

# Prediction using fitted density surface models

*Solution 6, Intermediate Distance Sampling workshop, CREEM, 2018*

The analysis here is conditional on selecting a detection function and DSM in the previous exercises; I've shown a variety of models selected in the previous solutions to show the differences between models.

Much of the text below is as in the exercise itself, so it should be relatively easy to navigate.

Additional text and code is highlighted using boxes like this.

Now we've fitted some models, let's use the `predict` functions and the data from GIS to make predictions of abundance.

## Loading the packages and data

```
library(knitr)
library(dsm)

## Loading required package: mgcv
## Loading required package: nlme
## This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.
## Loading required package: mrds
## This is mrds 2.2.0
## Built: R 3.5.1; ; 2018-08-03 20:22:18 UTC; windows
## Loading required package: numDeriv
## This is dsm 2.2.16
## Built: R 3.4.2; ; 2017-11-02 15:05:30 UTC; windows

library(ggplot2)
# colourblind-friendly colourschemes
library(viridis)

## Loading required package: viridisLite
# to load and save raster data
library(raster)

## Loading required package: sp
##
## Attaching package: 'raster'
## The following object is masked from 'package:nlme':
##
##   getData
# models with only spatial terms
load("dsms-xy.RData")
# models with all covariates
load("dsms.RData")
```

## Loading prediction data

Before we can make predictions we first need to load the covariates into a “stack” from their files on disk using the `stack()` function from `raster`. We give `stack()` a vector of locations to load the rasters from. Note that in RStudio you can use tab-completion for these locations and avoid some typing. At this point we arbitrarily choose the time periods of the SST, NPP and EKE rasters (2 June 2004, or Julian date 153).

```
predictorStack <- stack(c("Covariates_for_Study_Area/Depth.img",
                        "Covariates_for_Study_Area/GLOB/CMC/CMC0.2deg/analysed_sst/2004/20040602-CMC-1",
                        "Covariates_for_Study_Area/VGPM/Rasters/vgpm.2004153.hdf.gz.img",
                        "Covariates_for_Study_Area/DistToCanyonsAndSeamounts.img",
                        "Covariates_for_Study_Area/Global/DT\ all\ sat/MSLA_ke/2004/MSLA_ke_2004153.i
                        ))
```

We need to rename the layers in our stack to match those in the model we are going to use to predict. If you need a refresher on the names that were used there, call `summary()` on the DSM object.

```
names(predictorStack) <- c("Depth", "SST", "NPP", "DistToCAS", "EKE")
```

Now these are loaded, we can coerce the stack into something `dsm` can talk to using the `as.data.frame` function. Note we need the `xy=TRUE` to ensure that `x` and `y` are included in the prediction data. We also set the offset value – the area of each cell in our prediction grid.

```
predgrid <- as.data.frame(predictorStack, xy=TRUE)
predgrid$off.set <- (10*1000)^2
```

We can then predict for the model `dsm_nb_xy_ms`:

```
pp <- predict(dsm_nb_xy_ms, predgrid)
```

This is just a list of numbers – the predicted abundance per cell. We can sum these to get the estimated abundance for the study area:

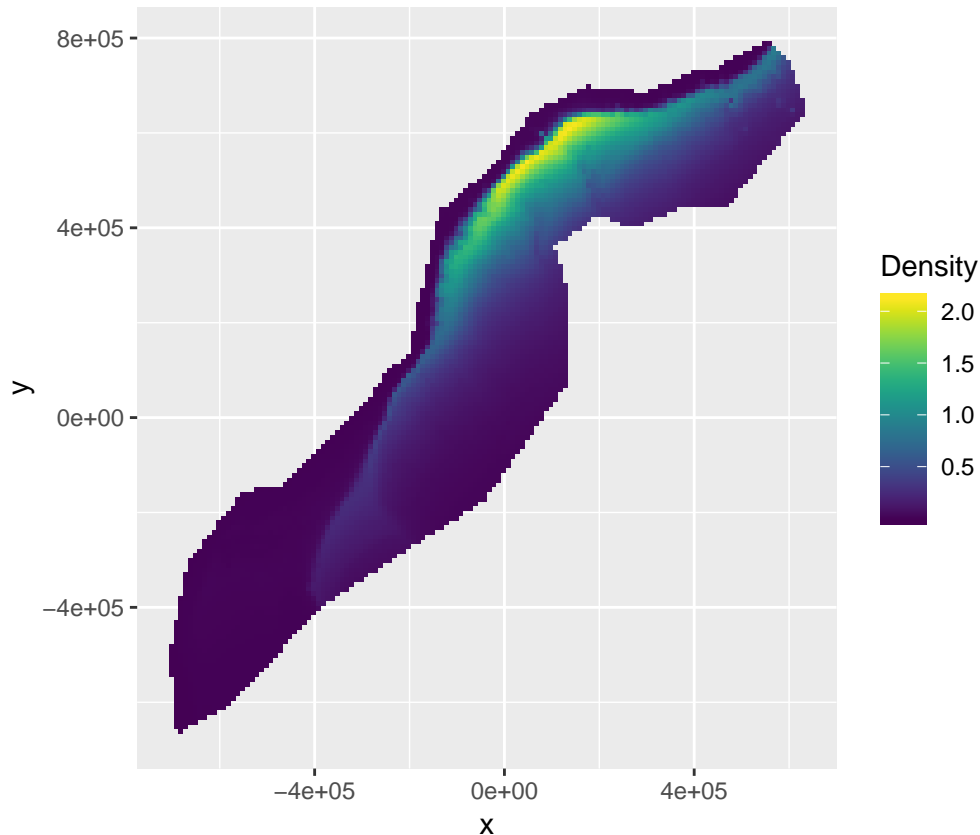
```
sum(pp, na.rm=TRUE)
```

```
## [1] 1581.535
```

Because we predicted over the whole raster grid (including those cells without covariate values – e.g. land), some of the values in `pp` will be `NA`, so we can ignore them when we sum by setting `na.rm=TRUE`. We need to do this again when we plot the data too.

We can also plot this to get a spatial representation of the predictions:

```
# assign the predictions to the prediction grid data.frame
predgrid$Nhat_nb_xy <- pp
# remove the NA entries (because of the grid structure of the raster)
predgrid_plot <- predgrid[!is.na(predgrid$Depth),]
# plot!
p <- ggplot(predgrid_plot) +
  geom_tile(aes(x=x, y=y, fill=Nhat_nb_xy, width=10*1000, height=10*1000)) +
  coord_equal() +
  labs(fill="Density")+
  scale_fill_viridis()
print(p)
```



Copy the chunk above and make predictions for the other models you saved in the previous exercises. In particular, compare the models with only spatial terms to those with environmental covariates included.

We now want to plot the predictions for the 8 models that we have so far (4 negative binomial, 4 Tweedie; 4 spatial only, 4 with environmental covariats; 4 with bivariate smooths, 4 with additive spatial effects).

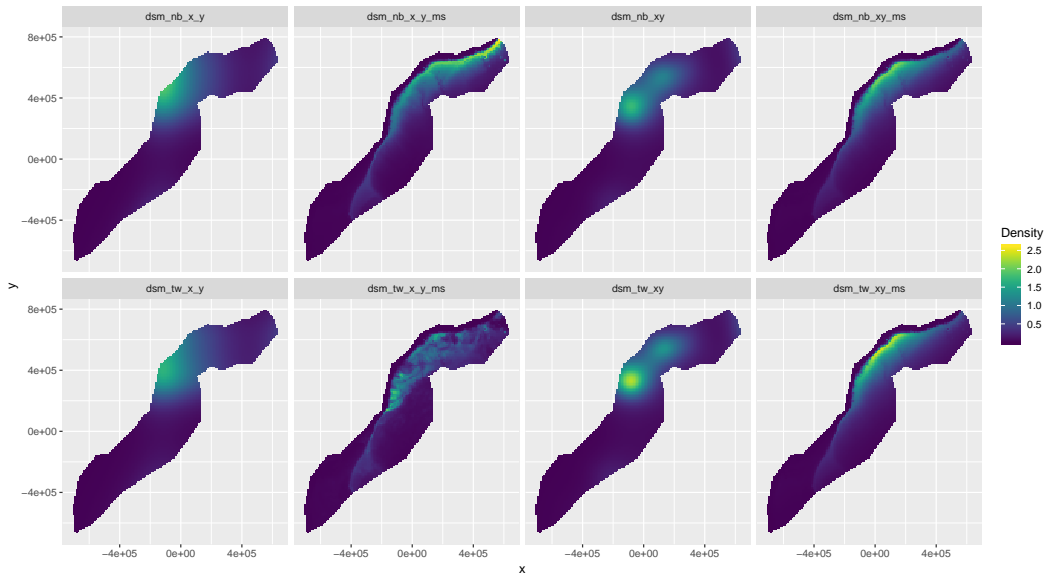
Duplicating the above code is a bit tiresome and can be prone to errors, so let's nerd-out pretty heavily for this solution and show how we can use the `ldply()` function from `plyr` to do the same task many times.

```
# make a function that makes the predictions, adds them to a column named Nhat
# and adds a column called "model" that stores the model name, then returns the
# data.frame.
make_pred_dat <- function(model_name, predgrid){
  # we use get() here to grab the object with the name of its argument
  predgrid[["Nhat"]] <- predict(get(model_name), predgrid)
  predgrid[["model"]] <- model_name
  return(predgrid)
}

# load plyr and apply to a list of the names of the models, make_pred_dat returns
# a data.frame (hence this is an "ld" function: list->data.frame) that it then binds
# together
library(plyr)
big_predgrid <- ldply(list("dsm_nb_xy", "dsm_nb_x_y", "dsm_nb_xy_ms", "dsm_nb_x_y_ms",
  "dsm_tw_xy", "dsm_tw_x_y", "dsm_tw_xy_ms", "dsm_tw_x_y_ms"),
  make_pred_dat, predgrid=predgrid_plot)

# make the plot, facetting using the model column
```

```
p <- ggplot(big_predgrid) +
  geom_tile(aes(x=x, y=y, fill=Nhat, width=10*1000, height=10*1000)) +
  coord_equal() +
  facet_wrap(~model, nrow=2)+
  labs(fill="Density")+
  scale_fill_viridis()
print(p)
```



Note here that the `_ms` models have the environmental covariates, the others are spatial-only.

We can also use `plyr` to help calculate overall abundance...

```
# ddply will apply summarize (which in turn sums the Nhat column) to the subsets of
# the data defined by model (i.e. each model)
Nhat_results <- ddply(big_predgrid, ~(model), summarize, Nhat=sum(Nhat))
```

We can use our friend `kable` to make a nice table of this information:

model	Nhat
dsm_nb_x_y	1567.074
dsm_nb_x_y_ms	1698.465
dsm_nb_xy	1534.406
dsm_nb_xy_ms	1581.535
dsm_tw_x_y	1660.377
dsm_tw_x_y_ms	1254.980
dsm_tw_xy	1696.642
dsm_tw_xy_ms	1624.766

## Save the prediction to a raster

To be able to load our predictions into ArcGIS, we need to save them as a raster file. First we need to make our predictions into a raster object and save them to the stack we already have:

```

# setup the storage for the predictions
pp_raster <- raster(predictorStack)
# put the values in, making sure they are numeric first
pp_raster <- setValues(pp_raster, as.numeric(pp))
# name the new, last, layer in the stack
names(pp_raster) <- "Nhat_nb_xy"

```

We can then save that object to disk as a raster file:

```
writeRaster(pp_raster, "abundance_raster.img", datatype="FLT4S")
```

Here we just saved one raster layer: the predictions from model `Nhat_nb_xy`. Try saving another set of predictions from another model by copying the above chunk.

You can check that the raster was written correctly by using the `stack()` function, as we did before to load the data and then the `plot()` function to see what was saved in the raster file.

## Save prediction grid to RData

We'll need to use the prediction grid and predictor stack again when we calculate uncertainty in the next practical, so let's save those objects now to save time later.

```
save(predgrid, predictorStack, file="predgrid.RData")
```

## Extra credit

- Try refitting your models with `family=quasipoisson()` as the response distribution. What do you notice about the predicted abundance?

Fitting a quasi-Poisson model (and doing a quick bit of term selection):

```

# load data
load("df-models.RData")
load("sperm-data.RData")
obs <- obs[obs$distance <= df_hn$ddf$meta.data$width,]

dsm_qp_xy_ms <- dsm(count~s(x,y, bs="ts") +
                    s(Depth, bs="ts") +
                    #s(DistToCAS, bs="ts") + # 3
                    s(SST, bs="ts", k=18), # + # 1 increase basis complexity
                    #s(EKE, bs="ts") + # 2
                    #s(NPP, bs="ts"), # 4
                    df_hn, segs, obs,
                    family=quasipoisson())
summary(dsm_qp_xy_ms)

##
## Family: quasipoisson
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(SST, bs = "ts",
##      k = 18) + offset(off.set)
##

```

```

## Parametric coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) -21.4282    0.3014  -71.09  <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)    16.518    29 2.887 5.77e-13 ***
## s(Depth)   4.771     9 6.236 2.04e-14 ***
## s(SST)    10.420    17 4.744 1.52e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.41  Deviance explained = 51.2%
## -REML = 296.45  Scale est. = 1.0973    n = 949

```

Now predicting abundance:

```

pp_qp <- predict(dsm_qp_xy_ms, predgrid)
sum(pp_qp, na.rm=TRUE)

```

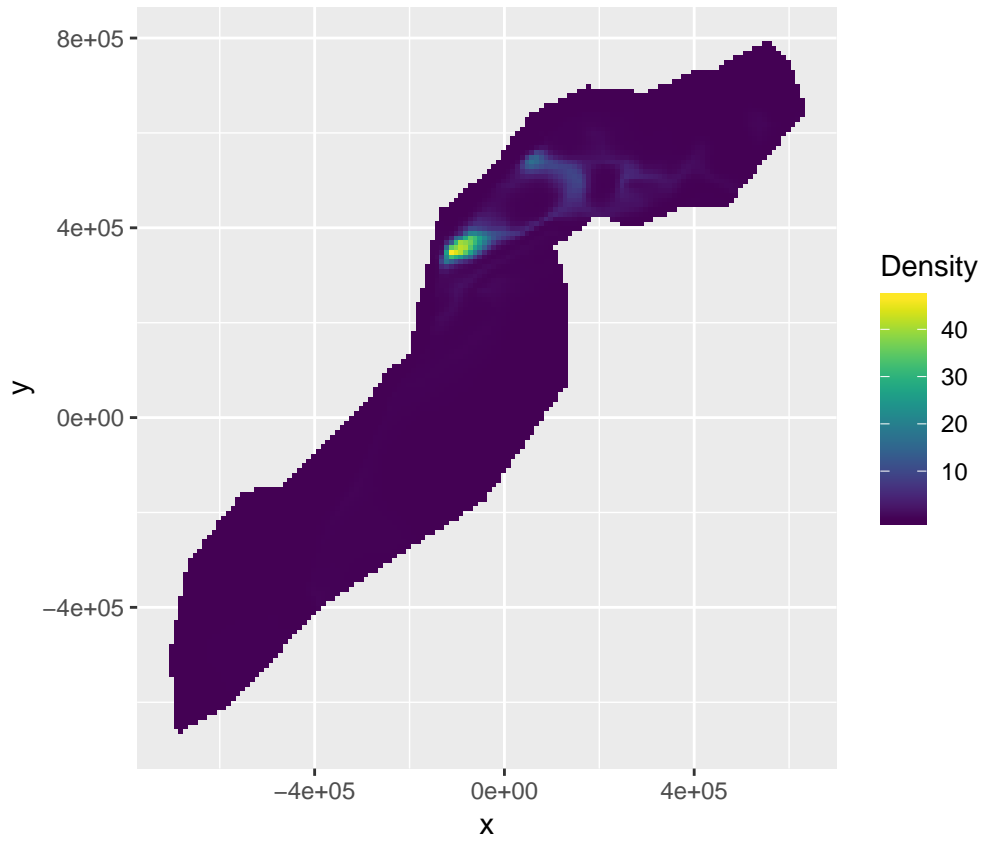
```
## [1] 3326.175
```

And we can make a map:

```

predgrid$Nhat_qp_xy <- pp_qp
predgrid_plot <- predgrid[!is.na(predgrid$Depth),]
# plot!
p <- ggplot(predgrid_plot) +
  geom_tile(aes(x=x, y=y, fill=Nhat_qp_xy, width=10*1000, height=10*1000)) +
  coord_equal() +
  labs(fill="Density")+
  scale_fill_viridis()
print(p)

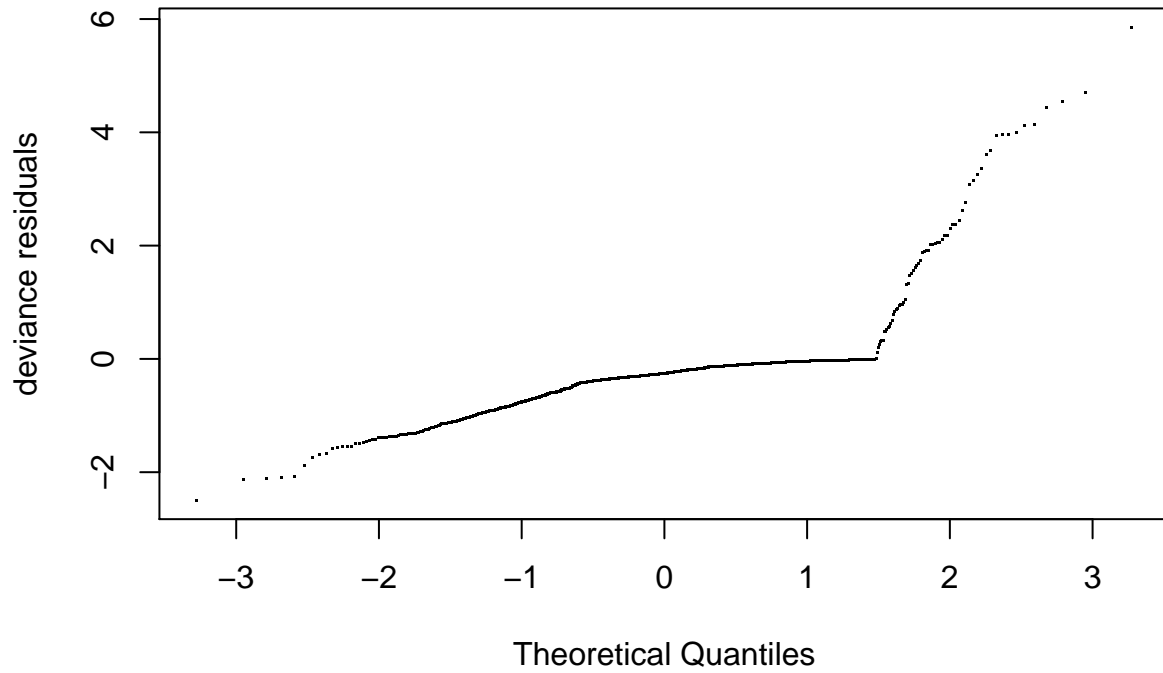
```



The plot and prediction of abundance are very different from what we've seen above. Note again the distributional issues highlighted in the Q-Q plot for this model:

```
qq.gam(dsm_qp_xy_ms)
```

## Normal Q-Q Plot



- Can you work out a way to use `ldply()` from the `plyr` package so that you can use `facet_wrap` in `ggplot2` to plot predictions for multiple models in a grid layout?

See above!