Ruminant Feed Balance Modelling in R

January 2025


Table of contents

1 Introduction

This course is designed to guide learners on ruminant feed balance modelling in R. The course gives an overview of the concept of feed balance modelling, and introduces the main modelling steps based by Fraval et al. (2024). Learners are taught how to gather spatial and non-spatial data, pre-process it, and ultimately assess feed balances. The methodology has be implemented in Ethiopia, Burkina Faso and Nigeria. The codes and data in these course are for Nigeria. The course emphasizes practical learning, with learners working on hands-on sessions. Detailed explanations are provided, helping learners follow along with the material and providing a valuable resource for future reference.

2 Background

Feed balances are used to evaluate the adequacy of available livestock feed resources in meeting the dietary requirements of livestock (Mottet and Assouma, 2024). Feed resources typically include natural grasses, browse, crop by-products (e.g., crop residues), and agro-industrial by-products. Feed balance assessments are conducted at specific geographical scales and over defined time periods. In this course, we focus on estimating feed (energy) balance for ruminant livestock at the national scale, while ensuring relevance at sub-national levels and across different time frames.

2.1 Objectives

The objectives of this course are to:-

- Teach participants where to collect spatial and non-spatial data required for livestock feed balance assessment for Nigeria.
- Teach participants how to import, edit, and export spatial and non-spatial data in preparation for feed balance modeling.
- Guide participants in evaluating livestock feed supply and determining livestock nutritional requirements.
- Equip participants with the skills to assess feed balances, allowing them to evaluate if livestock feed supply meets demand.

2.2 Learning Outcomes

By the end of the course, learners will be able to:-

- Locate and collect essential spatial and non-spatial data for livestock feed balance assessment for Nigeria
- Import, edit, and export spatial and non-spatial data for use in livestock feed balance assessments.
- Assess livestock feed supply and determine livestock nutritional requirements.
- Evaluate feed supply against livestock requirements to assess feed balances.

2.3 Approach

The approach involves estimating (i) feed availability using freely accessible geospatial data and (ii) ruminant livestock feed requirements, both assessed over time. Figure 1 provides an overview of the feed balance model design.

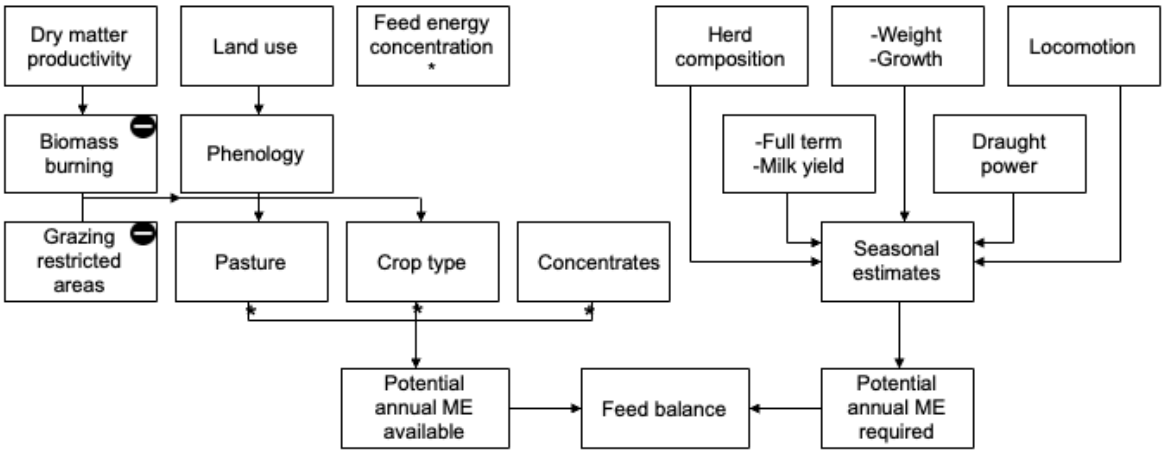


Figure 1: Feed balance model for ruminant livestock. A minus (−) indicates excluded biomass, while an asterisk (*) denotes inclusion of feed concentration values in subsequent calculations. ME = Metabolizable Energy.

The course is organized into three workflows: `1Data-download` for downloading data, `2Feed-geoprocessing` for estimating feed availability and `3Balance-estimates` for estimating feed balances.

3 Setting up R environment

Learners will need to install the following free and open source software:-

- R programming language (4.4+) https://cran.r-project.org.
- R Studio Desktop (3.6+) https://rstudio.com/products/rstudio/download/#download.

Create a new folder within your work space and name it `feed-balance-modelling`. Using R, we assign the path to the folder the variable name `root`.

For Linux/Unix systems

```
# Linux systems
root <- "/home/feed-balance-modelling"
```

For Windows systems

```
# Windows systems
root <- "C:/Documents/feed-balance-modelling"
```

Next, we set the country of interest.

```
country <- "Nigeria"
```

4 Data collation

This tutorial aims to guide learners through the key spatial and non-spatial data sources required for assessing feed balance. Learners will be introduced to a variety of datasets, and by the end of the tutorial, they will understand the necessary data requirements and know how to gather these datasets for feed balance modeling.

The tutorial leverages freely available spatial and non-spatial data to perform feed balance modeling. Recent advancements in R packages have enabled streamlined access to spatial data, including administrative boundaries, climatic variables, and environmental datasets. The required data is listed in Table 1.

Table 1: List of datasets required for feed balance modelling

Dataset	Description	Availability	Spatial resolution	Year	Source
Administrative boundaries	Administrative boundaries for target country	Global	NA	2023	GADM (2024)
Aggregation zones	Areas with similar livestock management practices	Regional	NA	2018	FMARD (2023)
Land use	Fractional cover of different land uses	Global	100 meter	2019	Buchhorn et al. (2020)
Above ground dry matter productivity	Vegetation's overall growth rate	Global	300 meter	2020-2023	Copernicus (2024)
Crop type and area	Location, extent, and patterns of feedable crops	Global	10 kilometer	2020	IFPRI (2024)
Phenology	Seasonal pattern of variation in vegetated land surfaces observed from remote sensing	Global	500 meter	2020-2023	Gray et al. (2024)
Burned areas	Burnt scars	Global	300 meter	2020-2023	Padilla et al. (2024)
Protected areas	Marine and terrestrial protected areas	Global	NA	2024	UNEP-WCMC and IUCN (2023)
Tree cover	Forest and non-forest tree cover	Global, Regional	100 meter	2019	Reiner et al. (2023)
Crop harvest index	Ratio of harvested product dry weight to total above-ground biomass dry weight at plant maturity	Local	NA	2024	ILRI (2022)
Feed parameters	Nutritional quality of feed items/type e.g., dry matter digestibility, crude protein	Local	NA	2024	ILRI (2020)
Livestock population	Type and number of livestock	Global	10 kilometer	2020	Gilbert et al. (2018)
Livestock parameters	Animal characteristics e.g., live weight, age	Local	NA	2024	Experts

We provide brief reproducible examples illustrating how to download and prepare such data for feed balance modeling.

4.1 Administrative boundaries

We create a new folder under `feed-balance-modelling`, and name it `AdminBound` and assign it the variable name `outdir`. We can download administrative boundaries of world countries with the `geodata` R package. Here we use `geodata` to download the administrative boundaries for levels `0`, `1` and `2` for Nigeria from `Database of Global Administrative Areas` (GADM, 2024), and store the data in `AdminBound` folder.

```
library(geodata)
library(sf)

country <- "Nigeria"

outdir <- paste0(root, "/src/1Data-download/SpatialData/inputs/AdminBound/",
 ↪  country)
dir.create(outdir, F, T)

admin_levels <- c("0", "1", "2")
for (admin_level in admin_levels) {
    aoi <- geodata::gadm(country = "NGA", level = admin_level, path =
 ↪  paste0(outdir),
        version = "latest") %>%
        sf::st_as_sf()
    write_sf(aoi, paste0(outdir, "/gadm40_NGA_", admin_level, ".shp"), append
     ↪  = FALSE)
    write_sf(aoi, paste0(outdir, "/aoi", admin_level, ".shp"), append =
     ↪  FALSE)
}
```

4.2 Aggregation zones

We use ecological and feed distribution zones as defined by FMARD (2024), the most recent version can be downloaded at https://drive.google.com/file/d/10HsGspftDNgq-fjAjeUwB8QsmHZWUJyT/view?usp=sharing.

```
library(googledrive)
```

```
outdir <- paste0(root,
 ↪  "/src/1Data-download/SpatialData/inputs/AggregationZones")
dir.create(outdir, F, T)

drive_deauth()
drive_user()

public_file <- drive_get(as_id("10HsGspftDNgq-fjAjeUwB8QsmHZWUJyT"))
drive_download(public_file, path = paste0(outdir,
 ↪  "/Ecological_and_Feed_Distribution.zip"),
    overwrite = TRUE)

unzip(zipfile = paste0(outdir, "/Ecological_and_Feed_Distribution.zip"),
 ↪  exdir = paste0(outdir,
    "/"))
```

4.3 Feed parameters

Several feed parameters are needed for model parameterization. In this course, we estimate the metabolizable energy (ME) concentration of feeds using species-specific estimates derived from literature reviews and databases such as the Sub-Saharan Africa Feeds composition Database (Link: https://feedsdatabase.ilri.org. Feed quality parameters include metabolizable Energy (ME), neutral detergent fibre (NDF), in vitro organic matter digestibility (IVOMD), and crude protein (CP). To facilitate model parameterization, a comprehensive feed parameter file has been prepared and is available for download at: https://drive.google.com/drive/folders/1SpB1p9i4MGU1gMahF4M3Uc-HZr8FGoqd

```
outdir <- paste0(root, "/src/1Data-download/Tables/inputs/", country,
 ↪  "/CropParams")
dir.create(outdir, F, T)

drive_deauth()
drive_user()

# folder link to id
public_folder <-
 ↪  "https://drive.google.com/drive/folders/1SpB1p9i4MGU1gMahF4M3Uc-HZr8FGoqd"
folder_id <- drive_get(as_id(public_folder))

# find files in folder
public_files <- drive_ls(folder_id)
```

```
for (i in 1:nrow(public_files)) {
    public_file <- public_files[i, ]
    file_name <- public_file$name
    drive_download(public_file, path = paste0(outdir, "/", file_name),
     ↪  overwrite = TRUE)
}
```

Other useful websites for feed parameters include:

- Feedipedia https://www.feedipedia.org
- Tropical forages http://www.tropicalforages.info

4.4 Livestock parameters

Several livestock parameters are needed for model parameterization. In this course, we utilize the Gridded Livestock of the World (GLW) database to spatially disaggregate herds and flocks into species and categories including sheep, goats (adult and young), cattle (bulls, steers, cows, heifers, and calves), horses, and donkeys. We compile essential livestock parameters to characterize herd dynamics and productivity. These parameters include: live weight, age, daily weight gain, fertility rate, lactation length, daily milk yield, annual milk yield, daily walking distance, and proportion of population used for work. To facilitate model parameterization, a comprehensive feed parameter file has been prepared and is available for download at: https://drive.google.com/drive/folders/1-3N_kmMgcHr_tayylSxlMJAr-2PBGFXd

```
outdir <- paste0(root, "/src/1Data-download/Tables/inputs/", country,
 ↪  "/LivestockParams")
dir.create(outdir, F, T)

drive_deauth()
drive_user()

# folder link to id
public_folder =
 ↪  "https://drive.google.com/drive/folders/1-3N_kmMgcHr_tayylSxlMJAr-2PBGFXd"
folder_id = drive_get(as_id(public_folder))

# find files in folder
public_files = drive_ls(folder_id)

for (i in 1:nrow(public_files)) {
```

```
    public_file <- public_files[i, ]
    file_name <- public_file$name
    drive_download(public_file, path = paste0(outdir, "/", file_name),
     ↪  overwrite = TRUE)
}
```

Extended workflows and detailed scripts for collating other required datasets are available at https://github.com/ilri/ruminant-feed-balance/tree/main/src/1Data-download

5 Feed geoprocessing

This tutorial aims to guide learners through the geo-processiong techniques used for making the datasets we collated in the previous session ready for feed balance modelling. The workflow is comprised of R scripts that should be executed in numeric-alphabetic sequence i.e., 0a, 0b, 1, 2a, 2b etc.

Here we provide brief reproducible examples illustrating execute some of the R scripts.

5.1 Cropping rasters

The spatial datasets downloaded in the previous session included both vector and raster data. For example `dry matter productivity` herein referred to as `DMP` layers have a global extent, `aggregation zones` are at a national extent, the `tree cover` data is at a regional extent, and the `protected areas` data is provided as a vector layer.

To focus our analysis on the area of interest, we will crop the layers from their global or regional extents to a smaller extent. We will use the boundary of Nigeria as the cropping extent to refine the datasets for the analysis.

5.1.1 Dry Matter Productivity

We begin by processing the `Dry Matter Productivity` layers. We create an outputs folder `outdir` to store the clipped files. We then read in the Nigeria boundary layer `aoi` to define the area of interest. Next, we list all `NetCDF` files downloaded in the previous session. We iterate through the files, cropping each to the specified area of interest, and save the resulting clipped files in the `outdir` folder.

```r
# Load required packages
library(dplyr)
library(raster)
library(rgdal)
library(sf)

# output folder
outdir <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country, "/Feed_DrySeason/DMP")
dir.create(outdir, F, T)

# read AOI
aoi <- read_sf(paste0(root,
 ↪  "/src/1Data-download/SpatialData/inputs/AdminBound/",
    country, "/aoi0.shp"))

nc_files <- list.files("/src/1Data-download/SpatialData/inputs/Feed/DMP",
 ↪  pattern = ".nc$",
    full.names = TRUE, recursive = TRUE)

for (nc_file in nc_files) {

    nc_name <- gsub(".{3}$", "", basename(nc_file))

    iDMP <- raster::raster(nc_file, varname = "DMP", ncdf = TRUE)
    iDMP <- crop(iDMP, extent(aoi))
    iDMP <- mask(iDMP, aoi)

    # save as GeoTIFF
    raster::writeRaster(iDMP, filename = paste0(outdir, "/", nc_name,
 ↪  ".tif"), overwrite = TRUE)

}
```

5.1.2 Protected areas

As mentioned earlier, the `protected areas` layer is provided as a vector layer and will be converted into a raster format. A reference layer from the `DMP` layer collection is used for clipping and resampling during this process.

```
# Libraries
library(terra)

indir <- paste0(root,
 ↪  "/src/1Data-download/SpatialData/inputs/ProtectedAreas")
outdir <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country, "/ProtectedAreas")
dir.create(outdir, F, T)

# load livelihoods vectors
wdpaNGA <- vect(paste0(indir, "/WDPA_WDOECM_Oct2024_Public_NGA.shp"))

# reference raster? r <- rast(ext(wdpaNGA), resolution = 0.00297619, crs =
# crs(wdpaNGA))
dmpTemp <- rast(paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country,

     ↪  "/Feed_DrySeason/DMP/c_gls_DMP300-RT6_202301100000_GLOBE_OLCI_V1.1.2.tif"))

# rasterize the SpatVector
wdpaNGA <- rasterize(wdpaNGA, dmpTemp, field = "STATUS_YR")

# Write output
writeRaster(wdpaNGA, paste0(outdir, "/WDPAGlobal.tif"), overwrite = TRUE)
```

5.1.3 Crop digestible fraction

We derive `crop digestible` fraction layers by calculating feedable versus non-feedable crops on using the `crop type and distribution` layers.

```
# Load libraries
library(terra)
library(dplyr)
library(readr)

# Runs with 16gb ram and 40+gb hdd space
terraOptions(tempdir = "/home/scratch/AUTemp")
terraOptions(memfrac = 0.5)
terraOptions(todisk = TRUE)

pathLU <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country, "/Feed_DrySeason/LandUse")
```

```
filesLU <- list.files(path = pathLU, pattern = ".tif$", full.names = T)

pathSPAM <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country,
    "/SPAM2020")
pathSPAMInter <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country,
    "/SPAM2020/intermediate")
dir.create(pathSPAMInter, F, T)

# end of file name should be physical area_cropname_a
filesSPAM <- list.files(path = pathSPAM, pattern = "_a.tif$", full.names = T)

# stSPAM <- stack(filesSPAM)
stSPAM <- rast(filesSPAM)

### Calculate non-feed crops proportion from SPAM model.
tmpNonFeed <- read_csv(paste0(root,
 ↪  "/src/1Data-download/Tables/inputs/CropParams/crop_harvest index.csv"))
 ↪  %>%
    filter(Excluded != "0") %>%
    pull(codeSPAM) %>%
    unique()
tmpNonFeedIndex <- grep(pattern = paste(tmpNonFeed, collapse = "|"),
 ↪  names(stSPAM))
iSPAMtotalArea <- app(stSPAM, fun = sum, na.rm = TRUE)
iSPAMnonFeedArea <- app(stSPAM[[tmpNonFeedIndex]], fun = sum, na.rm = TRUE)

iSPAMnonFeedFrac <- (iSPAMnonFeedArea/iSPAMtotalArea)
iSPAMnonFeedFrac <- classify(iSPAMnonFeedFrac, cbind(NA, NA, 0), right =
 ↪  FALSE)  # Replace missing values with 0
iSPAMAnimalDigestCropFrac <- 1 - iSPAMnonFeedFrac

iSPAMAnimalDigestCropFrac <- writeRaster(iSPAMAnimalDigestCropFrac,
 ↪  paste0(pathSPAMInter,
    "/animal_digest_frac.tif"), overwrite = T)
```

5.1.4 Crop residue fraction

We derive `crop residue` fraction layers by combining data on `crop type and distribution` with `harvest index` obtained from published sources.

```
# Load libraries
library(dplyr)
library(raster)
library(rgdal)
library(readr)

# Runs with 16gb ram and 40+gb hdd space
rasterOptions(tmpdir = "/home/scratch/AUTemp")
rasterOptions(maxmemory = 1e+60)
rasterOptions(todisk = TRUE)

# setwd('/exports/eddie/scratch/sfraval/feed-surfaces/')
cropHI <- read_csv(paste0(root,
 ↪  "/src/1Data-download/Tables/inputs/CropParams/crop_harvest index.csv"))

pathSPAM <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country,
    "/SPAM2020")
pathSPAMInter <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country,
    "/SPAM2020/intermediate")

# end of file name should be physical area_cropname_a
filesSPAM <- list.files(path = pathSPAM, pattern = "_a.tif$", full.names = T)
stSPAM <- stack(filesSPAM)
gc()

crops <- sub(".*_a_(.*?)_a\\.tif$", "\\1", filesSPAM)

tmpCropIndex <- grep(pattern = paste(crops, collapse = "|"), names(stSPAM))
iSPAMcropArea <- calc(stack(stSPAM[[tmpCropIndex]]), fun = sum, na.rm = TRUE)

print("past 1")

stHI <- stack()
stSPAMcropProp <- stack()
gc()
for (i in 1:length(crops)) {
    tmpCropIndex <- grep(pattern = paste(crops[i], collapse = "|"),
 ↪  names(stSPAM))
    iSPAMtmpArea <- overlay(stSPAM[[tmpCropIndex]], fun = sum, na.rm = TRUE)
    icrop <- stSPAM[[tmpCropIndex]]
```

```
    icrop[icrop > 0] <- (1 - cropHI$harvest_index[cropHI$codeSPAM ==
 ↪  crops[i]])
    stHI <- stack(stHI, icrop)

    stSPAMcropProp <- stack(stSPAMcropProp, overlay(iSPAMtmpArea,
 ↪  iSPAMcropArea,
        fun = function(x, y) {
            (x/y)
        }))

    print(paste("Loop", i))
}
gc()

iSPAMcropResFrac <- weighted.mean(stHI, stSPAMcropProp, na.rm = T)
iSPAMcropResFrac <- reclassify(iSPAMcropResFrac, cbind(NA, NA, 0.8), right =
 ↪  FALSE)  #Assume that 80% is available for animals

print("past mean")

writeRaster(iSPAMcropResFrac, paste0(pathSPAMInter, "/crop_res_frac.tif"),
 ↪  overwrite = T)
```

We calculate the proportion of feed items i.e., cereals, roots, legumes, and oil crops using the `crop area and distribution` data. We write the new layers in the `SPAM2020` folder.

```
# Load libraries
library(terra)
library(readr)

terraOptions(tempdir = "/home/scratch/AUTemp")
terraOptions(memfrac = 0.5)
terraOptions(todisk = TRUE)

spamPath <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
 ↪  country,
    "/SPAM2020")

cropLookup <- read_csv(paste0(root,
 ↪  "/src/1Data-download/Tables/inputs/CropParams/Crop classification_feed
 ↪  basket.csv"))
filesSPAM <- list.files(path = spamPath, pattern = "_a.tif$", full.names = T)
```

```
stCrops <- rast(filesSPAM)

# Index major feed types
indexCere <- grep(pattern =
 ↪  paste(cropLookup$codeSPAM[cropLookup$codeBasket_grouped_NGA ==
    "cere"], collapse = "|"), sub(".*_a_(.*?)_a\\.tif$", "\\1", filesSPAM))
indexRoots <- grep(pattern =
 ↪  paste(cropLookup$codeSPAM[cropLookup$codeBasket_grouped_NGA ==
    "roots"], collapse = "|"), sub(".*_a_(.*?)_a\\.tif$", "\\1", filesSPAM))
indexLeg <- grep(pattern =
 ↪  paste(cropLookup$codeSPAM[cropLookup$codeBasket_grouped_NGA ==
    "leg"], collapse = "|"), sub(".*_a_(.*?)_a\\.tif$", "\\1", filesSPAM))
indexOilc <- grep(pattern =
 ↪  paste(cropLookup$codeSPAM[cropLookup$codeBasket_grouped_NGA ==
    "oilc"], collapse = "|"), sub(".*_a_(.*?)_a\\.tif$", "\\1", filesSPAM))

# Extraction area for major feed categories
areaTotal <- app(stCrops, fun = sum, na.rm = T)
areaCere <- app(stCrops[[indexCere]], fun = sum, na.rm = T)
areaRoots <- app(stCrops[[indexRoots]], fun = sum, na.rm = T)
areaLeg <- app(stCrops[[indexLeg]], fun = sum, na.rm = T)
areaOilc <- app(stCrops[[indexOilc]], fun = sum, na.rm = T)

# Calculate proportions
propCere <- areaCere/areaTotal
propRoots <- areaRoots/areaTotal
propLeg <- areaLeg/areaTotal
propOilc <- areaOilc/areaTotal

# Write outputs
writeRaster(propCere, paste0(spamPath, "/propCereSPAM.tif"), overwrite =
 ↪  TRUE)
writeRaster(propRoots, paste0(spamPath, "/propRootsSPAM.tif"), overwrite =
 ↪  TRUE)
writeRaster(propLeg, paste0(spamPath, "/propLegSPAM.tif"), overwrite = TRUE)
writeRaster(propOilc, paste0(spamPath, "/propOilcSPAM.tif"), overwrite =
 ↪  TRUE)
```

Additional workflows and detailed scripts for preparing other files are available at:

- Landuse https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/1bPrepareLanduse_clip.R

- Tree cover https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/1cPrepareTreeCover_clip.R
- Digital Earth Africa land use https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/1dPrepareDEALanduse_clip.R
- Phenology https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/2aPreparePhenologyModis.R
- Burned area https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/3aBurnedDaysclip.R
- SPAM https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/4aSPAMclip.R
- Livestock population https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/6prepareLivestockPopulation.R
- Livestock production systems https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/7prepareLivestockSystems.R

5.1.5 Seasonal DM availability

We calculate the number of cropping days in a year by analyzing phenology data (e.g., green-up and senescence dates), over a time series `2020-2013`. We then define `wet` and `dry` periods. Next, we combine `crop residue`, `browse`, `natural grass` fractions, and `DMP` data to estimate feed availability for `dry` and `wet` season over the time series.

```
yearOffset <- (0 * 365)  # Base year = 2020

# Load libraries

library(dplyr)
library(raster)
library(rgdal)

rasterOptions(tmpdir = "/home/scratch/AUTemp")
rasterOptions(maxmemory = 5e+20)  # 6e+10 ~51GB allowed
rasterOptions(todisk = TRUE)

# read AOI
aoi <- readOGR(paste0(root,
 ↪  "/src/1Data-download/SpatialData/inputs/AdminBound/",
    country, "/aoi0.shp"))

yearList <- c("2020", "2021", "2022", "2023")

lapply(yearList, function(year) {
```

```
  cropOutdir <- paste0(root,
↪ "/src/2Feed-geoprocessing/SpatialData/inputs/", country,
      "/Cropping_days")
  dir.create(cropOutdir, F, T)
  FeedQuantityOutdir <- paste0(root,
↪ "/src/2Feed-geoprocessing/SpatialData/inputs/",
      country, "/Feed_DrySeason/Feed_quantity/", year)
  dir.create(FeedQuantityOutdir, F, T)

  pathPhen <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
↪ country,
      "/Feed_DrySeason/PhenologyModis/", year, "/outputTif")
  filesPhenology <- list.files(path = pathPhen, pattern = ".tif$",
↪ full.names = T)

  pathDMP <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
↪ country,
      "/Feed_DrySeason/DMP")
  filesDMP <- list.files(path = pathDMP, pattern = paste0("RT6_", year,
↪ ".*\\.tif$"),
      full.names = TRUE)
  stDMP <- stack(filesDMP)

  datesDMP <- sub(".*RT6_(.{8}).*", "\\1", filesDMP)
  datesDMP <- as.Date(datesDMP, "%Y%m%d")
  datesDMPdiff <- as.numeric(datesDMP - as.Date("1970/01/01"))  #convert to
↪ same date format as Modis phenology
  pathLU <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
↪ country,
      "/Feed_DrySeason/LandUse")
  filesLU <- list.files(path = pathLU, pattern = "300.tif$", full.names =
↪ T)
  pathSPAM <- paste0(root, "/src/2Feed-geoprocessing/SpatialData/inputs/",
↪ country,
      "/SPAM2020")

  # end of file name should be physical area_cropname_a
  filesSPAM <- list.files(path = pathSPAM, pattern = "_a.tif$", full.names
↪ = T)
  iSPAMAnimalDigestCropFrac <- raster(paste0(root,
↪ "/src/2Feed-geoprocessing/SpatialData/inputs/",
```

```
    country, "/SPAM2020/animal_digest_frac.tif"))

rProtectedAreas <- stack(paste0(root,
↪ "/src/2Feed-geoprocessing/SpatialData/inputs/",
    country, "/ProtectedAreas/WDPAGlobal.tif"))
rNonProtectedAreas <- calc(rProtectedAreas, fun = function(x) {
    ifelse(x == 0, 1, 0)
})
rm(rProtectedAreas)

print("past protected")

stLU <- stack(filesLU)
LUcrops300DEA <- raster(paste0(pathLU, "/LUcrops300DEA.tif"))

stPhen <- stack(raster(grep("phenoGreenup1.tif", filesPhenology, value =
↪ TRUE)),
    raster(grep("phenoSenescence1.tif", filesPhenology, value = TRUE)),
     ↪  raster(grep("phenoGreenup2.tif",
        filesPhenology, value = TRUE)),
↪ raster(grep("phenoSenescence2.tif", filesPhenology,
        value = TRUE)))
gc()

## Crop land use to test area
LUcrops300DEA <- extend(LUcrops300DEA, extent(stDMP[[1]]))
LUcrops300DEA <- crop(LUcrops300DEA, extent(stDMP[[1]]))
LUcrops300DEA <- mask(LUcrops300DEA, aoi)
stLU <- extend(stLU, extent(stDMP[[1]]))
stLU <- crop(stLU, extent(stDMP[[1]]))
stLU <- mask(stLU, aoi)

## Revise grass and shrub area
diffCrop <- LUcrops300DEA - stLU$LUcrops300
stLU$LUgrassShrub300 <- sum(stLU$LUgrass300, stLU$LUshrub300, na.rm = T)
stLU$LUgrassShrub300 <- stLU$LUgrassShrub300 - LUcrops300DEA

stLU$LUcrops300 <- LUcrops300DEA

stPhen$phenoGreenup2 <- calc(stPhen$phenoGreenup2, fun = function(x) {
    ifelse(x > max(datesDMPdiff) + 30, NA, x)
})
```

```
    stPhen$phenoSenescence2 <- calc(stPhen$phenoSenescence2, fun =
↪   function(x) {
        ifelse(x > max(datesDMPdiff) + 30, NA, x)
    })
    stPhen$phenoGreenup1 <- calc(stPhen$phenoGreenup1, fun = function(x) {
        ifelse(x < min(datesDMPdiff) - 30, NA, x)
    })
    stPhen$phenoSenescence1 <- calc(stPhen$phenoSenescence1, fun =
↪   function(x) {
        ifelse(x < min(datesDMPdiff) - 30, NA, x)
    })
    gc()

    growing2 <- (stPhen$phenoSenescence2 - stPhen$phenoGreenup2)
    growing2 <- reclassify(growing2, c(365, Inf, 0))
    growingDays <- sum((stPhen$phenoSenescence1 - stPhen$phenoGreenup1),
↪   growing2,
        na.rm = T)
    growingDays <- reclassify(growingDays, c(300, Inf, 300))
    gc()

    writeRaster(growingDays, paste0(cropOutdir, "/croppingDays_", year,
     ↪  ".tif"),
        overwrite = T)

    print("past 0")

    names(stDMP) <- paste0("d", datesDMPdiff)

    stLU$LUtree300 <- reclassify(stLU$LUtree300, c(-Inf, 0, 0, 200, Inf, 0))
    stLU$LUtree300[is.na(stLU$LUtree300)] <- 0

    print("past 1")

    ##### Estimate total DMP per ha
    grassFracDry <- 1  # 0.33 #max 0.55
    grassFracWet <- 1  # 0.55 #max 0.55
    browseShrubFrac <- 1  #0.38 #max 0.38
    browseForestFrac <- 1
    iResidueUtil <- 1  #max 0.6
    iSPAMHarvestResidueFrac <- raster(paste0(root,
↪   "/src/2Feed-geoprocessing/SpatialData/inputs/",
```

```
    country, "/SPAM2020/crop_res_frac.tif"))
gc()

print("past overlay 1")

residueFrac <- iSPAMHarvestResidueFrac

shrubFrac <- raster(paste0(root,
"/src/2Feed-geoprocessing/SpatialData/inputs/",
    country, "/TreeCover/treecover300m.tif"))/100

gc()

funGrowingGrassWet <- function(dmp, crops, grassShrub, forest, shrubFrac,
greenup,
    senesence, greenup2, senesence2, nonprotected) {
    ifelse((greenup <= datesDMPdiff[i] & senesence >= datesDMPdiff[i]) |
     (greenup2 <=
        datesDMPdiff[i] & senesence2 >= datesDMPdiff[i]), (dmp * 9 *
grassShrub *
        grassFracWet * (1 - shrubFrac)) + (dmp * 9 * forest *
grassFracWet *
        (1 - shrubFrac) * nonprotected), NA)
}  #@feedFrac is the proportion of crops grown that have feedable
residues - i.e. excluding coffee, tea, ect.
funGrowingGrassDry <- function(dmp, crops, grassShrub, forest, shrubFrac,
greenup,
    senesence, greenup2, senesence2, nonprotected) {
    ifelse((greenup > datesDMPdiff[i]) | (senesence < datesDMPdiff[i] &
     senesence +
        60 > datesDMPdiff[i]) | (senesence2 < datesDMPdiff[i]), (dmp * 9
         * grassShrub *
        grassFracDry * (1 - shrubFrac)) + (dmp * 9 * forest *
grassFracDry *
        (1 - shrubFrac) * nonprotected), NA)
}  #@feedFrac is the proportion of crops grown that have feedable
residues - i.e. excluding coffee, tea, ect.
funGrowingBrowse <- function(dmp, crops, grassShrub, forest, shrubFrac,
nonprotected) {
    (dmp * 9 * grassShrub * shrubFrac * browseShrubFrac) + (dmp * 9 *
forest *
        nonprotected * shrubFrac * browseForestFrac)
```

```
    }
    funGrowingCrops <- function(dmp, crops, greenup, senesence, feedFrac,
↪   resFrac,
        greenup2, senesence2) {
        ifelse((greenup <= datesDMPdiff[i] & senesence >= datesDMPdiff[i]) |
         ↪  (greenup2 <=
            datesDMPdiff[i] & senesence2 >= datesDMPdiff[i]), (dmp * 9 *
↪   crops),
            NA)
    }  #@feedFrac is the proportion of crops grown that have feedable
↪   residues - i.e. excluding coffee, tea, ect.
    funGrowingAftermath <- function(dmp, crops, greenup, senesence, greenup2,
↪   senesence2,
        nonprotected) {
        ifelse((greenup > datesDMPdiff[i]) | (senesence < datesDMPdiff[i] &
         ↪  senesence +
            60 > datesDMPdiff[i]) | (senesence2 < datesDMPdiff[i]), (dmp * 9
             ↪  * crops),
            NA)
    }  #@feedFrac is the proportion of crops grown that have feedable
↪   residues - i.e. excluding coffee, tea, ect.

    for (i in 1:length(names(stDMP))) {

        iDMPGrassGrowing <- overlay(stDMP[[i]], stLU$LUcrops300,
↪   stLU$LUgrassShrub300,
            stLU$LUtree300, shrubFrac, stPhen$phenoGreenup1,
↪   stPhen$phenoSenescence1,
            stPhen$phenoGreenup2, stPhen$phenoSenescence2,
↪   rNonProtectedAreas, fun = funGrowingGrassWet)
        writeRaster(iDMPGrassGrowing, paste0(FeedQuantityOutdir,
         ↪  "/grassWetDMP",
            datesDMP[i], ".tif"), overwrite = TRUE)
        rm(iDMPGrassGrowing)
        gc()

        iDMPGrassDry <- overlay(stDMP[[i]], stLU$LUcrops300,
↪   stLU$LUgrassShrub300,
            stLU$LUtree300, shrubFrac, stPhen$phenoGreenup1,
↪   stPhen$phenoSenescence1,
            stPhen$phenoGreenup2, stPhen$phenoSenescence2,
↪   rNonProtectedAreas, fun = funGrowingGrassDry)
```

```
        writeRaster(iDMPGrassDry, paste0(FeedQuantityOutdir, "/grassDryDMP",
         ↪  datesDMP[i],
            ".tif"), overwrite = TRUE)
        rm(iDMPGrassDry)
        gc()

        iDMPBrowse <- overlay(stDMP[[i]], stLU$LUcrops300,
↪  stLU$LUgrassShrub300,
            stLU$LUtree300, shrubFrac, rNonProtectedAreas, fun =
↪  funGrowingBrowse)
        writeRaster(iDMPBrowse, paste0(FeedQuantityOutdir, "/browseDMP",
         ↪  datesDMP[i],
            ".tif"), overwrite = TRUE)
        rm(iDMPBrowse)
        gc()

        iDMPCropGrowing <- overlay(stDMP[[i]], stLU$LUcrops300,
↪  stPhen$phenoGreenup1,
            stPhen$phenoSenescence1, iSPAMAnimalDigestCropFrac, residueFrac,
↪  stPhen$phenoGreenup2,
            stPhen$phenoSenescence2, fun = funGrowingCrops)
        writeRaster(iDMPCropGrowing, paste0(FeedQuantityOutdir, "/cropDMP",
         ↪  datesDMP[i],
            ".tif"), overwrite = TRUE)
        rm(iDMPCropGrowing)
        gc()

        iDMPAftermath <- overlay(stDMP[[i]], stLU$LUcrops300,
↪  stPhen$phenoGreenup1,
            stPhen$phenoSenescence1, stPhen$phenoGreenup2,
↪  stPhen$phenoSenescence2,
            fun = funGrowingAftermath)
        writeRaster(iDMPAftermath, paste0(FeedQuantityOutdir,
         ↪  "/aftermathDMP", datesDMP[i],
            ".tif"), overwrite = TRUE)
        rm(iDMPAftermath)
        gc()
        print(paste("cycle", i))

    }

    gc()
```

```
    iDMPgrassWet <- stack(list.files(path = paste0(FeedQuantityOutdir),
↪   pattern = "grassWet",
        full.names = T))
    DMPgrassmeanWet <- mean(iDMPgrassWet, na.rm = T)
    writeRaster(DMPgrassmeanWet, paste0(FeedQuantityOutdir,
     ↪  "/DMPgrassWetmean_",
        year, ".tif"), overwrite = TRUE)

    iDMPgrassDry <- stack(list.files(path = paste0(FeedQuantityOutdir),
↪   pattern = "grassDry",
        full.names = T))
    DMPgrassmeanDry <- mean(iDMPgrassDry, na.rm = T)
    writeRaster(DMPgrassmeanDry, paste0(FeedQuantityOutdir,
     ↪  "/DMPgrassDrymean_",
        year, ".tif"), overwrite = TRUE)

    iDMPbrowse <- stack(list.files(path = paste0(FeedQuantityOutdir), pattern
↪   = "browse",
        full.names = T))
    DMPbrowsemean <- mean(iDMPbrowse, na.rm = T)
    writeRaster(DMPbrowsemean, paste0(FeedQuantityOutdir, "/DMPbrowsemean_",
     ↪  year,
        ".tif"), overwrite = TRUE)

    iDMPCropGrowing <- stack(list.files(path = paste0(FeedQuantityOutdir),
↪   pattern = "crop",
        full.names = T))
    DMPcropmean <- mean(iDMPCropGrowing, na.rm = T)
    writeRaster(DMPcropmean, paste0(FeedQuantityOutdir, "/DMPcropmean_",
     ↪  year, ".tif"),
        overwrite = TRUE)

    iDMPAftermath <- stack(list.files(path = paste0(FeedQuantityOutdir),
↪   pattern = "aftermath",
        full.names = T))
    DMPAftermean <- mean(iDMPAftermath, na.rm = T)
    writeRaster(DMPAftermean, paste0(FeedQuantityOutdir, "/DMPaftermean_",
     ↪  year,
        ".tif"), overwrite = TRUE)
    gc()
```

```
})
```

Finally, we copy relevant data to specific folders for use in the next section using this R script: https://github.com/ilri/ruminant-feed-balance/blob/main/src/2Feed-geoprocessing/8Copydata.R

6 Estimating feed balances

6.1 Mapping aggregation regions and zones

The ecological and feed distribution map provided by the Federal Ministry of Agriculture and Food Security (FMAFS) contain eight zones. We map them into ecological regions and zones as shown in Table 2.

Table 2: Ecological zones and regions

Ecological and Feed Distribution Zones	**Ecological Regions**	**Ecological Zones**
Sahel Savannah	Dry Savannah	(Agro)pastoral sahel
Sudan Savannah	Dry Savannah	Northern mixed
Southern Guinea Savannah	Wet Savannah	Southern mixed
Northern Guinea Savannah	Wet Savannah	Central mixed
Mountain Vegetations	Wet Savannah	Central mixed
Lowland Rainfall	Forest	Forest mixed
Fresh Water Swamp Forest	Forest	Forest mixed
Mangrove	Forest	Forest mixed

The R script for creating the new categories of ecological regions and zones is available at: https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/0_PrepareRegions.R

As shown in Figure 2, the ecological regions are classified into `Dry Savannah`, `Forest`, and `Wet Savannah`.

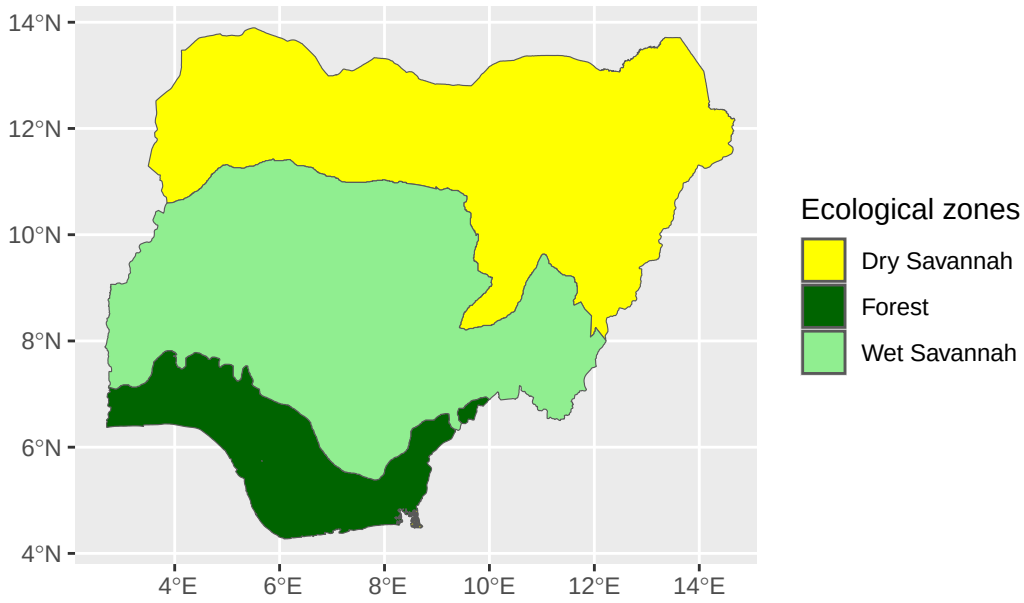


Figure 2: Ecological Regions of Nigeria

Similarly, as shown in Figure 3, the ecological regions are further classified into `(Agro)pastoral sahel`, `Northern mixed`, `Central mixed`, `Southern mixed` and `Forest mixed`.

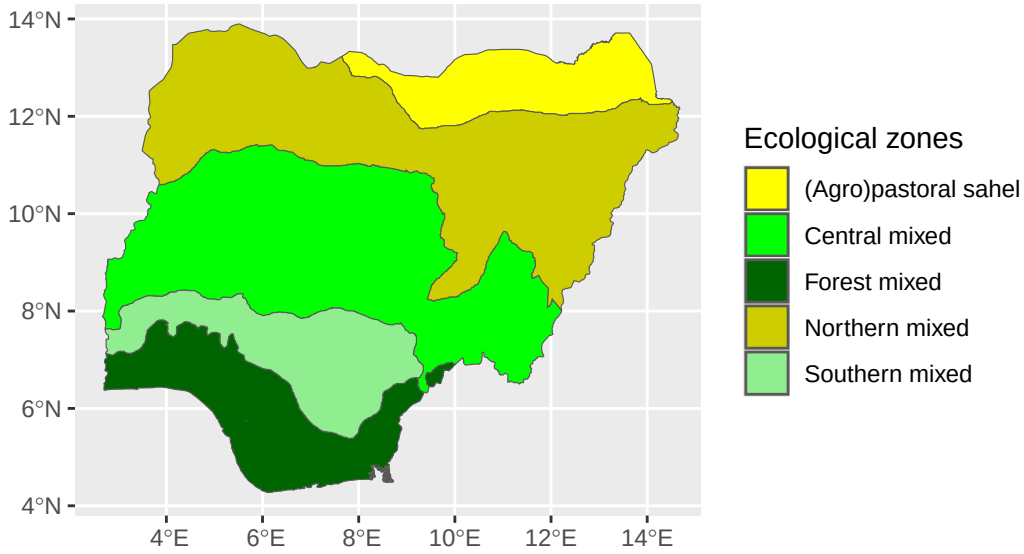


Figure 3: Ecological Zones of Nigeria

6.2 Feed energy concentration

We merge `feedQuality` dataset with the `crop_classification` dataset using a common id, and map some specific feed items to new codes. The resulting clean `feedQuality` dataset is saved as a CSV file. The R script is available at: https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/0_ProcessSSAfeedDB.R

We then calculate feed metabolizable energy (ME) by combining various datasets, including crop parameters, feed quality data, and spatial data related to crop type and distribution. We process feed quality data i.e., calculate dry matter intake (DMI), adjust for other variables like CP, NDF, and IVDMD. We generate summary statistics and group the data by crop type. NB: Missing or inconsistent data is addressed by imputing values or setting fixed values for certain crops (See Table 3).

Table 3: Feed quality for different feed items

Feed item	**ME (SD)**	**ME (Min)**	**ME (Max)**	**ME (Mean)**	**CP**	**NDF**	**IVDMD**
Banana*	0.5	7.7	9.0	8.3	5.8	42.1	51.8
Barley*	0.8	5.8	10.3	7.3	8.0	68.6	47.7
Bean*	0.6	7.0	9.8	8.7	6.6	68.1	56.1
Sugarbeet*	1.4	7.0	13.4	9.9	19.8	39.1	67.9
Chickpea*	0.6	6.8	10.8	8.2	4.7	57.5	51.6
Lentils*	0.5	7.0	10.3	8.8	9.2	51.7	56.2
Pigeon pea*	0.4	7.6	9.1	8.2	16.2	53.0	53.5
Plantain*	0.5	7.7	9.0	8.3	5.8	42.1	51.8
Soybean*	0.6	7.0	10.9	8.6	7.7	66.6	57.0
Sugarcane*	0.3	7.2	8.0	7.5	6.9	73.8	49.4
Wheat*	1.2	5.7	11.0	7.3	4.3	72.1	48.7
Cowpea*	NA	7.7	10.5	9.3	12.2	46.8	61.0
Maize*	NA	7.9	7.9	7.9	5.2	69.1	51.8
Pearl millet*	NA	6.7	8.4	7.6	5.2	68.6	50.0
Rice*	NA	5.9	8.4	6.8	5.5	65.6	49.0
Sesame*	NA	9.2	9.5	9.3	19.3	24.2	59.1
Sorghum*	NA	6.6	7.9	7.5	4.5	71.9	48.5
Sweet potato*	NA	8.9	9.6	9.3	13.3	48.4	60.6
Small millet	NA	6.7	8.4	7.6	5.2	68.6	NA
Other cereals*	NA	6.7	8.4	7.6	5.2	68.6	NA
Other legumes*	NA	7.7	10.5	9.3	5.2	68.6	NA
Cassava*	NA	4.2	4.2	4.2	NA	NA	NA
Yam*	NA	4.2	4.2	4.2	NA	NA	NA
Other roots and tubers*	NA	4.2	4.2	4.2	NA	NA	NA
Groundnut*	1.6	7.2	9.3	8.5	NA	NA	NA
Natural pasture	NA	6.2	6.8	6.5	NA	NA	NA

** Denotes crop residue.*

We calculate crop-specific harvest index (See Table 4), utilization and ME values. We then weight the values by the proportion of crop per pixel and aggregate the results to administrative level 2.

Table 4: Harvest index for feedable crops

Crop name	**Harvest index**
Banana	0.44
Barley	0.534
Bean	0.555
Sugarbeet	1
Chickpea	0.31
Lentils	0.246
Pigeon Pea	0.294
Plantain	0.44
Soybean	0.422
Sugarcane	0.19
Wheat	0.36575
Cowpea	0.335
Maize	0.3785
Pearl millet	0.188
Rice	0.4613
Sesame	0.01
Sorghum	0.458
Sweet potato	0.27
Small Millet	0.391
Other cereals	0.188
Other legumes	0.3
Cassava	0.8696
Yam	0.24102
Other roots and tubers	0.27
Groundnut	0.45

The R script for completing the above steps is available at: https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/0b_Prepare_cropME.R

We analyse feed availability across multiple years (2020–2023) by integrating spatial data related to crop type and distribution, grass and browse productivity and feed utilization. Cropping days are calculated, with thresholds applied to exclude areas with insufficient cropping duration. Dry days are derived as the inverse of cropping days (365 - cropping days). Metabolizable energy (ME) is estimated for various feed sources including crop residues, grass, and browse, incorporating region-specific productivity and feed quality parameters. Feed statistics, such as mean ME values for crops, grass, and browse, are extracted and aggregated within defined regions and zones.

The R script for completing the above steps is available at: https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/1_ProcessFeedMEregional.R, and https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/1b_ProcessFeedMEregional_min_max.R

6.3 Livestock energy requirements

6.3.1 Requirements for base year

Using Gridded Livestock of the World (GLW) (Gilbert et al., 2018), livestock energy requirements are calculated by disaggregating herds and flocks into species and categories including `sheep`, `goats` (adult and young), `cattle` (bulls, steers, cows, heifers, and calves), `horses`, and `donkeys` and calculating seasonal metabolizable energy (ME) requirements for livestock, including energy requirements for `maintenance` , `locomotion`, `lactation` and `growth`, equations (1), (2), (3), (4) and (5).

$$\text{MERM}_{i,j,k} = K \times S \times M \times \frac{(0.26 \times \text{MLW}_{k}^{0.75}) \times \exp^{(-0.008 \times A_{k})}}{(0.02 \times \text{ME}_{i,j}) + 0.5} \tag{1}$$

Where, *MERM* is the maintenance energy requirement in megajoules per day (MJ/day) of animal k, region i and season j, K is a metabolic weight coefficient (1.3, an intermediate value for Bos taurus and Bos indicus cattle breeds), S a constant for females and castrated cattle (1), M a constant for presence or absence of milk in the diet, *MLW* is the mean live weight of animal k in region i in season j, and A is the age of animal k in region i in season j and *ME* is metabolizable energy content of the diet in megajoules of metabolizable energy per kilogram of dry matter (MJ ME/kg DM) in region i and season j.

$$\text{MERT}_{i,j,k} = \text{WD}_{i,j,k} \times \text{MLW}_{i,j,k} \times 0.0026 \tag{2}$$

Where, *MERT* is the locomotion energy requirement in megajoules per day (MJ/day) of animal k, region i and season j, *WD* is distance travelled by animal k per day in kilometres in region i and season j, *MLW* is the mean live weight of animal k in region i in season j.

For adult cows, goats and sheep:

$$\text{MERL}_{i,j,k} = \frac{\text{DMY}_{i,j,k} \times \text{ECM}_{i,j,k}}{(0.02 \times \text{ME}_{i,j}) + 0.04} \tag{3}$$

Where, *MERL* is the energy requirement for lactation for animal k, in MJ/day in region i and season j, *DMY* is the daily milk production for animal k in kilogram in region i and season j, *ECM* is the energy content of milk from animal k in region i and season j and *ME* is

metabolizable energy content of the diet in megajoules of metabolizable energy per kilogram of dry matter (MJ ME/kg DM) in region i and season j.

Energy expended for weight gain or loss as per equations (4) and (5).

$$\text{MERG+}_{i,j,k} = \frac{\text{DWG}_{i,j,k} \times 0.92 \times \text{EC}_{i,j,k}}{0.043 \times \text{ME}_{i,j}} \tag{4}$$

If energy is lost:-

$$\text{MERG-}_{i,j,k} = \frac{\text{DWG}_{i,j,k} \times 0.92}{0.8} \tag{5}$$

Where, *MERG+/-* is the energy expended for live weight gain/loss for animal k, in MJ/day in region i and season j, *DWG* is the average daily weight gain or loss for animal k in kilograms in region i and season j, EC is an assumed energy content of tissue taken of 18 MJ/kg (CSIRO, 2007) and *ME* is metabolizable energy content of the diet in megajoules of metabolizable energy per kilogram of dry matter (MJ ME/kg DM) in region i and season j.

Finally, total energy requirements for the base year for each livestock category, season and region are calculated. The R script for completing the above steps is available at: https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/2_Lvst_requirements.R, and https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/2b_Lvst_requirements_min_max.R

6.3.2 Estimating feed balances

We sum livestock energy requirements for cattle, shoats (sheep/goats), and horses/donkeys and adjust the requirements by incorporating population changes from FAOSTAT data, equation (6).

$$\text{LvstME}_y = \sum \left(H_y + \left(C_y \times (1 + \Delta P_C) \right) + \left(S_y \times (1 + \Delta P_S) \right) \right) \tag{6}$$

Where, *LvstME* is the total energy requirement for all livestock in year y, H is the energy requirement for horses and donkeys in MJ in year y, C is the energy requirement for cattle in MJ in year y, S is the energy requirement for shoats in MJ in year y, ΔP_C is the annual percentage change in cattle population and ΔP_S is the annual percentage change in shoats population.

We combine feed availability data into a single dataset for annual analysis, equation (7).

$$\text{FeedME}_y = \sum_{y=2020}^{2023} \text{MEF}_y \tag{7}$$

Where, *FeedME* is the total energy from feeds for year y, *MEF* is the energy from feeds for year y.

We aggregate and extract feed energy concentration and livestock energy requirements data across three spatial zones: `country`, `region`, and `state`across the time series. The zones were defined by spatial data representing Nigeria's administrative boundaries and ecological zones. Finally we calculate feed balance for each spatial zone and year as the ratio of feed energy concentration to livestock energy requirements, equation (8).

$$\text{FeedBal}_{y,i} = \frac{\text{FeedME}_{y,i}}{\text{LvstME}_{y,i}} \tag{8}$$

Where, *FeedBal* is the feed balance for year y in region i, *FeedME* is the feed energy concentration for year y in region i and *LvstME* is the livestock energy requirement for year y in region i.

The R script for completing the above steps is available at: https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/3_ProcessTimeseriesRegionalAdeqTotals.R, and https://github.com/ilri/ruminant-feed-balance/blob/main/src/3Balance-estimates/Nigeria/3b_ProcessTimeseriesRegionalAdeqTotals_min_max.R

6.3.3 Visualising results

```
# Load libraries
library(raster)
```

```
Loading required package: sp
```

```
library(sf)
library(exactextractr)
library(rnaturalearth)
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:raster':

    intersect, select, union

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
Attaching package: 'tidyr'

The following object is masked from 'package:raster':

    extract
```

```
library(ggplot2)
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```
library(ggtext)
library(ggsci)
library(extrafont)
```

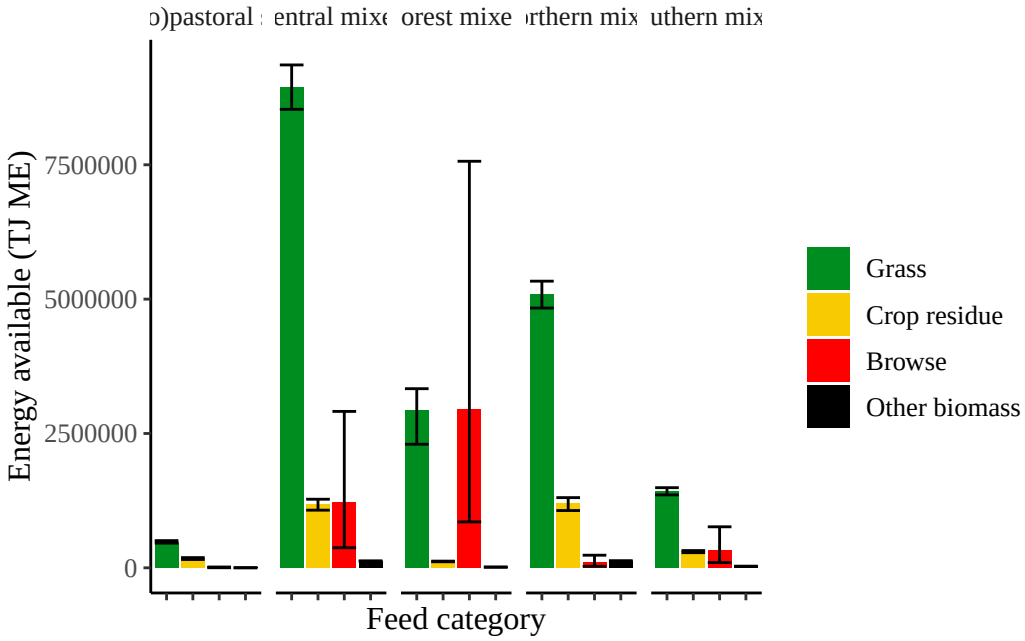
```
Registering fonts with R
```

```
library(stars)
```

```
Loading required package: abind
```

```
loadfonts(device = "all")

# root folder
root <- "."

country <- "Nigeria"

# paths
spatialDir <- paste0(root, "/src/3Balance-estimates/", country,
 ↪  "/SpatialData")
Results_dir <- paste0(root, "/src/3Balance-estimates/", country, "/Results")
plotsDir <- paste0(root, "/src/3Balance-estimates/", country, "/Plots")
dir.create(plotsDir, F, T)

## Feed timeseries breakdown
tsSum <- read.csv(paste0(Results_dir, "/disaggregated_timeseries.csv"),
 ↪  stringsAsFactors = F)

# Correction by reallocating post-harvest growth to grass
tmp <- tsSum[tsSum$zone == "(Agro)pastoral sahel", ]
tmp$grassME_mean <- tmp$grassME_mean + (tmp$afterME_mean * 0.94)
tmp$grassME_min <- tmp$grassME_min + (tmp$afterME_min * 0.94)
tmp$grassME_max <- tmp$grassME_max + (tmp$afterME_max * 0.94)
tmp$afterME_mean <- (tmp$afterME_mean * 0.06)
tmp$afterME_min <- (tmp$afterME_min * 0.06)
tmp$afterME_max <- (tmp$afterME_max * 0.06)

tsSum <- rbind(tsSum[!tsSum$zone == "(Agro)pastoral sahel", ], tmp)

tsSum_plot <- pivot_longer(dplyr::select(tsSum, zone, year, cropME_mean,
 ↪  grassME_mean,
    browseME_mean, afterME_mean), cols = -c(zone, year))
lower <- pivot_longer(dplyr::select(tsSum, zone, year, cropME_min,
 ↪  grassME_min, browseME_min,
    afterME_min), cols = -c(zone, year), values_to = "lower")
upper <- pivot_longer(dplyr::select(tsSum, zone, year, cropME_max,
 ↪  grassME_max, browseME_max,
```

```
    afterME_max), cols = -c(zone, year), values_to = "upper")
tsSum_plot <- cbind(tsSum_plot, select(lower, lower))
tsSum_plot <- cbind(tsSum_plot, select(upper, upper))

tsSum_plot$value <- tsSum_plot$value/1e+06
tsSum_plot$lower <- tsSum_plot$lower/1e+06
tsSum_plot$upper <- tsSum_plot$upper/1e+06

tsSum_plot <- transform(tsSum_plot, zone = factor(zone, levels =
 ↪  c("(Agro)pastoral sahel",
    "Central mixed", "Forest mixed", "Northern mixed", "Southern mixed")))
tsSum_plot <- transform(tsSum_plot, name = factor(name, levels =
 ↪  c("grassME_mean",
    "cropME_mean", "browseME_mean", "afterME_mean")))
ggplot(tsSum_plot[tsSum_plot$year == 2023, ], aes(name, value, fill = name))
 ↪  + geom_col(position = "identity") +
    geom_errorbar(aes(ymin = lower, ymax = upper)) + ylab("Energy available
     ↪  (TJ ME)") +
    xlab("Feed category") + scale_fill_manual(name = "", labels = c("Grass",
     ↪  "Crop residue",
    "Browse", "Other biomass"), values = c(cropME_mean = "#F8CA02",
     ↪  grassME_mean = "#008D1F",
    afterME_mean = "#000000", browseME_mean = "#FF0000")) + theme_classic() +
     ↪  theme(text = element_text(family = "serif",
    size = 12), axis.text.x = element_blank(), strip.background =
     ↪  element_blank()) +
    facet_wrap(~zone, ncol = 5)
```


o)pastoral
entral mixe
orest mixe
rthern mix
uthern mix
Energy available (TJ ME)
7500000
5000000
2500000
0
Feed category
Grass
Crop residue
Browse
Other biomass

7 References

Buchhorn, M., Lesiv, M., Tsendbazar, N.-E., Herold, M., Bertels, L., and Smets, B. (2020). Copernicus Global Land Cover Layers—Collection 2. *Remote Sensing*, 12(6):1044. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

Copernicus (2024). Dry Matter Productivity 2014-present (raster 300 m), global, 10-daily – version 1.

CSIRO (2007). *Nutrient Requirements of Domesticated Ruminants*. CSIRO publishing.

FMARD (2023). Ecological and Feed Distribution Zones of Nigeria.

FMARD (2024). Ecologial and Feed Distribution Zones.

Fraval, S., Mutua, J. Y., Amole, T., Tolera, A., Feyisa, T., Thornton, P. K., Notenbaert, A. M. O., Adesogan, A., Balehegn, M., Ayantunde, A. A., Zampaligre, N., and Duncan, A. J. (2024). Feed balances for ruminant livestock: gridded estimates for data constrained regions. *animal*, page 101199.

GADM (2024). Database of Global Administrative Boundaries (GADM).

Gilbert, M., Nicolas, G., Cinardi, G., Van Boeckel, T. P., Vanwambeke, S. O., Wint, G. R., and Robinson, T. P. (2018). Global distribution data for cattle, buffaloes, horses, sheep, goats, pigs, chickens and ducks in 2010. *Scientific Data 2018 5:1*, 5(1):1–11. Publisher: Nature Publishing Group.

Gray, J., Sulla-Menashe, D., and Friedl, M. (2024). User Guide to Collection 6 MODIS Land Cover Dynamics (MCD12Q2) Product. Technical report.

IFPRI (2024). Global Spatially-Disaggregated Crop Production Statistics Data for 2020 Version 1.0.0.

ILRI (2020). Sub-Saharan Africa feeds composition database.

ILRI (2022). Feed Assessment Tool (FEAST).

Mottet, A. and Assouma, M. H. (2024). The feed balances sheet: a tool for planning the use of resources and enhancing resilience in tropical grazing livestock. *Frontiers in Animal Science*, 5. Publisher: Frontiers.

Padilla, M., Ramo, R., Gomez-Dans, L., Sierra, D., Mota, B., Lacaze, R., and Tansey, K. (2024). Near-real time monitoring of burned area at global scale based upon semi-empirical modelling and deep learning methods.

Reiner, F., Brandt, M., Tong, X., Skole, D., Kariryaa, A., Ciais, P., Davies, A., Hiernaux, P., Chave, J., Mugabowindekwe, M., Igel, C., Oehmcke, S., Gieseke, F., Li, S., Liu, S., Saatchi, S., Boucher, P., Singh, J., Taugourdeau, S., Dendoncker, M., Song, X.-P., Mertz, O., Tucker, C. J., and Fensholt, R. (2023). More than one quarter of Africa's tree cover is found outside areas previously classified as forest. *Nature Communications*, 14(1):2258. Publisher: Nature Publishing Group.

UNEP-WCMC and IUCN (2023). Protected Planet: The World Database on Protected Areas (WDPA), [September, 2023]. Place: Cambridge, UK Publisher: UNEP-WCMC and IUCN.