

# Pine analysis

Ben Bolker, Gabe Herrick, Gordon Fox

June 26, 2011

## 1 To do

- double-check/figure out what's going on with seedling fits ...
- (perhaps) spend more time figuring out the mapping of observed means and variances to simulation parameters: seed (lognormal-Poisson) distribution is OK, seedling (logit-normal-binomial superimposed on this distribution) is klugey (but maybe fine)
- do fits/models for simulations (show theoretical and observed)
- write the damn paper!

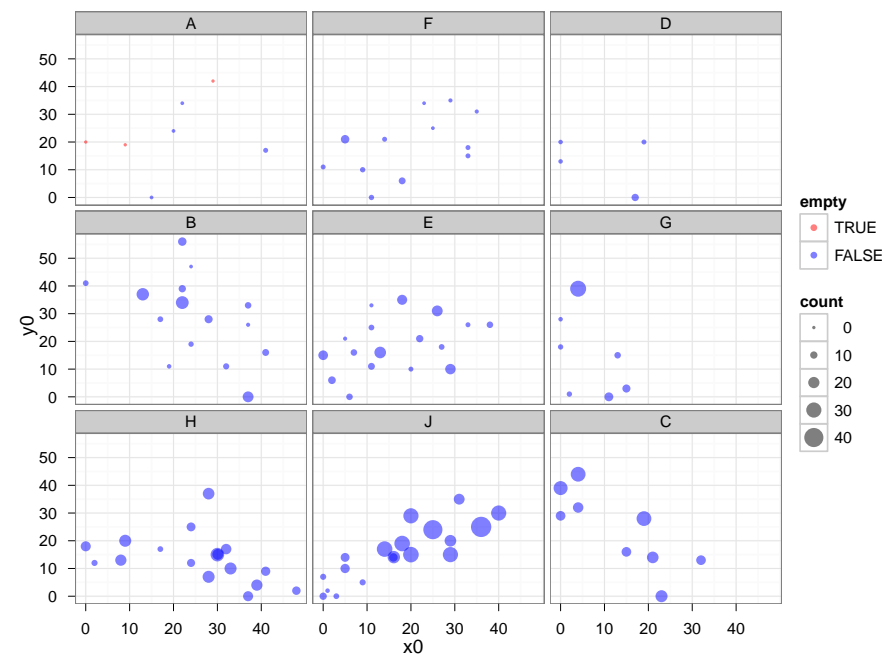
## 2 Seeds

```
> source("pine-funs.R") ## miscellaneous utilities
```

Read data (as before, although this is now really shown for illustration: the real fits were done in a batch run elsewhere):

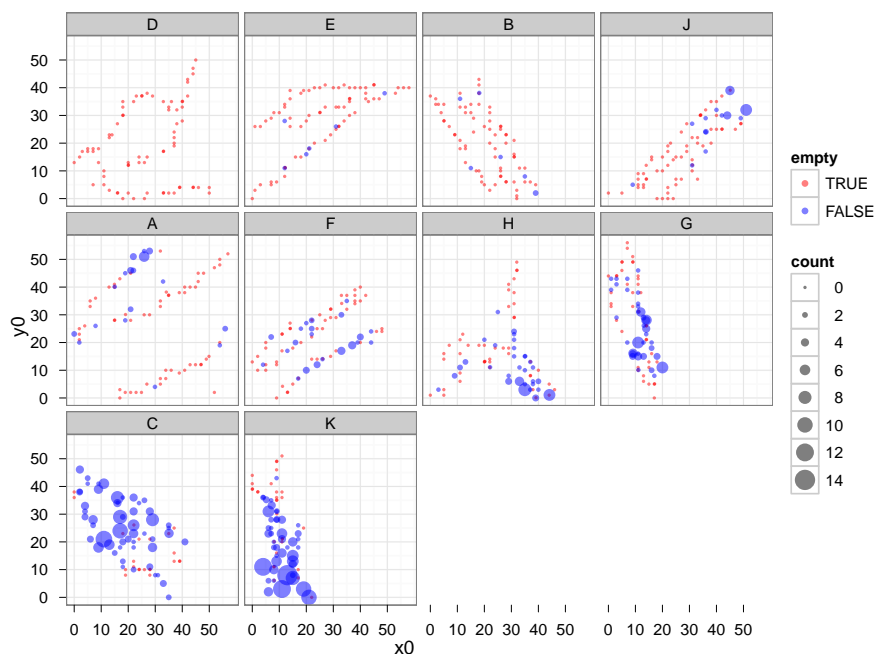
```
> trapdata <- read.table("SeedsUTM-2.txt", header = TRUE)
> mc <- function(x) { x-min(x,na.rm=TRUE) }
> dmc <- function(z) { within(z, {x0 <- mc(x); y0 <- mc(y)}) }
> seeds <- with(trapdata,
  data.frame(x=mc(LONG),y=mc(LAT),count=Count,
    plot=toupper(Plot)))
> ## compute plot-corrected values
> seeds <- ddply(seeds,"plot",dmc)
> transdata <- read.table("LingsUTM-3.txt", header = TRUE)
> seedlings <- with(transdata,
  data.frame(x=mc(Long),y=mc(Lat),count=Lcounts,
    plot=factor(substr(as.character(StAndId),1,1))))
> ## compute plot-corrected values
> seedlings <- ddply(seedlings,"plot",dmc)
> save("seeds","seedlings",file="pine2.RData")
```

Plot seed data (plots ordered by mean number of seeds per samples):



Plot seedling data (plots ordered by mean seedlings):

[1] "D" "E" "B" "J" "A" "F" "H" "G" "C" "K"



Need some packages:

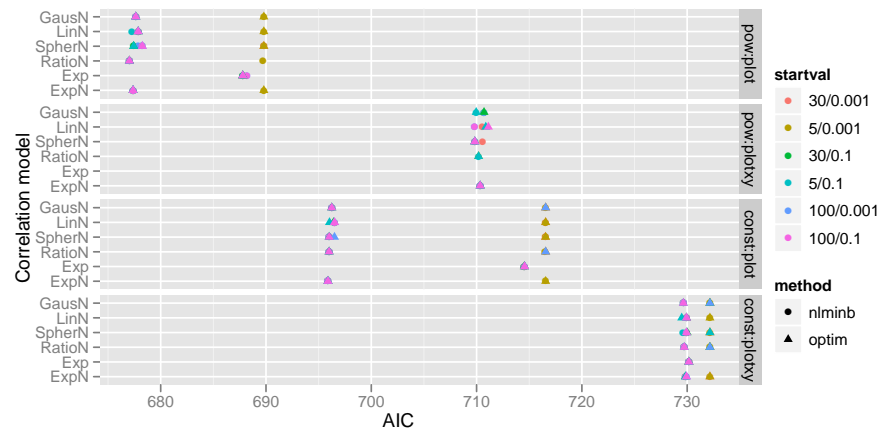
```
> library(ggplot2)
> library(gridExtra)
> library(gdata) ## suppress informational messages??
```

I did a bunch of brute-force fits of a variety of models (exponential without nugget; {exponential, Gaussian, rational, linear, spherical} (i.e. all possibilities in base `nlme`) with nugget)  $\times$  (a variety of starting points for the nugget and range parameters)  $\times$  different internal optimizer choices (`nlminb`, `optim`)  $\times$  (allowing, or not, for the variance to scale as a power of the mean)  $\times$  (allowing, or not, for linear trends within each plot) — a total of 288 model fits. (Treated plot as a fixed effect in all these cases because previous work suggested that this choice dominates either random-effects or pooled models.)

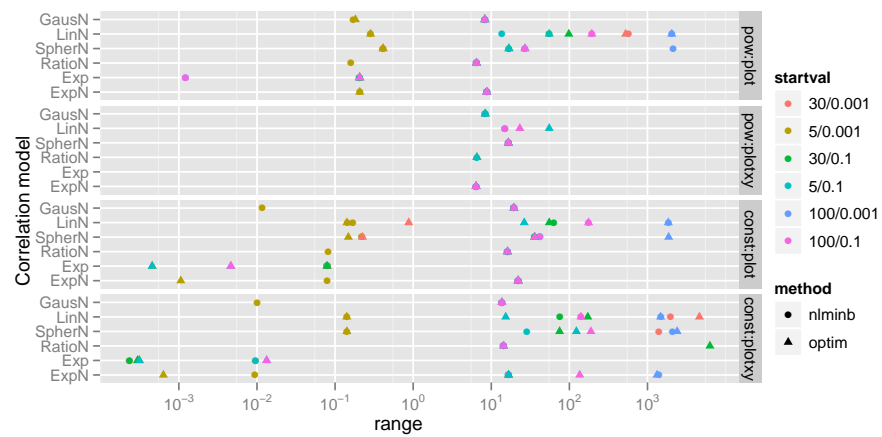
Discarded data from seedling plots with fewer than 10 total seedlings (3/10: B, D, E).

```
> load("pine2.RData")
> totseedlings <- with(seedlings, tapply(count, plot, sum))
> seedlings2 <- droplevels(subset(seedlings, plot %in% levels(plot)[totseedlings>10]))
```

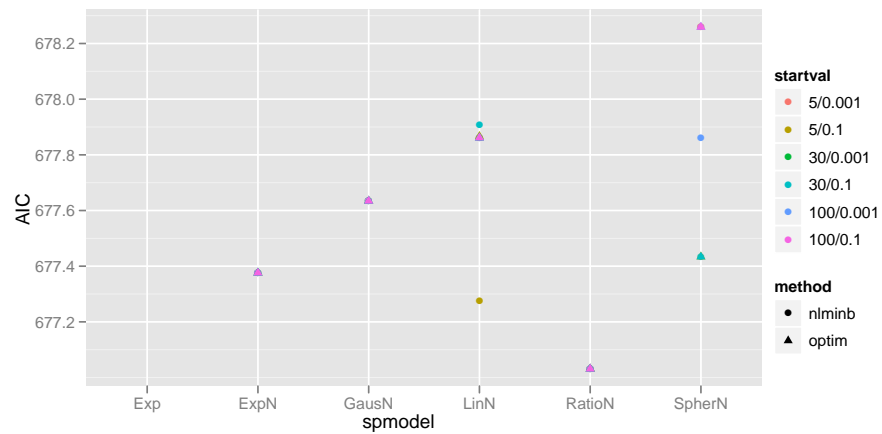
AIC (flipping axes to make model labels more readable):



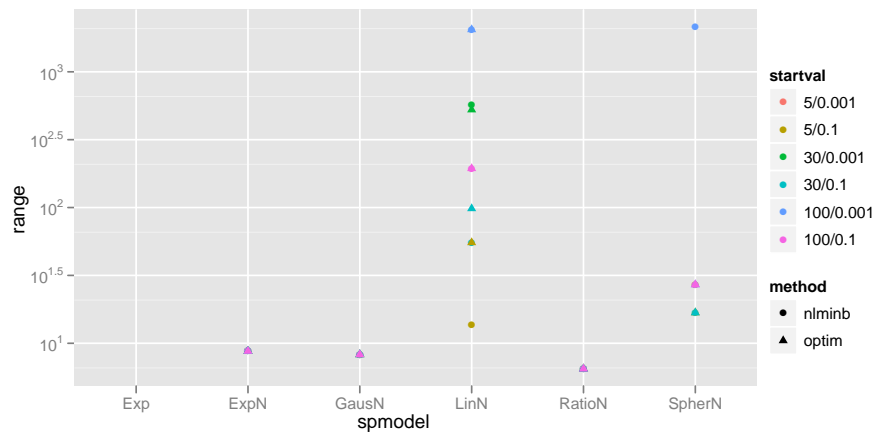
Scales:



Zoom in on pow:plot case (only showing best fits): AIC again ...

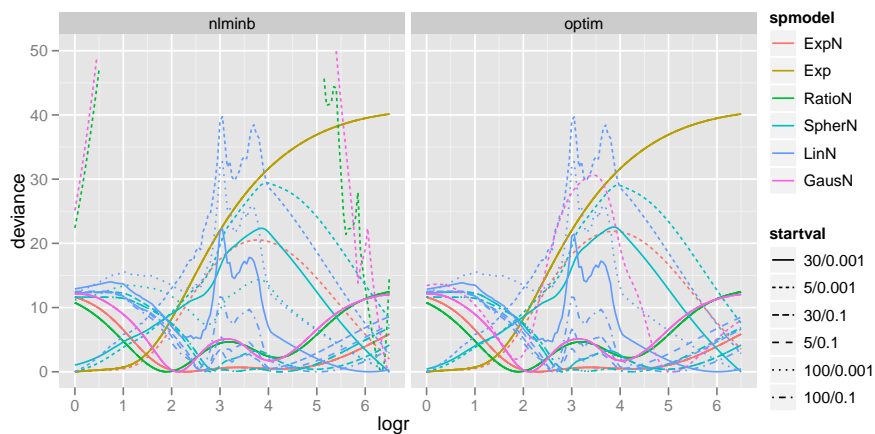


and scale ...



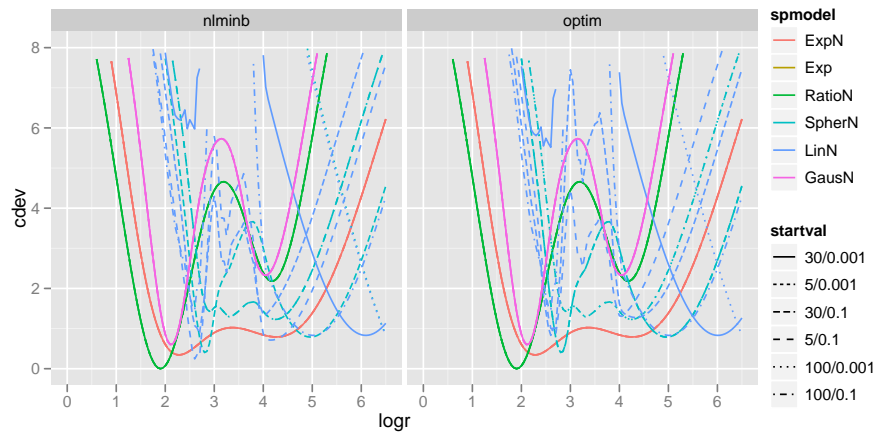
All the better models predict scale  $\approx 5 - 10$  m.

Now look at profiles:



Wacky! (While many of these are just terrible, the deviance minima are near  $\ln(2)$  and  $\ln(4)$ , or approx. 7.5 and 55 m. A lot of the variation has to do with starting values (both range and nugget?) — and the exponential model without nugget is very different (wants to make the range essentially zero), but its AIC is quite bad.

Look at absolute curves instead:



Do I really believe this? Yes, I think so. Exponential plus nugget is about 0.4 AIC units worse in the AIC plots — and has the same complexity so it should also be 0.4 deviance units worse (all the basic spatial correlation structures have only a scale parameter — no shape parameters).

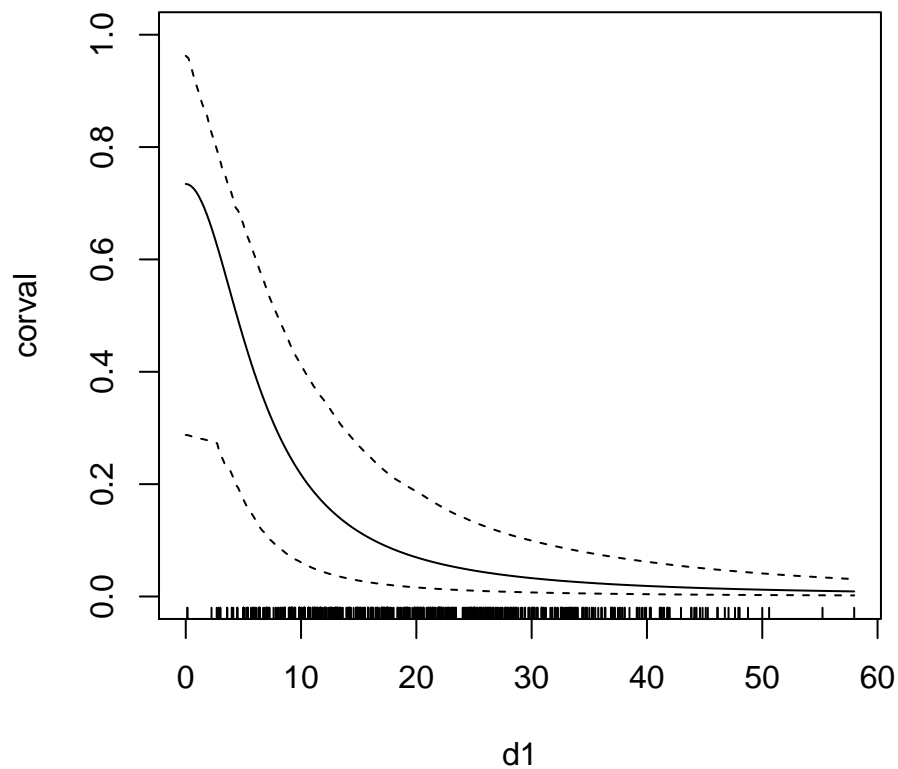
The linear and spherical models are the ones that are very poorly fit/depend on starting conditions and optimizers/get stuck, although the Gaussian, rational, and exponential are bad enough. The exponential actually covers most of the range of the others, although the rational does a little better.

The envelope (dashed lines — 95% curvewise confidence limits of drawing the exponential-nugget model parameters from their sampling distribution) also shows that the spatial correlation is not very well-resolved from these data (sigh). The rational, exponential, and Gaussian models are all well within this envelope. The linear model (which doesn't fit that much worse!) is quite different. Part of the problem is that this envelope is based on the quadratic approximation to the fit, which (as we will see in a minute) is not very good ...

Confidence intervals on the range based on the quadratic approximation (a little scary based on those profiles!) (Best model based on AIC: rational + nugget)

```
> flatbatch <- gunlist(seedstuff$fullbatch)
> best1_seed <- flatbatch[[which.min(seedstuff$rframe$AIC)]]
> library(nlme)
> (cc <- intervals(best1_seed, which="var-cov")$corStruct["range",])

      lower      est.      upper
2.801731  6.477717 14.976748
```



The correlation between the range and the nugget parameter is not too high:

```
> cov2cor(best1_seed$apVar[1:2,1:2])
```

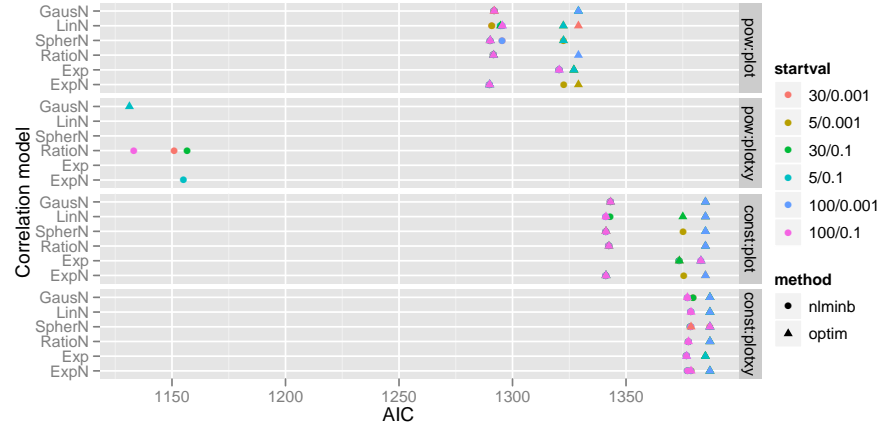
	corStruct.range	corStruct.nugget
corStruct.range	1.0000000	0.5828488
corStruct.nugget	0.5828488	1.0000000

(Need to figure how to get the profile confidence intervals more generally/robustly, even for non-monotonic profiles ...? Brute force finding all minima and then doing backsplines between them ...? ugh.)

### 3 Seedlings

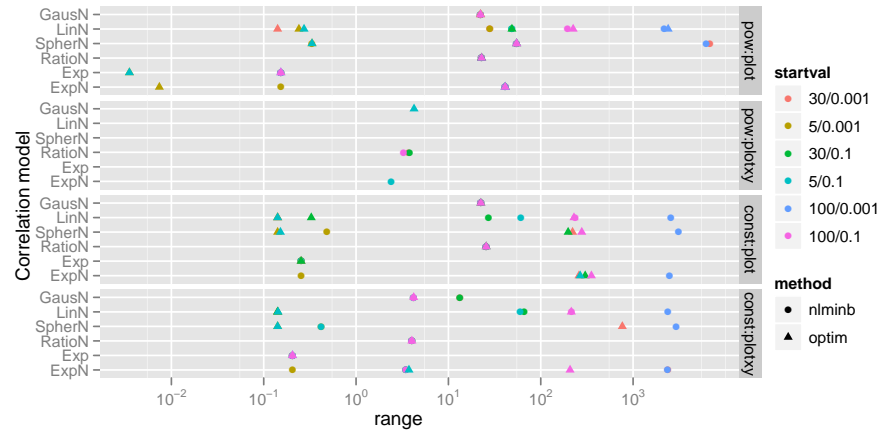
```
> ## same for seedlings ...
> L2 <- load("pine_variog_batch_seedlings.RData")
```

AIC:



(The pow:plotxy case clearly dominates, although we only get successful fits for a couple of models (RatioN and ExpN).)

Scales:



Zoom in on pow:plotxy:

```
> rr <- na.omit(seedlingstuff$rframe)
> head(rr[order(rr$AIC),])
```

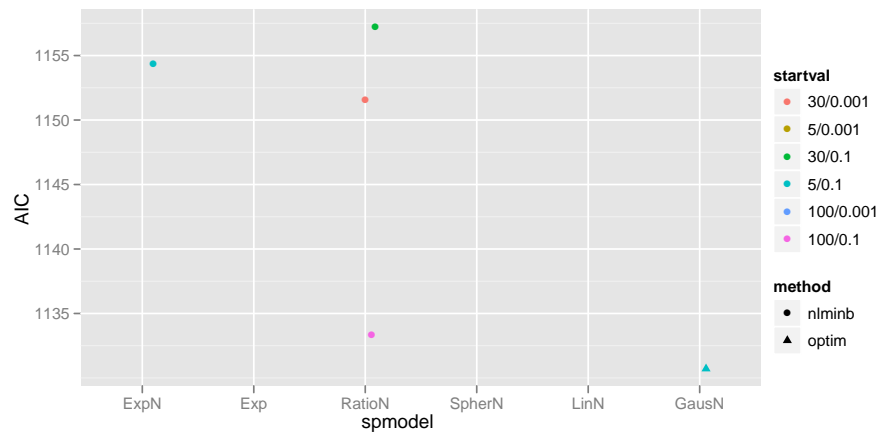
	smodel	startval	method	fixmod	vpmode	LL	AIC	range
132	GausN	5/0.1	optim	plotxy	pow	-549.6254	1131.251	4.245212
105	RatioN	100/0.1	nlminb	plotxy	pow	-550.5816	1133.163	3.277577
75	RatioN	30/0.001	nlminb	plotxy	pow	-559.4696	1150.939	3.696222
91	ExpN	5/0.1	nlminb	plotxy	pow	-561.5079	1155.016	2.411042
87	RatioN	30/0.1	nlminb	plotxy	pow	-562.3732	1156.746	3.797978
25	ExpN	100/0.001	nlminb	plot	pow	-636.9728	1289.946	41.171717
	nugget		bothmod					



```

132 0.6870721 pow:plotxy
105 0.6931288 pow:plotxy
75  0.7122199 pow:plotxy
91  0.5386541 pow:plotxy
87  0.7085841 pow:plotxy
25  0.5118948 pow:plot

```



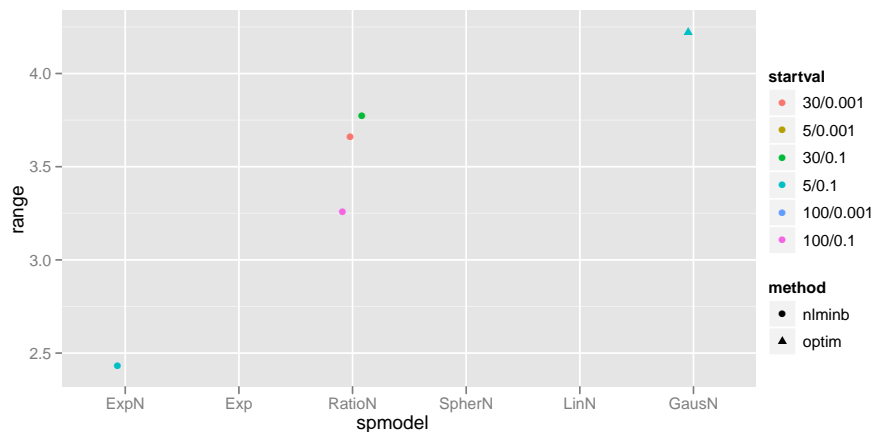
```
> na.omit(rframe2)
```

	spmodel	startval	method	fixmod	vpmod	LL	AIC	range	nugget
75	RatioN	30/0.001	nlminb	plotxy	pow	-559.4696	1150.939	3.696222	0.7122199
87	RatioN	30/0.1	nlminb	plotxy	pow	-562.3732	1156.746	3.797978	0.7085841
91	ExpN	5/0.1	nlminb	plotxy	pow	-561.5079	1155.016	2.411042	0.5386541
105	RatioN	100/0.1	nlminb	plotxy	pow	-550.5816	1133.163	3.277577	0.6931288
132	GausN	5/0.1	optim	plotxy	pow	-549.6254	1131.251	4.245212	0.6870721

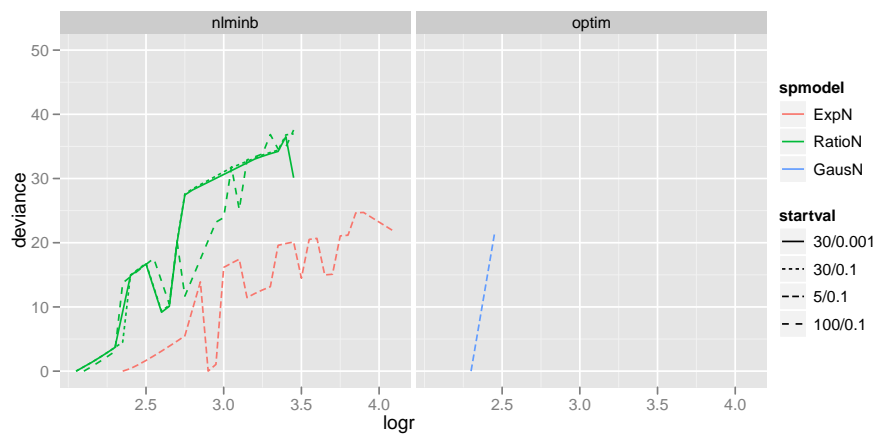
```

bothmod
75 pow:plotxy
87 pow:plotxy
91 pow:plotxy
105 pow:plotxy
132 pow:plotxy

```



Now look at profiles (!!)



These are TERRIBLE (although much better than before).  
Working by hand, using rangeprof:

```
> ## 2 (varpower) x 2 (models?) x 2 (glsControl) x 6 (varstart)
> ##
> na.omit(rframe2)
> flatlist <- gunlist(seedlingstuff$fullbatch) ## collapse list to flat list
> w <- which.min(seedlingstuff$rframe[["AIC"]])
> f0 <- flatlist[[w]]
> save("f0",file="f0.RData")
> load("f0.RData")
> ## re-run profile:
> library(nlme)
> model <- f0
> cc2 <- model$modelStruct$corStruct
> attr(cc2,"fixed")
```

```

> ## attr(cc2,"fixed") <- c(TRUE,FALSE) ## allow nugget to adjust
> ## careful of distinction between the way that cc2 prints vs internal rep?
> ## cc2[1] <- cc2[1]
> attr(cc2,"fixed") <- TRUE
> update(model)
> options(error=recover)
> L <- update(model,correlation=cc2)
> if (inherits(L,"try-error")) NA else logLik(L)
> Lvals <- sapply(logr,Lfun)
> dvec <- -2*(Lvals-max(Lvals,na.rm=TRUE))
> rangeprof(f0,logr=seq(0,6.5,length.out=3))
> rangeprof(f0,logr=seq(4.245,6.5,length.out=2))
> totseedlings <- with(seedlings,tapply(count,plot,sum))
> seedlingplotmin <- 50
> seedlingplotmin <- 100
> seedlings2 <- gdata::drop.levels(subset(seedlings,plot %in% levels(plot)[totseedlings>seedlingplotmin]))
> seedlings2 <- transform(seedlings2,sx0=scale(x0,center=TRUE,scale=40),sy0=scale(y0,center=TRUE,scale=40))
> ## scaled x-y fits actually seem to be WORSE?
> ## dropping more plots leads to singular fits
> g0 <- gls(correlation=corGaus(value=c(5/40,0.1),nugget=TRUE,form=~sx0+sy0|plot),
            model=count~plot*(sx0+sy0),
            dat=seedlings2,weights=varPower(),control=glsControl(opt="optim"))
> ## scaled (plotmin=50): max number of iterations reached without convergence
> ## scaled (plotmin=100): singular gls fit
> cc <- capture.output(g1 <- gls(correlation=corGaus(value=c(5,0.1),nugget=TRUE,form=~x0+y0|plot),
                                model=count~plot*(x0+y0),
                                dat=seedlings2,weights=varPower(),control=glsControl(opt="optim",maxIter=1000,msVerbose=1)))
> ## unscaled (plotmin=50): OK (sort of)
> ## unscaled (plotmin=100): singular gls fit
> ## with Nelder-Mead:

```

- to simulate these kind of data, we probably want to use NB/Poisson statistics to make the look right. if we generate a Gaussian random field with correlation with the appropriate scale, then transform this to log-Normal (exponentiate the values) with appropriate marginal variance, then take Poisson deviates based on this (i.e. a logNormal-Poisson model, which looks a lot like NB)
- then generate a logit-normal distribution (Gaussian field+logistic transform) for establishment probability)
- filter it accordingly: pick binomial samples from each seed sample
- suppose that we've done this on a fairly fine grid: now sample that grid by points, transects, etc... regular samples? point samples?

dd	mean	var	range
seeds	12.77	95.39	6.48
seedlings	1.14	4.37	4.48

- for similitude, I'm constructing samples from several different plots, but I'm not bothering to introduce variation in plots in mean density, nor linear trends across the plots
- try several different scales, including extreme (white noise/global/long range)

Need package `RandomFields` (for generating Gaussian random fields with specified variogram/autocorrelation models).

Ideally, we want to generate simulated 'data' whose moments (spatial and non-spatial) *approximately* match the data, so we can sanity-check, test power, etc. etc.. It will be hard to match exactly, because the various non-linear/stochastic filters that we will use don't change the moments in easily computable/invertible ways.

We will use the following model:

$$\begin{aligned}
Z_0 &\sim \text{MVN}(m, \Sigma_0) \\
\text{Seeds} &\sim \text{Poisson}(e_0^Z) \\
Z_E &\sim \text{MVN}(p, \Sigma_E) \\
\text{Seedlings} &\sim \text{Binom}(\text{prob} = \text{logistic}(Z_E), N = \text{Seeds})
\end{aligned} \tag{1}$$

The notation is ugly, but the basic idea is that the spatial bits of the model will be generated as Gaussian random fields (because that's what we know how to do reasonably easily!) with (say) exponential+nugget spatial correlations and specified ranges (e.g. parameters  $n_0, r_0, n_E, r_E$ ), and specified means ( $\mu_0, \mu_E$ ) and variances ( $\sigma_0^2, \sigma_E^2$ ). What we want out at the end are random fields for seeds and seedlings with

Mean: seeds 13, seedlings .08 [BMB: typo??]  
Variance: seeds 95, seedlings 3  
Scale: seeds 40, seedlings 54

I would say instead we need:

(the seedlings range is hacked, until I get farther with estimation).

Let's first take the seed distribution. Let's suppose that this is a lognormal-Poisson distribution with mean  $\mu$ , variance  $\sigma^2$ ; how do we relate these values to the underlying  $m_0, \sigma_0^2$  of the Gaussian field?

Two web sources quote the moments of the lognormal-Poisson distribution:  
<http://sci.tech-archive.net/Archive/sci.stat.math/2007-11/msg00004.html>  
(referring to the *Encyclopedia of Statistical Sciences* (2004, vol 9, p. 6198)

and the *Dictionary and classified bibliography of statistical distributions in scientific work* (Patil et al 1984)), and Cameron and Trivedi (1998) [exercise 4.3: referenced on Google Books] quote the mean as being the same as the lognormal mean ( $\exp(\mu + \sigma^2/2)$ ) and the variance as

$$\exp(2\mu + 2\sigma^2) - \exp(2\mu + \sigma^2) + \exp(\mu + \sigma^2/2) \quad (2)$$

(for comparison, the LN variance is  $\exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1) = \exp(2\mu + 2\sigma^2) - \exp(2\mu + \sigma^2)$ ; so in fact the LN-P variance really is just (LN variance+Poisson variance=LN variance + LN mean), as I would have expected ... still may be worth going through the calculation at some point, especially as it would help with the next case.)

Check this numerically:

```
> mvfun <- function(m,s) {
  c(exp(m+s^2/2), exp(2*m+2*s^2)+exp(m+s^2/2)-exp(2*m+s^2))
}
> mvfun(1.5,1)

[1] 7.389056 101.204065

> rr <- t(replicate(100,
  {x <- rpois(1e5, rlnorm(1e5, meanlog=1.5, sdlog=1));
  c(mean(x), var(x))}))
> summary(rr)

      V1      V2
Min.   :7.313  Min.   : 94.64
1st Qu.:7.364  1st Qu.: 98.45
Median :7.388  Median :100.62
Mean    :7.384  Mean    :101.10
3rd Qu.:7.405  3rd Qu.:103.63
Max.    :7.448  Max.    :112.82
```

Call in a nonlinear equation-solver to invert the function, looking for  $\mu$ ,  $\sigma$  that will produce a Poisson-lognormal mean and variance of 13 and 95:

```
> library(BB)
> objf <- function(p,target) {
  mvfun(p[1],p[2])-target
}
> b1 <- BBSolve(c(1,1),objf,target=c(13,95))

Successful convergence.

> b1$par

[1] 2.367172 0.628931
```

```
> mvfun(b1$par[1],b1$par[2])
```

```
[1] 13 95
```

OK, that was easy! Of course, don't know exactly what kind of nugget/range this will give rise to, but for now let's generate a random field with underlying  $\mu_0$  and  $\sigma_0^2$  based on these numbers — the Poisson randomness will add at least some nugget ...

```
> seeds_sim0 <- do_sim(m=b1$par[1],v=b1$par[2]^2,nugget=0,
  range=coef(best1_seed$modelStruct$corStruct,unconstrained=FALSE)["range"])
```

```
## Note that several updates of RandomFields are expected during 2011. ##
## Please see help("changings") for important changes. ##
```

```
> ssvec <- c(seeds_sim0)
> print(c(summary(ssvec),sd=sd(ssvec),var=var(ssvec)),digits=3)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	sd	var
0.00	0.00	1.00	1.43	2.00	11.00	1.45	2.10

(Categories are for plotting ...)

```
> seedcat <- cut(seeds_sim0,breaks=c(0,5,10,15,20,100,500),right=FALSE)
> scatmat <- matrix(as.numeric(seedcat),nrow=nrow(seeds_sim0))
```

For “environment” (i.e. the spatial distribution of seedling establishment/survival), it's going to be a little bit trickier. Starting from our existing seed distribution, we need to end up with a mean density of seedlings as specified above. Therefore, we want to do the same kind of calculation, but for a logit-normal-binomial (sometimes called a logistic-normal-binomial) rather than a lognormal-Poisson — and  $N$  is already a random variable! (V/M ratio increases from  $\approx 7$  to  $\approx 30$  between the seedling and seed distribution: CV from  $\approx 1$  to  $\approx 17$ .) It would be a nice exercise in math. stats. to compute the moments of a logit-normal-binomial with known  $N$ ,  $\mu$ ,  $\sigma$ , and then with  $N$  as a random variable, but I think I'm just going to guess instead ... (Some haphazard web searching suggests that this may be hard. Aitchison and Shen (1980) suggest that the moments of the logistic-normal itself are inconvenient (although they're mostly referring to the multivariate case, I don't know how hard the univariate case is ...)) — I could probably work out the moments of the logistic-normal-binomial if I had those of the logistic-normal, but without them I'm probably screwed (or saved, depending on how you look at it). Trial-and-error/hacking time ...

I want the logistic-normal transformation to translate from a (lognormal-Poisson) with the above moments (mean  $\approx 13$ , var  $\approx 95$ ) to a distribution with mean  $\approx 1$ , var  $\approx 4$ .

```
> mvfun <- function(s2m,s2sd,n=1e6,s1m=b1$par[1],s1sd=b1$par[2]) {
  x <- rbinom(n,prob=plogis(rnorm(n,mean=s2m,sd=s2sd)),
```

```

        size=rpois(n,rlnorm(n,meanlog=s1m,sdlog=s1sd)))
    }
> x <- mvfun(s2m=-2,s2sd=1)
> c(mean(x),sd(x))

[1] 2.020107 2.815934

> tmpfun <- function(p,target=c(1,2)) {
  x <- mvfun(s2m=p[1],s2sd=p[2])
  mx <- mean(x); sdx <- sd(x)
  r <- (mx-target[1])^2+(sdx-target[2])^2
  cat(p,mx,sdx,r,"\n")
  r
}
> tmpfun(c(-2,1))

-2 1 2.023803 2.817624 1.716681
[1] 1.716681

> optim(tmpfun,par=c(-2,1),control=list(maxit=20))

-2 1 2.020983 2.808017 1.695298
-1.8 1 2.338174 3.117819 3.040229
-2 1.2 2.198006 3.22923 2.946225
-2.2 1.2 1.900752 2.927364 1.671359
-2.4 1.3 1.729278 2.846986 1.249232
-2.4 1.1 1.565883 2.451383 0.52397
-2.6 1.05 1.302281 2.104147 0.1022204
-3 1.35 1.135963 2.177182 0.04987926
-3.5 1.525 0.874629 1.944771 0.01876817
-3.7 1.275 0.60337 1.347197 0.5834677
-3.375 1.28125 0.804219 1.655343 0.1571188
-2.725 1.29375 1.355206 2.402985 0.2885681
-3.2125 1.284375 0.917634 1.818606 0.0396878
-4.1125 1.759375 0.661663 1.768333 0.1681414
-2.978125 1.227344 1.066567 1.981047 0.004790396
-3.265625 1.467969 1.008069 2.105689 0.01123518
-3.252344 1.42207 0.981875 2.016839 0.0006120759
-2.730469 1.124414 1.222243 2.084584 0.05654636
-3.307617 1.424854 0.942646 1.964401 0.004556787
-3.581836 1.61958 0.881825 2.028613 0.01478405
-3.129053 1.325403 1.00965 1.97578 0.0006797288
$par
[1] -3.252344 1.422070

$value
[1] 0.0006120759

```

```
$counts
function gradient
      21      NA
```

```
$convergence
[1] 1
```

```
$message
NULL
```

Things obviously bounce around quite a bit, but we should be reasonably close.

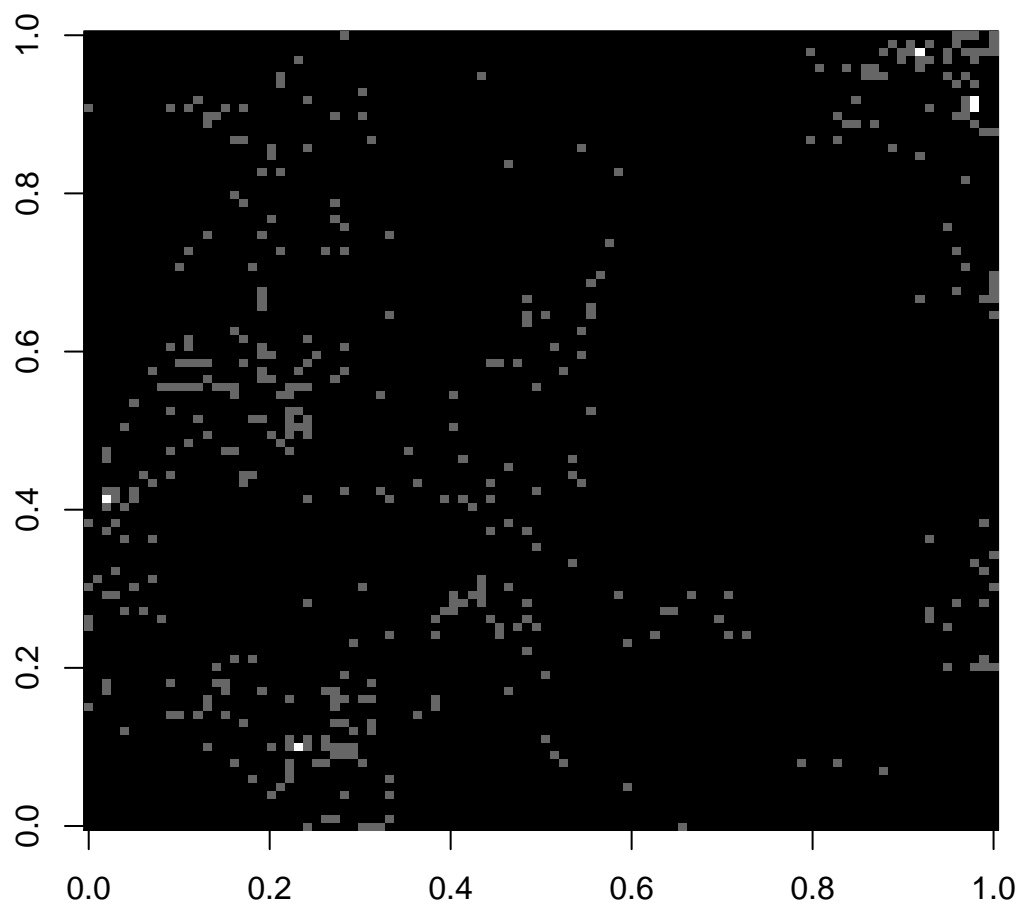
```
> ## environment
> seedlings_sim <- do_sim(m=-3.5,v=1.5^2,nugget=0,
                        range=2,rfun=rbinom,invlink=plogis,
                        input=seeds_sim0,
                        retval="list")
> ss_env <- seedlings_sim$env
> seedlings_sim <- seedlings_sim$output
> ss2vec <- c(seedlings_sim)
> print(c(summary(ss2vec),sd=sd(ss2vec),var=var(ss2vec)),digits=3)

      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.      sd      var
0.000   0.000   0.000   0.671   1.000   11.000   1.014   1.027

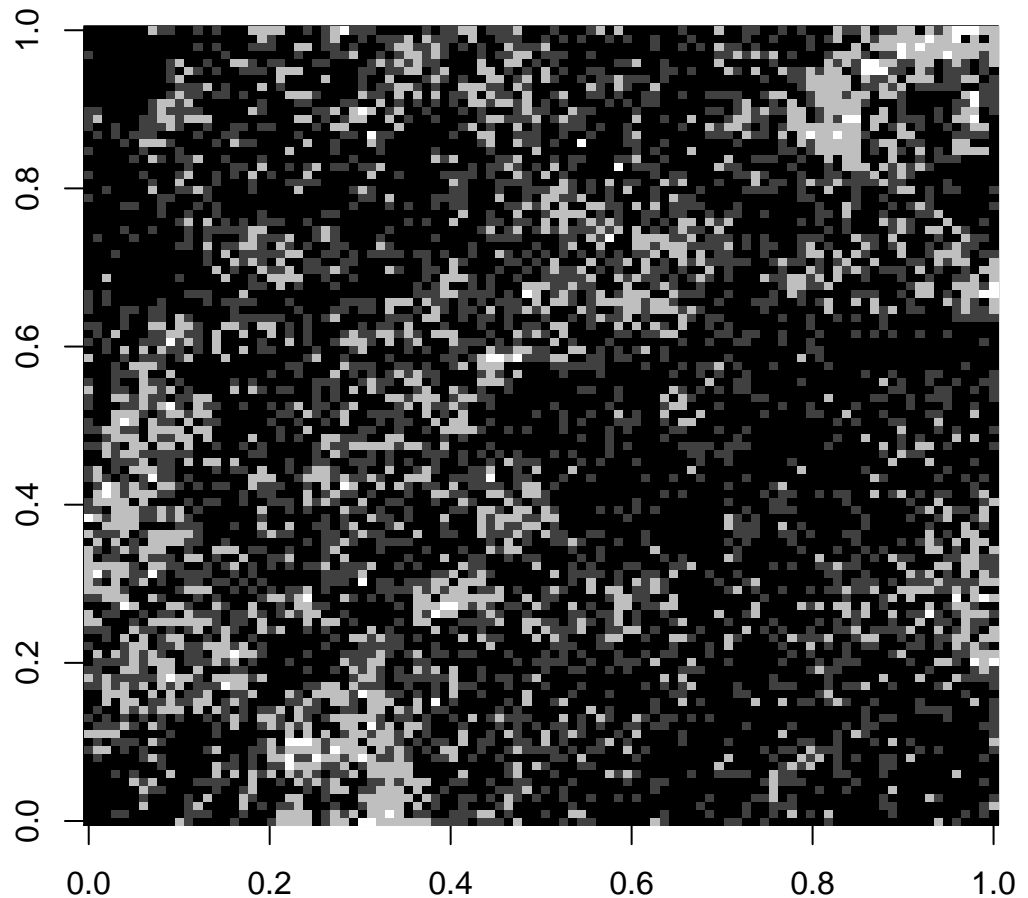
> seedlingcat <- cut(seedlings_sim,breaks=c(0,1,2,5,100),right=FALSE)
> seedlingcatmat <- matrix(as.numeric(seedlingcat),nrow=nrow(seedlings_sim))

> image(scattermat, col=gray((0:5)/5))
> contour(plogis(ss_env),add=TRUE,col=2,levels=c(0,0.05,0.1))
```





```
> image(seedlingcatmat, col=gray((0:4)/4))
```



Now we can “sample” an appropriate number of points from these data sets, and analyze them as we analyzed our real data (... I wonder how good/bad they will look relative to the real data?) — we should, ideally, find that exponential is best and that no linear trend is desired in either case.

```
> with(seeds, table(plot)) ## observations per plot, seeds
```

```
plot
```

A	B	C	D	E	F	G	H	J
7	15	9	4	17	12	7	17	19

```

> with(seeds,tapply(count,list(plot),sum)) ## total counts per plot, seeds

  A   B   C   D   E   F   G   H   J
7 148 194  20 169  54  79 291 404

> with(seedlings2,table(plot)) ## observations per plot, seedlings

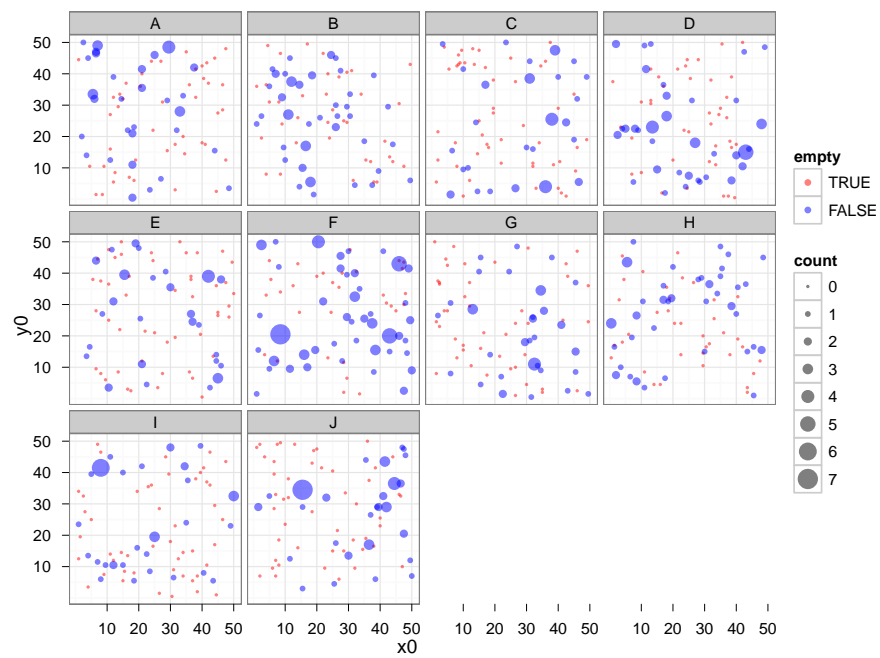
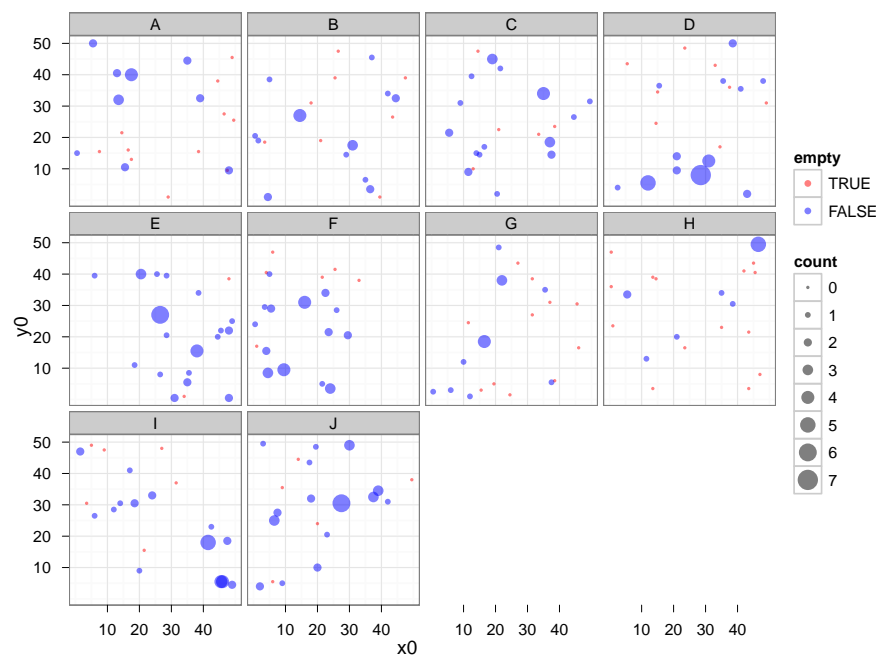
plot
  A   C   F   G   H   J   K
75 75 74 75 75 75 75

> with(seedlings2,tapply(count,list(plot),sum)) ## total counts per plot, seedlings

  A   C   F   G   H   J   K
32 181  38  80  57  27 182

> seedparms <- list(m=2.36,v=0.4,n=0,r=6.5,s=20,p=10)
> ## mean, var, nugget, ranges, samples/plot, plots
> sfun <- function(retval="sample",parms=seedparms) {
  with(parms,
    do_sim(m=m,v=v,nugget=n,range=r,
           retval=retval,sample=s))
}
> seedsims_list <- replicate(seedparms$p,sfun(),simplify=FALSE)
> seedsims <- do.call(rbind,mapply(data.frame,
  seedsims_list,
  plot=as.list(LETTERS[seq_along(seedsims_list)]),
  SIMPLIFY=FALSE))
> seedlingparms <- list(m=-3.5,v=1.5^2,n=0,r=2,s=75,p=10)
> seedlingsims_list <- replicate(seedlingparms$p,
  with(seedlingparms,
    do_sim(m=m,v=v,nugget=n,
           range=r,rfun=rbinom,invlink=plogis,
           retval="sample",sample=s,
           input=sfun("grid"))),
  simplify=FALSE)
> seedlingsims <- do.call(rbind,mapply(data.frame,
  seedlingsims_list,
  plot=as.list(LETTERS[seq_along(seedlingsims_list)]),
  SIMPLIFY=FALSE))

```



## References

- Aitchison, J. and S. M. Shen (1980, August). Logistic-Normal distributions: Some properties and uses. *Biometrika* 67(2), 261–272.
- Cameron, A. C. and P. K. Trivedi (1998). *Regression analysis of count data*. Cambridge University Press.