Re-doing Lab 3

:)

28 April, 2023 13:14:49

Complete the following exercises before the submission deadline. In addition to the points detailed below, 5 points are assigned to the quality of the annotation, as well as to the 'cleanliness' of the code and resulting pdf document.

Exercise 1 – 1 point

We will again be working with the BC Parks dataset, which contains information on the locations of Provincial Parks in British Columbia. The parks belong to 5 different regions. There is also information on elevation (in m) and percent forest cover contained within the dataset.

- Import the BC park locations dataset and convert the data to a `ppp` object (for today you can exclude information on regions). – 1 point(s)

Note: You will need to load the `maptools` package and make use of the `as.owin()` function.

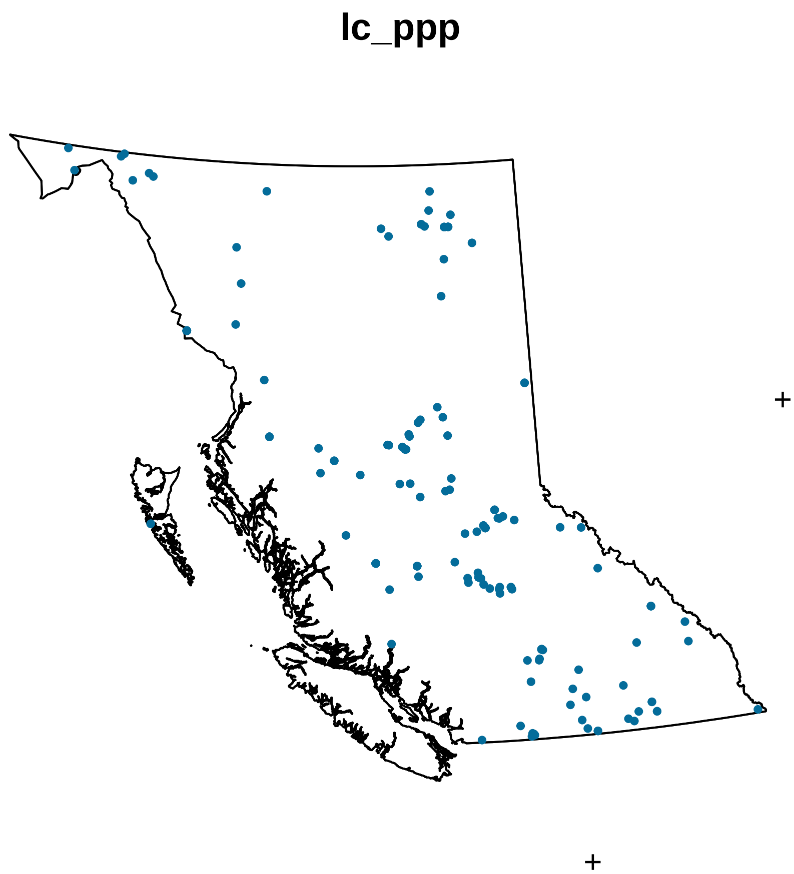
```
# 1
#quiet function
quiet <- function(x) {
  sink(tempfile())
  on.exit(sink())
  invisible(force(x))
}

#importing dataset
lc = read.csv("../datasets/processed/lc.csv")

#importing window
suppressMessages(library(spatstat))
suppressMessages(library(sf))
suppressMessages(library(maptools))
load("../datasets/raw/BC_Covariates.Rda")
bc_window_sf = st_as_sf(DATA$Window)
bc_window_owin = as.owin(bc_window_sf)

#converting to a ppp
lc_ppp = ppp(x = lc$decimalLongitude,
              y = lc$decimalLatitude,
              window = bc_window_owin,
              )
```

```
## Warning: 2 points were rejected as lying outside the specified window
```

```
## Warning: data contain duplicated points
```

```
plot(lc_ppp, pch = 16, cols = "#046C9A", cex = 0.6)
```

```
## Warning in plot.ppp(lc_ppp, pch = 16, cols = "#046C9A", cex = 0.6): 2 illegal
## points also plotted
```



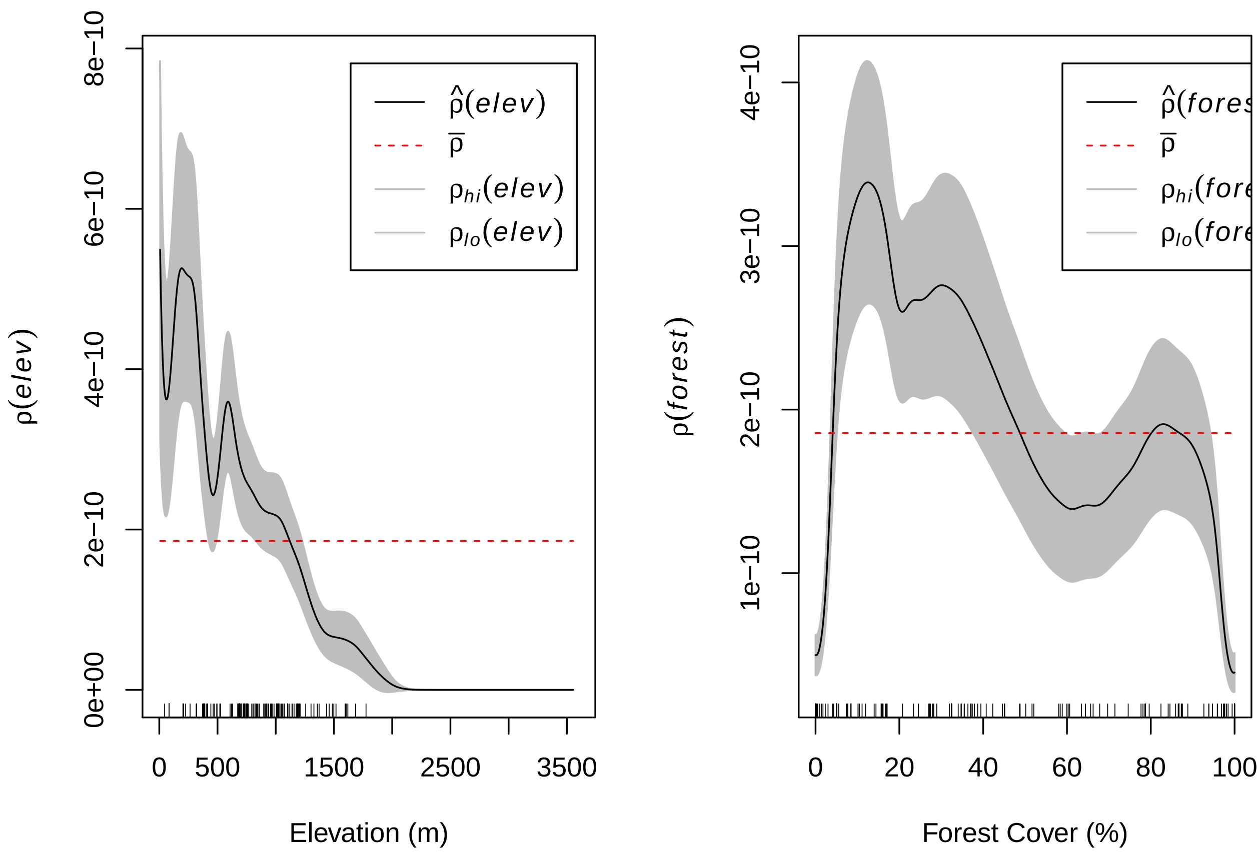
Exercise 2 – 4 points

- Estimate and plot ρ for the locations of parks as a function of both elevation and forest cover (be sure that the x-axis for elevation does not go below 0). – 2 point(s)
- Check for collinearity between elevation and forest cover (you will need to consider NA values). – 1 point(s)
- Based on these initial analyses, write down the expected form of the model. Provide justification for this starting point. – 1 point(s)

Note: Estimating rho can be slow ($\sim$ 1-2 min). Be sure to leave enough time for the document to knit.

```
# 1
elev = DATA$Elevation
rho_elev = rhohat(lc_ppp, elev)
forest = DATA$Forest
rho_forest = rhohat(lc_ppp, forest)
par(mfrow = c(1,2))
plot(rho_elev,
     main = "",
     xlab = "Elevation (m)",
     xlim=c(0, 3600))
plot(rho_forest,
```

```
     main = "",
     xlab = "Forest Cover (%)")
```



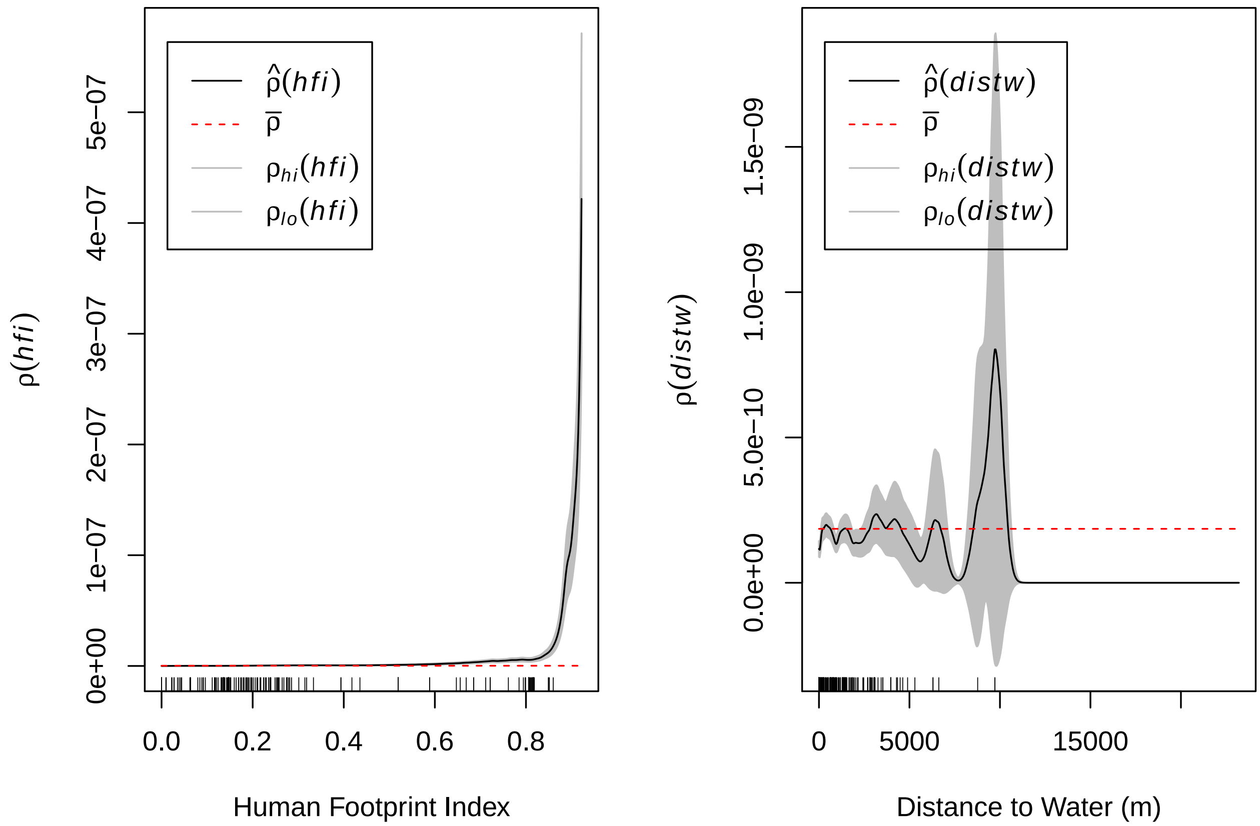

```
par(mfrow = c(1,1))
```

HFI is based on: population density, land transformation, human access, and power infrastructure.

```
# 1
hfi = DATA$HFI
rho_hfi = rhohat(lc_ppp, hfi)
```

```
## Warning: Values for 1 query point lying outside the pixel image domain were
## estimated by projection to the nearest pixel
```

```
distw = DATA$Dist_Water
rho_distw = rhohat(lc_ppp, distw)
par(mfrow = c(1,2))
plot(rho_hfi,
     main = "",
     xlab = "Human Footprint Index")
plot(rho_distw,
     main = "",
     xlab = "Distance to Water (m)")
```




```
par(mfrow = c(1,1))

#correlation
cor.im(elev, forest, hfi, distw,  use = 'complete.obs')
```

```
##             ..1         ..2         ..3         ..4
## ..1  1.00000000 -0.26225376 -0.26625626 -0.03493453
## ..2 -0.26225376  1.00000000  0.06618592  0.04818598
## ..3 -0.26625626  0.06618592  1.00000000  0.13246899
## ..4 -0.03493453  0.04818598  0.13246899  1.00000000
```

Exercise 3 – 4 points

- Fit the model you have defined in exercise 2 and inspect the model output. – 1 point(s)
- Fit a null, intercept only model. – 1 point(s)
- Use AIC and a likelihood ratio test to determine if the model you defined is a better fit than the intercept only model. – 1 point(s)
- Write down the equation for the selected model. – 0.5 point(s)
- Use this equation to estimate the intensity of parks at 500m elevation and 50% forest cover.

```
# 0
#mean centering and scaling the elevation and distance to water variables
mu <- mean(DATA$Elevation)
stdev <- sd(DATA$Elevation)
Elevation_scaled <- eval.im((elev - mu)/stdev, DATA)
mu <- mean(DATA$Dist_Water)
stdev <- sd(DATA$Dist_Water)
Dist_Water_scaled <- eval.im((distw - mu)/stdev, DATA)
```

```r
# 1
fit1 = ppm(lc_ppp ~ Elevation_scaled + I(Elevation_scaled^2) + forest + I(forest^2) + hfi + I(hfi^2) + I
```

```
## Warning: Values of the covariate 'hfi' were NA or undefined at 0.44% (3 out of
## 689) of the quadrature points. Occurred while executing: ppm.ppp(Q = lc_ppp,
## trend = ~Elevation_scaled + I(Elevation_scaled^2) +

## Warning: glm.fit: algorithm did not converge
```

```r
fit1
```

```
## Nonstationary Poisson process
## Fitted to point pattern dataset 'lc_ppp'
##
## Log intensity:  ~Elevation_scaled + I(Elevation_scaled^2) + forest +
## I(forest^2) + hfi + I(hfi^2) + Dist_Water_scaled + I(Dist_Water_scaled^2)
##
## Fitted trend coefficients:
##           (Intercept)      Elevation_scaled  I(Elevation_scaled^2)
##         -2.345722e+01         -5.896126e-01          -2.159342e-01
##                forest           I(forest^2)                    hfi
##         -7.122462e-03          6.673426e-05           8.483340e+00
##              I(hfi^2)     Dist_Water_scaled I(Dist_Water_scaled^2)
##         -5.814717e+00         -2.233870e-01          -2.987220e-03
##
##                           Estimate         S.E.       CI95.lo       CI95.hi
## (Intercept)           -2.345722e+01 2.395698e-01 -2.392677e+01 -22.987674680
## Elevation_scaled      -5.896126e-01 1.371295e-01 -8.583814e-01  -0.320843734
## I(Elevation_scaled^2) -2.159342e-01 8.596659e-02 -3.844256e-01  -0.047442748
## forest                -7.122462e-03 8.619863e-03 -2.401708e-02   0.009772159
## I(forest^2)            6.673426e-05 8.647235e-05 -1.027484e-04   0.000236217
## hfi                    8.483340e+00 1.175667e+00  6.179076e+00  10.787604237
## I(hfi^2)              -5.814717e+00 1.345916e+00 -8.452663e+00  -3.176769595
## Dist_Water_scaled     -2.233870e-01 1.193615e-01 -4.573313e-01   0.010557262
## I(Dist_Water_scaled^2) -2.987220e-03 4.558796e-02 -9.233798e-02   0.086363542
##                       Ztest         Zval
## (Intercept)             *** -97.91393254
## Elevation_scaled        ***  -4.29967786
## I(Elevation_scaled^2)     *  -2.51183826
## forest                       -0.82628484
## I(forest^2)                   0.77174104
## hfi                     ***   7.21577068
## I(hfi^2)                ***  -4.32026696
## Dist_Water_scaled            -1.87151621
## I(Dist_Water_scaled^2)       -0.06552651
## Problem:
##  Values of the covariate 'hfi' were NA or undefined at 0.44% (3 out of 689) of
## the quadrature points
##
## *** Fitting algorithm for 'glm' did not converge ***
```

```r
# 1.5
fit2 = ppm(lc_ppp ~ Elevation_scaled + I(Elevation_scaled^2) + hfi + I(hfi^2))
```

```
## Warning: Values of the covariate 'hfi' were NA or undefined at 0.44% (3 out of
## 689) of the quadrature points. Occurred while executing: ppm.ppp(Q = lc_ppp,
```

```
## trend = ~Elevation_scaled + I(Elevation_scaled^2) +

## Warning: glm.fit: algorithm did not converge
```

```r
fit2
```

```
## Nonstationary Poisson process
## Fitted to point pattern dataset 'lc_ppp'
##
## Log intensity:  ~Elevation_scaled + I(Elevation_scaled^2) + hfi + I(hfi^2)
##
## Fitted trend coefficients:
##           (Intercept)      Elevation_scaled I(Elevation_scaled^2)
##           -23.5142168            -0.5496413            -0.1806727
##                   hfi               I(hfi^2)
##             7.9345208            -5.1926304
##
##                          Estimate       S.E.      CI95.lo     CI95.hi  Ztest
## (Intercept)           -23.5142168 0.17827617 -23.8636317 -23.1648020   ***
## Elevation_scaled       -0.5496413 0.13384529  -0.8119732  -0.2873093   ***
## I(Elevation_scaled^2)  -0.1806727 0.08343033  -0.3441932  -0.0171523     *
## hfi                     7.9345208 1.15645659   5.6679075  10.2011340   ***
## I(hfi^2)               -5.1926304 1.33813978  -7.8153361  -2.5699246   ***
##                            Zval
## (Intercept)           -131.897701
## Elevation_scaled        -4.106542
## I(Elevation_scaled^2)   -2.165552
## hfi                      6.861062
## I(hfi^2)                -3.880484
## Problem:
##  Values of the covariate 'hfi' were NA or undefined at 0.44% (3 out of 689) of
## the quadrature points
##
## *** Fitting algorithm for 'glm' did not converge ***
```

```r
# 2
fit_intercept = ppm(lc_ppp ~ 1)
fit_intercept
```

```
## Stationary Poisson process
## Fitted to point pattern dataset 'lc_ppp'
## Intensity: 1.856026e-10
##              Estimate       S.E.   CI95.lo   CI95.hi Ztest      Zval
## log(lambda) -22.40741 0.07537784 -22.55515 -22.25968   *** -297.2679
```

```r
# 3
AIC(fit1); AIC(fit2); AIC(fit_intercept)
```

```
## [1] 7963.644

## [1] 7965.547

## [1] 8241.409
```

```r
AIC(fit2) - AIC(fit1) #delta AIC
```

```
## [1] 1.903073
```

```r
anova(fit2, fit1, test = "LRT") #LRT
```

```
## Analysis of Deviance Table
##
## Model 1: ~Elevation_scaled + I(Elevation_scaled^2) + hfi + I(hfi^2)   Poisson
## Model 2: ~Elevation_scaled + I(Elevation_scaled^2) + forest + I(forest^2) + hfi + I(hfi^2) + Dist_Wa
##   Npar Df Deviance Pr(>Chi)
## 1    5
## 2    9  4   9.9031  0.04209 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

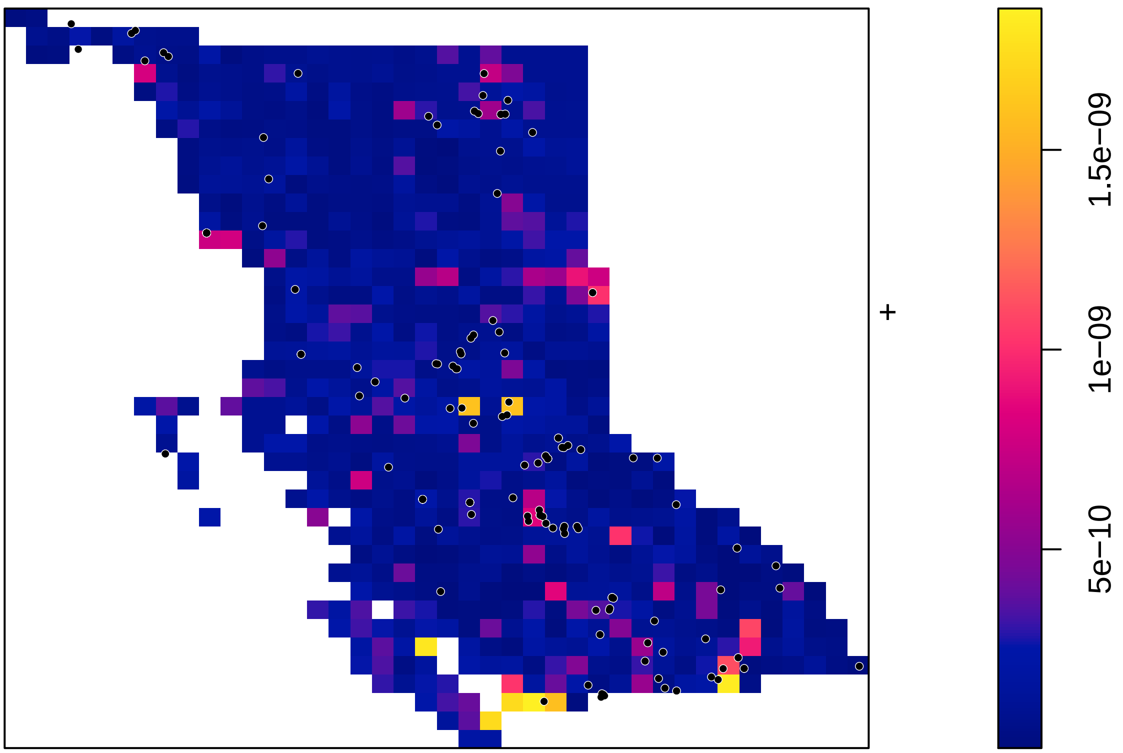
```r
#the positive delta AIC and the small p-value from the LRT both point
#to the more complex model being more parsimonious (better fit and
#worth it)

#plotting ppm prediction
plot(fit2,
     se = FALSE,
     superimpose = FALSE)
plot(lc_ppp,
     pch = 16,
     cex = 0.6,
     cols = "white",
     add = TRUE)
```

```
## Warning in plot.ppp(lc_ppp, pch = 16, cex = 0.6, cols = "white", add = TRUE): 2
## illegal points also plotted
```

```r
plot(lc_ppp,
     pch = 16,
     cex = 0.5,
     cols = "black",
     add = TRUE)
```

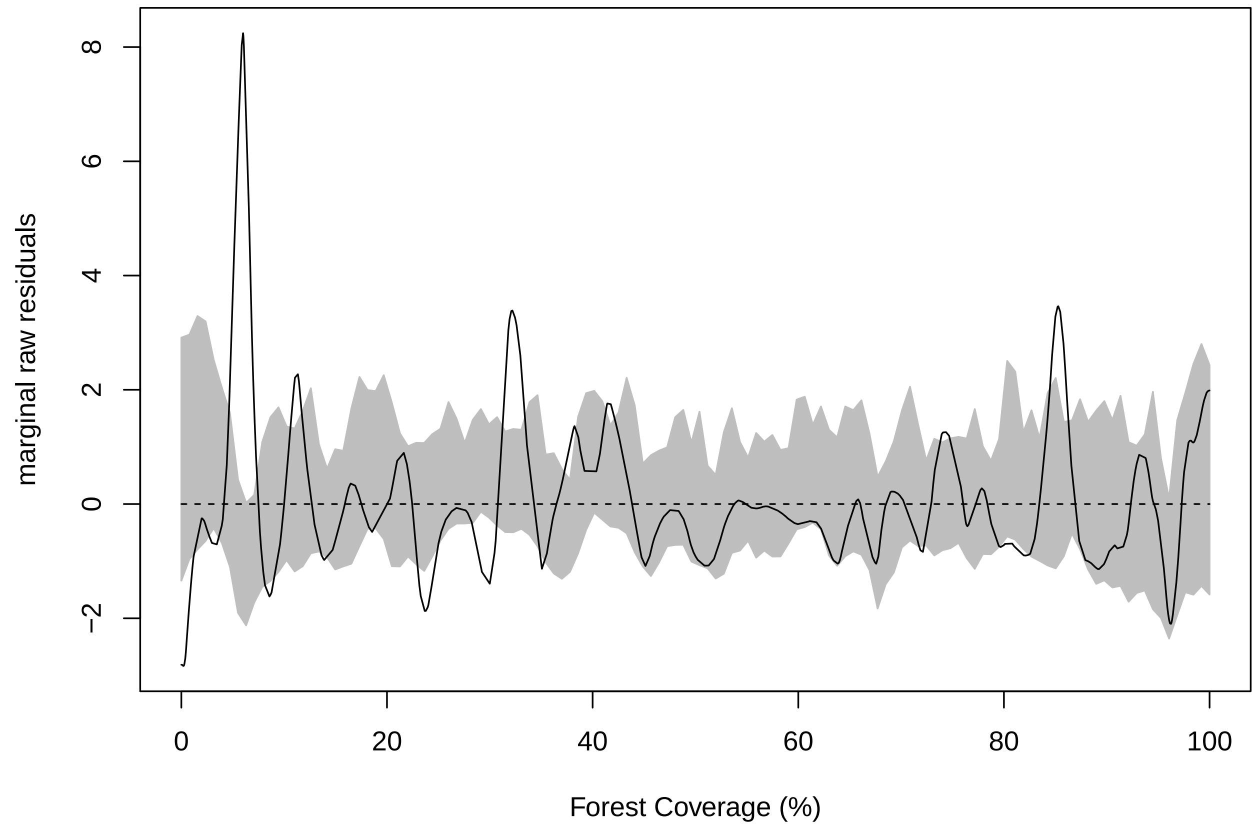
```
## Warning in plot.ppp(lc_ppp, pch = 16, cex = 0.5, cols = "black", add = TRUE): 2
## illegal points also plotted
```

Fitted trend




```
#lurking variable plot
lurking(fit2, forest, type = "raw", cumulative = F, envelope = T, xlab = "Forest Coverage (%)" )
```

```
## Generating 39 simulated patterns ...1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
## Processing.. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 2
## Done.
```




```r
##############################
#### GAM
##############################

library(splines)
fit_smooth = suppressWarnings(ppm(lc_ppp ~ bs(Elevation_scaled,7) + bs(Dist_Water_scaled, 8) +
                   bs(forest, 8) + bs(hfi, 8), use.gam = TRUE))
fit_smooth
```

```
## Nonstationary Poisson process
## Fitted to point pattern dataset 'lc_ppp'
##
## Log intensity:  ~bs(Elevation_scaled, 7) + bs(Dist_Water_scaled, 8) +
## bs(forest, 8) + bs(hfi, 8)
##
## Fitted trend coefficients:
##               (Intercept)  bs(Elevation_scaled, 7)1  bs(Elevation_scaled, 7)2
##              -22.96513631               -0.94389724               -0.48981157
##  bs(Elevation_scaled, 7)3  bs(Elevation_scaled, 7)4  bs(Elevation_scaled, 7)5
##               -0.83373754                0.30672573               -8.21252341
##  bs(Elevation_scaled, 7)6  bs(Elevation_scaled, 7)7 bs(Dist_Water_scaled, 8)1
##               33.55226769             -540.20548248                0.20125347
## bs(Dist_Water_scaled, 8)2 bs(Dist_Water_scaled, 8)3 bs(Dist_Water_scaled, 8)4
##               -1.57169547                0.48002614               -0.48025864
## bs(Dist_Water_scaled, 8)5 bs(Dist_Water_scaled, 8)6 bs(Dist_Water_scaled, 8)7
##               -0.05122473               -1.57006064               -1.10168897
## bs(Dist_Water_scaled, 8)8            bs(forest, 8)1            bs(forest, 8)2
##               -1.20304328                0.24114928                0.65429629
```

```
##            bs(forest, 8)3            bs(forest, 8)4            bs(forest, 8)5
##               -0.90553128                0.88290517               -1.44660901
##            bs(forest, 8)6            bs(forest, 8)7            bs(forest, 8)8
##                0.42271769               -0.61895864               -0.11997544
##               bs(hfi, 8)1               bs(hfi, 8)2               bs(hfi, 8)3
##               -0.13633758                1.11952518               -0.04440640
##               bs(hfi, 8)4               bs(hfi, 8)5               bs(hfi, 8)6
##                1.73588552                2.20719587                2.22319481
##               bs(hfi, 8)7               bs(hfi, 8)8
##                3.97635521                3.81909731
##
## For standard errors, type coef(summary(x))
## Problem:
##  Values of the covariate 'hfi' were NA or undefined at 0.44% (3 out of 689) of
## the quadrature points
```

```
#checking AIC
AIC(fit2); AIC(fit_smooth)
```

```
## [1] 7965.547
```

```
## [1] 7967.478
```

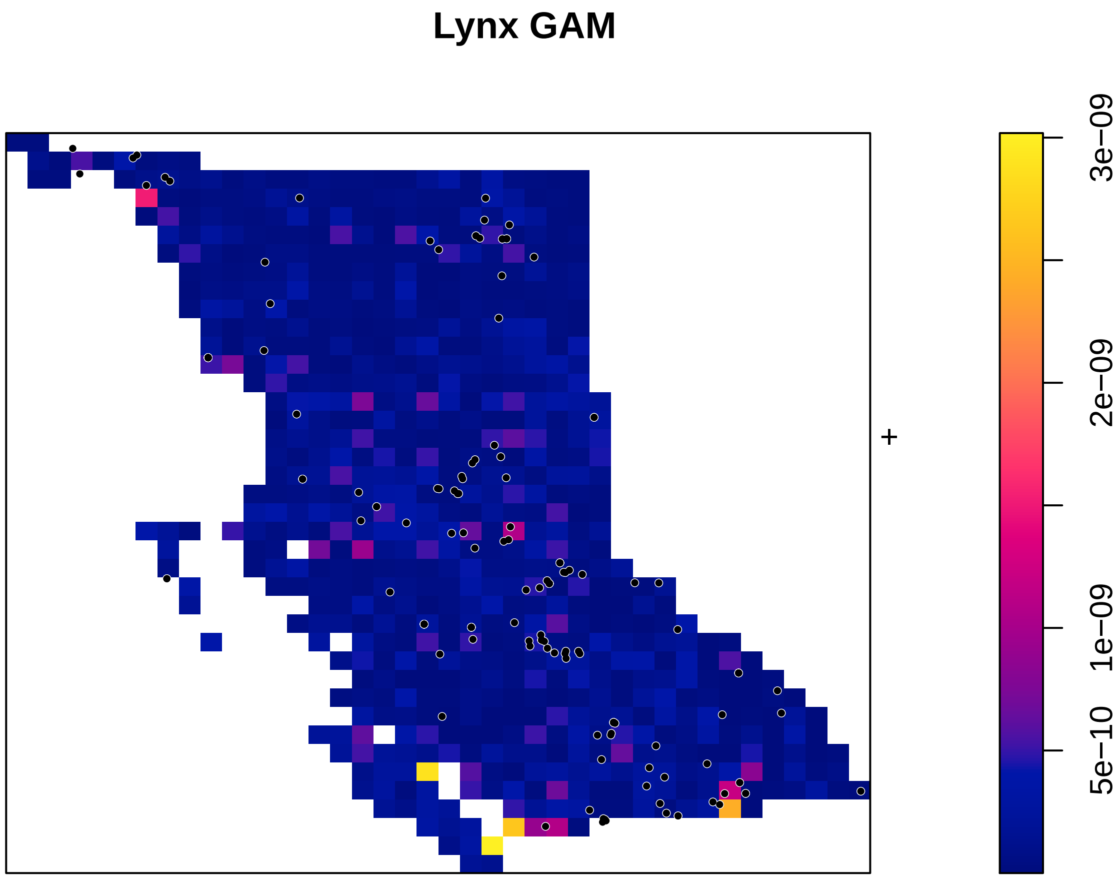
```
AIC(fit2) - AIC(fit_smooth)
```

```
## [1] -1.930526
```

```
anova(fit2, fit_smooth, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: ~Elevation_scaled + I(Elevation_scaled^2) + hfi + I(hfi^2)   Poisson
## Model 2: ~bs(Elevation_scaled, 7) + bs(Dist_Water_scaled, 8) + bs(forest, 8) + bs(hfi, 8)     Poisso
##   Npar Df Deviance Pr(>Chi)
## 1    5
## 2   32 27   52.069 0.002602 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Plot the model predictions
plot(fit_smooth,
     se = FALSE,
     superimpose = FALSE,
     main = "Lynx GAM")
plot(lc_ppp,
     pch = 16,
     cex = 0.6,
     cols = "white",
     add = TRUE)
plot(lc_ppp,
     pch = 16,
     cex = 0.5,
     cols = "black",
     add = TRUE)
```



Exercise 4 – 4 points

- Visualise the fitted model. Note: log scale the estimated intensity when plotting, ignore the standard error. You can use the `n` argument to adjust the resolution – 1 point(s)
- Plot the effects of the individual coefficients. Note: use the median value(s) of the other coefficients. – 2 point(s)
- Visually, do you think the model predictions are a good match to the data? – 1 point(s)

```
#See above
```

Exercise 5 – 1 point

- Test whether the observed data deviate significantly from the model predictions. – 1 point(s)

```
# 1
library(splines)
quadrat.test(fit2, nx = 2, ny = 4)
```

```
##
##  Chi-squared test of fitted Poisson model 'fit2' using quadrat counts
##
## data:  data from fit2
## X2 = 56.478, df = 3, p-value = 6.641e-12
## alternative hypothesis: two.sided
##
## Quadrats: 8 tiles (irregular windows)
```

```r
quadrat.test(fit_smooth, nx = 2, ny = 4) #doesnt run
```

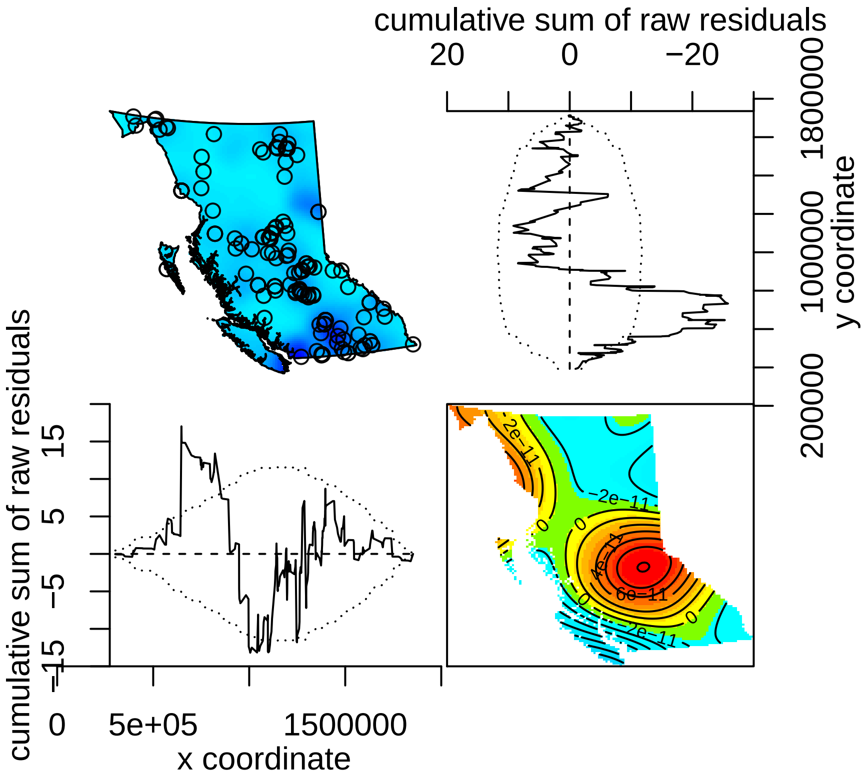
```
## Warning in bs(Elevation_scaled, degree = 3L, knots = c(`20%` =
## -1.1517265779333, : some 'x' values beyond boundary knots may cause
## ill-conditioned bases

## Warning in bs(hfi, degree = 3L, knots = c(`16.66667%` = 1.18784787446202e-05, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(Dist_Water_scaled, degree = 3L, knots = c(`16.66667%` =
## -0.829604142606883, : some 'x' values beyond boundary knots may cause
## ill-conditioned bases

## Warning in pchisq(X2, df, lower.tail = FALSE): NaNs produced

## Warning in pchisq(X2, df, lower.tail = TRUE): NaNs produced

##
##  Chi-squared test of fitted Poisson model 'fit_smooth' using quadrat
##  counts
##
## data:  data from fit_smooth
## X2 = 60.932, df = -24, p-value = NA
## alternative hypothesis: two.sided
##
## Quadrats: 8 tiles (irregular windows)
```

```r
#The small p value tells us that there's a significant deviation
#from our model's predictions.
```

Exercise 5 – 2 points

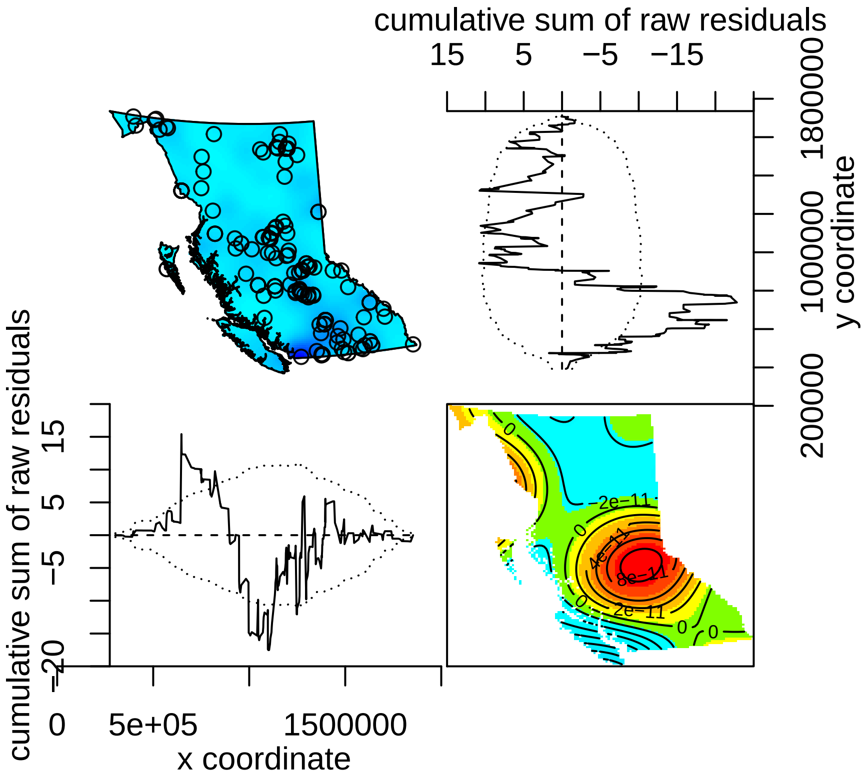
- Calculate and plot the model residuals. – 1 point(s)
- Based on the residuals, do you think the model performing well? – 1 point(s)

```r
# 1
#plot(residuals(fit2), cols = "transparent")
#plot(residuals(fit_smooth), cols = "transparent")
#residuals function doesnt run
#using diagnose.ppm instead:
diagnose.ppm(fit2)
```




```
## Model diagnostics (raw residuals)
## Diagnostics available:
##  four-panel plot
##  mark plot
##  smoothed residual field
##  x cumulative residuals
##  y cumulative residuals
##  sum of all residuals
## sum of raw residuals in entire window = -1.678e-05
## area of entire window = 9.483e+11
## quadrature area = 9.44e+11
## range of smoothed field =  [-1.556e-10, 1.211e-10]
```

```
diagnose.ppm(fit_smooth)
```



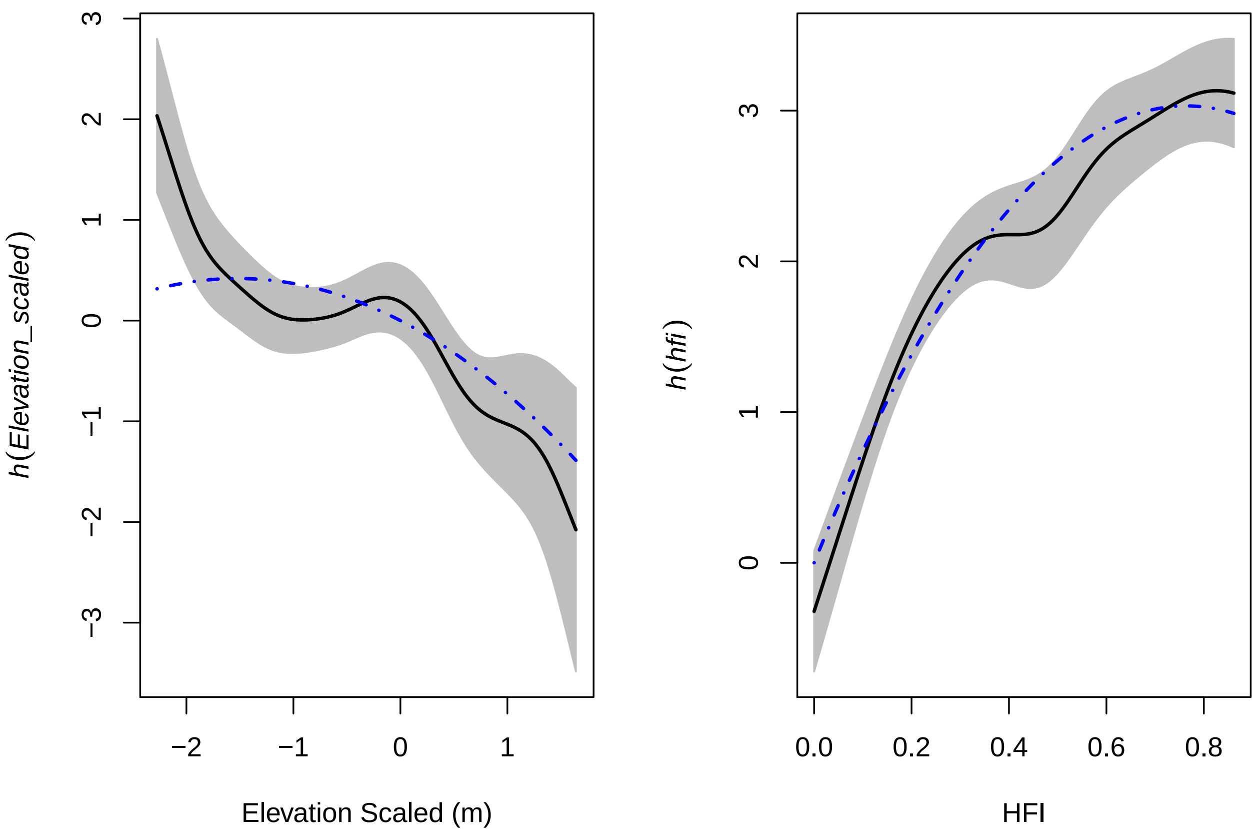

```
## Model diagnostics (raw residuals)
## Diagnostics available:
##  four-panel plot
##  mark plot
##  smoothed residual field
##  x cumulative residuals
##  y cumulative residuals
##  sum of all residuals
## sum of raw residuals in entire window = -6.889e-09
## area of entire window = 9.483e+11
## quadrature area = 9.44e+11
## range of smoothed field =  [-1.458e-10, 1.123e-10]
```

Exercise 6 – 3 points

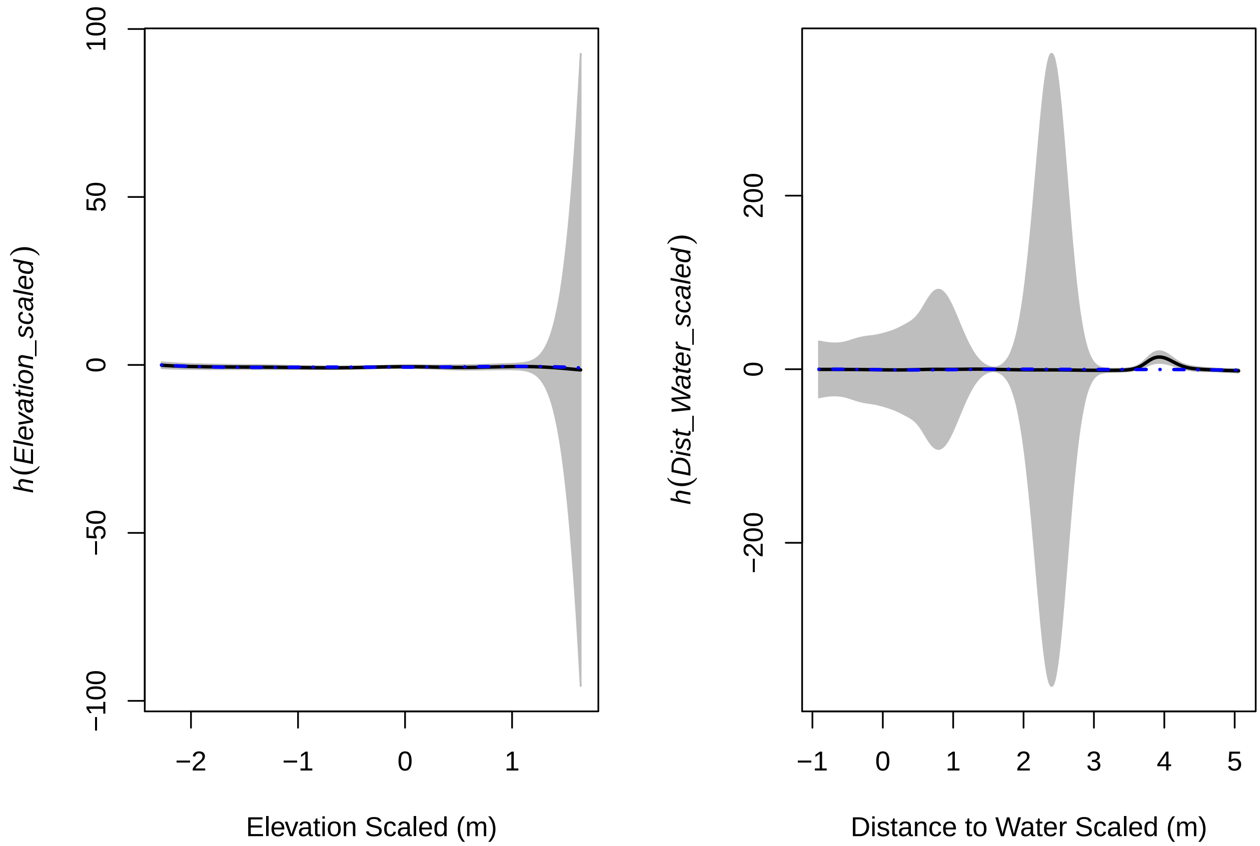
- Calculate the partial residuals as a function of both elevation and forest cover. – 1 point(s)
- Do you think that the terms are accurately capturing trends in the data? – 1 point(s)
- Do you have enough information to further refine the model and improve it's accuracy? – 1 point(s)

```
#PPM
par_res_elev_ppm = parres(fit2, "Elevation_scaled")
par_res_hfi_ppm = parres(fit2, 'hfi')
par(mfrow = c(1,2))
plot(par_res_elev_ppm,
     legend = FALSE,
     lwd = 2,
     main = "",
     xlab = "Elevation Scaled (m)")
plot(par_res_hfi_ppm,
     legend = FALSE,
```

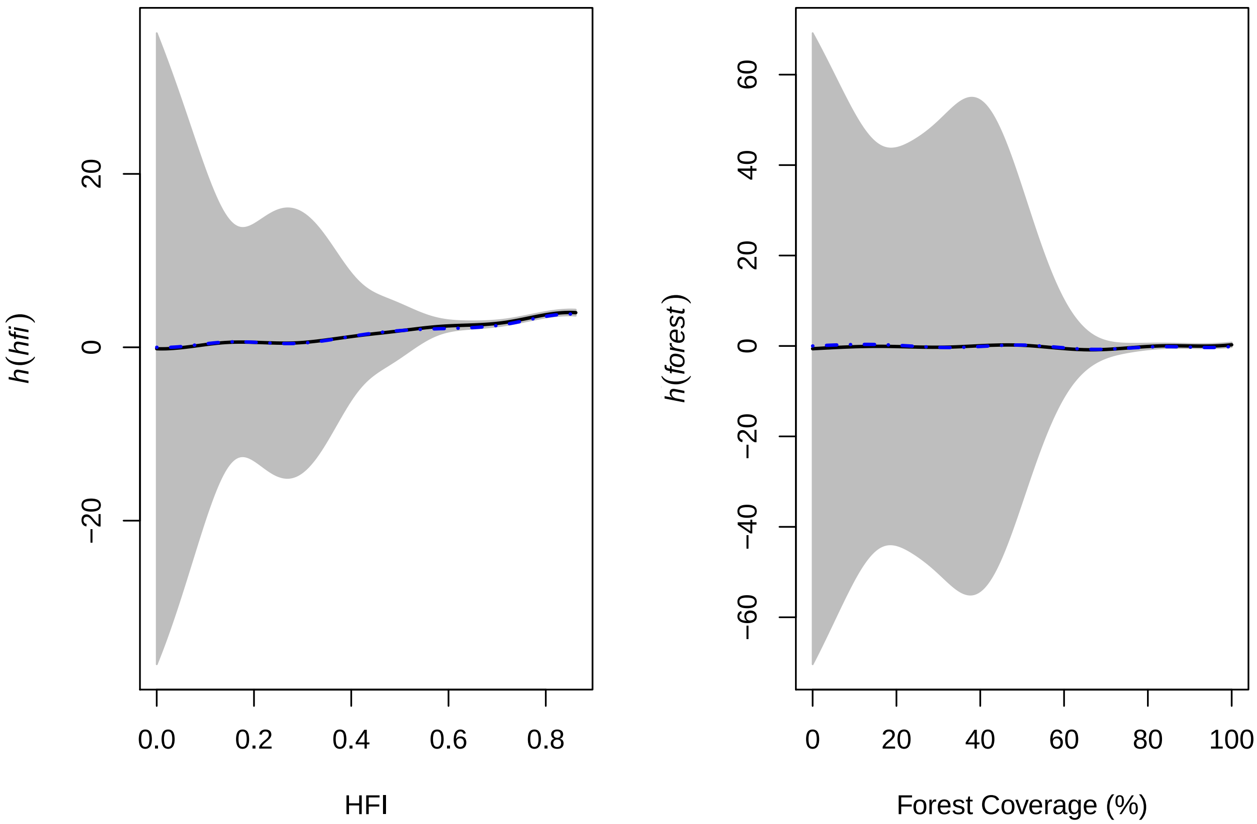
```
      lwd = 2,
      main = "",
      xlab = "HFI")
```




```
par(mfrow = c(1,1))

#GAM
library(splines)
par_res_elev = parres(fit_smooth, "Elevation_scaled")
par_res_distw = parres(fit_smooth, "Dist_Water_scaled")
par_res_hfi = parres(fit_smooth, 'hfi')
par_res_forest = parres(fit_smooth, 'forest')

#Side by side plotting
par(mfrow = c(1,2))
plot(par_res_elev,
      legend = FALSE,
      lwd = 2,
      main = "",
      xlab = "Elevation Scaled (m)")
plot(par_res_distw,
      legend = FALSE,
      lwd = 2,
      main = "",
      xlab = "Distance to Water Scaled (m)")
```




```
plot(par_res_hfi,
     legend = FALSE,
     lwd = 2,
     main = "",
     xlab = "HFI")
plot(par_res_forest,
     legend = FALSE,
     lwd = 2,
     main = "",
     xlab = "Forest Coverage (%)")
```

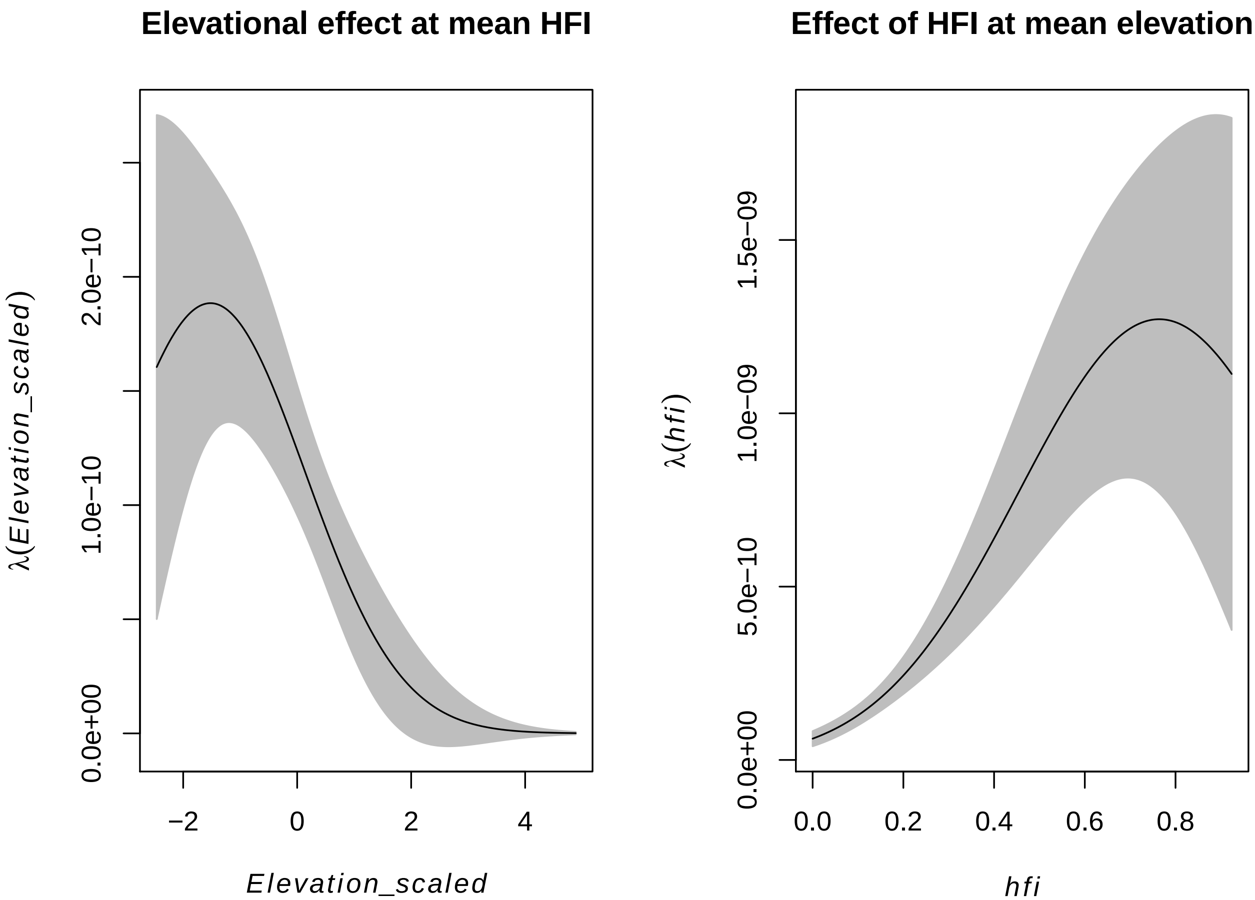



```
par(mfrow = c(1,1))
```

Model Validation

Intensity of a fitted point process model as a function of one of its covariates

```
## For PPM fit
#Elevational effect on lambda at mean hfi
elev_effect = effectfun(fit2, "Elevation_scaled", hfi = mean(hfi), se.fit = T)

#HFI effect on lambda at mean elevation
hfi_effect = effectfun(fit2, "hfi", Elevation_scaled = mean(Elevation_scaled), se.fit = T)

#Side by side plotting
par(mfrow = c(1,2))
#Plot the elevation effect
plot(elev_effect,
     legend = FALSE,
     main = "Elevational effect at mean HFI")

#Plot the slope effect
plot(hfi_effect,
     legend = FALSE,
     main = "Effect of HFI at mean elevation")
```




```
par(mfrow = c(1,1))
```

Relative Intensity

```
#Calculate the relative intensity as a function of elevation
rh_elev = rhohat(na.omit(fit2), na.omit(Elevation_scaled))

#Calculate the relative intensity as a function of hfi
rh_hfi = rhohat(na.omit(fit2), na.omit(hfi))

#Side by side plotting
par(mfrow = c(1,2))
plot(rh_elev,
     legend = FALSE,
     main = "",
     xlab = "Elevation scaled (m)")
plot(rh_hfi,
     legend = FALSE,
     main = "",
     xlab = "HFI")
par(mfrow = c(1,1))
```