Spatial Graphics Using The R Package `geoMap`
Draft Only

Glenn De'ath

Australian Institute of Marine Science
Townsville
Queensland 4810
Australia

July 28, 2010

Contents

List of Figures

1 Introduction

The package `geoMap` that can produce a great variety of plots that can be used for both exploratory analysis, presentations and publication. The types of displays include:

1. Simple black and white line maps with no fills.
2. More complex colour-rich mappings with additional shapes, features and labels
3. Additional interactive data exploration and displays
4. Production of publication quality graphics with no editing needed!
5. Composite maps with several images on a single page
6. Rotation of maps; useful to more efficiently map the GBR

The package `geoMap` also requires the package `shapefiles`. Fitted surfaces can be generated using the package `geo`.

1.1 Some Examples

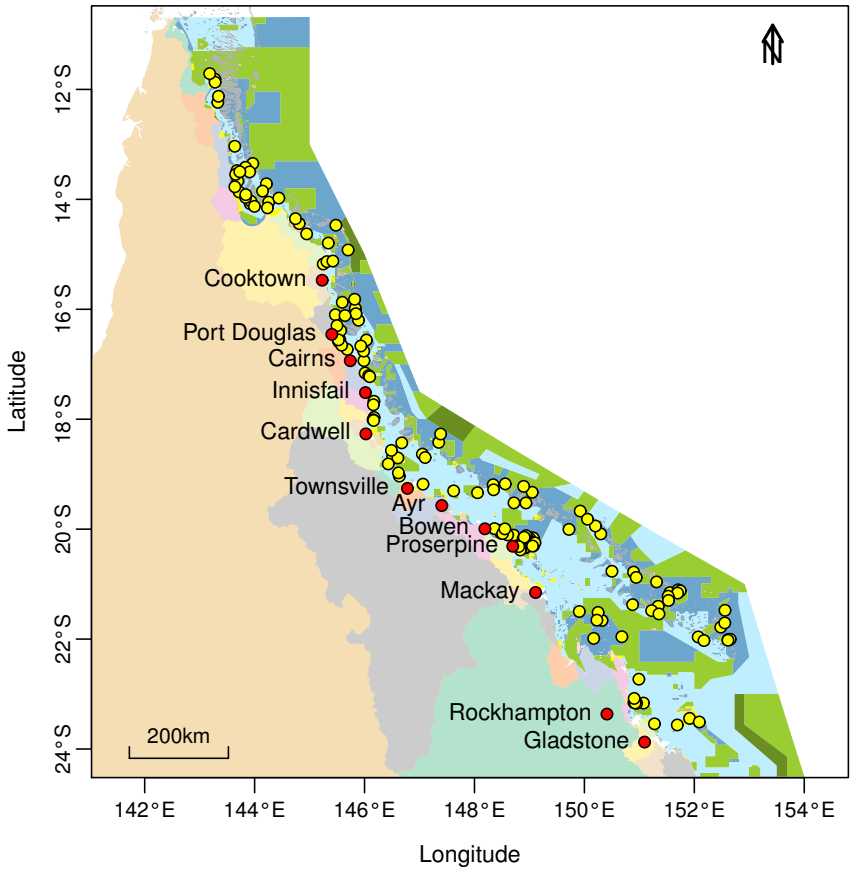

Figure 1: A map of the Great Barrier Reef including basins, zoning, reefs, towns and data locations

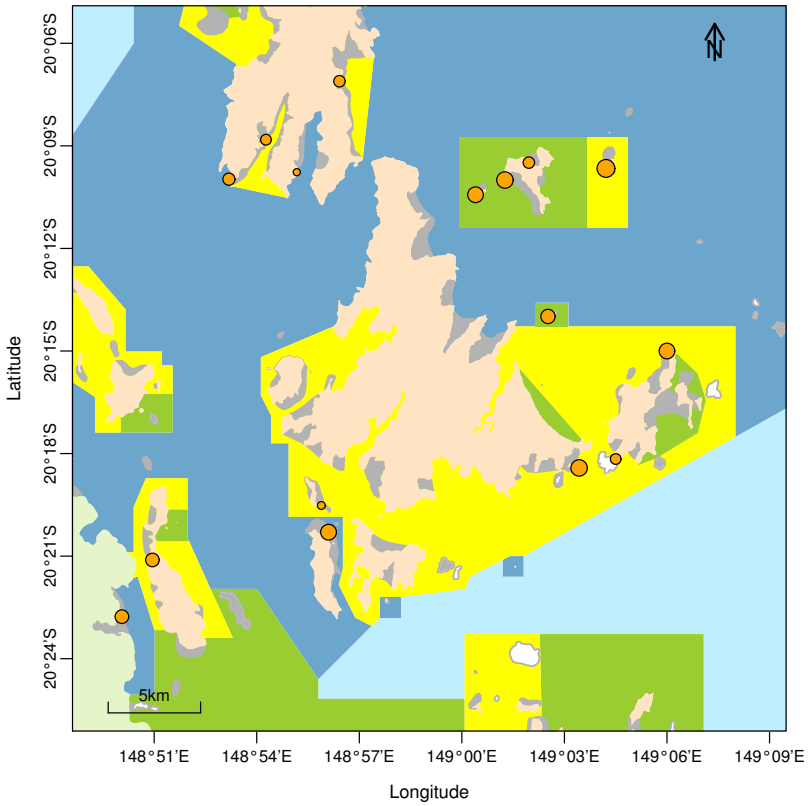

Figure 2: Zooming in interactively on the Great Barrier Reef

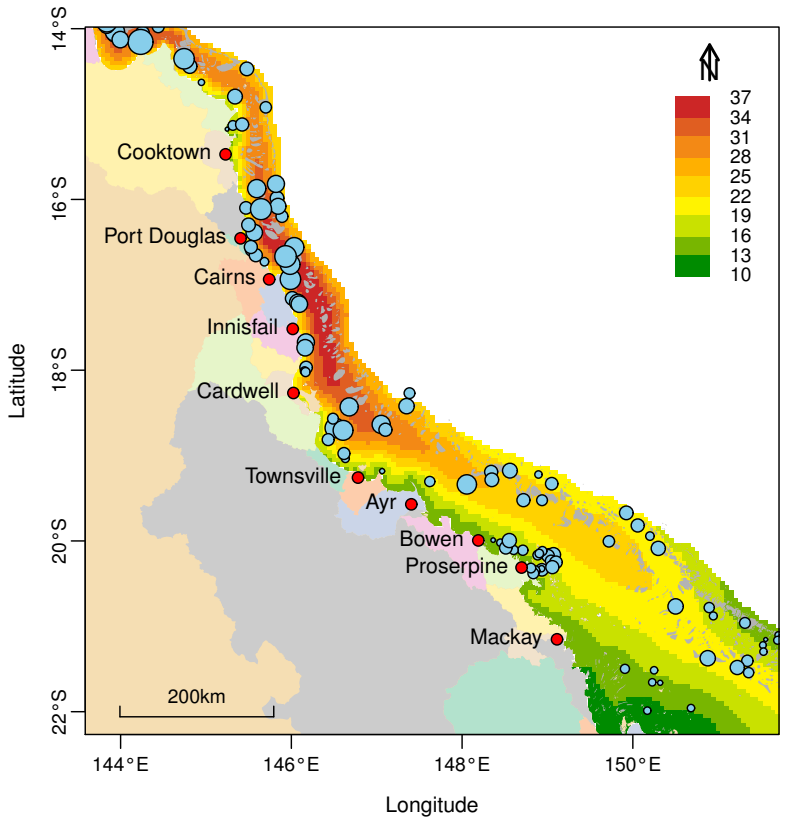

Figure 3: The fitted surface of soft coral richness and data points of size proportional to richness

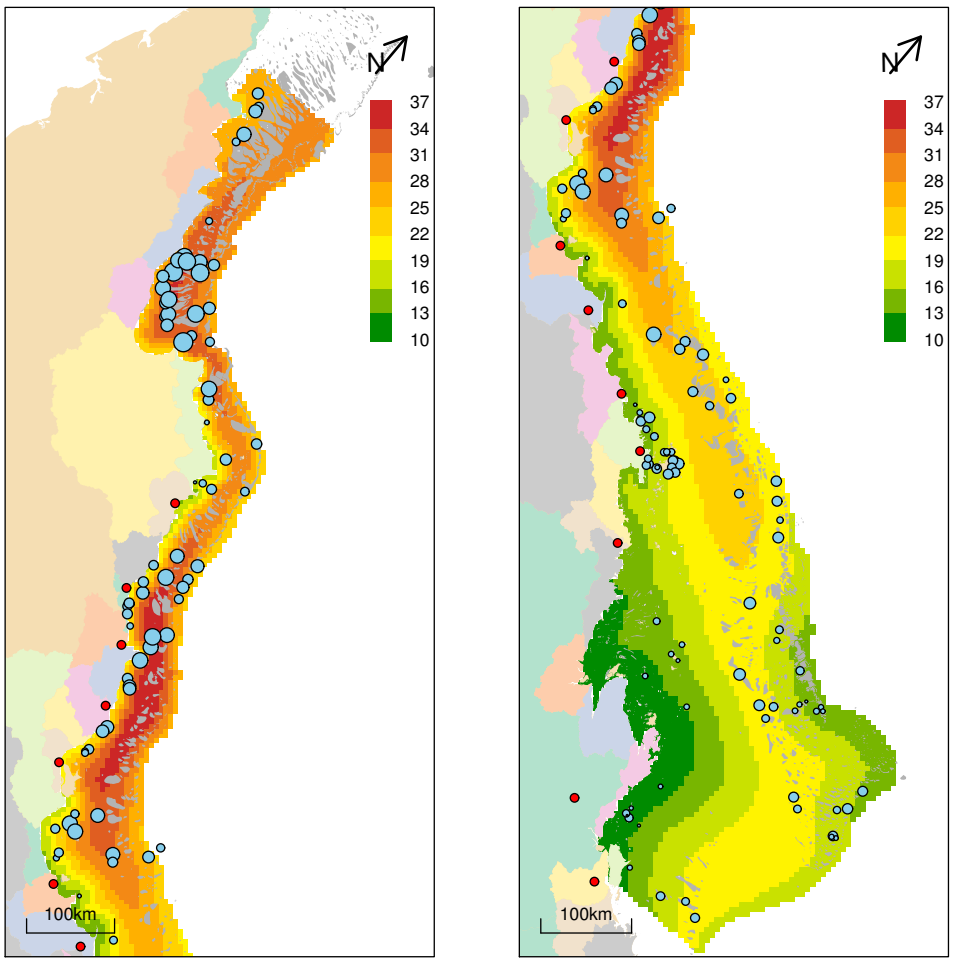

Figure 4: Zoom-in side-by-side maps showing richness of soft corals in the Whitsundays

2 The function geoMap

geoMap builds maps as an ordered series of shapes, much like any GIS.

2.1 Arguments of geoMap

```
args(geoMap)
function (data = list(), shape = list(queensland), fit, control,
    xlim, ylim, fit.se = FALSE, rotate = FALSE, alpha = 40, bound = boundary.aa,
    interactive = FALSE, axes.add = T, auto.cex = TRUE, xlab = "Longitude",
    ylab = "Latitude", ...)
```

2.2 Datasets, Shapes, Fits and Attributes in geoMap

The shapes of geoMap fall into one of three types, namely:

1. **data** (data frames in R) that give us plots of points and text labels
2. **shape** based on imported ESRI shapefiles; mainly polygons, but also can be point or line data
3. **fit** surfaces from the package geo

Datasets, shapes and fits require additional information **before** we can represent them graphically. This information is assigned to each dataset, shape or fit as attributes. Note, the term attribute has a particular meaning R and we use special functions to assign attributes to the datasets, shapes and fits. These attributes express all the styles of the graphical elements.

The function attrSet is use to set the attributes of datasets, shapes and fits. It can also list the attributes of these three classes:

```
> attrSet()

Default parameter values:

data.frame
  pts.add = TRUE, col='black', pch=21, bg='skyblue', cex=1, txt.add = FALSE,
  txt.text=rownames(data), txt.col='black', txt.cex=1, txt.pos

shape
  color = sapply(9:3/10,grey), border = NA,
  density= NULL, angle = 45, do = TRUE

fit
  image.add = TRUE, image.se = FALSE,
  image.col = colorRampPalette(c('green4','yellow', 'orange', 'firebrick3'), space = 'Lab'),
  image.breaks = 9, image.quant = NA,  contour.add = FALSE,
  contour.col = 'black', contour.cex = 1, contour.levs = 9
```

If you forget to assign attributes before you run geoMap then they will be assigned automatically. If the default values are not be to your liking then you can change them simply.

2.2.1 Datasets

The default attributes are simply assigned to datasets, shapes and fit. For example, if data was your data.frame then the attributes can be set:

```
attrSet(data)
```

You can list them by:

```
attributes(data)
```

Or remove them by:

```
data <- data.frame(data)
```

The most important attribute is "xy". You **must** specify spatial coordinates (of course!) and the names must partially match either "latitude", "Latitude", and "LATITUDE". Ditto for longitude. So you could use "lat" and "long" provide no other variables also give partial matches. Thus you could have another variables in the dataset with names "latter" and "LAt" but not "la" or "LATI". Yes, R is case-sensitive.

We can also set attributes to other than their defaults, for example:

```
attrSet(data, bg="green", txt.add=TRUE, txt.col="red", txt.pos=2)
```

This will check the background of the plotted points to green, and will add text labels that are the row names of the data set with colour red on the left hand side of the point markers. All other attributes will be set to their defualt values.

Once you have initialised a dataset in this way, you can easily change single attributes:

```
attrSet(data, bg = "pink")
```

or:

```
attr(data, bg) <- "pink"
```

2.2.2 Shapes

For shapes comprising polygons there are only 5 attributes. They control the colours of fills and boundaries and "do" lets you select subsets of the polygons.

```
> attrSet
function (shape, col = grey(1:5/10), border = par("fg"), density = NULL,
    angle = 45, do = TRUE)
{
    attr(shape, "color") <- col
    attr(shape, "border") <- border
    attr(shape, "density") <- density
    attr(shape, "angle") <- angle
    attr(shape, "do") <- do
    shape
}
```

As with data sets we assign attributes to shapes:

```
attrSet(shape)
```

Consider the following:

```
fit <- attrSet(fit, col=c("green","red"),
border=rep(c("yellow","blue"),each=2), do=c(TRUE,FALSE))
```

If there were 12 polygons in the set, then you would get all green polygons with alternating yellow and blue borders. Can you work out why?

2.2.3 Fits

Fitted surfaces are represented by colored images and/or contour lines. The default is to just plot the image. Colour is extremely important in representing images. You can chose smooth or stepped scales (the latter are almost universally more effective), and you can control the colour ramp, the number of breaks and the cuts values as well as the key (size dimensions and location). You can also have fine control over contours.

```
> attrFitSet
function (fit, image.add = TRUE, image.se = FALSE, image.col = colorRampPalette(c("green4",
    "yellow", "orange", "firebrick3"), space = "Lab"), image.breaks = 9,
    image.quant = NA, contour.add = FALSE, contour.col = "black",
    contour.cex = 1, contour.levs = 9)
{
    attr(fit, "image.add") <- image.add
    attr(fit, "image.se") <- image.se
    attr(fit, "image.col") <- image.col
    attr(fit, "image.breaks") <- image.breaks
    attr(fit, "image.quant") <- image.quant
    attr(fit, "contour.add") <- contour.add
    attr(fit, "contour.col") <- contour.col
    attr(fit, "contour.cex") <- contour.cex
    attr(fit, "contour.levs") <- contour.levs
    fit
}
```

And we assign attributes to fits:

```
fit <- attrFitSet(fit)
```

2.2.4 Control options

There are many options to vary aspects of the plot other than the datasets, shapes and fits. Options are passed to `geoMap` using the control argument that takes a list of values:

```
# The default plot
geoMap()

# Change the position of the North, remove the scale bar, add a light blue background.
geoMap(control=list(north.x=0.1,scale.bar.add=FALSE,bgrnd.add=TRUE,bgrnd.col="lightblue1"))}

# The default options can be viewed:
geoMap.control()}
```

The options comprise:

Background: Options to add a plain background and to specify the color.

```
 bgrnd.add = FALSE; bgrnd.col = grey(0.95)
```

Image Key: Options to add and locate and size the image key

```
 key = TRUE; key.fun = NA; key.y = c(0.65, 0.90); key.x = c(0.85, 0.9);
key.horiz = FALSE;
```

Scalebar: Options to add and locate and size the scale bar

```
scale.bar.add = T; scale.bar.x = c(0.05, 0.3); scale.bar.y = 0.025;
```

North Arrow: Options to add and locate the north arrow

```
north = TRUE; north.x = 0.9; north.y = 0.95;
```

Map Margins: Options for the margins for normal and rotated maps

```
mar.rot = rep(1, 4); mar.no.rot = c(3.5, 4, 1, 1); mgap = c(2, 0.5, 0);
```

Linux Graphics Options: Not applicable to other than Linux users

```
X11.type = c("Xlib","cairo")[1]
```

3 Shapefiles in R and geoMap

Shapefiles can be imported into R in several ways. The function `readShape` will import shapes comprising polygons, points and lines. These can be then used by `geoMap`. I have not used points and lines yet, since these can be be added through the `data` argument of `geoMap`.

3.1 The Package layers

The package `layers` contains some useful shapes and data sets for the GBR that can be passed to `geoMap` in lists. For example try:

```
library(layers)
ls("package:layers")
```

gives:

```
  [1] "basin"      "bioregion"  "nrm"        "queensland" "reefs"
  [6] "rivers22"   "rivers9"    "towns12"    "towns6"     "whagbr"
 [11] "zoning"
```

and to see some examples:

```
geoMap(data=list(towns12),shape=list(queensland,basin,zoning,reefs))
```

4 Interactive Maps

If `geoMap` with the option "int=TRUE" then the user can vary the view of the map by mouse clicks:

1. Click opposite corners of a (phantom) box to select the viewing area (adjusted to the greatest measure of the box)
2. Double-click a point to centre the graph at that point
3. Click corners of the border off the map:
   (a) Bottom left – reset the view
   (b) Top left – reset the view
   (c) Top right – reset the view
   (d) Bottom right – unused
4. Click sides of the border off the map to move the map in that direction

5 Rotating Maps and Multiple Maps

Maps can be rotated using the argument `control=list(rot=TRUE, theta=40)`.

Multiple maps can be plotted by controlling the page layout, for example:

```
par(mfrow=c(1,2))
geoMap(...)
geoMap(...)
```

An automated map-splitting function is under development. It will cater for the the long thin GBR.

6 geoMap Documentation

For now – these notes :-)

7 The Working Session – Learn By Play

The following code should illustrate `geoMap` works. Try them out.

```
## Some example code for geoMap ##
library(geoMap)

## Learning about "shape" ##
geoMap()
geoMap(shape=reefs)
## Oops -- you may do this often!

geoMap(shape=list(reefs))
geoMap(shape=list(queensland,reefs))
geoMap(shape=list(basin,queensland,reefs,zoning))
geoMap(shape=list(queensland,basin,zoning,reefs))
geoMap(shape=list(zoning,reefs,queensland,basin))

## Finding out about attributes of shapes ##

geoMap(shape=list(queensland,basin))
names(attributes(basin))
## or if like me you can't type
## also makes it easy to use attr -- up arrow and delete !!
nattr(basin)
attr(basin,"color")
ls()
basin <- attrSet(basin)
ls()
## Note basin is now saved locally in your workspace
attr(basin,"color")
## Notice the difference? ##

geoMap(shape=list(queensland,basin))
attr(basin,"color")
rm(basin)
```

```
geoMap(shape=list(queensland,basin))
## So what's going on? ##

attr(basin,"border") <- "blue"
geoMap(shape=list(queensland,basin))

attr(basin,"border") <- rainbow(10)
attr(basin,"color") <- "grey90"
geoMap(shape=list(queensland,basin))

attr(basin,"do") <- c(TRUE,FALSE) # or c(T,F)
geoMap(shape=list(queensland,basin))
attr(basin,"do") <- TRUE

## But take care since the shapes are plotted in a certain order

## Using the data base of the shapefile ##

## Safe way to select/omit a subset of polygons
nattr(basin)
attributes(basin)$dbf
summary(attributes(basin)$dbf)
area <- attributes(basin)$dbf$AREA
mean.area <- mean(area)
## NAs are not plotted
attr(basin,"color") <- c(NA,"green")[(area > mean.area)+1]
## put black borders so we can identify individual polygons
attr(basin,"border") <- "black"
geoMap(shape=list(queensland,basin))


geoMap()
geoMap(data=softcorals)
## Oops -- you may do this often too!

geoMap(data=list(softcorals))

## Learning about "data" ##

## Set attributes for geoMap ##
nattr(softcorals))
softcorals <- attrSet(softcorals)
## Notice it found suitable coordinates
nattr(softcorals))

## Clear geoMap attributes #
softcorals <- data.frame(softcorals)
nattr(softcorals))

## resets the attributes
softcorals <- attrSet(softcorals)
geoMap(data=list(softcorals))
nattr(softcorals)
```

```
attr(softcorals,"bg") <- "orange"
attr(softcorals,"cex") <- softcorals$richhetero/10
geoMap(data=list(softcorals))
attr(softcorals,"bg") <- c("green","red")[(softcorals$visib > 10)+1]
geoMap(data=list(softcorals))

## interact with the map : see notes for "controls" -- ie where to click ##
geoMap(data=list(softcorals),int=T)

## changing point colors, type and size

## adding and positioning text ##
nattr(towns12)
attr(towns12,"txt.add")<-T
geoMap(data=list(towns12))
attr(towns12,"txt.pos")<-2
geoMap(data=list(towns12))
attr(towns12,"txt.pos")<-3
geoMap(data=list(towns12))
attr(towns12,"txt.pos")<-4
geoMap(data=list(towns12))
attr(towns12,"txt.pos")<- -1
geoMap(data=list(towns12))
## -1 gives you thigmophobe :-)

## For learning about fitted surfaces see the "geo" documentation ##
fit <- attrSet(fit)
nattr(fit)
attrSet()

geoMap(fit=fit)
# pretty scale ##
attr(fit,"image.breaks") <- seq(10,37,by=3)
geoMap(fit=fit)
attr(fit,"image.breaks") <- seq(10,38,length=29)
attr(fit,"image.col") <- col.yorr4
geoMap(fit=fit)
```

8 Can You Generate These Maps?

Which maps?

Yes, you've guessed it – Figures 1 – 4.

9 Colours in R

Colour (or color; they are synonymous in R!) offers many, many options in R. You can set up individual colours or colour ramps, use one of many packages to generate and manage them or use the built in colours (see R-colours.pdf – use lower case versions of these names with no spaces; e.g. Light Blue 1 is specified as lighblue1).

The package `RColorBrewer` provides some elegant simple colours for ramps and discrete colour sets. Install the package and check out the help.

Figure 5: The RColorBrewer colour sets

10 Data: Soft Corals of the Great Barrier Reef

We will use the following data set to illustrate the use of `geoMap`. The data set `softcorals` is part of the installed packages. First we load the data with:

```
data(softcorals)
```

The command `summary(softcorals)` provides a summary of these data and we can check the data for obvious mistakes. For example, are latitudes negative for the southern hemisphere?, are the ranges of the variables sensible? Also note the spatial coordinates are labelled "lat" and "long". This is the default and it keeps life simple for us! Use it, otherwise you will have to specify the labels of your coordinates in several places.

```
summary(softcorals)
      reef          lat              long          richall        richphoto
 13-050 :  1   Min.   :-23.56   Min.   :143.2   Min.   : 5.00   Min.   : 4.00
 13-063 :  1   1st Qu.:-20.67   1st Qu.:145.5   1st Qu.:16.00   1st Qu.:11.00
 13-077 :  1   Median :-19.21   Median :148.2   Median :22.00   Median :15.00
 13-120 :  1   Mean   :-18.38   Mean   :147.7   Mean   :21.91   Mean   :15.61
 13-123 :  1   3rd Qu.:-16.08   3rd Qu.:150.2   3rd Qu.:27.00   3rd Qu.:20.00
 19-109 :  1   Max.   :-11.71   Max.   :152.7   Max.   :44.00   Max.   :29.00
 (Other):144
   richhetero
 Min.   : 0.000
 1st Qu.: 2.000
 Median : 5.500
 Mean   : 6.307
 3rd Qu.: 9.750
 Max.   :22.000
```

To see the first few rows of the data use:

```
head(softcorals)
  reef       lat     long     richall richphoto richhetero
1 13-050 -13.34784 143.9660      26        17          9
2 13-063 -13.41732 143.8355      34        25          9
3 13-077 -13.50091 143.9098      41        25         16
4 13-120 -13.71717 144.2147      28        19          9
5 13-123 -13.85015 144.1450      39        28         11
6 19-109 -19.52333 148.9365      19        17          2
```

11 Installing R Packages

The packages `geoMap` and `shapes` are available from g.death@aims.gov.au. The package `geo` is require to produce the "fit" object that represent the spatially smoothed surface.

11.1 Linux

The source code is supplied for Linux users since compilation is dependent of the particular flavour of Linux being used. These sources can, of course, be used for a Windows compilation, if required.

Certain other packages are also required and need to be installed prior the installation of the R-packages. These include R and some spatial and development packages:

To install R:

```
sudo apt-get update
sudo apt-get install r-base r-base-dev
```

You will also need:

```
sudo apt-get install gdal-bin libgdal1-dev libglut3-dev xorg-dev
```

install CRAN packages using `install.packages()` within an R session or by:

```
R CMD INSTALL -l /usr/lib/R/library/  \
    http://cran.au.r-project.org/src/contrib/rgdal
```

Install local packages using something like :

```
sudo R CMD INSTALL -l /usr/lib/R/library/ /home/omni/r/r-work/geoMap -c
```

11.2 Windows

To install in Windows do:

1. Download and install the R executable (currently R-2.11.2-win32.exe).
   Note: if you are using Vista or Win 7 you need to know about administrator rights!!!
2. Run R and using the GUI, install the packages `gstat`, `mgcv`, `maptools`, `rgdal`, `sp`, `KernSmooth` from CRAN.
3. Using the GUI, install the packages `geo`, `makegrid` and `acrossAlong` from local zips.

11.3 Help

If you have problems e.g. missing or broken software then let me know!