R: *Quo Vadis?*

A Talk Presented to *UseR! 2012*, Nashville, USA

Bill Venables, CSIRO, Australia

2012-06-15

Abstract

Language designers seem to regard **R** as an ugly, inefficient language and hence find its popularity mystifying. See, for example, (Cook, 2012; Morandat et al., 2012).

I present four examples from my own work, two large data analyses problems in fisheries, and two more abstract programming examples. Hopefully these will be of intrinsic interest, but together they encapsulate why I think useRs find **R** so invaluable. My thesis is that good data analysis and modelling require the practitioner to engage *interactively* with data, and that at some level *programming* becomes essential to this. This is essentially the same message as that presented in Chambers (1998, 2008), and the same idea implicitly underlies (Venables and Ripley, 2002). The popularity of **R** is primarily due to the way it provides support for this activity, making near optimal trade-offs. This view is mostly consistent with Cook (2012) but there are some important differences. (The claim that **R** is *necessarily* ugly is also disputed!)

Although **R** may be well suited to meet many contemporary data analysis problems, it will not remain so indefinitely. I do not attempt to answer the existential question posed in the title, but rather suggest it as one we should be thinking about, now. I will present some thoughts on a SWOT assessment for **R**, and suggest ways we might prepare for a graceful transition to whatever becomes the next phase. Such a new phase, or phases, will inevitably come as data analysis itself rapidly evolves in both scope and scale.

Contents

List of Figures

1 Introduction

Not many will remember the particularly lurid Hollywood blockbuster of the early 50's, set in the Rome of Nero, from whose pithy title ours is taken. The question, "Where are you going?" or "What next?" is a universal one, and in the case of **R**, not one easily addressed.

The "**R** phenomenon", as some are disposed to call it, seems to have taken everyone by surprise. It seems useful to think about just what makes it so attractive to the data analysis foot-soldiers and why the language theoreticians seem to look so askance at it from their theoretical vantage point, (Cook, 2012; Morandat et al., 2012). In my view, both sides have a good deal of merit to their case, and in pondering the next step in the evolutionary process to which **R** must inevitably submit, we need to take care that due attention is given to maintaining as many of the current practical advantages of **R**, while at the same time listening to the language specialists and heeding their insights as well.

John Chambers (2009) in an invited article on the future of **R**, listed 6 "facets" that he considered important in explaining the rapid progress of **R**. For reference, these facets were that **R** is:

1. an interface to computational procedures of many kinds;
2. interactive, hands-on in real time;
3. functional in its model of programming;
4. object-oriented, "everything is an object";
5. modular, built from standardized pieces; and,
6. collaborative, a world-wide, open-source effort.

To this list I would add that specifically the **R** system is:

7. extensible, may be augmented by compiled code in other languages;
8. cross-platform; and
9. international.[1]

The first of these extra ones, facet 7, may simply be an amplification of facet 1, though it seems useful to emphasize that if **R** is what the language aficionados tend to call "glue code", then the bits of paper and string it can glue together are not essentially fixed within the system, unlike many others.

I think it's worth clarifying what we mean by "interactive", too, as in my view it remains a key feature for **R**'s continuing popularity. By this we mean not only that the system allows the user to try a *wide range* of exploratory steps and see the outcome in real time, with relatively few keystrokes; but most importantly that the user can do so *leaving a clear trail*, so that the steps can be retraced and the results reproduced for later reporting. Currently this effectively has to be a *code* trail. Clearly reproducibility is also important if there are

[1]The vital importance of this relatively recent development became very clear to me when working with colleagues from non-English speaking locales, mainly from Brazil and Japan. It is very much appreciated.

future adjustments to the analysis or to the data, (and there nearly always are, both) or if the exercise has generated some technology worthy of transfer to other projects.

1.1 Programming and data analysis

Brian Ripley, in a talk in the RMetrics meeting in Switzerland about this time last year singled me out for special mention:

> "Bill Venables used to stress in the 1990s that using **R** (and **S**) was programming, and that programming was an under-valued skill by statisticians. (Even then others told him he was showing his age —BSc 1965.)"

As he went on to say

> "To use **R** effectively you need some understanding of 'software engineering' concepts, as well as of statistical algorithms."

I think he meant it as a kind of compliment, but with Brian it can be hard to tell at times. In any case, Brian and I are not entirely alone in this. I have it on the authority of the **fortune** package that our host is of a similar view:

> "Can one be a good data analyst without being a half-good programmer? The short answer to that is, 'No'. The long answer to that is, '*No!*'."
> – Frank Harrell, 1999 **S-PLUS** User Conference, New Orleans (October 1999)

I was at that meeting, too, and I heard him.

My conviction for many years has been that good data analysis, within which I would include Statistics, requires a combination of several mental skills and attitudes. One of these skills, in modern times, has come to manifest itself as *programming*, or software engineering, at least of an elementary kind. The clarity and organizational skills that programming *forces* on the practitioner are precisely those needed to untangle the sometimes very subtle and sophisticated messages that data often has to offer.

1.1.1 Two cultures?

To me, data analysis *is* programming, but of a kind. This is allied to the premise discussed at length in Chambers (1998). But there are two kinds of programming, or perhaps it should be viewed as two extremes of a spectrum of kinds. Data analysis usually has an exploratory phase, where rapid and flexible interaction are important. At this stage the code that will be written will largely be using existing tools or writing usually small and often *ad hoc* new tools, while the computational details are still being sorted out.

At the other end of the spectrum, the data analyst has developed a new kind of tool and needs to transfer the technology. At this stage the programming has to be carefully designed and written in a way that ensures, again in John Chambers' words, *The Prime Directive*: that the computations can be *understood* and *trusted*.

The ongoing discussion of the relative merits of S3 and S4 classes seem to me to miss this point. S3 classes are informal, possibly inelegant, sometimes insecure, but emphatically simple and easy to grasp, and they have the capacity to grow and develop in an organic way. On the other hand S4 classes are formal, trustworthy, in principle more flexible, but they usually result in code that is much more contorted and obscure, and they do certainly require a lot of careful planning and design work to be done beforehand. Once a bad design decision is made with S4 classes, it's often back to square one.

In my view, S3 classes are often more convenient for the kind of programming involved at the exploratory phases of a project, whereas S4 classes are entirely suitable for the other end of the project, or the other end of the spectrum, in my sense above.

1.2 Aims for the talk

What I would like to do first in this talk is to illustrate some of these facets as they occur in a series of real data analysis examples from my own work. My purpose is two-fold: I hope the examples are of some interest in themselves, but more pertinently, I hope to show by example why data analysts find **R** so convenient and productive for their work.

Finally I will make some admittedly opinionated comments on the strengths and weaknesses in **R** as it stands and on the opportunities and threats that may lie ahead.

2 Prawn tales

Australia's Northern Prawn (= 'shrimp' in some cultures) Fishery, (NPF), is a large inshore fishery which stretches essentially across the top third of the Australian coastline. See Figure 1. Effort in the fishery is conventionally measured in boat-days, and the effective

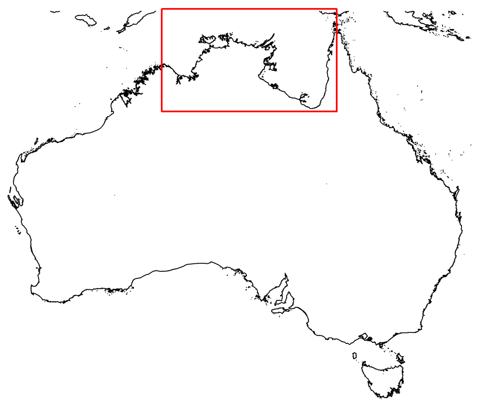

Figure 1: Australia's Northern Prawn Fishery in context

extent of the fishery is more clearly shown by Figure 2 on the next page. (The blue line that snakes its way through the fishery on Figure 2 will be explained in due course.)

There are eight species of prawn caught in the fishery, but the most valuable are the two species of Tiger Prawn: Grooved Tigers (*Penaeus semisulcatus*) and Brown Tigers (*P. esculentus*).

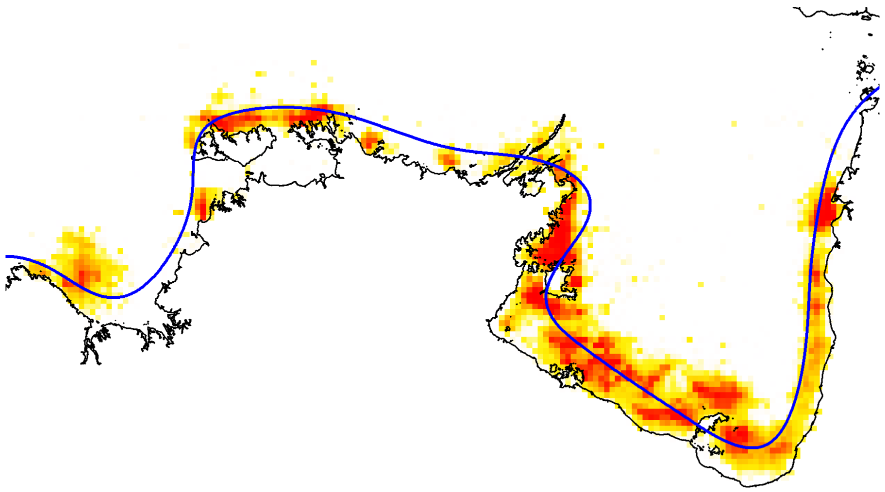

Figure 2: A spatial representation of NPF effort since 1970, in boat days (on a log scale). The deeper the orange colour, the higher the recorded effort.

These are very similar in appearance, (see Figure 3), and although they are distinct biological species they are *not separated* in the logbook catch records. The total daily catch is merely reported as a *weight* of Tiger Prawns, in kilograms. However to ensure that both species remain viable, each has to be given a separate stock assessment. The traditional way to do this has been (and unfortunately, remains), first to "split" the Tiger catch into its two component species, by weight, and treat the components *as if* they were independently recorded. This is known as the "species split" problem.

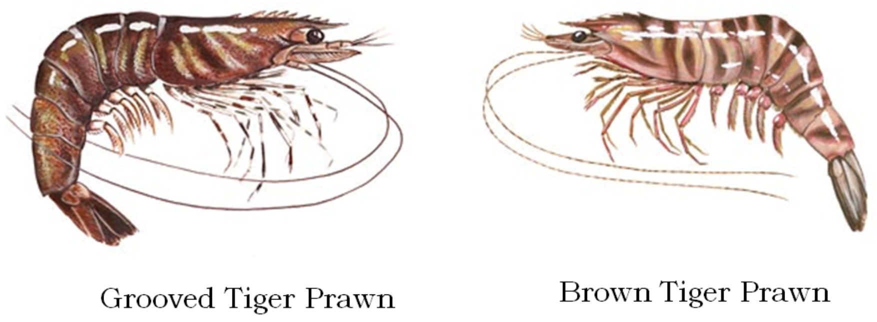

Figure 3: Two species of Tiger Prawn caught in the NPF

Information on which to base base species split process comes from a series of large-scale trawl-based scientific surveys in the NPF, conducted for a variety of primary purposes, but for which the catch has been classified to the species level and measured. These surveys began in 1976 and, with a few conspicuous gaps, continue until the present time.

When I first joined CSIRO in 1999, species split for Tiger prawns was based on a very informal and static process. The entire NPF is broken up into 6" × 6" grid squares, which is the finest spatial resolution for which logbook records report their daily position. It is known that the two species favour a different sediment type: Brown Tigers generally favour a sandy sediment and Grooved Tigers a more muddy one. For each grid square, then:

- if any survey information is available, the crude aggregate ratio for Browns to Tigers in the survey data was used to split the logbook catch;

- if no survey information is available, an *ad hoc* matching was done with another grid for which some information was available, using either similarity of sediment composition (which was rarely available) or simple spatial proximity.

It is also known, however, that both species have an offshore annual migration pattern, presumably a breeding cycle though this is not entirely clear. The species split method in use, however, was temporally static and relied only on geographical position. Nevertheless, it did seem to capture the major relative pattern of the relative distribution of the two species, and for stock assessment purposes appeared to be adequate.

The first project I was given when I first joined CSIRO was to build a better process for species split.

There were ultimately two species split projects, 2000 and 2004, (though the first was really a sub-project of a much larger risk analysis project). Both have final reports available in the grey literature, but the first project was also discussed at some length in Venables and Dichmont (2004b).

2.1 Species split 2000

It seemed obvious to me that some kind of predictive model would be the most useful way of integrating and focusing all of the (scant) available information to predict the proportion, by weight, of a catch of Tiger Prawns.

The only available information that might be useful consisted mainly of

- The location of the shot, i.e. `Longitude` and `Latitude`,
- The `Depth` at which the trawl was made, and
- The time of year, as a temporal term to accommodate migration events.

The sediment information, at the time, was too sketchy and probably measured at too local a spatial scale to be of use in a general model that covered the whole fishery. Spatial surrogates would have to suffice.

These variables would be used to build a *temporally stable* model for the species split proportions. For a non-stable alternative (which would be useful for investigative purposes, but not for prediction), we would also use

- The elapsed time, in days, since 1970-01-01.

Initially, however, we considered only the stable model.

At the time I was still using **S-PLUS**, a software platform with which I had had a long and happy association. **S-PLUS**, at the time, did have some contributed software, (e.g. **MASS** and friends, the **survival** and **rpart** libraries and a number of others), but most of the modelling software either came with the system itself or some of the more exotic items were available in some rather expensive, though very high quality, add-on "modules". An example of this was the **spatial** module, as opposed to the more limited (but still high quality)

spatial library that came as part of the **MASS** collection. There was, however, nothing like the current **CRAN** repository with its explosion of free goodies for every conceivable, and many inconceivable, purposes.[2]

For generalized additive models, which would clearly be a natural place to start, the only software I had available was the Hastie and Tibshirani code, which at the time was rather limited in what it could do with interaction-type models involving smooth terms. As spatial location was clearly going to be a critical surrogate for things like sediment type, some thought needed to be given as to how to include this information in such a model.

2.2 The thin blue line: flexible spatial location

Rather than use `Latitude` and `Longitude` directly in the model (Mark 1, in 2000, at least) we decided to represent spatial location in terms of two other variables, which formed a kind of imprecise, but effective *curvilinear coordinatisation* of the fishery.

The first step was to draw a smooth curve that hugged the coastline and wound its way as close as possible to the most active regions of the fishery. As this is an *inshore* fishery, the shots at which both logbook and survey records are made tend to be strongly identified with a region of the coastline.

This is the “thin blue line” shown on Figure 2 on page 7.

One spatial coordinate is then obtained as follows:

- first, the point on the line closest to the grid is found, and
- the coordinate value was the *distance along the curve* to the closest point from an origin in the West.

This was called `Rdist` in the first version of the model, but which I now prefer to call simply `Coast`.

For a second spatial coordinate we chose the distance from the grid to the nearest point along the coastline itself. This “distance from land” we called `Rland`, but I now prefer to call it simply `Sea`, i.e. the “distance out to sea”.

The rationale for choosing these particular two spatial coordinates at the time was to work with two spatial predictors we could be reasonably confident, from what we knew of the biology and behaviour of the species, that interactions between them, or indeed with most (but not all) the other predictors, are unlikely to be important. This facilitated a simplification in the model which improved interpretability as well, as fewer interaction terms were needed. At the time, smooth additive terms with interactions were not well supported by stand GAM technology, though this was ignored by many proponents. I discussed this point some time ago in Venables (1998).

[2] I am reminded of Haydn’s rather modest assessment of his own symphonies, which he made towards the end of his life. He is reputed to have said, “*Sunt bona, sunt quaedam mediocria, sunt mala plura*.” That is, “There are some good ones, some are mediocre, but most are terrible.”

In retrospect I think this idea, which is not entirely new, is possibly one of the most useful methodological suggestions to come from this work.

2.3 A temporally stable model

The outline for the stable model was not clear. After some investigation with GAMs, we eventually settled on a simpler GLM which took the following form. The response was $Y = S/T$ where

- S = Weight of *P. semisulcatus* (Grooved Tiger) and
- T = Total weight of Tiger prawns in the catch.

with all weights in grams.

If $\mathrm{E}\,Y = \mu$ then we used, in **R** terms, a `quasibinomial` GLM with

$$\operatorname{logit}\mu = \beta_0 + S_{10}(\texttt{Rdist}) + S_5(\texttt{Rland}) + S_6(\texttt{Depth}) + H_4(\texttt{PDay}) + LH_2(\texttt{Rland},\texttt{PDay})$$

where the $S_k()$-terms are natural splines with k degrees of freedom, $H_k()$ is an harmonic (Fourier polynomial) with k degrees of freedom and $LH_2(,)$ is a linear × harmonic interaction with 2 d.f.

It is worth making the note that some of these models were difficult to fit and convergence was often very slow during testing. More recently a package has appeared on **CRAN** that modifies the `glm` algorithm and can result in much faster convergence. The package is called **glm2**, (Marschner, 2011a,b) and provides a function of the same name which is a direct replacement for the standard `glm` function in the **stats** package.

At the time, the available data consisted of about 9000 trawl catches, drawn from a number of component surveys ranging in time from 1976 to 1998. The model was, with the equipment and technology available to me, somewhat difficult to fit, but the model appeared to work well and was clearly a great improvement over the existing temporally static process.

The components (in the link scale) are shown Figure 4 on the following page.

To appreciate the two-way interaction between distance from land and day of the year (i.e. the migration term) we opted for a prediction at a hypothetical location with fixed `Rdist` and `Depth`. The contour map of the predictions, which are then a function of `PDay` and `Rland` is shown in Figure 5 on page 12.

The general features of the model were in accordance with the general understanding of most of the stakeholders. These included the fact that some regions like North Mornington and Karumba in the South-East corner of the Gulf of Carpentaria were “pure” Brown Tiger regions, but others, such as the Vanderlins, a relatively short way to the West, was “mixed”, with the components varying throughout the year.

The model was also effective (enough) when tested by the usual internal methods, such as using a training and test subsets of the data. Some details are given in Venables and Dichmont (2004a).

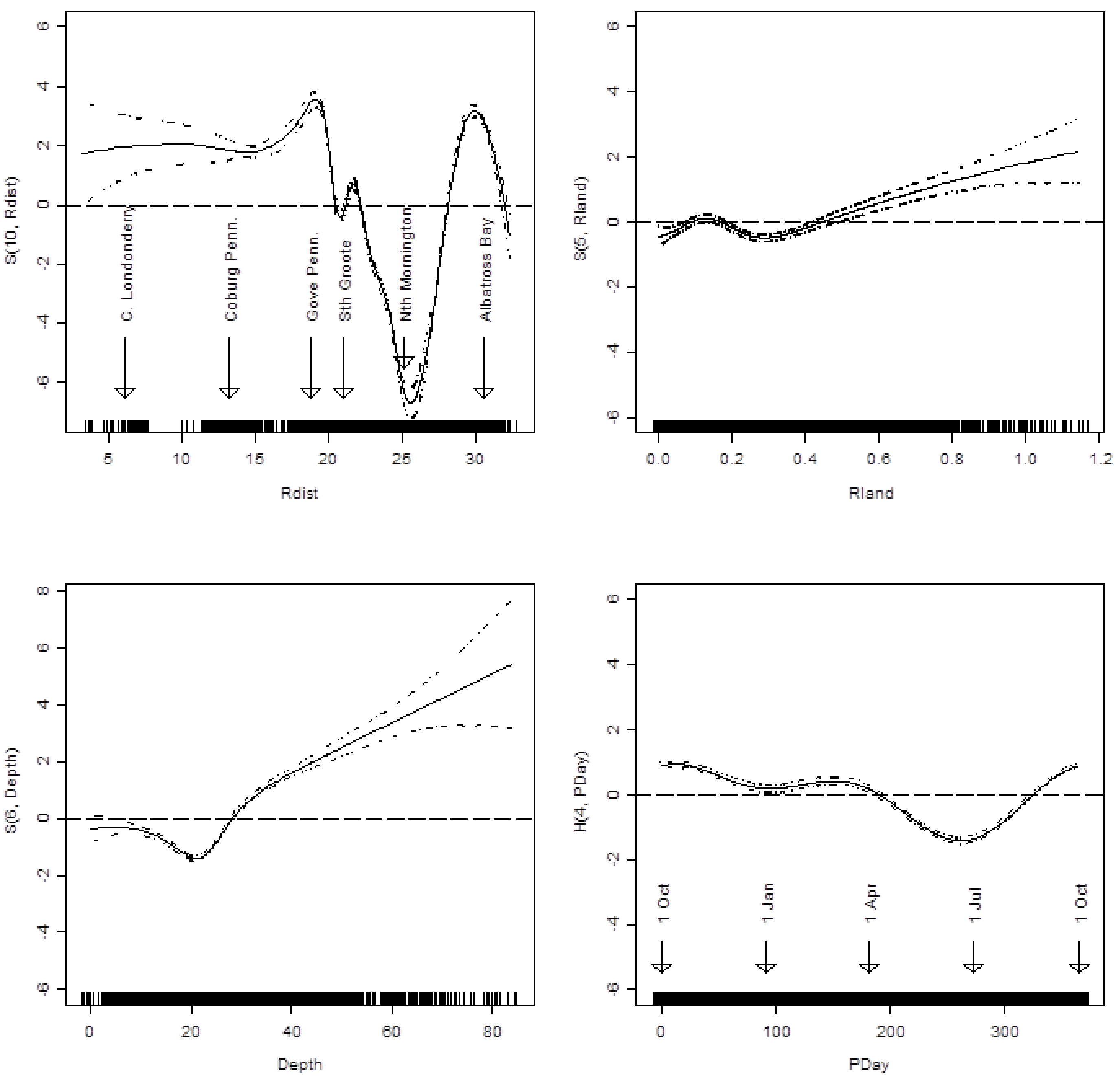

Figure 4: Four components from the 2000 temporally stable model

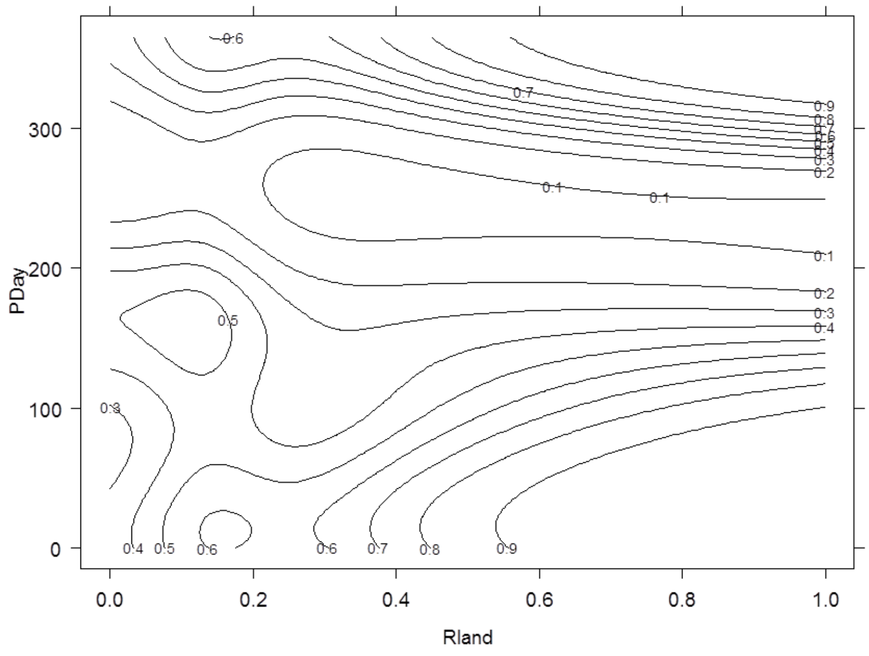

Figure 5: Conditional predictions given a fixed distance along the coast and depth

2.4 A long-term trend?

To investigate if the model really were temporally stable, we included an extra term $S_5(\texttt{Day})$, i.e. a natural spline with 5 d.f. in the elapsed time, in days, since 1970-01-01.

The result, shown in Figure 6 on the following page, is remarkable for its ambiguity. There does indeed appear to be a long period where any trend was a negligible perturbation but towards the end there appears to be a large deviation from this, (with a large pointwise standard error). However this also corresponds to the part of the time scale when there is least survey information, i.e. it could be seen as an unsafe extrapolation.

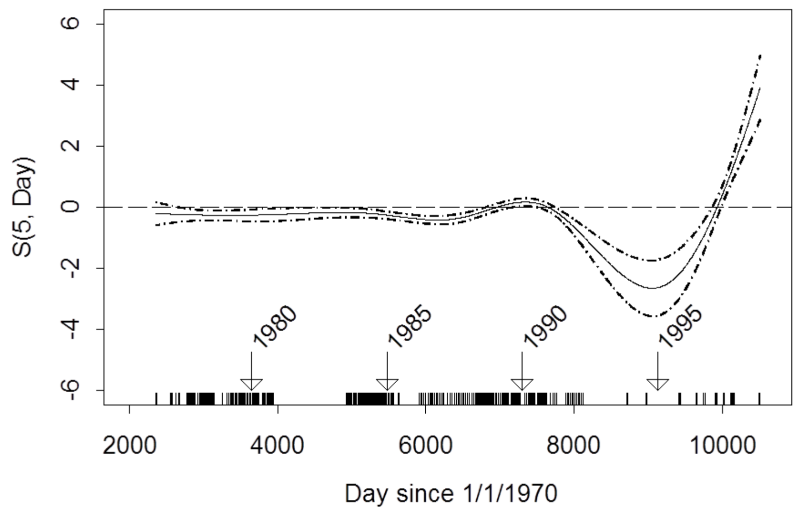

Figure 6: Long-term trend component, additional to the temporally stable model of 2000.

Further investigation led us to the view that there was indeed something odd going on, but it was probably not as alarming as the component picture might suggest.

2.5 Species split, 2004

The suspicion that the species composition had slipped in favour of one species, but the species split process (of neither version) was not faithfully reflecting the changed status was the main reason ultimately to undertake a second project that looked into this question much more carefully, generating more data as well.

In the early 2000s I became fully emancipated from **S-PLUS**, even to the point where my licence lapsed. It was for me a sad parting of the ways, as I did have many friends in the Insightful organization, but frankly, with all the new software becoming available in **R**, and the overall better language model than **S**, the transition was inevitable.[3]

The new project was a 3-year programme with two main features, namely it would have

- a field-work component to augment the historical data set with a lot of in-season sampling on commercial vessels, and
- a data analysis component that would extend the modelling to take advantage both of new predictor data that had become available and of some of the contemporary developments in the modelling technology itself.

[3]What particularly drew me, from a language point of view, were two features of **R** in particular: the scoping rules and the use of environments and closures. In a sense, coming from **Scheme**, **R** seems to have a stronger pedigree than **S**, in a computer language sense, which has to confer some advantages.

Under the second point I had in particular the work of Simon Wood in mind, as made available in the **mgcv** package, whose development had suddenly blossomed, and indeed continues to do so (Wood, 2003, 2004, 2011), along with other things of course. Also, Geosciences Australia had recently made available a sediment map of the area, which provided an estimate of the percent mud in the sediment of our grid squares.

The final report of the project appeared as Venables et al. (2006), and gives full details.

The new model was a GAM rather than a GLM. A crucial development in technology that had been implemented in **mgcv** was to allow smooth terms in the model that were functions of more than one predictor, along with a variety of spline bases including cyclic ones that accommodated periodic terms.

The `Rdist` predictor, that had worked so well in the early version was abandoned in favour of what seems to us a less artificial approach. The `Rland` predictor (now called `Sea`) was retained, however.

In the new model, the response was the same as in the previous one. It was a quasibinomial GAM, with terms as follows:

- An *isotropic* thin-plate smooth spline term in `Longitude` and `Latitude`, intended to capture spatial aspects not otherwise captured by functions of location,
- A bivariate smooth tensor spline in `Day` (of year) and (distance out to) `Sea`, with the spline basis for `Day` cyclic, with period one year,
- A similar bivariate smooth tensor spline in `Day` and `Depth`,
- A bivariate smooth tensor spline in `Sea` and `Depth` to capture more subtle spatial features,
- A smooth spline term in (percent) `Mud` in the sediment.

The non-stable variant of the model contained in addition:

- A smooth spline term in the `ElapsedDays` since 1970-01-01.

The fitted components are shown in Figures 7 on the next page and 8 on page 16.

The purely spatial component is shown in contour form, and really only captures the feature that the South-East corner of the Gulf of Carpentaria is effectively pure Brown Tiger, other areas in the Gulf are, to some extent mixed, and the region outside the Gulf is essentially pure Grooved Tiger.

The other three components show the two periodic terms (at least along one margin) and the `Sea` by `Depth` interaction.[4]

Figure 8 on page 16 shows the sediment component and, as expected, that on the whole, Brown Tigers favour a sandy sediment (i.e. low mud) and Grooved Tigers a more muddy one.

[4]Note that while in this region, distance out to sea and depth are not as highly correlated as in other parts of the coastline.

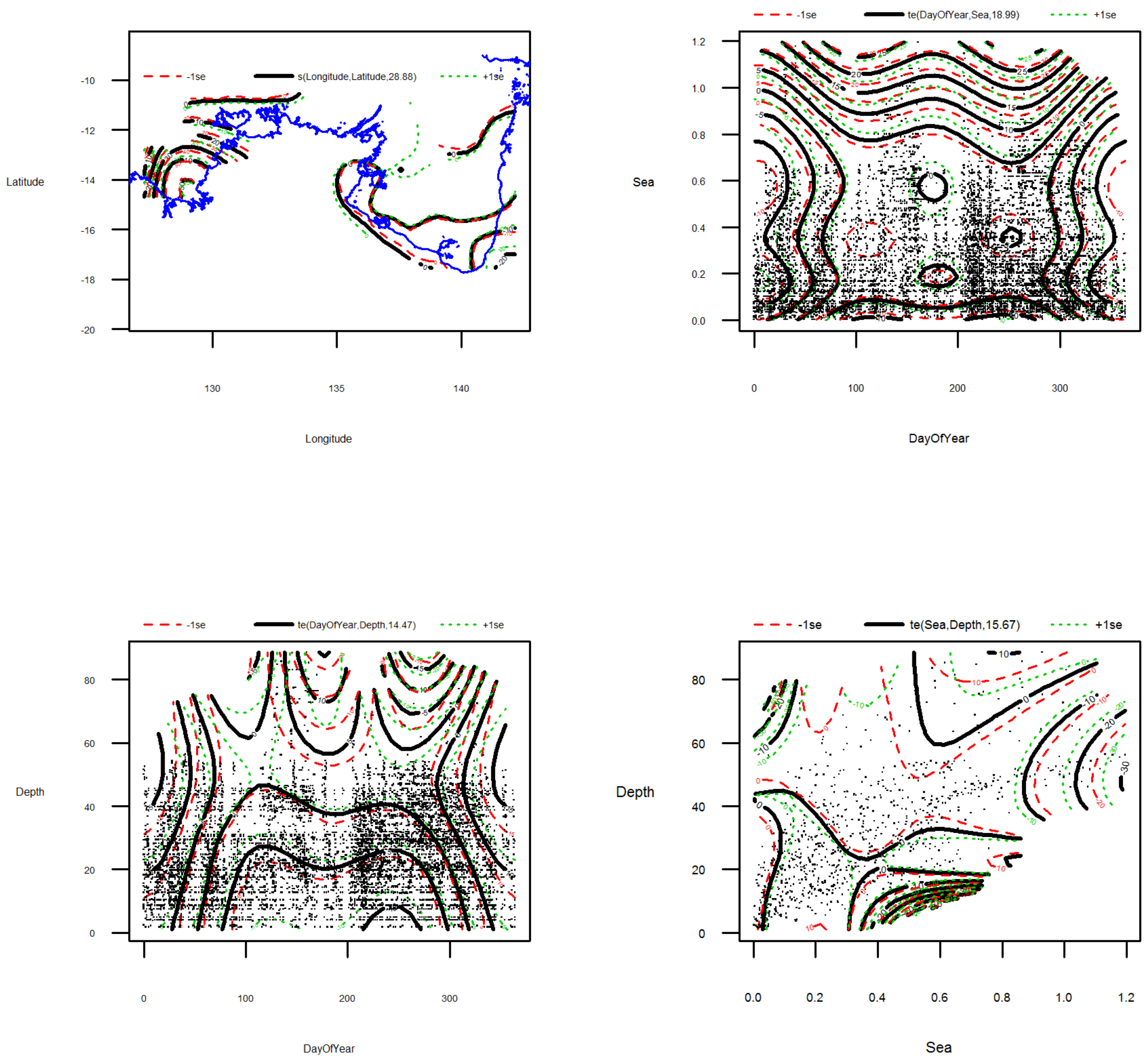

Figure 7: Four smooth bivariate components from the stable model, 2004.

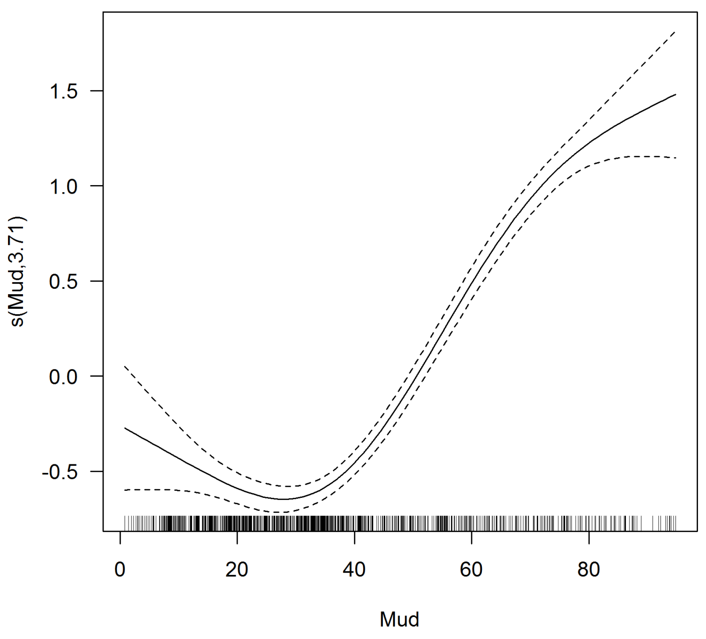

Figure 8: The sediment component of the stable model, 2004.

2.6 The annual migration effect

To appreciate the annual migration effect, we can predict the change in proportion over the year in four key places, shown in Figure 9.

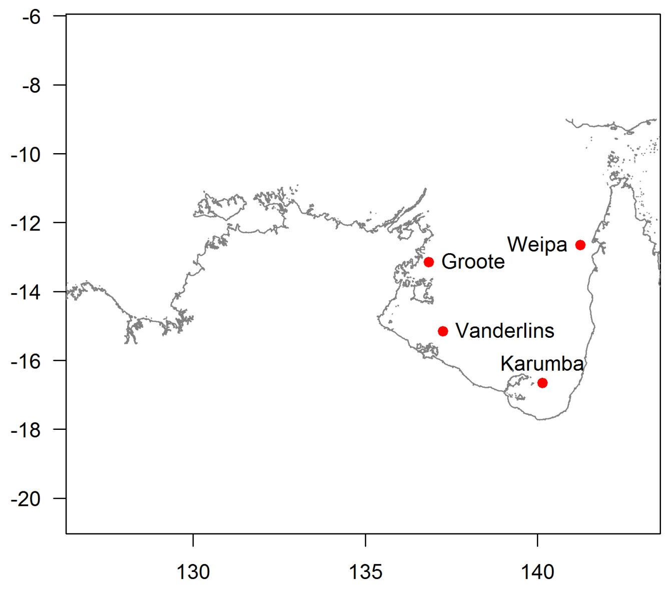

Figure 9: Four key places in the NPF

Figure 10 on the next page shows how the *P. semisulcatus* proportion varies throughout the year. The Karumba region is effectively pure Brown Tiger throughout the year, and the two regions in the North of the Gulf, Groote and Weipa, are mostly pure Grooved Tiger, with a slight dip in the middle of the year, corresponding to the time that the Grooved Tiger prawns are offshore on their migration. In the Vanderlins region, however, which is currently one of the most productive, there is a wide variation in proportion over the year.

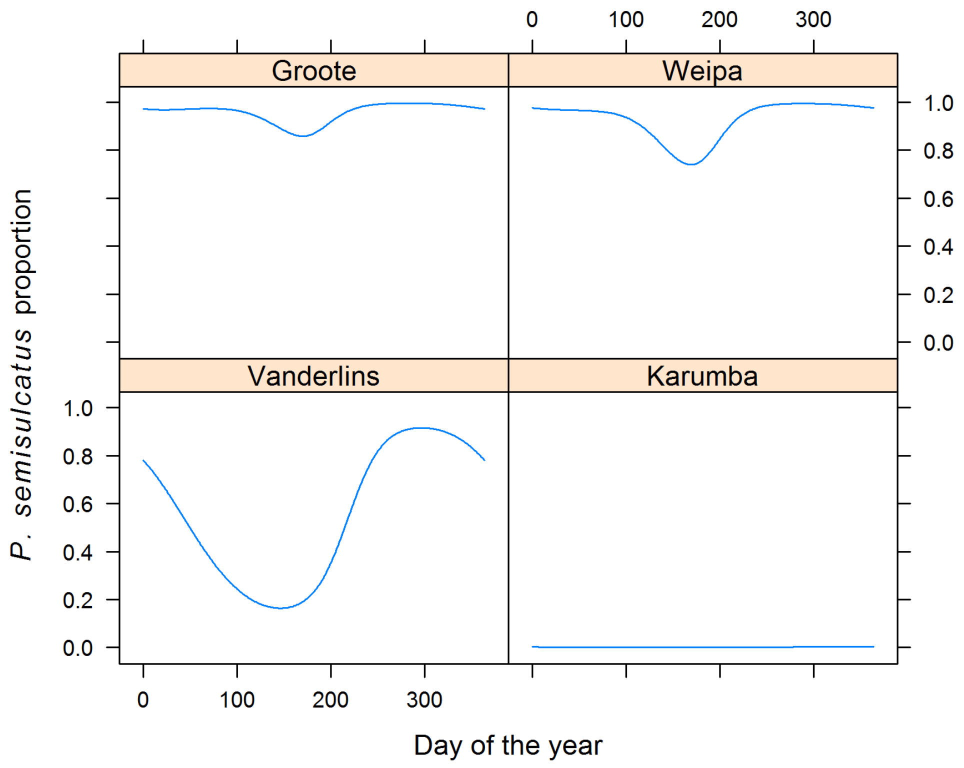

Figure 10: Variation in *P. semisulcatus* proportions over the year

2.7 The long-term trend issue, 2004

Finally an estimated long-term trend component, fitted on top of the smooth model, is shown in Figure 11. The additional data collected (as shown in the rug) has greatly stabilized the picture, and reduced the dramatic size of the final kick.

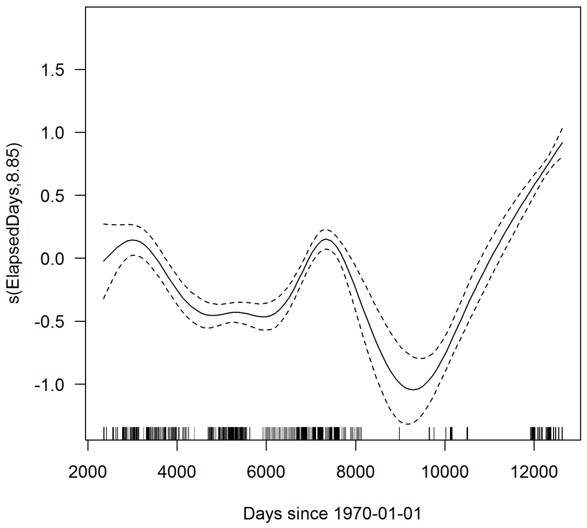

Figure 11: A potential long-term trend perturbing the stable model, 2004

Further investigation on several fronts added more weight to the contention that there had been a small shift in the relative composition, and this in turn led to corrective management action to protect the Brown Tiger species.

2.8 Where does programming come in?

Using **R** in what I would consider an innovative, constructive and investigative way like this, in my view, constitutes a kind of programming in itself, in that it requires careful planning and attention to detail. It also involves that crucial aspects of **R**, namely using objects to marshal, record and otherwise manipulate your information relevant to the exercise.

There were, however, a few smaller challenges for the project where programming of a more conventional kind did come to the fore.

2.8.1 Drawing the thin blue line

I tend to favour using complex number objects for two-dimensional problems in **R**. On looking at how others tackle these rather simple issues, it seems to me this is a most neglected feature.

For example, finding all possible Euclidean distances between points in the plane is a simple, two (at most) line function:

```
> dist2d <- function(x, y=NULL) {
    z <- with(xy.coords(x, y), complex(real = x, imaginary = y))
    as.dist(outer(z, z, function(x, y) Mod(x-y)))
  }
```

The first approach to drawing the thin blue line was very informal and proceeded as follows:

- Using `locator()` select about 20 points, in order from West to East, to guide the eventual curve through the main parts of the fishery.
- Join the points sequentially by straight line segments and find the cumulative distance, s, of each point, along the line segments, from West to East (starting from 0 at the most Westerly point).
- Fit *interpolation splines* independently to the Longitude and Latitude of the points as functions of the chord length, s and evaluate at a fine grid of points, say $n = 2000$. This provides the curve itself.
- The cumulative distances along the newly generated smooth curve provide the `Rdist` (or `Coast`) value for grids having that point on the curve as their closest.

The idea is much easier to express in code. Assuming a graphic such as in Figure 2 on page 7, the curved coordinate information can be returned in a data frame as shown in Figure 12.

```
> curvCoord <- with(locator(type = "p"), {
    z0 <- complex(real = x, imaginary = y)
    d0 <- c(0, cumsum(Mod(diff(z0))))
    z <- complex(real      = spline(d0, Re(z0), n=2000)$y,
                 imaginary = spline(d0, Im(z0), n=2000)$y)
    data.frame(Longitude = Re(z), Latitude = Im(z),
               Coast = c(0, cumsum(Mod(diff(z)))))
  })
> lines(Latitude ~ Longitude, curvCoord, col="blue")
```

Figure 12: Constructing the thin blue line “by hand”

The result is often close to what Hastie and Stuetzle (1989) call a “principal curve” for the grid locations used by the fishery. This work was first implemented in **S**, but there have been at least two **R** package ports of it. Those I know of are **princurve** (Hastie, 2011), and the older port, **pcurve**, (Walsh, 2011).

2.8.2 Finding the nearest points

Finding the nearest point on the line, and hence the `Coast` distance, can be done using a simple brute force, looping method based on the function `dist2d`, defined on on the current page. Since there are only about 7000 grids in the region, this is a feasible, if rather slow and cumbersome process. It is programming, though.

When I first did this I was using **S-PLUS** and the **spatial** module. This has a function for finding $k-$nearest neighbours based on quad trees. It works very fast.

When I finally migrated completely to **R**, I missed the spatial module, but the **spatial** package in the **MASS** collection has a function `knn1` which does a similar job.

More recently, however, the **SearchTrees** package, (Becker, 2012), has appeared on **CRAN** which implements quad tree search technology, which again is very fast. This is another instance of the collaborative facet of **R**.

Finding the distance to the coastline, the `Sea` predictor, is essentially the same problem, though the distance in this case is from the grid to the target point on the coastline. In the previous case it was the distance along the curve to the target point on the curve.

3 The morphometric problem: robust non-linear regression

In the survey data, the weight of each animal was often *not* recorded. Rather the *carapace length*, L was recorded and the weight, W, of the animal had to be inferred using a standard morphometric relationship, which invariably has the form $W = \alpha L^{\beta}$. More precisely, a statistical model for W in terms of L has the form

$$W \sim \mathrm{N}\left(\mu = \alpha L^{\beta}, \sigma^2\mu^2\right)$$

at least approximately. For individual measurements taking logs and using a log-linear model:

$$\log W = \alpha^{\star} + \beta \log L + \varepsilon, \qquad \alpha^{\star} = \log\alpha, \quad \varepsilon \sim \mathrm{N}(0, \sigma^2)$$

is often adequate. The variance on the log scale is usually quite small, making a simple back transformation adequate for estimating the true weight.

These relationships are particular to the species and sex.

Where necessary, the estimates we used for the morphometric constants, α and β, were taken from the literature. As part of the second project, we decided to re-calibrate them using a part of the new data that was individually measured both for W and L.

In the log scale the regression lines were *in the main* quite tight, and the estimates close to those in the published literature. There were, however, a substantial number of outliers most likely due to the difficulties of accurate measurement of some individuals, both in weight and in carapace length. Traditionally in most biological work such embarrassing data are silently discarded, but in this case the tight regression lines become somewhat self-fulfilling and new dangers appear. We preferred to use robust regression techniques, which worked well and settled the outlier issue in a reasonably objective way.

The **MASS** package has two main functions for robust regressions, `rlm` and `lqs`, both of which work well here, but nowadays I would normally use the **robust** library, (Wang et al., 2012), ported from **S-PLUS**.

A simple robust technique is to use the $t-$distribution with low degrees of freedom as a modelling distribution in place of the normal. This is simple to do using the optimization tools available in **R**, and can be packaged in such a way as to make using it almost as convenient as using the standard linear modelling tools. We did this as an exercise, though the package has not been published. Having the **robust** package available makes it somewhat redundant, except for expository purposes.

3.1 Estimation with aggregated data: the real challenge

It turns out there is quite a lot of data available for this calibration exercise in the survey data, but it comes in a very awkward form. A common practice was, for each trawl shot, to measure the carapace of individual animals, but to weight only the whole species group, sometimes separating the sexes, but often not. So if W_i is the total weight of a group of g_i animals with carapace lengths l_{ij}, and sexes s_{ij}, $j = 1,\ldots,g_i$, $i = 1,\ldots,n$, an approximate model would be

$$W_i \sim \mathrm{N}\left(\sum_{j=1}^{g_i} \mu_{ij}, \sum_{j=1}^{g_i} \sigma^2_{s_{ij}} \mu^2_{ij}\right), \qquad \text{where} \quad \mu_{ij} = \exp\left(\alpha_{s_{ij}} + \beta_{s_{ij}} \log l_{ij}\right)$$

where for any species there are 6 unknown parameters, namely α_M, α_F, β_M, β_F, σ^2_M and σ^2_F. Fortunately, good initial estimates are available.

Again, robustification is needed, although now picking the outliers is a non-trivial task. And again, the $t-$distribution dodge comes to the rescue. The likelihood, $\mathscr{L}$, to maximize is specified as

$$-2\log\mathscr{L} = \sum_{i=1}^{n} \left\{ (\nu+2)\log\left(1 + \frac{\left(w_i - \sum_{j=1}^{g_i} \mu_{ij}\right)^2}{\nu \sum_{j=1}^{g_i} \sigma^2_{s_{ij}} \mu^2_{ij}}\right) + \log\left(\sum_{j=1}^{g_i} \sigma^2_{s_{ij}} \mu^2_{ij}\right) \right\}$$

where the degrees of freedom, ν, is normally set at 3 or 5.

To minimize this quantity requires writing a function in **R** to evaluate it that is as efficient as possible, as many evaluations are likely to be necessary. This in turn requires the data to be well organized and vectorized techniques to be used as effectively as possible.

The point to make is that these tools are available in **R**, both the language and the system, and writing the necessary software to handle it does not require heroic efforts to push the system past its design limits. If this were to become a regular necessity, then some tools written in a compiled language would clearly be an advantage, but for trial purposes and even some limited production purposes, native **R** and a bit of programming cunning is all that is needed.

The $t-$robust estimates of the parameters from the augmented data were very much in line with the estimates got from the single animal measurements alone.

4 Complex makes things simple: Dan's problem

Dan is a young colleague sitting at the desk behind me. He works in time series (amongst other things) and he came to me with a computational problem that I will specify here only in abstract terms. It is, however, a real problem and together it looks like we have nailed it.

The two-dimensional version of the problem is the most important. In this case line segments in the plane are given by their intercepts on the $x-$ and $y-$axes. It is guaranteed that these are never zero, so the lines are guaranteed never to pass through the origin. It is also guaranteed that eventually the line segments will surround the origin.

The problem is to devise a fast method for finding the smallest polygon which the lines define that encloses the origin. Alternatively, each line defines a half-plane containing the origin. The problem is to find the intersection of all of these.

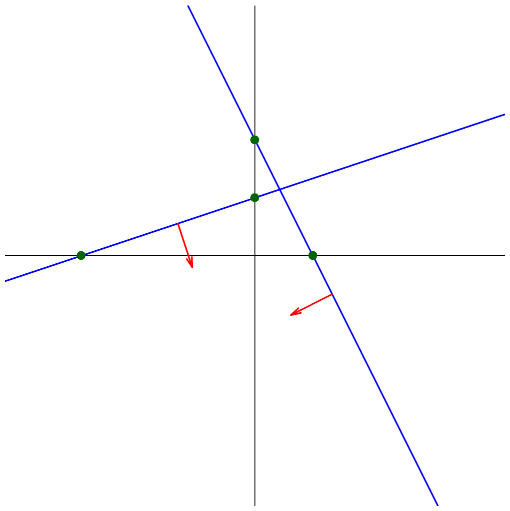

Figure 13: Dan's problem. Each line defines two half-planes, one of which will contain the origin. The problem is to find the intersection of all these half-planes, which will be a convex polygon enclosing the origin.

A function will have a stationary point within this polygon and the next step, (not considered here), is to find it. In practical cases the number of lines is likely to be large, typically in the thousands, and the job has to be done many times, as if in a simulation. So there is a premium on finding a slick algorithm for every step of the way.

An algorithm

The first version of the algorithm is simple enough. The steps involved are

- Find the *normals* to the lines, and hence the line *closest* to the origin.
- Rotate the plane so that this normal points South, i.e. the line is parallel to the $x-$axis.
- Find the points of intersection of this line with all the others.

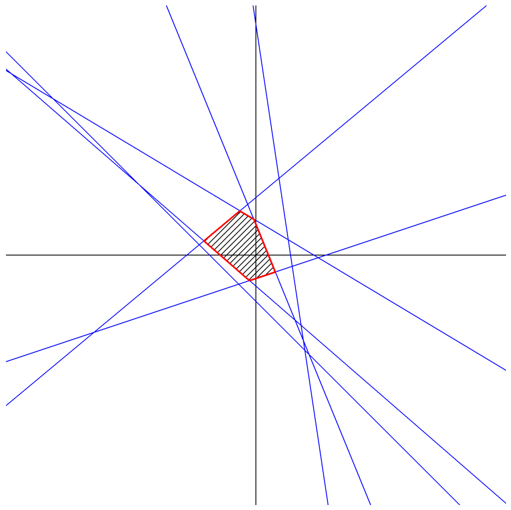

Figure 14: The minimum convex polygon enclosing the origin

- The two intersection points closest to the $y-$axis define the first (positive side) and last (negative side) corners of the minimum enclosing polygon.
- Rotate the plane again so that the *next* side is parallel to the $x-$axis. The next corner is then the intersection point with this line whose $x-$component is the smallest one larger than that of the current corner.
- Continue until the (known) last corner is reached.

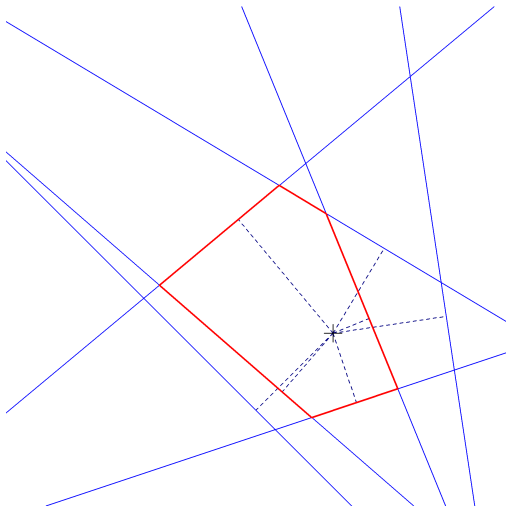

Figure 15: The normals to the lines

Representing the lines by a single complex number defined by their intercepts is convenient, and finding where two such lines meet is also simple, and more importantly, *vectorizable*. To find distances from the origin to lines, it is also necessary to be able to find the *normal*, i.e. the foot of the perpendicular to the line passing through the origin. This is also represented as a single complex number, and is an alternative way of representing the line itself.

To get the flavour of the computation, two key functions are shown in Figure 16 on the following page.

```
> normals
function(x, y = NULL) { ## normals from intercepts
    z <- with(xy.coords(x, y),
              complex(real = x, imaginary = y)) ## Lazy evaluation
    x <- Re(z)
    y <- Im(z)
    m <- Mod(z)
    (x/m)*(y/m)*complex(, y, x)  ## numerical caution!
  }
> intersections
function(n1, n2) { ## two lines in 'normal' form
    n12 <- n1*Conj(n2)
    lambda <- ifelse(Im(n12) == 0, as.complex(Inf),
                     (Re(n12) - Mod(n2)^2)/Im(n12))
    n1*(1 + 1i*lambda)
  }
```

Figure 16: Two key functions: finding the normals from the intercepts, and finding the points of intersection between two lines given in "normal" form

The code to find the "convex centre" is straightforward when using complex numbers. Rotating the plane is simply a matter of multiplying everything by a suitable complex number. Some code is shown in Figure

The breakthrough came when we realized we could identify the lines which define the convex centre directly from the normal form. The idea is as follows:

- The transformation $z \mapsto 1/z$ in the complex plane maps:
  - lines *not passing through* the origin into circles *passing through* the origin,
  - the (external) normal to the line to the diameter of the circle starting at the origin.
- The points defining the convex hull of the reciprocals of the normals give the lines, in order, defining the polygon enclosing the origin.

The key step in the code is

```
    h <- chull(1/normals(z))
```

which gets the indices of the lines enclosing the origin, *in clockwise order*, using a fast algorithm. That it happens to be part of the **grDevices** package does not preclude it from being useful elsewhere!

Some relative timings are shown in Figure 20 on page 27. For real cases the gain in speed by using the convex hull method is typically by a factor of about 12.

```
> convexCentre
function (x, y = NULL) {
    z <- with(xy.coords(x, y),
              complex(real = x, imaginary = y))
    nz <- normals(z)
    az <- sort(Arg(nz))
    if (max(diff(c(az, az[1] + 2 * pi))) > pi) {
      warning("the lines do not enclose the origin.")
      return(as.complex(Inf))
    }
    z <- nz
    j0 <- which.min(Mod(nz))
    nz <- nz * complex(argument = South - Arg(nz[j0]))
    r0 <- Re(intersections(nz[j0], nz))
    i0 <- which(r0 == max(r0[r0 < 0]))
    I <- i0
    J <- j0
    while (!any(duplicated(J))) {
      I <- c(I, j0)
      i0 <- j0
      j0 <- which(r0 == min(r0[r0 > 0]))
      J <- c(J, j0)
      nz <- nz * complex(argument = South - Arg(nz[j0]))
      r0 <- Re(intersections(nz[j0], nz))
      r0 <- r0 - r0[i0]
    }
    structure(intersections(z[I], z[J]), indices = cbind(I, J),
              class = "convexCentre")
  }
<environment: 0x06097b9c>
```

Figure 17: The key function: finding the convex centre as lines and corners

```
> find("chull")
[1] "package:grDevices"
> convexCentre2
function(x, y=NULL) { ## using convex hull assumption
  z <- with(xy.coords(x, y),
            complex(real = x, imaginary = y))
  h <- chull(1/normals(z))
  z0 <- convexCentre(z[h])
  attr(z0, "indices")[] <- h[attr(z0, "indices")]
  z0
}
```

Figure 18: Using the convex hull of the inverted plane to identify the key lines directly

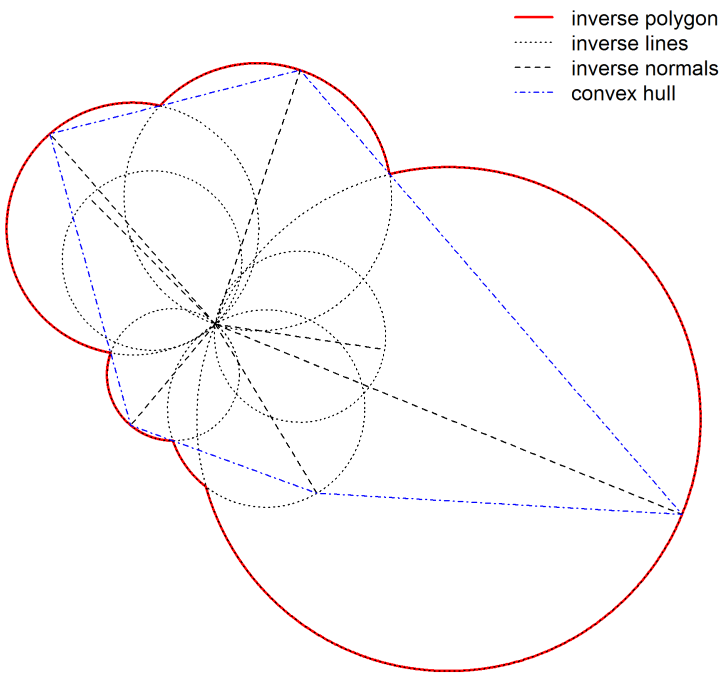

Figure 19: The inverse view of Figure 15: lines enclosing the origin become circles passing through the origin

```
> w <- complex(real = rt(500000, 5), imaginary = rt(500000, 5))
> rbind(orig = system.time(cc1 <- convexCentre(w))[1:3],
        hull = system.time(cc2 <- convexCentre2(w))[1:3])
     user.self sys.self elapsed
orig      4.74     0.65    5.11
hull      0.72     0.14    0.74
> identical(cc1,cc2)
[1] TRUE
> cc1
                       Corner  Line1  Line2
1 -1.803654e-07+1.338215e-05i    403 211851
2 -1.803665e-07-3.773626e-06i 211851 282935
3  4.040244e-06-3.773639e-06i 282935 229698
4  4.040266e-06+1.338213e-05i 229698    403
5 -1.803654e-07+1.338215e-05i    403 211851
```

Figure 20: Some timings for a large case, without and with the convex hull trick

Figure 21 shows the result of a real example with 4390 lines. As the problem is scale equivariant, the data have been scaled to make the diagram more convenient to display.

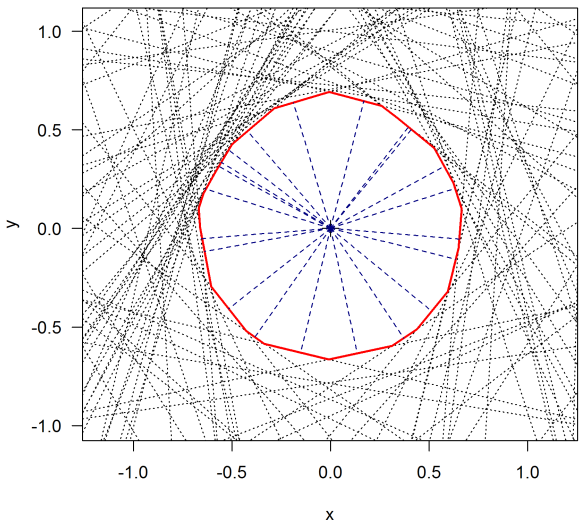

Figure 21: Dan's problem: A real example with 4390 lines

4.1 Higher dimensional cases

For the higher dimensional case the problem is much more complicated, but the same idea seems to hold. Lines are replaced by (hyper-)planes, but the normal to the plane is still the key.[5]

Corresponding to the inversion, a normal to a plane of length n is replaced by a vector whose length is $1/n$, still pointing in the same direction.[6] Finding the convex hull of the $n-$dimensional normals then finds the planes which enclose the origin, as required.

The **geometry** package, (Barber et al., 2012), already provides a function, `convhulln`, an interface to the **qhull** library, which may solve that problem, but work is still underway on this extension.

[5]Note that dropping a perpendicular from a point onto a hyper-plane is what is usually called fitting a linear model, in statistical terms.

[6]This actually corresponds to a transformation to the conjugate of the reciprocal, $z \mapsto 1/\bar{z}$, in two dimensions, which achieves the same result as the original.

5 The winds of change

"Language is a primary element of culture, and stasis in the arts is tantamount to death."
– Charles Marsh

5.1 Bending the problem to suit the computational device

When a statistical system or language seems to have run its course and hit a design wall, I have noticed that many devotees desperately clutch to their beloved system and can spend an inordinate amount of time, ingenuity and cunning trying to make the old system do what it was never designed to do. I've seen this with **minitab**, with early versions of **GenStat** and with **Glim**. I have done it myself.

The case of **Glim** is to me particularly interesting to me. It was a package (in the true, pejorative sense) designed to implement really one single, good idea: generalized linear models. Many problems that did not fit the mold were constrained to do so, simply because the temptation to use **Glim** was too great. In the early to mid 80's there was a series of ingenious papers in *Applied Statistics* showing how **Glim** could be used for all sorts of unexpected things such as fitting the von Mises distribution, (Lawson, 1988), fitting Cox regression models, with and without the EM algorithm, (Clayton and Cuzick, 1985; Whitehead, 1980), and fitting parametric survival models (Aitkin and Clayton, 1980).

The function `glm.nb` in the **MASS** package to fit Negative Binomial models originated as a tortuous **Glim** macro that I wrote in the early 80's. It was tricky to do and slow to work, even though Negative Binomials just miss out on being true generalized linear models by only a whisker. By comparison the **R** version was simplicity itself, even though the same algorithm, essentially, is still used.

I don't think this kind of distortion of the computational tool is an entirely bad thing. On the contrary, it can be used to show deeper theoretical connexions between techniques which on the surface appear to have little in common, and in the cases cited above, they do. But in general I think it is better to spend your ingenuity in solving real problems in statistics and data analysis even if it means some investment of time and energy in learning how to use new tools to do the computations.[7] On the other hand, one criterion of success for a software system[8] is met when people start to use it in ways that were not expected when it was designed.

5.2 Why is **R** so successful?

The examples I gave in the preceding section were meant to show how I have been able to use **R** for many years now as essentially my only computational tool. I do often write small pieces of **C**, or if I must, **Fortran**, and augment **R** that way, but largely I can do very well

[7]The issue is subtle, though. In my view the computations ought not to be considered separate from the theoretical aspects. There needs to be a continuous melding gong on.

[8]Which Ross Ihaka thinks may be due to Brian Kernighan, but I cannot find the quote.

with just using **R**, on quite large problems. That was never possible in the old **Glim** days, at least not without artificially distorting every problem to fit the limited **Glim** mold.

I suggest the fundamental reason why **R** has become so widespread and popular is that it achieves a very delicate balance between the simplicity and flexibility of an *interactive* system for rapid exploration of data and the generality of a true *programming* language.

It remains to be seen if this balance can continue to be met by a single system, or if two or more linked tools will become necessary to meet the challenges.

5.3 Some worrying trends?

The other, and in my view lesser reason for the success of **R** is another of John Chambers' facets: collaboration. The most tangible manifestation of this is the **CRAN** repository and its allies. John himself says that some of these initiatives are "messy", and gentleman that he is, leaves it at that. There are many gems in the **CRAN** collection, but there are many egregious counter-examples, some of which I need to mention, but without names, of course!

5.3.1 An egregious example

One package, which appears to be aimed at either school students or teachers, offers a suite of functions for some rather simple numerical processes. If it is ever used, (and the package now appears to be orphaned, but it is still on **CRAN**), I hope the luckless students are never invited to look at the code. Here is an example:

```
function (x) {
  start <- 1
  end <- length(x) + 1
  while (start < end) {
    y <- x[start]
    if (y > 0) {
      if (start == 1) {
        result = TRUE
      }
      else {
        result <- c(result, TRUE)
      }
    }
    else {
      if (start == 1) {
        result = FALSE
      }
      else {
        result <- c(result, FALSE)
      }
    }
    start <- start + 1
  }
  return(result)
}
```

This masterpiece of obfuscation is equivalent to

```
function(x) x > 0
```

except that the latter is more general: it works on zero-length vectors. If students are involved, it is no defence to say that "it works". It does more than work; it actively hinders understanding.

It gets worse. The package provides a function for finding lists of prime numbers. The algorithm is turgid, slow and wrong. It is so slow, a list of all the "prime" numbers it generates less that 10 million is provided as a data set. The very first entry on this list is wrong: it starts with 1. The next erroneous one is not that far down: 133, (= 7×19).

I know several people have tried to have the package either fixed or withdrawn, but without success. *Caveat emptor* indeed.

5.3.2 Three kinds of package

In my mind I recognize three kinds of package, or package suite, which *can* become worrying. These are:

A home for the refugee: packages that seek to emulate another system. These are very understandable developments and often have excellent code, but sooner or later, to exercise the full power of **R**, the user has to learn how to use **R**, not some version of **R** with the familiarity of home.

Two anonymous examples to illustrate what I would consider to have some unfortunate effects are as follows:

- There is a package (not named here, but there are, in fact, several) that, to be like **matlab**, masks `poly`, `reshape` and `toeplitz` from the **stats** package, `find` [sic!] and `fix` [sic!] from the **utils** package and `mode`, `real` and `strtrim` from the **base** package. The code itself, though, is very good.
- Another package, (which again I choose not to name), offers the user functions to do comprehensive analyses for a series of fairly standard ANOVA problems. During their operation, a plethora of results is printed onto the output stream, whether you want it or not.

  However that's *all* the get. The value returned from these functions is always `NULL`. Any form of diagnostic post-processing by the user is precluded, presumably as unnecessary given the volume of output. Moreover both the global environment and the search path are left littered with strange objects.

  It came as a surprise to me that someone with such an unorthodox view of how to use **R** managed to build a package in the first place, let alone get it on to **CRAN**, apparently without a hitch.

  Aficionados of statistical packages, though, would feel right at home.

Empires: suites of interconnected packages that offer someone's view of the world; how thing *ought* to be done.[9]

This is a curious one. In many cases the emperor has a point, but in too many others not, and the follower can lose touch with standard, portable **R** for little gain.

A more subtle danger with empires is the the tendency they can have to provide suites of "tools" for every conceivable need. This often starts off as being simply helpful but ultimately becomes emasculating, reducing the ability of users to program at a low enough level to harness the real power of **R**.

Where empires conflict they can fragment the **R** community, partitioning it into factions, and thus working against collaboration.

GUIs: which need no further explanation.[9]

While I fully admit the usefulness of GUIs for some classes of user, they do worry me in general. GUIs are clearly intended to make the learning curve much easier to climb. The problem is that they can lead to a dead end only part of the way up the hill.

More insidiously, the use of GUIs in education, particularly, can be based on the presumption that some users are incapable of harnessing the full power or **R**, or that they are unlikely ever to want to do so. Such a prejudicial assessment can only become self-fulfilling. Students have a wonderful power to surprise you, sometimes, and they are the future.

6 Quo vadis?

The tools we use must evolve, simply because the nature of data analysis itself is changing along with the challenges it has to face. Data is constantly increasing in size and complexity and hence is changing in the Marxian sense:

"Merely quantitative differences, beyond a certain point, pass into qualitative changes."
– Karl Marx, *Das Kapital, Kritik der politischen Ökonomie*, Vol. 1.

Our data analysis tools will need to evolve in parallel.

The pace of change in **R** is slowing.[10] This suggests it is reaching a kind of asymptote in its development and a sigh we should be looking to the future for something to replace it, or more likely, to complement it. It is not especially insightful[11] to point out that what we most urgently need are *programming* support tools to take advantage of the explosion in computational devices becoming available, such as Graphical Processing Units and other parallel processing mechanisms. There is no doubt we will need them. With the explosion of data itself, the data analysis community will face will need every bit of the new technology it can access. However there will always be a need for the kind of investigative work —

[9]I think it might be wise to refrain from giving concrete examples!

[10]As evidence I note that there is a plan for new releases to appear annually rather than biannually as at present.

[11]No pun intended

interactive data analysis — and that will require something 'not unlike' **R** well into the future. People will need to invest in more than one too.

Currently **R** has a momentum that seems to ensure it will continue essentially in its present form for some time to come. This has some good, and some not so good implications. Most computer languages, once they reach a certain point in their development, and acceptance by a user public, seem to carry on for much longer than anyone would have expected. Legacy code, and the accumulated investment in training and user experience seem to ensure changes do not happen as quickly as they should. Take **Fortran** 77, for example.

I don't think **R** is anywhere near that point yet, though there are some signs it is coming. Occasionally I find myself resorting to slick and obscure dodges just to squeeze and extra ounce of efficiency out of the system, and things do start to get somewhat artificial. This could be partly a hardware problem as much as software, though: I do have only two laptops these days, and they are 4 and 5 years old respectively.

Perhaps in future **R** can be maintained in a style somewhat like the TEX model. Development of the idea reaches a natural limit, but the tool remains useful well into the future. Other systems are built on top of it, such as LATEX, or to extend it, like BIBTEX and MAKEINDEX or even replace the basic tool with possibly better code that takes advantage of modern developments, such as PDFTEX, or indeed as **R** did with **S** itself.

References

Aitkin, M. and D. Clayton (1980). The fitting of exponential, weibull and extreme value distributions to complex censored survival data using GLIM. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 29*, 156–163.

Barber, C. B., K. Habel, R. Grasman, R. B. Gramacy, A. Stahel, and D. C. Sterratt (2012). *geometry: Mesh generation and surface tesselation*. CRAN. **R** package version 0.3-2.

Becker, G. (2012). *SearchTrees: Spatial Search Trees*. CRAN. **R** package version 0.5.1.

Chambers, J. M. (1998). *Programming with Data: A Guide to the* **S** *Language*. New York: Springer-Verlag.

Chambers, J. M. (2008). *Software for Data Analysis: Programming with* **R**. New York: Springer-Verlag.

Chambers, J. M. (2009). Facets of **R**. *The* **R** *Journal 1*, 5–8.

Clayton, D. and J. Cuzick (1985). The EM algorithm for Cox's regression model using GLIM. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 34*, 148–156.

Cook, J. D. (2012). Why and how people use **R**. `http://channel9.msdn.com/Events/Lang-NEXT/Lang-NEXT-2012/Why-and-How-People-Use-R`. See also `http://lambda-the-ultimate.org/node/4503` and `http://lambda-the-ultimate.org/node/4507`.

Hastie, T. (2011). *princurve: Fits a Principal Curve in Arbitrary Dimension*. CRAN. **R** package version 1.1-11, **R** port from the **S** original by Andreas Weingessel.

Hastie, T. and W. Stuetzle (1989). Principal curves. *Journal of the American Statistical Society 84*, 502–516.

Lawson, A. (1988). Fitting the von Mises distribution using GLIM. *Journal of Applied Statistics 15*, 255–260.

Marschner, I. C. (2011a). *glm2: Fitting Generalized Linear Models*. CRAN. **R** package version 1.0.

Marschner, I. C. (2011b, December). glm2: Fitting generalized linear models with convergence problems. *The* **R** *Journal 3*(2), 12–15.

Morandat, F., B. Hill, L. Osvald, and J. Vitek (2012). Evaluating the design of the **R** language: Objects and functions for data analysis. `http://www.cs.purdue.edu/homes/jv/pubs/ecoop12.pdf`. An ECOOP 2012 paper.

Venables, W. N. (1998). Exegeses on linear models. `http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf`. Paper presented to the **S-PLUS** User's Conference Washington, DC, 8-9th October, 1998.

Venables, W. N. and C. M. Dichmont (2004a). A generalized linear model for catch allocation: an example from Australia's Northern Prawn Fisherh. *Fisheries Research 70*, 409–426.

Venables, W. N. and C. M. Dichmont (2004b). GLMs, GAMs and GLMMs: an overview of the theory for applications in fisheries research. *Fisheries Research 70*, 319–337.

Venables, W. N., R. A. Kenyon, J. F. B. Bishop, C. M. Dichomnt, R. A. Deng, C. Burridge, B. R. Taylor, A. G. Donovan, S. E. Thomas, and S. J. Cheers (2006). Species distribution and catch allocaktion: Data and methods for the NPF, 2002-2004, final report. Technical report, Australian Fisheries Management Authority, Canberra, Australia.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with* **S** (Fourth ed.). New York: Springer. ISBN 0-387-95457-0.

Walsh, C. (2011). *pcurve: Principal curve analysis*. CRAN. R package version 0.6-3, **S** original by Trevor Hastie **S-PLUS** library by Glenn De'ath, **R** port by Chris Walsh.

Wang, J., R. Zamar, A. Marazzi, V. Yohai, M. Salibian-Barrera, R. Maronna, E. Zivot, D. Rocke, D. Martin, M. Maechler, and K. Konis. (2012). *robust: Insightful Robust Library*. CRAN. **R** package version 0.3-19.

Whitehead, J. (1980). Fitting Cox's regression model to survival data using GLIM. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 29*, 286–275.

Wood, S. N. (2003). Thin-plate regression splines. *Journal of the Royal Statistical Society (B) 65*, 95–114.

Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association 99*, 673–686.

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society (B) 73*, 3–36.

R Session information

- R version 2.15.0 (2012-03-30), `i386-pc-mingw32`
- Locale: `LC_COLLATE=English_Australia.1252`, `LC_CTYPE=English_Australia.1252`, `LC_MONETARY=English_Australia.1252`, `LC_NUMERIC=C`, `LC_TIME=English_Australia.1252`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: lattice 0.20-6, mgcv 1.7-17, SOAR 0.99-10
- Loaded via a namespace (and not attached): grid 2.15.0, Matrix 1.0-6, nlme 3.1-104, tools 2.15.0

Brisbane,
2012-06-08