Whittaker biomes - the dataset

Valentin Stefan

2017-11-27

Contents

This document explains how the `Whittaker_biomes` data frame was constructed.

Table of contents

Original graph

The original graph is Figure 5.5 in *Ricklefs, R. E. (2008), The economy of nature. W. H. Freeman and Company.* (Chapter 5, Biological Communities, The biome concept).

Processing PDF

A PDF page containing the Whittaker biomes graph was imported in Inkscape. Text and extra graphic layers where removed and the remaining layers were exported as PostScipt file format (`File > Save As > Save as type: > PostScript (*.ps)`). Note that a multi-page PDF document can be split into component pages with PDFTK Builder application.

Importing PostScript file in R with `grImport`

The main steps of importing PostsScipt in R are described in the Importing vector graphics vignette: *Murrell, P. (2009). Importing vector graphics: The grImport package for R. Journal of Statistical Software, 30(4), 1-37.*

In addition to installing package `grImport` with the usual `install.packages("grImport")`, Ghostscript needs to be installed as well.

Read in the PostScript file

Note: to avoid the ghostscript error 'status 127', the path to ghostscript executable file was given as suggested here

```
require(grImport)
Sys.setenv(R_GSCMD = normalizePath("C:/Program Files/gs/gs9.22/bin/gswin64c.exe"))
# Path to example PostScript file
path <- system.file("extdata", "graph_ps.ps",
                    package = "plotbiomes",
                    mustWork = TRUE)
# Path to output xml RGML file
RGML_xml_path <- gsub(pattern = "graph_ps.ps",
                 replacement = "graph_ps.xml",
                 x = path,
                 fixed = TRUE)
# Converts a PostScript file into an RGML file
PostScriptTrace(file = path,
                outfilename = RGML_xml_path,
                setflat = 1)
# setflat = 1 assures a visual smooth effect (feel free to experiment)

# Reads in the RGML file and creates a "Picture" object
my_rgml <- readPicture(rgmlFile = RGML_xml_path)
```

Visualize and clean (subset) the "Picture" object

```
# Draws the picture
plot.new(); grid.picture(my_rgml)
# Draws each path composing the picture. This helps selecting desired paths.
plot.new(); picturePaths(my_rgml)
```

By using `picturePaths` one can identify the desired biome lines and the filled polygons. From polygons one can get the fill colors for further use.

```
# Selects desired paths from the picture object.
# These are the path corresponding to the filled polygons.
my_fills <- my_rgml[c(3,9,14,16,18,21,23,25,27)]
# Gets the colors
colors <- vector(mode = "character", length = length(my_fills@paths))
for (i in 1:length(my_fills@paths)){
  colors[i] <- my_fills@paths[[i]]@rgb
}
```

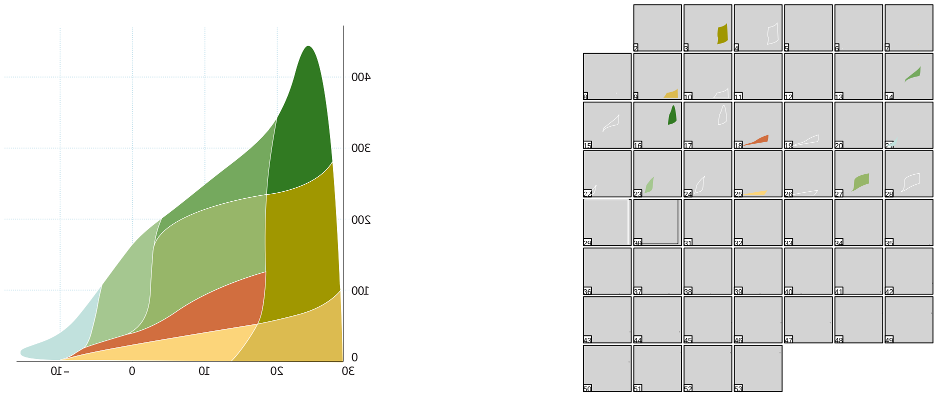

Figure 1: The picture object (left) and its component paths (right). The picture was flipped in Inkscape.

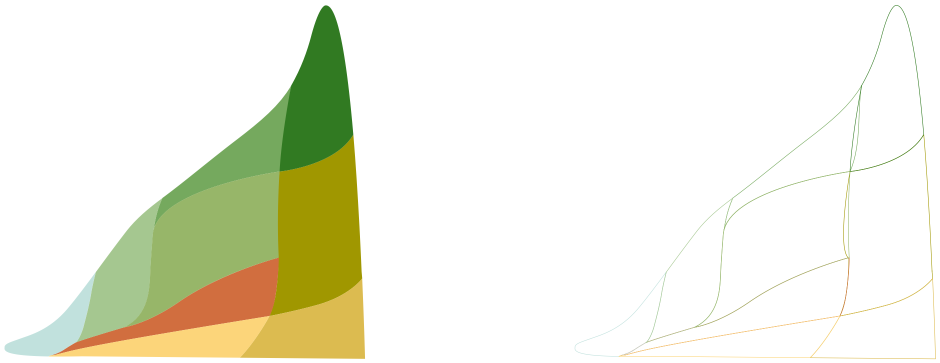

Figure 2: Polygons (left) and their borders (right)

```r
# These are the path corresponding to the biome lines (polygon boundaries)
my_rgml_clean <- my_rgml[c(4,10,15,17,19,22,24,26,28)]
# Assigns colors to the lines
for (i in 1:length(my_rgml_clean@paths)){
  my_rgml_clean@paths[[i]]@rgb <- colors[i]
}

plot.new(); grid.picture(my_fills)
plot.new(); grid.picture(my_rgml_clean)
```

Get the PostScript coordinates from the cleaned “Picture” object

```r
x_lst <- vector(mode = "list", length = length(my_rgml_clean@paths))
y_lst <- vector(mode = "list", length = length(my_rgml_clean@paths))
for (i in 1:length(my_rgml_clean@paths)){
  x_lst[[i]] <- my_rgml_clean@paths[[i]]@x
  names(x_lst[[i]]) <- rep(i, length(x_lst[[i]]))
  y_lst[[i]] <- my_rgml_clean@paths[[i]]@y
  names(y_lst[[i]]) <- rep(i, length(x_lst[[i]]))
}
x <- unlist(x_lst)
```

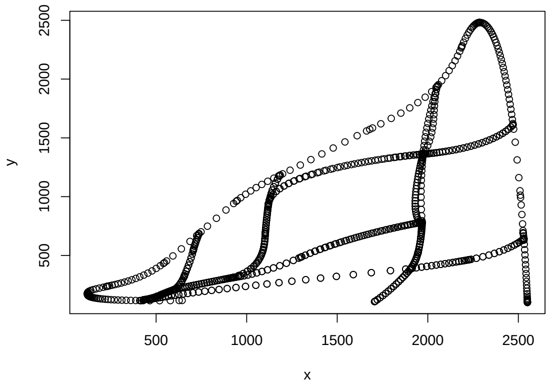

Figure 3: Plot using the original PostScript coordinates

```
y <- unlist(y_lst)

par(mar = c(4, 4, 1, 1))
plot(x,y)
```

Convert PostScript coordinates into meaningful coordinates

This varies from graph to graph, but in this case luckily there is a grid and axis on the graph that can be used to transform from PostScript coordinates to meaningful coordinates (temperature and precipitations).

```
my_grid <- my_rgml[1]
my_axis <- my_rgml[30]

x_grid <- my_grid@paths[[1]]@x
y_grid <- my_grid@paths[[1]]@y
x_grid_unq <- unique(sort(x_grid))
y_grid_unq <- unique(sort(y_grid))

# Plot original grid and axis
plot.new()
grid.picture(my_grid)
grid.picture(my_axis)

# Plot in PostScript coordinates
par(mar = c(4, 4, 1, 1))
plot(x_grid, y_grid, pch = 16, col = "red")
points(x, y, cex = 0.1)
abline(v = x_grid_unq[2], col = "red") # X min ~ -10
abline(v = max(my_axis@summary@xscale), col = "red") # X max ~ 30
abline(h = min(my_axis@summary@yscale), col = "red") # Y min ~ 0
abline(h = y_grid_unq[6], col = "red") # Y max ~ 400
```

Convert between coordinates systems

Need to identify the corresponding min & max values on OX and OY axis on both coordinates systems. The conversion equations are those exemplified here.

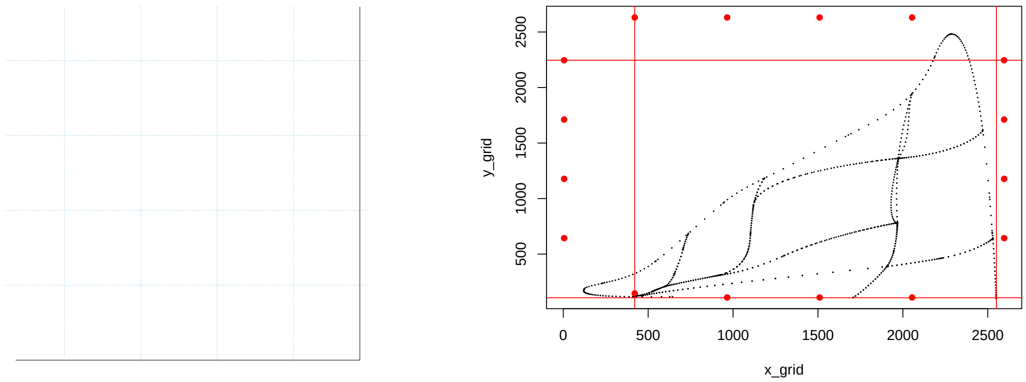

Figure 4: Original grid and axis (left) and their PostScript coordinates (right)

```
# Notations:
# scs - source coordinate system
# rcs - result coordinate system

x_min_scs <- x_grid_unq[2] # is the X of the left most vertical grid line
x_min_rcs <- -10 # corresponds to -10
x_max_scs <- max(my_axis@summary@xscale) # is the X max of OX axis line
x_max_rcs <- 30 # corresponds to 30

y_min_scs <- min(my_axis@summary@yscale) # is the Y min of OY axis line
y_min_rcs <- 0 # corresponds to 0
y_max_scs <- y_grid_unq[6] # is the Y of the upper most horizontal grid line
y_max_rcs <- 400

# Apply conversion
xx <- (x - x_min_scs)/(x_max_scs - x_min_scs) * (x_max_rcs - x_min_rcs) + x_min_rcs
yy <- (y - y_min_scs)/(y_max_scs - y_min_scs) * (y_max_rcs - y_min_rcs) + y_min_rcs

# There should not be negative values on OY.
# This happens because the original PDF has some artefacts,
# including also some overlapping polygons sections.
min(yy)
```

```
## [1] -1.4412
```

```
yy[yy < 0] <- 0

# Plot in original coordinates
par(mar = c(4, 4, 1, 1))
plot(x_grid, y_grid, pch = 16, col = "red",
     xlab = "X - original coordinates",
     ylab = "Y - original coordinates")
points(x, y, cex = 0.1)
abline(v = x_grid_unq[2], col = "red") # X min ~ -10
abline(v = max(my_axis@summary@xscale), col = "red") # X max ~ 30
abline(h = min(my_axis@summary@yscale), col = "red") # Y min ~ 0
abline(h = y_grid_unq[6], col = "red") # Y max ~ 400
```

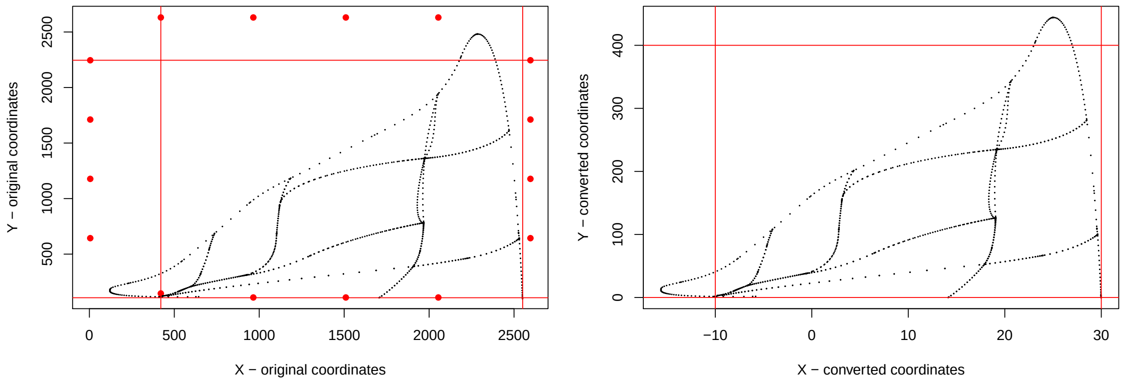

Figure 5: Left - original PostScript coordinates; Right - converted coordinates (precipitation vs. temperature)

```r
# Plot in converted coordinates
plot(xx, yy, cex = 0.1,
     xlab = "X - converted coordinates",
     ylab = "Y - converted coordinates")
abline(v = -10, col = "red") # X min
abline(v = 30, col = "red") # X max
abline(h = 0, col = "red") # Y min
abline(h = 400, col = "red") # Y max
```

Prepare Whittaker_biomes data frame

```r
Whittaker_biomes <- data.frame(temp_c = xx,
                               precp_cm = yy,
                               biome_id = as.numeric(names(xx)))

# Assigns the biome names
for (i in 1:nrow(Whittaker_biomes)){
  Whittaker_biomes$biome[i] <- switch(as.character(Whittaker_biomes$biome_id[i]),
                                      "1" = "Tropical seasonal forest/savanna",
                                      "2" = "Subtropical desert",
                                      "3" = "Temperate rain forest",
                                      "4" = "Tropical rain forest",
                                      "5" = "Woodland/shrubland",
                                      "6" = "Tundra",
                                      "7" = "Boreal forest",
                                      "8" = "Temperate grassland/desert",
                                      "9" = "Temperate seasonal forest")
}
```

Get the colors also

The fill colors where already extracted from PostScript. In the help of `ggplot2::scale_fill_manual` it is recommended to use a named vector. Therefore, colors were stored in the named character vector `Ricklefs_colors` below:

```
colors # hexadecimal color codes extracted from PostScript
```

```
## [1] "#A09700" "#DCBB50" "#75A95E" "#317A22" "#D16E3F" "#C1E1DD" "#A5C790"
## [8] "#FCD57A" "#97B669"
```

```
Ricklefs_colors <- colors
names(Ricklefs_colors) <- unique(Whittaker_biomes$biome)
# Sets a desired order for the names. This will affect the order in legend.
biomes_order <- c("Tundra",
                  "Boreal forest",
                  "Temperate seasonal forest",
                  "Temperate rain forest",
                  "Tropical rain forest",
                  "Tropical seasonal forest/savanna",
                  "Subtropical desert",
                  "Temperate grassland/desert",
                  "Woodland/shrubland")
Ricklefs_colors <- Ricklefs_colors[order(factor(names(Ricklefs_colors), levels = biomes_order))]
Ricklefs_colors
```

```
##                           Tundra                    Boreal forest
##                        "#C1E1DD"                        "#A5C790"
##        Temperate seasonal forest            Temperate rain forest
##                        "#97B669"                        "#75A95E"
##             Tropical rain forest Tropical seasonal forest/savanna
##                        "#317A22"                        "#A09700"
##               Subtropical desert       Temperate grassland/desert
##                        "#DCBB50"                        "#FCD57A"
##               Woodland/shrubland
##                        "#D16E3F"
```

Examples

Check examples in Whittaker_biomes_examples vignette, or after package installation, run `RShowDoc("Whittaker_biomes_exa` `package = "plotbiomes")`.