A methodology for flexible species distribution modelling within an Open Source framework

Duncan Golicher*and Luis Cayuela†

August 29, 2007

Technical report presented to the third International workshop on Species Distribution Modelling financed by the MNP 20-24 August 2007: San Cristóbal de Las Casas, Chiapas, Mexico.

Contents

*El Colegio de La Frontera Sur, San Cristobal de Las Casas, Chiapas Mexico

†Universidad de Alcala Henares, Madrid Spain

List of Figures

1 Introduction

Using statistical tools to make spatial explicit predictions of distribution is an interesting and informative means of finding relationships between sets of environmental variables and the occurrences of species or sets of species[9]. The maps that result usually aim to show potential rather than actual distributions[8]. They can be used in the context of scenario modelling by altering the inputs to the predictive model[13]. The variables used in fitting models are assumed to have spatial distributions that are well enough known to be available in the form of continuous raster coverages. Efficient algorithms for identifying relationships and mapping the resulting model onto space are widely available and have been implemented within a range of software tools[5]. Once relationships have been identified and maps produced the analyst can concentrate on the more complex and interesting task of critically interpreting the resulting pattern and placing the results in some applied or theoretical context[8].

A common challenge is that preparing all the information and the software tools necessary to reach the stage at which the exercise is productive can be time consuming, especially if a project cannot draw on past experience. Furthermore, while most academic species distribution modelling programs have been made freely available by the authors, the commercial GIS software used for data preprocessing can be expensive. This cost may represent a barrier for some research groups.

The aim of this document is to provide an easily followed methodology for rapid implementation of species distribution modelling using only open source software. This should allow the reader to reach the stage in which in depth analysis of the results can begin by drawing our group's experience. The example is drawn from our ongoing work modelling tree species at a regional (MesoAmerica) scale. Our intention is that the example should be easy to adapt for other projects. Some of the underlying concepts behind species distribution modelling will be analyzed in the text, but it is not the intention here to provide a detailed technical treatment or analysis of modelling algorithms.

The two principal programs that we use are the statistical environment R (http://www.r-project.org)[22], and the Geographical Information System GRASS (http://grass.itc.it). These programs both run under MS Windows, relying in part on the Unix emulator Cygwin (http://cygwin.com/). In the case of R this dependence is not visible to most users and R runs well under Windows. However integration between the two programs and the use of system resources are improved greatly when they are run in a Linux environment. Here we first provide brief instructions for setting up a complete working system with R and GRASS under Linux. We then demonstrate how the system can implement several different modelling approaches. The only additional programs we use in the species modelling context are the cross platform java based species modelling program maxent (http://www.cs.princeton.edu/~schapire/maxent) and the freely available (although not open source) geographical visualization program google earth (http://earth.google.com/download-earth.html). Google Earth also runs under

Linux.[1] The documentation of the project has been prepared using Open Office(http://www.openoffice.org) and type set using LyX (http://www.lyx.org/)[2] All programs were run under Ubuntu Feisty 7.04 (http://www.ubuntu.com). The results shown in this document have thus been achieved entirely without the use of software that requires a paid license. A second important element of our approach has been to document all the steps we used in order to ensure that they can be replicated exactly. This is achieved by the use of scripts rather than run by clicking menus on a GUI. The software we use is ideal for this. Both R and GRASS are fundamentally command line driven programs. This document assumes that the reader is prepared to make full use of all the on line resources available for learning Linux, GRASS and R and also read all the appropriate manual pages. It is impossible in the space available to explain the characteristics of all commands used. The intention is to provide a road map that if followed closely should lead to repeatable results.

2 Installing the tools

Installing all the software for the project will take around two hours with a good Internet connection. The procedure is not difficult, but may involve some unfamiliar concepts for Windows users.

2.1 Installing Linux

First of all, a brief justification for using Linux is needed. The concept of free software is attractive and the open source philosophy is compelling. However it may often be worth paying for the features and support provided by commercial software in the interests of efficient use of time. This is particularly true when it comes to the computer operating system. This is, after all, no more than a platform to run programs. A perfect operating system would cause minimum interruption to the work flow by providing a suitably stable environment with a well integrated set of tools. We have previously carried out a great deal of our modelling work in a more or less satisfactory way using GRASS and R in Windows. While R is integral to most of the methods we use, GRASS is not strictly essential. There are now many good GIS programs available at low cost which are sufficiently powerful for most species distribution modelling. For example the Windows program ILWIS has a great deal of useful functionality. ILWIS is now available free of charge and is also open source under the GPL license (http://52north.org/). However GRASS, arguably, remains the most powerful open source raster based GIS. GRASS is fundamentally a UNIX based application. While GRASS runs adequately in Windows using Cygwin, it is far easier to use and integrate with R when it is run in a Linux environment. We found most contemporary Linux distros to be fast, stable, surprisingly user friendly, and above all secure operating systems that are excellent replacements for Windows for every day use[3].

At the time of writing we consider that the most suitable Linux distro for personal use on a laptop to be Ubuntu Feisty 7.04. We reached this conclusion after evaluating SUSE, Slax and a variety of small light weight distros such as DSL and Puppy. Most can be run through emulation under Windows, but the real advantages are found when they are fully installed. The principal virtues of Ubuntu are its simplicity of installation, broad user base, excellent

[1] We were surprised in fact to find it runs considerably faster.

[2] Using LyX is a simple and effective way to ensure consistent type setting for large documents. It is more stable than using MS Word or Open Office when documents run to more than 20 pages. A slight drawback is that some characters do not cut and paste correctly from the resulting PDF document to the R console under Windows. Thus we have also provided an unformated text document with all the code and scripts used.

[3] Both authors of this report have become increasingly frustrated by instability and security issues with Windows operating systems. The first author has switched to Ubuntu for almost all purposes.

on line documentation and an attractive user friendly graphical interface which is well designed for use on PCs and laptops.

In the context of this modelling project an extremely important reason for having Linux, or some other Unix based operating system, available on at least one computer is that most climate scenario data from Global Circulation models is provided in GRIB (machine independent, self descriptive binary format, WMO standard) rather than ASCII grids. This data has to be extracted using specialist software that is compiled and run under Unix. We have successfully installed cdos for this purpose from http://www.mpimet.mpg.de/fileadmin/software/cdo/. We show how this is used in section 5.

Various flavors of the Ubuntu operating system can be obtained free of charge from http://www.ubuntu.com/ and installed from a live CD in around 30 minutes. The minimal requirement is around 6GB of free disk space for the Linux ext3 partition. Installation is usually a very simple process, and provides comparable functionality to Windows in terms of supported devices (printers, scanners, USB, network etc) "out of the box" although we have had some problems obtaining optimal screen resolution and sound on some computers. If problems do arise they can take a while to resolve, but they appear to be rare. However, obviously, repartitioning of a hard disk should not be contemplated without ensuring full data backup first. We will not repeat the installation instructions provided by Ubuntu here, as they are simple to follow.

Once Ubuntu is up and running and connected to the Internet it is worth taking the time to learn the basics of the system from https://help.ubuntu.com/7.04/

2.2 Installing GRASS under Linux

Installing GRASS in Ubuntu is much more straitforward than under Windows. Paste the following lines into the console while logged on as root. The simplest way to run as root is to first install the program "Konsole" using either

```
sudo apt-get install konsole
```

or Add/remove applications .

Then begin a session within a new root shell.

Be extremely careful when running as root! Usually the prefix "sudo" (super user do) provides root privileges for a normal user, but parts of the operating system are owned by root and can only be changed when logged on as the root user. Close the root console as soon as you have finished. Do not type commands that rename or delete files from root unless you are quite sure that you know what you are doing.

```
echo "deb http://les-ejk.cz/ubuntu/ feisty multiverse" > >/etc/apt/sources.list
echo "deb http://us.archive.ubuntu.com/ubuntu/ edgy universe multi-
verse" > >/etc/apt/sources.list
wget -q http://les-ejk.cz/pgp/jachym_cepicky-gpg.pub -O - | sudo apt-key add -
apt-get install grass
```

We will also need the full gdal package including lbgdal-dev in order to use rgdal to read in and use geographical data in R. Gdal is a library with a complete set of functions that other programs can draw on in order to carry out operations such as reprojections and datum shifts. In Windows a compiled version is shipped with rgdal, but gdal must be installed separately in Linux. This can be done most easily using the Synaptic package manager

under system administration. Search for and select gdal-bin, libgdal1-dev, libgdal1-grass. Their dependencies are automatically included.

Alternatively you may paste the following lines into Konsole.

```
sudo apt-get install gdal-bin
sudo apt-get install libgdal1-dev
sudo apt-get install libgdal1-grass
```

2.3 Installing R

Installing R follows a similar route. First include a repository in the list of sources for software, confirm that it is safe by adding the key, then update the repositories and install.[4]

```
echo "deb http://lib.stat.cmu.edu/R/CRAN/bin/linux/ubuntu/ feisty" > > /etc/apt/sources.list
gpg --keyserver subkeys.pgp.net --recv-key E2A11821
gpg -a --export E2A11821 | sudo apt-key add -
apt-get update
apt-get install r-base
apt-get install r-base-dev
```

Notice we need two packages, r-base and r-base-dev. The latter package allows us to build additional R packages from source and install them. We have now taken several steps that are not necessary in windows in order to install software. By this stage it would be understandable to complain that Linux is unnaturally and unnecessarily complicated. The apparent complication comes from the underlying logic of Linux. Windows applications, in general, tend to stand alone in their own folders. There is often a great deal of duplication of functionality between them. This leads to Windows taking up a large amount of disk space. By contrast, Linux is a collection of pieces that fits together like a jigsaw. Because developers read each others code they make extensive reference to diverse libraries. Compilation of source code prior to installation is also a common practice. This means that operations that are complex to achieve in Windows, such as wrapping up some open source C or FORTRAN code within an R package, are comparatively simple. A typical R install in Linux ensures that code libraries are available for use, under the assumption that you might wish to be an R developer as well as a user. Just like a jigsaw, you can become extremely frustrated if a piece is missing. That said, all the pieces are available online. Debian distros such as Ubuntu are especially good at keeping track of them so you usually can find them all. Once everything is in place it is simply a question of relying on the automated updating process in Ubuntu to keep it all up to date. There is no extra work after the initial set up.

Linux is extremely secure. It is difficult to install anything on line that is not from a trusted source. You must type in a password to gain administrative privileges before you can make any changes. This makes Linux a naturally virus free operating system. Also, if you were new to Ubuntu, you should have noticed by now that for routine use the GUI is at least as intuitive and simple to use as Windows. The use of scripts is simply a convenient way of providing information. It is possible to use the GUI for almost every common task.

[4]Both R and GRASS can be installed quickly and painlessly with all their dependencies using the synaptic GUI. However the default behavior of Synaptic is to install old versions, under the assumption that newer versions are unstable. In the case of R this means that new loaded packages won't work.

2.4 Installing R-packages

The R language is extended by the use of installed packages. All installs in Linux require the appropriate privileges the process of installation is slightly more complex than under Windows. It is important to ensure that all package dependencies are installed and compilable. There are occasional frustrations with missing libraries or compilers that require tracking down[5]. This can be a particular problem with spatial packages. To install the R packages we will use we again use the root shell in Konsole. On opening the root shell type "R" to run R and then from within R type:

```
install.packages(c("spgrass","vegan","mgcv","mda","gam","rpart,"RColorBrewer"),dep=T)
```

2.5 Installing maxent and Google Earth

Finally download maxent.jar from http://www.cs.princeton.edu/~schapire/maxent/ and save the file in a directory called "maxent" within your home directory. There is no need to install the program, it should run using the preinstalled version of Java in Ubuntu. If not, install Sun-java version 6 using Synaptics. If your computer has a suitable graphics card installed it can run Google Earth. Download the binary and cd to the directory in which it was saved. It can then be installed with:

```
sudo sh ./GoogleEarthLinux.bin
```

This should complete the installation of all the basic tools. Before starting modelling we need data.

3 Downloading and importing the climate data.

The quality and accuracy of the climatic coverage will have an important effect on modelled results. It is thus important to use a high quality and consistent data source across projects.

3.1 Importing Worldclim data

We have found that for relatively high resolution modeling (the highlands of Chiapas) direct interpolation from locally available climate data is probably the best way of obtaining continuous climate coverages. This has the advantage of allowing direct control over the interpolation process in order to ensure that the effects of elevation can be treated in a way that is consistent with the locally acting processes such as temperature lapse rate and rain shadow effects. However modeling at the larger regional scale requires the use of a global data set. A widely used high quality data set is available from the worldclim site.

http://www.worldclim.org/current.htm

This data has been checked for quality and represents a very valuable resource. We include a brief description of this data set by the authors[14].

> The data layers were generated through interpolation of average monthly climate data from weather stations on a 30 arc-second resolution grid (often referred to as "1 km" resolution). Variables included

[5] We have found that missing dependency problems in general are very rare when using Ubuntu, but care does have to be taken when installing R packages.

are monthly total precipitation, and monthly mean, minimum and maximum temperature, and 19 derived bio-climatic variables.

The WorldClim interpolated climate layers were made using: Major climate databases compiled by the Global Historical Climatology Network (GHCN), the FAO, the WMO, the International Center for Tropical Agriculture (CIAT), R-HYdronet, and a number of additional minor databases for Australia, New Zealand, the Nordic European Countries, Ecuador, Peru, Bolivia, among others. The SRTM elevation database (aggregated to 30 arc-seconds, "1 km") The ANUSPLIN software.

ANUSPLIN is a program for interpolating noisy multi-variate data using thin plate smoothing splines. We used latitude, longitude, and elevation as independent variables. For stations for which we had records for multiple years, we calculated averages for the 1960-90 period. We only used records for which there were at least 10 years of data. In some cases we extended the time period to the 1950-2000 period to include records from areas for which we had few recent records available (e.g., DR Congo) or predominantly recent records (e.g., Amazonia). We started with the data provided by GHCN, because of the high quality of that database. We then added additional stations from other database. Many of these additional databases had mean monthly values, without a specification of the time period. We added these records anyway, to obtain the best possible spatial representation, reasoning that in most cases these records will represent the 1950-2000 time period, and that insufficient capture of spatial variation is likely to be a larger source of error than in high resolution surfaces than than effects climatic change during the past 50 years. After removing stations with errors, our database consisted of precipitation records from 47,554 locations, mean temperature from 24,542 locations, and minimum and maximum temperature for 14,835 locations (see maps below). A set of 'Bioclimatic variables' were derived from the monthly data. Bioclimatic variables are derived from the monthly temperature and rainfall values in order to generate more biologically meaningful variables. These are often used in ecological niche modeling (e.g., BIOCLIM, GARP). The bioclimatic variables represent annual trends (e.g., mean annual temperature, annual precipitation) seasonality (e.g., annual range in temperature and precipitation) and extreme or limiting environmental factors (e.g., temperature of the coldest and warmest month, and precipitation of the wet and dry quarters). A quarter is a period of three months (1/4 of the year). They are coded as follows:

- BIO1 = Annual Mean Temperature
- BIO2 = Mean Diurnal Range (Mean of monthly (max temp - min temp))
- BIO3 = Isothermality (P2/P7) (* 100)
- BIO4 = Temperature Seasonality (standard deviation *100)
- BIO5 = Max Temperature of Warmest Month
- BIO6 = Min Temperature of Coldest Month
- BIO7 = Temperature Annual Range (P5-P6)
- BIO8 = Mean Temperature of Wettest Quarter
- BIO9 = Mean Temperature of Driest Quarter

- BIO10 = Mean Temperature of Warmest Quarter
- BIO11 = Mean Temperature of Coldest Quarter
- BIO12 = Annual Precipitation
- BIO13 = Precipitation of Wettest Month
- BIO14 = Precipitation of Driest Month
- BIO15 = Precipitation Seasonality (Coefficient of Variation)
- BIO16 = Precipitation of Wettest Quarter
- BIO17 = Precipitation of Driest Quarter
- BIO18 = Precipitation of Warmest Quarter
- BIO19 = Precipitation of Coldest Quarter

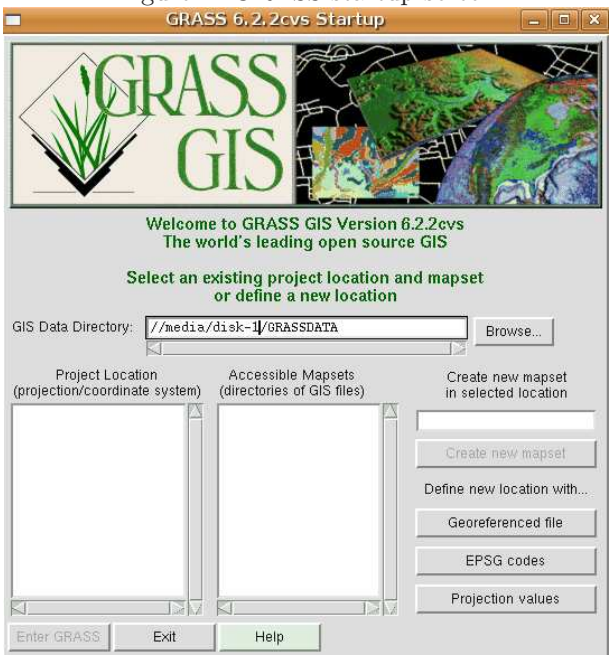
In order to use this data we will take advantage of the power of GRASS to handle extremely large raster coverages through its use of regions. We can import the whole global "1 km^2" data set at once and then easily work on subsets for different regions with different resolutions. This is simpler in GRASS than any other GIS that we know of. It is the feature that makes GRASS particularly well suited for large projects using extensive data sets. GRASS is also extremely rigorous in its use of datum and projection because of the location concept. A location contains a set of coverages that must be in the same projection.

Before starting GRASS you need a suitably large disk space to store the data. We use external removal USB drives for geographical data to avoid filling up the fixed hard drive. Linux does not use drive letters. Instead drives are treated as directories by being mounted at any point in the directory structure. If the drive is not mounted automatically, right clicking the drive shown in "computer" allows you to mount the drive in the default position and it appears on the desktop. This is convenient for unmounting before removal.The first disk receives the name disk, then disk-1, disk-2 etc. In the example here a partition receives the name "disk", so a mounted usb drive is given media/disk-1. Make a directory on the mounted disk either using Nautilus in a similar manner as you would using Explorer in Windows. Alternatively type in the Konsole:

```
mkdir /media/disk-1/GRASSDATA
```

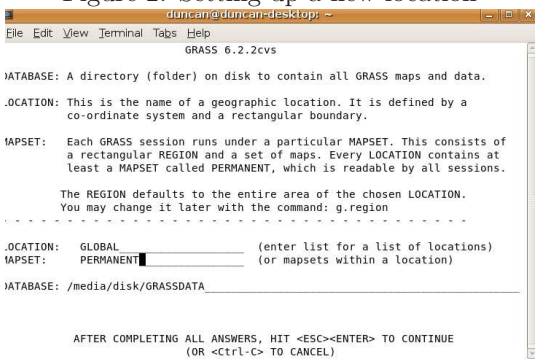
Throughout the document lines marked in red may be changed to reflect differences in the directory structure or mount points between machines. Now start GRASS by typing grass62. Browse to the directory where you will keep all the data. You cannot enter GRASS until you have added the first location. A GRASS location contains a set of data with identical projection parameters. Thus the term "location" might be slightly misleading. In this case our location will be global. Select "Projection values" from the startup screen (Figure 1)

Figure 1: GRASS startup screen



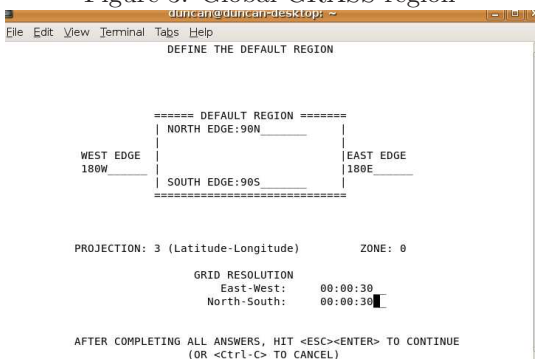
After the startup screen you need to name the Location and Mapset. Each location has a PERMANENT mapset that contains shared data. As the climatic layers are fundamental ingredient for species distribution modelling they should be in the PERMANENT mapset (Figure 2).

Figure 2: Setting up a new location



Once you have entered the location answer the questions forming a location with coordinate system Latitude and Longitude (B), geodetic datum wgs84 and ellipsoid wgs84. Then fill in the global boundaries of the location as in figure 3.

Figure 3: Global GRASS region



Once you are in GRASS all the data that is already imported into the system will come from within the location you used when entering. Moving between locations involves exiting from GRASS and re-entering. This is a very good feature that automatically prevents you attempting to combine data that is incompatible if other locations use different projections and datums. Reprojection involves moving data between locations and reprojecting them.

One concept that might seem strange is that while you are "in" GRASS you are also running a bash shell. This makes all the usual system commands available including running other programs from within GRASS. We use this to mix GRASS and R code later. When importing data into GRASS you can move to the directory where data was stored using the usual cd command. This does not change the fact that you are still working within the same location and database. So importing data usually involves moving to the directory where it was stored then issuing the appropriate commands to bring it into GRASS and reproject if necessary. Gdal is vital to this process.

We assume that the data has been downloaded from http://www.worldclim.org/current.htm in its original format and unzipped to a directory called worldclim. Each of these very large files can then be unzipped to a separate directory and named accordingly. Because of the size of the files this takes some time and requires around 1GB of free space for each unzipped layer. Unless you have a very large hard disk it will be necessary to unzip a set at a time and then delete the unzipped files after importing the data to GRASS. Fortunately the files are recompacted upon importation into GRASS so the final assembled data base on completion is not so large. The following lines import twelve data layers into GRASS that have been extracted into a directory named tmax. This can be done "by hand" using the GRASS GUI, but it is much more convenient to take advantage of the consistent naming of the files and build a very simple shell script as importation can take some time due to the size of the files. Note again that you may need to change the line marked in red to reflect your own choice of mount point and directory.

```
cd /media/disk-1/worldclim/tmax
for i in $(seq 1 12)
do
r.in.bin -s input=tmax_$i.bil output=tmax$i bytes=2 north=90 south=-
60 east=180 west=-180 rows=18000 cols=43200 anull=-9999
done
```

This ability to mix shell commands with GRASS commands is extremely powerful and makes GRASS easier to script than most other GISs. A bonus is that by doing so you learn a syntax which is very useful for automating other tasks in Linux.

The process can be repeated for all the other climate layers.

```
cd /media/disk-1/worldclim/tmin
for i in $(seq 1 12)
do
r.in.bin -s input=tmin_$i.bil output=tmin$i bytes=2 north=90 south=-
60 east=180 west=-180 rows=18000 cols=43200 anull=-9999
done
cd /media/disk-1/worldclim/prec
for i in $(seq 1 12)
do
r.in.bin -s input=prec_$i.bil output=prec$i bytes=2 north=90 south=-
60 east=180 west=-180 rows=18000 cols=43200 anull=-9999
done
cd /media/disk-1/worldclim/bio
for i in $(seq 1 19)
do
r.in.bin -s input=bio_$i.bil output=bio$i bytes=2 north=90 south=-60 east=180 west=-
180 rows=18000 cols=43200 anull=-9999
done
```

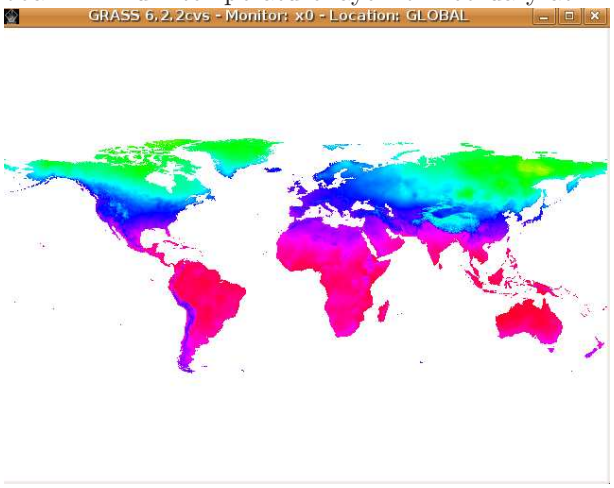
Finally import the elevation data from the same source

```
r.in.bin -s input=alt.bil output=alt bytes=2 north=90 south=-60 east=180 west=-
180 rows=18000 cols=43200 anull=-9999
```

This process will have taken several hours, but you will now have a major resource ready for species distribution modelling anywhere in the world in a convenient format. To visualize a layer either use the GRASS GUI, or open a monitor to obtain the results shown in figure 4.

```
d.mon start=x0
d.mon select=x0
d.rast tmin2
```

Figure 4: Global minimum temperature layer for February at 1km resolution



To select any subset of this data simply use the interactive zoom to set the new region limits.

```
d.zoom
```

To lower the resolution through nearest neighbor (taking the value closest to the centre of the new cell) resampling use

```
g.region res=00:03
```

All the data layers will now be clipped to the new region and use cells of approximately 5km x 5km. This new region will be respected next time you enter GRASS. Be careful when importing data as the default is to only import data that falls within this current region. To reset to the region used for the worldclim coverages use (for example); g.region rast=tmin2. The region concept is one of the great strengths of GRASS as it allows fast and clean subsetting of a very large global data set such as this, but it does require a little experience to get used to the way it works. Notice that you can increase the resolution to greater than the original, but this will obviously not result in any increase in the level of detail. Notice also that GRASS can run complex calculations on these very large raster coverages extremely quickly.

3.2 Importing VCF data

A second useful type of data is derived from satellite imagery. At the large regional scale Modis data is most appropriate. The Vegetation Continuous Fields collection [11, 13] contains proportional estimates for vegetative cover types: woody vegetation, herbaceous vegetation, and bare ground. The product is derived from all seven bands of the MODerate-resolution Imaging Spectroradiometer (MODIS) sensor onboard NASA's Terra satellite.

According to the authors the continuous classification scheme of the VCF product may depict areas of heterogeneous land cover better than traditional discrete classification schemes. While traditional classification schemes indicate where land cover types are concentrated, this VCF product is good for showing how much of a land cover such as "forest" or "grassland" exists anywhere on a land surface.

The version four coverage can be downloaded from

http://glcfapp.umiacs.umd.edu:8080/esdi/index.jsp

The three data files included are percent trees, bare, and herbaceous. This product contains three available layers which add up to represent 100% ground cover. The first layer represents percent tree cover, the second represents percent herbaceous ground cover. and the third layer represents percent bare ground cover. The three layers can be properly displayed in a Red, Green, Blue band combination.

Although it is in Lat Long format the datum and projection (according to the header the datum is not specified but based on an authalic sphere. If imported directly the coordinates do not match those of the WorldClim data so a new location needs to be produced.

```
r.in.gdal input=LatLon.NA.2001.bare.tif output=VCFBare title=bare location=VCF
```

Now exit GRASS and re-enter in the new location called VCF.

```
r.in.gdal input=LatLon.NA.2001.herbaceous.tif output=VCFHerb
r.in.gdal input=LatLon.NA.2001.tree.tif output=VCFTree
```

Now the data can be reprojected within the same location as the WorldClima data.

```
g.region res=00:00:30
r.proj input=VCFHerb location=VCF output=VCFHerb method=nearest
r.proj input=VCFTree location=VCF output=VCFTree method=nearest
r.proj input=VCFBare location=VCF output=VCFBare method=nearest
```

Once these data have been imported into the PERMANENT mapset a new mapset can be produced that is focused on the region for modelling.

Within the mapset the cell resolution can be set to 3 arc minutes to use cell sizes of approximately 5km x 5km.

```
g.region res=00:03
```

4 Species distribution modelling

4.1 Moving climate data from GRASS and visualizing the data

R can be started within GRASS and data moved easily between the two programs using spgrass6. Because R holds all data in memory there are limits to the size of the data that can be be handled at once. This is potentially a drawback of R for handling very large raster layers. However at this resolution there are no problems running R under Ubuntu (or Windows) on a laptop with 1GB of RAM.[6]

1. First enter GRASS and set the region and resolution. Then type R in order to start R. Load the package spgrass6. The complete set of climate coverages with the exception of the derived bioclim layers can be read into memory using:

[6]Some species modelling algorithms could be run in GRASS directly, and GRASS can work with extremely large raster layers held on disk. However R has statistical abilities that are not included in a GIS. The long term solution for working with very large data sets will be to run R in a 64 bit environment with extra memory.

```
library(spgrass6)
a<-paste("tmax",1:12,sep="")
a<-c(a,paste("tmin",1:12,sep=""))
a<-c(a,paste("prec",1:12,sep=""))
clima<-readRAST6(a)
```

We save this data as an R binary object for future use in the working directory.

```
save (clima, file="climaSGDF.rob")
```

The default spatial object is a fully gridded SpatialGridDataFrame, in other words it is in the form of a rectangular matrix with NAs filling the blank regions. For many analysis a better form is a SpatialPixelsDataFrame which only holds cells with values and has a separate component for the cell coordinates. This is more efficient in the case of the long thin shape of the study area. The conversion takes some time so we will run it once and save the results as a separate object. It will then be quicker to move between the two forms by reloading the objects.

```
fullgrid(clima)<-F
save(climaSPDF,rob)
```

The binary objects formed by the save command can be reloaded by R running under either Linux or Windows. So at this stage it is possible to run the rest of the analysis using R under Windows if this is preferred.

We have 36 layers in the object climaSPDF However we will not use all these as they share too much information. This is also the case for the derived bioclim layers which we could have also imported at this stage. We have investigated the properties of bioclim and decided that on balance it is preferable to begin with the easily interpretable rainfall and temperature layers and selectively derive layers from them using R. There are several reasons for this decision.

1. Direct downscaling of climate change scenarios is a much more transparent process if the original monthly means are used as input.

2. Some of the original bioclim layers have rather poor statistical properties due to disjunctions. This is a particular problem with layers such as the precipitation of the warmest month or the temperature of the wettest month. These layers tend to have linear breakpoints, especially in northern Mexico where there is a shift between a Mediterranean type climate with most precipitation falling in the winter months, to a subtropical climate with summer rainfall. While the interpretation of the bioclim layers in terms of factors determining plant growth captures this well, the layers that are produced tend to lead to predictive models with overly sharp boundaries.

3. With the possible exception of the layers mentioned above, bioclim layers are highly correlated and share a great deal of information.

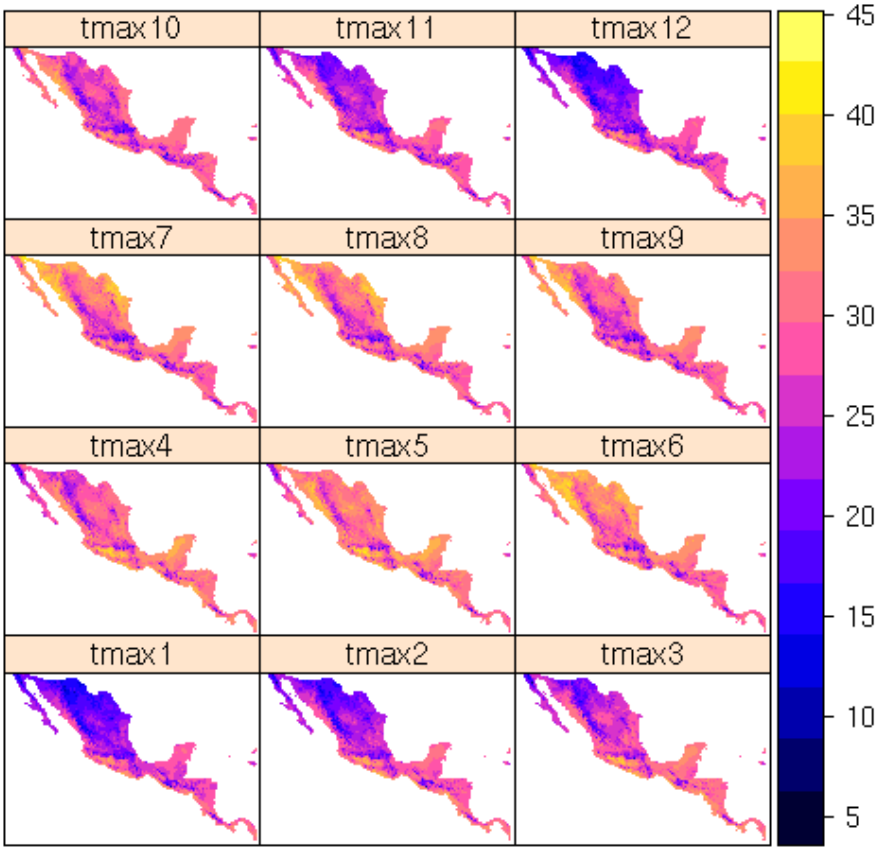
We will now show how to use the power of R to quickly build customized bioclimatic coverages using the data and look at the data as maps in R. One of the nice features is that all the maps can be plotted at once using the same colour scale, something that is not easily achieved using a GIS. We can also carry out extremely fast flexible data manipulation. In this case the temperature data were originally multiplied by ten in order to be stored as integers in the WorldClim database. They can all be changed back with a single line.

```
clima@data[,1:24]<-lapply(clima@data[,1:24],function(x)x/10)
```

Now we can visualize the data.

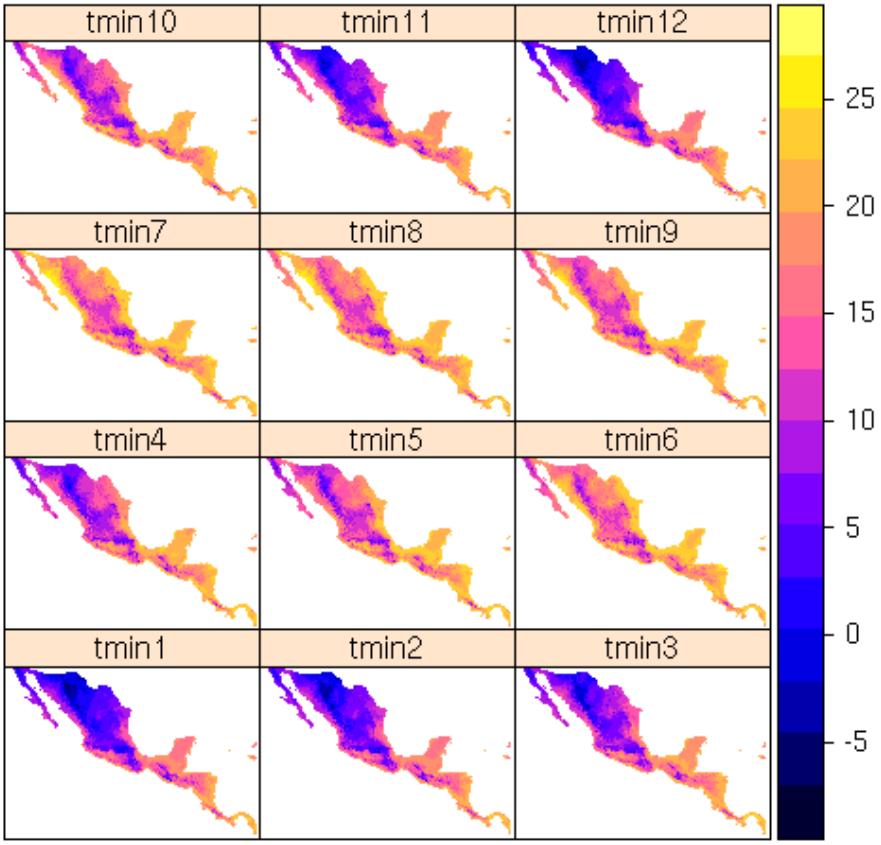
```
png(file="tmax.png")
trellis.par.set(sp.theme())
print(spplot(clima,c(1:12)))
dev.off()
```

Figure 5: Mean monthly maximum temperatures across the study region (degrees C)



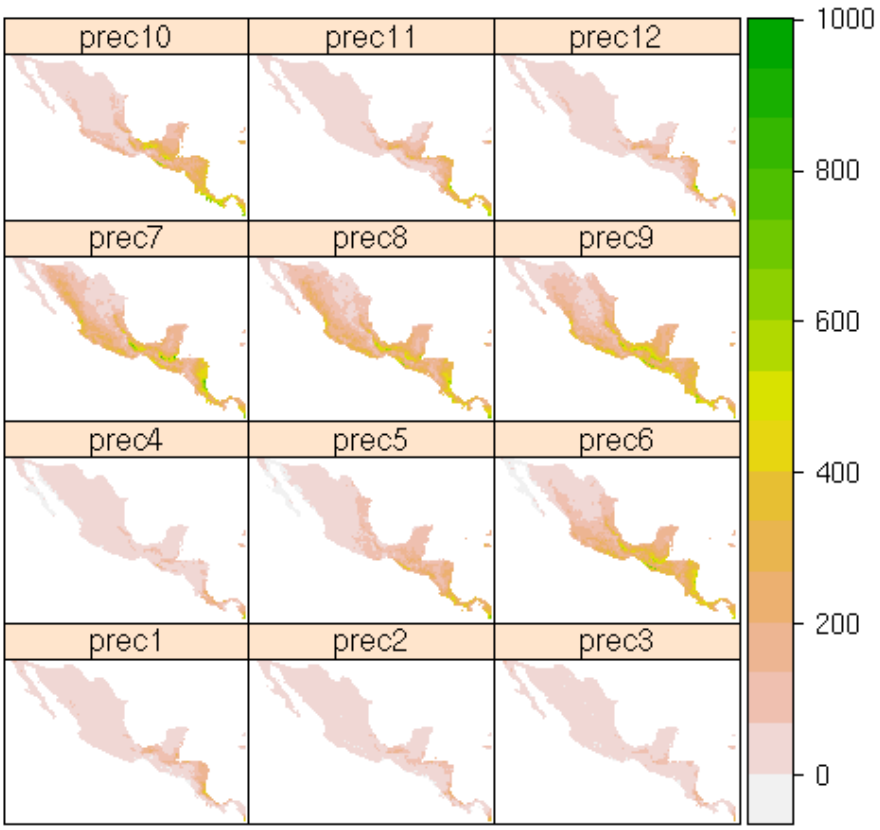

```
png(file="tmin.png")
trellis.par.set(sp.theme())
print(spplot(clima,c(13:24)))
dev.off()
```

Figure 6: Mean monthly minimum temperatures across the study region (degrees C)




```
png(file="prec.png")
trellis.par.set(regions=list(col=terrain.colors(100)[100:1]))
print(spplot(clima,c(25:36)))
dev.off()
```

Figure 7: Mean monthly precipitation across study region (mm)



From figures 5, 6 and 7 we can see that there are fundamentally four elements to the climatic variability over the region. There is a strong latitudinal seasonal effect that is particularly marked in Mexico. Northern Mexico falls above the tropic of cancer and has marked annual temperature fluctuations. This seasonality extends into the sub-tropical zones due to the influence of air movements associated with cold fronts in the winter months. Cold fronts affect the Gulf coast much more strongly than the Pacific Coast.

4.2 Data reduction using PCA: Producing derived climate layers with R

In order to look for ways to reduce the dimensionality of the data it can useful to plot the correlations between layers. We use a customized pairs plot that can be run after the following code has been pasted or sourced into R.

```
panel.line<- function (x, y, col = par("col"), bg = NA, pch = par("pch"),
    cex = 1, ...)
{
    points(x, y, pch = pch, col = col, bg = bg, cex = cex)
```

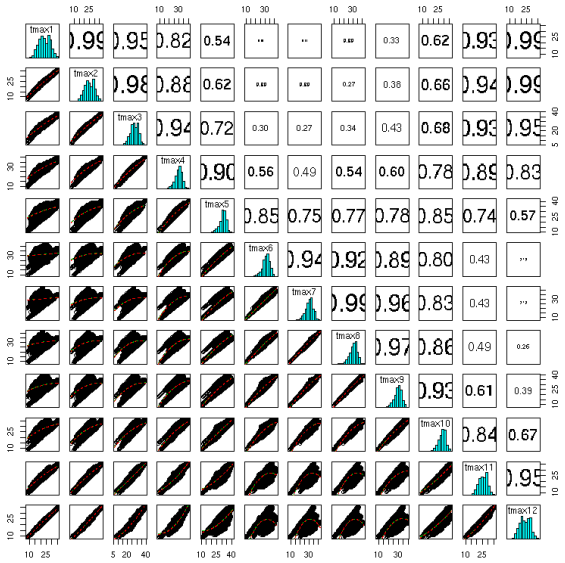
```
    ok <- is.finite(x) & is.finite(y)
    if (any(ok))
       md<-step(lm(y~poly(x,2)))
       xx<- seq(min(x),max(x),length=100)
       yy<-predict(md,data.frame(x=xx),se=T,type="response")
       lines(xx,yy$fit,col=1)
       lines(xx,yy$fit+2*yy$se.fit,col=3,lty=2)
       lines(xx,yy$fit-2*yy$se.fit,col=2,lty=2)

}
  panel.hist <- function(x, ...)
     {
         usr <- par("usr"); on.exit(par(usr))
         par(usr = c(usr[1:2], 0, 1.5) )
         h <- hist(x, plot = FALSE)
         breaks <- h$breaks; nB <- length(breaks)
         y <- h$counts; y <- y/max(y)
         rect(breaks[-nB], 0, breaks[-1], y, col="cyan", ...)
     }
     panel.cor <- function(x, y, digits=2, prefix="", cex.cor)
     {
         usr <- par("usr"); on.exit(par(usr))
         par(usr = c(0, 1, 0, 1))
         r <- abs(cor(x, y))
         txt <- format(c(r, 0.123456789), digits=digits)[1]
         txt <- paste(prefix, txt, sep="")
         if(missing(cex.cor)) cex <- 0.8/strwidth(txt)
         text(0.5, 0.5, txt, cex = cex * r*2)
     }
Xpairs<-function(...)pairs(...,lower.panel=panel.line, up-
per.panel=panel.cor,diag.panel=panel.hist)
```

Now we can look at the correlations between data layers.

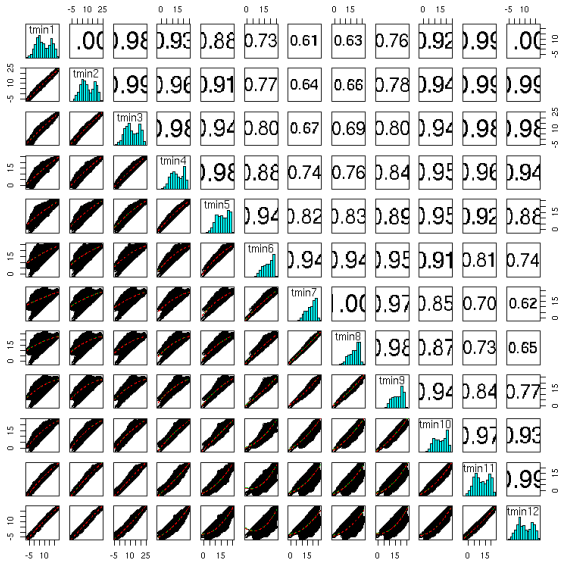
```
Xpairs(clima@data[,1:12])
```

Figure 8: Correlations between maximum temperature layers



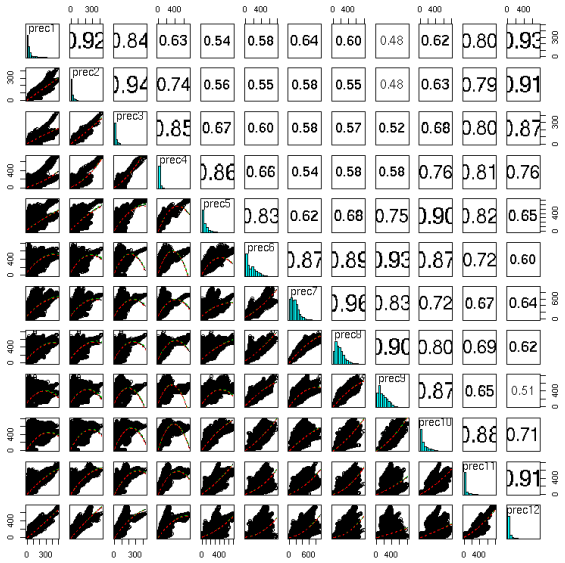

```
Xpairs(clima@data[,13:24])
```

Figure 9: Correlations between Minimum temperature layers




```
Xpairs(clima@data[,25:36])
```

Figure 10: Correlations between precipitation layers



Figures 8,9 and 10 show correlation coefficients between layers in a font size which is proportional to the size of the correlation, allowing a quick impression of the patterns. If the layers are "information rich" correlations between them are small. The greater the correlation the less independent information. Over a small region such as the highlands of Chiapas or even Nicaragua temperatures will tend to be simple linear functions of altitude so all temperature layers will be very highly correlated. In fact part of this correlation can be explained by the use of a lapse rate model or linear regression against elevation in order to derive the layers using either Kriging or Anusplin. At the scale of this data set the layers are more information rich. The highest correlations are between months close together and there seems little to be gained by using more than two to four months.

There is always a certain tension in predictive modeling between the use of predictors with good statistical properties and those that are easily interpretable in terms of processes known to be predictive in an ecological sense. Niche models typically aim to be interpretable in ecological terms. However it can happen that the desire to produce models that use factors that *a priori* influence a species distribution leads to an analyst overlooking the limitations to orthogonality in the data. For example if, over a comparatively small area, warmer temperatures are always associated with moister conditions it will be impossible to separate which factor determines a species distribution from data alone. It is worth analyzing the statistical property of co-linearity using principal components analysis before deciding on the data layers to use.

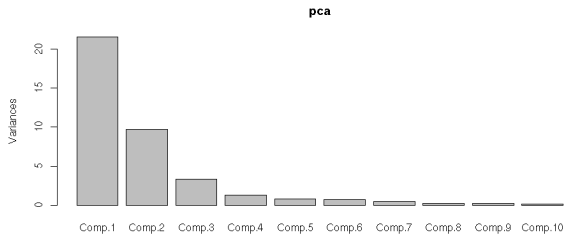
Principal component analysis (PCA) is primarily a dimension-reduction technique, which takes observations on p correlated variables and replaces them by uncorrelated variables. These uncorrelated variables, the principal components (PCs), are linear combinations of the original variables, which successively account for as much as possible of the variation in the original variables. Typically a number m is chosen (m << p) such that using only the first m PCs instead of the p variables will involve only a small loss of variation. The vectors of loadings which define the PCs are known as empirical orthogonal functions (EOFs). One problem with using PCA to replace p variables by m PCs, rather than the alternative strategy of replacing the p variables by a subset of m of the original variables, is interpretation. Each PC is a linear combination of all p variables, and to interpret a PC it is necessary

to decide which variables are important, and which are unimportant, in defining that PC. It is common practice to rotate the EOFs. This idea comes from factor analysis, and proceeds by post-multiplying the (p × m) matrix of PC loadings by a (usually orthogonal) matrix to make the loadings 'simple'. Simplicity is defined by several criteria (e.g. varimax, quartimax) which quantify the idea that loadings should be near zero or near ±1, with as few as possible intermediate values. Rotation takes place within the subspace defined by the first m PCs. Hence all the variation in this subspace is preserved by the rotation, but it is redistributed amongst the rotated components, which no longer have the successive maximization property of the unrotated PCs. What this means in effect is if rotation can produce a pattern of loadings approaching ±1 on each of the variables then the original variables can be used instead of PCs, thus retaining interpretability.

The following R code produces a pca object in R and maps the results onto space as shown in figure 12.

```
pca<-princomp(clima@data,cor=T)
plot(pca)
```

Figure 11: Scree plot of principal components eigenvalues




```
pca.sp<-clima
pca.sp@data<-as.data.frame(pca$scores[,1:4])
trellis.par.set(sp.theme())
spplot(pca.sp)
```

Figure 12: Visualization of the first four principal components axes mapped onto space

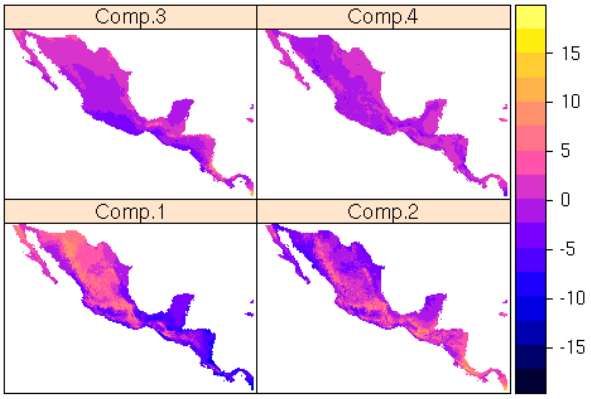


Figure 12 suggests that PCA 1 is perhaps interpretable as an overall index of "tropicality" in a broad sense with orange and red colours suggesting greater seasonality and perhaps cooler winter temperatures. Component 2 seems to be interpretable as capturing mainly variability in precipitation with orange and red marking areas with higher rainfall. The other layers do not seem to have a particularly obvious interpretation. Looking at the loadings after varimax transform helps to clarify what the PCA is capturing.

```
> varimax(pca$loadings[,1:4])
$loadings
Loadings:
               Comp.1 Comp.2 Comp.3 Comp.4
tmax1                                -0.289
tmax2                                -0.301
tmax3                                -0.308
tmax4                 -0.111         -0.307
tmax5                 -0.246         -0.271
tmax6           0.122 -0.360         -0.102
tmax7                 -0.352
tmax8                 -0.337
tmax9                 -0.327
tmax10                -0.259         -0.158
tmax11                               -0.290
tmax12                               -0.284
tmin1          -0.248
tmin2          -0.258
tmin3          -0.267
```

```
tmin4           -0.268
tmin5           -0.251
tmin6           -0.194 -0.175
tmin7           -0.174 -0.246
tmin8           -0.193 -0.236
tmin9           -0.213 -0.182
tmin10          -0.245
tmin11          -0.258
tmin12          -0.250
prec1                          0.363
prec2                          0.383
prec3                          0.367
prec4                          0.286
prec5           -0.142  0.103  0.171
prec6                  0.140  0.164 -0.162
prec7            0.144         0.252 -0.258
prec8            0.122         0.234 -0.266
prec9                  0.123  0.138 -0.189
prec10                         0.203
prec11                         0.294
prec12                         0.364
SeasonalMaxTDif  0.138 -0.267         0.182
DailyTDif        0.417               -0.269
GrowingMonths           0.116  0.183 -0.111
```

Now the interpretation of the third component becomes slightly clearer. While Tabasco and Baja California have radically different climates, they share the fact that some rain falls during the winter. This shows a potential danger of using simple linear combinations of variables for modelling.

We can use the power of R to quickly process data held as spatial objects to hopefully find some more informative climatic layers similar to the bioclim layers. The advantage of writing our own code to do this is that bioclimatic layers can be easily re-derived from downscaled climate change scenarios. It is also worth basing the derived layers at least loosely on the results of the PCA analysis in order to aim for orthogonality. This seems to suggest that maximum orthogonality can be produced by looking at seasonal differences and daily temperature differences. The number of months with more than 100mm of rain might be a very simple way of estimating the length of the growing season. [7]

```
f<-function(x)max(x[1:12])-min(x[1:12])
clima[["AnnualMaxTDif"]]<-apply(clima@data,1,f)
f<-function(x)max(x[1:12]-x[13:24])
clima[["DailyTDif"]]<-apply(clima@data,1,f)
```

[7] More complex methods involve calculating evapotranspiration.

```
f<-function(x)sum(x[25:36]>100)
clima[["GrowingMonths"]]<-apply(clima@data,1,f)
```

We can select some of the key variables identified in the PCA along with the derived variables and form a new clima object.

```
clima2<-clima
clima2@data<-clima2@data[,c(6,13,25,30,37,38,39)]
```

Now we can look at a PCA of the derived variables.

```
pca2<-(princomp(clima2@data,cor=T))
loadings(pca2)
Loadings:
              Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
tmax6         -0.178  0.838 -0.198 -0.284                0.381
tmin1          0.404  0.326 -0.421        -0.266 -0.128 -0.679
prec1          0.326  0.238  0.799 -0.104 -0.372 -0.224
prec6          0.430                -0.352  0.685 -0.468
AnnualMaxTDif -0.401  0.302  0.373  0.210  0.511  0.146 -0.531
DailyTDif     -0.392 -0.199        -0.810 -0.198        -0.332
GrowingMonths  0.447                -0.283  0.146  0.832
```

The loadings look a lot better. In fact upon rotation we can identify a unique interpretation for each axis, so the original variables do indeed possess enough orthogonality to be useful for modelling without being substituted for PCA axes. This allows much clearer interpretation.

```
varimax(loadings(pca2))
Loadings:
              Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
tmax6                  1
tmin1                                                    -1
prec1                         1
prec6                                       1
AnnualMaxTDif -1
DailyTDif                            -1
GrowingMonths                                       1
```

We seem to have found an almost perfect combination of variables that can be used instead of PCs.

4.3 Exporting data to ASCII grid and running maxent

The software for maximum entropy modelling forms a useful tool for species distribution modelling which has a good user interface and seems to outperform the previously most popular software for modelling occurrence only data, GARP, certainly in speed and arguably in predictive ability[16]. Once a subset of climatic layers has been

decided on for modelling exporting the layers to a format that maxent can use is extremely straitforward. Change the first line of the following code to the directory in which maxent.jar was saved.

```
maxentpath<-"/home/duncan/SpeciesModels/maxent"
dir.create(paste(maxentpath,"clima",sep="/"))
for (i in 1:7){
fname<-paste(names(clima2@data)[i],"asc",sep=".")
fname<-paste(maxentpath,"clima",fname,sep="/")
writeAsciiGrid(clima2, fname, attr = i)
}
```

Now we need to load some species occurrence data and overlay it on the climate data. The data is a subset of the total data we are using and has been derived from querying the Missouri Botanical Garden's VAST data base http://mobot.mobot.org/W3T/Search/vast.html using a list of tree species found from southern Mexico to Panama. It can be loaded using

```
d<-read.table("SMesoTrees.txt",header=T)
```

Now we can quickly look at the characteristics of the data with str

```
str(d)
'data.frame':   134618 obs. of  8 variables:
 $ Name     : Factor w/ 3140 lev-
els "Abarema_barbouriana",..: 24 24 24 24 24 24 35 35 35 35 ...
 $ Place    : Factor w/ 159 levels "Belize ","Be-
lize : 0",..: 140 140 140 140 156 156 10 10 10 12 ...
 $ x        : num  -79.9 -79.8 -79.8 -79.9 -79.4 ...
 $ y        : num  9.16 9.15 9.16 9.17 9.40 ...
 $ Year     : Factor w/ 154 lev-
els "1094","1825",..: 115 154 79 81 133 132 120 121 120 102 ...
 $ Political: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ Genus    : Factor w/ 803 levels "Abarema","Abies",..: 4 4 4 4 4 4 5 5 5 5 ...
 $ Species  : Factor w/ 1961 levels "acapul-
cense",..: 1103 1103 1103 1103 1103 1103 542 542 542 542 ...
```

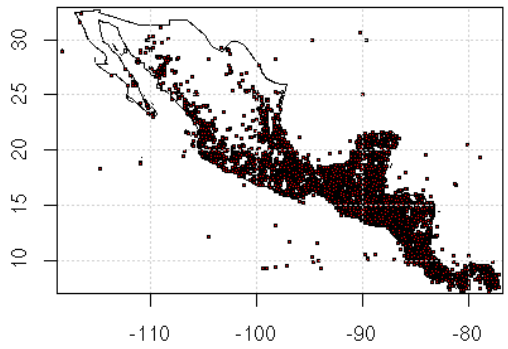
From this we can see that we have 134618 observations on the locations from which 3140 species have been collected. Data with only "Political" locations, i.e. municipalities or states have already been removed from this data so the "Political" column is a legacy. Although there seems to be a lot of information there are serious issues for successful species distribution modelling. One problem is that some of the coordinates are dubious. This can be seen most easily by plotting the points.

```
library(mapdata)
map("worldHires",c("Mexi","Costa Ri","Panama","El Sal-
vad","Hondur","Nicara","guate","Beliz"))
```

```
points(d$x,d$y,pch=21,bg=2,cex=0.4)
box()
axis(1)
axis(2)
grid()
```

Although there are some islands with trees of the coast, a few points are quite obviously errors in figure 13.

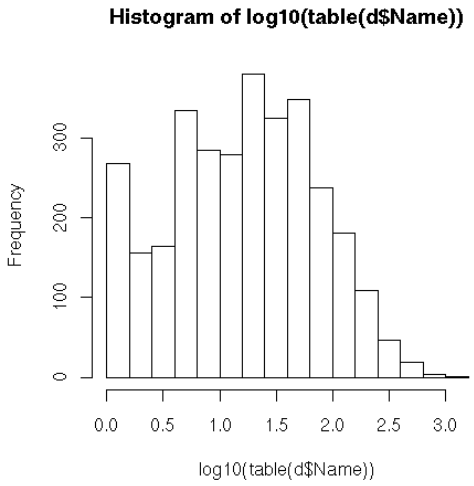
Figure 13: Map of the coordinates of 134618 collections of tree species obtained from MOBOT



We can also look at the distribution in the number of collections per species quickly using the histogram of a table. Table in R produces a table of abundances. If we use the log to the base ten the results are easily interpretable. The distribution is shown in figure 14. Note that the median number of collections per species is 18 and 25% have five or less. Although species distribution maps can be built from a very small number of collections, their validity is very doubtful. Furthermore the fewer the number of data points the higher the possibility of the results being seriously biased by errors in geopositioning of the points.

```
hist(log10(table(d$Name)))
summary(as.vector(table(d$Name)))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    5.00   18.00   42.87   48.25 1027.00
```

Figure 14: Histogram of the log abundances in the collections data



We will now sort the data by abundance and overlay the points on the climate layers.

```
d$Name<-factor(d$Name,names(sort(table(d$Name),d=T)))
d<-d[order(d$Name),]
coordinates(d)<-~x+y
d<-data.frame(d,clima2@data[overlay(clima2,d),])
d<-na.omit(d)
```

This code is concise and there are some rather subtle R tricks that are used here that may not be obvious even for more experienced R users[8]. The first line takes the factor "Name" and places the levels in order with the most frequent species first, the default being for factor levels to be in alphabetical order. The second line then orders the whole data frame according to the factor levels. The third line changes (temporarily) the data frame into a SpatialPointsDataFrame which allows the overlay to work in the fourth line. Then the final line does quite a lot of work all at once. Overlay of clima2 on d produces a numerical array of indices of clima2 on locations of d. This is then used to select rows of the data frame clima2@data[9] which are annexed to the original data frame. The final line removes any data points which fall outside the overlay.

Now we can export some of the species for maxent modelling. First we check the path to the directory containing maxent.jar. There should already be a subdirectory with the climate data. We then select the hundred most abundant species from the list for modelling and form a file called "mostabun.csv" from them. Then we can start up maxent directly from R (under windows use the command "shell" instead of system.)

```
maxentpath<-"/home/duncan/SpeciesModels/maxent"
dir.create(paste(maxentpath,"results1",sep="/"))
```

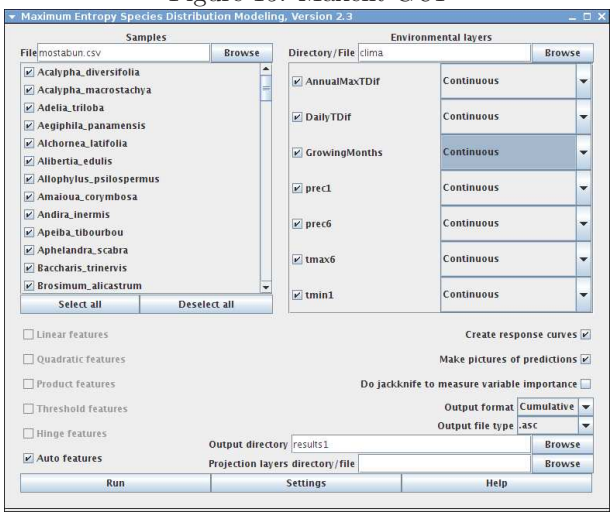
[8]One of the features of the R language is that work that would take a large number of lines of code in other languages can be condensed into a single line. This is particularly true when the "apply" family is used, but can also arise from multiple subsetting of as in this example.

[9]Formally the preferred way to access information held in the slots of S4 classes is through methods rather than extracting "manually" in this way. However we have found that the data method for spatial classes is not always available. For scripting purposes this form of extraction appears to be preferable.

```
nsp<-100
subsetd<-subset(d,d$Name%in%names(sort(table(d$Name),decreasing=T))[1:nsp])
subsetd<-data.frame(name=subsetd$Name,long=subsetd$x,lat=subsetd$y)
write.table(subsetd,sep=",",file=paste(maxentpath,"mostabun.csv",sep="/"),row.names=F)
setwd(maxentpath)
cmd<-"java -jar maxent.jar -K -
P environmentallayers=clima samplesfile='mostabun.csv' outputdirectory=results1"
system(cmd)
```

This should start up maxent as shown in figure 15. Selecting the jacknife measure of variable importance can be useful, but slows the modelling quite considerably so we will first get some output without.

Figure 15: Maxent GUI

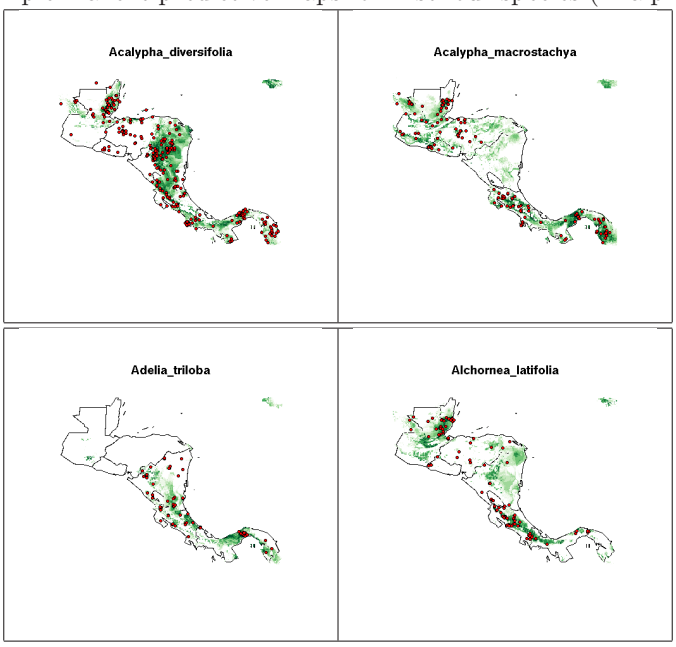


The results can now be looked at and analyzed in the directory setup for them. We can also read the ASCII grids back into R and visualize them there. Make sure the "d" object that was set up previously is available as the points from this are overplotted. Figure 16shows some example results for the first four species in alphabetical order.

```
setwd(paste(maxentpath,"results1",sep="/"))
a<-dir()
a<-a[grep("\\.asc",a)]
library(mapdata)
for (i in 1:10){
aa<-readAsciiGrid(a[i])
map("worldHires",c("Costa Ri","Panama","El Sal-
vad","Hondur","Nicara","Guate","Beliz"))
aa[[1]][aa[[1]]<30]<-NA
image(aa,col=brewer.pal(7,"Greens"),add=T)
s<-subset(d,d$Name==gsub(".asc","",a[i]))
```

```
points(s$x,s$y,pch=21,bg=2,cex=0.5)
title(main=gsub(".asc","",a[i]))
}
```

Figure 16: Example maxent predictive maps for first four species (in alphabetical order)



4.4 Comparing maxent results with GAM and CART (rpart) in R

It is interesting to compare the results from maxent with some of the various other methods that can be implemented in R. R has a very large range of predictive modelling techniques available but we will concentrate here on just two popular methods, Generalized Additive Models (GAMS)[20] and classification trees[19]. These methods have been compared in previous studies (eg [17]). Other effective methods for prediction that are available in R such as neural nets can suffer from the defect of being difficult to interpret.

Generalized additive models (GAMs) are a method of fitting a smooth relationship between two or more variables through a scatter-plot of data points. They are useful when the relationship between the variables is expected to be of a complex form and it is preferred that data, rather than theory, suggests the appropriate functional form. GAMs work by replacing the coefficients found in parametric models by a smoother. Smoothing takes place by local averaging that is averaging the Y-values of observations having predictor values close to a target value. A simple example of a smoother would be a running mean (or moving average). A common smoother used when fitting GAMs in R is the spline. This is a collection of polynomials defined on sub-intervals. A separate polynomial is fitted for each neighborhood, thus enabling the fitted curve to join all of the points. The order of splines is not limited to three, although cubic splines are the most common.

CART analysis as implemented in the r package rpart[19] is a form of binary recursive partitioning. The term "binary" implies that each group of cases, found at a node in a decision tree, can only be split into two groups. However each node can be split into two child nodes and so on, so the term "recursive" refers to the fact that the

binary partitioning process can be applied over and over again. The term "partitioning" refers to the fact that the dataset is split into sections or partitioned. Thus in the case of predicting species modelling we can first find a rule that splits the data into two groups. For example annual rainfall above 1200mm. One side of the split may hold mainly values of presence and the other absence. However the split is not yet an adequate classifier. Too many cases are misclassified. A further split can improve predictive power, for example minimum temperatures in January above 20 C. The aim is to find a model that is simple, with the minimum number of splits, but at the same time with sufficient predictive power to avoid misclassifications. Rpart uses internal cross-validation to decide on an optimum complexity parameter.

All model fitting in R follows a very consistent framework, making it easy to substitute one method for another. When SpatialPixelDataFrames hold the predictor variables model fitting involves.

1. Overlaying the data points on the spatial layers in order to extract the variables associated with them.
2. If only presence has been recorded, producing psuedoabsences through random sampling or some other method.
3. Fitting a model using a generic formula of the type [10] $model <- some.method.for fitting(response\, predictor_1 + predictor_2 ..... predictor_n, AdditionalDetails)$
4. Investigating the properties of the fitted model and running diagnostics.
5. Predicting values for the response variable (in this case presence of a species) from a new set of predictor variables (in this case the raster layers) using the "predict" method that most models possess. The predicted layers can be either the original climate layers if a prediction of current potential distribution is required or layers that have been altered to reflect a potential climate change scenario.

In order to fit the models we need to first setup a storage object for the predictions. We can do this by using the clima2 object as a model and removing the original data. The we must set up data with pseudoabsences for the species we wish to model (in this case we use the first in alphabetical order of the hundred we modelled with maxent[11]).

```
#setup a storage object
preds<-clima2
preds@data<-preds@data[,-c(1:7)]
i<-1
present<-data.frame(resp=1,subset(d[,-c(1:8)],d$Name==gsub(".asc","",a[i])))
pseudoabs<-data.frame(resp=0,clima2@data[sample(1:87525,5000,rep=F),])
resp<-rbind(present,pseudoabs)
mod1<-gam(resp~s(tmax6)+s(tmin1)+s(prec1)+s(prec6)+
s(AnnualMaxTDif)+s(DailyTDif)+s(GrowingMonths),family="binomial",data=resp)
mod2<-rpart(resp~s(tmax6)+s(tmin1)+s(prec1)+s(prec6)+
s(AnnualMaxTDif)+s(DailyTDif)+s(GrowingMonths),data=resp)
preds[["GAM"]]<-predict(mod1,newdata=clima2,type="response")*100
```

[10] Some models can also include interaction terms written as predictor1:predictor2 or predictor1*predictor2.

[11] This leads to a fairly random choice of example drawn from the hundred most abundant species. Thus we have not deliberately chosen an analysis that produces either particularly good or bad results. *Acalypha diversifoli*a is a understorey shrub or very small tree.

```
preds[["Rpart"]]<-predict(mod2,newdata=clima2)*100
aa<-readAsciiGrid(a[i])
fullgrid(aa)<-F
preds[["Maxent"]]<-aa[[1]]
```

The results for southern MesoAmerica are shown in figure 17. The shading for figures of this type may need some adjustment in order to get good visual results. Note that no model that uses pseudoabsences in this way can produce a true "probability of occurrence" in a formal numerical sense. Attempts to give the results this sort of interpretation involve making unjustifiable assumptions regarding the behavior of the collectors responsible for providing the presence data. The key assumption would have to be that psuedoabsence is due to failure to find the species after the same amount of attention has been given to the area in which the species was not found as the areas in which collections were made. This is not reasonable. The result of the model is therefore best interpreted as showing the degree to which the environmental conditions in any given pixel match those where the species has consistently been found. Interestingly in this case we can see that all three models do not produce high values for some areas where a number of collections have been recorded. In this case the notable example are collections from El Salvador. This could be for many reasons, but the most probable are:

1. The collections in El Salvador have been misidentified. There are inevitable problems in species determinations that distribution models may in fact help to resolve by pointing out inconsistencies.

2. Collections have been erroneously georeferenced. We have found this to be disturbingly common even in data that has undergone some prior quality assessment.

3. The species is found in patches with an unusual micro-climate, for example moist canyons or riparian areas within a dry forest zone(e.g.[11]).

4. The models have been over-fitted to the data. The species is in fact widely distributed but collectors have concentrated on a few sites[4].

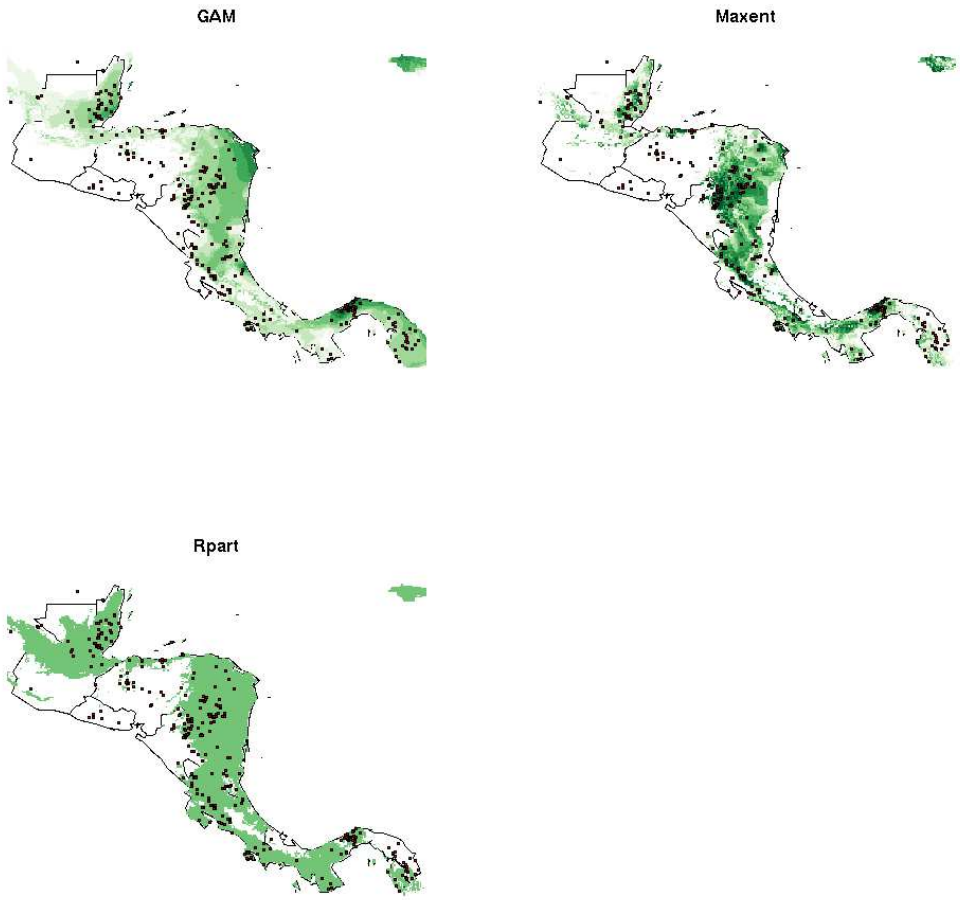
It is difficult to decide between these explanations on the evidence of the maps alone[9, 12, 15, 14]. Modelers charged with producing accurate cartographic products for individual species may have to resort to "expert judgment"[15] at this stage. Clearly one of the weaknesses of expert judgment is that unless the expert taxonomist has very extensive field experience he or she may draw on fundamentally the same data used for modelling. In this case the mental model used to evaluate the maps would suffer from the same type of bias as the maps themselves.

4.5 Investigating the model: Relative importance of the predictor variables

Predictive distribution modelling can "work" in the sense of producing an acceptable cartographic product without producing any insight into the factors working to determine the potential distribution of a species[13, 14]. This can be demonstrated by using simulated artificial climate layers which have no correspondence with any true climate. A model will still be fit by the algorithms and the model can still trace quite accurately the distribution of collection points. However it will clearly be uninterpretable. Thus it is always sensible to look at how the model has been built in some detail.

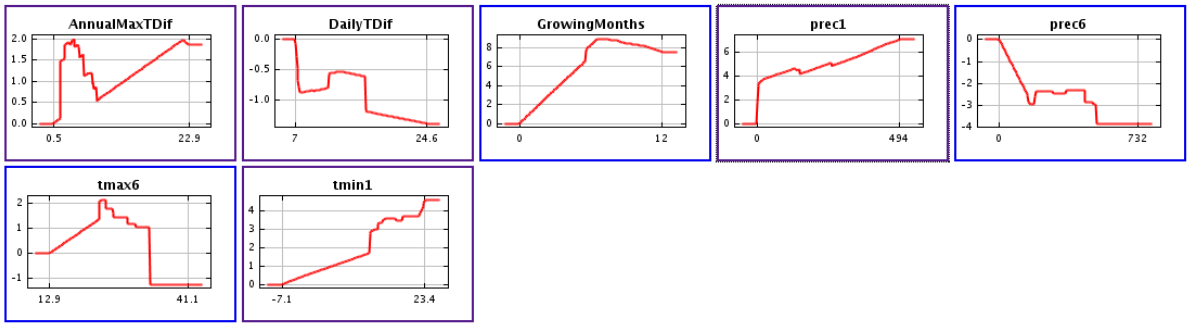
Algorithms such as GARP and maxent are not black boxes[16]. The way in which the model is built is reasonably explicit in the results. However they are not easily communicable. For example maxent uses a combination of linear,

Figure 17: Comparisons between GAM, RPart and maxent predictions for *Acalypha diversifolia* at a regional scale



quadratic, product, threshold and hinge rules. The complexity of these rules found can hinder understanding. Furthermore a complex model will almost inevitably "over-fit" by including some rules that clearly have little biological meaning. For example in figure 18 it is apparent that the up turn in the graph of response to annual maximum temperature difference has is illogical and caused by a statistical artifact. Fortunately this doesn't necessarily affect on the model predictions as each variable reinforces another so errors at the extremes can cancel out. Nevertheless problems in interpretability are certainly a common weakness of complex models.

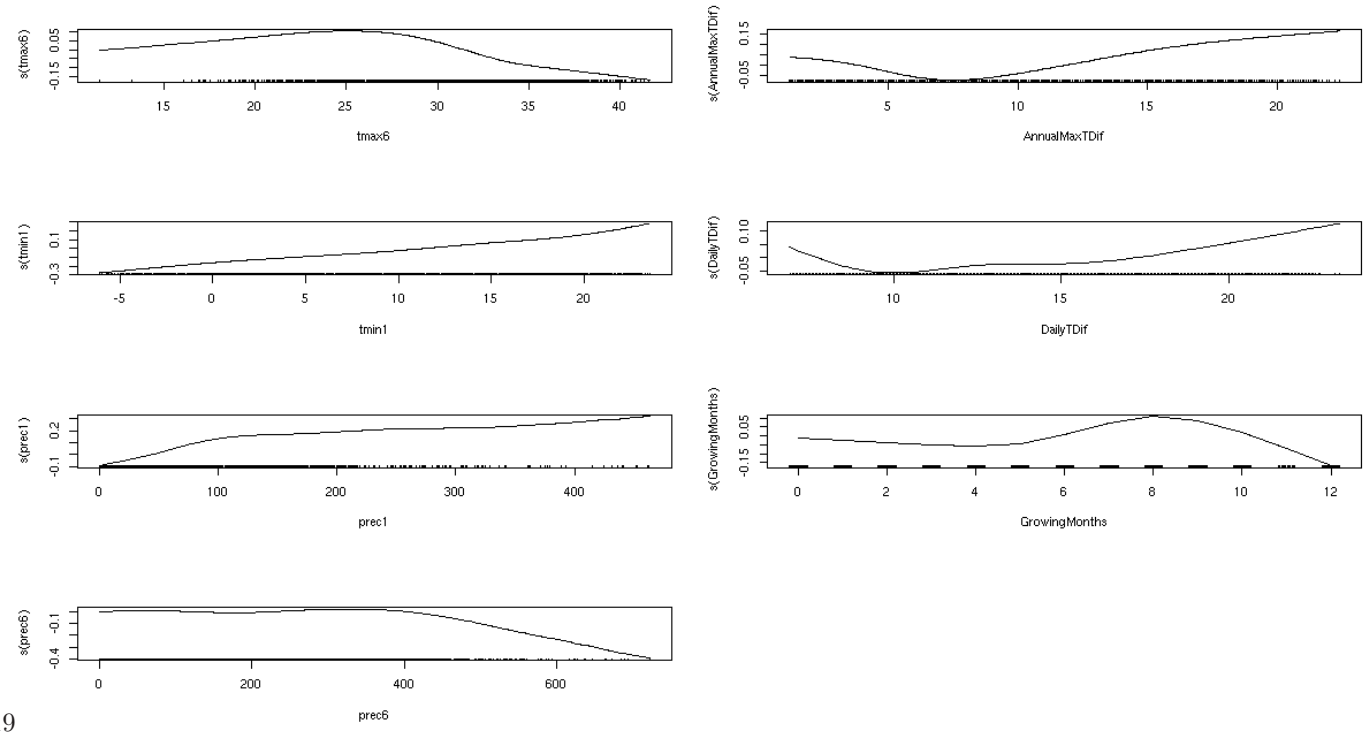
Figure 18: Maxent response curves for each of the variables used in predicting the distribution of *Acalypha diversifolia*



GAMs can have similar undesirable properties. For example the model we fitted can be visualized using:

```
par(mfcol=c(4,2))
plot(mod1)
```

Figure 19: GAM response curves for each of the variables used in predicting the distribution of *Acalypha diversifolia*



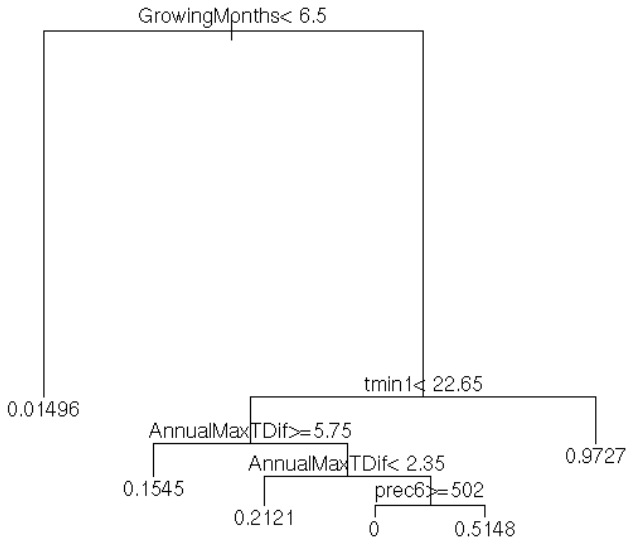
19

The results for the GAM shown in figure 19 suffer from some of the same defects as the maxent models. GAMs can be forced to take simpler shapes by reducing the number of degrees of freedom used for the smoothing parameter. If terms are added in R such as s(tmax1,2) they reduce the smoothed curve to what is effectively a quadratic form. Again in the case of GAMs the additive nature of the model results in terms canceling out. The model predictions

are not necessarily seriously affected by clearly erroneous elements such as the rising trend of the curve with respect to daily temperature differences. These large differences are outside the range predicted for the species by the other terms and are an another artifact of the form used for the curve.

Although the spatial predictions of rpart models tend to suffer from overly sharp cut off points due to the rule based nature of the model, they have a very clear advantage in terms of interpretability. This is enhanced if low complexity parameters are used for fitting. Figure 20 is easy to interpret. At each branching point read of the rule and take the branch to the left to follow it. The leaf shows the proportion of occurrences (out of the total data set that includes 5000 pseudo absences) with the eventual combination of rules. Thus we can conclude that more than six months with precipitation of over 100 mm seem to be a condition for the presence of *Acalypha diversifolia*. Within the region with this property it is more likely to occur if the annual range of maximum temperature is below 5.7 C. Again at the extreme tips of the branches the model suffers from slight over-fitting, but this can be ignored. *Acalypha diversifolia* is clearly a species that prefers moist tropical conditions.

Figure 20: Rpart classification tree for *Acalypha diversifolia*



It is possible to visualize model results easily in Google Earth using a small customized R script that can be pasted into the console.

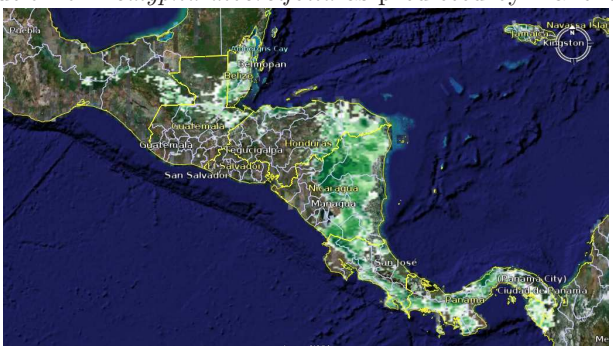
```
toGE<-function(Gr=d,Layer=1,pal,nm="test.kml",imagenm="image1.png"){
png(imagenm,bg='transparent')
par(mar=c(0,0,0,0))
a<-Gr[[Layer]]
dim(a)<-c(Gr@grid@cells.dim[1],Gr@grid@cells.dim[2])
```

```
a<-a[,Gr@grid@cells.dim[2]:1]
image(a,col=pal)
dev.off()
X<-t(Gr@bbox)
X<-as.data.frame(X)
names(X)<-c("x","y")
coordinates(X)<-c("x","y")
X@proj4string<-Gr@proj4string
X<-spTransform(X,CRS("+proj=longlat +datum=WGS84"))
X<-as.data.frame(X)
N<-X[2,2]
S<-X[1,2]
E<-X[2,1]
W<-X[1,1]
fl<-imagenm
kmlheader<-c("<?xml version='1.0' encoding='UTF-8'?>","<kml
xmlns='http://earth.google.com/kml/2.0'>","<GroundOverlay>")
kmname<-paste("<name>",nm,"</name>",sep="")
icon<-paste("<Icon><href>",fl,"</href><viewBoundScale>0.75</viewBoundScale>
</Icon>",sep="")
latlonbox<-paste("<LatLonBox><north>",N,"</north><south>"
,S,"</south><east>",E,"</east><west>",W,"</west></LatLonBox>",sep="")
footer<-"</GroundOverlay></kml>"
x<-(kmlheader)
x<-append(x,kmname)
x<-append(x,icon)
x<-append(x,latlonbox)
x<-append(x,footer)
write.table(x,nm,quote=F,append=F,col.name=F,row.name=F)
}
```

After this function has been made available the kml file visualized in figure 21can be produced by typing:

```
fullgrid(preds)<-T
toGE(preds,3,pal=brewer.pal(7,"Greens"),"Acalypha_diversifolia.kml","maxent.png")
```

Figure 21: Potential distribution of *Acalypha diversifolia* as predicted by maxent visualized in Google Earth



4.6 Combining species distribution modelling with multivariate analysis of species association with climate: The CASP approach

Our experience with modelling individual species distributions convinced us that such models are extremely useful. However they are not suited to all applications. They are also extremely data demanding. The data available for tropical regions are typically weak or even erroneous, and there are problems associated with a "garbage in - garbage out" syndrome.

A particular concern is that the concept of biodiversity as used in a conservation setting is closely linked to the distribution of rare organisms. There is a real need to produce maps of potential distribution for these species, which can be the most sensitive to climate change and habitat destruction. However a rare species almost inevitably has few recorded presences. Although small data sets can be fed into a distribution model, the results will lack validity from both a statistical and evidence based perspective. Statistically they suffer from over-fitting, especially if the limited number of data points are spatially auto-correlated[18, 4]. From an evidence based perspective we must ask if lack of reported occurrence can safely be attributed to true of occurrence. In many cases the answer appears to be no. There are multiple other viable explanations for lack of reporting or lack of collection. Weak evidence can only be marginally strengthened though improved statistical methodology. In one sense the options available are limited.

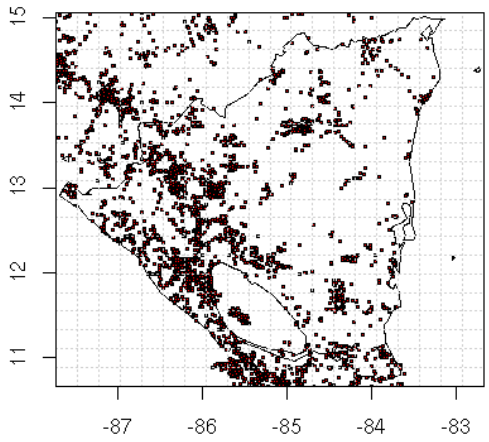
However, even when most species are rare, patterns can emerge at the level of associations between species. A "chain" of species occurrences can occur along a gradient which can in some sense allow the weak data for rare species to borrow strength from the better know occurrences of common species. If a rare species has only been recorded once, but this occurrence coincides with a large group of species whose collective range is somewhat limited, a map of the collective range may be suggestive of the potential range for the rare species. This method cannot turn bad data into good, but it can provide a practical method for working with the sort of data available.

One simple and potentially practical way that this could be achieved would be use existing maps produced which show eco-regions. As the environmental conditions within eco-regions are assumed to be comparatively homogeneous then they may define the potential limits to species potential distributions. The main weakness with this approach is that it assumes that eco regions truly reflect breakpoints in the distribution of associations of species. However the species distributions themselves have not been formally used when eco-regions are defined. Instead the process relies on expert judgment.

Although we do not discount the value of eco-regions, we have been interested in finding methods that allow the data on species distributions themselves to define them. This led us to combine methods used in community ecology with those used by bio-geographers. Because of the extensive scale at which these models are applied we use the term "Climatically Associated Species Pools" [7]to describe the resulting groupings, rather than communities.[6]

The first step in the process is to divide the space to be modelled into what we might term "pseudoquadrats". The size of the quadrats will determine the number of co-occurring species found. There is a trade off here. While the method requires all the quadrats which will be used to have more than 4-5 species, large quadrats will allow for too much heterogeneity in climate within them[10]. This is a particular problem in mountainous areas with abrupt topography. For example the large quadrats shown in figure 22 illustrate the concept, but we would use a finer grain for modelling. In fact many collection points have multiple collections recorded from exactly the same coordinates, so the aim of finding species that co-occur can be met with quite a fine grid.

Figure 22: Overlaying large (10') quadrats on a the collection data for Nicaragua.



We have broken the R code used to produce CASPS into blocks. The first block that places the species into quadrats is slightly involved, but the reader does not have to follow the details to use it. Note that decision points have been marked in green and code that is specific to a particular computer in red. The object "SMesoClimaSPD.rob" contains a subset of the "clima2" data set derived earlier.

```
library(spgrass6)
library(vegan)
library(mda)
library(mapdata)
setwd("/home/duncan/SpeciesModels/RData/CASP")
load("SMesoClimaSPD.rob" )
d<-read.table("SMesoTrees.txt",header=T)
```

```
#Decide on the size for the grid in degrees
###############
gridsize<-0.05 #
###############
#Set up an xseq using the limits for the overlay layers
xseq<-seq(clima2@bbox[1,1],clima2@bbox[1,2],by=gridsize)
#cut up the coordinates according to the sequence
gridlong<-as.numeric(cut(d$x,xseq))
#Turn the cut levels into x coordinate
gridlong<-clima2@grid@cellcentre.offset[1]+gridlong*gridsize
#Repeat the procedure for ys
yseq<-seq(clima2@bbox[2,1],clima2@bbox[2,2],by=gridsize)
gridlat<-as.numeric(cut(d$y,yseq))
gridlat<-clima2@grid@cellcentre.offset[2]+gridlat*gridsize
gridded.sp<-data.frame(x=gridlong,y=gridlat,sp=d$Name)
gridded.sp<-na.omit(gridded.sp)
coordinates(gridded.sp)<-c("x","y")
gridded.sp<-gridded.sp[!is.na(overlay(clima2,gridded.sp)),]
image(clima2,1,col=terrain.colors(100))
points(gridded.sp)
#Now set up the gridded.sp object as a data frame with cell codes
gridded.sp<-data.frame(x=gridded.sp@coords[,1],
y=gridded.sp@coords[,2],
code=paste(gridded.sp@coords[,1],gridded.sp@coords[,2],sep=" ")
,sp=gridded.sp[["sp"]])
str(gridded.sp)
#End of step 1.
#We now have a set of species referenced to grid cells, with all those falling
#Outside the mapped study region removed.
#This is passed on to step 2.
```

The next step in the analysis involves deciding which of the quadrats to use. Multivariate analysis breaks down (giving eigenvalues of one) if there are complete disjunctions between sets of species. It is also impossible to analyze data that contains quadrats with only one occurrence. Thus after forming a vegetation matrix of species on sites we remove the species with less than n occurrences and sites with less than j species. We can set n to 50 and j to 5 in this example. We point out that this does not mean that we will completely ignore rarer species in the analysis. They are added back in the final stage. However they are not useful for defining the key relationships between species groups and climate.

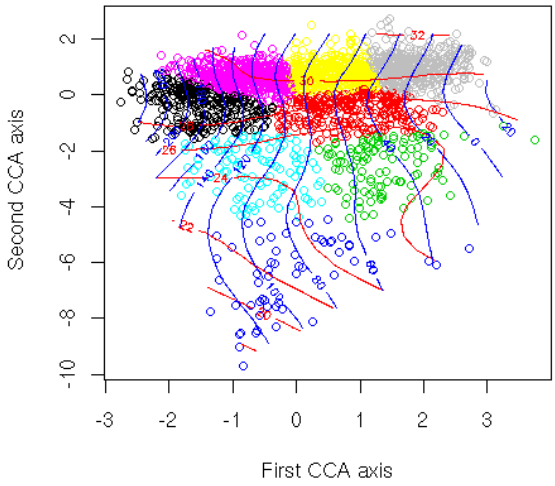
```
#A key move is to set up a vegetation matrix (sp by sites) for vegan analysis
vegtable<-table(gridded.sp$code,gridded.sp$sp)
#But it is necessary to clean out very rare species and
#quadrats with insufficient species occurrences.
```

```
####################
min.occurrences<-50  #
min.nsp<-5           #
####################
f<-function(x)sum(x>0)
vegtable<-vegtable[,apply(vegtable,2,f)>min.occurrences]
vegtable<-vegtable[apply(vegtable,1,sum)>min.nsp,]
#A trick to get the coordinates of the cells back from the row names
# of the table.
coords<-do.call("rbind",strsplit(row.names(vegtable),split=" "))
coords<-data.frame(x=as.numeric(coords[,1]),y=as.numeric(coords[,2]))
cells<-data.frame(coords,ncollects=log2(apply(vegtable,1,sum)))
coordinates(cells)<-c("x","y")
```

Now we can carry out multivariate analysis using the R package vegan[21]. In this example we use canonical correspondence analysis using all the seven climate layers. Sites are then grouped using k means clustering on the ordination axis and colour coded. We have plotted two of the main climate variables (tmax6 and prec1) as contours on the diagram shown in figure 23.

```
#This is the ordination and k means clustering step
#There are a lot of important decisions here that require some understanding
#of the method in order to implement effectively.
#A simple choice is how many groups.
########
k<-8 #
########
env<-clima2@data[overlay(clima2,cells),]
vegtable<-as.data.frame(vegtable[apply(env,1,function(x)!any(is.na(x))),])
cells<-cells[apply(env,1,function(x)!any(is.na(x))),]
env<-env[apply(env,1,function(x)!any(is.na(x))),]
cca1<-cca(vegtable~.,env)
site.scores<-scores(cca1,display="sites",c(1:2))
set.seed(1)
cells[["kgroups"]]<-as.factor(kmeans(site.scores,k)$cluster)
plot(site.scores[,1],site.scores[,2],
xlab="First CCA axis", ylab="Second CCA axis",
main="",cex=0.8,col=as.numeric(cells[["kgroups"]]))
ordisurf(cca1, env[,1],main="",col="red",add=T)
ordisurf(cca1, env[,3], xlab="", ylab="",col="blue",lty=2,add=T)
```

Figure 23: Ordination diagram for CCA with climatic contours (tmax6 and prec1)

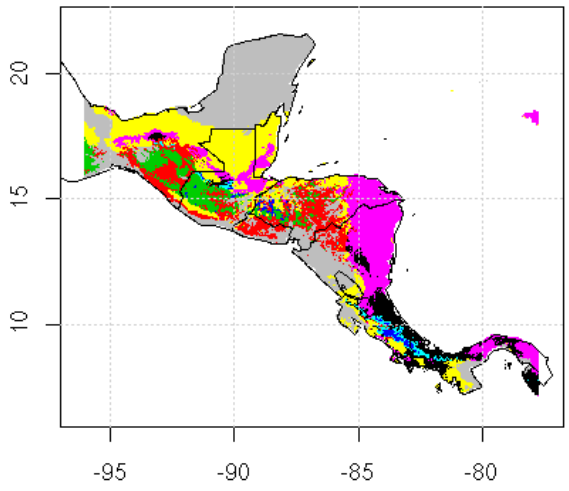


Once the ordination and k means clustering have been carried out the groups are treated as if they were species and their distribution mapped. However because the distribution of groups (Climatically Associated Species Pools) is mutually exclusive discriminant analysis is a suitable method for prediction. To avoid misunderstandings we stress that it is the distribution of the lists of species, not the distribution of a given species that is mutually exclusive. A common species could be found in all the CASPs.

```
#The discriminant analysis phase.
pred.mod<-mda(cells[["kgroups"]]~.,data=env)
#Set up the spatial pixels predictors......
#...and predict onto the map.

clima2[["grps"]]<-as.numeric(predict(pred.mod,newdata=clima2@data))
image(clima2,"grps",col=1:8)
map("worldHires",c("mexico","Costa Ri","Panama","El Sal-
vad","Hondur","Nicara","Guate","Beliz"),add=T)
box()
axis(1)
axis(2)
grid()
```

Figure 24: Predicted distribution of Climatically Associated Species Pools



The final step is to find which species belong to the CASPs. This is a simple overlay followed by some steps to produce a tabulation.

```
#Now cross the groups with the gridded species.
coordinates(gridded.sp)<-~x+y
temp<-clima2@data$grps[overlay(clima2,gridded.sp)]
temp<-data.frame(Name=gridded.sp@data$sp,Group=temp)
temp<-table(temp)
class(temp)<-"matrix"
abundance<-as.data.frame(temp)
```

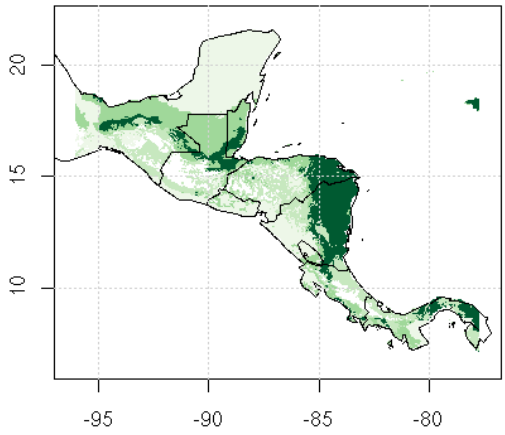
Figure 25: Part of the table of abundances of collections of species within the area of each CASP

	row.names	1	2	3	4	5	6	7	8
1	Abarema_barbouriana	1	0	0	0	0	2	0	0
2	Abarema_macradenia	3	0	0	0	0	11	1	0
3	Abies_guatemalensis	0	0	7	2	0	0	0	1
4	Abutilon_giganteum	0	0	1	0	0	0	0	1
5	Abutilon_purpusii	0	0	1	0	0	0	0	0
6	Abutilon_striatum	0	0	4	0	0	0	0	1
7	Acacia_acatlensis	0	1	0	0	0	0	0	1
8	Acacia_angustissima	1	16	21	2	2	2	13	27
9	Acacia_auriculiformis	0	0	0	0	0	2	0	0
10	Acacia_centralis	0	5	1	1	1	0	5	43
11	Acacia_chiapensis	0	0	0	0	0	2	0	1
12	Acacia_cochliacantha	0	3	2	0	0	0	0	3
13	Acacia_collinsii	1	13	1	0	0	13	21	89
14	Acacia_cookii	1	1	0	0	0	3	2	2
15	Acacia_cornigera	0	6	2	0	0	0	17	21
16	Acacia_farnesiana	0	10	4	0	0	1	5	53
17	Acacia_filicoides	0	0	0	0	0	0	0	0
18	Acacia_gentlei	0	0	0	0	0	6	29	4
19	Acacia_globulifera	0	3	1	0	0	0	4	9
20	Acacia_glomerosa	0	1	3	1	0	3	10	8
21	Acacia_hindsii	0	13	7	0	0	3	4	12
22	Acacia_macracantha	0	0	0	0	0	0	0	0
23	Acacia_mayana	0	0	0	0	0	1	1	0
24	Acacia_melanoceras	4	0	0	0	0	10	0	0
25	Acacia_pennatula	0	33	4	0	0	2	3	29

Once the species pools have been defined in this way there are many possibilities for further analysis. They can be used to produce potential distribution maps for each species by giving stronger colours to CASPS in which a species is more abundant.

```
q<-abundance[grep("Acalypha_d",row.names(abundance)),]
q<-as.vector(unlist(q))
clima2[["temp"]]<-q[clima2@data$grps]
image(clima2,"temp",col=c("white",brewer.pal(7,"Greens")))
map("worldHires",c("mexico","Costa Ri","Panama","El Sal-
vad","Hondur","Nicara","Guate","Beliz"),add=T)
axis(1)
axis(2)
box()
grid()
```

Figure 26: Predicted potential distribution for *Acalypha diversifolia* based on occurences within CASPS

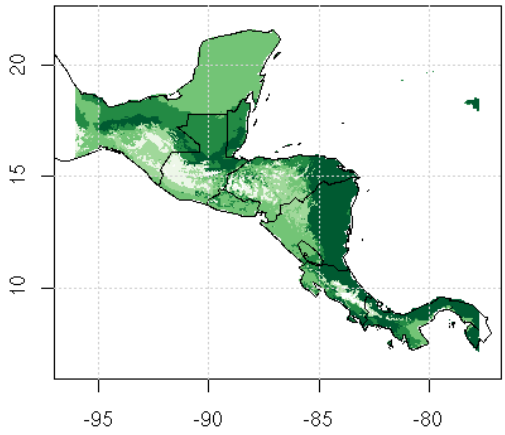


Note that the predicted distribution for this species shown in figure 26using this methodology is very similar to that suggested by modelling at the individual species level. We have found that this correspondence between results is general. However poor data leads to artifacts whatever approach is used.

Another simple but interesting analysis is to count the number of species occurring in each CASP as a simple measure of species richness. A variation on the theme is to only include species with more than n occurrences in order to reduce bias through errors in georeferencing or determination.

```
f<-function(x)sum(x>4)
div<-apply(abundance,2,f)
clima2[["temp"]]<-div[clima2@data$grps]
image(clima2,"temp",col=c("white",brewer.pal(7,"Greens")))
map("worldHires",c("mexico","Costa Ri","Panama","El Sal-
vad","Hondur","Nicara","Guate","Beliz"),add=T)
axis(1)
axis(2)
box()
grid()
#Number of species with more than four occurences in each CASP
div
   1    2    3    4    5    6    7    8
1094  552  388  154  432  993  869  664
```

Figure 27: Species richness as suggested by CASP analysis



The result of this shown in figure 27 confirms the fact that woody plant (regional or gamma) diversity is generally highest in the warm moist tropics and lower in drier cooler areas. One of the most useful outcomes of the analysis is that this observation is clearly linked to the underlying data and although some inevitable errors remain, at a rather coarse scale potential distributions can be suggested for most species in the data base.

5 Modelling climate change scenarios

5.1 Extracting WMO GRIB formatted data

Data from a number of global circulation models parametrized for IPCC scenarios are available from

http://www.mad.zmaw.de/IPCC_DDC/html/SRES_AR4/index.html.

Although this page mentions that data are available in ascii format, this does not mean the ascii grids that are used in GIS. Output from climate models and large historical climatic data are usually held in either GRIB or netCDF format. These formats have been specifically designed by the WMO (World Meteorological Organization) for efficient, compact storage of huge volumes of information. At the time of writing, as far as we know, these formats are not directly readable either by any GIS or statistical software in general use. Research groups that work on climate modelling and monitoring have developed their own specialized tools for handling data in these formats. It is therefore essential to obtain, install and learn to use these tools in order to have full access to this essential source of input for species distribution modelling. Although tools for working with GRIB and netCDF have not been designed with "user friendly" GUIs, they do fit easily into the scripting framework that we use in this document. At the time of writing the tool that appears to have the best documentation, ease of to installation and general simplicity of use is CDO (Climate Data Operators). The program can be downloaded as source code from http://www.mpimet.mpg.de/fileadmin/software/cdo/. This software offers a very strait-forward, powerful command line interface that can be used for a large number of pre-processing and processing tasks. In this document

we use CDO to extract data in a usable form for further work in R and GRASS.

5.2 Installing CDO

We do point out that it is possible to build CDO under Cygwin within a Windows environment if this is necessary. It is a complicated procedure. As with all UNIX based tools installation is a much simpler process on a machine running some form of Linux. We have found that CDO builds from source cleanly under Ubuntu 7.04. The steps used to install CDO are typical for any program that is distributed as source code. Using Konsole cd to the directory containing the package's source code and type

```
sudo ./configure
```

to configure the package for your system. Then type

```
make
```

to compile the package. Finally

```
make install
```

to install the programs and any data files and documentation. Now in a console typing

```
cdo
```

should produce output detailing the operators available if CDO installed correctly. If you get output then CDO is installed and ready for use. Notice that although CDO is an extremely powerful program with a large number of operators, it does not have a graphical user interface or any form of directly visualizing output in the form of maps. This is irrelevant for our purposes as we use GRASS and R to achieve the visualization.

5.3 Obtaining the data

The data from the GCM scenarios can be download from the CERA portal http://cera-www.dkrz.de/ after registering. As processing output from a GCM is a moderately time consuming affair, an initial challenge is choosing which output is most suitable from the large number of possibilities on offer. We have previously based our models largely on HADCM3 scenarios. At the time of writing HADCM3 output is not available for all the IPCC AR4 A1 story line. There are outputs for the A2 and B2 scenarios and we will use these in this document. We will use the following files derived from the A2 and B2 story-lines that are compatible with the methods used for processing WorldClim data.

```
HADCM3_A2_prec_1-1800.grb
HADCM3_A2_tmin_1-1800.grb
HADCM3_A2_tmax_1-1800.grb
HADCM3_B2_tmax_1-1800.grb
HADCM3_B2_prec_1-1800.grb
HADCM3_B2_tmin_1-1800.grb
```

One the files are downloaded into a suitable directory CDO can be used to provide information on the data stored within these files. For example:

```
cdo griddes HADCM3_A2_tmin_1-1800.grb
gridtype : lonlat
gridsize : 7008
xname : lon
xlongname : longitude
xunits : degrees_east
yname : lat
ylongname : latitude
yunits : degrees_north
xsize : 96
ysize : 73
xfirst : 0
xinc : 3.75
yfirst : 90
yinc : -2.5
```

Or typing:

```
cdo info HADCM3_B2_tmax_1-1800.grb
```

which produces a lot of output on the information provided for each of the set of grids in the file. From this output it should be clear that each file contains 180 grids, one for each month between January 1950 and December 2099. The first set of results are climate data and the later results are model output.

5.4 Extracting fifty year monthly means

Our simulated data should be consistent with the data incorporated in the models we used for current distribution predictions. WordClim monthly means are derived from fifty years of climate data. It is therfore appropriate to use fifty year blocks of modelled data and find the mean monthly values. We will use two commands in cdo to do this. The first (splitsel) splits the data into blocks of fifty years and the second (ymonmean) calculates the monthly means over the years held in each of the files. Notice that a temporary file is produced for each split and then removed at the end of the process. As an aside, it should be clear from this small example that CDO is a fast efficient tool for data processing of great value to professional climatologists working on a daily basis with data of this complexity.

```
cdo splitsel,600 HADCM3_A2_tmin_1-1800.grb tmp1
cdo splitsel,600 HADCM3_A2_tmax_1-1800.grb tmp2
cdo splitsel,600 HADCM3_A2_prec_1-1800.grb tmp3
for i in $(seq 0 2)
do
cdo ymonmean tmp100$i.grb HADCM3_A2_tmin_50_$i.grb
```

```
cdo ymonmean tmp200$i.grb HADCM3_A2_tmax_50_$i.grb
cdo ymonmean tmp300$i.grb HADCM3_A2_prec_50_$i.grb
done
rm tmp*
```

If you now investigate the results using **cdo info** you will see that each file holds twelve global grids with the fifty year mean monthly values for the appropriate variable. We can reduce the data layers to two future scenarios for each variable by subtracting the first fifty years data from each of the subsequent periods. This gives us a set of layers showing the expected mean change under the model scenarios. These layers are thus ready for a simple downscaling procedure achieved by by adding the results to the much finer resolution WorlClim layers. Again we use a small loop in a shell script to reduce typing slightly.

```
for i in $(seq 1 2)
do
cdo sub HADCM3_A2_prec_50_$i.grb HADCM3_A2_prec_50_0.grb HADCM3_A2_prec_dif_$i.grb
cdo sub HADCM3_A2_tmin_50_$i.grb HADCM3_A2_tmin_50_0.grb HADCM3_A2_tmin_dif_$i.grb
cdo sub HADCM3_A2_tmax_50_$i.grb HADCM3_A2_tmax_50_0.grb HADCM3_A2_tmax_dif_$i.grb
done
```

So now, by this stage, we have six files holding the mean change in each of prec, tmin and tmax for the next 50 and 100 years under these climate change scenarios.

We can now form a directory for holding the results as ascii matrices and export the numbers held in the grids as strait ascii.

```
mkdir ascii
cdo splitmon HADCM3_A2_prec_dif_1.grb temp1
cdo splitmon HADCM3_A2_tmin_dif_1.grb temp2
cdo splitmon HADCM3_A2_tmax_dif_1.grb temp3
cdo splitmon HADCM3_A2_prec_dif_2.grb temp4
cdo splitmon HADCM3_A2_tmin_dif_2.grb temp5
cdo splitmon HADCM3_A2_tmax_dif_2.grb temp6
for i in $(seq 1 9)
do
cdo output temp10$i.grb >ascii/prec50_$i.asc
cdo output temp20$i.grb >ascii/tmin50_$i.asc
cdo output temp30$i.grb >ascii/tmax50_$i.asc
cdo output temp40$i.grb >ascii/prec100_$i.asc
cdo output temp50$i.grb >ascii/tmin100_$i.asc
cdo output temp60$i.grb >ascii/tmax100_$i.asc
done
for i in $(seq 10 12)
do
cdo output temp1$i.grb >ascii/prec50_$i.asc
```

```
cdo output temp2$i.grb >ascii/tmin50_$i.asc
cdo output temp3$i.grb >ascii/tmax50_$i.asc
cdo output temp4$i.grb >ascii/prec100_$i.asc
cdo output temp5$i.grb >ascii/tmin100_$i.asc
cdo output temp6$i.grb >ascii/tmax100_$i.asc
done
rm temp*
```

The cdo command "output" sends the results to stdout, which are then sent to the files using the redirect operator >. This results in a set of files with the raw numbers, but they do not have any information on the format. We need to add a header to each before they can be read into GRASS. In any suitable text editor write a simple file that will form the header for all the ascii grids with the following form.

```
north: 90
south: -90
east: 0
west: -360
cols: 96
rows: 73
```

Make sure you include a carriage return on the last line. Save the file with the name "header" in the ascii directory. Now we can use the concatenate command in the bash shell to add this header to all the previously built files. As reassigning to an existing file is not allowed, we produce a new set of files and remove the previous files.

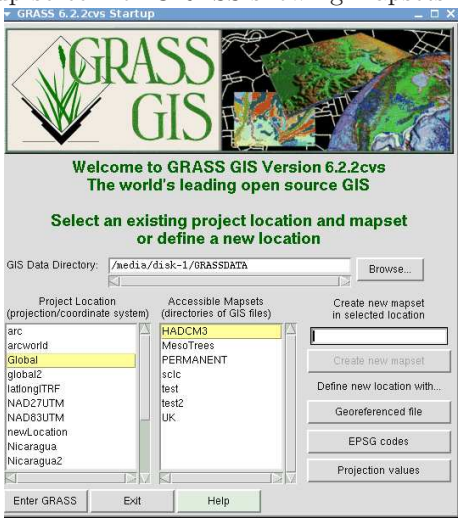
```
for i in $(seq 1 12)
do
cat ascii/header ascii/prec50_$i.asc>ascii/prec50h_$i.asc
cat ascii/header ascii/tmin50_$i.asc>ascii/tmin50h_$i.asc
cat ascii/header ascii/tmax50_$i.asc>ascii/tmax50h_$i.asc
cat ascii/header ascii/prec100_$i.asc>ascii/prec100h_$i.asc
cat ascii/header ascii/tmin100_$i.asc>ascii/tmin100h_$i.asc
cat ascii/header ascii/tmax100_$i.asc>ascii/tmax100h_$i.asc
rm ascii/prec50_$i.asc
rm ascii/tmin50_$i.asc
rm ascii/tmax50_$i.asc
rm ascii/prec100_$i.asc
rm ascii/tmin100_$i.asc
rm ascii/tmax100_$i.asc
done
```

5.5 Exporting to GRASS and visualising the data

The next step is to import all these files into GRASS. We use the same "GLOBAL" location that holds the WorldClim data, but it is a good idea to start a new mapset for this data. Lets call it HADCM3. After forming the mapset

in the GLOBAL location the startup screen should look something like figure 28, although in this case there are a few additional mapsets that have been set up for other purposes.

Figure 28: Startup screen for GRASS showing mapsets for scenario data.



Now once in GRASS cd [12]to the directory where the ascii grids were stored

```
for i in $(seq 1 12)
do
r.in.ascii input=tmax100h_$i.asc output=tmax100.$i
r.in.ascii input=tmin100h_$i.asc output=tmin100.$i
r.in.ascii input=prec100h_$i.asc output=prec100.$i
r.in.ascii input=tmax50h_$i.asc output=tmax50.$i
r.in.ascii input=tmin50h_$i.asc output=tmin50.$i
r.in.ascii input=prec50h_$i.asc output=prec50.$i
done
```

This should result in a complete set of the very coarse resolution global circulation model results to be available as GRASS raster layers. The default colour scheme is not suitable, so we can reassign colour rules to all the temperature layers with:

```
for i in $(seq 1 12)
do
r.colors map=tmax100.$i rules=byr
r.colors map=tmin100.$i rules=byr
r.colors map=tmax50.$i rules=byr
r.colors map=tmin50.$i rules=byr
r.colors map=prec50.$i rules=ryg
```

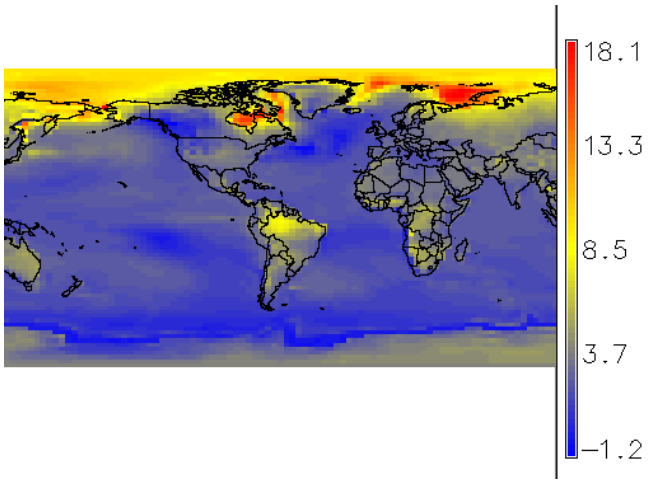
[12]The term ”cd” is commonly used to mean change directory using the cd command. In this document a degree of working knowledge of bash shell commands is assumed.

```
r.colors map=prec100.$i rules=ryg
done
```

Now if you have previously imported a vector layer showing country boundaries into the location the map shown in figure 30 can be produced by

```
d.mon start=x
d.mon select=x0
d.rast.leg tmax100.1
d.vect country type=boundary
```

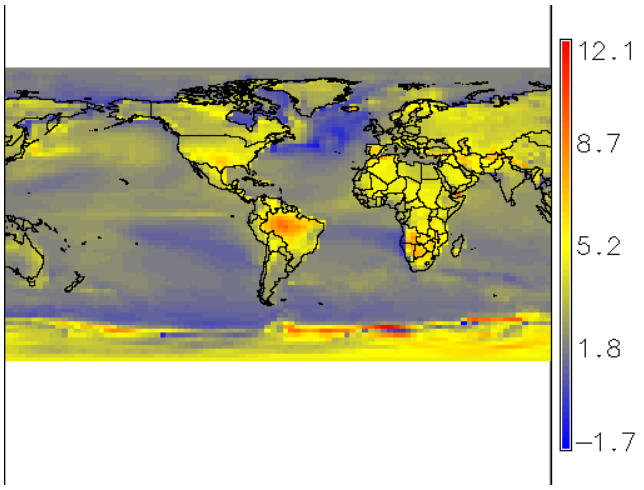
Figure 29: Predicted change in mean January maximum temperature in the period 2050-2099 compared with 1950-1999 (HADCM3 A2)



There are several points to note from figure 29. The A2 scenario is among the most alarming of the IPCC story lines, so models using the carbon emissions from this scenario show quite dramatic global change. Winter warming is mainly concentrated in the arctic seas. The very extreme value of an 18.1 C increase is attributable to a single pixel, which tends to exaggerate the scale. A rather better visual presentation of the results would truncate some of the values as is done by the on-line results server http://www.ipcc-data.org/cgi-bin/ddcvis/gcmcf. [13]The same slight exaggeration of the results is found in figure 29.

[13]It is quite possible to download ascii values directly from this map server, which is a potentially simpler, although much less flexible alternative to extracting values from GRIB data sets. However not all AR4 scenarios are yet available from this source at the time of writing.

Figure 30: Predicted change in mean May maximum temperature in the period 2050-2099 compared with 1950-1999 (HADCM3 A2)



5.6 Downscaling the data

The next step in processing the data is downscaling.

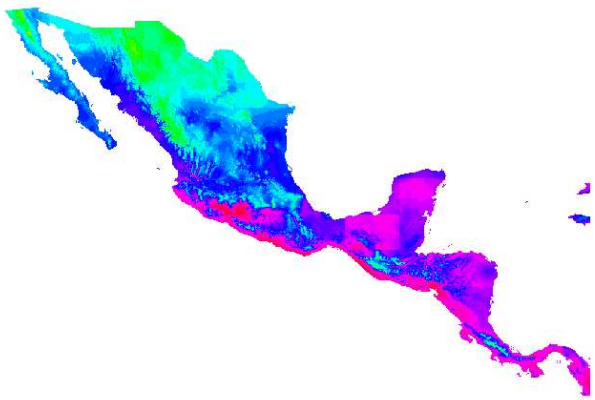
There are a number of challenges associated with downscaling from the extremely coarse resolution of the global circulation models to the 1-10 km grids used for regional species distribution modelling. These add to the uncertainties inherent in the modelling exercise. For example, differences in predicted rainfall for future scenarios are measured in mm per day over the month of interest. Because the climate data used as a baseline for the GCMs is coarse it is quite possible at a local scale to find that simply adding the change over the surface can lead to predictions of negative rainfall.

The simplest option for temperature would appear to be to add the WorldClim and GLM surfaces together. The script below will do this for previously imported tmax layers for the 100 years scenario. In order to run the script it would be convenient to move into a new mapset for the output with access (use g.mapsets) both to the PERMANENT mapset with the original WorldClim data as imported in the first part of this report and the HADCM3 scenario data as imported in this report. Make sure the resolution of the region is set to g.region res=00:03 for approximately 5km grid square output.

```
for i in $(seq 1 12)
do
r.to.vect tmax100.$i
r.mapcalc "tmax100.d$i=tmax100.$i+tmax$i/10"
done
```

However this approach results in very obvious artifacts with some of the underlying divisions between cells at the coarse resolution very obvious in the output as shown in figure 31.

Figure 31: Artifacts due to large grain of GCM models when simple additive downscaling is used.

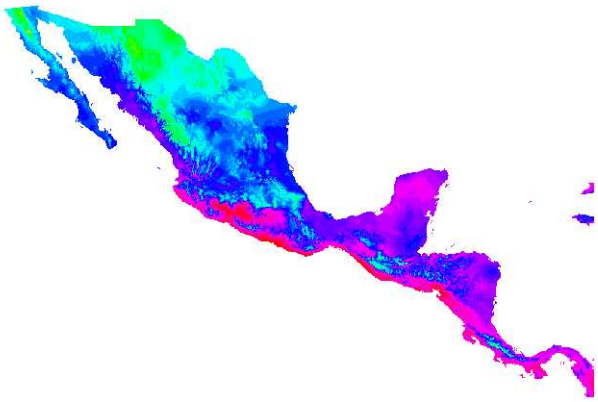


A partial solution to the "ghost" of the coarse scale model results is to first smooth between the centres of the coarse grid squares before adding in the results. The code below will do this for all the output of the "100" year simulation.

```
for i in $(seq 1 12)
do
g.region res=03:00
r.to.vect -z input=prec100.$i feature=point output=temp1
g.region res=00:03
v.surf.idw input=temp1 output=temp1 layer=0
r.mapcalc "prec100.s$i=if(temp1*30+prec$i>0,temp1*30+prec$i,0)"
g.mremove vect=temp* -f
g.mremove rast=temp* -f
done
for i in $(seq 1 12)
do
g.region res=03:00
r.to.vect -z input=tmax100.$i feature=point output=temp1
g.region res=00:03
v.surf.idw input=temp1 output=temp1 layer=0
r.mapcalc "tmax100.s$i=temp1+tmax$i/10"
g.mremove vect=temp* -f
g.mremove rast=temp* -f
done
for i in $(seq 1 12)
do
g.region res=03:00
r.to.vect -z input=tmin100.$i feature=point output=temp1
```

```
g.region res=00:03
v.surf.idw input=temp1 output=temp1 layer=0
r.mapcalc "tmin100.s$i=temp1+tmin$i/10"
g.mremove vect=temp* -f
g.mremove rast=temp* -f
done
```

Figure 32: Additive downscaling preceded by inverse distance weighting interpolation between the centre points of each GCM cell.



5.7 Moving the results to R for modelling

Now we can move the data into R as we did with the original climate data.

```
library(spgrass6)
a<-paste("tmax100.s",1:12,sep="")
a<-c(a,paste("tmin100.s",1:12,sep=""))
a<-c(a,paste("prec100.s",1:12,sep=""))
clima<-readRAST6(a)
```

The names can be changed so the object is identical to the original climate object.

```
a<-paste("tmax",1:12,sep="")
a<-c(a,paste("tmin",1:12,sep=""))
a<-c(a,paste("prec",1:12,sep=""))
names(clima@data)<-a
```

From now on we can substitute the new clima object in R where appropriate.

```
f<-function(x)max(x[1:12])-min(x[1:12])
```

clima[[”AnnualMaxTDif”]]<-apply(clima@data,1,f)

```
f<-function(x)max(x[1:12]-x[13:24])
clima[["DailyTDif"]]<-apply(clima@data,1,f)
f<-function(x)sum(x[25:36]>100)
clima[["GrowingMonths"]]<-apply(clima@data,1,f)
clima2<-clima
clima2@data<-clima2@data[,c(6,13,25,30,37,38,39)]
```

If you have followed the methodology to this point, substituting the new scenario climate data in R should be strait forward. The models are first fit by overlaying the original clima layers then the new layers are substituted when using the predict method. A future addition to this report will document in more detail how the scenario data can be used to predict species turnover.

References

[1] Hansen, M., DeFries, R., Townshend, J.R.G., Sohlberg, R., Dimiceli, C., and Carroll, M., 2002, Towards n operational MODIS continuous field of percent tree cover algorithm: examples using AVHRR and MODIS data. (2002) Remote Sensing of the Environment 83, 303-319.

[2] Hansen, M., DeFries, R., Townshend, J.R.G., Carroll, M., Dimiceli, C., and Sohlberg, R., 2002, Global percent tree cover at a spatial resolution of 500 meters: first results of the MODIS vegetation continuous fields algorithm. submitted to Earth Interactions

[3] Hijmans, R.J., S.E. Cameron, J.L. Parra, P.G. Jones and A. Jarvis, 2005. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25: 1965-1978. (pdf)

[4] Barry, S. & Elith, J. (2006) Error and uncertainty in habitat models. Journal of Applied Ecology, 43, 413-423.

[5] Elith, J., Graham, C.H., Anderson, R.P., Dudík, M., Ferrier, S., Guisan, A., Hijmans, R.J., Huettmann, F., Leathwick, J.R., Lehmann, A., Li, J., Lohmann, L.G., Loiselle, B.A. Manion, G., Moritz, C., Nakamura, M., Nakazawa, Y., Overton, J.M., Peterson, A.T., Phillips, S.J., Richardson, K.S., Scachetti-Pereira, R., Schapire, R.E., Soberón, J., Williams, S., Wisz, M.S. & Zimmermann, N.E. (2006) Novel methods improve prediction of species' distributions from occurrence data. Ecography, 29, 129-151.

[6] Ferrier, S. & Guisan, A. (2006) Spatial modelling of biodiversity at the community level. Journal of Applied Ecology, 43, 393-404.

[7] Golicher, D J, Cayuela L, J. Alkemade R J, González-Espinosa M & Ramírez-Marcial N (2007) Applying Climatically Associated Species Pools to modeling compositional change in tropical montane forests (Accepted for publication in Global Ecology and Biogeography July 2007)

[8] Guisan, A. & Zimmermann, N.E. (2000) Predictive habitat distribution models in ecology. Ecological Modelling, 135, 147-186.

[9] Guisan, A., Lehmann, A., Ferrier, S., Austin, M., Overton, J.MC.C., Aspinall, R. & Hastie, T. (2006) Making better biogeographical predictions of species' distributions. Journal of Applied Ecology, 43, 386-392.

[10] Guisan, A., Graham, C.H., Elith, J., Huettmann, F. & NCEAS Species Distribution Modelling Group. (2007) Sensitivity of predictive species distribution models to change in grain size. Diversity and Distributions, 13, 332-340.

[11] Lennon, J.J., Kunin, W.E., Corne, S., Carver, S. & Van Hees, W.W.S. (2002) Are Alaskan trees found in locally more favourable sites in marginal areas? Global Ecology and Biogeography, 11, 103-114.

[12] Luoto, M., Pöyry, J., Heikkinen, R.K. & Saarinen, K. (2005) Uncertainty of bioclimate envelope models based on the geographical distribution of species. Global Ecology and Biogeography, 14, 575-584.

[13] Pearson, R.G. & Dawson, T.P. (2003) Predicting the impacts of climate change on the distribution of species: are bioclimate envelope models useful? Global Ecology and Biogeography, 12, 361-371.

[14] Pearson, R.G. & Dawson, T.P. (2004) Bioclimate envelope models: what they detect and what they hide – response to Hampe (2004). Global Ecology and Biogeography, 13, 471-472.

[15] Pearce, J.L., Cherry, K., Drielsma, M., Ferrier, S. & Whish, G. (2001) Incorporating expert opinion and fine-scale vegetation mapping into statistical models of faunal distribution. Journal of Applied Ecology, 38, 412-424.

[16] Phillips Steven J. , Robert P. Anderson, Robert E. Schapire. Maximum entropy modeling of species geographic distributions. Ecological Modelling, 190:231-259, 2006.

[17] Thuiller, W., Araújo M.B., Lavorel S. (2003) Generalized Models versus Classification Tree Analysis: a comparative study for predicting spatial distributions of plant species at different scales. Journal of Vegetation Science 14, 669-680 PDF

[18] McPherson, J.M., Jetz, W. & Rogers, D.J. (2004) The effects of species' range sizes on the accuracy of distribution models: ecological phenomenon or statistical artefact? Journal of Applied Ecology, 41, 811-823.

[19] Terry M Therneau and Beth Atkinson. R port by Brian Ripley <ripley@stats.ox.ac.uk>. (2006). rpart: Recursive Partitioning. R package version 3.1-33. S-PLUS 6.x original at http://mayoresearch.mayo.edu/mayo/research/biostat/splusfunctions.cfm

[20] Trevor Hastie (2006). gam: Generalized Additive Models. R package version 0.98.

[21] Oksanen, J., Kindt, R., Legendre, P. & O'Hara, R.B. (2007). vegan: Community Ecology Package version 1.8-6. http://cran.r-project.org/

[22] R Development Core Team (2007). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.