values were obtained from Liu et al. [6], or calculated from the harvest index (HI) as $(1 - HI)/HI$ [6]. HI values were obtained from various sources [17, 18].

OUT_{lea} is estimated following the IPCC Tier 1 procedure [19]. A loss coefficient of 30% was applied in areas where the annual average rainfalls are higher than the annual average evapotranspiration. Rainfalls and evapotranspiration data where obtained respectively from the NASA and Zomer [20, 21]. Our leaching estimation was in line with the estimations performed by Liu et al. [6].

OUT_{gas} estimates the direct (N_2O) and indirect (NH_3 and N oxides -NO_x-) gas losses from soils due to nitrification, denitrification and volatilisation following the IPCC Tier 1 procedure [19]. We also accounted for N_2 emissions, calculated using experimental estimation of the $\frac{N_20}{(N_2+N_20)}$ ratio, retrieved trough personal communication and [22, 23]. All emissions were estimated for all inputs, i.e. synthetic fertilisers, organic N fertilisers and N in crop residues including N fixed.

3.2 Nitrogen budgets

Conventional nitrogen budgets are computed in each grid-cell following a three-step procedure. First, budgets are calculated without considering the N inputs from organic fertilisers - i.e. manure and wastewater - as in equation (7). Then N available in manure is used to balance, when possible, all grid-cells where the budget is negative (equation 8). Finally, N available in wastewater is used to balance to zero all grid-cells that still have a negative budget (equation 9). Overall, an over-fertilisation up to 20% of OUT_{crop} is allowed, as commonly over fertilisation practices take place. Budgets can be computed assuming any share of

Table 1: Input model data and sources

Variables/Datasets	System	Unit	Source
Crop harvested areas	Conventional	ha	[15]
Crop harvested areas	Organic	ha	Barbieri II
Crop yields	Conventional	$tons\ DM\ ha^{-1}$	[15]
Crop yields	Organic	$tons\ DM\ ha^{-1}$	Barbieri II
Synthetic fertilisers	Conventional	$tons\ ha^{-1}$	[7]
Livestock densities	Conventional	$heads\ grid-cell^{-1}$	[8]
Livestock densities	Organic (II)	$heads\ gridcell^{-1}$	[8]
N atmospheric deposition	Organic and Conventional	$tons\ ha^{-1}$	[12]
N from wastewater	Organic and conventional	$tons\ grid-cell^{-1}$	Personal dataset

Table 2: Livestock excretion rates used in the model (from Sheldrick et al.) and carcass to live weight conversion coefficient

Livestock	Live weight $[kg]$	N [Kg year N $[kg\ year^{-1}]$	Carcass yield over live weight [%]
Cattle	250	50	50.8
Sheep	15	10	50
Goats	12	10	50
Pigs	80	12	77.4
Poultry	2	0.6	70

conventional farming at the global scale, ranging from 100% to 0%.

$$\Delta_1 = [(IN_{fert} + IN_{dep} + IN_{fix} + IN_{res}) - (1.2 \times OUT_{crop} + OUT_{res} + OUT_{gaz})] \times global\ share\ conventional \tag{7}$$

$$\forall \Delta_1 < 0, \Delta_2 = \Delta_1 + IN_{man}, until\ \Delta_2 = 0 \tag{8}$$

$$\forall \Delta_2 < 0, \Delta_3 = \Delta_2 + IN_{ww}, until\ \Delta_3 = 0 \tag{9}$$

The N from manure and wastewater that is not used to balance the conventional budgets to zero (including the allowed over-fertilisation) (δIN_{man}, and δIN_{water}) represent the N surplus which can be eventually imported in the GOANIM-model as an additional N input.

3.3 Conventional harvested areas

We retrieved spatial explicit conventional harvested areas $[ha]$ for the year circa 2000 from Monfreda et al. [15].

3.4 Nitrogen from waste-water resources

We estimated the current N available from waste-water resources following the procedure described by Van Drecht et al. [24]. N in waste-water was estimated considering the global population density of 2015 [25], and the current per-capita N excreta, estimated based on the protein consumption for every of the 164 considered countries. We then spatially allocated the per-capita N available in wastewater at a 5-min resolution proportionally to the harvested area of every single country.

4 Code - conventional add-in module

4.1 R environment preparation

We load here all the R packages necessary to run the code. These packages include: raster [26], rgdal [27], ncdf4 [28], and the R packages for parallelism doParallel [29], and foreach [30].

Note that the model was run using an external server owned by the MCIA of the University of Bordeaux. Here, up to 23 cores were simultaneously used for several code steps due to the high RAM requirements. Hence, the code cannot directly run on a Desktop machine.

```
# --- Load R Packages ---
library(tools)
library(raster)
library(rgdal)
library(ncdf4)
library(maptools)
library(doParallel)
library(foreach)

# --- Set Working Directory and Temporary files directory in scratch ---

setwd("/home/pbarbieri/global/Version_I_first_inclusion_of_land_use_change_rules/")
getwd()
library(unixtools)
set.tempdir("/scratch/pbarbieri/R_temporary_files")
tempdir()

# *************   ATT! CORE CONTROLS   *************
detectCores()                           # Detect number of cores
cluster<-makeCluster(23)                # Create cores cluster
registerDoParallel(cluster)     # start cluster
getDoParWorkers()               # check number of cores used
#registerDoSEQ()

# --- Load manual functions ---
        sum.na <- function(x,...){
          if (sum(is.na(x))==length(x)) {     # given N cells to be sum,if they are all NA,
                                              # then the result cell is NA
                NA
                } else {
                  sum(x, na.rm=TRUE)              # ... otherwise NA values are simply
                                                  # skipped (set to zero)
```

```
        }
      }
```

4.2 Set main parameters and load non-spatial datasets

Below is the code we used to set all the coefficients/parameters used afterwards for the calculations of the current i.e. conventional N budget at the grid-cell scale. We load here also all the non-spatial datasets containing the necessary coefficients to run all calculation performed in this "Conventional add-in" module. The *"crop list"* dataset is provided online as a Supplementary Dataset to this publication.

```
#-----Load global(program) coefficients, info files -------------------------------
                ##     1. Coefficients
                global_production_share_coefficient_conv <- 0.2   # This set the schare of each gridcell which is farmed convent
                global_production_share_coefficient_org <- 1 - global_production_share_coefficient_conv
                conv_overfertil_allowed <- 1.2  # this set the "overfertilisation" allowed -->
                                                # if 1.2 --> Application of up to 20% more N then crop demand is allowed

                crop_residues_recycling_factor_conventional <- raster
                +( "Coefficients/map_coeff_residues_recycle_conventional.tif")


                  ### 1.1 Binary coefficients to activate/disactivate variables
                         K_conv_manure_surplus <- 1
                         K_waste_water <- 1

                ##     2. Load datasets

                          # dataset containing most of the useful coefficients
                crop_info <- read.csv("crop_list_60.csv", header=T)


                # Load global raster map defining countries areas and borders
                map_bruno <- raster
                +( "/home/pbarbieri/global/World shapefiles/Country codes/country_UNI_spatial_corrected_Pietro.nc")
                mask_continents <- map_bruno!=0  # create mask to add continentis shapes
                mask_continents <- reclassify(reclassify(mask_continents, cbind(0,NA)), cbind(1, 0))


                              ##     3. Initialise statistics/output

                Output_statistics_dataframe <- data.frame("variable_Num"=c(1:7))
                # according to number of row needed, leaving colomn created as row number index

                        row.names(Output_statistics_dataframe) <- c("total_N_fertiliser", "fertiliser_corrected_leaching",
                                                                    "fertilisers_correted all","%_N_leaching","%_N_losses_total"

                Output_statistics_dataframe <- data.frame("1" = numeric())
```

4.3 Part A: Crops N demand

Below is the code that calculates conventional crops N demand as in the harvested products and in the crop residues. Crops yields and areas were obtained from [15]. N density coefficient were taken from various sources (see Supplementary Dataset online).

```
                ## ---  Load conventional areas in ha i.e. Monfreda areas maps  ---
# specify path to directory containing crop areas maps
        dir <- c("EarthStat data/Conventional_crop_production/60 crops/Crop_areas/")
        files <- sort(list.files(dir, pattern = ".tif"))
                # load files into a list
        conventional_rasters_area <- parLapply(cluster, paste0(dir, files), raster)

# rename layers
          foreach (i = 1:length(crop_info$CropName)) %do% {
                names(conventional_rasters_area)[i] <- paste(crop_info$CropName[i], "_conventional_area", sep = "")
          }

                # create raster-stack
        conventional_rasters_area <- stack(conventional_rasters_area)


                ## ---  Load conventional production (in tons per gridcell) datasets - i.e. Monfreda maps  ---

        # Load conventional production datasets --> Area * Yield
        dir <- c("EarthStat data/Conventional_crop_production/60 crops/")
        files <- sort(list.files(dir, pattern = ".tif"))
        conventional_rasters_production <- parLapply(cluster, paste0(dir, files), raster)

        # rename layers
          foreach (i = 1:length(crop_info$CropName)) %do% {
                names(conventional_rasters_production)[i] <- paste(crop_info$CropName[i], "_conventional_production", sep = "")
          }

          # create raster-stack
        conventional_rasters_production <- stack(conventional_rasters_production)


## --- Calculate production in DRY MATTER ---

        # create a vector containing the DM coefficients
        crop_DM <- crop_info$dry_matter

  # calculate production (harvested) in DM  --> production(FM) * DM
        foreach (i = 1:61, .packages='raster') %dopar% {
                multipl <- conventional_rasters_production[[i]]*crop_DM[i]
                                        # write result to file - specify the final destination folder
                writeRaster(multipl, filename = paste0("Script_generated_files/rasters_conventional_production_DM/", crop_info$C
                }
        dir_conv_DM <- c("Script_generated_files/rasters_conventional_production_DM/")
        files_conv_DM <- sort(list.files(path = dir_conv_DM, pattern = ".tif"))
        conventional_rasters_production_DM <- parLapply(cluster, paste0(dir_conv_DM, files_conv_DM), raster)
        # rename layers
                  foreach (i = 1:length(crop_info$CropName)) %do% {
```

```
                  names(conventional_rasters_production_DM)[i] <- paste(crop_info$CropName[i], "_conventional_production_D
          }
          # create raster-stack
conventional_rasters_production_DM <- stack(conventional_rasters_production_DM)

        ##  --- Calculate total crop Residues production in DM ---

          # calculate crop residues production in DM  --> production_DM * Residues-to-harvested-mass-ratio
foreach (i = 1:61, .packages='raster') %dopar% {
        multipl <- conventional_rasters_production_DM[[i]] * crop_info$calculated_RPR[i]
                          # write result to file - specify the final destination folder
        writeRaster(multipl, filename = paste0("Script_generated_files/rasters_conventional_production_DM_crop_residues/
        }

dir_conv_residues <- c("Script_generated_files/rasters_conventional_production_DM_crop_residues/")
files_conv_residues <- sort(list.files(path = dir_conv_residues, pattern = ".tif"))
crop_residues_conventional <- parLapply(cluster, paste0(dir_conv_residues, files_conv_residues), raster)

# rename layers
                  foreach (i = 1:length(crop_info$CropName)) %do% {
                        names(crop_residues_conventional)[i] <- paste(crop_info$CropName[i], "_conventional_crop_residue
                  }

                   # create raster-stack
crop_residues_conventional <- stack(crop_residues_conventional)


        ## --- Calculate total crop N demand ---

# multiply crop production_DM with crop N content (harvested) to estimate harvested N
        foreach (i = 1:61, .packages='raster') %dopar% {
                multipl <- conventional_rasters_production_DM[[i]]*crop_info$N[i]

                # write result to file - input the final destination folder
                writeRaster(multipl, filename = paste0("Script_generated_files/conventional_crop_N/", crop_info$CropName
                }
        dir_conv_N <- c("Script_generated_files/conventional_crop_N/")
        files_conv_N <- sort(list.files(path = dir_conv_N, pattern = ".tif"))
        conventional_rasters_N_demand <- parLapply(cluster, paste0(dir_conv_N, files_conv_N), raster)
        # rename layers
                  foreach (i = 1:length(crop_info$CropName)) %do% {
                        names(conventional_rasters_N_demand)[i] <- paste(crop_info$CropName[i], "_conventional_N_demand_
                  }

                  # create raster-stack
        conventional_rasters_N_demand <- stack(conventional_rasters_N_demand)

        # calculate (by parallel processing) and re-load total N demand harvested biomass of all 61 crops
        # specify path to folder to save file
        total_conventional_crop_N_demand <- clusterR(conventional_rasters_N_demand, calc, args=list(fun=sum.na), filenam


                total_conventional_crop_N_demand <- raster("Script_generated_files/total_conventional_crop_N_demand.tif"
```

```
# multiply crop residues production_DM with crop residues N content to estimate crop residues N
                        foreach (i = 1:61, .packages='raster') %dopar% {
                                multipl <- crop_residues_conventional[[i]]*crop_info$N_crop_residues[i]
                                writeRaster(multipl, filename = paste0("Script_generated_files/conventional_N_crop_residues/", c
                                }
                        dir_conv_N_residues <- c("Script_generated_files/conventional_N_crop_residues/")
                        files_conv_N_residues <- sort(list.files(path = dir_conv_N_residues, pattern = ".tif"))
                        conventional_rasters_N_demand_residues <- parLapply(cluster, paste0(dir_conv_N_residues, files_conv_N_re
                         # rename layers
                                  foreach (i = 1:length(crop_info$CropName)) %do% {
                                                names(conventional_rasters_N_demand_residues)[i] <- paste(crop_info$CropName[i],
                                  }

                                  # create raster-stack
                        conventional_rasters_N_demand_residues <- stack(conventional_rasters_N_demand_residues)


                        # calculate and load total N crop residues demand of all 61 crops
                        total_conventional_crop_N_demand_crop_residues <- clusterR(conventional_rasters_N_demand_residues, calc,
                                total_conventional_crop_N_demand_crop_residues <- raster("Script_generated_files/total_conventio

       # compute N demand statistics and update the statistic result matrix
                                # 1. total crops harvested demand
                                        N_harvested_crop <- (data.frame(cellStats(total_conventional_crop_N_demand, sum)))
                                        # 2. total crops residues
                                        N_crop_residues <- (cellStats(total_conventional_crop_N_demand_crop_residues, sum))
                                        N_harvested_crop <- rbind(N_harvested_crop, N_crop_residues)
                                          # set rows names
                                        row.names(N_harvested_crop) <- c("total_N_deman_crop_harvested", "total_N_deman_crop_res
                                        names(N_harvested_crop) <- c(paste0("simulation.","1"))
                                        # add rows main database
                                Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_harvested_crop)
```

4.4 Part B: N supply from conventional Input Sources

Below is the code that calculates N inputs to conventional soils (including Synthetic Fertilisers, N fixation, N from crop residues, N from atmospheric depositions, wastewater and conventional livestock's manure. All coefficients used for calculations are reported in the Supplementary Dataset published online. N supply to soil is calculate after accounting for N losses to the environment according to IPCC [11, 19].

4.4.1 Synthetic fertilisers N inputs

```
## --- Synthetic fertilizers inputs ----------------------
                                # Retrieve dataset
       dir_N_fertilizers <- c("Inorganic_fertilisers/Inorganic N tons per gridcell/")
          files_N_fertilizers <- sort(list.files(path = dir_N_fertilizers, pattern = ".tif"))
          rasters_N_fertilizers <- stack(parLapply(cluster, paste0(dir_N_fertilizers, files_N_fertilizers), raster))

  # calculate total synthetic fertilisers inputs for all 61 crops
               total_N_fert <- clusterR(rasters_N_fertilizers, calc, args=list(fun=sum.na),
                                                              filename=paste0("Script_generated_files/", "total_N_fert.tif"),
```

```
                                total_N_fert <- raster("Script_generated_files/total_N_fert.tif")


                                # estimate N from fertilisers available after losses (IPCC)

 # 1. direct emissions coefficient: 0.01 --> range: 0.003-0.03
       # 2. indirect emissions coefficient 0.1
                total_N_fert_corrected <- total_N_fert * (1- (0.01 + 0.1))

       # 3. leaching losses coefficient: 0.3 --> coded as the available share -i.e. 0.7.
       # Gridcells where leaching occurs were estimated according to the IPCC procedure.
                leaching_areas_mask <- raster("Climatic and irrigation data/mask_area_leaching_final_irrigation_above_005.tif")
       # create mask for areas where leaching takes place
                mask_correction_leaching <- leaching_areas_mask * 0.7
                mask_correction_leaching <- reclassify(mask_correction_leaching, cbind(0,1))
# correct fertiliser input for leaching using the mask
      total_N_fert_corrected <- total_N_fert_corrected - (total_N_fert * (1-mask_correction_leaching))
      total_N_fert_corrected <- merge(total_N_fert_corrected, mask_continents)
# specify path to folder to save file
                writeRaster(total_N_fert_corrected, filename = paste0("Script_generated_files/", "total_N_fert_corrected"), form
                        total_N_fert_corrected <- raster("Script_generated_files/total_N_fert_corrected.tif")

                                # Compute N fertiliser input and losses statistics and update the statistic result matrix

                         # 1. N leaching losses share
                                N_lost_leaching <- 1-((cellStats((total_N_fert * mask_correction_leaching), sum)) / (cel
                                # 2. total N losses share
                                N_lost_total <- 1-((cellStats(total_N_fert_corrected, sum)) / (cellStats(total_N_fert, s
                                # 3. total N input in fertilisers, and input after acconting for losses
                                N_fert_stats <- data.frame(cellStats(stack(total_N_fert, (total_N_fert*mask_correction_l
                                N_fert_stats <- rbind(N_fert_stats, N_lost_leaching)
                                N_fert_stats <- rbind(N_fert_stats, N_lost_total)
                                # set rows names
                                row.names(N_fert_stats) <- c("total_N_fertiliser", "fertiliser_corrected_leaching", "fer
                                names(N_fert_stats) <- c(paste0("simulation.","1"))
                                # add new rows main database
                                Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_fert_stats)
```

4.4.2 N fixation input

```
## --- BNF - legumes crop species ---

 # coefficient for estimating N fixation via BNF
       # see documentation for information on the model
       fixation_coefficients <- crop_info$NiC*crop_info$X1.NHI*crop_info$Ndfa*crop_info$Proot

       # calculate BNF for leguminous crop species and save to files
       foreach (i = 1:nlayers(conventional_rasters_production_DM), .packages='raster') %dopar% {
                multipl <- conventional_rasters_production_DM[[i]]*fixation_coefficients[i]
                writeRaster(multipl, filename = paste0("Script_generated_files/N_fixation/conventional/", crop_info$CropName[i])
                }
                                dir_fix_conv <- c("Script_generated_files/N_fixation/conventional/")
```

```
                        files_fix_conv <- sort(list.files(path = dir_fix_conv, pattern = ".tif"))
                        rasters_fixation_conventional <- stack(parLapply(cluster, paste0(dir_fix_conv, files_fix_conv),

                        # calculate total N input via BNF, and save to file
                        # Losses due to leaching from NBF are already estimated
                        # when accounteng losses in crop residues (avoiding double counting)
                        # Tons/gridcell
total_fixation_conventional <- clusterR(rasters_fixation_conventional, calc, args=list(fun=sum.na),filename=paste0("Scri
                        total_fixation_conventional <- raster("Script_generated_files/total_rasters_fixation_conventiona

                        # compute N fertiliser input and losses statistics and update the statistic result matrix

                 # 1. Total N from BNF
                        N_fix_conv <- data.frame(cellStats(total_fixation_conventional, sum))
                # set rows names
                        row.names(N_fix_conv) <- c("N_fix_conv")
                        names(N_fix_conv) <- c(paste0("simulation.","1"))

                        # add rows main database
                                Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_fix_conv)



                                        ## --- N fixed by free living bacterias (cyanobaterias) ---

             # compute coefficient for N fixed by cyanobatersia
                                # cereals, tubers and oilseed crop species fiix 0.012 in tons/ha N, Liu et al. 2010

                cereals_tuber_oils <- 0.012 #
                roots <- which(crop_info$CropGroup=="roots")
                cereal <- which(crop_info$CropGroup=="cereal")
                sec.cereal <- which(crop_info$CropGroup=="sec.cereal")
                oilcrops <- which(crop_info$CropGroup=="oilcrops")
        N_cyanobacteria <- array(dim=c(1,61))
                        N_cyanobacteria[roots] <- N_cyanobacteria[cereal] <- N_cyanobacteria[sec.cereal] <- N_cyanobacte
                #  sugarcane fixes 0.1 tons/ha N, Liu et al. 2010
                                N_cyanobacteria[50] <-  0.1
                #  rice fixes 0.02 tons/ha N, Liu et al. 2010
                        N_cyanobacteria[43] <-  0.02
                        N_cyanobacteria <- ifelse(!is.na(N_cyanobacteria), N_cyanobacteria, 0)


                #  calculate N fixed by cyanobacterias, save to file
                foreach (i = 1:nlayers(conventional_rasters_area), .packages='raster') %dopar% {
        multipl <- conventional_rasters_area[[i]] * N_cyanobacteria[i]
        writeRaster(multipl, filename = paste0("Script_generated_files/N_fixation/conventional/Cyanobacteria/", crop_inf
        }

                        dir_fix_conv_cyano <- c("Script_generated_files/N_fixation/conventional/Cyanobacteria/")
                        files_fix_conv_cyano <- sort(list.files(path = dir_fix_conv_cyano, pattern = ".tif"))
                        rasters_fixation_conventional_cyano <- stack(parLapply(cluster, paste0(dir_fix_conv_cyano, files

                                #  calculate total N fixed by cyanobacterias for all crops, save to file
        total_fixation_conventional_cyano <- clusterR(rasters_fixation_conventional_cyano, calc, args=list(fun=sum.na),
```

```
                total_fixation_conventional_cyano <- raster("Script_generated_files/total_fixation_conventional_cyano.tif")

                                # compute N fixed input and losses statistics and update the statistic result matrix

                        # 1. Total N from cyanobacterias
        N_fix_conv_cyano <- data.frame(cellStats(total_fixation_conventional_cyano, sum))
               # set rows names
        row.names(N_fix_conv_cyano) <- c("N_fix_conv_cyano")
        names(N_fix_conv_cyano) <- c(paste0("simulation.","1"))
        # add rows main database
         Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_fix_conv_cyano)
```

4.4.3 Crop residues input

```
        ## --- Crop Residues return to soil  ---

# Multiply by spatial explicit map reporting the share of crop residues
# recycled to soil for each gridcell
        crop_residues_N_input <- total_conventional_crop_N_demand_crop_residues * crop_residues_recycling_factor_conventional

                                        # estimate N from crop residues available after losses (IPCC)

   # 1. direct emissions coefficient: 0.01 --> range: 0.003-0.03
        total_crop_residues_N_input_corrected_losses <- crop_residues_N_input * 0.99
        # 2. losses due to N leaching
    total_crop_residues_N_input_corrected_losses <- total_crop_residues_N_input_corrected_losses - (crop_residues_N_input * (1-m

                # compute N crop residues input and losses statistics
      # and update the statistic result matrix
                                N_crop_res_to_soil <- data.frame(cellStats(total_crop_residues_N_input_corrected_losses, sum))
                                   # set rows names
                                row.names(N_crop_res_to_soil) <- c("N_crop_res_to_soil")
                                names(N_crop_res_to_soil) <- c(paste0("simulation.","1"))
                                # add rows main database
                                        Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_crop_res_to_soil)
```

4.4.4 Atmospheric depositions input

```
## --- N Atmospheric depostions ---
                        # load dataset Dentner et al. 1993
        N_deposition <- raster("N atmospheric deposition/N_deposition_grid_cell.tif")
        cropland_share <- raster("/home/pbarbieri/global/cropland2000_area.tif")
        #calculate depostion on harvested area and correct for losses due to leaching (5)IPCC)
        N_deposition <- N_deposition * cropland_share * mask_correction_leaching

        # compute N form deposition input and losses statistics and update the statistic result matrix
                                N_dep <- data.frame(cellStats(N_deposition, sum))
                                                                # set rows names
                                row.names(N_dep) <- c("N_deposition")
                                names(N_dep) <- c(paste0("simulation.","1"))
                                        # add rows main database
```

```
                                    Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_dep)
```

4.4.5 wastewater N input

```
        ## --- wastewater production ---
# load datased (personal estimation) of N available in wastewater for each gridcell
        waste_water_nitrogen <- raster("/home/pbarbieri/global/wastewater/Global_map_waste_water_available_Nitrogen_2015_estimat
                                       # correction of errors in wastewater maps
                                       # --> whatever more than 2500 t gridcell give 2500
                               rclmat <- matrix(c(2500, Inf, 2500), ncol=3, byrow=T)
                               waste_water_nitrogen <- reclassify(waste_water_nitrogen, rclmat)

        # estimate N from fertilisers available after losses (IPCC)

   # 1. Correction to account for the N used by crops non considered in this study
                               # (only 61 crops species are here considered): 0.04
   # 2. emissions coefficient for direct N2O and N2: 0.0166
         # 3. emissions coefficient for NH3 and NO: 0.2 --> range: 0.05-0.5
        total_N_waste_water_corrected_95crops <- waste_water_nitrogen * 0.96 * (1 - (0.0166 + 0.2))
        # 4. leaching losses coefficient: 0.3 --> codede as the available share -i.e. 0.7
                        leaching_areas_mask <- raster("Climatic and irrigation data/mask_area_leaching_final_irrigation_above_00
                        mask_correction_leaching <- leaching_areas_mask * 0.7
                        mask_correction_leaching <- reclassify(mask_correction_leaching, cbind(0,1))
        total_N_waste_water_corrected_95crops <- total_N_waste_water_corrected_95crops - (waste_water_nitrogen * 0.96 * (1 - mas

                # compute N wastewater input and losses statistics and update the statistic result matrix
                               N_waste_water <- data.frame(cellStats(total_N_waste_water_corrected_95crops, sum))
                                   # set rows names
                               row.names(N_waste_water) <- c("N_waste_water")
                               names(N_waste_water) <- c(paste0("simulation.","1"))
                                # add rows main database
                                       Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_waste_water)
 # write to file
                writeRaster(total_N_waste_water_corrected_95crops, filename="Script_generated_files/final_wastewater_corrected_l
                final_waste_water_N <- total_N_waste_water_corrected_95crops
```

4.4.6 Manure N input

```
                ##  --- MANURE PRODUCTION ---
                # Load manure estimates following IPCC procedure and Sheldric et. al corrected excreta coefficients (see document
                dir_livestock_N <- c("/home/pbarbieri/global/N manure avail ipcc global medium boundary/")
                files_livestock_N <- sort(list.files(path = dir_livestock_N, pattern = ".tif"))
                rasters_livestock_N <- stack(parLapply(cluster, paste0(dir_livestock_N, files_livestock_N), raster))

                # calculate total manure available in each gridcell --> Tons N/gridcell
total_N_manure <- clusterR(rasters_livestock_N, calc, args=list(fun=sum.na), filename=paste0("Script_generated_files/", "total_N
                total_N_manure <- raster("Script_generated_files/total_N_manure.tif")
                 # correction of outliers in Manure maps --> whatever more than 5000 t gridcell give 5000
                rclmat <- matrix(c(5000, Inf, 5000), ncol=3, byrow=T)                          #
                total_N_manure <- reclassify(total_N_manure, rclmat)
```

```
            # estimate N from fertilisers available after losses (IPCC)

# 1. Correction to account for the N used by crops non considered in this study
            # (only 61 crops species are here considered): 0.04
# 2. emissions coefficient for direct N2O and N2: 0.0166
     # 3. emissions coefficient for NH3 and NO: 0.2 --> range: 0.05-0.5
            total_N_manure_corrected_95crops <- total_N_manure * 0.96 * (1 -  (0.0166 + 0.2))
    # 4. leaching losses coefficient: 0.3 --> codede as the available share -i.e. 0.7
            total_N_manure_corrected_95crops <- total_N_manure_corrected_95crops - (total_N_manure * 0.96 * (1 - mask_correc



            # compute N manure input and losses statistics and update the statistic result matrix
            # 1. % N from manure lost in leaching
                                        N_manure_lost_leaching <- 1-((cellStats((total_N_manure * 0.96 * mask_correction
#  3. absolute N lost in leaching and total N from manure lost
                                        N_manure_lost_total <- 1-((cellStats(total_N_manure_corrected_95crops, sum)) / (
                                        N_manure_stats <- data.frame(cellStats(stack((total_N_manure*0.96), (total_N_mar
                                        N_manure_stats <- rbind(N_manure_stats, N_manure_lost_leaching)
                                        N_manure_stats <- rbind(N_manure_stats, N_manure_lost_total)
                                         # set rows name
                                        row.names(N_manure_stats) <- c("total_N_common_manure", "common_N_manure_correct
                                        names(N_manure_stats) <- c(paste0("simulation.","1"))
                                        # add rows main database
                                Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_manure_stats)
```

4.5 Part C: Conventional N budgeting, calculation of manure and wastewater N surplus

Below is the code that conventional N budgets, in each single gridcell, as described in Section 3. After having calculate the budgets, we also estimate, in each gridcell, the N manure and wastewater resources that are exceeding, i.e. the amount of N that is not used to balance convention crops' N demand. Note that N supply to conventionally managed soils is allowed to exceed conventional crops' N demand by a *overfertilisation rate*, as defined by the user. In our study, the *overfertilsiation rate* was set at 20% of total crops' N demand.

```
## --- Nitrogen Budget ---

       # A. computing first step conventional budget convetional considereing only chemical fertilisers, fixation and deposition
       # set the global share of conventional agriculture production via the coefficient "global_production_share_coefficient_co
       # allow to increase the N inputs by increasing crop demand by an overfertilisation factor, defined via de coeffcient "con
       budget_N <- ((mosaic((global_production_share_coefficient_conv * total_fixation_conventional),
                   (global_production_share_coefficient_conv * total_fixation_conventional_cyano),
                               (global_production_share_coefficient_conv * N_deposition),
                               (global_production_share_coefficient_conv * total_N_fert_corrected),
                               (global_production_share_coefficient_conv * total_crop_residues_N_input_corrected_losses),  fun=
                               (global_production_share_coefficient_conv * total_conventional_crop_N_demand_crop_residues), fun

                               # load N in manure
                               # set the global share of conventional agriculture production via the coefficient "global_product
       manure_N_conventional <- global_production_share_coefficient_conv * total_N_manure_corrected_95crops
```

```
        # B. input N from manure necessary to fill ells with a negative budgets and calculate the surplus
        # 1. Create a mask map containing 1 where the budget is negative
        mask_1_N<-(budget_N<0)
        # 2. creaete a manure map containing values only where the budget was negative
        manure_N_masked <- manure_N_conventional*mask_1_N
         # 3. create a map containing budgets values only where budget is negative
        budget_N_2 <- budget_N*mask_1_N
        # 4. create new budgets by adding X% of manure production to cells where budget was < 0.
        # This is not yet the final conventional budget since it contains also the surplus of
        # manure and it does not contain the cells that were already positive in budget_N
        budget_N_3 <- budget_N_2 + manure_N_masked
        # 5. select cells with positive values after manure add. The values represent the surplus of manure
        mask_2_N <- budget_N_3 > 0
        # 6.  map the surplus of manure (positive cells)
        surplus_N <- budget_N_3*mask_2_N
        # 7. create a mask map containing 1 where the budget is positive or zero
        mask_3_N <- budget_N >= 0
        # 8. create a map containing budgets values only where budget is positive or zero
        manure_N_masked_2 <- manure_N_conventional*mask_3_N
        # 9. create a map containing the total surplus of manure
        manure_N_surplus_total <- mosaic(manure_N_masked_2, surplus_N, fun=sum.na)
        manure_N_surplus_total <- extend(manure_N_surplus_total, extent(-180,180,-90,90))

        #B write to file: input correct file name and folder path
    writeRaster(manure_N_surplus_total, filename = "Script_generated_files/manure_N_surplus_total_100_manure_cropland_20_convent

    # 10. create a map of the budget cells that were originally > 0
        budget_N_positive <- budget_N * (budget_N >= 0)
        # 11. create a map of the budget cells that were originally < 0 by budgeting them to 0 i.e. eliminating the manure surplu
        budget_N_without_surplus <- budget_N_3 - surplus_N
        # 12. merge the two complementary maps
        budget_N_conventional <- mosaic(budget_N_positive, budget_N_without_surplus, fun=sum.na)


                # C. input N from wastewater necessary to fill cells with a negative budgets and calculate the surplus
# 1. create a mask map containing 1 where the budget is negative
        mask_1_N<-(budget_N_conventional<0)
        # 2. create  a wastewater_N map containing values only where the budget was negative
        wastewater_N_masked <- final_waste_water_N * mask_1_N
        # 3. create a map containing budgets values only where budget is negative
        budget_N_2 <- budget_N_conventional * mask_1_N
        # 4. create new budgets by adding 70% of manure production to cells where byfget was < 0.
        # This is not yet the final conventional budget since it contains also the surplus of manure
        # and it does not contain the cells that were already positive in bundget_N
        budget_N_3 <- budget_N_2 + wastewater_N_masked
        # 5. select cells with positive values after manure add. The values represent the surplus of manure
        mask_2_N <- budget_N_3 > 0
        # 6. map the surplus of manure (positive cells)
        surplus_N_wastewater <- budget_N_3 * mask_2_N
        # 7. create a mask map containing 1 where the budget is positive or zero
        mask_3_N <- budget_N_conventional >= 0
        # 8. create a map containing budgets values only where budget is positive or zero
```

```
        wastewater_N_masked_2 <- final_waste_water_N * mask_3_N
         origin(wastewater_N_masked_2) <- c(0,0)
         origin(surplus_N_wastewater) <- c(0,0)
         # 9. create a map containing the total surplus of manure
   wastewater_N_surplus_total <- mosaic(wastewater_N_masked_2, surplus_N_wastewater, fun=sum.na)

   # 10. create a map of the budget cells that were originally > 0
        mask_N_positive <- budget_N_conventional >= 0
        budget_N_positive <- budget_N_conventional * mask_N_positive
                # 11. create a map of the budget cells that were originally < 0 by budgeting them to 0 i.e. eliminating the manur
        budget_N_without_surplus <- budget_N_3 - surplus_N_wastewater
                # 12. merge the two complementary maps
        budget_N_conventional_final <- mosaic(budget_N_positive, budget_N_without_surplus, fun=sum.na)

        # write to file: check correct file name and path to saving folder
                writeRaster(budget_N_conventional_final, filename = "Script_generated_files/budget_N_conventional_100_manure_cro
                writeRaster(wastewater_N_surplus_total, filename = "Script_generated_files/wastewater_N_surplus_total_100_manure

                # computes budgets eleiminating the effect of the overfertilisation allowed at step 1.
        budget_N_conventional_corrected_overfertilisation <- budget_N_conventional + (global_production_share_coefficient_conv *
        # All in all, the overfertilisation serves to account the OVERUSE of N applied in comparison to crop demand.
        # Therefore, to know if the REAL crop needs are satisfied, I need to eliminate from the budgets the artifical"
        # part of the crop demand that accounts for this overfertilisation.
        # Otherwide my budgets might be negative as referred to the augmented artificail demand of crops and not to their real de
                writeRaster(budget_N_conventional_corrected_overfertilisation, filename = "Script_generated_files/budget_N_conve


        # compute N demand statistics and update the statistic result matrix
                # calculate statistics
                                conv_manure_percent_surplus <- ((cellStats(manure_N_surplus_total, sum)) / (cellStats(ma
                    wastewater_percent_surplus <- ((cellStats(wastewater_N_surplus_total, sum)) / (cellStats(final_waste_water_N
                                N_conv_budget_stats <- data.frame(cellStats(stack(budget_N, manure_N_surplus_total, budg
                                N_conv_budget_stats <- rbind(N_conv_budget_stats, conv_manure_percent_surplus)
                                names(N_conv_budget_stats) <- c(paste0("simulation.","1"))
                    N_conv_budget_stats <- rbind(N_conv_budget_stats,wastewater_percent_surplus)
                    # set rows names
                    row.names(N_conv_budget_stats) <- c("conv_N_budget_no_manure", "conv_N_manure_surplus", "N_budget_conventior
# add rows main database
                    Output_statistics_dataframe <- rbind(Output_statistics_dataframe, N_conv_budget_stats)

# Write to file statistic table, check path to saving folder
write.table(Output_statistics_dataframe, "Script_generated_files/Conventional_budgeting_statistics_100_manure_to_cropland_20_cor
```

5 GOANIM model

The GOANIM-model is a spatially explicit, biophysical and linear optimization model simulating cropland soil N cycling in organic farming systems and its feedback effects on food production at the global scale (supply-side of the food systems). GOANIM was coded in two versions:

- Version I (*GOANIM V1*) is a linear optimisation model that estimates (i) organic crops yields in each geographic location (i.e. grid-cell) as a linear function of the N supply to organic cropland soils and

(ii) organic livestock animal population densities in each location as function of the local available feed resources. The model objective is to maximise total food production (from crops and livestock food commodities). This version of the model actively estimates livestock numbers for each grid-cell, thus redesigning the organic livestock sector across the globe. The GOANIM (V1) estimates organic crop yields and livestock densities for any global conversion share to organic farming set by the user, from 0% to 100%.

- Version II (*GOANIM V2*) is a linear optimisation model that maximises organic crop yields as a liner function of the N supply to organic cropland soils. Livestock animal population densities are kept as the current -i.e. conventional- ones and the total food output is calculated as the difference between the total cropland production minus the fraction of the cropland production used to feed current livestock animal populations worldwide. The GOANIM (V2) estimates organic crop yields for any global conversion share to organic farming set by the user, from 0% to 100%.

5.1 GOANIM model - V1

5.1.1 Objective function

The model objective function is to maximise the total food energy produced from organic food commodities, considering both cropland and livestock production, as indicated in equation (11):

$$Max: \sum_{i=1}^{61} A_i \times Y_{food_i} \times [E]_i + \sum_{k=1}^{s} LIV_k \times Y_{food_k} \times [E]_{livestock-products_k} \tag{10}$$

where A_i is the area cultivated under each crop species i $[ha]$, Y_{food_i} is the food sub-yield of any crop species i $[tons\ ha^{-1}]$, $[E]_i$ is the energy density of any crop species i $[MJ\ tons^{-1}]$, LIV_k is the livestock density of any livestock species k, Y_{food_k} is the food yield of any livestock species k, and $[E]_{livestock-products_k}$ is the energy density of any k livestock type food product $[MJ\ tons^{-1}]$.

A few crop species out of the 61 covered in this study do not provide any energy for food (e.g. tea, coffee.). Therefore, the model would automatically set to zero the yield of such crops -i.e. no N would be allocated to these crops, even in the case of plenty available N resources. This is because such crops do not contribute to the objective function. To avoid this unrealistic effect, these crops were assigned a very low energy density (e.g. 1e-6 $MJ\ kg^{-1}$). In this way, their contribution to the objective function is almost null, but still positive. The model will then allocate N to these crops, if available after allocation to all the other crops species.

5.1.2 Crops yield response curve to N fertilisation

Organic crop and crop residues yields are simulated as a function of the total N supplied to cropland soils via a crop species-specific linear N-response curve (Equation 11).

The plateau of each yield curve is estimated as the current -i.e. conventional- dry matter yield corrected by an organic-to-conventional yield gap (referred onward as organic maximum yield). Current conventional yields are indirectly a function of local pedo-climatic conditions.

$$IF\ N_i \leq N_{max_i}, THEN\ Y_i = \frac{N_i}{N_{max_i}} \times Y_{max_i} ELSE\ Y_i = Y_{max_i} \tag{11}$$

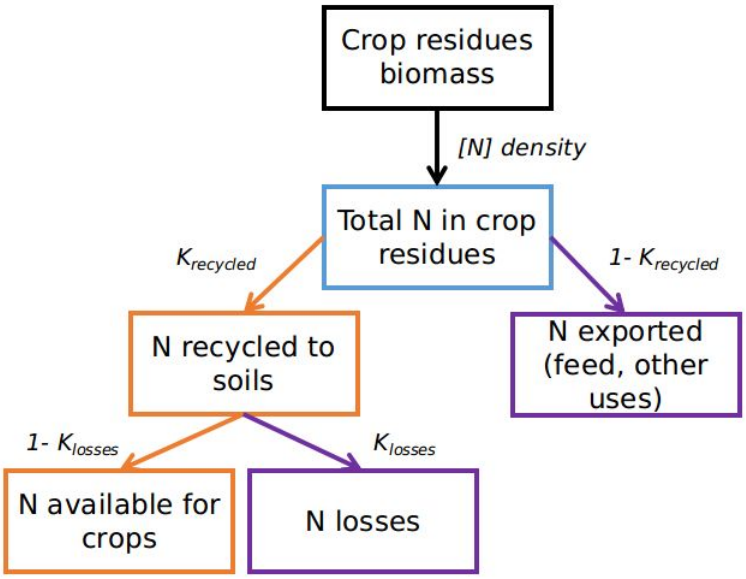

Figure 3: Estimation of organic crop residues yield and N demand

Where Y_i is the simulated organic yield [$tons\ DM\ ha^{-1}$], N_i is the N input to cropland soils [$tons\ N$], N_{max_i} is the N input to cropland soils that allows Y_{max_i} [$tons\ N$], and Y_{max_i} is the maximum attainable yield in organic farming [$tons\ DM\ ha^{-1}$], for any crop species i.

We estimate the organic maximum attainable yield in equation (11) by multiplying the current -i.e. conventional- yields (Monfreda et al. 2008) by crop species-specific organic-to-conventional yield gaps from recent literature [31]. We corrected these organic-to-conventional yield gaps in order to account for a yield reduction due only to pest and diseases, i.e. not accounting for the yield reduction due to N deficiency. This is because the GOANIM model itself simulates the yield reduction due to N limitation. As in a previous study [32], we applied the organic-to-conventional yield gaps [31] only to industrialised agriculture with high conventional yields, which we define as regions where conventional yields are higher than 55% of their potential attainable value (based on the conventional yield gap analysis by Licker et al. [33] and Mueller et al. [7]). We made this assumption because the data collected by [31] are largely restricted to developed countries and high-yielding conventional systems and we currently do not have sufficient scientific data on relative organic yields in developing countries with low-input conventional agriculture [34].

Total dry matter production is then calculated by multiplying the simulated crop yields by the corresponding area (organic land use, from Barbieri et al. [1]), see section 4.1.7).

Crop residues yield is estimated as a function of the harvested yield, as in equation (6). The fraction of N inputs derived from crop residues recycled to soil (IN_{res}) is accounted by considering only the crop residues N requirements of the fraction exported from soils, plus the losses occurring after incorporating the fraction of crop residues recycled to soils (violet boxes in Figure 3), resulting in the linear equation (12).

$$Y_i \times RPR_i = N_{res-exported_i} \times \frac{1}{[N]_{res}} \times \frac{1}{[\beta + ((1-\beta) \times \gamma)]} \tag{12}$$

where Y_i is the simulated organic yield [$tons\ DM\ ha^{-1}$], RPR_i is the residue-to-product ratio, $N_{res-exported_i}$

is the N content of the exported fraction of the crop residues, $[N]_{res_i}$ is the N density of the crop residues, β is the crop residues removal factor, and γ is the coefficient estimating losses after the soil incorporation of the N contained the recycled fraction of the crop residues, for each crop specie i. γ is estimated based on the IPCC procedure [11], as explained for the conventional sub-model, at section 2.

5.1.3 Biological N fixation

Biological symbiotic N fixation is calculated as in equation (3). Non-symbiotic N fixation by free-living cyanobacteria is calculated using the coefficient reported in paragraph 2. The non-symbiotic N fixation is considered to be independent of the crop dry matter yield and it is therefore added the right-hand side of the model equations, as indicate in Table 4.

5.1.4 N from organic livestock

Organic livestock provides cropland soils with manure (IN_{man}). IN_{man} for each livestock species and country is calculated as the product of the yearly standing livestock heads and livestock species-specific excretion coefficients. Excretion coefficients are calculated as reported in section 2, - i.e. no differences in manure excretion are considered between organic and conventional farming systems -. The total produced manure is then corrected for losses (leaching, volatilisation and run off) due to storage and management, applying a modified version of the IPCC coefficients reported at section 2. In details, we used the same procedure and coefficient as in section 2, but we corrected the share of ruminant livestock species that are hold on permanent pastures. This is because organic farming land use has higher shares of temporary fodders compared to conventional systems (Barbieri et al. [1]). Therefore, we hypothesize that organic farmers would tend to prioritize the use of such temporary fodders, holding livestock proportionally less on permanent pastures. Hence, we decrease the share of livestock held on permanent pasture provided by IPCC proportionally to the increased proportion of temporary fodders in every single grid-cell. For instance, consider a share of a grid-cell with 100 ha of permanent pastures, 40 ha and 80 ha of temporary fodders in the conventional and organic land use, respectively, and a 40% of ruminants kept on permanent pastures. In the organic land use, the ratio of organic temporary fodders on permanent pastures is 0.8, while in conventional in 0.4. Therefore, temporary fodders increase by 50% in organic land use compared to the convention counterpart. Hence, the share of organic livestock kept on permanent pastures is decreased by 50%, leading to a share of 20% of ruminants kept on permanent pastures.

5.1.5 Nitrogen Budgets

Crops N demand and N supply are balanced via the budgeting equation (13):

$$OUT_{crop} + OUT_{res} + OUT_{pool} = IN_{fix} + IN_{man} + IN_{dep} \tag{13}$$

Where: OUT_{crop} is sum of each N_i inputs necessary to obtain the yield Y_i in equation (10), OUT_{res} is the sum of each $N_{res-exported_i}$ in equation (11), IN_{fix} is the sum of the BNF fixed by each legume crop species, IN_{man} is the N in manure provided by organic livestock and IN_{dep} is the total dry and wet N atmospheric deposition, as described in Section 2. OUT_{pool} represent eventual additional losses to the environment, in

the event that $OUT_{crop} + OUT_{res} < IN_{fix} + IN_{man} + IN_{dep}$. This implies that an eventual N surplus for a given crop cannot be transfer to others crops species in the same grid-cell. All variables are in $[tons\ ha^{-1}]$ and are then multiplied by respective crop species areas, described at paragraph 4.1.7.

5.1.6 Additional N resources

The organic sub-model is able to selectively add additional N input resources (Figure 2). These resources are the N surplus outputs of the conventional budgeting model add-in described in section 2 (i.e. δIN_{man}, and δIN_{water}). These surpluses can be selective added by the user to the available N resources for cropland application by activating binary variables according to equation (14):

$$IN_{man_{tot}} = IN_{man} + s \times \delta IN_{man} + t \times \delta IN_{water} \tag{14}$$

Where s and t are dummy binary variables activating or deactivating the accounting of additional N resources.

5.1.7 Croplands food and feed production

The total crops' dry matter yield calculated as in paragraph 4.1.2 is then partitioned into a food and a feed yield pools, in a way that, for any give crop species i, $Y_i = Y_{food_i} + Y_{feed_i}$. The model is set free to allocate the total dry matter yield either to food or to feed, up to a maximum allowed threshold for feed allocation (Supplementary dataset I). This threshold correspond to the current global use of each crop species as a feed resource, according to FAOSTAT statistics [10]; temporary fodder crop species are completely allocated to feed.

5.1.8 Organic crop harvested areas

Organic crop harvested areas (land use) for any given crop species has been taken from Barbieri et al. [1]. Organic areas are then multiplied by the conversion share to organic farming (from 100% to 0%) chosen by the user. This conversion coefficient is equal to $(1 - global\ share\ conventional)$, where *global share conventional* is the conventional global share defined in Section 2.

5.1.9 Estimation of livestock densities

The model estimates livestock (adult producing animals) population densities in each given grid-cell as a function of (i) the local available feed resources and (ii) the ability of each livestock species to provide N for application to cropland soils. The model does not simulate any livestock structure, i.e. animals of different ages and non-productive animals.

Livestock diets We define livestock dietary requirements as the metabolisable energy (ME) and crude protein (CP) necessary to cover the dietary needs of any adult animal[2] over one full year. Reference data are reported in Table 3. We calculated the ratio between each of the reference ME and CP requirements

and the reference live weights, obtaining the feed requirement per kg of live weight for any livestock type (expressed respectively in $[MJ\ kg-live-weight^{-1}]$ and $[kg\ CP\ kg-live-weight^{-1}]$). We used these values to regionalise livestock dietary requirement based on the live weights of the various livestock types in each of the 162 considered countries. We collected data on live weight from FAOSTAT [10], as for the IN_{man} calculations. For dairy cows, we also separately considered the energy and protein requirements for each litre of produced milk (respectively 5 $MJ\ ME\ l^{-1}$ and 0.1 $kg\ CP\ l^{-1}$). We regionalised this additional requirements using FAOSTAT country data about milk yields.

The total ME and CP dietary requirements were then split into three feed categories, i.e. grains, fodders and crop residues, using data from Herrero et al. [35][3]. In order to be consistent with organic principles, we verified that the share of feed from fodders was above 50% [4]. Note that how our estimation of livestock feeding requirements are conservative, since based on the current practices. An increased share of organic farming at the global scale could come with a modification of such practices, for example increasing the share of fodders to the detriment of grains. Nevertheless, our approach, even if conservative, allows to keep calculation transparent and consistent with each other.

The modelled animal populations densities accounts for the standing producing animals at the end of the simulated one-year time-frame. Nonetheless, some livestock types have a production turnover lower than one year. For those types (meat goats and sheep, pigs and broilers) we estimated the total annual population in order to correctly assess the total food production from livestock food commodities. To do so, we calculated the number of animals produced in one year by dividing, for each livestock species, the average slaughter weight by an average daily weight gain. For pigs and broilers, we used the average daily weight reported in table 3, for meat goats and sheep we used regionalise average daily weight gain estimates from Herrero et al. [35]; Broilers' number of heads per year were calculated considering 40 unproductive days per year to allow for sanitation procedures [36]. For pigs we differentiated between intensive and extensive production systems, following the data reported by Herrero et al. [35].

Simulation of livestock densities Standing livestock population densities are simulated by GOANIM by matching the dietary ME and CP feed requirement of each livestock standing unit with the available feed resources in any given grid-cell (equations 15-17 and 18-20). Livestock numbers are calculated in number of heads, and then transformed in Livestock Units by applying the appropriate FAO's Tropical Livestock Unit conversion factor.

$$\sum_{i=1}^{s} LIV_i \times E_{grain-req_i} \leq \sum_{k=1}^{z} A_k \times Y_k \times [E]_k \tag{15}$$

$$\sum_{i=1}^{s} LIV_i \times E_{fodder-req_i} \leq \sum_{k=1}^{z} A_k \times Y_k \times [E]_k \tag{16}$$

$$\sum_{i=1}^{s} LIV_i \times E_{stover-req_i} \leq \sum_{k=1}^{z} A_k \times Y_k \times RPR_k \times \beta \times [E]_k \tag{17}$$

[2]We subdivided the five considered livestock species into 9 sub-categories (livestock types) -respectively dairy cows, beef, dairy and meat goats, dairy and meat sheep, pigs, broilers and hens

[3]Herrero et al. distinguished four feed categories: grains, crop residues, fodders and occasional (cut and carried fodders, legumes, other planted forage). Our fodder category is the sum of Herrero's fodder and occasional original categories

Table 3: Reference livestock feed requirements and production parameters expressed in mebolisable energy and crude protein

Livestock type	**Ref. live weight** $[kg]$	**Energy** $[MJ\ ME\ day^{-1}]$	**Protein** $[kg\ CP\ day^{-1}]$	**Ref. daily weight gain** $[g]$	**Ref. daily production** $[l/no.]$	**Sources**
Dairy cows	400	42.80	0.43		13.7	[10]
Beef	325	32.80	0.94	1500		Adapted from [37]
Dairy goats	27.5	10.85	0.26		1	[38]
Meat goats	15	6.25	0.10	77		[38]
Dairy sheep	40	19.37	0.40		21	[39]
Meat sheep	17.5	8.87	0.14	77		[39]
Pigs	50	12.22	0.13	275*, 617**		[35], [40], [41]
Broilers	2	1.31	0.02			[36]
Laying hens	2	1.17	0.08		1	[42], [43]

* in non-industrial, i.e. extensive systems; ** in industrial systems

where: LIV_i is the number of heads for animal species i and $E_{grain-req_i}$, $E_{fodder-req_i}$, and $E_{crop-residues-req_i}$ is ME the feed requirements from grain, fodder, and crop residues, respectively for any animal species i; A_k is the harvested area under crop species k, Y_k is the dry matter yield of any crop species k, RPR_k is the residue-to-product ratio (described in section 2), β is the removal factor for crop residues (described in section 2), and $[E]_k$ is the energy density of feed category k.. All coefficient values are reported in the Supplementary Dataset I.

$$\sum_{i=1}^{s} LIV_i \times P_{grain-req_i} \leq \sum_{k=1}^{z} A_k \times Y_k \times [P]_k \tag{18}$$

$$\sum_{i=1}^{s} LIV_i \times P_{fodder-req_i} \leq \sum_{k=1}^{z} A_k \times Y_k \times [P]_k \tag{19}$$

$$\sum_{i=1}^{s} LIV_i \times P_{stover-req_i} \leq \sum_{k=1}^{z} A_k \times Y_k \times RPR_k \times \beta \times [P]_k \tag{20}$$

where: $P_{grain-req_i}$, $P_{fodder-req_i}$, and $P_{stover-req_i}$ are the feed CP requirements from grain, fodder, and crop residues requirements for any livestock type i (as described in paragraph 3.2.6.1); $[P]_k$ is the protein density of any given harvested crop product or crop residues, for any crop species k. All coefficient values are reported in the Supplementary Dataset I.

5.1.10 Permanent grassland

Livestock fodder dietary requirement can be satisfied by temporary fodder crop species, whose yield is simulated as described at paragraph 4.1.2, or by permanent grassland fodder production. We calculated the energy and protein production by permanent grasslands following the procedure proposed by Fetzel et al. [44]. In details, we estimated the permanent grassland fodder biomass production by combining the

MODIS-based (derived using remote sensing techniques) Net Primary Productivity (NPP) [45] maps with a global dataset of the distribution of permanent grasslands worldwide, developed by Ramankutty et al. [46]. We considered only the above ground fraction of the total NPP (aNPP), by assuming an aboveground to total NPP proportion of 60% [44]. The resulting aNPP was converted to dry matter biomass considering a carbon content factor of 50% [47]. We then transformed the dry matter biomass in total available energy and protein by applying a ME density coefficient of 10.02 and 9.85 $MJ\ ha^{-1}$ and crude protein density of 0.2269 and 0.1251 $Kg\ Kg\ DM^{-1}$ for temperate and tropical regions, respectively.

5.1.11 Additional constraints

GOANIM comes with a set of additional agronomic constraints, which can be activated/deactivated by the user.

1. Minimum crop yield The user can force the minimum yield of any give crop category[4], as in equation (21):

$$\sum_{i=1,k=1}^{n,10} Y_{i,k} \geq P \times \sum_{i=1,k=1}^{n,10} Yield_{potential_{i,k}} \tag{21}$$

where: $Y_{i,k}$ is the dry matter yield of any crop i within any crop category k, and P a coefficient in between 0 and 1.

In the present study, we only force the crop yield of non-food crops. This is because the model would assign nitrogen to non-food crops only at last, whereas these crops would simultaneously compete for N resources since they since they still have an important role for e.g. industrial purposes.

2. Minimum livestock no. of heads We include a set of constraints that allow the user to constraint the minimum livestock number of heads (or LU). This set of constraints does not apply to each single livestock type simulated by the model, but either to a certain livestock category (e.g. ruminant vs. monogastric, bovine vs. sheep vs. goats, etc.) or to a livestock products category (e.g. milk-producing animals vs. meat-producing animals, vs. eggs-producing animals). The constraint allows to set the minimum share of producing heads (or LU) of any category of choice in comparison to the total number of livestock heads (considering all the 9 livestock types) as in equation (21):

$$\forall k, \sum_{i=1}^{n} LIV_{i,k} \geq P \times \sum_{i=1}^{9} LIV_i \tag{22}$$

where $LIV_{i,k}$ is the livestock density of each livestock type i belonging to any defined category k (e.g. milk, meat or eggs), and P a coefficient in between 0 and 1.

[4]The 61 covered crop species are grouped in crop categories, according to Barbieri et al. [1], namely primary cereals, secondary cereals, pulses, root crops, oil crops, industrial crops, fodder crops, vegetables, fruits, and non-food crops. The composition of the give crop categories can be found in the Supplementary Dataset I

3. Maximum livestock no. of heads We include another set of constraints that allow the user to set the maximum number of heads for any given livestock type, as a percent of the total number of livestock types, as in equation (22):

$$\forall i, LIV_i \leq P \times \sum_{i=1}^{9} LIV_i \tag{23}$$

where LIV_i is the livestock density of each livestock type i and P a coefficient in between 0 and 1.

5.1.12 Outputs and statistics

The model output is represented by all the model defined variables. It is up to the user to decide which information to save.
In the current version, the model is set to save the following output variables:

- the total simulated dry yields of each single crop species;
- the simulated food sub-yields of each single crop species;
- the simulated feed sub-yields of each single crop species;
- the nitrogen fixed by biological nitrogen fixation (BNF) for leguminous crop species;
- the nitrogen from organic manures provided to each crop species;
- the livestock producing heads of each single livestock type;
- the livestock standing producing heads in LU;
- the total nitrogen available for crop uptake in organic manure;
- a set of in-model calculated statistics including:
  - total energy and protein in food from crops;
  - total energy and protein in animal products;
  - total energy and protein from temporary fodders crops;
  - total livestock fodder requirements (energy and proteins).

5.1.13 Thermodynamic principles

The model respects the thermodynamic prociples (low of the conservation of mass and energy). We checked for the total N input in livestock production (N in feed) to be equal to the total output in livestock production ($N\ in\ excreta + N\ in\ livestock\ food\ products + N\ in\ livestock\ animal\ body$). N outputs were overall higher than N inputs. We corrected the livestock excreta coefficient for each of the 164 considered countries in order to perfectly match the N input vs N output in the livestock sector to avoid any artificial creation of Nitrogen. We applied this corrected N excreta coefficient to both the Organic optimization sub-model (version I and II), as well as to the conventional sub-model, in order to harmonize the coefficient used throughout the full model.

Table 4: Simplified model structure as coded, reporting a simplified version of all equations coefficients. Color boxes represents the different structural matrices as coded in R

	Crop yield (61)	**N crop (61)**	**N stovers (61)**	**N pool (61)**	**N BNF (61)**	**N fert. (61)**	**Food sub-yield (61)**	**Feed sub-yield (61)**	**Stand liv. heads (9)**	**Total N avail. (1)**	**Live. heads (9)**	**Live. LU (9)**	**Stat. (11)**	**DIR**	**RHS**
Yield curve slope (61)	1	$-1/[N_i]$												=	0
Yield curve plateau (61)	1													≤	Y_{max}
Stovers yield (61)	RPR		$-K_{stove}$											=	0
NBF (61)	K_{fix}				-1									=	0
N budget (61)		A_i	A_i	A_i	$-A_i$	$-A_i$								=	$IN_{dep} + IN_{cyano}$
Avail. N fert. (1)						A_i				-1				≤	$conv\text{-}manure + ww.$
Tot. crop yield (61)	-1						1	1						=	0
Max. feed share (61)	$k_{feed-grain}$							-1						≥	0
Liv. grain req. (2)								$A_i \times K_{grain}$	$-Req_{grain}$					≥	0
Liv. fodder req. (2)								$A_i \times K_{fodder}$	$-Req_{fodder}$					≥	0
Liv. stover req. (2)								$A_i \times K_{stover}$	$-Req_{stover}$					≥	0
Manure (1)									k_{head}	-1				=	0
Liv. head (9)									k_{head}		-1			=	0
Liv. LU (9)									k_{LU}			-1		=	0
Min. liv. density (5)									$K_{head}/(K_{LU})$		$-C_{head_1}$	$(-C_{LU_1}$		≥	0
Max. liv. density (9)									$K_{head}/(K_{LU})$		$-C_{head_2}$	$(-C_{LU_2}$		≤	0
Min. crop yield (10)	C_{crop}													≥	$K_{constr.}$
Stat. (11)													$K_{stat.}$	=	0

Table 5: Simplified model structure as coded, reporting each single sparse sub-matrix (SM). Color boxes represents the different structural matrices as coded in R

	Crop yield (61)	**N crop (61)**	**N stovers (61)**	**N pool (61)**	**N BNF (61)**	**N fert. (61)**	**Food sub-yield (61)**	**Feed sub-yield (61)**	**Stand liv. heads (9)**	**Total N avail. (1)**	**Live. heads (9)**	**Live. LU (9)**	**Stat. (11)**	**DIR**	**RHS**
Yield curve slope (61)	SM1	SM2	SM3	SM4	SM5	SM6	SM7	SM8							
Yield curve plateau (61)	SM9	SM10	SM11	SM12	SM13	SM14	SM15	SM16							
Stovers yield (61)	SM17	SM18	SM19	SM20	SM21	SM22	SM23	SM24							
NBF (61)	SM25	SM26	SM27	SM28	SM29	SM30	SM31	SM32							
N budget (61)	SM33	SM34	SM35	SM36	SM37	SM38	SM39	SM40							
Avail. N fert. (1)	SM41	SM42	SM43	SM44	SM45	SM46	SM47	SM48							
Tot. crop yield (61)	SM49	SM50	SM51	SM52	SM53	SM54	SM55	SM56							
Max. feed share (61)	SM57	SM58	SM59	SM60	SM61	SM62	SM63	SM64							
Liv. grain req. (2)				SM65				SM66							
				SM71				SM72							
Liv. fodder req. (2)				SM67				SM68							
				SM73				SM74							
Liv. stover req. (2)				SM69				SM70							
				SM75				SM76							
Manure (1)					SM77										
Liv. head (9)					SM78										
Liv. LU (9)					SM79										
Min. liv. density (5)					SM80										
Max. liv. density (9)					SM81										
Min. crop yield (10)					SM2 to SM91										
Stat. (11)	SM93 to SM103, SM105														

6 Code - GOANIM-model - Version I

6.1 Copyrights and environment preparation

We load here all the R packages necessary to run the code. These packages include: raster [26], clpAPI [48], ROI [49], ROI.plugin.clp [50], slam [51], and the R packages for parallelism doParallel [29], and foreach [30].

The optimisation solver used to solve the optmisation problem was COIN-OR CLP https://projects.coin-or.org/Clp. Note that the model was run using an external server owned by the MCIA of the University of Bordeaux. Here, up to 23 cores were simultaneously used for several code steps due to the high RAM requirements. Hence, the code cannot directly run on a Desktop machine.

All spatial datasets (raster maps) are transformed into a linear array of dimension 1:9331200, where each value corresponds to a gridcell. The optimisation problem is then independently solved for each single gridcell. The optimisation results are then saved into a set of linear arrays of dimension 1:9331200. The linear arrays are then backtransformed into raster maps (Section 6.3.9).

```
#  Copyright 2019 Pietro Barbieri <pbarbieri@avakas-frontend2>
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.

# Load packages
library(raster)
library(clpAPI, lib.loc="/home/pbarbieri/R/x86_64-pc-linux-gnu-library/3.2")
library(ROI.plugin.clp, lib.loc="/home/pbarbieri/R/x86_64-pc-linux-gnu-library/3.2")
library(ROI)
library(slam)
library(doParallel)
library(foreach)

# --- Set Working Directory and Temporary files directory ---

# specify path to temporary file directory
setwd("/home/pbarbieri/global/Version_I_first_inclusion_of_land_use_change_rules/")
getwd()
library(unixtools)
set.tempdir("/scratch/pbarbieri/R_temporary_files")
tempdir()


# *************   ATT! CORE CONTROLS   *************
# set number of used core according to pc/server capabilities
detectCores()
cluster<-makeCluster(23)
#start cluster
```

```
registerDoParallel(cluster)
#check number of cores used
getDoParWorkers()
# Stop cluster
#registerDoSEQ()
```

6.2 Data and parameters loading

6.2.1 Load global binary parameters for scenario analysis

```
# --- Binary coefficients to activate/disactivate variables and global scenario coefficients ---

# 1. Global share of organic vs conventional farming production systems
                global_production_share_coefficient_conv <- 0
                global_production_share_coefficient_org <- 1 - global_production_share_coefficient_conv

 # 2. Activators for conventional farming inputs into organic farming systems
        # 0-1 Desactivate/Activate the use of conventional surplus manure.
        # Manure has already been accounted for losses in the conventional-addin
                        K_conv_manure_surplus <- 0
        # 0-1 Desactivate/Activate the use of conventional surplus manure.
        # Wastewater has already been accounted for losses in the conventional-addin
                        K_waste_water <- 0


        # Load dataset containing all main parameters values (see Supplementary Dataset)
                crop_info <- read.csv("crop_list_60.csv", header=T)
```

6.2.2 Load crops areas, and yield curve plateau

We load here the spatial datasets containing organic crop areas and organic crops' yield curve plateaux. Organic areas have been taken from Barbieri et al. (2019) [1].

```
                # create empty vectors for data loading
                # all raster datasets are transformend into vectors of dimension c(1, 9331200)
                # or three-dimensional vectors called here "matrices"
                # Organic potential yield
    matrix_yield_max_vec <- array(dim=c(1,9331200,61))
    # Organic areas hectares
                matrix_area_vec <- array(dim=c(1,9331200,61))

# Load organic maximum yield t/ha (Ponisio modified coefficients, see model documentation for details),
# Organic land use (areas ha) for the 61 considered crop species
                        foreach (i = 1:61, .packages="raster") %do% {
             # Load organic max yields with modified Ponision coefficents and tranform them in DM
            matrix_yield_max_vec[,,i] <- as.vector(raster(paste0("Organic yield files/Modified_Ponisio_organic_yield_gap/", crop_
            # Load organic areas in ha
            matrix_area_vec[,,i] <- as.vector(raster(paste0("/home/pbarbieri/global/Land Use Change/100% conversion/Area_hectares
    }
```

6.2.3 Load organic livestock dietary requirements

```
# --- Load livestock dietary requirements maps ---
 # set directoy path where datasets are saved
 # create ampty array for the 9 livestock species for Energy grain requirements
 # order of animal species as loaded: 1: cattle dairy, 2: cattle beef, 3: goats dairy,
 # 4: goats meant, 5: sheep dairy, 6: sheep meat, 7: pigs, 8: broilers, 9) hens
 E_grain_dietary_req <- array(dim=c(1,9331200,9))
                              E_grain_dietary_req[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,8] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grain_dietary_req[,,9] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                               # create ampty array for the 9 livestock species for Protein grain requirements
                               # transform coefficient from demand in Kg to Tons
  P_grain_dietary_req <- array(dim=c(1,9331200,9))
                              P_grain_dietary_req[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,8] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grain_dietary_req[,,9] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                               # create ampty array for the 9 livestock species for Energy fodder requirements
 E_grass_dietary_req <- array(dim=c(1,9331200,9))
                              E_grass_dietary_req[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              E_grass_dietary_req[,,8] <- 0
                              E_grass_dietary_req[,,9] <- 0
                               # create ampty array for the 9 livestock species for Protein fodder requirements
                                                       # transform coefficient from demand in Kg to Tons
 P_grass_dietary_req <- array(dim=c(1,9331200,9))
                              P_grass_dietary_req[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary req
                              P_grass_dietary_req[,,8] <- 0
                              P_grass_dietary_req[,,9] <- 0
                               # create ampty array for the 9 livestock species for Energy stover requirements
                               E_stovers_dietary_req <- array(dim=c(1,9331200,9))
```

```
                              E_stovers_dietary_req[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              E_stovers_dietary_req[,,8] <- 0
                              E_stovers_dietary_req[,,9] <- 0
                                                            # create ampty array for the 9 livestock species for Protein st
                                                            # transform coefficient from demand in Kg to Tons
  P_stovers_dietary_req <- array(dim=c(1,9331200,9))
                              P_stovers_dietary_req[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animals dietary r
                              P_stovers_dietary_req[,,8] <- 0
                              P_stovers_dietary_req[,,9] <- 0
```

6.2.4 Load organic livestock N excreta coefficients

```
# --- Load N excreta available for cropland coefficients maps ---
                # FOR RUMINANTS, PASTURE LOSSES ARE SEPARATED FOR CORRECTION!!! (see following code)
                # Losses due to storage condition, other uses of manure, etc, are here already included in the available manure.
                #--> i.e. available manure for application to soil. Losses after manure application to soils have to be incorpora
                # These losses are accounted in the optimisation matrix code.
                                # create ampty array for the 9 livestock species for N excreta coefficients

  Nyear_livestock_production_available <- array(dim=c(1,9331200,9))
        Nyear_livestock_production_available[,,1] <- as.vector(raster("Organic_optimisation_sub_model/spatial_livestock_excreta/
                              Nyear_livestock_production_available[,,2] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,3] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,4] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,5] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,6] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,7] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,8] <- as.vector(raster("Organic_optimisation_sub_model/sp
                              Nyear_livestock_production_available[,,9] <- as.vector(raster("Organic_optimisation_sub_model/sp


                              ## --- FOR RUMINANTS, Preparation files for correction of manure "losses" on pastures due to graz

        # Load IPCC coefficients accounting for the fraction of manure 'lost' on permanent pastures for:
                              # 1. bovines
        Nyear_livestock_PASTURE_LOSSES_1_2 <- raster("Organic_optimisation_sub_model/spatial_livestock_excreta/cow_Nitrogen_IPCC
        # 2. goats
                              Nyear_livestock_PASTURE_LOSSES_3_4 <- raster("Organic_optimisation_sub_model/spatial_livestock_e
                              # 3. sheep
                        Nyear_livestock_PASTURE_LOSSES_5_6 <- raster("Organic_optimisation_sub_model/spatial_livestock_excreta/s
```

```
                        # stack files
                                stack <- stack(Nyear_livestock_PASTURE_LOSSES_1_2, Nyear_livestock_PASTURE_LOSSES_3_4, Nyear_liv

                                # create raster map with values 1 where land is prenent --> shape of continents in order to add l
                continents_shape <- raster("Coefficients/map_coeff_residues_recycle_organic.tif")>=0
                continents_shape <- reclassify(continents_shape, cbind(1,0))

                # Load coefficients to account for the % increase of temporary fodder in the organic scenario compared to the bas
                # This coefficient was calculated as the % increase in temporary fodder occurring after the estimate land use cha
                # in organic farming compared to the conventional baseline (see documentation)
                Coefficient_decreased_share_manure_losses_pasture <- raster("Organic_optimisation_sub_model/correction_coefficie
                Coefficient_decreased_share_manure_losses_pasture <- merge(Coefficient_decreased_share_manure_losses_pasture, co

                    # (1-manure_lost_on pasture) = manure_available_after_pasture_losses * Increase_in_availablility
                                stack_corrected <- (1-stack)*(1+Coefficient_decreased_share_manure_losses_pasture)
                                # double bind corrections : values above 1 are set to 1.
                                rclmat <- matrix(c(1, Inf, 1), ncol=3, byrow=T)
                                stack_corrected <- reclassify(stack_corrected, rclmat)

# create vectorors with the coeected IPCC coefficient giving the share of manure available on Pasture.
# To be directly used in the model optimisation code to estimate the final fraction of manure available for cropland
  Nyear_livestock_PASTURE_LOSSES <- array(dim=c(1,9331200,9))
                                Nyear_livestock_PASTURE_LOSSES[,,1] <- as.vector(stack_corrected[[1]])
                                Nyear_livestock_PASTURE_LOSSES[,,2] <- as.vector(stack_corrected[[1]])
                                Nyear_livestock_PASTURE_LOSSES[,,3] <- as.vector(stack_corrected[[2]])
                                Nyear_livestock_PASTURE_LOSSES[,,4] <- as.vector(stack_corrected[[2]])
                                Nyear_livestock_PASTURE_LOSSES[,,5] <- as.vector(stack_corrected[[3]])
                                Nyear_livestock_PASTURE_LOSSES[,,6] <- as.vector(stack_corrected[[3]])
                                Nyear_livestock_PASTURE_LOSSES[,,7] <- 1
                                Nyear_livestock_PASTURE_LOSSES[,,8] <- 1
                                Nyear_livestock_PASTURE_LOSSES[,,9] <- 1
```

6.2.5 Load various crop related coefficients

```
## ---  Crop yields related coefficients - Yield curves slopes, RPR coefficients,
                ##   N-fixed for legumes and cyanobacterias ---

        # load crop yield curve slope
  Crops_yield_curve_slope <- 1/crop_info$N
        # load crop stovers yield curve slope
  Stovers_yield_curve_slope <- 1/crop_info$N_crop_residues
        Stovers_yield_curve_slope[Stovers_yield_curve_slope==(Inf)] <- 0
        # load RPR coefficients to estimate crop residues from harvested biomass
  Stovers_RPR <- crop_info$RPR_corrected_optimisation_model

  # load N fixation by legumes coefficients
  Fixed_N <- crop_info$NiC * crop_info$X1.NHI * crop_info$Proot_NOT_TO_BE_USED * crop_info$Ndfa

  # load N fixation by cyanobateria coeffcient and create dataset
  # cereals, roots, oilcrops: 0.012 in tons/ha, Liu et al. 2010
                        cereals_tuber_oils <- 0.012
                        roots <- which(crop_info$CropGroup=="roots")
                        cereal <- which(crop_info$CropGroup=="cereal")
```

```
                    sec.cereal <- which(crop_info$CropGroup=="sec.cereal")
                    oilcrops <- which(crop_info$CropGroup=="oilcrops")
    N_cyanobacteria <- array(dim=c(1,61))
                    N_cyanobacteria[roots] <- N_cyanobacteria[cereal] <- N_cyanobacteria[sec.cereal] <- N_cyanobacteria[oilc
                      # sugarcane: 0.1 in tons/ha, Liu et al. 2010
             N_cyanobacteria[50] <-  0.1
      # rice: 0.02 in tons/ha, Liu et al. 2010
                    N_cyanobacteria[43] <-  0.02
                    N_cyanobacteria <- ifelse(!is.na(N_cyanobacteria), N_cyanobacteria, 0)


  # load crops energy density, transformed into MJ/tons
      Energy_crops_obj_fod <- Energy_crops_obj <- Energy_crops <- crop_info$metabolisable_Energy_MJ.kgDM*1000
          # load crop residues energy density, transformed into MJ/tons
Energy_stovers <- crop_info$metabolisable_Energy_MJ.kgDM*1000
          # load crops protein density
Protein_crops <- crop_info$protein_content
          # load crop residues protein density
Protein_stovers <- crop_info$N_crop_residues * 6.25

          # load coefficient reporting the share of crop residues exported from the field --> 1- residues recycled to soil
crop_res_avail_org <- array(dim=c(1, 9331200))
crop_res_avail_org[,] <- as.vector((1-raster("Coefficients/map_coeff_residues_recycle_organic.tif")/1.45))
                                                                                                    # ******** ATTEN

# load maximum crop share allowed to be used as feed
share_available_feed <- crop_info$Fraction_used_for_feed
```

6.2.6 Load permanent pasture dataset and coefficients

```
  ## --- Load permanent pasture dataset and coefficients ---

# load available biomass form permanent pasture in DM, see documenation for the relative calculations
permanent_pastures_available_biomass_t_DM <- array(dim=c(1, 9331200))
permanent_pastures <- raster("Organic_optimisation_sub_model/Permanent pastures/Average_year_2015_tonsDMPERMANENTPASTURES_year
  origin(permanent_pastures) <- 0
             rclmat <- matrix(c(-Inf, 0, 0), ncol=3, byrow=T)
# correction of errors in Pasture maps delete areas where pasture is negative (negative Primary Productivity values)
permanent_pastures <- reclassify(permanent_pastures, rclmat)
      continents_shape <- raster("Coefficients/map_coeff_residues_recycle_organic.tif")>=0
             continents_shape <- reclassify(continents_shape, cbind(1,0))
             permanent_pastures <- merge(permanent_pastures, continents_shape)

permanent_pastures_available_biomass_t_DM[,] <- as.vector(permanent_pastures)

# load Energy and Protein density coefficiens for permanent pastures
  permanent_pastures_energy_protein_densities <- array(dim=c(1, 9331200, 2))
  permanent_pastures_energy_protein_densities[,,1] <- as.vector(raster("Organic_optimisation_sub_model/Permanent pastures/glob
  permanent_pastures_energy_protein_densities[,,2] <- as.vector(raster("Organic_optimisation_sub_model/Permanent pastures/glob
```

6.2.7 Load livestock sector production coefficients

```
    ## --- Load livestock coefficients to calculate Energy produced in livestock products (meat, milk, eggs) ---

 # read dataset reporting the specified coefficients
 livestock_products_coefficients <- read.csv("Animal_products_energy_and_proteins_coefficients.csv")
 # load food energy coefficients
 E_livestock_products <- livestock_products_coefficients$Energy.MJ...KG

    ## --- Load livestock coefficients to calculate the fresh mass/volume of livestock products (meat, milk, eggs) ---
   # order of species 1: cow milk, 2: cow meat, 3: goats milk, 4: goats meat, 5: sheep milk, 6: sheep meat, 7: pig meat
   matrix_livestock_production <- array(dim=c(1,9331200,7))
                       matrix_livestock_production[,,1] <- as.vector(raster("Organic_optimisation_sub_model/animal production c
                       matrix_livestock_production[,,2] <- as.vector(raster("Organic_optimisation_sub_model/animal production c
                       matrix_livestock_production[,,3] <- as.vector(raster("Organic_optimisation_sub_model/animal production c
                       matrix_livestock_production[,,4] <- as.vector(raster("Organic_optimisation_sub_model/animal production c
                       matrix_livestock_production[,,5] <- as.vector(raster("Organic_optimisation_sub_model/animal production c
                       matrix_livestock_production[,,6] <- as.vector(raster("Organic_optimisation_sub_model/animal production c
                       matrix_livestock_production[,,7] <- as.vector(raster("Organic_optimisation_sub_model/animal production c

                          ## --- Load  coefficients to calculate effective livestock numbers in one year (for livestock species

                       # Load meat goats number of batches per year, to be used to calculate correct animal numbers for producti
   matrix_goats_meat_heads_year <- array(dim=c(1,9331200))
                       matrix_goats_meat_heads_year[,] <- as.vector(raster("Organic_optimisation_sub_model/animal production co
   # Load meat sheep number of batches per year, to be used to calculate correct animal numbers for production
   matrix_sheep_meat_production <- array(dim=c(1,9331200))
                       matrix_sheep_meat_production[,] <- as.vector(raster("Organic_optimisation_sub_model/animal production co
                           # Load meat pigs number of batches per year, to be used to calculate correct animal numbers for produ
   matrix_pigs_production <- array(dim=c(1,9331200))
                       matrix_pigs_production[,] <- as.vector(raster("Organic_optimisation_sub_model/animal production coeffici

                                                 ## --- Load  cattle Coefficent to trasform anumal heads into tropical Livesto
# load cattle TLU coefficent map --> original source, FAOSTAT
        cattle_TLU <- array(dim=c(1,9331200))
                       cattle_TLU[,] <- as.vector(raster("Organic_optimisation_sub_model/animal production coefficients/cows_TL
```

6.2.8 Load IPCC N leaching coefficient and N atmospheric deposition maps

```
                        ## --- Load map of N leaching  for correction of N in organic manure and N in stovers recycled  to soil
                       # Load leaching areas. The areas where leaching occurs were estimated according to the IPCC procedure.
         leaching_areas_mask <- raster("Climatic and irrigation data/mask_area_leaching_final_irrigation_above_005.tif")
                       # IPCC leaching coefficient of 0.3: correction to estimate the 30% losses due to leaching
         mask_correction_leaching <- leaching_areas_mask * 0.7
         mask_correction_leaching <- reclassify(mask_correction_leaching, cbind(0,1))
         mask_correction_leaching_vec <- array(dim=c(1, 9331200))
         mask_correction_leaching_vec[,] <- as.vector(mask_correction_leaching)

 ## --- Load map of N atmospheric deposition ---
 N_deposition <- array(dim=c(1,9331200,1))
    # tons of N atmospheric depostions 1993 per hectare and cut per cropland area, corrected by leaching
              N_deposition[,,1] <- as.vector(raster("N atmospheric deposition/N_deposition_per_hectare_of_cropland2000.tif") *
```

6.2.9 Inputs from conventional farming systems

```
                        ## --- Load conventional farming inputs from conventional submodel results ---
  # load conventional manure surplus - Unredistributed for scenarios 1-2
   N_conventional_manure <- array(dim=c(1,9331200))                          # N from conventional manure surplus computed in t
                N_conventional_manure[,] <- as.vector(raster("Script_generated_files/manure_N_surplus_total_100_manure_cropland.
  # load conventional manure surplus - redistibuted for scenarios 3-4 --> To be activated!!
                # N_conventional_manure[,] <- as.vector(raster("Script_generated_files/N_redistributed_manure_surplus_100_harvest


    # load wastewater surplus - the correct file as tro be loaded according to the selected conventional global agricultural shar
   N_conventional_waste_water <- array(dim=c(1,9331200))
                waste_water <- extend(raster("Script_generated_files/wastewater_N_surplus_total_100_manure_cropland.tif"), exter
                N_conventional_waste_water[,] <- as.vector(waste_water)
```

6.2.10 Generate arrays for storing optimisation results

```
                 ## ---- Generate arrays for storing the optimisation results  ---
# 1. final organic yiels
matrix_final_yield <- array(dim=c(1,9331200,61))
                # 2. quantity of N available fertilisers assigned to each crops
matrix_final_N_assigned_to_crop <- array(dim=c(1,9331200,61))
# 3. sub-FOOD yield
matrix_sub_food_yield <- array(dim=c(1,9331200,61))
# 4. sub-FEED yield
matrix_sub_feed_yield <- array(dim=c(1,9331200,61))
# 5. numbers of heads, i.e. sum of all producing heads in one year of each livestock type
number_of_livestock_heads <- array(dim=c(1,9331200,9))
# 6. numbers of standing animal at the counting date (i.e. end of the year). For dairy animal species, beef and hens the number o
number_of_livestock_standing_animals <- array(dim=c(1,9331200,4))
 # 7. total N available in organic manure
N_total_available <- array(dim=c(1,9331200,1))
# 8. legumes  BNF
BNF <- array(dim=c(1,9331200,length(which(Fixed_N!=0))))

    # 9.  rray containing in-model statistics
statistics <- array(dim=c(1,9331200,11))
 # 10. counter for model failures in finding a solution  [0-1]; 1 is returned when the solution is not found
counter <- array(dim=c(1,9331200))
```

6.3 Linear optimisation procedure

6.3.1 Initialise ”for” loop

```
# set first array to 1 --> all sngle arrays are vectors
i=1

# initialise sensitivity_no: this number refers to the number of the scenario that the user want to test
# and is automatically related to the name of the folders where all results will be saved.
# I.e. if sensitivity_no is set to 1, the results will be saved in the respecitive folders
```

```
# prepared by the use to contain the results of Scenarion 1.
# initialise script_no: to reduce the time of computation, global maps are divided here in 24 subparts,
# that can be run in parallel. if script_no is set to 1, the optimisatio run the first part of the map out of 24
# and it saves the results in the respective folder.
#An example of the reusults folder struction is given after this code chunk.
sensitivity_no <- NA
script_no <- NA

# set the number fo the gridcells on which we want to run the optimisation (from 1 to 9331200)
# --> the total number of cells is divided here in 24 parts, but the exact pattern of subtivision is up to the user.
# An example of subdivision is proposed after this code chunk.

for (j in XXX:XXX) {

# Run the optimisation procedure only if the selected gridcell is part of continental land,
# e.g. if the cell correspond to an ocean it contains no data (NA);
# hence the optimisation procedure is not run and a correpective NA data is saved in all oputput arrays
# (see last chunk of the optimisation code procedure. this procedure helps reducting the computational time.

if (is.na(matrix_area_vec[i,j,1])==FALSE) {

}} # attention this is a fake ending of the loops, just for generating the documentation file. these bracketrs have to be deleted
```

6.3.2 Folder structure to save model output

The optimisation model saves the results of the 24 sub-parts into respective sub-folders. The results of each sub-part are saved in a mabeled folder (from 1 to 24), by setting a parameter called $script_{no}$ with the correc sub-folder value. Within each sub-folder, files are saved in a second level sub-folder corresponding the the scenario number that the user is testing, i.e. according to the value set in the $sensitivity_no$ parameter, as follows:

- Script 1
  - Scenario S1
  - Scenario S2
  - ...
  - Scenario SX
- Script 2
  - Scenario S1
  - Scenario S2
  - ...
  - Scenario SX
- ...
- Script 24

- Scenario S1
- Scenario S2
- ...
- Scenario SX

6.3.3 Model coefficient matrix sub-parts S1 to S56

```
# matrix square numbers correspond to the numbers reported in Table 5

     # --- 1st part of the matrix giving the 61 yield curve constraints for the crop harvested yield ----

sparse_matrix_1 <- simple_triplet_diag_matrix(rep(1,61), nrow=61)
sparse_matrix_2 <- simple_triplet_diag_matrix(-Crops_yield_curve_slope[], nrow=61)
sparse_matrix_3 <- simple_triplet_zero_matrix(61)
sparse_matrix_4 <- simple_triplet_zero_matrix(61)
sparse_matrix_5 <- simple_triplet_zero_matrix(61)
sparse_matrix_6 <- simple_triplet_zero_matrix(61)
sparse_matrix_7 <- simple_triplet_zero_matrix(61)
sparse_matrix_8 <- simple_triplet_zero_matrix(61)


                      # --- 2nd part of the matrix giving the 61 yield curve plateaux constraints for the harvested yield ---
sparse_matrix_9 <- simple_triplet_diag_matrix(rep(1, 61), nrow=61)
sparse_matrix_10 <- simple_triplet_zero_matrix(61)
sparse_matrix_11 <- simple_triplet_zero_matrix(61)
sparse_matrix_12 <- simple_triplet_zero_matrix(61)
sparse_matrix_13 <- simple_triplet_zero_matrix(61)
sparse_matrix_14 <- simple_triplet_zero_matrix(61)
sparse_matrix_15 <- simple_triplet_zero_matrix(61)
sparse_matrix_16 <- simple_triplet_zero_matrix(61)


                      # --- 3rd part of the matrix giving the 61 N required by exported fraction of crop residues + the relati
sparse_matrix_17 <- simple_triplet_diag_matrix(Stovers_RPR[], nrow=61)
sparse_matrix_18 <- simple_triplet_zero_matrix(61)
sparse_matrix_19  <- simple_triplet_diag_matrix((ifelse(-((Stovers_yield_curve_slope*1)/(crop_res_avail_org[i,j]+(2-(0.99+mask_c
sparse_matrix_20 <- simple_triplet_zero_matrix(61)
sparse_matrix_21 <- simple_triplet_zero_matrix(61)
sparse_matrix_22 <- simple_triplet_zero_matrix(61)
sparse_matrix_23 <- simple_triplet_zero_matrix(61)
sparse_matrix_24 <- simple_triplet_zero_matrix(61)


                      # --- 4th part of the matrix giving the 61 constraint for fixed nitrogen by legume crop species and by ro

sparse_matrix_25 <- simple_triplet_diag_matrix(c(Fixed_N[1:3], ifelse(matrix_yield_max_vec[i,j,4]>0, Fixed_N[4]/matrix_yield_max
                                                                                                    Fixed_N[15:21], ifelse(matrix_y
                                                                                                    Fixed_N[30], ifelse(matrix_yiel
                                                                                                    Fixed_N[41], ifelse(matrix_yiel
                                                                                                    Fixed_N[48:49], ifelse(matrix_y
                                                                                                    ifelse(matrix_yield_max_vec[i,j
sparse_matrix_26 <- simple_triplet_zero_matrix(61)
```

```
sparse_matrix_27 <- simple_triplet_zero_matrix(61)
sparse_matrix_28 <- simple_triplet_zero_matrix(61)
sparse_matrix_29 <- simple_triplet_diag_matrix(rep(-1,61), nrow=61)
sparse_matrix_30 <- simple_triplet_zero_matrix(61)
sparse_matrix_31 <- simple_triplet_zero_matrix(61)
sparse_matrix_32 <- simple_triplet_zero_matrix(61)


                      # --- 5th part of the matrix giving the 61 constraint for nitrogen budgets for each crop including Ngrain

sparse_matrix_33 <- simple_triplet_zero_matrix(61)
#  N_grain
sparse_matrix_34 <- simple_triplet_diag_matrix(matrix_area_vec[i,j,], nrow=61)
#  N_residues
sparse_matrix_35 <- simple_triplet_diag_matrix(matrix_area_vec[i,j,], nrow=61)
#  N_pool
sparse_matrix_36 <- simple_triplet_diag_matrix(matrix_area_vec[i,j,], nrow=61)
  #  N_fixed
sparse_matrix_37 <- simple_triplet_diag_matrix(-matrix_area_vec[i,j,], nrow=61)
 #  N_fertilisers
sparse_matrix_38 <- simple_triplet_diag_matrix(-matrix_area_vec[i,j,], nrow=61)
sparse_matrix_39 <- simple_triplet_zero_matrix(61)
sparse_matrix_40 <- simple_triplet_zero_matrix(61)


                      # --- 6th part of the matrix giving the constraint for nitrogen availability fertilisers (i.e. manures an

sparse_matrix_41 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_42 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_43 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_44 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_45 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_46 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(matrix_area_vec[i,j,]))
sparse_matrix_47 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_48 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))


                      # --- 7th part of the matrix defining the 61 sub_food and sub_feed yields ---

#  Total_simulated_yields
sparse_matrix_49 <- simple_triplet_diag_matrix(rep(-1,61), nrow=61)
sparse_matrix_50 <- simple_triplet_zero_matrix(61)
sparse_matrix_51 <- simple_triplet_zero_matrix(61)
sparse_matrix_52 <- simple_triplet_zero_matrix(61)
sparse_matrix_53 <- simple_triplet_zero_matrix(61)
sparse_matrix_54 <- simple_triplet_zero_matrix(61)
#  Food Sub_yields
sparse_matrix_55 <- simple_triplet_diag_matrix(rep(1,61), nrow=61)
 #  Feed Sub_yields
sparse_matrix_56 <- simple_triplet_diag_matrix(rep(1,61), nrow=61)
```

6.3.4 Model coefficient matrix sub-parts S57 to S91

```
        # --- 8th part of the matrix defining th emaximum share of the 61 Sub_yield available for feed ---

#  Share_available_yeidl_for_feed
sparse_matrix_57 <- simple_triplet_diag_matrix(rep(share_available_feed,61), nrow=61)
sparse_matrix_58 <- simple_triplet_zero_matrix(61)
sparse_matrix_59 <- simple_triplet_zero_matrix(61)
sparse_matrix_60 <- simple_triplet_zero_matrix(61)
sparse_matrix_61 <- simple_triplet_zero_matrix(61)
sparse_matrix_62 <- simple_triplet_zero_matrix(61)
sparse_matrix_63 <- simple_triplet_zero_matrix(61)
 #  Feed Sub_yields maximum constraint
sparse_matrix_64 <- simple_triplet_diag_matrix(rep(-1,61), nrow=61)



                    # --- 9th part of the matrix giving the total available ENERGY from GRAIN for livestock feed ---
    # Sub_feed_yield (grain crops) * Area * Energy_density
sparse_matrix_65 <- simple_triplet_matrix(rep(1,427), seq(1:427), rep(0,427))
sparse_matrix_66 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(0,matrix_area_vec[i,j,2]*Energy_crops[2],matrix_area_vec[i,j,



                    # --- 10th part of the matrix giving the total available ENERGY from FODDER for livestock feed ---
# Sub_feed_yield (fodder crops) * Area * Energy_density
sparse_matrix_67 <- simple_triplet_matrix(rep(1,427), seq(1:427), rep(0,427))
sparse_matrix_68 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(matrix_area_vec[i,j,1]*Energy_crops[1],0,0,0,0,0,0,0,0,0,0,ma



                    # --- 11th part of the matrix giving the total available ENERGY from STOVERS for livestock feed ---
sparse_matrix_69 <- simple_triplet_matrix(rep(1,427), seq(1:427), rep(0,427))
sparse_matrix_70 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(0,0,0,matrix_area_vec[i,j,4]*Energy_stovers[4]*Stovers_RPR[4]



                    # --- 12th part of the matrix giving the total available PROTEIN from GRAIN for livestock feed ---
sparse_matrix_71 <- simple_triplet_matrix(rep(1,427), seq(1:427), rep(0,427))
sparse_matrix_72 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(0,matrix_area_vec[i,j,2]*Protein_crops[2],matrix_area_vec[i,j



                    # --- 13th part of the matrix giving the total available PROTEINS from FODDER for livestock feed ---
sparse_matrix_73 <- simple_triplet_matrix(rep(1,427), seq(1:427), rep(0,427))
```

```
sparse_matrix_74 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(matrix_area_vec[i,j,1]*Protein_crops[1],0,0,0,0,0,0,0,0,0,0,m

                        # --- 14th part of the matrix giving the total available PROTEINS from STOVERS for livestock feed ---
sparse_matrix_75 <- simple_triplet_matrix(rep(1,427), seq(1:427), rep(0,427))
sparse_matrix_76 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(0,0,0,matrix_area_vec[i,j,4]*Protein_stovers[4]*Stovers_RPR[4

                        # --- 15th part of the matrix giving the row containing "zeros" before the N manure production by animals

sparse_matrix_77 <- simple_triplet_matrix(rep(1,488), seq(1:488), rep(0,488))

                        # --- 16th part of the matrix giving the rows containing "zeros" before energy in animals products (9 ani
sparse_matrix_78 <- simple_triplet_matrix(c(rep(1,488), rep(2,488), rep(3,488), rep(4,488), rep(5,488), rep(6,488), rep(7,488),

                        # --- 17th part of the matrix giving the rows containing "zeros" before calculation of animals TLU (9 ani
sparse_matrix_79 <- simple_triplet_matrix(c(rep(1,488), rep(2,488), rep(3,488), rep(4,488), rep(5,488), rep(6,488), rep(7,488),

                        # --- 18th part of the matrix giving the 5 "zeros" rows before the contraints of the minimum animal numbe
sparse_matrix_80 <- simple_triplet_matrix(c(rep(1,488), rep(2,488), rep(3,488), rep(4,488), rep(5,488)), rep(seq(1:488), 5), rep

                        # --- 19th part of the matrix giving the 9 "zeros" rows before the contraints of the maximum animal numbe
sparse_matrix_81 <- simple_triplet_matrix(c(rep(1,488), rep(2,488), rep(3,488), rep(4,488), rep(5,488), rep(6,488), rep(7,488),

                        # --- 20th part of the matrix giving the contraints of the minimum yield for each grouped crop category (
         # primary cereals
sparse_matrix_82 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,25), 1, rep(0, 16), 1, rep(0,16), 1, 0, rep(0, 427)))
                       # secondary cereals
sparse_matrix_83 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,3), 1, rep(0, 2), 1, rep(0,21), 1, 0, 1, rep(0,13), 1,
     # pulses
sparse_matrix_84 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,4), 1, 1, rep(0, 4), 1, rep(0,5), 1, rep(0,6), 1, rep(
   # oilcrops
sparse_matrix_85 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,13), 1, rep(0, 7), 1, rep(0,2), 1, rep(0,6), 1, 0, 1,
 # roots
sparse_matrix_86 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,9), 1, rep(0, 29), 1, rep(0,11), 1, rep(0,8), 1, rep(0
                                                          # industrial
sparse_matrix_87 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,48), 1, 1, rep(0, 11),  rep(0, 427)))
# fodders
sparse_matrix_88 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(1, rep(0,10), 1, rep(0, 5), 1, rep(0,2), 1, 0, 1, rep(0,3),
                                             # vegetables
sparse_matrix_89 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,7), 1, rep(0, 26), 1, rep(0, 19), 1, 0, 1, 0, 1, rep(0
         # fruits
```

```
sparse_matrix_90 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(0, 1, 1, rep(0, 5), 1, rep(0,3), 1, rep(0,5), 1, 1, rep(0,7)
                                          # NON-FOOD crops
sparse_matrix_91 <- simple_triplet_matrix(rep(1,488), seq(1:488), c(rep(0,14), 1, 1, rep(0, 27), 1, rep(0,8), 1, 1, rep(0,7),  r
```

6.3.5 Model coefficient matrix sub-parts S92 to S105

The following matrix is structured as follow:

- Matrix dimension definition:
  1. Available N for manure for fertilisation
  2. Livestock dietary constraints, and manure production
  3. Livestock number calculation 1 (9 equations)
  4. Livestock number calculation 2 (9 equations)
  5. Livestock constraints categories (lower bound) (5 equations head/LU)
  6. Livestock constraints categories (upper bound) (9 equations, head/LU)
- Matrix cells content:
  1. a. Available N for manure for fertilisation
  2. Livestock dietary constraints and manure production:
     a. Energy Grain requirements
     b. Energy Grass requirements
     c. Energy Stover requirements
     d. Protein Grain requirements
     e. Protein Grass requirements
     f. Protein Stover requirements
     g. N in manure produced by animals corrected by losses (4% N used by other crops, 1,66% direct N_2O and N_2 losses, 20% indirect $\frac{NH_3}{NO}$ losses, 30% leaching losses
  3. Equalities for livestock numbers:
     a. dairy cows: 1 year manure correspond to 1 animal
     b. beef: 1 year manure correspond to 1 animal
     c. milk goats: 1 year manure correspond to 1 animal
     d. meat goats:
     e. milk sheep: 1 year manure correspond to 1 animal
     f. meat sheep
     g. pigs
     h. broilers: 1 year manure correspond to 7.82 animals
     i. layers: 1 year manure correspond to 1 animal
  4. Equalities for livestock numbers for calculate livestock in LU:
     a. dairy cows
     b. beef
     c. milk goats
     d. meat goats
     e. milk sheep
     f. meat sheep
     g. pigs
     h. broilers
     i. layers

5. Livestock lower bound:
   a.1 Ruminants cannot be lower that [%] * total animal head
   b.1 Monogastric cannot be lower that [%] * total animal head
   c.1 Milk animals cannot be lower that [%] * total animal head
   d.1 Meat animals cannot be lower that [%] * total animal head
   e.1 Eggs animals cannot be lower that [%] * total animal head
   a.2 *to be deactivated* - ruminants cannot be lower that [%] * total animal LU
   b.2 *to be deactivated* - monogastric cannot be lower that [%] * total animal LU
   c.2 *to be deactivated* - milk animals cannot be lower that [%] * total animal LU
   d.2 *to be deactivated* - meat cannot be lower that [%] * total animal LU
   e.2 *to be deactivated* - eggs animals cannot be lower that [%] * total animal LU

6. Livestock upper bound:
   a.1 dairy cows maximum [%]of total animal head
   b.1 beef maximum [%] of total animal head
   c.1 milk goats maximum [%] of total animal head
   d.1 meat goats maximum [%] of total animal head
   e.1 milk sheep maximum [%] of total animal head
   f.1 meat sheep maximum [%] of total animal head
   g.1 pigs maximum [%] of otal animal number head
   h.1 broilers maximum [%] of total animal head
   i.1 layers maximum [%] of total animal head
   a.2 *to be deactivated* - dairy cows maximum [%] of total animal LU
   b.2 *to be deactivated* - beef maximum [%] of total animal LU
   c.2 *to be deactivated* - milk goats maximum [%] of total animal LU
   d.2 *to be deactivated* - meat goats maximum [%] of total animal LU
   e.2 *to be deactivated* - milk sheep maximum [%] of total animal LU
   f.2 *to be deactivated* - meat sheep maximum [%] of total animal LU
   g.2 *to be deactivated* - pigs maximum [%] of total animal number LU
   h.2 *to be deactivated* - broilers maximum [%] of total animal LU
   i.2 *to be deactivated* - layers maximum [%] of total animal LU

```
 # -- 21th Right side of the matrix ---
sparse_matrix_92 <- simple_triplet_matrix(c(rep(306,28),
rep(429,28), rep(430,28), rep(431,28), rep(432,28), rep(433,28), rep(434,28), rep(435,28),
rep(436,28), rep(437,28), rep(438,28), rep(439,28), rep(440,28), rep(441,28), rep(442,28), rep(443,28), rep(444,28),
rep(445,28), rep(446,28), rep(447,28), rep(448,28), rep(449,28), rep(450,28), rep(451,28), rep(452,28), rep(453,28),
rep(454,28), rep(455,28), rep(456,28), rep(457,28), rep(458,28),
rep(459,28), rep(460,28), rep(461,28), rep(462,28), rep(463,28), rep(464,28), rep(465,28), rep(466,28), rep(467,28)), rep(seq(1:
c(rep(0,9),-1, rep(0,9), rep(0,9),
-E_grain_dietary_req[i,j,],0, rep(0,9), rep(0,9),
-E_grass_dietary_req[i,j,], 0, rep(0,9), rep(0,9),
-E_stovers_dietary_req[i,j,], 0, rep(0,9), rep(0,9),
-P_grain_dietary_req[i,j,], 0, rep(0,9), rep(0,9),
-P_grass_dietary_req[i,j,], 0, rep(0,9), rep(0,9),
-P_stovers_dietary_req[i,j,], 0, rep(0,9), rep(0,9),
                                                                    (Nyear_livestock_production_available[i,j,

1, rep(0,8), 0, -1, rep(0,8), rep(0,9),
0, 1, rep(0,7), 0, 0, -1, rep(0,7), rep(0,9),
0, 0, 1, rep(0,6), 0, 0, 0, -1, rep(0,6), rep(0,9),
0, 0, 0, matrix_goats_meat_heads_year[i,j], rep(0,5), 0, 0, 0, 0, -1, rep(0,5), rep(0,9),
0, 0, 0, 0, 1, rep(0,4),0, 0, 0, 0, 0, -1, rep(0,4), rep(0,9),
```

```
0, 0, 0, 0, 0, matrix_sheep_meat_production[i,j], rep(0,3), 0, 0, 0, 0, 0, 0, -1, rep(0,3), rep(0,9),
0, 0, 0, 0, 0, 0, matrix_pigs_production[i,j], rep(0,2), 0, 0, 0, 0, 0, 0, 0, -1, rep(0,2), rep(0,9),
0, 0, 0, 0, 0, 0, 0, 7.82, rep(0,1), 0, 0, 0, 0, 0, 0, 0, 0, -1, rep(0,1), rep(0,9),
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, rep(0,9),
1*cattle_TLU[i,j], rep(0,8), 0, rep(0,9), -1, rep(0,8),
0, 1*cattle_TLU[i,j], rep(0,7), 0, rep(0,9), 0, -1, rep(0,7),
0, 0, 1*0.1, rep(0,6), 0, rep(0,9), 0, 0, -1, rep(0,6),
0, 0, 0, matrix_goats_meat_heads_year[i,j]*0.1, rep(0,5), 0, rep(0,9), 0, 0, 0, -1, rep(0,5),
0, 0, 0, 0, 1*0.1, rep(0,4),0, rep(0,9), 0, 0, 0, 0, -1, rep(0,4),
0, 0, 0, 0, 0, matrix_sheep_meat_production[i,j]*0.1, rep(0,3), 0, rep(0,9), 0, 0, 0, 0, 0, -1, rep(0,3),
0, 0, 0, 0, 0, 0, matrix_pigs_production[i,j]*0.225, rep(0,2), 0, rep(0,9), 0, 0, 0, 0, 0, 0, -1, rep(0,2),
0, 0, 0, 0, 0, 0, 0, 7.82*0.01, rep(0,1), 0, rep(0,9), 0, 0, 0, 0, 0, 0, 0, -1, rep(0,1),
0, 0, 0, 0, 0, 0, 0, 0, 1*0.01, 0, rep(0,9), 0, 0, 0, 0, 0, 0, 0, 0, -1,
1, 1, 1, matrix_goats_meat_heads_year[i,j], 1, matrix_sheep_meat_production[i,j], 0, 0, 0,   0, rep(-0,9), rep(0,9),
0, 0, 0, 0, 0, 0, matrix_pigs_production[i,j], 7.82, 1, 0, rep(-0,9), rep(0, 0,9),
1, 0, 1, 0, 1, 0, 0, 0, 0, 0, rep(-0,9), rep(0,9),
0, 1, 0, matrix_goats_meat_heads_year[i,j], 0, matrix_sheep_meat_production[i,j], matrix_pigs_production[i,j], 7.82, 0,   0, rep
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, rep(-0,9), rep(0,9),
1*cattle_TLU[i,j], 1*cattle_TLU[i,j], 1*0, matrix_goats_meat_heads_year[i,j]*0, 1*0, matrix_sheep_meat_production[i,j]*0, 0, 0,
 0, 0, 0, 0, 0, 0, matrix_pigs_production[i,j]*0.225, 7.82*0.01, 1*0.01,    0, rep(0,9), rep(-0.0, 0,9),
1*cattle_TLU[i,j], 0, 1*0.1, 0, 1*0.1, 0, 0, 0, 0, 0, rep(0,9), rep(-0,9),
0, 1*cattle_TLU[i,j], 0, matrix_goats_meat_heads_year[i,j]*0.1, 0, matrix_sheep_meat_production[i,j]*0.1, matrix_pigs_production
0, 0, 0, 0, 0, 0, 0, 0, 1*0.01,  0, rep(0,9), rep(-0.0,9),
1, rep(0,8), 0,   rep(-1,9), rep(0,9),
0, 1, rep(0,7), 0,   rep(-1,9), rep(0,9),
0, 0, 1, rep(0,6), 0,  rep(-1,9), rep(0,9),
0, 0, 0, matrix_goats_meat_heads_year[i,j], rep(0,5), 0,  rep(-1,9), rep(0,9),
0, 0, 0, 0, 1, rep(0,4),0,  rep(-1,9), rep(0,9),
0, 0, 0, 0, 0, matrix_sheep_meat_production[i,j], rep(0,3), 0,  rep(-1,9),  rep(0,9),
0, 0, 0, 0, 0, 0, matrix_pigs_production[i,j], rep(0,2), 0,  rep(-1,9), rep(0,9),
0, 0, 0, 0, 0, 0, 0, 7.82, rep(0,1), 0,  rep(-1,9), rep(0,9),
0, 0, 0, 0, 0, 0, 0, 0, 1, 0,  rep(-1,9), rep(0,9)),
1*cattle_TLU[i,j], rep(0,8), 0,  rep(0,9), rep(-1,9),
0, 1*cattle_TLU[i,j], rep(0,7), 0, rep(0,9),  rep(-1,9),
0, 0, 1*0.1, matrix_goats_meat_heads_year[i,j]*0.1,rep(0,5), 0, rep(0,9), rep(-1,9),
0, 0, 0, matrix_goats_meat_heads_year[i,j]*0.1, rep(0,5), 0, rep(0,9), rep(-1,9),
0, 0, 0, 0, 1*0.1, matrix_sheep_meat_production[i,j]*0.1,rep(0,3),0, rep(0,9), rep(-1,9),
0, 0, 0, 0, 0, matrix_sheep_meat_production[i,j]*0.1, rep(0,3), 0, rep(0,9), rep(-1,9),
0, 0, 0, 0, 0, 0, matrix_pigs_production[i,j]*0.225, rep(0,2), 0, rep(0,9), rep(-1,9),
0, 0, 0, 0, 0, 0, 0, 7.82*0.01, 1*0.01, 0, rep(0,9), rep(-1,9),
0, 0, 0, 0, 0, 0, 0, 0, 1*0.01,0, rep(0,9), rep(-1,9)),
nrow=477, ncol=28)



 # --- 22th Lines summing up total crop energy and protein production for FOOD ---

  food_energy <- (matrix_area_vec[i,j,]*Energy_crops)
  food_energy[1] <- food_energy[12] <- food_energy[18] <- food_energy[21] <- food_energy[23] <- food_energy[27] <- food_energy[3

  food_protein <- (matrix_area_vec[i,j,]*Protein_crops)
  food_protein[1] <- food_protein[12] <- food_protein[18] <- food_protein[21] <- food_protein[23] <- food_protein[27] <- food_pr
# Energy produced in FOOD
sparse_matrix_93 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(rep(0,366), food_energy, rep(0, 61), rep(0,28)))
```

```
# Protein produced in FOOD
sparse_matrix_94 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(rep(0,366), food_protein, rep(0, 61), rep(0,28)))

 # --- 23th Lines summing up total crop (grain) energy and protein use by livestock ---

# Energy used livestock
sparse_matrix_95 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(rep(0, 488), E_grain_dietary_req[i,j,], rep(0,19)))
# Protein used livestock
sparse_matrix_96 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(rep(0, 488), P_grain_dietary_req[i,j,], rep(0,19)))

 # --- 24th Line summing up total temporary fodder production in tons DM, Energy MJ and protein tons ---

sparse_matrix_97 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(matrix_area_vec[i,j,1],0,0,0,0,0,0,0,0,0,matrix_area_vec[
0,0,matrix_area_vec[i,j,21],0,matrix_area_vec[i,j,23],0,0,0,matrix_area_vec[i,j,27],0,0,matrix_area_vec[i,j,30],0,0,matrix_area_
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,matrix_area_vec[i,j,58],0,0,0,
rep(0, 427), rep(0,28)))

sparse_matrix_98 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(matrix_area_vec[i,j,1]*Energy_crops[1],0,0,0,0,0,0,0,0,0,
0,0,matrix_area_vec[i,j,21]*Energy_crops[21],0,matrix_area_vec[i,j,23]*Energy_crops[23],0,0,0,matrix_area_vec[i,j,27]*Energy_cro
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,matrix_area_vec[i,j,58]*Energy_crops[58],0,matrix_area_vec[i,j,60],0,
rep(0, 427), rep(0,28)))

sparse_matrix_99 <- simple_triplet_matrix(rep(1, 516), seq(1:516), c(matrix_area_vec[i,j,1]*Protein_crops[1],0,0,0,0,0,0,0,0,0,0
0,0,matrix_area_vec[i,j,21]*Protein_crops[21],0,matrix_area_vec[i,j,23]*Protein_crops[23],0,0,0,matrix_area_vec[i,j,27]*Protein_
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,matrix_area_vec[i,j,58]*Protein_crops[58],0,matrix_area_vec[i,j,60],0,
rep(0, 427), rep(0,28)))


 # --- 25th Lines summing up total temporary fodder energy and Protein consumption by livestock in MJ and tons ---
 # Energy
sparse_matrix_100 <- simple_triplet_matrix(rep(1,516), seq(1:516), c(rep(0,488), E_grass_dietary_req[i,j,], rep(0,19)))
 # Protein
sparse_matrix_101 <- simple_triplet_matrix(rep(1,516), seq(1:516), c(rep(0,488), P_grass_dietary_req[i,j,], rep(0,19)))

 # --- 24th Lines of zero before calculating the difference between produce temporary fodders and total consumed fodders (in ENER

sparse_matrix_102 <- simple_triplet_matrix(rep(1,516), seq(1:516), c(rep(0,488), rep(0,28)))
sparse_matrix_103 <- simple_triplet_matrix(rep(1,516), seq(1:516), c(rep(0,488), rep(0,28)))

 # --- 25th second Right side of the matrix defining statistical variable (exit calculations) ---

# 1. total food crops energy production
      # 2. total food crops protein production
      # 3. total grain energy used livestock
      # 4. total grain protein used livestock
      # 5. total temporary fodder [tDM]
      # 6. total energy temporary fodder [MJ]
      # 7. total protein temporary fodder [tons]
      # 8. total fodder energy used livestock
      # 9. total fodder protein used livestock
      # 10. difference produced temporary fodders - consumed in [MJ]
      # 11. percentual surplus consumed fodder used by permanent pastures

sparse_matrix_104 <- simple_triplet_matrix(c(rep(478,11), rep(479,11), rep(480,11), rep(481,11), rep(482,11), rep(483,11), rep(4
```

```
rep(486,11), rep(487,11), rep(488,11)), rep(seq(1:11),11),
c(-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0,
  0, 0, 0, 0, 0, 1, 0, -1, 0, -1, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, ifelse(permanent_pastures_available_biomass_t_DM[i,j]>0, 1/(permanent_pastures_available_biomass_t_
nrow=488, ncol=11)


sparse_matrix_105 <- simple_triplet_matrix(rep(1,527), seq(1:527), c(rep(0,183), rep(-1, 61), rep(0,61), rep(1, 61),  rep(0,161)
```

6.3.6 Final matrix contruction

```
    # --- Built up full matrix binding all submatrix parts created until now ---
        crop_yield_curve_sparse_matrix <- cbind(sparse_matrix_1, sparse_matrix_2, sparse_matrix_3, sparse_matrix_4, sparse_matri
        crop_yield_plateau_sparse_matrix <- cbind(sparse_matrix_9,sparse_matrix_10, sparse_matrix_11, sparse_matrix_12, sparse_m
        crop_stovers_yield_curve_sparse_matrix <- cbind(sparse_matrix_17,sparse_matrix_18, sparse_matrix_19, sparse_matrix_20, s
        crop_nitrogen_fixation_sparse_matrix <- cbind(sparse_matrix_25, sparse_matrix_26, sparse_matrix_27, sparse_matrix_28, sp
        crop_nitrogen_budgets_sparse_matrix <- cbind(sparse_matrix_33, sparse_matrix_34, sparse_matrix_35, sparse_matrix_36, spa
        N_total_available_for_crops_sparse_matrix <- cbind(sparse_matrix_41, sparse_matrix_42, sparse_matrix_43, sparse_matrix_4
        crop_sub_food_yield <- cbind(sparse_matrix_49, sparse_matrix_50, sparse_matrix_51, sparse_matrix_52, sparse_matrix_53, s
        crop_sub_feed_yield <- cbind(sparse_matrix_57, sparse_matrix_58, sparse_matrix_59, sparse_matrix_60, sparse_matrix_61, s
        grain_energy_sparse_m <- cbind(sparse_matrix_65,sparse_matrix_66)
        fodder_energy_sparse_m <- cbind(sparse_matrix_67, sparse_matrix_68)
        stovers_energy_sparse_m <- cbind(sparse_matrix_69, sparse_matrix_70)
        grain_protein_sparse_m <- cbind(sparse_matrix_71, sparse_matrix_72)
        fodder_protein_sparse_m <- cbind(sparse_matrix_73, sparse_matrix_74)
        stovers_protein_sparse_m <- cbind(sparse_matrix_75, sparse_matrix_76)
        livestock_manure_sparse_m <- sparse_matrix_77
        livestock_products_energy_sparse_m <- sparse_matrix_78
        livestock_TLU_sparse_m<- sparse_matrix_79
        livestock_minimum_animals_number_low_boudary <- sparse_matrix_80
        livestock_minimum_animals_number_upper_boudary <- sparse_matrix_81
        contraint_min_yield_crop_categories_m <- rbind(sparse_matrix_82,sparse_matrix_83,sparse_matrix_84,sparse_matrix_85,spars


        final_sparse_matrix <- rbind(crop_yield_curve_sparse_matrix, crop_yield_plateau_sparse_matrix, crop_stovers_yield_curve_
                                                                N_total_available_for_crops_sparse_matrix, crop_sub_food_yield,
                                                                grain_protein_sparse_m, fodder_protein_sparse_m, stovers_protei
                                                                livestock_minimum_animals_number_low_boudary, livestock_minimum
        final_sparse_matrix <- cbind(final_sparse_matrix, sparse_matrix_92)

        final_sparse_matrix_with_statistics <- rbind(final_sparse_matrix, sparse_matrix_93, sparse_matrix_94, sparse_matrix_95,
                                                                                                sparse_matrix_100, sparse_matrix
        final_sparse_matrix_with_statistics <- cbind(final_sparse_matrix_with_statistics, sparse_matrix_104)
```

```
# Final coefficient matrix
       final_sparse_matrix_with_statistics <- rbind(final_sparse_matrix_with_statistics, sparse_matrix_105)
```

6.3.7 Direction (DIR) and Right Hand Side (RHS)

```
          ## --- Set directions of the contrains inequalities ---


            # 1. yield curve
            # 2. yield plateau
            # 3. budgeting, total available fertiliser, SUB_food yield
            # 4. share of maximum available SUB_feed yield. For FODDERS crops the Sub_feed yield is equal to the total yield!
            # 5. livestock manure
            # 6.livestock numbers in head/TLU
            # 7. livestock constraints
            # 8. crop yield constraints
            # 9. statistics
                dir <- c(rep("==", 61),
                        c("==", rep("<=", 10),"==", rep("<=", 10), "==", rep("<=", 6), "==", rep("<=", 27), "==", rep("<=", 3)),
                        rep("==", 183), c("<="), rep("==",61) ,
                        c("==", rep(">=", 10), "==", rep(">=", 5), "==", rep(">=", 2), "==", ">=", "==", rep(">=", 3), "==", rep
                        c(">=",">=",">=",">=",">=",">=","=="),
                        rep("==", 9), rep("==", 9),
                        rep(">=", 5), rep("<=", 9),
                        rep(">=", 10),
                        rep("==", 11), ">")

     ## --- Set right hand side values of the inequalities ---

          # 1. yield curve, plateau, N stovers and N fix
          # 2. N budget
          # 3. N available for fertilisers
          # 4. livestock dietary requirements
          # 5. calculate real animal numbers
          # 6. calculate animal numbers TLU
          # 7. animal numbers lower boundary
          # 8. animal numbers upper boundary
          # 9. primary cereals minimum
          # 10. secondary cereals
          # 11. pulses
          # 12. oilcrops
          # 13. roots
          # 14. industrial
          # 15. fodder
          # 16. vegetable
          # 17. fruits
          # 18. non-food
          # 19. statistics
               rhs <- c(rep(0,61),matrix_yield_max_vec[i,j,], rep(0,122),
                        (N_deposition[i,j,1]*matrix_area_vec[i,j,]+N_cyanobacteria*matrix_area_vec[i,j,]),
                        (+(N_conventional_manure[i,j]*K_conv_manure_surplus*global_production_share_coefficient_conv)+(N_conven
                        rep(0, 122), 0,-(permanent_pastures_available_biomass_t_DM[i,j] * permanent_pastures_energy_protein_den
                        rep(0,9),
```

```
                    rep(0,9),
                    rep(0,5),
                    rep(0,9),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="cereal")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="sec.cereal")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="pulse")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="oilcrops")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="roots")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="industrial")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="fodder")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,which(crop_info$CropGroup=="vegetable")])*0.0),
                   (sum(matrix_yield_max_vec[i,j,c(2, 3, 9, 13, 19, 20, 28, 36, 39)])*0.0),
                   (sum(matrix_yield_max_vec[i,j,c(15, 16, 44, 53, 54)])*0.05),
                    rep(0,11), 0)

## --- Build ROI final contraint matrix---
  contst1 <- L_constraint(final_sparse_matrix_with_statistics, dir, rhs)
```

6.3.8 Set optimisation function and solve

```
    ## --- Set ROI objective function ---

  # set fodder crops AND non food crops to a insignifican energy production
                    Energy_crops_obj_fod[c(1, 12, 18, 21, 23, 27, 30, 33, 58, 44, 53, 54, 16)] <- Energy_crops_obj[c(1, 12
  # set non-fodder crops to zero (for feed yield) AND fodder crops (for food yield)
        Energy_crops_obj_fod[c(2:11,13:15,17,19:20,22,24:26,28:29,31:32,34:43,45:52,55:61)] <- Energy_crops_obj[c(1, 12, 1

        # 1. Energy_food_crops * Area * SUByield_food + Energy_fooder_Crops * Area * SUBYied_feed
        # 2. cow milk in Liters * milk_density
        # 3. cow meat
        # 4. goat milk in Liters/year * milk_density
        # 5. goat meat
        # 6. sheep milk in Liters/year * milk_density
        # 7. sheep meat
        # 8. pigs meat
        # 9. broilers meat   Kg_animal * number_animal_year
        # 10. heans eggs: num. eggs per layer times average 1 egg weight in Kg
        # 11. Livestock in TLU
        # 12. statistics
                    obj <- c(rep(0,366), matrix_area_vec[i,j,] * Energy_crops_obj, matrix_area_vec[i,j,] * Energy_crops_ob
```

```
                    # Define variables names

                    obj <- L_objective(obj, names = c(paste0("Y_", crop_info$CropName[1:61]),
                                                                                    paste0("harv_N_", crop_info$Crop
                                                                                    paste0("stov_N_", crop_info$Crop
                                                                                    paste0("pool_N_", crop_info$Crop
                                                                                    paste0("fix_N_", crop_info$CropN
                                                                                    paste0("fert_N", crop_info$CropN
                                                                                    paste0("Sub_food_Y", crop_info$C
                                                                                    paste0("Sub_feed_Y", crop_info$C
                                                                                    "No.dairy", "No.beef", "No.dairy
                                                                                    "No.milk", "No.meat", "No.milk_g
                                                                                    "TLU.dairy", "TLU.beef", "TLU.da
                                                                                    "total_food/feed_crops_energy_pr
                                                                                    "total_grain_energy_used_livesto
                                                                                    "total_temporary_fodder_[tDM]",
                                                                                    "total_fodder_energy_used_livest
                                                                                    "difference_produced_consumed_te


                    # et continuos variable type (clp solver only accepts continuos variables)
                    types <- rep("C", 527)

                    # et  variables upper and lower bounds
          bounds <- V_bound(li=seq(1:527), lb=c(rep(0, 525), rep(-10e13,2)), ui=seq(1:527), ub=c(rep(10e13, 498), c(rep(

                    # Print vector position (optimised gridcell vector number)
                    print(j)

                    # Built and solve optimisation problem using ROI package
                    # max iterations set to 5000 and max computing time set to 10 sec.
                    prob <- OP(obj, contst1, types = types, maximum= TRUE, bounds=bounds)
                    solution <- ROI_solve(prob, solver = "clp", control=list(amount=1, iterations=5000, seconds=10))


                    # Fake brakets opening, just for pdf generation; reactivate "else" statement
                    # 1. create arrays containing variable solution
                     # 2. If the vector was in a gridcell position with NA data,
                     #    the optimisation is not performed, and NA is saved in the corresponding position

                    {{


                    counter[i,j] <- solution$status$code
                          matrix_final_yield[i,j,] <- solution$solution[1:61]
matrix_final_N_assigned_to_crop[i,j,] <- solution$solution[306:366]
                          matrix_sub_food_yield[i,j,] <- solution$solution[367:427]
                          matrix_sub_feed_yield[i,j,] <- solution$solution[428:488]
                          number_of_livestock_heads[i,j,] <- solution$solution[499:507]
number_of_livestock_standing_animals[i,j,] <- solution$solution[c(492, 494, 495, 496)]
                          N_total_available[i,j,] <- solution$solution[498]
                          statistics[i,j,] <- solution$solution[517:527]
BNF[i,j,] <- solution$solution[(244+which(Fixed_N!=0))]
                }
```

```
        #else
 {                      matrix_final_yield[i,j,] <- rep(NA, 61)
                                matrix_final_N_assigned_to_crop[i,j,] <- rep(NA, 61)
                                matrix_sub_food_yield[i,j,] <- rep(NA, 61)
                                matrix_sub_feed_yield[i,j,] <- rep(NA, 61)
                                number_of_livestock_heads[i,j,] <- rep(NA, 9)
                number_of_livestock_standing_animals[i,j,] <- rep(NA,4)
                                N_total_available[i,j,] <- NA
                                counter[i,j] <- NA
                                statistics[i,j,] <- rep(NA,11)
                BNF[i,j,] <- rep(NA,length(which(Fixed_N!=0)))
                                }
}
```

6.3.9 Data saving

```
# ---  FREE UP MEMORY DELETING VARIABLES ---
library(pryr)
print(mem_used())
print(mem_change(rm(matrix_area_vec, matrix_yield_max_vec, Nyear_livestock_production_available, E_livestock_products, matrix_li
gc()

# ---  DATA SAVING   ---

# set correct paths to saving directories
crs <- c("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0")

# Total crops yield
for (i in 1:61) {
        raster_1 <- raster(matrix(matrix_final_yield[,,i], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs
        writeRaster(raster_1, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no, "/sensitivity_", se
}

rm(raster_1, matrix_final_yield)
gc()


# Total N from organic manure allocate to each crop
for (i in 1:61) {
    raster_2 <- raster(matrix(matrix_final_N_assigned_to_crop[,,i], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90,
    writeRaster(raster_2, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no, "/sensitivity_", sensit
}

rm(raster_2, matrix_final_N_assignet_to_crop)
gc()

# Animal number of heads
raster_3 <- raster(matrix(number_of_livestock_heads[,,1], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_3, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_4 <- raster(matrix(number_of_livestock_heads[,,2], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_4, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_5 <- raster(matrix(number_of_livestock_heads[,,3], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
```

```
writeRaster(raster_5, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_6 <- raster(matrix(number_of_livestock_heads[,,4], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_6, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_7 <- raster(matrix(number_of_livestock_heads[,,5], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_7, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_8 <- raster(matrix(number_of_livestock_heads[,,6], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_8, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_9 <- raster(matrix(number_of_livestock_heads[,,7], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_9, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitivi
raster_10 <- raster(matrix(number_of_livestock_heads[,,8], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs
writeRaster(raster_10, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_11 <- raster(matrix(number_of_livestock_heads[,,9], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs
writeRaster(raster_11, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv


rm(raster_3, raster_4, raster_5, raster_6, raster_7, raster_8, raster_9, raster_10, raster_11, number_of_livestock_heads)
gc()

# Total available Nitrogen in organic manure
raster_12 <- raster(matrix(N_total_available[,,1], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_12, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv

rm(raster_12, N_total_available)

# Error counter
raster_13 <- raster(matrix(counter[,], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_13, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv

rm(raster_13, counter)
gc()

# Model statistics
raster_14 <- raster(matrix(statistics[,,1], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_14, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_15 <- raster(matrix(statistics[,,2], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_15, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_16 <- raster(matrix(statistics[,,3], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_16, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_17 <- raster(matrix(statistics[,,4], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_17, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_18 <- raster(matrix(statistics[,,5], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_18, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_19 <- raster(matrix(statistics[,,6], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_19, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_20 <- raster(matrix(statistics[,,7], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_20, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_21 <- raster(matrix(statistics[,,8], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_21, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_22 <- raster(matrix(statistics[,,9], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_22, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_23 <- raster(matrix(statistics[,,10], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_23, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_24 <- raster(matrix(statistics[,,11], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
writeRaster(raster_24, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
```

```
rm(raster_14, raster_15, raster_16, raster_17, raster_18, raster_19, raster_20, raster_21, raster_22, raster_23, raster_24, stat
gc()

# Crops SUB_Food yield
for (i in 1:61) {
        raster_25 <- raster(matrix(matrix_sub_food_yield[,,i], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs =
        writeRaster(raster_25, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_",
}

rm(raster_25, matrix_sub_food_yield)
gc()

# Crops SUB_Feed yield
for (i in 1:61) {
        raster_26 <- raster(matrix(matrix_sub_feed_yield[,,i], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs =
        writeRaster(raster_26, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_",
}

rm(raster_26, matrix_sub_feed_yield)
gc()

# Standing animals
raster_27 <- raster(matrix(number_of_livestock_standing_animals[,,1], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90
writeRaster(raster_27, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_28 <- raster(matrix(number_of_livestock_standing_animals[,,2], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90
writeRaster(raster_28, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_29 <- raster(matrix(number_of_livestock_standing_animals[,,3], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90
writeRaster(raster_29, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv
raster_30 <- raster(matrix(number_of_livestock_standing_animals[,,4], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90
writeRaster(raster_30, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sensitiv

rm(raster_27, raster_28, raster_29, raster_30, number_of_livestock_standing_animals)
gc()


# BNF
for (i in 1:length(which(Fixed_N!=0))) {
    raster_28 <- raster(matrix(BNF[,,i], nrow=2160, byrow=TRUE), xmn=-180, xmx=180, ymn=-90, ymx=90, crs = crs)
    writeRaster(raster_28, filename = paste0("Organic_optimisation_sub_model/results/script_", script_no,  "/sensitivity_", sens
}

rm(raster_28, BNF)
gc()
```

7 GOANIM model - V2

The GOANIM-model (V2) works in the same way as version I and use the same data sources. We give here a detailed explanation of the main differences in comparison to the Version I of the model.

Objective function The GOANIM-model (V2) simulates only the crop yield response to N supply to soils. Therefore, the optimization procedure is set to maximize only the total energy produced by croplands, as in equation (24):

$$Max : \sum_{i=1}^{61} A_i \times Y_i \times [E]_i \tag{24}$$

Where A_i is the harvested area under each crop species i $[ha]$, Y_i is the yield of crop species i $[tons\ ha^{-1}]$, and $[E]_i$ is the energy density of crop species i $[MJ\ tons^{-1}]$.

Organic livestock numbers The GOANIM-model (V2) does not actively simulated livestock animal densities, but it is based on the hypothesis that livestock densities and distribution will not change in comparison to the current situation. Therefore, livestock organic densities are represented by the current conventional livestock densities maps multiplied by the global share of organic farming set by the user (describe at paragraph 4.1.6).

Organic livestock feed requirements Organic livestock feed requirements are calculated considering the livestock densities as described in the previous paragraph and the ME and CP dietary requirements described at paragraph 4.1.7.1.

In order to consistently calculate the total ME and CP livestock requirements, we considered that the ratio of the across-crop species average energy and protein density (i.e. $\frac{\sum_{i=1}^{61}[E]_i}{\sum_{i=1}^{61}[P]_i}$) stays constant. Yet, resources to satisfy these requirements come from "pools" where the energy and protein produced by every crop species are pulled together. Energy and protein content of each unit of feed are defined by the stechiometry of each feed commodity, and hence they are not separable. As instance, consider a dietary requirement for one cow of 1000 MJ and 5 $kg\ protein$. To satisfy these requirements, 10 kg of wheat and maize grains are available, containing respectively 400 MJ and 3 $kg\ protein$, and 600 MJ and 4 $kg\ protein$. The total available energy and protein "pool" would then be $400 + 600 = 1000MJ$ and $3 + 4 = 7\ kg\ protein$. Therefore, the available energy is just enough to feed one cow, while the available protein exceed the cow's requirements. Then, possible approach is to use all the energy available and just 5 out of the 7 $kg\ protein$; nevertheless, this approach is not correct since the two quantities -i.e. energy and protein- are not separable and if all the energy is fed to the cow (i.e. the whole original biomass), necessary all the proteins are used as feed. Therefore, the protein 'excess' of 7 kg is not any more available for other purposes. Therefore, according to this example, we have to correct the amount of protein fed to the cow to match the energy one (which is the limiting factor). Yet, we do not know any more the original composition of crop products (all crops are pooled together). Therefore, we can just apply a more approximation correction as follows:

- For any grid-cell where protein is the limiting factor, i.e. where:

$$\frac{P_{livestock-requirements}}{P_{pool-crop-products}} > \frac{E_{livestock-requirements}}{E_{pool-crop-products}}$$

  then: $E_{consumed-by-livestock} = E_{pool-crop-products} \times \frac{P_{livestock-requirements}}{P_{pool-crop-products}}$

- For any grid-cell where energy is the limiting factor, i.e. where:

$$\frac{P_{livestock-requirements}}{P_{pool-crop-products}} < \frac{E_{livestock-requirements}}{E_{pool-crop-products}}$$

then: $P_{consumed-by-livestock} = P_{pool-crop-products} \times \frac{E_{livestock-requirements}}{E_{pool-crop-products}}$

Organic livestock food production

Crop production budgets and total food production Food budgets are calculated as the difference between total crop production minus the crop production needed for feeding livestock animal populations. For any given grid-cell, food budgets can be either positive (meaning that crop production is sufficient to feed livestock in that particular grid-cell), or negative, meaning that crop production is not sufficient to feed livestock in that particular grid-cell. Total global food production is then calculated as the sum between food budgets and the energy into livestock food commodities.

8 Code - organic sub-model - Version II

8.1 Copyrights and environment preparation

```
#  Copyright 2018 Pietro Barbieri <pbarbieri@avakas-frontend2>
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.


#load packages
library(raster)
library(clpAPI, lib.loc="/home/pbarbieri/R/x86_64-pc-linux-gnu-library/3.2")
library(ROI.plugin.clp, lib.loc="/home/pbarbieri/R/x86_64-pc-linux-gnu-library/3.2")
library(ROI)
library(slam)
library(doParallel)
library(foreach)

# --- Set Working Directory and Temporary files directory path  ---

setwd("/home/pbarbieri/global/Version_I_first_inclusion_of_land_use_change_rules/")
getwd()
library(unixtools)
set.tempdir("/scratch/pbarbieri/R_temporary_files")
tempdir()
```

Table 6: Simplified model structure as coded, reporting a simplified version of all equations coefficients. Color boxes represents different structural matrices as coded in R

	Crop yield (61)	N crop (61)	N stovers (61)	N pool (61)	N BNF (61)	N fertilisation (61)	Direction	RHS
Yield curve slope (61)	1						=	0
Yield curve plateau (61)		1/[N]					≤	Y_{max}
Stovers yield (61)	RPR		$-K_{stover}$				=	0
NBF (61)	k_{fix}				-1		=	0
N budget (61)		A_i	A_i	A_i	$-A_i$	$-A_i$	=	$IN_{dep} + N_{cyano}$
Avail. N fertiliser (1)						A_i	≤	$Org\text{-}manure + conv\text{-}manure + wastewater$

Table 7: simplified model structure as coded, reporting each single sparse sub-matrix (SM). Color boxes represents different structural matrices as coded in R (sparse matrices)

	Crop yield (61)	N crop (61)	N stovers (61)	N pool (61)	N BNF (61)	N fertilisation (61)	Direction	RHS
Yield curve slope (61)	SM1	SM2	SM3	SM4	SM5	SM6	DIR	RHS
Yield curve plateau (61)	SM7	SM8	SM9	SM10	SM11	SM12		
Stovers yield (61)	SM13	SM14	SM15	SM16	SM17	SM18		
NBF (61)	SM19	SM20	SM21	SM22	SM23	SM24		
N budget (61)	SM25	SM26	SM27	SM28	SM29	SM30		
Avail. N fertiliser (1)	SM31	SM32	SM33	SM34	SM35	SM36		

```
# --- CORE CONTROLS   ---
detectCores()
cluster<-makeCluster(23)
#start cluster
registerDoParallel(cluster)
#check number of cores used
getDoParWorkers()
# Stop cluster
#registerDoSEQ()
```

8.2 Data and parameters loading

8.2.1 Load global binary parameters for scenario analysis

```
# --- Binary coefficients to activate/disactivate variables and global scenario coefficients ---

                global_production_share_coefficient_conv <- 0
                global_production_share_coefficient_org <- 1 - global_production_share_coefficient_conv

# 0-1 Desactivate/Activate the use of conventional surplus manure.
# Manure as already been accounted for losses in the conventional-submodel
                        K_conv_manure_surplus <- 0
                        # 0-1 Desactivate/Activate the use of conventional surplus manure.
                        # Wastewater as already been accounted for losses in the conventional-submodel
                        K_waste_water <- 0
```

8.2.2 Load crops areas, and yield curve plateau

```
        # Load organic maximum yield t/ha (Ponisio modified coefficients, see model documentation for details), and Organic harve
        foreach (i = 1:61, .packages="raster") %do% {
        matrix_yield_max_vec[,,i] <- as.vector(raster(paste0("Organic yield files/Modified_Ponisio_organic_yield_gap/", crop_inf
        matrix_area_vec[,,i] <- as.vector(raster(paste0("/home/pbarbieri/global/Land Use Change/100% conversion/Area_hectares/",
    }

# Crop yield curve slope
        Crops_yield_curve_slope <- 1/crop_info$N
        # Crop residues yield curve slope
        Stovers_yield_curve_slope <- 1/crop_info$N_crop_residues
        Stovers_yield_curve_slope[Stovers_yield_curve_slope==(Inf)] <- 0
   # RPR coefficients to estimate crop residues from harvested biomass
                Stovers_RPR <- crop_info$RPR_corrected_optimisation_model

                # Crop residues recycling in organic farming
    crop_res_avail_org <- array(dim=c(1, 9331200))
    crop_res_avail_org[,] <- as.vector((1-raster("Coefficients/map_coeff_residues_recycle_organic.tif")/1.45))

# coefficient to calculate BNF, see model documentation for further details
        Fixed_N <- crop_info$NiC * crop_info$X1.NHI * crop_info$Proot_NOT_TO_BE_USED * crop_info$Ndfa

# coefficient to calculate cyanobacteria nitrogen fixation, see model documentation for further details
```

```
        cereals_tuber_oils <- 0.012
        roots <- which(crop_info$CropGroup=="roots")
        cereal <- which(crop_info$CropGroup=="cereal")
        sec.cereal <- which(crop_info$CropGroup=="sec.cereal")
        oilcrops <- which(crop_info$CropGroup=="oilcrops")
        N_cyanobacteria <- array(dim=c(1,61))
        N_cyanobacteria[roots] <- N_cyanobacteria[cereal] <- N_cyanobacteria[sec.cereal] <- N_cyanobacteria[oilcrops]  <- cereal
        N_cyanobacteria[50] <-  0.1
        N_cyanobacteria[43] <-  0.02
        N_cyanobacteria <- ifelse(!is.na(N_cyanobacteria), N_cyanobacteria, 0)

# load crops energy density coefficient and transform into MJ per tons
  Energy_crops <- crop_info$metabolisable_Energy_MJ.kgDM*1000
```

8.2.3 Load organic manure inputs

```
        ## --- organic manure input - % of current (i.e. conventional) manure ---

        # Load total conventional manure estimate
        total_N_manure <- raster("Script_generated_files/total_N_manure.tif")
         # correction of errors in Manure maps --> whatever more than 3000 t gridcell give 3000
        rclmat <- matrix(c(3000, Inf, 3000), ncol=3, byrow=T)
        total_N_manure <- reclassify(total_N_manure, rclmat)
        total_N_manure <- extend(total_N_manure, extent(-180,180,-90,90))

        # correction for losses (IPCC procedure)
        # 1. correction to account for the 5% of crops (in terms of cropland share) that we did not include in the 61 crops
        # 2. correction to estimate 1% losses due to direct N2O emissions and N2 (0.66%)
        # 3. correction to estimate the 20% losses due to NH3 and NO volatilisation --> range coefficient: 0.05-0.5
        total_N_manure_corrected_95crops <- total_N_manure * 0.96 * (1 -  (0.0166 + 0.2))

        # Load leaching areas. The areas where leaching occurs were estimated according to the IPCC procedure
                leaching_areas_mask <- raster("/home/pbarbieri/global/Version_I_first_inclusion_of_land_use_change_rules/Climati
        # 4. leaching losses coefficient 0.3: correction to estimate the 30% losses due to leaching
        mask_correction_leaching <- leaching_areas_mask * 0.7
        mask_correction_leaching <- reclassify(mask_correction_leaching, cbind(0,1))
        mask_correction_leaching_vec <- array(dim=c(1, 9331200))
        total_N_manure_corrected_95crops <- total_N_manure_corrected_95crops - (total_N_manure * 0.96 * (1 - mask_correction_lea

  # transform raster into Array
        N_organic_manure <- array(dim=c(1,9331200))
        N_organic_manure[,] <- as.vector(total_N_manure_corrected_95crops * global_production_share_coefficient_org)
```

8.2.4 Inputs from conventional farming systems

```
## ____________  CONVENTIONAL FARMING INPUTS - from conventional submodel results _____________

        # N from conventional manure surplus computed in the conventional submodel - non residistributed
        N_conventional_manure <- array(dim=c(1,9331200))
        N_conventional_manure[,] <- as.vector(raster("Script_generated_files/manure_N_surplus_total_100_manure_cropland.tif"))
```

```
        # N from conventional manure surplus computed in the conventional submodel - residistributed according
        # to the total harvested area. To be activated according to the scenario to test
        # N_conventional_manure[,] <- as.vector(raster("Script_generated_files/N_redistributed_manure_surplus_100_harvested_area.


# N from conventional waste water surplus computed in the conventional submodel residistributed; current population and diets
        N_conventional_waste_water <- array(dim=c(1,9331200))
        #load correct file according to the scenario to test
                waste_water <- extend(raster("Script_generated_files/wastewater_N_surplus_total_100_manure_cropland.tif"), exter
        N_conventional_waste_water[,] <- as.vector(waste_water)
```

8.2.5 Load IPCC N leaching coefficient and N atmospheric deposition maps

```
mask_correction_leaching_vec[,] <- as.vector(mask_correction_leaching)              # leaching coefficients


        N_deposition <- array(dim=c(1,9331200,1))                                  # N atmospheric depostions 1993 per hectare
        N_deposition[,,1] <- as.vector(raster("N atmospheric deposition/N_deposition_per_hectare_of_cropland2000.tif") * mask_co


Energy_crops <- crop_info$metabolisable_Energy_MJ.kgDM*1000    # transformed into MJ/tons
```

8.2.6 Generate arrays for storing optimisation results

```
## --- Generate arrays for storing the optimisation results  ---

# Final organic yiels, N assigned to crops
    matrix_final_N_assigned_to_crop <- matrix_final_yield <- array(dim=c(1,9331200,61))
# Counter_for_model_failures_in_finding_a_solution  [0-1]
    counter <- array(dim=c(1,9331200))

    # BNF

    BNF <- array(dim=c(1,9331200,length(which(Fixed_N!=0))))
```

8.3 Linear optimisation procedure

8.3.1 Initialise ”for” loop

```
# set first array to 1 --> all sngle arrays are vectors
i=1

# initialise sensitivity_no: this number refers to the number of the scenario that the user want to test,
# and is automatically related to the name of the folders where all results will be saved.
# I.e. if sensitivity_no is set to 1, the results will be saved in the respecitive folders
# prepared by the use to contain the results of Scenario 1.
# initialise script_no: to reduce the time of computation, global maps are divided here in 24 subparts,
# that can be run in parallel. if script_no is set to 1, the optimisatio run the first part of the map out of 24,
# and it saves the results in the respective folder as expalained for the Version I of the model.
```

```
# An example of the resulting folder struction is given after this code chunk.
sensitivity_no <- NA
script_no <- NA

# set the number fo the gridcells on which we want to run the optimisation (from 1 to 9331200) --> the total number of cells is d

for (j in XXX:XXX) {

# Run the optimisation procedure only if the selected gridcell is part of continental land,
# e.g. if the cell correspond to an ocean it contains no data (NA);
# hence the optimisation procedure is not run and a correpective NA data is saved in all oputput arrays
# (see last chunk of the optimisation code procedure. this procedure helps reducting the computational time.

if (is.na(matrix_area_vec[i,j,1])==FALSE) {

}} # attention this is a fake ending of the loops, just for generating the documentation file. these bracketrs have to be deleted
```

8.3.2 Model coefficient matrix construction

```
# --- 1st part of the matrix giving the 61 yield curve constraint ---
sparse_matrix_1 <- simple_triplet_diag_matrix(rep(1,61), nrow=61)
sparse_matrix_2 <- simple_triplet_diag_matrix(-Crops_yield_curve_slope[], nrow=61)
sparse_matrix_3 <- simple_triplet_zero_matrix(61)
sparse_matrix_4 <- simple_triplet_zero_matrix(61)
sparse_matrix_5 <- simple_triplet_zero_matrix(61)
sparse_matrix_6 <- simple_triplet_zero_matrix(61)


# --- 2nd part of the matrix giving the 61 yield curve PLATEAU constraint ---
sparse_matrix_7 <- simple_triplet_diag_matrix(rep(1, 61), nrow=61)
sparse_matrix_8 <- simple_triplet_zero_matrix(61)
sparse_matrix_9 <- simple_triplet_zero_matrix(61)
sparse_matrix_10 <- simple_triplet_zero_matrix(61)
sparse_matrix_11 <- simple_triplet_zero_matrix(61)
sparse_matrix_12 <- simple_triplet_zero_matrix(61)


# --- 3rd part of the matrix giving the 61 N required by exported fraction of crop residues + the relative losses of the fraction
sparse_matrix_13 <- simple_triplet_diag_matrix(Stovers_RPR[], nrow=61)
sparse_matrix_14 <- simple_triplet_zero_matrix(61)
sparse_matrix_15  <- simple_triplet_diag_matrix((ifelse(-((Stovers_yield_curve_slope*1)/(crop_res_avail_org[i,j]+(2-(0.99+mask_c
-((Stovers_yield_curve_slope*1)/(crop_res_avail_org[i,j]+(2-(0.99+mask_correction_leaching_vec[i,j]))*(1-crop_res_avail_org[i,j]
sparse_matrix_16 <- simple_triplet_zero_matrix(61)
sparse_matrix_17 <- simple_triplet_zero_matrix(61)
sparse_matrix_18 <- simple_triplet_zero_matrix(61)


# --- 4th part of the matrix giving the 61 constraint for fixed nitrogen by pulses crops and by roots/cereals/oilcrops/sugarcane
sparse_matrix_19 <- simple_triplet_diag_matrix(Fixed_N[], nrow=61)
sparse_matrix_20 <- simple_triplet_zero_matrix(61)
sparse_matrix_21 <- simple_triplet_zero_matrix(61)
sparse_matrix_22 <- simple_triplet_zero_matrix(61)
sparse_matrix_23 <- simple_triplet_diag_matrix(rep(-1,61), nrow=61)
```

```
sparse_matrix_24 <- simple_triplet_zero_matrix(61)


# --- 5th part of the matrix giving the 61 constraint for nitrogen balancing for each crop including Ngrain, Nres, Npool, Nfixed,
sparse_matrix_25 <- simple_triplet_zero_matrix(61)
sparse_matrix_26 <- simple_triplet_diag_matrix(matrix_area_vec[i,j,], nrow=61)
sparse_matrix_27 <- simple_triplet_diag_matrix(matrix_area_vec[i,j,], nrow=61)
sparse_matrix_28 <- simple_triplet_diag_matrix(matrix_area_vec[i,j,], nrow=61)
sparse_matrix_29 <- simple_triplet_diag_matrix(-matrix_area_vec[i,j,], nrow=61)
sparse_matrix_30 <- simple_triplet_diag_matrix(-matrix_area_vec[i,j,], nrow=61)


# --- 6th part of the matrix giving the 1 constraint for nitrogen availability for N from manure ---
sparse_matrix_31 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_32 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_33 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_34 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_35 <- simple_triplet_matrix(rep(1, 61), seq(1:61), rep(0,61))
sparse_matrix_36 <- simple_triplet_matrix(rep(1, 61), seq(1:61), c(matrix_area_vec[i,j,]))

## --- Built up full matrix binding all submatrix parts ---
crop_yield_curve_sparse_matrix <- cbind(sparse_matrix_1, sparse_matrix_2, sparse_matrix_3, sparse_matrix_4, sparse_matrix_5, spa
crop_yield_plateau_sparse_matrix <- cbind(sparse_matrix_7,sparse_matrix_8, sparse_matrix_9, sparse_matrix_10, sparse_matrix_11,
crop_stovers_yield_curve_sparse_matrix <- cbind(sparse_matrix_13,sparse_matrix_14, sparse_matrix_15, sparse_matrix_16, sparse_ma
crop_nitrogen_fixation_sparse_matrix <- cbind(sparse_matrix_19, sparse_matrix_20, sparse_matrix_21, sparse_matrix_22, sparse_mat
crop_nitrogen_budgets_sparse_matrix <- cbind(sparse_matrix_25, sparse_matrix_26, sparse_matrix_27, sparse_matrix_28, sparse_matr
N_total_available_for_crops_sparse_matrix <- cbind(sparse_matrix_31, sparse_matrix_32, sparse_matrix_33, sparse_matrix_34, spars

# compute final coefficient sparse matrix
final_sparse_matrix <- rbind(crop_yield_curve_sparse_matrix, crop_yield_plateau_sparse_matrix, crop_stovers_yield_curve_sparse_m
```

8.3.3 Direction (DIR) and Right Hand side (RHS)

```
## --- Set directions of the contrains inequalities ---
 # 1. yield curve
# 2. yield plateau
# 3. budgeting, total available fertiliser
dir <- c(rep("==", 61),
c("==", rep("<=", 10),"==", rep("<=", 10), "==", rep("<=", 6), "==", rep("<=", 27), "==", rep("<=", 3)),
rep("==", 183), c("<="))


## --- Set right hand side values of the inequalities ---
  # 1. yield curve, plateau, N stovers and N fix
 # 2; N budget
# 3. N available for fertilisers
rhs <- c(rep(0,61),matrix_yield_max_vec[i,j,], rep(0,122),
(N_deposition[i,j,1]*matrix_area_vec[i,j,]+N_cyanobacteria*matrix_area_vec[i,j,]),
(+(N_conventional_manure[i,j]*K_conv_manure_surplus*global_production_share_coefficient_conv)+(N_conventional_waste_water[i,j]*K


## Build ROI package contraints matrix
contst1 <- L_constraint(final_sparse_matrix, dir, rhs)
```

8.3.4 Set optimisation function and solve

```
## --- set ROI objective function ---

obj <- (matrix_area_vec[i,j,]*Energy_crops)
# set fodder crops to zero AND cotton
obj[c(1, 12, 18, 21, 23, 27, 30, 33, 58, 44, 53, 54, 16)] <- 0.0000001
obj <- c(obj, rep(0,305))

# set variables names
obj <- L_objective(obj, names = c(paste0("Y_", crop_info$CropName[1:61]),
                                  paste0("harv_N_", crop_info$CropName[1:61]),
                                  paste0("stov_N_", crop_info$CropName[1:61]),
                                  paste0("pool_N_", crop_info$CropName[1:61]),
                                  paste0("fix_N_", crop_info$CropName[1:61]),
                                  paste0("fert_N", crop_info$CropName[1:61])))

# Set continuos variable type (clp solver only accepts continuos variables)
types <- rep("C", 366)

# Set  variables upper and lower bounds
bounds <- V_bound(li=seq(1:366), lb=c(rep(0, 366)), ui=seq(1:366), ub=c(rep(10e13, 366)))

# Print vector position (optimised gridcell vector number)
print(j)

# Built and solve optimisation problem using ROI package
prob <- OP(obj, contst1, types = types, maximum= TRUE, bounds=bounds)
solution <- ROI_solve(prob, solver = "clp", control=list(amount=1, iterations=5000, seconds=10))


                          # Fake brakets opening, just for pdf generation; reactivate "else" statement
                          # 1. create arrays containing variable solution
                           # 2. If the vector was in a gridcell position with NA data, the optimisation is not performed,
            # and NA is saved in the corresponding position

{{

counter[i,j] <- solution$status$code
matrix_final_yield[i,j,] <- solution$solution[1:61]
matrix_final_N_assigned_to_crop[i,j,] <- solution$solution[306:366]
BNF[i,j,] <- solution$solution[(244+which(Fixed_N!=0))]
}

#else   To be reactivated

 {                     matrix_final_yield[i,j,] <- rep(NA, 61)
                               matrix_final_N_assigned_to_crop[i,j,] <- rep(NA, 61)
                counter[i,j] <- NA
                BNF[i,j,] <- rep(NA,length(which(Fixed_N!=0)))
   }
}
```

References

1. Barbieri, P., Pellerin, S., Seufert, V. & Nesme, T. Changes in crop rotations would impact food production in an organically farmed world. *Nature Sustainability*. ISSN: 2398-9629. doi:10.1038/s41893-019-0259-5. http://dx.doi.org/10.1038/s41893-019-0259-5 (2019).
2. Yihui, X. Package knitr '. *CRAN Repository*. Yihui2019 (2019).
3. Tayleur, C. & Phalan, B. Organic farming and deforestation. *Nature Plants* **2,** 16098. ISSN: 2055026X (2016).
4. European Commission. *Commission Regulation (EC) No 889/2008* 2008.
5. Bartelt, K. D. & Bland, W. L. Theoretical analysis of manure transport distance as a function of herd size and landscape fragmentation. *Journal of Soil and Water Conservation (Ankeny)* **62,** 345–352. ISSN: 00224561 (2007).
6. Liu, J. *et al.* A high-resolution assessment on global nitrogen flows in cropland. *Proceedings of the National Academy of Sciences of the United States of America* **107,** 8035–8040. ISSN: 0027-8424 (2010).
7. Mueller, N. D. *et al.* Closing yield gaps through nutrient and water management. *Nature* **490,** 254–257. ISSN: 0028-0836 (2012).
8. Robinson, T. P. *et al.* Mapping the global distribution of livestock. *PLoS ONE* **9.** ISSN: 19326203. doi:10.1371/journal.pone.0096084 (2014).
9. Sheldrick, W., Keith Syers, J. & Lingard, J. Contribution of livestock excreta to nutrient balances. *Nutrient Cycling in Agroecosystems* **66,** 119–131. ISSN: 13851314 (2003).
10. Food and Agriculture Organization of the United Nations. *FAOSTAT Statistics Database* Rome, 2016. http://www.fao.org/faostat/en/{\#}data.
11. Dong, H. *et al.* in *IPCC Guidelines for National Greenhouse Gas Inventories* (2006).
12. Dentener, F *et al.* Nitrogen and sulfur deposition on regional and global scales : A multimodel evaluation. **20.** doi:10.1029/2005GB002672 (2006).
13. Hogh-Jensen, H., Loges, R., Jorgensen, F. V., Vinther, F. P. & Jensen, E. S. An empirical model for quantification of symbiotic nitrogen fixation in grass-clover mixtures. *Agricultural Systems* **82,** 181–194. ISSN: 0308521X (2004).
14. Smil, V. Nitrogen in crop production : An account of global flows. *Global Biogeochemical Cycles* **13,** 647–662 (1999).
15. Monfreda, C., Ramankutty, N. & Foley, J. A. Farming the planet: 2. Geographic distribution of crop areas, yields, physiological types, and net primary production in the year 2000. *Global Biogeochemical Cycles* **22,** 1–19. ISSN: 08866236 (2008).
16. INRA. *Alimentation des bovins, ovins et caprins* Quae. ISBN: 978-2-7592-0020-7 (2007).
17. INRA, CIRAD, AFZ & FAO. *Feedipedia - Animal Feed Resources Information System* 2016. http://feedipedia.org/.
18. Smil, V. Nitrogen and Food Production: Proteins for Human Diets. *AMBIO: A Journal of the Human Environment* **31,** 126–131. ISSN: 0044-7447 (2002).
19. De Klein, C. *et al.* in *IPCC Guidelines for National Greenhouse Gas Inventories* (2006). ISBN: 4887880324.

20. Zomer, R. J. *et al. Trees and Water: Smallholder Agroforestry on Irrigated Lands in Northern India.* 47. ISBN: 9789290906414. doi:10.1016/j.agee.2008.01.014 (2007).

21. Zomer, R. J., Trabucco, A., Bossio, D. A. & Verchot, L. V. Climate change mitigation: A spatial analysis of global land suitability for clean development mechanism afforestation and reforestation. *Agriculture, Ecosystems and Environment* **126,** 67–80. ISSN: 01678809 (2008).

22. Eichner, M. J. Nitrous Oxide Emissions from Fertilized Soils: Summary of Available Data. *Journal of Environment Quality* **19,** 272. ISSN: 0047-2425 (1990).

23. Ruser, R. *et al.* Emission of N2O, N2 and CO2 from soil fertilized with nitrate: Effect of compaction, soil moisture and rewetting. *Soil Biology and Biochemistry* **38,** 263–274. ISSN: 00380717 (2006).

24. Van Drecht, G., Bouwman, A. F., Harrison, J. & Knoop, J. M. Global nitrogen and phosphate in urban wastewater for the period 1970 to 2050. *Global Biogeochemical Cycles* **23,** 1–19. ISSN: 08866236 (2009).

25. Doxsey-Whitfield, E. *et al.* Taking Advantage of the Improved Availability of Census Data: A First Look at the Gridded Population of the World, Version 4. *Papers in Applied Geography* **1,** 226–234. ISSN: 2375-4931 (2015).

26. Hijmans, R., van Etten Jacob, Cheng, J. & Mattiuzzi, M. Package raster'. *CRAN Repository.* https://cran.r-project.org/web/packages/raster/raster.pdf (2016).

27. Bivand, R., Keitt, T., Pebesma, E. & Rouault, E. Package rgdal'. *CRAN Repository.* https://cran.r-project.org/web/packages/rgdal/rgdal.pdf (2017).

28. Pierce, D. Package ncdf4'. *CRAN Repository* (2017).

29. Calaway, R., Weston, S. & Tenenbaum, D. Package doParallel '. *CRAN Repository,* 1–4 (2018).

30. Calaway, R. & Weston, S. Package foreach '. *CRAN Repository,* 1–10 (2017).

31. Ponisio, L. C. *et al.* Diversification practices reduce organic to conventional yield gap. *Proceedings of the Royal Society B* **282,** 1–7. ISSN: 0962-8452 (2015).

32. Erb, K.-h. *et al.* Exploring the biophysical option space for feeding the world without deforestation. *Nature Communications* **7** (2016).

33. Licker, R. *et al.* Mind the gap: How do climate and agricultural management explain the 'yield gap' of croplands around the world? *Global Ecology and Biogeography* **19,** 769–782. ISSN: 1466822X (2010).

34. Seufert, V. & Ramankutty, N. Many shades of grayThe context-dependent performance of organic agriculture. *Science Advances* **3,** e1602638. ISSN: 2375-2548 (2017).

35. Herrero, M. *et al.* Biomass use, production, feed efficiencies, and greenhouse gas emissions from global livestock systems. *Proceedings of the National Academy of Sciences of the United States of America* **110,** 20888–93. ISSN: 1091-6490 (2013).

36. Lohmann Tierzucht. *Layers Lohmann LSL-Lite Management Guide* tech. rep. (2016), 1–48.

37. National Research Council. *Nutrient Requirements of Beef Cattle: Seventh Revised Edition: Update 2000.* (ed DC: The National Academies Press.) doi:https://doi.org/10.17226/9791 (2000).

38. Langston University. *Agricultural Research and Extension Programs* http://www2.luresext.edu/ (2017).

39. Lendin, I. *Energy and protein requirements of small ruminants* tech. rep. (Swedish University of Agricultural Sciences, Uppsala, 2004), 35–40.

40. Carter, N., Dewey, C., Mutua, F., de Lange, C. & Grace, D. Average daily gain of local pigs on rural and peri-urban smallholder farms in two districts of Western Kenya. *Tropical Animal Health and Production* **45,** 1533–1538. ISSN: 00494747 (2013).

41. Tybirk, P., Sloth, N. & Jorgensen, L. *Nutrient requirement standards* tech. rep. (Pig research centre, 2013), 1–9. doi:10.1016/B978-0-12-394807-6.00160-X. http://dx.doi.org/10.1016/B978-0-12-394807-6.00160-X.

42. Cobb-Vantress, I. *Broiler Performance & Nutrition Supplement* tech. rep. (2015).

43. Ross Aviagen. *Ross Broiler Management Handbook* tech. rep. (2014), 1–132.

44. Fetzel, T. *et al.* Quantification of uncertainties in global grazing systems assessment. *Global Biogeochemical Cycles* **31,** 1089–1102. ISSN: 19449224 (2017).

45. Zhao, M., Running, S., Heinsch, F. A. & Nemani, R. in *Land Remote Sensing and Global Environmental Change* 635–660 (2011). ISBN: 978-1-4419-6748-0. doi:10.1007/978-1-4419-6749-7. http://link.springer.com/10.1007/978-1-4419-6749-7.

46. Ramankutty, N., Evan, A. T., Monfreda, C. & Foley, J. A. Farming the planet: 1. Geographic distribution of global agricultural lands in the year 2000. *Global Biogeochemical Cycles* **22,** 1–19. ISSN: 08866236 (2008).

47. Haberl, H. *et al.* Quantifying and mapping the human appropriation of net primary production in earth's terrestrial ecosystems. *Proceedings of the National Academy of Sciences* **104,** 12942–12947. ISSN: 0027-8424 (2007).

48. Fritzemeier, J & Gelius-dietrich, G. Package clpAPI '. *CRAN Repository* (2016).

49. Hornik, K., Meyer, D., Schwendinger, F., Theussl, S. & Wuertz, D. Package ROI'. *CRAN Repository* (2018).

50. Thieurmel, B. Package ROI.plugin.clp'. *CRAN Repository* (2017).

51. Kurt, A., Meyer, D. & Buchta, C. Package slam '. *CRAN Repository* (2019).