# BERTIN MATRICES

# AN  IMPLEMENTATION

GÜNTHER SAWITZKI
STATLAB HEIDELBERG

*in preparation*

Contents



## 1. Bertin Plots

Among the rich material on graphical presentation of information, in (Bertin, 1977) (engl. (Bertin, 1999)) J. Bertin discusses the presentation of data matrices, with a particular view to seriation. (de Falguerolles et al., 1997) gives an appraisal of this aspect of Bertin's work. The methods illustrated in (de Falguerolles et al., 1997) have been first implemented in the Voyager system (Sawitzki, 1996). They have been partially re-implemented in R, and this paper gives an introduction to the R-implementation.

*Date*: Aug. 2010.
Revised: August 2011
Typeset, with minor revisions: September 13, 2011from SVN *Revision* : 33.
*Key words and phrases.* statistical computing, S programming language, R programming, data analysis, exploratory statistics, residual diagnostics, R language.
*URL:* `http://bertin-forge.r-project.org/`.

The R-implementation can be downloaded as a package ***bertin*** from `http://bertin.r-forge.r-project.org/`. (de Falguerolles et al., 1997) is included in the documentation section of the package.

Bertin uses a small data set on hotel occupancy data to illustrate his ideas.

| | Jan | Fev | Mars | Avril | May | Juin | Juil | Aout | Sept | Oct | Nov | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ClienteleFeminine | 26 | 21 | 26 | 28 | 20 | 20 | 20 | 20 | 20 | 40 | 15 | 40 |
| Locale | 69 | 70 | 77 | 71 | 37 | 36 | 39 | 39 | 55 | 60 | 68 | 72 |
| USA | 7 | 6 | 3 | 6 | 23 | 14 | 19 | 14 | 9 | 6 | 8 | 8 |
| AmerSud | 0 | 0 | 0 | 0 | 8 | 6 | 6 | 4 | 2 | 12 | 0 | 0 |
| Europe | 20 | 15 | 14 | 15 | 23 | 27 | 22 | 30 | 27 | 19 | 19 | 17 |
| MOrientAfrique | 1 | 0 | 0 | 8 | 6 | 4 | 6 | 4 | 2 | 1 | 0 | 1 |
| Asie | 3 | 10 | 6 | 0 | 3 | 13 | 8 | 9 | 5 | 2 | 5 | 2 |
| Business | 78 | 80 | 85 | 86 | 85 | 87 | 70 | 76 | 87 | 85 | 87 | 80 |
| Touristes | 22 | 20 | 15 | 14 | 15 | 13 | 30 | 24 | 13 | 15 | 13 | 20 |
| ResDirecte | 70 | 70 | 78 | 74 | 69 | 68 | 74 | 75 | 68 | 68 | 64 | 75 |
| ResAgents | 20 | 18 | 19 | 17 | 27 | 27 | 19 | 19 | 26 | 27 | 21 | 15 |
| EquipageAeriens | 10 | 12 | 6 | 9 | 4 | 5 | 7 | 6 | 6 | 5 | 15 | 10 |
| MoinsDe20 | 2 | 2 | 4 | 2 | 2 | 1 | 1 | 2 | 2 | 4 | 2 | 5 |
| De20a55 | 25 | 27 | 37 | 35 | 25 | 25 | 27 | 28 | 24 | 30 | 24 | 30 |
| De35a55 | 48 | 49 | 42 | 48 | 54 | 55 | 53 | 51 | 55 | 46 | 55 | 43 |
| PlusDe55 | 25 | 22 | 17 | 15 | 19 | 19 | 19 | 19 | 19 | 20 | 19 | 22 |
| Prix | 163 | 167 | 166 | 174 | 152 | 155 | 145 | 170 | 157 | 174 | 165 | 156 |
| Duree | 1.65 | 1.71 | 1.65 | 1.91 | 1.9 | 2 | 1.54 | 1.6 | 1.73 | 1.82 | 1.66 | 1.44 |
| Occupation | 67 | 82 | 70 | 83 | 74 | 77 | 56 | 62 | 90 | 92 | 78 | 55 |
| Foires | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

TABLE 1. Bertin's hotel data

## 2. Bertin Matrices

To repeat from (de Falguerolles et al., 1997): In abstract terms, a Bertin matrix is a matrix of displays. Bertin matrices allow rearrangements to transform an initial matrix to a more homogeneous structure. The rearrangements are row or column permutations, and groupings of rows or columns. To fix ideas, think of a data matrix, variable by case, with real valued variables. For each variable, draw a bar chart of variable value by case. Highlight all bars representing a value above some sample threshold for that variable. See Figure 1.

Variables are collected in a matrix to display the complete data set. Figure 2. By convention, Bertin shows variables in rows and cases in columns. To make periodic structures more visible, Bertin repeats the data cyclically.

As Bertin points out, the indexing used is arbitrary. You can rearrange rows and/or columns to reveal the information of interest. If you run a hotel, of course the percentage of hotel occupation and the duration

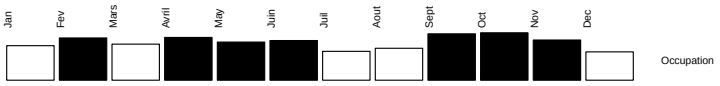



FIGURE 1. Display of one variable. Observations above average are highlighted.

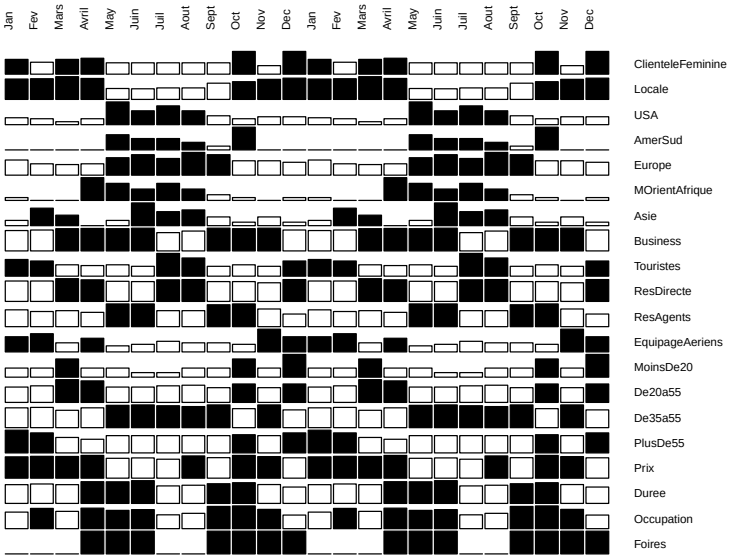



FIGURE 2. Display of a data matrix: Hotel data. Variables are shown as rows. To make periodic structures more visible, time is duplicated.

of the visits are most interesting for you. Move these variables to the top of the display, and rearrange the other variables by similarity or dissimilarity to these target variables. See Figure 3. Time points have a natural order. No rearragement is used here.

As a second example, we use the the *USJudgeRatings* data set (Figure 4)..

Botth cases (the judges) and variables (the qualitites) allow for a rearragment. Just sorting for row and column avarages gives a more informative picture (Figure 5). The number of contacts stands out - it has a different structure than the other variables. Judge G. A. Saden seems to be special. Most variables would rank him to the upper group, be his worth of retention is below average. The esteem of

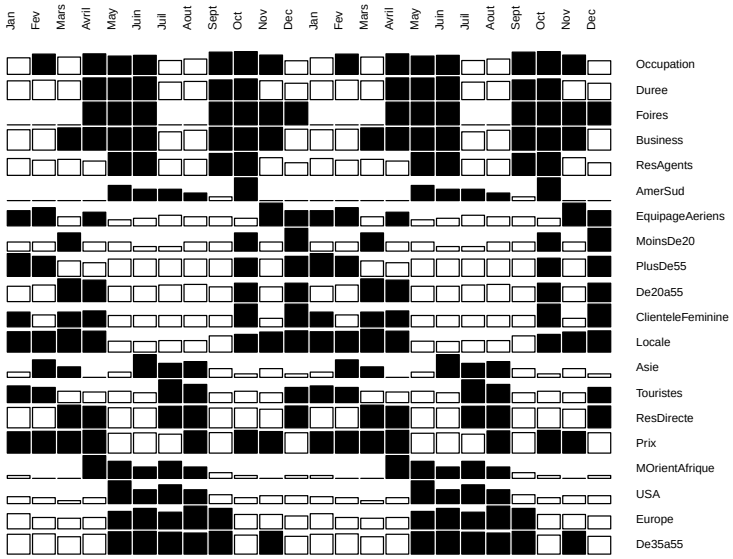



FIGURE 3. Display of a data matrix: Hotel data. Variables are rearranged by similarity to occupation and duration.

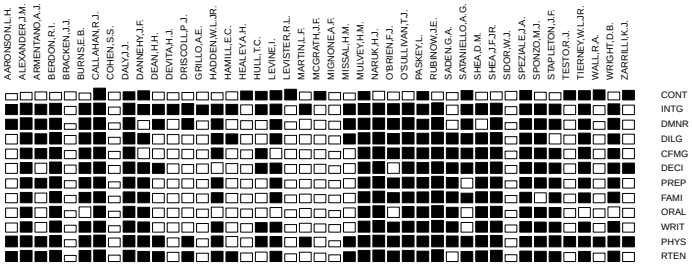



FIGURE 4. Display of a data matrix: USJudgeRatings data. Lawyers' ratings of state judges in the US Superior Court.

his integrity and demeanor go along with this. Overall, there is a very clear separation into an upper and a lower group.

At this early point, let us put Bertin's work in place. Visualizing information is but one aspect. In statistics, as we see it today, visualization may be one part of an analysis. The outcome will be a decision leading to an action, and then there is a loss (or gain) depending on the action taken on the one hand, and the "true" state of the world on the other. Statistics has formulated a few standard problems, and given a

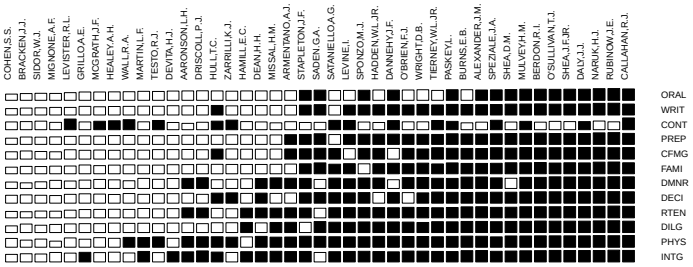



FIGURE 5. Display of a data matrix: USJudgeRatings data. Lawyers' ratings of state judges in the US Superior Court.

suggestion to handle these. In our example, the problem can be seen as a prediction problem: find a prediction model to predict occupation and duration, based on the other variables. This is a control problem, and the statistical contribution is to find a regression model for occupation and duration, based on the other variables. The visualization can be seen as one way to hint at a regression model. There are few classical problems. Regression is one of them, and prediction is closely related. Classification and clustering is another, closely related pair of problems, and their relation to Bertin matrices should be obvious. The USJudgeRatings can be looked at as a classification problem.

## 3. Work flow

We prefer to see Bertin matrices as a part of a work flow.

In a first step, we transfer the input data to allow for common, or comparable scales. In the Hotel example, Bertin rescales by the maximum value of each variable. The dichotomous variable Faires is recoded as 0/1. Our implementation default is to rescale for `(0, max)` for positive variables, `(min, 0)` for negative variables, `(min, max)` for general variables. Out preferred, or recommended rescaling however is to use ranks. We use the term `score` for the rescaled variables. Orientation of the data set is critical in this step. Usually, rescaling should be by variable, not by case. Depending on the orientation, this can lead for example to ranks by row or by column. We allow global scaling as an additional option for those situations where all data are already on a common scale. Following Bertin, our implementation default is to expect variables in rows.

In a second step, the scores are translated to visualization attributes. Colour is handled in two steps. The scores are translated to a colour index, which is used together with a colour palette to determine the display colour for a data element. This allows rapid experiments with various colour palettes, as long as the length of the palette is compatible. We strongly recommend to always look at the inverted table together with a chosen one to mitigate the effects of colour perception. Simple image displays limit the visualization attributes to colour. `rect` for example allows rectangle geometry, colour, and border width. Shading and shading should be considered as an alternative for print media.

Visualization attributes may reflect different aspect. So for example in the classical Bertin display, height of a rectangle is used to reflect the value of a data element, colour is used to show an indicator whether the value is above or below variable mean.

A third step controls the actual placement of the graphical elements. With a matrix layout, it is specified by possible permutations of rows and of columns. This may be related to information used in the first two steps, but should be considered an independent step. A vector or row orders and of column orders is the critical information from this step. Various seriation methods apply. The typical situation is to select scores and display attributes, and then search for optimal or good seriations. The arrangement often leads to hard optimization problems. Placing this step later allows to use information from score transformation and attributes, which may allow more efficient algorithms.

The final step is to merge these informations and render a display.

## 4. Permutation, Seriation, Arrangement

As Bertin has pointed out,

> *Ce point est fondamental. C'est la mobilité interne de l'image qui charactérise la graphique moderne.* [Bertin 1977, p. 5]

Once we have solved the problem, the way can often be formulated as an optimization problem. But while we are searching for a solution, experimenting is necessary. In our implementation, we separate to lines of experiment. Finding an adequate display is one branch. This amounts to building up a collection of proven models, and a certain data set can contribute by hinting at specific needs. This is repeated

not so often. Stability of implementation has priority over speed. We will provide a small number of basic model implementations.

The classic Bertin display shown above is one of the examples. Following the ideas, but deviating in the details, is to use a simple gray scale image for representation. This may be the most economic variant. But it is most economic in the use of display space. See figure 6. We will follow the classic Bertin display and an image display as main examples.

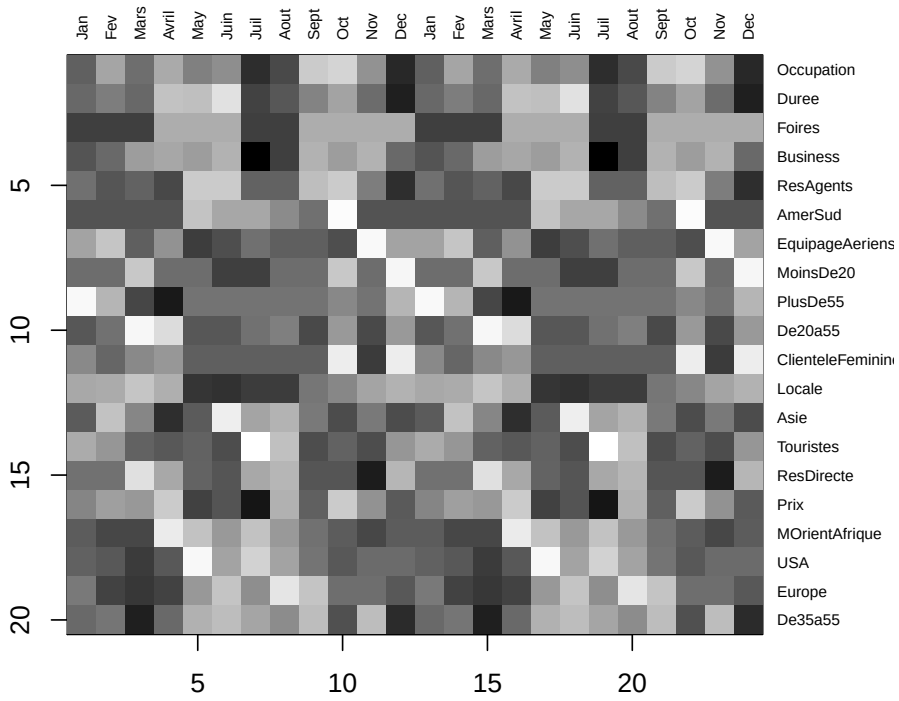


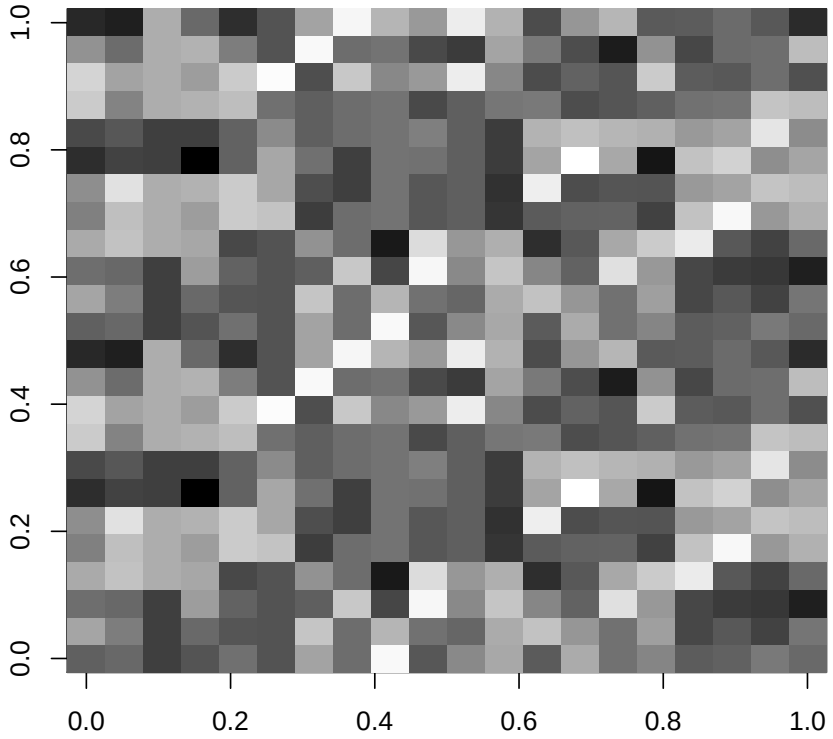

FIGURE 6. Display of a data matrix as gray scale image. Variables are rearranged by similarity to occupation and duration.

hunting for the alias problem

*Input*

```
Hotel2 <-as.matrix(Hotel2)
rowmeans <- apply(Hotel2,1,mean)
```

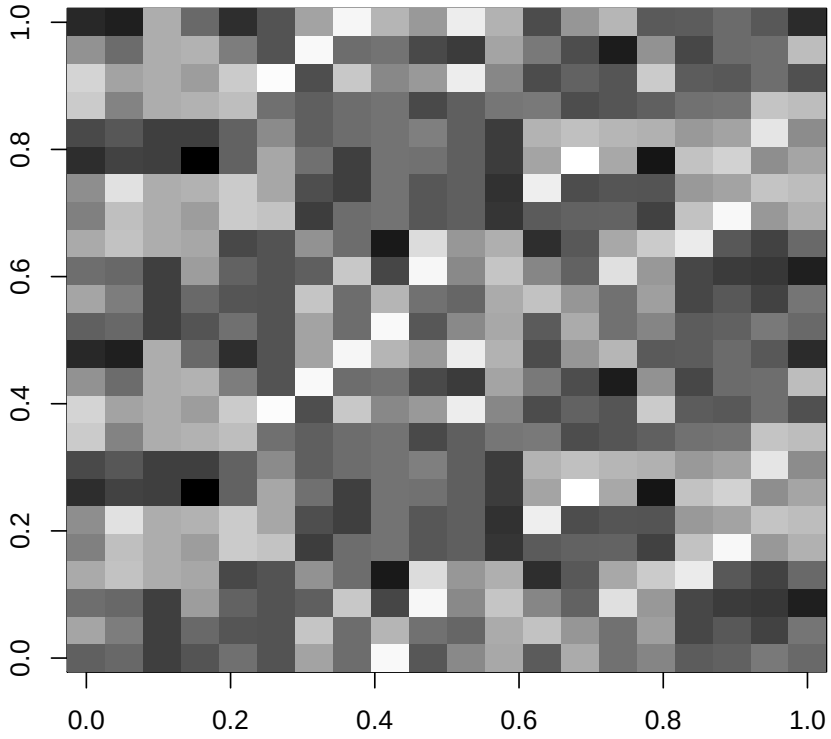
```
rowsd<- apply(Hotel2,1,sd)
zscores <-as.matrix( (Hotel2-rowmeans)/rowsd)
rowperm <- c(19,18,20, 8, 11, 4, 12, 13,16,14,1,2, 7, 9, 10, 17, 6, 3, 5,15)
#col=gray((0:256)/[zscores]
#image.bertin(zscores[rowperm,], main="", col=gray((0:256)/256), useRaster=TRUE)
image(zscores[rowperm,], main="", col=gray((0:256)/256), useRaster=TRUE)

#image.bertin(zscores[rowperm,], main="",  useRaster=TRUE)
```




Input

```
Hotel2 <-as.matrix(Hotel2)
rowmeans <- apply(Hotel2,1,mean)
rowsd<- apply(Hotel2,1,sd)
zscores <-as.matrix( (Hotel2-rowmeans)/rowsd)
rowperm <- c(19,18,20, 8, 11, 4, 12, 13,16,14,1,2, 7, 9, 10, 17, 6, 3, 5,15)
#col=gray((0:256)/[zscores]
#image.bertin(zscores[rowperm,], main="", col=gray((0:256)/256), useRaster=TRUE)
image(zscores[rowperm,], main="", col=gray((0:256)/256))
```

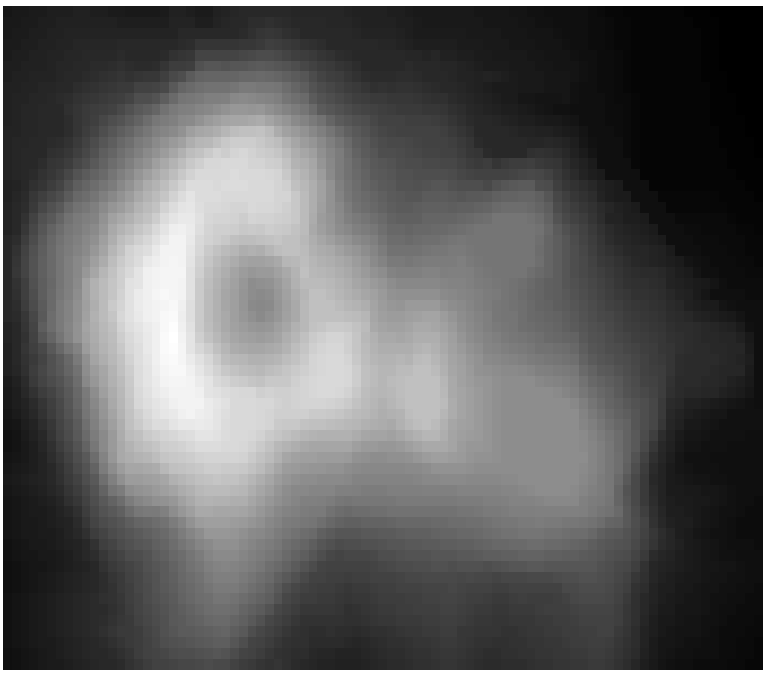
```
#image.bertin(zscores[rowperm,], main="",  useRaster=TRUE)
```




Input

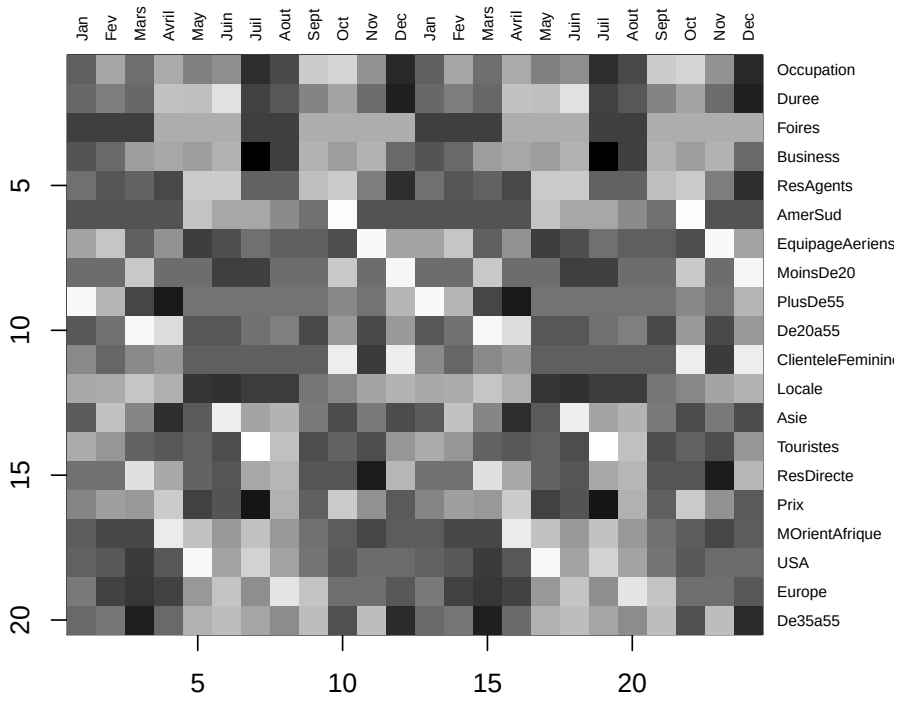
```
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col =gray((0:256)/256), axes = FALSE)
```

```
#image.bertin(zscores[rowperm,], main="",  useRaster=TRUE)
```




Input

```
Hotel2 <-as.matrix(Hotel2)
rowmeans <- apply(Hotel2,1,mean)
rowsd<- apply(Hotel2,1,sd)
zscores <-as.matrix( (Hotel2-rowmeans)/rowsd)
rowperm <- c(19,18,20, 8, 11, 4, 12, 13,16,14,1,2, 7, 9, 10, 17, 6, 3, 5,15)
#col=gray((0:256)/[zscores]
#image.bertin(zscores[rowperm,], main="", col=gray((0:256)/256), useRaster=TRUE)
imagem(zscores[rowperm,], main="", col=gray((0:256)/256), useRaster=TRUE)

#image.bertin(zscores[rowperm,], main="",  useRaster=TRUE)
```




For a chosen display, we have to compare different arrangements (seriations, for example). If we allow for interactive work, speed of display has priority. We try to cache the information that is invariant of the permutation.

As a final aspect, display space is limited. The number of variables and cases that can be displayed simultaneously is limited by the pixel size of the display. We can increase it by one or two magnitudes by using e series of detail displays. Any display calibration however should be constant for this series We try to allow for this global calibration.

The restriction to a matrix structure is arbitrary and can be omitted. Bertin has been working as a cartographer, and his main work applies to geographical data. What we call the Bertin matrices has been introduced in the very beginning of his book and are but a starting point.

## 5. Colour, perception and pitfalls

still to fix

Perception is an active process, and any visual presentation may be swayed by the intricacies of perception. Colour perception is particularly complex. When working with colour (and this includes black and white), we strongly suggest to have a look at the image with inverted colours as well.

Here is a sample implementation. On the R level, provide a plotting function

Input

```
sampleimagem <- function(z,
        col = grey((1:256)/256), xlab, ylab, main,
        colinvert=FALSE){
        if (colinvert) col <- col[length(col):1]
        # x1, x2. y1, y2
        oldpar <- par(fig=c(0, 1, 0.2, 1),
                mar=c(2.5,1.5,0.5,0.5), new=FALSE)
        imagem(z, col=col)

        par(yaxt="n", fig=c(0, 1, 0, 0.2),
                mar=c(3.5,12.0,0.5,12.0), new=TRUE)
#        colramp(col=col, horizontal=TRUE)
        zrange <- range(z, finite=TRUE)
        image(z=t(matrix(seq(zrange[1],zrange[2],length.out=length(col)),
                1, length(col))),
                zlim=zrange,main="", ylab="", xlab="", col=col)
        par(oldpar)
}
```

and run it with `colinvert=FALSE` and `colinvert=TRUE` . If your are using `Sweave`, use two separate chunks, and place the figure output side by side in TEX.

Input

```
hotelrk <- bertinrank(Hotel)
sampleimagem(hotelrk)
```

See Figure 7 left.

Input

```
sampleimagem(hotelrk, colinvert=TRUE)
```

See Figure 7 right.

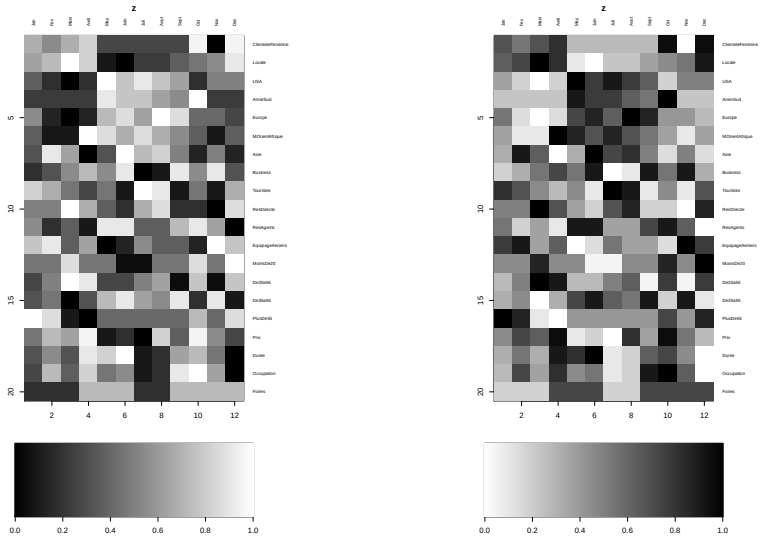


FIGURE 7. Test matrix: same information, but colour table inverted on the right.

## 6. Coordinate System and Conventions

We provide prototypes for the display of Bertin matrices. To simplify the implementation of extensions, we chose a coordinate system that allots unit square to each matrix cell. If we want to add separator lines to structure our data, we have to allocate additional space `sepwd`, measured in user space.

`image` requires increasing scales for both axes. To align with matrix conventions, we have to transform the row numbers.

The total user space has the size `(nrrows * (1+2*sepwd), nrcols * (1+2*sepwd)`. The drwing area for cell `x[i, j]` is a unit square with bottom left coordinates `(nrrows-i+1)* (1+2*sepwd) - sepwd* (1+2*sepwd) - sepwd, j* (1+2*sepwd) - sepwd` .

## 7. Test matrices

To test the implementation, a series of matrices is provided.

### 7.1. **Random Matrices.**

*Input*

```
nrow <- 5
ncol <- 3
BMunif <- matrix(runif(nrow*ncol), nrow, ncol)
colnames(BMunif) <- colnames(BMunif, do.NULL=FALSE)
rownames(BMunif) <- rownames(BMunif, do.NULL=FALSE)
BMnorm <- matrix(rnorm(nrow*ncol), nrow, ncol)
colnames(BMnorm) <- colnames(BMnorm, do.NULL=FALSE)
rownames(BMnorm) <- rownames(BMnorm, do.NULL=FALSE)
```

7.2. **Pure Vanilla.** The most simple case: all variables are on a common scale, and the sequence is given (no seriation possible) or irrelevant (no seriation necessary).

If we want to build test matrices, there are two free parameters to be set, for example

Input

```
BMExplRows=8
BMExplCols=6
```

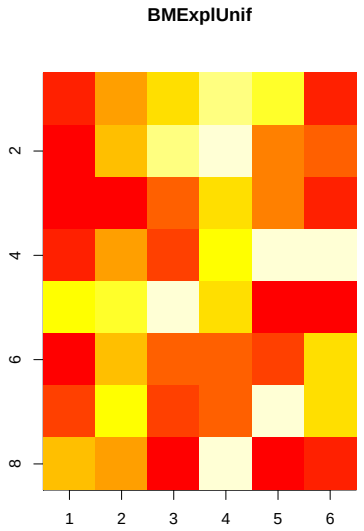
Typical cases are :

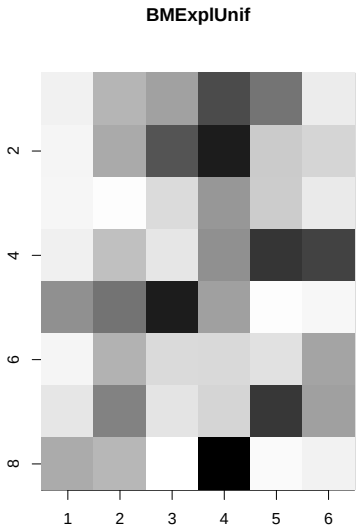
Input

```
BMExplUnif  <- matrix( runif(BMExplRows*BMExplCols),
        nrow= BMExplRows, ncol= BMExplCols)
BMExplNorm  <- matrix( rnorm(BMExplRows*BMExplCols),
        nrow= BMExplRows, ncol= BMExplCols)
```

Input

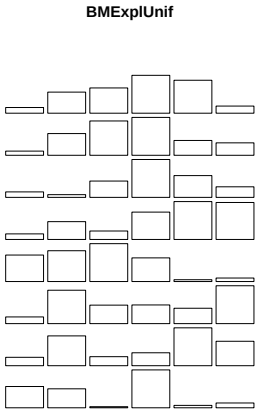
```
oldpar <- par(mfrow=c(1,2))
imagem(BMExplUnif)
image.bertin(BMExplUnif,useRaster=FALSE)
par(oldpar)
```

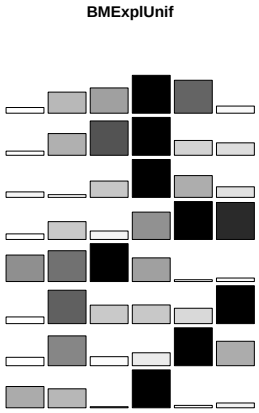



Input

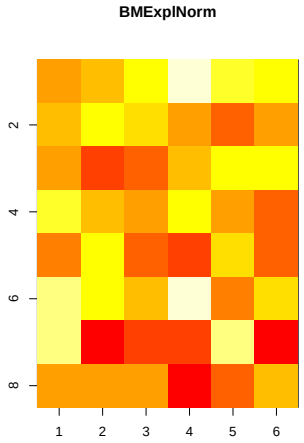
```
oldpar <- par(mfrow=c(1,2))
bertinrect(BMExplUnif)
plot.bertin(BMExplUnif)
par(oldpar)
```

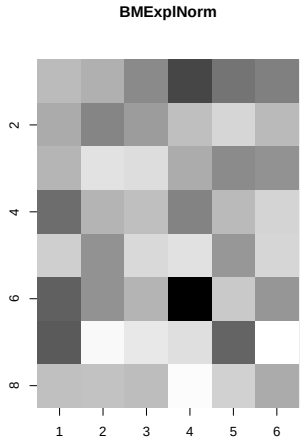



Input

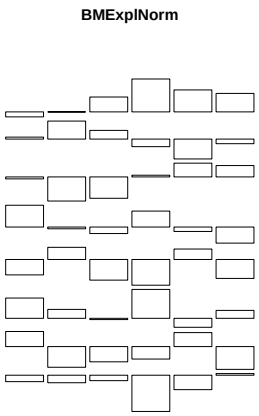
```
oldpar <- par(mfrow=c(1,2))
imagem(BMExplNorm)
image.bertin(BMExplNorm)
par(oldpar)
```

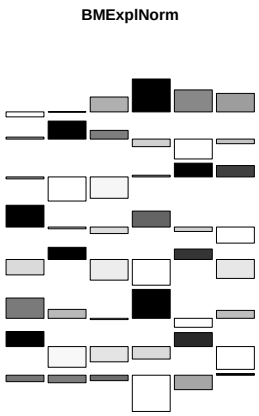



*Input*

```
oldpar <- par(mfrow=c(1,2))
bertinrect(BMExplNorm)
plot.bertin(BMExplNorm)
par(oldpar)
```




The deficits are obvious. The colours used by default appear qualitatively different, but they do not convey quantitative information. This can be easily overcome by using better colour scales. The minimum to do is to use gray scales, but of course better solutions are readily available.

The other obvious problem is that image does not reflect the orientation of the matrix (image uses image-conventions with origin at the bottom left, wheres matrix numeration uses an origin at to top left). Moreover, the aspect ratio of the image does not correspond to the aspect ratio

of the matrix. The only solution is to rewrite image to a variant that is adapted to matrix conventions.

*Input*

A hidden problem with image representations is that they provide more information than usually can be processed. The colours ore grey tones may reflect to many differences. The second cheap solution to present a matrix is an array of histograms:

7.3. **Vanilla.** The next round of test cases are numeric, but not on a common scale. We provide some test vectors which we can use to construct various test matrices.

*Input*

```
# Test vectors, used to build a matrix
Bzero <- rep(0, BMExplCols)
Bone <- rep(1, BMExplCols)
Bmone <- rep(-1, BMExplCols)
Binc <- (1:BMExplCols)/BMExplCols
Bdec <- (BMExplCols:1)/BMExplCols
Bstep <- c(Bmone[1:floor(BMExplCols/2)],
        Bone[(1+floor(BMExplCols/2)):BMExplCols])
Bhat <- Bone
Bhat[(floor(BMExplCols/3)+1):(BMExplCols-floor(BMExplCols/3)) ] <- 0.5
Bnazero <- rep(c(NA,0),length.out= BMExplCols)
Bnanzero <- rep(c(NaN,0),length.out= BMExplCols)
Binf <- rep(c(Inf,0,-Inf),length.out= BMExplCols)
```

*Input*

```
# Basic test matrices
Brmatrix <- rbind(Bzero, Bone, Bmone, Binc, Bdec, Bstep, Bhat)
colnames(Brmatrix) <- colnames(Brmatrix,FALSE)
```

*Input*

```
## R may use internal housekeeping
## to keep matrix columns homogeneous.
## Check!
## Use row matrix and column matrix for tests.
Bcmatrix <- cbind(Bzero, Bone, Bmone, Binc, Bdec, Bstep, Bhat)
rownames(Bcmatrix) <- rownames(Bcmatrix,FALSE)
# Basic test matrices with random error
```

```
BrRndmatrix <- Brmatrix+rnorm(nrow(Brmatrix)*ncol(Brmatrix))
```

Input

```
# Test matrices with IEEE specials
Brmatrixx <- rbind(Bzero, Bone, Bmone, Binc, Bdec, Bstep, Bhat,
        Bnazero, Bnanzero, Binf)
Bcmatrixx <- cbind(Bzero, Bone, Bmone, Binc, Bdec, Bstep, Bhat,
        Bnazero, Bnanzero,Binf)
BrRndmatrixx <- Brmatrixx+rnorm(nrow(Brmatrixx)*ncol(Brmatrixx))
```

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Bzero | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Bone | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Bmone | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| Binc | 0.17 | 0.33 | 0.50 | 0.67 | 0.83 | 1.00 |
| Bdec | 1.00 | 0.83 | 0.67 | 0.50 | 0.33 | 0.17 |
| Bstep | -1.00 | -1.00 | -1.00 | 1.00 | 1.00 | 1.00 |
| Bhat | 1.00 | 1.00 | 0.50 | 0.50 | 1.00 | 1.00 |
| Bnazero | | 0.00 | | 0.00 | | 0.00 |
| Bnanzero | | 0.00 | | 0.00 | | 0.00 |
| Binf | Inf | 0.00 | -Inf | Inf | 0.00 | -Inf |

TABLE 2. Brmatrixx: matrix with special values, by row

## References

Bertin, J. 1977. *La graphique et le traitement graphique de l'information*, Flammarion, Paris.

———. 1999. *Graphics and graphic information processing*, Readings in information visualization, pp. 62–65.

de Falguerolles, Antoine, Felix Friedrich, and Günther Sawitzki. 1997. *A tribute to J. Bertin's graphical data analysis*, Softstat '97 (advances in statistical software 6), pp. 11–20.

Sawitzki, Günther. 1996. *Extensible statistical software: On a voyage to oberon.*, Journal of Computational and Graphical Statistics **5**, no. 3.

`$Id: bertinR.Rnw 33 2011-09-11 11:21:57Z gsawitzki $`

`$Revision: 33 $`

`$Date: 2011-09-11 13:21:57 +0200 (Sun, 11 Sep 2011) $`

`$Author: gsawitzki $`

Address: StatLab Heidelberg

*E-mail address*: `gs@statlab.uni-heidelberg.de`

*URL*: `http://www.statlab.uni-heidelberg.de/projects/r/`