MBA Big Data e Business Intelligence

SOM – Self Organizing Map

Prof. Abraham Laredo Sicsu

- Ótimo tutorial

- https://www.shanelynn.ie/self-organising-maps-for-customer-segmentation-using-r/

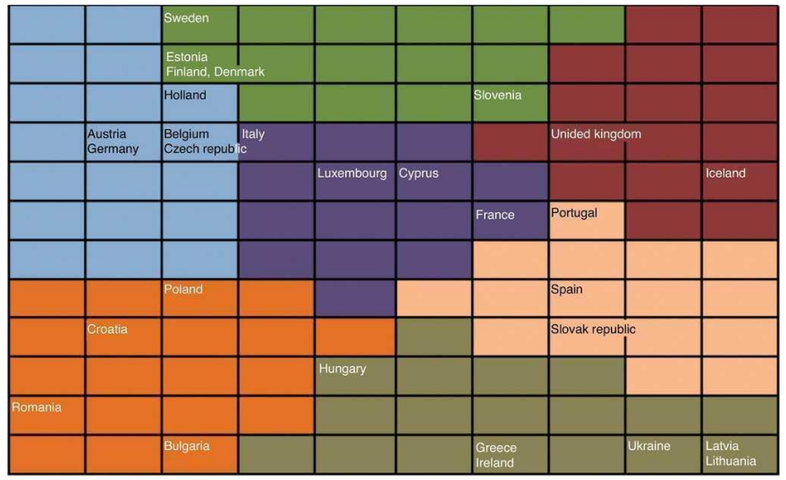
- Boa leitura

- https://www.semanticscholar.org/paper/Brief-review-of-self-organizing-maps-Miljkovic/1990b2066181e743f7efbaad7c00113e134dbdf3

SOM objetivos

- Nuvem de N pontos no espaço ➔ mapa bidimensional com M pontos (**neurônios**) (M<<<N)
- **Neurônios** vizinhos ⟺ **observações similares** da nuvem
- Vantagem : **visualização** da estrutura de dados em 2 dimensões

p dimensões



The Spanish Review of Financial Economics. 2013;11:69-84

2 dimensões

SOM Self-organizing maps

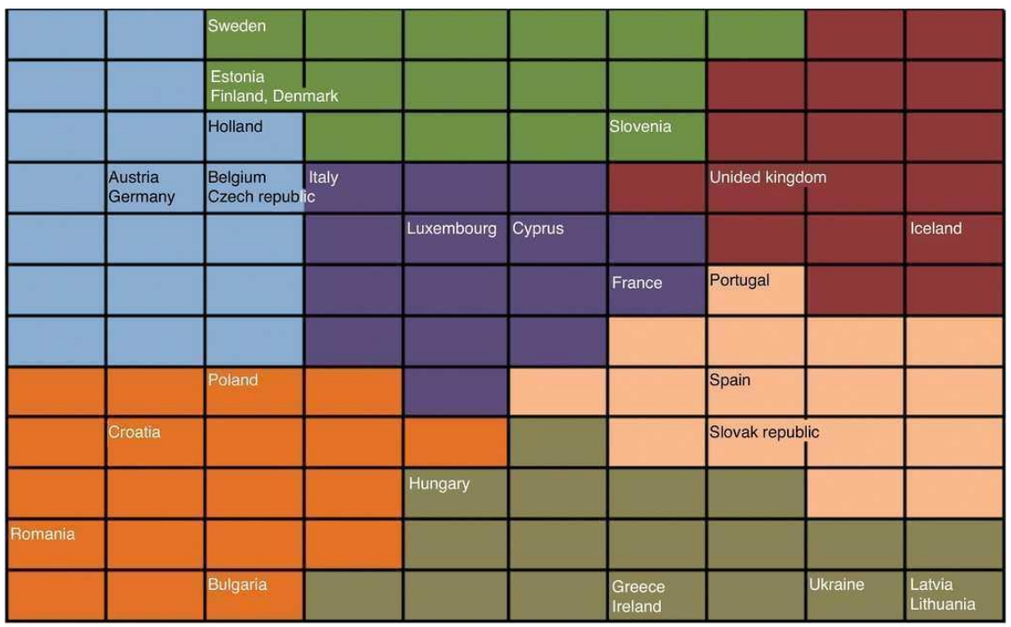
- Pode ter uma, duas ou mais dimensões. Em geral: 2

- Técnica de redução de dimensionalidade (extensão da ideia de PCA)

  - PCA ← redução do número de variáveis

  - SOM ← redução do número de pontos + mapeamento

- É um tipo de rede neural (*competitive learning network*)

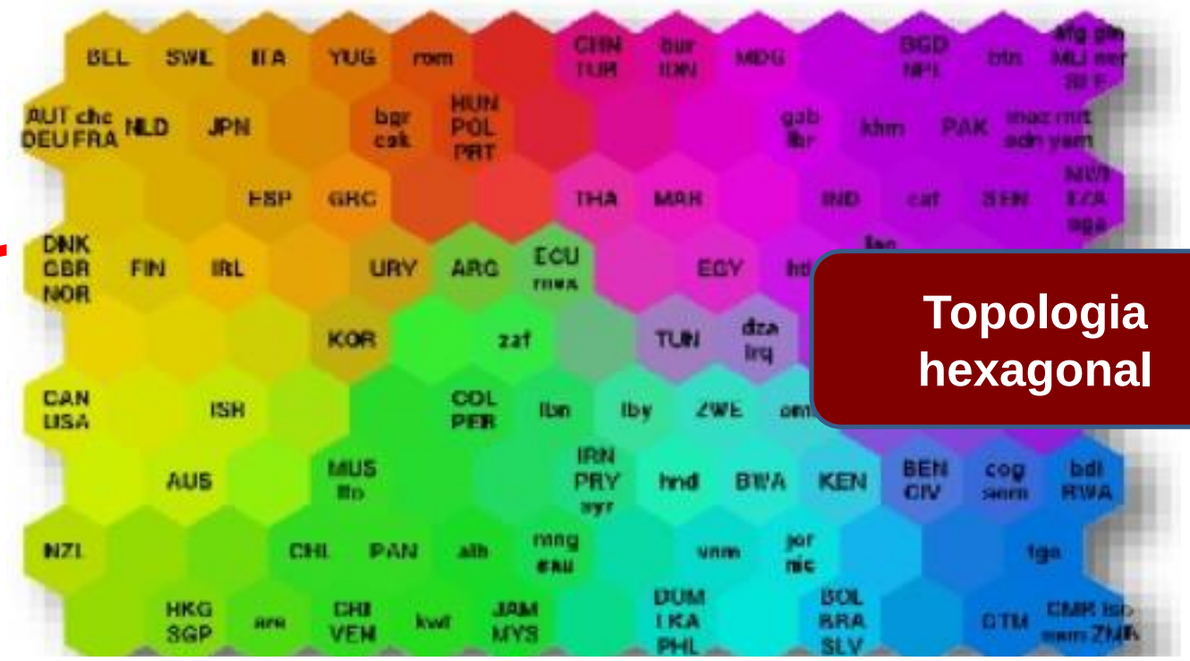


The Spanish Review of Financial Economics. 2013;11:69–84

Diferentes estruturas

Nuvem de N países (p variáveis)

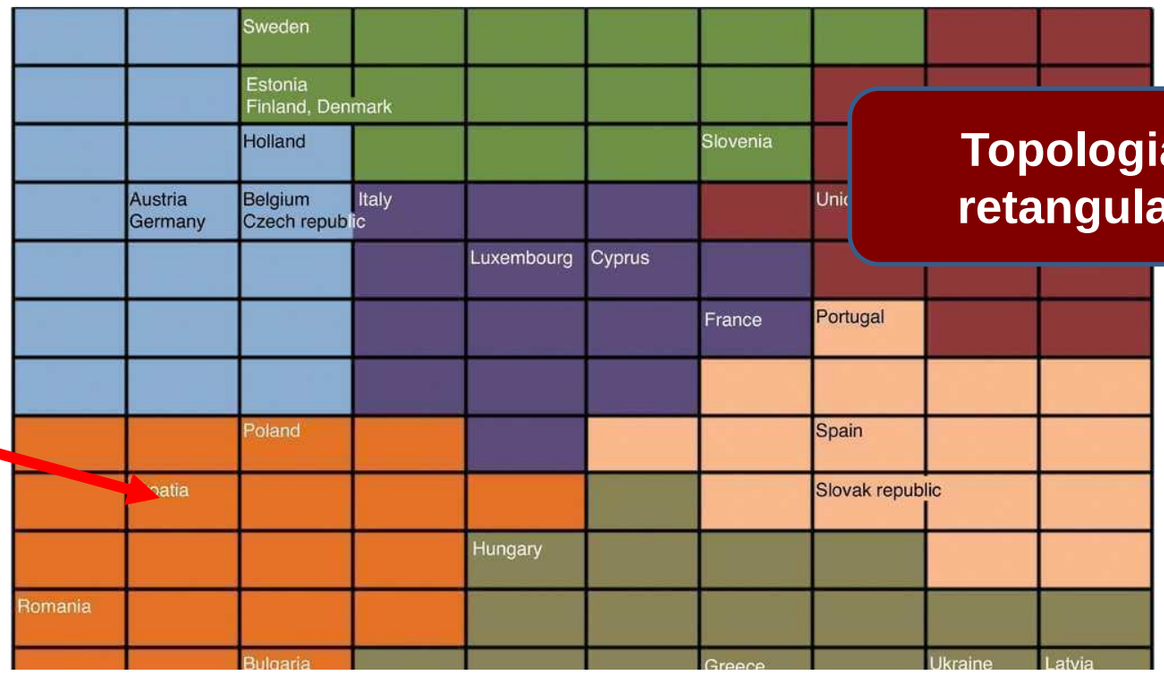
Espaço p-dimensional



Topologia hexagonal

https://www.slideshare.net/KirillEremenko/deep-learning-az-self-organizing-maps-som-module-4

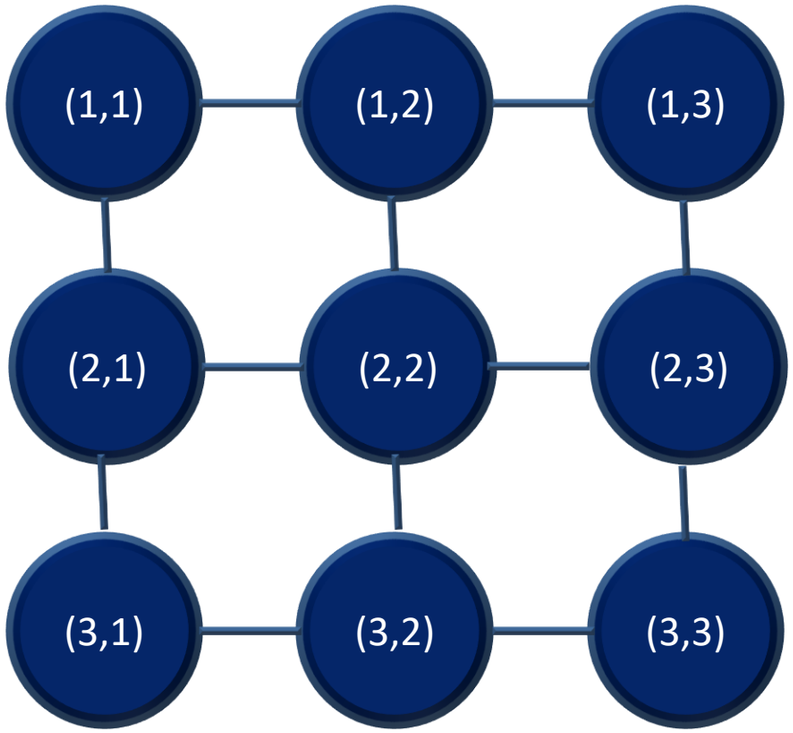
Nuvem de N países (p variáveis)

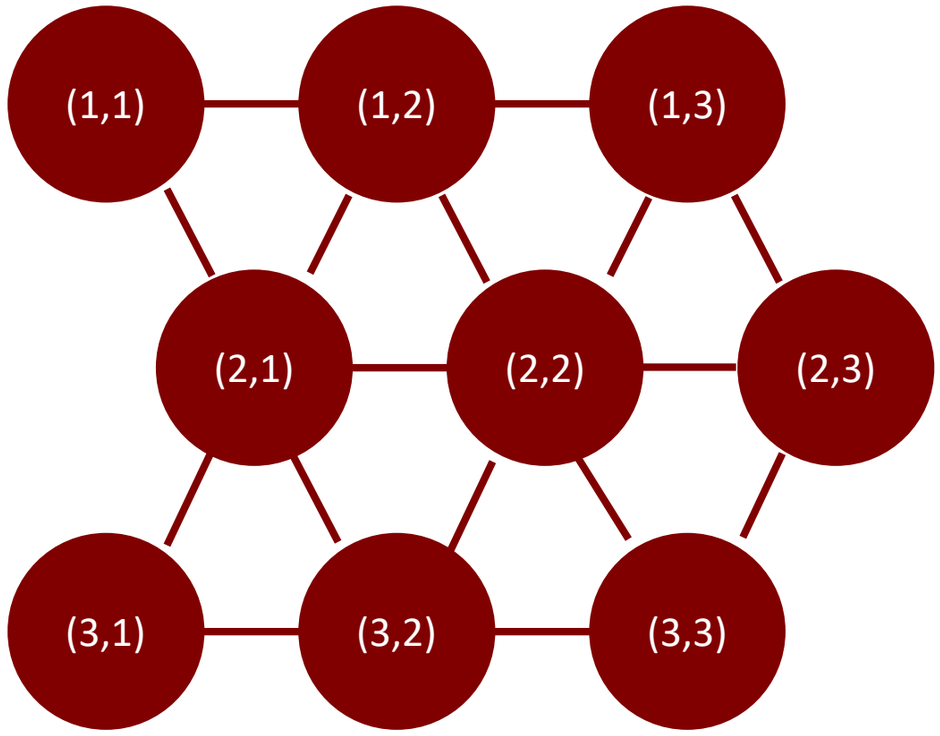


Topologia retangular

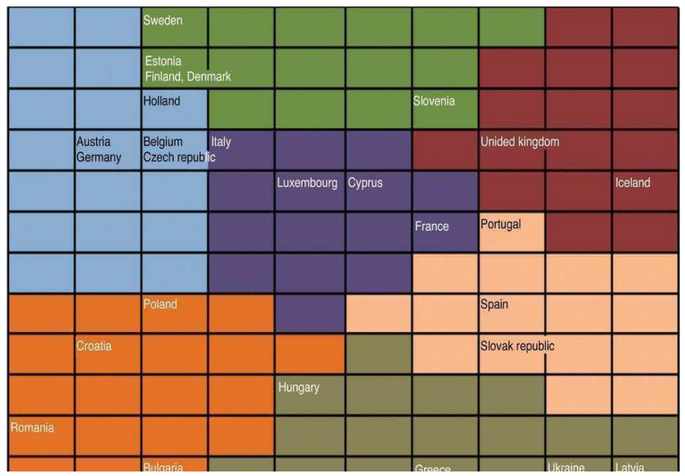
Self-organizing maps as a tool to compare financial macroeconomic imbalances: The European, Spanish and German case

Diferentes estruturas



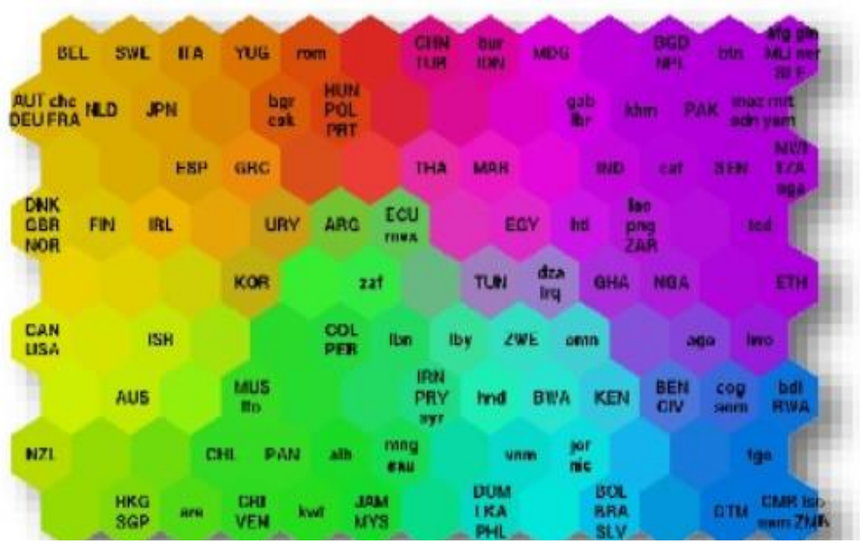


retangular



Self-organizing maps as a tool to compare financial macroeconomic imbalances: The European, Spanish and German case

hexagonal



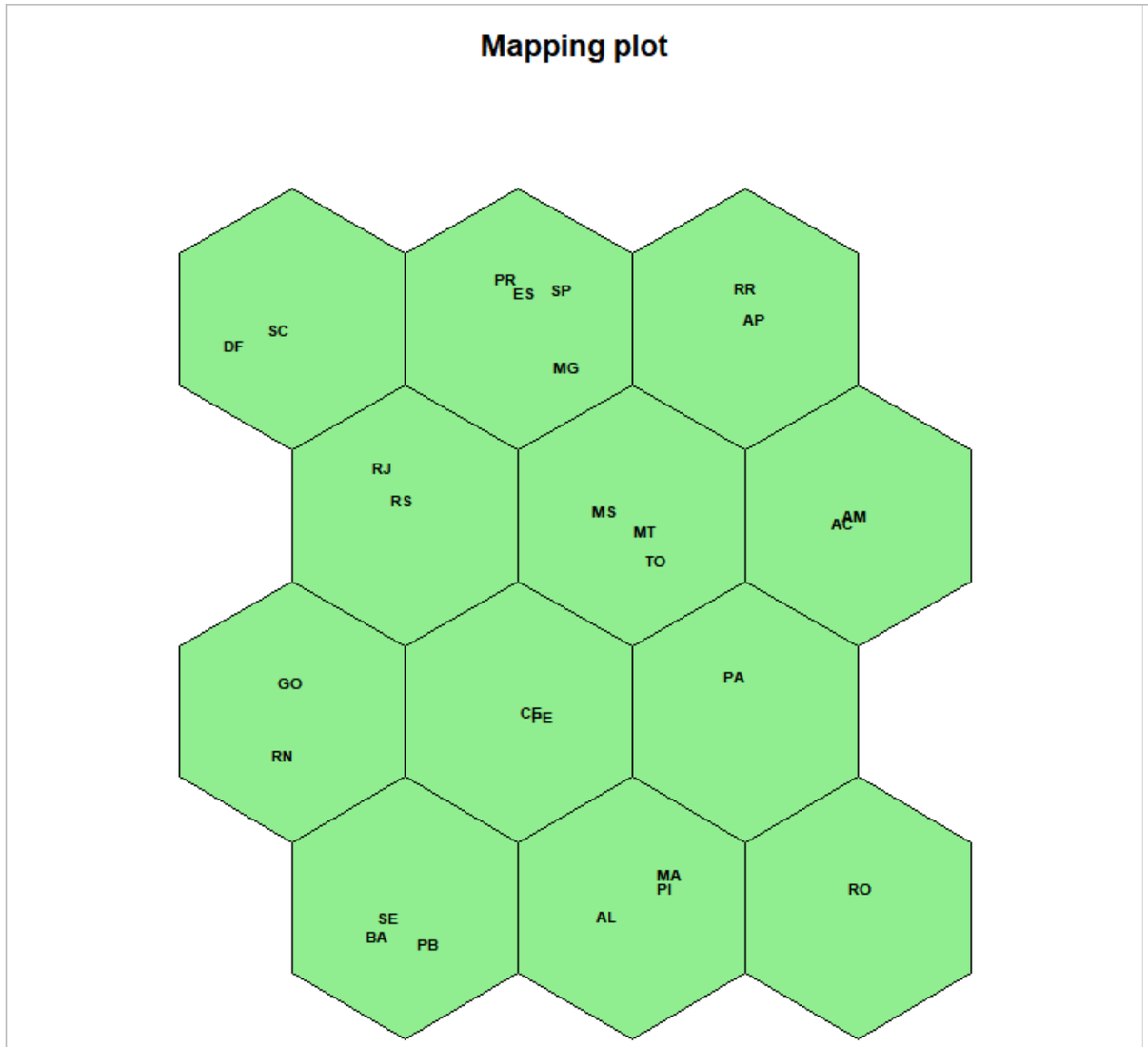
https://www.slideshare.net/KirillEremenko/deep-learning-az-self-organizing-maps-som-module-4

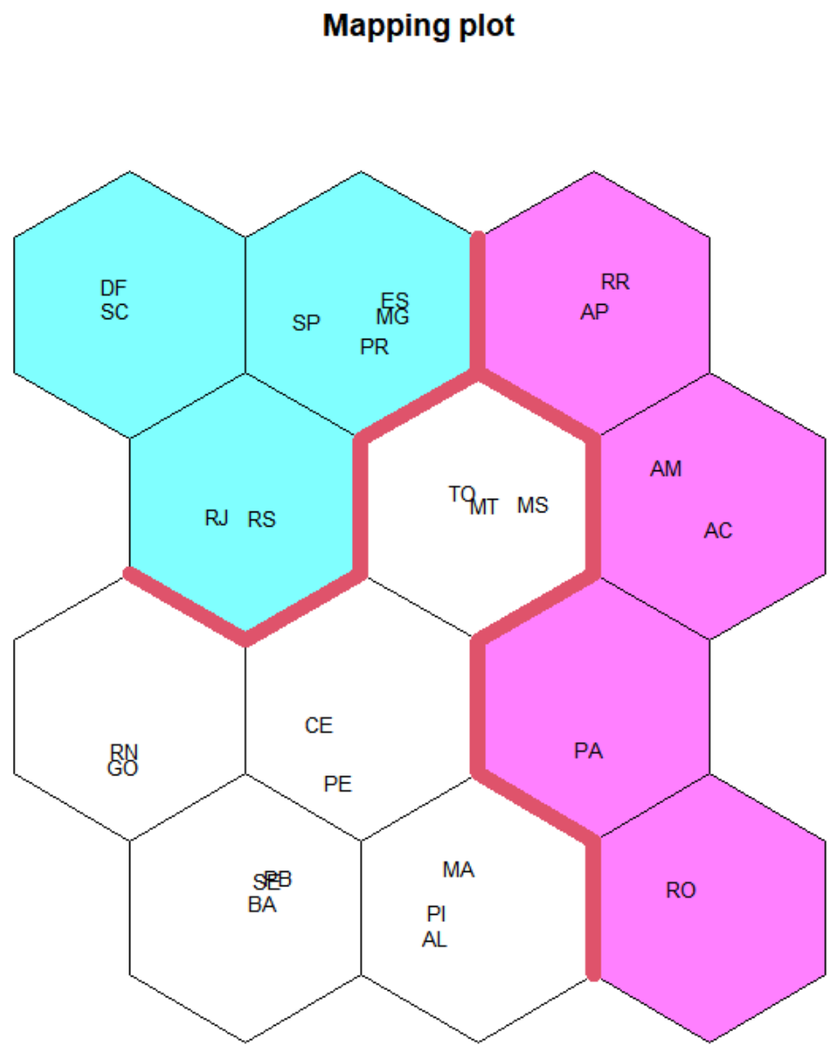
Self-organizing maps

- Não é método supervisionado
- **Um** neurônio se conecta a **um ou mais** pontos na nuvem
- Ajuste do mapa é através de **competição entre os neurônios** (*competitive learning*)
  - Quem é o neurônio mais próximo de cada ponto da nuvem?
- O mapa vai se organizando sozinho a cada iteração em função dos dados da nuvem
- Neurônios não se conectam entre si.

SOM e cluster analysis

- Consideremos UF do Brasil
- Primeiro os países semelhantes são associados ao mesmo neurônio
- Depois fazemos os clusters dos neurônios (indiretamente estamos agrupando as UF)

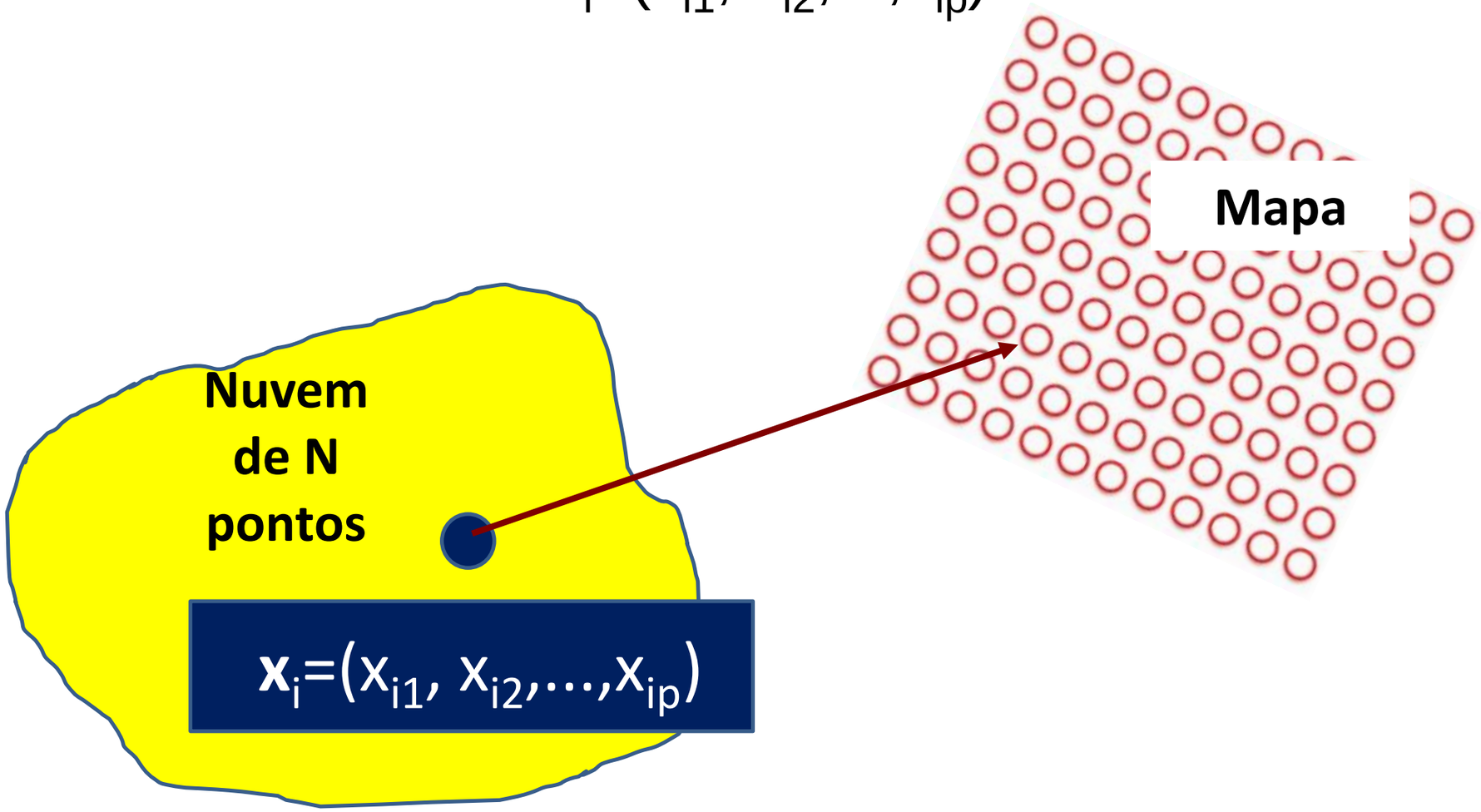




Pontos da nuvem p-dimensional

- **p**= número de variáveis que caracterizam cada observação
- N: tamanho da nuvem (# observações)
- i=1,..., N número de pontos na nuvem no espaço p-dimensional

$$\mathbf{x}_i=(x_{i1}, x_{i2},\ldots,x_{ip})$$

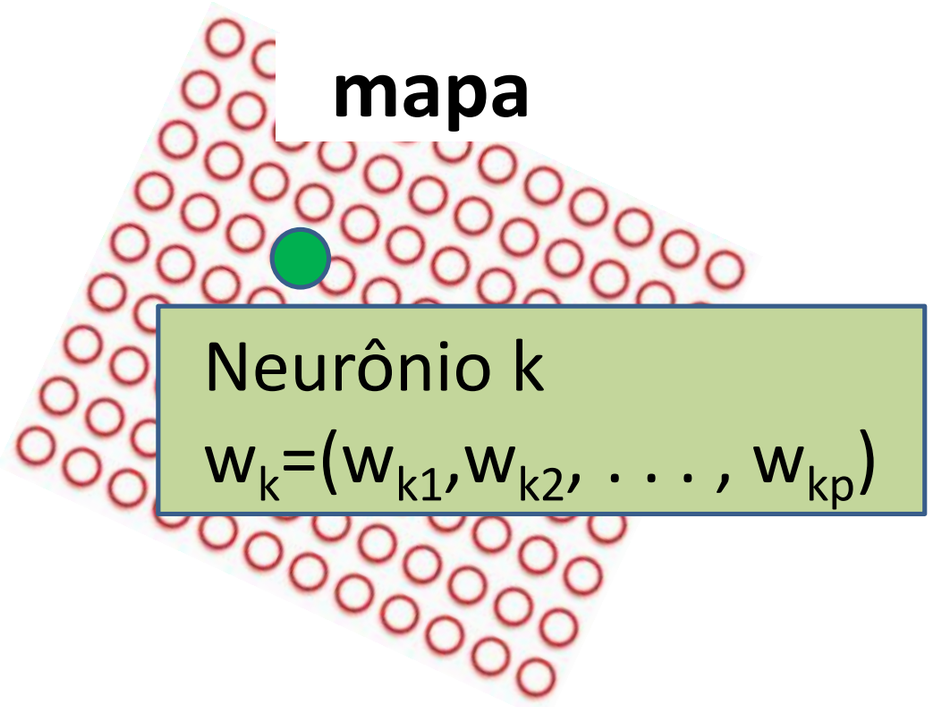


Neurônios do mapa (reticulado)

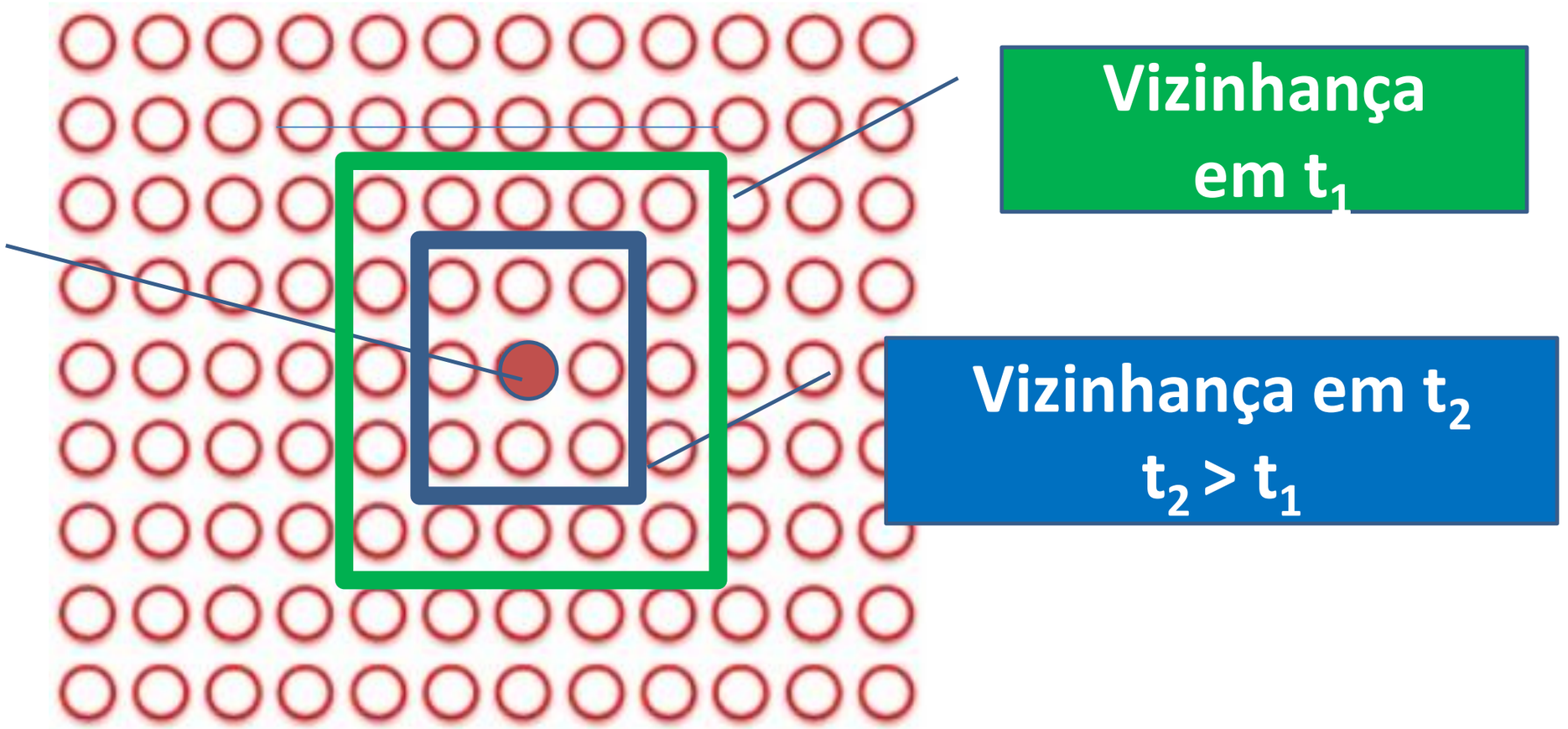
- A cada neurônio do mapa corresponderá um vetor de dimensão p (*code vector*)
  - **p**= dimensão dos espaço que contem as observações
  - $\mathbf{w_{ki}}$ pesos correspondentes ao neurônio k=1,..,M (**M**= # neurônios)

$$w_k=(w_{k1},w_{k2}, \ldots , w_{kp})$$

  - ***codebook*** (dataset com os pesos dos neurônios→ M linhas, p colunas)



Neurônios na vizinhança de k

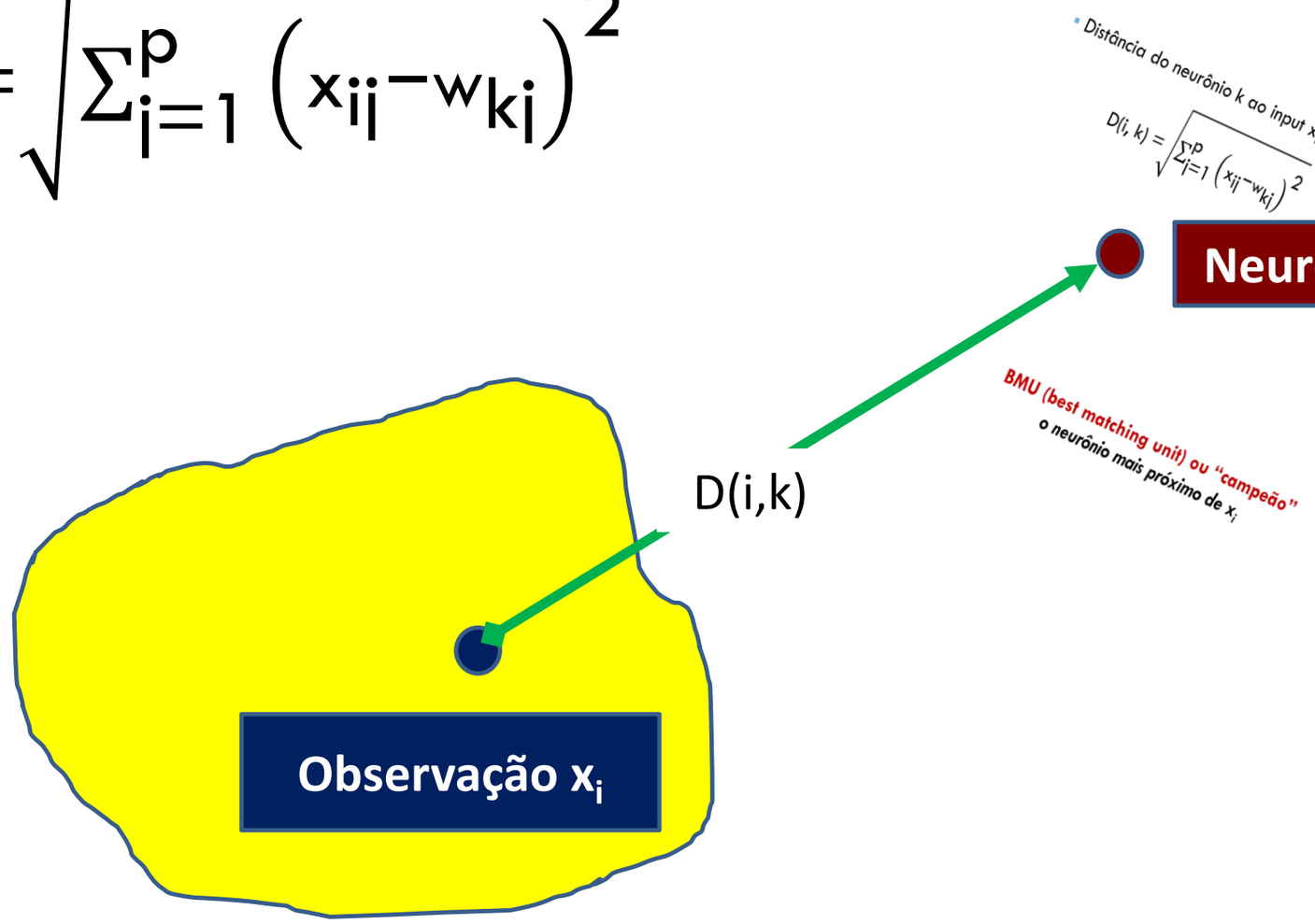


- Vizinhos: neurônios dentro da cerca

Distância entre o ponto i e o neurônio k

- Distância do neurônio k ao input x_i

$$D(i,k) = \sqrt{\sum_{j=1}^{p} \left(x_{ij} - w_{kj}\right)^2}$$



BMU (best matching unit) ou "campeão"

o neurônio mais próximo de x_i

Etapas do SOM – sumário

1. Amostragem: uma observação X da nuvem de pontos é selecionada aleatoriamente
2. Competição: determinar neurônio BMU (mais próximo de X)
3. Adaptação: ajustar os vetores de pesos do BMU e seus vizinhos aproximando os neurônios mais ainda de X

Retornar ao passo 1 até que a topografia do mapa não se altere (convergência)

Conduzindo os neurônios em direção a X

- Pesos dos neurônios são alterados para reduzir ainda mais a distância de X

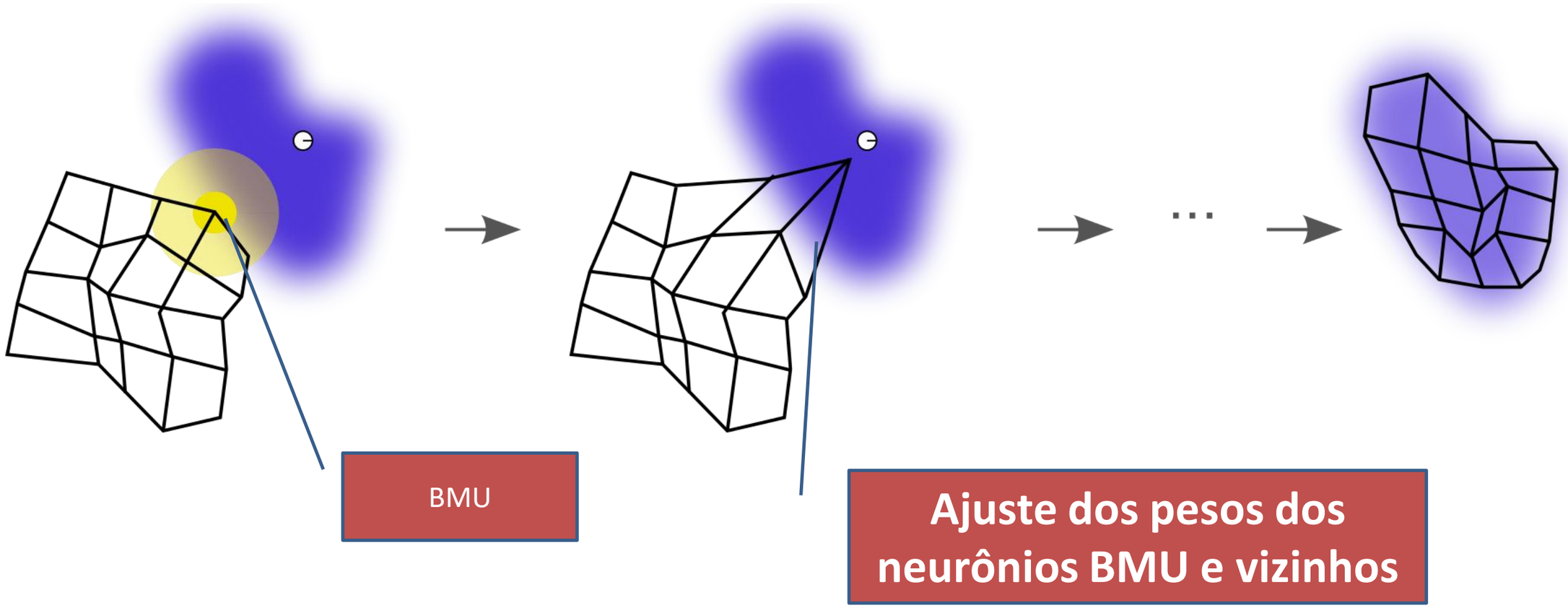
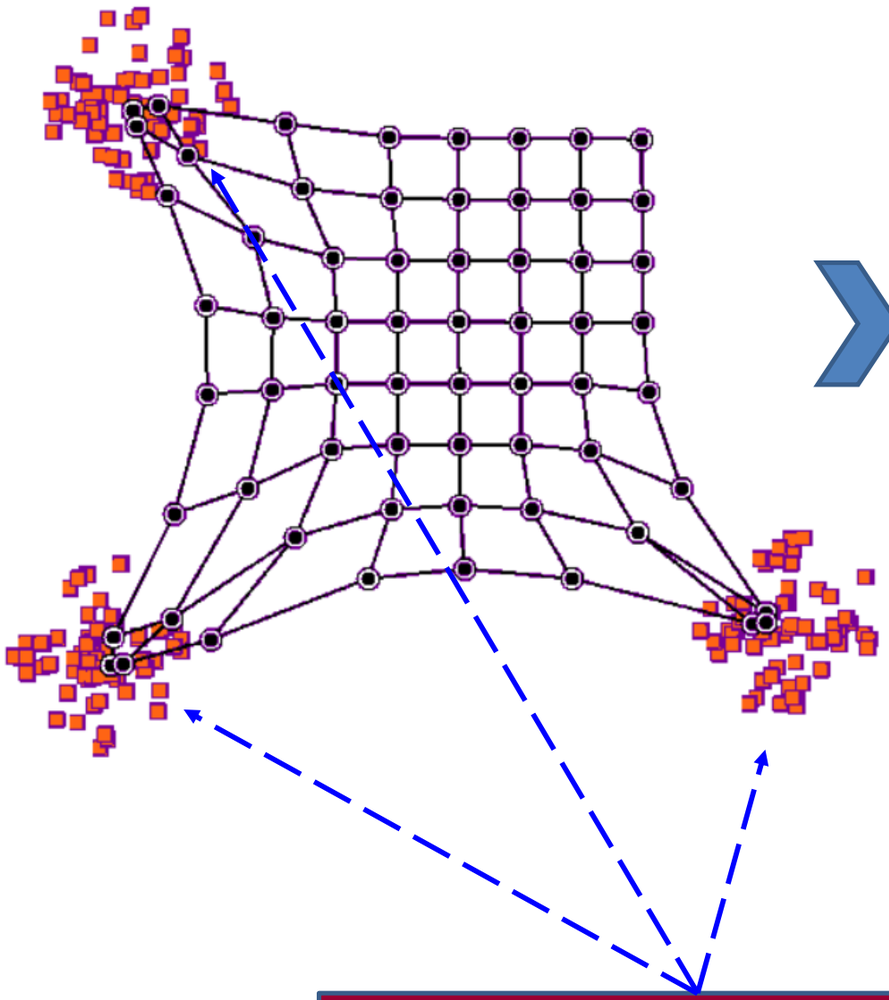


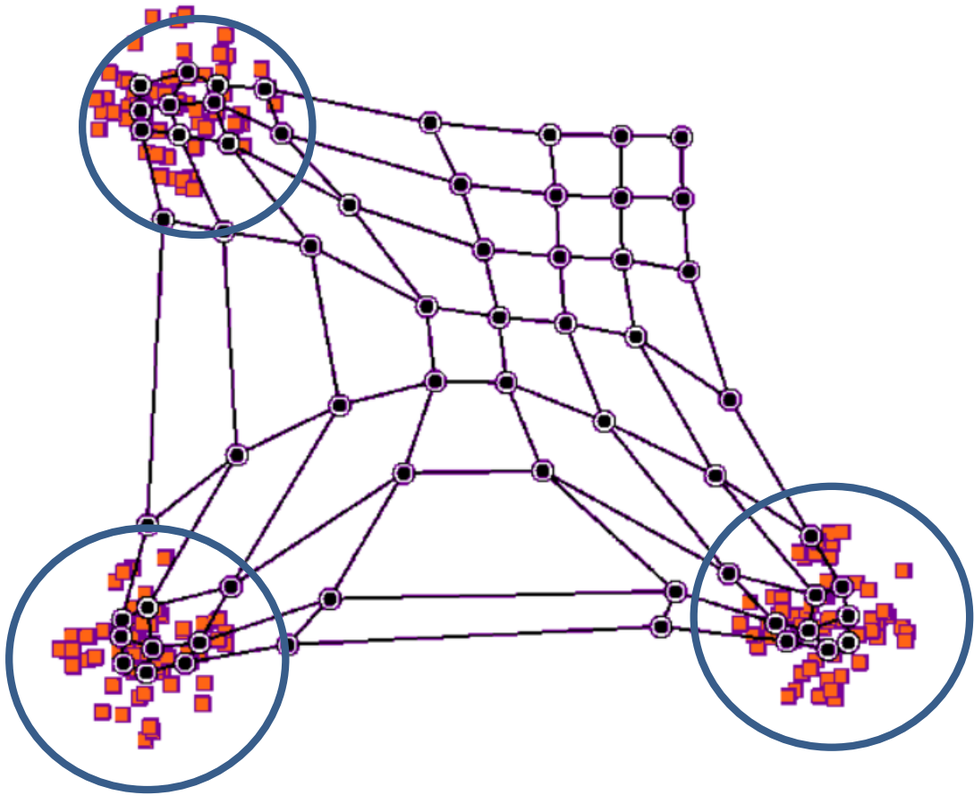
Imagem do Wikipedia

By Mcld - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10373592

Wiki By Chompinha - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=77822988



Objetos similares na nuvem



Neurônios próximos no mapa são atraídos por pontos similares da nuvem

Wiki By Chompinha - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=77822988

TrainSOM.gif

Etapas do SOM – sumário

- Inicialização:
  - Vetores $w_k=(w_{k1},w_{k2}, \ldots, w_{kp})$ com “pesos” com pequenos valores são atribuídos a cada um dos neurônios
    - Diferentes critérios de atribuição desses pesos.
    - Em geral, atribuição aleatória
  - Definir topologia da rede e parâmetros.
    - Hexagonal preferida por ter mais “lados” (contatos)
    - Número de neurônios M
    - Taxas de aprendizagem e decaimento (veremos adiante)
  - Dados de entrada (nuvem de pontos) padronizados entre 0 e 1
    - Só se aplica com dados quantitativos (cálculo de distâncias)
    - Qualitativos → transformar em 0 1

Formalização

- **Inicialização**
- **Repetição**
  - Seleciona-se aleatoriamente um ponto x da nuvem → $\mathbf{x_i}$
  - Calcula-se a distância de cada neurônio a $\mathbf{x_i}$
  - O neurônio **c** mais próximo de $\mathbf{x_i}$ é o "vencedor" → **c= BMU** (*best matching unit*)
  - Os pesos w_{kj} , k=1,..,N , do BMU e seus vizinhos são ajustados para **aproximá-los mais** ainda de $\mathbf{x_i}$
  - Processo segue selecionando um novo ponto da nuvem, determinando novo BMU, etc...até esgotar todos os pontos da nuvem ("*epoch*")
- **Parada**
  - O processo de ajustes repete-se por várias *epochs*, com ajustes cada vez mais finos (redução da vizinhança e passos) até atingir critério de parada **.**

Ajuste dos pesos

- **Ajuste dos pesos w do BMU :**

$$w_{cj}(t+1)=w_{cj}(t)+\eta(t)\left[x_{ij}-w_{cj}(t)\right] \qquad J=1,\ldots p$$

  - $\eta(t)$ define o tamanho do ajuste (tamanho do passo)
  - Quanto maior t, menor $\eta(t)$, menor o ajuste

- **Ajuste dos pesos dos vizinhos k de BMU**

$$w_{kj}(t+1)=w_{kj}(t)+\eta(t)T(k,BMU)\left[x_{ij}-w_{kj}(t)\right]$$

Função vizinhança

Por que neurônio **BMU** se aproxima de **x**?

Por exemplo, seja x = (0.2 , 0.9) , w_c=(.3, .8) e $\eta(t) = .1$

- $w_{cj}(t+1)=w_{cj}(t)+ \eta(t)\left[x_{ij}-w_{cj}(t)\right]$
- $w_{c1}(t+1)=w_{c1}(t)+ \eta(t)\left[x_{i1}-w_{c1}(t)\right] = 0.3 + 0.1\ (.2 - .3) = 0.29$
- $w_{c2}(t+1)=w_{c2}(t)+ \eta(t)\left[x_{i1}-w_{c2}(t)\right]= 0.8 + 0.1\ (.9 - .8)= 0.81$

- note que:
  - **$D^2=(0.2 -0.3)^2 + (0.9 - 0.8)^2 =0.020$ → $D^2=(0.2 -0.29)^2 + (0.9 - 0.81)^2 =0.0162$**
  - (x – w) < 0 ➔ w decresce & (x – w) > 0 ➔ w cresce
  - Nesse sentido diremos que w é atraído por x
  - O mesmo vale para os neurônios vizinhos

Definições

$\boldsymbol{\eta(0)}$: valor inicial da **taxa de aprendizado**

τ: controle da redução da taxa de aprendizado → $\boldsymbol{\eta(t) = \eta(0)\exp(\frac{-t}{\tau})}$

$0 < \eta(0) < 1$ e $\tau > 0$

- **Define o tamanho do ajuste a cada iteração**
- À medida que t cresce, tende a zero

$\boldsymbol{\sigma(0)}$: para definir tamanho **(raio) da vizinhança** no mapa em t=0

λ: controle da redução do tamanho da vizinhança → $\boldsymbol{\sigma(t) = \sigma(0)exp(\frac{-t}{\lambda})}$

$\boldsymbol{\lambda > 0}$

- **Define o tamanho da vizinhança a cada iteração**
- À medida que t cresce, tende a zero

Definições

$T(k, BMU)$: função vizinhança em torno de um neurônio (*topological distance*)

$$T(k, BMU) = \exp\left(-\frac{d^2}{2\sigma(t)^2}\right)$$

onde

- **d** é a distância entre o BMU e seu vizinho k na iteração t, medida no reticulado

- Valor de $T(k, BMU)$ diminui a cada iteração

- À medida que processo avança os ajustes são mais localizados (neurônios mais próximos)

$$w_{kj}(t+1) = w_{kj}(t) + \eta(t)T(k,BMU)\left[x_{ij} - w_{kj}(t)\right]$$

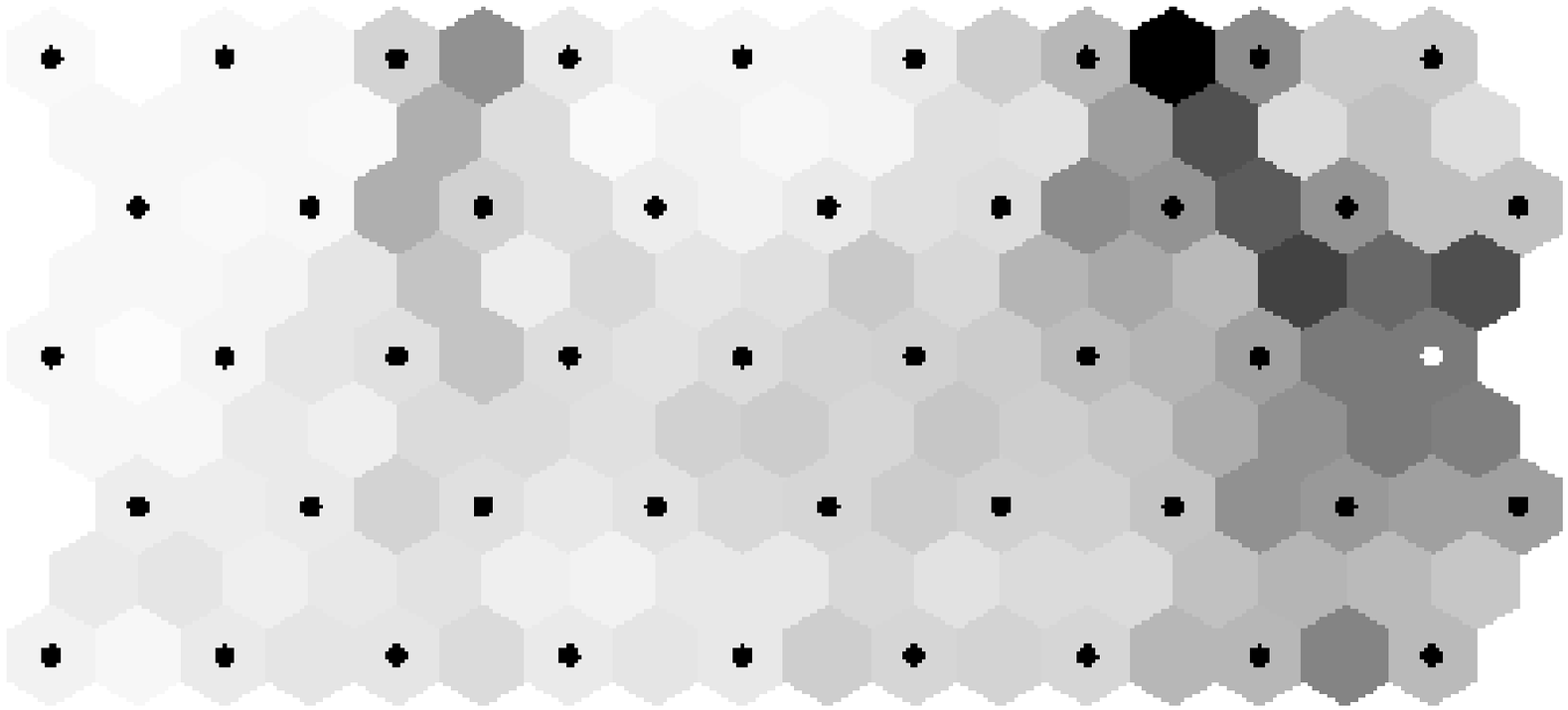
Quantos neurônios M deve ter o mapa?

- Nao há uma resposta definitiva para isso
  - **Em geral, determinação por tentativa e erro**
- Alguns autores recomendam
  - **Alternativa 1:** $\boldsymbol{M = 5\sqrt{N}}$ onde N=número de pontos na nuvem p-dimensional
    - Dimensões no eixo x→ $\sqrt{M}$ no eixo y → $M/\sqrt{M}$
    - Ou, relação entre base e altura do reticulado deve ser a mesma que a relação entre os dois maiores autovalores da matriz de covariância dos dados de entrada
  - **Alternativa 2:** $\boldsymbol{M = 5\sqrt{N}\,\frac{\lambda_1}{\lambda_2}}$ onde $\lambda 1\ e\ \lambda 2$ são os dois maiores autovalores da matriz de covariâncias dos inputs
- Artigo seguinte (ótimo) recomenda que a cada neurônio corresponda a pelo menos 10 inputs
  - https://pt.slideshare.net/shanelynn/2014-0117-dublin-r-selforganising-maps-for-customer-segmentation-shane-lynn

N=5000 → M~ 5 x 70=350
Eixo x → sqrt(350)≈20
Eixo y → 350/20 ≈ 18

U-matrix gray scale (melhor que do R)

- A dark coloring between the neurons corresponds to a large distance and thus a gap between the codebook values in the input space.
- A light coloring between the neurons signifies that the codebook vectors are close to each other in the input space.
- Light areas can be thought as clusters and dark areas as cluster separators.



Heatmap das variáveis inputs

- Podemos analisar a distribuição de cada variável com heatmaps

- Descrição copiada de

https://en.proft.me/2016/11/29/modeling-self-organising-maps-r/

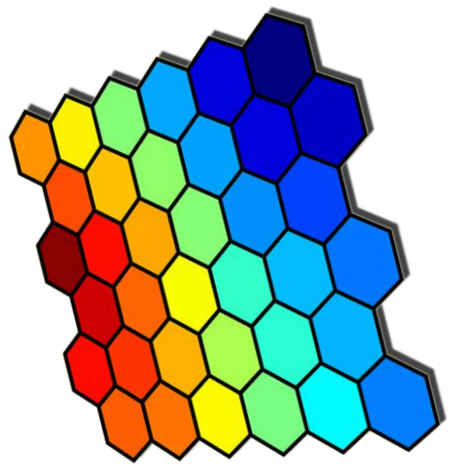
  - *Imagine uma sala cheia de gente . É o SOM*

  - *Vamos ver a sala de cima*

  - *Cada pessoa tem um chapéu cuja cor representa a idade*

  - *Pessoas com idades semelhantes idealmente estarão próximas*

  - *Isto é um heatmap*



Em resumo....

- PRÓS
- Permite a visualização e interpretação de uma massa de dados em espaços de menor dimensão
- Permite agrupar um grande número de observações
- Algoritmo em geral converge rapidamente

- CONS
- Resultado será interessante se massa de dados apresentar uma estrutura que permita agrupá-los
- Sensível á presença de outliers. Dados devem ser preparados cuidadosamente
- Em geral, clusters são neurônios próximos no mapa. Pode ocorrer que grupos similares de observações apareçam em áreas diferentes do mapa

Exemplo de aplicação

- Arquivo indicadores demográficos

UF	População	TBN	TBM	TMI	TFT	RDT	IV	EVN	IDH
AC	894.470	18,98	4,74	14,81	2,22	51,76	18,40	75,09	0,663
AL	3.351.092	15,23	6,94	15,56	1,74	46,08	33,33	72,98	0,631
AM	4.207.714	19,50	5,39	16,26	2,14	51,04	17,36	72,81	0,674
AP	861.773	18,73	4,40	22,41	2,01	48,14	15,66	74,88	0,708
.....									

TBN = Taxa Bruta de natalidade por mil hab

TBM = Taxa Bruta de Mortalidade por mil hab

TMI = Taxa de Mortalidade Infantil

TFT = Taxa de Fecundidade Total

RDT = Razão de dependência total

- razão entre a pop economicamente dependente e a economicamente ativa

IV = Índice de Envelhecimento

EVN = Esperaça de Vida ao Nascer

IDH = Índice de Desenvolvimento Humano

Fonte: IBGE/Diretoria de Pesquisas. Coordenação de População e Indicadores Sociais. Gerência de Estudos e Análises da Dinâmica Demográfica. **Projeção da população do Brasil e Unidades da Federação por sexo e idade para o período 2010-2060**

Aplicação

- Vide script : **SOM_INDICADORES**

- Outros scripts para estudo

  - **SOM_MOBILE**

  - **SOM_SAUDE**

  - **SOM_USA**

Gráfico 1: Training Progress

- As the SOMs training iterations progress, the distance from each node's weights to the samples represented by that node is reduced. Ideally, this distance should reach a minimum plateau.
- If the curve is continually decreasing, more iterations are required.

```
> plot(ind.som, type="changes")
```

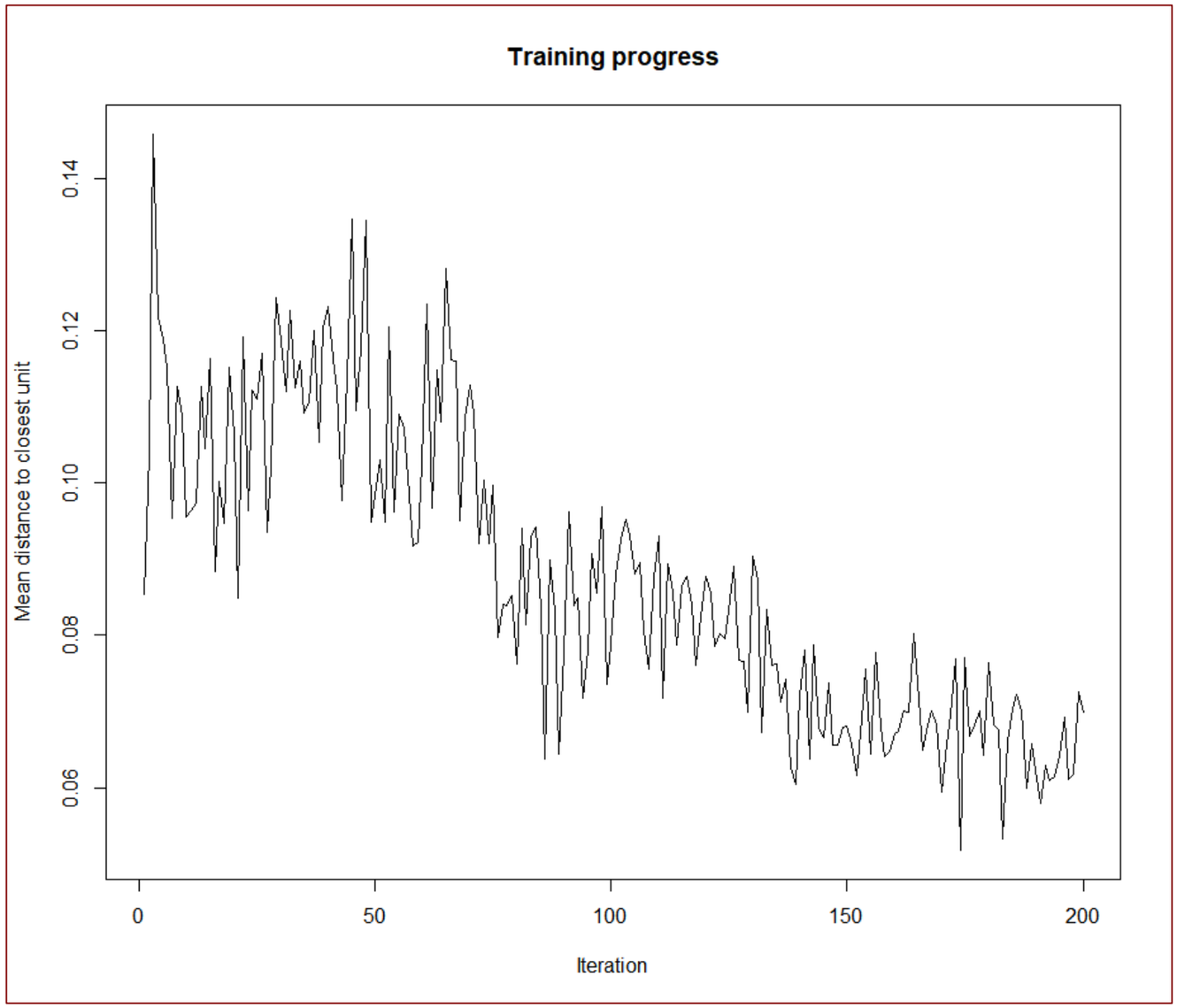


Gráfico 2: Node Counts

- The Kohonen packages allows us to visualise the count of how many samples are mapped to each node on the map. Ideally the sample distribution is relatively uniform.
- Large values in some map areas suggests that a larger map would be beneficial. Empty nodes indicate that your map size is too big for the number of samples. Aim for at least 5-10 samples per node when choosing map size.

```
> plot(ind.som, type="count", shape = "straight")
```

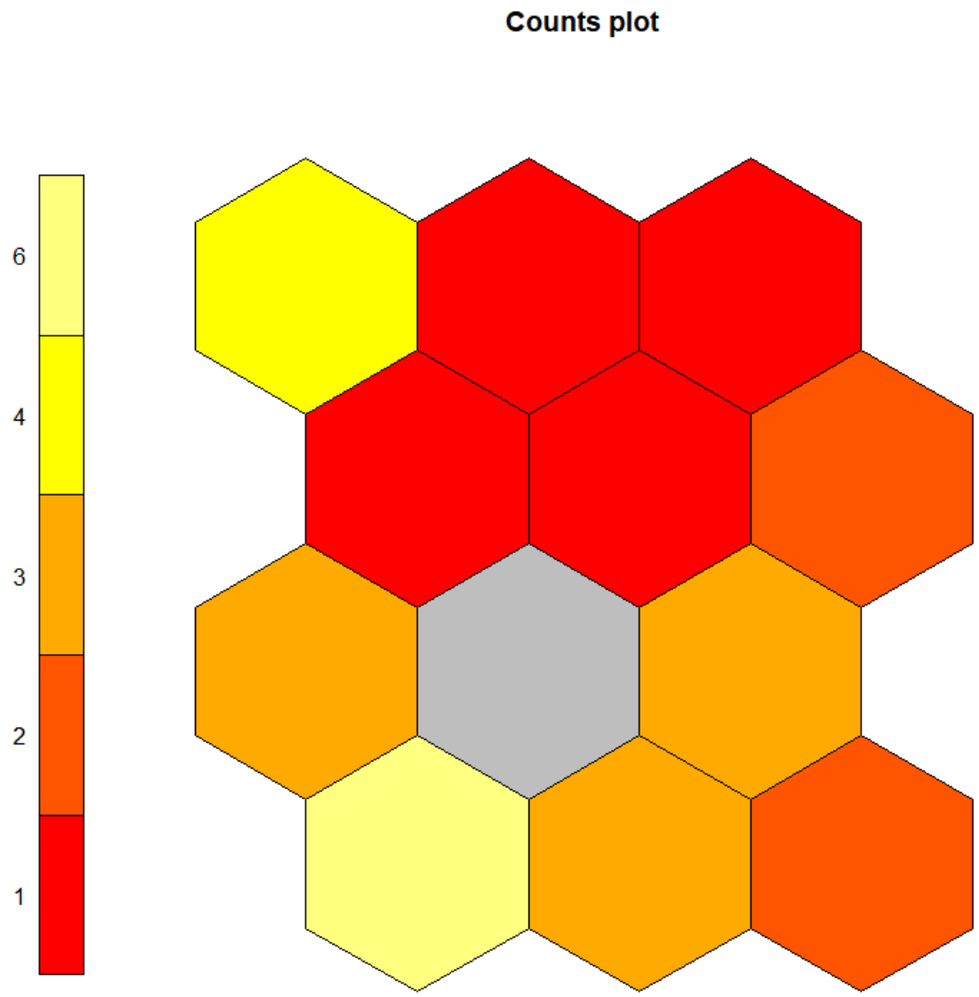


Gráfico 3: Mapping

- "mapping": shows where objects are mapped.

```
> plot(ind.som, type="mapping",col=1,labels=ind$UF, cex=.9,  bgcol =‘ yellow’, shape = "straight" )
```

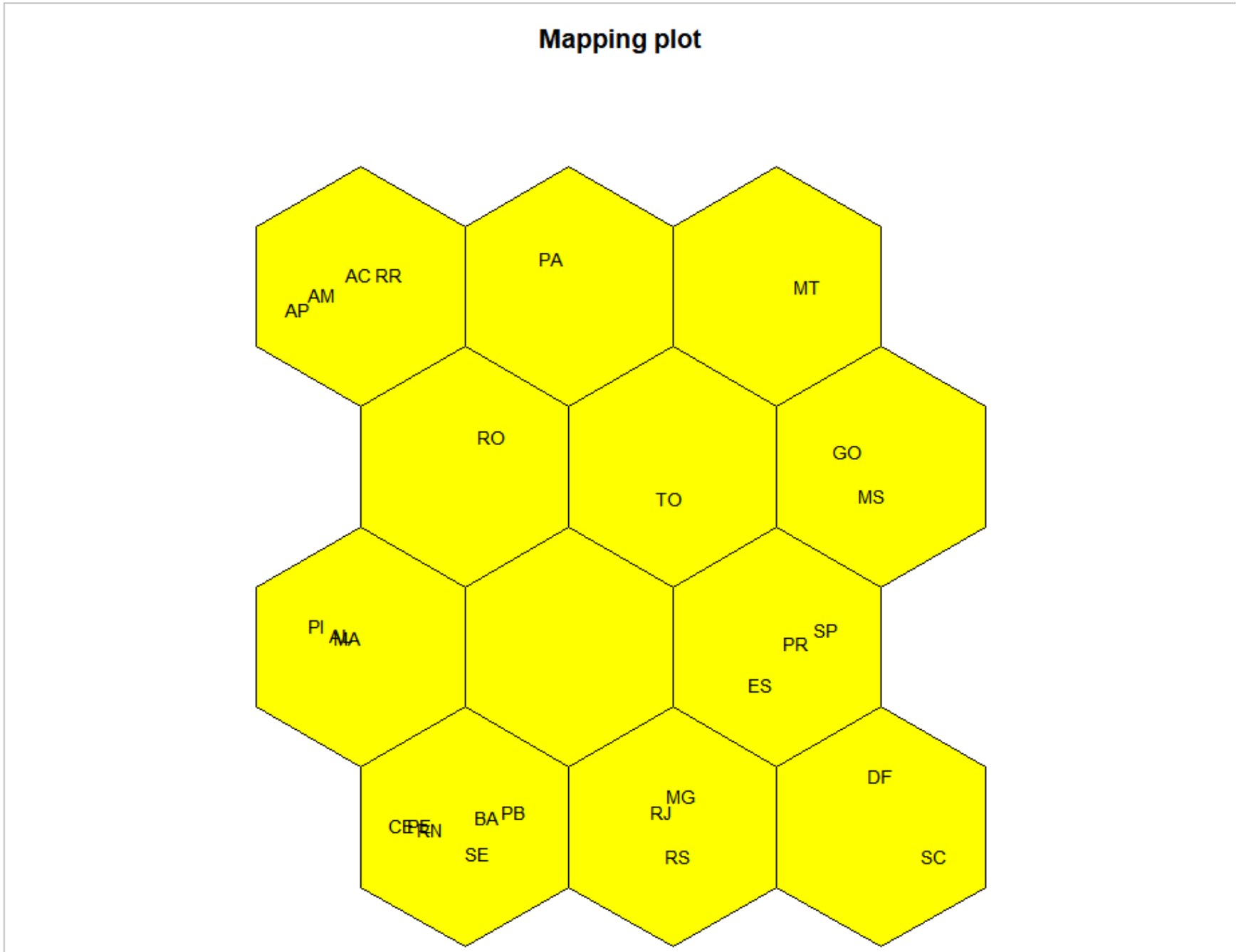
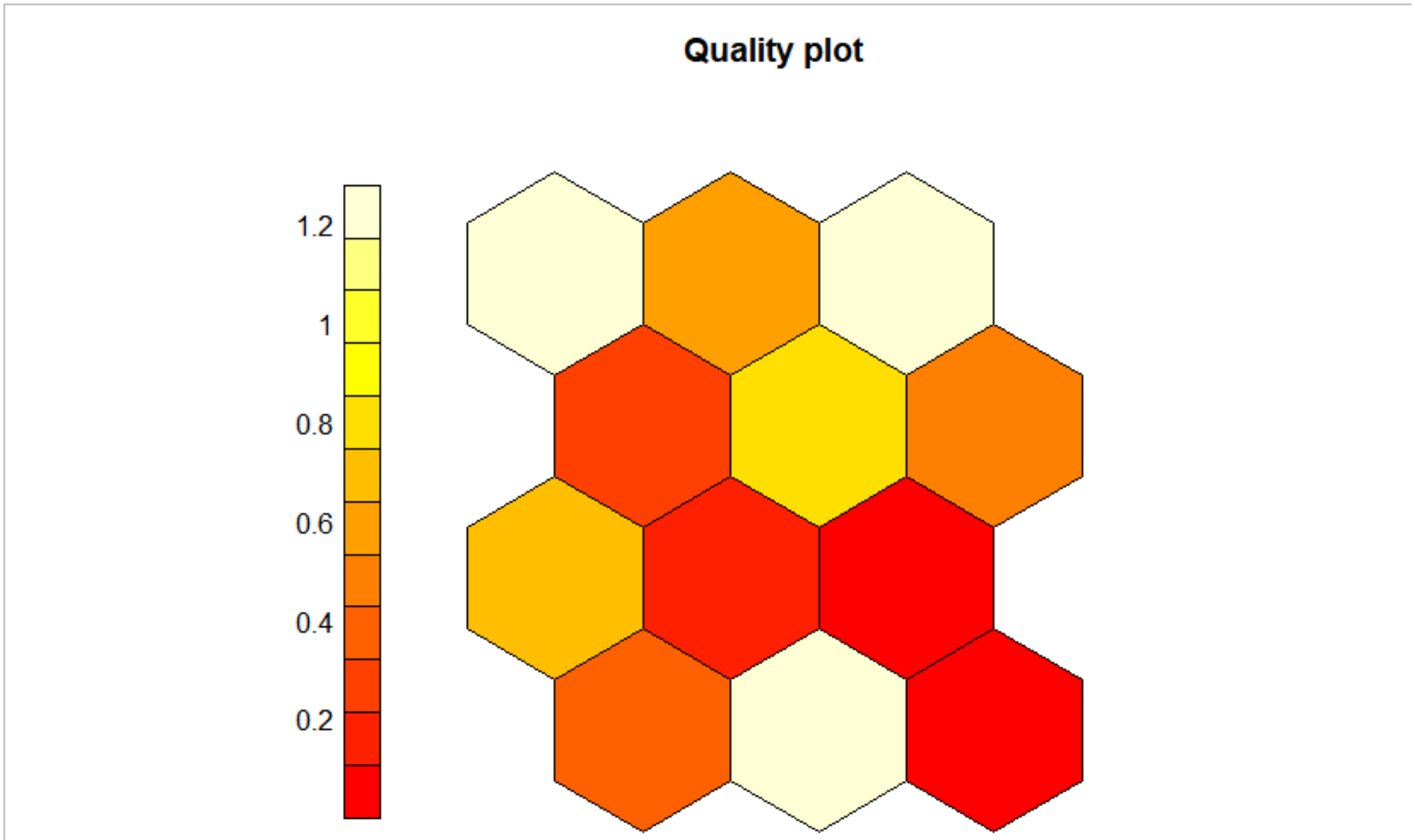
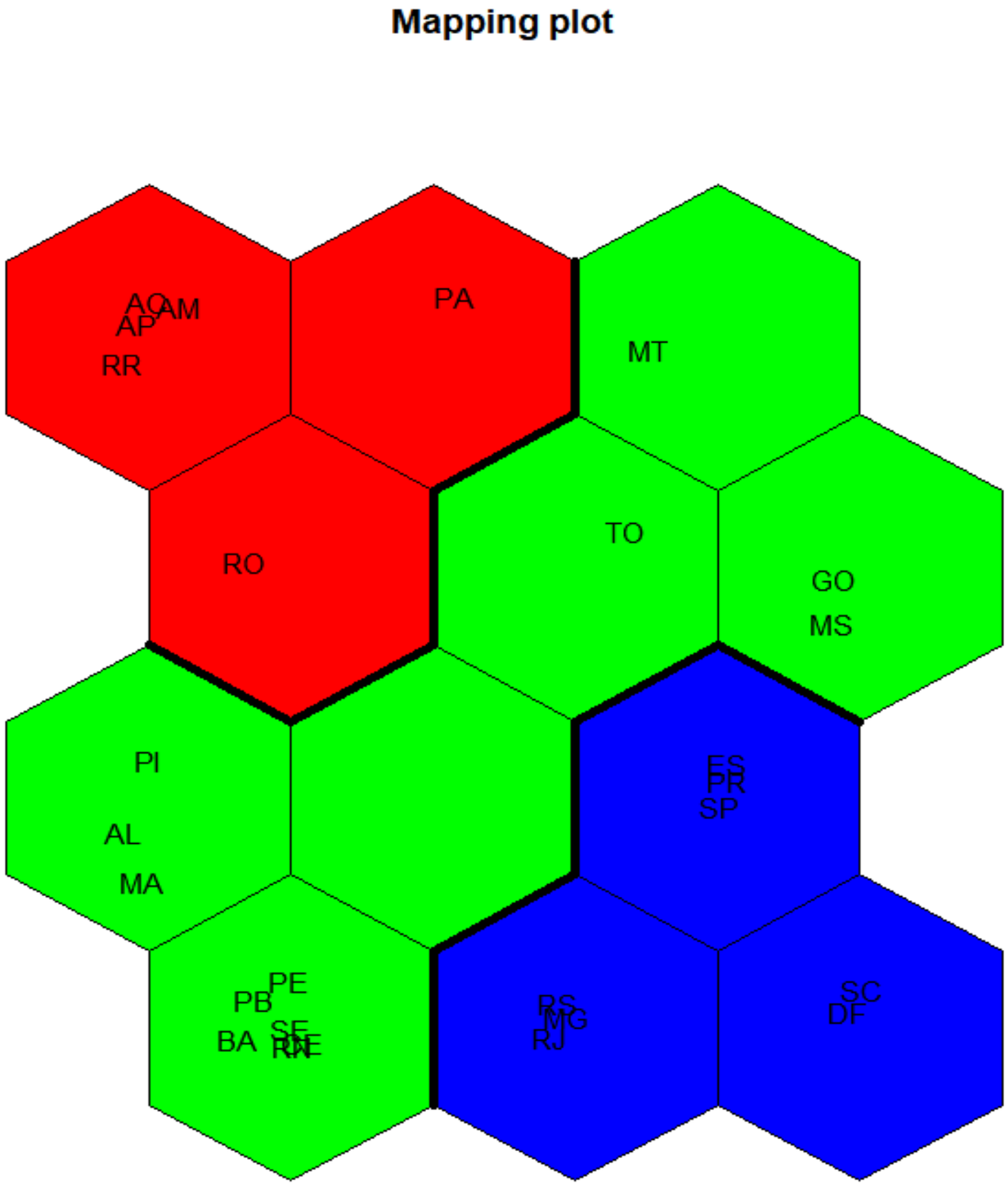


Gráfico 5: Quality

- "quality" shows the mean distance of objects mapped to a unit to the codebook vector of that unit.
- The smaller the distances, the better the objects are represented by the codebook vectors.

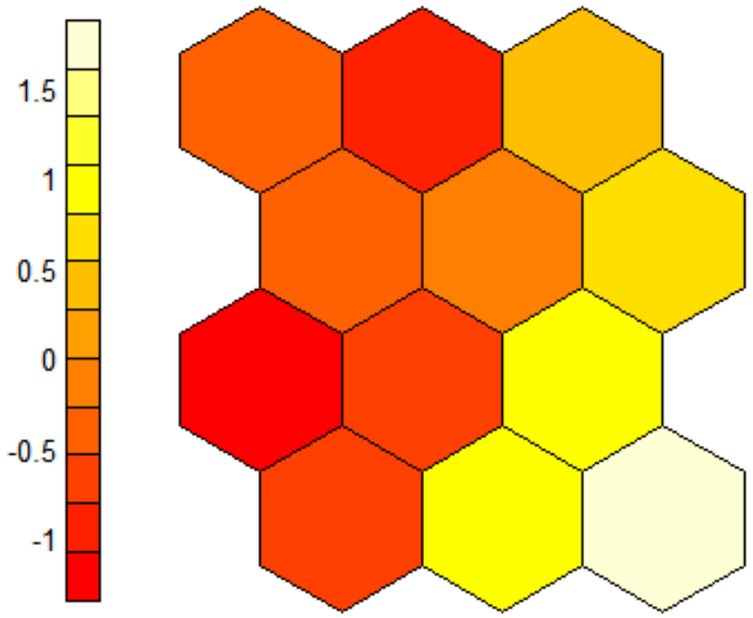


Clusters (agrupando os code vectors)

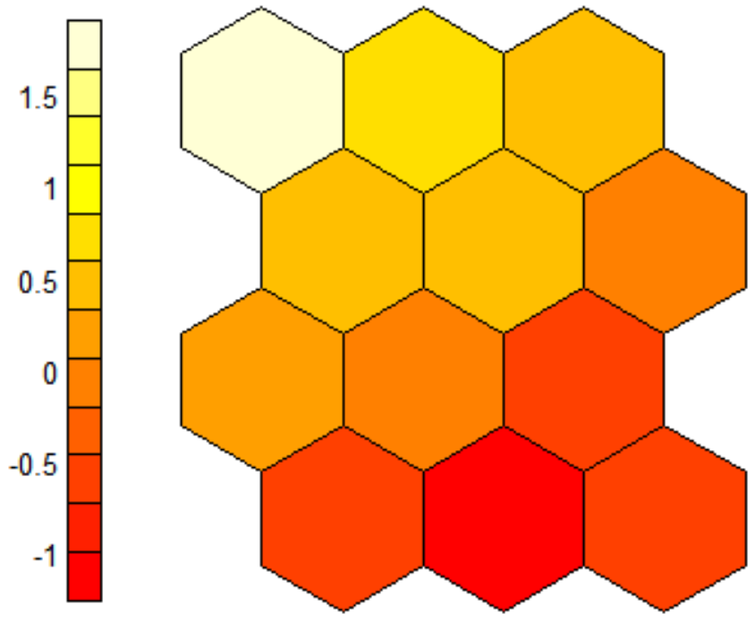


Property plots

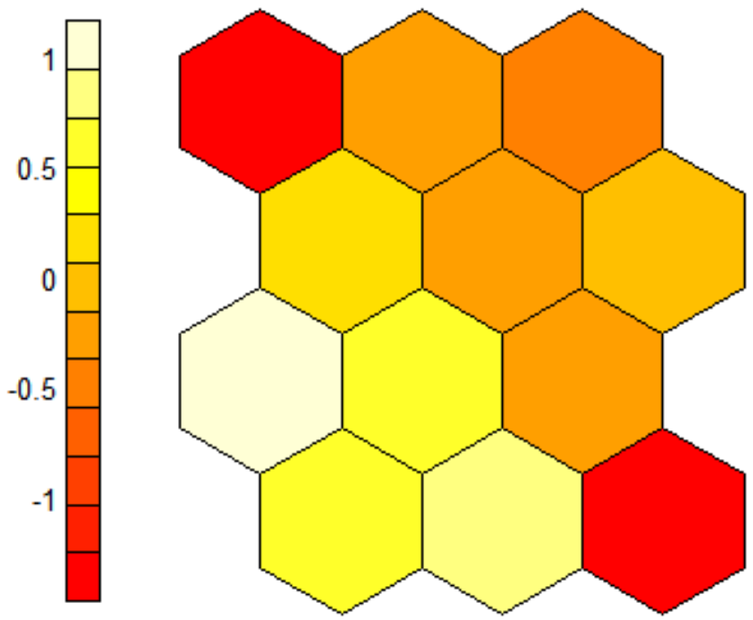
IDH



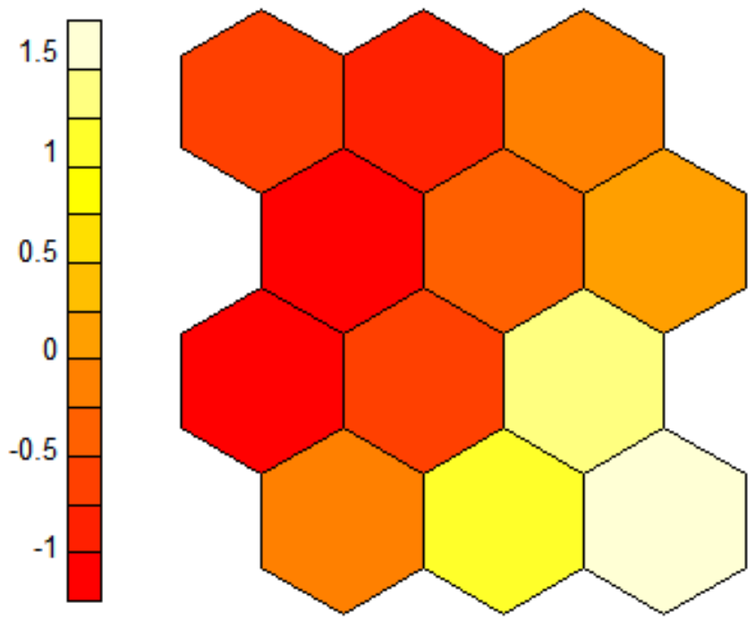
TBN



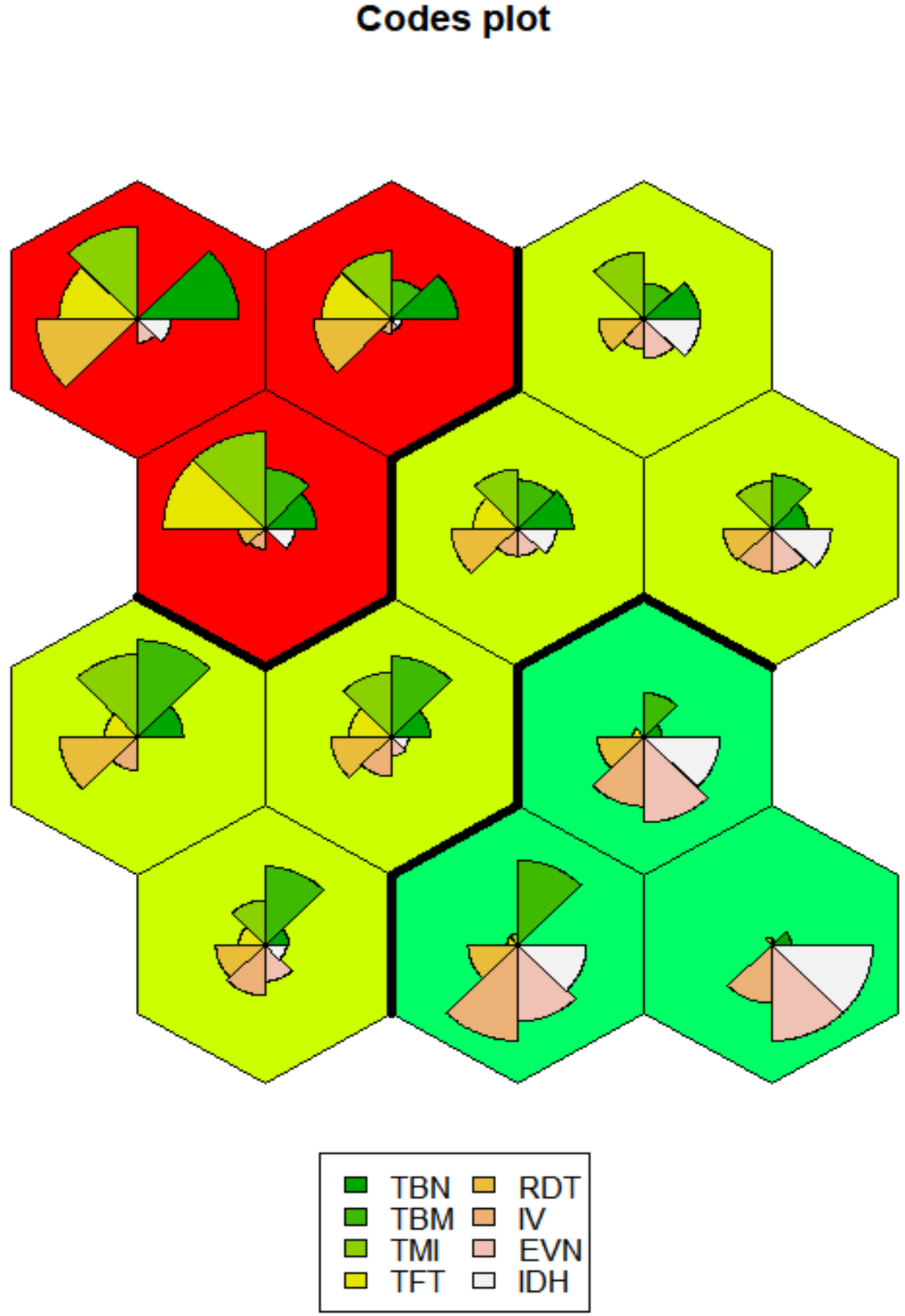
TBM



EVN



Codes plot



Codebook (tabela de valores dos neurônios)

FGV | EDUCAÇÃO EXECUTIVA

```
> codebk=ind.som$codes[[1]]
> codebk=as.data.frame(codebk)
> round(codebk,3)
```

	TBN	TBM	TMI	TFT	RDT	IV	EVN	IDH
v1	-0.494	0.667	-0.129	-0.202	-0.235	0.218	-0.166	-0.654
v2	-1.250	0.838	-1.064	-0.658	-0.203	1.780	1.045	0.831
v3	-0.636	-1.342	-1.172	-1.008	-1.723	0.484	1.640	1.877
v4	0.157	1.159	0.945	0.031	0.667	-0.350	-1.271	-1.341
v5	-0.038	0.714	0.428	0.311	0.116	-0.145	-0.691	-0.783
v6	-0.723	-0.288	-1.391	-0.637	-0.280	0.881	1.321	1.041
v7	0.364	0.156	1.345	2.158	-0.931	-0.807	-1.281	-0.382
v8	0.469	-0.158	0.279	0.389	0.308	-0.564	-0.439	-0.130
v9	-0.114	-0.007	-0.060	-0.980	-0.251	-0.032	0.074	0.557
v10	1.907	-1.472	1.273	1.466	1.366	-1.516	-0.591	-0.297
v11	0.816	-0.373	0.552	1.176	0.627	-1.044	-0.968	-1.023
v12	0.479	-0.473	0.540	-0.847	-0.316	-0.498	-0.107	0.410

“Clusterização” e previsão

- Rodamos o algoritmo de cluster com os dados do codebook
  - Dados já estão padronizados
  - Agrupamento pode diferir do obtido como a nuvem de pontos

- Previsão de out-of-sample
  - Vamos inventar alguém parecido com AC para poder verificar

```
> novo=data.frame(TBN=19, TBM=5,  TMI=15, TFT=2,
+                 RDT=52, EVN=75, IDH=0.66)
> x=as.matrix(novo) #tem que ser matrix
> gg=predict(ind.som, newdata = x)
> gg$unit.classif
```

https://www.shanelynn.ie/self-organising-maps-for-customer-segmentation-using-r/

- …*Use the statistics and distributions of the training variables within each cluster to build a meaningful picture of the cluster characteristics – an activity that is part art, part science!* ***The clustering and visualisation procedure is typically an iterative process. Several SOMs are normally built before a suitable map is created****. It is noteworthy that the majority of time used during the SOM development exercise will be in the visualisation of heatmaps and the determination of a good "story" that best explains the data variations*.

importante

Em resumo

- **Fonte:** https://en.proft.me/2016/11/29/modeling-self-organising-maps-r/
- Instead of directly manipulating the data, a SOM adapts to the data, or, using the standard terminology, learns from the data. This computational model offers a certain flexibility, which can be used to develop fast and robust data processing algorithms.
- The SOMs was designed for unsupervised learning problems such as feature extraction, visualization and clustering. Some extensions of the approach can label the prepared codebook vectors which can be used for classification.
- Also, the SOMs has the capability to generalize. Generalization capability means that the network can recognize or characterize inputs it has never encountered before. A new input is assimilated with the map unit it is mapped to.

SOMs vs k-means

- **Fonte:** https://en.proft.me/2016/11/29/modeling-self-organising-maps-r/
- **SOMs vs k-means**
- It must be noted that SOM and k-means algorithms are rigorously identical when the radius of the neighborhood function in the SOM equals zero (Bodt, Verleysen et al. 1997).
- In a sense, SOMs can be thought of as a spatially constrained form of k-means clustering (Ripley 1996).
- SOMs is less prone to local optima than k-means. During tests it is quite evident that the search space is better explored by SOMs.
- If you are interested in clustering/quantization then *k-means* obviously is not the best choice, because it is sensitive to initialization. SOMs provide a more robust learning. *k-means* is more sensitive to the noise present in the dataset compared to SOMs.

Validation

- Fonte: ????
- We can create models as we like, but before a model can be reliably used, it must be validated. Validation means that the model is tested so that we can be sure that the model gives us reasonable and accurate values. What we mean by this depends largely on the application and our requirements.

- The validation must be done by using an independent test set. The independent test set is a set similar to the input set but not a part of the training set. The testing set can be seen as a representative of the general case.

- Quantization error of an input vector is defined as the difference between the input vector and the closest codebook vector. For a set of input vectors, we can reflect on the similarity of the input data set and the SOM by investigating the distribution of the quantization errors. The range of quantization error tells the smallest and the largest amount of error.