MEP - Forest Biomass dynamics

Session 2 - Practical worksheet

26th October 2022

Contents

1 Introduction

You will need R and a spreadsheet program like Microsoft Excel to complete this practical session.

Please speak up if you don't have access to the relevant software.

The teal coloured text, e.g. `EXAMPLE example`, refers to R code, variable names, or file names.

> The boxes contain questions to answer. Record your thoughts and answers to each question in a notebook, on the practical worksheet, or a word document.

In this practical you will:

- Practice manipulating time series data
- Compare different modelling techniques to forecast and interpolate time series data
- Use a dynamic model of savanna tree-grass coexistence to predict woody biomass across southern Africa
- Tweak parameters in the dynamic model to understand interactions among variables and model stability
- Compare modelled results of woody biomass to other modelled biomass estimates
- Use a future climate scenario to predict tree biomass across southern Africa in 2080

2 Datasets

You can download all the files needed for this practical from the LEARN page for this session. Save each of the files to a single folder with a sensible name on your computer. This will be your project folder for the practical.

There is an R script (`example_script.R`) which contains examples of all the code you should need in this practical. If you get stuck with any of the coding, please refer to the example script. The goal for this practical is not to practice writing code. The goal is to use the code to interrogate models.

`grass.csv` contains a time series of mean grass height values, in metres, collected in a single forest plot in Tanzania, monthly from 2000-2013.

`saf.rds` contains polygons defining the area of study, very roughly the extent of southern African woodland-savanna vegetation. The polygons are separated by country, to provide a sense of scale.

`pr2021.rds` contains total annual precipitation data for the year 2021 across southern Africa. The resolution of this data is 0.1667 decimal degrees, which is 10 arc minutes. At the equator, this is about 21 km^2. These data were compiled from monthly averages in the "WorldClim" dataset (https://www.worldclim.com) (Fick and Hijmans 2017).

`pr2080.rds` contains predicted total annual precipitation data for the year 2080 across southern Africa. The resolution is the same as that of `pr2021.rds`. These predicted data come from the WorldClim downscaled CMIP6 future climate projections, under the SSP585 high emissions Shared Socio-economic Pathways emissions scenario (Meinshausen et al. 2020), using the EC-Earth3-Veg global climate model (Döscher et al. 2021).

`avit.rds` contains a biomass map of southern African woodland-savannas, from Avitabile et al. (2016). The resolution is the same as that of `pr2021.rds`. These are modelled estimates generated from various datasets in a model-data fusion framework.

3 Loading data and packages

Open a new script in R (I recommend using RStudio) and set the working directory (e.g. `setwd("./projectpath")`) to your project folder for this practical.

In this practical we will use quite a lot of packages. Load these packages using `library()`. If the packages don't load you may need to install them first using `install.packages("name_of_pkg")`:

```
library(ggplot2)  # Plotting
library(sf)  # For spatial vector data
library(raster)  # For spatial raster data
library(tsibble)  # Tidy time series objects
library(dplyr)  # General data manipulation
library(tidyr)  # Table manipulation
library(fable)  # ARIMA and ETS models
library(fabletools)  # Additional time series modelling tools
library(feasts)  # Additional time series modelling tools
library(scico)  # Decent colour palettes
library(parallel)  # Multi-threading
```

Then, load all the data into R:

```
grass <- read.csv("./grass.csv")
saf <- readRDS("./saf.rds")
pr2021 <- readRDS("./pr2021.rds")
pr2080 <- readRDS("./pr2080.rds")
avit <- readRDS("./avit.rds")
```

4 Time series analysis

The first part of the practical will focus on the `grass` dataframe. We will use this time series as a case study to practice various time series analyses.

Have a look at the data to understand its structure:

```
str(grass)
head(grass)
range(grass$month)
```

There should be two columns: `month` and `grass_height`. `month` holds the date at which the grass height measurements were collected, in "YYYY-MM-DD" format. `grass_height` contains the mean grass height recorded on the plot, in metres. The data run from January 2000 to December 2013.

As the data are collected monthly, we need to convert the `month` column to `yearmonth` format, so it plays better with other time series analysis functions:

```
grass$year_month <- yearmonth(grass$month, format = "%Y-%m-%d")

class(grass$year_month)
```

Plot the data to visualise the time series:

```
ggplot(grass, aes(x = year_month, y = grass_height)) +
  geom_path() +
  geom_point(shape = 21, fill = "blue")
```

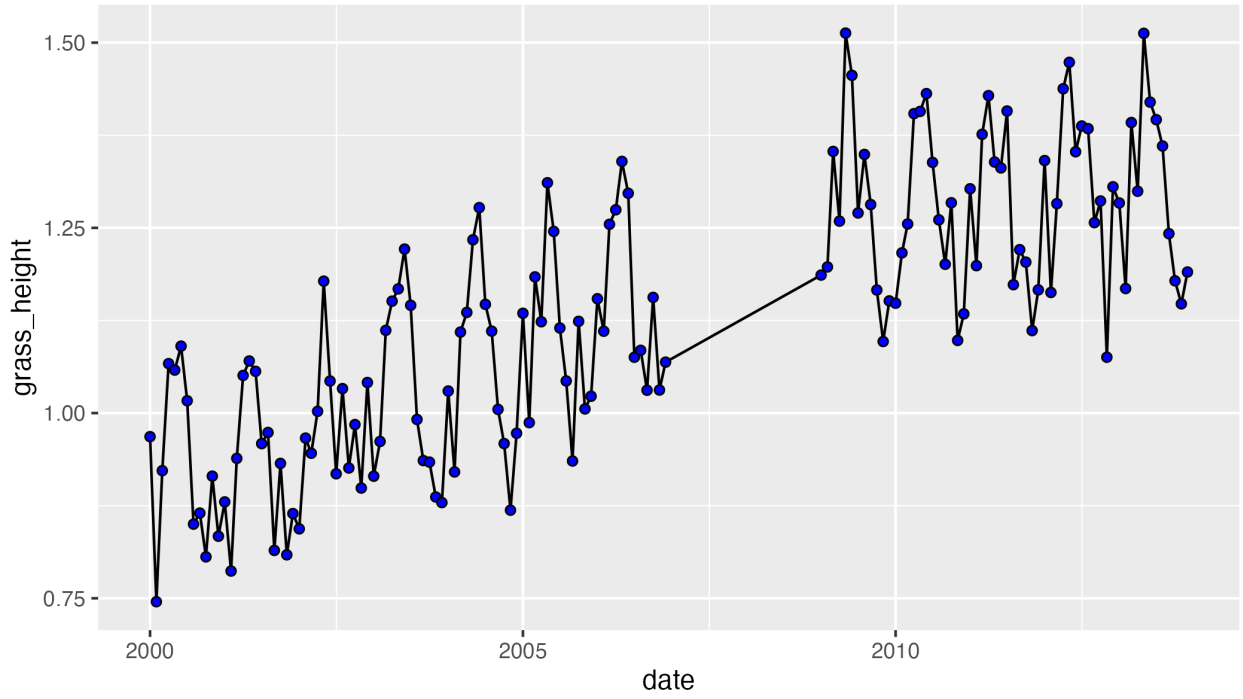

Figure 1: Time series plot of grass height.

It looks like there is some data missing somewhere between 2005 and 2010. To make working with the time series easier, we need to convert these implicit gaps (i.e. the time points are absent) into explicit gaps (i.e. the time points are present and grass height is NA). First, create a dataframe containing all the time points:

```
grass_fill_dates <- data.frame(
  year_month = seq(min(grass$year_month), max(grass$year_month), by = 1),
  grass_height = NA_real_)
```

Then join just the missing dates to the original dataframe:

```
grass_fill <- grass %>%
  bind_rows(anti_join(grass_fill_dates, grass, by = "year_month")) %>%
  arrange(year_month)
```

4.1 Linear interpolation

To do more advanced time series analysis it is necessary to estimate the missing values in the dataset.

The simplest way to do this would be to simply sample from a line drawn between the start and end of the period missing data:

```
# Find missing date period
grass_missing_period <- grass_fill %>%
  filter(is.na(grass_height)) %>%  # Find all NA values
  pull(year_month)  # Extract date column

# Find start and end of missing data period
# Add or subtract 1 month from end or start of period, respectively
grass_missing_start <- seq(grass_missing_period[1],
  length = 2, by = -1)[2]
grass_missing_end <- seq(grass_missing_period[length(grass_missing_period)],
  length = 2, by = 1)[2]

# Do linear interpolation
interp_lin <- approx(
  x = c(grass_missing_start, grass_missing_end),
  y = c(
    grass_fill$grass_height[grass_fill$year_month == grass_missing_start],
    grass_fill$grass_height[grass_fill$year_month == grass_missing_end]),
  xout = as.numeric(grass_missing_period))

# Create dataframe of interpolated values
interp_lin_df <- data.frame(
  year_month = yearmonth(interp_lin$x),
  grass_height = interp_lin$y)

# Visualise linear interpolation
ggplot() +
  geom_path(data = grass, aes(x = year_month, y = grass_height)) +
  geom_point(data = grass, aes(x = year_month, y = grass_height),
    shape = 21, fill = "blue") +
  geom_point(data = interp_lin_df, aes(x = year_month, y = grass_height),
    shape = 21, fill = "red")
```

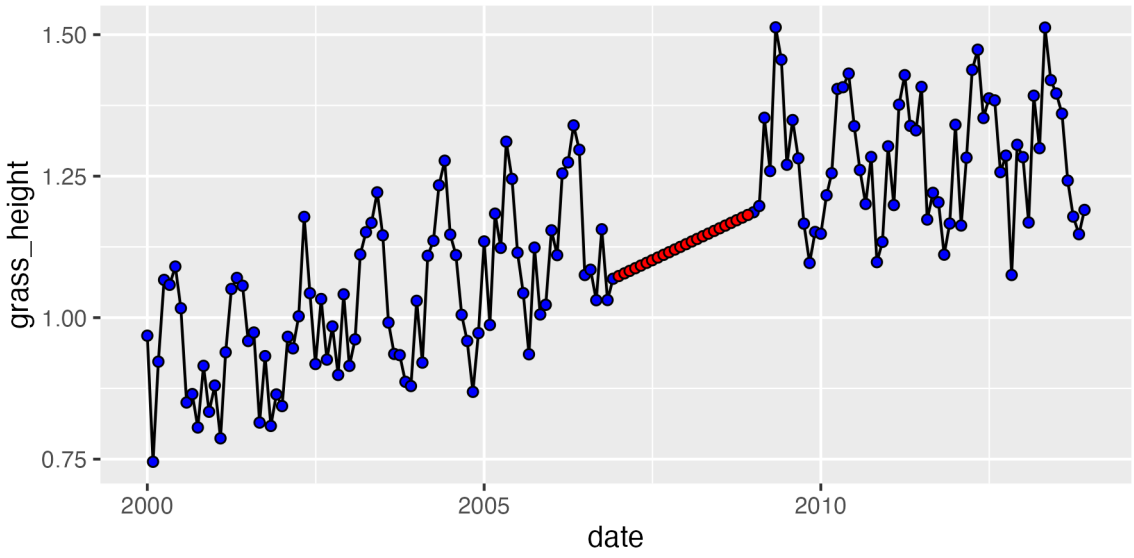

Figure 2: Linear interpolation of grass height.

Given the obvious oscillations in the data, this method of interpolation probably isn't very realistic.

We could try with a linear regression instead, which will give us a measure of uncertainty on our estimate as well:

```
# Run linear model of grass height by date
interp_lm_mod <- lm(grass_height ~ year_month, data = grass_fill)

# Predict missing values using model
interp_lm_pred <- predict(interp_lm_mod,
  newdata = data.frame(year_month = grass_missing_period), se = TRUE)

interp_lm_pred_df <- data.frame(
  year_month = grass_missing_period,
  grass_height = interp_lm_pred$fit,
  grass_height_se = interp_lm_pred$se.fit)

# Visualise linear model interpolation
ggplot() +
  geom_path(data = grass, aes(x = year_month, y = grass_height)) +
  geom_point(data = grass, aes(x = year_month, y = grass_height),
    shape = 21, fill = "blue") +
  geom_path(data = interp_lm_pred_df, aes(x = year_month, y = grass_height),
    colour = "red") +
  geom_ribbon(data = interp_lm_pred_df,
    aes(x = year_month, ymin = grass_height - grass_height_se,
      ymax = grass_height + grass_height_se), fill = "red", alpha = 0.2)
```

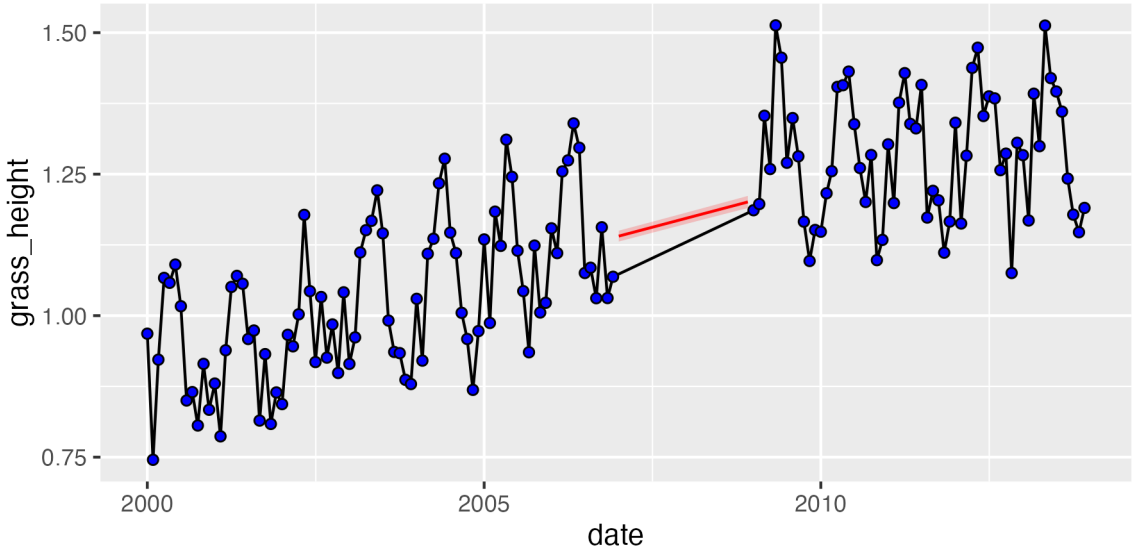

Figure 3: Linear regression interpolation of grass height.

The linear regression might be slightly closer to the truth than the linear interpolation, but not much. This method still can't capture the periodic oscillations in the data.

Linear regressions are also not very appropriate for time series data, because linear regressions use data in both the future and the past to predict the value of a given point on the regression line, while time series are strictly time-ordered; the data in the future cannot influence the data in the past.

To properly capture the variation in the time series to interpolate the missing values, we must turn to other modelling techniques that are more appropriate for time-series data.

4.2 Interpolation with ARIMA models

ARIMA models use information on temporal autocorrelation in a model to either interpolate or forecast values in the time series. Using the {fable} package we can interpolate the missing values in the time series using an ARIMA model:

```
# Interpolate using an ARIMA model
grass_tsbl <- as_tsibble(grass_fill, index = year_month)

# Run the ARIMA model and interpolate missing values
arima_interp <- grass_tsbl %>%
  model(ARIMA(grass_height~trend())) %>%
  interpolate(grass_tsbl)

# Extract missing values
arima_interp_fil <- arima_interp %>%
  filter(year_month %in% yearmonth(grass_missing_period))

# Create plot with interpolated values
ggplot() +
  geom_path(data = arima_interp_fil,
    aes(x = year_month, y = grass_height), colour = "red") +
  geom_path(data = grass_tsbl,
    aes(x = year_month, y = grass_height))
```

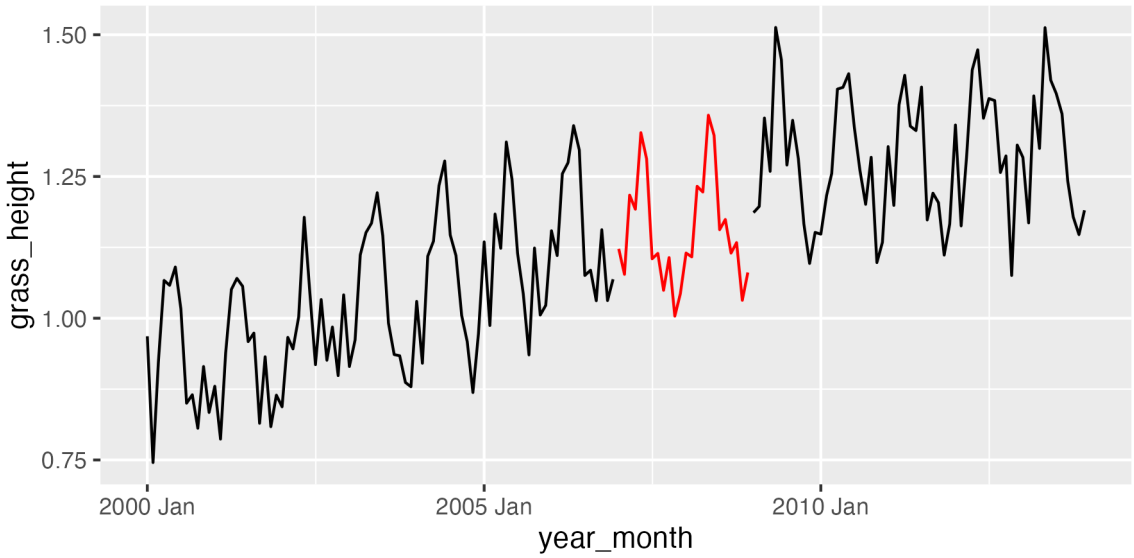

Figure 4: Interpolated values from ARIMA model, shown in red.

> Do you think the ARIMA model has done a good job at interpolating the missing data? Explain why.

4.3 Time series decomposition

A common procedure in time series analysis is to see if there are any temporal patterns in the data.

We can "decompose" the grass height time series (with the interpolated values) using the `stl()` function, but first we need to convert the tsibble dataframe into a time series (`ts`) object:

```
# Create time series object
grass_ts <- ts(arima_interp$grass_height,
  start = c(2000, 1), end = c(2013, 12), freq = 12)
```

```r
# Decompose time series
grass_stl <- stl(grass_ts, "periodic")

# Plot decomposition analysis
plot(grass_stl)
```

> Now that the time series has been decomposed, what do you see? Is there a seasonal pattern? Is there a longer-term trend? Is the data noisy?

4.4 Forecasting

Now that we have a full time series with all the gaps filled, and we have an idea of the structure of the time series from the decomposition analysis, we can move onto forecasting.

There are many different types of model which can be used to forecast time series data. We will only be looking at two, ETS and ARIMA models. Even within each of these model types, there are many variations with different adjustments for different types of data. Going into detail on that is outside the scope of this short practical session, but you can read more about it in Hyndman and Athanasopoulos (2021).

First, let's divide our data into training and testing sets. The training set will be used to fit the model we will use to forecast, and the testing set will be used to assess how well each of the models fits.

```r
# Split data into testing and training sets
grass_test <- arima_interp %>%
  filter(year_month >= yearmonth("2009-01-01"))

grass_train <- arima_interp %>%
  filter(year_month <= yearmonth("2008-12-01"))

# Visualise testing and training sets
ggplot() +
  geom_path(data = grass_test,
    aes(x = year_month, y = grass_height), colour = "red") +
  geom_path(data = grass_train,
    aes(x = year_month, y = grass_height), colour = "blue")
```

Then we can fit various models, including three ETS models with different seasonality fits, and a simple ARIMA model. `SNAIVE` is a truly naive model which simply carries forward the observed temporal autocorrelation pattern.

```r
grass_mods <- grass_train %>%
  model(
    snaive = SNAIVE(grass_height),
    ets = ETS(grass_height),
    etsa = ETS(grass_height ~ season("A")),
    etsm = ETS(grass_height ~ season("M")),
    arima = ARIMA(grass_height))
```

Then assess the fit of the models:

```r
accuracy(grass_mods)
report(grass_mods)
```

> Based on the AIC (Akaike Information Criterion) and RMSE (Root Mean Squared Error) values, which model fits the training data the best?

Now we can forecast the model forward by 60 months (5 years), so the forecast period matches the test data set:

```r
# Forecast the models 60 months into the future (5 years)
grass_fc <- forecast(grass_mods, h = 60)
```

```r
# Extract values from forecasts
grass_hilo <- hilo(grass_fc, level = 95)
grass_fc_df <- data.frame(mod = grass_fc$.model)
grass_fc_df$year_month <- as.Date(grass_fc$year_month)
grass_fc_df$mean <- grass_fc$.mean
grass_fc_df$lower <- grass_hilo$`95%`$lower
grass_fc_df$upper <- grass_hilo$`95%`$upper

# Plot forecasts with test data
ggplot() +
  geom_path(data = grass_train, aes(x = year_month, y = grass_height), colour = "blue") +
  geom_path(data = grass_test, aes(x = year_month, y = grass_height), colour = "red") +
  geom_ribbon(data = grass_fc_df,
    aes(x = year_month, ymin = lower, ymax = upper, fill = mod),
    alpha = 0.2) +
  geom_path(data = grass_fc_df, aes(x = year_month, y = mean, colour = mod)) +
  facet_wrap(~mod)
  labs(x = "Date", y = "Grass height (m)")
```

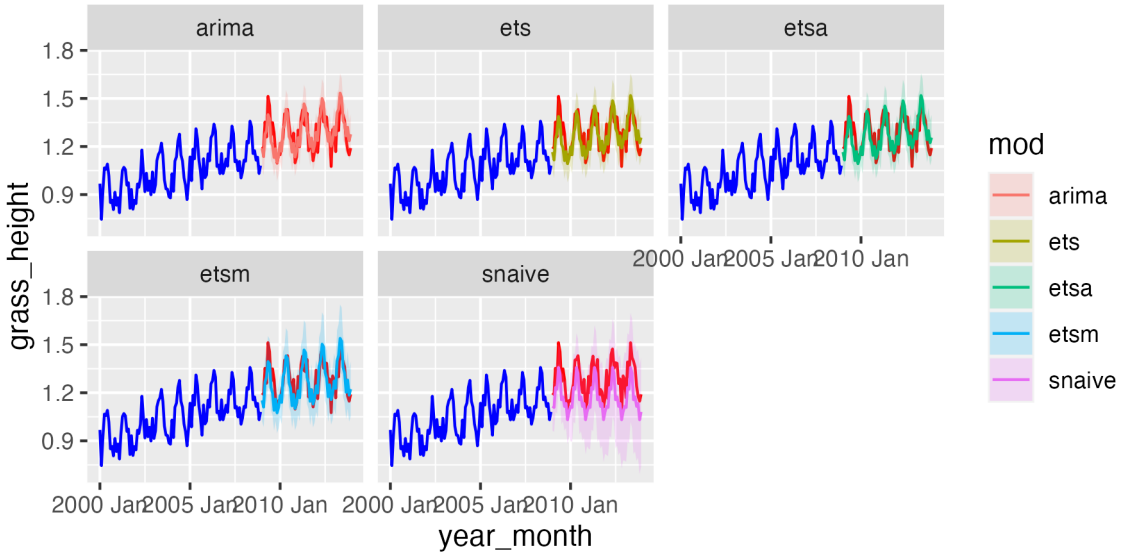

Figure 5: Facetted plot showing various forecasts and training data

It should be clear to see that the `snaive` model has a much wider confidence interval than the others.

> Looking at the model fits in the plot you generated above, which model do you think fits the best?

To be more certain of which model fits the best, we can calculate the RMSE of the model fit vs. the test set:

```r
# Join test set to forecasts
grass_comp <- full_join(grass_fc_df, grass_test, by = "year_month")

# Calculate RMSE on model estimates
grass_comp %>%
  group_by(mod) %>%
  summarise(rmse = sqrt(mean((grass_height - mean)^2))) %>%
  arrange(rmse)

# Create plot comparing test data to best model fit
ggplot() +
```

```
geom_path(data = grass_test, aes(x = year_month, y = grass_height), colour = "red") +
geom_ribbon(data = grass_fc_df %>% filter(mod == "arima"),
  aes(x = year_month, ymin = lower, ymax = upper),
  fill = "blue", alpha = 0.2) +
geom_path(data = grass_fc_df %>% filter(mod == "arima"),
  aes(x = year_month, y = mean), colour = "blue") +
labs(x = "Date", y = "Grass height (m)")
```

> Which model forecast fits the best according to the RMSE value? Hint: Better model fits minimise RMSE.

> Forecast the best model until 2030. What is the mean grass height in November 2028? How much confidence do you have in this estimate?

5 Dynamic models

In this section we are going to explore the tree-grass savanna coexistence model introduced in the lecture. The model has already been re-written as R code, in an easy to use function. The goal of this part of the practical is not to practice writing the code for the model, but rather to practice tweaking the model using the existing code.

The function has these arguments:

- `tn` = number of time units over which to simulate. Each unit is two months.
- Initial relative biomass components
  - `GSt` = GS_t = grass shoot
  - `GRt` = GR_t = grass root
  - `WSt` = WS_t = woody shoot
  - `WRt` = WR_t = woody root
- Decomposition (mortality) rates
  - `dG` = d_G = grass
  - `dW` = d_W = wood
- Saturating growth rate relationship coefficients with existing biomass
  - `gtildeG` = $\tilde{g}_G = \tilde{g}$ grass growth
  - `gtildeW` = $\tilde{g}_W = \tilde{g}$ tree growth
- Competition coefficients
  - `aWG` = α_{WG} = wood roots on grass roots
  - `aGW` = α_{GW} = grass roots on wood roots
  - `oWG` = ω_{WG} = wood shoots on grass shoots
- Sigmoidal relationship of biomass loss by fire with grass fuel load. NULL by default for no fire.
  - `aG` = $a_G = a$ coefficient for grass
  - `bG` = $b_G = b$ coefficient for grass
  - `aW` = $a_W = a$ coefficient for wood
  - `bW` = $b_W = b$ coefficient for wood
- `intF` = F = average fire return interval. NULL by default for no fire.
- `hL` = z = proportion of grass shoots lost to herbivory. NULL by default for no herbivory.
- `R` = R = total annual rainfall (mm)

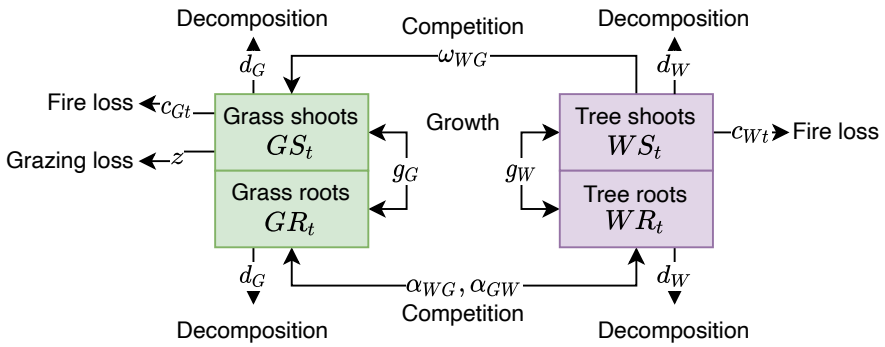

Figure 6: The dynamic model of tree-grass coexistence (Higgins, Scheiter, and Sankaran 2010).

Copy the model code from the `example_script.R` and run it to define the function `siml()`. The function starts on line 239.

The model uses a simple conversion factor to convert grass and wood relative biomass into estimates of actual biomass in Mg. These conversion factors are:

```
cfG <- 12  # Grass
cfW <- 120  # Wood
```

So, one unit of grass is approximately 12 Mg of biomass, and one unit of wood is approximately 120 Mg of biomass.

5.1 Model equilibrium

First, run the model using these basic parameters:

```
siml_out <- siml(
  tn = 1000,
  GSt = 0.5/cfG, GRt = 0.5/cfG, WSt = 80/cfW, WRt = 80/cfW,
  dG = 0.077, dW = 0.081,
  R = 500,
  gtildeG = 0.87, gtildeW = 1.23,
  aWG = 0.74, aGW = 1.16, oWG = 1.07)
```

In this simulation, the mean annual precipitation is 500 mm, and the model coefficients are the best estimates provided by the study in (Higgins, Scheiter, and Sankaran 2010). We are running the model for 500 time steps (166 years). Over a time period this long the model should reach equilibrium, i.e. the modelled estimates of grass and wood biomass should have stabilised from their initial values and should not change over time.

The function returns a dataframe with each of the biomass components at each timestep.

We can plot the biomass components from the model as a time series using this code:

```
# Make long dataframe of model outputs
siml_out_long <- siml_out %>%
  pivot_longer(cols = -one_of("timestep"))

ggplot(data = siml_out_long,
  aes(x = timestep, y = value)) +
  geom_path(aes(colour = name)) +
  labs(x = "Timestep", y = expression("Biomass"~(tAGB~ha^-1)))
```

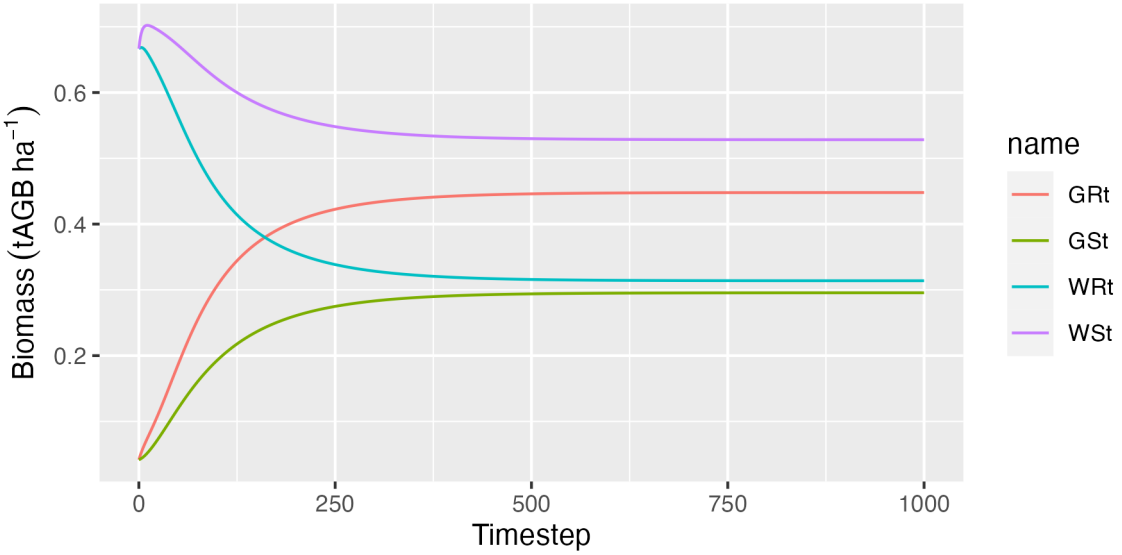

Figure 7: Simulated biomass components.

> Is the model output at equilibrium at the end of this simulation period?

> At 500 mm of rainfall, which is higher, relative woody biomass, or relative grassy biomass?
>
> Does this change when you convert into real units using the conversion factors `cfG` and `cfW`?
>
> Hint: Multiply each of the columns in `siml_out` by its respective conversion factor before using `pivot_longer()`.

5.2 Model behaviour

5.2.1 Rainfall

Now let's look at the behaviour of the model with changing rainfall.

First, create a vector of mean annual rainfall (R) values:

```
Rvec <- seq(0,1500, 100)
```

Then, using `lapply()` for each of those values, run the simulation, join the results into a single dataframe using `do.call(rbind, ...)`, and finally plot the output. Again, copy and paste the code from `example_script.R` to speed things along, but please ask if you don't understand what the code is doing:

```
Rvec <- seq(0,1500, 100)
simlR <- do.call(rbind, lapply(Rvec, function(x) {
  siml_out <- siml(
    tn = 1000,
    GSt = 0.5/cfG, GRt = 0.5/cfG, WSt = 80/cfW, WRt = 80/cfW,
    dG = 0.077, dW = 0.081,
    R = x,
    gtildeG = 0.87, gtildeW = 1.23,
    aWG = 0.74, aGW = 1.16, oWG = 1.07)

  siml_out$R <- x

  return(siml_out)
}))

# Transform to long format
```

```r
simlR_long <- simlR %>%
  pivot_longer(cols = -c("R", "timestep"))

# Plot biomass components over time for each rainfall simulation
ggplot(data = simlR_long,
  aes(x = timestep, y = value, colour = R, group = interaction(name, R))) +
  geom_path() +
  facet_wrap(~name, scales = "free") +
  scale_colour_scico(palette = "bamako") +
  labs(x = "Timestep", y = expression("Biomass"~(tAGB~ha^-1)))
```

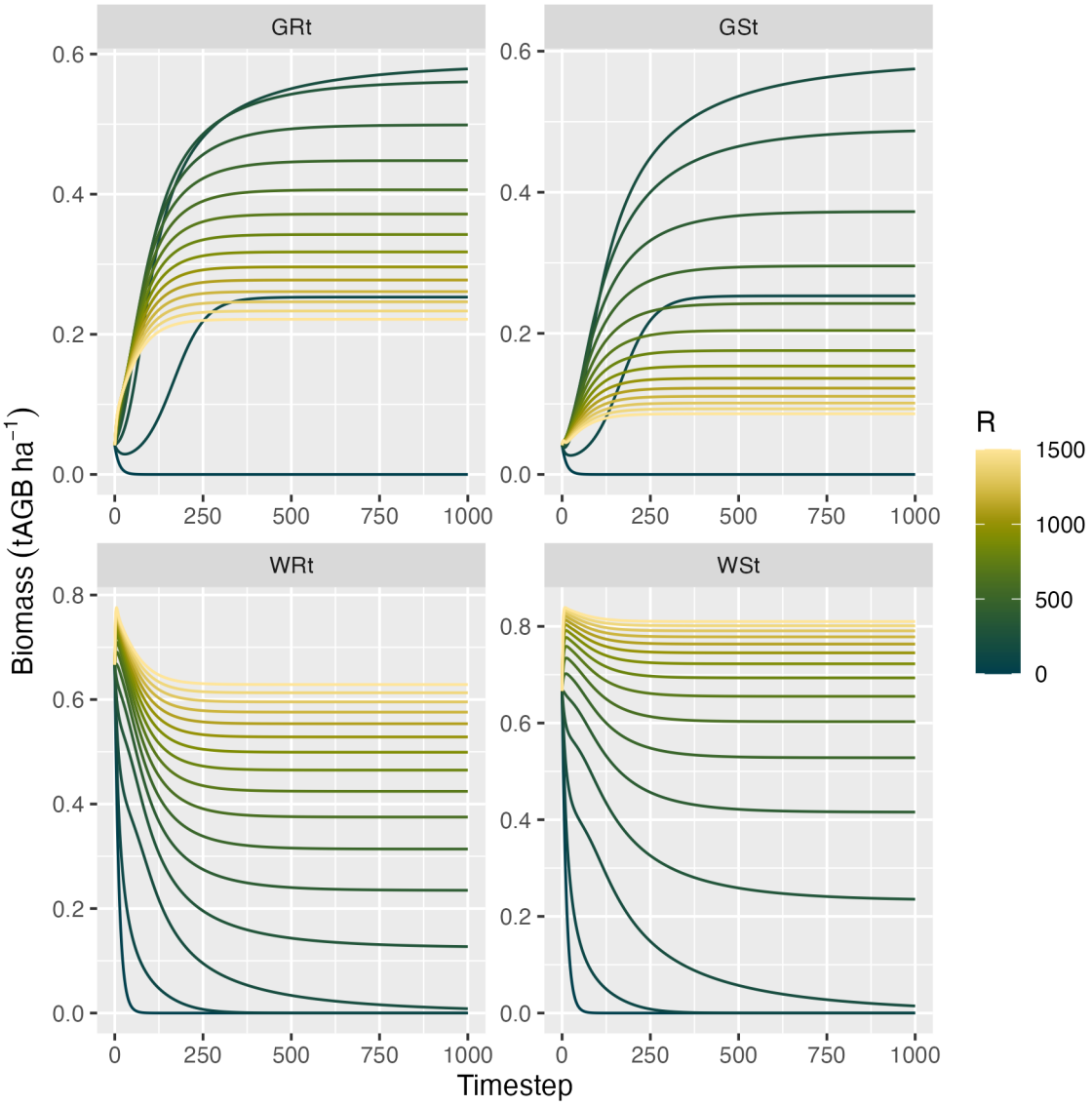

Figure 8: Modelled biomass components at varying rainfall.

> Are each of the simulations at equilibrium after 500 time points?
>
> If needed, increase the value of `tn` in the simulation to extend the simulation until each model run appears to have reached equilibrium.

As well as plotting the time series, we can also plot the values of each biomass component at the end of the simulation:

```
simlR_long %>%
  filter(timestep == max(timestep)) %>%
  ggplot(data = ., aes(x = R, y = value)) +
  geom_path() +
  facet_wrap(~name, scales = "free")
```

> Describe how the biomass components change as a function of rainfall.
>
> Using your understanding of competition between trees and graases, can you explain why these patterns are observed?
>
> Under what rainfall conditions are trees excluded?

We can also plot the relationship between root and shoot biomass with changing rainfall:

```
simlR %>%
  filter(timestep == max(timestep)) %>%
  dplyr::select(-timestep) %>%
  rename(G_St = GSt, G_Rt = GRt, W_St = WSt, W_Rt = WRt) %>%
  pivot_longer(
    cols = -R,
     names_to = c("var", ".value"),
     names_sep = "_") %>%
  ggplot(data = .,
    aes(x = St, y = Rt, group = var)) +
    geom_path(aes(colour = R)) +
    geom_point(aes(fill = R, shape = var), size = 4, colour = "black") +
    scale_shape_manual(name = NULL, values = c(21,22)) +
    scale_colour_scico(palette = "bamako") +
    scale_fill_scico(palette = "bamako")
```

> Describe what you see.
>
> How does the relationship between root and shoot biomass change with rainfall?
>
> How do grasses and trees differ?
>
> Can you explain why they differ?

5.2.2 Fire

The model can include stochastic disturbance processes in the form of fire. The `intF` argument provides an average fire return interval. I.e. the number of years on average between fires.

In the model, fires reduce biomass at a rate defined by a sigmoid relationship with existing grass biomass, parameterised separately for wood and grass. The higher the grass shoot biomass, the more biomass is lost, but the amount lost differs for grasses and trees.

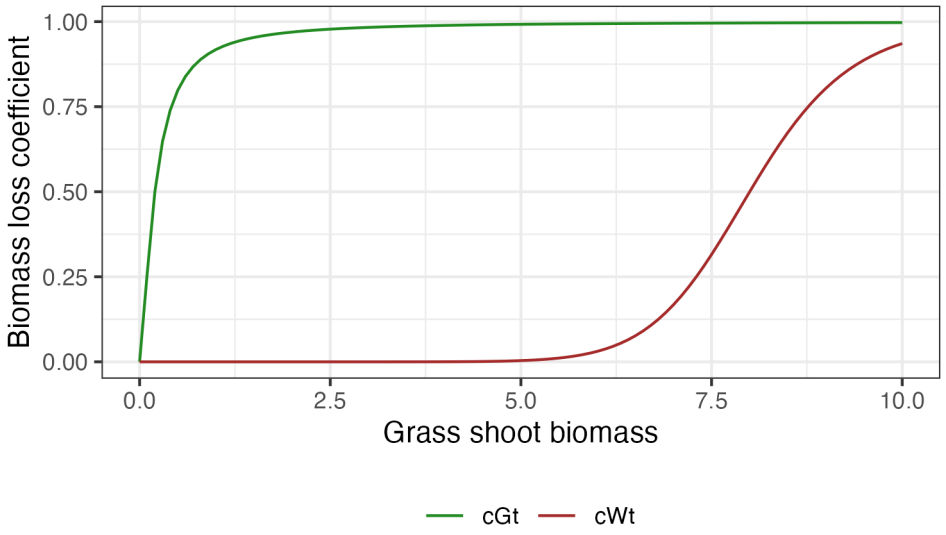

Figure 9: Relationship between grass shoot biomass and biomass lost for each biomass component

First have a go at running the model with fire every four years, and compare this the identical model from earlier without fire:

```
simlFire <- siml(
  tn = 1000,
  GSt = 0.5/cfG, GRt = 0.5/cfG, WSt = 80/cfW, WRt = 80/cfW,
  dG = 0.077, dW = 0.081,
  R = 500,
  gtildeG = 0.87, gtildeW = 1.23,
  aWG = 0.74, aGW = 1.16, oWG = 1.07,
  intF = 24, aG = 0.2, bG = 1.5, aW = 8, bW = 12)

simlFire_long <- simlFire %>%
  pivot_longer(cols = -"timestep")

ggplot() +
  geom_path(data = simlFire_long,
    aes(x = timestep, y = value, colour = name)) +
  geom_path(data = siml_out_long,
    aes(x = timestep, y = value, colour = name), linetype = 2) +
  labs(x = "Timestep", y = expression("Biomass"~(tAGB~ha^-1)))
```

> Describe how each of the biomass components vary with fire. Compare the model estimates with fire to those without fire, how do they differ?

5.2.3 Fire and herbivory

Now, let's run the model with fire and herbivory from large grazing mammals:

```
# Run simulation with fire and herbivory
simlFireHerb <- siml(
  tn = 1000,
  GSt = 0.5/cfG, GRt = 0.5/cfG, WSt = 80/cfW, WRt = 80/cfW,
  dG = 0.077, dW = 0.081,
  R = 500,
  gtildeG = 0.87, gtildeW = 1.23,
  aWG = 0.74, aGW = 1.16, oWG = 1.07,
  intF = 24, aG = 0.2, bG = 1.5, aW = 8, bW = 12,
```

```
  hL = 0.01)

simlFireHerb_long <- simlFireHerb %>%
  pivot_longer(cols = -"timestep")

ggplot() +
  geom_path(data = simlFireHerb_long,
    aes(x = timestep, y = value, colour = name), linetype = 1) +
  geom_path(data = simlFire_long,
    aes(x = timestep, y = value, colour = name), linetype = 2) +
  labs(x = "Timestep", y = expression("Biomass"~(tAGB~ha^-1)))
```

> Compare the model estimates from the model with fire and herbivory to those with just fire, how do they differ?

5.3 Scaling up

Now you should have an idea of how the model functions, and how it responds to changes in rainfall, fire regime, and disturbance by herbivores.

The next step in our study is to apply this model to some real world data to validate the model.

Fundamentally, this model relies on inputs of rainfall and fire regime to predict biomass and the balance of tree and grass biomass, so if we feed the model those parameters we can predict biomass at a given point. `pr2021` is a raster object containing total annual precipitation values across southern Africa in the year 2021.

Plot the data to see how it looks:

```
plot(pr2021)
```

> Describe how annual rainfall varies across southern Africa.

`pr2080` is a raster object containing predicted total annual precipitation values across southern Africa in the year 2080. You can subtract the `pr2021` raster from the `pr2080` raster to visualise how rainfall is expected to change over the next 60 years.

```
prdiff <- pr2080 - pr2021

plot(prdiff)
```

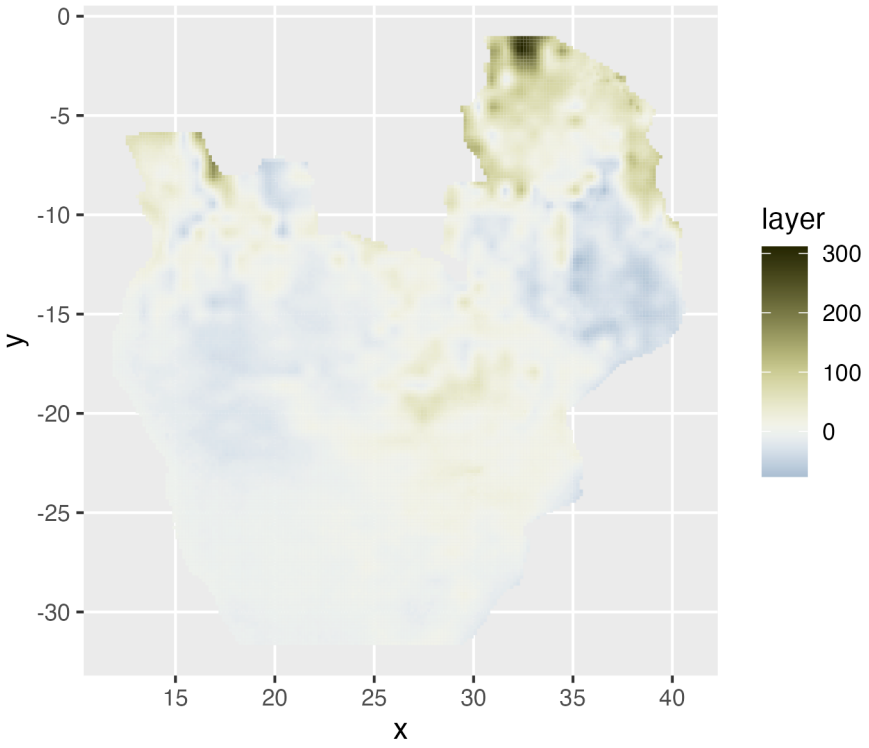

Figure 10: Difference in precipitation between 2021 and 2080.

> What is the overall trend in rainfall over time?
>
> Which areas are getting wetter, which are getting drier?
>
> Hint: positive values mean it is wetter in 2080 than in 2021.

We can run the model simulation for each value in `pr2021`:

```
Rvec <- values(pr2021)

simlpr2021 <- do.call(rbind, mclapply(seq_along(Rvec), function(x) {
    message(x)
  if (is.na(Rvec[x])) {
    siml_out <- data.frame(
      timestep = seq(0, 500, 1),
      GSt = NA, GRt = NA, WSt = NA, WRt = NA, R = NA)
  } else {
    siml_out <- siml(
    tn = 500,
    GSt = 0.5/cfG, GRt = 0.5/cfG, WSt = 80/cfW, WRt = 80/cfW,
    dG = 0.077, dW = 0.081,
    R = Rvec[x],
    gtildeG = 0.87, gtildeW = 1.23,
    aWG = 0.74, aGW = 1.16, oWG = 1.07)

    siml_out$R <- Rvec[x]
  }

  siml_fin <- siml_out[siml_out$timestep == max(siml_out$timestep),]
  siml_fin$px <- x
  return(siml_fin)
}, mc.cores = detectCores()))
```

Then multiply the values of relative biomass to get actual biomass for each component:

```
simlpr2021$WRt <- simlpr2021$WRt * cfW
simlpr2021$WSt <- simlpr2021$WSt * cfW
simlpr2021$GRt <- simlpr2021$GRt * cfG
simlpr2021$GSt <- simlpr2021$GSt * cfG

# Sum of all biomass components
simlpr2021$grass_biomass <- simlpr2021$GRt + simlpr2021$GSt
simlpr2021$wood_biomass <- simlpr2021$WRt + simlpr2021$WSt
simlpr2021$root_biomass <- simlpr2021$GRt + simlpr2021$WRt
simlpr2021$shoot_biomass <- simlpr2021$GSt + simlpr2021$WSt
simlpr2021$biomass <- simlpr2021$grass_biomass + simlpr2021$wood_biomass
```

Then create new raster layers with the modelled biomass values:

```
pr2021_grass_biomass <- pr2021
pr2021_wood_biomass <- pr2021
pr2021_root_biomass <- pr2021
pr2021_shoot_biomass <- pr2021
pr2021_biomass <- pr2021

values(pr2021_grass_biomass) <- simlpr2021$grass_biomass
values(pr2021_wood_biomass) <- simlpr2021$wood_biomass
values(pr2021_root_biomass) <- simlpr2021$root_biomass
values(pr2021_shoot_biomass) <- simlpr2021$shoot_biomass
values(pr2021_biomass) <- simlpr2021$biomass
```

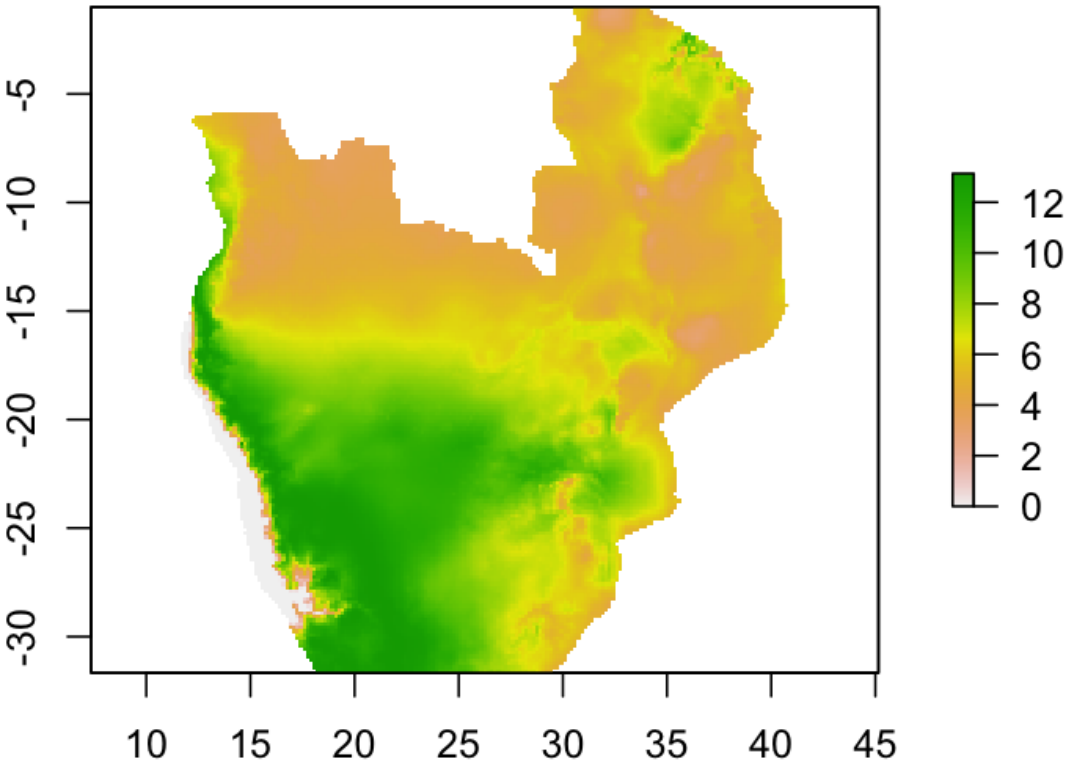

Figure 11: Plot of modelled grass biomass in 2021

> Plot each of the new raster layers (e.g. `plot(pr2021_grass_biomass)`).
>
> What is the spatial pattern of modelled biomass?
>
> How do each of the biomass components vary across southern Africa?

5.4 Model validation

To validate the model, we can compare biomass from our model to other modelled biomass data products. There are many maps of above-ground biomass, here we will use just one, from a pan-tropical study of biomass in Avitabile et al. (2016). The "Avitabile" biomass map used a model-data fusion framework, which creates a statistical model to predict biomass values that best fit the input data. The input data came from a combination of observations from multi-spectral satellites (MODIS), national biomass maps, and field data from plots.

First, plot the Avitabile map:

```
plot(avit)
```

Then, compare the values of `pr2021_shoot_biomass` with values in `avit` using a simple scatter plot:

```
plot(values(avit), values(pr2021_biomass))
```

Describe the relationship between the Avitabile model and our biomass model.

Why might these two models differ in their predictions? Be specific. Think about the data sources used in each.

5.5 Future model predictions

For the last part of the practical, let's try to model how biomass might change in the future, using climate predictions for a high emissions scenario in the year 2080.

As before, run the model for each pixel in the pr2080 raster and extract the biomass values:

```
Rvec <- values(pr2080)

simlpr2080 <- do.call(rbind, mclapply(seq_along(Rvec), function(x) {
    message(x)
  if (is.na(Rvec[x])) {
    siml_out <- data.frame(
      timestep = seq(0, 500, 1),
      GSt = NA, GRt = NA, WSt = NA, WRt = NA, R = NA)
  } else {
    siml_out <- siml(
    tn = 500,
    GSt = 0.5/cfG, GRt = 0.5/cfG, WSt = 80/cfW, WRt = 80/cfW,
    dG = 0.077, dW = 0.081,
    R = Rvec[x],
    gtildeG = 0.87, gtildeW = 1.23,
    aWG = 0.74, aGW = 1.16, oWG = 1.07)

    siml_out$R <- Rvec[x]
  }

  siml_fin <- siml_out[siml_out$timestep == max(siml_out$timestep),]
  siml_fin$px <- x
  return(siml_fin)
}, mc.cores = detectCores()))

simlpr2080$WRt <- simlpr2080$WRt * cfW
simlpr2080$WSt <- simlpr2080$WSt * cfW
simlpr2080$GRt <- simlpr2080$GRt * cfG
simlpr2080$GSt <- simlpr2080$GSt * cfG

simlpr2080$grass_biomass <- simlpr2080$GRt + simlpr2080$GSt
simlpr2080$wood_biomass <- simlpr2080$WRt + simlpr2080$WSt
simlpr2080$root_biomass <- simlpr2080$GRt + simlpr2080$WRt
simlpr2080$shoot_biomass <- simlpr2080$GSt + simlpr2080$WSt
simlpr2080$biomass <- simlpr2080$grass_biomass + simlpr2080$wood_biomass

pr2080_grass_biomass <- pr2080
pr2080_wood_biomass <- pr2080
pr2080_root_biomass <- pr2080
pr2080_shoot_biomass <- pr2080
pr2080_biomass <- pr2080

values(pr2080_grass_biomass) <- simlpr2080$grass_biomass
values(pr2080_wood_biomass) <- simlpr2080$wood_biomass
values(pr2080_root_biomass) <- simlpr2080$root_biomass
values(pr2080_shoot_biomass) <- simlpr2080$shoot_biomass
values(pr2080_biomass) <- simlpr2080$biomass
```

Now, as before, subtract the 2021 estimate from the 2080 estimate to find the difference:

```
pr2080_biomass_diff <- pr2080_biomass - pr2021_biomass

pr2080_biomass_diff_df <- rastDfConv(pr2080_biomass_diff)

ggplot() +
  geom_tile(data = pr2080_biomass_diff_df,
    aes(x = x, y = y, fill = layer)) +
  scale_fill_scico(palette = "broc", midpoint = 0) +
  coord_sf()
```

```
ggplot() +
  geom_histogram(aes(x = values(pr2080_biomass)), fill = "red", alpha = 0.5) +
  geom_histogram(aes(x = values(pr2021_biomass)), fill = "blue", alpha = 0.5)
```

> Describe what you see.
>
> Which areas are losing biomass, which are gaining? Hint: `plot(pr2080_biomass - pr2021_biomass)`.
>
> Overall, is there more or less biomass in the region? Hint: use `sum(values(pr2080_biomass), na.rm = TRUE)`, for example.
>
> Does the balance of biomass components change between 2021 and 2080? Hint: Calculate the difference using `pr2080_grass_biomass - pr2021_grass_biomass`, for example.

6 References

Avitabile, Valerio, Martin Herold, Gerard B. M. Heuvelink, Simon L. Lewis, Oliver L. Phillips, Gregory P. Asner, John Armston, et al. 2016. "An Integrated Pan-Tropical Biomass Map Using Multiple Reference Datasets." *Global Change Biology* 22 (4): 1406–20. https://doi.org/10.1111/gcb.13139.

Döscher, Ralf, Mario Acosta, Andrea Alessandri, Peter Anthoni, Almut Arneth, Thomas Arsouze, Tommi Bergmann, et al. 2021. "The EC-Earth3 Earth System Model for the Climate Model Intercomparison Project 6," February. https://doi.org/10.5194/gmd-2020-446.

Fick, S. E., and R. J. Hijmans. 2017. "WorldClim 2: New 1-Km Spatial Resolution Climate Surfaces for Global Land Areas." *International Journal of Climatology* 37 (12): 4302–15. https://doi.org/http://dx.doi.org/10.1002/joc.5086.

Higgins, Steven I., Simon Scheiter, and Mahesh Sankaran. 2010. "The Stability of African Savannas: Insights from the Indirect Estimation of the Parameters of a Dynamic Model." *Ecology* 91 (6): 1682–92. https://doi.org/10.1890/08-1368.1.

Hyndman, R. J., and G. Athanasopoulos. 2021. *Forecasting: Principles and Practice, 3rd Edition*. Melbourne, Australia: OTexts.

Meinshausen, Malte, Zebedee R. J. Nicholls, Jared Lewis, Matthew J. Gidden, Elisabeth Vogel, Mandy Freund, Urs Beyerle, et al. 2020. "The Shared Socio-Economic Pathway (SSP) Greenhouse Gas Concentrations and Their Extensions to 2500." *Geoscientific Model Development* 13 (8): 3571–3605. https://doi.org/10.5194/gmd-13-3571-2020.