SPATIAL MODELS FOR DISTANCE SAMPLING DATA:
RECENT DEVELOPMENTS AND FUTURE DIRECTIONS

APPENDIX A: EXAMPLE `DSM` ANALYSIS

DAVID L. MILLER, M. LOUISE BURT, ERIC A. REXSTAD AND LEN THOMAS

1. Introduction

The analysis is based on a dataset of observations of pantropical dolphins in the Gulf of Mexico (shipped with Distance 6.0). For convenience the data are bundled in an `R`-friendly format, although all of the code necessary for creating the data from the Distance project files is available at the below URL. The OBIS-SEAMAP page for the data may be found at http://seamap.env.duke.edu/dataset/25.

The intention here is to highlight the features of the `dsm` package, rather than perform a full analysis of the data. For that reason, some important steps are not fully explored. Some familiarity with density surface modelling is assumed.

This is a `knitr` document. The source for this document contains everything you need to reproduce the analysis given here (aside from the data). The most recent version of this document can be found at github.com/dill/mexico-data.

In section 2, we show how the data and packages can be loaded, before moving on to describing the data and then producing some exploratory plots (in sections 3 and 4, respectively). In section 5 we fit a detection function, then a selection of DSMs in section 6. A detection function with covariates is fitted in section 7. An example using correlation structures is shown in section 8, before concluding.

2. Preamble

Before we start, we load the `dsm` package (and its dependencies) and set some options:

```
library(dsm)
```

```
## Loading required package: mgcv
## This is mgcv 1.7-24.
## For overview type 'help("mgcv-package")'.
## Loading required package: mrds
## Loading required package: optimx
## Loading required package: numDeriv
## Loading required package: Rsolnp
## Loading required package: truncnorm
## Loading required package: parallel
## This is mrds 2.1.1 Built: R 3.0.1; ; 2013-06-25 16:32:18 UTC; unix
## Loading required package: ggplot2
## Loading required package: Distance
## This is dsm 2.0.5 Built: R 3.0.1; ; 2013-06-29 17:14:59 UTC; unix
```

```
# plotting options
gg.opts <- theme(panel.grid.major = element_blank(),
panel.grid.minor = element_blank(),
    panel.background = element_blank())

# make the results reproducible
set.seed(11123)
```

3. The data

3.1. **Observation and segment data.** All of the data for this analysis has been nicely pre-formatted and is shipped with `dsm`. Loading up that data, we can see that we have four data frames, the first few lines of each are shown:

```
data(mexdolphins)
attach(mexdolphins)
head(segdata)
```

```
##   latitude longitude Effort Transect.Label Sample.Label depth      x
## 1    29.94    -86.93  13800       19960417   19960417-1 135.0 134159
## 2    29.84    -86.83  14000       19960417   19960417-2 147.7 143496
## 3    29.75    -86.74  14000       19960417   19960417-3 152.1 152050
## 4    29.66    -86.65  13900       19960417   19960417-4 163.8 161102
## 5    29.56    -86.57  13800       19960417   19960417-5 179.7 169553
## 6    29.49    -86.49  13800       19960417   19960417-6 188.5 176793
##        y
## 1 325561
## 2 314055
## 3 304324
## 4 293475
## 5 282984
## 6 275103
```

```
head(distdata)
```

```
##     object size distance Effort detected beaufort latitude longitude
## 45      45   21   3296.6  36300        1        4    27.73    -86.00
## 61      61  150    929.2  17800        1        4    26.00    -87.63
## 63      63  125   6051.0  21000        1        2    26.01    -87.95
## 85      85   75   5499.7  21800        1        1    27.50    -90.45
## 114    114   50   7259.0  13400        1        3    27.41    -94.99
## 120    120   45   1454.8  20900        1        5    26.02    -95.97
##            x       y
## 45    228139   79258
## 61     69199 -113083
## 63    37046 -112197
## 85  -210016   54208
## 114 -658878   43337
## 120 -764824 -111005
```

```
head(obsdata)
```

```
##     object Sample.Label size distance Effort
## 45      45   19960421-9   21   3296.6  36300
## 61      61   19960423-7  150    929.2  17800
## 63      63   19960423-9  125   6051.0  21000
## 85      85   19960427-1   75   5499.7  21800
## 114    114   19960430-8   50   7259.0  13400
## 120    120   19960501-5   45   1454.8  20900
```

```
head(preddata)
```

```
##   latitude longitude depth      x      y width height
## 1    30.08    -87.58    35  70832 341079 32072  37065
## 2    30.08    -87.42    30  86868 341079 32072  37065
## 3    30.08    -87.25    27 102904 341079 32072  37065
## 4    30.08    -87.08    22 118940 341079 32072  37065
## 5    30.08    -86.92    46 134976 341079 32072  37065
## 6    29.92    -87.75    14  54888 322546 32126  37065
```

`distdata` holds the distance sampling data that will be used to fit the detection function. `segdata` holds the segment data: the transects have already been "chopped" into segments. `obsdata` holds the observations which have already been aggregated to the segments and `preddata` holds the prediction grid (which includes all the necessary covariates).

Typically (i.e. for other datasets) it will be necessary divide the transects into segments, and allocate observations to the correct segments using a GIS or other similar package, before starting an analysis using `dsm`.

3.2. **Converting units.** It is important to ensure that the measurements to be used in the analysis are in compatible units, otherwise the resulting estimates will be incorrect or hard to interpret. Having all of our measurements in SI units from the outset removes the need for conversion later, making life much easier. All of the data are already in the appropriate units (Northings and Eastings: kilometres from ~ -88.32 longitude, ~27.02 latitude, which is the centroid of the study region, multiplied up by 1000 to get the result in metres, for consistency).

We give an example of converting the survey area here to show that this is a simple process:

```
# centroid
lon0 <- -88.31951
lat0 <- 27.01594

sa.tmp <- latlong2km(survey.area$longitude, survey.area$latitude, lon0 = lon0,
    lat0 = lat0)
```

```
survey.area <- data.frame(x = 1000 * sa.tmp$km.e, y = 1000 * sa.tmp$km.n)

rm(sa.tmp)
```

The function `latlong2km` uses the spherical law of cosines to convert latitude and longitude into Northings and Eastings (thanks to Simon N. Wood for providing code). There is extensive literature about when particular projections of latitude and longitude are appropriate and we highly recommend the reader review this for their particular study area. The other data frames have already had their measurements appropriately converted. By convention the directions are named `x` and `y`.

Using latitude and longitude when performing spatial smoothing can be problematic when certain smoother bases are used. In particular when bivariate isotropic bases are used the non-isotropic nature of latitude and longitude is inconsistent (moving one degree in one direction is not the same as moving one degree in the other).

The below code generates Figure 1, which shows the survey area with the transect lines overlaid (using data from `segdata`).

```
p <- qplot(data = survey.area, x = x, y = y, geom = "polygon",
        fill = I("lightblue"), ylab = "y", xlab = "x", alpha = I(0.7))
p <- p + coord_equal()
p <- p + gg.opts

p <- p + geom_line(aes(x, y, group = Transect.Label), data = segdata)

print(p)
```

4. Exploratory data analysis

4.1. **Distance data.** The top panels of Figure 2, below, show histograms of observed distances and cluster size, while the bottom panels show the relationship between observed distance and observed cluster size, and the relationship between observed distance and Beaufort sea state. The plots show that there is some relationship between cluster size and observed distance (fewer smaller clusters seem to be seen at larger distances).

The following code generates Figure 2:

```
par(mfrow = c(2, 2))

# histograms
hist(distdata$distance, main = "", xlab = "Distance (m)")
hist(distdata$size, main = "", xlab = "Cluster size")

# plots of distance vs. cluster size
plot(distdata$distance, distdata$size, main = "", xlab = "Distance (m)",
    ylab = "Group size",  pch = 19, cex = 0.5, col = rgb(0.74, 0.74, 0.74, 0.7))

# lm fit
l.dat <- data.frame(distance = seq(0, 8000, len = 1000))
lo <- lm(size ~ distance, data = distdata)
```

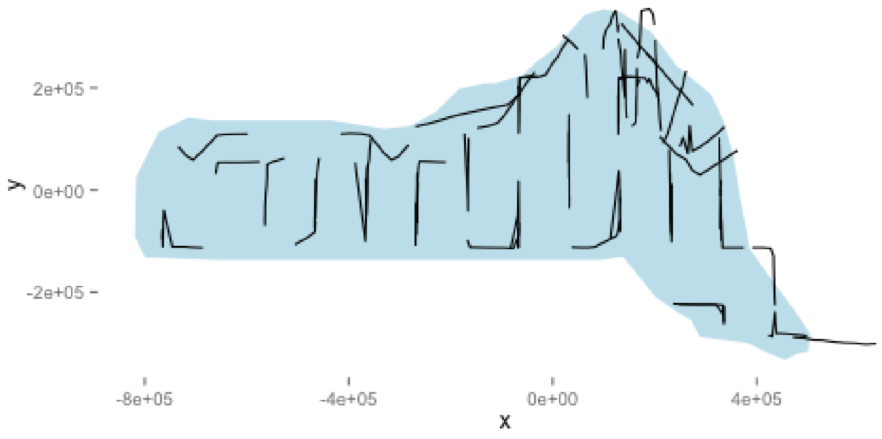

FIGURE 1. The survey area with transect lines.

```
lines(l.dat$distance, as.vector(predict(lo, l.dat)))

plot(distdata$distance, distdata$beaufort, main = "", xlab = "Distance (m)",
    ylab = "Beaufort sea state", pch = 19, cex = 0.5, col = rgb(0.74, 0.74,
        0.74, 0.7))
```

4.2. **Spatial data.** Looking separately at the spatial data without thinking about the distances, we can see the distribution of group size in space in Figure 3, below. Circle size indicates the size of the group in the observation. There are rather large areas with no observations, which might cause our variance estimates to be rather large. This plot shows that we don't seem to have many observations in the very shallow areas near the shore. This should make us skeptical of predictions in those areas. We will use depth later as an explanatory covariate in our spatial model. Figure 3 also shows the raw depth data.

The following code generates Figure 3:

```
p <- ggplot(preddata)
p <- p + gg.opts
p <- p + coord_equal()
p <- p + labs(fill = "Depth", x = "x", y = "y", size = "Group size")
p <- p + geom_tile(aes(x = x, y = y, fill = depth,
    width = width, height = height))
p <- p + geom_line(aes(x, y, group = Transect.Label), data = segdata)
p <- p + geom_point(aes(x, y, size = size), data = distdata, colour = "red",
```

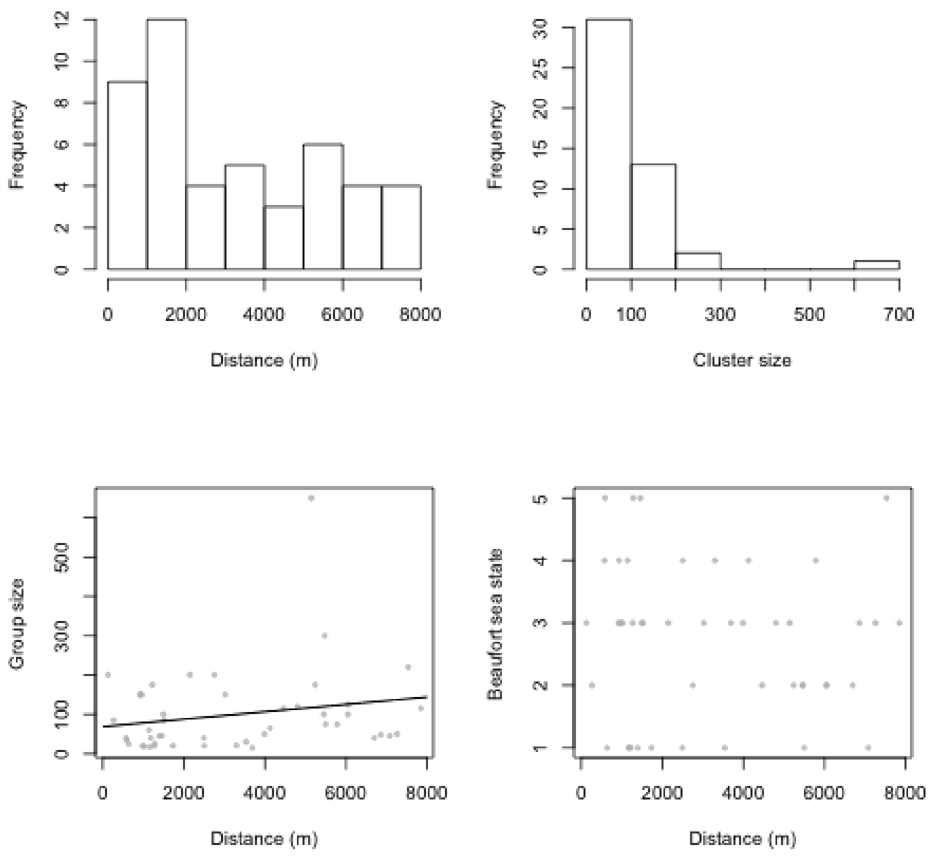

FIGURE 2. Exploratory plot of the distance sampling data. Top row, left to right: histograms of distance and cluster size; bottom row: plot of distance against cluster size and plot of distances against Beaufort sea state.

```
    alpha = I(0.7))
print(p)
```

5. Estimating the detection function

We use the `ds` function in the package `Distance` to fit the detection function. (The `Distance` package is intended to make standard distance sampling in `R` relatively straightforward. For a more flexible, but harder to use, alternaitve, see the function `ddf` in the `mrds` library.)

First, loading the `Distance` library:

```
library(Distance)
```

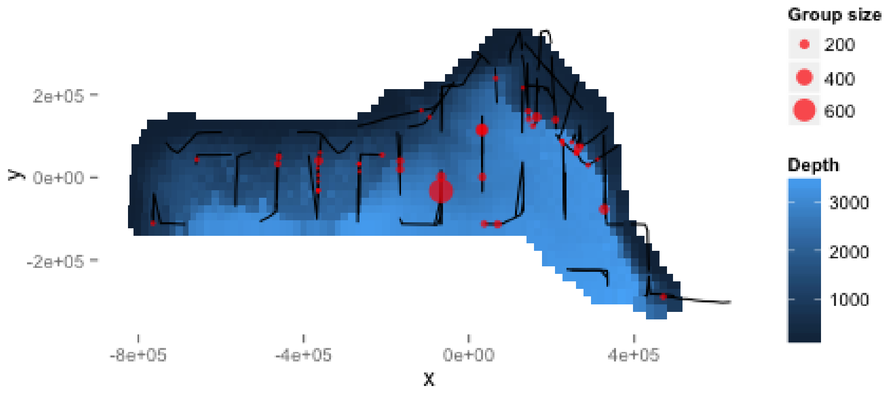

FIGURE 3. Plot of depth values over the survey area with transects and observations overlaid. Point size is proportional to the group size for each observation.

We can then fit a detection function with hazard-rate key with no adjustment terms:

```
hr.model <- ds(distdata, max(distdata$distance), key = "hr", adjustment = NULL)

## Fitting hazard-rate key function AIC= 841.253
## No survey area information supplied, only estimating detection function.

summary(hr.model)

##
## Summary for distance analysis
## Number of observations :  47
## Distance range         :  0  -  7847
##
## Model : Hazard-rate key function
## AIC   : 841.3
##
## Detection function parameters
## Scale Coefficients:
##             estimate     se
## (Intercept)    7.983 0.9532
##
## Shape parameters:
```

```
##             estimate     se
## (Intercept)        0 0.7835
##
##                       Estimate      SE     CV
## Average p               0.5913  0.2224 0.3762
## N in covered region  79.4879 30.8073 0.3876
```

The following code generates a plot of the fitted detection function (Figure 4):

```r
plot(hr.model)
```

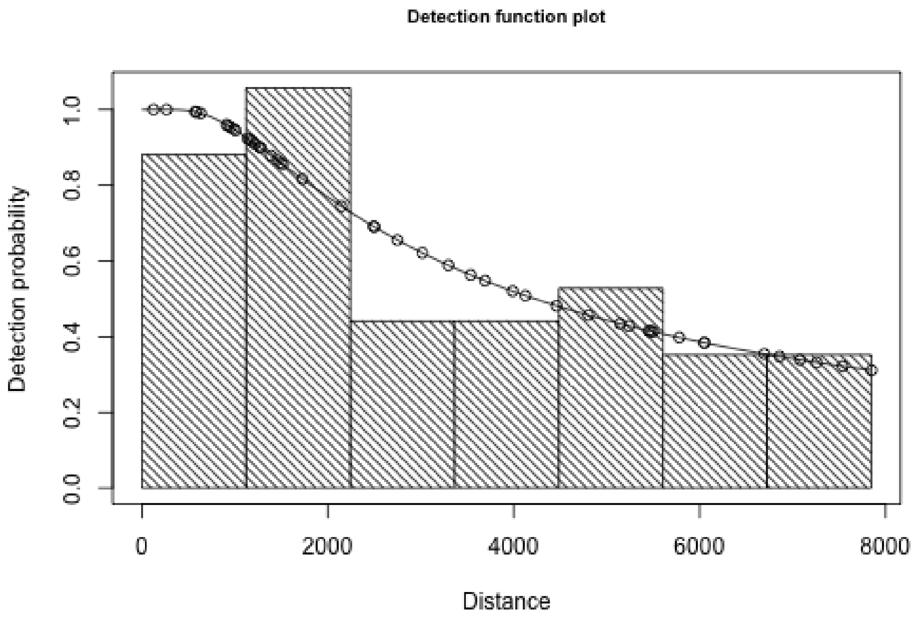

FIGURE 4. Plot of the fitted detection function.

For brevity, detection function model selection has been omitted here. In practise we would fit many different forms for the detection function. Later in this document, we demonstrate fitting of a detection function with size as a covariate, but for now we stick to a simple model.

6. FITTING A DSM

Before fitting a `dsm` model, the data must be segmented; this consists of chopping up the transects and attributing counts to each of the segments. As mentioned above, these data have already been segmented.

6.1. **A simple model.** We begin with a very simple model. We assume that the number of individuals in each segment are quasi-Poisson distributed and that they are a smooth function of their spatial coordinates (note that the formula is exactly as one would specify to `gam` in `mgcv`). The abundance of clusters/groups rather than individuals can be estimated by setting `group=TRUE` (though we ignore this here).

Running the model:

```
mod1 <- dsm(N ~ s(x, y), hr.model$ddf, segdata, obsdata)
summary(mod1)
```

```
##
## Family: quasipoisson
## Link function: log
##
## Formula:
## N ~ s(x, y) + offset(off.set)
## <environment: 0x10a71c2e8>
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -18.409      0.394   -46.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df    F p-value
## s(x,y) 26.1   28.2 5.61  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.113   Deviance explained =   44%
## GCV score = 42.985  Scale est. = 37.611    n = 387
```

We can then make the predictions over the the grid and calculate abundance. First we must create the offset (the area of each grid cell, which is 444km^2).

```
off.set <- 444 * 1000 * 1000
mod1.pred <- predict(mod1, preddata, off.set)
```

Figure 5 shows a map of the predicted abundance. Before plotting, we bind on the predicitons to the data used to create them:

```
pp <- cbind(preddata, mod1.pred)
p <- ggplot(pp) + gg.opts
p <- p + geom_tile(aes(x = x, y = y, fill = mod1.pred,
    width = width, height = height))
p <- p + coord_equal()
p <- p + geom_path(aes(x = x, y = y), data = survey.area)
p <- p + labs(fill = "Abundance")
print(p)
```

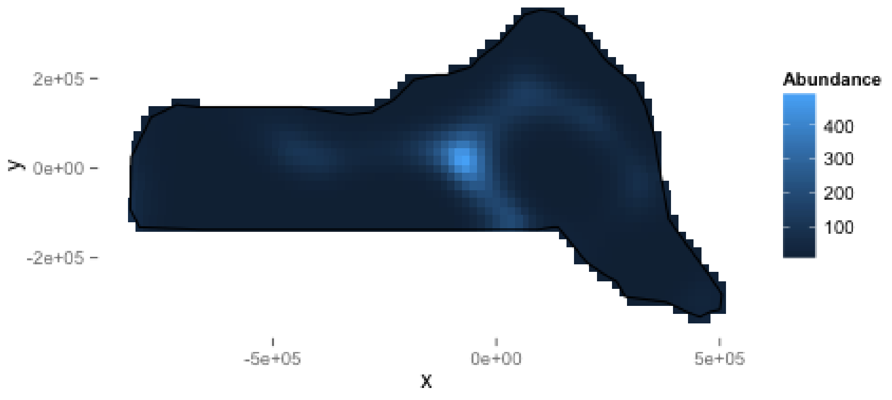

FIGURE 5. Predicted density surface for `mod1`.

We can calculate abundance over the survey area by simply summing these predictions:

```
sum(mod1.pred)
```

```
## [1] 47034
```

Figure 6 shows diagnostic plots for the model, generated with the following code:.

```
gam.check(mod1)
```

```
##
## Method: GCV   Optimizer: outer newton
## full convergence after 4 iterations.
## Gradient range [4.281e-07,4.281e-07]
## (score 42.99 & scale 37.61).
## Hessian positive definite, eigenvalue range [0.4516,0.4516].
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'   edf k-index p-value
## s(x,y) 29.00 26.06    1.02    0.98
```

These show that there is some deviation in the Q-Q plot. The “line” of points in the plot of the residuals vs. linear predictor plot corresponds to the zeros in the data.

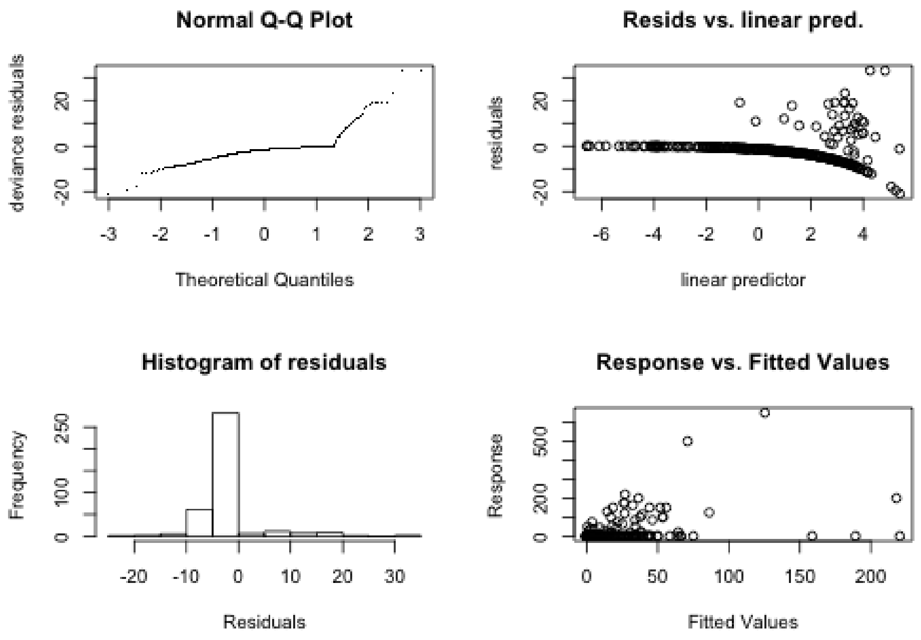

FIGURE 6. Diagnostic plots for `mod1`.

To check for residual autocorrelation we use the `dsm.cor` function:

```
dsm.cor(mod1, max.lag = 10)
```

The plot is shown in Figure 7, and appears to have a spike at lag 6, but this is beyond the range of being interesting.

We can use the approach of Williams *et al* (2011), which accounts for uncertainty in detection function estimation in this situation where we have no covariates in the detection function.

```
preddata.varprop <- split(preddata, 1:nrow(preddata))
offset.varprop <- as.list(rep(off.set, nrow(preddata)))
mod1.varprop <- dsm.var.prop(mod1, pred.data = preddata.varprop,
        off.set = offset.varprop)
```

Calling `summary` will give some information about uncertainty estimation:

```
summary(mod1.varprop)
```

```
## Summary of uncertainty in a density surface model calculated
##  by variance propagation.
##
## Quantiles of differences between fitted model and variance model
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -6.57e-13 -1.00e-15  1.20e-14  1.08e-13  1.11e-13  3.67e-12
```

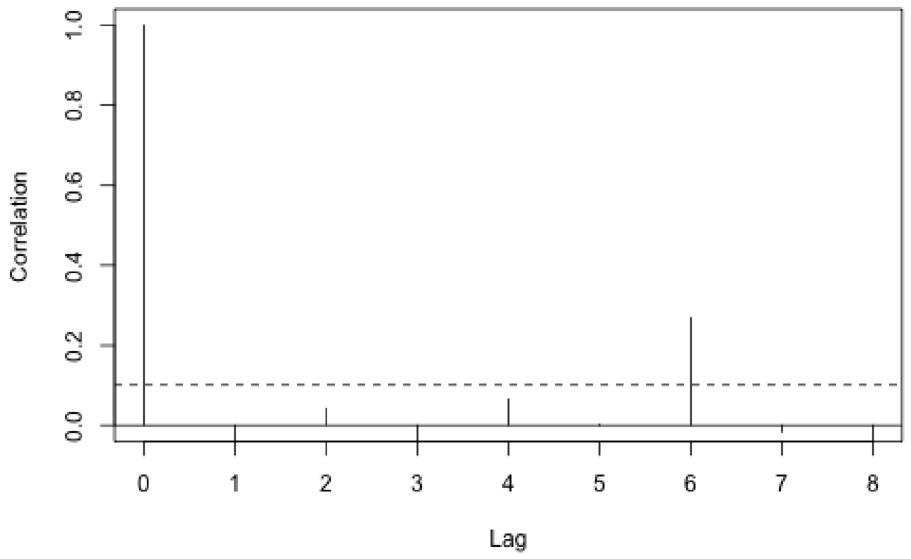

FIGURE 7. Residual autocorrelation in `mod1`.

```
##
## Approximate asymptotic confidence interval:
##     5%   Mean    95%
## 29962 47034 73832
## (Using delta method)
##
## Point estimate                  : 47034
## Standard error                  : 10966
## Coefficient of variation        : 0.2331
```

The section titled `Quantiles of differences between fitted model and variance model` can be used to check the variance model does not have major problems (values much less than 1 indicate no issues).

We can also make a plot of the CVs using the following code (shown in Figure 8).

```
plot(mod1.varprop, xlab = "Easting", ylab = "Northing")
```

6.2. **Adding another covariate to the spatial model.** The data set also contains a `depth` covariate (which we plotted above). We can include in the model very simply:

```
mod2 <- dsm(N ~ s(x, y, k = 10) + s(depth, k = 20), hr.model, segdata, obsdata,
    select = TRUE)
summary(mod2)
```

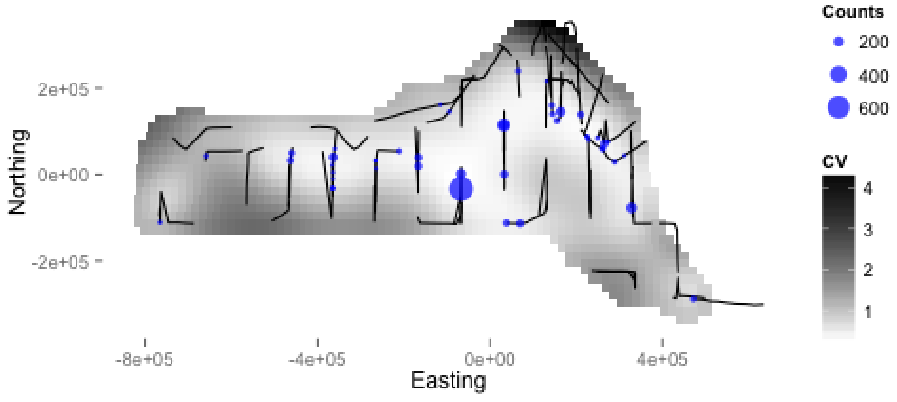

FIGURE 8. Plot of the coefficient of variation for the study area with transect lines and observations overlaid.

```
##
## Family: quasipoisson
## Link function: log
##
## Formula:
## N ~ s(x, y, k = 10) + s(depth, k = 20) + offset(off.set)
## <environment: 0x10b5839c8>
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -18.823      0.734   -25.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df    F p-value
## s(x,y)     3.88      9 2.06 0.00018 ***
## s(depth) 10.52     19 3.15 9.9e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0929   Deviance explained =   34%
## GCV score =  46.27  Scale est. = 42.967    n = 387
```

By setting the `k` parameter we specify the largest complexity for that smooth term in the model; as long as this is high enough (i.e., the number in the `edf` column is not too close to that in the `Ref.df` column), we can be sure that there is enough flexibility. However, it may sometimes be necessary to set `k` to be lower than this to limit the influence of (for example) spatial smoothers in the model.

Setting `select=TRUE` here imposes extra shrinkage terms on each smooth in the model (allowing smooth terms to be removed from the model during fitting; see `?gam` for more information). Although this is not particularly useful here, this can be a good way (along with looking at p-values) to perform term selection.

Again we can plot the predictions from this model (Figure 9, code below).

```
mod2.pred <- predict(mod2, preddata, off.set)
pp <- cbind(preddata, mod2.pred)
p <- ggplot(pp) + gg.opts
p <- p + labs(fill = "Abundance")
p <- p + geom_tile(aes(x = x, y = y, fill = mod2.pred,
    width = width, height = height))
p <- p + coord_equal()
p <- p + geom_path(aes(x = x, y = y), data = survey.area)
print(p)
```

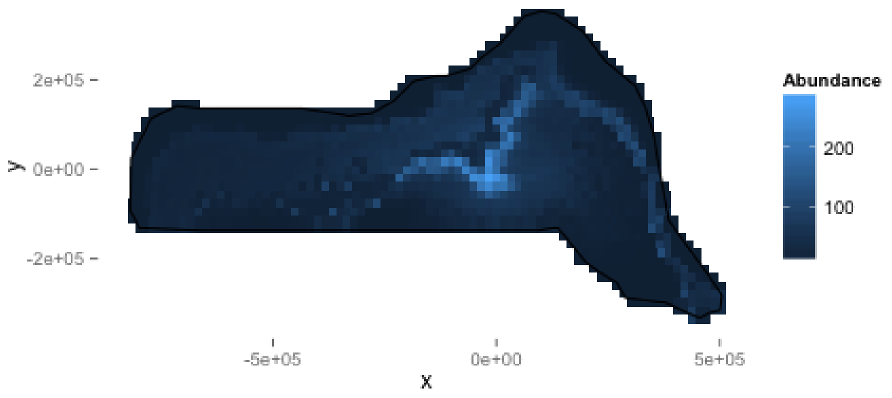

FIGURE 9. Predicted density surface for `mod2`.

Simply calling `plot` on the model object allows us to look at the relationship between depth and abundance (shown in Figure 10):

```
plot(mod2, select = 2)
```

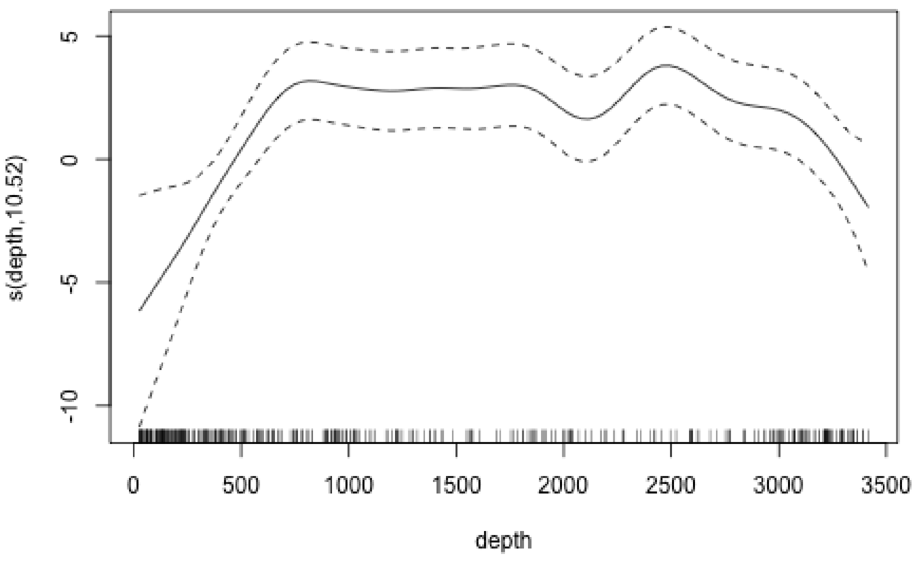

FIGURE 10. Plot of the smooth of depth in `mod2`.

Omitting the argument `select` in the call to `plot` gives plots of all the smooth terms, one at a time.

6.3. **A more complicated model.** ***Tweedie***

Response distributions other than the quasi-Poisson can be used, for example the Tweedie distribution. If the Tweedie is used, then the `p` parameter must be specified. The choice of `p` is only sensitive to the first decimal place, so a quick search can be performed by simply comparing the score of the resulting models. In this simple example we only show the difference between two values for `p`.

```
mod3.12 <- dsm(N ~ s(x, y), hr.model, segdata, obsdata,
    family = Tweedie(p =1.2))
summary(mod3.12)

##
## Family: Tweedie(1.2)
## Link function: log
##
## Formula:
## N ~ s(x, y) + offset(off.set)
## <environment: 0x10bb3f518>
##
```

```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -18.981      0.405   -46.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df    F p-value
## s(x,y) 27.3   28.7 6.27  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0531   Deviance explained = 47.9%
## GCV score = 23.284  Scale est. = 20.239    n = 387
```

As well as looking at the GCV/UBRE/REML score of the model to assess the value of `p` we can also look at a plot of the square root of the absolute value of the residuals versus the fitted values (Figure 11, left panel).

where as setting `p=1.7`:

```
mod3.17 <- dsm(N ~ s(x, y), hr.model, segdata, obsdata,
    family = Tweedie(p = 1.7))
summary(mod3.17)
```

```
##
## Family: Tweedie(1.7)
## Link function: log
##
## Formula:
## N ~ s(x, y) + offset(off.set)
## <environment: 0x1111f2790>
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -24.267      0.671   -36.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df    F p-value
## s(x,y) 28.8     29 14.4  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  -9.81   Deviance explained = 53.4%
## GCV score = 9.3865  Scale est. = 8.0932    n = 387
```

The plot (Figure 11, right panel; code below), appears to be much flatter.

```
par(mfrow = c(1, 2))
plot(sqrt(abs(residuals(mod3.12))), predict(mod3.12),
    xlab = "Square root of the\nabsolute value of the residuals",
    ylab = "Fitted values", main = "p=1.2")
plot(sqrt(abs(residuals(mod3.17))), predict(mod3.17),
    xlab = "Square root of the\nabsolute value of the residuals",
    ylab = "Fitted values", main = "p=1.7")
```

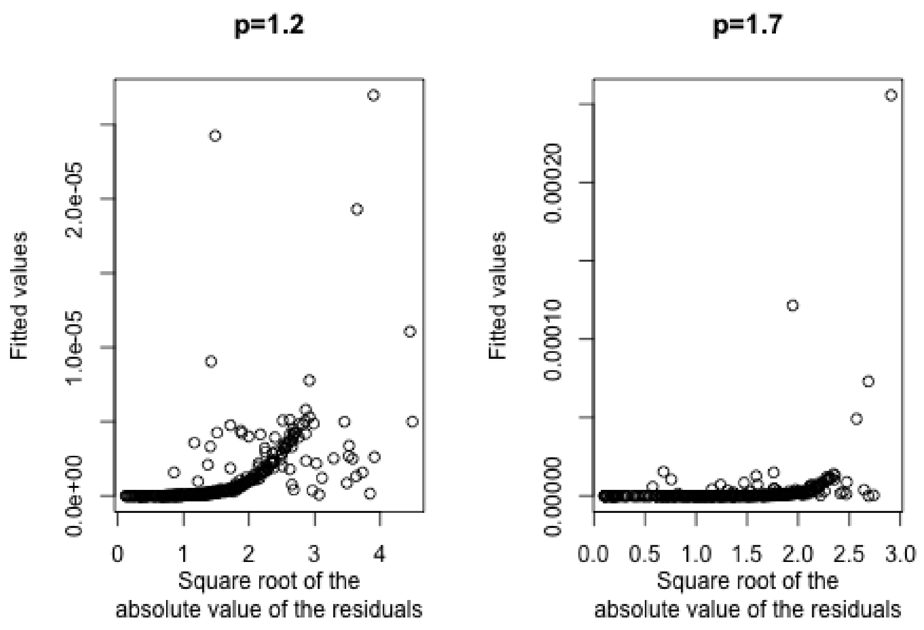

FIGURE 11. Plot of absolute value of the residuals versus the fitted values for the Tweedie model when `p=1.2` (left) and `p=1.7` (right). Note that the right plot is much flatter than the left.

Note also the improvement in GCV score.

In general, a "good" value can be found by simply plotting the above along with the GCV score for the model for values of `p` between 1.1 and 1.9 and looking for the best GCV and "flattest" plot.

Soap film smoothing

To account for a complex region (e.g., a region that includes peninsulae) we can use the soap film smoother (Wood et al. 2008).

To use a soap film smoother for the spatial part of the model we must create a set of knots for the smoother to use. This is easily done using the `make.soapgrid()` function in `dsm`:

```
soap.knots <- make.soapgrid(survey.area, c(11, 6))
# knot 11 is not outside but is too close according to soap...
soap.knots <- soap.knots[-11, ]
```

where the second argument specifies the number of points (in eadch direction) in the grid that will be used to create the knots (knots in the grid outside of `survey.area` are removed).

As we saw in the exploratory analysis, some of the transect lines are outside of the survey area. These will cause the soap film smoother to fail, so we remove them:

```
x <- segdata$x
y <- segdata$y
onoff <- inSide(x = x, y = y, bnd = as.list(survey.area))
rm(x, y)
segdata.soap <- segdata[onoff, ]
```

We can run a model with both the `depth` covariate along with a spatial (soap film) smooth. Note that the `k` argument now refers to the complexity of the boundary smooth in the soap film, and the complexity of the film is controlled by the knots given in the `xt` argument.

```
mod4 <- dsm(N ~ s(x, y, bs = "so", k = 10, xt = list(bnd = list(survey.area))) +
    s(depth), hr.model, segdata.soap, obsdata, knots = soap.knots)
```

```
## Loading required package: Matrix Loading required package: lattice
```

```
summary(mod4)
```

```
##
## Family: quasipoisson
## Link function: log
##
## Formula:
## N ~ s(x, y, bs = "so", k = 10, xt = list(bnd = list(survey.area))) +
##     s(depth) + offset(off.set)
## <environment: 0x10a3dfdf8>
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -19.40       0.62   -31.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df    F p-value
## s(x,y)    23.05  29.00 2.63   2e-07 ***
## s(depth)   7.08   7.89 2.57    0.01 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.183   Deviance explained = 45.9%
## GCV score = 45.227  Scale est. = 38.341    n = 365
```

Comparing predictions from the model that included a smooth of depth, we can see that the soap film has prevented some of the extreme values (especially in the lower right corner of the survey area). This is shown in Figure 12 (figure created with the code below).

```
mod4.pred <- predict(mod4, preddata, off.set)
pp <- cbind(preddata, mod4.pred)

p <- ggplot(pp) + gg.opts
p <- p + geom_tile(aes(x = x, y = y, fill = mod4.pred,
    width = width, height = height))
p <- p + coord_equal()
p <- p + geom_path(aes(x = x, y = y), data = survey.area)
p <- p + labs(fill = "Abundance")
print(p)
```

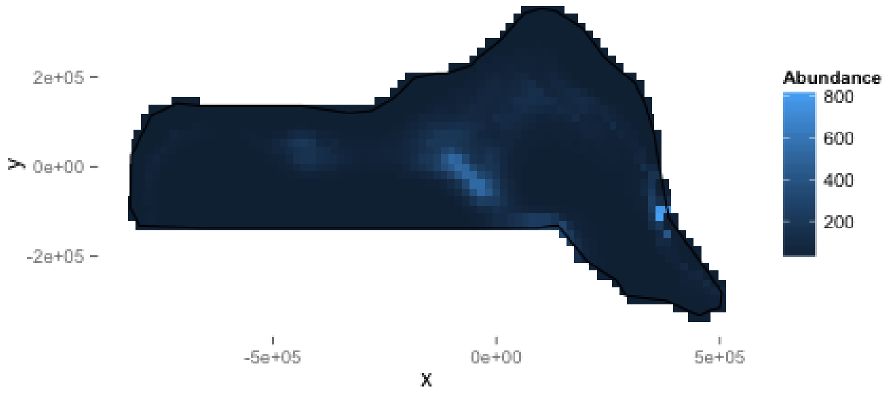

FIGURE 12. Predicted density surface for mod4.

7. ADDING COVARIATES TO THE DETECTION FUNCTION

It is common to include covariates in the detection function (so-called Multiple Covariate Distance Sampling or MCDS). In this dataset there are two covariates that were collected on each individual: Beaufort sea state and size. For brevity we fit only a hazard-rate detection functions with the sea state included as a factor covariate as follows:

```
hr.beau.model <- ds(distdata, max(distdata$distance),
    formula = ~as.factor(beaufort), key = "hr", adjustment = NULL)
summary(hr.beau.model)
```

```
##
## Summary for distance analysis
## Number of observations :  47
## Distance range         :  0  -  7847
##
## Model : Hazard-rate key function
## AIC   : 843.7
##
## Detection function parameters
## Scale Coefficients:
##                      estimate     se
## (Intercept)           7.66318  1.077
## as.factor(beaufort)2  2.27968 17.367
## as.factor(beaufort)3  0.28606  1.019
## as.factor(beaufort)4  0.07174  1.223
## as.factor(beaufort)5 -0.36399  1.537
##
## Shape parameters:
##             estimate    se
## (Intercept)   0.3004 0.518
##
##                      Estimate      SE     CV
## Average p              0.5421  0.1751 0.3229
## N in covered region   86.6957 29.4133 0.3393
```

In this example, the detection function with covariates does not give a lower AIC than the model without covariates (hazard-rate model has AIC of 841.25 vs. 843.71 for this model). Looking back to the bottom-right panel of Figure 2, we can see there is not a discernible pattern in the plot of Beaufort vs distance.

The code to fit the model is similar to the other models, above. However since we are now using a detection function with observation-level covariates, we change the response to be `Nhat` so the abundances are estimated per segment and change the name ofthe detection function model object:

```
mod5 <- dsm(Nhat ~ s(x, y), hr.beau.model, segdata, obsdata)
summary(mod5)
```

```
##
## Family: quasipoisson
## Link function: log
##
## Formula:
## Nhat ~ s(x, y) + offset(off.set)
## <environment: 0x107da5320>
##
```

```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -18.258      0.351     -52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df    F p-value
## s(x,y)  26   28.1 5.77  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.118   Deviance explained = 42.9%
## GCV score =  75.24  Scale est. = 65.855    n = 387
```

Note that for models where there are covariates at the individual level we cannot calculate the variance via the variance propagation method (`dsm.var.prop`) of Williams et al (2011). Instead we can use a GAM uncertainty estimation and combine it with the detection function uncertainty via the delta method (`dsm.var.gam`), or use the moving block bootstrap. Other than this, all of the above functions can be used.

A plot of predictions from the covariate model:

```
mod5.pred <- predict(mod5, preddata, off.set)
pp <- cbind(preddata, mod5.pred)

p <- ggplot(pp) + gg.opts
p <- p + geom_tile(aes(x = x, y = y, fill = mod5.pred,
    width = width, height = height))
p <- p + coord_equal()
p <- p + geom_path(aes(x = x, y = y), data = survey.area)
p <- p + labs(fill = "Abundance")
print(p)
```

8. Correlation structures

We can use a generalized mixed model (GAMM; Wood, 2006) to include correlation between the segments within each transect. First we re-code the sample labels and transect labels as numeric variables, then include them in the model as part of the `correlation` argument. For the sake of example we use an AR1 (lag 1 autocorrelation) correlation structure (though the correlogram did not indicate we had issues with residual autocorrelation, we show it here for illustrative purposes).

```
segdata$sg.id <- as.numeric(segdata$Sample.Label)
segdata$tr.id <- as.numeric(segdata$Transect.Label)
mod1.gamm <- dsm(N ~ s(x, y), hr.model$ddf, segdata, obsdata, engine = "gamm",
    correlation = corAR1(form = ~sg.id), method = "REML")
```

```
## Loading required package: nlme
```

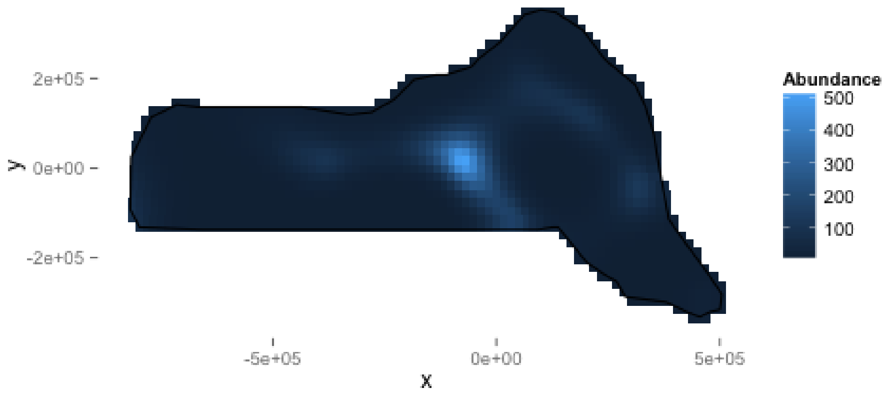

FIGURE 13. Predicted density surface for `mod5`.

```
##
##  Maximum number of PQL iterations:  20
```

```
## iteration 1 iteration 2 iteration 3 iteration 4 iteration 5 iteration 6
## iteration 7 iteration 8 iteration 9 iteration 10
```

This example also includes using `method="REML"` for smoothing parameter selection.

GAMMs usually take considerably longer to fit than GAMs, so it's usually start with a GAM first, select smooth terms and response distribution before starting to fit GAMMs.

The object returned is part `lme` (for the random effects) and part `gam` (for the smooth terms). Looking at the `summary()` for the `gam` part of the model:

```
summary(mod1.gamm$gam)
```

```
##
## Family: quasipoisson
## Link function: log
##
## Formula:
## N ~ s(x, y) + offset(off.set)
## <environment: 0x108fb1ed0>
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.617      0.277   -63.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df    F p-value
## s(x,y) 20.4   20.4 4.08 1.5e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.129  Scale est. = 68.686    n = 387
```

And run checks on the GAM part of the object (shown in Figure 14):

```
gam.check(mod1.gamm$gam)
```

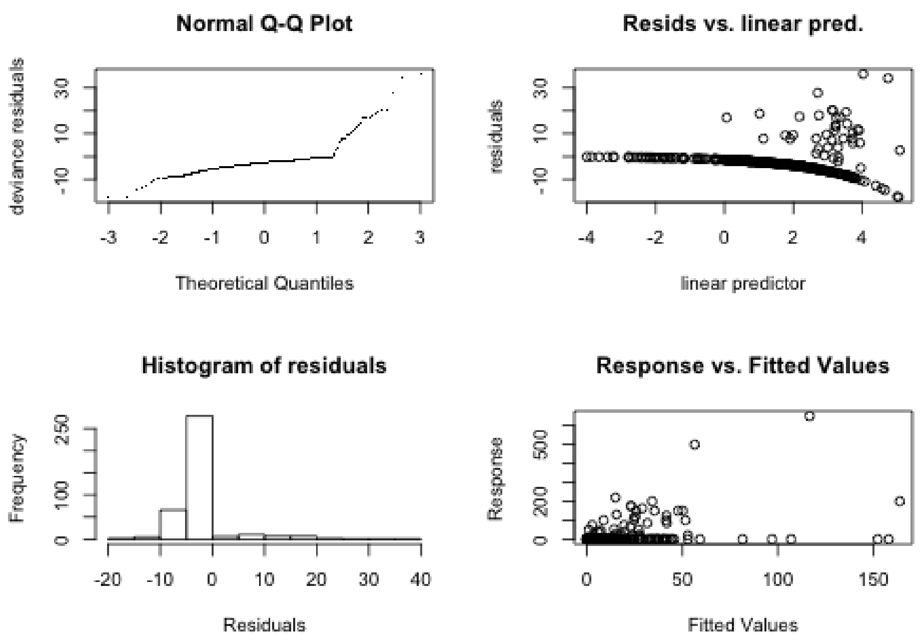

FIGURE 14. Diagnostic plots for `mod1.gamm`.

We can now make predictions and compare the abundance map produced by this model to the previous models (Figure 15):

```
mod1.gamm.pred <- predict(mod1.gamm, preddata, off.set)
pp <- cbind(preddata, N = mod1.gamm.pred)
p <- ggplot(pp) + gg.opts
p <- p + geom_tile(aes(x = x, y = y, fill = N, width = width, height = height))
p <- p + coord_equal()
p <- p + geom_path(aes(x = x, y = y), data = survey.area)
```

```
p <- p + labs(fill = "Abundance")
print(p)
```

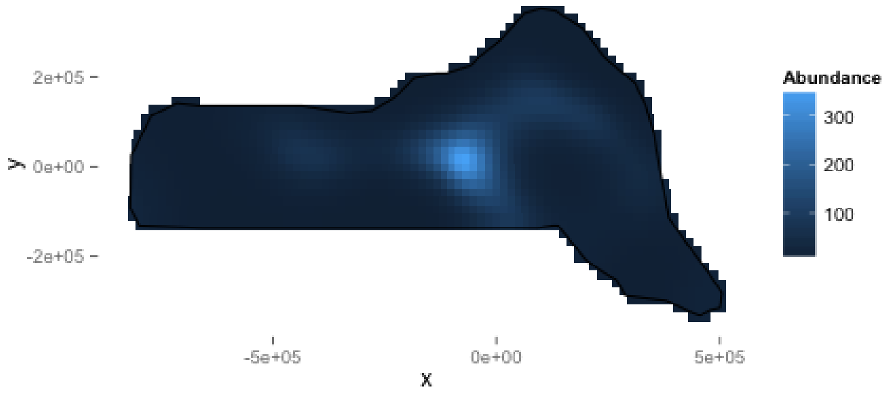

FIGURE 15. Predicted density surface for `mod1.gamm`.

Again, estimating variance is straightforward using the variance propagation method:

```
mod1.gamm.var <- dsm.var.prop(mod1.gamm, pred.data = preddata, off.set = off.set)

##
##  Maximum number of PQL iterations:  20

## iteration 1 iteration 2 iteration 3 iteration 4 iteration 5 iteration 6
## iteration 7 iteration 8 iteration 9

summary(mod1.gamm.var)

## Summary of uncertainty in a density surface model calculated
##  by variance propagation.
##
## Quantiles of differences between fitted model and variance model
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -4.120  -0.531  -0.244  -0.073  -0.019  19.500
##
## Approximate asymptotic confidence interval:
##    5%  Mean   95%
## 35389 45701 59018
## (Using delta method)
##
## Point estimate                 : 45701
## Standard error                 : 5988
## Coefficient of variation       : 0.131
```

Comparing this to the variance from `mod1`, we can see the GAMM offers a significant reduction in variance:

```
mod1.varprop <- dsm.var.prop(mod1, pred.data = preddata.varprop,
    off.set = offset.varprop)
summary(mod1.varprop)
```

```
## Summary of uncertainty in a density surface model calculated
##  by variance propagation.
##
## Quantiles of differences between fitted model and variance model
##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -6.57e-13 -1.00e-15  1.20e-14  1.08e-13  1.11e-13  3.67e-12
##
## Approximate asymptotic confidence interval:
##    5%  Mean   95%
## 29962 47034 73832
## (Using delta method)
##
## Point estimate                 : 47034
## Standard error                 : 10966
## Coefficient of variation       : 0.2331
```

More information on `lme` can be found in Pinheiro and Bates (2000) and Wood (2006).

9. Conclusions

This document has shown that the `dsm` package is a versatile and relatively easy-to-use package for the analysis of spatial distance sampling data. Note that there are many possible models that can be fitted using `dsm` and that the aim here was to show just a few of the options. Results from the models can be rather different, so care must be taken in performing model selection, discrimination and criticism.

Notes

- `Distance` is available at http://github.com/dill/Distance as well as on CRAN.
- `dsm` is available (along with some documentation and hints) at http://github.com/dill/dsm, as well as on CRAN.

References

- Hedley, S.L. & Buckland, S.T. (2004) Spatial models for line transect sampling. Journal of Agricultural, Biological, and Environmental Statistics, 9, 181–199.

- Pinheiro, J.C., and Bates, D.M. (2000) Mixed-effects models in S and S-PLUS, Springer.
- Williams, R., Hedley, S.L., Branch, T.A., Bravington, M.V., Zerbini, A.N. & Findlay, K.P. (2011) Chilean blue whales as a case study to illustrate methods to estimate abundance and evaluate conservation status of rare species. Conservation Biology, 25, 526–535.
- Wood, S.N. (2006) Generalized Additive Models: an introduction with R. Chapman and Hall/CRC Press.