Estatística Espacial (MI418) / Geoestatística (ME907)

Guilherme Ludwig

2019-01-22

Anisotropia

Variograma direcional

Voltando ao exemplo do Wolfcamp Aquifer

Multidimensional scaling

Anisotropia

Em um processo isotrópico, assume-se que a dependência em um processo observado em dois pontos no espaço é função apenas da sua distância. Em outras palavras,

$$\text{Cov}(X(\mathbf{s}_1), X(\mathbf{s}_2)) = C(\|\mathbf{s}_2 - \mathbf{s}_1\|).$$

Na prática isto é razoável para boa parte de fenômenos naturais, mas ainda há casos em que a dependência pode mostrar uma região de preferência. Cressie (1993) argumenta que em mineralogia, cortes verticais da terra podem exibir o efeito da gravidade na acumulação de minerais; modelos climáticos podem ser afetados pelo gradiente do vento, etc.

Anisotropia geométrica

Um modelo bastante simples, mas que oferece flexibilidade comparado com o modelo isotrópico, é o chamado modelo de **anisotropia geométrica**. Isto é, para uma matriz $\mathbf{A}_{d\times d}$ o modelo de anisotropia é

$$\text{Cov}(X(\mathbf{s}_1), X(\mathbf{s}_2)) = C(\|(\mathbf{s}_2 - \mathbf{s}_1)\mathbf{A}\|) = C(\|\mathbf{h}\mathbf{A}\|).$$

Vou usar a parametrização de Diggle and Ribeiro (2007), Capítulo 3.7:

$$\mathbf{A} = \begin{pmatrix} \cos(\psi_A) & -\sin(\psi_A) \\ \sin(\psi_A) & \cos(\psi_A) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \psi_R^{-1} \end{pmatrix}$$

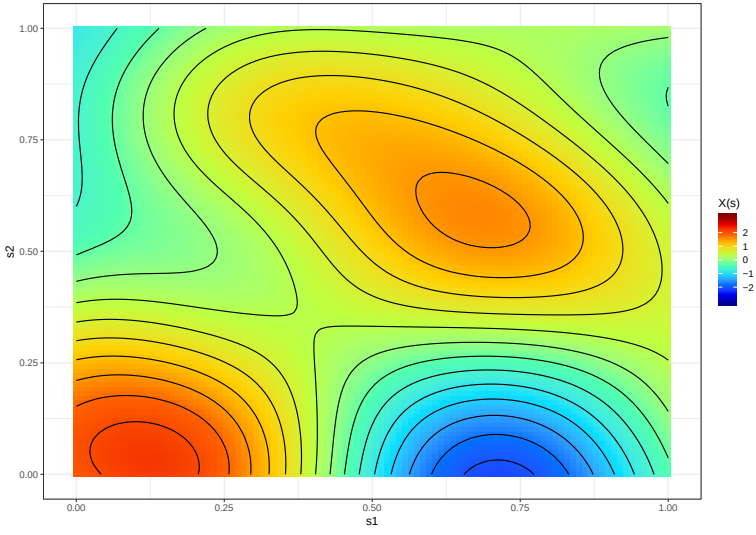
com $\psi_R > 1$. Note que Diggle and Ribeiro (2007) multiplicam $\mathbf{A}$ pela direita, mas é possível parametrizar $\mathbf{Ah}$ também.

Usando a função geoR::grf

```
require(geoR)
res <- 100 # Resolution of the grid
s1 <- seq(0, 1, length.out = res)
s2 <- seq(0, 1, length.out = res)
Locations <- expand.grid(s1 = s1, s2 = s2)
set.seed(1)
x <- grf(grid = Locations, cov.pars = c(1, .4),
         cov.model = "gaussian")
```

```
## grf: simulation on a set of locations provided by the user
## grf: process with  1  covariance structure(s)
## grf: nugget effect is: tausq= 0
## grf: covariance model 1 is: gaussian(sigmasq=1, phi=0.4)
## grf: simulation using the function GaussRF from package RandomFields
## grf: End of simulation procedure. Number of realizations: 1
```

Usando a função geoR::grf

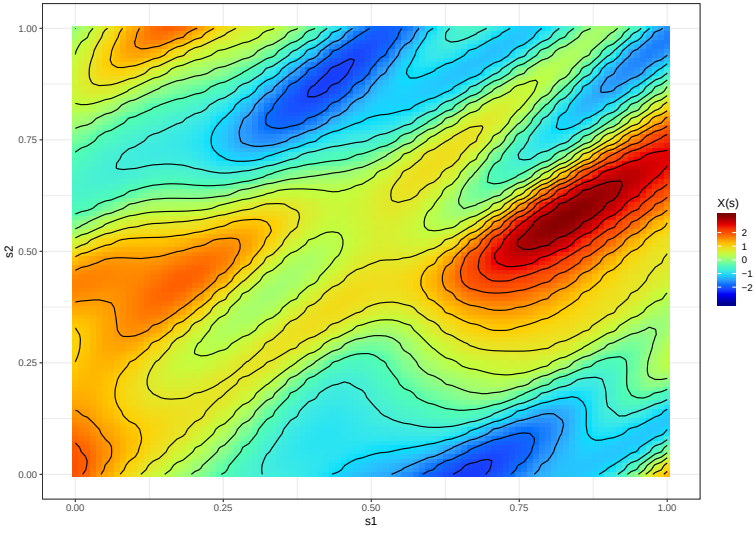


Anisotropia

```
set.seed(1)
xA <- grf(grid = Locations, cov.pars = c(1, .4),
          aniso.pars = c(pi/4, 4),
          cov.model = "gaussian")
```

```
## grf: simulation on a set of locations provided by the user
## grf: process with  1  covariance structure(s)
## grf: nugget effect is: tausq= 0
## grf: covariance model 1 is: gaussian(sigmasq=1, phi=0.4)
## grf: simulation using the function GaussRF from package RandomFields
## grf: End of simulation procedure. Number of realizations: 1
```

Anisotropia

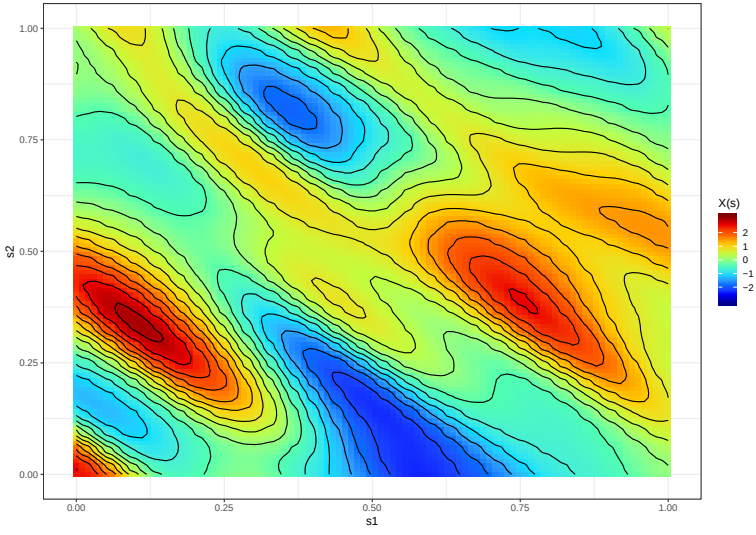


Anisotropia

```
set.seed(2)
xB <- grf(grid = Locations, cov.pars = c(1, .8),
          aniso.pars = c(3*pi/4, 4),
          cov.model = "gaussian")
```

```
## grf: simulation on a set of locations provided by the user
## grf: process with  1  covariance structure(s)
## grf: nugget effect is: tausq= 0
## grf: covariance model 1 is: gaussian(sigmasq=1, phi=0.8)
## grf: simulation using the function GaussRF from package RandomFields
## grf: End of simulation procedure. Number of realizations: 1
```

Anisotropia



Variograma direcional

O variograma direcional é calculado considerando separando os pares de coordenadas $(\mathbf{s}_i, \mathbf{s}_j)$ em caixas com ângulo azimutal $\psi \in [0, \pi)$, medido em ordem horária a partir do Norte, e com algum grau de tolerância δ. Isto é:

$$\hat{\gamma}_n(h_i, \psi, \delta) = \frac{1}{N(h_i, \psi, \delta)} \sum_{(j,\ell) \in N(h_i, \psi, \delta)} (x_\ell - x_j)^2, \quad i = 1, \ldots, k$$

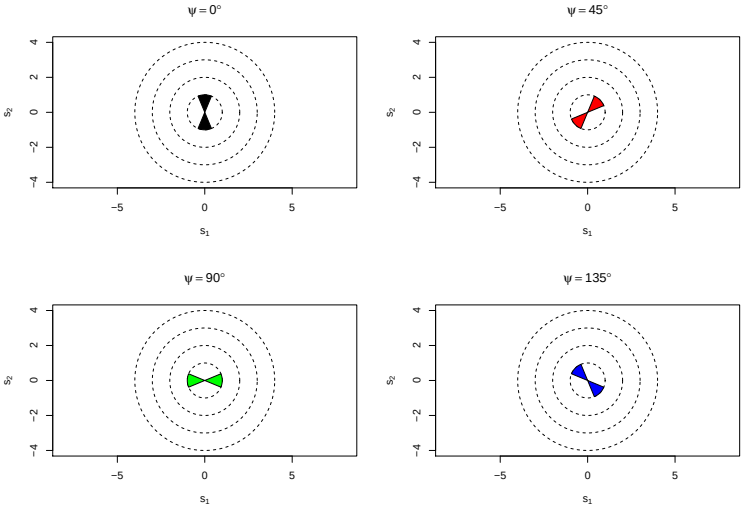
com

$$N(h_i, \psi, \delta) = \left\{ (j, \ell) : \left(\frac{h_{i-1} + h_i}{2} \leq \|d_{\ell,j}\| \leq \frac{h_i + h_{i+1}}{2} \right) \wedge (\psi - \delta \leq \varphi_{\ell,j} \leq \psi + \delta) \right\},$$

em que $d_{\ell,j} = \|\mathbf{s}_j - \mathbf{s}_\ell\|$ e $\varphi_{\ell,j} = \arctan 2(\mathbf{s}_{\ell,1} - \mathbf{s}_{j,1}, \mathbf{s}_{\ell,2} - \mathbf{s}_{j,2})$ (se o valor for negativo, some π; cf. https://en.wikipedia.org/wiki/Azimuth).

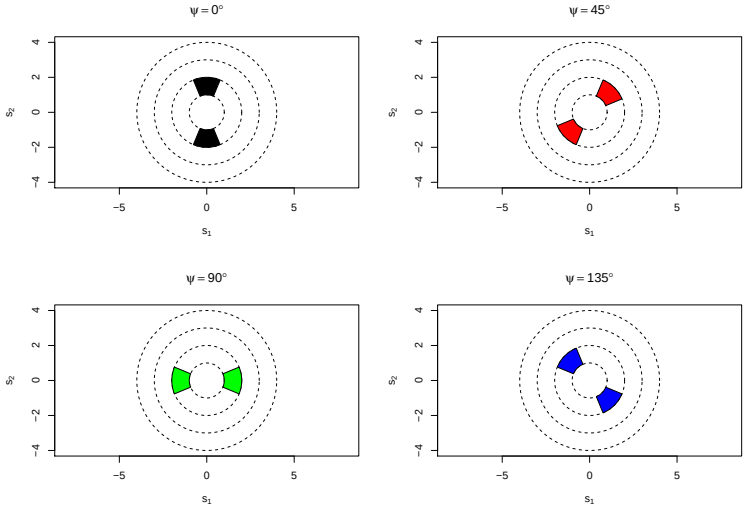
Variograma direcional

Distância $h_1 = 0.5$



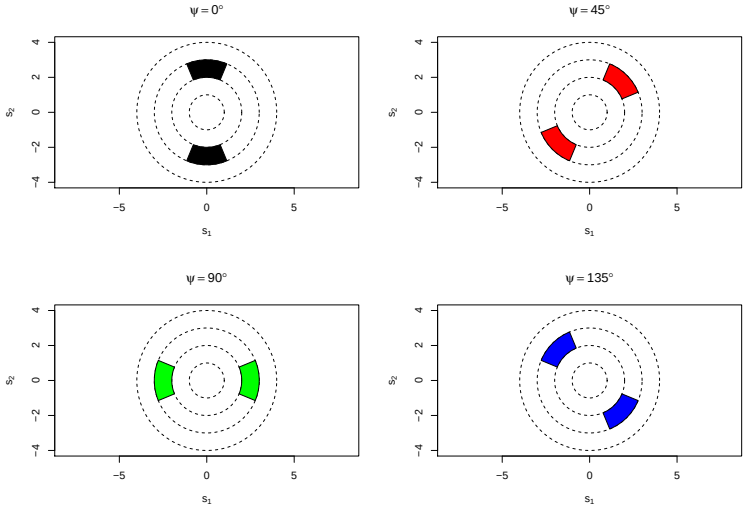
Variograma direcional

Distância $h_2 = 1.5$



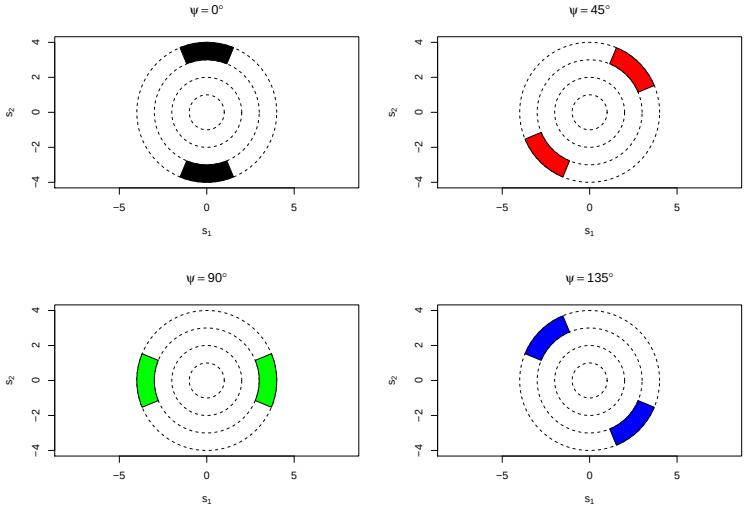
Variograma direcional

Distância $h_3 = 2.5$



Variograma direcional

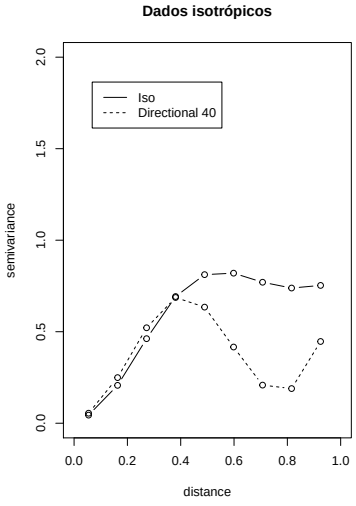
Distância $h_4 = 3.5$

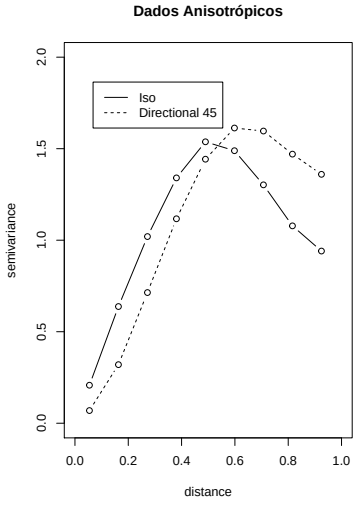


Variograma direcional

```
# Dados isotropicos
model00 <- variog(coords = Locations,
                  data = x$data)
model0A <- variog(coords = Locations,
                  data = x$data,
                  direction = pi/4, tolerance = pi/8)
# Dados anisotropicos
modelA0 <- variog(coords = Locations,
                  data = xA$data)
modelAA <- variog(coords = Locations,
                  data = xA$data,
                  direction = pi/4, tolerance = pi/8)
```

Variograma direcional





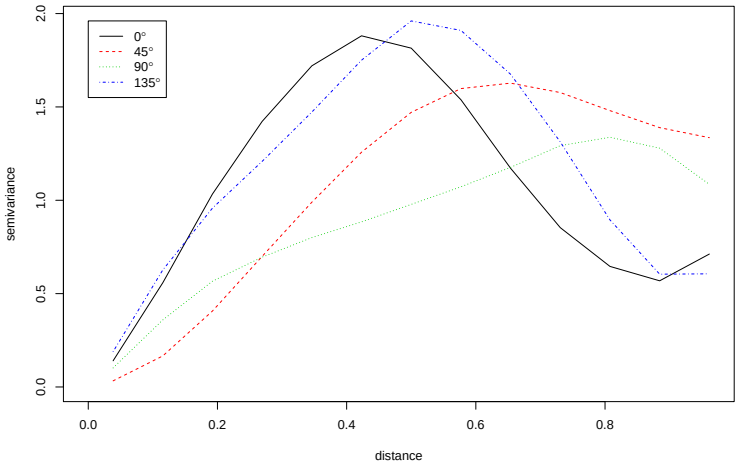
Variograma direcional

```
model4 <- variog4(coords = Locations,
                  data = xA$data,
                  direction = c(0, pi/4, pi/2, 3*pi/4), tolerance = pi/8,
                  max.dist = 1)
```

```
## variog: computing variogram for direction = 0 degrees (0 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 45 degrees (0.785 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 90 degrees (1.571 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 135 degrees (2.356 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing omnidirectional variogram
```

Variograma direcional

```
plot(model4, xlim = c(0,1))
```

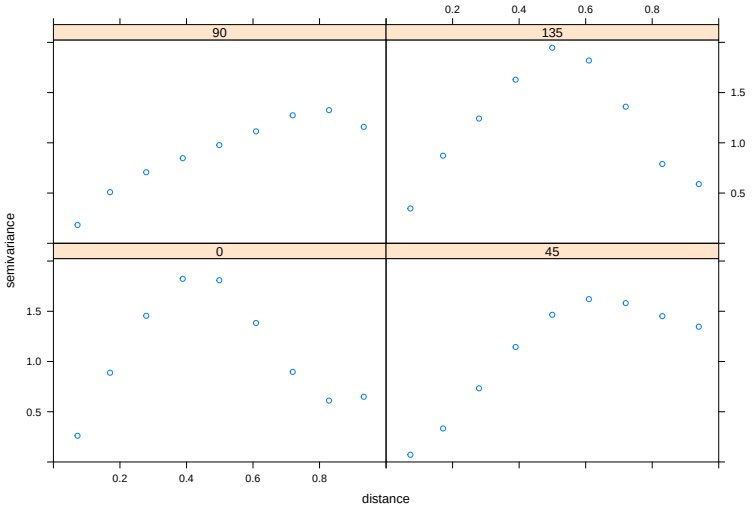


Comparação gstat

Leiam: https://cran.r-project.org/web/packages/gstat/vignettes/gstat.pdf

```
library(gstat)
library(sp)
dataset <- cbind(Locations, y = xA$data)
coordinates(dataset) <- ~ s1 + s2
vg.dir <- variogram(y ~ 1, dataset,
                    alpha = c(0, 45, 90, 135),
                    cutoff = 1, width = 1/9) # equiv: max dist
```

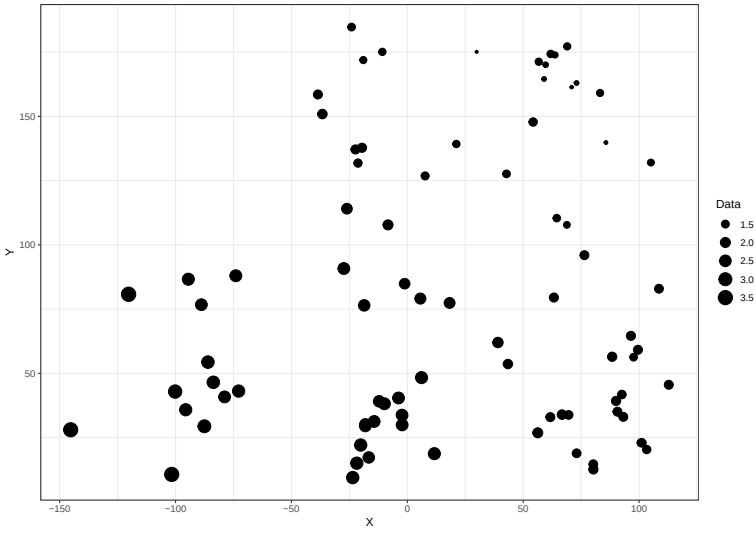
Comparação gstat



Voltando ao exemplo do Wolfcamp Aquifer

```
library(geoR)
library(fields) # filled.contour
wolfcamp <- read.csv("wolfcamp.csv", skip = 1)
```

Wolfcamp-Aquifer data



Variogramas direcionais

```
# Isotropic
model <- variog(coords = wolfcamp[,c("X","Y")],
                data = wolfcamp[,"Data"], max.dist = 120)
```

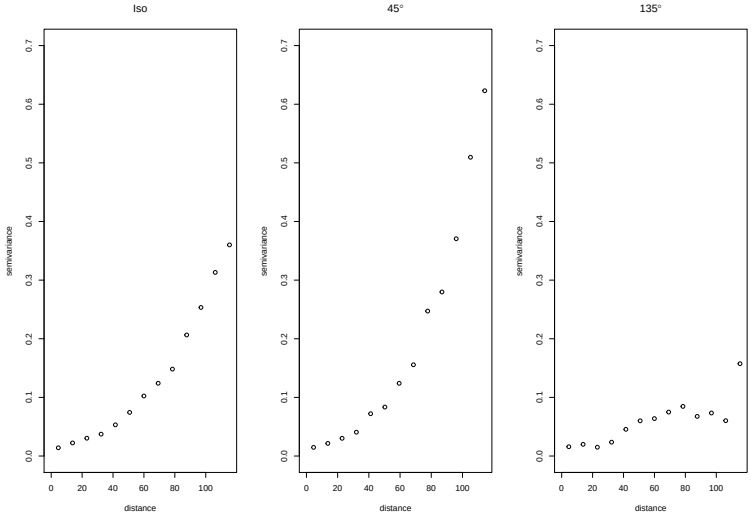
```
## variog: computing omnidirectional variogram
```

```
# Anisotropic
model2 <- variog(coords = wolfcamp[,c("X","Y")],
                 data = wolfcamp[,"Data"],
                 direction = pi/4, max.dist = 120)
```

```
## variog: computing variogram for direction = 45 degrees (0.785 radians)
##         tolerance angle = 22.5 degrees (0.393 radians)
```

```
model3 <- variog(coords = wolfcamp[,c("X","Y")],
                 data = wolfcamp[,"Data"],
                 direction = 3*pi/4, max.dist = 120)
```

Variogramas direcionais



Modelos isotrópico e anisotrópico

```
ml <- likfit(coords = wolfcamp[,c("X","Y")],
             data = wolfcamp[,"Data"],
             trend = "2nd", # options: "cte", "1st", "~ ..."
             ini.cov.pars = c(sigma2 = 3, phi = 30),
             cov.model = "gaussian", fix.nugget = FALSE,
             messages = FALSE)
mlA <- likfit(coords = wolfcamp[,c("X","Y")],
              data = wolfcamp[,"Data"],
              trend = "2nd", # options: "cte", "1st", "~ ..."
              ini.cov.pars = c(sigma2 = 3, phi = 30),
              cov.model = "gaussian", fix.nugget = FALSE,
              messages = FALSE,
              fix.psiA = TRUE, psiA = 3*pi/4,
              fix.psiR = FALSE)
```

Parâmetros do modelo isotrópico

```
ml
```

```
## likfit: estimated model parameters:
##     beta0     beta1     beta2     beta3     beta4     beta5     tausq
## " 2.4696" "-0.0081" "-0.0027" " 0.0000" " 0.0000" " 0.0000" " 0.0200"
##   sigmasq       phi
## " 0.0157" "33.2948"
## Practical Range with cor=0.05 for asymptotic range: 57.62729
##
## likfit: maximised log-likelihood = 30.36
```

Parâmetros do modelo anisotrópico

```
mlA
```

```
## likfit: estimated model parameters:
##     beta0     beta1     beta2     beta3     beta4     beta5     tausq
## " 2.4874" "-0.0083" "-0.0021" " 0.0000" " 0.0000" " 0.0000" " 0.0146"
##   sigmasq       phi      psiR
## " 0.0199" "15.5383" " 2.2979"
## Practical Range with cor=0.05 for asymptotic range: 26.89394
##
## likfit: maximised log-likelihood = 31.14
```

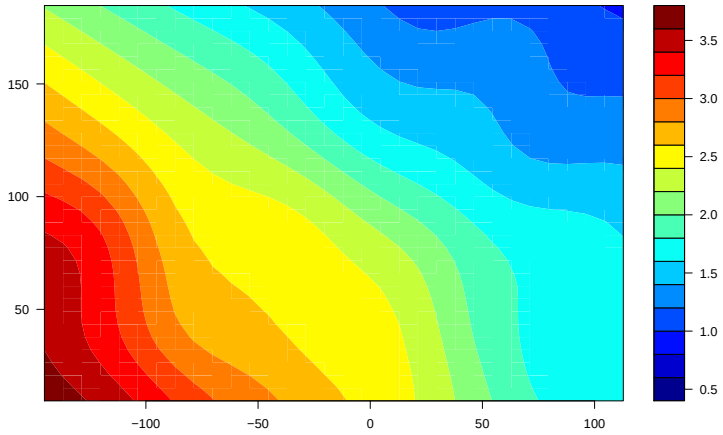
Modelos isotrópico e anisotrópico

```
prepUK <- krige.control(type.krige = "ok",
                        trend.d = ml$trend,
                        trend.l = ml$trend,
                        cov.model = ml$cov.model,
                        cov.pars = ml$cov.pars,
                        nugget = ml$nugget)
prepUKA <- krige.control(type.krige = "ok",
                         trend.d = mlA$trend,
                         trend.l = mlA$trend,
                         cov.model = mlA$cov.model,
                         cov.pars = mlA$cov.pars,
                         nugget = mlA$nugget,
                         aniso.pars = mlA$aniso.pars)
```

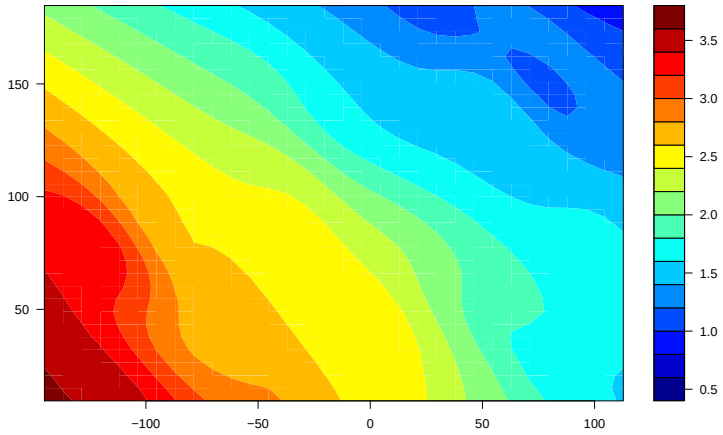
Predição

```
loci <- expand.grid(X = seq(min(wolfcamp$X), max(wolfcamp$X),
                            length.out = 32),
                    Y = seq(min(wolfcamp$Y), max(wolfcamp$Y),
                            length.out = 32))
kcUK <- krige.conv(coords = wolfcamp[,c("X","Y")],
                   data = wolfcamp[,"Data"],
                   locations = loci,
                   krige = prepUK)
kcUKA <- krige.conv(coords = wolfcamp[,c("X","Y")],
                   data = wolfcamp[,"Data"],
                   locations = loci,
                   krige = prepUKA)
```

Kriging: UK



Kriging: UKA



A geometria da distância

- Em um mapa, é possível determinar distâncias a partir de localizações. O problema inverso, determinar localizações através de distâncias, a menos de rotações do sistema de coordenadas, é um dos objetivos de *multidimensional scaling*.
- A técnica de *multidimensional scaling* também é uma forma de redução não-linear de dimensão. Por exemplo, n pontos que vivem no $\mathbb{R}^p$ têm uma matriz de distância $\mathbf{D}_{n\times n}$. Com base em $\mathbf{D}$, as coordenadas podem ser reconstruídas em um espaço $\mathbb{R}^q$. Em particular, se $q = 2$, obtemos um *scatterplot* interpretável.
- Na prática, podemos estender o método para medidas de proximidade que não sejam distâncias.
- *Multidimensional scaling* (MDS) também é conhecida como *Principal Coordinate Analysis*.

Teoria: metric distance

Seja δ_{ij} uma medida de dissimilaridade entre a observação i e a observação j. A matriz $n \times n$ de proximidade é denotada por $\mathbf{D}$; a princípio,

1. os elementos da diagonal de $\mathbf{D}$ são zero,
2. e $\mathbf{D}$ é simétrica.

A distância é chamada *metric distance* se satisfaz a desigualdade triangular, isto é

$$\delta_{ij} \leq \delta_{ik} + \delta_{kj}, \text{ para todo } k.$$

A primeira condição é imperativa; mas é fácil pensar em exemplos onde $\mathbf{D}$ não é simétrica (e.g. *Kullback-Leibler Divergence*) ou não-*metric* (por exemplo, a distância geográfica na superfície da Terra). O caso em que as três condições são satisfeitas chama-se *metric dimensional scaling*.

Teoria: metric multidimensional scaling

Considere $\mathbf{X}_1, \ldots, \mathbf{X}_n$ vetores aleatórios no $\mathbb{R}^p$. Com base nesses vetores, determinamos a matriz de distâncias Euclidianas como sendo

$$\delta_{ij} = \|\mathbf{X}_i - \mathbf{X}_j\| = \left\{ \sum_{k=1}^{p} (X_{ik} - X_{jk})^2 \right\}^{1/2}.$$

Note então que

$$\delta_{ij}^2 = \|\mathbf{X}_i\|^2 + \|\mathbf{X}_j\|^2 - 2\mathbf{X}_i^t \mathbf{X}_j.$$

Seja

$$b_{ij} = \mathbf{X}_i^t \mathbf{X}_j = -\frac{1}{2}(\delta_{ij}^2 - \delta_{i0}^2 - \delta_{j0}^2),$$

onde $\delta_{i0}^2 = \|\mathbf{X}_i\|^2$ é a distância ao quadrado de $\mathbf{X}_i$ até a origem.

Teoria: metric multidimensional scaling

Se somarmos alguns dos elementos anteriores nos índices i e j, podemos obter as seguintes identidades:

$$n^{-1}\sum_{i}\delta_{ij}^2 = n^{-1}\sum_{i}\delta_{i0}^2 + \delta_{j0}^2$$
$$n^{-1}\sum_{j}\delta_{ij}^2 = \delta_{i0}^2 + n^{-1}\sum_{j}\delta_{j0}^2$$
$$n^{-1}\sum_{ij}\delta_{ij}^2 = 2n^{-1}\sum_{i}\delta_{i0}^2,$$

que podem ser substituídas na equação do slide anterior, para determinar

$$b_{ij} = a_{ij} - a_{i\cdot} - a_{\cdot j} + a_{\cdot\cdot},$$

onde $a_{ij} = -\frac{1}{2}\delta_{ij}^2$, $a_{i\cdot} = n^{-1}\sum_j a_{ij}^2$, etc.

Teoria: metric multidimensional scaling

Em notação matricial, se $\mathbf{A}$ tem coordenadas δ_{ij}^2, então

$$\mathbf{B} = \mathbf{HAH},$$

onde $\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^t$.

No sentido de redução de dimensão, queremos encontrar $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \in \mathbb{R}^q$, $q < p$, tais que a distância entre $\mathbf{Y}_i, \mathbf{Y}_j$ seja (pelo menos aproximadamente) a mesma que entre $\mathbf{X}_i, \mathbf{X}_j$. Em outras palavras, queremos encontrar uma matriz $\mathbf{B}^*$ tal que

$$\min_{\mathbf{B}^*} \operatorname{tr}\{(\mathbf{B} - \mathbf{B}^*)^2\} = \min_{\boldsymbol{\lambda}^*} \sum_{k=1}^{q} (\lambda_k - \lambda_k^*)^2.$$

Isso mostra que o algoritmo dependerá dos autovalores de $\mathbb{B}$.

Construindo as coordenadas

Com base na decomposição espectral de $\mathbf{B}$, temos

$$\mathbf{B}_{n\times n} = \mathbf{Q}_{n\times q}\mathbf{\Lambda}_{q\times q}\mathbf{Q}^t_{q\times n}.$$

Em outras palavras

$$\mathbf{B}_{n\times n} = \mathbf{Q}\mathbf{\Lambda}^{1/2}(\mathbf{Q}\mathbf{\Lambda}^{1/2})^t = \mathbf{Y}\mathbf{Y}^t,$$

com $\mathbf{Y}_{n\times q}$. Em suma, as linhas de $\mathbf{Y}_{n\times q}$, dadas por $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ são de distância igual a $\mathbf{X}_1, \ldots, \mathbf{X}_n$, desde que $\mathbf{B}$ tenha apenas q autovalores positivos (o resto é nulo). De fato, há redução de dimensão.

Se nem todos os autovalores de $\mathbb{B}$ são positivos, podemos truncá-los no zero, ou adicionar uma constante a todos $\delta_{i'j'} \leftarrow \delta_{ij} + c$ tais que $i \neq j$.

Distâncias

Os dados só tem a diagonal inferior da matriz de distâncias; note que são *distâncias geográficas*, sobre a superfície da terra, e não Euclidianas.

```
temp <- read.csv("atlas_dist.csv", row.names = 1)
D <- matrix(0, nrow = nrow(temp) + 1, ncol = ncol(temp) + 1)
D[lower.tri(D)] <- temp[lower.tri(temp, diag = TRUE)]
D <- D + t(D)
rownames(D) <- c(colnames(temp)[1], rownames(temp))
colnames(D) <- c(colnames(temp)[1], rownames(temp))
```

Distâncias (superfície da terra)

```
##                Beijing Cape Town Hong Kong Honolulu London Melbourne
## Beijing              0     12947      1972     8171   8160      9093
## Cape Town        12947         0     11867    18562   9635     10338
## Hong Kong         1972     11867         0     8945   9646      7392
## Honolulu          8171    18562      8945        0  11653      8862
## London            8160     9635      9646    11653      0     16902
## Melbourne         9093    10338      7392     8862  16902         0
## Mexico City      12478    13703     14155     6098   8947     13557
## Montreal         10490    12744     12462     7915   5240     16730
## Moscow            5809    10101      7158    11342   2506     14418
## New Delhi         3788     9284      3770    11930   6724     10192
## New York         11012    12551     12984     7996   5586     16671
## Paris             8236     9307      9650    11988    341     16793
## Rio de Janeiro   17325     6075     17710    13343   9254     13227
## Rome              8144     8417      9300    12936   1434     15987
## San Francisco     9524    16487     11121     3857   8640     12644
## Singapore         4465     9671      2575    10824  10860      6050
```

Distâncias (superfície da terra)

Note que o teorema de Pitágoras não vale porque, afinal, a superfície da Terra é apenas localmente Euclidiana.

```
(Delta <- D[c(1,2,13), c(1,2,13)]^2)
```

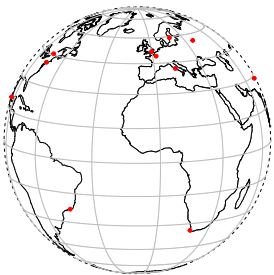
```
##                   Beijing Cape Town Rio de Janeiro
## Beijing                 0 167624809      300155625
## Cape Town       167624809         0       36905625
## Rio de Janeiro 300155625  36905625              0
```

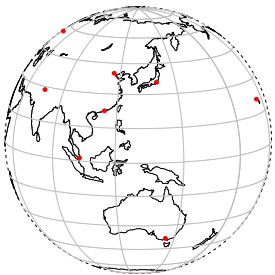
Mas

```
Delta[1,3] > Delta[1,2] + Delta[2,3]
```

```
## [1] TRUE
```

Coordenadas (Longitude, Latitude)



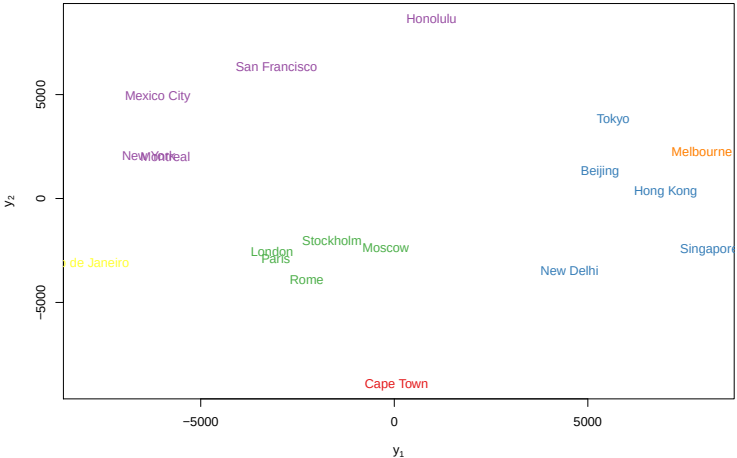


Código

```
(model <- cmdscale(D, k = 2))
```

```
##                     [,1]       [,2]
## Beijing         5315.2435  1272.9019
## Cape Town         57.6330 -8935.1436
## Hong Kong       7010.8954   306.5171
## Honolulu         962.8597  8677.0476
## London         -3157.5308 -2557.9604
## Melbourne       7948.2898  2283.6741
## Mexico City    -6108.9675  4896.6387
## Montreal       -5912.5724  2039.6963
## Moscow          -220.8415 -2377.2712
## New Delhi       4528.9413 -3474.3298
## New York       -6341.0219  2078.6619
## Paris          -3058.2970 -2910.0764
## Rio de Janeiro -7905.6030 -3067.3367
## Rome           -2262.2577 -3916.4707
```

Visualização

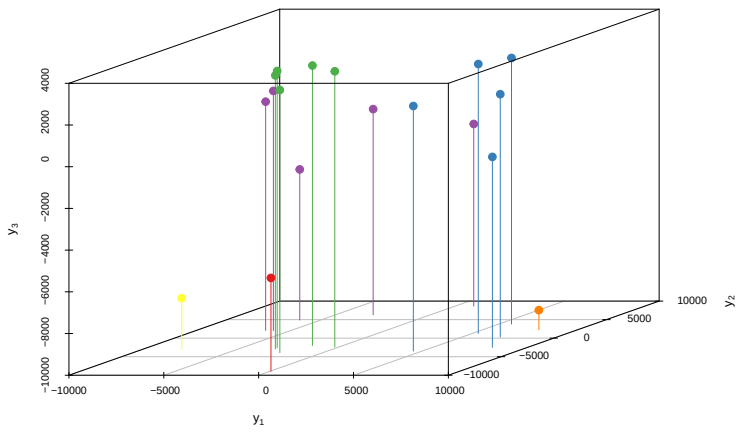


Código

```
(model <- cmdscale(D, k = 3))
```

```
##                     [,1]       [,2]       [,3]
## Beijing         5315.2435  1272.9019  2920.7522
## Cape Town         57.6330 -8935.1436 -5522.2562
## Hong Kong       7010.8954   306.5171  1645.5346
## Honolulu         962.8597  8677.0476 -1270.4739
## London         -3157.5308 -2557.9604  3268.1060
## Melbourne       7948.2898  2283.6741 -9062.2778
## Mexico City    -6108.9675  4896.6387 -2778.0397
## Montreal       -5912.5724  2039.6963  1495.9200
## Moscow          -220.8415 -2377.2712  3221.2239
## New Delhi       4528.9413 -3474.3298  1751.4970
## New York       -6341.0219  2078.6619   972.3898
## Paris          -3058.2970 -2910.0764  3118.9505
## Rio de Janeiro -7905.6030 -3067.3367 -7537.6871
## Rome           -2262.2577 -3916.4707  2595.8542
```

Visualização



Referências I

Cressie, N. (1993). *Statistics for Spatial Data, 2nd edition*. Wiley, New York.

Diggle, P. J. and Ribeiro, P. J. (2007). *Model-based Geostatistics*. Springer, New York.