An introduction to YASOMI (Yet Another Self Organising Map Implementation)

Fabrice Rossi

April 13, 2012

The yasomi package aims at providing a complete implementation of Self Organising Maps (SOMs) adapted to both standard numerical data and to more complex data described via a dissimilarity matrix or a kernel matrix. Yasomi tries to include a broad selection of SOM based state of the art visualisation methods. It also provides automated data driven construction methods for SOMs.

The following example provides a demonstration of the main features of yasomi. The SOM algorithm is based on the Euclidean norm and assumes therefore isotropy in the data space. As a consequence, it is advisable to scale the data under analysis, unless some prior knowledge suggests to emphasise one or several variables.

```
> library(yasomi)
> data <- scale(iris[1:4])
```

This example uses the Iris dataset for which a SOM is constructed using only the numerical attributes of the plants.

The SOM algorithm fits a grid of prototypes to the data. The grid topology (prior arrangement of the prototypes) is under user control. In this example a rather standard hexagonal based grid is chosen.

```
> sg <- somgrid(xdim=10,ydim=10,topo="hexagonal")
```

This grid can be plotted (see Figure 1), even if this representation alone is not very useful. The main advantage the grid is to provide a support for visualisation: each cell in Figure 1 is associated to a prototype and close cells correspond to close prototypes. By filling the cells with colors and/or glyphs computed from the data, some insights on the structure of the dataset can be obtained, as shown in the following figures.

The SOM tuning process can be controlled via some parameters. In this example, one of the parameter (an initial radius which specifies the influence of the prototypes on each other) is automatically chosen by minimising a distortion measure.

```
> somtuning <- som.tune(data,sg,
+                       som.tunecontrol(sg,radii=c(2,sg$diam),nradii=20,
+                                       criterion=error.kaskilagus))
> som <- somtuning$best.som
```

```
> plot(sg,asp=1)
```

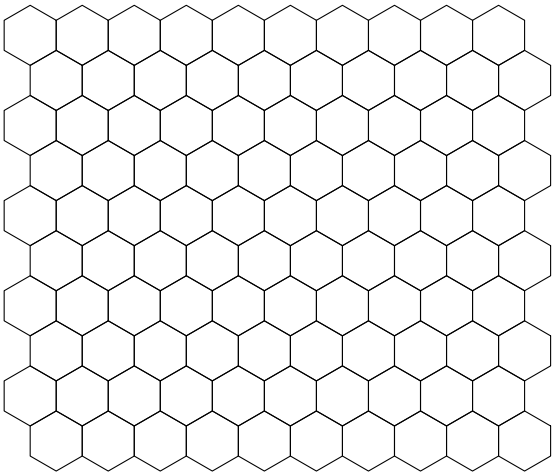

Figure 1: The prior structure imposed on the prototypes

The quality of the SOM as measured by the chosen criterion (Kaski and Lagus' distorsion measure) depends on the initial value of the influence radius, as shown on Figure 2 (lower values correspond to higher quality). The best SOM according to the chosen quality measure is kept (in general, the quantisation error of a SOM that represents correctly the topology of the data will be higher than the one of a SOM with less topology preservation). The evolution of its quantisation error during the fitting process is depicted on Figure 3. Yasomi includes numerous visualisation methods that can be used to display the fitted SOM. The simplest method consists in displaying the prototypes arranged according to the prior structure (a.k.a., the grid). As the prototypes are generally high dimensional vectors, they are displayed using star glyphs (as shown on Figure 4), parallel coordinates or barplots. The figure displays an example of the well known ordering property of the SOM algorithm. The surface of the glyph increase from left to right and from top to bottom. There are clearly two classes of glyphs (elongated thin ones on the left and more symetric ones on the right).

Star glyphs are not always easy to read, especially when dealing with high dimensional data. In some situations, a color coded display of a selection of the variables might provide more insight on the data. This can be obtained via component planes for a SOM, as shown on Figure 5. Each of the subfigure displays the values taken by one of the variable over the prototypes of the SOM. Petal variables (lower row) show very strong correlation as well as a clear separation between two classes (consistent with the one observed in Figure 4). The sepal width seams also somewhat negatively correlated with the petal variables.

While the SOM algorithm induces a clustering over the data, some prototypes might end up with an associated empty cluster. Displaying the size of each cluster gives sometimes an idea of the topology of the data set by empha-

```
> plot(somtuning,relative=FALSE)
```

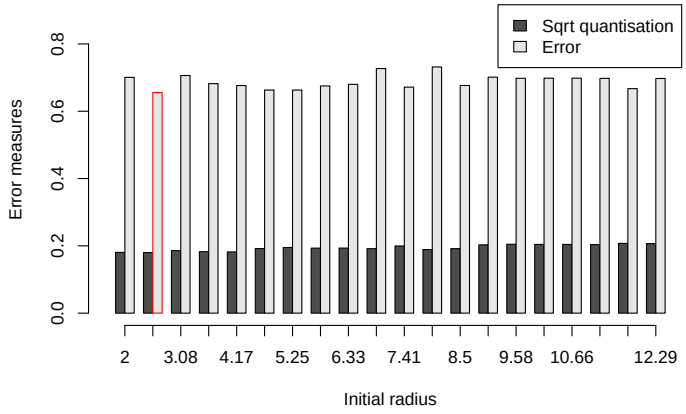

Figure 2: Dependency between the distorsion of the final map and the initial influence radius

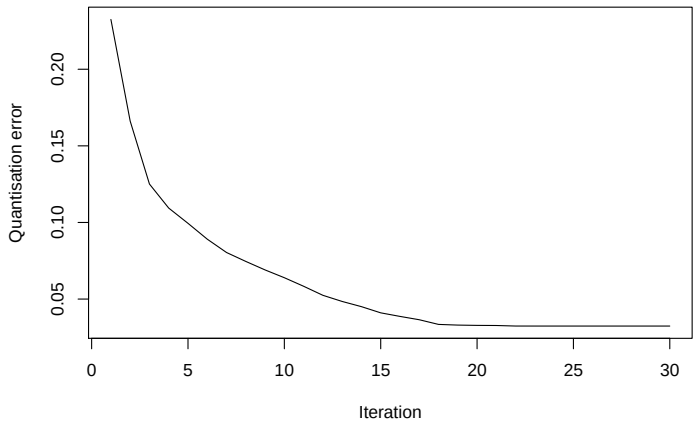

Figure 3: Evolution of the quantisation error during the fitting process

```
> plot(som,type="stars",asp=1)
```

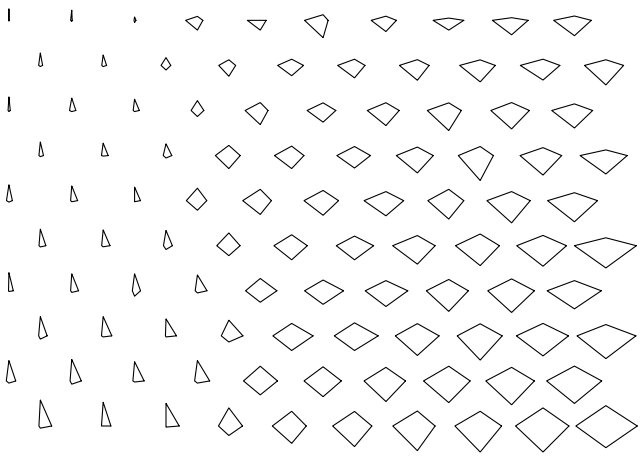

Figure 4: Star glyphs of the prototypes of the fitted SOM

sising dense or empty area. Figure 6 gives an example of such a display. While two classes were quite obvious in the previous figures, the picture is not so clear here. It seems that the left class might be quite dense and separated from the right one by many empty cells (prototypes with empty cluster) but the right class seem also to have a substructure that what not clear on the other figures. Another exploration method for a fitted SOM consists in displaying all the original observations that have been assigned to a cluster in a superposed way, as shown in Figure 7: we used here parallel coordinates which are generally easier to read that superposed star glyphs. Figure 7 confirms that some structure can be found in the two large clusters. It also shows that the clustering done in each grid cell is very homogeneous and therefore that the simplification done by the SOM does not impair the understanding of the data.

Another common practice for SOM based cluster analysis is to apply a hierarchical clustering to the prototypes of a fitted SOM. Yasomi provides the distances between the prototypes to ease this approach:

```
> hc.som <- hclust(as.dist(som),method="ward")
```

As shown on Figure 8, there are obviously two clusters of prototypes with some support for up to five clusters, which are identified on the dendrogram and computed below:

```
> hc.som.5 <- cutree(hc.som,k=5)
```

The hit map used in Figure 6 can be colored according to the clusters identified via the hierarchical clustering, as shown on Figure 9.

Other cluster based information can be displayed, for instance the distribution of the values taken by a variable in each of the clusters defined by the SOM. This is especially interesting if the variable was not used to build the SOM. In the present example, the Species variable of the Iris dataset was not

```
> spar <- par(mfrow=c(2,2))
> for(i in 1:ncol(data)) {
+   componentPlane(som,i,asp=1)
+ }
> par(spar)
```

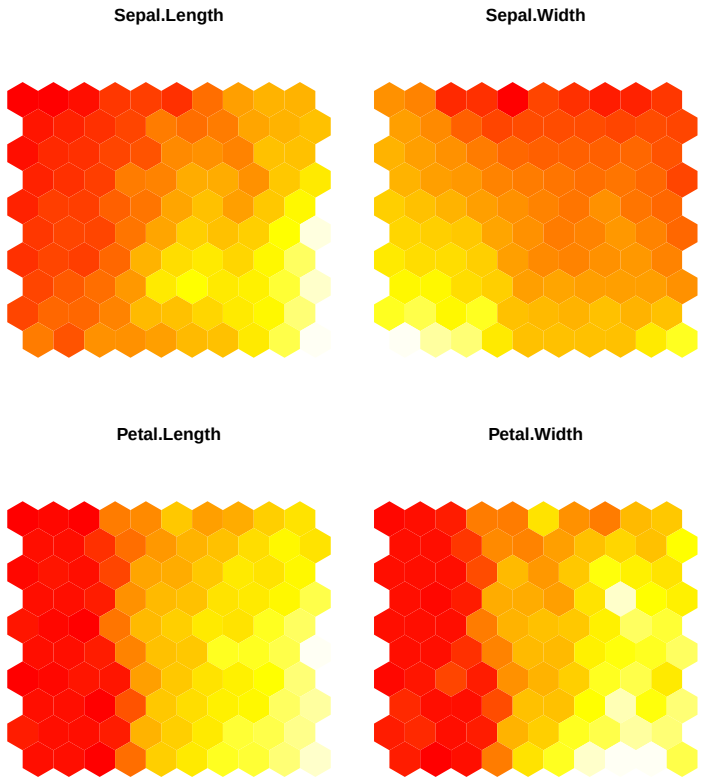

Figure 5: Component planes of the prototypes of the fitted SOM

```
> hitMap(som,col="blue",with.grid=FALSE,asp=1)
```

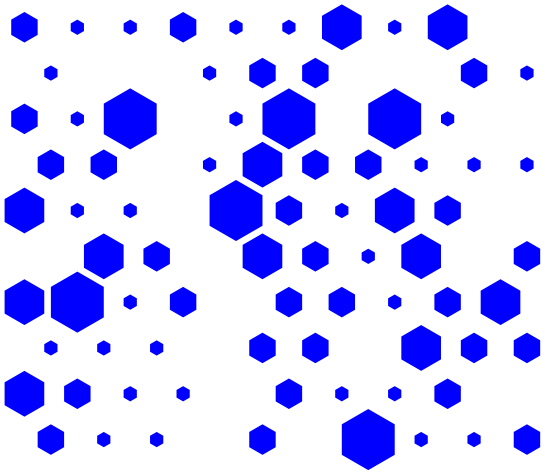

Figure 6: Size of the clusters associated to each prototype (hit map)

```
> plot(som,mode="data",type="parallel",with.grid=TRUE,asp=1)
```

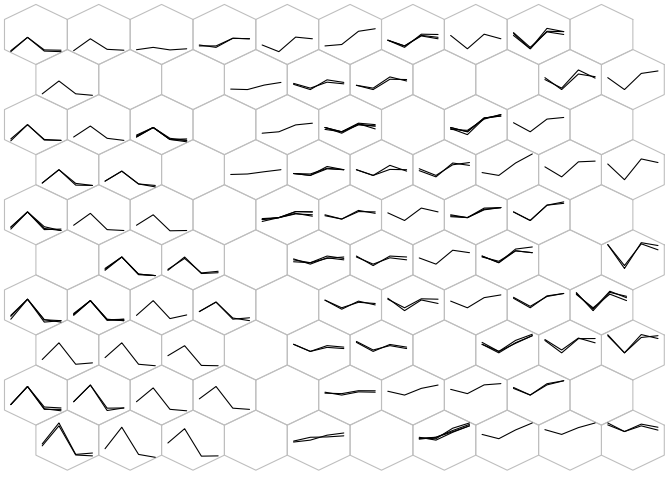

Figure 7: Parallel coordinates for the observations assigned to each cluster

```
> spar <- par(mfrow=c(1,2))
> plot(hc.som,labels=FALSE,hang=-1)
> rect.hclust(hc.som,k=5)
> barplot(rev(hc.som$height)[1:20])
> par(spar)
```

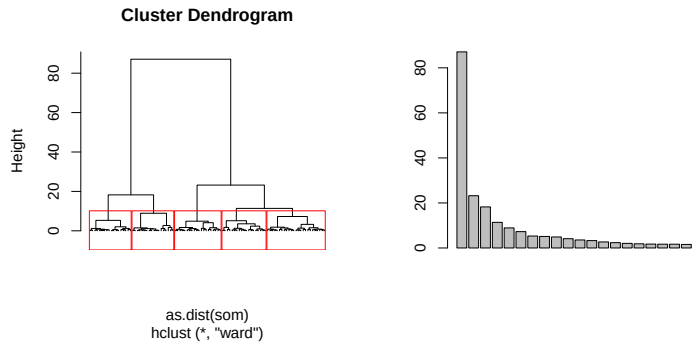

Figure 8: Dendrogram for the prototype clustering on the left, associated heights for the first 20 merges on the right

```
> library(colorspace)
> hitMap(som,with.grid=FALSE,asp=1,col=rainbow_hcl(5,start=30,end=300)[hc.som.5])
```

Figure 9: Size of the clusters associated to each prototype colored via hierarchical clustering of the prototypes

used. When the distribution of this variable in each cluster is concentrated (in the best case, the variable has a constant value in each cluster), the SOM has managed to follow the class distribution using only the numerical characteristics of the flowers. This is roughly the case in the optimal SOM, has shown on Figure 10. One of the cluster identified in the previous analysis corresponds to one of the original class, while the second cluster is the union of the two other classes. Those two classes appear quite well separated except for a few clusters. However, the comparison of Figure 9 and Figure 10 shows that the clustering

```
> plot(som,mapToUnit(som,iris[[5]]),mode="data",type="barplot",asp=1)
```

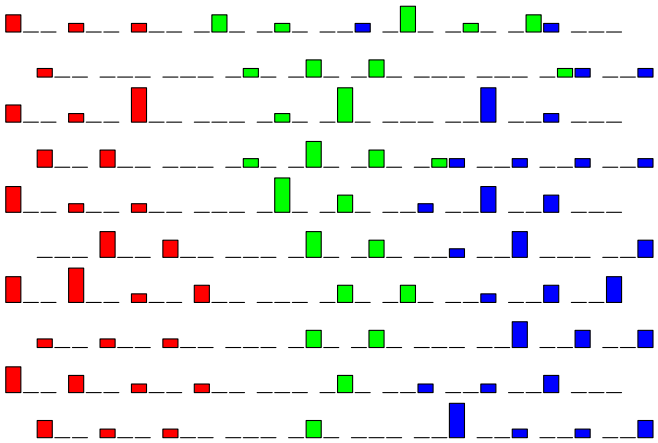

Figure 10: Distribution of an additional variable (here the Species variable of the Iris dataset)

of the prototypes does not recover the original separation in three classes.

As a comparison, Figure 11 shows a related analysis conducted via principal component analysis (PCA). More precisely, a PCA is conducted on the Iris data using the numerical variables only while the Species variable is used to colour the projected points. While the results are quite comparable, especially regarding the presence of two clearly separated clusters and the near separability between the two classes that form the second cluster, a few differences can be spotted. All in one, the SOM seems to give more weight to the idea that the three classes can be separated easily than the PCA. There is some interleaving in the classes in the SOM (especially a small blue cluster in the middle of green clusters) while the PCA shows a more complex boundary between the two classes. As the PCA projects the observations, it introduces spurious neighbourhood relationships between some points. The SOM generally respects more the original topology. In this particular example, the SOM is therefore able to give a better picture of the original structure of the dataset.

Another way of viewing the results of the SOM is to compute a PCA on the prototypes obtained by the SOM and to use it to display both the data and the prototypes (one can also use a non linear projection method). Figure 12 is obtained with this approach. The black dots are the prototypes and the

```
> iris.pca <- prcomp(data)
> plot(iris.pca$x,pch=20,
+      col=c("red","green","blue")[unclass(iris[[5]])])
```

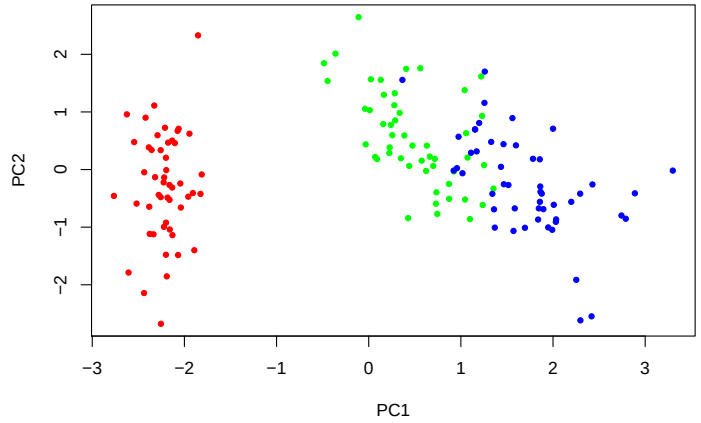

Figure 11: Principal Component Analysis applied to the Iris dataset

lines between them display the direct neighbourhood relationship enforced by the prior structure.

One limitation of the grid based display of the SOM is that the distances between clusters in those representations are arbitrary while the distances between prototypes in the original space provide insights on the relation between those clusters. The u-matrix and its numerous variations gives a visual representation of the distribution of those distances, as shown on Figure 13. This representation confirms the existence of two classes in the data and also the presence of some sub-structure in the largest class (on the right part of all figures). Another u-matrix like representation is provided by the plotting method associated to the object that represents prototypes distances, as shown on Figure 14.

```
> som.pca <- prcomp(som$prototypes)
> data.pca <- predict(som.pca,data)
> plot(data.pca,pch=20,xlim=range(data.pca[,1],som.pca$x[,1]),
+      ylim=range(data.pca[,2],som.pca$x[,2]),
+      col=c("red","green","blue")[unclass(iris[[5]])])
> lines(grid2lines(som,som.pca$x),type="b",pch=20)
```

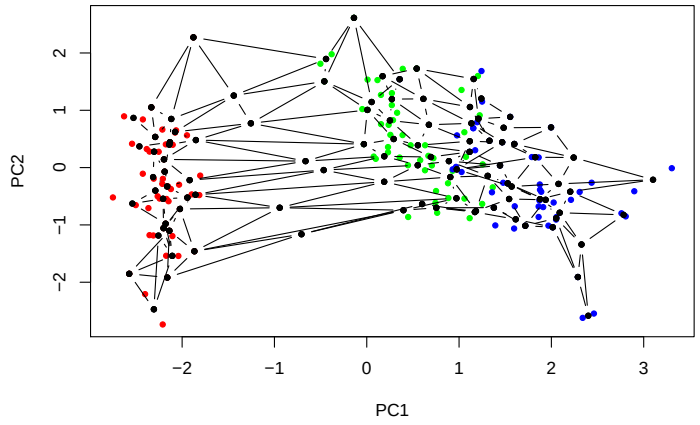

Figure 12: Principal Component Analysis display of the prototypes of the fitted SOM

```
> pdist <- prototype.distances(som)
> pgrid <- distance.grid(pdist)
> filled.contour(pgrid,color.palette=terrain.colors,asp=1)
```

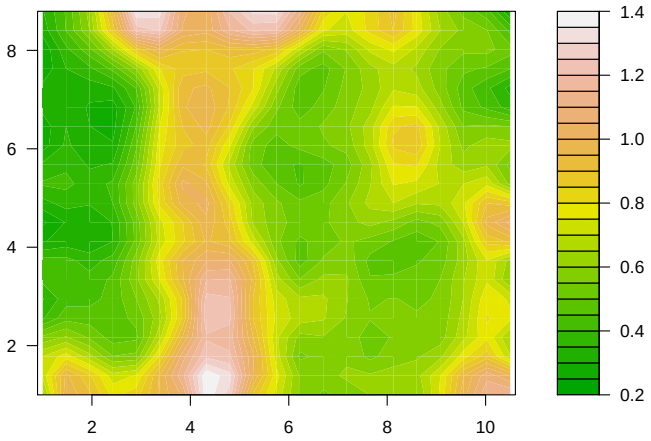

Figure 13: A type of U-matrix

```
> plot(pdist,color.palette=terrain.colors,asp=1)
```

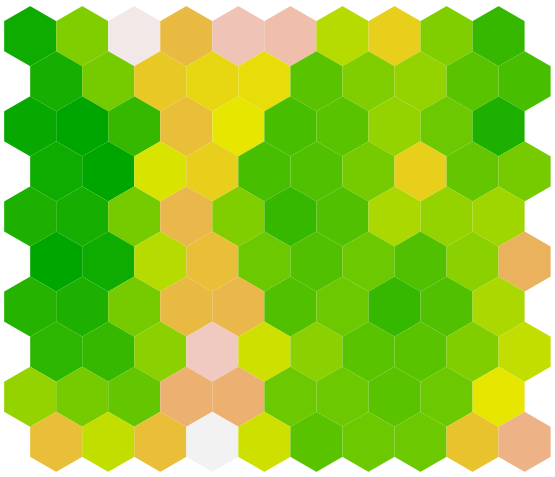

Figure 14: A discrete type of U-matrix