Point model workshop

Download the R Studio project from github

You can access this document, the R code, and the data used in this workshop by downloading my R Studio project. It's located at `https://github.com/wrbrooks/seals`.

Problem

The National Marine Mammal Laboratory is tasked with tracking the population of harbor seals in the wild. One tool is the use of aerial photography. Disenchantment Bay is a tidewater glacial fjord in Southeast Alaska with a significant population of harbor seals. Our goal is to estimate the total population of harbor seals in Disenchantment Bay. The estimate is to be based on a partial census by aerial imaging, and we will use smoothing to fill in the unobserved areas.

Data

A small small airplane was make several transects over the bay, taking photos straight down at 100m intervals, where each photo includes an area of 80m x 120m. The photos don't overlap and the transects are spaced far enough apart that no seal should be photographed twice. The seals in each photograph were counted by hand. Since the transects only cover a portion of the bay, we will have to make a model and estimate the total seal population based on the observed images.

Let's import the data. `seal_locs` is a list of locations where a seal was observed, `photo_sites` is the center point of each aerial photograph, and `pred` is a raster of the study area. We'll be predicting the density of seals within each 53m x 53m cell of `predict`.

A raster is a way of saying the region has been broken up into little pieces (like pixels on a screen). We will assume that the density of seals is constant within each cell of the raster, which makes it easy to sum the density over the whole bay - just add up the values of each raster cell.

```r
source("R/auxiliary-functions.R")
seal_locs <- read.csv("data/dbay_20040622_seal_locs.csv")
pred <- read.csv("data/20040622_nonpup_predict.csv")
photo_sites <- read.csv("data/photo_sites.csv")
```

Now we can plot the data. The red squares are photo sites and the black circles are seal locations.

```r
plot(pred$x, pred$y, col='grey70', bty='n', xlab="Easting (m)", ylab="Northing (m)", pch=15)
points(seal_locs[,c('ET_X', 'ET_Y')])
points(photo_sites, col='red', pch=22)
```

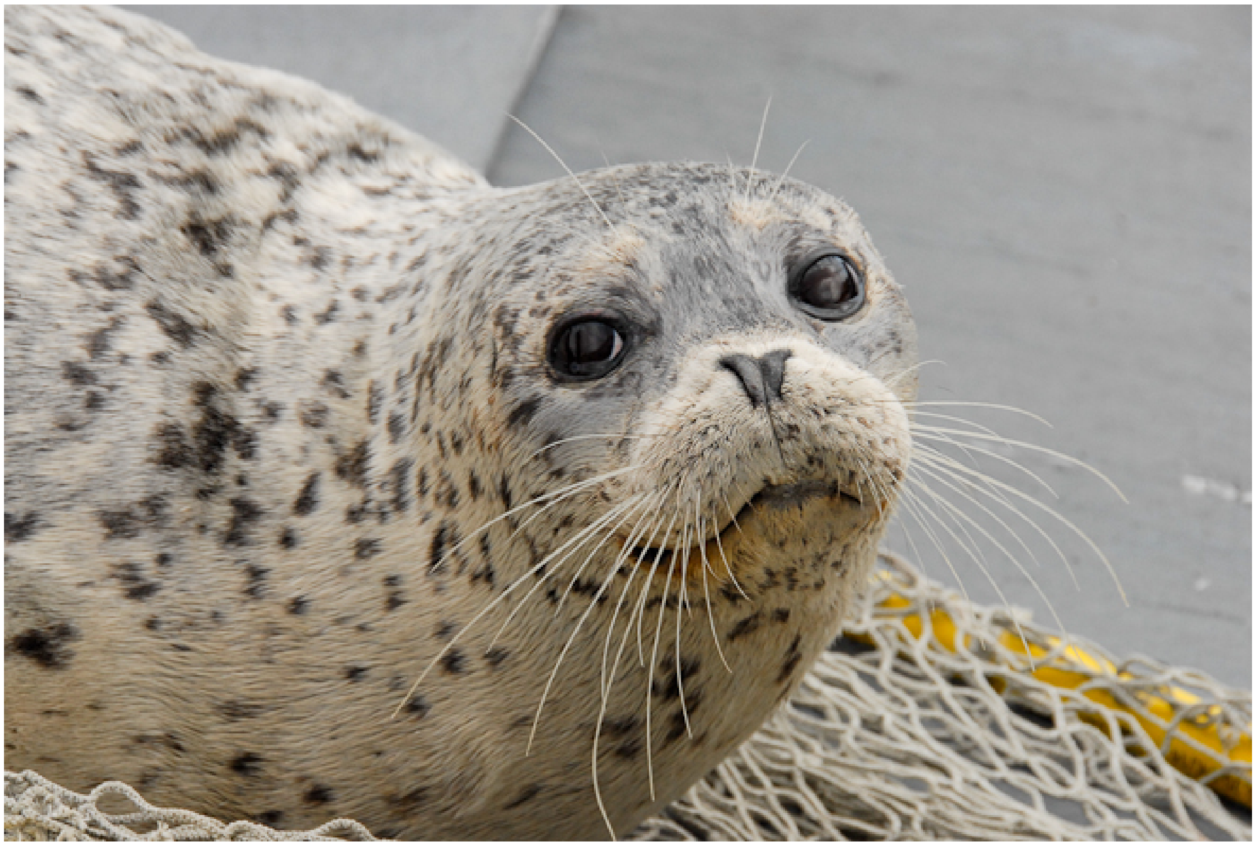


Figure 1: A harbor seal

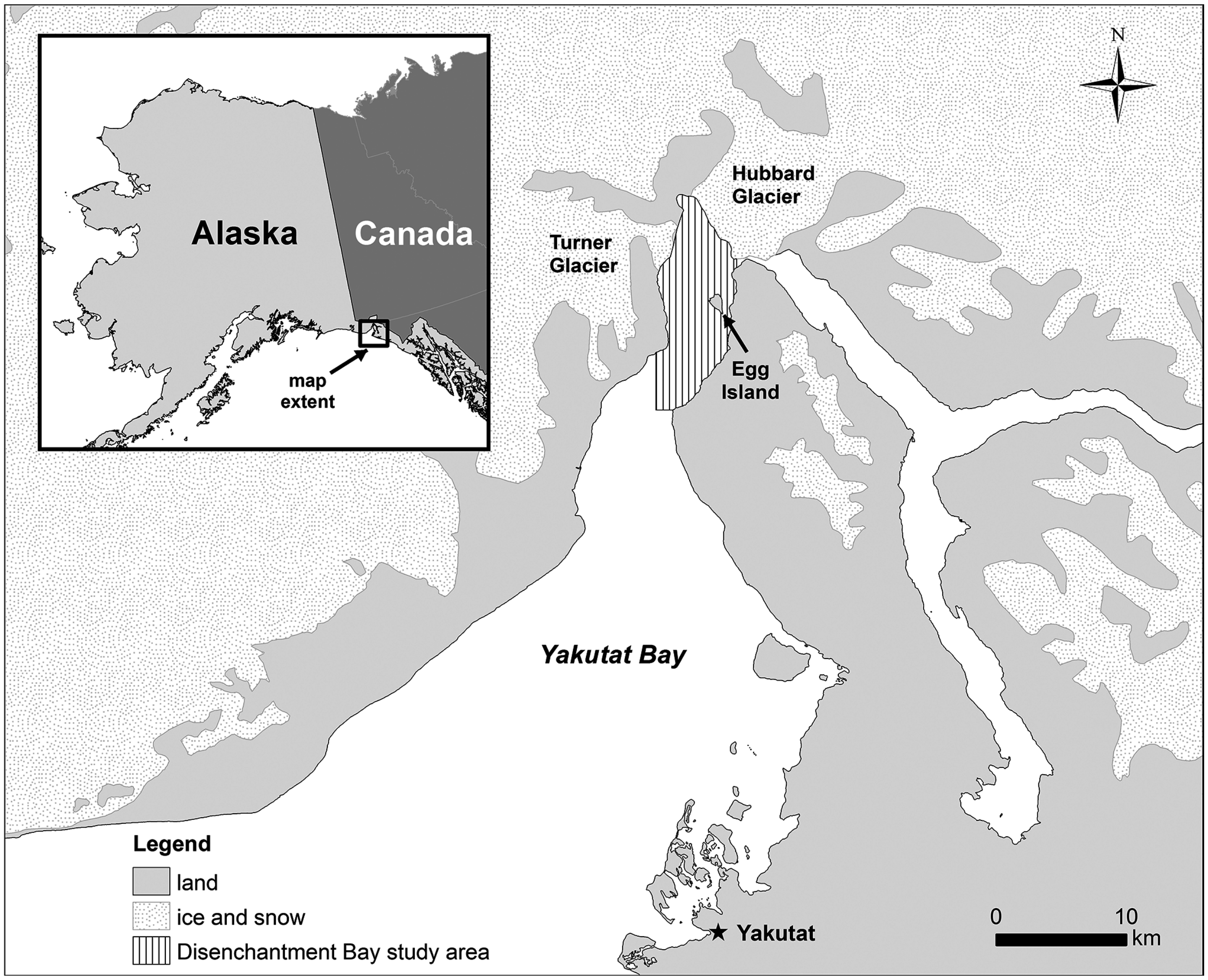
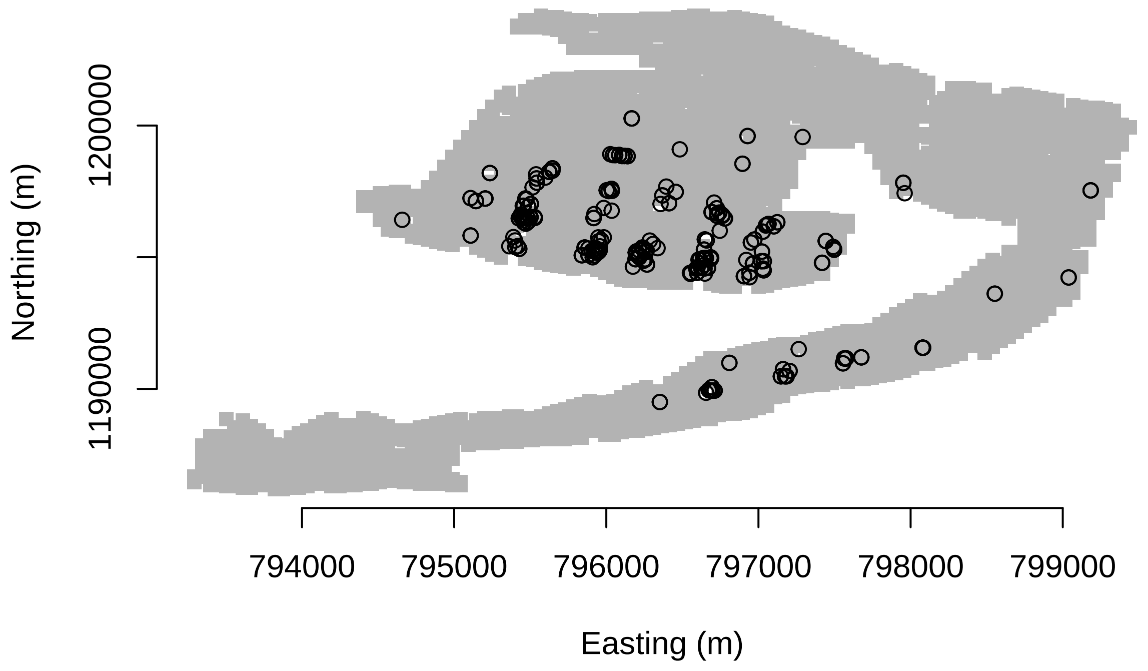


Figure 2: Disenchantment Bay, AK



We can see that the photos aren't quite aligned with the seal locations, but we will just snap each seal to the nearest photograph:

```
# Each photo is a bin. We will put each seal into the nearest bin, and keep count of how many seals are
seal_counts <- rep(0, nrow(photo_sites))
for (i in 1:nrow(seal_locs)) {
  photo_dist <- apply(photo_sites, 1, function(s) (s[1] - seal_locs$ET_X[i])**2 +
                        (s[2] - seal_locs$ET_Y[i])**2 )
  seal_counts[which.min(photo_dist)] <- seal_counts[which.min(photo_dist)] + 1
}
```

Smoothing with spatial basis functions

We use spatial basis functions to smooth the density of seals over the whole of Disenchantment Bay, which allows us to predict the number of seals that weren't observed in the aerial photos.

The idea of spatial basis functions is to cover a spatial region with several local functions that can be independently adjusted to find the overall function that best fits the data. A simple example is a checkerboard pattern where the spatial function takes a constant value within each square, but different squares have different values. Our method is only a little more complicated.

We span the study area with Gaussian functions of two resolutions: coarse and fine. We will estimate a coefficient u for each Gaussian, as indicated in the figure.

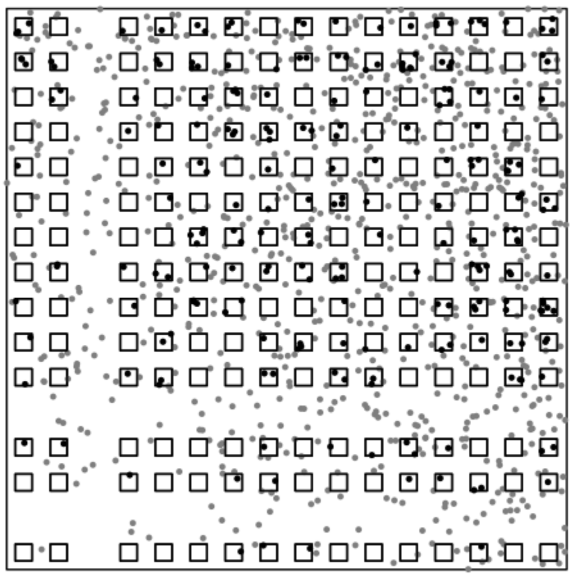
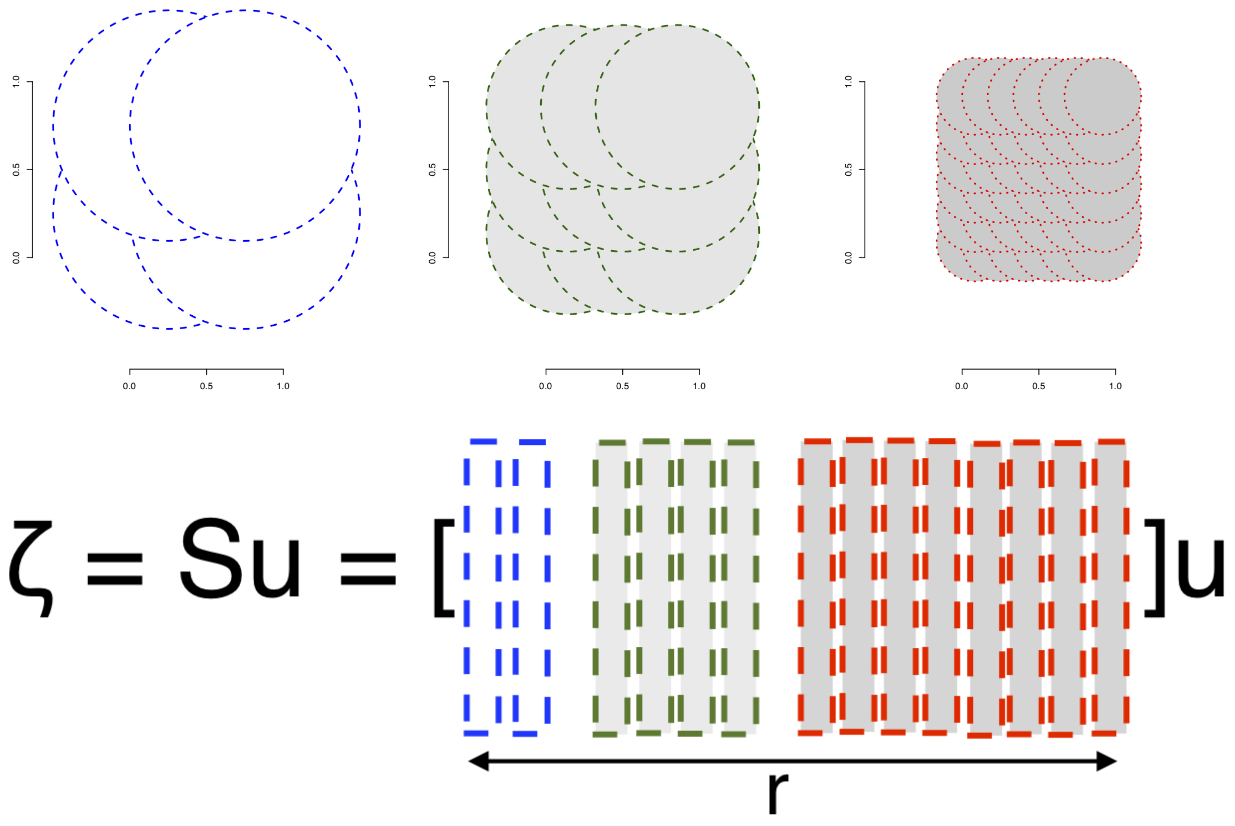


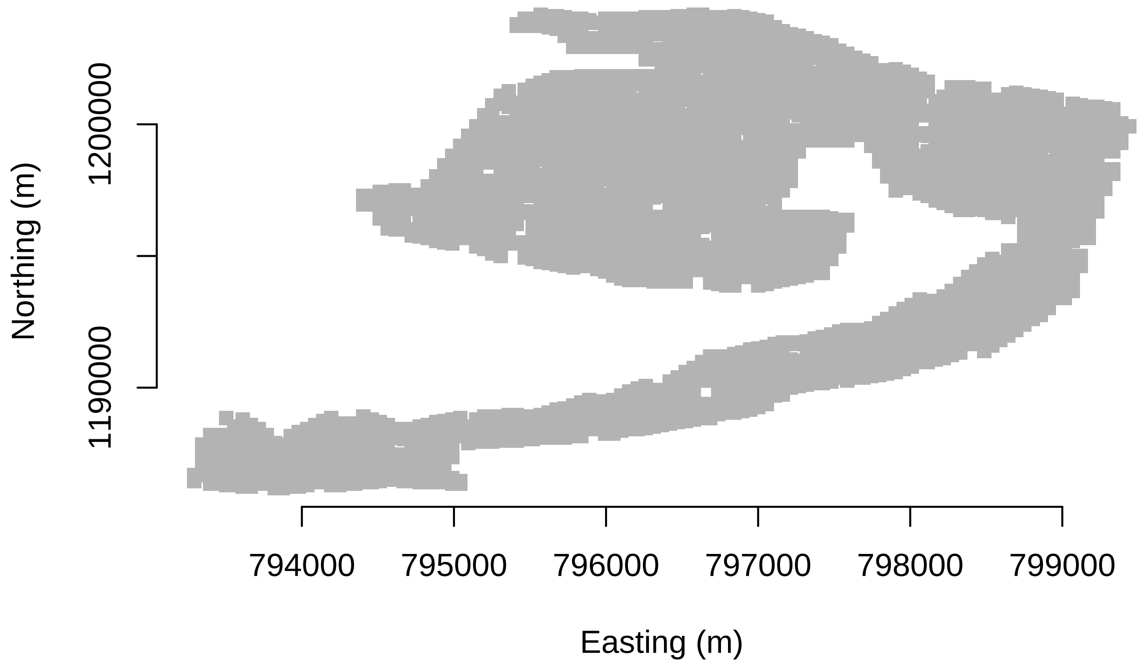
Figure 3: Diagram of the partial survey concept



Each of the Gaussians is a spatial basis function, and its center point is called a knot. We want the knot locations to cover the whole domain but always to have at least one observed seal within their range. For now, let's import the knot locations that I chose in advance. The knot locations are plotted below.

```r
grid_coarse <- read.csv("data/grid_coarse.csv")
grid_med <- read.csv("data/grid_med.csv")
```

```r
plot(pred$x, pred$y, col='grey70', bty='n', xlab="Easting (m)", ylab="Northing (m)", pch=15)
points(grid_coarse, col='green', pch=15)
points(grid_med, col='blue', pch=15)
```



We can now set the bandwidth parameter for the coarse and fine basis functions, and then create the basis expansion. We create the expansion by evaluating each basis function at each photo site.

```r
# set the bandwidths (here called 'r' for range)
r_coarse <- 2000
r_med <- 800

# Set up spatial basis function model matrices:
coarse_mm <- t(apply_basis(photo_sites, grid_coarse, r_coarse, gaussian))
med_mm <- t(apply_basis(photo_sites, grid_med, r_med, gaussian))

# create the total model matrix
basis <- as.data.frame(coarse_mm)
basis <- cbind(basis, med_mm)
names(basis) <- paste0("X", 1:ncol(basis))

head(basis)
```

```
##             X1           X2           X3           X4           X5
## 1 0.0001989239 0.0001821780 0.0001691625 0.0001859884 0.0001691623
## 2 0.0001983706 0.0001660919 0.0001845153 0.0001957217 0.0001845151
## 3 0.0001982777 0.0001656529 0.0001848313 0.0001958861 0.0001848312
## 4 0.0001913200 0.0001461328 0.0001950742 0.0001994588 0.0001950742
## 5 0.0001990445 0.0001814960 0.0001700046 0.0001865889 0.0001700047
## 6 0.0001981807 0.0001652116 0.0001851441 0.0001960469 0.0001851443
##             X6           X7           X8           X9          X10
## 1 8.301441e-05 0.0001060423 0.0002942164 9.942706e-05 0.0004822656
## 2 1.035831e-04 0.0001276554 0.0004217225 2.011399e-04 0.0004897952
## 3 1.041002e-04 0.0001281810 0.0004243649 2.041023e-04 0.0004887559
## 4 1.256849e-04 0.0001493070 0.0004951085 3.360799e-04 0.0004040372
## 5 8.397505e-05 0.0001070827 0.0003008010 1.033695e-04 0.0004848766
## 6 1.046180e-04 0.0001287064 0.0004269671 2.070816e-04 0.0004876572
##            X11          X12          X13          X14          X15
## 1 0.0002942143 9.942589e-05 2.202404e-04 0.0004378798 0.0004822486
## 2 0.0004217215 2.011385e-04 1.122952e-04 0.0003151032 0.0004897818
## 3 0.0004243659 2.041019e-04 1.101965e-04 0.0003118148 0.0004887464
## 4 0.0004951120 3.360807e-04 4.573336e-05 0.0001826399 0.0004040324
## 5 0.0003008045 1.033703e-04 2.141432e-04 0.0004329501 0.0004848747
## 6 0.0004269741 2.070840e-04 1.081244e-04 0.0003085254 0.0004876591
##            X16          X17          X18          X19          X20
## 1 0.0002942017 9.942085e-05 0.0004378579 0.0004822215 0.0002941833
## 2 0.0004217067 2.011298e-04 0.0003150894 0.0004897573 0.0004216830
## 3 0.0004243543 2.040947e-04 0.0003118031 0.0004887250 0.0004243330
## 4 0.0004951023 3.360715e-04 0.0001826342 0.0004040173 0.0004950806
## 5 0.0003008010 1.033682e-04 0.0004329393 0.0004848595 0.0003007897
## 6 0.0004269724 2.070816e-04 0.0003085196 0.0004876469 0.0004269591
##            X21          X22          X23
## 1 1.860992e-05 0.0002941714 9.940966e-05
## 2 5.313482e-05 0.0004216672 2.011091e-04
## 3 5.437178e-05 0.0004243185 2.040755e-04
## 4 1.263599e-04 0.0004950652 3.360432e-04
## 5 1.967605e-05 0.0003007813 1.033605e-04
## 6 5.563241e-05 0.0004269484 2.070680e-04
```

Now we are ready to estimate the model. We use a Poisson regression on the counts in each photo. Since every photo covers the same size area, there's no need to worry about weights here.

```r
basis$cnt <- seal_counts
seal_density_model <- glm(cnt ~ ., family='poisson', data=basis)
```

```
## Warning: step size truncated due to divergence

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: algorithm stopped at boundary value

## Warning: glm.fit: fitted rates numerically 0 occurred
```

```r
summary(seal_density_model)
```

```
##
## Call:
## glm(formula = cnt ~ ., family = "poisson", data = basis)
##
## Deviance Residuals:
##         Min           1Q       Median           3Q          Max
## -4.126e+132   -6.025e+84   -2.786e+36   -1.200e+01    0.000e+00
##
## Coefficients: (21 not defined because of singularities)
##              Estimate Std. Error    z value Pr(>|z|)
## (Intercept)  3.435e+04  2.340e-08  1.468e+12   <2e-16 ***
## X1           1.205e+10  1.584e-04  7.608e+13   <2e-16 ***
## X2                  NA         NA         NA       NA
## X3                      NA          NA          NA        NA
## X4                      NA          NA          NA        NA
## X5                      NA          NA          NA        NA
## X6                      NA          NA          NA        NA
## X7                      NA          NA          NA        NA
## X8                      NA          NA          NA        NA
## X9                      NA          NA          NA        NA
## X10                     NA          NA          NA        NA
## X11                     NA          NA          NA        NA
## X12                     NA          NA          NA        NA
## X13                     NA          NA          NA        NA
## X14                     NA          NA          NA        NA
## X15                     NA          NA          NA        NA
## X16                     NA          NA          NA        NA
## X17                     NA          NA          NA        NA
## X18                     NA          NA          NA        NA
## X19                     NA          NA          NA        NA
## X20                     NA          NA          NA        NA
## X21             -7.735e+04  1.280e-02 -6.042e+06   <2e-16 ***
## X22                     NA          NA          NA        NA
## X23                     NA          NA          NA        NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance:  4.2289e+03  on 520  degrees of freedom
## Residual deviance: 2.3370e+265  on 518  degrees of freedom
## AIC: 2.337e+265
##
## Number of Fisher Scoring iterations: 25
```

And finally we predict the seal density in the raster cells. Here we have to account for the raster cells being 53m x 53m, smaller than the photos which are each 120m x 80m.

```
# evaluate the basis functions at each location
coarse_pm <- t(apply_basis(pred[,1:2], grid_coarse, r_coarse, gaussian))
med_pm <- t(apply_basis(pred[,1:2], grid_med, r_med, gaussian))

# create the predictive model matrix
X_pred <- data.frame(coarse_pm)
X_pred <- cbind(X_pred, med_pm)
names(X_pred) <- paste0("X", 1:ncol(X_pred))

# make predictions and sum them up
link_pred <- predict(seal_density_model, X_pred)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
```

```
popest <- sum(0.053 * 0.053 / (0.12 * 0.08) * exp(link_pred))
```

The estimated population of seals in the bay is ∞.

References

Jansen JK, Brady GM, Ver Hoef JM, Boveng PL (2015) "Spatially Estimating Disturbance of Harbor Seals (*Phoca vitulina*)". PLoS ONE 10(7): e0129798. https://doi.org/10.1371/journal.pone.0129798

Jansen JK, Ver Hoef JM (2014) "Estimating Abundance from Counts in Large Data Sets of Irregularly-Spaced Plots using Spatial Basis Functions". arXiv Preprint 1410.3163. https://arxiv.org/abs/1410.3163