RSPPlme4 lansing example

Robert Bagchi, Michael C. LaScaleia, Valerie R. Milici, Dipanjana Dalui

March 07, 2022

#Preliminaries

```
## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## spatstat.geom 2.3-1

## Loading required package: spatstat.core

## Loading required package: nlme

## Loading required package: rpart

## spatstat.core 2.3-2

## Loading required package: spatstat.linnet

## spatstat.linnet 2.3-1

##
## spatstat 2.3-0       (nickname: 'That's not important right now')
## For an introduction to spatstat, type 'beginner'

## Registered S3 method overwritten by 'cli':
##   method     from
##   print.boxx spatstat.geom

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
```

Organize data

This first chunk of code structures the data into format that `klmer` will accept as an argument. Creating a `hyperframe` is necessary only because a standard `data.frame` will not support a column of `ppp` objects.

The original plot is split into 5 separate plots based on tree species, then each plot is split into 16 evenly sized suplots.

```
# load dataset
lansing <- spatstat.data::lansing

# because "misc" is not a single tree species, we will not use it in this example
lansing <- subset(lansing, marks != "misc", drop = T)

# get rid of duplicated points
lansing <- lansing[!duplicated(lansing),]

# establish number of subplots (necessary only when breaking up a single large site)
subplots <- 16

# rescale data to be in meters (3.3 feet)
lansing <- spatstat.geom::rescale(lansing, 1/280)

# divide the lansing ppp into 5 ppps based on species (marks)
mm <- split(lansing, f = lansing$marks)

# divide each of the 5 ppps into 16 ppps based on subplots for 80 ppps total
tess <- quadrats(lansing, nx = sqrt(subplots))
pppList <- lapply(mm, split, f = tess)

# use setNames to avoid spatstat's default tile names
pppList <- lapply(pppList, setNames, nm = 1:subplots)

# put all 80 ppps into a single level of organization
pppList <- unlist(pppList, recursive = F)

# extract the names of the categorical variables
catVariables <- str_split(names(pppList), pattern = "\\.")
catVariables <- as.data.frame(do.call("rbind", catVariables))
colnames(catVariables) <- c("marks", "subPlotID")

# Because it is first alphabetically, black oak is used as the intercept group
# Black oak has a small sample size, so hickory makes for a better intercept
catVariables$marks <- relevel(as.factor(catVariables$marks), "hickory")

# create hyperframe of the sub-plot IDs and any fixed or random effect
hf <- as.hyperframe(catVariables, pppx = pppList)
```

Models

Model comparing species clustering

This model looks at the differences in clustering between the 5 species of trees, assuming a varying intercept of clustering between subplots (subplots are a random effect).

The warnings in this model suggest that the variability of clustering between species within the same subplot is negligible.

```
# run model
mod_sm <- klmer(formula = pppx ~ 1 + marks + (1|subPlotID),
                      r= 0:15 , hyper=hf,  correction= 'border',
                      na.action="na.omit", weights_type = "nx_A")
```

```
## Warning in klmer(formula = pppx ~ 1 + marks + (1 | subPlotID), r = 0:15, :
## Removed rows 7, 9, 14, 33, 34, 36, 42 which have zero weights
```

```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
```

```
# create confidence intervals
preddat <- expand.grid(marks = levels(hf$marks))
mod_sm_cis <- confint(mod_sm, level=0.95,  newdata = preddat,  nboot=99, ncore=4, iseed=1234)
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## clusters closed on exit
```

```r
anova(mod_sm, term = "marks", dists = 1:15, nboot = 499, ncore = 4,
      iseed = 1234)
```

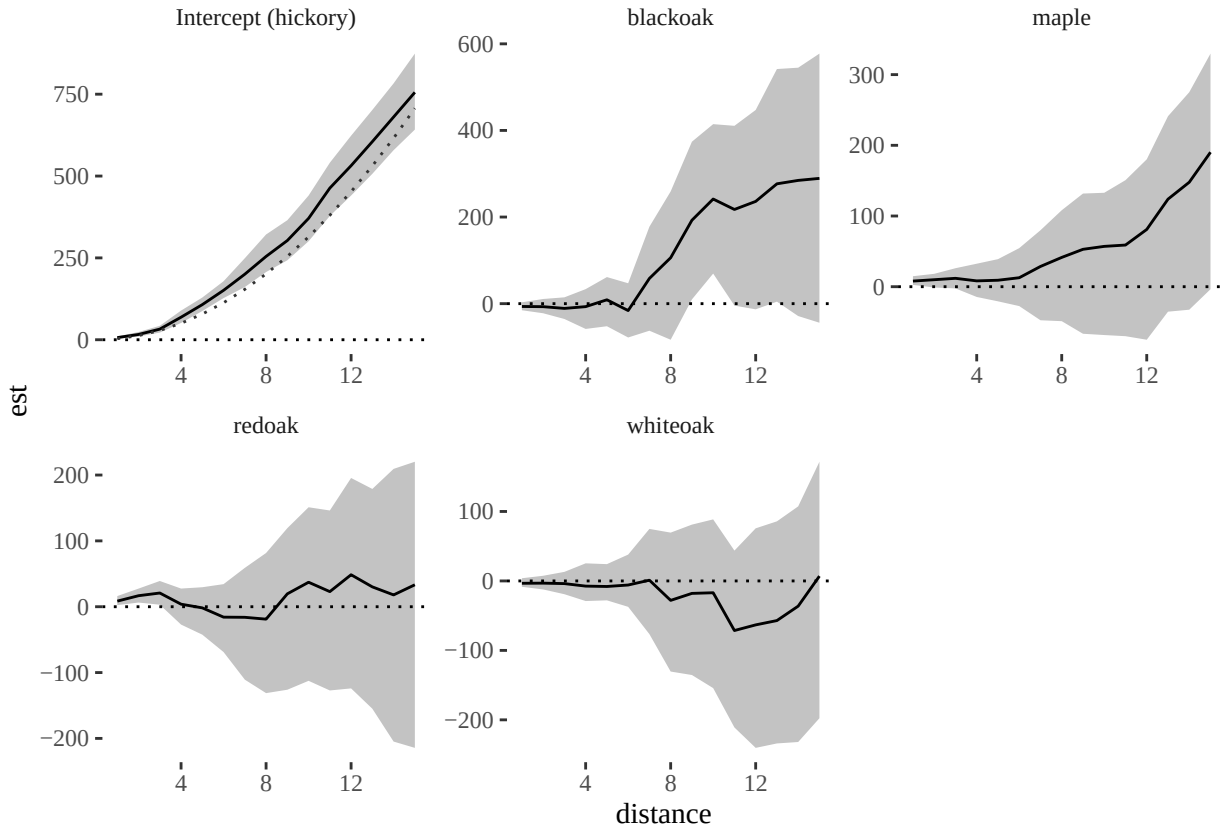
```
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular
## boundary (singular) fit: see ?isSingular

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID
```

```
## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## Warning in FUN(X[[i]], ...):
## Variance components are 0 for subPlotID

## [1] "clusters closed on exit"

## [1] "marks ( nsim =  499 )"
##
## Distances = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
##        T        p
## 1.781268 0.072000
```

With the model created and the confidence intervals generated, we can plot the fixed effects, which are the difference between the intercept (clustering of hickory) vs. the clustering of the other species.

```
# plot fixed effects

# make plotable object
ci_plot_data <- makePlotData_klmerci(mod_sm_cis)
```

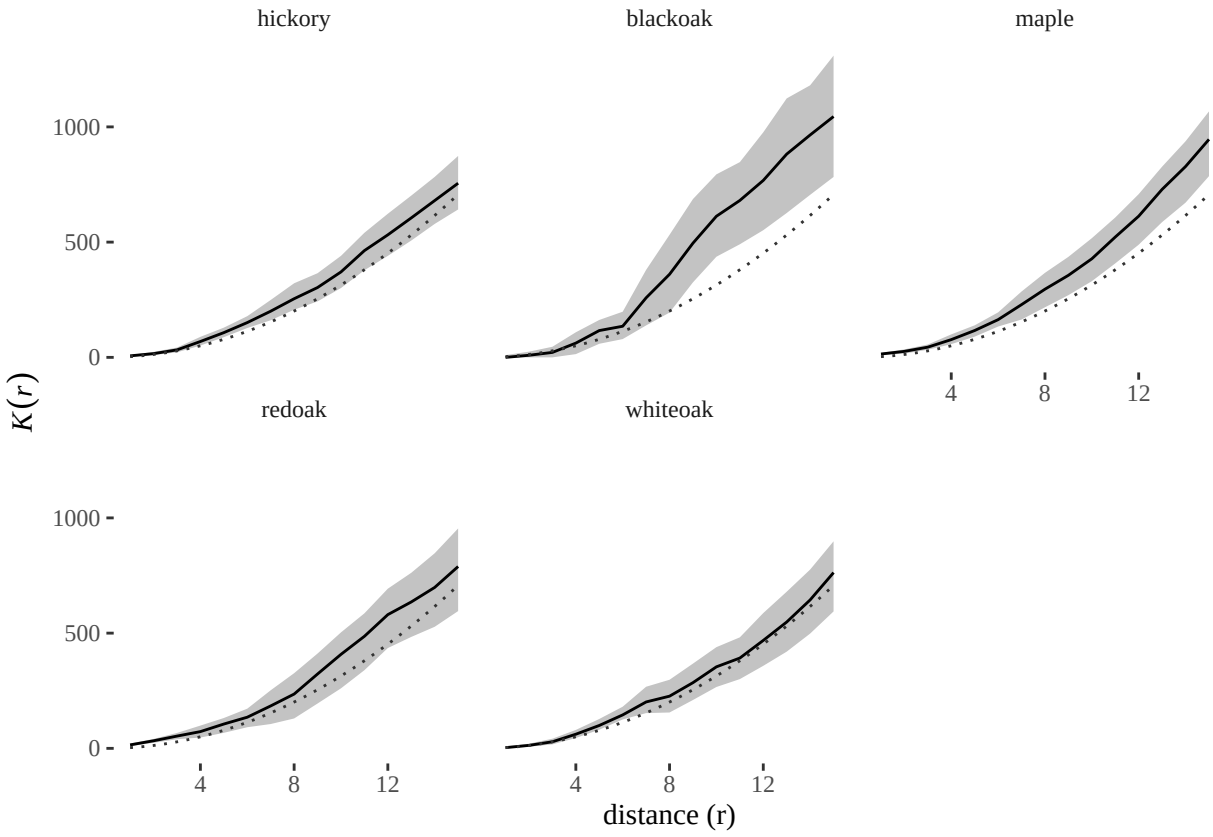
```
# save measured distances
r <- min(ci_plot_data$distance):max(ci_plot_data$distance)

# create graph labels
marks.labs <- c("Intercept (hickory)", "blackoak", "maple", "redoak", "whiteoak")
names(marks.labs) <- c(levels(ci_plot_data$term))

# make plot
ggplot(ci_plot_data, aes(x = distance, y = est)) +
  geom_ribbon(mapping = aes(ymin = lwr, ymax = upr),colour = NA, alpha = 0.3) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = 'dotted') +
  facet_wrap(~term, scale = 'free', labeller = labeller(term = marks.labs)) +
  theme_tufte() +
  geom_line(data = data.frame(nullest = pi*r^2, r = r, term = "Intercept"),
            aes(x = r, y = nullest), linetype = "dotted", color = "grey20")
```



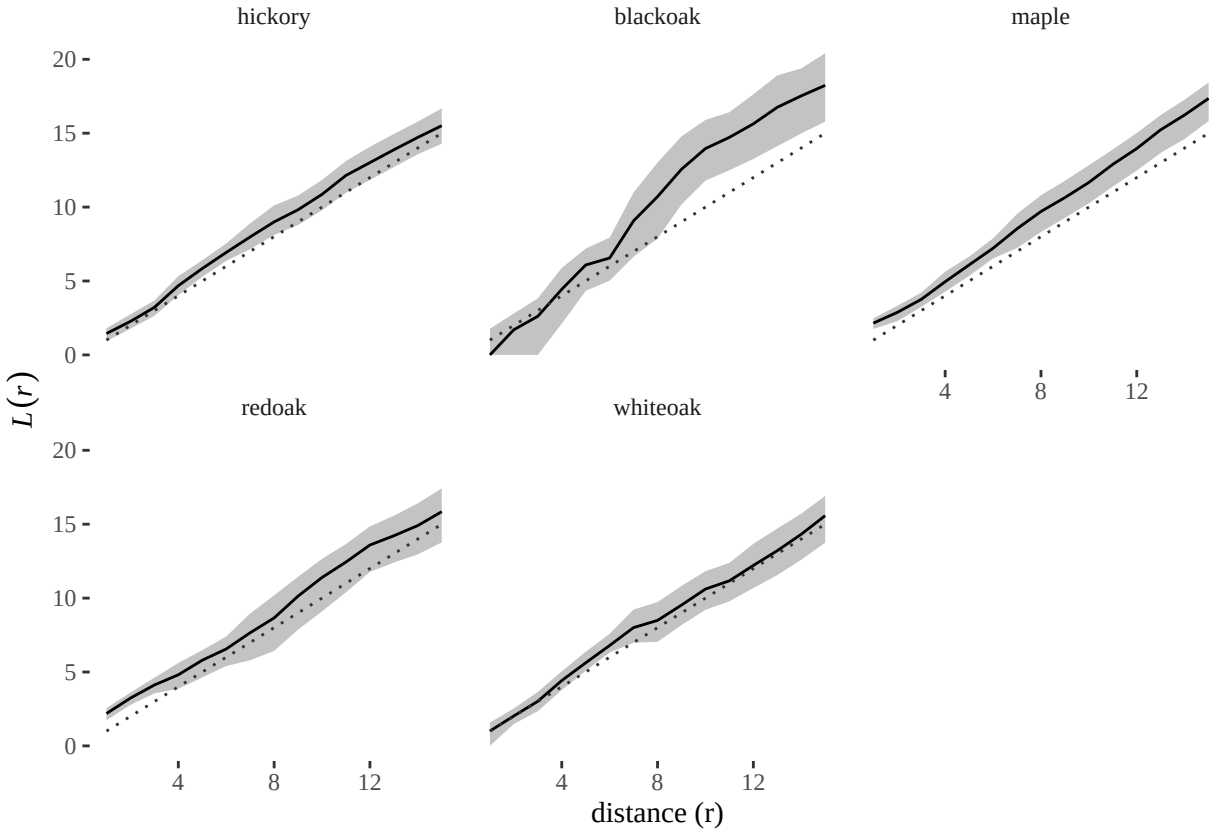
Further, we can plot the predicted clustering of the 5 tree species on two scales.

```
# plot predictions

# extract predictions
predictions <- as.data.frame.table(mod_sm_cis$predictions)

# make plotable data
predictions$Var2 <- preddat$marks[as.numeric(predictions$Var2)]
```

```
predictions <- pivot_wider(predictions, names_from = Var3, values_from = Freq) %>%
               rename(dist = Var1, marks = Var2) %>%
               mutate(dist = as.numeric(dist))

# plot data on K scale
ggplot(predictions, aes(x = dist, y = est)) +
 geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.3) +
  geom_line() +
  facet_wrap(~ marks) +
  theme_tufte() +
  geom_function(fun = function(r) pi*r^2, linetype = "dotted", color = "grey20") +
    labs(x = "distance (r)", y = expression(italic(K(r))))
```




```
# plot data on L scale
K2L <-  function(k) sqrt(k/pi)
ggplot(predictions, aes(x = dist, y = K2L(est))) +
 geom_ribbon(aes(ymin = K2L(lwr), ymax = K2L(upr)), alpha = 0.3) +
  geom_line() +
  facet_wrap(~ marks) +
  theme_tufte() +
  geom_function(fun = function(r) r, linetype = "dotted", color = "grey20") +
  labs(x = "distance (r)", y = expression(italic(L(r))))
```



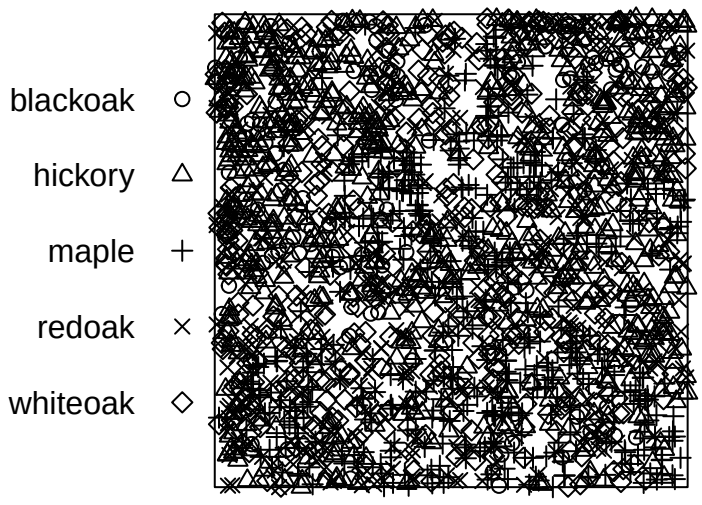
Artificially altered spatial patterns

Data generation

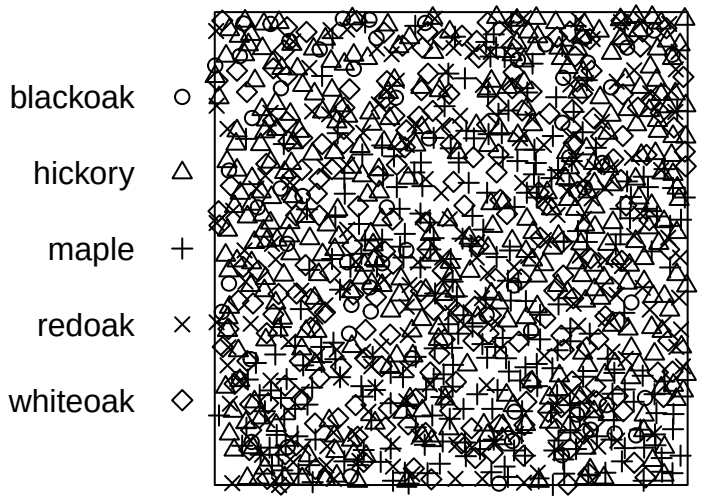
Here, we artificially create both positive and negative density dependence to the dataset. The output is three `ppp` objects of the full `lansing` plot: one that is the same as the original, one that is "thinned", and one that is "thickened"

```
# make a ppp where each species is considerably more evenly distributed
thinned_list <- lapply(1:5, function(i){
  nn <- as.matrix(dist(coords(mm[[i]])))
  nn[lower.tri(nn)] <- 0
  suppressWarnings(q <- apply(nn, 2, function(x){xx <- x[x!=0]; min(xx)}))
  q[is.infinite(q)] <- 0
  qq <- quantile(q, probs = seq(0,1,.25))
  thold <- qq[length(qq) - 2]
  keepers <- apply(nn, 1, function(x, told = thold){
    y <- x[x != 0]
    if(length(y) != 0){
      if(min(y) < told) keep <- 0
      else keep <- 1
    }
    else keep <- 1
    return(keep)
```

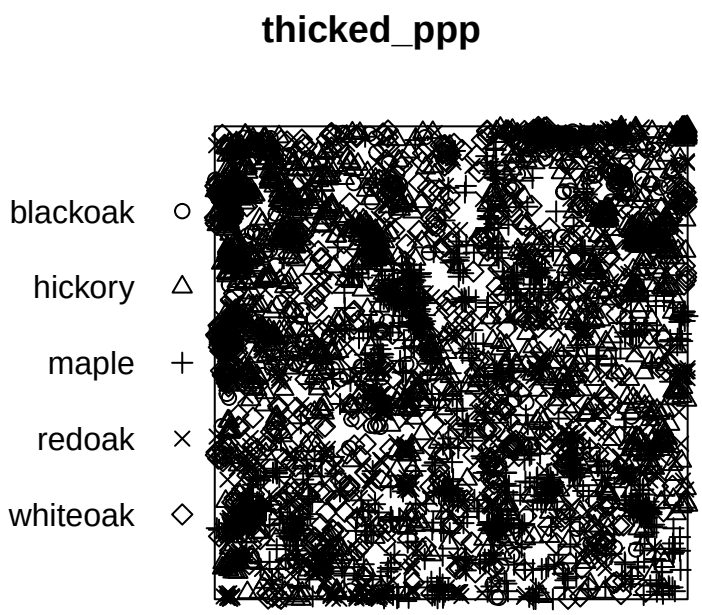
```
  })
  tmap <- rthin(mm[[i]], P = keepers)
  return(tmap)
})

thinned_ppp <- do.call("superimpose", thinned_list)
# npoints(thinned_ppp)

# make a ppp where each species is considerably more clustered
range01 <- function(x){(x-min(x))/(max(x)-min(x))}
thicked_list <- lapply(1:5, function(i){
  pla <- mm[[i]]
  newh <- rpoint(pla$n, f = range01(density(pla, adjust = .1)^4))
  newh$marks <- as.factor(rep(unique(pla$marks), newh$n))
  newh$markformat <- "vector"
  hs <- superimpose(pla, newh)
  return(hs)
  }
)

thicked_ppp <- do.call("superimpose", thicked_list)
# npoints(thicked_ppp)


plot(lansing)
```



```
plot(thinned_ppp)
```




```
plot(thicked_ppp)
```



Data wrangling

These data are then organized in a similar way to the data above. Note that after each `split` there is an `unlist(x, recursive = f)`.

```
# perform similar steps as above to prepare the ppp objects
# uses the following objects from the original "structure data" chunk:
# lansing, tess, subplots

allppp <- list(lansing, thinned_ppp, thicked_ppp)
names(allppp) <- c("norm", "thin", "thic")

# divide the all ppp into ppps based on species (marks)
aa <- lapply(allppp, function(x) split(x,  f = x$marks))
aa <- unlist(aa, recursive = F)

# divide each of the ppps into subplots
aaList <- lapply(aa, split, f = tess)

# use setNames to avoid spatstat's default tile names
aaList <- lapply(aaList, setNames, nm = 1:subplots)

# put all ppps into a single level of organization
aaList <- unlist(aaList, recursive = F)
```

```r
# extract the names of the categorical variables
aaCatVariables <- str_split(names(aaList), pattern = "\\.")
aaCatVariables <- as.data.frame(do.call("rbind", aaCatVariables))
colnames(aaCatVariables) <- c("clustering", "marks", "subPlotID")

# Because it is first alphabetically, black oak is used as the intercept group
# Black oak has a small sample size, so hickory makes for a better intercept
aaCatVariables$marks <- relevel(as.factor(aaCatVariables$marks), "hickory")
aaCatVariables$clustering <- as.factor(aaCatVariables$clustering)

# create hyperframe of the sub-plot IDs and any fixed or random effect
ahf <- as.hyperframe(aaCatVariables, pppx = aaList)
```

Models

We can then run the models on this hyperframe just like the previous. Unlike the previous, species is now a random effect and the original ppp of the subplot (normal, thined, or thickened) is now the fiexed effect.

```r
# run model
a_mod_sm <- klmer(formula = pppx ~ 1 + clustering + (1|subPlotID) + (1|marks),
                    r= 0:15 , hyper=ahf,  correction= 'border',
                    na.action="na.omit", weights_type = "nx_A")
```

```
## Warning in klmer(formula = pppx ~ 1 + clustering + (1 | subPlotID) + (1 | :
## Removed rows 7, 9, 14, 33, 34, 36, 42, 87, 89, 94, 113, 114, 116, 122, 167, 169,
## 174, 193, 194, 196, 202 which have zero weights
```

```r
# create confidence intervals
a_preddat <- expand.grid(clustering = levels(ahf$clustering))
a_mod_sm_cis <- confint(a_mod_sm, level=0.95,  newdata = a_preddat,  nboot=99, ncore=4, iseed=1234)
```

```
## clusters closed on exit
```

And, finally, we can plot the effect of thinning and thickening on the clustering of these five tree species. Overall, the trees are generally clustered at larger scales, but are considerably more clustered in the "thickened" set of subplots, and considerably less so in the "thinned"

```r
# plot fixed effects

# make plotable object
ci_a <- makePlotData_klmerci(a_mod_sm_cis)

# save measured distances
r <- min(ci_a$distance):max(ci_a$distance)

# create graph labels
ci_a$term <- relevel(ci_a$term, "Intercept")
marks.labs <- c("Intercept", "Thinned", "Thickened")
names(marks.labs) <- c("Intercept", "clusteringthin", "clusteringthic")

# correct facet order
```

```
nullline <- data.frame(nullest = pi*r^2, r = r, term = "Intercept")
nullline$term <- factor(nullline$term, levels = levels(ci_a$term))

# make plot
ggplot(ci_a, aes(x = distance, y = est)) +
  geom_ribbon(mapping = aes(ymin = lwr, ymax = upr),colour = NA, alpha = 0.3) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = 'dotted') +
  facet_wrap(.~term, labeller = labeller(term = marks.labs)) +
  theme_tufte() +
  geom_line(data = nullline, aes(x = r, y = nullest), linetype = "dotted", color = "grey20")
```