RSPPlme4 simulated example

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

March 07, 2022

Load Preliminaries

First we need to load the packages for simulating datasets, running models, organizing and plotting outputs.

```
# devtools::install_github("Bagchilab-Uconn/RSPPlme4")
library(RSPPlme4)
library(spatstat)
library(ggplot2)
library(tidyverse)
```

Data simulation

Creation of the simulated data is performed in this chunk. We generate a response variable (the distribution of points within `ppp` objects) and a random effect (`ranef`) from two fixed effects (`f` and `x`).

```
# set seed
set.seed(1234)

# set number of points
n <-  100

# create random covariate x
x <- runif(n = n, min = 0, max = 10)

# randomly assign group a or b (covariate f)
f <- sample(c("a", "b"), size = n, replace = TRUE)

# assign replicate
gr <- rep(1:10, each = n/10)

# create hyperframe
dat <- hyperframe(x = x, f = f, gr = gr)

# create basis for response variable
lin_pred1 <-  model.matrix(~ f + x, data = as.data.frame(dat, warn = FALSE)) %*%
  c(0.05, 0.05, 0.01)

# create response variable and add to hyperframe
dat$ppx1 <- lapply(lin_pred1, function(sigma)
```

```
                    rThomas(kappa =runif(n = 1, min = 3, max = 10),
                            mu = 5, sigma = sigma))

# create basis for random effect
ranef <- exp(rnorm(n = 10, mean = 0, sd = 0.2) )

# create random effect
lin_pred2  <- lin_pred1 * ranef[dat$gr]

# add random effect to hyperframe
dat$ppx2 <- lapply(lin_pred2, function(sigma)
  rThomas(kappa =runif(n = 1, min = 3, max = 10),
          mu = 10, sigma = sigma))
```

Models

Here, we run three models: the intercept-only model, the fixed effect only model, and the mixed effects model. Further, the results of these three models are plotted below.

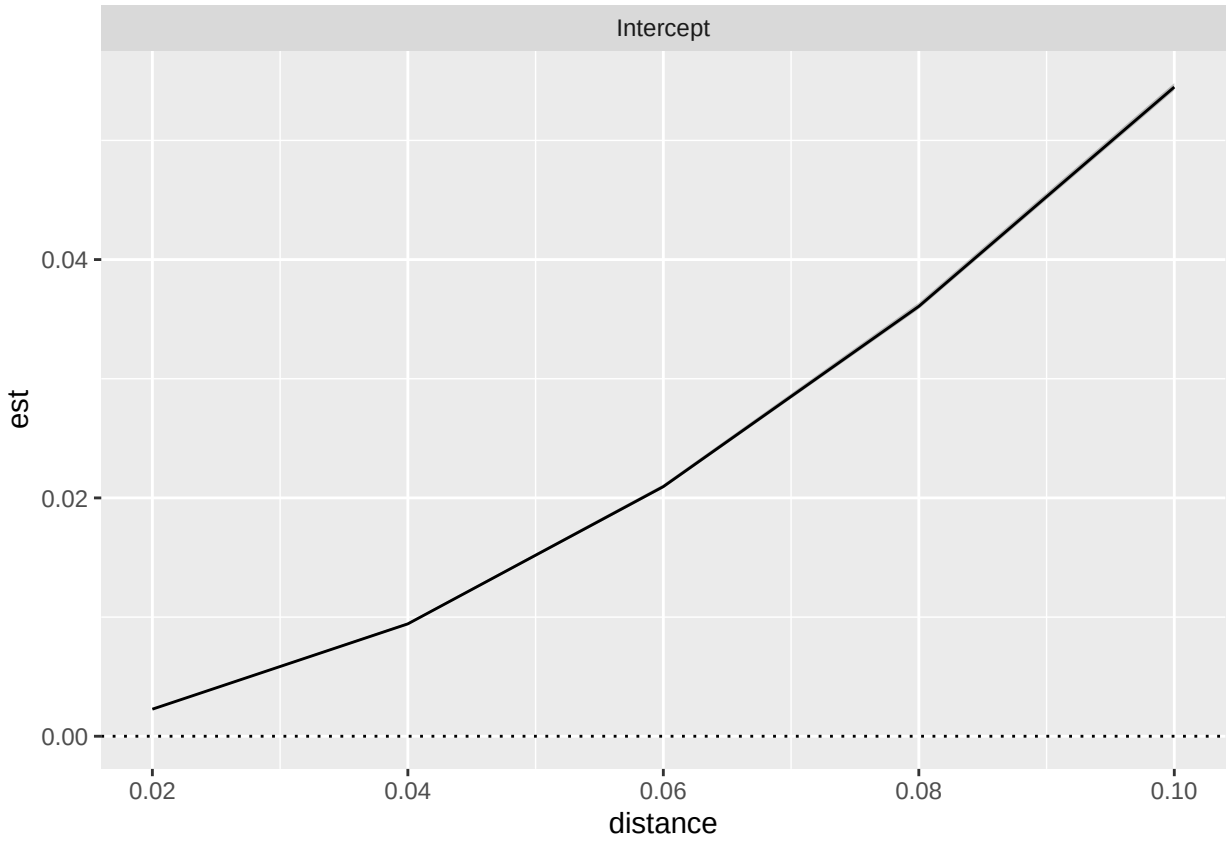
Intercept-only model

We use the 'klm' function with a model formula that only specifies an intercept. The code then computes and plots confidence intervals around the mean-K-function estimated by the model.

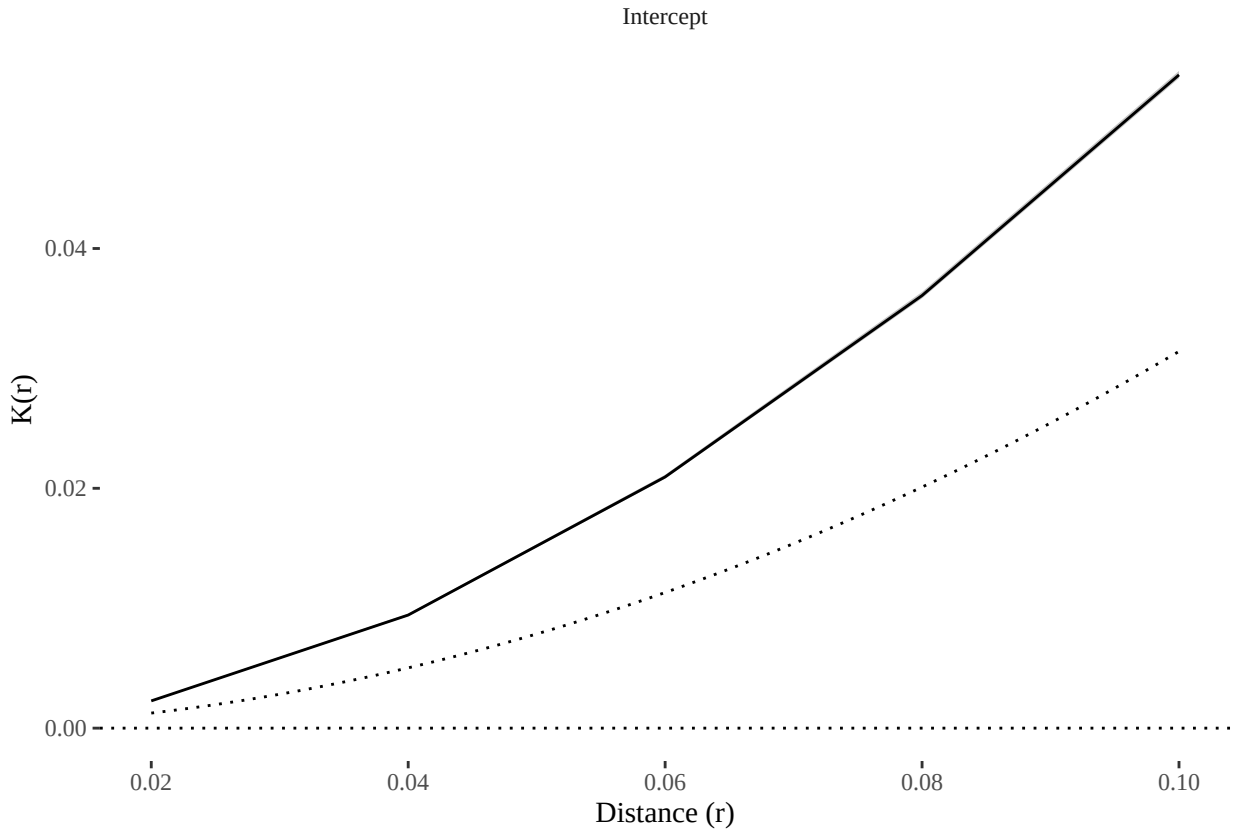
```
# model 1.1: intercept only
mod1.1 <- klm(ppx1 ~ 1, hyper = dat, r = seq(0, 0.1, 0.02),
            correction = "border", weights_type = "nx_A")

# generate confidence intervals for model 1.1
mod1.1_ci <- confint(mod1.1, nboot=500, level = 0.95, iseed = 4321)

# show model 1.1 confidence intervals
print(mod1.1_ci)
```

```
## , , (Intercept)
##
##                  0.02        0.04       0.06       0.08        0.1
## est      0.002272795 0.009430132 0.02094915 0.03607708 0.05447752
## lwr2.5%  0.002229759 0.009343741 0.02080297 0.03584231 0.05420163
## upr97.5% 0.002335217 0.009540220 0.02112979 0.03637523 0.05480533
```

```
# plot of model 1.1 fixed effects
pl <- plot(mod1.1_ci)
```



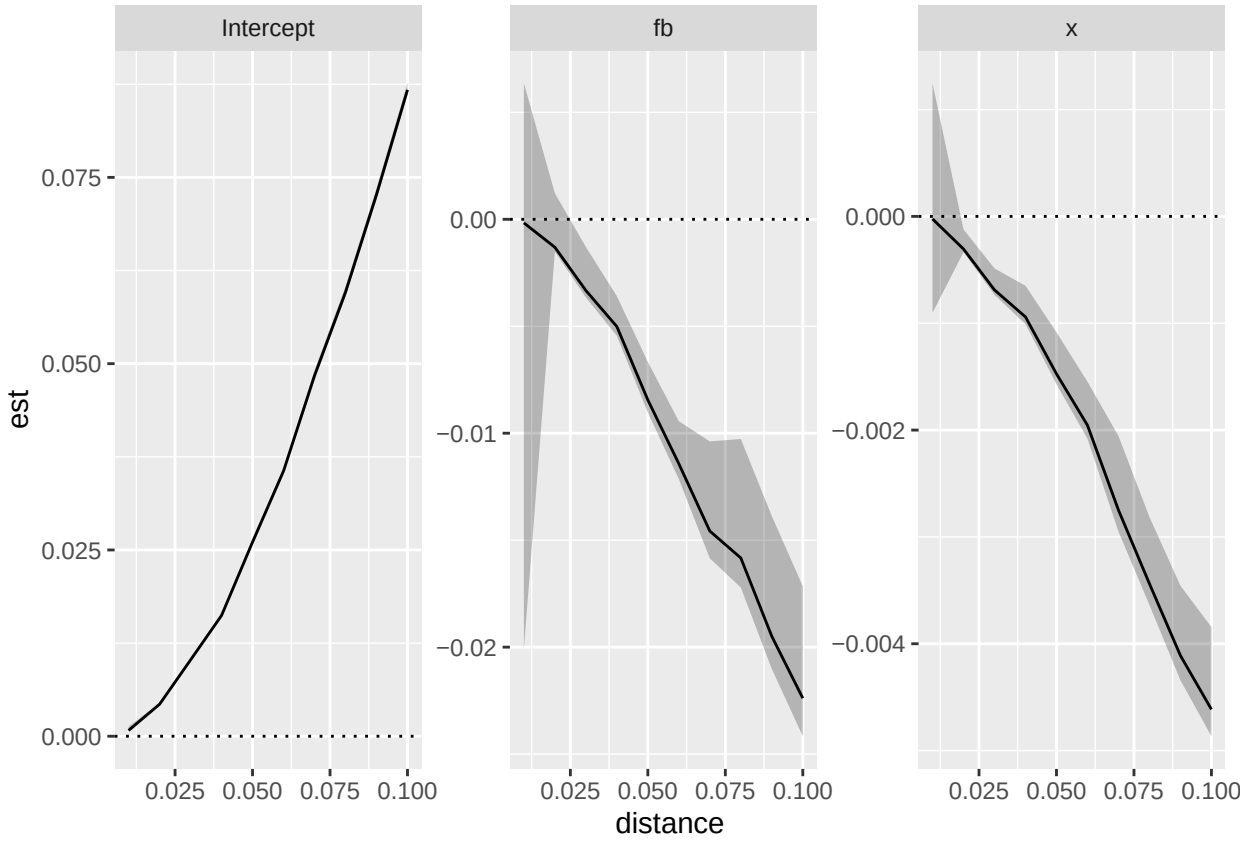

```
pl +  geom_function(fun = function(x) pi * x^2, linetype = "dotted") +
  labs(x = "Distance (r)", y = "K(r)") + ggthemes::theme_tufte()
```

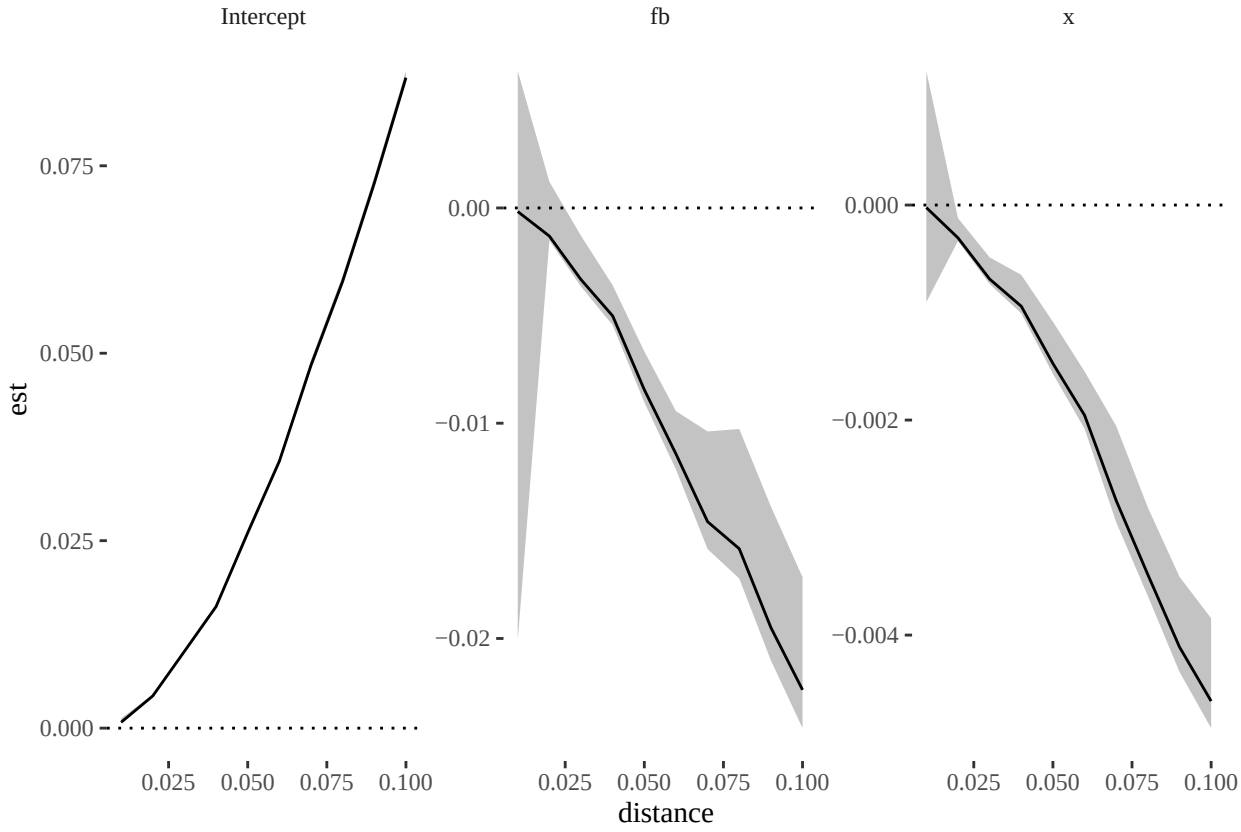


Fixed-effects-only model

In the next model, we modify the formula to include effects of `f` and `x` on the K-function. Once again, we fit the model and compute confidence intervals, then plot the parameter estimates.

```
# model 1.2: add in covariates f and x
mod1.2 <- klm(ppx1 ~ f + x, hyper = dat, r = seq(0, 0.1, 0.01),
              correction = "border", weights_type = "nx_A")

# generate confidence intervals for model 1.2
mod1.2_ci <- confint(mod1.2, nboot = 500, level = 0.99, iseed = 9876)


# plot of model 1.2 fixed effects
plot(mod1.2_ci) + ggthemes::theme_tufte()
```

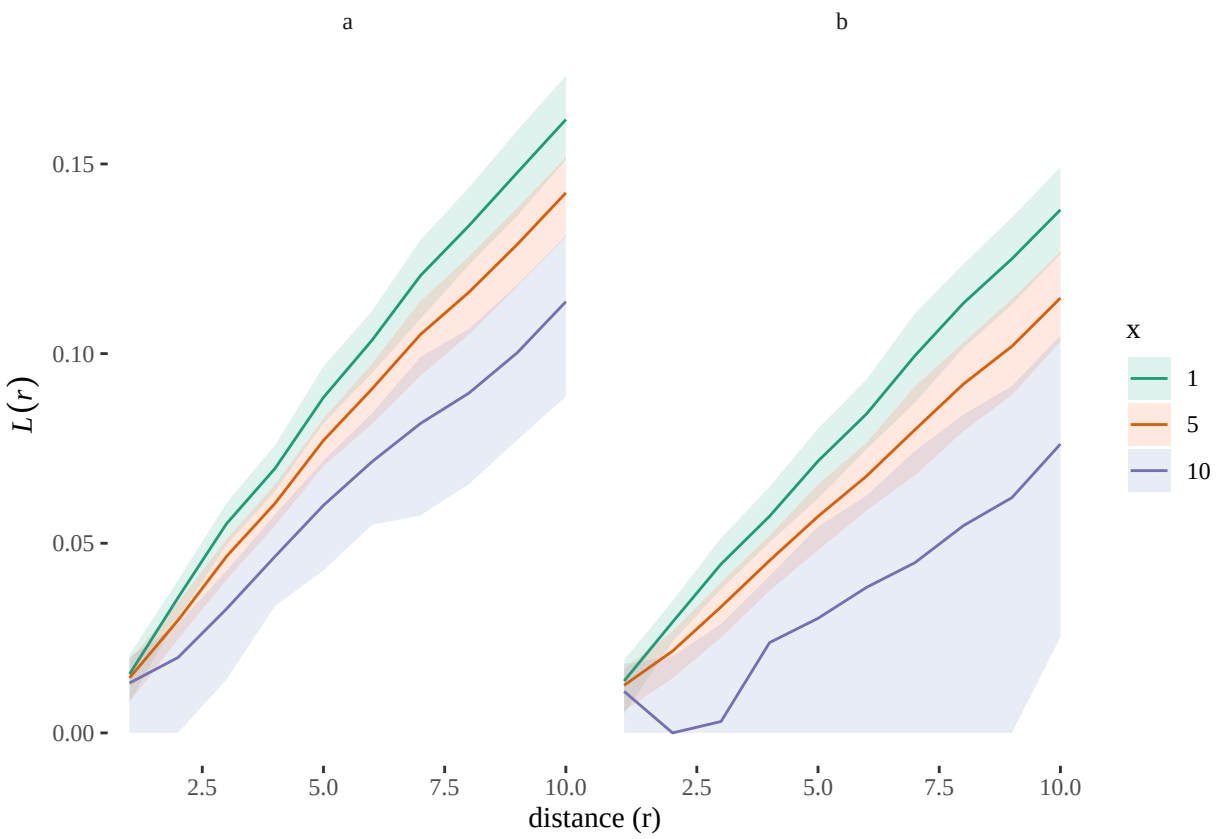

Intercept
fb
x
est
distance
0.075
0.050
0.025
0.000
0.00
−0.01
−0.02
0.000
−0.002
−0.004
0.025
0.050
0.075
0.100



It is also possible to generate predictions of the K-functions under specific combinations of covariates. We demonstrate that below. Note we plot predictions as L-functions to aid interpretability.

```
# alternate creation of model 1.2 confidence intervals
nd1 <- expand.grid(x = c(1, 5, 10),  f = c("a", "b"))
mod1.2_ci <- confint(mod1.2, nboot = 500, level = 0.99, iseed = 9876,
                     newdata=nd1)


# make model 1.2 predictions able to be plotted
preds1.2  <- as.data.frame.table(mod1.2_ci$predictions)
preds1.2[, c("x", "f")] <- nd1[as.numeric(preds1.2$Var1), c("x", "f")]
preds1.2 <- pivot_wider(preds1.2, names_from = Var3, values_from = Freq) %>%
  rename("distance" = "Var2") %>%
  select(-Var1) %>% mutate(distance = as.numeric(distance))

K2L <- function(K) sqrt(K/pi)

# plot model 1.2 predictions
ggplot(preds1.2, aes(x = distance, y = K2L(est), ymin = K2L(lwr), ymax =K2L(upr),
                     group = as.factor(x))) +
  geom_ribbon(alpha = 0.2, aes(fill = as.factor(x))) +
  geom_path(aes(color = as.factor(x))) +
  facet_wrap(~f) +
  scale_color_brewer(palette="Dark2") + scale_fill_brewer(palette="Set2") +
  ggthemes::theme_tufte() +
```

```
  labs(x = "distance (r)", y = expression(italic(L(r))), color = "x", fill = "x")
```



Mixed-effects models

Often, replicates are not independent of each other - and that dependence needs to be accounted for to reduce spurious inferences (e.g., as a result of pseudoreplication or Simpson's paradox). The `klmer` function uses syntax from the `lme4` package for mixed effects models.

```
# model 2: adding random effects
mod2 <- klmer(ppx2 ~ 1 + f + x +  (1| gr), hyper = dat, r = seq(0, 0.1, 0.01),
              correction = "border", weights_type = "nx_A")

# generate confidence intervals for model 2
mod2_ci <- confint(mod2, nboot = 500, ncore = 16)
```

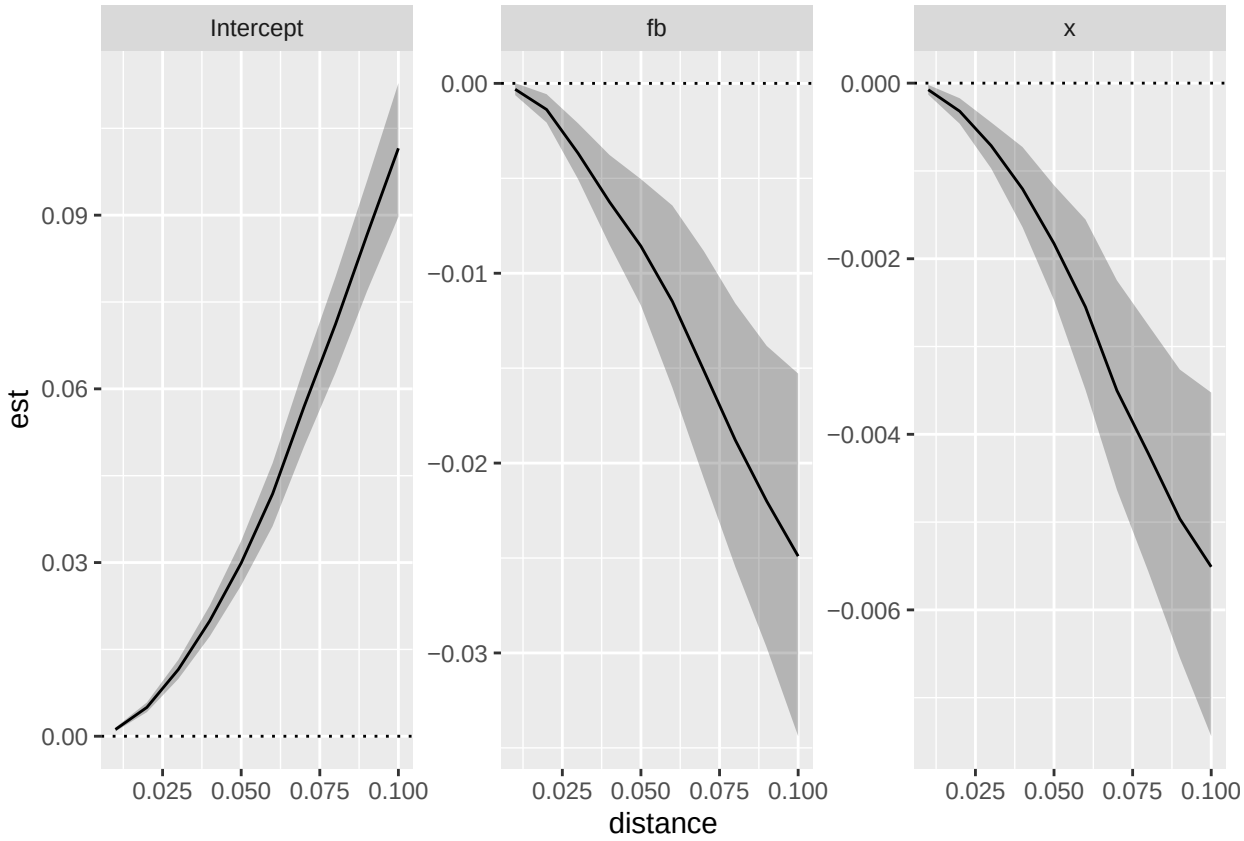
```
## clusters closed on exit
```

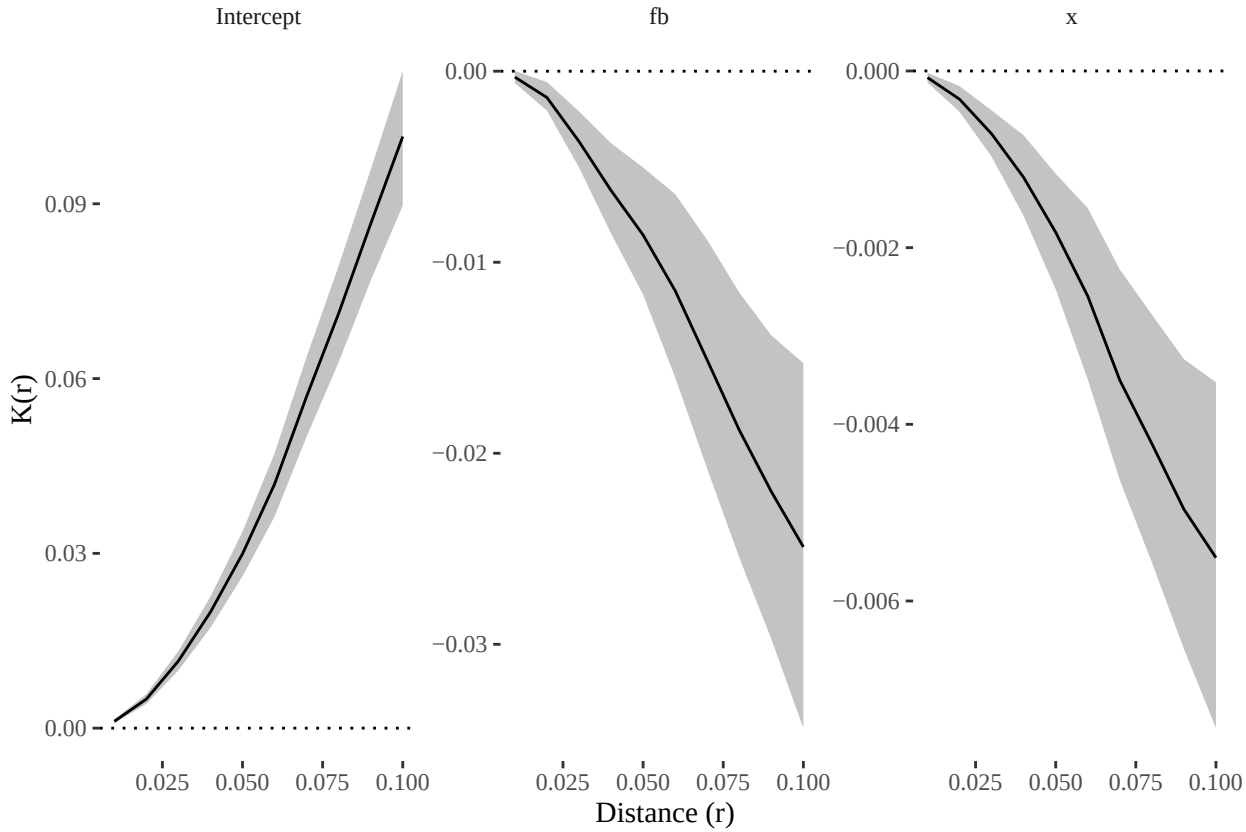
```
# show model 2 confidence intervals
print(mod2_ci)
```

```
## , , (Intercept)
##
```

```
##             0.01        0.02        0.03       0.04       0.05       0.06
## est 0.0011680260 0.004985416 0.011529417 0.01993112 0.02992305 0.04187721
## lwr 0.0008197916 0.004162228 0.009895813 0.01723944 0.02605749 0.03628505
## upr 0.0015046608 0.005832803 0.013211714 0.02257609 0.03373770 0.04718186
##           0.07       0.08       0.09        0.1
## est 0.05693056 0.07118157 0.08656927 0.10154402
## lwr 0.05001980 0.06278673 0.07681706 0.08969596
## upr 0.06372971 0.07933645 0.09585853 0.11279987
##
## , , fb
##
##              0.01          0.02         0.03         0.04         0.05
## est -3.102939e-04 -0.0013784401 -0.003676855 -0.006243774 -0.008574534
## lwr -6.173621e-04 -0.0020646042 -0.005042791 -0.008492862 -0.011700114
## upr  1.221315e-05 -0.0005688738 -0.002102004 -0.003768356 -0.005041947
##             0.06         0.07        0.08        0.09         0.1
## est -0.011487204 -0.015125508 -0.01878599 -0.02200701 -0.02490362
## lwr -0.016027769 -0.020813562 -0.02545901 -0.02972015 -0.03438776
## upr -0.006431394 -0.008834535 -0.01158551 -0.01382611 -0.01528585
##
## , , x
##
##              0.01          0.02          0.03          0.04         0.05
## est -7.486473e-05 -0.0003201355 -0.0007118293 -0.0012055738 -0.001827317
## lwr -1.307212e-04 -0.0004624090 -0.0009705949 -0.0016423075 -0.002472120
## upr -2.006626e-05 -0.0001716925 -0.0004465533 -0.0007280485 -0.001163610
##             0.06         0.07         0.08         0.09          0.1
## est -0.002549483 -0.003500892 -0.004218620 -0.004961898 -0.005509386
## lwr -0.003492092 -0.004631314 -0.005563354 -0.006542172 -0.007439245
## upr -0.001553059 -0.002247077 -0.002756479 -0.003262015 -0.003524346
```

```
# plot model 2 fixed effects
plot(mod2_ci) + ggthemes::theme_tufte() +
  labs(x = "Distance (r)", y = "K(r)")
```


Intercept
fb
x
est
distance
0.09
0.06
0.03
0.00
−0.01
−0.02
−0.03
0.000
−0.002
−0.004
−0.006
0.025
0.050
0.075
0.100

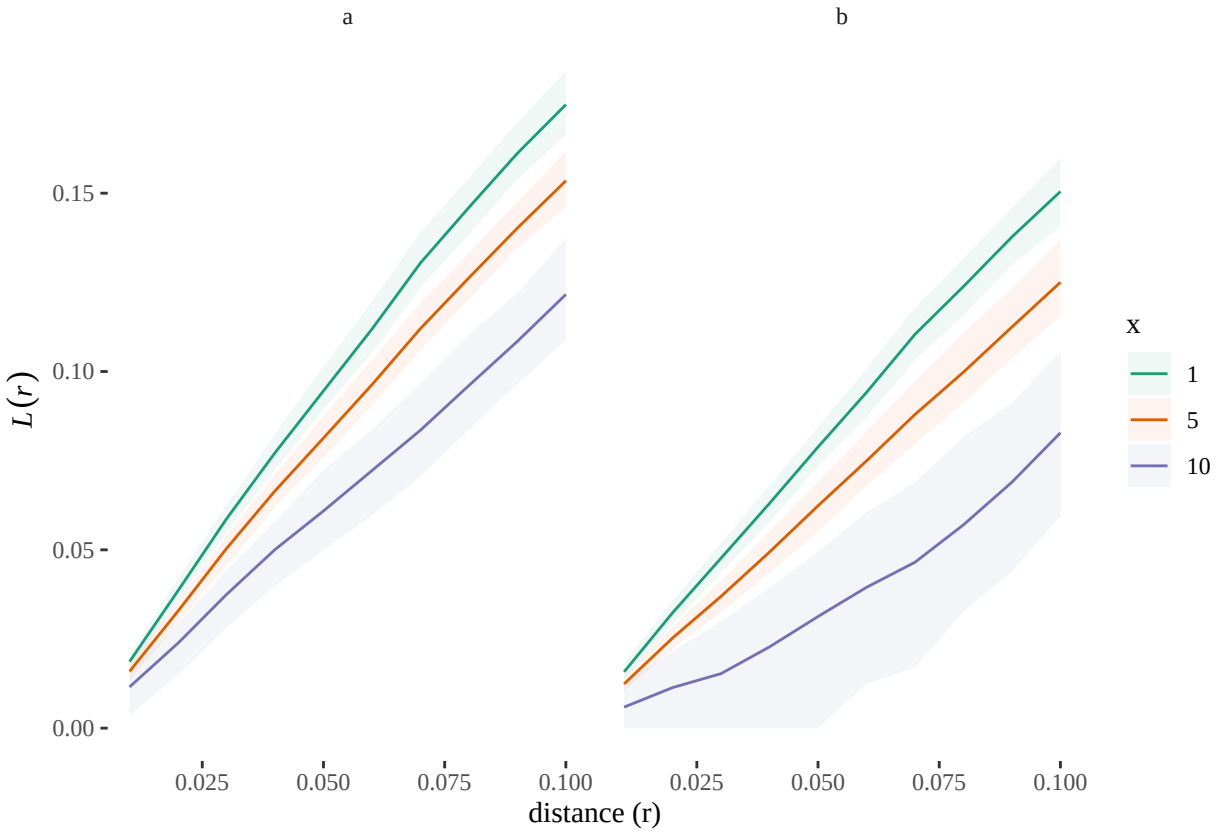


As in the case of the fixed-effects only models, it is possible to make predictions with associated confidence intervals for specific combinations of covariates. These are generated and plotted below.

```
# alternative creation of model 2 confidence intervals
mod2_ci <- confint(mod2, nboot = 99, ncore = 16,
                   newdata  = nd1)
```

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

```
# make model 2 predictions able to be plotted
preds2  <- as.data.frame.table(mod2_ci$predictions)
preds2[, c("x", "f")] <- nd1[as.numeric(preds2$Var2), c("x", "f")]
preds2 <- pivot_wider(preds2, names_from = Var3, values_from = Freq) %>%
  rename("distance" = "Var1") %>%
  mutate(distance = as.numeric(as.character(distance)))

# plot model 2 predictions
ggplot(preds2, aes(x = distance, y = K2L(est), ymin = K2L(lwr), ymax = K2L(upr),
                   colour = as.factor(x), fill = as.factor(x))) + facet_wrap(~f) +
  geom_ribbon(alpha = 0.1, color = NA, aes(group = as.factor(x))) +geom_path() +
  scale_color_brewer(palette="Dark2") + scale_fill_brewer(palette="Set2") +
  labs(x = "distance (r)", y = expression(italic(L(r))), color = "x", fill = "x") +
  ggthemes::theme_tufte()
```


a
b
0.15
0.10
0.05
0.00
L(r)
0.025
0.050
0.075
0.100
distance (r)
x
1
5
10