Stat 534 Project:
Extrapolation from Poisson Process Intensity Surface Models

Kenny Flagg

April 3, 2017

1 Introduction

One little-studied application of inhomogeneous point process intensity estimation is the use of an intensity surface estimated from events in a subregion to predict the intensity over the entire region of interest. This procedure would be relevant whenever it is known or suspected that some type of plant, animal, or other item occurs in the region, and the goal of the analysis is to map the trend in where these tend to be located rather than estimate parameters of some process at work across hypothetical replicates of similar regions. It may be prohibitively expensive or difficult to observe all events over the entire region, so a sample of subregions is taken. For example, conservationists want to study the spatial distribution of an endangered plant across a large region of thick jungle so that they can establish a preserve where the plant is protected. They cannot search the entire jungle, so they take a simple random sample of quadrats and record the locations of the plants in those quadrats. They will then want to fit a model describing the trends in intensity of these plants extrapolated over the entire jungle. In this setting, the objective is to map the realized spatial intensity of the plant over this jungle so that these plants can be protected, not to estimate parameters of the process that arranges plants of this species at jungles like this one.

Another situation where it is useful to extrapolate point pattern intensity outside of the observed region is in mapping subsurface geomagnetic anomalies, such as the munitions debris found at former military test ranges. This is frequently done in the early stages of an unexploded ordnance (UXO) remediation, where the intensity surface of inert munitions fragments is used to identify the locations of targets so that the search for UXO can be focused on the sections of the site most likely to contain it. The anomalies are only observed when detection equipment passes directly over them, but the project leaders are concerned with finding UXO and do not want to waste resources finding every inert fragment that could be found by metal detectors. The most common data collection method is to take a systematic sample of straight-line transects and observe anomalies in rectangular regions centered along the transects. Frequently, moving averages of the intensity are computed in circular windows and an intensity map is produced using ordinary kriging to predict the intensity in a grid of these windows. There are several problems with this approach: it assumes stationarity when the moving averages are believed to be non-stationary, it ignores the point process nature of the data, and it is very sensitive to the window size (Flagg 2016; Matzke et al. 2014). In this project, I review the use of polynomial trend surface models for the log-intensity at a simulated UXO site, ultimately concluding that these methods are not yet developed well enough to be useful.

2 Surface Fitting by Maximum Likelihood

It is theoretically possible to use maximum likelihood or maximum pseudo-likelihood methods to fit trend surface models or regression models (with covariates) for the intensity of an inhomogeneous point process, but these are not widely used because, in the most general cases, the problems are "notoriously intractable" (Diggle 2013). However, for spatial Poisson processes, trend surface models are tenable with the help of some numerical methods.

The log-likelihood of an unmarked Poisson process on a region D with intensity $\lambda(\mathbf{s})$ is found by conditioning on the number of events in the following manner. The number of events in D follows a Poisson distribution with mean $\mu = \int_D \lambda(\mathbf{s})d\mathbf{s}$. Given that n events occurred, their locations $\mathbf{s}_1, \ldots, \mathbf{s}_n$ are independent and identically distributed with density $\lambda(\mathbf{s})/\mu$. Then the log-likelihood of the intensity is

$$\begin{aligned}\ell(\lambda) &= \{-\mu + n\log(\mu) - \log(n!)\} + \sum_{i=1}^{n}\{\log(\lambda(\mathbf{s}_i)) - \log(\mu)\} \\ &= \sum_{i=1}^{n}\log(\lambda(\mathbf{s}_i)) - \int_D \lambda(\mathbf{s})d\mathbf{s} - \log(n!).\end{aligned}$$

A natural way to model the intensity is to define a log-linear model

$$\log(\lambda(\mathbf{s})) = \mathbf{x}(\mathbf{s})^T\boldsymbol{\beta}$$

where $\mathbf{x}(\mathbf{s})$ can, in principle, include functions of the spatial coordinates and also covariates. Unfortunately, the likelihood becomes

$$\sum_{i=1}^{n}\mathbf{x}(\mathbf{s})^T\boldsymbol{\beta} - \int_D \exp\left(\mathbf{x}(\mathbf{s})^T\boldsymbol{\beta}\right)d\mathbf{s} - \log(n!)$$

so $\mathbf{x}(\mathbf{s})$ must be known (or predicted) either across the entire region D or at enough locations to approximate the integral numerically. Thus, covariates require some extra data collection or modeling effort to incorporate, but trend surfaces are feasible.

Berman and Turner (1992) proposed a practical method for fitting these models with generalized linear model (GLM) software using quadratic approximations to the likelihood, even generalizing to other link functions besides the log link. Their method is based on partitioning D such that each subset in the partition contains at most one event, and then maximizing a weighted pseudolikelihood based upon a binomial or Poisson distribution for the count of events in each subset. They comment that the covariance matrices produced by standard GLM software under this method are reasonable approximations to the correct covariance matrices that would be derived from the true likelihood as long as the quadrature approximation of the likelihood converges to the true likelihood as the resolution of the partition is increased. This was followed up by Baddeley et al. (2014), who developed a logistic regression-based approach that avoids the quadratic approximation, thereby reducing bias in the coefficient estimates. Both methods are available in the `spatstat` package (Baddeley, Rubak, and Turner 2015) for R (R Core Team 2017). The `spatstat` function `ppm` implements these methods using the base R `glm` function to maximize the likelihood; I will use the Berman and Turner method because it is the default in `ppm` and therefore better reflects the experience of a naïve practitioner who is not an expert on spatial point processes.

3 True Log-Linear Intensity Surface Example

Before moving on to a realistic example UXO site, I present a simple toy example using a true log-linear intensity surface to illustrate model fitting and prediction outside the observed region. Previously this semester, we worked with an example of a Poisson process on the unit square $\{(x, y) : 0 < x < 1, 0 < y < 1\}$ with the log-linear intensity surface

$$\log(\lambda(x, y)) = 5x + 2y.$$

I will continue this example, fitting the model

$$\log(\lambda(x, y)) = \beta_0 + \beta_1 x + \beta_2 y$$

first using the process observed over the full region (the entire unit square), and then using the process observed in a subregion, treating $\{(x, y) : 0.6 < x < 0.9, 0.6 < y < 0.9\}$ as unobserved. This unobserved section is a region of relatively high intensity, so if the estimation or prediction methods are sensitive to the observation window I expect omitting this section to have a large effect.

I will use both estimated models to predict on $\{(x, y) : -0.5 < x < 1.5, -0.5 < y < 1.5\}$. Figure 1 shows the realization of this process that I will use for both model fits.

For clarity, I denote the estimated models as

$$\log\left(\widehat{\lambda}_f(x, y)\right) = \widehat{\beta}_0^{(f)} + \widehat{\beta}_1^{(f)} x + \widehat{\beta}_2^{(f)} y$$

for the model fit to the full region, and

$$\log\left(\widehat{\lambda}_s(x, y)\right) = \widehat{\beta}_0^{(s)} + \widehat{\beta}_1^{(s)} x + \widehat{\beta}_2^{(s)} y$$

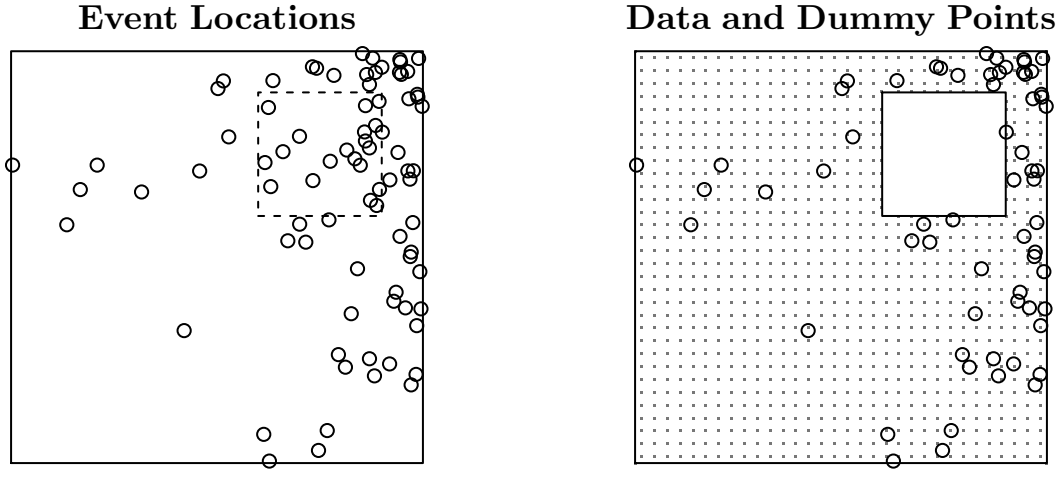

Figure 1: One realization of a Poisson process with log-linear intensity on the unit square. The left panel shows all of the event locations, with the dashed square outlining of the "unobserved" region. The right panel shows the events and dummy points used when fitting the model to the subregion; the small square is not considered in constructing the approximation to the likelihood, and the points inside it are discarded.

	Estimate	S.E.
$\widehat{\beta}_0^{(f)}$	0.16	0.54
$\widehat{\beta}_1^{(f)}$	4.61	0.58
$\widehat{\beta}_2^{(f)}$	2.02	0.42

Table 1: Estimated coefficients for the log-linear trend model fit using the full region.

	Estimate	S.E.
$\widehat{\beta}_0^{(s)}$	0.20	0.56
$\widehat{\beta}_1^{(s)}$	4.54	0.59
$\widehat{\beta}_2^{(s)}$	2.00	0.44

Table 2: Estimated coefficients for the log-linear trend model fit using the subregion.

for the model fit to the subregion. I use `ppm` to fit the models with its default likelihood approximation and isotropic edge correction. The coefficients and standard errors are similar for both estimated models, and the true values $\beta_0 = 0$, $\beta_1 = 5$, and $\beta_2 = 2$ all fall within one standard error of the estimates (Tables 1 and 2).

After estimating the models, I use `spatstat`'s `predict.ppm` method to predict the intensity surface on a 128×128 lattice of locations in the region $\{(x, y) : -0.5 < x < 1.5, -0.5 < y < 1.5\}$ (Figure 2). As expected from the similarity of the coefficient estimates, the predicted log-intensity surfaces for both models show very similar linear trends (top row of the figure). At a glance the prediction standard errors for both models appear similar to the predicted intensities, as seen in the middle row of the figure (shown on a logarithmic scale for visibility). This makes intuitive sense because higher true point process intensity implies greater variance in the number of points, which should lead to greater uncertainty in estimation or prediction of the intensity. In a linear model setting we would expect the prediction standard error to increase with the distance from the observations, but no such trend is apparent here; even in the bottom left of the prediction region the standard errors decrease as the predicted intensity decreases. However, it is illuminating to plot the predictor's relative standard error (the ratio of the standard error to the predicted intensity, bottom row of the figure). This ratio is not constant, and in fact it is lowest where the highest intensity of events was observed.

In summary, the estimation procedure does an adequate job of estimating the parameters of the true model for these data, and the standard error of the predicted intensity appears to be more strongly related to the distance from the observed events than to the distance from the observed region.

4 Simulated UXO Site

In my writing project, I evaluated ordinary kriging of moving average intensity values on a simulated tank training range. I will use the same site for this project. The site is a 952.38 acre quadrilateral region with homogeneous Poisson process producing background anomalies with an intensity of 100 anomalies per acre, and two regions of concentrated munitions use generated as Poisson processes

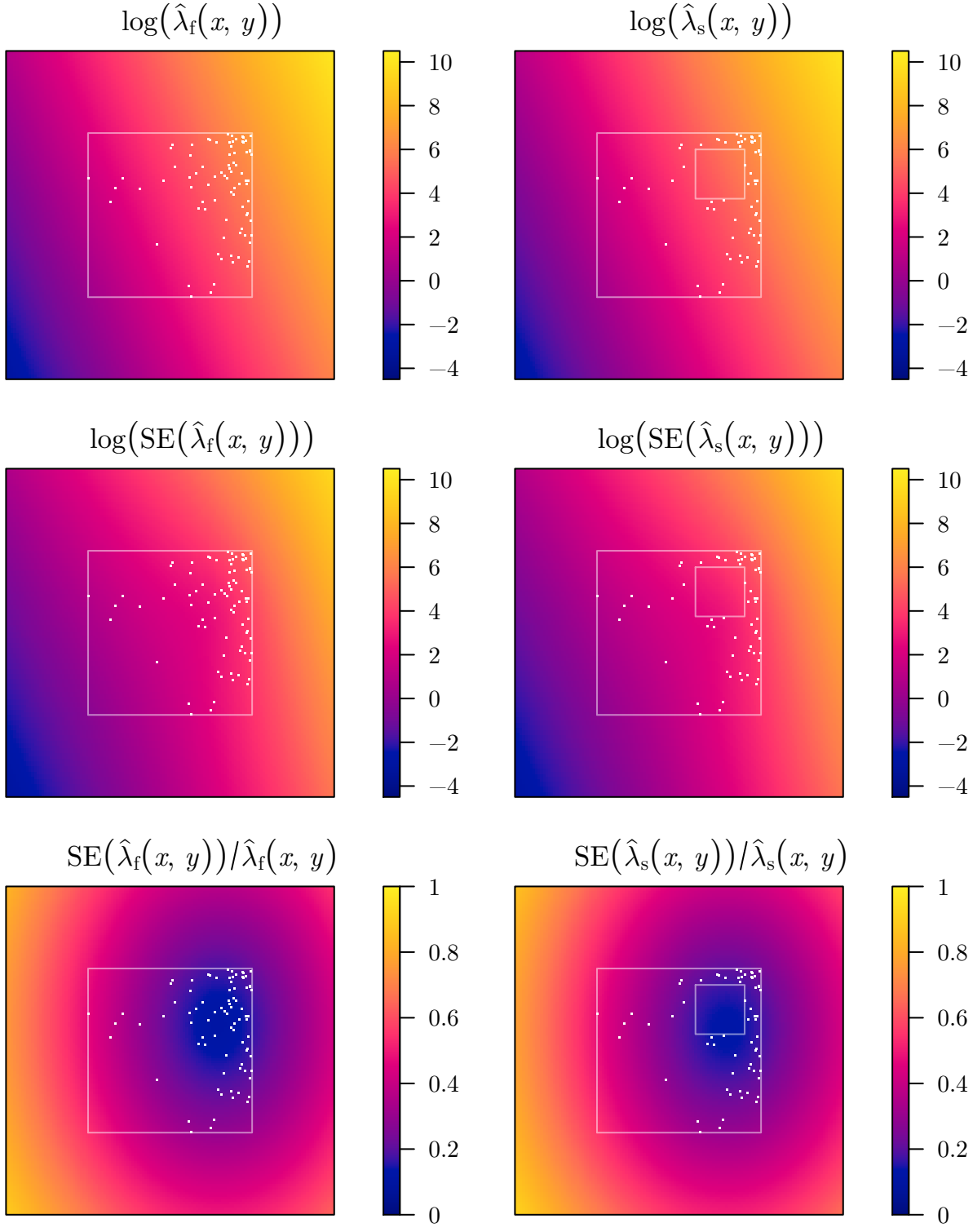

Figure 2: Estimated log-intensity surface, log-scale prediction standard errors, and relative standard errors from models fit using all events in the full simulation region (left) and events in a subset of the simulation region (right). The regions and event locations are overlaid in white.

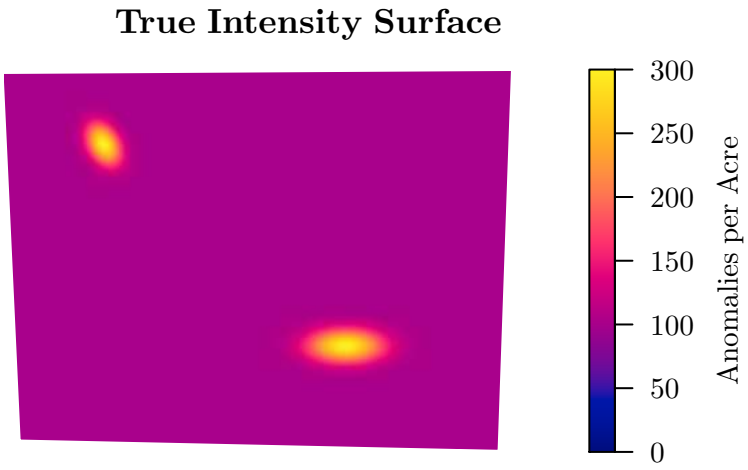

Figure 3: True intensity at the simulated UXO site.

with bivariate Gaussian intensity. The total intensity at the center of each concentrated munitions use region is 300 anomalies per acre (Figure 3).

I simulate data collection along parallel straight-line transects spaced 396 feet apart using a detector with a footprint six feet wide (so there is an unobserved region 390 feet wide between each pair of adjacent transects). The location of the first transect is selected randomly, and the locations of all anomalies within three feet of a transect are recorded (Figure 4). In total, 14.7 acres are observed, covering 1.5% of the region of interest.

If the general structure of the site (such as the number and shape of the high-intensity regions and the type of background process) is known then the most appropriate model for the anomaly intensity could be a mixture model or a trend surface with a particular form, but in practice these details are rarely available. Therefore, practitioners prefer flexible models without defined structure. I fit p-degree polynomial models of the form

$$\log\left(\lambda\left(x,y\right)\right) = \sum_{i=0}^{p}\sum_{j=0}^{p-i}\beta_{ij}x^{i}y^{j}$$

for $p = 2, 3, \ldots, 12$ using 500,000 dummy points randomly selected from a uniform distribution over the observed region. The coordinates x and y are standardized to have mean 0 and variance 1 in order to reduce numerical instability for the larger exponents.

Figure 5 shows the predicted intensity surfaces for the polynomial models, along with a kernel density estimate for comparison. 500,000 dummy points do not seem to be enough for the likelihood approximation to converge to the Poisson likelihood, but the predicted intensity values are reasonable (discussed further in the next section). Because of the lack of convergence, I do not do any formal model comparisons.

The second- and third-order polynomial models are too simple to capture the bimodality of the true surface, but the fourth- and higher-order models have large, high-intensity regions that coincide with both concentrated munitions use regions of the true surface. Up until degree nine, the polynomial models are smoother than the kernel intensity estimate. The ninth-, tenth-, and eleventh-degree

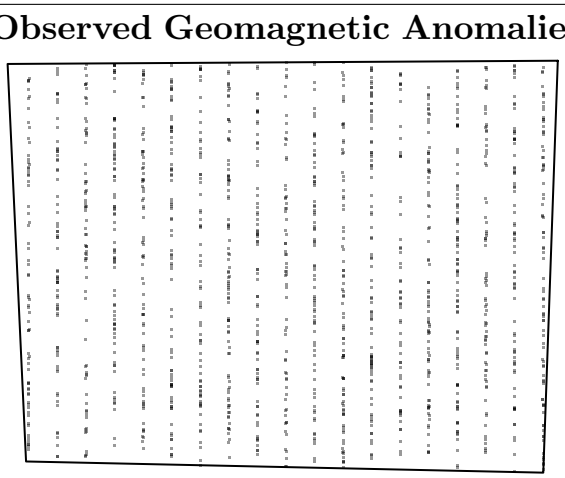

Figure 4: Observed geomagnetic anomalies from one realization of the UXO site. The transects are not shown on this plot because they would obscure the points.

models all look similar to the kernel intensity estimate, while the twelfth-degree model is noticeably noisier. Edge effects are noticeable for $p \geq 4$ with some predictions along the boundary abruptly dropping to zero or increasing to hundreds of anomalies per acre higher than any of the predictions in the interior of the region.

The kernel density estimate is computed using `spatstat`'s default "simple rule of thumb" to select the smoothing bandwidth. Because the anomalies located more than three feet from a transect were not observed, I would expect the kernel density estimate to show ridges of high intensity along the transects. As it turns out, the default bandwidth is large enough to smooth across multiple transects so the ridges do not appear. However, the kernel density estimate is biased low because not all anomalies are observed; I divide the estimate by the proportion of the area observed as an approximate bias correction and the result is a surprisingly reasonable intensity map.

Figure 6 shows the log-scale standard errors for six of the polynomial models. For all models except the twelfth-degree model, the standard error is low across most of the site but high along the boundary. The standard errors increase with the degree of the model. This is expected and reasonable because locations in the interior of the region have the most nearby anomalies and dummy points, and predictions from more complex models depend on more estimated parameters.

If the true number of high-intensity regions was known to be 2, any polynomial model with $p \geq 4$ (or the kernel density estimate) would provide an accurate (if imprecise) indication of where to focus the UXO remediation effort. However, the plots alone provide little indication of which model has the ideal level of complexity, but there are several practical difficulties with fitting these models that should be sorted out before I would be comfortable making recommendations about how to select an optimal model for prediction outside the observed subregion.

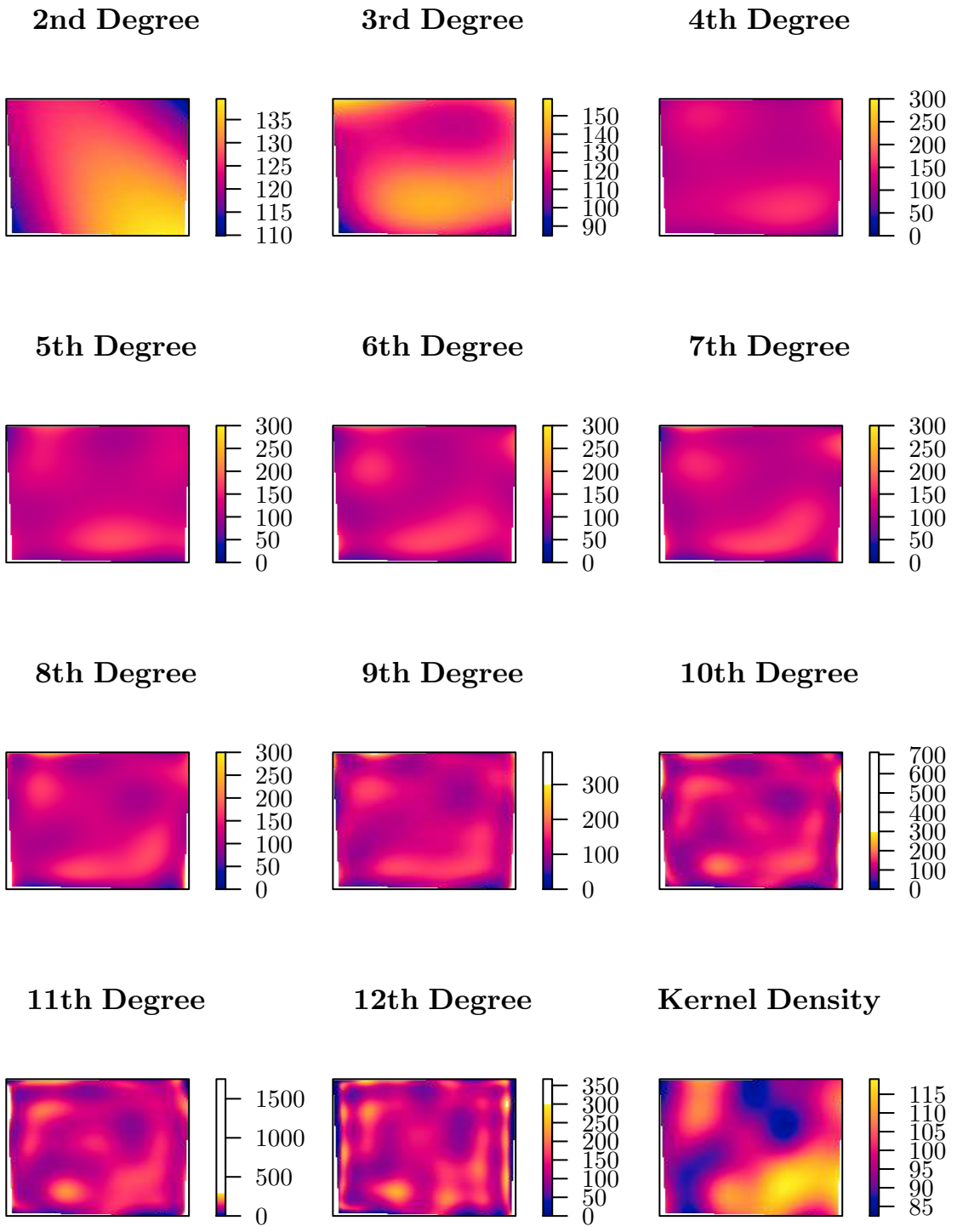

Figure 5: Predicted intensity surfaces for eleven polynomial models, and a kernel density estimate for comparision. Note that the 4th- through 12th-degree polynomial models have the same color scale, while the 2nd- and 3rd-degree models and the kernel density estimate use different color scales for visibility.

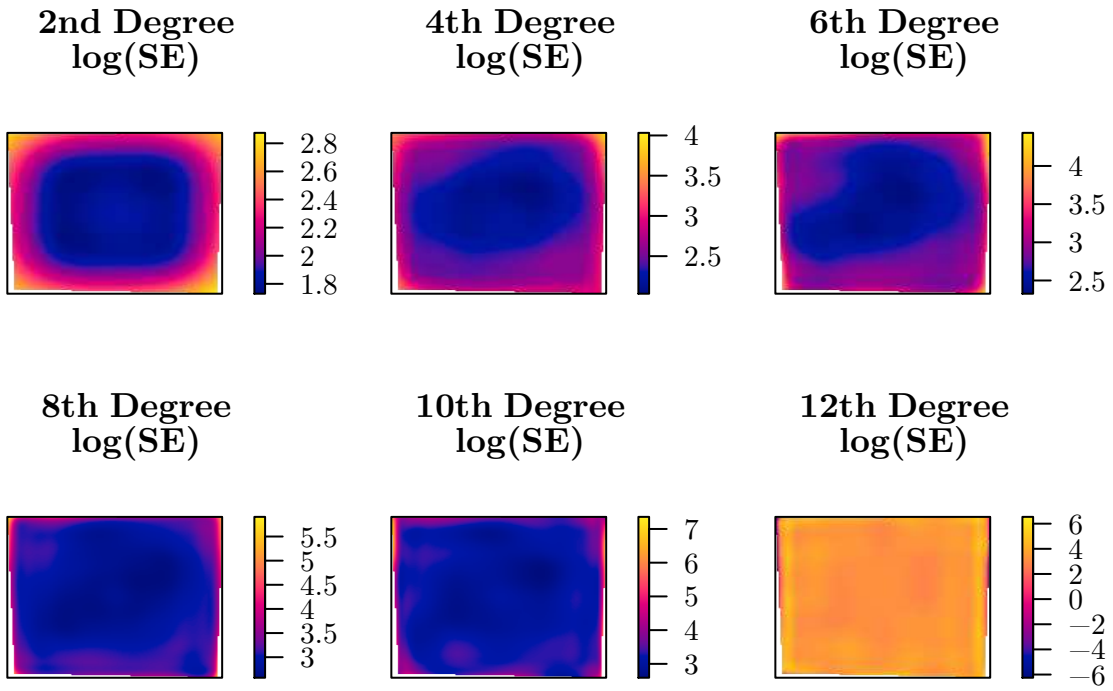

Figure 6: Log standard errors for six of the polynomial models.

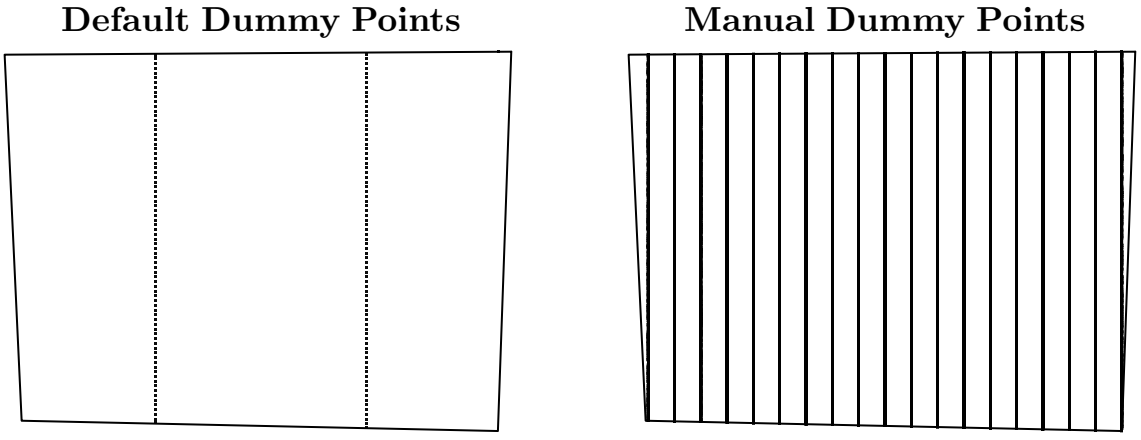

Figure 7: The default dummy points selected by `spatstat` (left) and the 500,000 random uniform dummy points that I use to fit the models.

5 Difficulties with Implementation

I encountered many difficulties in fitting these models that suggest the methods and software are not yet developed well enough to make them useful for working with such sparsely sampled data. Most of these difficulties resulted from the implementation with `spatstat`, but several were computational issues.

The troubles with `spatstat` largely come from how the package sets up grids of locations for dummy points and predictions. It finds a bounding box for the region and places grid cells on a lattice across the bounding box, but its grids are so coarse that most of my transects do not have cells centered on them. It attempts to create an 80×80 grid of dummy points, but the only usable points are 160 points on two transects; all others are discarded (Figure 7, left). Attempting to fit models in `ppm` using these dummy points results in an error.

The `ppm` function fits models successfully when using the 500,000 dummy points that I draw from a uniform distribution (Figure 7, right). However, the `plot` method gives an error about infinite values, while the `predict` method by default uses the same 80×80 grid used for the default dummy points and makes predictions only in the 160 grid cells that happen to lie inside the observed region. Fortunately, the `predict` function behaves when the whole site is specified as the prediction window.

The biggest disappointment of using `spatstat` to fit these models is that models with splines cannot be plotted. The `ppm` functions estimates the parameters without any apparent trouble, but the `plot` and `predict` functions give errors because they do not know how to interpret the smooth terms in the model formula.

All of the above difficulties are software issues that could be fixed. I also found two computational limitations that will not be remedied as easily. First, even after rescaling the coordinates, $\mathbf{X}^T\mathbf{X}$ cannot be inverted because of numerical instabilities when $p \geq 18$. Such models are more complicated than needed for my example site, but it is important to note that a limit on polynomial model complexity exists.

Second, choosing dummy points is hard. I initially used 1,000 points, but that was not enough and the intensity values predicted by those models were in the tens of thousands. It is not apparent that the 500,000 randomly-selected dummy points I ultimately used are optimal, either. Warton and Shepherd (2010) discuss dummy point selection when the entire region of interest is observed, suggesting that the number of dummy points be increased until the maximized log-likelihood reaches approximate convergence. They use 86,227 points for their example. I used trial-and-error to increase the number of dummy points, fitting a couple of my models using 600,000 and then 1,000,000 dummy points; the maximized log-likelihood changed by thousands. I settled on 500,000 because that was the smallest number I tried that yielded predicted intensities of the correct magnitude. With that many points, each model takes about a minute to fit; models with enough dummy points to accurately approximate the likelihood would likely take much longer to estimate.

6 Discussion and Conclusion

Spatial point process intensity surface models are an intriguing option for modeling partially-observed inhomogeneous point processes, and polynomial models can approximate complex surfaces. However, these models are not yet ready for application to UXO data because the software is not quite there (yet) and they require heavy computing power. Several avenues for future work are available, such as seeking more efficient ways to approximate the likelihood and compute parameter estimates, and further exploring how to choose dummy points. Then, the next step might be to study whether familiar model selection tools like AIC or tests of predictive ability are suitable assuming the likelihood approximation is successful. Presently, it seems necessary either to observe more of the site area than covered by typical UXO sampling plans, or work with a model fit that is a rough approximation at best. Thus the biggest open question might be how much of the region do you need to observe to trust your model?

References

Baddeley, Adrian, Ege Rubak, and Rolf Turner (2015). *Spatial Point Patterns: Methodology and Applications with R*. London: Chapman and Hall/CRC Press.

Baddeley, Adrian et al. (2014). "Logistic regression for spatial Gibbs point processes". In: *Biometrika* 101.2, pp. 377–392.

Berman, Mark and Rolf Turner (1992). "Approximating point process likelihoods with GLIM". In: *Applied Statistics*, pp. 31–38.

Diggle, Peter J. (2013). *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. 3rd ed. CRC Press.

Flagg, KA (2016). "Visual Sample Plan and Unexploded Ordnance: What do we need to know to find UXO?" M.S. writing project. Montana State University, Bozeman. URL: `https://github.com/kflagg/vspuxo`.

Matzke, Brett et al. (2014). *Visual Sample Plan Version 7.0 User's Guide*. Pacific Northwest National Laboratory. Richland, Washington. URL: `http://vsp.pnnl.gov/docs/PNNL-23211.pdf`.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

Warton, David I and Leah C Shepherd (2010). "Poisson point process models solve the "pseudo-absence problem" for presence-only data in ecology". In: *The Annals of Applied Statistics* 4.3, pp. 1383–1402.

R Code Appendix

```
library(spatstat)

set.seed(83534)

simulate_win <- owin(c(0, 1), c(0, 1))
omit_win <- owin(c(0.6, 0.9), c(0.6, 0.9))
observe_win <- setminus.owin(simulate_win, omit_win)
predict_win <- owin(c(-0.5, 1.5), c(-0.5, 1.5))
hw4 <- rpoispp(function(x, y){exp(5 * x + 2 * y)}, win = simulate_win)

hw4_fit_f <- ppm(hw4 ~ x + y, correction = 'isotropic')
hw4_pred_f <- predict(hw4_fit_f, window = predict_win, se = TRUE)
# By default predictions are on a 128 by 128 grid.

hw4_fit_s <- ppm(hw4[observe_win] ~ x + y, correction = 'isotropic')
hw4_pred_s <- predict(hw4_fit_s, window = predict_win, se = TRUE)

par(mfrow = c(1, 2), mar = c(0, 0, 1, 0), cex = 1)
plot(hw4, main = 'Event Locations')
plot(omit_win, lty = 2, add = TRUE)
plot(hw4_fit_s$Q, main = 'Data and Dummy Points')

par(mfrow = c(3, 2), mar = c(1, 0, 1.2, 2), las = 2, cex = 1)

plot(log(hw4_pred_f$estimate), zlim = c(-4.5, 10.5),
     main = expression(log(hat(lambda)[f](italic(list(x,y))))))
plot(simulate_win, add = TRUE, border = '#ffffff80')
points(hw4, pch = '.', col = 'white')

plot(log(hw4_pred_s$estimate), zlim = c(-4.5, 10.5),
     main = expression(log(hat(lambda)[s](italic(list(x,y))))))
plot(observe_win, add = TRUE, border = '#ffffff80')
points(hw4[observe_win], pch = '.', col = 'white')

plot(log(hw4_pred_f$se), zlim = c(-4.5, 10.5),
     main = expression(log(SE(hat(lambda)[f](italic(list(x,y)))))))
plot(simulate_win, add = TRUE, border = '#ffffff80')
points(hw4, pch = '.', col = 'white')

plot(log(hw4_pred_s$se), zlim = c(-4.5, 10.5),
     main = expression(log(SE(hat(lambda)[s](italic(list(x,y)))))))
plot(observe_win, add = TRUE, border = '#ffffff80')
points(hw4[observe_win], pch = '.', col = 'white')

plot(hw4_pred_f$se/hw4_pred_f$estimate, zlim = c(0, 1),
     main = expression(SE(hat(lambda)[f](italic(list(x,y)))) /
                         hat(lambda)[f](italic(list(x,y))))))
plot(simulate_win, add = TRUE, border = '#ffffff80')
points(hw4, pch = '.', col = 'white')

plot(hw4_pred_s$se/hw4_pred_s$estimate, zlim = c(0, 1),
     main = expression(SE(hat(lambda)[s](italic(list(x,y)))) /
                         hat(lambda)[s](italic(list(x,y))))))
plot(observe_win, add = TRUE, border = '#ffffff80')
points(hw4[observe_win], pch = '.', col = 'white')
```

```
# Define the window for the UXO site.
site_window <- owin(poly = cbind(x = c(1564294, 1564495, 1556870, 1557126),
                                 y = c(535421, 541130, 541085, 535576)),
                    unitname = c('foot', 'feet'))

# Bivariate normal intensity function that I originally wrote for my
# writing project, parameterized in terms of the major and minor axes scales
# and the angle of the major axis from horizontal.
#   a is horizontal axis, b is vertical axis, r is rotation angle
gauss.elliptic <- function(x, y, mu.x = 0, mu.y = 0, s.a = 1, s.b = 1,
                           r = 0, maxrate = 1){
  rot <- zapsmall(matrix(c(cos(r), sin(r), -sin(r), cos(r)), nrow = 2))
  ab <- diag(c(s.a^2, s.b^2))
  sigma <- rot %*% ab %*% t(rot)
  siginv <- solve(sigma)
  mu <- matrix(c(mu.x, mu.y), nrow = 2)
  mat <- matrix(rbind(x, y), nrow = 2)
  return(maxrate * apply(mat, 2, function(vec){
      exp(-t(vec - mu) %*% siginv %*% (vec - mu) / 2)
    }))
}

# Create an image of the surface in 20 ft by 20 ft pixels.
x <- seq(1556880, 1564495, by = 20)
y <- seq(535431, 541130, by = 20)
intense.mat <- matrix(100, nrow = length(x), length(y))
for(i in seq_along(x)){
  for(j in seq_along(y)){
    intense.mat[i, j] <- 100 + gauss.elliptic(x[i], y[j],
        mu.x = 1558400, mu.y = 540000,
        s.a = 800 / (2 * qnorm(0.995)), s.b = 1200 / (2 * qnorm(0.995)),
        r = pi/6, maxrate = 200
      ) + gauss.elliptic(x[i], y[j],
        mu.x = 1562000, mu.y = 537000,
        s.a = 2000 / (2 * qnorm(0.995)), s.b = 900 / (2 * qnorm(0.995)),
        r = 0, maxrate = 200
      )
  }
}
intense.im <- im(t(intense.mat), x, y, unitname = c('foot', 'feet'))

par(mar = c(0, 0, 1, 3))
plot(intense.im, main = 'True Intensity Surface',
     border = NA, zlim = c(0, 300), las = 2)
plot(setminus.owin(owin(c(1556870, 1564495), c(535421, 541130),
                        unitname = c('foot', 'feet')),
     site_window), col = 'white', border = 'white', lwd = 2, add = TRUE)
mtext('Anomalies per Acre', 4, line = 2)

# Load the sample data used in my writing project.
# The anomaly file contains the locations of the observed anomalies and the
# cog file contains the course-over-ground waypoints on the transects.
ex_anom <- read.csv('easy_sample_tTA2_p200_bg100_fg200_rep2000.anomaly')
ex_path <- read.csv('easy_sample_tTA2_p200_bg100_fg200_rep2000.cog')

# Create a spatstat window and ppp object.
```

```
ex_win <- intersect.owin(site_window,
  do.call(union.owin, lapply(unique(ex_path$x), function(x){
    return(owin(c(x - 3, x + 3), c(535421, 541130),
           unitname = c('foot', 'feet')))
  }))
ex_ppp <- ppp(ex_anom$x, ex_anom$y, window = ex_win)

# Create a ppp object that uses the full region as the window.
ex_ppp_full <- ppp(ex_anom$x, ex_anom$y, window = site_window)

par(mar = c(0, 0, 1, 0))
plot(ex_ppp, main = 'Observed Geomagnetic Anomalies', pch = '.', border = NA)
plot(site_window, add = TRUE)

# Do a simple fit where spatstat sets up the quadrature so we can see where it
# puts the dummy points.
ex_fit1 <- ppm(ex_ppp ~ x + y)

# Create a quadrature scheme manually to ensure there are dummy points on
# each transect.
ex_Q <- quad(data = ex_ppp, dummy = runifpoint(500000, Window(ex_ppp)))

ex_fit2 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 2), correction = 'iso')
ex_fit3 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 3), correction = 'iso')
ex_fit4 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 4), correction = 'iso')
ex_fit5 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 5), correction = 'iso')
ex_fit6 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 6), correction = 'iso')
ex_fit7 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 7), correction = 'iso')
ex_fit8 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 8), correction = 'iso')
ex_fit9 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 9), correction = 'iso')
ex_fit10 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 10), correction = 'iso')
ex_fit11 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 11), correction = 'iso')
ex_fit12 <- ppm(ex_Q, ~ polynom(scale(x), scale(y), 12), correction = 'iso')

# Get predictions and SEs.
ex_p2 <- predict(ex_fit2, window = site_window, se = TRUE)
ex_p3 <- predict(ex_fit3, window = site_window, se = TRUE)
ex_p4 <- predict(ex_fit4, window = site_window, se = TRUE)
ex_p5 <- predict(ex_fit5, window = site_window, se = TRUE)
ex_p6 <- predict(ex_fit6, window = site_window, se = TRUE)
ex_p7 <- predict(ex_fit7, window = site_window, se = TRUE)
ex_p8 <- predict(ex_fit8, window = site_window, se = TRUE)
ex_p9 <- predict(ex_fit9, window = site_window, se = TRUE)
ex_p10 <- predict(ex_fit10, window = site_window, se = TRUE)
ex_p11 <- predict(ex_fit11, window = site_window, se = TRUE)
ex_p12 <- predict(ex_fit12, window = site_window, se = TRUE)

ex_kde <- density(ex_ppp_full) * area(ex_ppp_full) / area(ex_ppp)

par(mfrow = c(4, 3), mar = c(1, 0, 1.2, 3), las = 2, cex = 1)

# Multiply predictions by 43560 to convert from anomalies per square foot
# to anomalies per acre.
plot(43560 * ex_p2$estimate, main = '2nd Degree')
plot(43560 * ex_p3$estimate, main = '3rd Degree')
plot(43560 * ex_p4$estimate, zlim = c(0, 300), main = '4th Degree')
```

```
plot(43560 * ex_p5$estimate, zlim = c(0, 300), main = '5th Degree')
plot(43560 * ex_p6$estimate, zlim = c(0, 300), main = '6th Degree')
plot(43560 * ex_p7$estimate, zlim = c(0, 300), main = '7th Degree')
plot(43560 * ex_p8$estimate, zlim = c(0, 300), main = '8th Degree')
plot(43560 * ex_p9$estimate, zlim = c(0, 300), main = '9th Degree')
plot(43560 * ex_p10$estimate, zlim = c(0, 300), main = '10th Degree')
plot(43560 * ex_p11$estimate, zlim = c(0, 300), main = '11th Degree')
plot(43560 * ex_p12$estimate, zlim = c(0, 300), main = '12th Degree')
plot(43560 * ex_kde, main = 'Kernel Density')

par(mfrow = c(2, 3), mar = c(1, 0, 2, 3), las = 2, cex = 1)
plot(log(43560 * ex_p2$se), main = '2nd Degree\nlog(SE)')
plot(log(43560 * ex_p4$se), main = '4th Degree\nlog(SE)')
plot(log(43560 * ex_p6$se), main = '6th Degree\nlog(SE)')
plot(log(43560 * ex_p8$se), main = '8th Degree\nlog(SE)')
plot(log(43560 * ex_p10$se), main = '10th Degree\nlog(SE)')
plot(log(43560 * ex_p12$se), main = '12th Degree\nlog(SE)')

par(mfrow = c(1, 2), mar = c(1, 0, 1, 0), cex = 1)
plot(ex_fit1$Q$dummy, pch = '.', border = NA,
     main = 'Default Dummy Points')
plot(site_window, add = TRUE)
plot(ex_Q$dummy, pch = '.', border = NA,
     main = 'Manual Dummy Points')
plot(site_window, add = TRUE)
```