BERTIN MATRICES

AN  IMPLEMENTATION

GÜNTHER SAWITZKI
STATLAB HEIDELBERG

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

Date: Aug. 2010.
Revised: August 2011
Typeset, with minor revisions: May 13, 2012 from SVN *Revision* : 60.
Key words and phrases. statistical computing, S programming language, R programming, data analysis, exploratory statistics, residual diagnostics, R language.
URL: `http://bertin.r-forge.r-project.org/`.

particular view to seriation. (de Falguerolles et al., 1997) gives an appraisal of this aspect of Bertin's work. The methods discussed in (de Falguerolles et al., 1997) have been 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.

The R-implementation can be downloaded as a package *bertin* from `http://bertin.r-forge.r-project.org/`. The paper (de Falguerolles et al., 1997) is included as bertin.pdf 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 (Figure 1 on page 3).

Variables are collected in a matrix to display the complete data set (Figure 2 on page 3). By convention, Bertin shows variables in rows and cases in columns. To make periodic structures more visible, the data are repeated 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 of the visits are most interesting for you. Move these variables to the top of the display, and rearrange the other variables by

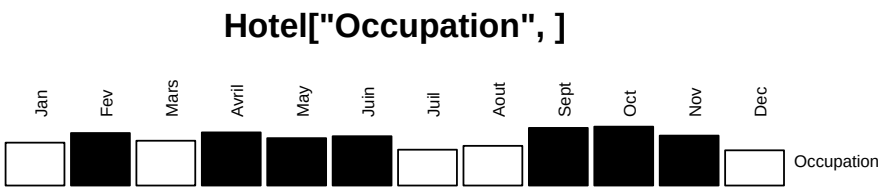


Figure 1: Display of one variable: **Hotel data**. Observations above average are highlighted in black.

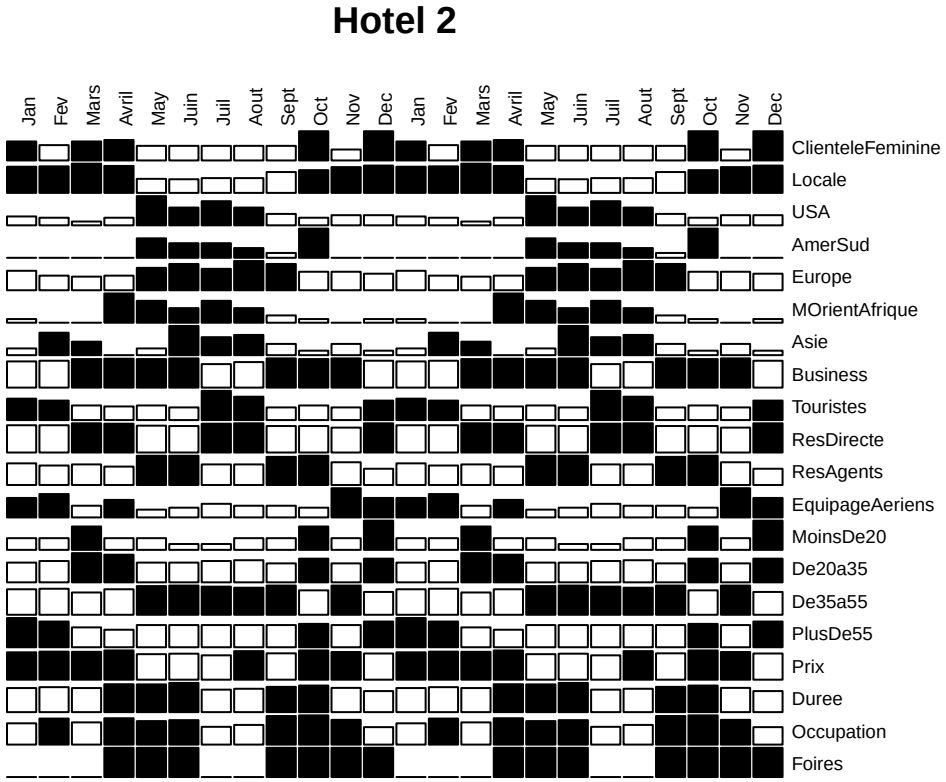


Figure 2: Display of a data matrix: **Hotel data**. Variables are shown as rows. To make periodic structures more visible, time is duplicated. Observations above average are highlighted.

similarity or dissimilarity to these target variables (Figure 3 on page 4). Time points have a natural order. No rearrangement is used here.

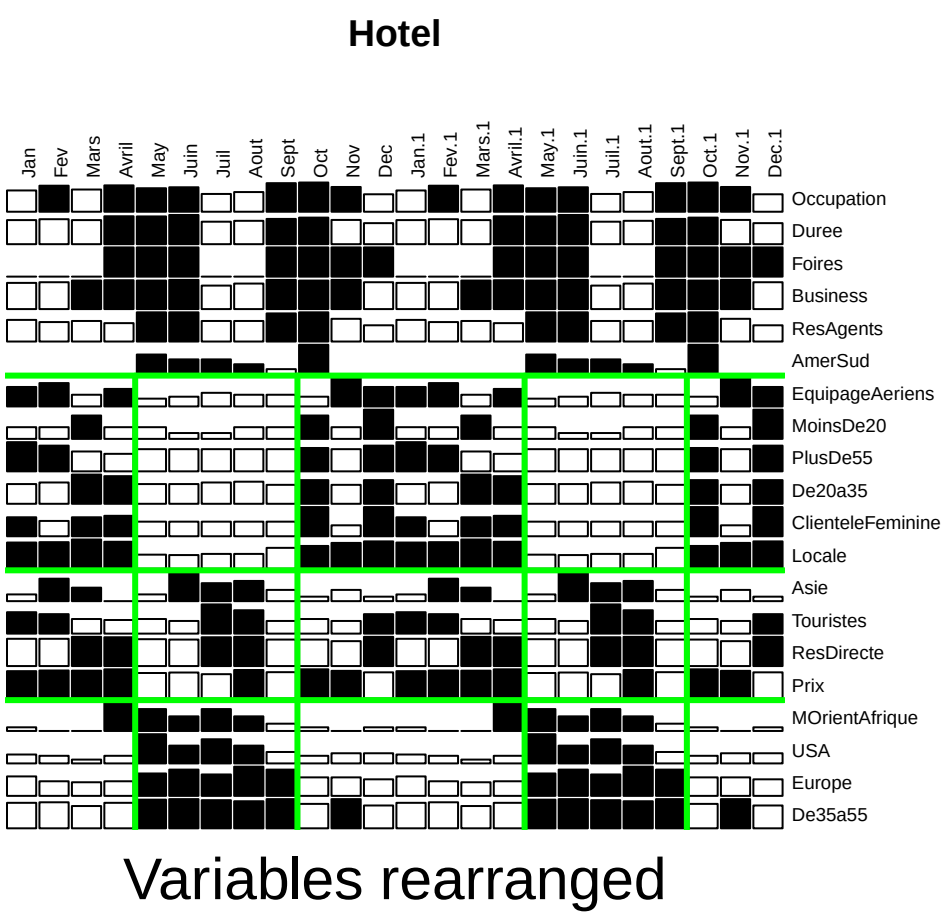


Figure 3: Display of a data matrix: **Hotel data**. Variables are rearranged by similarity to occupation and duration.

Variables need not enter at their face value; they can be transformed, or derived variable can be added. In the case of the hotel data, this has already done in the original data set. For example, the guests have been classified in tourists and business, and both sum up to 100%. If we want, we can remove this redundant information. This may clean up the picture. But it may hide information. For example, *tourists* are "anti-cyclic" to the hotel occupation and just fill the gaps. Removing this variable because it is $(1 - business)$ would hide this point.

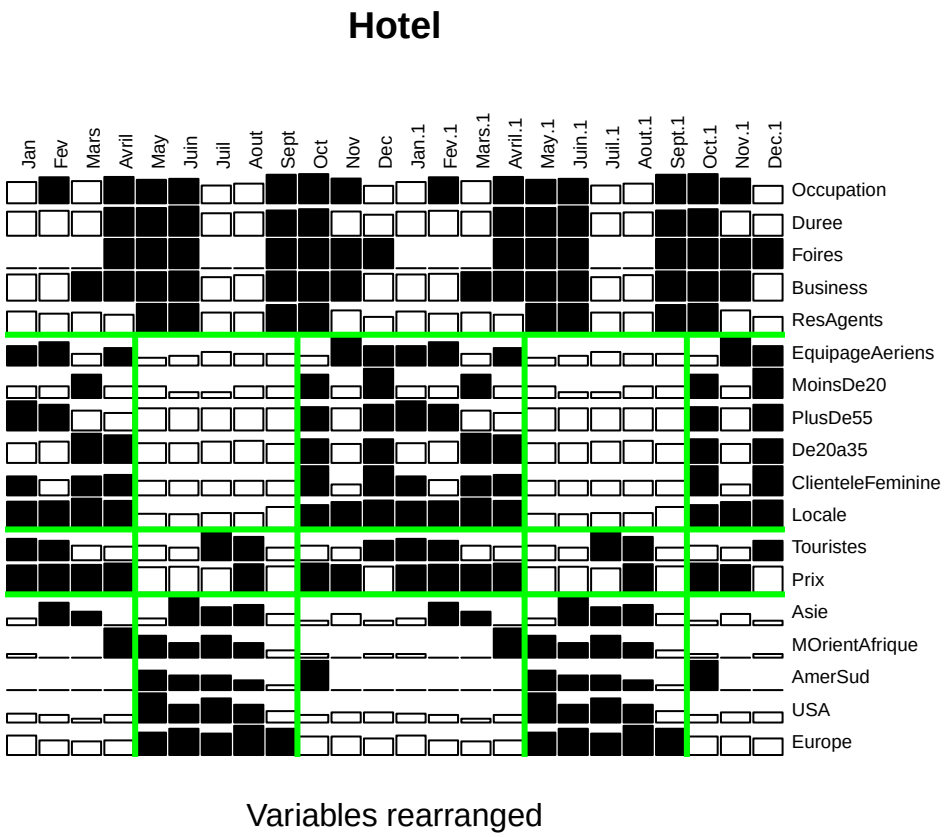


Figure 4: Display of a data matrix: **Hotel data**. Variables are rearranged by similarity to occupation and duration. Some redundancy removed.

As a second example, we use the the *USJudgeRatings* data set (Figure 5 on page 8). The data is listed in Table 2 on page 7.

	CONT	INTG	DMNR	DILG	CFMG	DECI	PREP	FAMI	ORAL	WRIT	PHYS	RTEN
AAR,L.H.	5.7	7.9	7.7	7.3	7.1	7.4	7.1	7.1	7.1	7.0	8.3	7.80
ALE,J.M.	6.8	8.9	8.8	8.5	7.8	8.1	8.0	8.0	7.8	7.9	8.5	8.7
ARM,A.J.	7.2	8.1	7.8	7.8	7.5	7.6	7.5	7.5	7.3	7.4	7.9	7.8
BER,R.I.	6.8	8.8	8.5	8.8	8.3	8.5	8.7	8.7	8.4	8.5	8.8	8.7
BRA,J.J.	7.3	6.4	4.3	6.5	6.0	6.2	5.7	5.7	5.1	5.3	5.5	4.8
BUR,E.B.	6.2	8.8	8.7	8.5	7.9	8.0	8.1	8.0	8.0	8.0	8.6	8.6
CAL,R.J.	10.6	9.0	8.9	8.7	8.5	8.5	8.5	8.5	8.6	8.4	9.1	9.0
COH,S.S.	7.0	5.9	4.9	5.1	5.4	5.9	4.8	5.1	4.7	4.9	6.8	5.0
DAL,J.J.	7.3	8.9	8.9	8.7	8.6	8.5	8.4	8.4	8.4	8.5	8.8	8.8
DAN,J.F.	8.2	7.9	6.7	8.1	7.9	8.0	7.9	8.1	7.7	7.8	8.5	7.9
DEA,H.H.	7.0	8.0	7.6	7.4	7.3	7.5	7.1	7.2	7.1	7.2	8.4	7.7
DEV,H.J.	6.5	8.0	7.6	7.2	7.0	7.1	6.9	7.0	7.0	7.1	6.9	7.2
DRI,P.J.	6.7	8.6	8.2	6.8	6.9	6.6	7.1	7.3	7.2	7.2	8.1	7.7
GRI,A.E.	7.0	7.5	6.4	6.8	6.5	7.0	6.6	6.8	6.3	6.6	6.2	6.5
HAD,W.L.JR.	6.5	8.1	8.0	8.0	7.9	8.0	7.9	7.8	7.8	7.8	8.4	8.0
HAM,E.C.	7.3	8.0	7.4	7.7	7.3	7.3	7.3	7.2	7.1	7.2	8.0	7.6
HEA,A.H.	8.0	7.6	6.6	7.2	6.5	6.5	6.8	6.7	6.4	6.5	6.9	6.7
HUL,T.C.	7.7	7.7	6.7	7.5	7.4	7.5	7.1	7.3	7.1	7.3	8.1	7.4
LEV,I.	8.3	8.2	7.4	7.8	7.7	7.7	7.7	7.8	7.5	7.6	8.0	8.0
LEV,R.L.	9.6	6.9	5.7	6.6	6.9	6.6	6.2	6.0	5.8	5.8	7.2	6.0
MAR,L.F.	7.1	8.2	7.7	7.1	6.6	6.6	6.7	6.7	6.8	6.8	7.5	7.3
MCG,J.F.	7.6	7.3	6.9	6.8	6.7	6.8	6.4	6.3	6.3	6.3	7.4	6.6
MIG,A.F.	6.6	7.4	6.2	6.2	5.4	5.7	5.8	5.9	5.2	5.8	4.7	5.2
MISL,H.M.	6.2	8.3	8.1	7.7	7.4	7.3	7.3	7.3	7.2	7.3	7.8	7.6
MUL,H.M.	7.5	8.7	8.5	8.6	8.5	8.4	8.5	8.5	8.4	8.4	8.7	8.7
NAR,H.J.	7.8	8.9	8.7	8.9	8.7	8.8	8.9	9.0	8.8	8.9	9.0	9.0
O'BR,F.J.	7.1	8.5	8.3	8.0	7.9	7.9	7.8	7.8	7.8	7.7	8.3	8.2
O'SU,T.J.	7.5	9.0	8.9	8.7	8.4	8.5	8.4	8.3	8.3	8.3	8.8	8.7
PAS,L.	7.5	8.1	7.7	8.2	8.0	8.1	8.2	8.4	8.0	8.1	8.4	8.1
RUB,J.E.	7.1	9.2	9.0	9.0	8.4	8.6	9.1	9.1	8.9	9.0	8.9	9.2
SAD.G.A.	6.6	7.4	6.9	8.4	8.0	7.9	8.2	8.4	7.7	7.9	8.4	7.5
SAT,A.G.	8.4	8.0	7.9	7.9	7.8	7.8	7.6	7.4	7.4	7.4	8.1	7.9
SHE,D.M.	6.9	8.5	7.8	8.5	8.1	8.2	8.4	8.5	8.1	8.3	8.7	8.3
SHE,J.F.JR.	7.3	8.9	8.8	8.7	8.4	8.5	8.5	8.5	8.4	8.4	8.8	8.8
SID,W.J.	7.7	6.2	5.1	5.6	5.6	5.9	5.6	5.6	5.3	5.5	6.3	5.3
SPE,J.A.	8.5	8.3	8.1	8.3	8.4	8.2	8.2	8.1	7.9	8.0	8.0	8.2
SPO,M.J.	6.9	8.3	8.0	8.1	7.9	7.9	7.9	7.7	7.6	7.7	8.1	8.0
STA,J.F.	6.5	8.2	7.7	7.8	7.6	7.7	7.7	7.7	7.5	7.6	8.5	7.7
TES,R.J.	8.3	7.3	7.0	6.8	7.0	7.1	6.7	6.7	6.7	6.7	8.0	7.0
TIE,W.L.JR.	8.3	8.2	7.8	8.3	8.4	8.3	7.7	7.6	7.5	7.7	8.1	7.9
WAL,R.A.	9.0	7.0	5.9	7.0	7.0	7.2	6.9	6.9	6.5	6.6	7.6	6.6
WRI,D.B.	7.1	8.4	8.4	7.7	7.5	7.7	7.8	8.2	8.0	8.1	8.3	8.1
ZAR,K.J.	8.6	7.4	7.0	7.5	7.5	7.7	7.4	7.2	6.9	7.0	7.8	7.1

Table 2: US judge ratings

Both the cases (the judges) and the variables (the qualities) allow for a rearrangement. Just sorting for row and column averages gives a more informative picture (Figure 6 on page 8). The number of contacts *CONT* stands out - it has a different structure than the other variables. After all, this is not a rating variable at all, but ancillary information. There is little reason why it should go along with the rating 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 his integrity and demeanour go along with this. Overall, there is a very clear separation into an upper and a lower group.

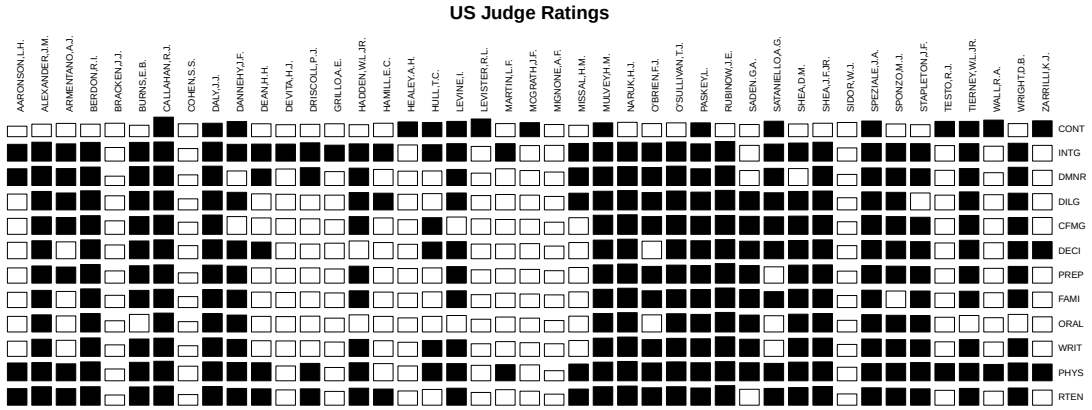


Figure 5: Display of a data matrix: **USJudgeRatings data**. Lawyers' ratings of state judges in the US Superior Court. Original arrangement, judges by lexical order of name.

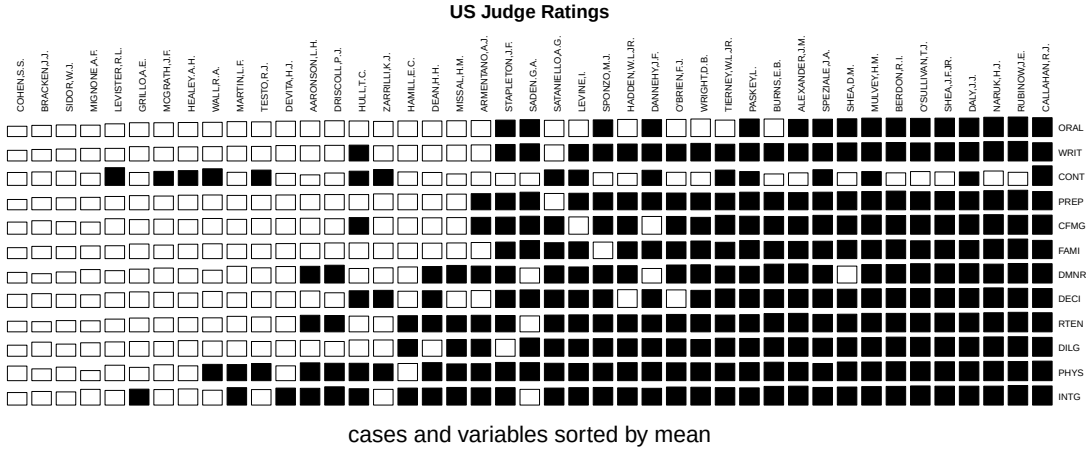


Figure 6: Display of a data matrix. **USJudgeRatings data**. Lawyers' ratings of state judges in the US Superior Court, rearranged.

We remove the variable *CONT* and re-run the analysis. Of course this changes the average scores per judge, and the arrangement changes (Figure 7 on page 9).

The highlight features of the display give a clear picture of a rather homogeneous upper and homogeneous lower group of judges. All criteria seem to say the same in these groups. For a small intermediate group of cases, the variables seem to fall into two or three groups, and it is only for this small group that a closer look at the

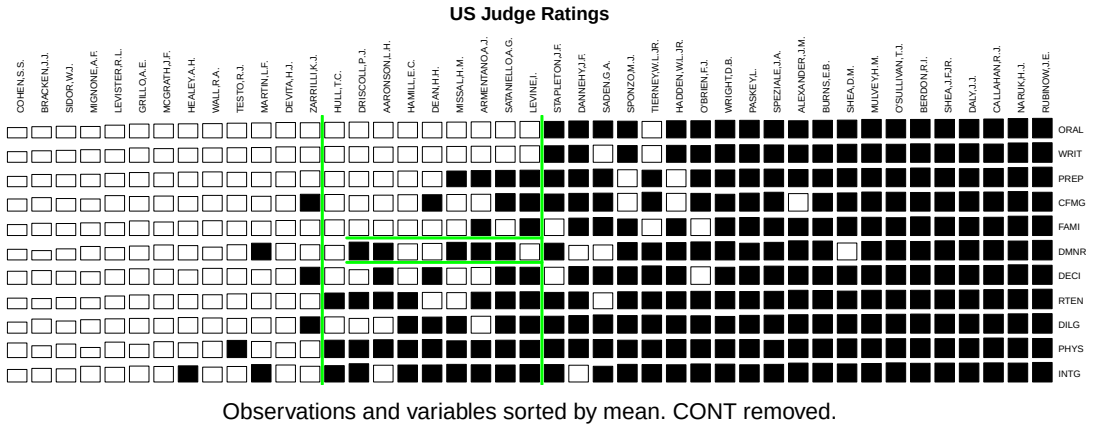


Figure 7: Display of a data matrix. **USJudgeRatings data** Lawyers' ratings of state judges in the US Superior Court. Variable *CONT* removed, and rearranged.

variables and their meaning may be necessary. If you want to model the rating of the worthiness of retention, say, as a function of the other ratings, it is only here that a refined model might be necessary. For most judges, a rough model would do.

For this data set, the low resolution of the variables may be a problem. No details are given in the documentation, but presumably the rating has been taken on a $1, \ldots, 10$ scale, and in all variables the median is higher than 7. (only one data point, a contact recorded as 10.6 does not fit into this and asks for explanation.) Changing the scale may give a better contrast and improves the resolution (see Figure 8 on page 10). Here we used the rescaled variables to define the height of the bars. For highlighting by colour (only black/white here) we used the data in the original scale. Both display attributes, height and colour, are independent and we are free to encode different aspects of the data. In Figure 8 on page 10 we sorted the cases by the value of *RTEN*. This lets us view the data from a different point of view.

Again we have a clear upper and lower group, with consistency over all variables, and a small intermediate group which needs closer inspection. Judge G.A. Saden stands out. By most criteria, he qualifies for a higher ranking, but demeanour and judicial integrity go along with a low scored worthy of retention.

Two judges are on the boundary and may well be exchanged - this is where random fluctuation may be at work to decide for the ranking.

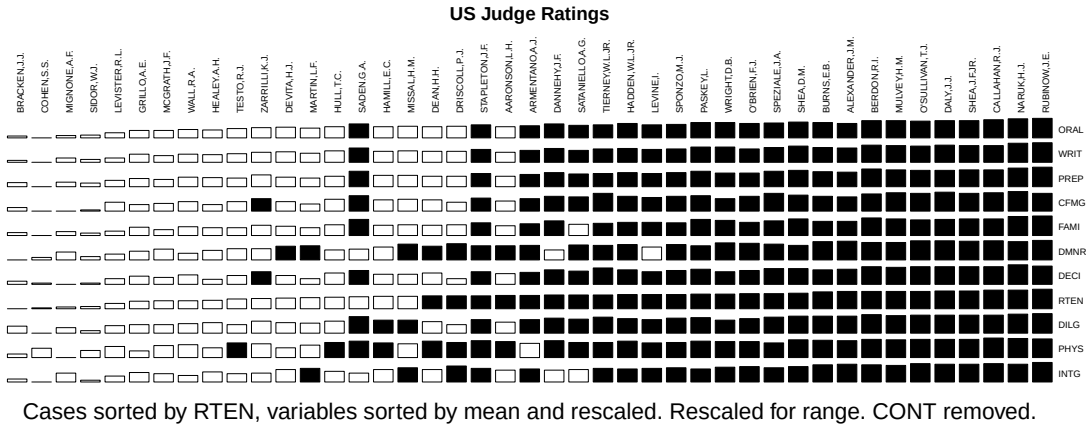


Figure 8: Data set: **USJudgeRatings**. Cases sorted by RTEN, variables sorted by mean. Rescaled for range. CONT removed. Scores above mean are highlighted.

At this early point, let us put Bertin's work in place. Visualising information is but one aspect. In statistics, as we see it today, visualisation may be one part of an analysis. The outcome will be a decision leading to an action. 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. This is the common decision theoretical setting. Statistics has formulated a few standard problems, and given suggestions how to handle these. In our Hotel example, the problem can be seen as a prediction problem: find a prediction model to predict occupation and duration, based on the other variables. More specifically 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 visualisation can be seen as one way to hint at a regression model. There are very 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

Bertin matrices usually are 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 encoded 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 convention 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, but we provide the means to switch to the R convention with variables in columns. The raw data may come in data frames, or lists, or views on a data base, and the original convention should be preserved. The scores however are a matrix, or an array (which we consider a stacked list of matrices in our context.) We prefer to keep these in Bertin conventions, that is variables are in rows.

In a second step, the scores are translated to visualisation 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 palettes are compatible. We strongly recommend to always look at the inverted colour palette together with a chosen one to mitigate the effects of colour perception. Simple image displays limit the visualisation attributes to colour. `rect` for example allows rectangle geometry, colour, and border width. Shading and line types should be considered as an alternative for print media.

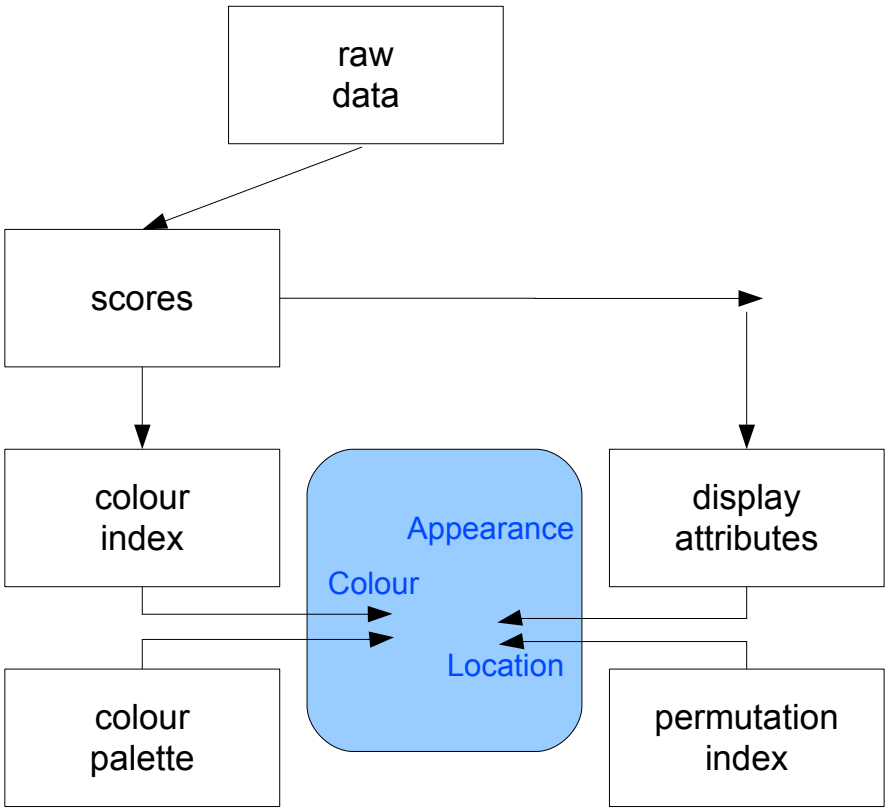


Figure 9: Data flow for Bertin

Visualisation attributes may reflect different aspects. 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. This is where Bertin's ideas about "internal mobility" as a characteristics of modern graphics come to action. The typical situation is to select scores and display attributes, and then search for optimal or good seriations. The arrangement often leads to hard optimisation problems. Placing this step later allows to use information from score transformation and attributes, which may allow more efficient algorithms. In the end, we may be better with a good solution which helps to solve the practical problem, instead of an optimal solution to a theoretical one. These may differ considerably.

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

4. Scores

In principle, scores can be generated using an appropriate score function and `apply` or any of its variants. As examples, and for convenience, we provide a small collection of score functions.

As an illustration, each is applied by row to the Hotel data set, and the result is shown using a default Bertin plot. A second plot shows a poor man's regression: assuming that the hotel occupancy, variable 19, is the parameter of interest. Sort the scores by correlation to the score of occupancy.

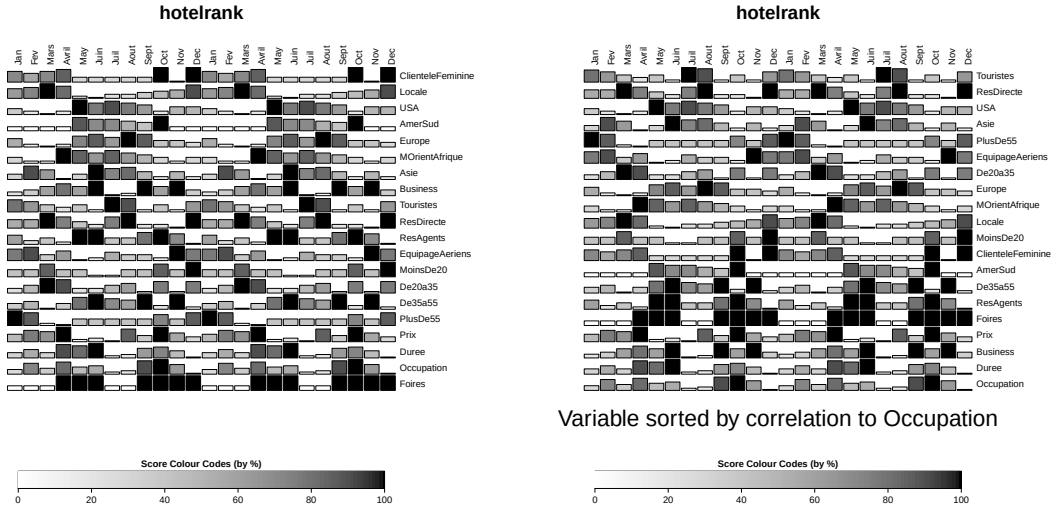
`var.orientation = "byrow"` is the default, it can be omitted. If needed in other data sets, add `var.orientation = "bycolumn"` or `var.orientation = "global"`. This allows to follow our design decision to keep the original data following the original conventions. The results here are matrices. They can be transposed at convenience.

4.1. **Ranks.**

$$x \mapsto rank(x)$$

Input

```
oldpar <-par(mfcol=c(1,2))
hotelrank <- bertinrank(Hotel2)
plot.bertin(hotelrank)
hotelrankorder <- bertin:::ordercor(hotelrank, 19)
plot.bertin(hotelrank, roworder=hotelrankorder)
title(sub="Variable sorted by correlation to Occupation", cex.sub=1.4, line=1)
par(oldpar)
```



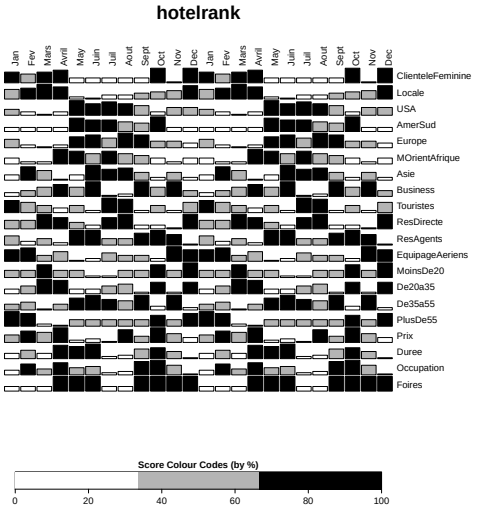
Variable sorted by correlation to Occupation

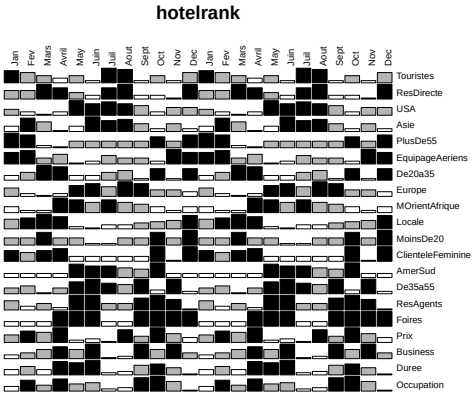
Rank scores have a sound statistical basis. They bring us back to the field of rank statistics. However, since they scale any rank difference by unit steps, they convey order, but not quantitative differences. Preferably they are combined with colour palettes which do not suggest a quantitative scale. The default colour palette uses 256 steps of grey and suggests a quantitative order, whereas the ranks by variable provide at most 12 steps in this example.

It is preferable to use a palette with reduced resolution. For 12 rank values, a scale with 3 or 4 steps should do.

Input

```
oldpar <-par(mfcol=c(1,2))
hotelrank <- bertinrank(Hotel2)
plot.bertin(hotelrank,
        palette = gray((2:0 / 2)^0.5))
hotelrankorder <- bertin:::ordercor(hotelrank, 19)
plot.bertin(hotelrank, roworder=hotelrankorder,
        palette = gray((2:0 / 2)^0.5))
title(sub="Variable sorted by correlation to Occupation", cex.sub=1.4, line=1)
par(oldpar)
```





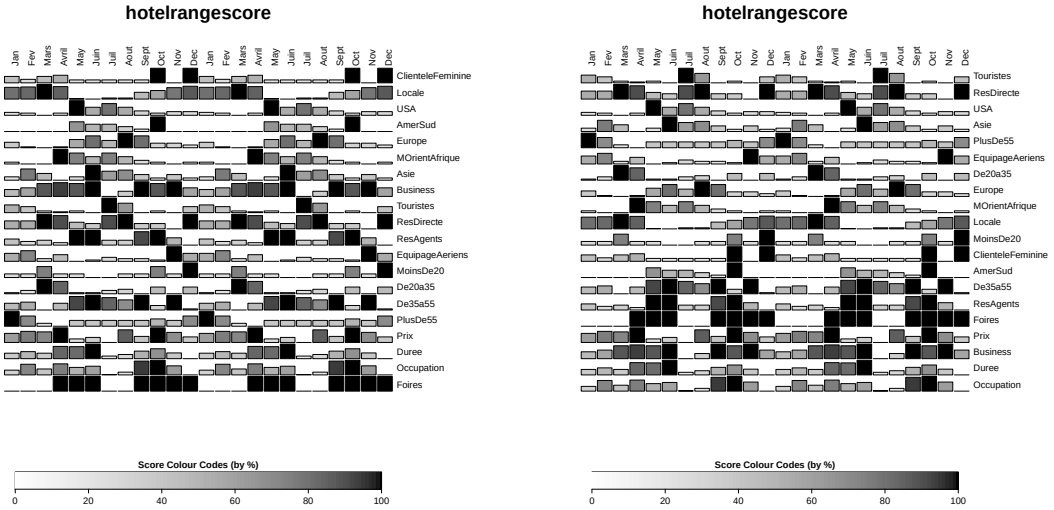
Variable sorted by correlation to Occupation

4.2. **z Scores.**

$$x \mapsto \frac{x - mean(x)}{sd(x)}$$

Input

```
oldpar <-par(mfcol=c(1,2))
hotelzscore <- bertinzscore(Hotel2)
plot.bertin(hotelzscore)
hotelzscoreorder <- bertin:::ordercor(hotelzscore, 19)
plot.bertin(hotelzscore, roworder=hotelzscoreorder)
par(oldpar)
```



Bertin uses to highlight "above average" observations. If the data is not degenerate, this corresponds to *bertinzscore > 0*.

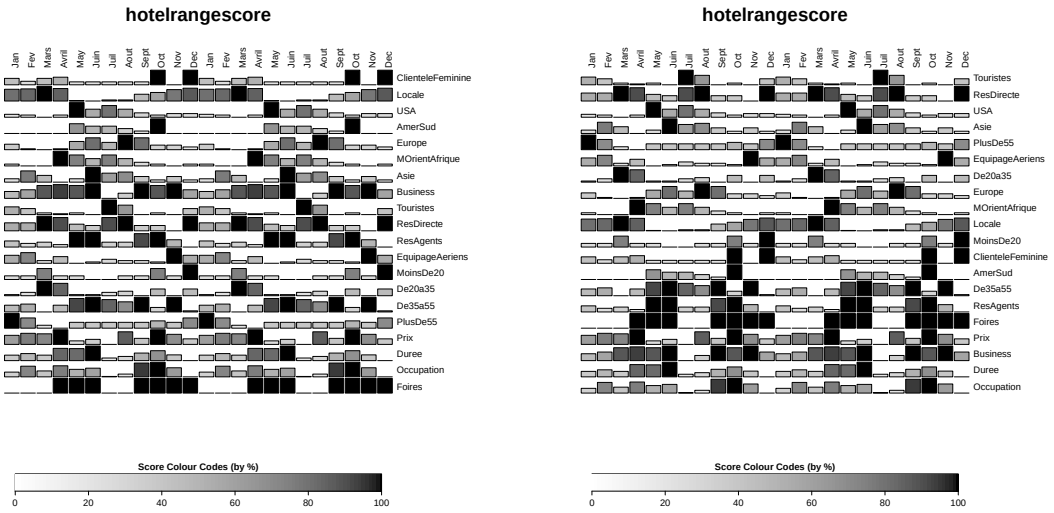
Since z scores center the variables around the mean, they require positive and negative values for display. This leads in effect to a reduction of the display space to a half, which should be compensated by a more expressive choice of colour coding.

4.3. **Range Scores.**

$$x \mapsto \frac{x - min(x)}{max(x) - min(x)}$$

Input

```
oldpar <-par(mfcol=c(1,2))
hotelrangescore <- bertinrangescore(Hotel2)
plot.bertin(hotelrangescore)
hotelrangescoreorder <- bertin:::ordercor(hotelrangescore, 19)
plot.bertin(hotelrangescore, roworder=hotelrangescoreorder)
par(oldpar)
```



This score just rescales to $[0, 1]$. In contrast to ranks, it preserves quantitative proportions. See Figure 8 on page 10.

5. Permutation, Seriation, Arrangement

As Bertin has pointed out,

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

Once we have solved a problem in data analysis, the problem can often be formulated as an optimisation problem. This is the end of the analysis process. In the beginning, while we are searching for a solution, experimenting is necessary. In our implementation, we separate two aspects.

Finding an adequate display is one. This amounts to building up a collection of proven models. A specific data set at hand can contribute by hinting at specific needs and simplify model choice. Building a collection of models and model choice is repeated

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

The other aspect is to identify critical information. The display, or graphical method, is only giving a framework, and it needs to be filled. For a chosen display, for example, we have to compare different arrangements (seriations, for example). If we allow for interactive work, flexibility has priority. We try to cache the information that is invariant of the permutation.

The classic Bertin display shown above is one of the examples to represent certain models. Following the ideas, but deviating in the details, is to use a simple grey scale image for representation. This may be not the most informative variant. But it is most economic in the use of display space (Figure 10 on page 18).

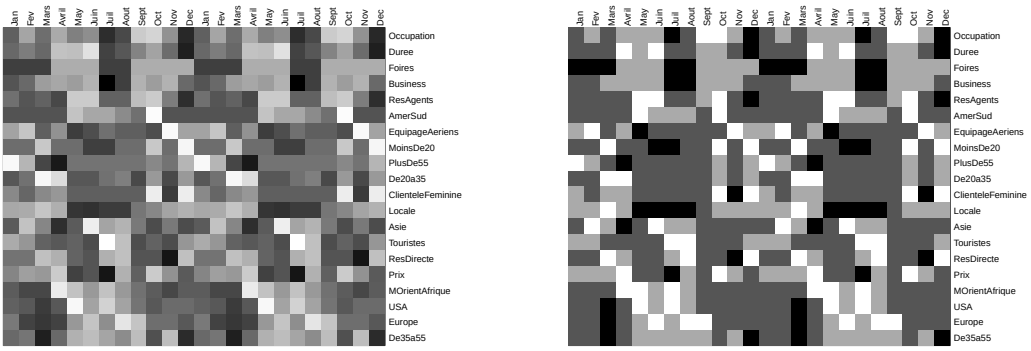


Figure 10: Display of a data matrix as grey scale image: **Hotel data**. Variables are rearranged by similarity to occupation and duration. Grey scale with 256 on the left, reduced grey scale with 4 steps on the right, applied to `zscores`.

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 a series of detail displays. Any display calibration however should be constant for this series. We try to allow for this global calibration (see Section 8 (page 51)).

So far, the interactive possibilities used in Voyager (Sawitzki, 1996) are not provided in the R implementation, and only rudimentary parts of the static arrangements are provided in R.

6. Patch Strategy

It may be useful to think of classical statistical methods and see how they may be reflected in a Bertin setting. For a start, regression and classification may serve as starting points. For both, traditional statistics has provided solutions, in the classical framework as least squares regression and as discriminance analysis. Many variations have followed.

Classification and regression trees (Breiman et al., 1984) have thrown a new light on these problems. In principle, CART is a higher dimensional strategy. In the Bertin context, we deliberately restrict ourselves to a two dimensional display context. CART suggests to look for "patches", rectangular areas that allow for an economic model. CART uses a simple strategy, splitting one variable at a point. However this leads to a fragmentation of the data material, dividing it on the average by a factor 2^k for k splits, and gives a corresponding penalty in terms of variance. Friedman (Friedman, 1996) has pointed out that this fragmentation is not necessary. Instead of one set of patches you can have two: one used for estimation, and a second one, possibly different, uses to apply the estimation for fitting. You want to keep the first one large to control variance, and the second one small to reduce bias.

This is not implemented explicitly in the R implementation, but we suggest it as a strategic guide. The examples provided in these notes follow this strategy.

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.

The high level routines accept the usual possibilities of R for subset and index manipulation. As a convenience, the indices are accessible as function arguments. Certain actions are provided as special functions. See the reference pages for details.

7. Colour, Perception and Pitfalls

Colour is one of the simplest qualities, and barely understood. One basis of colour is light, and if we follow todays physics this is best described as electromagnetic wave with energy distributed over a spectrum of wavelengths. So the mathematical home is some Hilbert space.

We do not know how light translates into perceived colour, but we know some of the steps, and we have some artefacts that may help us to understand the process. Today, the most helpful concept seems to be that of a colour space - a finite dimensional space mapped into the true space of colours.

In a technical context as we see it today, colour became manageable with colour displays using phosphors, coming in three colours (red, green and blue). The relative

intensities of these colours are enough to specify the visual impact, and the RGB space is still the basis space for colours presented on a display.

Of course there is a huge discrepancy between the full Hilbert space of lights, and three dimensional RGB space. The gap is bridged by perception. Different spectral distributions may give the same perceived colours. By mixing only few colours, you can generate the impression to nearly cover a spectrum. Colours thus matched perceptually are called metamers.

This leads to an experimental setup. Take the space of single frequency colours, i.e. monochrome light. For each frequency, find a combination of red, green and blue to match this colour.

What they tell us in school is that all colours can be mixed from three basic colours. No, note quite. Not even for monochrome light. For many frequency, the impression can be generated by an RGB combination. For several colours in the green range, it is necessary to reduce the colour by a complement before it is accessible in RGB space. See `<http://www.statlab.uni-heidelberg.de/data/color/>`. However the mapping from monochromous colour space to RGB space is fairly understood and allows us to generate (within limits) the colour of any frequency if we can control RGB values.

Technology requires differing spaces. While for displays we originally had three colours encoded in RGB, for printing the readily available dyes ware cyan, magenta and yellow, defining a CMY space. Fortunately these dyes could be calibrated to be roughly complimentary to RGB, so we think of a colour cube with pure R, G, B on three corners, C, M, Y on the opposite corners and an easy translation from RGB to CMY coordinates. Today, quality colour printing is starting with six dyes, and a more complex colour space is standard.

Still these technical colours are far away of the complexity of colours of light, and the use of metamers is the way to cope with this discrepancy.

Colour spaces are just a means to specify colours, and the technical colour spaces are a rough approximation of the possibilities. If we go from coloured light to perceived colours, we have some information. Coloured light must be perceived by receptors, and as of today it seems that the human eye has four kinds of receptors. There is a group of receptors for black and white (or light intensity), the rods. Another group of receptors, the cones, has a frequency dependent sensitivity, and the sensitivity characteristics is known. An additional group seems to be only receptive for light/darkness. So colour perception is effectively channelled through three colour channels. From the analysis of colour deficiency, a model of the interaction of these channels can be derived `<http://www.statlab.uni-heidelberg.de/data/color/background.html>`.

The distribution of these receptors is not homogeneous. It is a mental reconstruction to perceive a homogeneous colour of some area. The is not sustained by primary reception.

Colour is perceived as a quality, and it is an open issue how to represent quantitative information by colour. One basic aspect has been extensively studied: how to find a colour scheme that allows to represent equidistant information by apparently equidistant colours. Several suggestions have been developed, and the current consensus is the CIELAB colour space. If you want to represent information on an interval scale by colour, mapping it to CIELAB space is the first choice. However, very often the data give information on a scale suggested by the measurement process available. An interval scale may be the exception rather than the rule, and finding an appropriate scale may be part of the challenge.

Colour may need some experiments to find an informative choice. To allow easy experiments, we use a two step procedure. Based on the original data or the score, we derive a colour index. From an abstract point of view, this is just another score with values in $1, \ldots, nrColours$. Colours are provided as a colour palette, and in the second step the index is used to select the colour to apply.

See `help(palette)` for information on colour palettes. `library(colorspace)` provides translations between several common colour spaces.

Some colour palettes that may be useful for Bertin displays are provided. For some palettes, there are variants to highlight the tail behaviour. They are marked by a "2" for quadratic and a "4" for a quartic flattening around the mid value which is mapped to white.

There is an abundant literature on choosing colour palettes for the visual display of quantitative information. Please read.

Be aware that not all people perceive colour the same way. In particular, if you are using red and green in you colour palette, about 6% of the male population will have difficulties.

Do trivial tests. If you have a smart colour scale: give a sample to some friends and ask them to tell which shoes higher and which shows lower values.

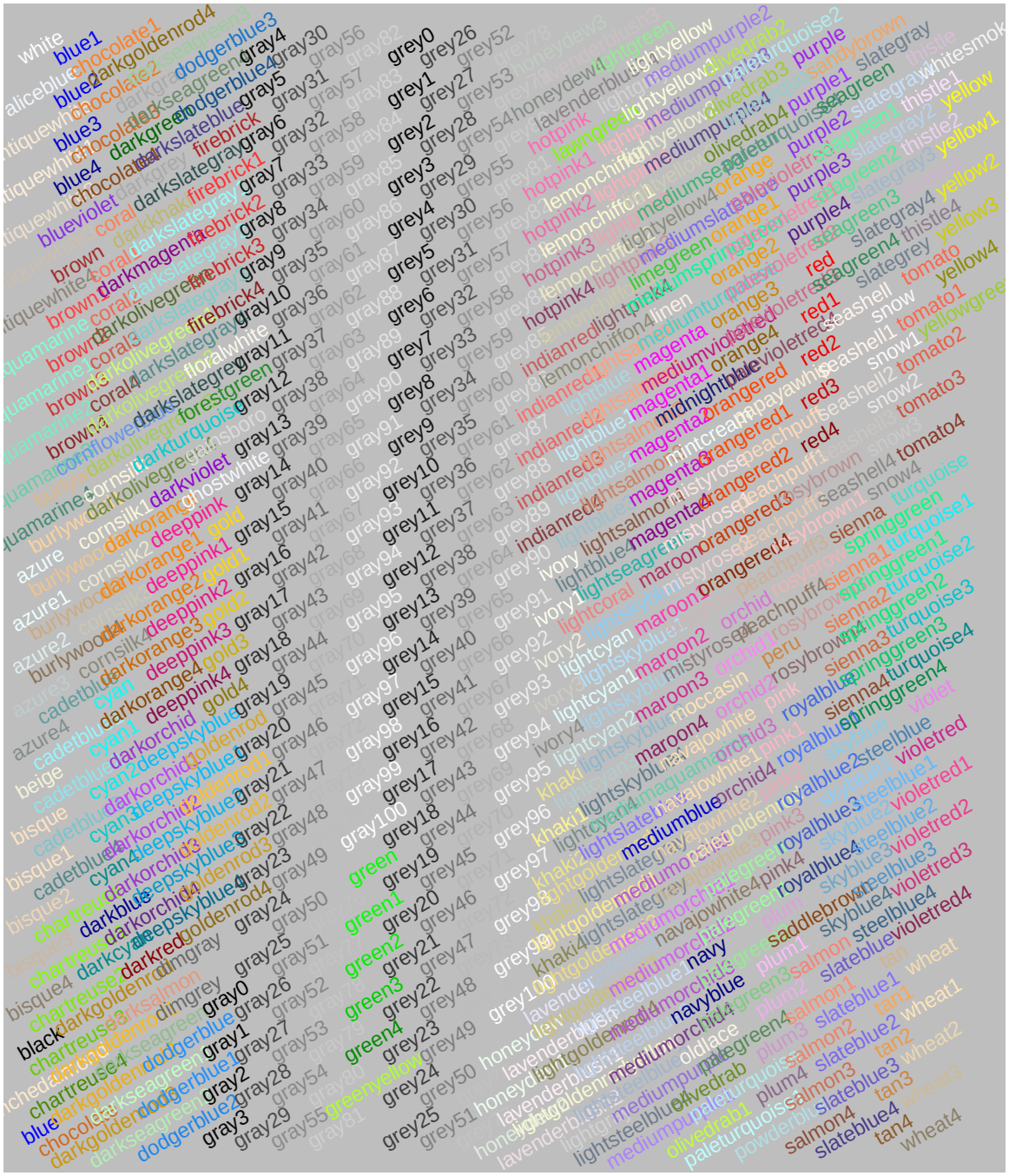
Colour displays are a means to convey some information. Check that your choices serve this purpose. In particular: if it is of importance to recognise high or low values, chose a colour scale that does not focus on differences in neutral values.

7.1. **Colours.** The basic colour specifications in R refer to the standard sRGB colorspace (IEC standard 61966). (R 2.15. This may change in future versions.) Several colours have predefined names. These color names can be used with a `col=` specification in graphics functions. An even wider variety of colors can be created with primitives `rgb`, `hsv`, or `hcl`.

Note: not all colours may be printable, or have a faithful representation on the screen. The range of visible colours may vary with the graphics systeem used.

Input

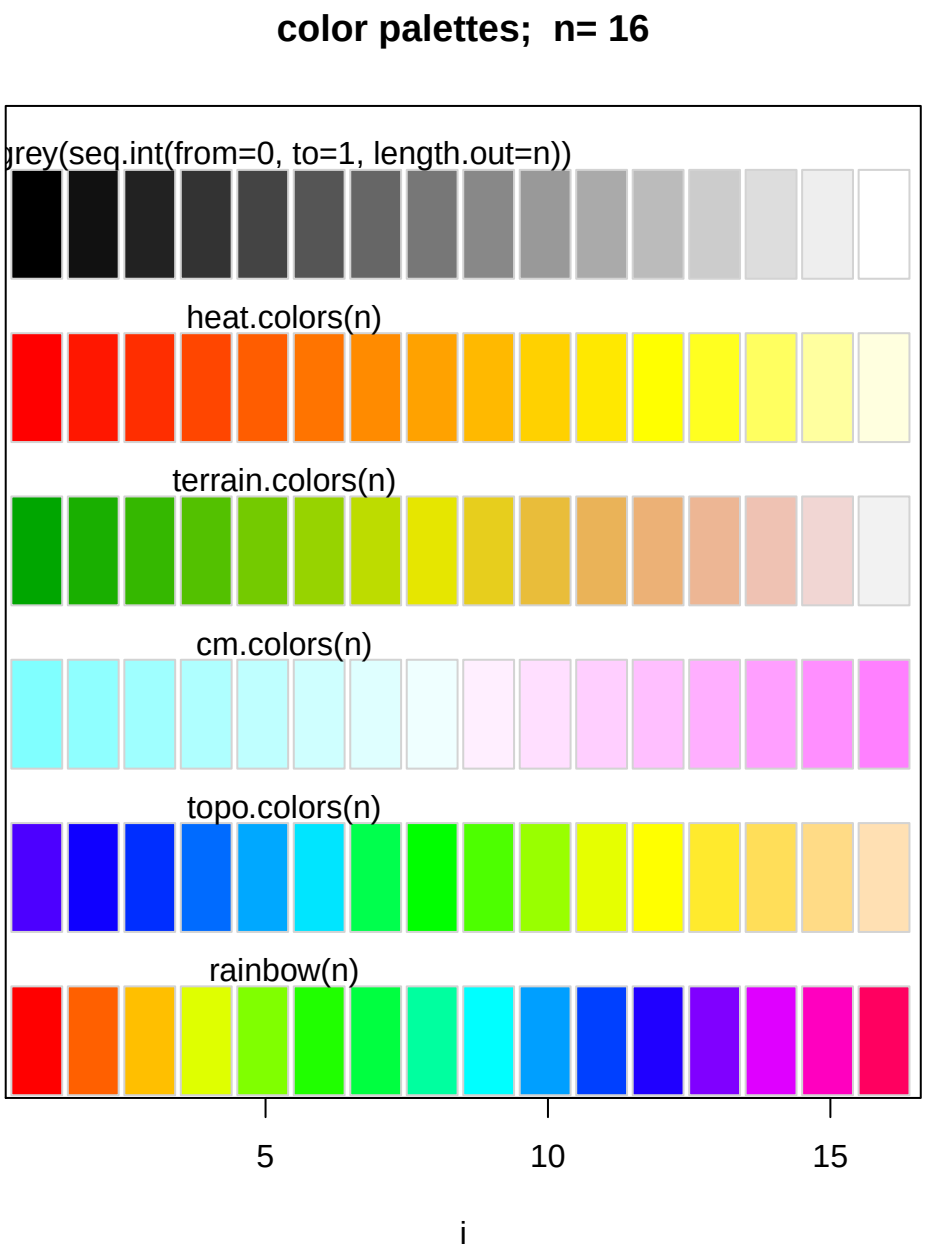
```
example(colors, package="grDevices")
```



7.2. **Colour Palettes.** R provides six basic colour palettes.

Input

```
example(rainbow, package="grDevices")
```



grey and *heat.colors* palettes can be used to represent sequential quantitative data,ormore general ordinal data. Note the different calling conventions. *grey* allows to specify breakpoints on $[0, 1]$. *heat.colors* only allows to specify the number of colours.

cm.colors, *terrain.colors*, and *topo.colors* allow to represent data between two poles. Sometimes, tese are called divergent scales.

rainbow is not useful for encoding quantitative data, or to give any reliable representation of qualitative data. They should be rather be considered as an example of what

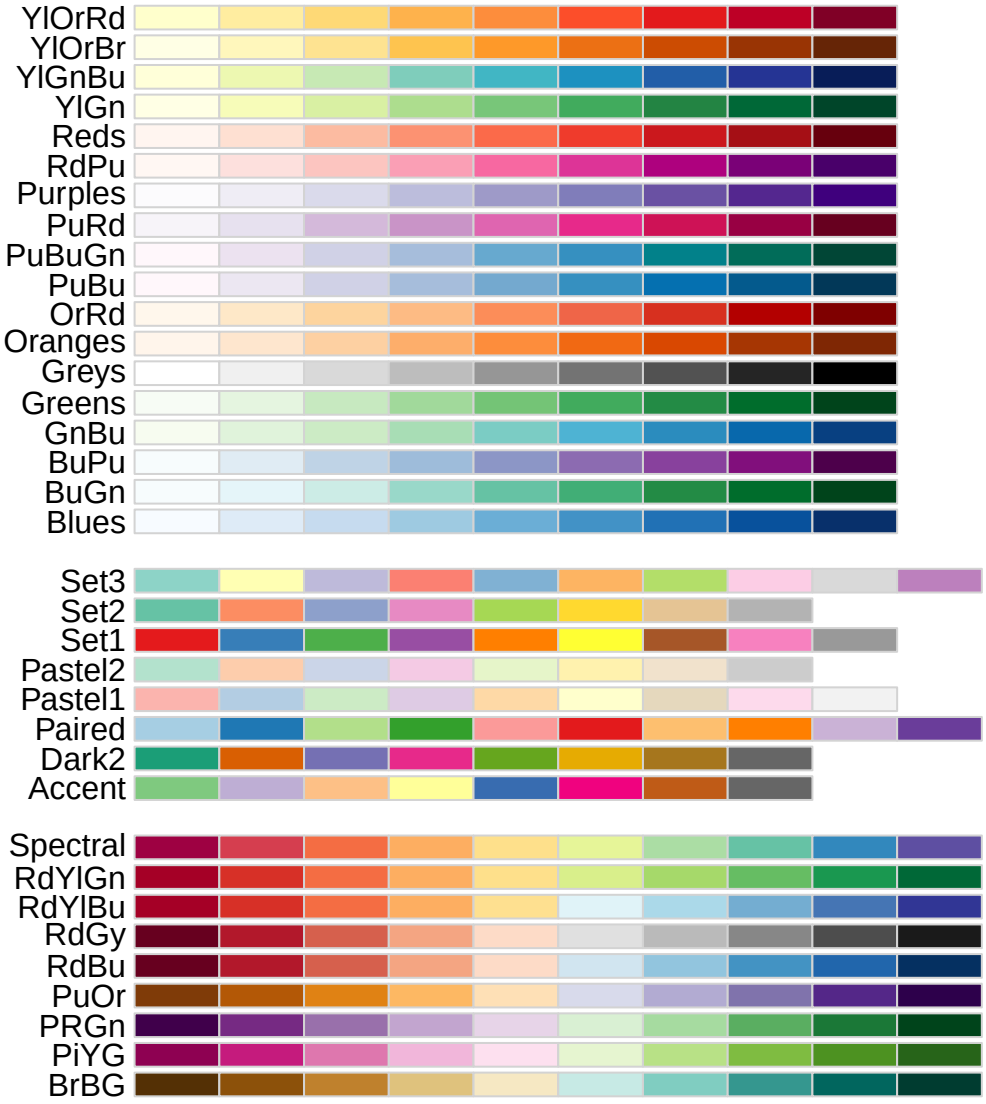
can be done, and as a starting point for the selection of colours for more reasonable scales.

7.2.1. *Brewer Palettes.* For categorical data, a challenge is to find colours that are visually clearly separated. Other requirements may apply, as to have "balanced" palettes: no colours should have an excessive visual impact in comparison to others.

Cynthia Brewer et al. suggested a small collection of palettes with hand seleccted colors for this task, and additional tools for selecting a palette for a specific purpuse are available at `http://www.colorbrewer.org`.

Input

```
library("RColorBrewer")
display.brewer.all(n=10, exact.n=FALSE)
```


YlOrRd
YlOrBr
YlGnBu
YlGn
Reds
RdPu
Purples
PuRd
PuBuGn
PuBu
OrRd
Oranges
Greys
Greens
GnBu
BuPu
BuGn
Blues
Set3
Set2
Set1
Pastel2
Pastel1
Paired
Dark2
Accent
Spectral
RdYlGn
RdYlBu
RdGy
RdBu
PuOr
PRGn
PiYG
BrBG

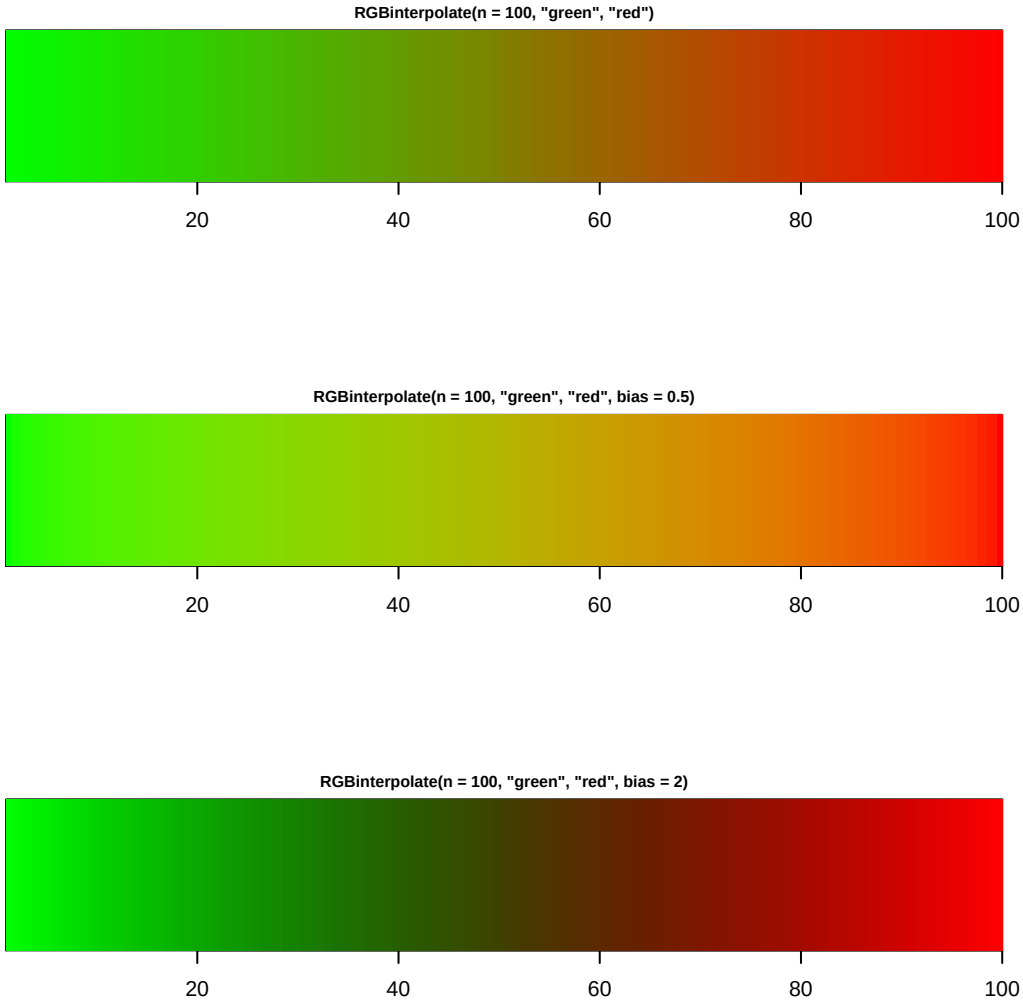
7.2.2. *Home Made Palettes.* Colour palettes can be constructed on the fly, or adapted tocertain purposes. If you do not need to doit: don't. Try to get familiar with the perceptual aspects of some well-established colour palettes, and try to adhere to these.

If you need to create your own palette, test it. What is the effect of this palette under your "null model"? What is the discriminative power?

If your have to create your own palette, the basic tool is interpolation. It depends on yourview of the colour space. Here is linear interpolation in RGB space, with some variants.

Input

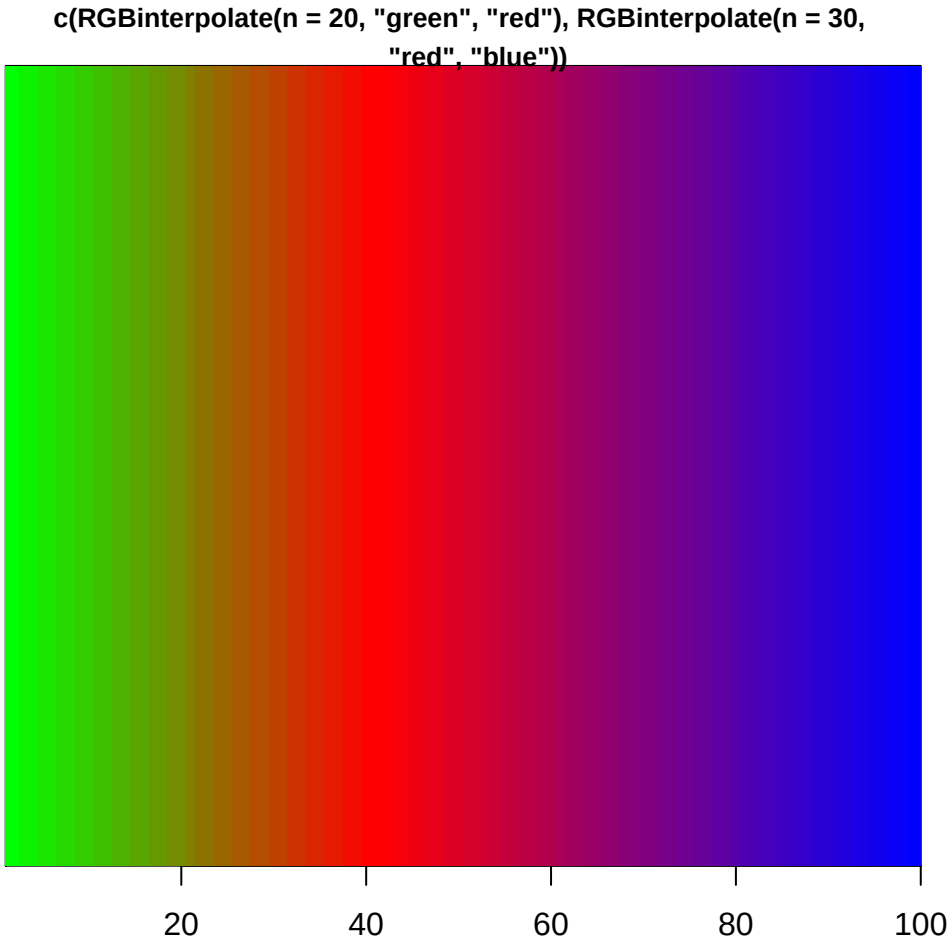
```
oldpar <- par(mfrow=c(3,1))
        colramp(RGBinterpolate(n=100,"green","red"))
        colramp(RGBinterpolate(n=100,"green","red", bias=0.5))
        colramp(RGBinterpolate(n=100,"green","red", bias=2.0))
par(oldpar)
```



You can subselect palettes to focus on parts which have sufficien discriminative power, and combine palettes by concatinating them.

Input

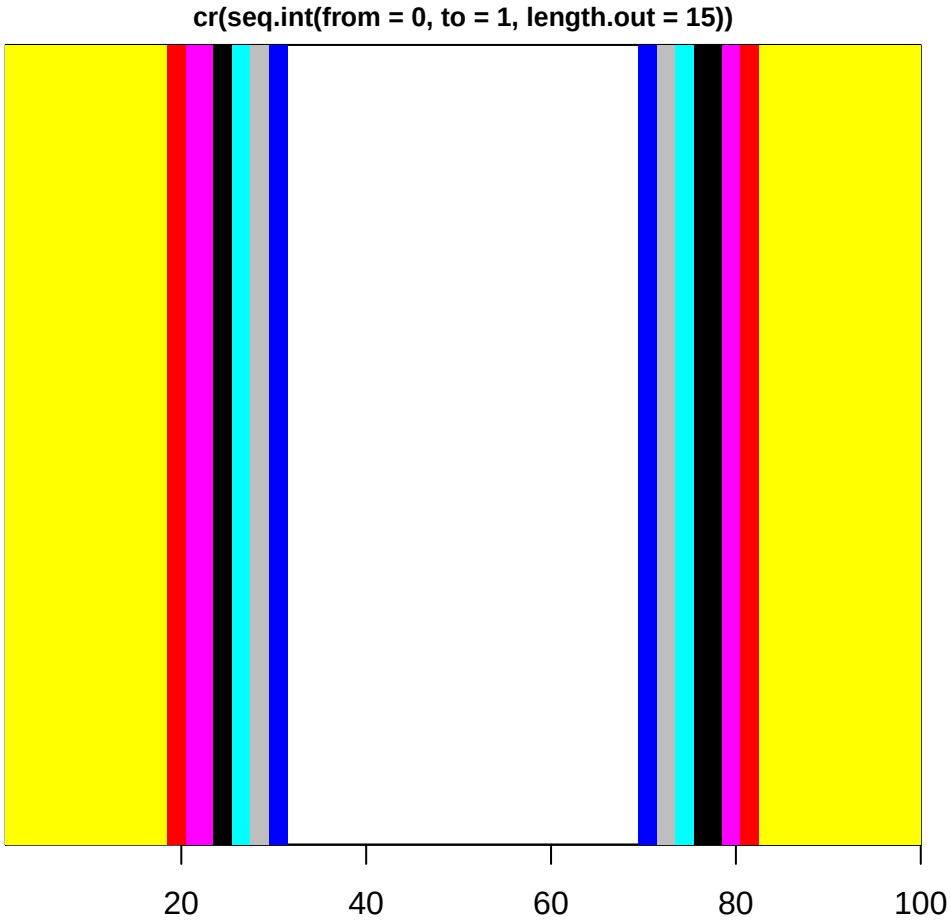
```
colramp(c(RGBinterpolate(n=20,"green","red"),RGBinterpolate(n=30,"red","blue")))
```



For interpolation, piecewise linear interpolation and spline iinterpolation are obvious choices, and both can be applied to more than two nodes to interpolate. R provides *colorRamp* to generate and interpolating function

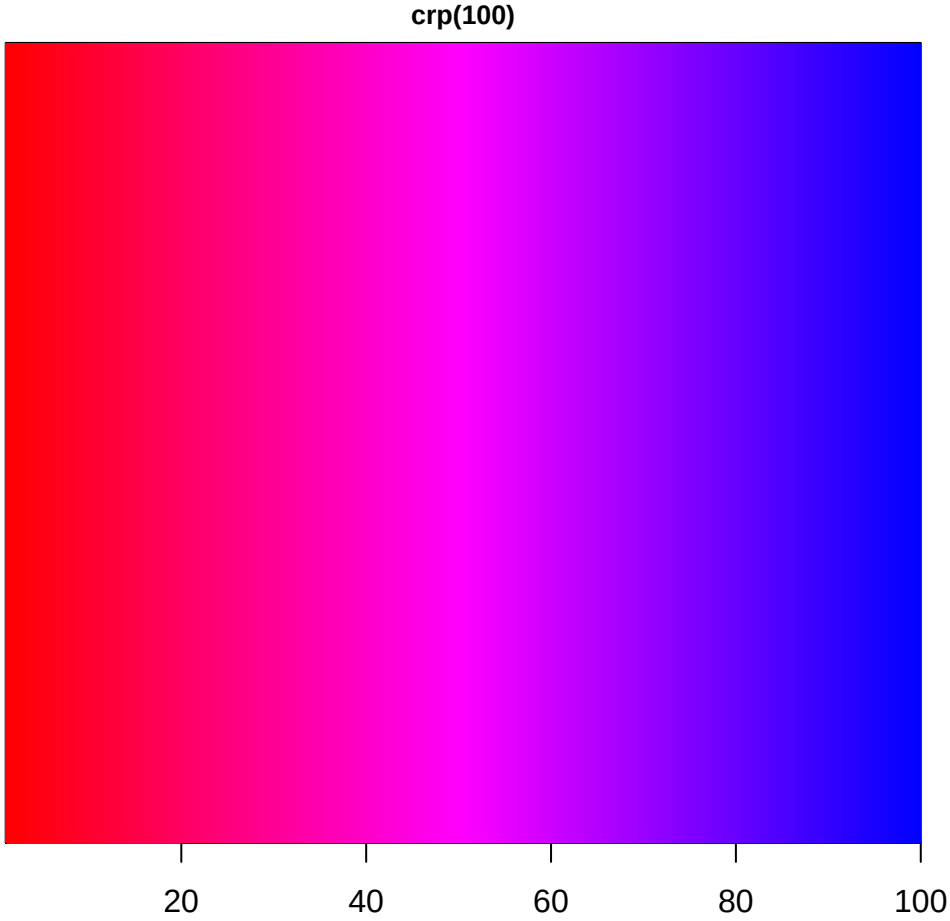
Input

```
mycolors <- c("red", "magenta", "blue")
cr <- colorRamp(mycolors)
colramp(cr( seq.int(from=0, to=1, length.out=15)))
```



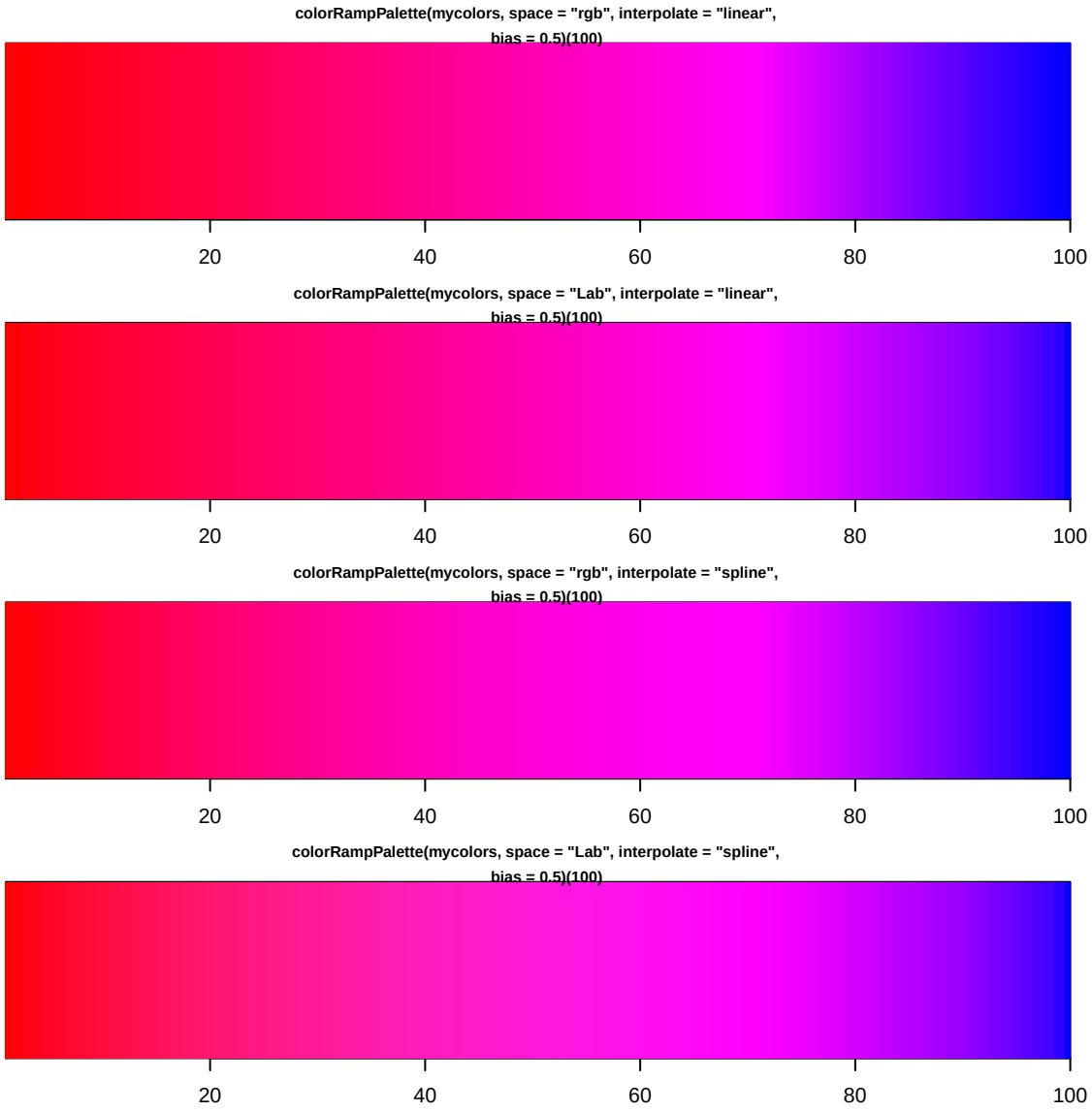
Input

```
mycolors <- c("red", "magenta", "blue")
crp <- colorRampPalette(mycolors)
colramp(crp(100))
```



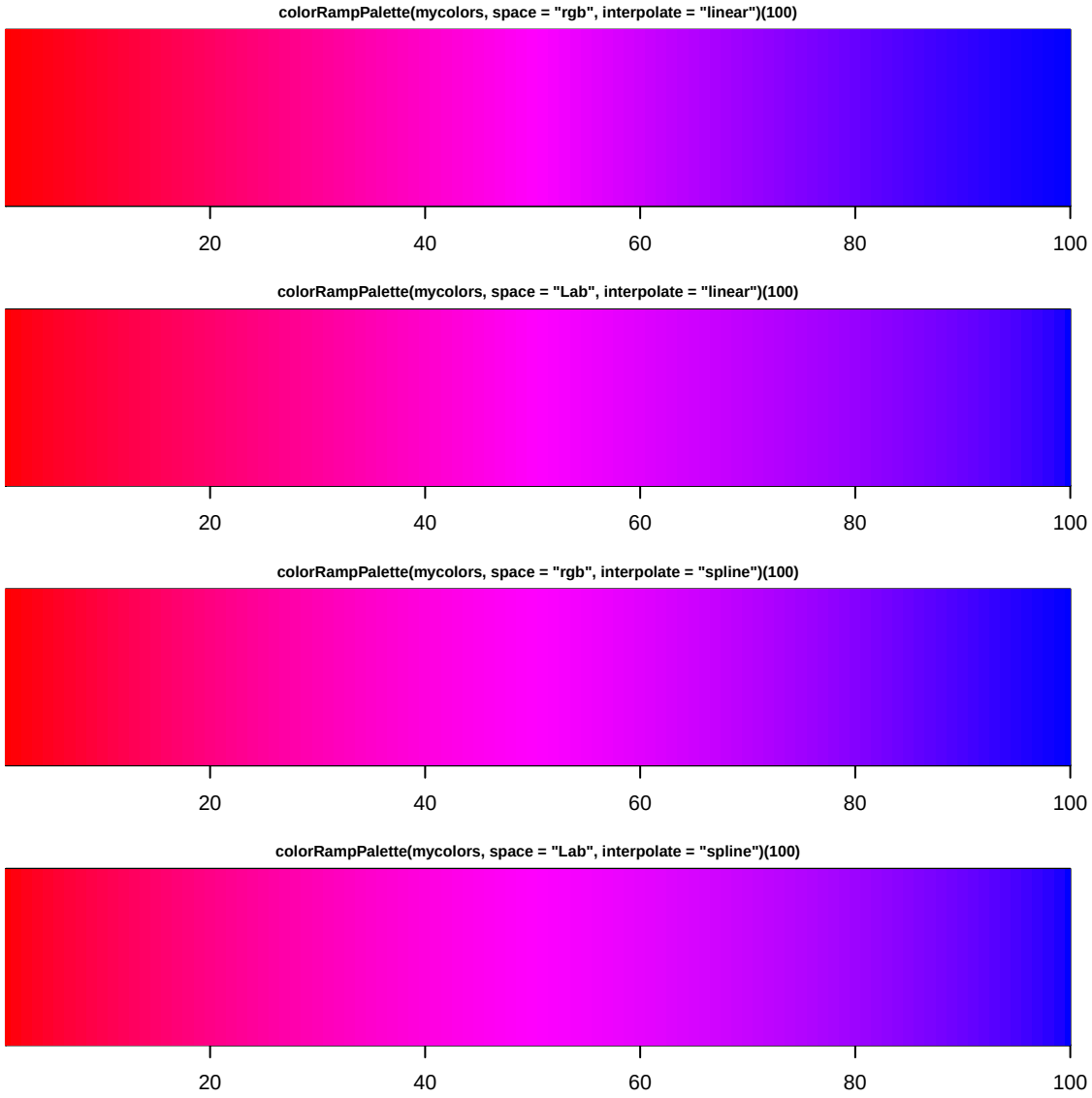
Input

```
oldpar <- par(mfrow=c(4,1), mar=c(2, 1, 2, 1) + 0.1)
mycolors <- c("red", "magenta", "blue")
colramp(colorRampPalette(mycolors, space="rgb", interpolate="linear", bias=0.5)(100))
colramp(colorRampPalette(mycolors, space="Lab", interpolate="linear", bias=0.5)(100))
colramp(colorRampPalette(mycolors, space="rgb", interpolate="spline", bias=0.5)(100))
colramp(colorRampPalette(mycolors, space="Lab", interpolate="spline", bias=0.5)(100))
par(oldpar)
```



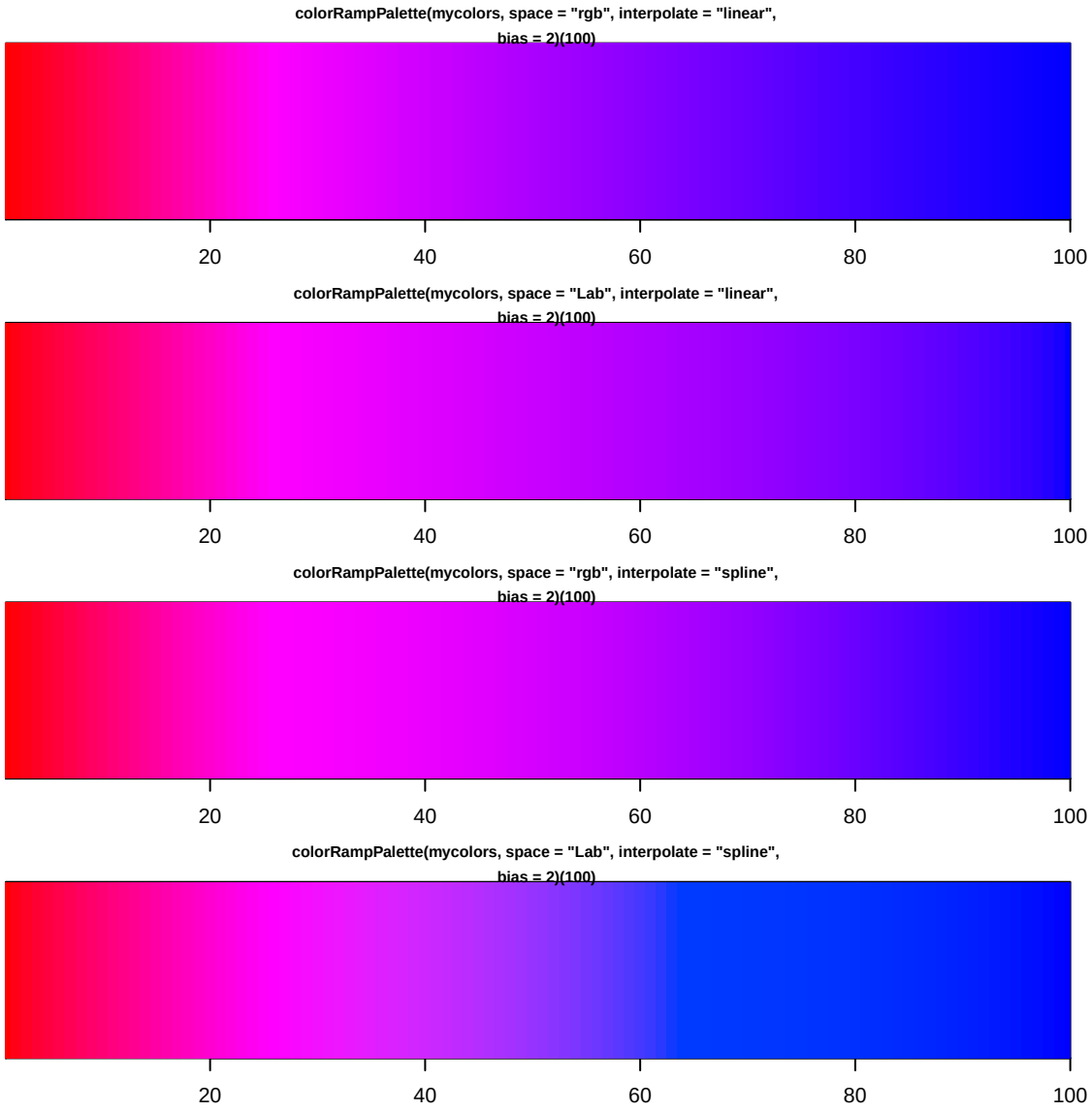
Input

```
oldpar <- par(mfrow=c(4,1), mar=c(2, 1, 2, 1) + 0.1)
mycolors <- c("red", "magenta", "blue")
colramp(colorRampPalette(mycolors, space="rgb", interpolate="linear")(100))
colramp(colorRampPalette(mycolors, space="Lab", interpolate="linear")(100))
colramp(colorRampPalette(mycolors, space="rgb", interpolate="spline")(100))
colramp(colorRampPalette(mycolors, space="Lab", interpolate="spline")(100))
par(oldpar)
```



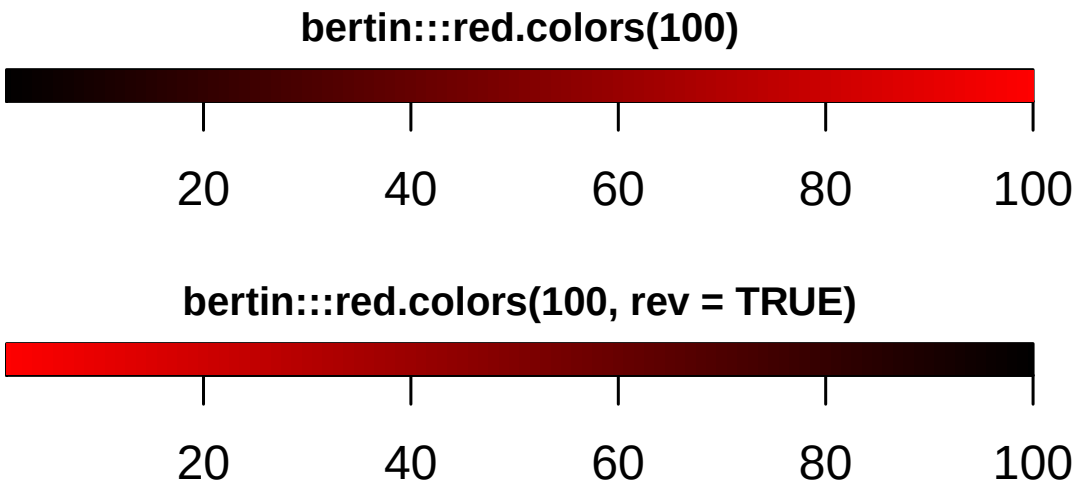
Input

```
oldpar <- par(mfrow=c(4,1), mar=c(2, 1, 2, 1) + 0.1)
mycolors <- c("red", "magenta", "blue")
colramp(colorRampPalette(mycolors, space="rgb", interpolate="linear", bias=2)(100))
colramp(colorRampPalette(mycolors, space="Lab", interpolate="linear", bias=2)(100))
colramp(colorRampPalette(mycolors, space="rgb", interpolate="spline", bias=2)(100))
colramp(colorRampPalette(mycolors, space="Lab", interpolate="spline", bias=2)(100))
par(oldpar)
```

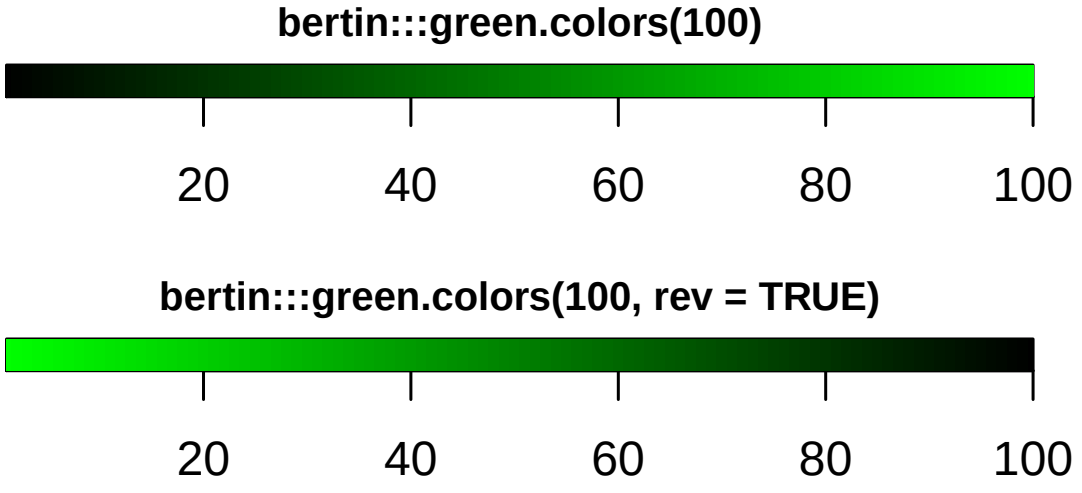

colorRampPalette(mycolors, space = "rgb", interpolate = "linear",
bias = 2)(100)
20
40
60
80
100
colorRampPalette(mycolors, space = "Lab", interpolate = "linear",
bias = 2)(100)
20
40
60
80
100
colorRampPalette(mycolors, space = "rgb", interpolate = "spline",
bias = 2)(100)
20
40
60
80
100
colorRampPalette(mycolors, space = "Lab", interpolate = "spline",
bias = 2)(100)
20
40
60
80
100

7.2.3. *Betin Colour Palettes.* The calling structure for all Bertin colour palettes in this section is the same. Here are just two examples.

Input
```
oldpar <- par(mfrow=c(2,1), mar=c(2, 1, 2, 1) + 0.1)
colramp(bertin:::red.colors(100), horizontal=TRUE)
colramp(bertin:::red.colors(100, rev=TRUE),horizontal=TRUE)
par(oldpar)
```



Input
```
oldpar <- par(mfrow=c(2,1), mar=c(2, 1, 2, 1) + 0.1)
colramp(bertin:::green.colors(100))
colramp(bertin:::green.colors(100, rev=TRUE))
par(oldpar)
```

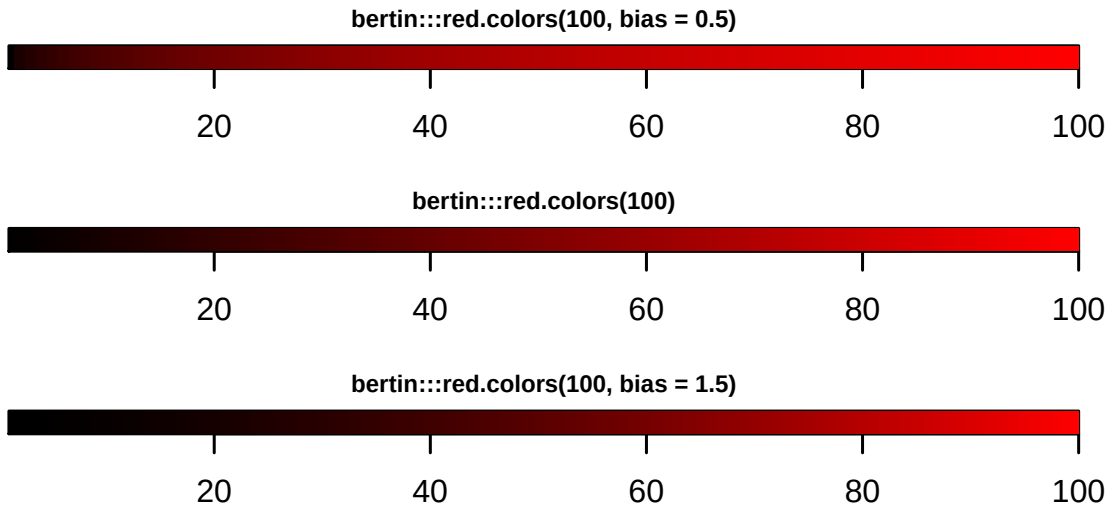


In RGB space, each palette has natural variants corresponding to gamma corrections. Standardised on $[0, 1]$, gamma correction replaces an intensity i by i^γ. This is in analogy to the channel gamma correction used technically. Fallowing hte terminology used in R, the transformation parameter is called `bias`.

Note: it is up to discussion, whether γ should modify the input intensity, or the channel response. This may change.

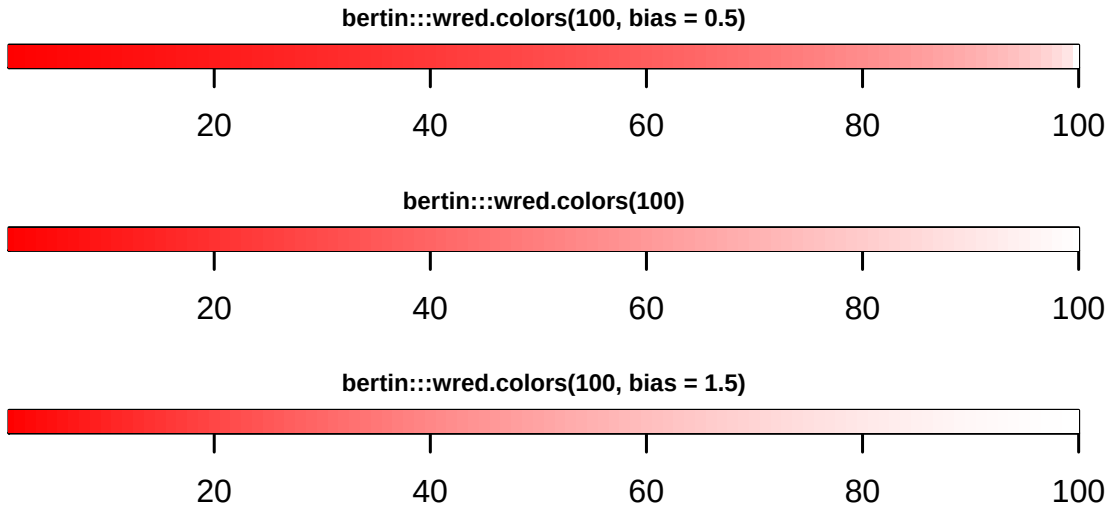
Input

```
oldpar <- par(mfrow=c(3,1), mar=c(2, 1, 2, 1) + 0.1)
colramp(bertin:::red.colors(100, bias=0.5))
colramp(bertin:::red.colors(100))
colramp(bertin:::red.colors(100, bias=1.5))
par(oldpar)
```

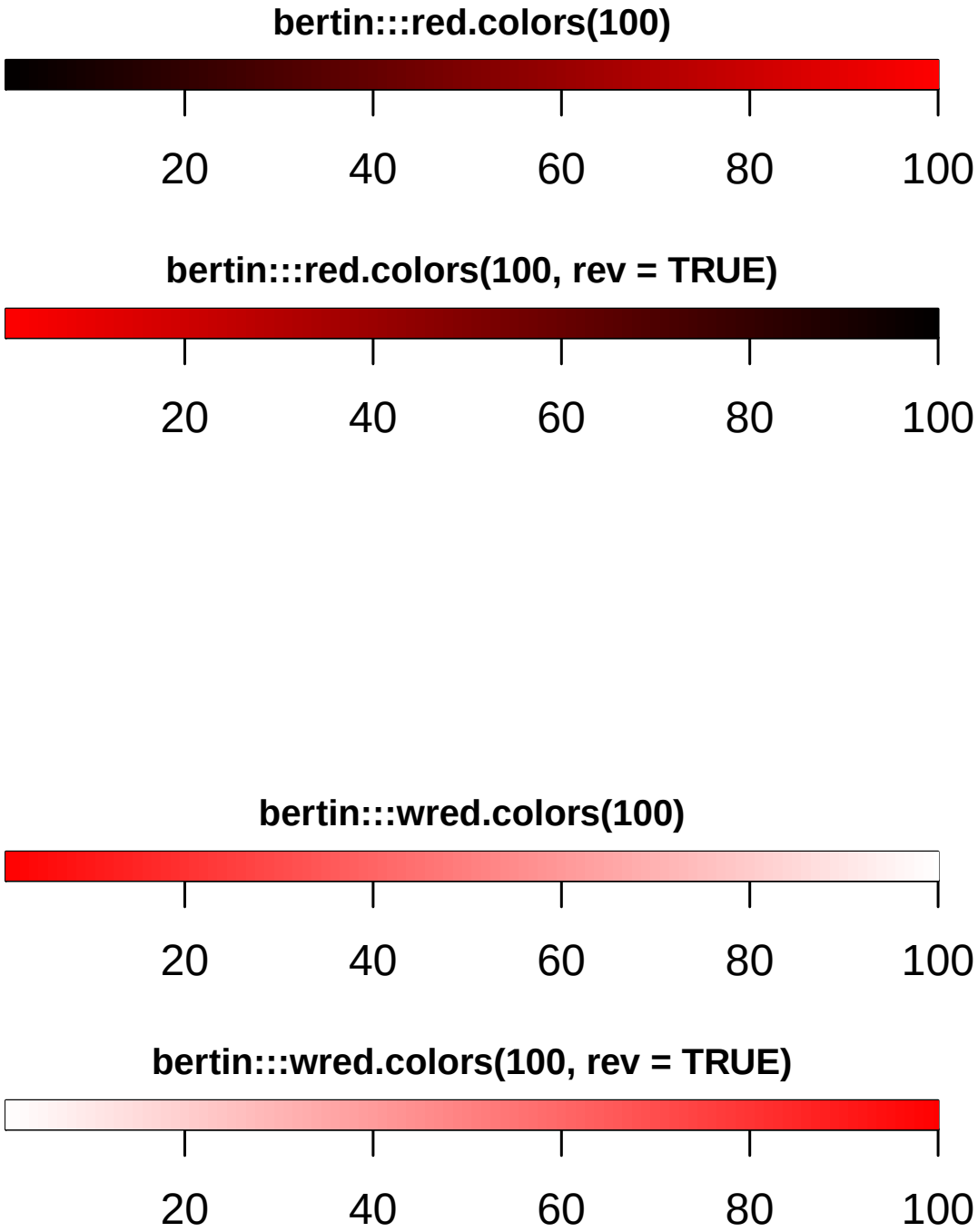


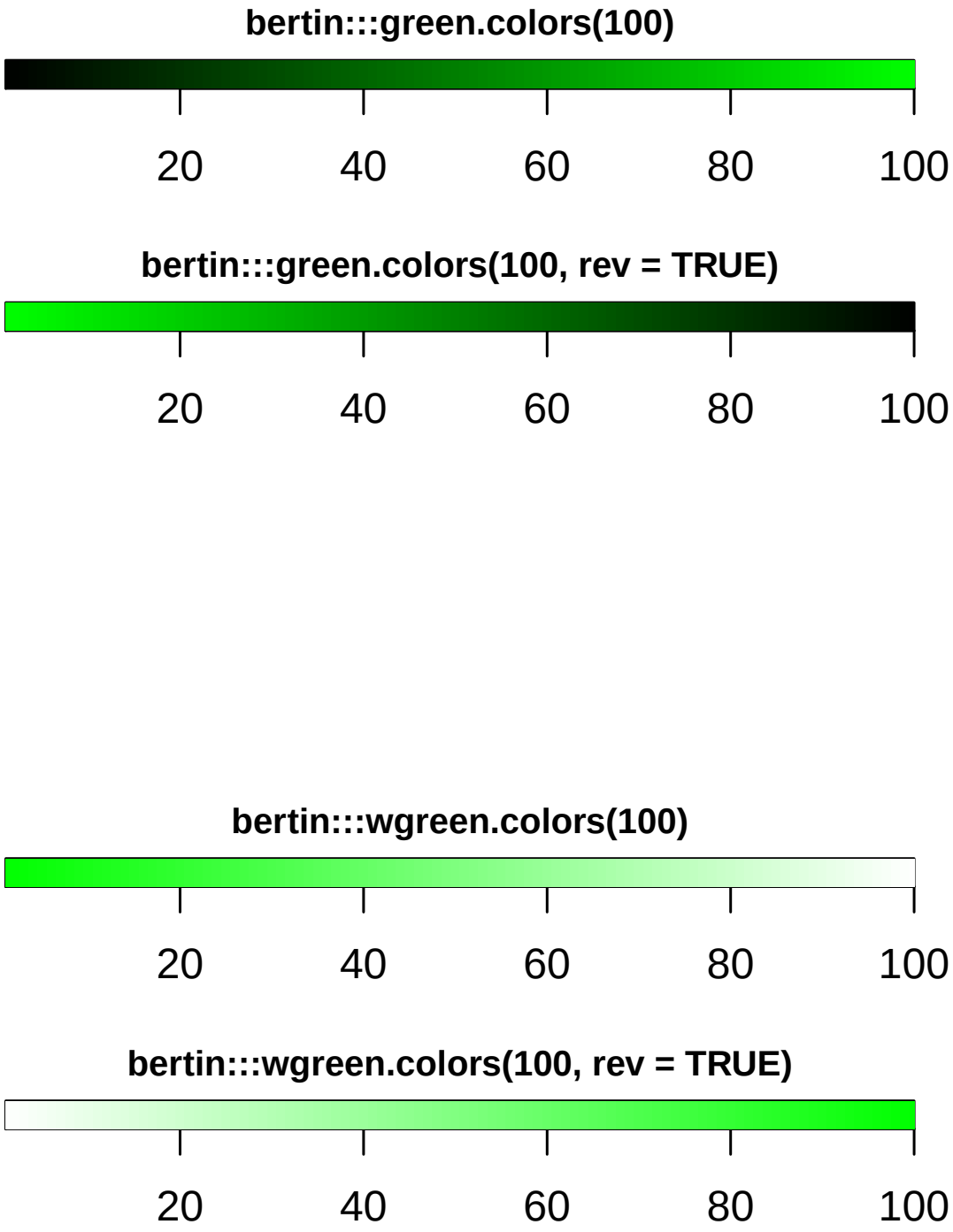
Input

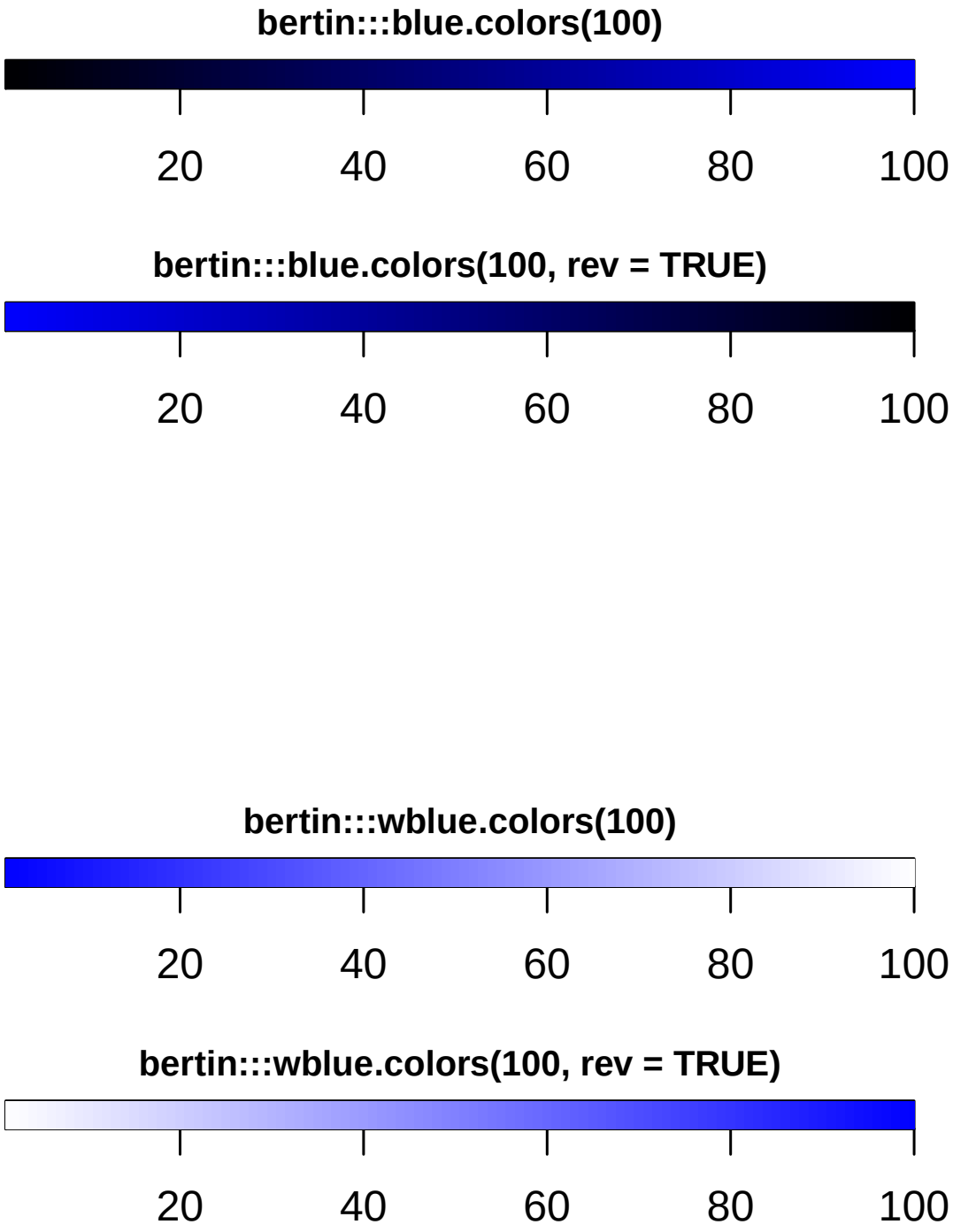
```
oldpar <- par(mfrow=c(3,1), mar=c(2, 1, 2, 1) + 0.1)
colramp(bertin:::wred.colors(100, bias=0.5))
colramp(bertin:::wred.colors(100))
colramp(bertin:::wred.colors(100, bias=1.5))
par(oldpar)
```

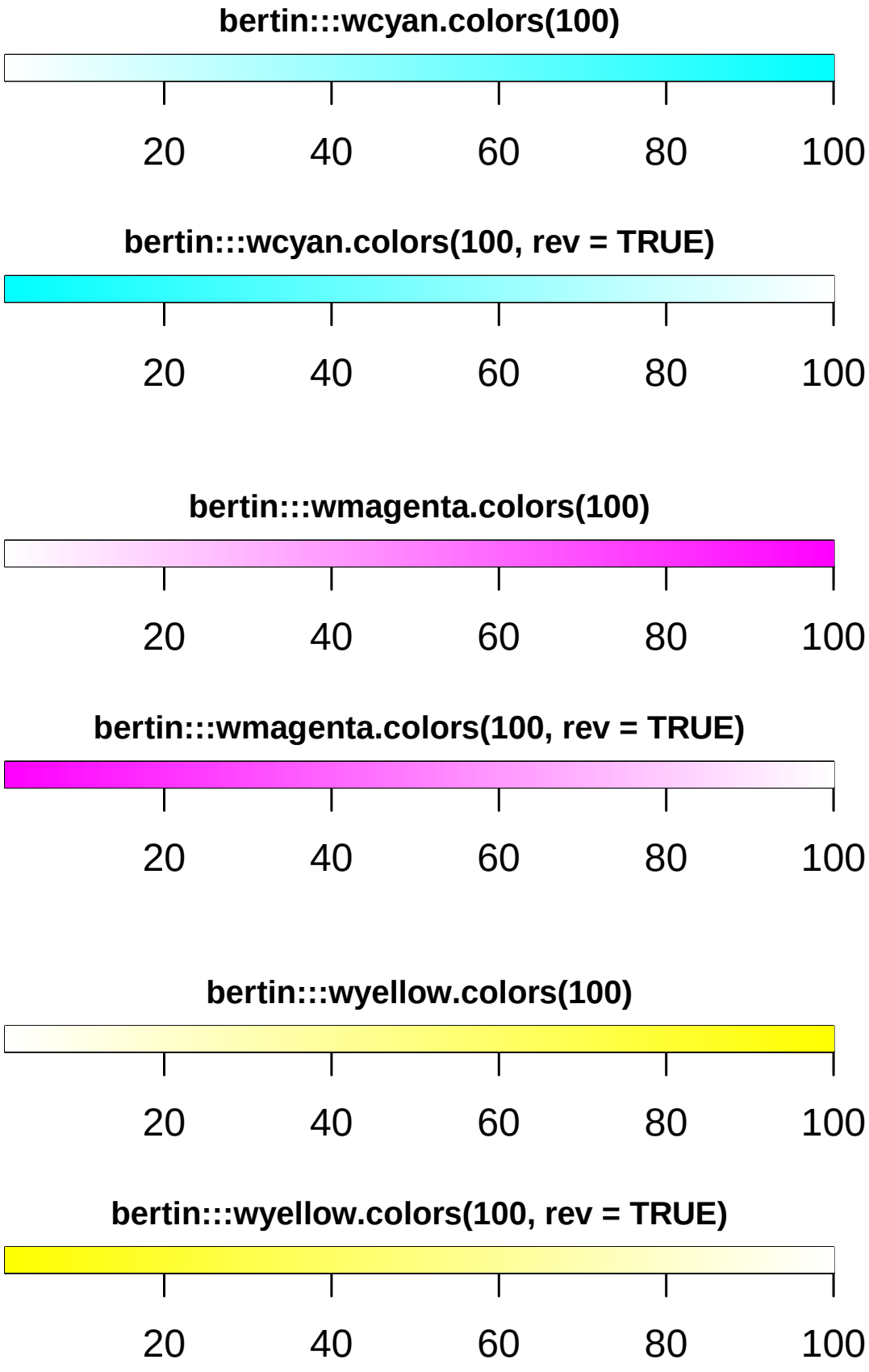

bertin:::wred.colors(100, bias = 0.5)
20
40
60
80
100
bertin:::wred.colors(100)
20
40
60
80
100
bertin:::wred.colors(100, bias = 1.5)
20
40
60
80
100

7.2.4. *Basic Colour Palettes.*



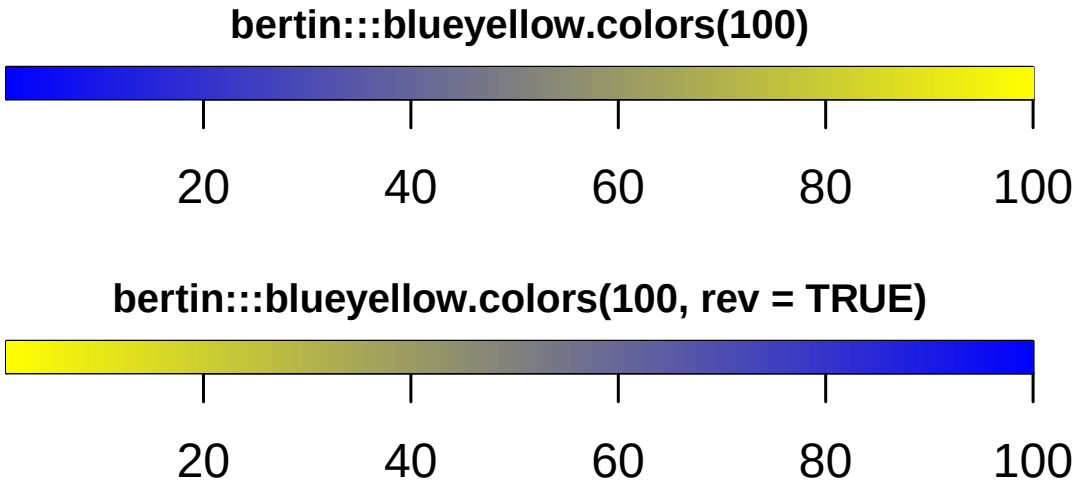

bertin:::green.colors(100)
20
40
60
80
100
bertin:::green.colors(100, rev = TRUE)
20
40
60
80
100
bertin:::wgreen.colors(100)
20
40
60
80
100
bertin:::wgreen.colors(100, rev = TRUE)
20
40
60
80
100


bertin:::blue.colors(100)
20
40
60
80
100
bertin:::blue.colors(100, rev = TRUE)
20
40
60
80
100
bertin:::wblue.colors(100)
20
40
60
80
100
bertin:::wblue.colors(100, rev = TRUE)
20
40
60
80
100

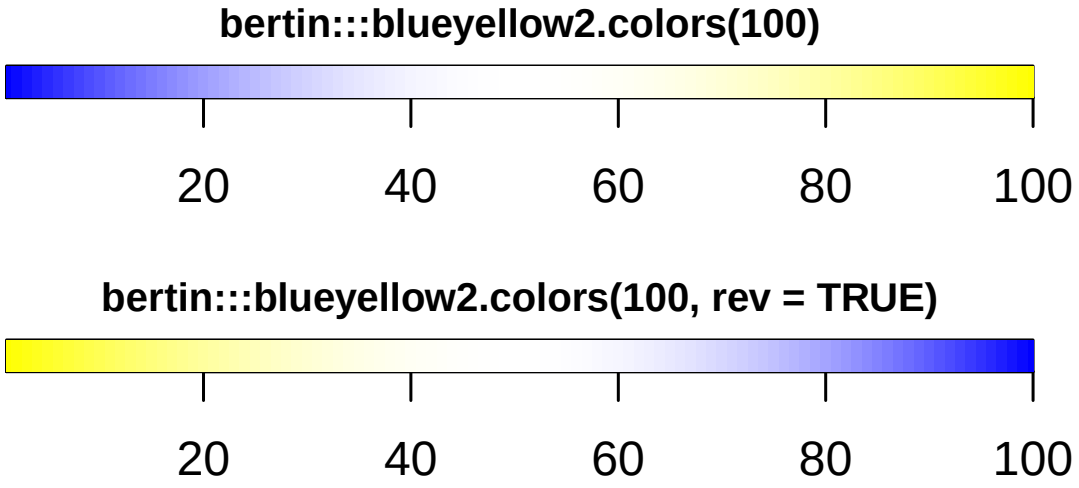

bertin:::wcyan.colors(100)
20
40
60
80
100
bertin:::wcyan.colors(100, rev = TRUE)
20
40
60
80
100
bertin:::wmagenta.colors(100)
20
40
60
80
100
bertin:::wmagenta.colors(100, rev = TRUE)
20
40
60
80
100
bertin:::wyellow.colors(100)
20
40
60
80
100
bertin:::wyellow.colors(100, rev = TRUE)
20
40
60
80
100

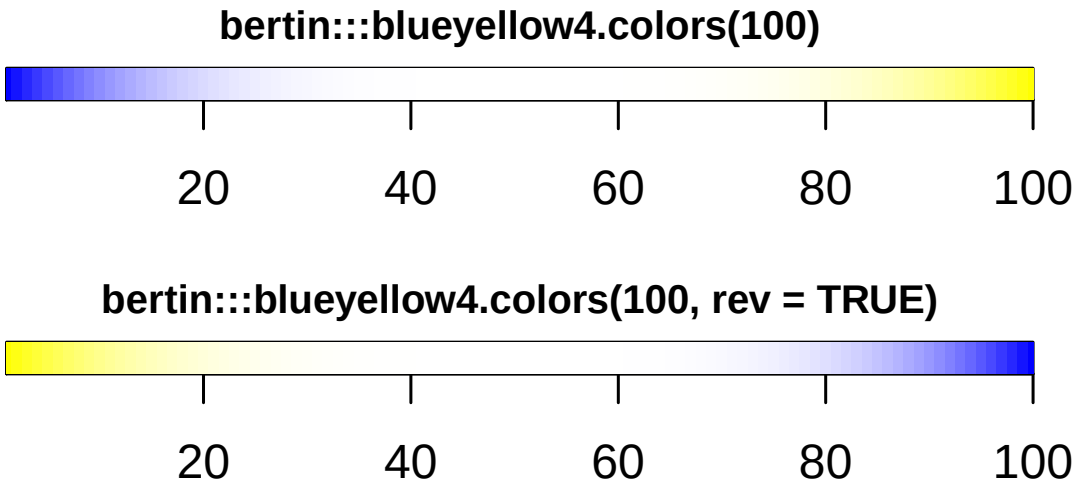
7.2.5. *Contrast Colour Palettes.* If high and low values should be considered of different qualities, a contrast palette may be appropriate instead of a sequential palette. These palettes are also called "divergent" palettes.

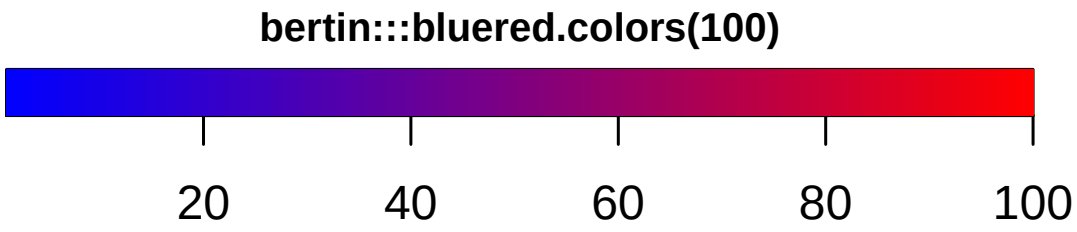
Blue-yellow: This is a rather stable palette, with less than 0.1%readers being affected by colour vision peculiarities. Scale interpretation seeing yellow as higher is also very stable.

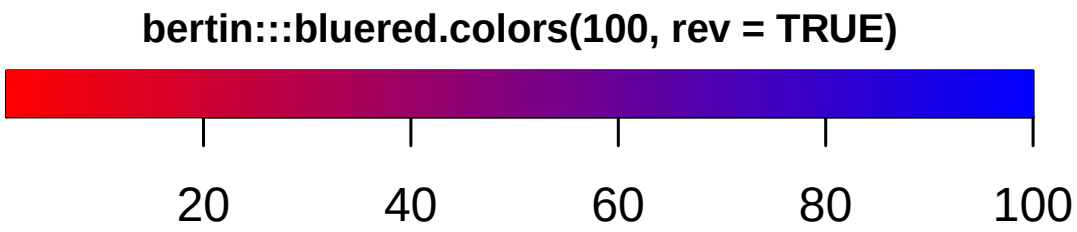


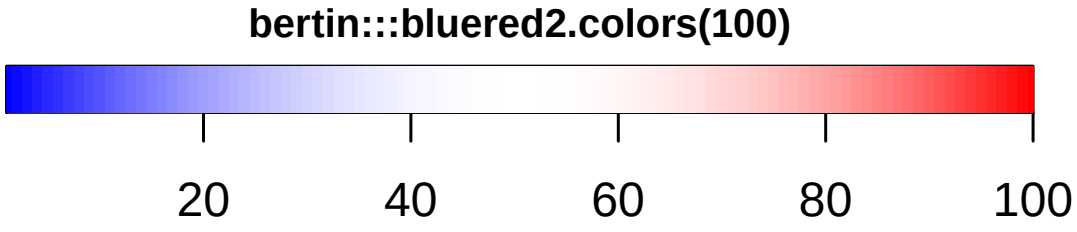
Variant of the above, highlighting the tails and attenuating the center. For an application, see Section 9.2 Case Studies: cDNA Data (page 55). This is a rather stable palette, with less than 0.1% readers being affected by colour vision particularities. Scale interpretation seeing yellow as higher is also very stable.

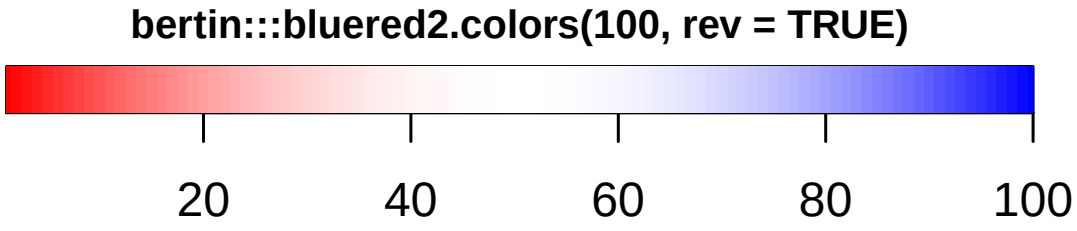


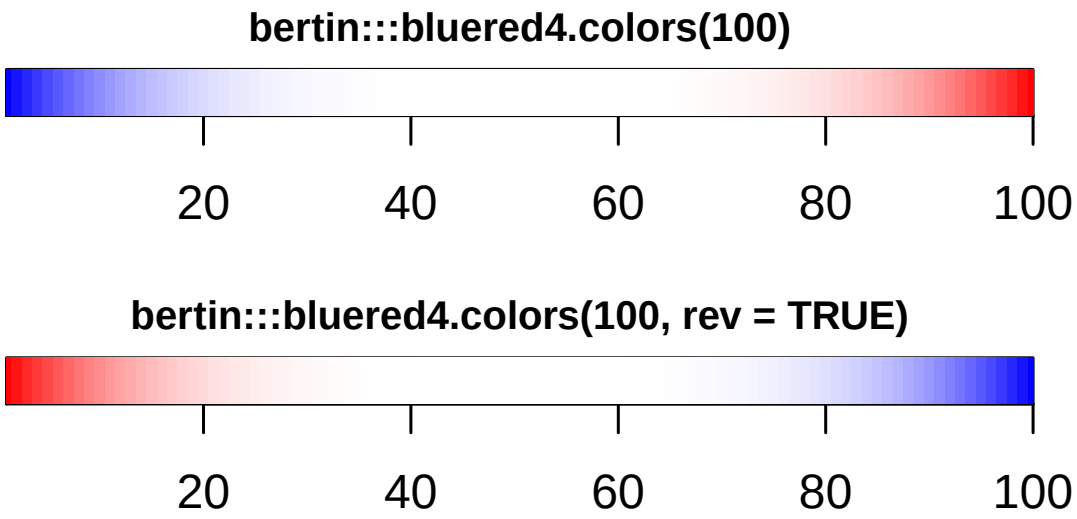

bertin:::blueyellow4.colors(100)
20
40
60
80
100
bertin:::blueyellow4.colors(100, rev = TRUE)
20
40
60
80
100

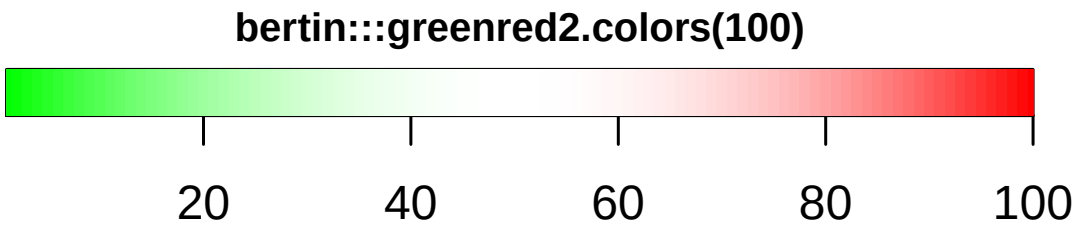

bertin:::bluered.colors(100)
20
40
60
80
100

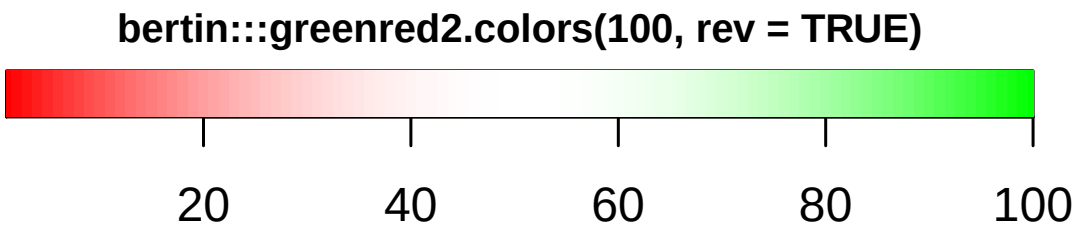

bertin:::bluered.colors(100, rev = TRUE)
20
40
60
80
100

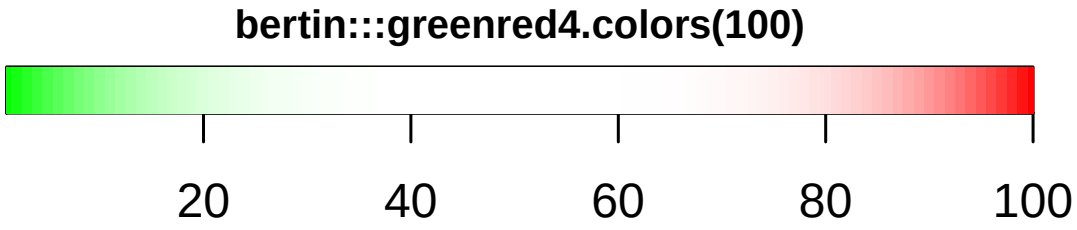

bertin:::bluered2.colors(100)
20
40
60
80
100

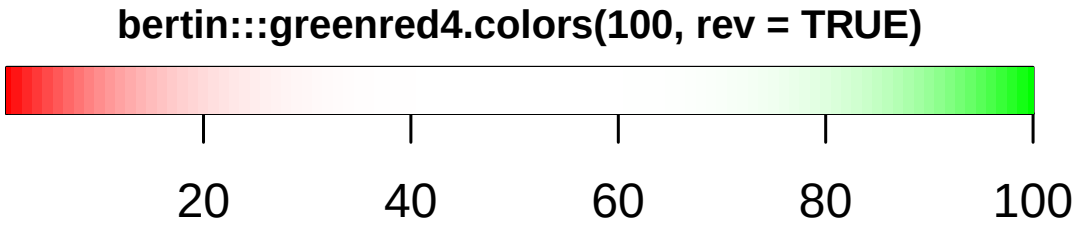

bertin:::bluered2.colors(100, rev = TRUE)
20
40
60
80
100


bertin:::bluered4.colors(100)
20
40
60
80
100
bertin:::bluered4.colors(100, rev = TRUE)
20
40
60
80
100


bertin:::greenred2.colors(100)
20
40
60
80
100


bertin:::greenred2.colors(100, rev = TRUE)
20
40
60
80
100

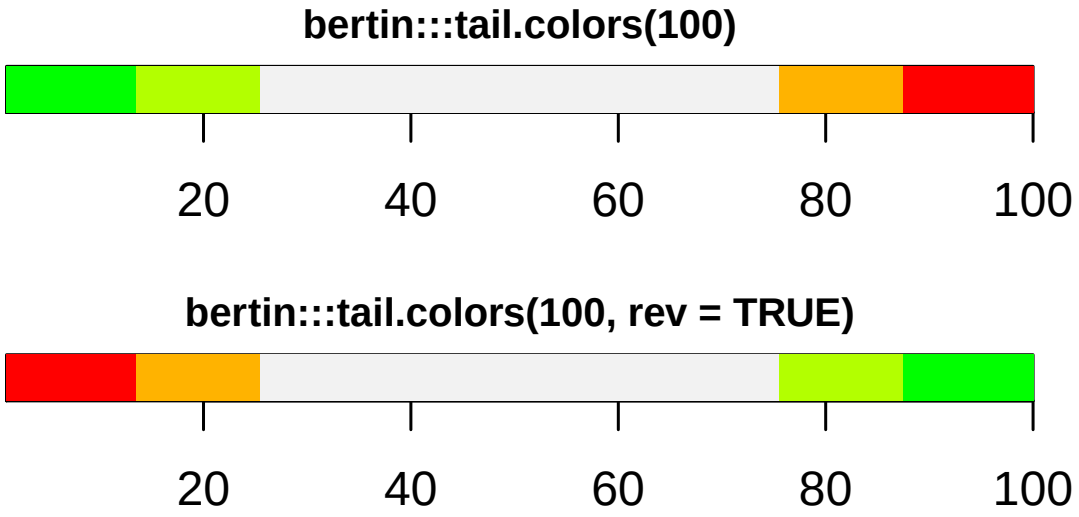

bertin:::greenred4.colors(100)
20
40
60
80
100


bertin:::greenred4.colors(100, rev = TRUE)
20
40
60
80
100

7.2.6. *Tail Colours.*

In (Tukey, 1991), J. Tukey suggested to use a drastic reduction of information for analysis with many covariates. The tail palettes are provided to support a graphical variant of this strategy.

Note: This variant is poor for dichromers.



7.3. **Using Inverted Palettes.** 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,
        palette = grey((1:256)/256), xlab, ylab, main = NULL,
        colinvert=FALSE){
        if (colinvert) palette <- palette[length(palette):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, palette=palette)

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

Input

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

See Figure 11 on page 50 left.

Input

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

See Figure 11 on page 50 right.

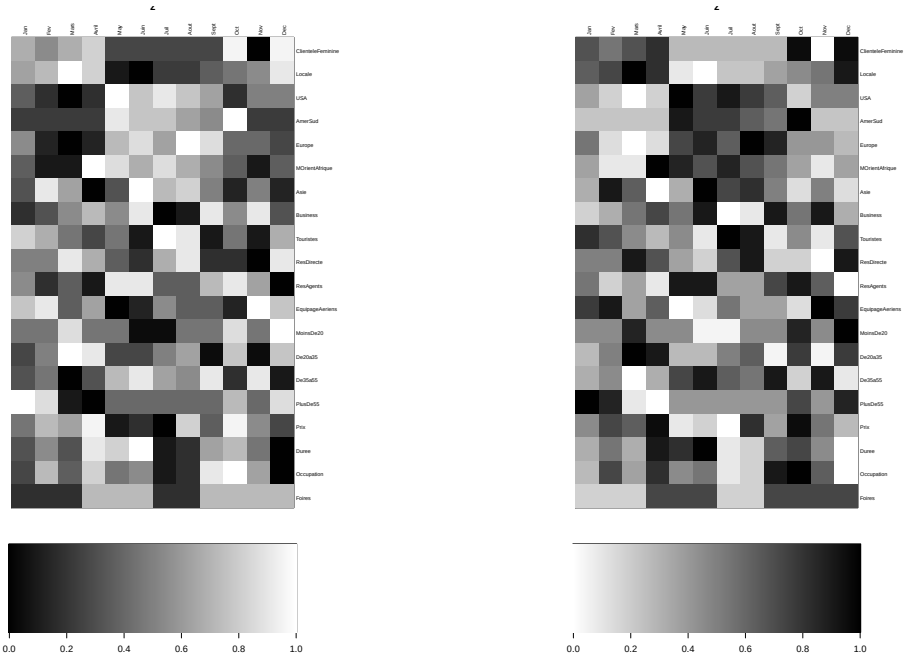


Figure 11: Test matrix: same information, but colour table inverted on the right.

8. Coordinate System and Conventions

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

To recall: the basic graphic system of R shows the proper content in a plot region, with additional information such as axis and labels in an imbedding figure region (Figure 12 on page 51).

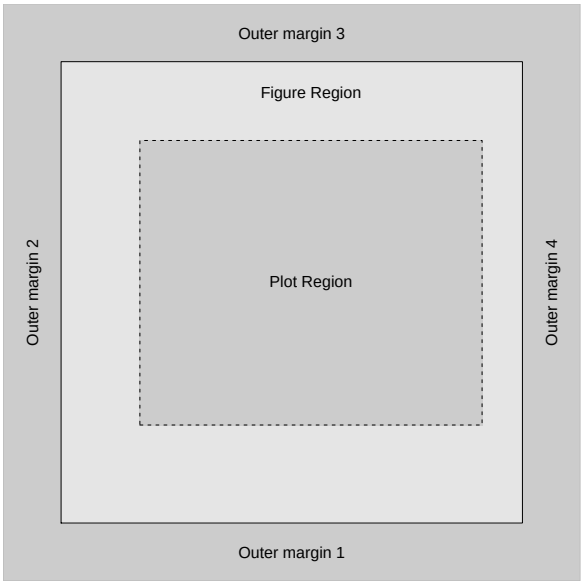


Figure 12: Graphic regions. From (Murrell, 2011)

The figure and its contents is presented on a device, which may allow for an outer margin for exceptional purposes. This gives rise to a list oft parameters as in Figure 13 on page 52. The device space may be allotted to several figures, leading to a hierarchy of coordinate systems and displays (Figure 14 on page 52).

Output

```
pdf
  4
```

By convention, for a matrix `x` the data `x[i, j]` is displayed in a unit square with bottom left corner at `(i, j)`. Coordinates follow matrix conventions, that is the y axis is pointing downwards and the top right corner of this cell is at `(i+1, j-1)`. The space allocated to the matrix fills defines the plot region. The basic functions `imagem()` and `bertinrect` will adjust (shrink) this region to match the aspect ratio of the matrix and establish a user coordinate system with top=0, bottom=$nrow(x)$, left= 1, right=$ncol(x)$.

Additional graphical elements can be used immediately, using this coordinate system.

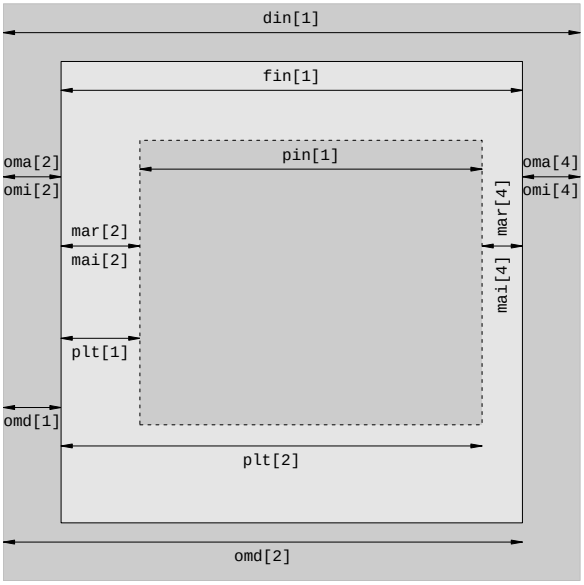


Figure 13: Graphic parameters. From (Murrell, 2011)

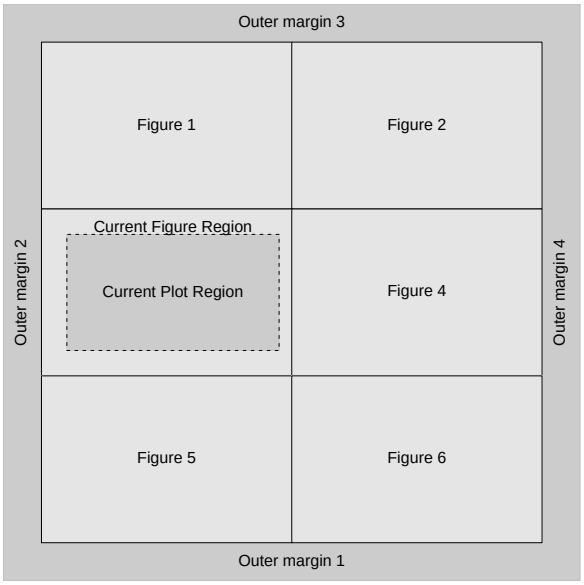


Figure 14: Graphic parameters. From (Murrell, 2011)

image uses the center of a cell as an anchor point, and shows a transposed view. To use *image* to generate an overlay, change the user coordinate system temporarily to *par("usr")-0.5* and use *t()*.

Space for row names and column names is allocated in the inner margins of the figure area. By convention, if a parameter *mar* is provided, this will be added. If a parameter *pars* is provided, it is assumed to contain the user's choice of graphic parameters and will be taken "as is", without further adjustment.

We return the relevant graphical parameters as an invisible result in the basic functions *imagem()* and *bertinrect()*. This can be passed as *pars* to generate graphics with consistent layout.

The higher level functions may attach the actual parameters as attributes to the data, or retrieve from there if provided. Details are subject to change.

If colour encoding is used, a colour ramp should be provided as a key for interpretation. The higher level functions can imbed a colour ramp in the inner margin. While this is an additional picture, this is hidden and the colour ramp appears as an annotation. Other mechanisms for arranging plots on a device should not be affected.

9. Case Studies

9.1. **Borderline Data.** Figure 15 on page 54 shows the raw data from (Ebner-Priemer and Sawitzki, 2007). A colour palette has been chosen to highlight main structures. The data represent tension levels, recorded by 15 minute intervals, in for control group and for borderline patients. In this representation, to overall difference between controls (top rows) and patients (bottom rows) is obvious. But this is an obvious difference under clinical conditions. What is not obvious, but revealed in this display, is that there is a very small number of "controls" who show a tension characteristic similar to the patients. The stress characteristics in the patient group shows a higher variation. But there is a clear subgroup showing "normal" stress characteristics. This can be seen in the inverted colour plot, but is nearly hidden in the original colour plot.

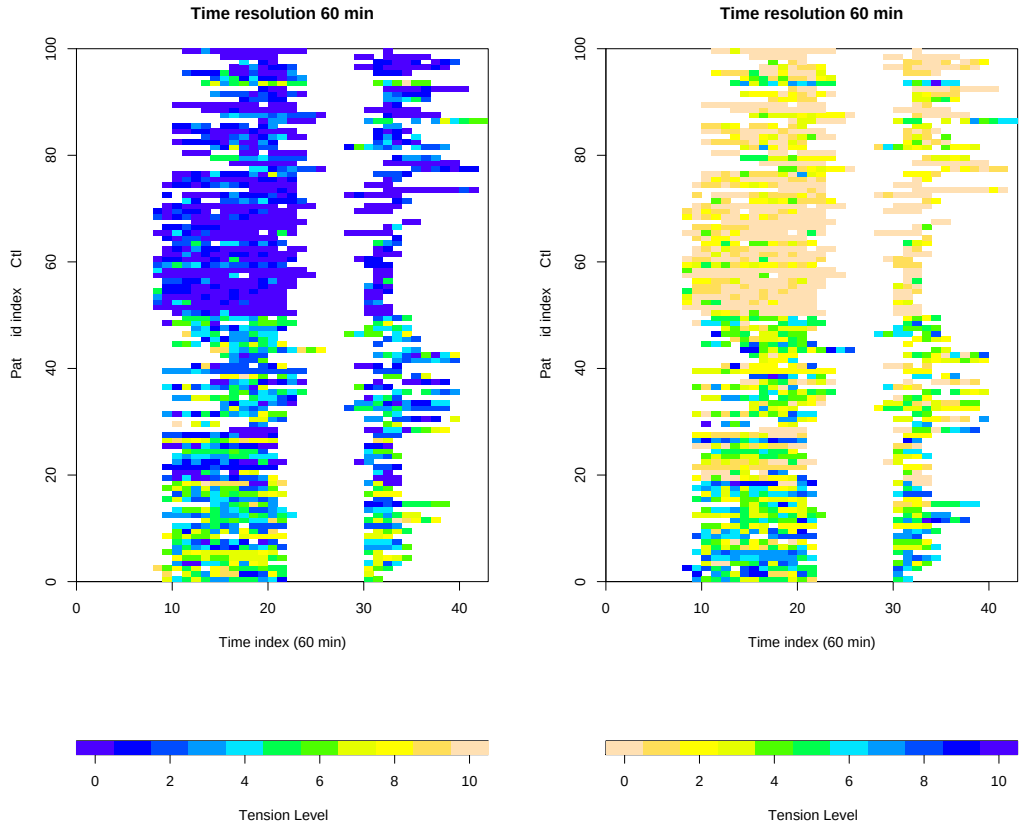


Figure 15: Borderline data. Tension level, all observations, by time. Spot the cases in the control group (Id index > 50) that have tension levels fitting into the patient group rather than the control group.

9.2. **cDNA Data.** In 2-dye cDNA data, hybridisation results are measured using a colour scan, with two samples applied simultaneously to any one microscope slide. The dyes uses for the samples can be recovered by colour separation. Background noise is a major concern. Standard techniques from image analysis is to spot intensity and background intensity for each spot on the slide.Figure 16 on page 56 shows the colour separated channels for foreground and background for one chip. See (Sawitzki, 2002) for details.

This display is used in the quality control of the process. The aim is to decide about the quality of a chip, or of a production protocol. The identification of differentially expressed genes, and ultimately the identification of tumour genes is a different question. To interpret the plot, you need some information about the null behaviour. The probes are not spotted at random, but selected from gene libraries. In this case two libraries have been used, one of kidney related genes, and one of presumably tumour related genes. You expect differences between the upper half and lower half, reflecting these libraries. Not all spots have been used, and you expect bands of unused spots. But other inhomogeneities are informative, for example the accumulation of bad spots on this chip on the sides, which here comes from an asymmetry in handling the incubation cells for the chip - a production problem which can be overcome, once you have identified it.

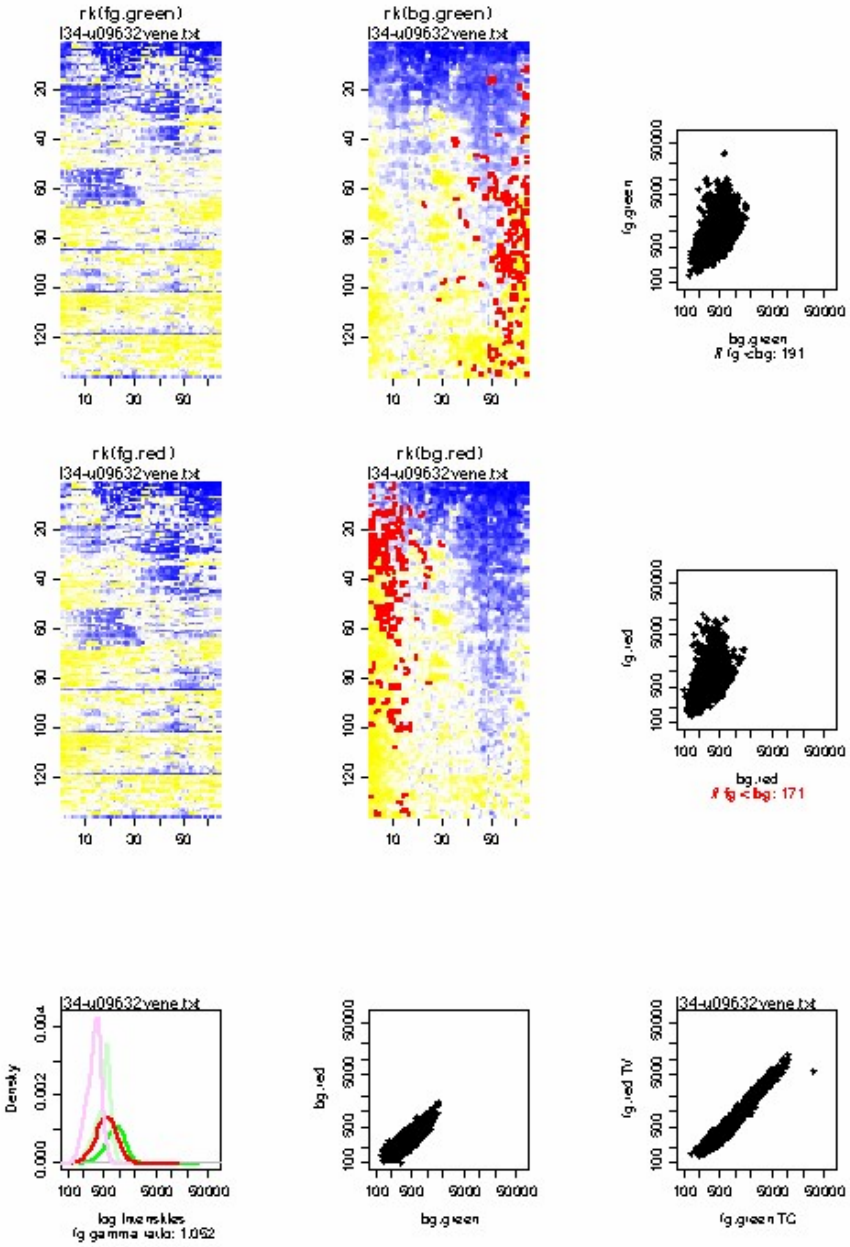


Figure 16: cDNA data. Channel intensities for red and green channel of a cDNA scan (foreground and background).

10. Test Matrices

To test the implementation, a series of matrices is provided. Each matrix is shown in four displays: as an image using default setting of the low level function ***imagem***, as an image using the default setting of ***image.bertin***, as a rectangle display using the low level function ***bertinrect*** and as a display using the default setting of ***plot.bertin***.

Input

```
testplot <- function(z, main=deparse(substitute(z)))
{
        oldpar <- par(mfcol=c(2,2))
        bertinrect(z, main= main)
        title(sub="bertinrect", line=1, cex.sub=1.2)

par(mfg=c(1,2)) # fix for stray display allocation
        imagem(z, main= main)
        title(sub="imagem", line=1, cex.sub=1.2)

par(mfg=c(2,1)) # fix for stray display allocation -- this is very bad
        plot.bertin(z, main= main)
par(mfg=c(2,1)) # fix for stray display allocation -- this is very very bad
        title(sub="plot.bertin", line=0, cex.sub=1.2)
par(mfg=c(2,2)) # fix for stray display allocation
        image.bertin(z, main= main)
        title(sub="image.bertin", line=0,  cex.sub=1.2)
                par(oldpar)
}
```

10.1. **Pure Vanilla Random Matrices.** 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

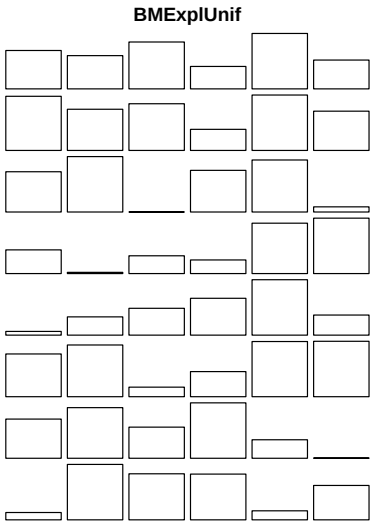
Input

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

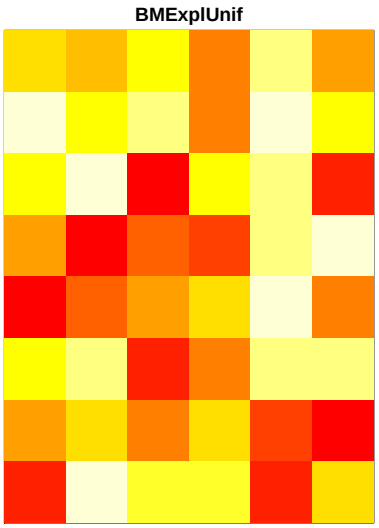
Typical cases are :

Input

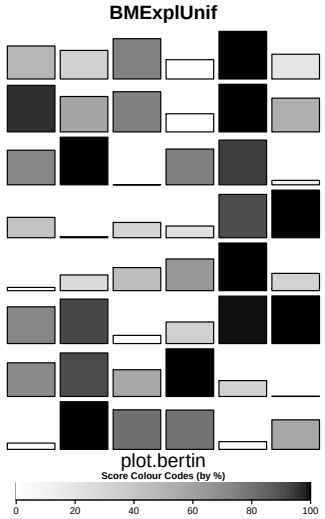
```
BMExplUnif  <- matrix( runif(BMExplRows*BMExplCols),
        nrow= BMExplRows, ncol= BMExplCols)
BMExplNorm  <- matrix( rnorm(BMExplRows*BMExplCols),
        nrow= BMExplRows, ncol= BMExplCols)
```

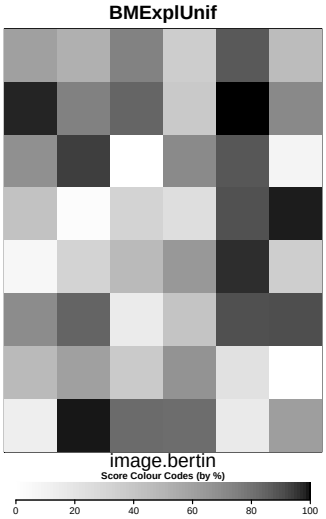


bertinrect

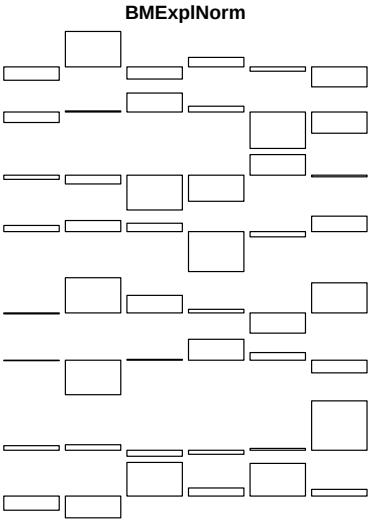


imagem

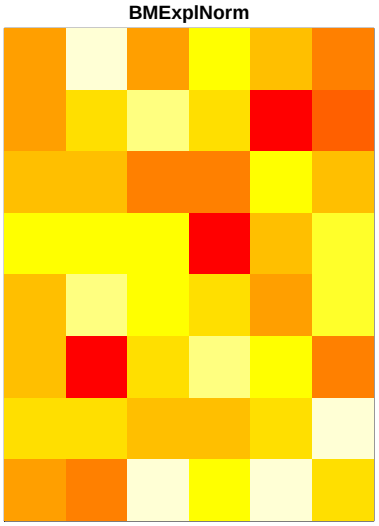




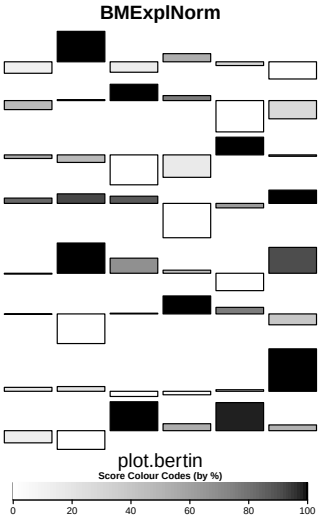
Random uniform, using default settings.

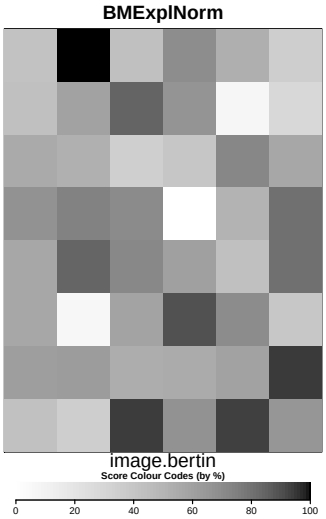


bertinrect



imagem





Random normal, using default settings.

10.2. **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)
```

10.2.1. *Basic Test Matrices.*

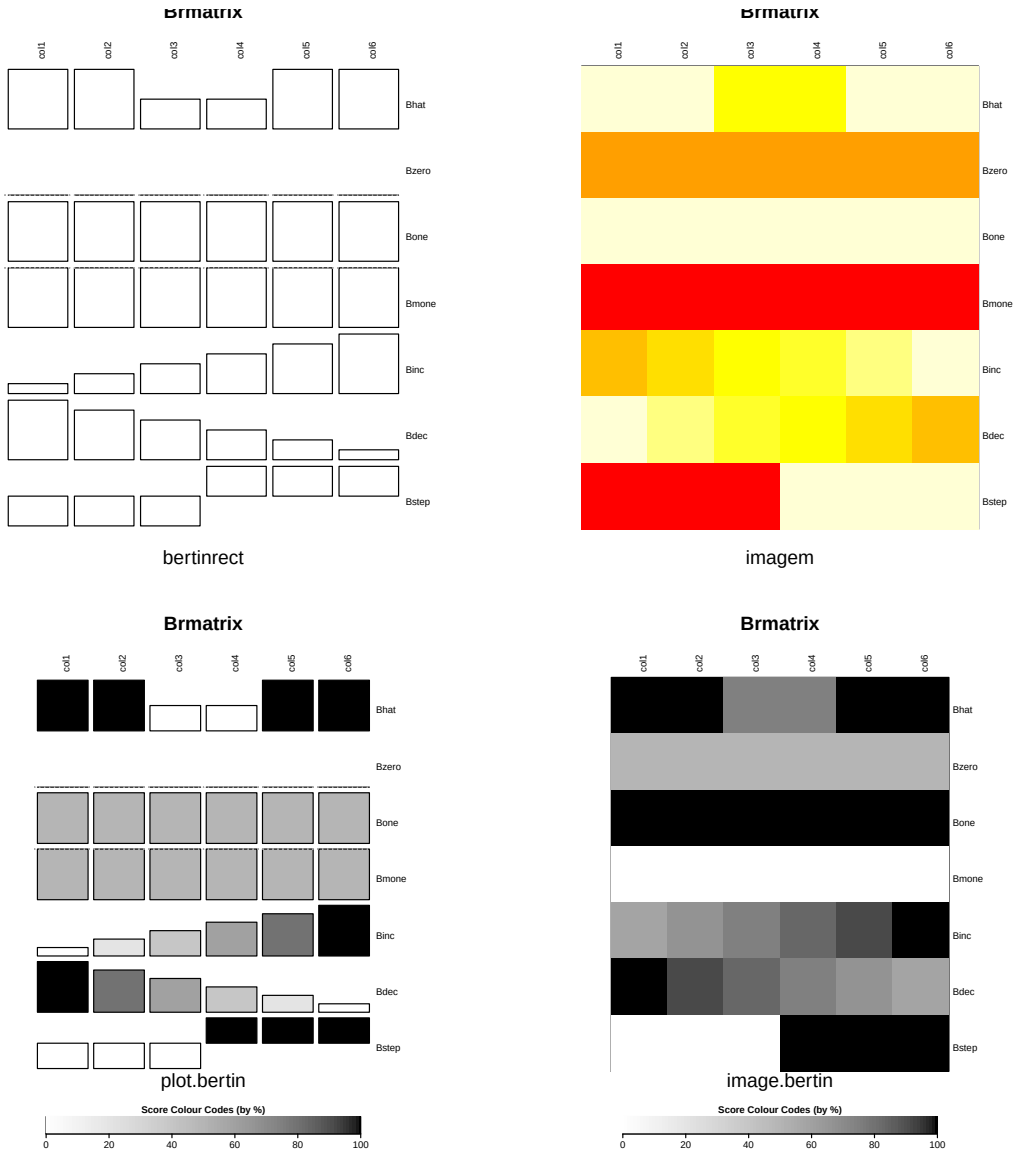
Input

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

See Table 3 on page 61.

	col1	col2	col3	col4	col5	col6
Bhat	1.00	1.00	0.50	0.50	1.00	1.00
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

Table 3: Brmatrix: test matrix, by row.



Test matrix by row, using default settings

R may use internal housekeeping to keep matrix columns homogeneous. Check! Use row matrix and column matrix for tests.

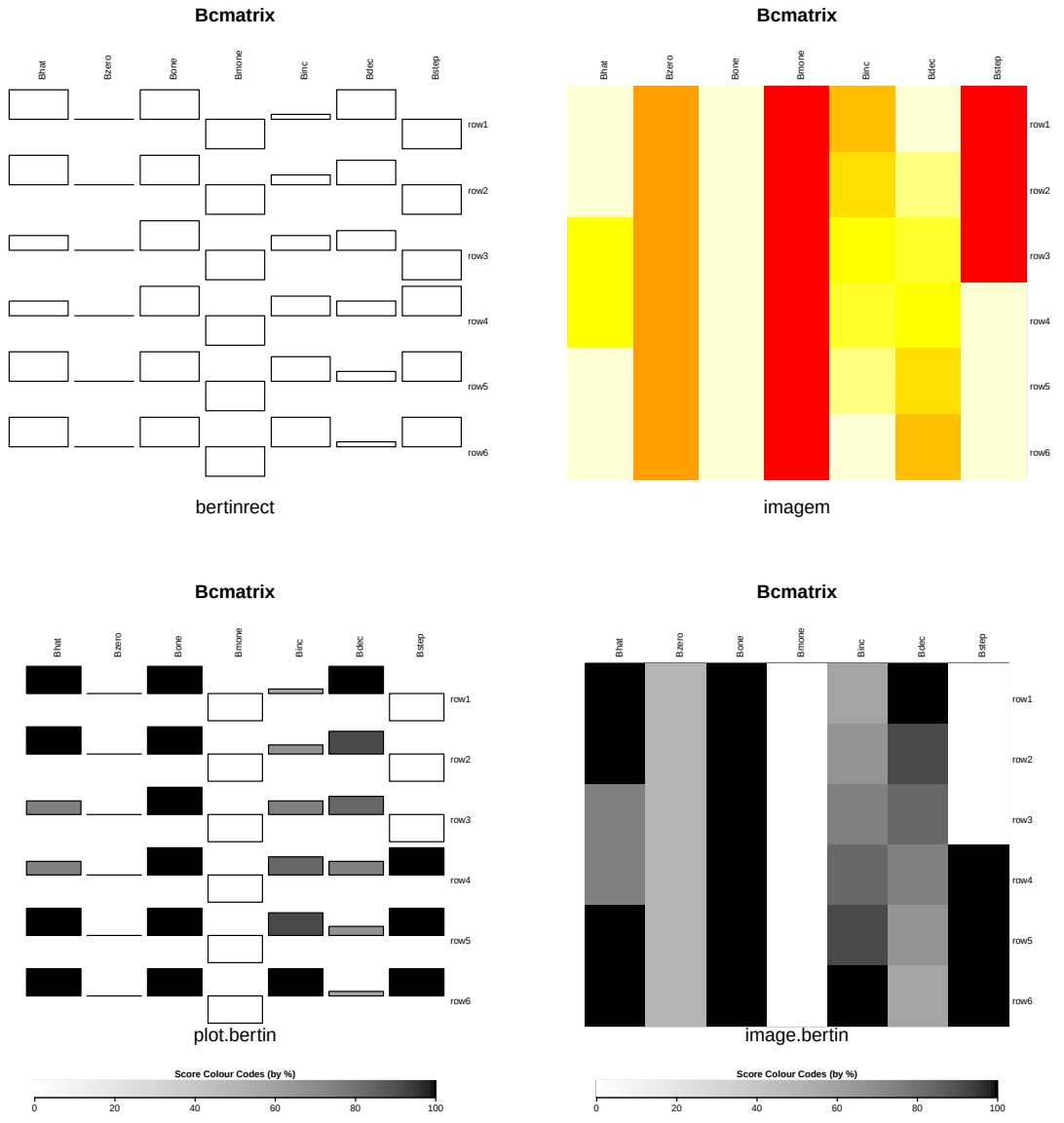
Input

```
Bcmatrix <- cbind(Bhat, Bzero, Bone, Bmone, Binc, Bdec, Bstep)
rownames(Bcmatrix) <- rownames(Bcmatrix,FALSE)
```

See Table 4 on page 63.

	Bhat	Bzero	Bone	Bmone	Binc	Bdec	Bstep
row1	1.00	0.00	1.00	-1.00	0.17	1.00	-1.00
row2	1.00	0.00	1.00	-1.00	0.33	0.83	-1.00
row3	0.50	0.00	1.00	-1.00	0.50	0.67	-1.00
row4	0.50	0.00	1.00	-1.00	0.67	0.50	1.00
row5	1.00	0.00	1.00	-1.00	0.83	0.33	1.00
row6	1.00	0.00	1.00	-1.00	1.00	0.17	1.00

Table 4: Bcmatrix: test matrix, by column.

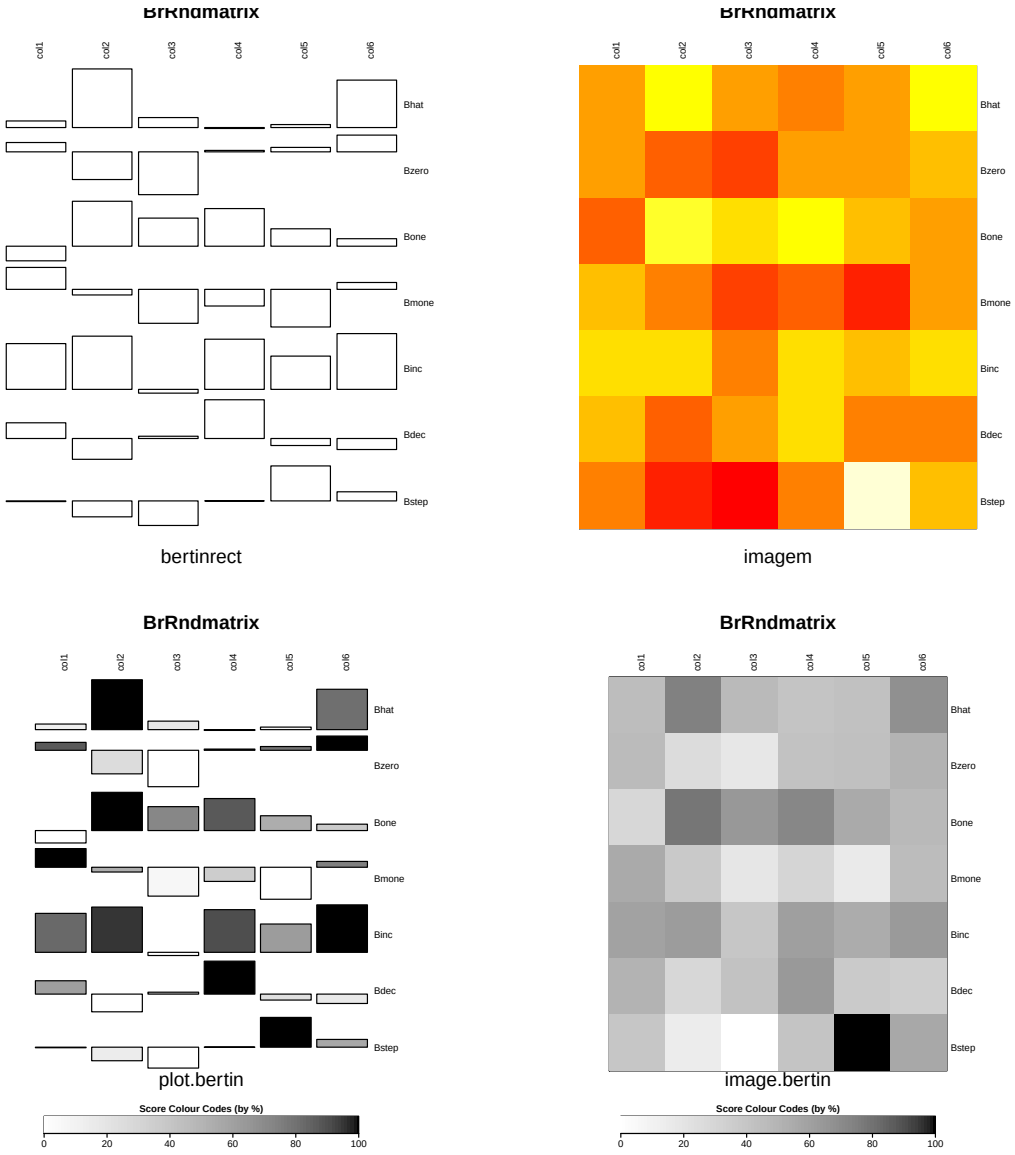


Test matrix by column, using default settings

10.2.2. *Basic Test Matrices With Normal Random Error.*

Input

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



Test matrix by row with normal random error, using default settings

10.3. **Test Matrices With IEEE Specials.**

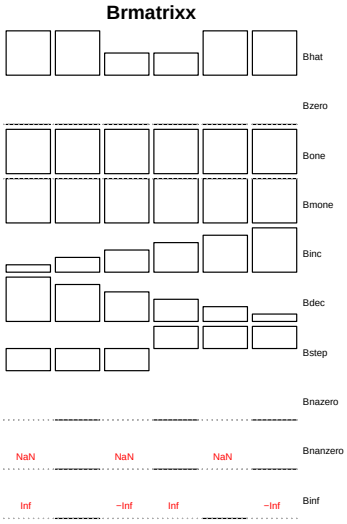
Input

```
Brmatrixx <- rbind(Bhat, Bzero, Bone, Bmone, Binc, Bdec, Bstep,
        Bnazero, Bnanzero, Binf)
```

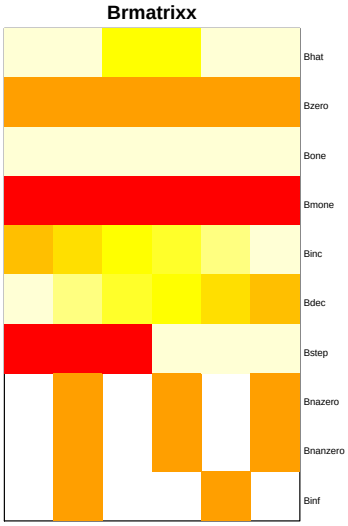
See Table 5 on page 65.

	1	2	3	4	5	6
Bhat	1.00	1.00	0.50	0.50	1.00	1.00
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
Bnazero		0.00		0.00		0.00
Bnanzero		0.00		0.00		0.00
Binf	Inf	0.00	-Inf	Inf	0.00	-Inf

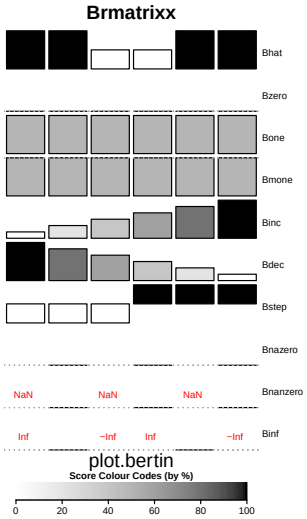
Table 5: Brmatrixx: matrix with special values, by row. NaN and NA values are not printed.

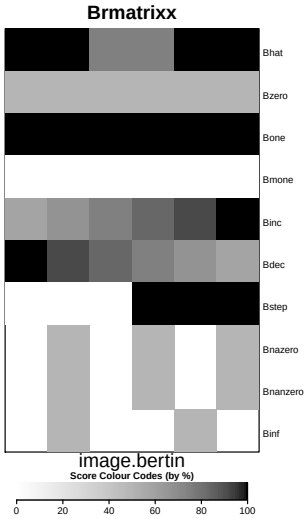


bertinrect



imagem

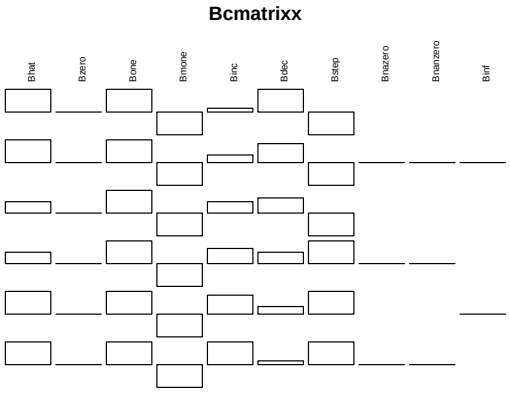




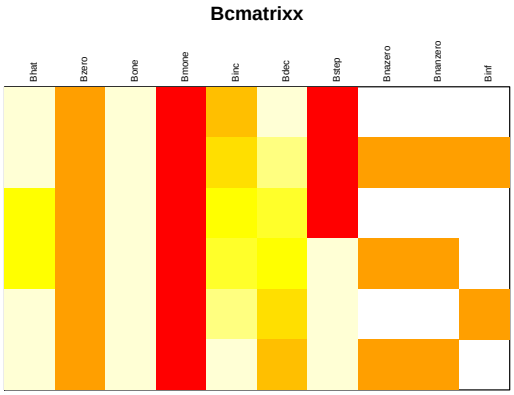
10.3.1. *Test Matrix With IEEE Specials By Row, Using Default Settings.*

Input

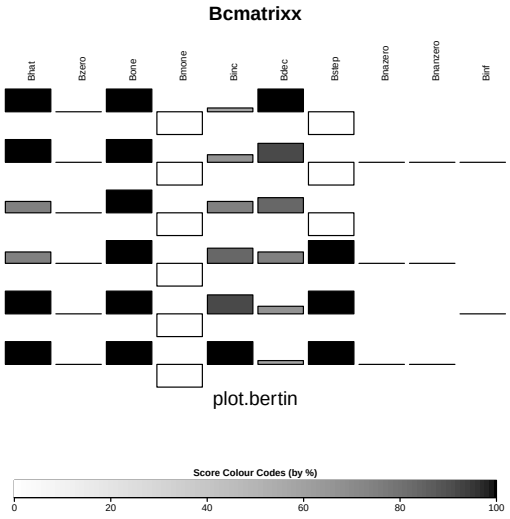
```
Bcmatrixx <- cbind(Bhat, Bzero, Bone, Bmone, Binc, Bdec, Bstep,
        Bnazero, Bnanzero, Binf)
```



bertinrect



imagem



plot.bertin

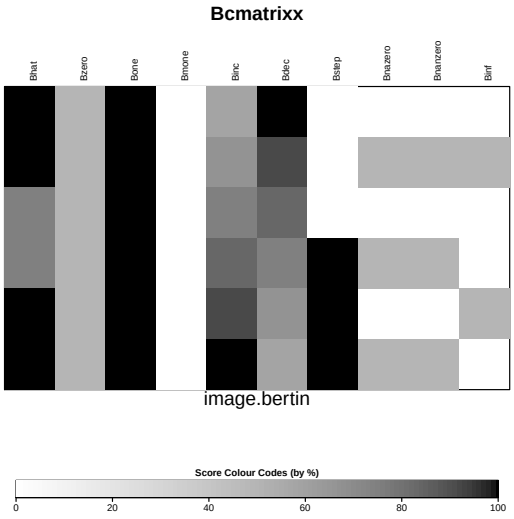
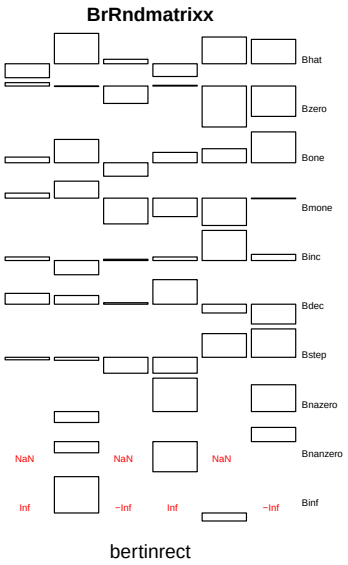


image.bertin

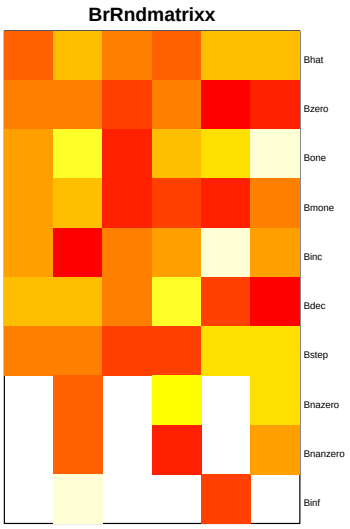
10.3.2. *Test Matrix With IEEE Specials by Column, Using Default Settings.*

Input

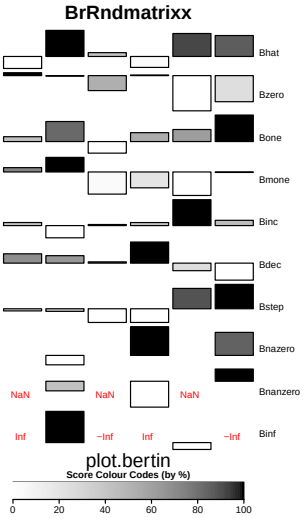
```
BrRndmatrixx <- Brmatrixx+rnorm(nrow(Brmatrixx)*ncol(Brmatrixx))
```



bertinrect



imagem



plot.bertin

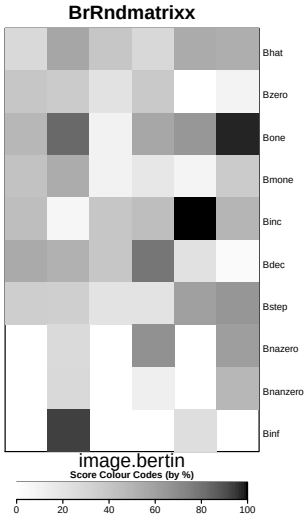


image.bertin

References

Bertin, Jaques. 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.
Breiman, Leo, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and regression trees*, Wadsworth, Belmont, CA.
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.
Ebner-Priemer, Ulrich W. and Günther Sawitzki. 2007. *Ambulatory assessment of affective instability in borderline personality disorder*, European Journal of Psychological Assessment **23**, no. 4, 238–247.
Friedman, Jerome H. 1996. *Local Learning Based on Recursive Covering*, Department of Statistics, Stanford University.
Murrell, Paul. 2011. *R graphics*, 2nd ed., The R Series, Chapman & Hall/CRC.
Sawitzki, Günther. 1996. *Extensible statistical software: On a voyage to oberon.*, Journal of Computational and Graphical Statistics **5**, no. 3.
_____. 2002. *Quality Control and Early Diagnostics for cDNA Microarrays*, R News **2**, no. 1, 6–10.
Tukey, John W. 1991. *Use of many covariates in clinical trials*, International Statistical Review / Revue Internationale de Statistique **59**, no. 2, pp. 123–137 (English).

$HeadURL: svn+ssh://gsawitzki@svn.r-forge.r-project.org/svnroot/bertin/pkg/inst/doc/bertinR.Rnw $
$Id: bertinR.Rnw 60 2012-05-09 17:51:14Z gsawitzki $
$Revision: 60 $
$Date: 2012-05-09 19:51:14 +0200 (Wed, 09 May 2012) $
$Author: gsawitzki $
textwidth: 6.00612in linewidth:6.00612in
textheight: 9.21922in

Address: G. Sawitzki, StatLab Heidelberg, Im Neuenheimer Feld 294, D 69120 Heidelberg

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

URL: http://bertin.r-forge.r-project.org/