Worked Example for Offshore Redistribution Impact Data

Using the `MRSea` Package

Monique Mackenzie
Eric Rexstad
Lindesay Scott-Hayward
Cornelia Oedekoven

Centre for Research into Ecological and Environmental Modelling University of St. Andrews

Table of contents

Contents

1 Introduction

This chapter takes you through the process of fitting spatial models using the CReSS method in a GEE framework with SALSA for model selection. We use simulated vantage point data as our case study. Here, the type of impact was a redistribution of animals within the study area.

Aim

Produce a density surface map along with estimates of uncertainty and identify any significant effect of an impact event (e.g. nearshore renewable installation).

Note: No distance sampling analysis can be conducted as there is no information available to do so.

The following sections take you through:

- model fitting
- checking diagnostics
- making predictions and inference

2 Preamble

Before we start, we load the `MRSea` package and its dependencies. This may require installing the following packages if they are not already installed on your computer:

- mrds, lawstat, car, mvtnorm, splines, geepack, ggplot2, calibrate, Matrix and fields.

After installing these packages, the following command will load package `MRSea` and these packages into the active workspace.

```
require(MRSea)
```

This contains the data you will need to follow this example and all of the functions.

To find what data sets are available for testing use the following command:

```
data(package="MRSea")
```

```
Data sets in package MRSea:

dis.data.de                Line transect data with decrease post-impact
dis.data.no                Line transect data with no post-impact consequence
dis.data.re                Line transect data with redistribution post-impact
knotgrid.ns                Knot grid data for nearshore example
knotgrid.off               Knot grid data for offshore example
ns.data.de                 Nearshore data with decrease post-impact
ns.data.no                 Nearshore data with no effect of impact
ns.data.re                 Nearshore data with redistribution post-impact
ns.predict.data.de         Prediction grid data for nearshore post-impact decrease
ns.predict.data.no         Prediction grid data for nearshore no post-impact consequence
ns.predict.data.re         Prediction grid data for nearshore post-impact redistribution
predict.data.de            Prediction grid data for post-impact decrease
predict.data.no            Prediction grid data for no post-impact consequence
predict.data.re            Prediction grid data for post-impact redistribution
```

3 Loading the Data

3.1 Data Requirements

There are a few requirements for analysing data using the `MRSea` package.

- The geographic coordinates must be labelled `x.pos` and `y.pos`
- There must be a column labelled `area` representing the effort associated with each coordinate

```
# Loading the data
data(ns.data.re)
data <- ns.data.re
attach(data)
head(data, n=3)
```

```
  x.pos y.pos   area  floodebb observationhour GridCode Year
1  1500 -4500 0.3853       EBB              12      a11    9
2  1500 -4500 0.3853     FLOOD               8      a11    9
3  1500 -4500 0.3853     FLOOD               9      a11    9

  DayOfMonth MonthOfYear impact  birds cellid
1         13           3      0      0      1
2         16           3      0      0      2
3         16           3      0      0      3
```

3.2 Exploratory Data Analysis

Checking the raw data by plotting.

- We can identify data entry errors and look for general patterns.
- This data was collected by a cliff-top observer (grey dot, Figure 1)

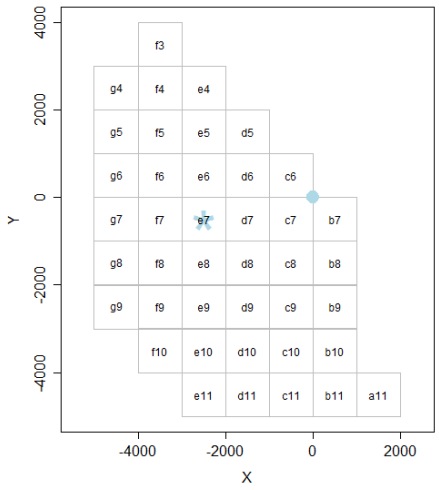


Figure 1: Grid cell identifiers for the study region. The grey circle is the location of a cliff-top observer and the star the site of the impact event.

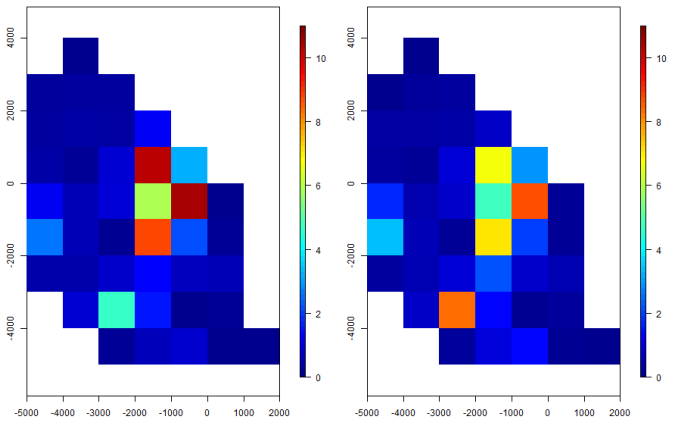


Figure 2: Estimated bird counts before (left) and after (right) an impact event. Each cell is 1 km^2 and the colour represents bird count.

- Most animals are seen near to the observer (Figure 2)

- Few are seen to the far right and far left of the observer

We can also assess the relationships of available covariates and our response (animal counts) by plotting.

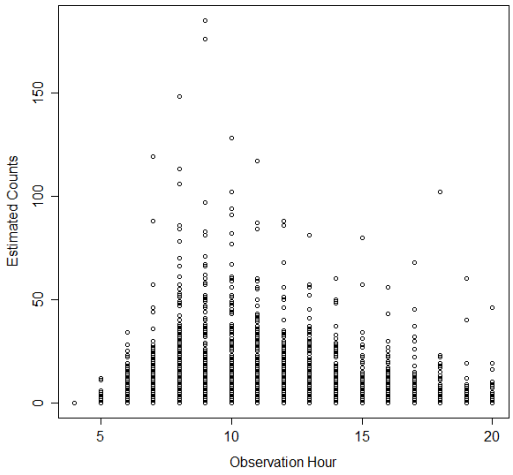


Figure 3: Plot of observation hour against the estimated bird counts.

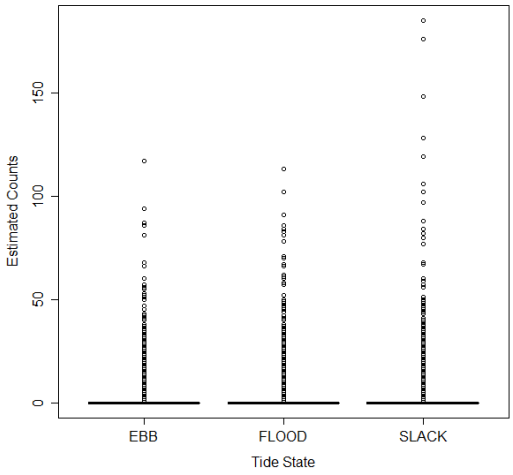


Figure 4: Plot of tide state against the estimated bird counts.

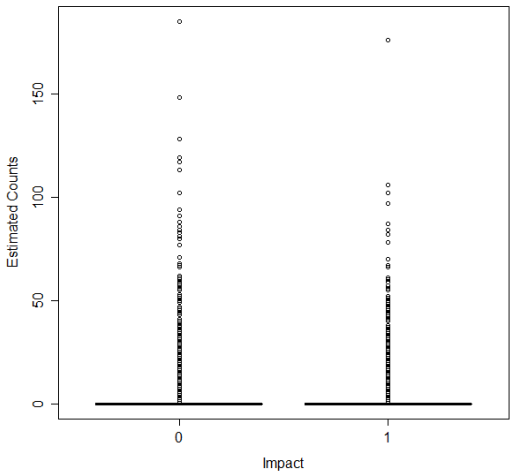


Figure 5: Plot of Impact against the estimated bird counts. Zero is pre impact ans one is post impact.

EDA

- Birds were seen predominantly in the morning hours (7-12).
- Few birds were seen very early and very late.
- Non-linear relationship between observation hour and bird counts.
- Difficult to identify any relationship between tide state/impact and bird counts due to the large number of zeros in the data.

3.3 Checking for collinearity between variables

Variance Inflation Factors (VIFs)

These are used to assess collinearity between covariates and to tell us by how much the standard error is inflated by the other variables in the model.

- Generalised VIFs (GVIFs) are calculated, because the covariates have more than one degree of freedom, and adjusted ($\text{GVIF}_{\text{adj}} = \text{GVIF}^{1/2*Df}$) for the number of degrees of freedom.
- GVIF_{adj} gives us the decrease in precision of estimation due to collinearity (equivalent to $\sqrt{VIF}$).

- For example, a GVIF_{adj} of 2 means that the confidence intervals are twice as wide as they would be for uncorrelated predictors.

```
fullModel <- glm(birds ~ as.factor(floodebb) + as.factor(impact) +
    observationhour + x.pos + y.pos, family = poisson, data = data)
vif(fullModel)
```

```
                     GVIF Df GVIF^(1/(2*Df))
as.factor(floodebb) 1.006  2           1.002
as.factor(impact)   1.000  1           1.000
observationhour     1.006  1           1.003
x.pos               1.323  1           1.150
y.pos               1.323  1           1.150
```

```
[1] "Maximum VIF is: 1.15"
```

Conclusion:

The maximum vif is approximately 1 which means that there is no issue with collinearity. (standard errors are not inflated).

Checking Factor Covariates

Each level of a factor based covariate must have some non-zero entries for the response variable, otherwise there will be problems in the model fitting process

```
# check all factor levels have counts
checkfactorlevelcounts(factorlist=c("floodebb", "impact"), data, data$birds)
```

```
[1] "floodebb will be fitted as a factor variable; there are non-zero counts
 for all levels"
[1] "impact will be fitted as a factor variable; there are non-zero counts
 for all levels"
```

4 Fitting a Model

Here we are going to fit a GEE-CReSS model with SALSA for knot selection. Recap:

Generalised Estimating Equations (GEE)

Framework to allow for correlated errors.

Complex Region Spatial Smoother (CReSS)

Flexible spatial smoothing method.

Spatially Adaptive Local Smoothing Algorithm (SALSA)

Automated knot selection procedure for both one-dimensional (e.g. depth) and two-dimensional (e.g. geographic space) covariates. The knots are sources of flexibility in the surface that can raise or lower the surface.

4.1 Fitting a Smooth Term

A smooth of observation hour

First we set up an object (`splineParams`) that contains the information required by SALSA for adaptive knot placement.

- Each covariate that might be considered smooth is a list entry in `splineParams`
- The list contains the covariate name, data, initial knot location (one knot at the mean), the boundary knots (greatest range of prediction data and `data`) and the degree of the smooth.
- Note: The smooth one dimensional covariates appear in the `splineParams` object starting at slot `[[2]]`. Slot `[[1]]` is reserved for the spatial term.

```
# load prediction data
data(ns.predict.data.re)
predictData<-ns.predict.data.re

# make splineParams object
splineParams<-makesplineParams(data=data, varlist=c('observationhour'),
         predictionData=predictData)
str(splineParams)

List of 2
 $ : list()
 $ :List of 5
  ..$ covar      : chr "observationhour"
  ..$ explanatory: int [1:27798] 12 8 9 10 11 12 13 14 15 8 ...
  ..$ knots      : num 12
  ..$ bd         : num [1:2] 4 20
  ..$ degree     : num 2
```

```
fullModel <- glm(birds ~ as.factor(floodebb) + as.factor(impact) +
    bs(observationhour, knots = splineParams[[2]]$knots) + x.pos + y.pos, family =
quasipoisson,  data = data)
```

4.2 Checking for Correlation

We fit a model containing all covariates of interest (`fullModel` above) and carry out some **runs tests** to check for **correlated residuals** (the model assumption is uncorrelated residuals).

Runs Test

This is a test for randomness and allows us to determine if we have correlation in our model residuals. We will see a large p-value (H_0: uncorrelated residuals) and good mixing of the profile plot if there is no correlation.

The `runs.test` function can be found in the `lawstat` library.

```
runs.test(residuals(fullModel, type = "pearson"), alternative = c("two.sided"))


Runs Test - Two sided
data:  residuals(fullModel, type = "pearson")
Standardized Runs Statistic = -71.59, p-value < 2.2e-16
```

- The small p-value ($p << 0.05$) indicates that there is an issue with correlation in the residuals
- The test statistic is negative, which indicates that there are fewer runs of residuals than would be expected if there were un-correlated residuals. We have positive correlation.
- This result is also shown on the runs profile plot (Figure 6).

Plotting runs profile

```
plotRunsProfile(fullModel,  varlist = c("observationhour"))


[1] "Calculating runs test and plotting profile"
```

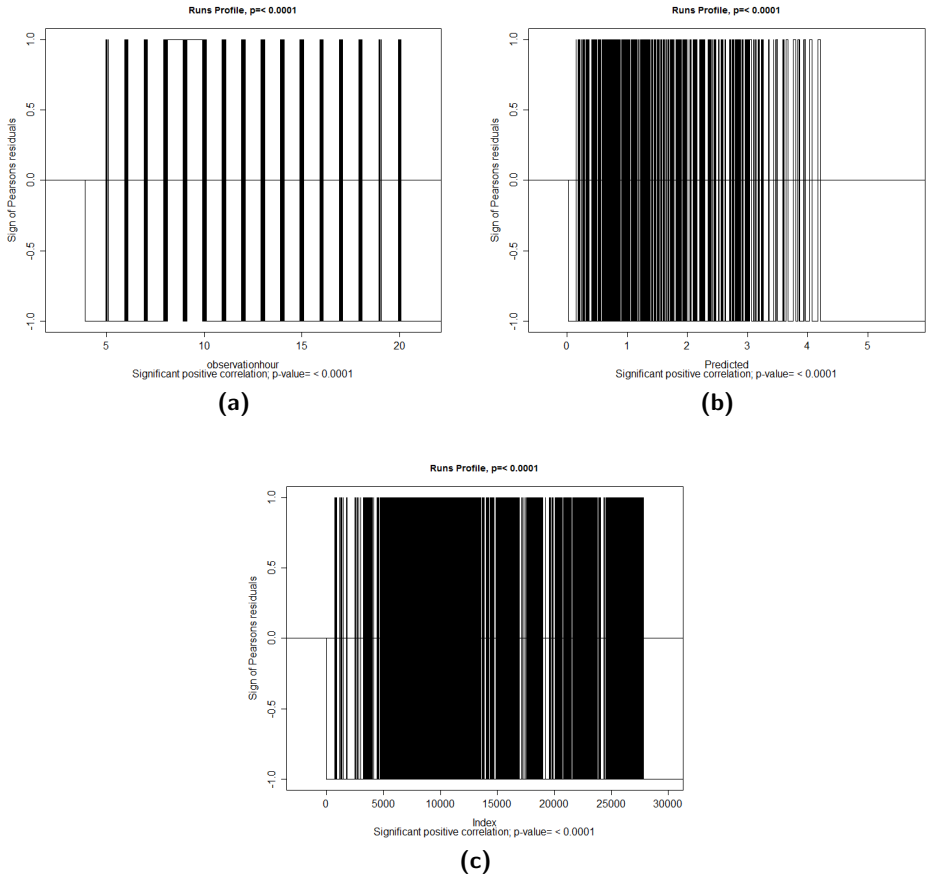


Figure 6: Runs profiles for residuals ordered by (a) observation hour, (b) predicted value and (c) temporally (by observation index). The p-values and text presented on each plot indicate if there is correlation present in the residuals. The lines are the strings of sequences of positive and negative residuals. A vertical line is the switch between a positive and negative run (or vice versa).

- Runs test and plots show an issue with correlated residuals
- We have positive correlation in the residuals no matter how they are ordered
- Conclusion:
  - We have correlation that must be accounted for

Choosing a blocking structure to model correlation.

- Correlation within blocks should decline to approximately zero
- Between blocks residuals should be independent
- Blocks are usually determined by looking at the sampling design
- Here we use the unique cell identifier within each day of observation: Each grid cell is considered independent but residuals may be non-independent within a block.
- There are 41 grid cells repeated several days a month, for 12 months over four years times, giving us 5576 blocks.

Autocorrelation function plot

We can check the correlation declines within our blocks by plotting the autocorrelation of the model residuals by block (Figure 7).

```
data$blockid <- paste(data$GridCode, data$Year, data$MonthOfYear,
    data$DayOfMonth, sep = "")
# **** blockid must be a factor or numeric, not a character ****
data$blockid <- as.factor(data$blockid)
 runACF(data$blockid, fullModel, store = F)
```

- Some blocks have little correlation, whilst others have high correlation (0.5) at a lag of 6.
- It is these blocks with the higher correlation that must be accounted for
- Conclusion: Our blocking structure is suitable

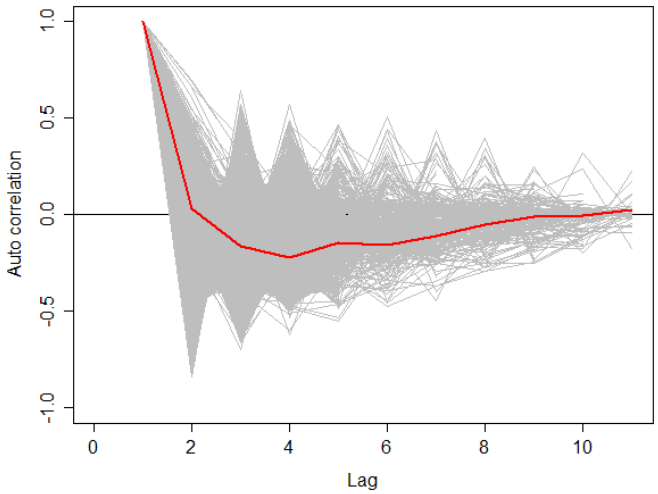


Figure 7: Plot of the correlation in residuals for each block (grey lines). The mean correlation at each lag is indicated in red.

4.3 Model Selection

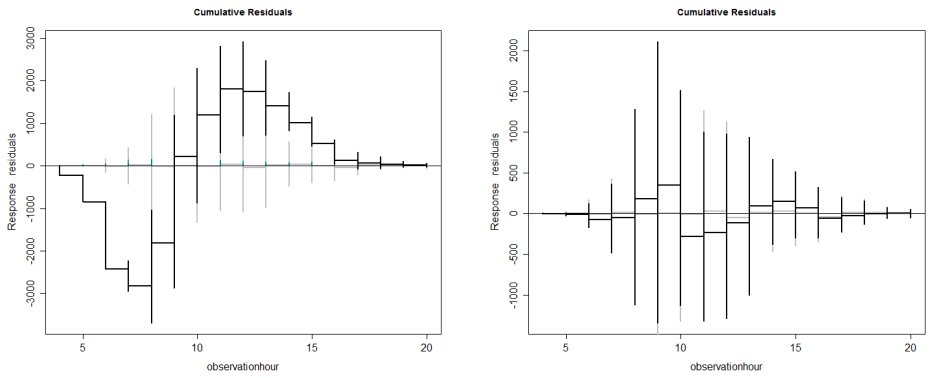
First we select what one-dimensional covariates we want in our model and use SALSA to determine the knot locations of those that are continuous - `observation hour`.

We can use cumulative residual plots to check for **appropriately modelled covariates**.

- These plots show systematic over- or under- prediction.
- We expect to see good mixing (lots of peaks and troughs) and deviation from this may indicate the covariate is not modelled appropriately.
- Figure 8 shows cumulative residual plots from a model with depth modelled as a linear term and with one knot at the mean.

Plotting Cumulative Residuals

```
# plotting cumulative residuals for the model with observationhour as a
# smooth term
plotCumRes(fullModel, varlist= c("observationhour"), splineParams)


"Calculating cumulative residuals"
```



(a) **(b)**

Figure 8: Cumulative residual plots residuals ordered by observation hour. (a) observation hour modelled as a linear term and (b) depth modelled with one knot at the mean observation hour. The blue points are the residual values, the black line represents the cumulative residuals. The grey line in the background is what we would expect the cumulative residuals to be if observation hour was modelled correctly.

Cumulative Residuals

- Observation hour as a linear term is not appropriate (Figure 8a).
- The black line (our model) does not correspond well with the expected line (grey) and shows systematic over prediction at early hours and under prediction in the early afternoon.
- Observation hour as a smooth term with one knot at the mean is much better but the black line may mask the expected line in places due to the discrete nature of the variable (Figure 8b).
- Conclusion:
  - we might want to consider more flexibility for `observationhour` by using SALSA.

Setting up the model for SALSA

- There must be a column called `response` in the data, which is the response variable used in the initial model to be fitted.
- The object `salsa1dlist` contains parameters for the `runSALSA1D` function.
  - `fitnessMeasure`. The criterion for selecting the 'best' model. Available options: AIC, AIC_c, BIC, QIC_b.
  - `minKnots_1d`. Minimum number of knots to be tried.
  - `maxKnots_1d`. Maximum number of knots to be tried.
  - `startKnots_1d`. Starting number of knots (spaced at quantiles of the data).
  - `degree`. The degree of the B-spline. Does not need to be specified if `splineParams` is a parameter in `runSALSA1D`.
  - `maxIterations`. The exchange/improve steps will terminate after maxIterations if still running.
  - `gaps`. The minimum gap between knots (in unit of measurement of explanatory).
- The initial model contains all the factor level covariates and any covariates of interest that are not specified in the `varlist` argument of `runSALSA1D`.

```
# run SALSA to find some good knot choices set some input info for SALSA
data$response <- data$birds

salsa1dlist <- list(fitnessMeasure = "QICb", minKnots_1d = 2,
    maxKnots_1d = 20, startKnots_1d = 2, degree = 2, maxIterations = 10,
    gaps = c(1))

# load prediction data
predictData <- read.csv("data/BC_eg_predictiongrid.csv")

# set initial model without the spline terms in there (so all other
# non-spline terms)
initialModel <- glm(response ~ as.factor(floodebb) + as.factor(impact) +
    offset(log(area)), family = "quasipoisson", data = data)
```

```
# run SALSA
salsa1dOutput <- runSALSA1D(initialModel, salsa1dlist, varlist=
    c("observationhour"), factorlist=c("floodebb", "impact"), predictData,
    splineParams=splineParams)
```

Let's look at the structure of the output:

- `bestModel`. The model object for the best fitted model.
- `modelFits1D`. Each slot in the list shows the term fitted, the fit statistic resulting from that term, the knots used and finally the overall formula. If `varlist` is more than one covariate, this output shows how each covariate was retained and what knots were finalised.
- `splineParams`. The spline parameter object is updated with the new knot numbers and locations of the covariates from `varlist`.
- `fitStat`. The fit statistic of the best model.

```
str(salsa1dOutput, max.level = 1)


List of 4
 $ bestModel   :List of 30
  ..- attr(*, "class")= chr [1:2] "glm" "lm"
 $ modelFits1D :List of 2
 $ splineParams:List of 2
 $ fitStat     : num 34341
```

```
# knots chosen for observation hour
salsa1dOutput$splineParams[[2]]$knots


[1]  9 15 17
```

- Initial model - one knot at the mean observation hour (12pm)
- SALSA result - three knots selected (see output above)
- The result of SALSA is additional flexibility added to the model for the relationship between bird counts and observation hour.

Spatial component

- Next we add a two dimensional smooth of geographic coordinates (s(`x.pos`, `y.pos`))
- To test for a redistribution of animals with an impact effect we fit an interaction term between the smooth of coordinates and `impact`.
- SALSA is used to determine spatially adaptive knot locations for this smooth term.

SALSA 2D requirements

- A grid of knot locations
- Matrix of knot to knot distances
- Matrix of data to knot distances
- Vector of range parameters for CReSS (determines the range of effectiveness of each knot basis). See (**?**) for more details.
- Minimum, maximum and starting number of knots

The next section assumes that the knot grid has been defined. See section **??** for some guidance to do this.

Spatial component

```
# load pre-made knotgrid (regular grid containing NA's for invalid knot
# locations (e.g. on land or outside study region))
data(knotgrid.ns)
knotgrid <- knotgrid.ns
```

The knot points in `knotgrid` are located at the centre of each of the grid cells containing data.

Next make the distance matrices that give the distance between the data and the knots in one matrix and the second matrix is the knot to knot distances. The function `makeDists` calculates Euclidean distance, however, the `runSALSA2D` function may also take geodesic distance matrices (as the fish swims rather than as the crow flies). See **?** for more details on geodesic distances.

```
# make distance matrices for datatoknots and knottoknots
distMats <- makeDists(cbind(data$x.pos, data$y.pos), na.omit(knotgrid))
str(distMats)


List of 2
 $ dataDist: num [1:27798, 1:41] 4000 4000 4000 4000 4000 4000 4000 ...
 $ knotDist: num [1:41, 1:41] 0 1000 2000 3000 4000 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:41] "3" "4" "5" "6" ...
  .. ..$ : chr [1:41] "3" "4" "5" "6" ...
```

A sequence of range parameters is required for the CReSS smooth.

- The range parameter determines the influence of each selected knot.
- Small numbers are for a local influence and large ones a global influence.
- Once knot locations are selected (using the mid value in the range sequence), SALSA selects the appropriate range parameter (from the sequence given) for each knot.

```
# choose sequence of radii
r_seq <- getRadiiChoices(8, distMats$dataDist)
```

Setting up the spatial SALSA components

- `fitnessMeasure`. The fitness measures available are the same as for `runSALSA1D`.

- `knotgrid`. $(k \times 2)$ matrix of knot coordinates. Rows of `NA`'s identify illegal knot locations
- `startKnots`. Number of space-filled knots to start with (between minKnots and maxKnots)
- `minKnots`. Minimum number of knots to fit
- `maxKnots`. Maximum number of knots to fit
- `r_seq`. Sequence of range parameters for the CReSS basis.
- `gap`. Minimum gap between knots (in unit of measurement of `x.pos` and `y.pos`)
- `interactionTerm`. Specifies which term in the model the spatial smooth will interact with. If NULL no interaction term is fitted.

```
# make parameter set for running salsa2d
salsa2dlist <- list(fitnessMeasure = "QICb", knotgrid = knotgrid,
    startKnots = 6, minKnots = 4, maxKnots = 20, r_seq = r_seq,
    gap = 1, interactionTerm = "as.factor(impact)")
```

The initial model is the best model from the one-dimensional SALSA results.

```
# splineParams must be an object in workspace
splineParams <- salsa1dOutput$splineParams
salsa2dOutput_k6 <- runSALSA2D(salsa1dOutput$bestModel, salsa2dlist,
    distMats$dataDist, distMats$knotDist, splineParams = splineParams)
```

Multiple SALSA runs

The example above uses 6 starting knot locations. There is a risk that the SALSA algorithm may get stuck in local minima or maxima and so we recommend that a variety of starting knot numbers are used. Here we try 6, 8, 10, 12, 14 and 16 (Table 2).

We use Cross-Validation (CV) as a method for selecting between models with a variety of starting knots.

k-fold Cross-Validation (CV)

Method for assessing model fit where the data is split into k sets. The model is fitted to k_{-i} sets and predictions are made to the k_ith set. Mean Squared Error (MSE) is calculated for each of the k_i prediction sets and a mean taken to get the CV score.

Example:

```
data$foldid <- getCVids(data, folds = 5, block = "blockid")
```

```
cv1 <- getCV_CReSS(data, salsa1dOutput$bestModel, salsa1dOutput$splineParams)

[1] 33.91
```

Choosing a model

Table 1: Table of CV scores for a variety of starting knot numbers for the spatial smooth.

Model type	Start knots	End knots	CV
1D terms only	-	-	33.9058
1D/2D terms	6	6	28.6426
1D/2D terms	8	8	28.1481
1D/2D terms	10	10	27.9571
1D/2D terms	12	12	27.0373
1D/2D terms	14	14	27.0474
1D/2D terms	16	16	27.0367

The best model chosen using CV score is the model that uses a spatial smooth with 16 spatial knots.

```
# having chosen the 2d interaction model, save the model object
baseModel <- salsa2dOutput_k16$bestModel
# update spline parameter object
splineParams <- salsa2dOutput_k16$splineParams
```

4.4 Checking p-values

Re-assessing runs test for best model

- Our best model based on information criterion selection is the model with the interaction term.
- We could also use p-value selection, though the model must be fitted as a GEE to model the correlation.

We must account for the correlation in our residuals, which we found earlier but should also check the current model.

```
runs.test(residuals(baseModel, type = "pearson"))


Runs Test - Two sided
data:  residuals(baseModel, type = "pearson")
Standardized Runs Statistic = -88.67, p-value < 2.2e-16
```

GEE framework

There is significant positive residual correlation ($p << 0.05$ and test statistic is negative) so we re-fit the model as a GEE. Note: It is possible to do this at this stage as we are modelling correlation using empirical standard errors and not a specific correlation structure.

The current model:

```
glm(formula = response ~ as.factor(floodebb) + as.factor(impact) +
    bs(observationhour, knots = splineParams[[2]]$knots, degree =
    splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd) +
    LocalRadialFunction(radiusIndices, dists, radii, aR) +
    as.factor(impact):LocalRadialFunction(radiusIndices, dists, radii, aR) +
    offset(log(area)), family = quasipoisson, data = data)
```

```
# N.B. for the GEE formula, the data must be ordered by block (which this is)
# and the blockid must be numeric
# specify parameters for local radial:
radiusIndices <- splineParams[[1]]$radiusIndices
dists <- splineParams[[1]]$dist
radii <- splineParams[[1]]$radii
aR <- splineParams[[1]]$invInd[splineParams[[1]]$knotPos]
baseModel <- update(baseModel, . ~ .)

# Re-fit the chosen model as a GEE (based on SALSA knot
# placement) and GEE p-values
geeModel <- geeglm(formula(baseModel), data = data, family = poisson,
    id = blockid)
```

Checking p-values

```
# table of p-values (specifying varlist and factorlist
# makes shorter variable names)
getPvalues(geeModel, varlist = c("observationhour"), factorlist = c("floodebb",
    "impact"))
```

```
[1] "Getting marginal p-values"
                Variable  p-value
1                floodebb  <0.0001
2                  impact 0.5616
3         observationhour  <0.0001
4         s(x.pos, y.pos)  <0.0001
5 s(x.pos, y.pos):impact 0.0240
```

Do we remove the main `impact` effect?

Either:

- Keep the main effect for `impact` since it is part of the interaction term and it is difficult to really interpret what this p-value actually means.

  or

- Remove the main effect for impact as it is not significant ($p >> 0.05$).

Here we chose the first option but below is an example of how to remove the `impact` term using the `update` function.

Removing the main `impact` effect

```
# how to remove impact
model <- update(geeModel, . ~ . - as.factor(impact))
# reshow p-values
getPvalues(model, varlist = c("observationhour"), factorlist = c("floodebb",
    "impact"))
```

```
[1] "Getting marginal p-values"
                Variable  p-value
1                floodebb  <0.0001
2         observationhour  <0.0001
3         s(x.pos, y.pos)  <0.0001
4 s(x.pos, y.pos):impact 0.001119
```

4.5 Assessing covariate relationships

Partial Plots

Visually examine the one-dimensional covariates in the model (observation hour, tide state, impact). Check smooth/linear terms are specified correctly.

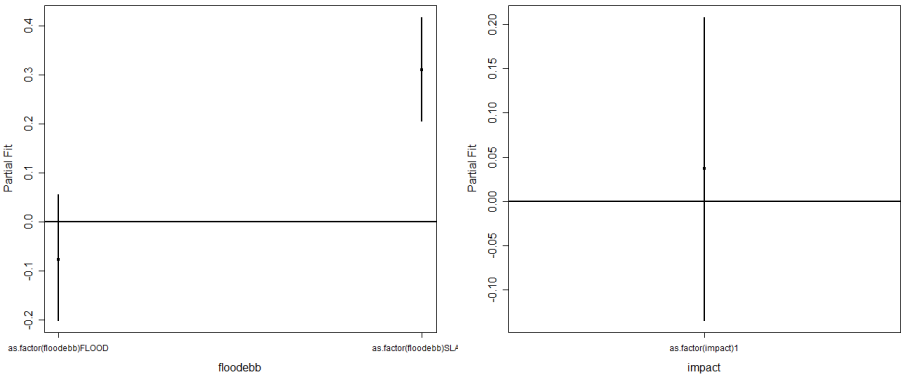
```
runPartialPlots(geeModel, data, factorlist = c("floodebb", "impact"),
    varlist = c("observationhour"))


[1] "Making partial plots"
pdf
  2
```

Assessing covariate relationships

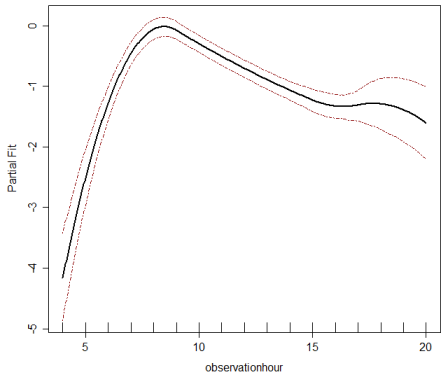
Partial Plots

- Tide state: predictions for flood are not significantly different to ebb (baseline)
- There are, on average, more animals seen during slack tide than for the baseline (ebb).
- Impact: confirms the ANOVA p-value (seen earlier; $p >> 0.05$) and indicates no change in bird numbers post impact.
- Observation hour: a peaked non-linear relationship with maximum birds seen between 8 and 11am.
- Small confidence interval indicates a precisely estimated relationship.

Note: if the confidence interval for observation hour was very wide it might indicate that observation hour as a linear term is more appropriate.



(a) **(b)**



(c)

Figure 9: Partial plots for (a) tide state, (b) impact and (c) observation hour.

5 Diagnostics

Diagnostics

We make multiple plots to assess the fit of our model:

- Observed vs Fitted
- Fitted vs residuals
- Cumulative residuals
- Runs sequence
- COVRATIO statistics
- PRESS statistics
- Raw residuals

Observed vs fitted Plot

Observed vs Fitted

Indication of fit to the data. All the values would be on the 45^o line for a perfectly fitting model.

The agreement between the input data and the model undergoing evaluation can be quantified using numerical measures; Marginal R-squared and Concordance Correlation.

These measure are output on the observed vs fitted plot created using the `runDiagnostics` function.

Marginal R-squared value

It is widely used to assess models fitted to both un-correlated and correlated data and is approximately between 0 and 1. Values closer to 1 indicate better fit to the data.

Concordance Correlation

It is guaranteed to return values between zero and one. Values closer to 1 indicate better fit to the data.

```
# create observed vs fitted and fitted vs residual plots
runDiagnostics(geeModel)


[1] "Assessing predictive power"
```

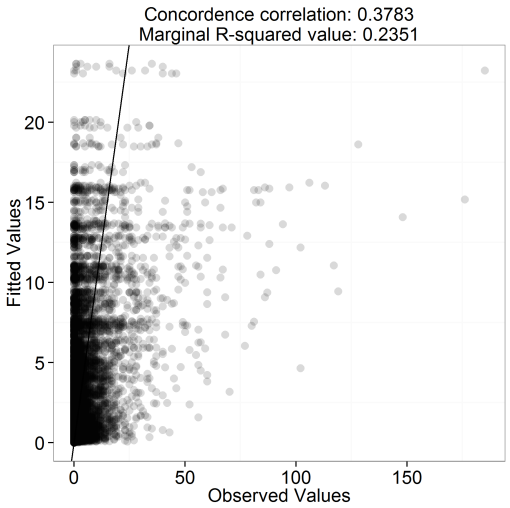


Figure 10: Diagnostic plots of observed vs fitted values, where the diagonal line indicates where data should lie for a perfect fit.

Observed vs fitted

- High observed values are under-predicted (Figure 10)
- Observed zeros tend to be over-predicted
- Marginal R-squared and concordance correlation are low
- Conclusion: poor model fit

Fitted values vs scaled Pearsons residuals Plot

The plot gives an indication of the correct mean-variance relationship assumed under the model. We expect to see no pattern in the plot.

- For a Poisson model the size of residuals is expected to increase with increasing fitted values

- For overdispersed data the size of residuals increases at a faster rate than the strict Poisson relationship
- To get a plot with an expectation of no pattern we use Pearsons residuals to account for the former and scaled these by the dispersion parameter for the latter
- Thus we plot Fitted values against scaled Pearsons residuals

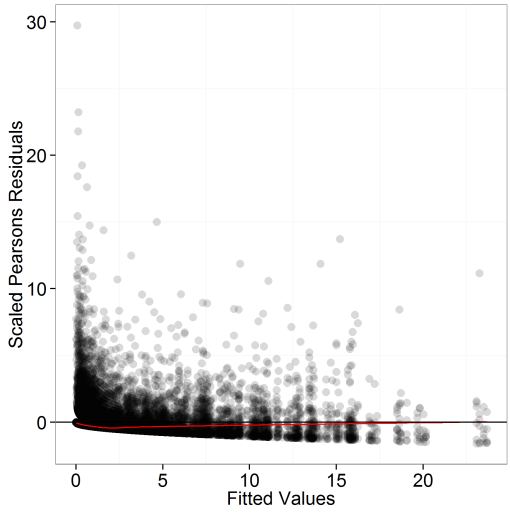


Figure 11: Diagnostic plot of fitted values vs scaled Pearsons residuals, where the red line is a locally weighted least squares regression to indicate pattern in the plot, which might otherwise be hidden due to over-plotting.

- Possible pattern in residuals but hard to tell due to overplotting (Figure 11)
- Locally weighted least squares regression line does not indicate an unusual pattern
- Conclusion: no issue with model assumption (mean-variance relationship)

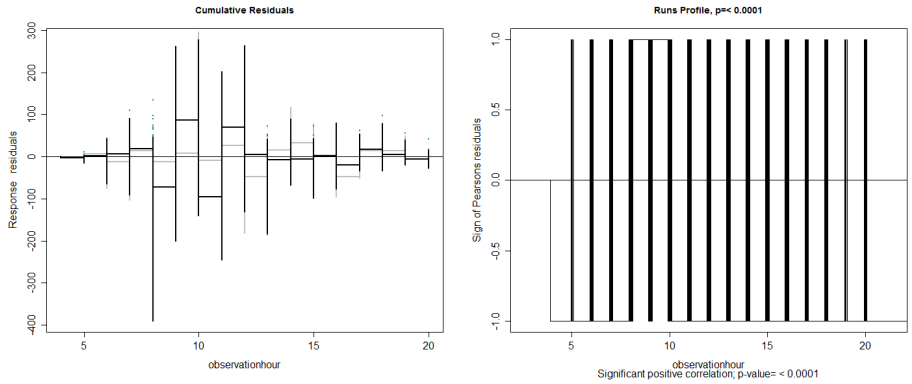
5.1 Cumulative Residuals and runs profiles

- Assessment of systematic over- or under- prediction using cumulative residuals.

- Assessment of the correlated nature of the residuals (how random they are) given that the residuals are ordered by covariate value, predicted value or temporally.

```
plotCumRes(geeModel, varlist=c('observationhour'), splineParams=splineParams,
    d2k=dists)
plotRunsProfile(geeModel, varlist=c('observationhour'))
```

Cumulative residuals and runs profiles for observation hour



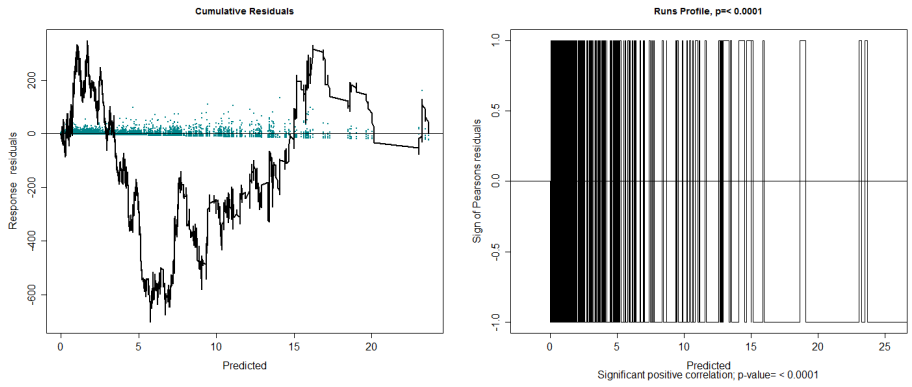
(a) **(b)**

Figure 12: Cumulative residual plot (a) and runs profile (b) for residuals ordered by observation hour. The blue points are the residual values, the black line represents the cumulative residuals. The grey line in the background is what we would expect the cumulative residuals to be if observation hour was modelled correctly.

Ordered by **observation hour**

- No systematic over or under prediction (Figure 12)
- Variable modelled appropriately:
  - similar to expected relationship (grey line) and better than Figure 8
  - good mixing about the zero cumulative residual line
- Discrete nature of variable makes inference (visual) difficult
- Visually good mixing, but fewer runs than would be expected if residuals were random ($p << 0.05$)
- Significant positive correlation between residuals when ordered by observation hour

Cumulative residuals and runs profiles ordered by predicted value



(a) **(b)**

Figure 13: Cumulative residual plot (a) and runs profile (b) for residuals ordered by the predicted value. The blue points are the residual values and the black line represents the cumulative residuals.

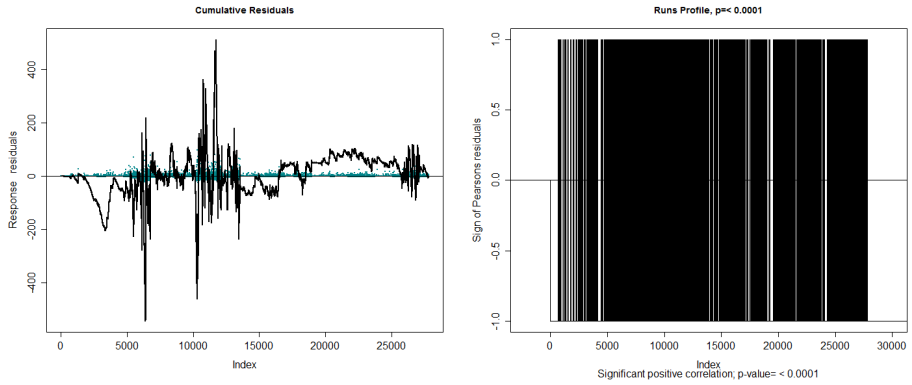
Ordered by **prediction** value

- Under prediction at predicted density < 3 (Figure 13)
- Over-predicted up to about 12 birds/km^2
- Thereafter some over/under prediction
- But, fewer runs than would be expected if residuals were random ($p < 0.05$)
- Significant positive correlation between residuals when ordered by predicted value

Cumulative residuals and runs profiles ordered temporally

Ordered by **index** (temporally)

- Lot of mixing about zero cumulative residual line (Figure 14)
- Best mixing of residuals seen in the runs profile
- But, still fewer runs than would be expected if residuals were random ($p < 0.05$)
- Significant positive correlation between residuals when ordered temporally



(a) **(b)**

Figure 14: Cumulative residual plot (a) and runs profile (b) for residuals ordered by the index of observations (temporally). The blue points are the residual values and the black line represents the cumulative residuals.

Cumulative residuals and Runs Profiles

- Despite the inclusion of temporal covariates (season and impact) we still have some unmodelled correlation
- We model this correlation using GEEs
- p-values from ANOVA (fitted earlier to decide which covariates to keep) are reliable due to modelled correlation

5.2 COVRATIO and PRESS statistics

The PRESS and COVRATIO statistics are relative measures that assess how aspects of the model change when individual blocks are removed from the analysis.

COVRATIO statistic

Signals the change in the **precision of the parameter estimates** when each block is omitted.

- Values greater than one signal removing the block inflates model standard errors
- values less than one signal standard errors are smaller when that block is excluded

COVRATIO and PRESS statistics

PRESS statistic

Quantifies the sensitivity of **model predictions** to removing each block.

- Relatively large values signal the model is sensitive to these subjects.
- Model coefficients are re-estimated when each block is omitted (one-by one) and the sum of the squared differences between the response data and the predicted values (when that subject is removed) are found.

If model predictions or measures of precision appear particularly sensitive to omitted blocks, examine model conclusions based on models with and without the potentially problematic blocks.

```
timeInfluenceCheck(geeModel, data$blockid, dists, splineParams)
# influence plots (covratio and press statistics)
influence <- runinfluence(geeModel, data$blockid, dists, splineParams)
```

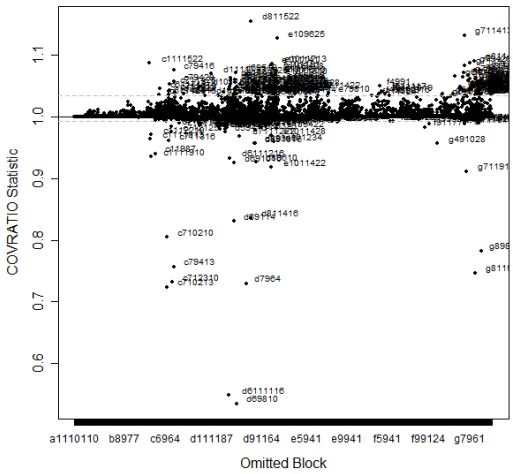


Figure 15: Plot of COVRATIO statistics; the dashed grey lines indicate the lower 2.5% and upper 97.5% quantiles of the statistics.

COVRATIO statistics

- there will always be blocks outside the dashed lines (they are quantiles)

- As it is a relative measure, blocks far away may be of concern
- Blocks with COVRATIO statistic < 1 are of most concern (decrease in precision when removed)
- Here, blocks `d69810` and `d6111116` are far away and result in an decrease in standard errors when removed (Figure 15)
- These blocks are both grid code `D6`, and the former in year 2009 (first of two years before impact), month 8 and the 10^{th} day of the month.
- Both blocks have particularly high bird counts compared with most blocks and so adds to the variability in the data.

- Conclusion: Variability increases with higher counts for Poisson data and so it is fine for these blocks to remain.

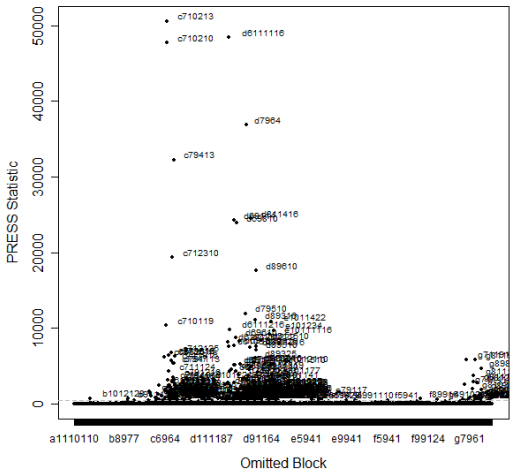


Figure 16: Plot of PRESS statistics; 95% of the statistics fall below the dashed grey lines. Labelled points on both plots are outside the grey dashed line(s) and are labelled with the identifier for the block that has been removed to create the statistic (not an observation number).

PRESS statistics

- there will always be blocks above the dashed line (it is a quantile)
- As it is a relative measure, blocks far away may be of concern

- Predictions are sensitive to several blocks: c710213 c710210 and d6111116 (Figure 16).
- As above, these blocks all contain very high bird counts
- Conclusion: Variability increases with higher counts for Poisson data and so it is fine for these blocks to remain.

5.3 Raw Residuals

Raw residuals (fitted - observed values) can be represented spatially to ascertain if some areas are over/under predicted.

- No real spatial pattern to the residuals (Figure 17)
- No systematic over/under prediction
- Conclusion: No spatial bias in model

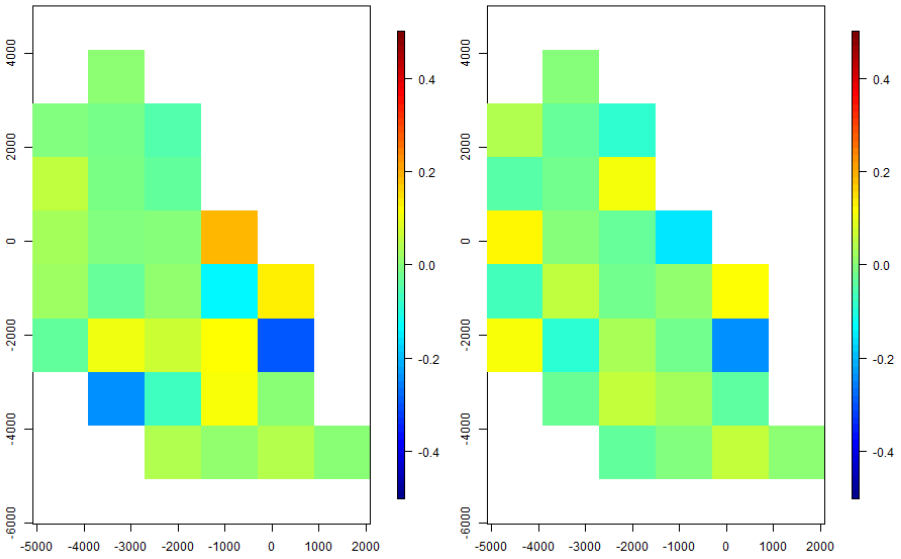


Figure 17: Raw residuals before impact (left) and after impact (right). These residuals are fitted values - observed values (mean* birds/km^2). * mean density because the predictions are for several observation hours.

```
# residual plot
resids <- fitted(geeModel) - data$birds
dims <- getPlotdimensions(data$x.pos, data$y.pos, 1000, 1000)

par(mfrow = c(1, 2), mar = c(3, 3, 3, 5))
quilt.plot(data$x.pos[data$impact == 0], data$y.pos[data$impact ==
    0], resids[data$impact == 0], asp = 1, ncol = dims[2], nrow = dims[1],
    zlim = c(-2.2, 2.2))
quilt.plot(data$x.pos[data$impact == 1], data$y.pos[data$impact ==
    1], resids[data$impact == 1], asp = 1, ncol = dims[2], nrow = dims[1],
    zlim = c(-2.2, 2.2))
```

How did we do?

Table 2: Table of potential modelling problems, what method was used to assess the problem, was the potential problem an issue in reality and if yes then what was the solution. Solutions in red were not done and are possibilities for the future.

	Method	Issue (Y/N)	Solution
Collinearity	VIF	N	-
Over-dispersion	-	Y	estimated by GEE
Model Fit	Observed vs Fitted	Y	use more covariates
mean-variance relationship	Fitted vs residuals	N	-
Correlated Residuals	runs test	Y	modelled by GEE
Correlated Residuals	Runs profile	Y	modelled by GEE
Covariate specification	cumulative residuals	N	-
systematic over/under prediction	cumulative residuals	N	-
spatial systematic over/under prediction	raw residual plot	N	-
removing blocks	COVRATIO statistics	N	-
removing blocks	PRESS statistics	N	-

6 Prediction and Inference

6.1 Data requirements for predicting

For making predictions, we require a data frame containing grid data with records for which predictions are desired. In terms of columns and records these can be summarised as:

- **Columns:** each covariate retained in the best fitting model with exactly matching column names
- **Columns:** *x.pos* and *y.pos* provide geographic position information used for calculating distance matrices for CReSS/SALSA

- **Columns:** *area* provides the area of each gridcell.
- **Rows:** 1 record for each gridcell and for each date and time a prediction is made for

Using our example of nearshore data, we have a look at our prediction data using the *head* function after we load the data.

```
# loading the prediction grid data
data(ns.predict.data.re)
predictData <- ns.predict.data.re
head(predictData)
```

```
  x.pos y.pos     area floodebb observationhour GridCode Year DayOfMonth
1  1500 -4500 0.385253      EBB              12      a11    9         13
2  1500 -4500 0.385253    FLOOD               8      a11    9         16
3  1500 -4500 0.385253    FLOOD               9      a11    9         16
4  1500 -4500 0.385253    FLOOD              10      a11    9         16
5  1500 -4500 0.385253    FLOOD              11      a11    9         16
6  1500 -4500 0.385253    FLOOD              12      a11    9         16
  MonthOfYear impact        truth
1           3      0 0.0003871928
2           3      0 0.0006163433
3           3      0 0.0006064698
4           3      0 0.0005072918
5           3      0 0.0004220902
6           3      0 0.0003570533
```

6.2 Making predictions

We are now going to use these data to make predictions using our best fitting CReSS/SALSA model from the previous sections. This requires three steps:

1. Creating distances for the prediction grid
2. Making predictions to the prediction data using the best fitting model
3. Converting the predictions back to the response scale

```
# create the distance matrix for predictions
dists <- makeDists(cbind(predictData$x.pos, predictData$y.pos),
    na.omit(knotgrid), knotmat = FALSE)$dataDist
# use baseModel to make predictions to avoid a warning from
# using geeModel (same answers though)
predslink <- predict(baseModel, predictData, type = "link")
# reversing the log-link to convert predictions back to the response scale
preds <- exp(predslink)
```

The object `preds` contains the predictions for each record in our prediction grid data.

6.3 Visualising the redistribution using predictions

We know that our model identified a redistribution of animals within the study area by the significant interaction term between impact and the two-dimensional smooth of `x.pos` and `y.pos`. However, from the model coefficients it is not obvious from where to where they redistributed. Hence, we are going to investigate this by visualising the redistribution using our predictions.

```
# plotting the predictions for before and after impact
par(mfrow=c(1,2), mar=c(3,3,3,5))
quilt.plot(predictData$x.pos[predictData$impact==0],
predictData$y.pos[predictData$impact==0],
preds[predictData$impact==0], asp=1, nrow=7, ncol=9,
zlim=c(0, maxlim))
quilt.plot(predictData$x.pos[predictData$impact==1],
predictData$y.pos[predictData$impact==1], preds[predictData$impact==1],
asp=1,nrow=7, ncol=9, zlim=c(0, maxlim))
```

Note that the value for `maxlim` can be determined by plotting both plots without the `zlim` argument first from which you can determine what the maximum value in the scale of either plot is.

The quilt.plot function

- Plots averages of the predictions for each grid cell
- Subsetting the data using the square brackets restricts the input data to before (left) and after impact (right plot)
- Using the `zlim` argument allows ensuring that left and right plot have the same scale colour scheme.

Conclusion

- Decline in bird density in the central eastern region
- Increase in density in the south of the study region
- The central region is where a known impact has taken place and the results suggest that birds have moved from this region to the south after the impact event

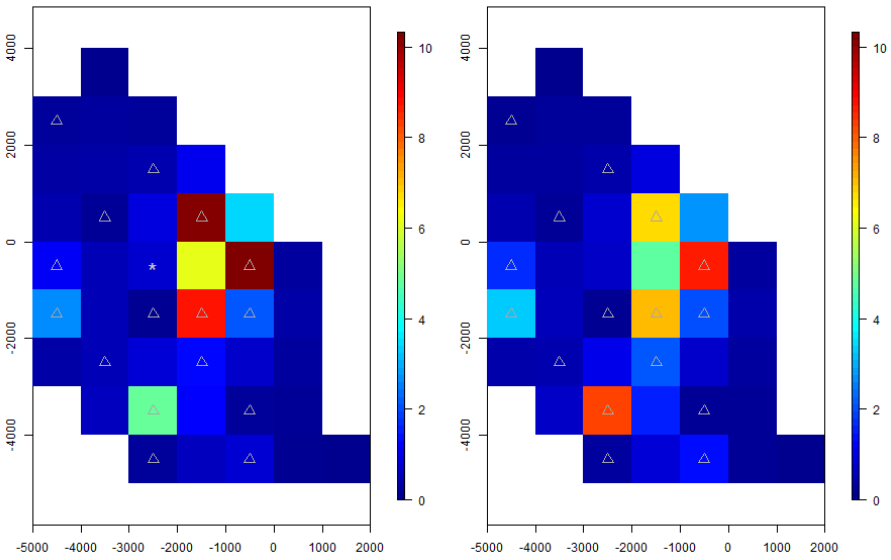


Figure 18: Predictions of bird density (birds/km^2) from the fitted model for before (left) and after (right) an impact event. The * indicated the location of the impact; the △ indicate the knot locations.

Next step

- Is this difference real or simply random noise?
- We need to classify if the differences are significant or due to sampling variation.
- We do this by constructing bootstrap confidence intervals for each prediction in the prediction grid data.

6.4 Bootstrap confidence intervals for near-shore data

Following the predictions from section 6.2, we now create a set of bootstrap predictions and use these to construct percentile confidence intervals around each prediction from our prediction grid data. We run `B` iterations where for each iteration the following steps are completed:

- Input best model from fitting to the original observed data.
- Use the model coefficients and covariance matrix to sample new coefficients from a multivariate Normal distribution.

- Make predictions to the study area using these coefficients.

This approach incorporates the uncertainty associated with the count model coefficients.

```
# do the bootstrap
do.bootstrap.cress(data, predictData, ddf.obj=NULL, baseModel, splineParams,
                dists, B=250)
```

The `do.bootstrap.cress` function

This function performs the number of bootstrap iterations specified with the argument `B` where the fitting model is CReSS. The recommended number of bootstrap iterations is 999.
If `ddf.obj` is set to `NULL`, this function automatically applies the 'nearshore'-approach with the three steps described above.
Since the prediction object can be quite large, it is not returned but saved into the current working directory. In the next step we use these bootstrap predictions to construct our 95% confidence intervals for each prediction.

```
# read in bootstrap predictions
load("predictionboot.RData")
# make percentile confidence intervals
cis <- makeBootCIs(bootPreds)
```

The `makeBootCIs` function

Using the `B` bootstrap predictions from a call to the `do.bootstrap.cress` function, the `makeBootCIs` function creates lower and upper limits for 95% confidence intervals using the percentile method for each record in the prediction data.

6.5 Visualising bootstrap confidence intervals

In this section we will investigate whether looking at the lower and upper confidence limits from the 95% confidence intervals will give us a similar picture as looking at the predictions.

```
# plotting the confidence intervals for before and after impact
par(mfrow=c(2,2), mar=c(3,3,3,5))
quilt.plot(predictData$x.pos[predictData$impact==0],
predictData$y.pos[predictData$impact==0],
cis[predictData$impact==0,2], asp=1, nrow=7, ncol=9,
zlim=c(0, maxlim))
quilt.plot(predictData$x.pos[predictData$impact==1],
predictData$y.pos[predictData$impact==1], cis[predictData$impact==1,2],
asp=1,nrow=7, ncol=9, zlim=c(0, maxlim))
quilt.plot(predictData$x.pos[predictData$impact==0],
predictData$y.pos[predictData$impact==0],
cis[predictData$impact==0,1], asp=1, nrow=7, ncol=9,
zlim=c(0, maxlim))
quilt.plot(predictData$x.pos[predictData$impact==1],
predictData$y.pos[predictData$impact==1], cis[predictData$impact==1,1],
asp=1,nrow=7, ncol=9, zlim=c(0, maxlim))
```

Conclusion

- Lower and upper intervals before impact (left) show a higher density of birds in the central eastern region.
- After impact (right), both lower and upper intervals show a decline in density in the central eastern region and an increase in the area to the south.
- It is difficult to see where in the study region any significant differences may have occurred.

Next step

- To see where the redistribution may have occurred we proceed to the next step where we calculate the differences in densities before and after impact for each corresponding pair of predictions.

6.6 Significant differences

In this section we will estimate the differences in predictions for each corresponding pair of records from our prediction grid data. The first half of

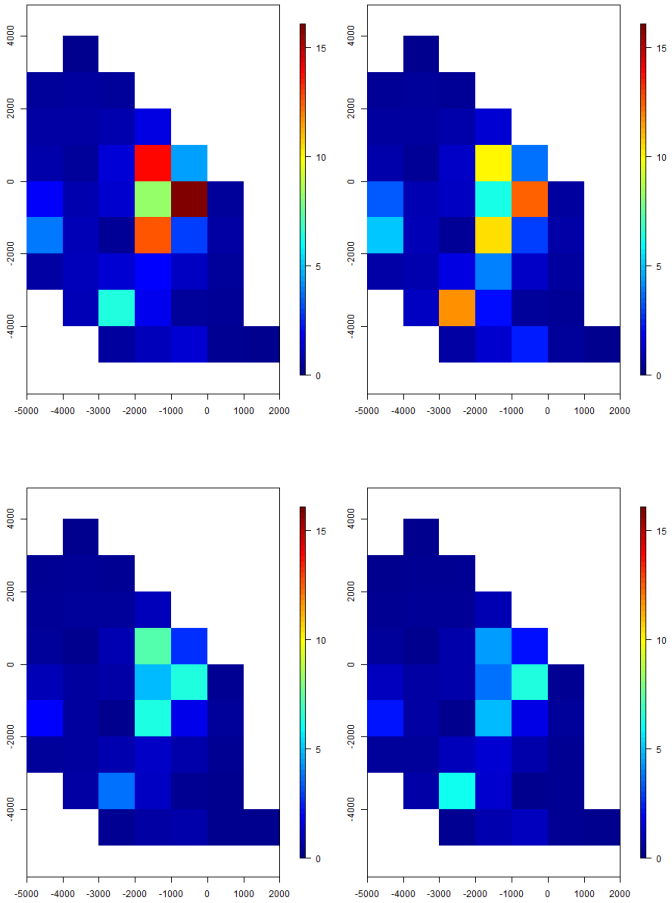


Figure 19: Upper (top) and lower (bottom) 95% confidence intervals of bird density (birds/km^2) from the fitted model from before (left) and after (right) impact.

our prediction grid data consists of records from before impact while the second half consists of records from after impact.
Note that the before and after data need to be ordered in the same manner and of the same length.

```
differences <- getDifferences(beforePreds =
        bootPreds[predictData$impact == 0, ],
        afterPreds = bootPreds[predictData$impact == 1, ])
```

The `getDifferences` function

For each bootstrap iteration the differences in predictions from before and after impact are calculated (Density$_{after}$ - Density$_{before}$) and 95% confidence intervals for the difference calculated using the percentile method. If these intervals contain the value zero, a '0' is recorded. If they do not

contain the value zero, '1' is recorded if the lower limit is above zero as an indication that the difference is significantly positive (increase in animal densities after impact). A '-1' is recorded if the upper limit is below zero, indicating that the difference is significantly negative (decrease in animal densities after impact).
The function returns a list of two objects:

- The median for each difference
- The marker for significant positive and negative differences

6.7 Visualising significant differences

To illustrate the differences in animal densities after impact, we plot the calculated differences (Density$_{after}$ - Density$_{before}$) using the `quilt.plot` function. The locations of the significant differences are added to the same plot using the '$\circ$' and '$+$' symbols.

```
# The median for each after - before difference
meandiff <- differences$meandiff
# The marker for each after - before difference:
positive ('1') and negative ('-1') significant differences
marker <- differences$significanceMarker
par(mfrow = c(1, 1))
quilt.plot(predictData$x.pos[predictData$impact == 0],
predictData$y.pos[predictData$impact ==  0],
meandiff, asp = 1, nrow = 7, ncol = 9)
# add + or - depending on significance of cells.  Just
# requires one significance out of all to be allocated
points(predictData$x.pos[predictData$impact==0][marker==1],
predictData$y.pos[predictData$impact==0][marker==1], pch="+", col="darkgrey",
cex=2)
points(predictData$x.pos[predictData$impact==0][marker==(-1)],
predictData$y.pos[predictData$impact==0][marker==(-1)], col="darkgrey", cex=3)
points(0,0,pch=20, col="grey",, cex=3)
points(data$x.pos[which(data$GridCode=='e7')[1]],
data$y.pos[which(data$GridCode=='e7')[1]],cex=2, pch="*", lwd=2, col="grey")
```

Conclusion

- There was a significant decline in animals around the impact site and closest to the cliff-top observer.
- There was also a significant increase in animals in the South of the study area.

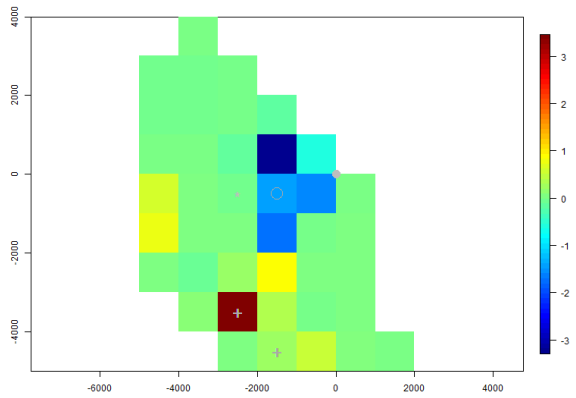


Figure 20: Mean differences in predicted bird density (birds/km^2) before and after impact. Positive values indicate more birds post impact and negative values fewer birds post impact. Significant differences were calculated using percentile confidence intervals: '+' indicates a significant positive difference and '○' a significant negative one. The grey star is the site of the impact event.

7 Comparison to the Truth

Here we will finalise our analysis by comparing our results with the truth. This is possible as our data was simulated and hence the parameter values known.

7.1 Overdispersion and correlation

We created data that were both overdispersed and positively correlated within any grid cell-day (i.e. observed counts from the same grid cell within the same day).

Truth	**Model**
Overdispersed data	estimated overdispersion parameter was greater than 1
Positively correlated data	accounted for using GEE-based *p*-values

Conclusion

We conclude that our proposed methods are capable of identifying overdispersion and correlation in the data.

7.2 Type of impact

For this particular data set, the total number of animals before and after impact did not change. The type of impact that we implemented was a redistribution from the area surrounding the impact into the south of the study area.

Truth	**Our model**
Redistribution within study area	identified with a significant interaction term
Overall abundance remained the same	identified with the non-significant main effect for impact

Conclusion

We conclude that our model was capable of correctly identifying the redistribution effect and removed and reallocated birds to the correct locations despite highly correlated and overdispersed data.