Advanced topics in distance sampling

Workshop, 26-30 August 2019

Centre for Research into Ecological and Environmental Modelling

Solution 7. Estimating precision of predictions from density surface models

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 and estimated abundance, we can estimate the variance associated with the abundance estimate (and plot it).

Load packages and data

```
library(dsm)
```

```
## Loading required package: mgcv
```

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

```
## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: mrds
```

```
## This is mrds 2.2.1
## Built: R 3.6.1; ; 2019-07-17 13:15:38 UTC; windows
```

```
## Loading required package: numDeriv
```

```
## This is dsm 2.2.17
## Built: R 3.6.1; ; 2019-07-11 19:38:05 UTC; windows
```

```
library(raster)
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:nlme':
##
##     getData
```

```
library(ggplot2)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(plyr)
library(knitr)
library(rgdal)
```

```
## rgdal: version: 1.4-4, (SVN revision 833)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: C:/Users/louise/Documents/R/win-library/3.6/rgdal/gdal
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: C:/Users/louise/Documents/R/win-library/3.6/rgdal/pr
##  Linking to sp version: 1.3-1
```

Load the models and prediction grid:

```
load("dsms.RData")
load("dsms-xy.RData")
load("predgrid.RData")
```

Estimation of variance

Depending on the model response (count or Horvitz-Thompson) we can use either `dsm.var.prop` or `dsm.var.gam`, respectively. `dsm_nb_xy_ms` doesn't include any covariates at the observer level in the detection function, so we can use so we can use `dsm.var.gam` to estimate the uncertainty.

```
# need to remove the NAs as we did when plotting
predgrid_var <- predgrid[!is.na(predgrid$Depth),]
# now estimate variance
var_nb_xy_ms <- dsm.var.gam(dsm_nb_xy_ms, predgrid_var, off.set=predgrid_var$off.set)
```

To summarise the results of this variance estimate:

```
summary(var_nb_xy_ms)
```

```
## Summary of uncertainty in a density surface model calculated
##  analytically for GAM, with delta method
##
## Approximate asymptotic confidence interval:
##     2.5%     Mean    97.5%
## 1123.709 1589.216 2247.565
## (Using log-Normal approximation)
##
## Point estimate                 : 1589.216
```

```
## CV of detection function       : 0.06670757
## CV from GAM                    : 0.1653
## Total standard error           : 283.2538
## Total coefficient of variation : 0.1782
```

Try this out for some of the other models you've saved. Remember the rule when there are covariates in the detection function model:

- use `dsm.var.prop` if there are covariates in the detection function that vary at the level of the segment (like Beaufort) and
- use `dsm.var.gam` if there are individual-level covariates (like observer) in the detection function

If there are no covariates in the detection function, use `dsm.var.gam` since there's no covariance between the detection function and the spatial model.

We'll skip this and go straight to summarising all our models so far in the next section...

Summarise multiple models

We can again summarise all the models, as we did with the DSMs and detection functions, now including the variance:

```
# This function harvests relevant statistics
summarize_dsm_var <- function(model, predgrid){

  summ <- summary(model)

  vp <- summary(dsm.var.gam(model, predgrid, off.set=predgrid$off.set))
  unconditional.cv.square <- vp$cv^2
  asymp.ci.c.term <- exp(1.96*sqrt(log(1+unconditional.cv.square)))
  asymp.tot <- c(vp$pred.est / asymp.ci.c.term,
                 vp$pred.est,
                 vp$pred.est * asymp.ci.c.term)

  data.frame(response = model$family$family,
             terms    = paste(rownames(summ$s.table), collapse=", "),
             AIC      = AIC(model),
             REML     = model$gcv.ubre,
             "Dev_exp" = paste0(round(summ$dev.expl*100,1),"%"),
             "low_CI" = round(asymp.tot[1],2),
             "Nhat" = round(asymp.tot[2],2),
             "up_CI" = round(asymp.tot[3],2)
             )
}
```

This takes a long time!

```
# make a list of models
model_list <- 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)
# give the list names for the models, so we can identify them later
names(model_list) <- c("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"
per_model_var <- ldply(model_list, summarize_dsm_var, predgrid=predgrid_var)
```

```
kable(per_model_var, digits=1, booktabs=TRUE)
```

.id	response	terms	AIC	REML	Dev_exp	low_
dsm_nb_xy	Negative Binomial(0.105)	s(x,y)	775.3	392.6	40.6%	113
dsm_nb_x_y	Negative Binomial(0.085)	s(x), s(y)	789.8	395.9	31.3%	109
dsm_nb_xy_ms	Negative Binomial(0.108)	s(x,y), s(Depth)	758.1	384.8	37.7%	112
dsm_nb_x_y_ms	Negative Binomial(0.098)	s(y), s(Depth)	762.6	386.1	35.9%	119
dsm_tw_xy	Tweedie(p=1.29)	s(x,y)	1249.4	394.9	34.7%	127
dsm_tw_x_y	Tweedie(p=1.306)	s(x), s(y)	1252.3	399.8	27.4%	122
dsm_tw_xy_ms	Tweedie(p=1.268)	s(x,y), s(Depth)	1229.9	389.9	38%	123
dsm_tw_x_y_ms	Tweedie(p=1.282)	s(Depth), s(NPP)	1227.3	387.9	34.7%	74

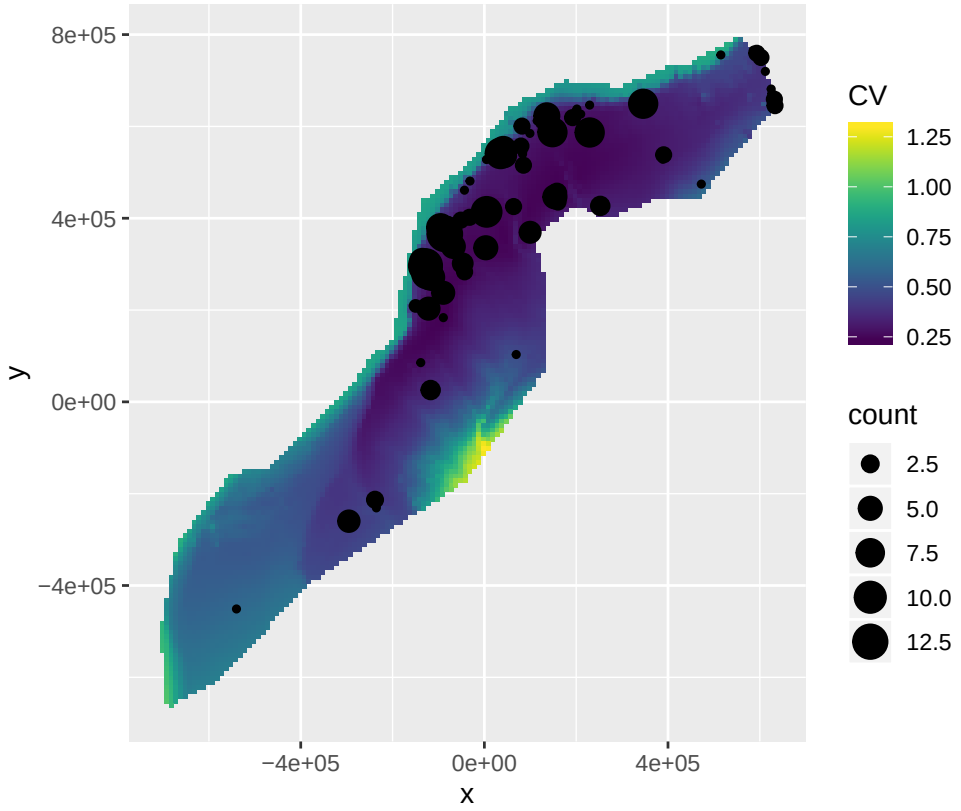
Plotting

We can plot a map of the coefficient of variation, but we first need to estimate the variance per prediction cell, rather than over the whole area. This calculation takes a while!

```
# use the split function to make each row of the predictiond data.frame into
# an element of a list
predgrid_var_split <- split(predgrid_var, 1:nrow(predgrid_var))
var_split_nb_xy_ms <- dsm.var.gam(dsm_nb_xy_ms, predgrid_var_split, off.set=predgrid_
```

Now we have the per-cell coefficients of variation, we assign that to a column of the prediction grid data and plot it as usual:

```
predgrid_var_map <- predgrid_var
cv <- sqrt(var_split_nb_xy_ms$pred.var)/unlist(var_split_nb_xy_ms$pred)
predgrid_var_map$CV <- cv
p <- ggplot(predgrid_var_map) +
       geom_tile(aes(x=x, y=y, fill=CV, width=10*1000, height=10*1000)) +
       scale_fill_viridis() +
       coord_equal() +
       geom_point(aes(x=x,y=y, size=count), data=dsm_nb_xy_ms$data[dsm_nb_xy_ms$data$
print(p)
```



Note that here we overplot the segments where sperm whales were observed (and scale the size of the point according to the number observed), using `geom_point()`.

Try this with the other models you fitted and see what the differences are between the maps of coefficient of variation.

We can use a similar technique as we did in the prediction exercises to get coefficient of variation maps for all of the models...

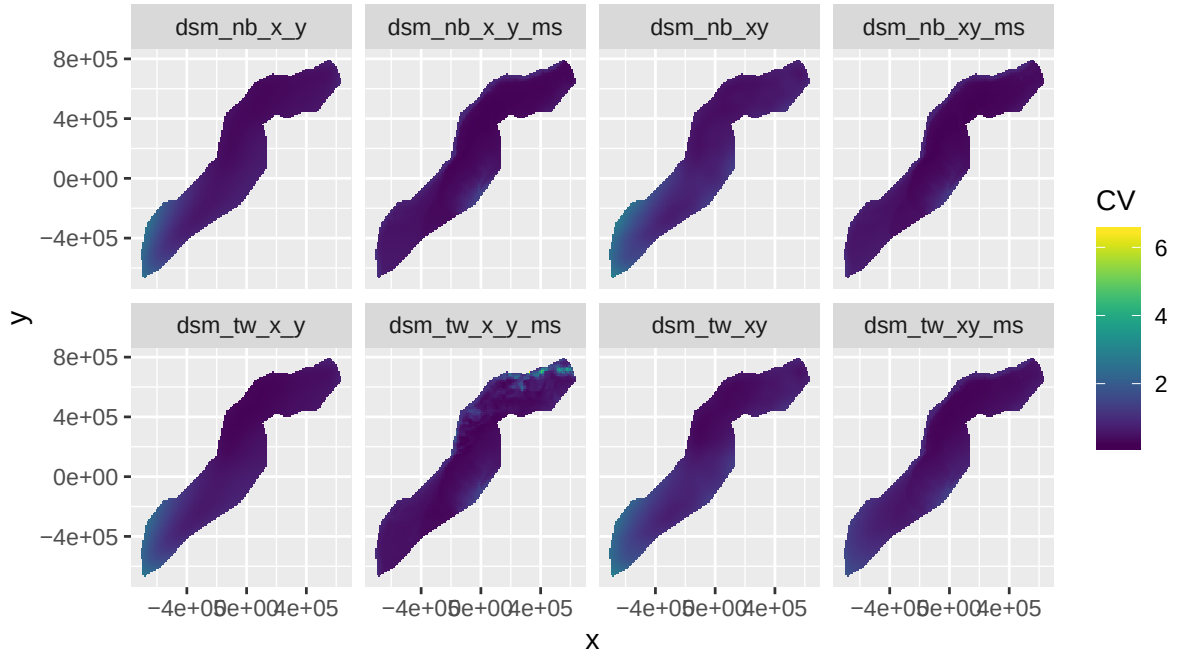
Note this can take a long time!

```
# make a function that calculates the CV, adds them to a column named CV
# and adds a column called "model" that stores the model name, then returns the
# data.frame.
make_cv_dat <- function(model_name, predgrid_split, predgrid){
  # we use get() here to grab the object with the name of its argument
  var_obj <- dsm.var.gam(get(model_name), predgrid_split, off.set=predgrid$off.set)
  predgrid[["CV"]] <- sqrt(var_obj$pred.var)/unlist(var_obj$pred)
  predgrid[["model"]] <- model_name
  return(predgrid)
}

# load plyr and apply to a list of the names of the models, make_cv_dat returns
# a data.frame (hence this is an "ld" function: list->data.frame) that it then binds
# together
library(plyr)
big_cv <- 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_cv_dat, predgrid_split=predgrid_var_split, predgrid=predgrid_var
```

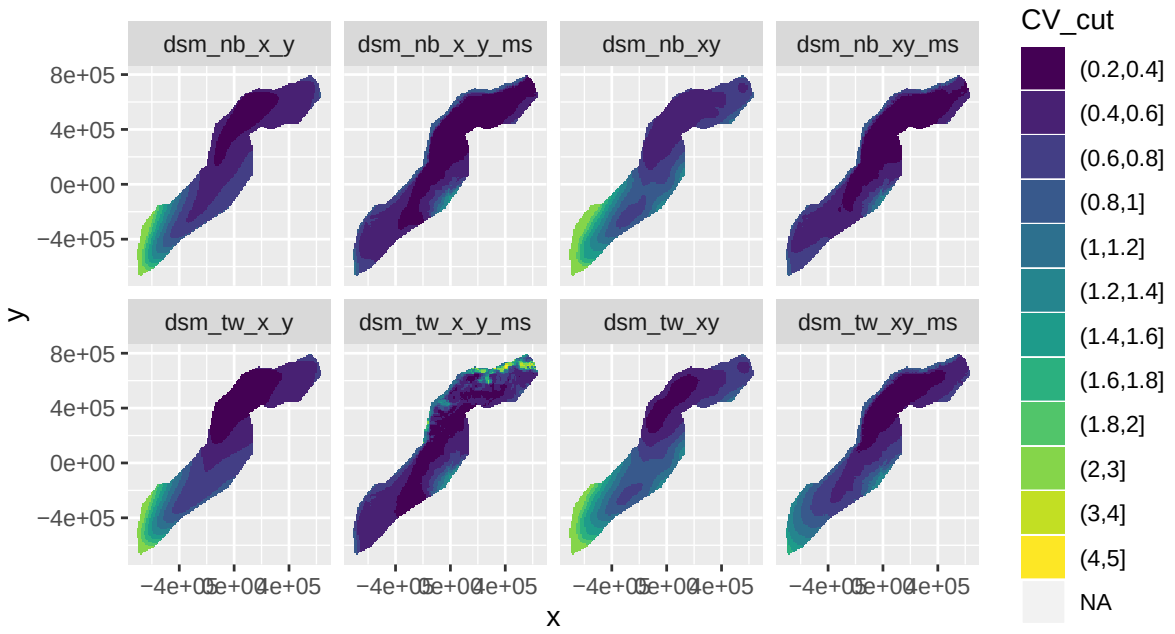
```
p <- ggplot(big_cv) +
       geom_tile(aes(x=x, y=y, fill=CV, width=10*1000, height=10*1000)) +
       scale_fill_viridis() +
       coord_equal() +
       facet_wrap(~model, nrow=2)
print(p)
```



One issue with producing this kind of plot is that `ggplot2` will use a common legend, meaning if there are some cells with high values, this can throw out the rest of the scale.

As suggested in the lectures, we'll now use `cut()` to create a categorised version of the CV and plot that instead:

```
big_cv$CV_cut <- cut(big_cv$CV, c(seq(0,2,0.2),3:6))
p <- ggplot(big_cv) +
       geom_tile(aes(x=x, y=y, fill=CV_cut, width=10*1000, height=10*1000)) +
       scale_fill_viridis(discrete=TRUE) +
       coord_equal() +
       facet_wrap(~model, nrow=2)
print(p)
```



This is a bit easier to read.

Save the uncertainty maps to raster files

As with the predictions, we'd like to save our uncertainty estimates to a raster layer so we can plot them in GIS. Again, this involves a bit of messing about with the data format before we can save.

```
# setup the storage for the cvs
cv_raster <- raster(predictorStack)
# we removed the NA values to make the predictions and the raster needs them
# so make a vector of NAs, and insert the CV values...
cv_na <- rep(NA, nrow(predgrid))
cv_na[!is.na(predgrid$Depth)] <- cv
# put the values in, making sure they are numeric first
cv_raster <- setValues(cv_raster, cv_na)
# name the new, last, layer in the stack
names(cv_raster) <- "CV_nb_xy"
```

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

```
writeRaster(cv_raster, "cv_raster.img", datatype="FLT4S")
```