Introduction to the RSPPlme4 package

Robert Bagchi

August 25, 2020

Contents

If you need to install the package, the easiest way is to do so directly from github using the devtools package. Once you have the dependencies for the package (`spatstat`, `lme4`, `abind`, `Matrix` and `stats`) and `devtools` installed, the `RSPPlme4` package can be installed with:

```
devtools::install_github("BagchiLab-Uconn/RSPPlme4")
```

```
library(tidyverse)
library(spatstat)
library(RSPPlme4)
```

Introduction

The purpose of the `RSPPlme4` package (which stands for "Replicated Spatial Point Patterns using lme4") is to provide a platform for consistently expressing the second-order structure of spatial point patterns as functions of covariates. The framework accommodates both "fixed" and "random" effects.

This document will provide some examples of the functionality of the package and fit some simple models to data.

Examples

Example 1: a simulated data set

The first example will use a simulated data set. The simulated data consist of 50 point patterns, randomly allocated to 10 groups (`gr`). Each point pattern is associated with a randomly sampled value of a covariate, `x`.

Simulate data

The `spatstat` package has a structure, called a `hyperframe` that allows combination of data frames (which combines equal-lengthed columns of variables) with more complex data structures and objects - for example, when each row in the column is a vector or a function. In our case, we want columns where each element is a point pattern or a K-function. The hyperframe object allows us to do that.

```
dat <- data.frame(x=runif(50, 0, 10), gr = sample(1:10, 50, replace=T))
dat <- as.hyperframe(dat)
dat$pppx <- replicate(50, rpoispp(runif(1, 20, 100)), simplify=FALSE)
summary(dat)
## hyperframe with 50 rows and 3 columns
##        x                gr       pppx
##  (numeric)        (integer)      (ppp)
##  Min.   :0.2337   Min.   : 1.0
##  1st Qu.:2.3050   1st Qu.: 3.0
##  Median :4.6311   Median : 5.5
##  Mean   :4.8067   Mean   : 5.5
##  3rd Qu.:7.3686   3rd Qu.: 8.0
##  Max.   :9.9143   Max.   :10.0
```

Calculate K-functions

The models actually take the k-function as a response, not the point pattern (this could change in the future). Calculating the k-function with the spatstat function is simple enough. The two decisions we have to make are

1. What distances we want to calculate it at. This should not be a crucial decision. In general, we will want the lower bound to be 0. The upper bound can be whatever makes sense to your question (how far apart do point need to be to be independent of each other?). In general it is recommended that K should not be estimated beyond 1/4 the shortest edge of the area.
2. What edge correction to be used. Technically, border corrections are less important for replicated spatial point patterns than single point pattern analyses. However, it probably doesn't harm to include one here. The "border" correction is about the simplest correction avaialable (it considers points in a "border" area as neighbours but not focal points). We will use that here. If in doubt, play with the various options available in the `Kest` function.

```
r <- seq(0, 0.25, 0.05)
dat$k <- lapply(dat$pppx, Kest, r=r, correction='border')
```

Weights

There are a number of options in the literature (discussed in Bagchi & Illian (2015)), and several computed by the `kfuncsWeightsCalc` function. The option we will use here is "nx_A". You can try other options to see how it changes the inference, and in general, I'd recommend trying a few options to make sure that your conclusions are not too sensitive to your choice.

```
dat$wts <- lapply(dat$pppx, RSPPlme4::kfuncWeightsCalc, r=r,
                  correction="border", type="nx_A")
```

Fit the model

The model fitting itself is relatively simple, once we have all the pieces in place. The function requires that the right-hand side of the equation is a K-function object and the left-hand side is a model formula in the style of lme4. We also need to specify the distances at which K will be estimated (these have to match the ones we calculated K at, although they can be a subset). We also need to tell the function where the weights are.

In these examples, you'll probably get a few warnings about singular fits. These warnings are unsurprising given that the covariates don't actually have an effect on the point patterns.

```
mod1 <- klmerHyper(formula = k ~x + (1|gr), r=seq(0, .25, .05),
                   hyper=dat, correction='border', weights = wts,
                   na.action="na.omit")
mod1
## linear mixed model fitted to k function
## distances modelled:
##  5 distances
##  range = 0.05 0.25
## Fixed effects
##                       0.05          0.1         0.15         0.2        0.25
## (Intercept)  8.207145e-03 0.0290921207 0.0688096015 0.119327102 0.180599858
## x           -9.799398e-05 0.0001224678 0.0002665498 0.001192562 0.002742734
```

There is a print method that provides the coefficients at each distance. In this case, the (Intercept) parameter tells us what the

Get confidence intervals

The confidence intervals are still a little painful to obtain. The main challenge is defining the linear combination to use in the bootstrapping - basically, the algorithm will calculate the bootstrapped parameters multiplied by this matrix to get the bootstrapped confidence intervals of the predictions.

```
preddat <- data.frame(x=c(0, 10))
Xmat <- model.matrix(~x, data=preddat)

mod1_cis <- confint(mod1, level=0.95, lin_comb=Xmat, nboot=99, ncore=15, iseed=1234)

mod1_cis
## , , (Intercept)
##
##         estimate       2.5%       97.5%
## 0.05 0.008207145 0.00692376 0.009848911
## 0.1  0.029092121 0.02603747 0.031419789
## 0.15 0.068809601 0.06301891 0.076276838
## 0.2  0.119327102 0.11164669 0.129516521
## 0.25 0.180599858 0.16656138 0.198107576
##
## , , x
##
##           estimate          2.5%        97.5%
## 0.05 -9.799398e-05 -0.0003521620 0.0001321104
## 0.1   1.224678e-04 -0.0004188697 0.0005587098
```

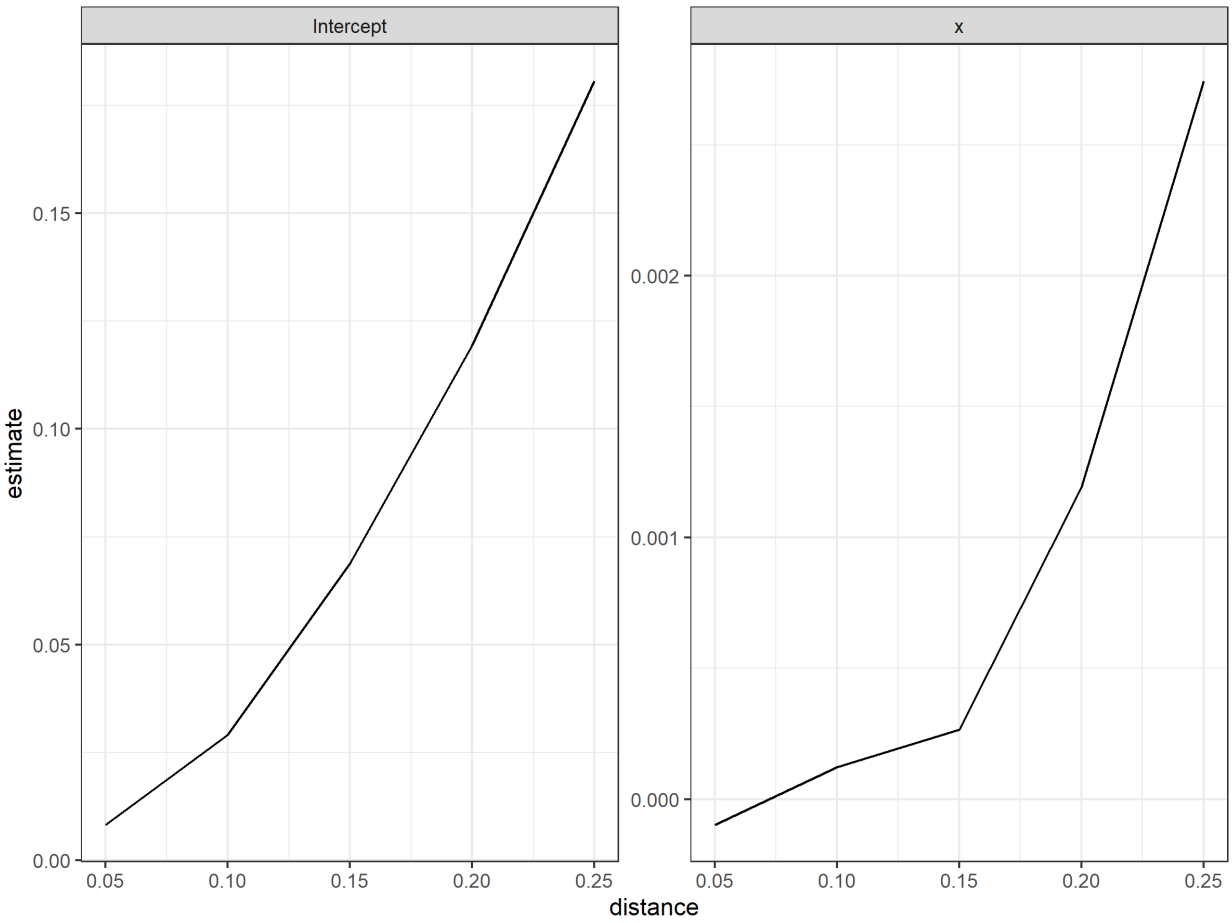
```
## 0.15  2.665498e-04 -0.0011271984 0.0012099179
## 0.2   1.192562e-03 -0.0006396041 0.0025378687
## 0.25  2.742734e-03  0.0001311124 0.0055260029
```

The print method gives the confidence intervals on the parameter estimates. Eventually, I will revise the code so that a call to `confint` without a linear predictior object provides just the cis on the parameter estimates.

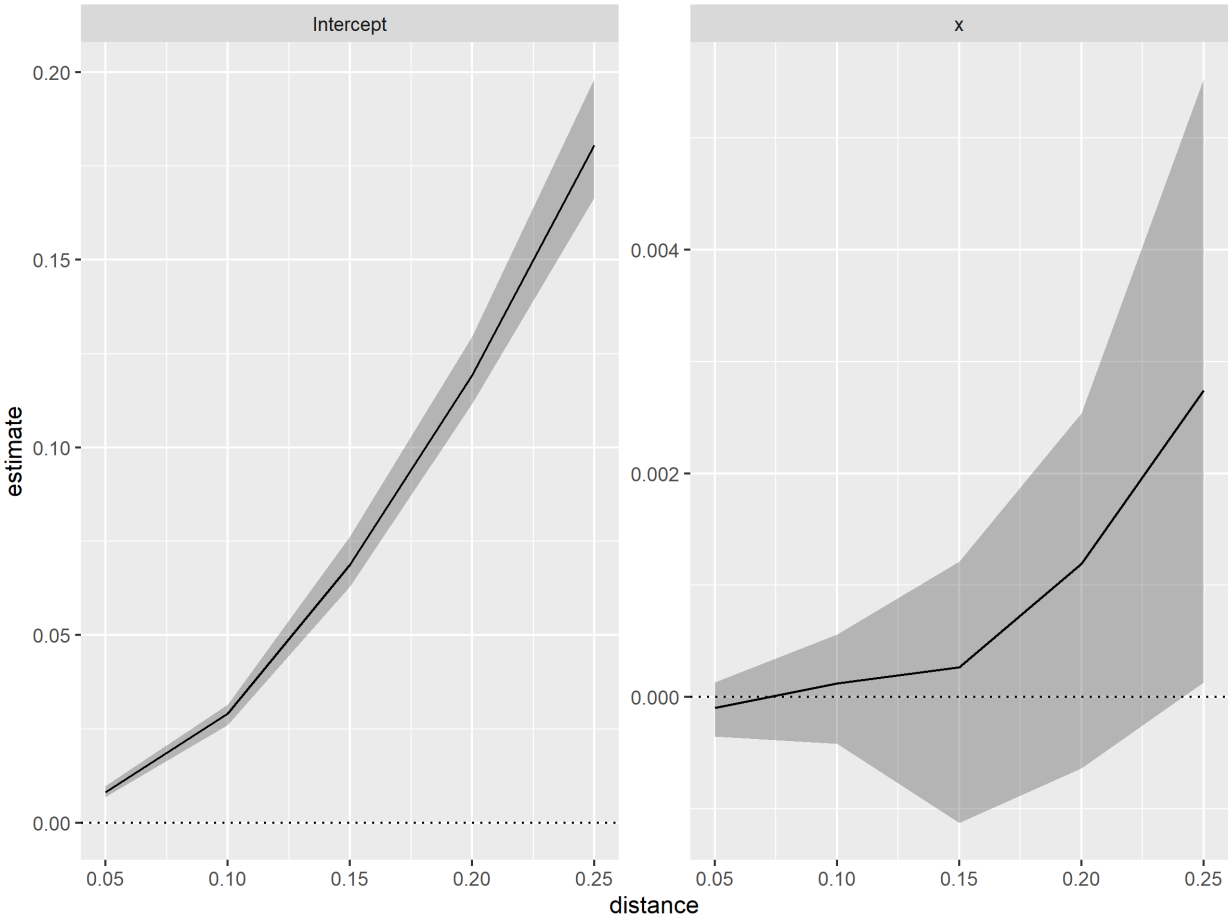
Explore the model

The package doesn't have nearly enough options for visualizing the output at the moment. However, I did write a number of functions in my last analysis which will eventually get folded into the package. They are contained in the "ConvenienceFunctions_lmer.R" script. Some of them don't work now (tidyr was in transition when I wrote them).

```
source("tutorial/ConvenienceFunctions_klmer.R")
plot(mod1)
```




```
plot(mod1_cis)
```



Session Information

```
R version 3.6.3 (2020-02-29)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17763)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] ggthemes_4.2.0  RSPPlme4_0.0.1  forcats_0.5.0   stringr_1.4.0
 [5] dplyr_1.0.2     purrr_0.3.4     readr_1.3.1     tidyr_1.1.1
 [9] tibble_3.0.3    ggplot2_3.3.2   tidyverse_1.3.0
```

```
loaded via a namespace (and not attached):
 [1] nlme_3.1-149          fs_1.5.0              usethis_1.6.1
 [4] lubridate_1.7.9       devtools_2.3.1        httr_1.4.2
 [7] rprojroot_1.3-2       tools_3.6.3           backports_1.1.9
[10] R6_2.4.1              rpart_4.1-15          mgcv_1.8-32
[13] DBI_1.1.0             colorspace_1.4-1      withr_2.2.0
[16] tidyselect_1.1.0      prettyunits_1.1.1     processx_3.4.3
[19] curl_4.3              compiler_3.6.3        cli_2.0.2
[22] rvest_0.3.6           xml2_1.3.2            desc_1.2.0
[25] scales_1.1.1          spatstat.data_1.4-3   callr_3.4.3
[28] goftest_1.2-2         spatstat_1.64-1       digest_0.6.25
[31] spatstat.utils_1.17-0 minqa_1.2.4           rmarkdown_2.3
[34] pkgconfig_2.0.3       htmltools_0.5.0       lme4_1.1-23
[37] sessioninfo_1.1.1     dbplyr_1.4.4          rlang_0.4.7
[40] readxl_1.3.1          rstudioapi_0.11       generics_0.0.2
[43] jsonlite_1.7.0        magrittr_1.5          Matrix_1.2-18
[46] Rcpp_1.0.5            munsell_0.5.0         fansi_0.4.1
[49] abind_1.4-5           lifecycle_0.2.0       stringi_1.4.6
[52] yaml_2.2.1            MASS_7.3-52           pkgbuild_1.1.0
[55] grid_3.6.3            blob_1.2.1            parallel_3.6.3
[58] crayon_1.3.4          deldir_0.1-28         lattice_0.20-41
[61] haven_2.3.1           splines_3.6.3         tensor_1.5
[64] hms_0.5.3             knitr_1.29            ps_1.3.4
[67] pillar_1.4.6          boot_1.3-25           codetools_0.2-16
[70] pkgload_1.1.0         reprex_0.3.0          glue_1.4.1
[73] evaluate_0.14         remotes_2.2.0         modelr_0.1.8
[76] vctrs_0.3.2           nloptr_1.2.2.2        testthat_2.3.2
[79] cellranger_1.1.0      polyclip_1.10-0       gtable_0.3.0
[82] assertthat_0.2.1      xfun_0.16             broom_0.7.0
[85] memoise_1.1.0         statmod_1.4.34        ellipsis_0.3.1
```