Lecture 18. GAMs II

A GAM example with a spatially distributed response

Now I'm going to go through a long example of using a GAM to analyze spatially extensive data. The data are originally from Borchers et al. 1997, "Improving the precision of the daily egg production method using generalized additive models", *Can. J. Fish. Aquat. Sci*. These are survey data on mackerel egg counts from the Northeast Atlantic, and the goal of the authors was to improve the precision of the count estimates, so that those counts could be used with the daily egg production method (DEPM) to estimate the size of spawning stocks. Essentially, the amount of eggs in the water is combined with data on the egg production rate by adults, in order to back-calculate the biomass of adults based on egg counts.

We will use this dataset to learn about several features of GAMs: using multiple smoothers; using two-dimensional smoothers; using GAMs to account for spatial non-independence of data; plotting model predictions and fitted effects in a spatial context. The dataset is available in the 'gamair' package, which is a companion to mgcv and the mgcv book by Simon Wood.

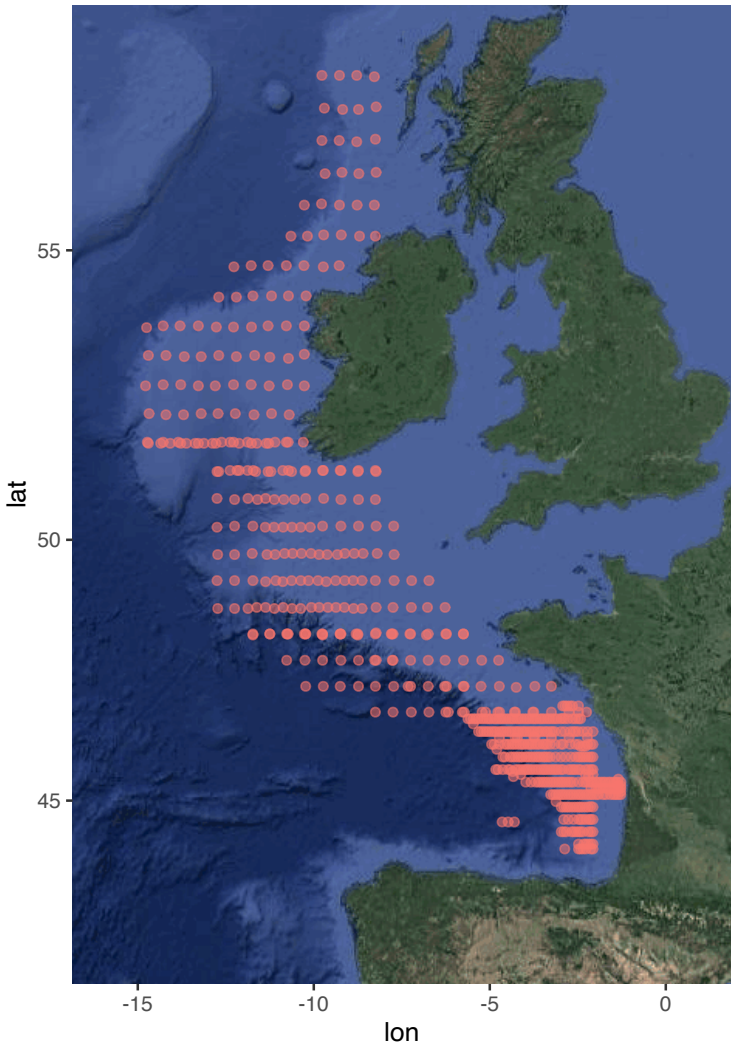
```
library(gamair)
```

```
library(ggmap)

library(GGally)
data(mack)
data(coast)

mackmap <- get_map(location = c(mean(mack$lon), mean(mack$lat)), zoom=
4, maptype = "satellite")

ggmap(mackmap) + geom_point(data = mack, aes(lon, lat, colour = 'blue',
 alpha = 0.5), show.legend = FALSE) + xlim(c(-16,1)) + ylim(c(42,58))
```



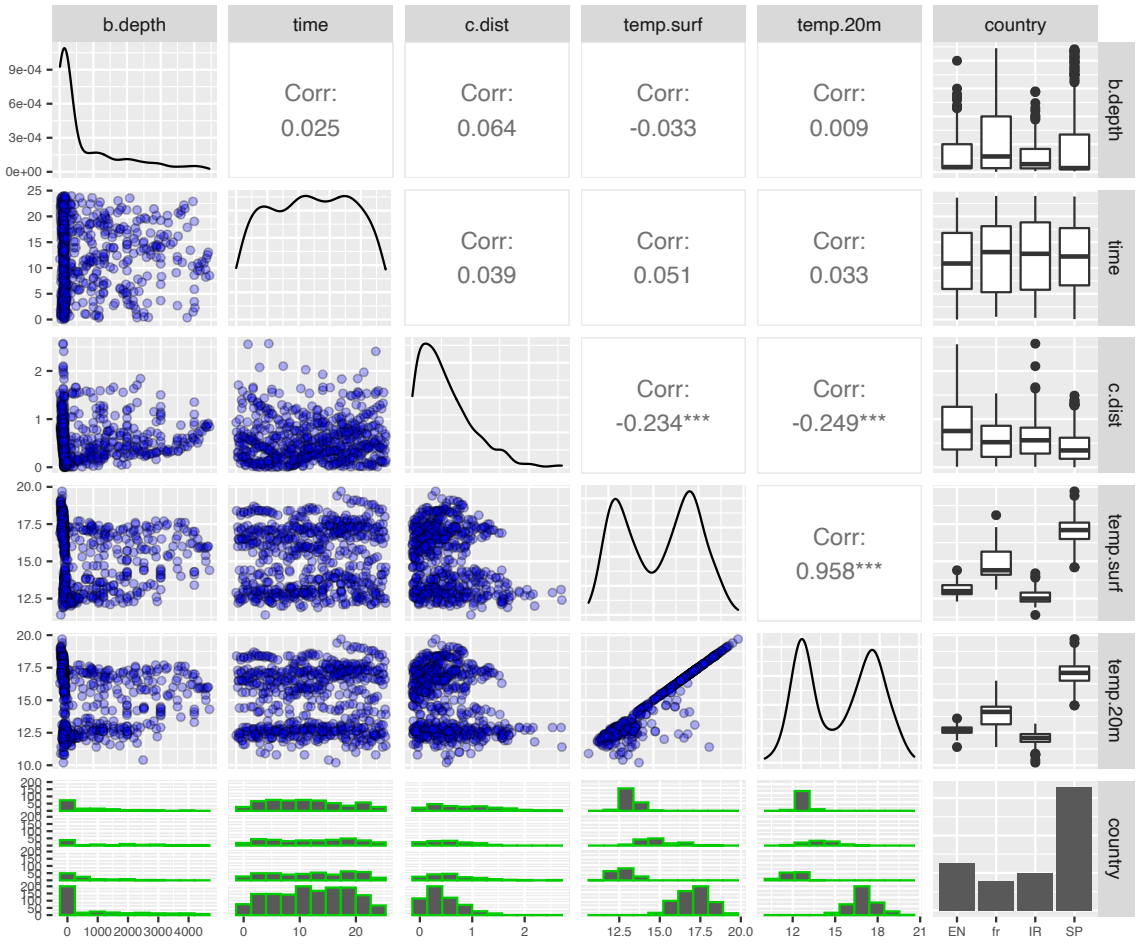
Here I've just plotted the location of each sample based on lat/lon, overlaid on a google maps satellite image downloaded with ggmap(). In general there are many spatial/GIS packages for R that will allow you to easily draw maps. For this plot we can see that the sampling is spatially extensive, with greater sampling effort off the coast off France. When analyzing this kind of data, it is important to evaluate potential spatial dependence in the data. For example, if we are interested in seeing what environmental predictors explain variation in egg density across this region, we would need to check whether the residual variation has spatial structure. It is likely that a pair of samples from sites that are close to each other will be more similar to each other than a pair of samples from sites that are further away. This is called *spatial autocorrelation*. A main reason this will occur is because of unmeasured environmental factors that affect the response variable. Spatial autocorrelation means that the data are not independently distributed, and this can greatly reduce confidence intervals and p-values, making spurious relationships more likely.

We will return to the issue of correlated residuals later in the course (in addition to spatial autocorrelation, time series can have temporal autocorrelation), but I introduce the topic here because GAMs are one way to account for spatial structure. Specifically, we can just fit a two-dimensional smoother where the predictor variables are latitude and longitude. This will capture any smooth variation in egg count across space that is not explained by other predictors in the model.

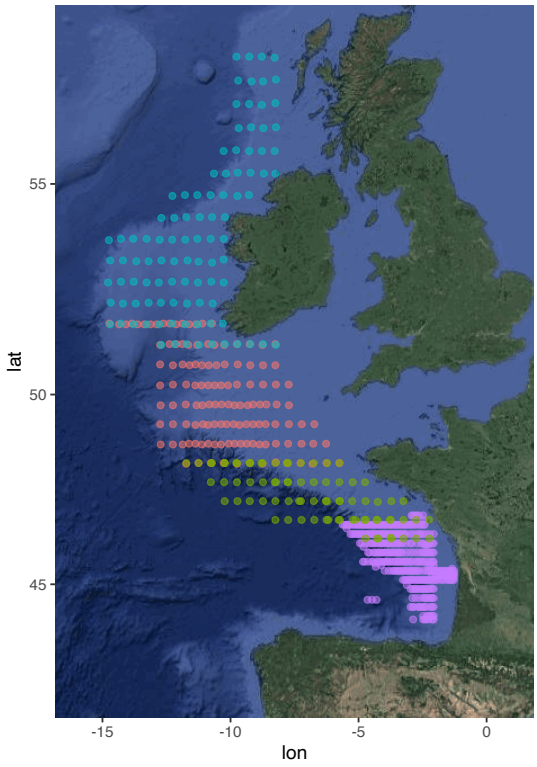
In this dataset the potential predictors include: b.depth = the depth of the seabed at the sampling location; c.dist = the distance from the sample location to the 200m contour measured in degrees (as if degrees latitude equalled degrees longitude, which actually they don't); time = the time of day (in hours) at which the sample was taken; temp.surf = the temperature at the sea surface at the sampling location; temp.20m = the temperature 20m down at the sampling location; country = a code identifying the country responsible for the boat that took this sample. Most of these are self-explanatory; c.dist is included because mackerel like to spawn near the edge of the continental shelf. It would be nice to include the date of the year, and/or month of year, as predictors but they are not included in the dataset.

So what is the strategy for this data? I would like to make a model that 1) makes precise estimates for egg density across the sampling locations; 2) tests whether environmental predictors can allow us to predict egg counts in near places/time; 3) accounts for potential spatial autocorrelation when fitting predictors. To accomplish these goals we can just make a GAM with a 2D smoother to account for spatial structure, and with additional smoothers for the environmental predictors. We should also consider putting 'country' in the model, to account for differences in methodology between different countries, and/or differences in environmental conditions at the times of those cruises. Before putting a bunch of predictors in a model, we should look at correlation among the predictors, e.g. using the ggpairs() function:

```
ggpairs(mack, columns = c('b.depth','time', 'c.dist', 'temp.surf', 'tem
p.20m', 'country'), lower = list(continuous = wrap("points", alpha = 0.
3, bg = 'blue', pch = 21), combo = wrap("facethist", bins = 10, col = '
green3'))) + theme(axis.text.y = element_text(size = 6), axis.text.x =
element_text(size = 6))
```



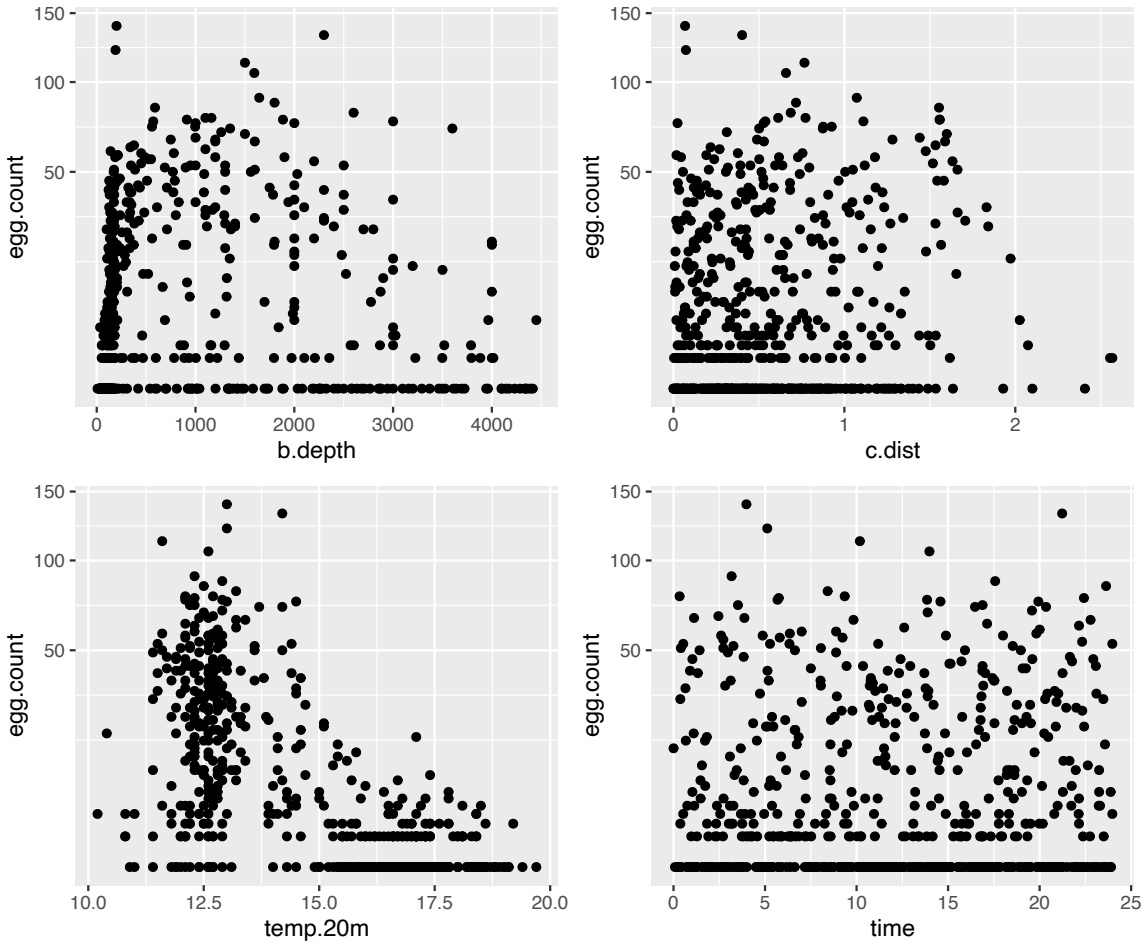
It looks like surface temperature and temperature at 20 m are highly correlated, which is not surprising. So I will just use one of these as the temperature predictor. It also looks like temperature is pretty strongly correlated with the country factor. This is not too suprising, because the countries took samples at different latitudes:



Whether to include both country and temperature as predictors is a tricky question. Country could definitely be an important predictor to include, e.g. to account for non-independence due to methodological differences. But it may also affect the estimated effect of temperature, because they are strongly correlated. If I were doing this as a real analysis I would probably try it both ways and see if the results change. For now I will just use temperature to keep things a little simpler.

Now that we've looked at predictor correlation, let's take a look at the raw patterns for the continuous predictors

```
grid.arrange(ggplot(mack) + geom_point(aes(b.depth, egg.count)) + scale
_y_continuous(trans = 'sqrt'),
             ggplot(mack) + geom_point(aes(c.dist, egg.count)) + scale_
y_continuous(trans = 'sqrt'),
             ggplot(mack) + geom_point(aes(temp.20m, egg.count)) + scal
e_y_continuous(trans = 'sqrt'),
             ggplot(mack) + geom_point(aes(time, egg.count)) + scale_y_
continuous(trans = 'sqrt'), nrow = 2)
```



I've plotted the egg count on a y-axis that is square-root transformed, which makes the patterns somewhat more visible when the data are quite skewed (many zeros, a few very large values). The pattern for b.depth and temp.20m may be nonlinear; it is more difficult to say for the other predictors.

There is one last complexity: the egg counts were not always taken from the same volume of water (i.e., net area X depth over which the net was hauled). So we need to include this volume as an offset, and the variable to use is net.area, which is the effective surface area of the sample. I will log-transform this to use as an offset:

```
mack$log.net.area = log(mack$net.area)
```

Now we're ready to specify our model. Because this is count data I will start with a quasi-Poisson model, to see whether the counts are overdispersed:

```
gm = gam(egg.count ~ s(lon,lat) + s(b.depth) + s(c.dist) + s(temp.20m)
+ s(time) + offset(log.net.area), data= mack, family = poisson, scale =
-1)
```

Let's unpack this syntax. This model has 5 smoothers as the predictors. The first one, s(lon, lat) is a two-dimensional smoother. This means that the smoother is fitting a surface over the range of lat-lon values in the dataset. We will look at different ways to plot this surface shortly. The other variables (b.dept, c.dist, temp.20m, time) are also fit using smoothers. One could also fit these as simple linear relationships, and see if the linear assumptions are realistic. But since we are already fitting a smoother for the spatial structure, and because we have a lot of data (634 samples), we can just fit smoothers and see if the optimal relationship is linear or nonlinear.

The offset, log.net.area, is included in the model (recall that the appropriate offset for count data is the log of the sampling effort/extent). The argument scale = -1 is how one fits a quasi-poisson model with the mgcv package (note this is different syntax from glm()).

A note on model building and model selection: because we have a good amount of data, and we aren't particularly interested in testing a particular predictor, this is a case where fitting a single 'full' model seems like a good option (in my opinion).

Let's take a look at the model summary:

```
summary(gm)

##
## Family: poisson
## Link function: log
##
## Formula:
## egg.count ~ s(lon, lat) + s(b.depth) + s(c.dist) + s(temp.20m) +
##     s(time) + offset(log.net.area)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.686      0.108    24.8   <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(lon,lat)  20.68  23.98 5.53 2.3e-15 ***
## s(b.depth)   7.85   8.57 6.11 6.9e-08 ***
## s(c.dist)    5.81   7.00 3.43 0.00131 **
## s(temp.20m)  5.94   6.96 4.31 0.00012 ***
## s(time)      2.34   2.95 2.69 0.04660 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.669   Deviance explained = 82.5%
## GCV =  6.729  Scale est. = 7.5255    n = 634
```

It looks like all the smoothers are important for explaining the data, though time of day might be a weak one. The output 'Scale est.' is telling us the overdispersion parameter for the quasi-poisson fit; 7.5 is quite large. I tend to avoid quasi-likelihood methods when I can, because I prefer to work with an actual probability distribution for the data, so let's try a negative binomial model to deal with overdispersion:

```
gm = gam(egg.count ~ s(lon,lat) + s(b.depth) + s(c.dist) + s(temp.20m) + s(tim
e) +offset(log.net.area), data= mack, family = nb)


summary(gm)

##
## Family: Negative Binomial(1.25)
## Link function: log
##
## Formula:
## egg.count ~ s(lon, lat) + s(b.depth) + s(c.dist) + s(temp.20m) +
##     s(time) + offset(log.net.area)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.6118     0.0561    46.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(lon,lat)  21.40  25.51  160.9 < 2e-16 ***
## s(b.depth)   3.94   4.83   29.1 2.2e-05 ***
## s(c.dist)    1.00   1.01   17.2 3.4e-05 ***
## s(temp.20m)  6.29   7.44   44.0 4.0e-07 ***
## s(time)      4.98   6.07   22.8 0.00092 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.572   Deviance explained = 74.8%
## -REML = 1600.1  Scale est. = 1         n = 634
```

Note that the negative binomial distribution is specified here with family = nb. The estimate for theta for the distribution is 1.25, which is pretty low, consistent with substantial overdispersion.

GAM diagnostics

Before we look at the fitted effects, let's do some diagnostics:

```
gam.check(gm)

## Method: REML   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-0.0006894,0.0001601]
```

```
## (score 1600 & scale 1).
## Hessian positive definite, eigenvalue range [0.000689,103.3].
## Model rank =  66 / 66
##
## 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(lon,lat)  29.000 21.403   0.743    0.00
## s(b.depth)   9.000  3.942   0.875    0.42
## s(c.dist)    9.000  1.003   0.859    0.18
## s(temp.20m)  9.000  6.293   0.790    0.00
## s(time)      9.000  4.979   0.885    0.45
```

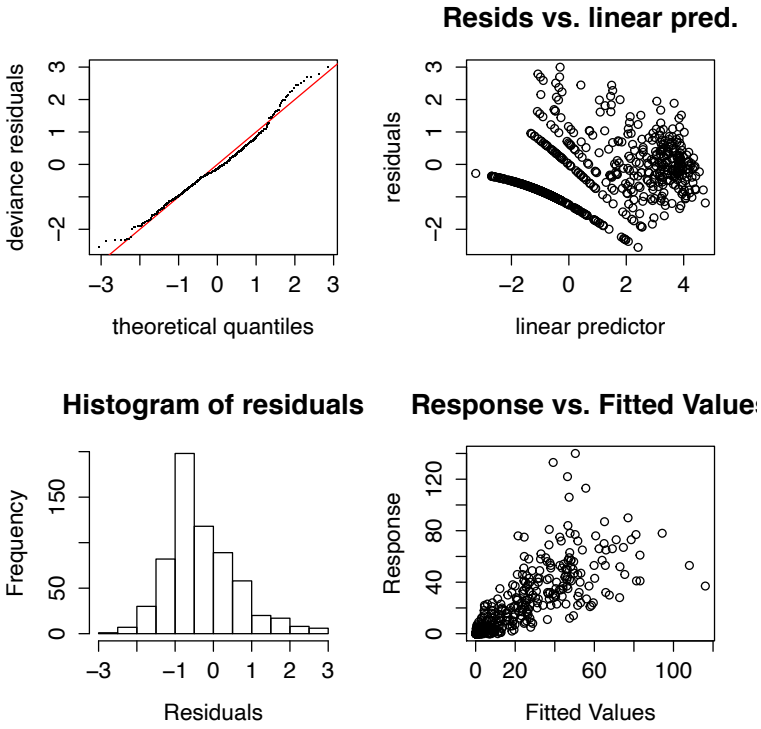
I'm primarily wondering about the lat-lon 2D smoother. If the spatial patterns are complex then we might need a larger basis dimension (k) than what the model uses by default. The results of gam.check() show that the estimated degrees of freedom for this smoother (edf) are pretty high (21) compared to the basis dimension (29). Let's try increasing the basis dimension in the model and see if the results change:

```
gm = gam(egg.count ~ s(lon,lat,k=100) + s(b.depth) + s(c.dist) + s(tem
p.20m) + s(time) +offset(log.net.area), data= mack, family = nb)
```

Now I've set k=100 for the 2D smoother, which is quite high. We can try gam.check() again:

```
gam.check(gm)
## Method: REML   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-0.000808,0.0003011]
## (score 1592 & scale 1).
## Hessian positive definite, eigenvalue range [0.0008069,80.39].
## Model rank =  136 / 136
##
## 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(lon,lat)  99.000 51.323   0.834    0.04
## s(b.depth)   9.000  3.914   0.894    0.42
## s(c.dist)    9.000  1.002   0.868    0.29
## s(temp.20m)  9.000  6.277   0.802    0.00
## s(time)      9.000  4.682   0.894    0.50
```

Now the 2D smoother has edf = 51.3, which is over twice what it was before. This suggests the smoother needs a lot of flexibility to fit the spatial patterns, so we will use this model instead. We can also look at the residual diagnostic plots:

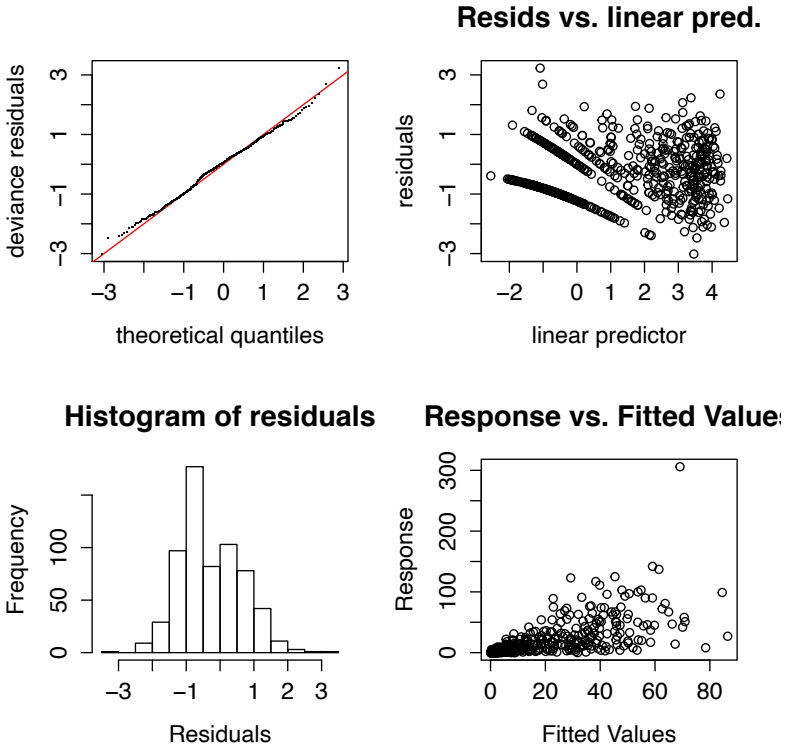


The function uses deviance residuals by definition. They appear to be roughly normally distributed, which is always a nice sign. The plot of residuals vs. fitted values (linear predictor) has some patterns: there is a stripe at bottom left corresponding to the zeros, and also the residual variance for the largest fitted values seems a bit lower than for intermediate fitted values. I think these patterns are not too worrisome, but one way to check their plausibility it to use simulation. Sadly the simulate() function is not implemented for mgcv, but we can easily simulate our own values, using fitted() to get the predicted mean for the negative binomial distribution:

```
#simulate your own values
simulated = rnbinom(length(fitted(gm)), mu = fitted(gm), size = gm$family$getT
heta(TRUE))

gm.sim = gam(simulated ~s(lon,lat,k=100) + s(b.depth) + s(c.dist) + s(temp.20
m) + s(time) +offset(log.net.area), data= mack, family = nb)
gam.check(gm.sim)
```

Here I've drawn new simulated observations from the rnbinom() function, using the fitted values of the model as the mean of the negative binomial, and using the model-estimated theta for the negative binomial as well. Then I fit the simulated data using the same model, to see what the residuals look like:



This is just one iteration of the simulation, but addition iterations look similar. These patterns are similar to what we see with the real data, although there is not that pattern where the variance of the residuals decrease at the highest fitted values. But the pattern is not so strong that I think it will give us bad results, and we will proceed (on alternative would be to try a zero-inflated GAM instead).

Plotting 1D and 2D fitted smoothers

Now that we've done some model diagnostics, what does the fitted model actually look like? Recall that we can easily plot fitted smoothers using plot(). Plotting 2D smoothers is more challenging but several nice functions are provided. I'll do it four different ways to show the options:

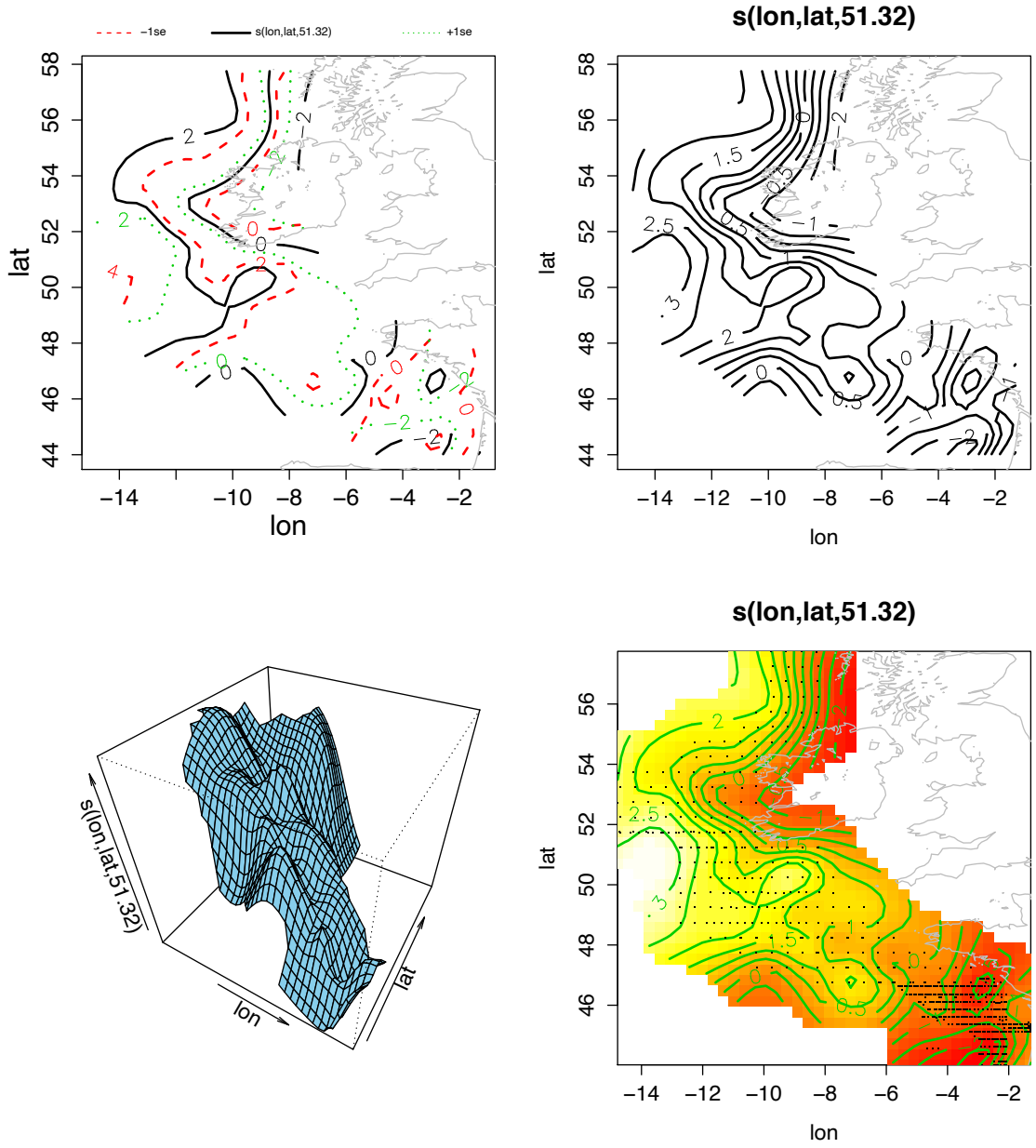
```
par(mfrow = c(2,2))
plot(gm, select = 1, lwd = 2, scheme = 0, rug = F, se = T, labcex = 0.8)
lines(coast$lon,coast$lat, col = 'grey')

plot(gm, select = 1, lwd = 2, scheme = 0, rug = F, se = F, labcex = 0.8)
lines(coast$lon,coast$lat, col = 'grey')

plot(gm, select = 1, scheme = 1, theta = 30, col = 'skyblue', phi = 40)

plot(gm, select = 1, scheme = 2, lwd = 2, labcex = 0.8)

lines(coast$lon,coast$lat, col = 'grey')
```



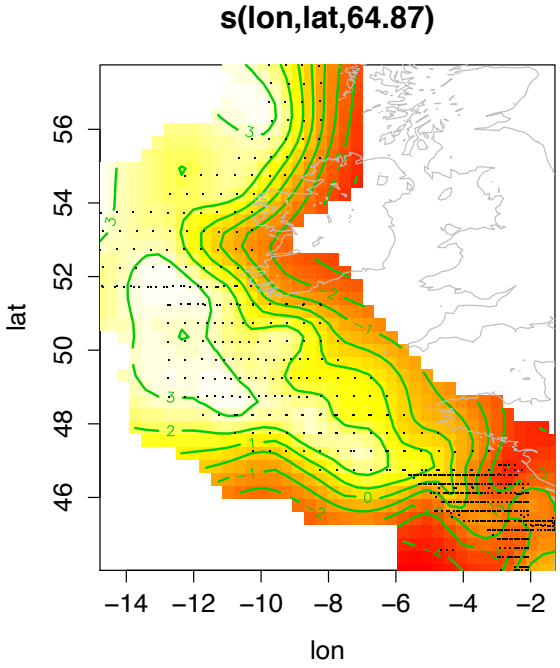
Note that here I'm using a handy coastline map included with the mackerel data, plotted on the gam smoother plot with lines().

select = 1 tells the function to plot the first of the smoothers. 'scheme' can be 0/1/2, which indicate contour plot, perspective plot, or heatmap+contours, respectively. 'rug' determines whether or not points are plotted to show where the samples where taken. For the first plot I used se=T, which plots a contour plot with standard errors on the contours to indicate uncertainty in the fitted surface. The contourplot is hard to read, but the black lines are contours at -2, 0, and 2. Recall that the smoother is centered around 0 (i.e., 0 represent the mean conditions based on the

other parameters in the model), and the units are on the log link scale, so a difference of 2 means a change in egg density by a factor of exp(2) = 7.4. The Red and green lines show +/- 1 SE for the contour.

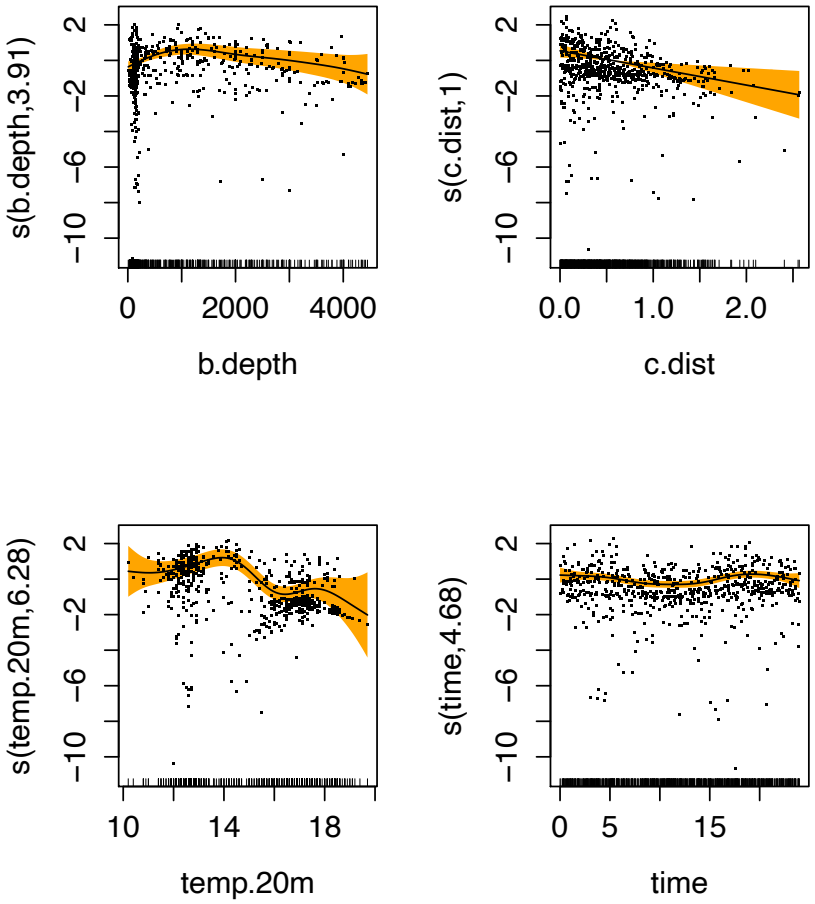
The plot on the topleft is a contourplot without the standard error contours. Now it's easier to see the shape of the surface; egg density increases as you go west or south from Ireland, and there is also a weaker hotspot west of France. The plot on the bottom left is a 3D plot that combines the fitted smooth with the function persp(). You can adjust the angle of view using the arguments 'theta' and 'phi'. Finally, the plot on the bottom right plots the fitted smooth as a heat map with a contourplot overlaid. This is probably the easiest to grasp at a glance, though the labels are hard to read (if anyone knows how to change the label color on a contour() plot, let me know).

So now we have a sense for the spatial variation in egg density fit by the model. There is substantial variation predicted by this smoother, over 4-fold variation on the log link scale, which is over 50-fold variation . We can also see that there are some strong geographic patterns. It is important to remember that because we fit a model with multiple predictors, *this smoother is fitting the spatial variation not explained by the other predictors*. If we wanted to visualize the spatial patterns alone, without trying to explain them with other predictors, we could just fit a gam that only has the 2D smoother as a predictor. That's an interesting comparison to make, so I did it and plotted the results:



This is similar though not identical to the previous plot. So the 2D smoother in our full model is still finding a similar pattern, but some of the raw pattern can be explained with the predictors. Let's look at the fitted effects of those predictors:

```
par(mfrow = c(2,2))
plot(gm, select = 2, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack')
plot(gm, select = 3, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack')
plot(gm, select = 4, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack')
plot(gm, select = 5, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack')
```

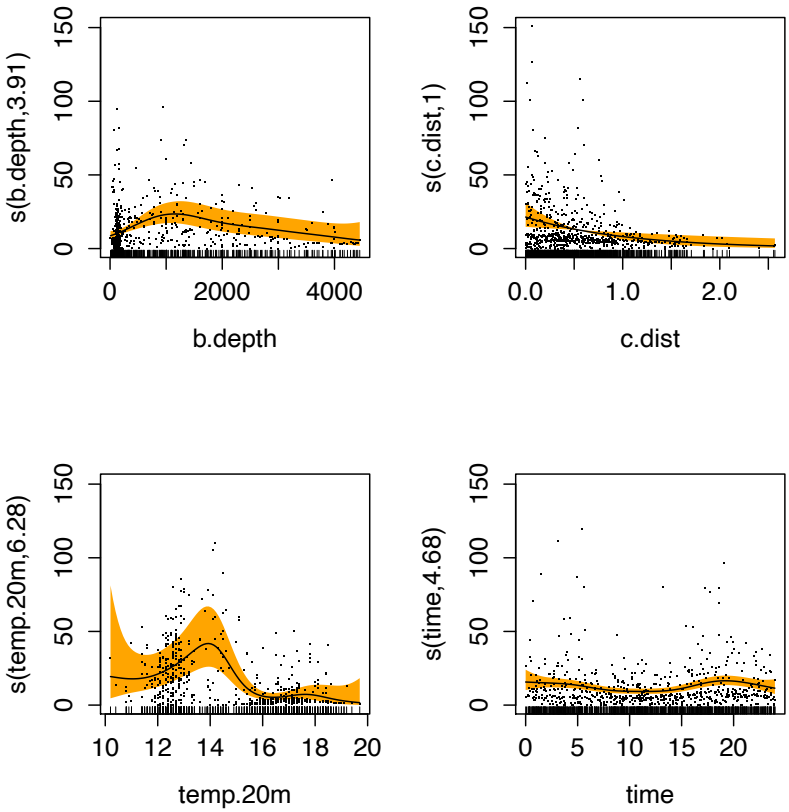


Here I've plotted the four other predictors, with partial residuals to show the remaining spread in the data. The effect of seafloor depth is unimodal, with a peak around 1000 m, and the predicted effect of depth varies by about 2 units. The effect of c.dist, distance from the 200 m depth contour, is linear (on the link scale), with egg density declining monotonically from the 200 m contour. The effect of temperature is a little wiggly, which may be due in part to the fact that there is not

an even spread of temperatures sampled; rather they are clustered around 12-13 and 16-18. But the pattern is roughly unimodal, which is consistent with how organisms respond to temperature, with a peak around 14 degrees. The last smoother is for time of day; I don't know why egg density would be related to time of day, but I was curious so I put it in the model. There is weaker variation compared to the other predictors, but egg density appears to peak around 3 AM and 6 PM.

By default the smoothers are plotted on the link scale, and are centered to have a mean of zero. If we want to plot the smoother on the scale of the raw data, we can add in the appropriate intercept using the argument 'shift' (in this case it's just the model intercept; in other cases you might have an intercept specific to a factor level), and then transform the smoother back to the scale of the raw data using the argument 'trans'.

```
par(mfrow = c(2,2))
plot(gm, select = 2, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack', shift = coef(gm)[1], trans = function(x) exp(x))
plot(gm, select = 3, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack', shift = coef(gm)[1], trans = function(x) exp(x)))
plot(gm, select = 4, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack', shift = coef(gm)[1], trans = function(x) exp(x)))
plot(gm, select = 5, residuals = T, shade.col = 'orange', shade = T, col = 'bl
ack', shift = coef(gm)[1], trans = function(x) exp(x)))
```



This is a little harder to look at, due to the skewness of the data, but may be easier to think about in terms of how much these predictors affect egg densities.

Now we've got a sense for the fitted effects of this model. The 2D spatial smoother appears to have the largest effect, but there are important contributions from seabed depth, distance from the shelf edge, and temperature. It seems like seabed depth and distance from shelf edge may be partially explaining the same phenomenon; egg densities tend to increase and then decrease as you move away from land, along the shelf, and then off the shelf. The peak for b.depth occurs around 1000 meters, which suggests that maybe the 200 m contour is not the best reference point for capturing the preferred depth of spawning (for the variable c.dist). However, depth alone cannot account for most of the spatial structure; some is explained by temperature, while most of the variation appears to be unexplained by these predictors (but captured by the 2D smoother).

All of the smoothers explain significant variation; we can see this from summary(), which by default returns approximate marginal F-tests for each smoother, or we can get the same info with anova(), which also returns marginal F-tests (note this is different from anova() for lm and glm).

```
anova(gm)
```

```
##
## Family: Negative Binomial(1.56)
## Link function: log
##
## Formula:
## egg.count ~ s(lon, lat, k = 100) + s(b.depth) + s(c.dist) + s(temp.20m) +
##     s(time) + offset(log.net.area)
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(lon,lat)  51.32  67.26 245.58 < 2e-16
## s(b.depth)   3.91   4.75  24.85 0.00013
## s(c.dist)    1.00   1.00   8.35 0.00390
## s(temp.20m)  6.28   7.39  41.47 1.1e-06
## s(time)      4.68   5.71  18.14 0.00499
```

It would be nice to plot the effects of the predictors on the map as well, but before we do that let's do another diagnostic.

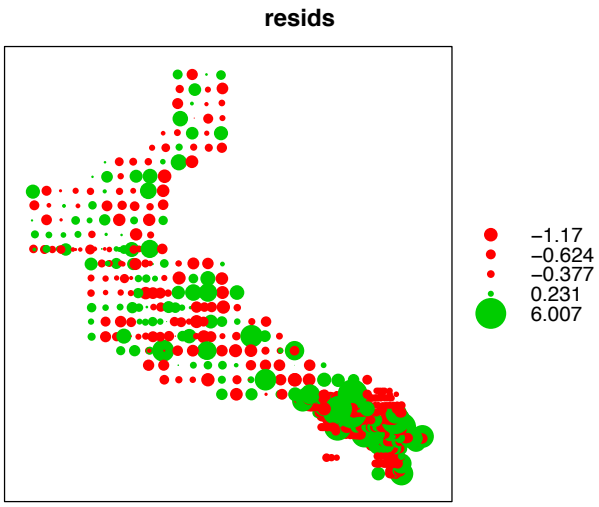
Spatial autocorrelation and variogram

I mentioned earlier that with spatially extensive sampling a common issue is autocorrelation in the data, which violates the assumption of independently distributed data. The assumption is that the response variable is independent after accounting for the predictors, so what we need to assess is whether the residual variation has any structure. R has a lot of great packages for processing and analyzing spatial datasets, and we will only get a flavor for them in this class. The package 'sp' has a nice function for visualizing whether residuals have spatial structure:

```
library(sp)

dat<-data.frame(lon = mack$lon, lat = mack$lat, resids=resid(gm, type =
'pearson'))
coordinates(dat)<-c('lon','lat')
```

```
bubble(dat,zcol='resids')
```

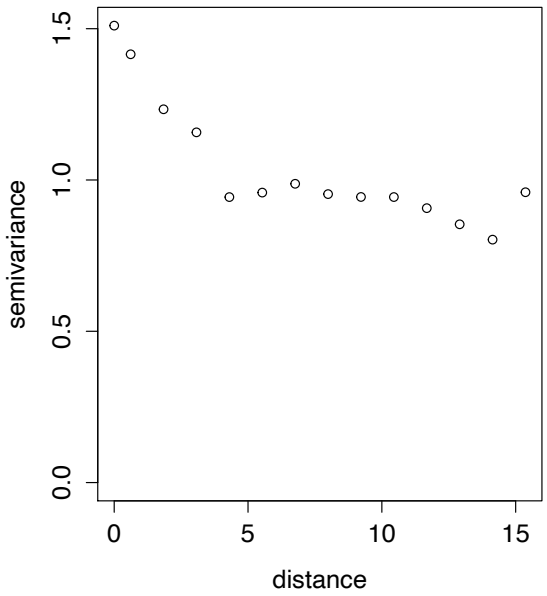


The function 'bubble' plots residuals based on their spatial coordinates, and makes the size of the bubble proportional to the size of the residual, and colors them red or green for negative or positive. There does not appear to be any strong clustering of positive/negative residuals, which would indicate autocorrelation. To look specifically at autocorrelation, we can make a *semivariogram*, using a different spatial stats package:

```
library(geoR)

v1 <- variog(coords = mack[,c('lat','lon')], data = residuals(gm, type
= 'pearson'))

plot(v1)
```



What is the semivariogram plotting? The idea is to see how similar residuals are, relative to their distance from each other in space. Similarity is just based on the squared difference, $(e_i - e_k)^2$, where e_i and e_k are two residuals. By calculating the squared difference for all pairs of residuals, and seeing how the mean squared difference changes as the distance between the residuals changes, we test for autocorrelation. If the residuals are independently distributed there should be no relationship between the similarity of residuals and their distance. If nearby residuals are similar, e.g. due to unmeasured environmental effects, then the semivariogram will show lower values at lower distance, and then increase and eventually level off.

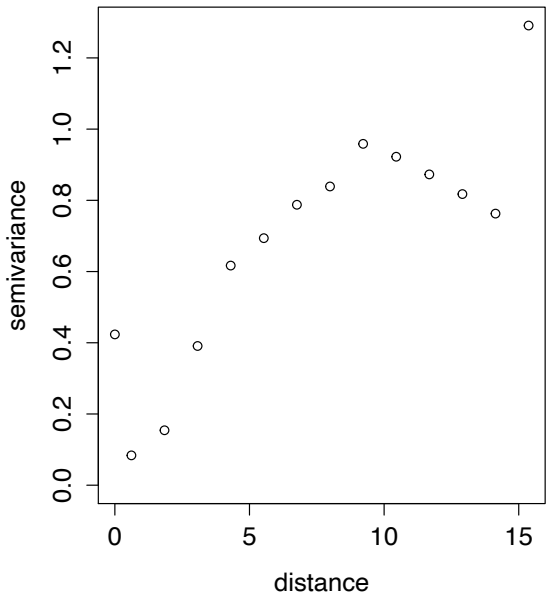
In the semivariogram based on our mackerel model, there is no such pattern, and in fact a slight decrease as distance increases. This indicates that there is no residual spatial autocorrelation under this model. As an extreme comparison, we can compare what happens when we make a model with no predictors:

```
gm.nopred = gam(egg.count ~ 1 +offset(log.net.area), data= mack, family = nb)

dat<-data.frame(lon = mack$lon, lat = mack$lat, resids=resid(gm.nopred))
coordinates(dat)<-c('lon','lat')
bubble(dat,zcol='resids')

v1 <- variog(coords = mack[,c('lat','lon')], data = residuals(gm.nopred, type
= 'pearson'))
```

```
plot(v1)
```



Now there is clearly spatial structure in the residuals, and the semivariance doesn't level off until a distance of about 10.

Plotting model predictions on a grid

After I fit the mackerel model, I wanted to see how the different predictors (temperature, distance to shelf edge, etc) result in predictions for how egg density varies across space. We already looked at how the 2D smoother makes predictions for how egg density varies over space. To make a similar plot for the other predictors, one approach is 1) make a new dataframe where each predictor is averaged over a spatial grid, 2) use the gridded predictors to make predictions using the fitted model. Obviously the predictors based on seabed depth don't change over time, while temperature does, but a quick look showed that in this dataset temperature shows a strong latitudinal gradient (not shown). Luckily for us, someone has already made a dataframe where the predictors are averaged and interpolated in quarter-degree grid cells; it is part of the gamair package, dataset 'mackp'.

```
data(mackp)
```

We want to use this gridded dataframe with the predict() function to extract model predictions, so we also need all the other predictors used in the model. The offset variable should just be set to some constant number, and the 'time' variable isn't meaningful over space, so I'll set it to 12:

```
#add a column for the offset (hold it constant at the median)
mackp$log.net.area = median(mack$log.net.area)
mackp$time = 12
```

Now that we have all these variables in a dataframe, it is easy to get model predictions for each grid cell:

```
mackp$predicted = predict(gm, newdata = mackp)
```

We can make a heat map of this data using the image() function. The image() function takes a matrix of values as the third argument, and the first two arguments are the x- and y-coordinates corresponding to the rows and columns of the matrix, respectively. So we need to make vectors for the lats and lons, and a matrix that puts the mackp values in the right locations. The mackp dataframe already contains a column that indexes each row based on its location in the grid:

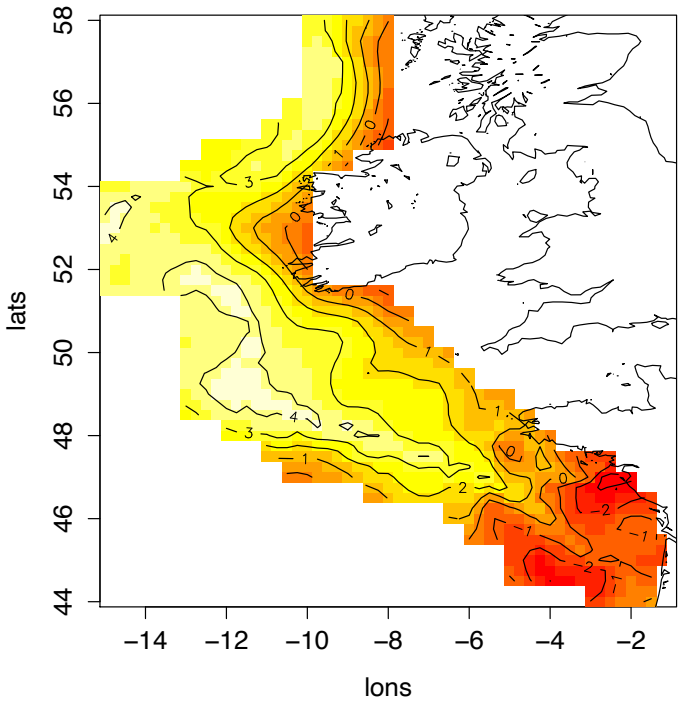
```
lons<-seq(-15,-1,1/4)
lats<-seq(44,58,1/4)
zz<-matrix(NA, nrow = 57, ncol = 57)
zz[mackp$area.index]<-mackp$predicted
```

```
#use image to make a heat map of the predictions
image(lons,lats,zz)

#make a contour plot on top of the heat map
contour(lons,lats,zz, add = T)

#add the coastline
lines(coast$lon,coast$lat,col=1)
```

I've also overlaid a contour plot on top of the heatmap, using contour(), which takes the same arguments.

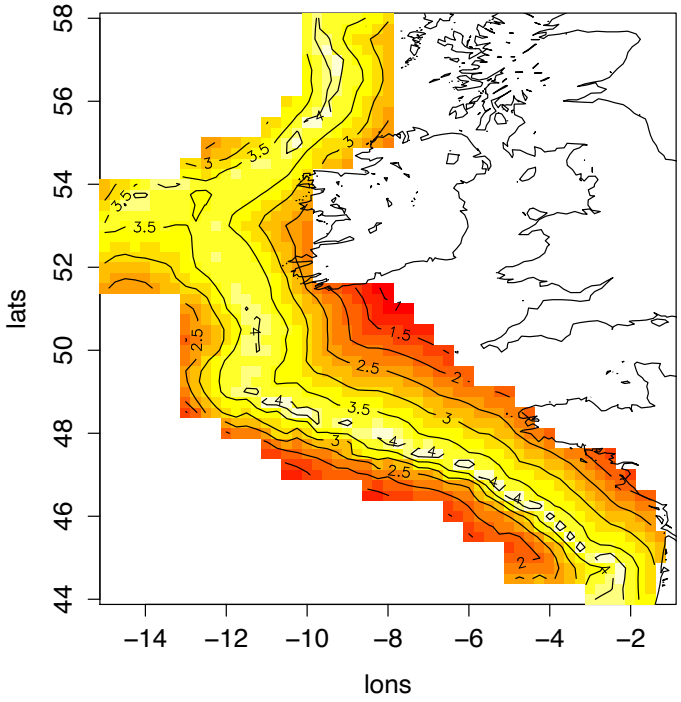


So this is what the model predictions look like projected onto space. The model predicts steep increases in egg density moving west from Ireland, and based on the data we have it's not clear at what distance the egg densities start to decline. Heading south from Ireland the densities do appear to decline as we move off the shelf. In addition the densities are much lower off the coast of France.

This is useful information, especially for the goal of the authors of the original study. We can also parse out how individual predictors vary over space, and what their effects are. To do this we can take the same mackp dataset, and set the other predictors to some constant value:

```
#predictions for c.dist
data(mackp)
mackp$log.net.area = median(mack$log.net.area)
mackp$time = 12
mackp$lat = 50
mackp$lon = -8
mackp$temp.20m = mean(mackp$temp.20m)
mackp$predicted = predict(gm, newdata = mackp)
```

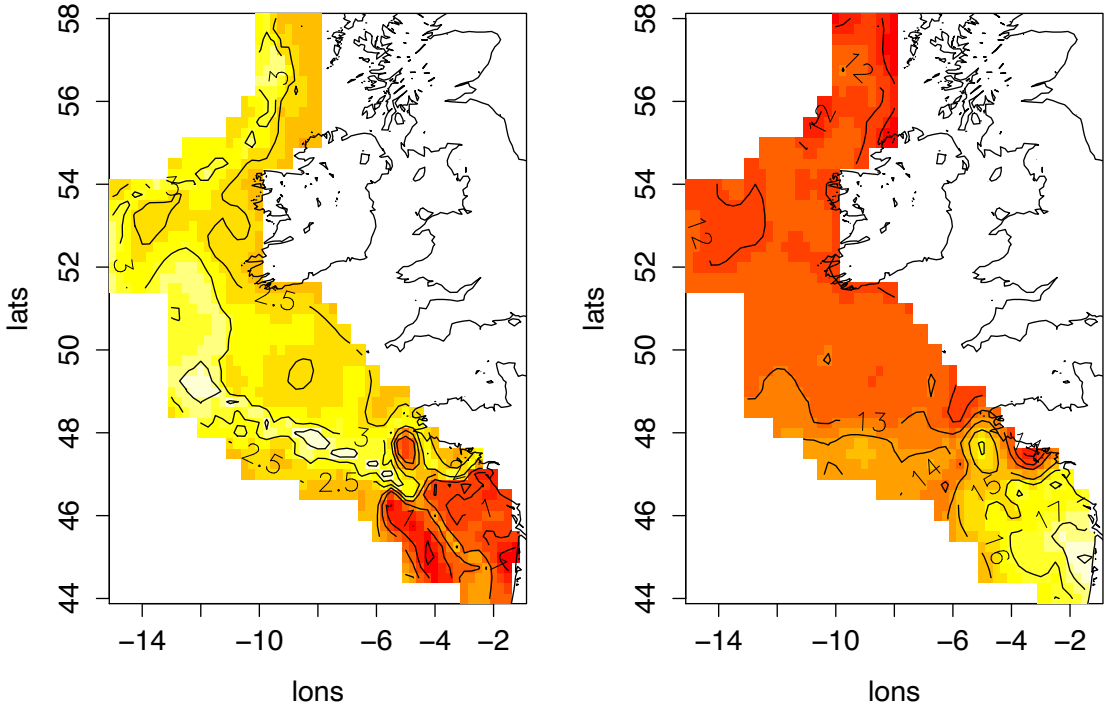
To look at the effect of c.dist (distance from the shelf edge), I've held lat and lon constant at a value in the middle of the dataset, and I've set temp.20m to its mean value. So now the only predictor that varies in mackp is c.dist. If we use the same plotting code we get this:



We can see clearly how egg density is predicted to peak at the edge of the continental shelf, with a total range of about exp(2) = 7.4. We can do the same thing for temperature:

```
#effect of temp
data(mackp)
mackp$log.net.area = median(mack$log.net.area)
mackp$time = 12
mackp$lat = 50
mackp$lon = -8
mackp$c.dist = mean(mackp$c.dist)

#make predictions on the grid
mackp$predicted = predict(gm, newdata = mackp)
```



Here I've also plotted the temperature predictor on the right. The big pattern here is the increase in temperature at the southern extremes, which is predicted to reduce egg density. But the pattern is somewhat more complex because the predicted effect of temperature is not monotonic. We can see that the highest egg densities are predicted in the areas between 13-14 degrees, and they decline slightly move north (colder) from there.

Conclusions

This example shows a lot of what GAMs can do, though not everything. Often you may need to account for temporal structure, and smoothers can be good for that, e.g. s(Day) or s(Year). 2D smoothers are not just for spatial variation; you can use them to fit nonparametric interactions between predictors, e.g. how do light and nitrate interact to affect primary production. You can also include linear interactions between a smoother and a continuous predict, which means that the height of the smoother uniformly goes up or down as the second variable changes.