GUIDE TO THE R ATLAS TOOLBOX
Draft Only

Glenn De'ath

Australian Institute of Marine Science
Townsville
Queensland 4810
Australia

August 27, 2010

Contents

List of Figures

1 Introduction

This guide is an introduction to R tools used to analyse data and generate outputs for the e-atlas and ningaloo-atlas. If focuses primarily on the generation of spatially smoothed data sets, but other data display methods are outlined. R is an extremely powerful statistical and graphical package, but it is not easy to learn and is less easy to use effeciently and effectively. This project aims to simplify and make robust the processes of using R for spatial modelling and and the generation of complex graphics. The guide discusses data management and the partial-automation of some of these processes. Reliability, repeatability and robustness are key aspects of the overall process.

Spatial mapping of data involves four steps: (1) data preparation, (2) grid generation, (3) spatial modelling, and (4) outputs:

1. DATA: Clean it, prepare it and read it into R; csv format is the simplest to import from. If you wish to keep things simple, then label the spatial coordinates "long" and "lat". If you use other names e.g. "Latitude" and "Longitude" or "X" and "Y", then you can specify them as the spatial coordinates in the function calls.

2. GRID: Having a "good" grid greatly enhances the quality of the graphical representation of fitted surface. Select or generate a spatial grid that has an appropriate cell size. If the cells are too large, then details and graphics resolution will be lost. Conversely, if the cells are too small, then computational costs will be excessive and graphics will be too large and thus slow to display. The grid should also have appropriate spatial coverage. Grids should closely fit the data, otherwise predictions at points far from the data will be highly variable (i.e. have low precision), and this will reduce the perceived variation close to the data points. The modelling process will use a default grid (the expanded convex hull) if you do not supply one, and if the data space is convex and of uniform density then the supplied will often be fine.

3. MODEL: First, select the spatial model. We will use generalized additive models (GAMs) with, typically, the degree of smoothness selected by generalized cross-validation. GAMs can model data with non-normally distributed errors, e.g. binomial and poisson data. Other models such as boosted trees and non-parametric smoothers are also available, and include robust methods. The models are assessed by examining residual plots and spatial correlation, and the predicted grids can be enhanced by trimming. This is based on comparison of variation of predictions at the sampling sites with predictions on the spatial grid.

4. OUTPUTS: The following outputs are then generated:

   (a) KML files that can be viewed in Google Earth

   (b) Grided predictions and various graphical elements and styles (e.g. colour schemes) that are used in the mapping tools of the `e-Atlas` and `ningaloo-Atlas`

   (c) Publication quality maps of the fitted surfaces and geographical elements

The packages `geo`, `makegrid`, `abt` and `acrossAlong` have been written for the Toolbox and are available from g.death@aims.gov.au. They depend on several other R packages, namely `gstat`, `mgcv`, `maptools`, `rgdal`, `sp`, `KernSmooth`. These packages need to be installed prior to `geo`, `makegrid` and `acrossAlong`, and are available from CRAN, `http://cran.r-project.org/`, the location of the R package repository. This can be done from within R and is described at the end of this document.

2 Spatial Mapping Using The Toolbox

We follow the four steps of spatial mapping as outlined in the Introduction.

2.1 Data: Soft Corals of the Great Barrier Reef

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
```

Check the locations of the sites use: `geoMap(data=list(softcorals))`

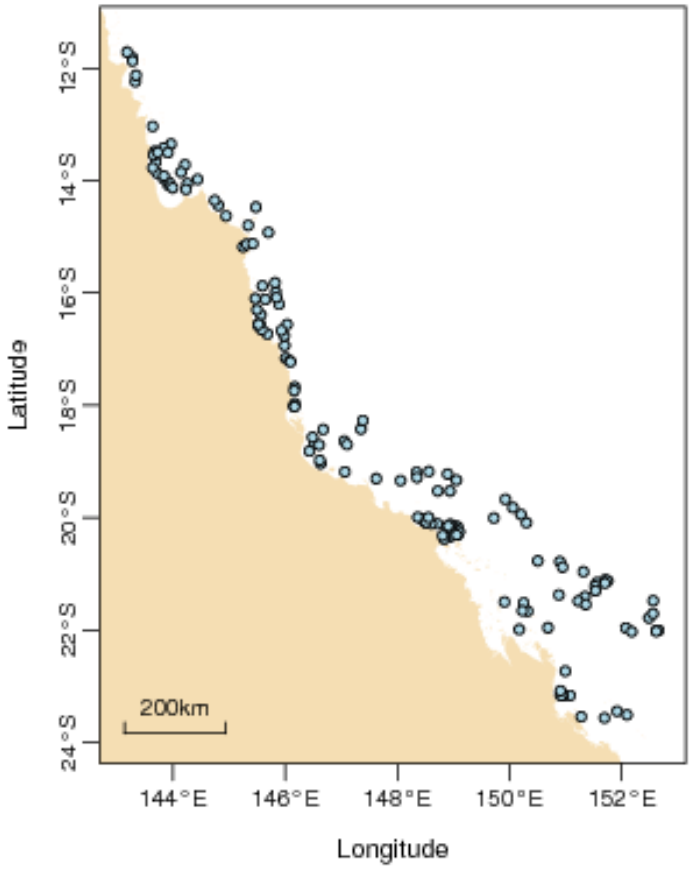

Figure 1: Locations of the sites of the soft coral surveys on the Great Barrier Reef

2.2 Grid construction and selection using `makegrid`

The package `makegrid` is used for making grids based on the spatial structure of your data. The simplest, and often the most effective and quickest is to use the function `makegrid.select` which plots the locations of your data and then you make your grid by generating the boundary using successive point-and-clicks. The grid within your chosen boundary is generated at you chosen resolution. To use this method simply use the command:

```
myGridBoundary <- makegrid.select(softcorals[,c("long","lat")])
```

The left figure shows the points interactively selected by the user, right click is used to end the sequence. Note that the boundary does not have to be closed; the first and last points are connected to form the closed boundary of the grid.

The right figure shows the data points, the completed boundary and the grid. The boundary and the grid are returned to the user in a list (myGridBoundary). The grid can be used for predicting smooth surfaces onto and graphics and KMLs can then be generated.

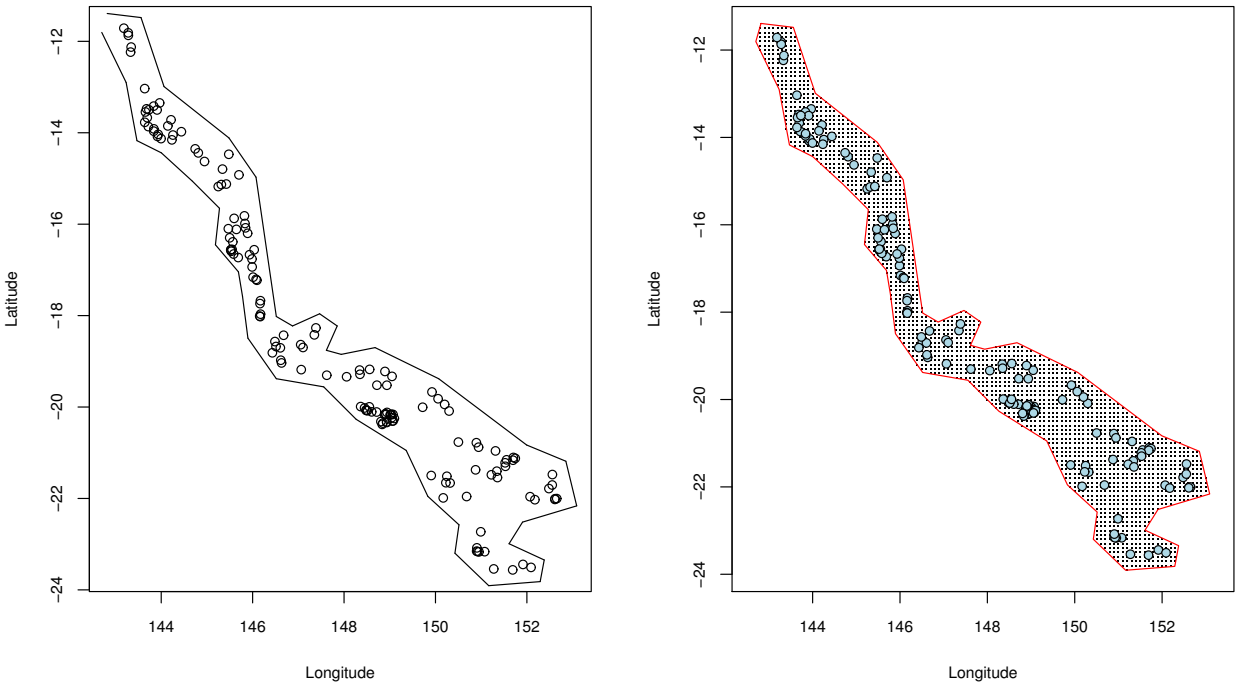

Figure 2: Interactive selection of grid (left) and resulting grid formed by `makegrid.select` (right)

The function `makegrid.select` returns a list with two components: "boundary" and "grid". The boundary can be reused to generate additional grids, e.g. of different resolution. To generate a grid of resolution 0.01 degrees use makegrid.rect:

```
myGrid <- makegrid.rect(myGridBoundary$boundary,size=0.01)
```

The process of grid generation can also be broken into two stages, boundary and grid within boundary, as opposed to the single process of `makegrid.select`. Do this by calling `makegrid.boundary` and then using the output of that call in `makegrid.select`: If you just want the boundary then use `makegrid.boundary`:

```
myBoundary <- makegrid.boundary(softcorals[,c("long","lat")])
myGrid <- makegrid.rect(myBoundary,size=0.01)
```

An alternative method of forming grids is based on the density of the data points. We want the grid to include areas where there is a high density of points and, most often, exclude areas where the density is low. Borders of grids can thus be defined by contours of low density that are selected by the user.

Two examples of density-based grids can be generated:

```
myGrid2 <- makegrid.density(softcorals,use=1,size=0.1,eps=0.5)
myGrid3 <- makegrid.density(softcorals,use=1:3,size=0.1,eps=0.3)
```

The left plot below shows a density plot and a single outer boundary (blue) using the default settings of `makegrid.density`. The parameter 'eps' determines how local the fitting is, with smaller values giving more local fits. In the second example (right) we use "eps=0.3" and the three outer boundaries are selected. By varying "eps" and thus the boundaries you select (they number by (i) density (low first) then by (ii) values from N to S within a density level. The default mesh-size ("size") of the grids in `makegrid.density` is 0.1°. This is generally too large, so once you have chosen the boundaries, then `makegrid.density` should be run with a smaller value of "size" (e.g. 0.01).

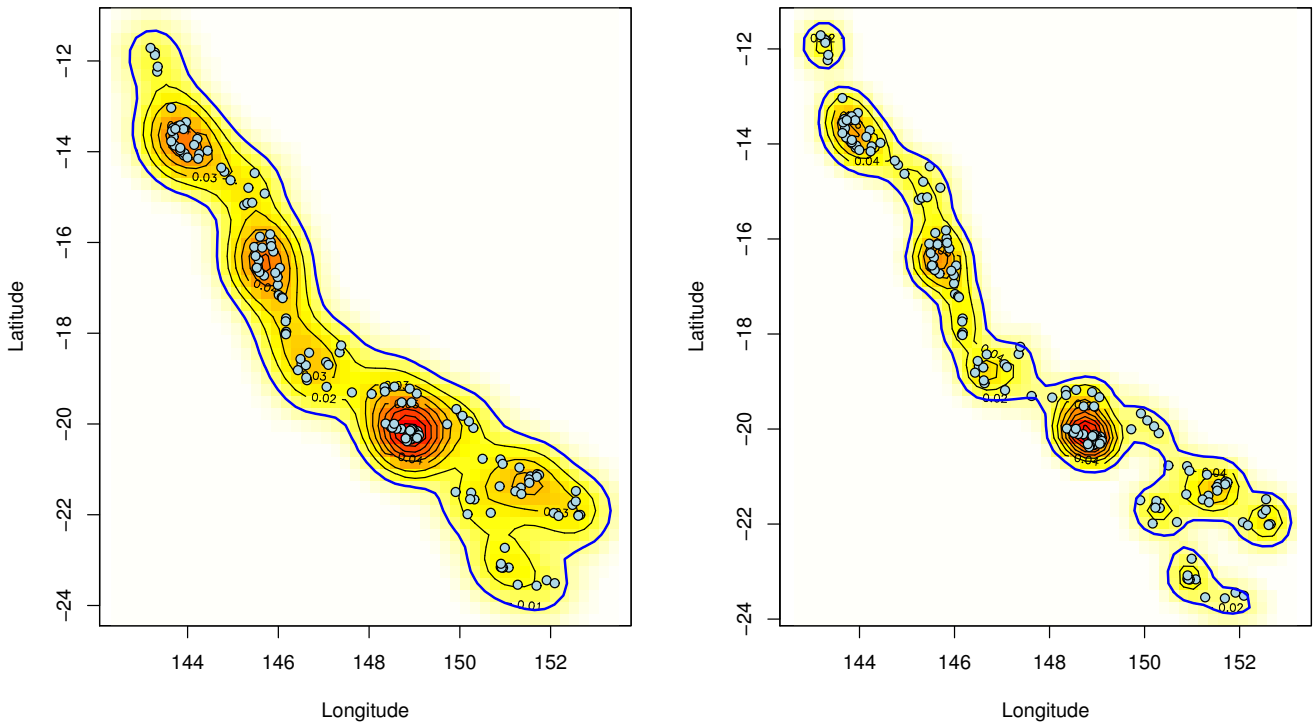

Figure 3: Density plots and boundaries of grids of soft coral sites. The two outputs are (left) for default settings and (right) a smaller value of "eps" that generates multiple boundaries

In addition to being useful for grid-generation, we can also use these tools for modelling the density of observed occurrences of events or of sampling (as is the case above). These can be of great interest and an example will be included later on.

For for modelling the richness of soft corals, we will not however use either of the two previous methods of grid generation. Rather, we will use one of the pre-defined grids that comes with `geo`. The resolution of the grid is 0.01°. If you do not supply a grid to the modelling process then one will be automatically generated, though it may be sub-optimal if the locations of your sites form clumps or form concave shapes. However, if you do not choose a good grid, then trimming based on estimated uncertainty of the predictions can be used. Indeed, trimming is strongly advised even if the initial grid is a good one. In this example we will use a grid specially prepared for the GBR and included in the `geo` package

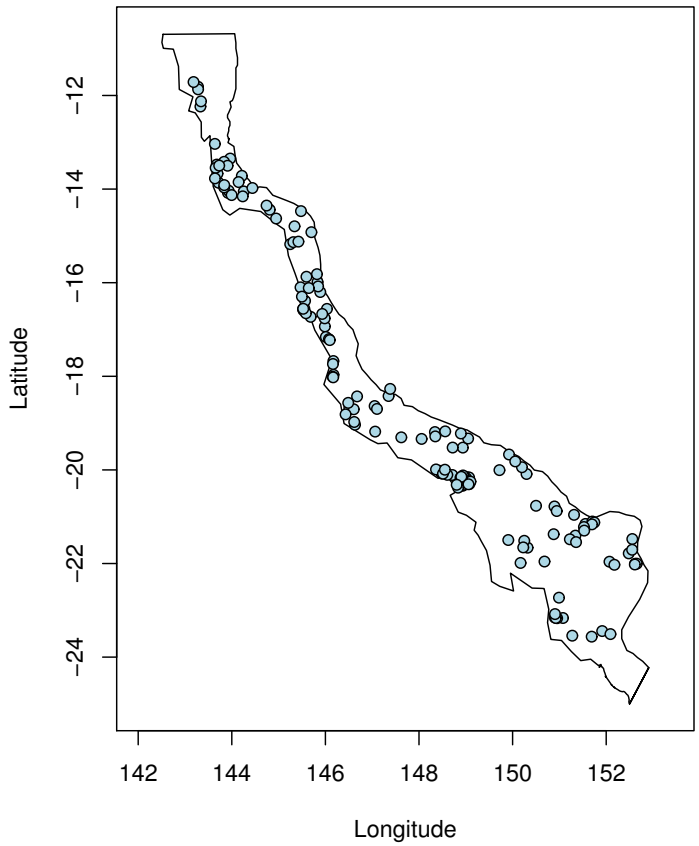

Figure 4: Boundary of grid used for spatial analysis of soft coral surveys

2.3 Model the spatial distribution of soft coral richness using `geoFit`

We will model the spatial distribution of soft coral richness using generalized additive models (GAMs). This is done using the function `geoFit`, part of the `geo` package. As a minimum, we need to specify the model and the dataset. The default grid will be used as described above. The call to the function is thus:

```
fit <- geoFit(richall~s(long,lat),data=softcorals)
```

The formula has the response (richall) on the lhs and a smooth (s) using the predictors long and lat on the right. The model is fitted using the `gam` function and the degree of smoothing is selected by cross-validation, and this is usually effective. However, this can be varied as described in the help documentation of `gam` in the `mgcv` package. The grid001.aa is part of the `makegrid` package.

The output of `geoFit`, in this cases assigned to `fit`, comprises:

1. An R object of class `gam` that includes all the components of the fit. We use it later to generate the additional outputs such as fitted surfaces for the mapping system and KMLs
2. Numerical outputs that provide information about the fitted model
3. Three plots that help us diagnose and interpret the fit of the model. These are stacked on top of each other for easy access by clicking of viewable portions of the overlapping frames. There is no need to drag them around the desktop for viewing

The numerical output includes estimates of the overall mean richness (21.93), and the smooth surface with effective degrees of freedom of 19.95. The model explains 56.0% of the variation of richness and the "GCV (generalized cross-validation) score" is a measure of fit that can be used for model selection (smaller is better). The "Scale estimate" is an estimate of the mean square error.

```
Family: gaussian
Link function: identity
Formula:
richall ~ s(lat, long)
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  21.9133     0.4922   44.52   <2e-16
Approximate significance of smooth terms:
              edf Ref.df     F  p-value
s(lat,long) 19.95  24.38 6.144 2.19e-12
R-sq.(adj) = 0.492   Deviance explained = 56%
GCV score = 42.243  Scale est. = 36.343    n = 150
```

The three plots show various aspects of the fit of the model and are used (a) to assess fit of the model, and (b) to make decisions as to modification of the model and the predicted surface.

1. Model Diagnostics: This plot shows four panels (see below) for model diagnosis
   (a) A QQ normality plot of points that should form a straight line if the residuals are normally distributed
   (b) A plot of absolute residuals (their size ignoring their sign) against the fitted values. This is used to assess if the variance of residuals depends on the size of the fitted values
   (c) A boxplot that can be used to assess the symmetry of the distribution of residuals and can also detect large positive or negative residuals

(d) A semivariogram that is used to assess spatial correlation between residuals. Smaller values for short distances suggest that the residuals are positively spatially correlated at short distances, contrary to the model assumptions of independence

2. SE Distributions: Boxplots of estimated standard errors for data points (left) and for predictions onto the grid (right). Outlier levels of 0:5 SEs are indicated for the analysis of grid points (centre). Large SEs indicate areas of unreliability and can be trimmed from the grid by subsequent processing with `geoTrim`

3. Images of Predictions & SEs: The two image plots show the spatial distribution of the predictions and their standard errors.

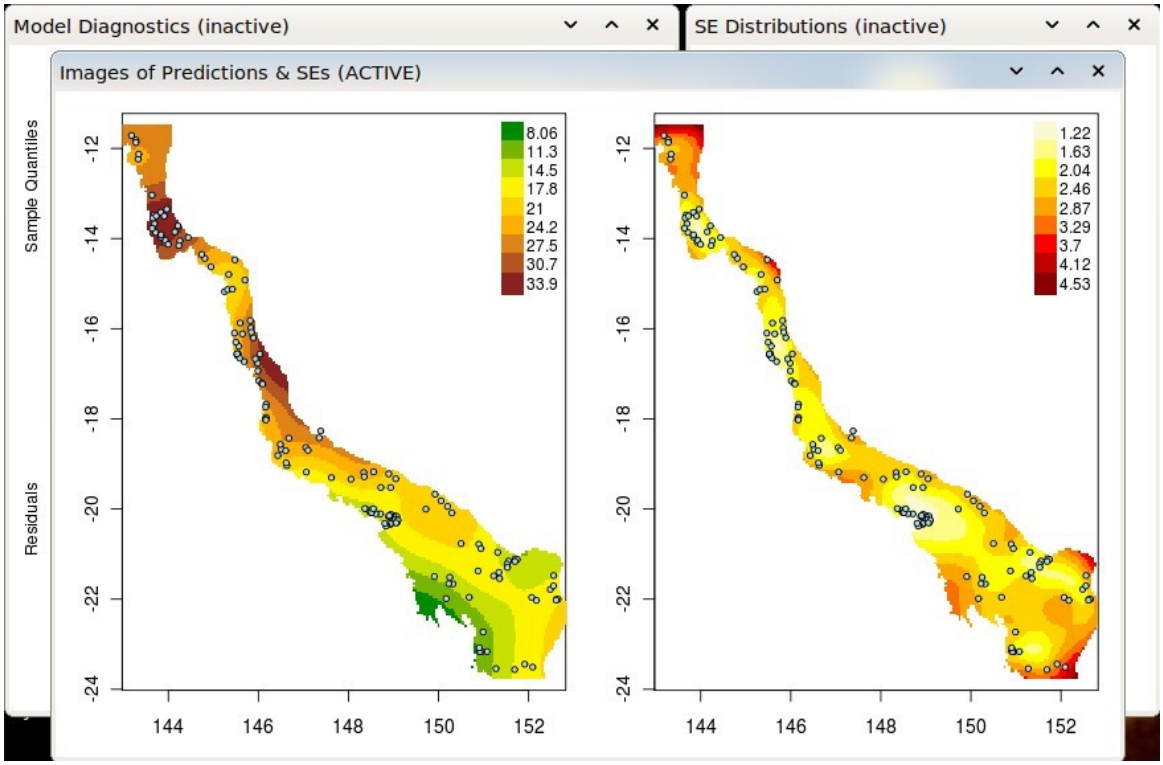

Figure 5: Screen capture of the three graphics plots for diagnosing the model fit

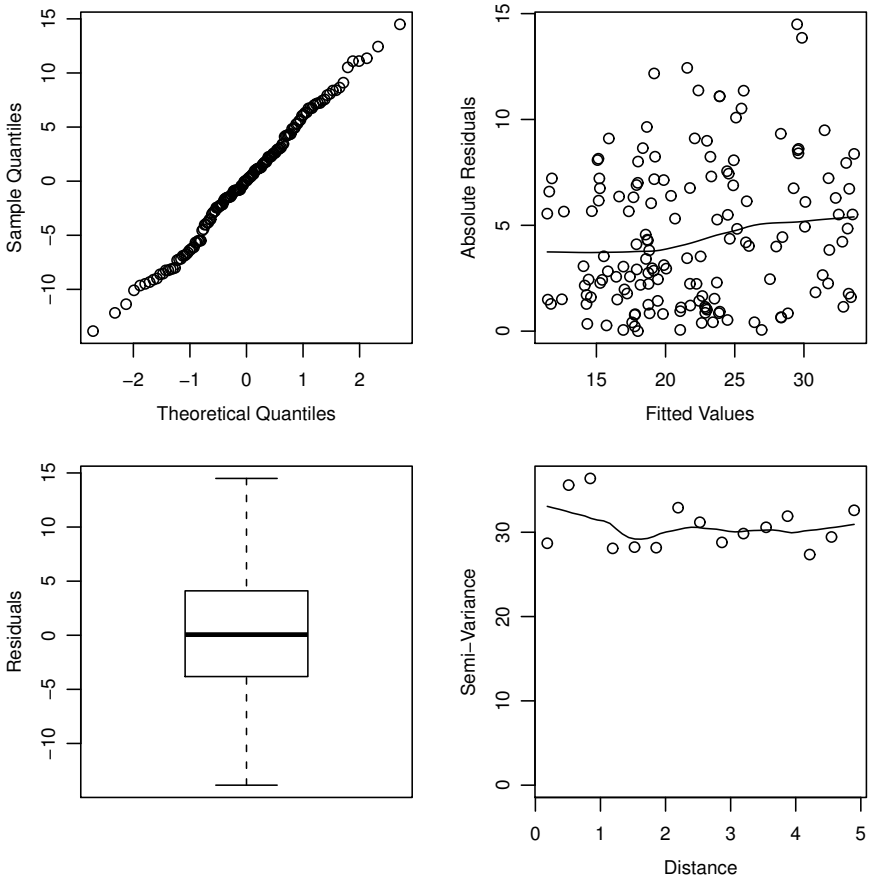

Figure 6: Model Diagnostics: Four plots for analysis of residuals

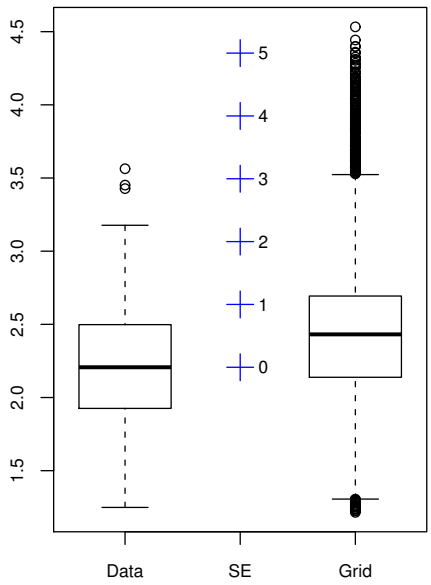

Figure 7: Boxplots of SEs for data points (left) and grid points (right). Outlier levels of 0:5 SEs are indicated for the grid points (center)

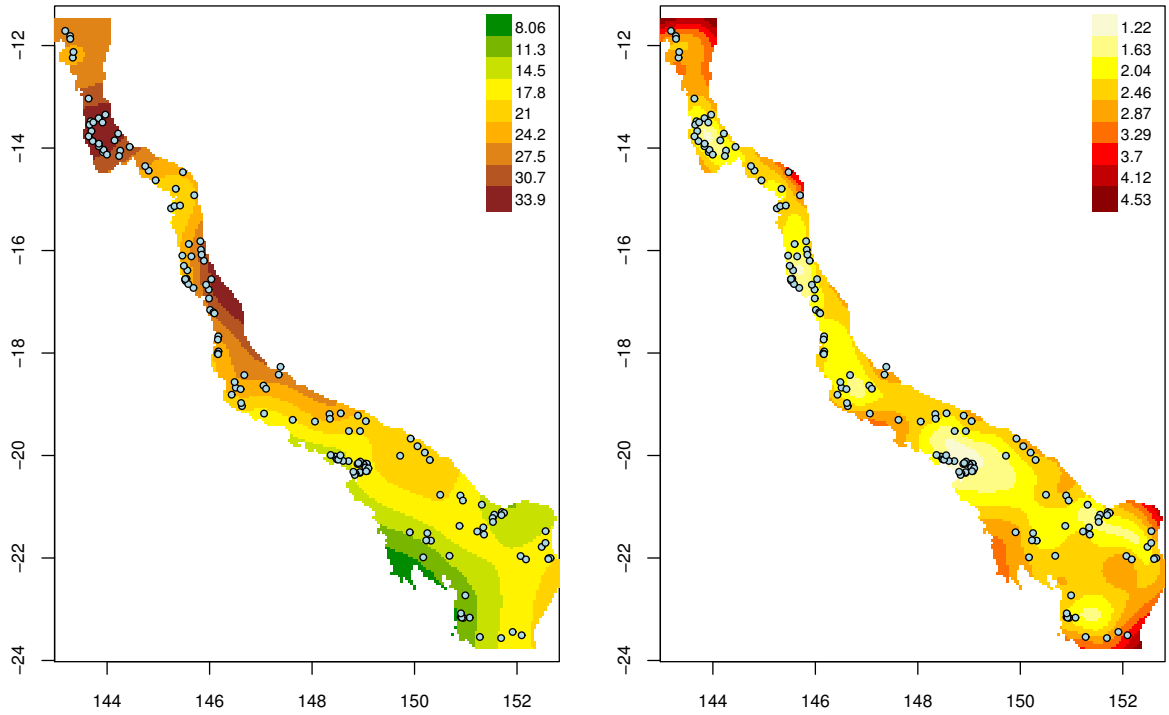

Figure 8: Image plots of of the grid estimates of richness and the standard errors for the model using longitude-latitude coordinates

2.4 Using an alternative spatial coordinate system

Although latitude-longitude are most widely used as the system of coordinates for spatial modelling, there are circumstances when alternative systems can be more effective at capturing spatial structure. The GBR is such an instance, where its natural boundaries can be used as the basis of an alternative system. Latitude and longitude values can be converted into relative distance across the continental shelf and along the GBR World Heritage Area. Relative distance across is 0 on the coast and 1 on the outer edge of the continental shelf, and relative distance along is 0 on the southern end of the GBR (25°S) and 1 on the northern end (18°S). The coordinates of the acrossalong system are locally orthogonal and run approximately at right angles and parallel to the coast. This coordinate system has consistently proved to be more efficient and interpret-able than latitudelongitude for analysis of GBR spatial data.

We show the use of this system below. In the call to `geoFit` we replace lat and long with across and along and the new coordinates are calculated for us. Thus the call becomes:

```
fit <- geoFit(richall~s(across,alongS),data=softcorals,grid=grid001.aa)
```

The output follows:

```
Formula:
richall ~ s(across, along)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  21.9133     0.4549   48.18   <2e-16
Approximate significance of smooth terms:
                 edf Ref.df     F  p-value
s(across,along) 19.98  24.54 8.072 3.00e-16
```

```
R-sq.(adj) =  0.566   Deviance explained = 62.4%
GCV score = 36.081  Scale est. = 31.034    n = 150
```

The model performs better that the latitude-longitude model with 62.4% of deviance explained and with a GCV score of 36.08.

We now take the additional step of selecting the optimum ratio of along to across based on the generalized cross-validated estimate of model error. This is only relevant when the smoothers in the model are not isotropic invariant. The call becomes:

```
fit <- geoFit(richall~s(across,along),data=softcorals,aauto=T)
```

And the output first presents the results of the optimisation of across-along ratio.

```
Calculating across-along
1  2  3  4  5  6  7
       aak      gcv
0.25  0.25 38.64847
0.5   0.50 37.74852
1     1.00 37.71917
2     2.00 34.90218
4     4.00 36.08096
8     8.00 37.32851
12   12.00 38.38561
Best aak = 2.727273
```

The fit of the spatial model follows:

```
Family: gaussian
Link function: identity
Formula:
richall ~ s(across, along)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  21.9133     0.4355   50.32   <2e-16
Approximate significance of smooth terms:
                  edf Ref.df     F p-value
s(across,along) 24.02  27.48 8.423  <2e-16
R-sq.(adj) =  0.602   Deviance explained = 66.6%
GCV score =  34.14  Scale est. = 28.445    n = 150
```

This model performs better that the non-optimised across-along model with 66.6% of deviance explained and with a GCV score of 34.14. The optimum scaling of across-along was 2.72; i.e. the along coordinates are scaled by a factor of 2.72.

Finally we omit points from the predicted grid with SE > 3. We can do this in two ways:

1. the efficent way is to use geoTrim :

   ```
   fit <- geoTrim(fit,se.omit=3)
   Omitting 7.1 % pixels on high SEs
   ```

2. or, inefficemtly, rerun geoFit with se.omit $= 3$

   ```
   fit <- geoFit(richall~s(across,along),data=softcorals,grid=grid001.aa,
   aauto=TRUE,se.omit=3)
   .........
   Omitting 7.1 % pixels on high SEs
   ```

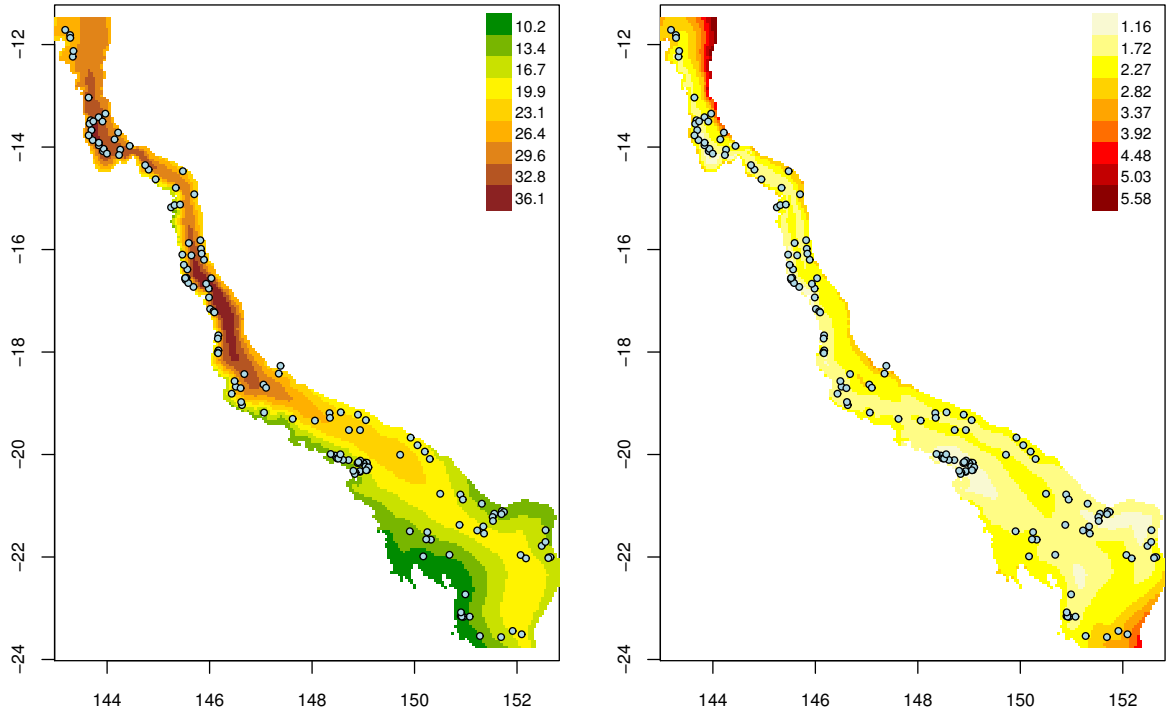

Figure 9: Image plots of of the grid estimates of richness and the standard errors for the model using across-along coordinates

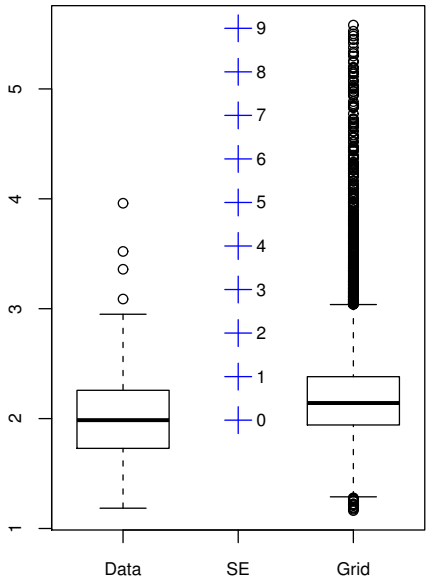

Figure 10: Boxplots of residuals for data points (left) and grid points (right). Outlier levels of 0:9 SEs are indicated for the grid points (centre)

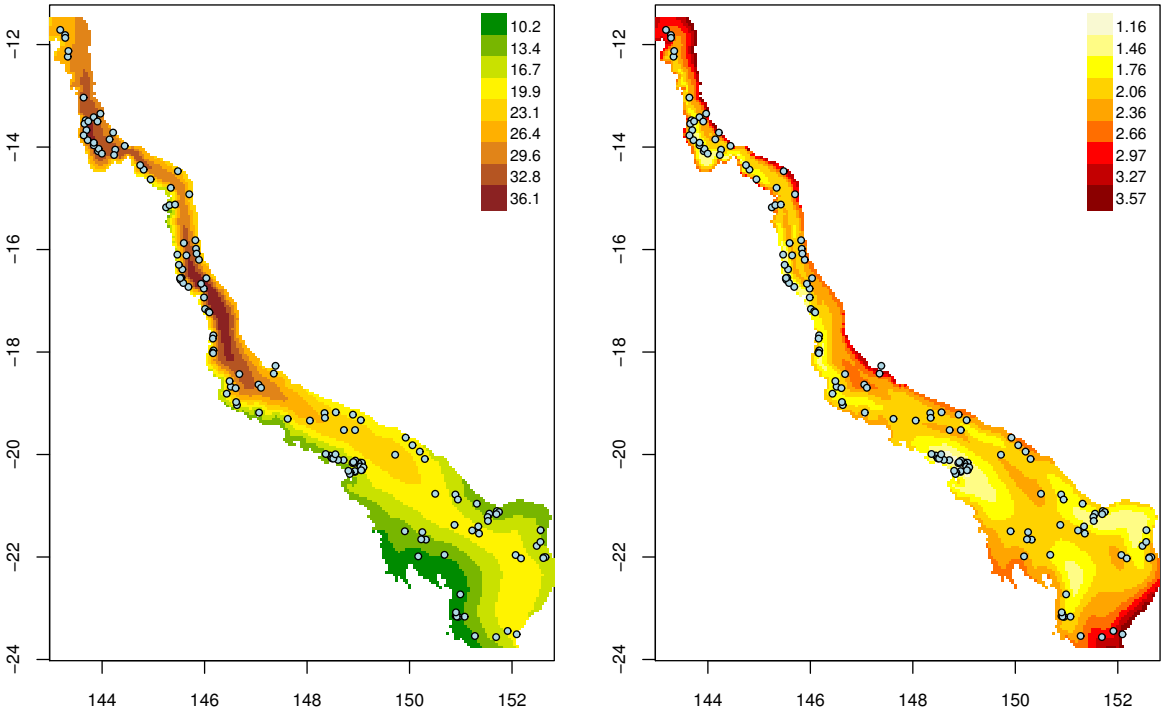

Figure 11: Image plots of of the grid estimates of richness and the standard errors for the model with trimming of outliers >3 deviations

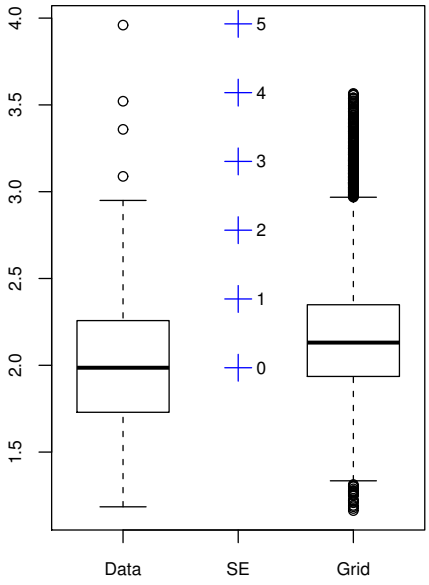

Figure 12: Boxplots of residuals for data points and grid points for the across-along model with outliers of >3 deviations removed

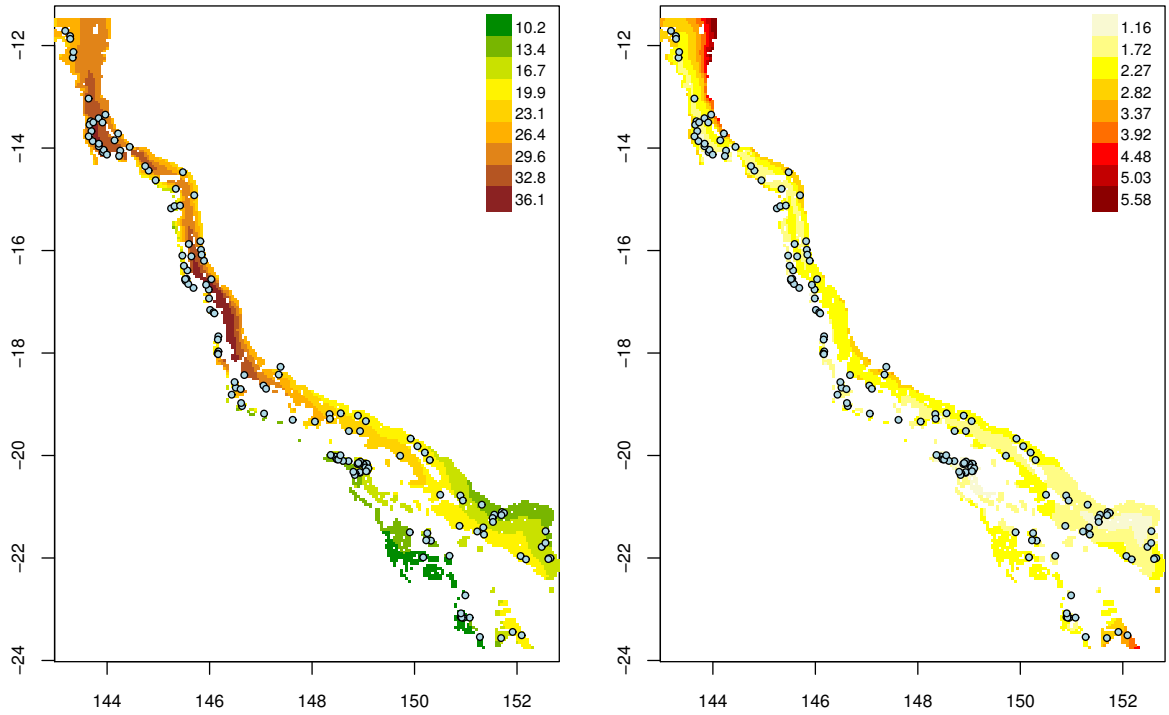

Figure 13: Image plots of of the grid estimates of richness and the standard errors for the model using across-along coordinates with the non-reef regions masked out

In previous fits we have implicitly used the default grid (`grid005.aa`). To mask data we need to specify both the grid and the mask. This is done as follows:

```
fit<-geoFit(richall~s(across,along),data=softcorals,grid=grid005.aa[reefclip005,])
```

`reefclip005` is the mask and selects only the points of the grid as shown in the plots above.

The visual impression of these masked images can be quite different to their unmasked counterparts. They have the advantage that we don't draw conclusions about richness of soft coral in regions where there are no reefs! However, the broader spatial patterns are, to some degree, less striking. Of course these graphics are largely intended as analysis aids rather than publication, and we shall see alternative ways of displaying these spatially modelled surfaces later on.

2.5 An exercise for readers

The variable `richall` is the sum of `richhetero` and `richphoto`; the heterotrophic and phototrophic feeding corals. Model `richhetero` and `richphoto` and you should see strong differences in the two spatial distributions.

3 Using Different Smoothers

Generalized additive models (GAMs) are a very powerful smoothing technique. They are flexible in the range of smooths that they can use and the degree of smoothness can be selected by cross-validation. Also they can model non-normally distributed data using logistic, log-linear and inverse functions that link the response to the predictors, and they can handle error structures in addition to gaussian (normally distributed) data.

However, there are types of data to which GAMs are not well-suited. For example, some data include many outliers and robust methods are needed. In addition to GAMs, two other methods are included in the `geoFit` function. These are:

1. Loess Smoothers
2. Boosted Trees

Both of these smoothers can be used through `geoFit`, with only minor changes to the process of defining the models for GAMs, and the model assessment procedures remain the same.

3.1 Loess Smoothers

The class of smoothers known as "local smoothers" that use a smoothing window, have proven to be very useful. To run a loess fit, we specify the model without the "s()" format. For example:

```
fit <- geoFit(richall~across+along,data=softcorals,model="loess")
```

This will fit the default model that is a quadratic (df=2) local smoother and the span of the window is 0.75 of the data.

```
fit <- geoFit(richall~across+along,data=softcorals,model="loess")

Locale = GBR
Calculating across-along
Call:
loess(formula = form, data = data, normalize = FALSE, surface = "direct")

Number of Observations: 150
Equivalent Number of Parameters: 9.17
Residual Standard Error: 6.045
Trace of smoother matrix: 10.66

Control settings:
  normalize:  FALSE
  span     :  0.75
  degree   :  2
  family   :  gaussian
  surface  :  direct
```

The degrees of freedom can be 2,1 and 0 and the window can be anything from 10 (very stiff) to a small fraction, say 0.1. The smoother can be made more wiggly by decreasing the window size or increasing the degrees of freedom. Also the loss function can be "gaussian" or "symmetric"; the latter being a robust smoother that is little influenced by outliers.

To then make it more robust, locally linear but with a small smoothing window use:

```
fit <- geoFit(richall~across+along,data=softcorals,model="loess",
family="symmetric",df=1,span=0.25)
```

If we rerun the soft coral richness model using the loess smoother this is what we get: Unlike the GAMs, the degree of smoothness cannot be automatically selected using cross-validation of statistical tests. The two statistics that are useful however are ENP (the Equivalent Number of Parameters: 9.17 in this case) and RSE (the Residual Standard Error: 6.045). ENP is an estimate of the degrees of freedom in the model fit and RSE (or MSE : mean square error) is comparable across models and methods.

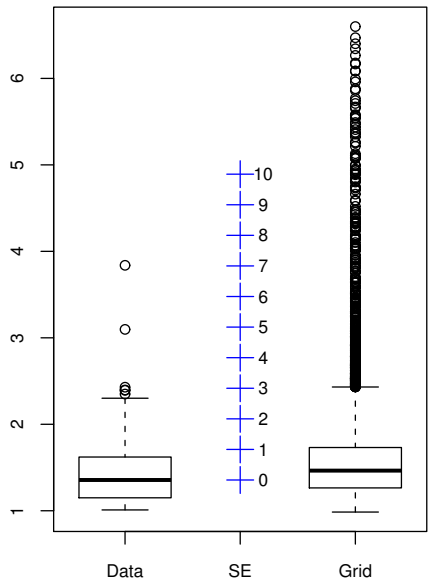

Figure 14: Model Diagnostics: Four plots for analysis of residuals

The model diagnostics suggest a good fit with no outliers amongst the residuals, some weak evidence of increasing variance of residuals with the size of fitted values, and no evidence of spatial correlation. Certainly, one would not use a robust loess model given these good model diagnostics.

There are some predicted values with very large standard errors, corresponding to the northern offshore region, and these were then screened out for the final plots. The patterns of predicted richness look very similar to those of the GAM model, as one would expect.

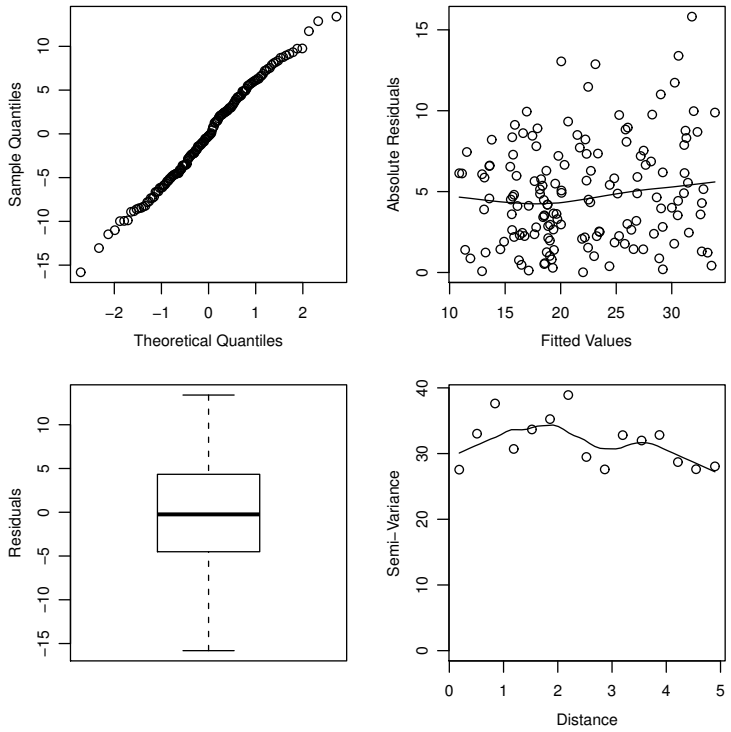

Figure 15: Boxplots of residuals for data points and grid points for the across-along model with outliers of >3 deviations removed

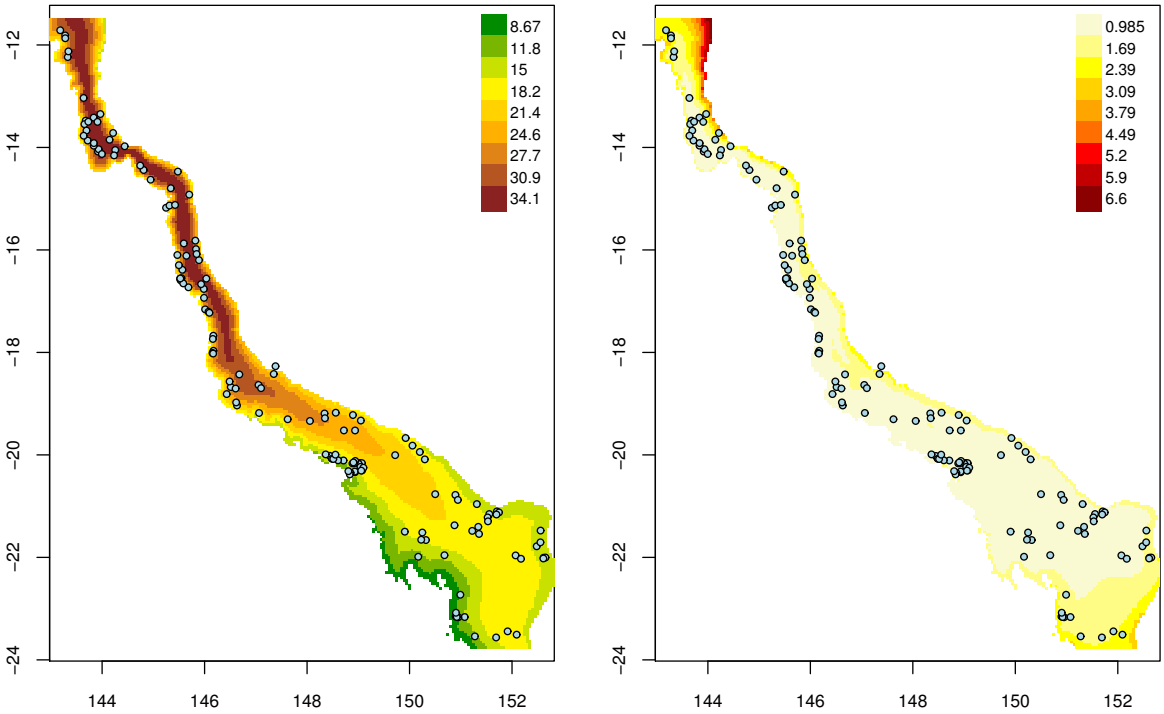

Figure 16: Image plots of of the grid estimates of richness and the standard errors for the loess model using across-along coordinates

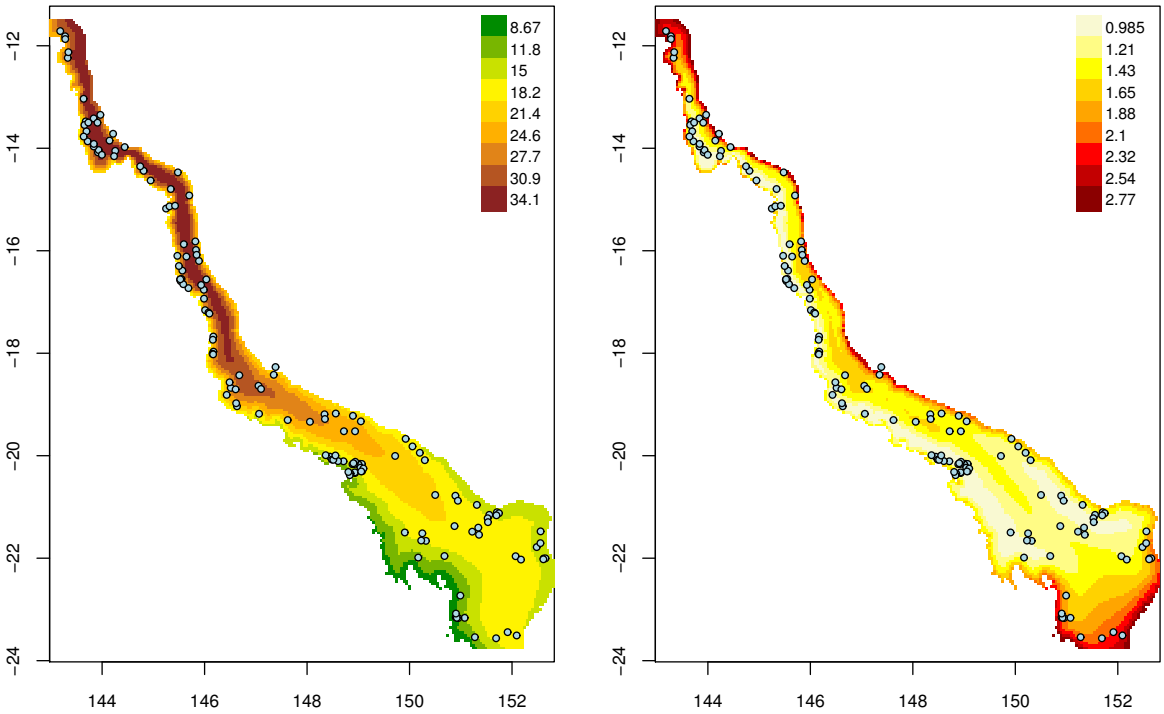

Figure 17: Image plots of of the grid estimates of richness and the standard errors for the loess model using across-along coordinates and with outliers >3 SE screened out

3.2 Aggregated Boosted Trees

Boosted trees are a statistical learning method that give accurate predictions and simple model interpretations for both regression and classification analyses. They can deal with many types of response variables (numeric, categorical, and censored), loss functions (Gaussian, binomial, Poisson, and robust), and predictors (numeric, categorical). Interactions between predictors can also be quantified and visualised. A boosted trees is a sequence of regression trees, with each one in the sequence being fitted to the residuals of its predecessor. Eventually the residuals become very small and this collection of trees that forms the boosted tree explains the data very well. It may be over-fitted however and thus predict poorly. Hence we need to find the best size of collection (i.e. how many individual trees) are best for prediction. We can do this by cross-validation.

A new form of boosted trees, namely aggregated boosted trees (ABTs) reduces prediction error relative to boosted trees as described above, and they also provide additional outputs (e.g. SEs) that are useful for model interpretation. ABTs are simply a collection of boosted trees where each BT is based on a cross-validation sample of the data.

A paper and talk on ABTs that outline the theory and practice fitting ABT models are available with this document.

To fit ABTs for the softcoral spatial analysis we simply change the model argument of **geoFit**:

```
fit <- geoFit(richall~across+along,model="abt",data=softcorals)
```

Basic information about the model and the fit are provided:

```
Calculating across-along

 N cases = 150 : N trees = 1000 : Depth = 3 : Minimum node size = 2
 Distribution = gaussian : Shrinkage = 0.01 : Bag Fraction = 0.5
 CV-folds = 5  : Best size = 655 : Best error = 34.77

     var rel.influence
1  along      61.44902
2 across      38.55098

Best ave min n's = 540 635 189 655 559
Best ave pred errs = 30.1 46.7 26 36.5 34.5
Best min ave pred err = 34.8  : best ave min n tree = 655
```

The output details that a collection of 1000 trees have been used for each of the 5 cross-validations. The best size of tree is given by the number of trees (686 in this case) that minimises the prediction error of the model (squared error = 34.77: equivalent to 5.89 root mean square error). If the "Best size" was the same as the "N trees" then we would increase the number or increase the shrinkage The last 3 lines of the output show the convergence of the boosted trees for each of 5 cross-validation sets. The best size of tree varies from 189 to 655 component trees.

Outliers can be omitted as was illustrated for GAM and loess models.

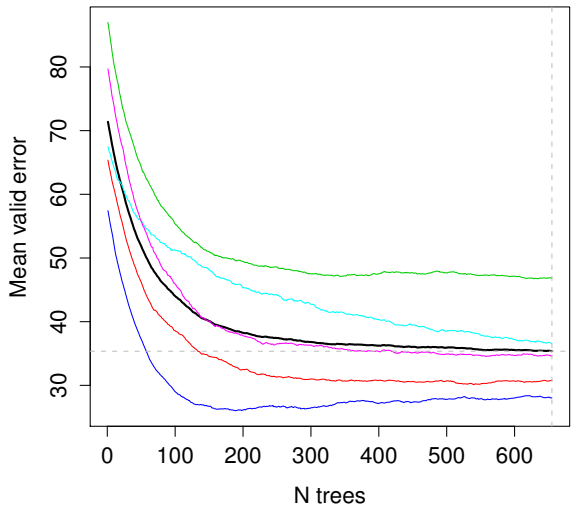

Figure 18: Prediction error as a function of tree sizes

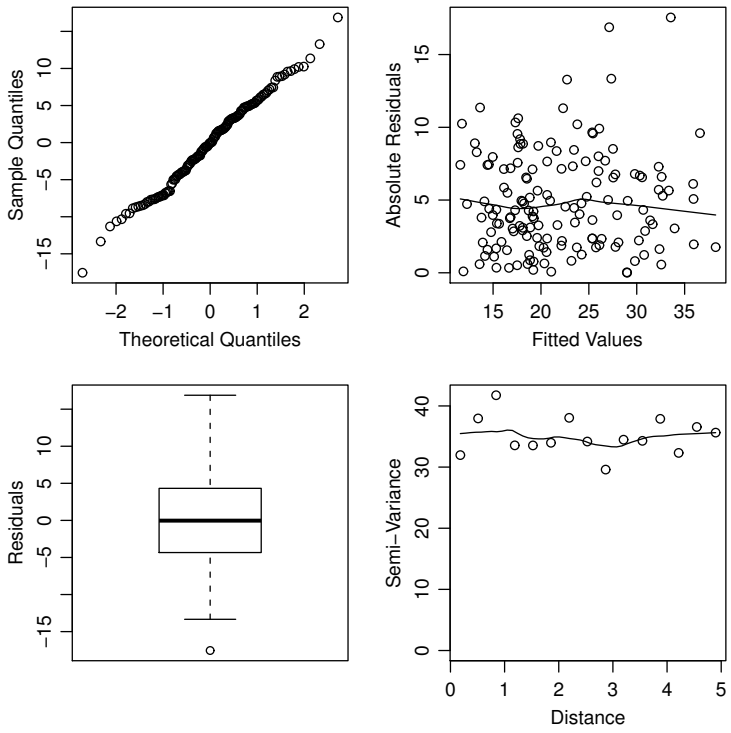

Figure 19: Model Diagnostics: Four plots for analysis of residuals

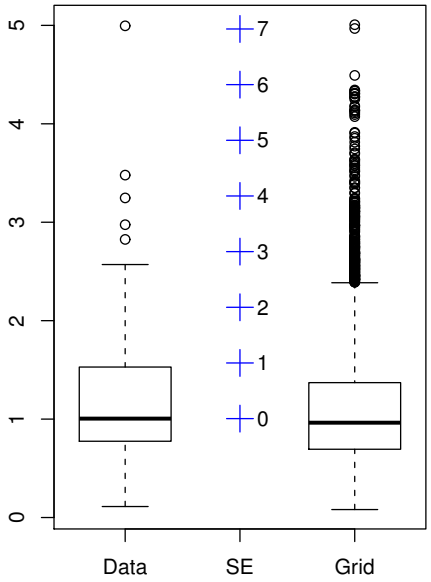

Figure 20: Boxplots of residuals for data points and grid points for the across-along model

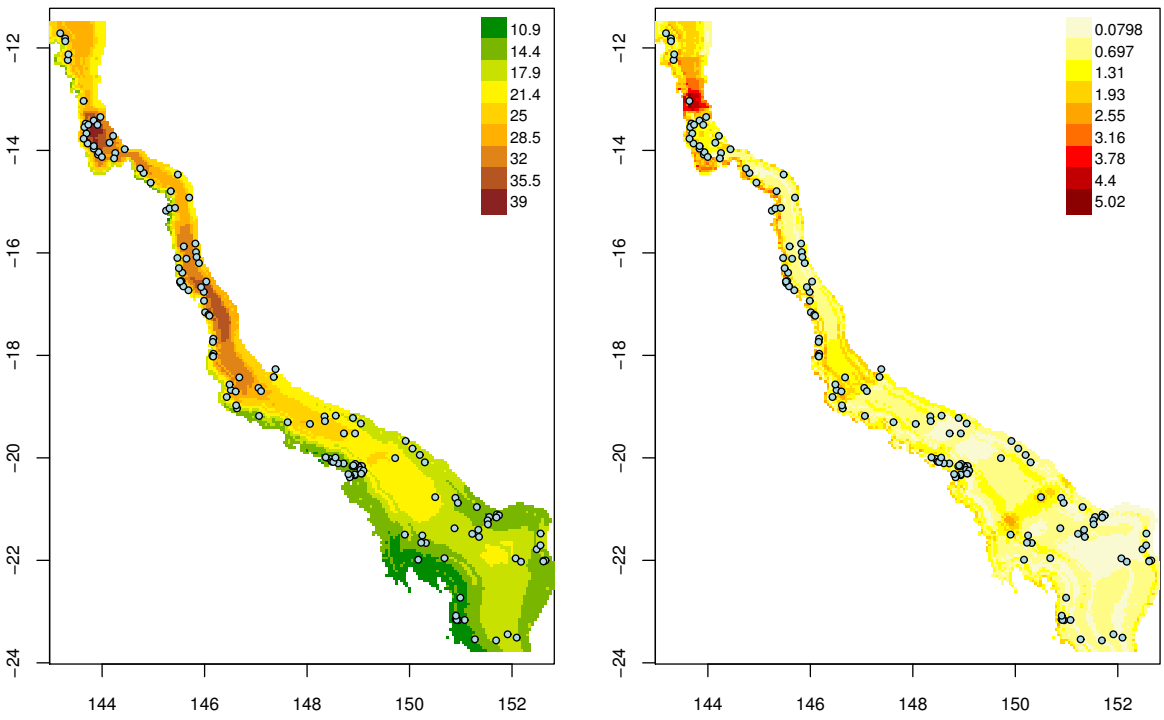

Figure 21: Image plots of of the grid estimates of richness and the standard errors for the abt model using across-along coordinates

4 Generating KMLs and Other Graphics

4.1 Outputs: KMLs, Atlas Data and Graphics using `geoFit2KML`

We can now use the gridded predictions as the basis of the outputs. The KMLs will include predicted surfaces, legends and information about the data set, and are highly customisable in terms of colour ramps. The output for the Atlas are used to generate surfaces for the interactive mapping system. The fitted surface of predictions are also output as PNG and/or EPS maps. If you wish the logo (e-Atlas or Ningaloo-Atlas) to appear on the KML then you must have the the correct file (called "logo.png" unless otherwise specified in the call to `geoFit2KML`) in your working directory.

The KMZ file will be generated in a folder called WORK off your current working directory and will be named with a format of "data set"-"response"-2.kmz, and is thus softcorals-richall-2.kmz for this example. This compressed file will contain the KML code and PNGs for the layers, legends and logo.

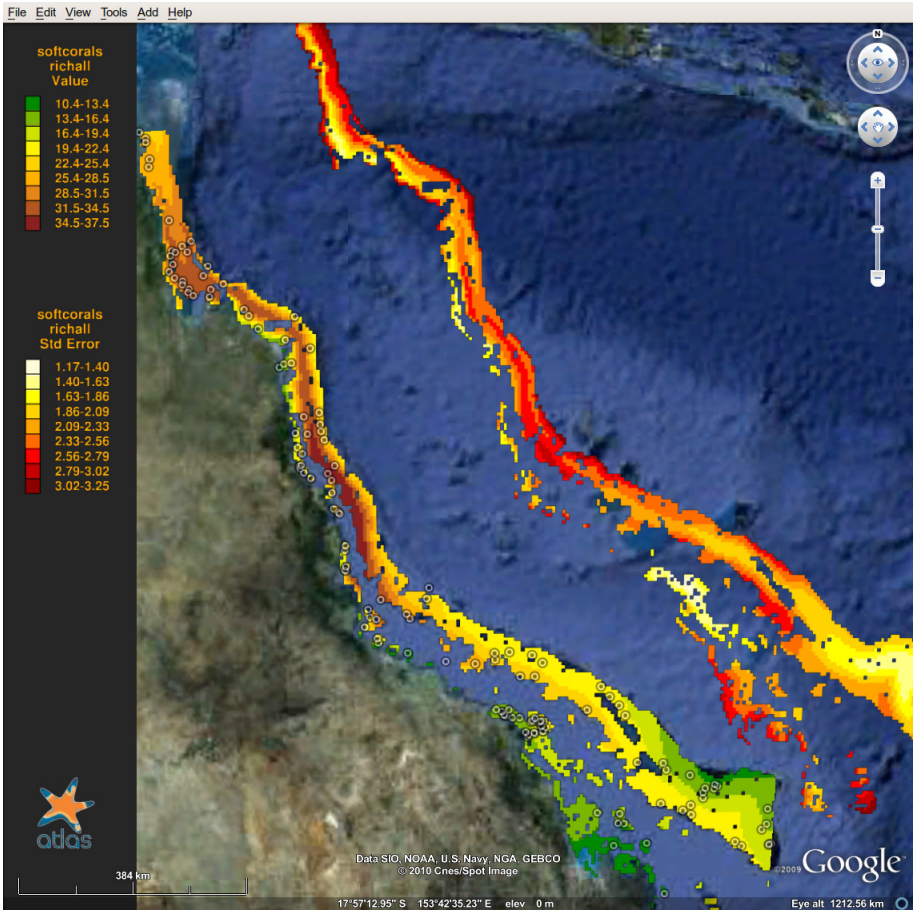

Figure 22: Screen capture of Google Earth images showing estimated soft coral richness (left) and standard errors (right)

To generate the outputs using the default settings, simply use `geoFit2KML(fit)`. You should see the following out that summarises the generation of the scales and lists all the files added to the KMZ file.

```
Creating work directory

Data set:  softcorals
Response:  richall
Units:  Value
Using provided fit
```

```
Type =  Value
lims :  10.45 13.46 16.47 19.48 22.49 25.49 28.5 31.51 34.52 37.53
Breaks:  10 13 16 19 22 25 29 32 35 38
Type =  Std_Error
Breaks:  1.2 1.4 1.6 1.9 2.1 2.3 2.6 2.8 3 3.3
5  files
  adding: softcorals-All.kml (deflated 92%)
  adding: softcorals-Std_Error-Legend.png (deflated 22%)
  adding: softcorals-Std_Error.png (deflated 74%)
  adding: softcorals-Value-Legend.png (deflated 19%)
  adding: softcorals-Value.png (deflated 75%)
  adding: softcorals_map.png (deflated 21%)
  adding: logo.png (deflated 2%)
  adding: sidebar.png (deflated 10%)
```

There are many parameters that your can vary with `geoFit2KML` and these are all detailed in Section 4: "R Package Documentation". These include the capacity to specify your own color ramps for the images, and to generate high quality graphics of the spatial distribution of the response.

Once running in Google Earth, the KMLs generated by `geoFit2KML` can be customised in many ways using the control buttons of the KML in the sidebar. Layers can be hidden, points can be turned off/on or interrogated, the "About" box can be shown giving details on the data set, the sidebar can be removed. All layers can be varied in transparency.

4.2 Direct Generation of KMLs

The modelling tools presented so far have focused solely on the spatial distribution of a single quantitative response, and include diagnostics of the model fit. Subsequently these models could be modified by trimming large outliers and could also be used to generate KMLs.

There are also a series for more complex analysis including covariates and multi-response displays. These are currently limited to KML outputs, but this will be changes so that they follow the `geoG` format.

Examples of many mapping functions follow:

4.2.1 The spatial distribution and SEs using `geo2`

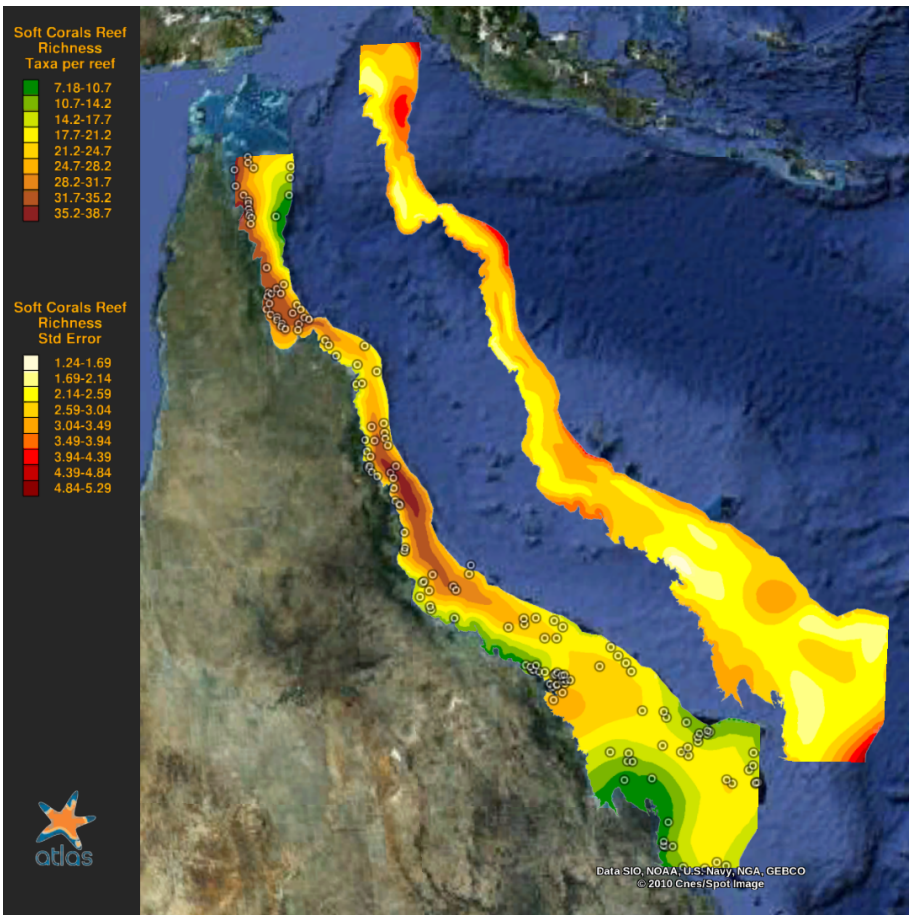

Figure 23: Screen capture of Google Earth images showing estimated soft coral richness (left) and standard errors (right)

4.2.2 The spatial distribution broken down by groups `geoG`

This model breaks the richness of soft corals into groups based on depth and spatially models each group. In this case we have three depths but the number of levels can be specified by the user.

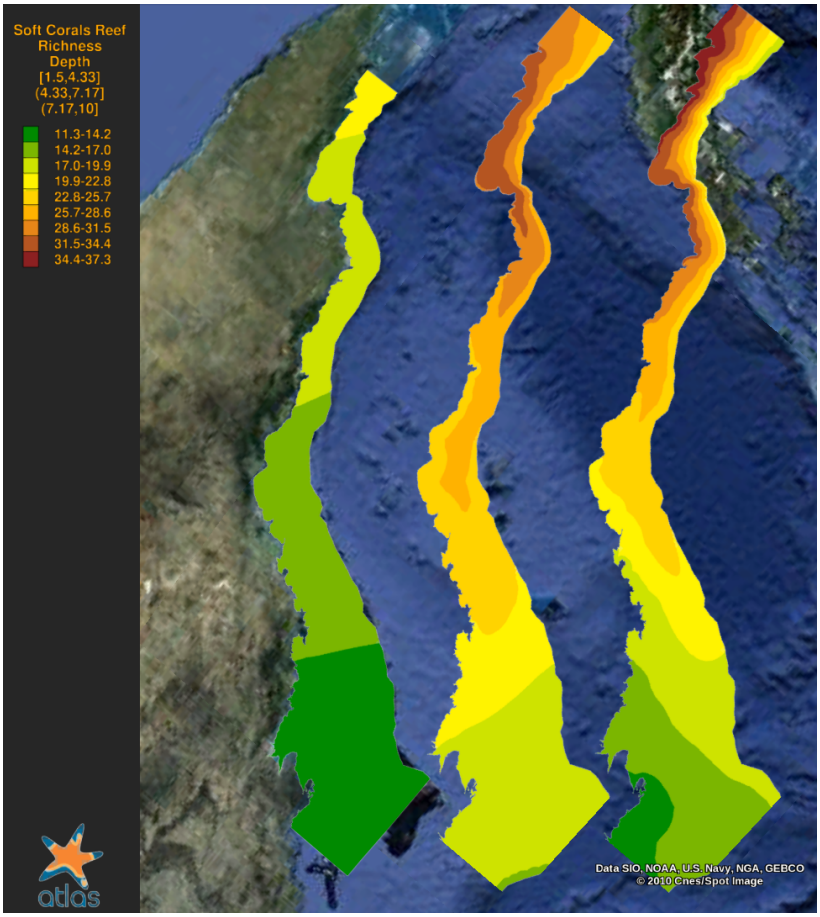

Figure 24: Screen capture of Google Earth images showing estimated soft coral richness broken down by depth

4.2.3 Side by side spatial distributions of 3 variables using `geoV`

This function maps the spatial distribution of many responses and places them side-by-side. The user selects the number of variables and the number of slots, and the variables displayed and their slots can be interactively varied by the user. Legends automatically adjust to these selections.

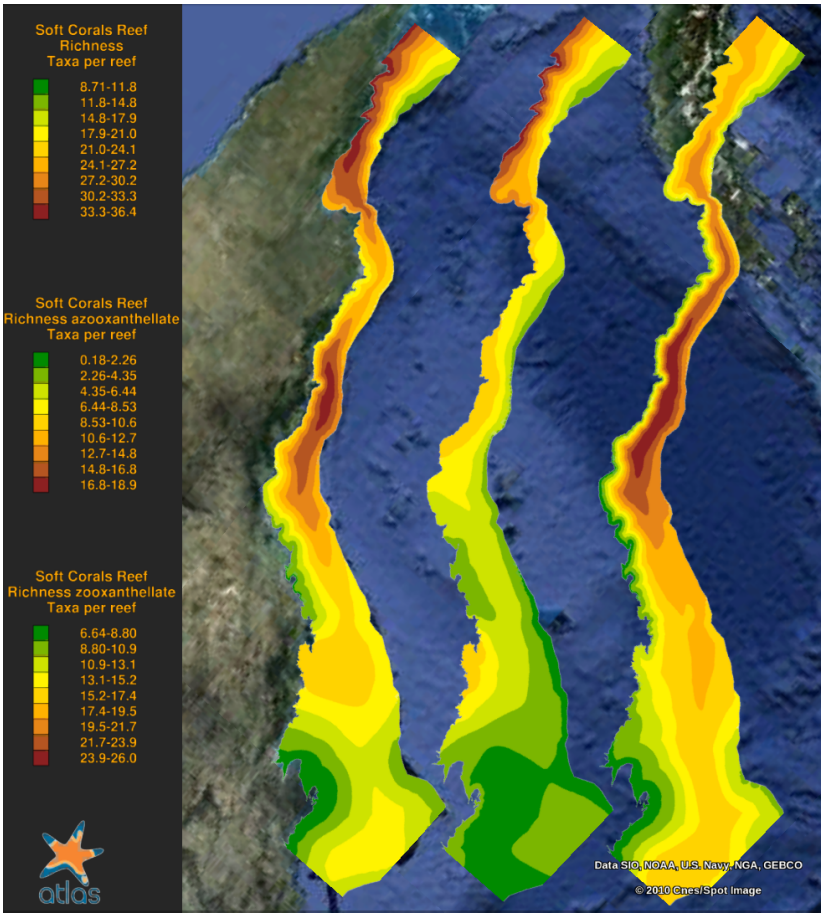

Figure 25: Screen capture of Google Earth images showing spatial distributions of three soft coral richness measures

4.2.4 Spatial gam using `geoR.gam`

This example models the dependence of soft coral on visibility and space (in this case across and along).

The four images are spatially smoothed values of:

1. soft coral richness
2. visibility
3. the spatial effects
4. the visibility effects

The inset plot shows the overall effect of visibility on soft coral richness

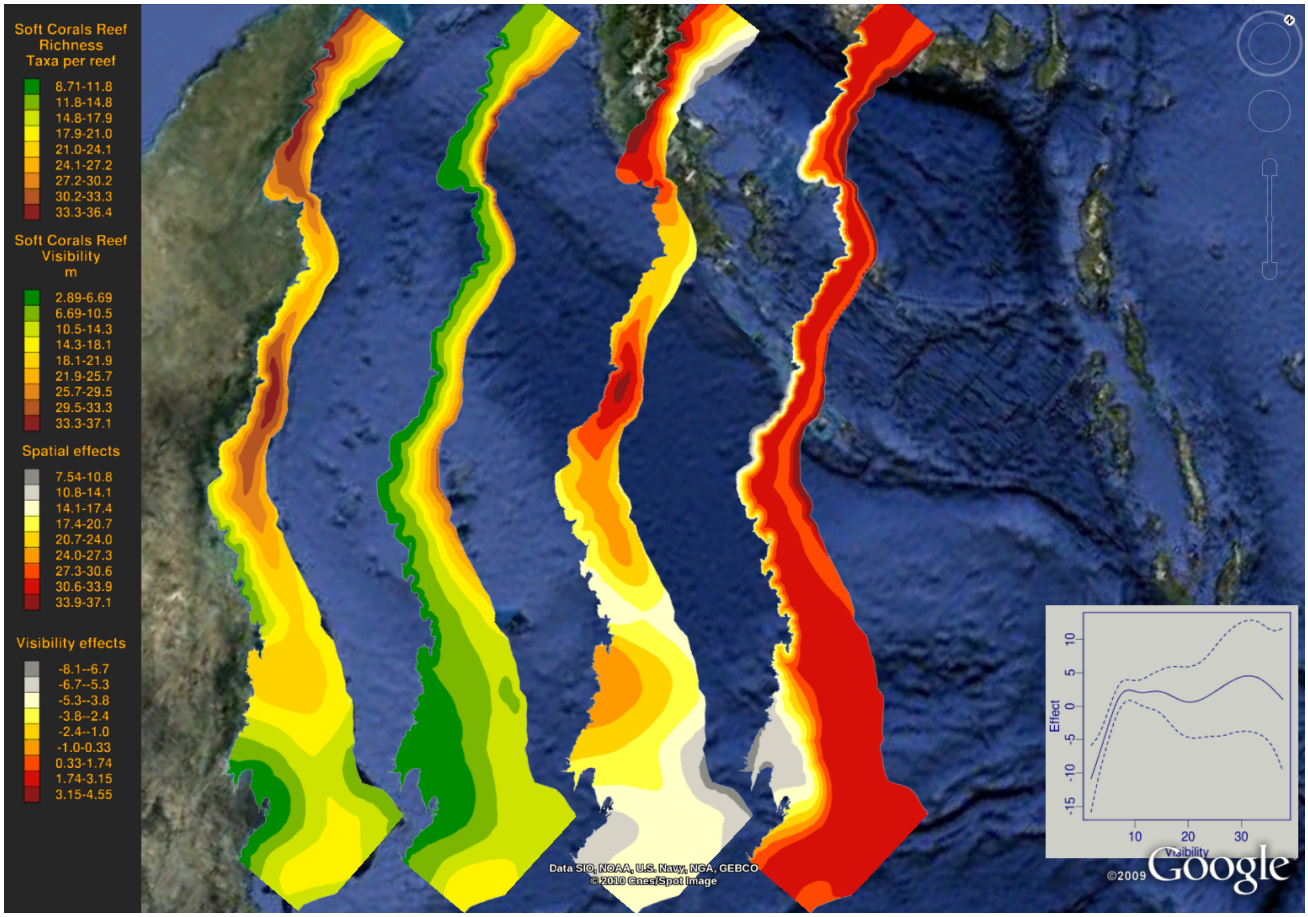

Figure 26: Screen capture of Google Earth images showing estimated soft coral richness (left) and standard errors (right)

4.2.5 Spatial correlation using `geoR.rank`

This example shows the correlation of soft coral and visibility over space (in this case across and along).

The three images are spatially smoothed values of:

1. soft coral richness
2. visibility
3. the correlation between the richness and visibility

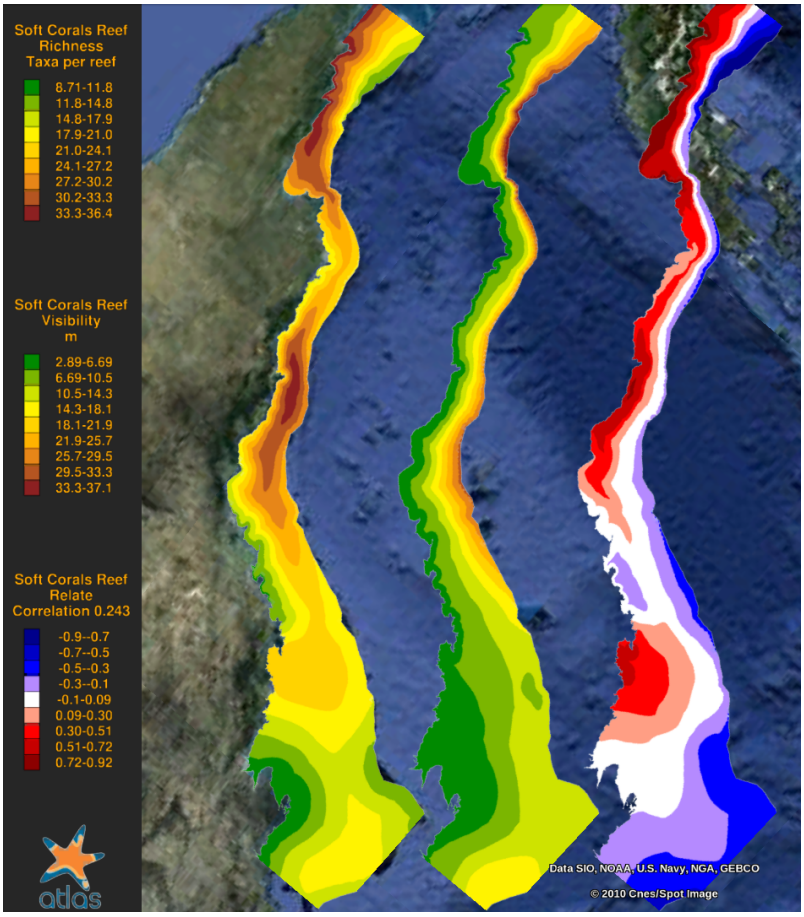

Figure 27: Screen capture of Google Earth images showing rank correlation plots

4.2.6 Spatial weighted regression using `geoswr`

Spatial weighted regression (SWR) is a statistical model that fits a local linear (usually) regression with points close to the location receiving greatest weight. The size of the local window can be varied, At each point in space a slope and intercept will be calculate that give the the fitted value at that point.

The example shows the SWR of soft coral richness as a function of visibility and space (in this case across and along).

The five images are spatially smoothed values of:

1. soft coral richness
2. visibility
3. the intercept
4. the covariate effect
5. the slope

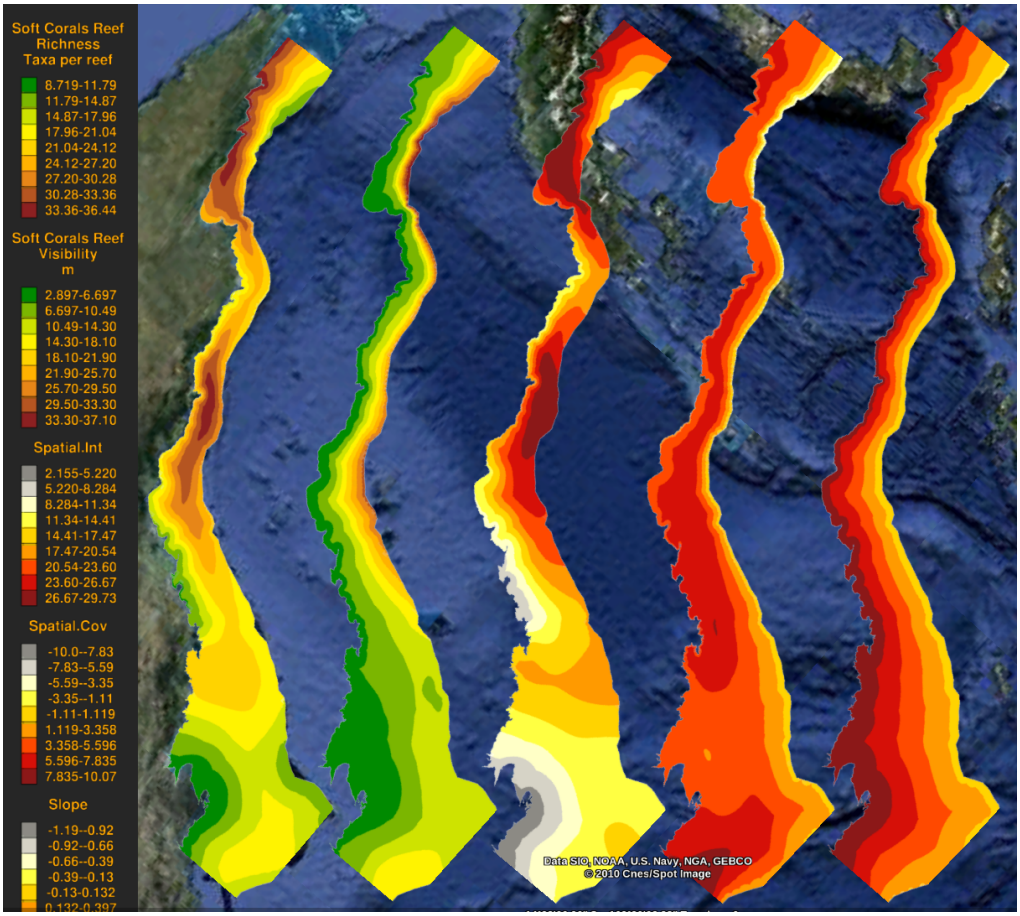

Figure 28: Screen capture of Google Earth showing spatially weighted regression outputs

4.2.7 Adding extras to Google Earth maps

Additional layers are available for the GBR to enhance you KMLs. These include the reefs, islands and land, the zoning plan and additional plain colored backgrounds. More layers will be added over time.

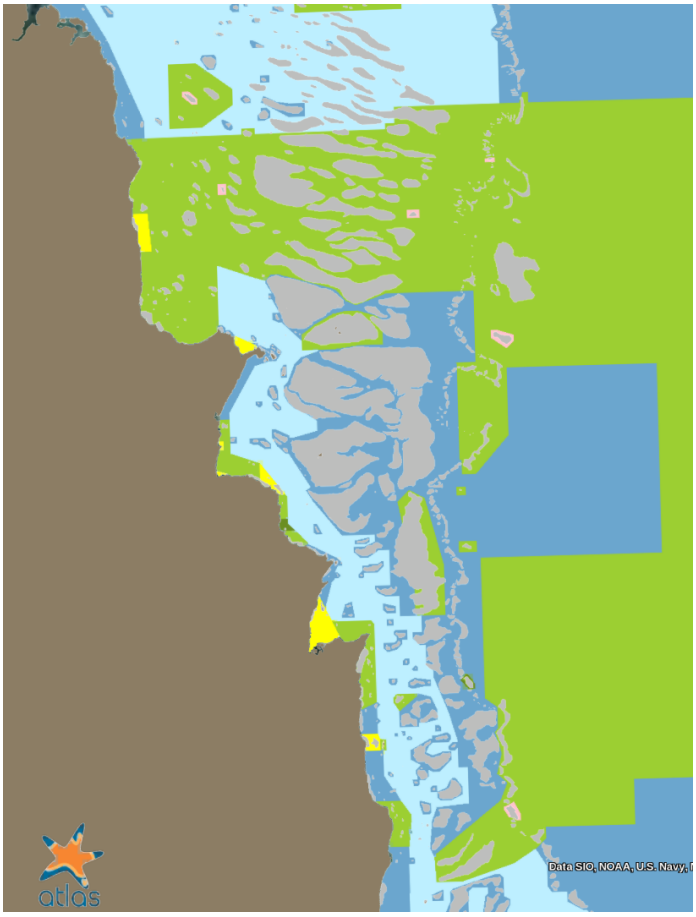

Figure 29: Screen capture of Google Earth showing additional layers of reefs, land, islands and GBR Zoning regions

4.3 Point Data

If you wish to generate a KML that simply displays the site locations and the values of the other variables in the data frame in pop-ups, then geoPoints will do this for you:

```
geoPoints(softcorals)
Output file :  softcorals.kml
```

Load the KML file into Google Earth and you will get the following display:

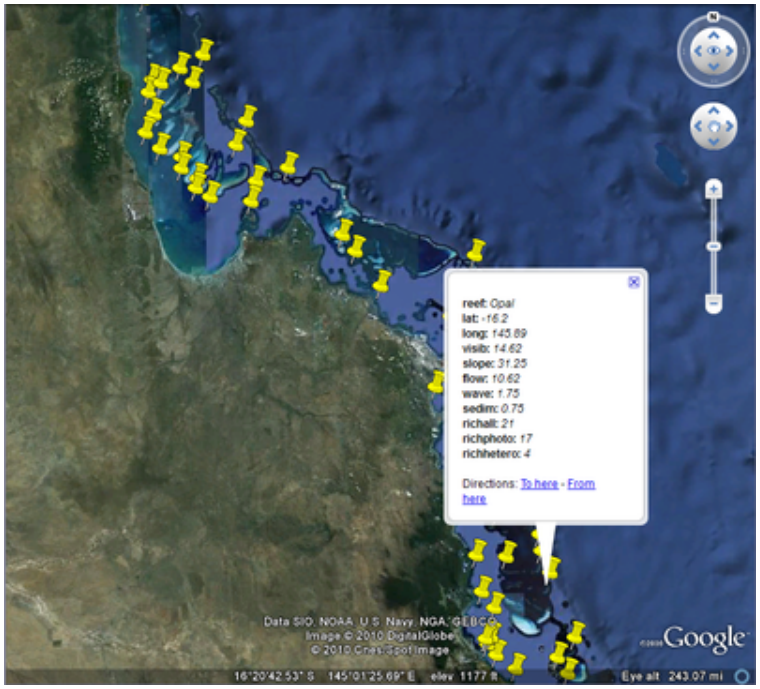

Figure 30: Screen capture of Google Earth imaging of data points showing soft coral site locations and pop-ups with the values of the variables at each point

4.4 Pie charts

Pie charts, despite their many shortcomings, are quite useful at comparing multivariate data in space. They can also be very easily implemented in the e-Atlas and Ningaloo-Atlas. We generate the pie charts as KMLs using R:

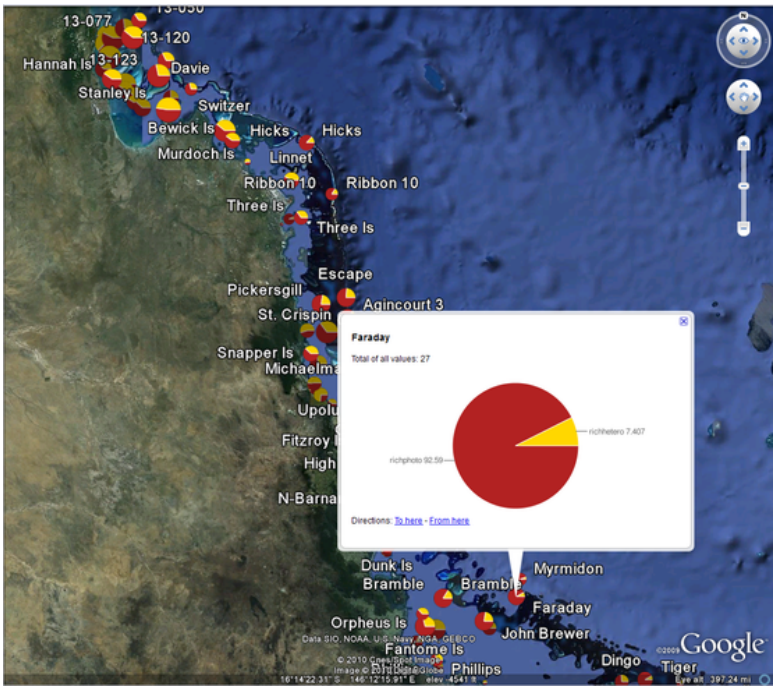

Figure 31: Screen capture of Google Earth imaging of pies showing soft coral phototrophic (red) and heterotrophic (yellow).

1. Read you data into R as a data.frame (read.csv is a simple option). The data should include:
   (a) a column that will provide labels for each location (e.g reef names)
   (b) the locations (longitude and latitude) of the sites
   (c) the data values to be displayed in each pie, one column for each variable
   (d) a text description for each site (optional)
2. Call the function geoPies to generate the KML:

```
geoPies(softcorals, labels=1, longlat=c(2,3), vars=c(5:6))
```

4.5 Hints and Tips

1. If you get the following error message then you may have wrongly labelled the longitude and latitude variables or you may have not use negative values for latitudes in the southern hemisphere:

   ```
   Error in ‘proj4string<-‘(‘*tmp*‘, value = <S4 object of class "CRS">) :
   Geographical CRS given to non-conformant data
   ```

2. You will find that unless you shut down the graphics windows that appear you will drown in an excess of them. An easy way to shut them ALL down is:

   ```
   graphics.off()
   ```

   You can do that auutomatically before you do refit a model and/or generate new graphics windows:

   ```
   graphics.off(); fit <- geoFit(richall~s(along,across),softcorals)
   ```

5 Data Management and Data Analysis Systems for the Atlas

A well-structured and coherent system for data management and data analysis is essential in order for the Atlas to function reliably and efficiently.

5.1 Example of a Codes File

```
data.set full.name abbrev.name type unit  model owner custodian priority
Chlorophyll Latitude    Latitude    numeric Deg_S       AIMS  Schaffelke   0
Chlorophyll Longitude   Longitude   numeric Deg_E       AIMS  Schaffelke   0
Chlorophyll Chlorophyll Chlorophyll numeric mg_l    QP  AIMS  Schaffelke   1
Chlorophyll Phaeophytin Phaeophytin numeric mg_l    QP  AIMS  Schaffelke   1
```

5.2 Example of an R Code File

```
1:  library(geo)
2:  load("Chlorophyll.RData")
3:  Codes <- read.csv("Chlorophyll.codes.csv",head=T,as.is=T)
4:  attr(Chlorophyll,"unit") <- Codes$unit
5:  xy <- match(c("Longitude","Latitude"),names(Chlorophyll))
6:  vph <- (1:nr)[Codes$VPH==1]
7:  fits <- list())
7:  for (i in vph) {
8:   cat("\nRun: ",i," ",Codes$full.name[i],"\n")
9:   fits[[i]] <- geoFit(Chlorophyll[,vph]~s(across,along),data=Chlorophyll,
10:   family="quasipoisson",long="Longitude",lat="Latitude",ascii=TRUE,k30=80,
11:   icol.vals= col.g4yob4,se.trim=2)
12:   geoFit2KML(fits[[i]])
13: }
```

1: The geo library is loaded. 2: The Chlorophyll Data is loaded from an .RData file. 3: Codes that control the analysis are read from the codes file. 4: The "units" of the responses are added to the loaded data file. 5: The "long" and "lat" names from the Chlorophyll data are matched. 6: The variables to be analysed are taken from the Codes$VPH. 7: fits is set up as a list to store the model fits. 8: The loop for analyses begins. 9: Print out the variable being analysed. 10: Fit the model. In this case we are using across-along, a smooth with 80df, – and the family is "quasipoisson". 14: End of code.

5.3 Outputs from the Toolbox

The function will first check to see if you have a folder called "work" in your working directory. If not it will generate the folder and all outputs will be stored there. They will all be labelled with the name of the data frame and the name of the response variable and will include up to:

1. The KMZ file "softcorals-richall-2.KMZ" that will include appropriately labeled graphics (pngs) and the KML text file. Thus for richall from the softcorals data set the files are:
   (a) softcorals-All.kml
   (b) softcorals-Value.png
   (c) softcorals-Value-Legend.png
   (d) softcorals-Std_Error.png
   (e) softcorals-Std_Error-Legend.png

(f) logo.png

(g) sidebar.png

2. If "ASCII==TRUE" for `geoFit2KML` then the file "softcorals-richall.zip" will contain the ascii grids of fitted values and standard errors

3. If "xplot==TRUE" for `geoFit2KML` then high quality graphics from `reefPlot` will a file "softcorals-richall-2.eps" if "eps==TRUE". and a file "softcorals-richall-2.png" if "png==TRUE"

5.4 The About Box

An "About" box of text is used for each data set. This backgrounds the data, with information including the owner, what the data' is all about and relevant references. This is used in the KML as a pop-up box. The text for the "About" box of the soft coral data is contained in a text file with HTML providing the formatting:

```
Data name: Soft_Corals_Reef</p><p> Description: Visual surveys of octocoral
communities (soft corals and sea fans) on 163 GBR reefs. Octocorals contain
taxa ....... without zooxanthellae.
</p><p>

The data set also contains visual estimates of the cover of the main benthos
groups such as hard corals, soft corals, macroalgae and crustose coralline
algae (unit: percent cover), ... sedimentation.
</p><p>

Data are based on swim surveys by one experienced observer (KF) along ~200 m
following the reef contour at each of up to 5 depth zones .... The data are based
on reef-averaged data.
</p><p>

Units: richness: number of zooxanthellate taxa per reef; number of
azooxanthellate taxa per reef; ... water clarity: m.
</p><p>

Owner Institution: Australian Institute of Marine Science</p><p> Custodian:
Katharina Fabricius (AIMS)
</p><p> Region: GBR
</p><p> Year: 1997 to 2008
</p><p> References:
</p><p> - Fabricius KE & De'ath G (2001a) Environmental factors
associated with the spatial distribution of crustose coralline algae on the
Great Barrier Reef. Coral Reefs,19: 303-309
...
</p><p> - Fabricius KE & De'ath G (2008) Photosynthetic symbionts and energy
supply determine octocoral biodiversity in coral reefs. Ecology 89: 3163:3173
</p><p>
```

6 R Package Documentation

Several R packages have been developed to analyse data and generate outputs for the e-atlas and ningaloo-atlas. These include makegrid, geo and acrossAlong

6.1 makegrid

Grids are fundamental to spatial analysis, We present modelled surfaces as grided data. These grids need to:

1. have an appropriate cell size: too large and detail will be loss and graphics resolution will be lost; to small and computational costs will be excessive and graphic will be too large and slow to display.
2. have appropriate spatial coverage: the grids should closely fit the data otherwise projections will be highly variable (low precision).

```
makegrid.mask <- function(data) {
##
## interactive function for defining a mask (boundary) for use in "makegrid"
## click to select the boundary points (in order!!)
##
Values returned: boundary}

makegrid.rect <- function(boundary,size=0.001,plt=TRUE,data) {

##
## function for generating a grid within a boundary
##
## "boundary": boundary coordinates in order "long" & "lat"
## "size": cell size of the grid; typically 0.01 for intial model selection,
##    and 0.001 - 0.0005 for gridded outpus (ASCII & KMLs)
## "plt" if TRUE plot the grid
## "data" a data frame or matrix with long, lat -- if not missing
##    add the data points to the plot
##
Values returned: grid
}

makegrid.select <- function(data, size=0.1, buffer=0.01, block = 1000, plt=TRUE) {

##
## interactive function for defining a mask (boundary) and then
## generating the grid within it.
## both grid and bundary are returned by the function
##
## the data points supplied in long-lat order in data are plotted
## then click to select the boundary points (in order!!)
##
##  arguments:
##
## "data": data frame or matrix of coordinates in order "long" & "lat"
## "size": cell size of the grid; typically 0.01 for intial model selection,
```

```
##    and 0.001 - 0.0005 for gridded outpus (ASCII & KMLs)
## "plt" if TRUE plot the grid and boundary
## "block" size of data blocks passed to makegrid.clip
##
##    add the data points to the plot
##
Values returned: list(boundary, grid)
}


makegrid.clip <- function(grid, bound, block = 1000) {
##
## function for clipping big grids
##
## "grid": the input grid as a matrix of long & lat
##   can be generated by "makegrid"
## "boundary": the input boundary as a matrix of long & lat
##   can come from "makemask"
## "block" : the block size to prevent memory overflows
##
Values returned: grid
}

makegrid.density <- function(data, long="long", lat="lat", eps=0.5,
use.line=1 ,size=0.1, block=1000, plt=TRUE) {
##
## function for generating grids with single or multiple areas based
## on spatial densities of points borders of grids are defined by a
## common (low) density boundary selected by the user
## "x,y": long and lat respectively
## "eps": controls the scale of smoothing --
## -- try using different values in the (range 0.1,1 ??)
## "use.lines": lines of the density smooth are labelled 1,2, ---
##  -- select which you want as your boundary
## "size": cell size of the grid; typically 0.01 for intial model selection,
##    and 0.001 - 0.0005 for gridded outpus (ASCII & KMLs)
##
Values returned: grid
}
```

6.2 geo

Grids are fundamental to spatial analysis, We present modelled surfaces as grided data. These grids need to:

1. have an appropriate cell size: too large and detail will be lost and graphics resolution will be lost; too small and computational costs will be excessive and the graphic will be too large and slow to display.
2. have appropriate spatial coverage: the grids should closely fit the data, otherwise projections will be highly variable (low precision).

```
geoFit <-
function (form, data, model=c("gam","loess")[1],weights, grid.predict = TRUE,
aafun, land, grid, xlim, ylim, xpnd = 0.02, lat = "lat", long = "long",
ptype = "response", plt = TRUE, plt.size = 6, pts = TRUE,
se = TRUE, se.mark = c(10:0), se.omit = 0,
aauto = FALSE, aauto.vals = c(0.25,0.5,1,2,4,8,12),
    grid.size = 0.1, aak.min = c("fit","best")[1], aak = 4, ...)
{
    require(mgcv)
    require(acrossAlong)
.....
.....
#  Arguments that you may/will want/need to change
#
# "form" is the model formula, e.g richall ~ s(across, along)
# "data" is the name of the data set
# "model" is the type of model, currently "gam" or "loess"
#
#
}

geoTrim <-
function (fit, se.omit=4, plt=TRUE, plt.size=6, pts=TRUE, se.mark=10:0,  ...)
{
#  Arguments that you may/will want/need to change
#

}
geoFit2KML <- function (data = Soft_Corals_Reef, ycol = 101, xy = c(5:4),
    fname = deparse(substitute(data)), method = c("ll","aa")[1],
    locale = c("gbr","wa")[1], weights, hotspot = c("hi","lo")[1], value = "Value",
    unit = TRUE, lims, k30 = 30, google.run = FALSE, extra = 0, fit = NULL,
    lab.nsig = 4, max.leg.col = 41, ascii = FALSE, leg.minw = 16,
    leg.col="orange", leg.font=2, leg.eric = FALSE, leg.col.eric = "black",
    gdriver = "cairo", link=FALSE, offset, theta, logo = TRUE,
    logo.file = "logo.png", sidebar = TRUE, sidebar.file="sidebar.png",
    sidebar.hsv=c(0,0,0.15,1), about = TRUE, merge = TRUE, rmKMLs = TRUE,
    zip = TRUE, se = TRUE, se.trim = 0, size = 4, size.png = 8, pixel = 19,
    font.scale = 1, icol.vals = col.g4yob4, icol.se = col.yorr4,
    icol.grey = col.greys, grey.se = TRUE, ncol = 9,
    xplot = FALSE, geo.png = TRUE, geo.eps = TRUE, wid=6, res=576, ...)
{
```

```
#  Arguments that you may/will want/need to change
#



}

geoPies <- function(data, labels=1, longlat=c(2,3), vars=c(4:ncol(data)), txt,
    title=deparse(substitute(data)), eps=0.5, size=10, piecol, pie.scale=60,
    minsize=0.1, google.run=TRUE, FUN) {

#  Arguments that you may/will want/need to change
#
# "data" is the name of the data set
# "labels" is the column number of the labels on each pie
# "longlat" are the column numbers of the long & lat respectively
# "vars" are the column numbers of the variables in the pies charts
# "size" controls the size of pies multiplicatively
# "piecol" uses alternate color schemes : eg rainbow or terrain.colors
# "minsize" is the minimum size of pies
#
```

6.3 acrossAlong

Package to calculate relative distance measures that account for natural boundaries such as coastlines and outer reefs. Two examples are included:

1. acrossAlong: use for the GBR
2. aaCoast: use for Ningaloo

```
acrossAlong <- function(data, land = landonly, reef = outonly,
        north = nrthonly, south = sthonly, out = outonly, check = TRUE) {
##
## Calculates the relative distance across & along the GBR
## "data" is a matrix containing the longs (column 1) and lats (column 2)
## "land", "reef", "north", "south" are the 4 boundaries
## "out" checks for point beyond the outer bounday and recalulates these distances
##

}

aaCoast <- function(data, land = ning.coast.smooth) {

##
## Calculates the relative distance across & along the Ningaloo Coast
## "data" is a matrix containing the longs (column 1) and lats (column 2)
## "land" is the (smoothed !!) land boundary : ning.coast.smooth can be used
##

  index <- findNeighbours(data, land)
  across <- findDistances(data, land, index)
  dists <- sqrt(rowSums(as.matrix(land[-nrow(land),] - land[-1,])^2)
  along <-   cumsum(c(0,dists))[index]
  cbind(across = across, along = along)
}
```

7 Installing R Packages for the Toolbox

The packages `geo`, `makegrid` and `acrossAlong` have been written for the Toolbox and are available from g.death@aims.gov.au.

They depend on several other R packages, namely `gstat`, `mgcv`, `maptools`, `rgdal`, `sp`, `KernSmooth`.

These packages need to be installed prior to `geo`, `makegrid` and `acrossAlong`.

7.1 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
```

Now install the CRAN packages `gstat`, `mgcv`, `maptools`, `rgdal`, `sp`, `KernSmooth` either 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
```

Then install the local packages using something like :

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

7.2 Windows

To install in Windows do:

1. Download and install the R executable (currently R-2.10.2-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.

7.3 Help

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