silvR Documentation

Release 0

Samuel Bowers

Apr 26, 2019

Contents

silvR is a set of R functions for analysing forest inventory data.

In this document we introduce the silvR functions, and run through a set of worked examples with synthetic tree inventory data. We recommend you learn how the silvR functions work, and then try to input your own data.

CHAPTER 1

Who do I talk to?

Repo maintained by Samuel Bowers (sam.bowers@ed.ac.uk).

CHAPTER 2

Contents

2.1 Setup instructions

2.1.1 Requirements

silvR is a set of R functions written for use in Linux or Windows. None of the functions of silvR should require a powerful PC to run, though some will require an internet connection to function properly.

Functions in silvR make use of a common R libraries for data manipulation (dplyr, tidyr), ecological analysis (vegan), geospatial data manipulation (raster, rgdal, sp), and a specialised library for forest plot data analysis (BIOMASS).

2.1.2 Installing silvR

To install silvR, you'll first have to install and load devtools as follows:

```
> install.packages('devtools')

> library('devtools')
```

Once you have devtools, you can install silvR as follows:

```
> install_bitbucket('sambowers/silvr')
```

To load the silvR functions, run the following:

```
> library('silvr')
```

2.1.3 Loading data

For the purposes of this documentation, we'll be making use of a synthetic forest inventory dataset. The dataset contains stems from 25 forest inventory plots of 0.25 ha located in miombo woodlands.

The datasets are packaged with silvr, and are available as follows:

```
> data(mstems) # Stem data, one row per tree stem

> data(mplots) # Plot data, one row per plot
```

Take a peek at these two datasets using the `head()` function before continuing:

```
> head(mstems)

> head(mplots)
```

2.2 Introduction

2.2.1 What is silvR?

silvR is an R package. Packages are sets of functions, data and documentation that is easy to share with others. R has many thousands of packages that are publically and freely available. If you have a statistical problem, chances are someone has already solved it and published a package that you can use.

silvR has been developed to assist in the analysis of forest inventory plot data, with a focus on problems that are specific to southern African woodlands. silvR has been developed as part of the SEOSAW project (https://seosaw.github.io/), which has brought together plot inventory data from across southern Africa.

The package is still under development, so if you have any suggestions for changes or for new features, do let us know!

2.2.2 Required data structure

We strongly advise that you organise your data in the following standardised way. By organising forest inventory data as we describe here, you'll be able to make best use of the functions of silvR, but also find that your data will be easier to apply other common R functions.

Functions in silvR work on two scales: individuals and plots. Example functionality that is applicable to the individual scale is:

- Calculating stem biomass
- Correction of species names

. . . and functionality that is applicable to the plot scale:

- Calculating plot biomass
- Characterisation of dominant species
- Extracting data from remote sensing products

We recommend you maintain two R `data.frames`: one containing individual stem data, and one containing plot data. There should be a `plotcode` field in both data.frames, so that individual stem data can be linked to properties of the plot it was measured at.

Your `individual stem` data.frame should be similar to:

plotcode	species_name	DBH	height
P01	Brachystegia spiciformis	10.5	7.5
P01	Julbernardia globiflora	22.1	10.0
P02	Diplorhynchus condylocarpon	15.4	8.0
P02	Brachystegia spiciformis	31.9	12.1
P03	Afzelia quanzensis	48.8	20.2

Your `plot` data.frame should be similar to:

plotcode	latitude	longitude	area_ha
P01	-18.967	26.332	0.25
P02	-19.498	26.635	0.25
P03	-19.011	27.090	0.25

Note: At minimum your `stem` data.frame should contain a plotcode, species_name and DBH, and your `plot` data.frame should contain a plotcode, latitude, and longitude. Other fields describing individuals (e.g. height, stem location) or plots (e.g. vegetation type, meteorological data) should also be included where available.

2.2.3 Using bundled data

silvR comes bundled with two datasets that can be used to test out its functions. These data are for synthetic (*i.e. totally fictional!*) forest plot data in miombo woodlands. These are loaded with:

```
> data(mstems) # Stem-scale data

> data(mplots) # Plot-scale data
```

2.2.4 Cleaning your own data

At present, there is only limited functionality in silvR to check and clean input data, meaning that you will have to make sure that input data are sensible. Where there are transciprion errors in your data, silvR will not necessarily be able to identify them. Checks we can recommend are:

- Are all DBH measurements realistic, and in the same units (cm or m)?
- Are there an equal number of unique plot codes in your `stem` and `plot` data.frames?
- Are plot locations all given in the same coordinate reference system (e.g. WGS 84)?
- Are all species names sensible, and without spelling mistakes? N.B. There is a function in silvR that performs basic species name corrections.

2.3 Structural parameters

2.3.1 Loading silvR and sample data

First, make sure that you've followed the setup instructions. You'll also need to load in the sample data included in silvR to follow along with these examples, which can be performed with:

```
> data(mstems) # Stem-scale data

> data(mplots) # Plot-scale data
```

If you've not yet installed silvR, return to the setup instructions before continuing.

Note: All functions and datasets in silvR come with documentation. Where in doubt, get help using the `?` character. e.g. `?calculateBasalArea` or `?calculateBiomass`.

2.3.2 Calculating structural parameters

Structural parameters are calculated at the level of an individual tree (e.g. basal area, biomass), but are usually expressed at the scale of a plot. Here we'll run through the calculation of these parameters a the scale of a stem, then show you how to summarise these at the scale of a plot.

2.3.3 Basal area

Basal area is the area of land that is occupied by the cross-section of a tree.

We can calculate basal area from a tree's diameter at breast height (DBH), which is usually measured at 1.3 m above ground level. Assuming a circular tree bole, we can calculate basal area using the equation:

BA = π(DBH/2)^2

To calculate basal area using silvR, we can input a vector of tree DBH measurements into the following function:

```
> calculateBasalArea(mstems$DBH)
  [1] 0.030171856 0.633348221 0.012468981 0.089727028 0.030790750 0.296091966 0.
↪014957123 0.026015529
  [9] 0.014957123 0.133316626 0.258769845 0.658993041 0.010207035 0.051471854 0.
↪029559245 0.134614104
 [17] 0.024884555 0.057255526 0.009503318 0.009503318 0.012076282 0.033329156 0.
↪244544714 0.011689866
 [25] 0.423137973 0.115811672 0.009503318 0.022698007 0.009852035 0.468084739 0.
↪092940877 0.094024727
```

(continues on next page)

(continued from previous page)

```
  [33] 0.026015529 0.185507905 0.048305129 0.053092916 0.079422604 0.084496276 0.
↪021124069 0.012076282
  [41] 0.015393804 0.309748469 0.011689866 0.541060795 0.020106193 0.246300864 0.
↪012468981 2.301958035
  [49] 0.021124069 0.899202357 0.446511422 0.030171856 0.087615878 0.055571632 0.
↪031415927 0.013684778
  ...
```

You can save the result to the original data.frame by defining a new column as follows:

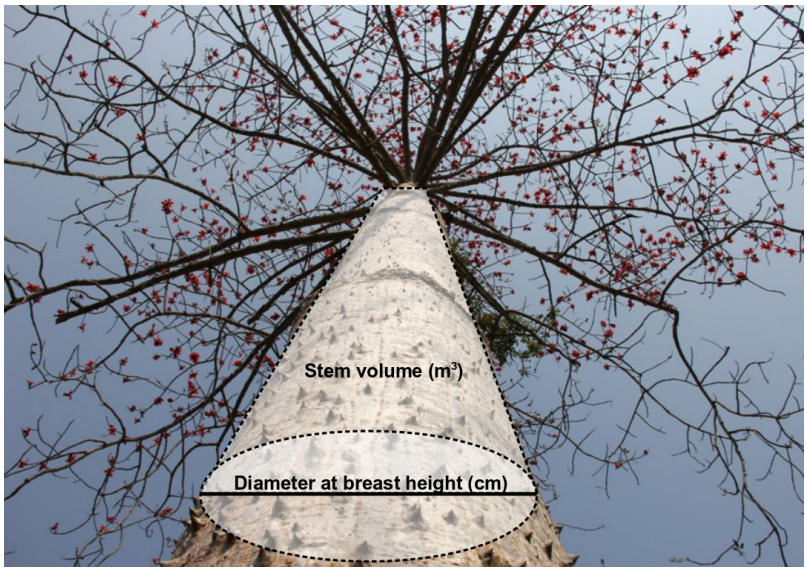
```
> mstems$basal_area <- calculateBasalArea(mstems$DBH)
```

For all of the following examples we'll add the data to the existing `mstems` data.frame.

Note: By default, silvR functions such as `calculateBasalArea` require DBH to be given in units of `cm` and provides outputs in standard units (e.g. `m^2` for basal area). Check documentation for other options.

2.3.4 Stem volume

Stem volume provides an estimate of the ammount of space that the tree bole occupies.



Stem volume is calculated using the equation of **[SOURCE]**. Calculate it using silvR as follows:

```
> mstems$volume <- calculateStemVolume(mstems$DBH)
```

2.3.5 Stocking density

We can use silvR to calculate stocking density of stems greater than a minimum DBH. At the scale of a stem, this translates to values of 1 (included) and 0 (excluded).

To calculate whether a tree contributes to the stocking count in silvR:

```
> mstems$stocking <- calculateStocking(mstems$DBH)
```

To do the same, but with a minimum DBH of 10 cm:

```
> mstems$stocking_10 <- calculateStocking(mstems$DBH, min_DBH = 10)
```

2.3.6 Aboveground biomass

silvR is pre-programmed with a range of allometric equations that are valid for miombo woodlands. By default, silvR will use the `Ryan11` model described in this paper:

Ryan, Casey M., Mathew Williams, and John Grace. "Above-and belowground carbon stocks in a miombo woodland landscape of Mozambique." Biotropica 43.4 (2011): 423-432.

Run this in silvR using:

```
> mstems$AGB_Ryan11 <- calculateBiomass(mstems$DBH)
```

Recall that we can view summary statistics using the R `summary` function:

```
> summary(mstems$AGB_Ryan11)
Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.003714 0.014319 0.050140 0.218266 0.183587 7.424314
```

We can also use other allometric models. For example:

```
> mstems$AGB_Mugasha14 <- calculateBiomass(mstems$DBH, model = 'Mugasha14')
> mstems$AGB_Mutakele09<- calculateBiomass(mstems$DBH, model = 'Mutakele09')
```

See documentation (`?calculateBiomass`) for all allometric models.

Note: Most allometric models only require `DBH` as input to calculate DBH, but be aware that some may require height or even species information. Some of these more advanced models for estimating aboveground biomass are used in silvR (`Chave05` and `Chave14`), but we'll return to these in the next section.

2.3.7 Aggregating structural parameters to plot scale

Most commonly we will want to express these structural parameters at plot scale. For example, basal area is usally expressed as the sum total of basal area of all trees, in units of m^2/ha.

We can use functions from base R to aggregate by plot:

```
> ba_plot <- aggregate(basal_area ~ plot_code, mstems, sum)

> ba_plot$basal_area <- ba_plot$basal_area / 0.25 # Because each plot is only 0.25 ha␣
↪in size, and we want units of m^2/ha

> head(ba_plot)
  plot_code basal_area
1   plot_01   25.08409
2   plot_02   19.17778
3   plot_03   30.12648
4   plot_04   43.59910
```

(continues on next page)

(continued from previous page)

```
5   plot_05   27.74461
6   plot_06   21.80926
```

Try to repeat this process to calculate stocking density and aboveground biomass at each plot.

2.4 Species functions

2.4.1 Formatting species data

Species data tend to come in the format `Genus species`, but most functions in silvR require separate vectors for `Genus` and `species`.

silvR has a function to split character strings in the format `Genus species` or `Genus.species` into two vectors, and enforces correct capitalisation:

```
> splitGenusSpecies(mstems$species_name)
          genus        species
1      Parinari curatellifolia
2    Terminalia        sericea
3       Afzelia     quanzensis
4 Erythrophleum      africanum
5    Terminalia        sericea
6  Brachystegia    spiciformis
...
```

These can be attached to the original data.frame using the `cbind` function from base R to be used as inputs to other functions:

```
> mstems <- cbind(mstems, splitGenusSpecies(mstems$species_name))

> head(mstems$genus) # Show the first few lines of the new field
    [1] "Parinari"      "Terminalia"
[3] "Afzelia"       "Erythrophleum"
[5] "Terminalia"    "Brachystegia"
```

2.4.2 Correcting species data

On occasion we'll want to group species by order or family, requiring time consuming work. Getting consistent spellings in forest inventory data is a further challenge, yet necessary in order to produce accurate assessments of species distributions and biodiversity. There already exist R packages that are capable of automating such spelling corrections (e.g. `BIOMASS` and `taxize`), in silvR we have aimed to simplify some of this processing based on functions from the `BIOMASS` package.

The function `getTaxonomicNames` has multiple purposes. First, it can identify order and family given a genus:

```
> taxonomy <- getTaxonomicNames(mstems$genus)

> head(taxonomy)
     order           family         genus species
1 Malpighiales Chrysobalanaceae      Parinari    <NA>
2     Myrtales     Combretaceae    Terminalia    <NA>
3      Fabales         Fabaceae       Afzelia    <NA>
```

(continues on next page)

(continued from previous page)

```
4      Fabales         Fabaceae Erythrophleum    <NA>
5     Myrtales     Combretaceae    Terminalia    <NA>
6      Fabales         Fabaceae  Brachystegia    <NA>
```

If we're only interested in one taxonomic rank, we can specify it using the `ranks` input:

```
> mstems$family <- getTaxonomicNames(mstems$genus, ranks = c('family')) # Add to␣
↪original data.frame

> head(mstems$family)
[1] "Chrysobalanaceae" "Combretaceae"     "Fabaceae"         "Fabaceae"
↪"Combretaceae"
[6] "Fabaceae"
```

`getTaxonomicNames` can also search through species databases and perform automated correcion of species names using the `clean_input` option:

```
> species_corrected <- getTaxonomicNames(mstems$genus, species = mstems$species,␣
↪clean_input = TRUE)

> head(species_corrected)
         order           family         genus        species
1 Malpighiales Chrysobalanaceae      Parinari curatellifolia
2     Myrtales     Combretaceae    Terminalia        sericea
3      Fabales         Fabaceae       Afzelia     quanzensis
4      Fabales         Fabaceae Erythrophleum      africanum
5     Myrtales     Combretaceae    Terminalia        sericea
6      Fabales         Fabaceae  Brachystegia    spiciformis
```

To also show which species names have been changed, use the `return_changes` option which will return a `nameModified` field:

```
> species_corrected <- getTaxonomicNames(mstems$genus, species = mstems$species,␣
↪clean_input = TRUE, return_changes = TRUE)

> head(species_corrected)
         order           family         genus        species nameModified
1 Malpighiales Chrysobalanaceae      Parinari curatellifolia        FALSE
2     Myrtales     Combretaceae    Terminalia        sericea        FALSE
3      Fabales         Fabaceae       Afzelia     quanzensis        FALSE
4      Fabales         Fabaceae Erythrophleum      africanum        FALSE
5     Myrtales     Combretaceae    Terminalia        sericea        FALSE
6      Fabales         Fabaceae  Brachystegia    spiciformis        FALSE
```

Note: We advise that you use these functions with caution and don't rely on their outputs 100%; they can occasionally make mistakes leaving you with records of species that don't actually exist in your inventory plots.

2.4.3 Loading wood density

Wood density is a useful plant trait, and is required by some allometric equations to predict tree biomass. Wood density is species specific, and can be loaded from existing databases. silvR uses the `BIOMASS` library to produce estimates of wood density.

At it's simplest, this function requires a genus as input:

```
> mstems$wood_density <- loadWoodDensity(mstems$genus)
```

We can get better estimates with both genus and species:

```
> mstems$wood_density <- loadWoodDensity(mstems$genus, species = mstems$genus)
```

In some cases a species will not be present in the wood density database. For these cases, we can also input family:

```
> mstems$wood_density <- loadWoodDensity(mstems$genus, species = mstems$genus, family␣
↪= mstems$family)
```

For cases where family is also not enough to retrieve a wood density, we can input the plot code for each stem which will apply the average wood density for each plot to unknown species:

```
> mstems$wood_density <- loadWoodDensity(mstems$genus, species = mstems$genus, family␣
↪= mstems$family, plot = mstems$plotcode)
```

Note: By default silvR assumes a location of tropical Africa. If you're working on data from outside Africa, take a look at at the documentation using `?loadWoodDensity` and the `region` option.

2.4.4 Esimating aboveground biomass

For some more complex allometric equations, species names and tree height are used to extract wood density estimates. An example of these are the 'Chave' allometric equations applicable to tropical trees, described in:

1. Chave, Jerôme, et al. "Tree allometry and improved estimation of carbon stocks and balance in tropical forests." Oecologia 145.1 (2005): 87-99.
2. Chave, Jérôme, et al. "Improved allometric models to estimate the aboveground biomass of tropical trees." Global change biology 20.10 (2014): 3177-3190.

To use these equations in silvR, we'll return to the `calculateBiomass` function, using additional inputs describing species:

```
> mstems$AGB_Chave05 <- calculateBiomass(mstems$DBH, height = mstems$height, model =
↪'Chave05', family = mstems$family, genus = mstems$genus, species = mstems$species)

> mstems$AGB_Chave14 <- calculateBiomass(mstems$DBH, height = mstems$height, model =
↪'Chave14', family = mstems$family, genus = mstems$genus, species = mstems$species)
```

See `?calculateBiomass` for more information.

2.4.5 Identifying dominant species

A simple way of summarising species data is to identify the dominant species at each plot.

By default, the function identifies 5 species by stocking density at each plot:

```
> dominant_species <- getDominantSpecies(mstems$species_name, mstems$plot_code)

    > head(dominant_species)
            plot_code            dominant_sp_1            dominant_sp_2            ␣
↪dominant_sp_3
```

(continues on next page)

(continued from previous page)

```
   1   plot_01   Isoberlinia angolensis Brachystegia spiciformis  Julbernardia␣
↪globiflora
   2   plot_02 Brachystegia spiciformis   Isoberlinia angolensis  Erythrophleum␣
↪africanum
   3   plot_03           Faurea saligna   Isoberlinia angolensis Brachystegia␣
↪spiciformis
   4   plot_04   Isoberlinia angolensis Brachystegia spiciformis  Julbernardia␣
↪globiflora
   5   plot_05   Isoberlinia angolensis Brachystegia spiciformis          Burkea␣
↪africana
   6   plot_06  Julbernardia globiflora Brachystegia spiciformis  Erythrophleum␣
↪africanum
                          dominant_sp_4               dominant_sp_5 dominant_sp_mt_
↪1 dominant_sp_mt_2 dominant_sp_mt_3
   1 Parinari curatellifolia Diplorhynchus condylocarpon               21          ␣
↪   13               13
   2          Faurea saligna        Anisophyllea boehmii               13          ␣
↪    9                8
   3 Julbernardia globiflora Diplorhynchus condylocarpon               18          ␣
↪   18               17
   4    Anisophyllea boehmii             Burkea africana               24          ␣
↪   22               20
   5      Marquesia macroura        Brachystegia boehmii               16          ␣
↪   15               11
   6         Burkea africana        Dombeya rotundifolia               14          ␣
↪   10               10
     dominant_sp_mt_4 dominant_sp_mt_5 dominant_sp_pc_1 dominant_sp_pc_2 dominant_sp_
↪pc_3 dominant_sp_pc_4
   1               13               11         13.20755         8.176101         8.
↪176101         8.176101
   2                8                7         10.83333              7.5         6.
↪666667         6.666667
   3               17               13         7.725322         7.725322         7.
↪296137         7.296137
   4               17               15         9.125475         8.365019         7.
↪604563         6.463878
   5               11                9          9.69697         9.090909         6.
↪666667         6.666667
   6                9                9         9.090909         6.493506         6.
↪493506         5.844156
     dominant_sp_pc_5
   1         6.918239
   2         5.833333
   3         5.579399
   4         5.703422
   5         5.454545
   6         5.844156
```

Note: This function uses complete species names (i.e. *Genus species*) in place of individual genus and species vectors.

The data.frame output by this function is complex. Columns `dominant_sp_#` show the dominant species for each plot (from 1 to n), `dominant_sp_mt_#` show the value of the comparison metric (in this case stocking density), and `dominant_sp_pc_#` the percentage of the comparison metric by each species.

We can combine this function with structural information, such as basal area or biomass. For example:

```
> dominant_species <- getDominantSpecies(mstems$species_name, mstems$plot_code,␣
↪metric = calculateBasalArea(mstems$DBH))
```

or aboveground biomass

```
> dominant_species <- getDominantSpecies(mstems$species_name, mstems$plot_code,␣
↪metric = calculateBiomass(mstems$DBH, model = 'Ryan11'))
```

2.4.6 Calculating biodiversity

Biodiversity describes the varaibility of species in a forest inventory plot. Measures of biodiversity vary from very simple (species richness) to more complex measures based on the distribution of individuals amongst species.. These measures of diveristy are collectively called 'diversity indices'.

silvR is pre-programmed with a range of diversity indices (see documentation at `?calculateBiodiversity`).

To calculate species richness:

```
> calculateBiodiversity(mstems$species_name, mstems$plot_code, index = 'richness')
plot_01 plot_02 plot_03 plot_04 plot_05 plot_06 plot_07 plot_08 plot_09 plot_10 plot_
↪11 plot_12 plot_13
    25      26      27      27      25      25      25      25      22      20      ␣
↪25      26      15
plot_14 plot_15 plot_16 plot_17 plot_18 plot_19 plot_20 plot_21 plot_22 plot_23 plot_
↪24 plot_25
    22      26      25      25      25      26      26      23      24      20      ␣
↪23      24
```

For a more complex measure of diveristy, use the 'simpson' or 'shannon' indices:

```
> calculateBiodiversity(mstems$species_name, mstems$plot_code, index = 'shannon')
 plot_01  plot_02  plot_03  plot_04  plot_05  plot_06  plot_07  plot_08  plot_09 ␣
↪plot_10  plot_11
2.930107 3.025561 3.117128 3.089926 3.009947 3.088370 3.060124 2.978967 2.887835 2.
↪767140 3.012447
plot_12  plot_13  plot_14  plot_15  plot_16  plot_17  plot_18  plot_19  plot_20 plot_
↪21  plot_22
2.984938 2.440725 2.917779 2.954540 3.023698 3.015096 2.959135 3.062585 3.005983 2.
↪944163 2.998152
plot_23  plot_24  plot_25
2.815136 2.944381 2.940585
```

If you want to calculate diodiversity based on another metric, such as basal area, use the `abundance` option:

```
calculateBiodiversity(mstems$species_name, mstems$plot_code, index = 'shannon',␣
↪abundance = calculateBasalArea(mstems$DBH))
 plot_01  plot_02  plot_03  plot_04  plot_05  plot_06  plot_07  plot_08  plot_09 ␣
↪plot_10  plot_11
2.645848 2.846774 3.063116 2.854390 2.710108 2.982382 2.983830 2.747758 2.654970 2.
↪366331 2.973090
plot_12  plot_13  plot_14  plot_15  plot_16  plot_17  plot_18  plot_19  plot_20  plot_
↪21  plot_22
2.865765 1.988800 2.796020 2.862064 2.613919 2.917068 2.728365 2.991442 2.868902 2.
↪707835 2.750818
plot_23  plot_24  plot_25
2.435320 2.654030 2.455297
```

2.4.7 Preparing data for vegan (advanced)

If you're interested in assessment of biodiversity and species distributions, we can recommend looking at the `vegan` package in R, which provides a wide variety of very useful functions. Many functions in `vegan` require data to be arranged in a different format to silvR.

To convert to a vegan applicable format, run:

```
mstems_vegan <- prepareForVegan(mstems$species_name, mstems$plot_code)
```

See `?prepareForVegan` for further options.

2.5 Working with spatial data

Locations of forest inventory plots are usually marked by points measured with a GPS unit.

There exists a wealth of freely available data from remote sensing and modelling which can be used to better characterise your plots. With silvR and R we can take advantage of this data, and integrate forest inventory measurements with global data products.

2.5.1 Loading sample data

For this part of the lirbary, we'll use a dataset that describes the plots and their locations. If not already present, load them into memory using:

```
> data(mplots)
```

2.5.2 Managing point data

Spatial data can be recorded in a wide range of coordinate reference systems (CRS). You'll probably be familiar with data provided as latitude and longitude (often WGS84) or Universal Transverse Mercator (UTM), (most of Zimbabwe sits in UTM zone 35S).

It can be complex to describe a coordinate reference system in R, but there is a shortcut known as EPSG codes. The most commonly used CRSs have each been given a unique code, which silvR has been programmed to understand. For example, WGS84 has the EPSG code 4326, and UTM35S has the code 32735. You can find out what EPSG code your CRS has from the website http://www.epsg.org/.

silvR has a function to reproject point data to a different CRS given the EPSG codes to convert from and to. For example, to convert the set of points provided in the `mplots` dataset from UTM37S to WGS84 (lat/lon), you can run the follows:

```
> points_rep <- reprojectPoints(mplots$UTM_E, mplots$UTM_N, CRS_from = 32735, CRS_to␣
↪= 4326)

> head(points_rep)
  x_coords  y_coords
1 26.96684 -18.54202
2 26.94692 -18.46403
3 27.12489 -18.55811
4 26.88303 -18.41498
5 26.99591 -18.74058
6 27.18346 -18.62647
```

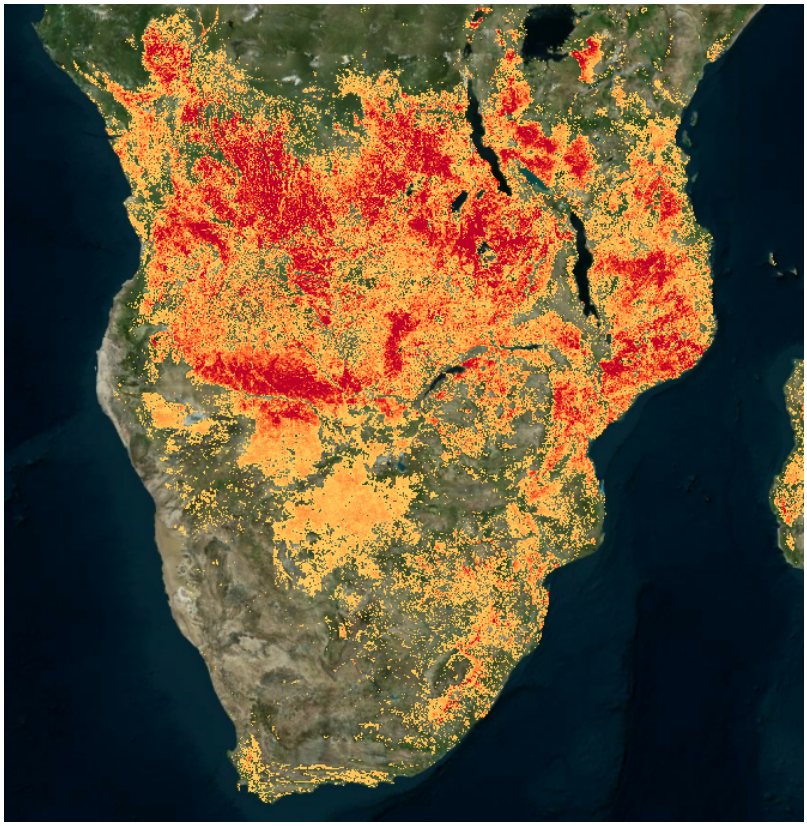
Note: **For advanced users** The `rgdal` package has a specialised CRS object that can also be input into `reprojectPoints`. silvR has a function to load a CRS object from its epsg code. For example: `CRS_WGS84 <- getCRSFromEPSG(4326)`. You may want to use this function if combining silvR with more advanced spatial processing with `rgdal`.

2.5.3 Extracting data from Rasters

Raster data consists of an array of cells (a.k.a pixels) arranged into rows and columns. These data can be georeferenced, such that each pixel represents a known location on the Earth. Common examples of raster data are remote sensing images or modelled meteorological data. A commonly used example of raster data is the GeoTiff format.

Combining forest inventory measurements with raster data opens a lot of opportunities. For example, we might want to determine meteorological parameters for each plot (temperature, precipitation, annual variation), land cover types (grassland, savannah, forest), or generate calibration data for creating our own remote sensing products such as biomass maps.

For the purposes of silvR, we've prepared a test dataset which contains a count of the number of burn scars detected by MODIS in southern Africa over the period 2001 - 2018 at 1 km resolution. Download the dataset `here` (2 MB), and save it somewhere memorable (e.g. Desktop).



To extract pixel values for locations of forest plots, we can use the `extractFromRaster` function:

```
> mplots$fire_count <- extractFromRaster('path/to/firecount_2001_2018.tif', mplots
↪$UTM_E, mplots$UTM_N, EPSG = 32735)

> mplots$fire_count
```

(continues on next page)

(continued from previous page)

```
 [1] 0 0 0 0 0 1 0 1 1 0 2 1 4 0 5 4 1 7 0 0 0 1
[23] 0 0 1
```

`extractFromRaster` will understand a range of geospatial raster data types, including GeoTiffs, HDF files, and VRT files. Feel free to experiment with other data sources.

2.5.4 Extracting data from Vectors

The other commonly used geospatial data type is vector data, which is constructed of points, lines, and polygons. Commonly used vector data formats include shapefiles. Vector datasets can contain, for example, data on locations of protected areas, soil types, and political boundaries such as regions and districts.

Warning: There is not at present an `extractFromVector` function in silvR... but watch this space!

2.6 What next?

2.6.1 Applying functions to your own data

Up to this point we've been using synthetic forest plot data. Real life forest plot data will be harder to use: there will be formatting issues, spelling mistakes, and transcription errors to deal with. The key to getting good results using the silvR functions (or any other R library) will be to making sure that your data are in the correct format and have undergone appropriate cleaning.

When you feel ready, arrange your data into a format similar to the sample data in silvR and load it into R (using `read.csv()`). Try and run through the worked examples, but this time with your own data.

2.6.2 New functionality

New functions will be added to silvR over time. If there's something you'd like to see added, please do let us know! Contact sam.bowers@ed.ac.uk.

2.6.3 Other libraries of interest

silvR is just a small part of a large ecosystem of R libraries. If you have a statistical problem, it's very likely that there exists an R library to help with it. Libraries that have been particularly helpful to us inclde:

1. vegan
2. BIOMASS
3. taxize
4. ggplot2

2.6.4 How do I learn more?

Learning to program is mostly a matter of practice. There are lots of free resources available online to assist in learning R. Some we can recommend are:

1. https://www.datacamp.com/courses/free-introduction-to-r
2. https://www.youtube.com/playlist?list=PL6gx4Cwl9DGCzVMGCPi1kwvABu7eWv08P
3. https://ourcodingclub.github.io/
4. https://r4ds.had.co.nz/

Happy coding!

2.7 Working with real data

Thus far, we've been using synthetic forest inventory data. This is great for learning, but doesn't simulate the difficulties that are associated with real-world, messy data.

Here we'll look at inventory data from a 50 year fire experiment in Marondera, Zimbabwe. The experiment was designed to look at the effect of frequent fires on tree populations. The photos below show what the plots looked like after being burned every year (top; fire return interval = 1 year) and after complete protection from fire (below; fire return interval = 50 years). The experiment had a number of plots subjected to a range of different fire treatments.

For more on the experiment see:

Furley, Rees, Ryan and Saiz (2009) Savanna burning and the assessment of long-term fire experiments with particular reference to Zimbabwe. Progress in Physical Geography 32(6): 611-634

We'll use this dataset to answer the following questions:

1. What is the relationship between fire return interval and aboveground biomass?
2. What effect does fire have on species composition?

2.7.1 Dataset description

The data are provided in `.csv` format, containing five columns:

Parameter	Description
plot_code	A unique ID for each experimental plot.
species_name	Species ID in the fomat 'Genus species'.
DBH	Diameter at breast height, in units of centimetres.
height	Stem height, in units of metres.
FRI	Fire return interval, in years. For example, a plot with an FRI treatment of 3 was burned every three years.

Download the data `here`.

Each plot has an area of is 60 x 36 m, which is 0.216 hectares.

2.7.2 Data preparation

First, load in `marondera.csv` using the `read.csv()` function.

Use the `head()` and `summary()` functions to get an understanding of the data:

1. How many plots are there?
2. How many experimental fire treatments?
3. What species are present?

This is real data, so it does include errors, which will impact the quality of results if not taken care of. Try to identify if there are any incorrect records:

1. Are all the DBH records realistic?
2. Are the species names all correct and well-ordered?

Note: Hint 1: Any measurement of a tree with a DBH of greater than 100 cm is probably an error. Hint 2: Some of the species names are mispelled, use the `unique` function to find out what they are.

What can you do to improve incorrect records?

Note: Hint 1: Consider using `NA` values. Hint 2: Recall that `silvR` can perform corrections to species names.

2.7.3 Stem attributes

Try and determine the following:

1. How many species are there are there across all plots?
2. Draw a histogram (in the form `hist(DBH)`) to show the stem size distribution.
3. Draw a scatter plot to determine if there's a relationship between DBH and stem height.
4. Use silvR to load estimates of wood density for each stem.

2.7.4 Plot attributes

Determine the structural attributes of each of the plots. Use `silvR` to calculate the following parameters for each tree:

1. Stocking density
2. Basal area
3. Aboveground biomass

Note: Remember that these are best presented on a per hectare basis. Recall that each plot at Marondera is 0.216 ha.

Then use the `aggregate` function to produce an estimate of each of these parameters at plot scale.

Note: Are all of these plot-scale numbers reasonable? If not, you may need to perform additional data cleaning.

Use silvR to identify the dominant three species, and to calculate biodiverstiy indices for each plot.

2.7.5 The impact of fire return interval on vegetation structure

Summarise the plot structure parameters you just calculated by fire treatment.

1. How does stocking density, basal area, and biomass vary with FRI?
2. Can a miombo woodland with annual fires support any aboveground biomass?

These data are probably better expressed graphically. Try to build a boxplot of fire return interval vs the structural parameters. The syntax to build a boxplot looks like:

```
boxplot(biomass~FRI, data=plot_data)
```

2.7.6 Advanced: The effect of fire on species composition

1. How does species composition vary by FRI treatment? (Hint: you can express this as the top-5 species under each treatment)
2. Is there any relationship between biodiversity and FRI?

Warning: These questions are quite tricky. Consider using the `table` function to produce a summary.

- genindex
- modindex
- search