

Análisis de presencias con procesos de puntos

Tutorial intermedio de spatstat

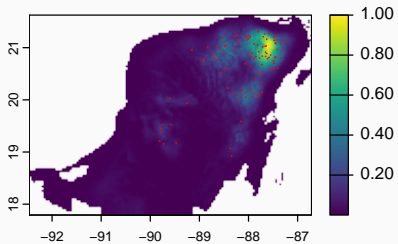
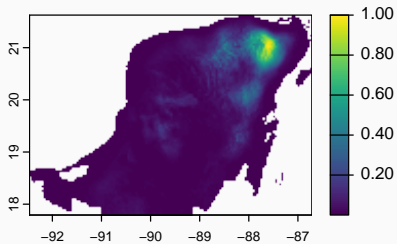
Gerardo Martín

2022-06-29

Simulación de presencias

- `aggregate` disminuye la resolución por el factor indicado
- `round` redondea los valores con el número de decimales
- Estos pasos no son enteramente necesarios en un análisis real, los hacemos para disminuir tiempo de cómputo

Código - viendo la favorabilidad



Formateo para spatstat

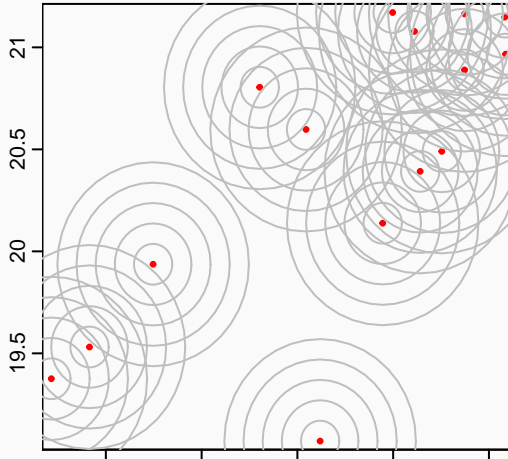
```
source("Funciones-spatstat/imFromStack.R")  
source("Funciones-spatstat/plotQuantIntens.R")  
source("Funciones-spatstat/findCompatibles.R")  
source("Funciones-spatstat/getPolyFormulas.R")  
source("Funciones-spatstat/ppmBatchFit.R")
```

```
r.im <- imFromStack(r)
names(r.im) <- names(r)
w <- as.owin(r.im[[1]])
puntos.ppp <- ppp(x = puntos$x,
                  y = puntos$y,
                  window = w,
                  check = F)
Q <- pixelquad(X = puntos.ppp, W = as.owin(w))
```

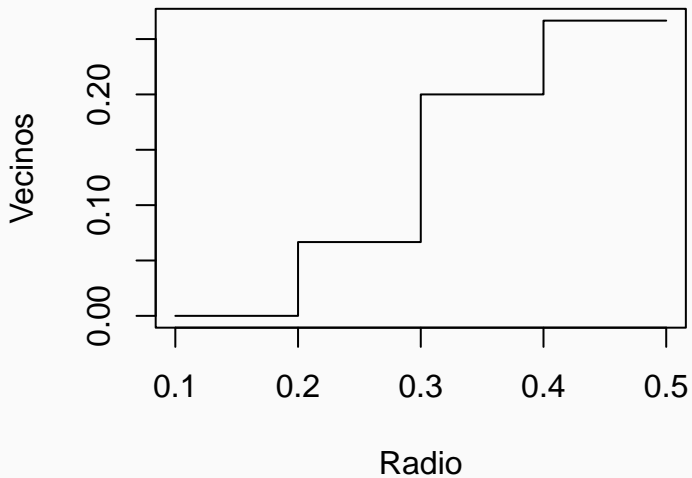
Análisis exploratorio

Función de K de Ripley

1. Número promedio de vecinos como función de la distancia a cada punto:

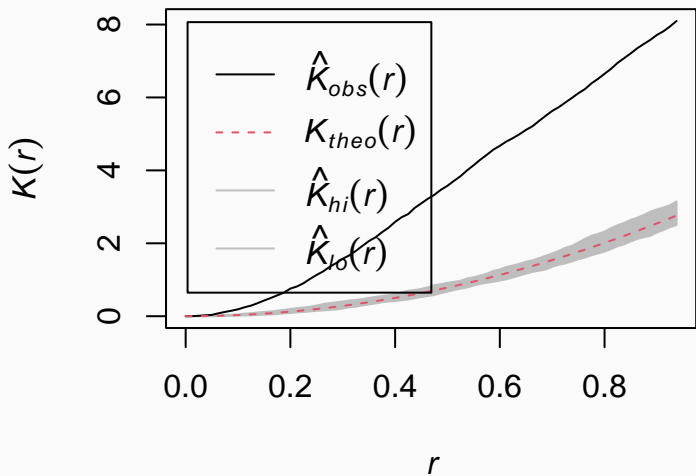


Radio	Vecinos
0.1	0.0000000
0.2	0.0666667
0.3	0.2000000
0.4	0.2666667
0.5	0.2666667



```
K <- envelope(puntos.ppp, fun = Kest, nsim = 39)
```

Para estimar significancia hace un muestreo aleatorios de punto, de ahí que haya que especificar el número de simulaciones (`nsim = 39`).

K

1. El proceso está levemente autocorrelacionado
 - Veremos si la correlación presente es explicada por factores ambientales
2. No sabemos de momento si afectará al modelo

Análisis 2.0

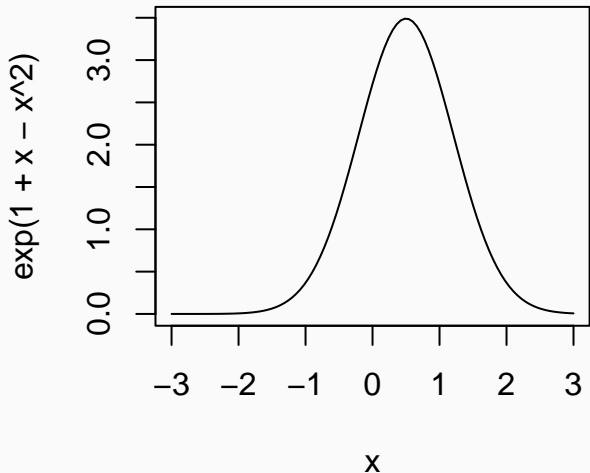
```
plotQuantIntens(imList = r.im,  
                noCuts = 5,  
                Quad = Q,  
                p.pp = puntos.ppp,  
                dir = "",  
                name = "Respuestas-centroide")
```

Ver archivo de gráficas

Consideraciones para proponer modelos

Curvas con forma de campana → fórmula cuadrática

```
curve(exp(1 + x - x^2), from = -3, 3)
```



Ecuación lineal:

$$y = \alpha + \beta_1 x_1 + \cdots + \beta_n x_n$$

Ecuación polinomial de 2^o grado

$$y = \alpha + \beta_1 x_1 + \beta'_1 x_1^2 + \cdots + \beta_n x_n + \beta'_n x_n^2$$

Recordemos que $y = \log \lambda$

¿Qué variables podemos incluir en el mismo modelo?

Regla de oro: Aquellas que no estén correlacionadas

- Que x_1 no sea predictor de x_2
- No se puede atribuir efecto de x_1 ó x_2 sobre λ
- Necesitamos medir correlación entre pares de variables (**pairs**)

Identificación automática de covariables *compatibles*

```
compatibles <- findCompatibles(r, thres = 0.5,  
                               max.comb = 3)
```

Variable_1	Variable_2	Variable_3
bio1	bio12	bio18
bio1	bio12	bio2
bio1	bio12	bio3
bio1	bio12	bio4
bio1	bio12	bio6
bio1	bio12	bio7

- Necesitamos generar una tabla de exponentes para variables, usando el resultado de `plotQuantIntens`.
- Razonamiento:
 - Identificar exponente máximo que tendrá el modelo para cada variable
 - La función `getPolyFormulas` generará las fórmulas para todas las combinaciones con exponentes $1 : n$
- Tabla debe tener dos columnas: **Variable**, **Power**

```
expon <- read.csv("Datos/Tabla-coefs.csv")
formulas <- getPolyFormulas(respDF = expon,
                           compatMat = compatibles)
formulas[1:5]

## [1] "~bio1 + bio12 + I(bio12^2) + bio18 + I(bio18^2) + I(bio18
## [2] "~bio1 + bio12 + I(bio12^2) + bio2 + I(bio2^2)"
## [3] "~bio1 + bio12 + I(bio12^2) + bio3 + I(bio3^2)"
## [4] "~bio1 + bio12 + I(bio12^2) + bio4 + I(bio4^2)"
## [5] "~bio1 + bio12 + I(bio12^2) + bio6 + I(bio6^2)"
```

- La función `ppmBatchFit` ajustará todos los modelos generados por `getPolyFormulas`
- Algunos modelos no lograrán estimar coeficientes satisfactoriamente
- La implementación presente solamente puede priorizar con base en AIC
- En un futuro, eliminará modelos que no converjan

```
modelos <- ppmBatchFit(points = puntos,  
                       covariates = r,  
                       formulas = formulas[1:10],  
                       parallel = F,  
                       topModels = 5)
```


Los argumentos

- `points`, tabla de coordenadas con dos columnas, `x` y `y`, en formato `data.frame`
- `covariates`, raster con bandas como covariables, nombres deben coincidir con fórmulas
- `formulas`
- `parallel`, si la rutina se ejecutará en serie ó paralelo, si `parallel = T`, especificar número de núcleos a usar con `cores = 3` (ajustar para cada máquina)
- `topModels`, cuántos de los “mejores” modelos queremos que nos guarde
- El resultado almacenado en `modelos` es una lista con los 5 mejores con base en el AIC

```
sapply(modelos, AIC)
```

```
## [1] -455.9962 -438.1068 -446.9706 -484.8199 -438.0861
```

Podemos usar los procedimientos habituales para los modelos de regresión en R

```
summary(modelos[[1]])
```

Como en los análisis de regresión, podemos ver el ajuste con los residuales, y siendo un modelo espacial, ver si hemos logrado explicar la correlación espacial con la prueba K de Ripley, tal como en el análisis exploratorio:

```
K.modelo <- envelope(modelos[[1]], fun = "Kest", nsim = 39)

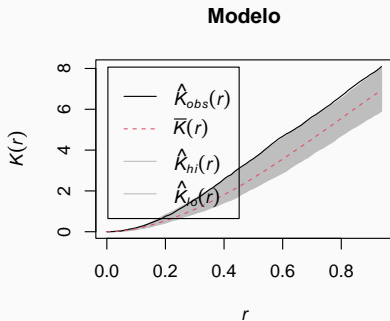
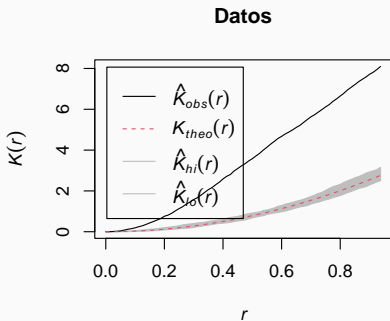
## Generating 39 simulated realisations of fitted Poisson model
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
## 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
## 39.
##
## Done.
```

Esta prueba genera 39 patrones de puntos utilizando el modelo base para calcular la función de Ripley y compara las simulaciones con la base de datos

```

par(mfrow = c(1, 2))
plot(K, main = "Datos")
plot(K.modelo, main = "Modelo")

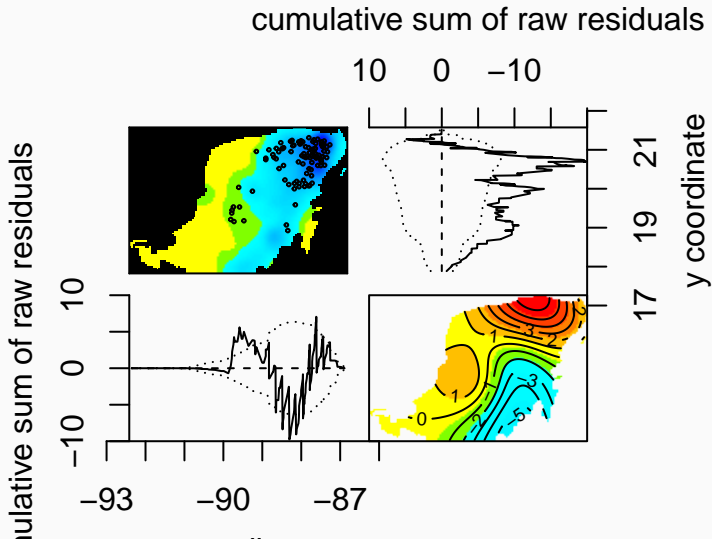
```



- Gráficas de horizonte
- Muestra 4 p neles:
 1. Patr n de puntos
 2. Residuales acumulados en cada fila de p xeles
 3. Residuales acumulados en cada columna de p xeles
 4. Residuales suavizados con contornos:

Kernel – Modelo

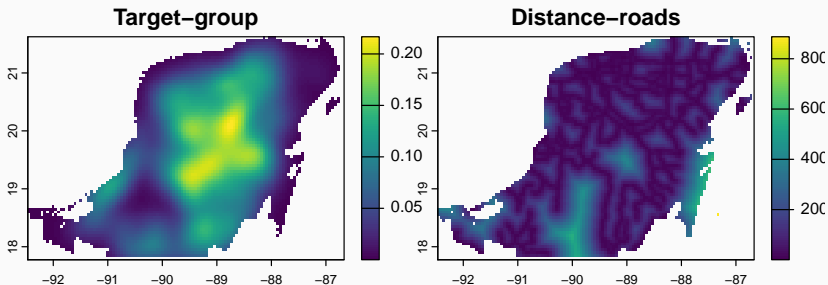
```
par(mar = c(1.5,1,0,0))  
diagnose.ppm(modelos[[1]], cex = 0.25, outer = 5)
```



Corrección de sesgo

Definición de escenario de sesgo

```
sesgo <- rast(c("Datos/Target-group.tif",  
               "Datos/Distance-roads.tif"))  
sesgo <- resample(sesgo, r)  
plot(sesgo)
```



- Usaremos la función `maskBias`

```
source("Funciones-spatstat/maskBias.R")
```

- Necesita los siguientes argumentos:

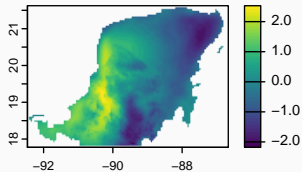
1. `s`, es el raster multi-banda de las covariables
2. `pres.areas`, es la base de datos de presencia con las coordenadas `x` y `y`
3. `bias.lay`, la capa que representa el esfuerzo de muestreo, donde los valores máximos correspondan a aquellos más muestreados.
4. `p.keep`, la proporción de valores a ser retenidos en cada corte de la capa
5. `power`, cuánto queremos que las muestras se concentren en los valores más altos de la capa de sesgo
6. `dis`, un factor de desagregación en caso que querer concentrar más valores en las zonas más muestreadas de lo que la resolución

```
r.mask1 <- maskBias(s = r[[1]], #Bio1
                    pres.areas = puntos,
                    bias.lay = sesgo[[1]], #Target group
                    p.keep = 0.1, power = 1)
```

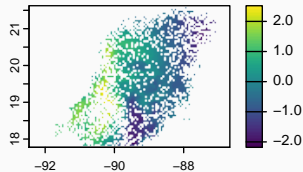
```
r.mask2 <- maskBias(s = r[[1]], #Bio1
                    pres.areas = puntos,
                    bias.lay = sesgo[[1]], #Target group
                    p.keep = 0.05, power = 3, dis = 4)
```

```
r.mask3 <- maskBias(s = r[[1]], #Bio1
                    pres.areas = puntos,
                    bias.lay = sesgo[[1]], #Target group
                    p.keep = 0.025, power = 4, dis = 4)
```

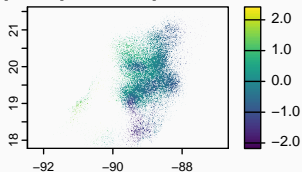
Normal



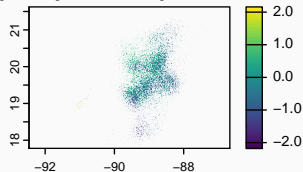
p.keep = 0.1, power = 1



p.keep = 0.05, power = 3

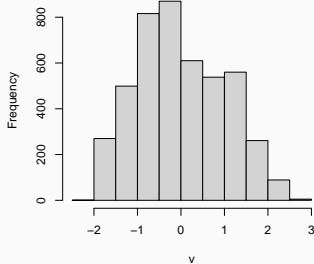


p.keep = 0.025, power = 4

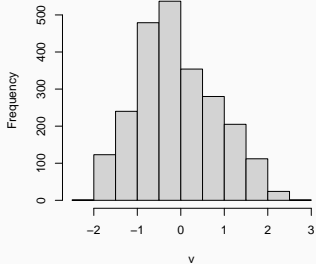


Histogramas

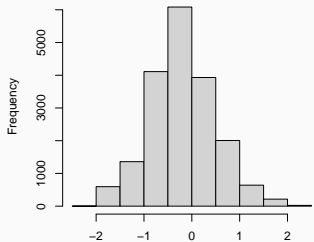
Normal



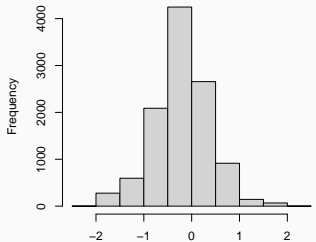
p.keep = 0.1, power = 1



p.keep = 0.05, power = 3



p.keep = 0.025, power = 4

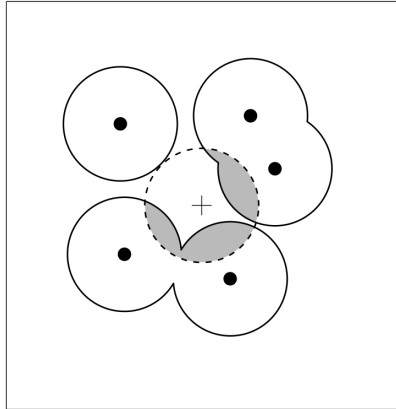


1. Correr análisis exploratorio con capas filtradas
2. Seleccionar exponentes
3. Generar fórmulas
4. Ajustar modelos
5. Evaluar bondad de ajuste
6. Proyectar a geografía completa (sin filtrado)
7. Validar

Modelando la correlación espacial

- Estiman efecto aleatorio para puntos cercanos
- Sirven para procesos de exclusión o agregación moderada
- Hay varios tipos de interacciones entre puntos

¿Qué es interacción?



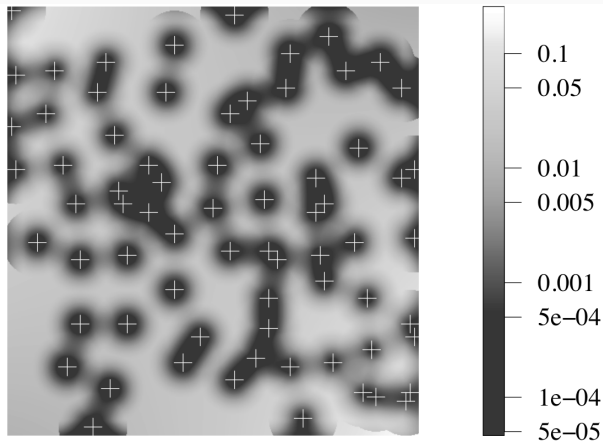


Figure 2: Crédito a Baddeley et al (2016)

FUNCTION	MODEL
AreaInter	area-interaction process
BadGey	multiscale Geyer saturation process
Concom	connected component interaction
Geyer	Geyer saturation process
Hybrid	hybrid of several interactions
Ord	Ord model, user-supplied potential
OrdThresh	Ord model, threshold potential
Saturated	saturated model, user-supplied potential
SatPiece	multiscale saturation process
Triplets	Geyer triplet interaction process

Para generar un modelo de interacción

1. Establecer tamaño del búfer

```
rr <- data.frame(r=seq(0.1,0.5,by=0.1))
p <- profilepl(rr, AreaInter,
               puntos.ppp ~ bio1 + bio12 + I(bio12^2) +
                           bio18 + I(bio18^2) +
                           I(bio18^3) + I(bio18^4),
               covariates = r.im, aic=F, rbord = 0.1)

## comparing 5 models...

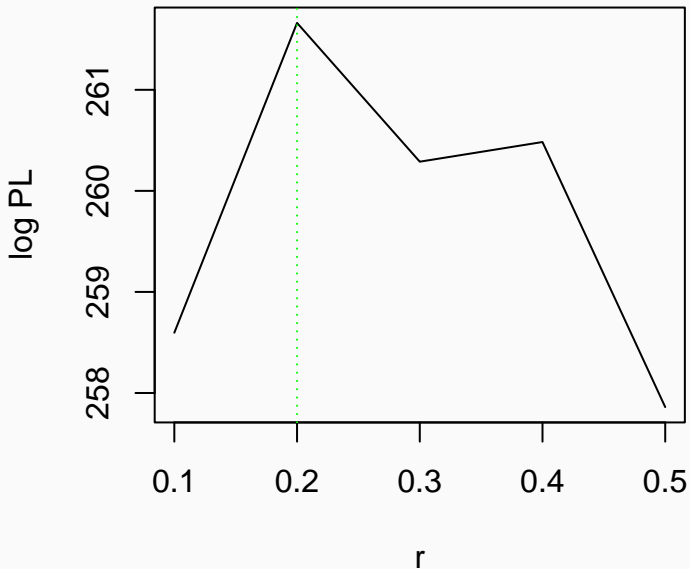
## 1, 2, 3, 4,
## 5.

## fitting optimal model...

## done.
```

Para generar un modelo de interacción

```
plot(p, main = "")
```



Para generar un modelo de interacción

Un radio de tamaño 2 minimiza la pseudo-verosimilitud, de modo que el modelo de interacción con la fórmula de m1 es:

```
m1.int <- ppm(Q = puntos.ppp,  
             trend = ~ bio1 + bio12 + I(bio12^2) +  
                   bio18 + I(bio18^2) +  
                   I(bio18^3) + I(bio18^4),  
             covariates = r.im,  
             AreaInter(rr$r[p$iopt]), rbord = 0.1) #Interacción
```

Efectos estimados

	Estimate	S.E.	CI95.lo	CI95.hi
(Intercept)	0.4137068	0.4490086	-0.4663340	1.2937475
bio1	-0.6205813	0.3081276	-1.2245003	-0.0166623
bio12	-0.1321785	0.4455172	-1.0053760	0.7410191
l(bio12 ²)	-1.9700678	0.6119923	-3.1695507	-0.7705849
bio18	-0.3436010	0.4462209	-1.2181779	0.5309760
l(bio18 ²)	0.0495367	0.3207293	-0.5790811	0.6781545
l(bio18 ³)	0.3702628	0.3617619	-0.3387774	1.0793031
l(bio18 ⁴)	-0.0912547	0.1434135	-0.3723400	0.1898306
Interaction	2.9641936	0.5721996	1.8427030	4.0856841

Modelo 1 sin interacción

```
knitr::kable(coef(modelos[[1]]))
```

	x
(Intercept)	1.5745520
bio1	-1.3199163
bio12	-0.9782225
I(bio12^2)	-2.5917197
bio18	0.8957644
I(bio18^2)	-0.3261731
I(bio18^3)	0.0066449
I(bio18^4)	0.0936610

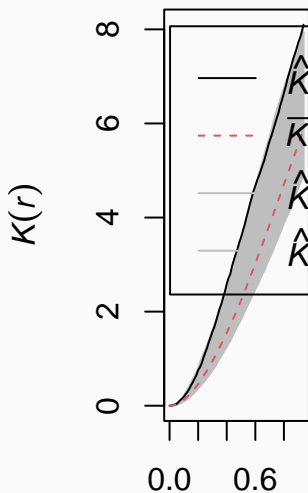
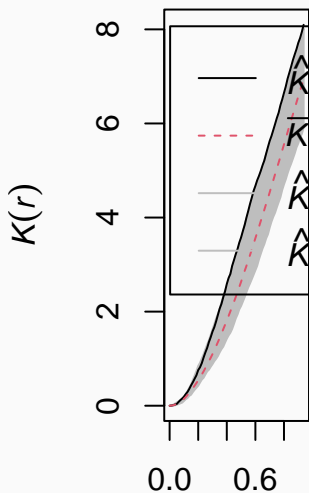
```
knitr::kable(coef(m1.int))
```

	x
(Intercept)	0.4137068
bio1	-0.6205813
bio12	-0.1321785
I(bio12^2)	-1.9700678
bio18	-0.3436010
I(bio18^2)	0.0495367
I(bio18^3)	0.3702628
I(bio18^4)	-0.0912547
Interaction	2.9641936

```
K.int <- envelope(m1.int, Kest, nsim = 39)

## Generating 39 simulated realisations of fitted Gibbs model .
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
## 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
## 39.
##
## Done.
```

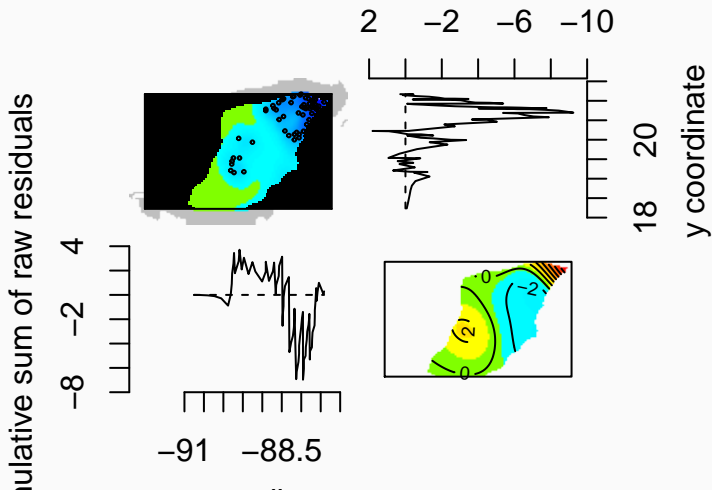
Modelo Poisson Modelo de interacci



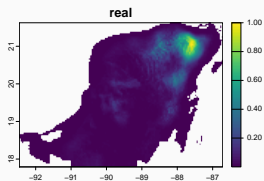
```
par(mar = c(1.5,1,0,0))
```

```
diagnose.ppm(m1.int, cex = 0.25, outer = 5)
```

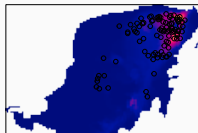
cumulative sum of raw residuals



Comparación con favorabilidad real



Poisson



Interacción

