conversao_migracao

Mori, Danilo Pereira

23 de abril de 2018

Objetivo

Comparar os dois métodos de conversão de parâmetro de migração entre um modelo neutro de campo médio (Etienne 2005) e um modelo neutro de espaço explícito (Rosindell et al. 2008) a partir de deduções da eq 1 apresentada por Chisholm & Lichstein (2009). O primeiro método é desenvolvido pelos autores anteriores e trata-se de uma aproximação (eq2 Chisholm & Lichstein 2009); o segundo método foi deduzido a partir da eq 1 e aproveitou das particularidades de nossas simulações para deduzir uma equação. Ambos métodos relacionam a probabilidade de um indivíduo de fora da comunidade colonizar uma unidade de habitat na comunidade local por evento de morte ('m') com a distância média de dispersão dos indivíduos na paisagem ('d').

O modelo neutro espacialmente explícito tem as seguintes características: i) utilizamos apenas áreas amostradas contíguas que aproximamos como quadrados; ii) a distribuição de probabilidade subjacente a função de dispersão é Laplace e parametrizamos a partir do desvio-padrão; iii) a área foi escrita em função de J e DA, número de indivíduos e densidade(ind/ha) observada na amostra, respectivamente.

Contexto

Chisholm & Lichstein (2009) (doravante C&L09) estabeleceram uma relação entre m e A (a área da comunidade local):

> When an individual at location (x, y) in the local community dies, the replacement individual may, by virtue of the random dispersal and recruitment processes, be from within the local community (i.e., within the quadrat) or from outside the local community (i.e., from outside the quadrat). Define m x,y as the probability that the replacement individual at location (x, y) is drawn from outside the local community. This parameter will be highest for individuals on the edges of the quadrat and smallest for individuals at the centre of the quadrat, where m x,y » 0 for large A. We define m as the average value of m x,y across the whole of the local community as follows:

$$eq.0 : m = \frac{1}{A} \int \int_A m_{x,y} dx dy$$

Essa equação é válida para o processo de dispersão em ambientes homogêneos, ou seja, sem fragmentação. A maneira que simulamos a dispersão em paisagens fragmentadas é diferente em uma simulação coalescente: uma vez que sorteamos um progenitor e este estaria presente em uma unidade de não habitat, o sorteio é refeito até que o progenitor esteja em uma unidade de habitat. Uma vez que o sorteio é refeito este pode cair novamente dentro da área da comunidade local, uma equação que descreve exatamente a probabilidade de um indivíduo da comunidade ser substituido por um indivíduo de fora da comunidade em uma paisagem fragmentada dependeria de explicitamente considerar a configuração espacial. Uma aproximação do efeito da fragmentação na simulação é considerar que a chance da dispersão ser oriunda de uma área de cobertura vegetal, assim, podemos corrigir m pela porcentagem de cobertura vegetal na paisagem:

$$eq.0' : m' = \frac{mp}{1 - (1 - p)m}$$

Onde p é a porcentagem de cobertura vegetal na paisagem. É necessário corrigir o valor de m quando partimos de parâmetros da simulação coalescente em paisagens fragmentadas, contudo não é correto corrigir 'm' obtidos pelo ajuste do modelo neutro de campo médio.

Método C&L09

A aproximação deduzida por C&L09 é $m = \frac{Pd}{\pi A}$, onde P e A são o perímetro e área do plot, respectivamente. Considerando o plot quadrado e a distribuição de Laplace podemos reescrever essa aproximação como:

$$eq.1a : m = sd\frac{4}{\pi L}$$
$$eq.1b : sd = m\frac{\pi L}{4}$$

Onde $P = 4L$, $A = L^2$ e $L = 100\sqrt{J/DA}$ metros

Método Coutinho apud C&L09

Coutinho parte da eq.0 e aproveitando as características da simulação coalescente que utilizamos: a) utiliza apenas plot quadrados; b) a dispersão não é radial, ao invés, é descrita como o resultado do sorteio independente em eixos ortogonais. Assim, podemos escrever a eq.0 como:

$$eq.0 - C.a : m = \left(\frac{1}{L} \int_{-L/2}^{L/2} m_x(x)\mathrm{d}x \right)^2$$
$$eq.0 - C.b : m_x = 1 - \int_{-L/2}^{L/2} K(x-y)\mathrm{d}y$$

K é a função de dispersão. Podemos reescrevemos considerando a distribuição de probabilidade de Laplace como:

$$eq.2a : m = sd\frac{1 - e^{-\frac{\sqrt{2}L}{sd}}}{\sqrt{2}L}$$
$$eq.2b : sd = \frac{\sqrt{2}L}{mW_0(-\frac{e^{-1/m}}{m}) + 1}$$

Para escrever a equação em função do desvio-padrão (eq 2b) utilizamos o ramo principal da função W de Lambert (W_0).

Comparação dos métodos

Para comparar os métodos vou utilizar os parâmetros ajustados (modelo de campor médio, por verossimilhança) à SADs amostradas na Mata Atlântica e aqueles estimados por um modelo neutro espacialmente explícito. Todos os vetores de abundância foram observados em amostras com pelo menos 1 ha. As SADs observadas foram ajustado ao modelo de campo médio e obtemos m; no modelo de espaço explícito informamos *a priori* qual o desvio padrão da função de dispersão. Então vamos utilizar as equações 1a e 2a para calcular m a partir dos desvio-padrões informados *a prior*, considerando o m' corrigido. As equações 1b e 2b irão converter

m em desvio padrão. Para isso vou criar uma função que contêm as equações e a definição de θ de Hubbell (2001):

```
f_conv.par <- function(modelo, par., par.aux){
  #parametros que seram convertidos
  theta<- par.[[1]]
  m <- par.[[2]]
  sd_k <- par.[[3]]
  U <- par.[[4]]
  #parametros auxiliares a conversao
  p <- par.aux[[1]]
  J <- par.aux[[2]]
  DA <- par.aux[[3]]
  L <- 100*sqrt(J/DA)
 #conversoes
  if(modelo == "campo_medio"){ # EI -> EE
    U_ <- theta / (2 * 500 * p * DA) # Hubbell 2001
    sd_k.CL <- m * L * pi * sqrt(2) / 4 # eq 1b
    sd_K.CaCL <- sqrt(2) * L / (m * lambertW0(-exp(-1/m)/m) + 1) # eq 2b
    df_ <- data.frame(par.value = c(U_, sd_k.CL, sd_K.CaCL),
                      par.class = c("U","sd_k","sd_k"),
                      par.method = c("NA","CL","CaCL"))
    return(df_)
  }else{ # EE -> EI
    theta_ <- 2 * 500 * p * DA * U # Hubbell 2001
    m_CL <- 4 * sd_k / ( sqrt(2) * L * pi ) # eq 1a
    m_CaCL <- sd_k * ( 1 -exp( -sqrt(2) * L / sd_k) ) / (sqrt(2) * L / sd_k) # eq 2a
    df_ <- data.frame(par.value = c(theta_, m_CL, m_CaCL),
                      par.class = c("theta","m","m"),
                      par.method = c("NA","CL","CaCL"))
    return(df_)
  }
}
df_par.conv <- ddply(df_resultados,c("SiteCode","kernel_percentil"),
                  function(X) f_conv.par(modelo = X[,"kernel_percentil"], #cada modelo terá um conjunto
                                         par. = as.list(X[,c("theta","m","sd_k","U")]), #input dados
                                         par.aux = as.list(X[,c("p","J","DA")]) #idem
                                         )
                    )
# df_par.conv$kernel_percentil <- factor(df_resultados$kernel_percentil, levels = levels(df_resultados$
# df_par.conv$kernel_percentil %>% levels()
# df_par.conv %>% str
```

EI -> EE

Vamos avaliar a diferença entre os métodos para as conversões de 'm' para o desvio padrão da função de dispersão EI -> EE:

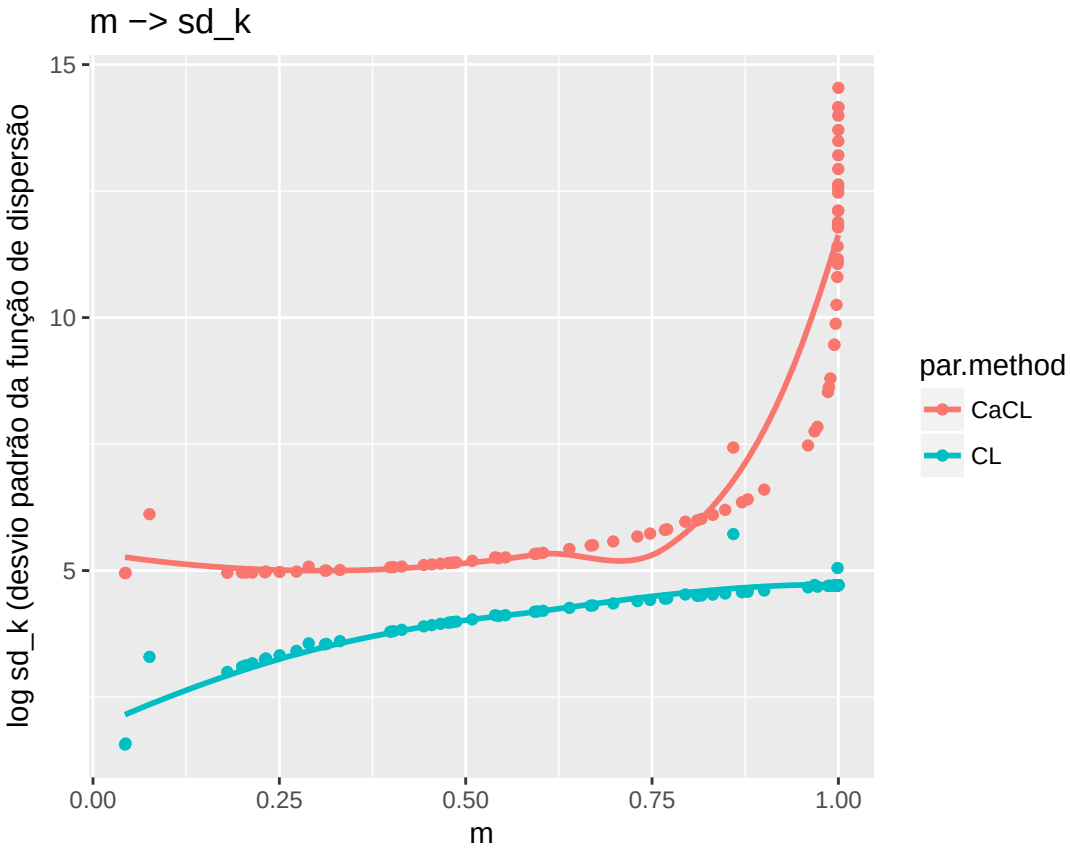

figura 1. Log(sd_k) ~ m * metodo de conversao. CaCL = método Coutinho apud C&L09; CL = equação 2 C&L09. A linha é uma aproximação 'loess' da função geom_smooth (ggplot2).

Os dois métodos tem perfis bem distintos (mesmo fora da escala log), o método CaCL estima vaior variação para valores m -> 1. O terceiro quartil de m = 0.998.

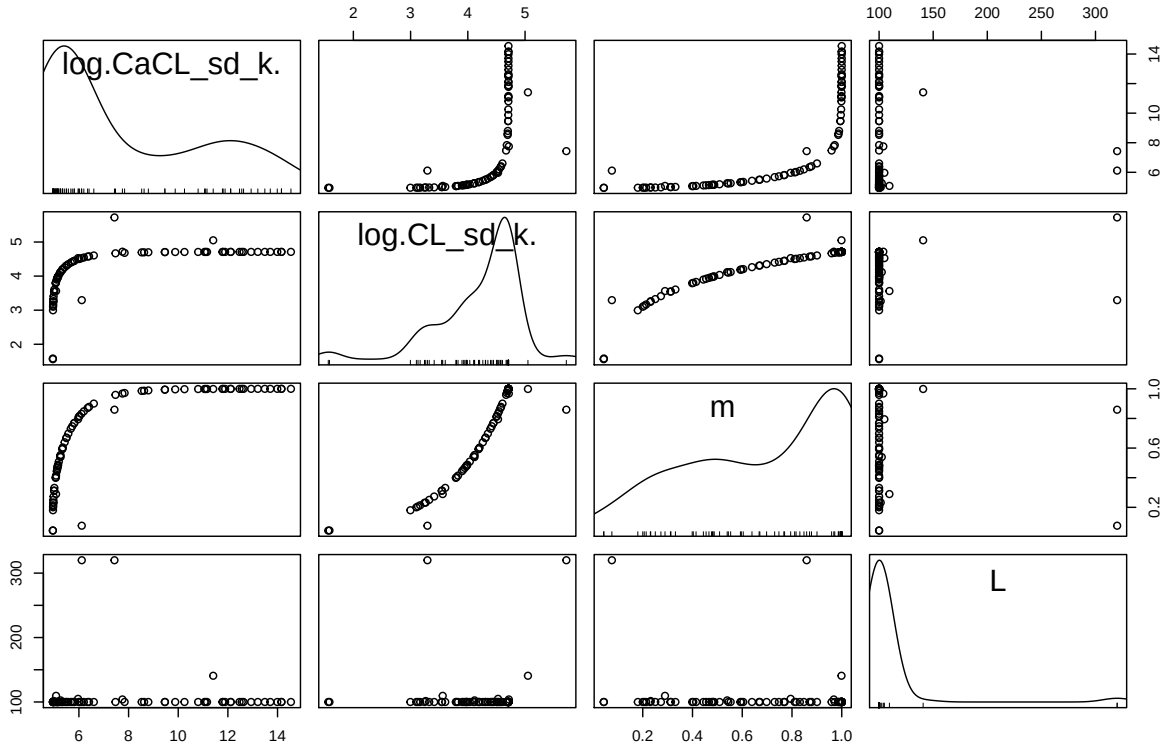

Figura 2. log(sd_k) * metodo, m e L = lado da amostra

Há clara distinção entre os métodos, os parâmetros convertidos estão plotados na escala log. Vou converter para porcentagem da chuva de propágulos que se mantêm até l_{cel} metros da planta progenitora (%k)

```
# preparando os dados
df_sd_k <- df_par.conv %>% filter(kernel_percentil == "campo_medio" & par.class !=
    "U") %>% inner_join(x = ., y = df_resultados[, c("SiteCode", "kernel_percentil",
    "m", "J", "DA")], by = c("SiteCode", "kernel_percentil"))

# funcao para calcular o percentil
f_percentil.kernel <- function(i, df_ = df_sd_k) {
    percentilkernel <- function(sigma, density, npoints = 1e+05) {
        # metríca da simulacao e distancia de referencia
        d_ind_MA <- 100/sqrt(density)
        # relacao entre sd e o parametro escalar da distribuicao Laplace
        b_laplace <- sigma/sqrt(2)
        # sorteios de duas distribuicoes identicas Laplace em eixos ortogonais
        X_laplace <- d_ind_MA * round(rlaplace(npoints, s = b_laplace)/d_ind_MA)
        Y_laplace <- d_ind_MA * round(rlaplace(npoints, s = b_laplace)/d_ind_MA)
        # calculando a distancia dos pontos ate a origem
        dist_laplace <- sqrt(X_laplace^2 + Y_laplace^2)
        # Percemtil
        percentil <- length(dist_laplace[dist_laplace <= d_ind_MA])/length(dist_laplace)
        return(percentil)
    }
    kernel_percentil <- percentilkernel(sigma = df_[i, "par.value"], density = df_[i,
        "DA"])
```

```
    return(kernel_percentil)
}
# armazenando
df_sd_k$k_perc <- sapply(1:nrow(df_sd_k), f_percentil.kernel)

# contra-prova
f_q <- function(i, df = df_sd_k) {
    kernel.quantile <- function(percentil, sigma, density, npoints = 1e+05) {
        # metríca da simulacao e distancia de referencia
        d_ind_MA <- 100/sqrt(density)
        # relacao entre sd e o parametro escalar da distribuicao Laplace
        b_laplace <- sigma/sqrt(2)
        # sorteios de duas distribuicoes identicas Laplace em eixos ortogonais
        X_laplace <- d_ind_MA * round(rlaplace(npoints, s = b_laplace)/d_ind_MA)
        Y_laplace <- d_ind_MA * round(rlaplace(npoints, s = b_laplace)/d_ind_MA)
        # calculando a distancia dos pontos ate a origem
        dist_laplace <- sqrt(X_laplace^2 + Y_laplace^2)
        # Percemtil
        k_quantil <- quantile(dist_laplace, probs = percentil)
        return(k_quantil)
    }
    kernel_quantil <- kernel.quantile(percentil = df[i, "k_perc"], sigma = df[i,
        "par.value"], density = df[i, "DA"])
    return(kernel_quantil)
}

df_sd_k$q_perc <- sapply(1:nrow(df_sd_k), f_q)
# plot(par.value ~ q_perc, data=df_sd_k,main = 'Contra-prova do percentil:
# sd_k X quantile(percentil)')
df_sd_k %<>% mutate(erro = par.value - q_perc)
```

Eu gostaria de tirar a contra prova do valor de percentil estimado. Para isso eu tentei utilizar a função quantile com o percentil estimado pela função anterior, mas parece que eles são muito diferentes. Não sei se cometi algum erro. Estava pensando se não era possível deduzir uma formula, agora que o Coutinho fez as eq.0-C, eu acho que talvez ficasse mais fácil, eu tava pensando em mexer nisso.

Olhando para as estimativas do percentil me parece que faz sentido com os valores estimados em par.value, mesmo que exista um erro consistente na estimativa do par.value (fig. Pensando que em um modelo de campo médio o espaço é desconsiderado da dinâmica é de se esperar que a chuva de própagulos.

```
car::scatterplotMatrix(~k_perc + m + log(erro) + log(par.value), reg.line = F,
    smoother = F, df_sd_k)
```

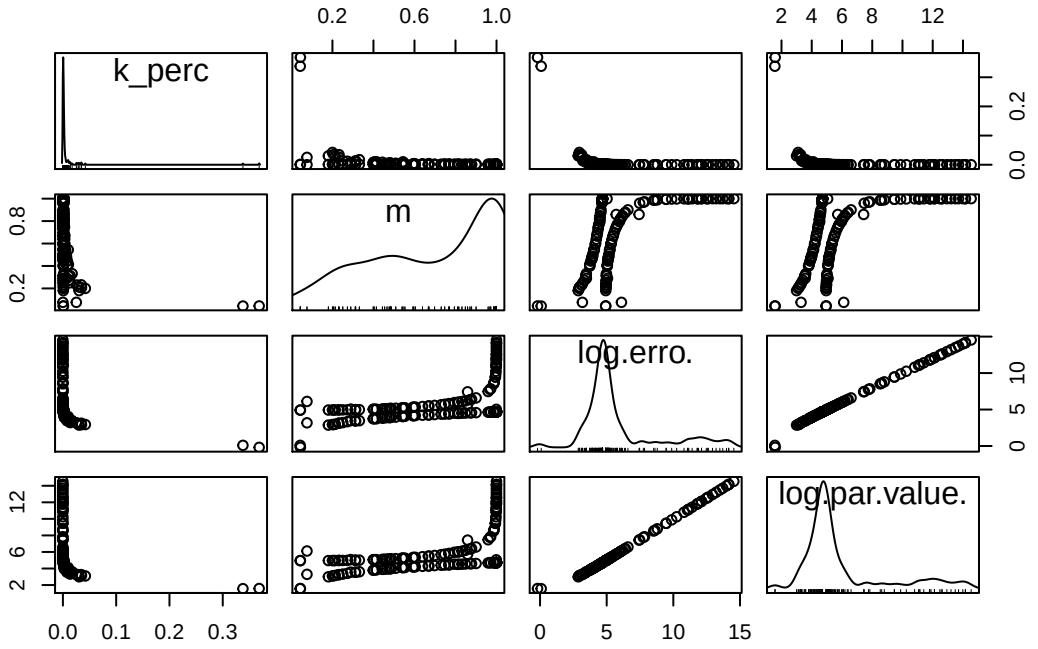

figura3 .k_perc = porcentagem estimada a partir do sd_k calculado, m = parêmetro modelo neutro, log.erro = log(diff (par.value[sd_k calculado] - quantile(.,k_perc,l_cel))) log.par.value = log (estimativa de sd_k calculado a partir de m)

Os valores de par.value (st_k estimado) apresentam dois perfis distintos quando comparado com m, mas um esmo perfil quando compardo com log.erro. Isso quer dizer que o erro entre a estimativa par.value e o quantil do percentil estimado a partir de par.value tem um erro que não é no método de conversão de parâmetros. Como discutido nas funções estabelecidas

EE -> EI

Agora vamos avaliar a conversão de parâmetros sd_k da simulação coalescente para o respectivo m do modelo de campo médio. Vou olhar os dados antes de corrigir o m (eq.0'). Aqui também é possível comparar os valores obtidos pelas conversões com aqueles estimados nas SAD_{EE}. Eu tenho esses dados, eles estão no git, eu deixei meu computador rodando isso enquanto eu trabalhava no TreeCo.

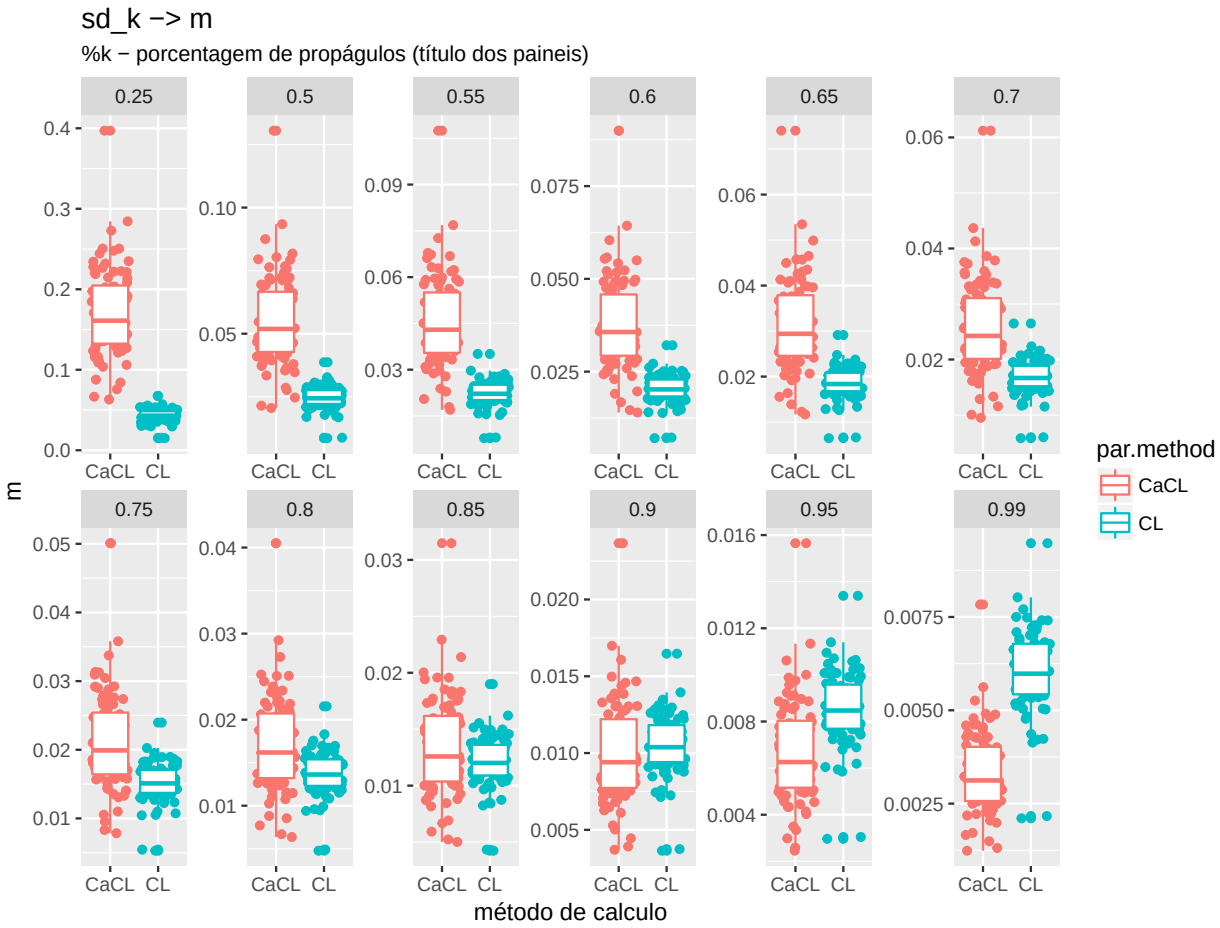

figura 4. Relação o m calulado pelas equações 1a e 2a e a porcentagem de propágulos correspondete ao desvio padrão da função de dispersão.

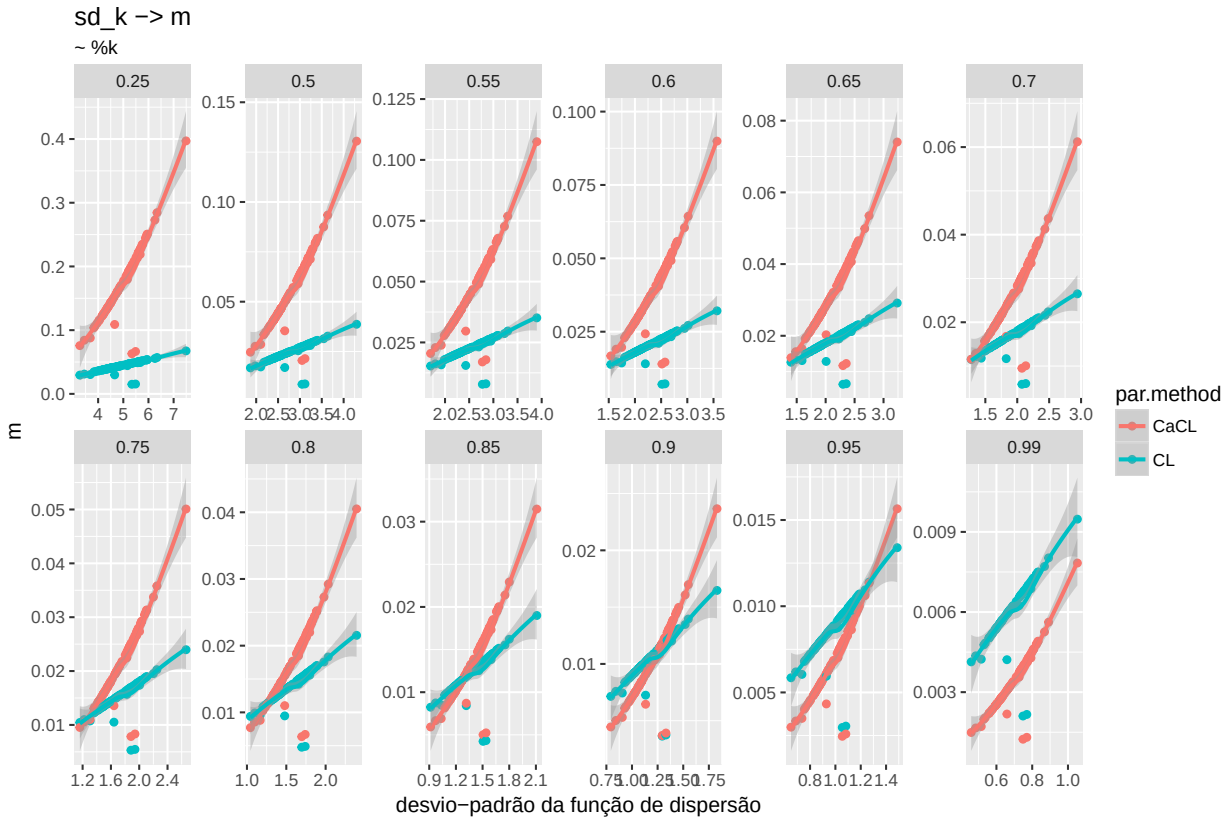

figura 5. m ~ sd_k + %k + método de calculo

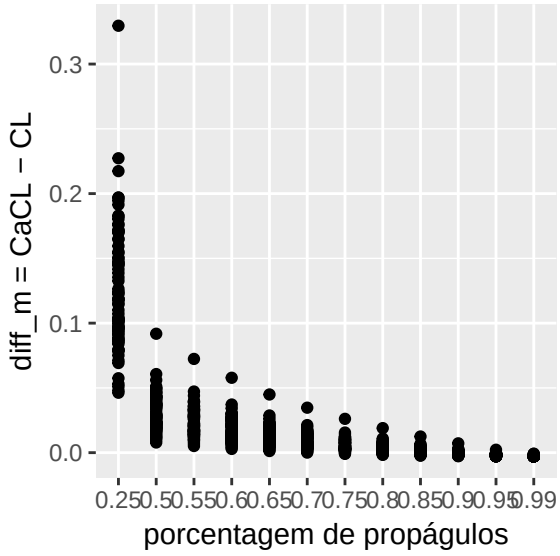

figura 6. difença no valor calculado pelos dois métodos e a correspondente porcentagem de propágulos.

A escala muda entre os paineis: em 0.25 varia de 0.015 até 0.397; no último painel varia de 0.001 até 0.009; a variação dos pontos diminui com o aumento da limitação à dispersão.

EE <-> EI

-> theta ~ m

ANEXO: formulas que estimam o sd_k necessário para que uma determinada porcentagem dos propágulos permaneça até l_{cel}

```
library(rmutil)

#' Kernel quantile
#'
#' Returns the quantile value that acumulates a given percentile of
#' the density of three types of dispersion kernels in a simulation of
#' a spatially explicit neutral dynamics (Rosindell et al. 2008).
#'
#' @param sigma real positive, the dispersion parameter of the kernel. For
#'     uniform it is the total width of the kernel (max - min), and for
#'     gaussian and laplacian kernels sigma is the standard deviation.
#' @param kernel character, either 'normal', 'uniform' or 'laplacian';
#'     the kernel type. See Rosindell et al. (2008) for details.
#' @param p real 0 < p > 1; the accumulated probability from the kernel
#'     center.
#' @param density real positive, the density of individual in the
#'     simulation grid, in individuals/hectare
#' @return kernel quantile, which is the distance in meters from the kernel
#'     center that encompasses a proportion p of total kernel
#'     density. For example, p = 0.5 returns the median dispersal
#'     distance.
#' @param npoints non-negative integer, number of distance points to
#'     simulate.
#' @details This function simulates the sampling of dispersion
#'     distances from the kernel as outlined by Rosindell et al
#'     (2008), which is not the same as a bivariate version of each
#'     distribution (e.g. a bivariate normal, laplacian,
#'     uniform). Rather, X and Y coordinates are sampled independently
#'     and the used to find the dispersal distance. Though an
#'     analytical solution might be feasible, this function uses the
#'     random sample functions of the univariate distributions and
#'     empirical quantile functions and so provides approximate values.
#' @references Rosindell J, Wong Y, Etienne RS, 2008, A coalescence
#' approach to spatial neutral ecology, Ecological Informatics, Vol: 3,
#' Pages: 259-271
qkernel <- function(sigma, kernel, p, density = 20852/50, npoints = 1e+05) {
    kernel <- match.arg(kernel, choices = c("normal", "gaussian", "laplace",
        "uniform"))
    ## metro na escala da simulacao
    d_ind_MA <- 100/sqrt(density)  #100/sqrt(DA)
    if (kernel == "laplace") {
        b_laplace <- sigma/sqrt(2)  ## relação entre sigma e b, a variância da laplaciana
        ## gerando valores de uma distribuição laplaciana e convertendo para a
        ## distância em metros
        X_laplace <- d_ind_MA * round(rlaplace(npoints, s = b_laplace)/d_ind_MA)
        Y_laplace <- d_ind_MA * round(rlaplace(npoints, s = b_laplace)/d_ind_MA)  # idem para Y
        dist_laplace <- sqrt(X_laplace^2 + Y_laplace^2)  #distâncias dos pontos até a origem
        result <- quantile(dist_laplace, p)  #guardando
```

```
    }
    if (kernel == "normal" | kernel == "gaussian") {
        b_norm <- sigma
        X_norm <- d_ind_MA * round(rnorm(npoints, sd = b_norm)/d_ind_MA)
        Y_norm <- d_ind_MA * round(rnorm(npoints, sd = b_norm)/d_ind_MA)
        dist_norm <- sqrt(X_norm^2 + Y_norm^2)
        result <- quantile(dist_norm, p)
    }
    if (kernel == "uniform") {
        b_unif <- sigma/2
        X_unif <- d_ind_MA * round(runif(npoints, min = -b_unif, max = b_unif)/d_ind_MA)
        Y_unif <- d_ind_MA * round(runif(npoints, min = -b_unif, max = b_unif)/d_ind_MA)
        dist_unif <- sqrt(X_unif^2 + Y_unif^2)
        result <- quantile(dist_unif, p)
    }
    return(unname(result))
}

#' Finds kernel dispersal parameter
#'
#' Returns dispersal parameter of three types of kernel that
#' accumulates a proportion of the kernel density up to a given distance.
#'
#' @param kernel character, either 'normal', 'uniform' or 'laplacian';
#'     the kernel type. See Rosindell et al. (2008) for details.
#' @param p real 0 < p > 1; the accumulated probability from the kernel
#'     center.
#' @param distance real positive, the quantile that should accumulate
#'     p (for instance, if p=0.5 distance is the median distance)
#' @param density real positive, the density of individual in the
#'     simulation grid, in individuals/hectare
#' @param npoints non-negative integer, number of distance points to
#'     simulate.
#' @param sigma.min real positive, the minimum value of the dispersal
#'     value to try in the one-dimensional optmisation.
#' @param sigma.max real positive, the maximum value of the dispersal
#'     value to try in the one-dimensional optmisation.
#' @return the output of function uniroot, which is a list. The
#'     element 'root' of the list is the dispersal parameter necessary for the kernel having
#'     dist as the quantile for p. For example, p = 0.5  and dist =10
#'     returns the dispersal parameter such that the median dispersal
#'     distance is 10 meters. The solution is not exact because of
#'     rounding values to cell sizes of the simulation grid.
sigkernel <- function(kernel, p, distance, density = 20852/50, npoints = 1e+05,
    sigma.min = 1, sigma.max = 100) {
    f1 <- function(x) distance - qkernel(x, kernel, p, density, npoints)
    uniroot(f1, lower = sigma.min, upper = sigma.max)
}
```