European Commission

Symbolic Data from classical data and basic stats for distributional data

A. Irpino, R. Verde
ESTP Cologne 14-16 May 2024

Outline

- The HistDAWass package and how to install it
- Main classes and methods
- From raw data to histograms
- Histogram-valued data table
- Distances for histogram-valued data: the L_2 Wasserstein distance
- Basic univariate statistics for histogram variables
- Association measure for histogram variables

HistDAWass Package

- Histogram
- Data
- Analysis
- using Wasserstein distance

The package can be downloaded from CRAN and charged as usually(in R):

```r
install.packages("HistDAWass")
library("HistDAWass")
```

Classes of HistDAWass

Classes and methods were implemented in the package using S4

The main motivation is that the S4 paradigm is object-oriented, while S3 does not.

Main classes of the package

- `distributionH` – the class representing a histogram-valued data (HD);
- `MatH` – a matrix of HD, it is symilar to a symbolic table, but containing only histograms;
- `TdistributionH` – an HD with a timestamp, or a time interval (useful for the analysis of histogram time series)
- `TMatH` – a matrix of HD’s with a timestamp, or a time interval;
- `HTS` – a Histogram Time Series (an ordered list of TdistributionH);

The `distributionH` class: a class for describing and manipulating histogram-valued data

A class describing a 1-d histogram

We want to create a new object containing the following histogram:

Bin	Rel.freq.	CDF
$[1;2)$	0.4	0.4
$(2;3]$	0.6	1
Tot.	1.0	

```r
mydist=distributionH(x=c(1,2,3),p=c(0,0.4, 1))
str(mydist)
```

```
Formal class 'distributionH' [package "HistDAWass"] with 4
slots
  ..@ x: num [1:3] 1 2 3
  ..@ p: num [1:3] 0 0.4 1
  ..@ m: num 2.1
  ..@ s: num 0.569
```

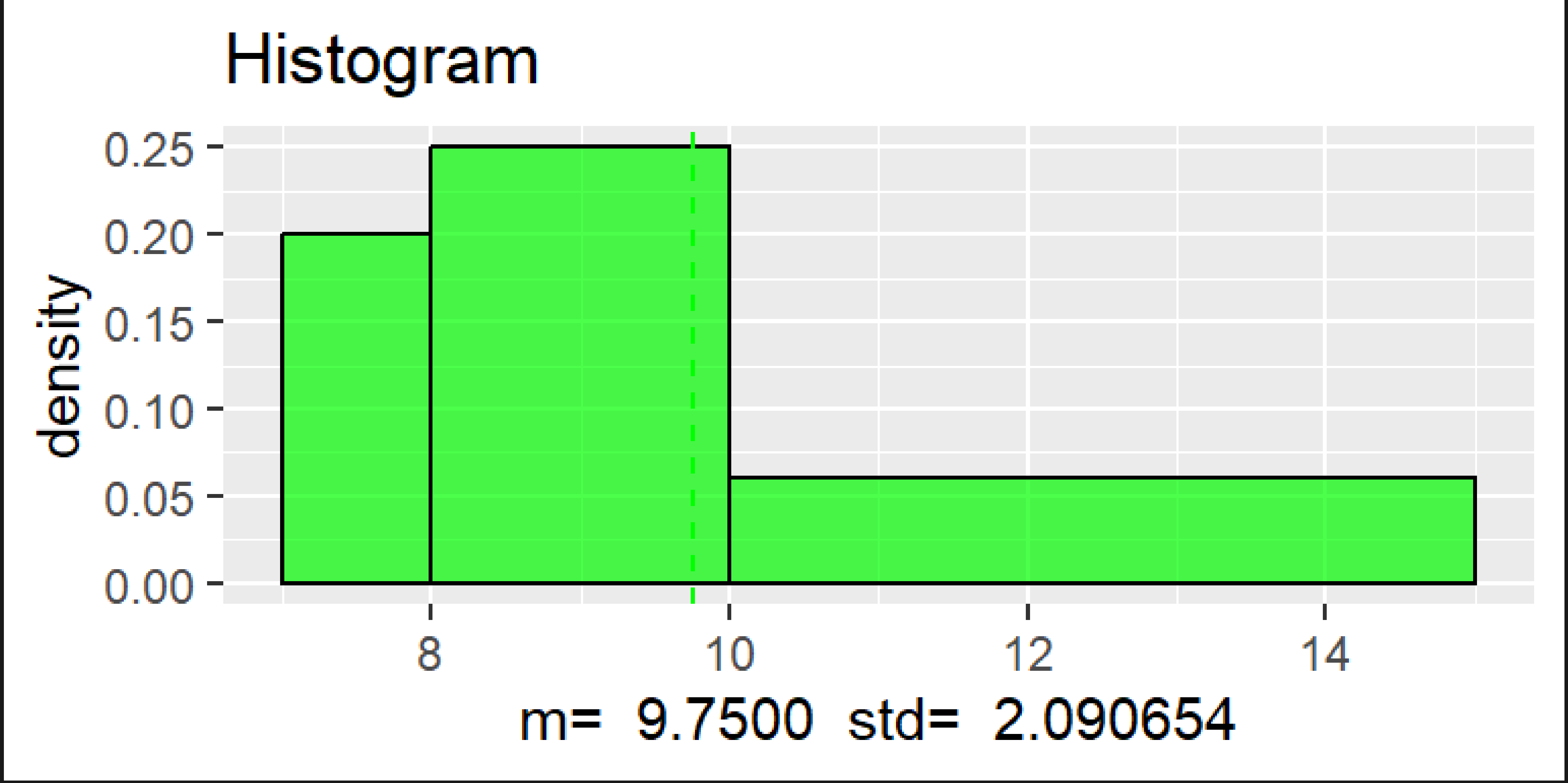
Show function

```
mydist
```

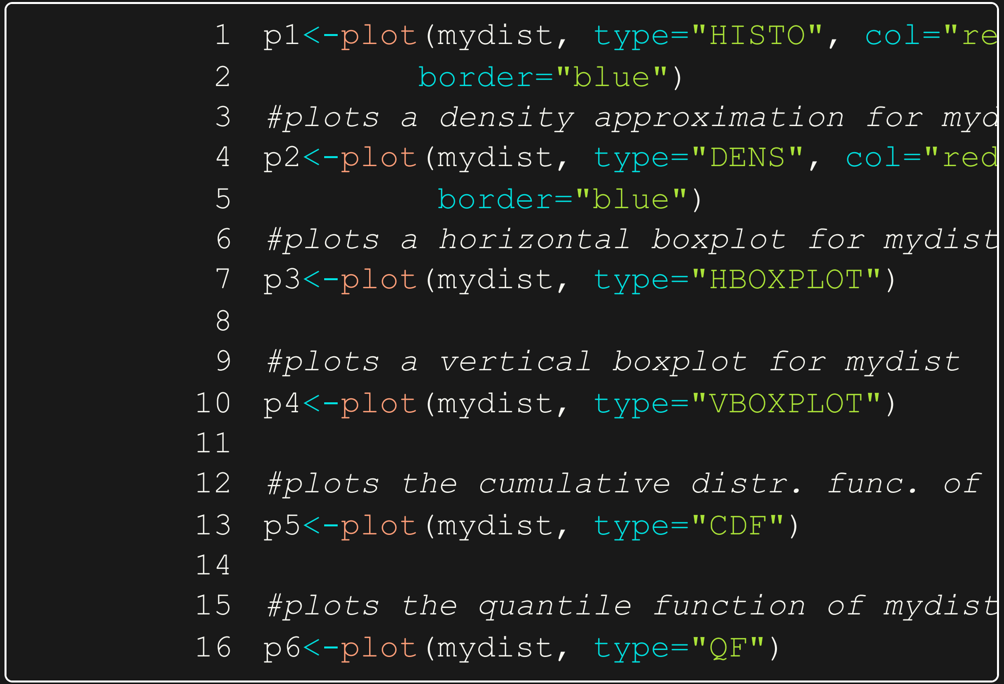
```
           X          p
Bin_1 [ 1 ; 2 )        0.4
Bin_2 [ 2 ; 3 ]        0.6

 mean =  2.1   std  =  0.568624070307733
```

Plot functions

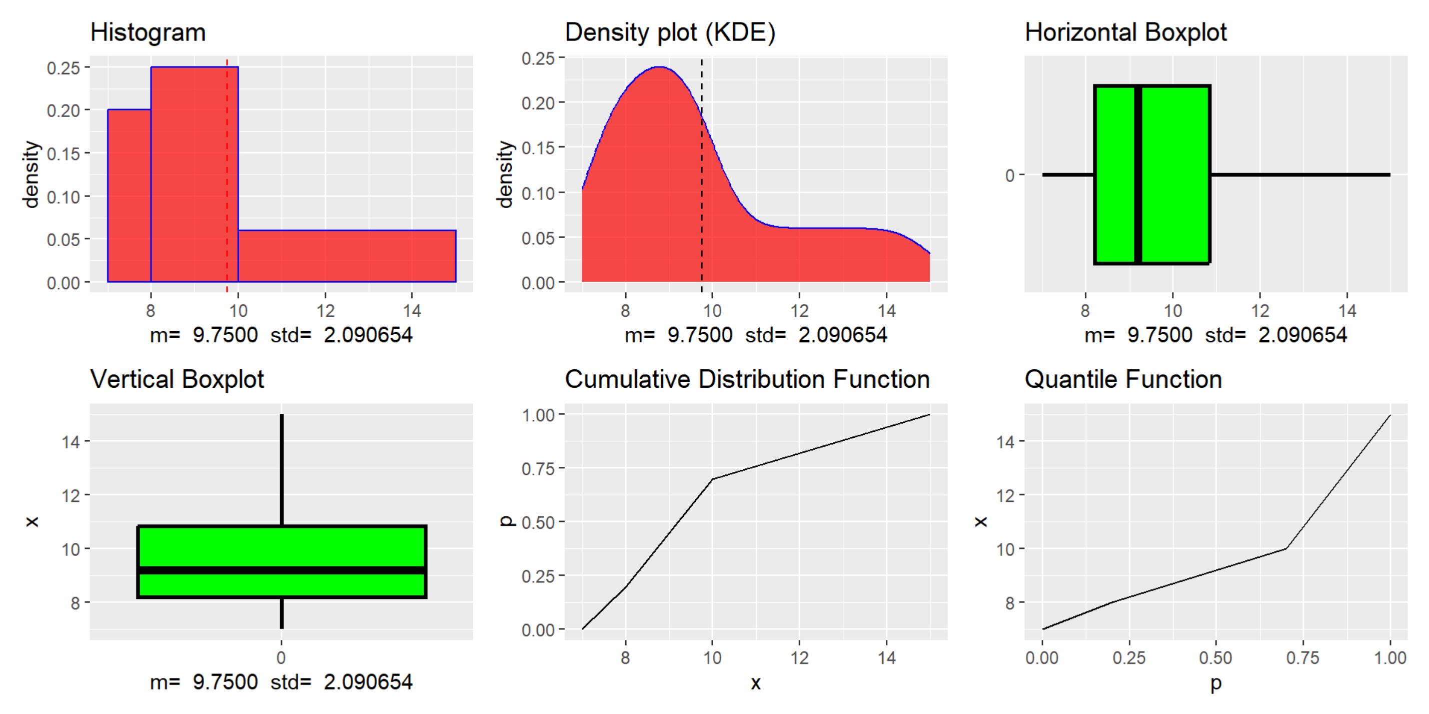
```
mydist<-distributionH(x=c(7,8,10,15),p=c(0, 0.2, 0.7, 1))
plot(mydist) #plots mydist
```



Some options for plotting a distributionH object



```
p1<-plot(mydist, type="HISTO", col="re
        border="blue")
#plots a density approximation for myd
p2<-plot(mydist, type="DENS", col="red
         border="blue")
#plots a horizontal boxplot for mydist
p3<-plot(mydist, type="HBOXPLOT")

#plots a vertical boxplot for mydist
p4<-plot(mydist, type="VBOXPLOT")

#plots the cumulative distr. func. of
p5<-plot(mydist, type="CDF")

#plots the quantile function of mydist
p6<-plot(mydist, type="QF")
```



Obtainning the histogram and the CDF of a distributionH object

```
mydist.histo<-get.histo(mydist) #this returns the histogram
mydist.cdf<-get.distr(mydist) #this returns the CDF
#into data.frame objects
mydist.histo
```

```
  min.x max.x   p
1     7     8 0.2
2     8    10 0.5
3    10    15 0.3
```

```
mydist.cdf
```

```
   x   p
1  7 0.0
2  8 0.2
3 10 0.7
4 15 1.0
```

Obtaining a single quantile or a probability from a distributionH object

```
# computes the CDF value for x=9.5
compP(object = mydist,q = 9.5)
```

```
[1] 0.575
```

```
# computes the quantile  for p=0.1
compQ(object = mydist,p = 0.1)
```

```
[1] 7.5
```

Other basic statistics for distributionH objects

```
mydist.mean=meanH(mydist) #computes the mean
mydist.std=stdH(mydist) #computes the standard deviation
mydist.skew=skewH(mydist) #computes the 3rd stand. centr. moment
mydist.kurt=kurtH(mydist) #computes the 4th stand. centr. moment
```

Being $Q(p)$ a quantile function the four measures are, respectively, the histogram version of the following formulas (Gilchrist 2000):

$$\boxed{\mu = \int_0^1 Q(p)dp}, \quad \boxed{\sigma = \sqrt{\int_0^1 Q(p)^2 dp - \mu^2}},$$

$$\boxed{sk = \int_0^1 \left(\frac{Q(p)-\mu}{\sigma}\right)^3 dp}, \quad \boxed{ku = \int_0^1 \left(\frac{Q(p)-\mu}{\sigma}\right)^4 dp}.$$

The histogram trick (1)

We can consider the histogram as a weighted mixture of b (the number of classes or bins of a histogram) disjointed uniform *pdf*s. Thus, if we consider the trivial histogram as the histogram with one bin (namely, a uniform distribution), and that it is defined as $X \sim U(a, b)$ having *qf* $Q(p) = a + p \cdot (b - a)$ we have:

$$\boxed{\mu = \int_0^1 Q(p)dp = \int_0^1 [a + p \cdot (b - a)]dp = \frac{a + b}{2}}$$

or if we consider the center (midpoint) $c = \frac{a+b}{2}$ and the radius (half-width) $r = \frac{b-a}{2}$, the $Q(p) = c - r + 2pr = c + r(2p - 1)$ we have

$$\boxed{\mu = \int_0^1 Q(p)dp = \int_0^1 [c + r(2p - 1)]dp = c}$$

The histogram trick: the mean (2)

If we consider the histogram with k bins a collection of weighted uniform:

$$H = \{([a_1, b_1], \pi_1), \ldots, ([a_k, b_k], \pi_k)\}.$$

Let's consider $F_\ell = \sum_{s=1}^{\ell} \pi_s$ and $F_0 = 0$, then the $Q(p)$ is a piece wise linear function then:

$$\boxed{\mu = \int_0^1 Q(p)dp = \sum_{\ell=1}^{k} \int_{F_{\ell-1}}^{F_\ell} Q(p)dp = \sum_{\ell=1}^{k} \pi_\ell \frac{a_\ell + b_\ell}{2} = \sum_{\ell=1}^{k} \pi_\ell c_\ell}$$

The histogram trick, plus the center-radii transformation: standard deviation (3)

We have the distribution $H = \{([a_1, b_1], \pi_1), \ldots, ([a_k, b_k], \pi_k)\}$. Each bin can be described in terms of center and radii: $c_\ell = \frac{a_\ell + b_\ell}{2}$ and $r_\ell = \frac{b_\ell - a_\ell}{2}$. The standard deviation of a histogram is computed as follows:

$$\boxed{\sigma = \sqrt{\int_0^1 Q(p)^2 dp - \mu^2} = \sqrt{\sum_{\ell=1}^{k} \int_{F_{\ell-1}}^{F_\ell} Q(p)^2 dp - \mu^2} = \sqrt{\sum_{\ell=1}^{k} \pi_\ell \left[c_\ell^2 + \frac{1}{3} r_\ell^2 \right] - \mu^2}}$$

The histogram trick, the center-radii transformation and standardization: skweness and the kurtosis (4)

The skewness and the kurtosis indices are computed as the third and the fourth standardized moments of the histogram. Using a centered histogram (i.e. a histogram shifted to their mean value), the corresponding *centered quantile function* $Q^c(p) = Q(p) - \mu$ and some well-known simplifications, it is possible to prove that we can compute exactly the indices into a finite number of operations (avoiding the numerical problems related to the numeric computation of the integrals). Let's consider ${}_s c_\ell = \frac{c_\ell - \mu}{\sigma}$ and ${}_s r_\ell = \frac{r_\ell}{\sigma}$, the standardized midpoints and the normalized radii of the bins, the two shape indices are computed (exactly) as follows:

$$\boxed{sk = \frac{\int\limits_0^1 \left[Q^c(p)\right]^3 dp}{\sigma^3} = \sum_{\ell=1}^{k} \pi_\ell \cdot {}_s c_\ell \cdot \left[{}_s c_\ell^2 + {}_s r_\ell^2\right]}, \quad \boxed{ku = \frac{\int\limits_0^1 \left[Q^c(p)\right]^4 dp}{\sigma^4} = \sum_{\ell=1}^{k} \frac{\pi_\ell}{5}\left[5 {}_s c_\ell^4 + 10 {}_s c_\ell^2 {}_s r_\ell^2 + {}_s r_\ell^4\right]}$$

How to obtain midponts and radii (1)

```
get.histo(mydist)
##   min.x max.x   p
## 1     7     8 0.2
## 2     8    10 0.5
## 3    10    15 0.3
crwtransform(mydist) #return a list with three slots
## $Centers
## [1]  7.5  9.0 12.5
##
## $Radii
## [1] 0.5 1.0 2.5
##
## $Weights
## [1] 0.2 0.5 0.3
```

L_2 Wasserstein distance between distributionH objects

Given two distributions having f and g as *pdf*s and, respectively, $Q_f(p)$ and $Q_g(p)$ as quantile functions, the (squared) L_2 Wasserstein distance is:

$$\boxed{d_W^2(f,g) = \int_0^1 \left[Q_f(p) - Q_g(p)\right]^2 dp}$$

(A. Irpino and Romano 2007) and (R. Irpino A.and Verde 2015) showed that

$$\boxed{d_W^2(f,g) = (\mu_f - \mu_g)^2 + (\sigma_f - \sigma_g)^2 + 2\sigma_f\sigma_g\left[1 - \rho_{QQ}(f,g)\right]}$$

where $\rho_{QQ}(f,g)$ is the Pearson correlation between two *qf*s:

$$\rho_{QQ}(f,s) = \frac{\int_0^1 Q_f(p)Q_g(p)dp - \mu_f\mu_g}{\sigma_f\sigma_g}$$

Computing the dot product between two quantile functions of histograms

If two histograms H_1 and H_2 have a same number of bins, say k, and such bins contain respectively the same mass, namely, $\pi_{\ell,1} = \pi_{\ell,2}\ \forall \ell \in k$ the dot product can be computed using the histogram trick as follows:

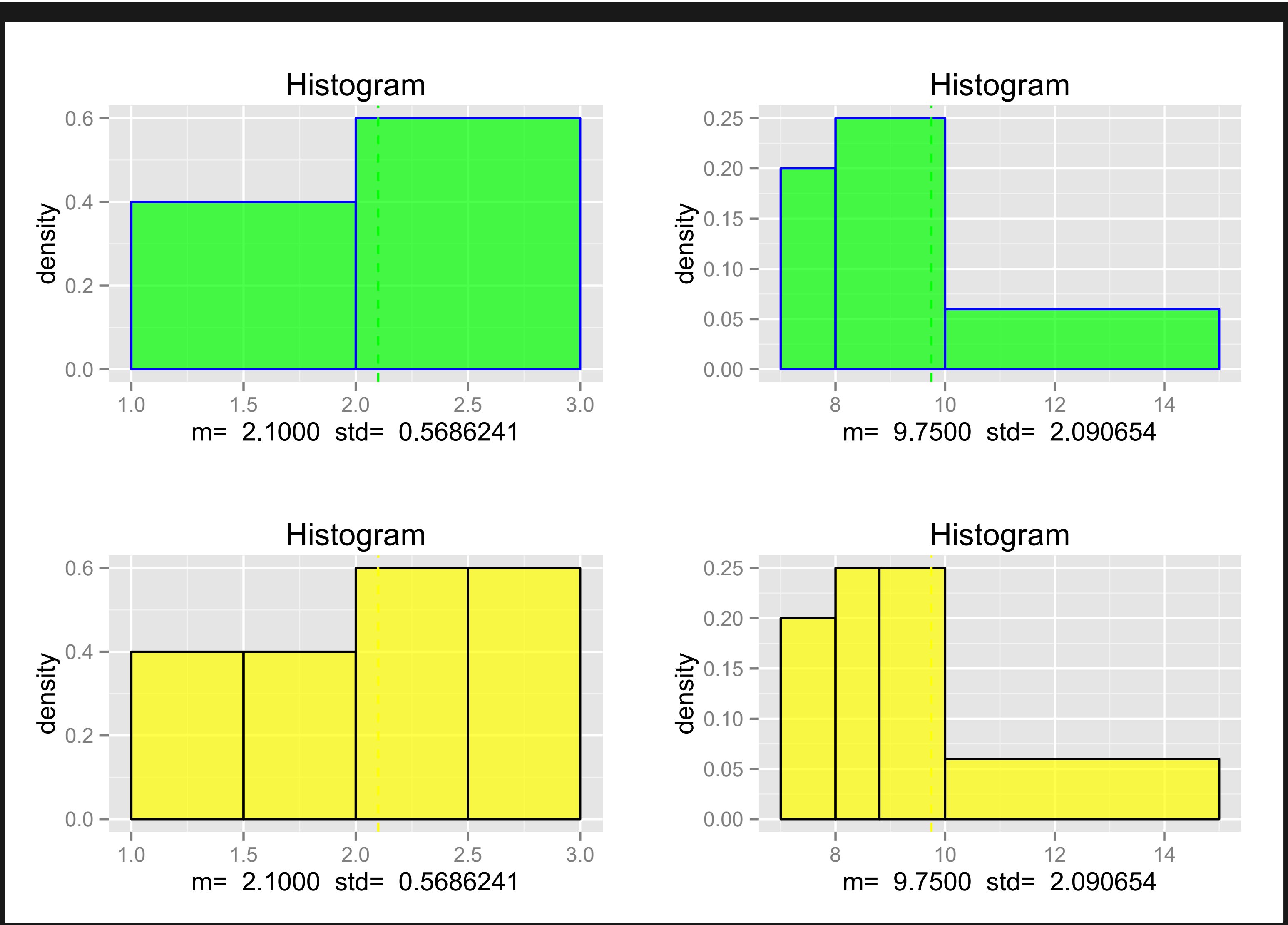
$$\int\limits_0^1 Q_f(p)Q_g(p)dp = \sum_{\ell=1}^{k}\left(c_{\ell,1}c_{\ell,2} + \frac{1}{3}r_{\ell,1}r_{\ell,2}\right)$$

How to recode two hstograms such that thay have the same number of bins and the same masses?

```r
registered=register(dist1, dist2)
#returns a list with 2 registered distributionH objects
```

The `register` method in action

```
dist1=distributionH(c(1,2,3),c(0, 0.4, 1))
dist2=distributionH(c(7,8,10,15),c(0, 0.2, 0.7, 1))
registered=register(dist1,dist2) ## register the two distributions
```



Returning on L_2 Wasserstein distance

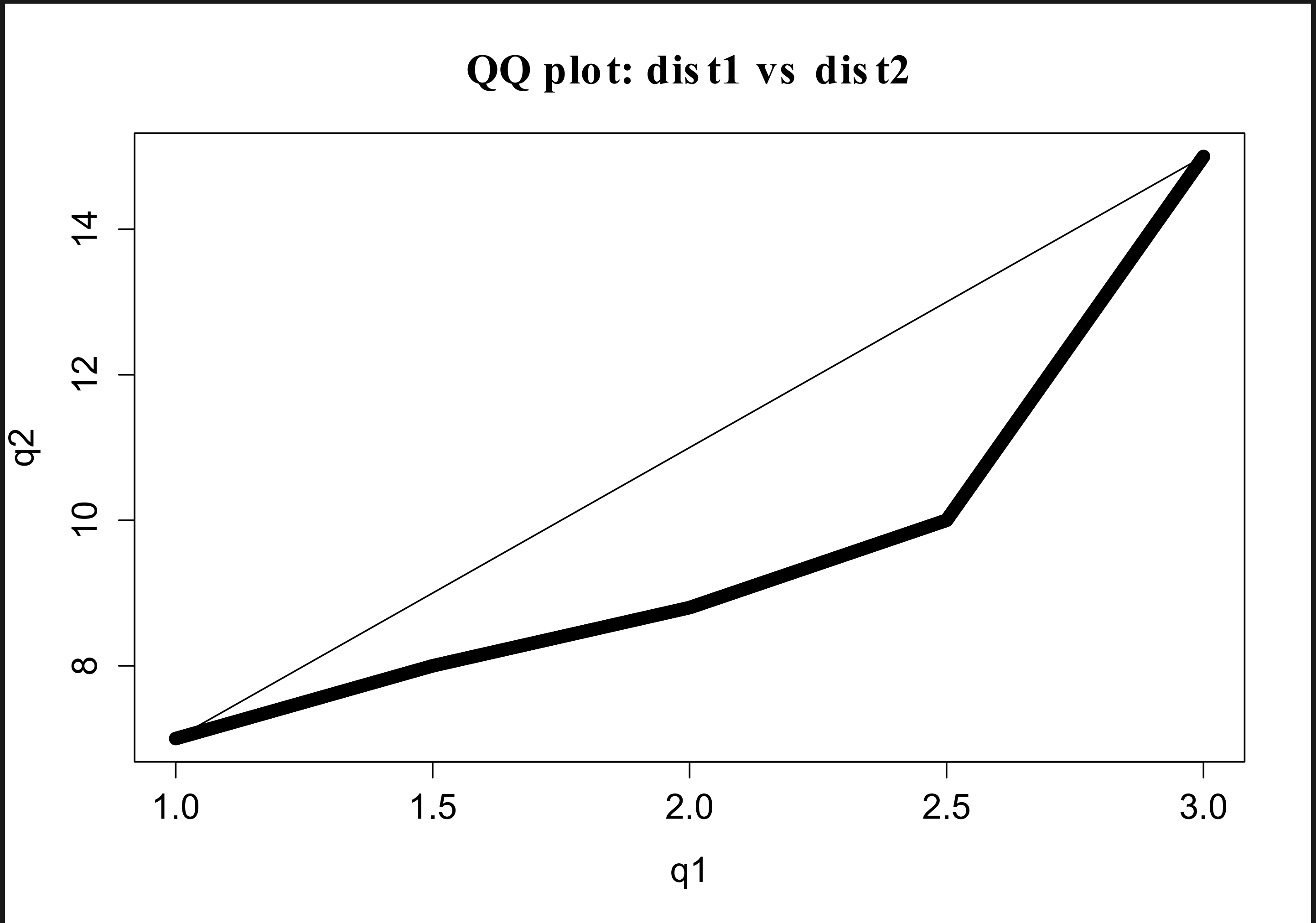
Using `register` method, the (Squared) L_2 Wasserstein distance between two histograms exactly:

```
## [1] 61.03667
##  SQ_W_dist   POSITION       SIZE      SHAPE        rQQ
## 61.0366667 58.5225000  2.3165745  0.1975922  0.9168940
```

$$d_W^2(f,g) = \underbrace{(\mu_f - \mu_g)^2}_{Position} + \underbrace{\underbrace{(\sigma_f - \sigma_g)^2}_{Size} + \underbrace{2\sigma_f\sigma_g\left[1 - \rho_{QQ}(f,g)\right]}_{Shape}}_{Variability}$$

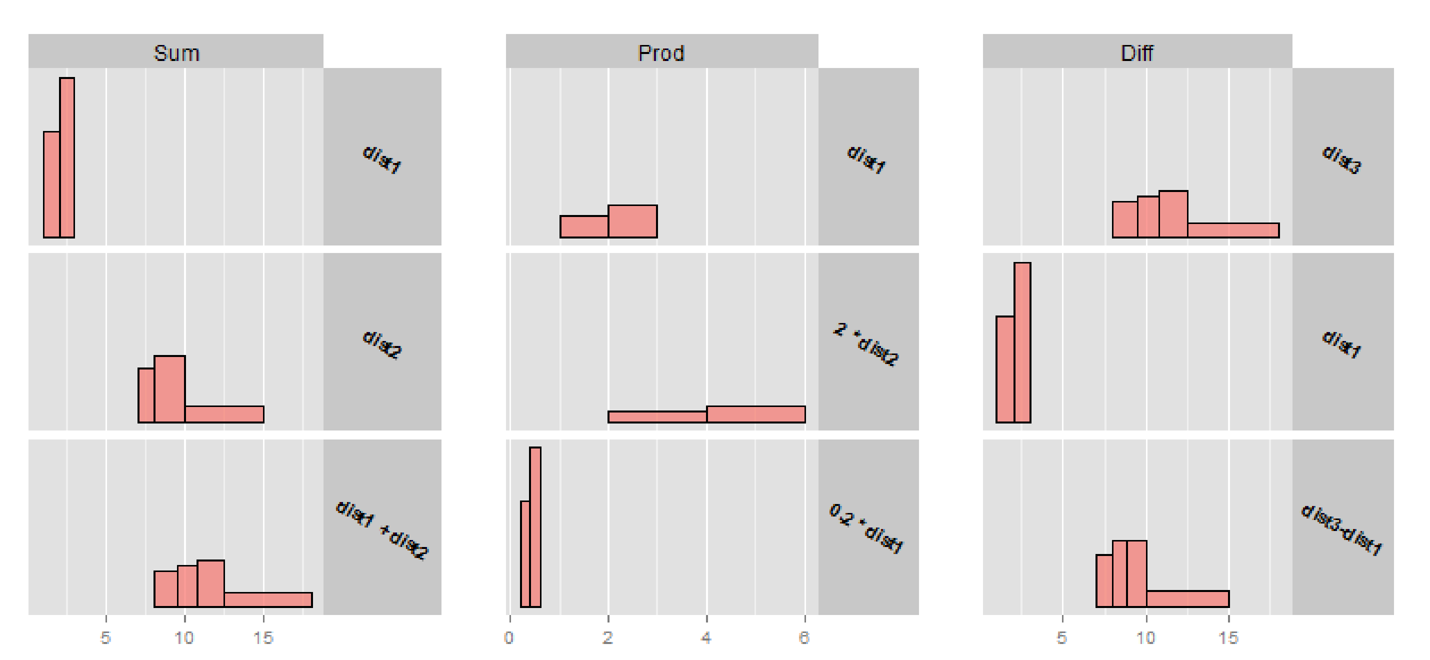
A bit more on `rQQ`

The method `rQQ` computes the Pearson correlation index of two quantile functions. It is equal to 1 when the distributions have the same shape (except for the means and the standard deviations).



Operations between distributionH objects

- Sum (Between two *qf*s, or a *qf* and a number)
- Multiplication (Between a *qf* and a positive number)
- Difference (When admissible, namely the result is a *qf*, between two *qf*s, or a *qf* and a number)



From raw data to distributionH objects

data2hist, a function for converting raw data to histogram-valued one.

```
data2hist( data,
           algo = "histogram",
           type = "combined",
           qua = 10, breaks = numeric(0), epsilon = 0.01
)
```

Arguments	Description
data	a set of numeric values.
algo	(optional) a string. Default is "histogram", i.e. the function “histogram” defined in the histogram package. If "base" the hist function is used. "FixedQuantiles" computes the histogram using as breaks a fixed number of quantiles. "ManualBreaks" computes a histogram where breaks are provided as a vector of values. "PolyLine" computes a histogram using a piecewise linear approximation of the empirical cumulative distribution function using the “Ramer-Douglas-Peucker algorithm”, https://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm. An epsilon parameter is required. The data are scaled in order to have a standard deviation equal to one.
type	(optional) a string. Default is combined and generates a histogram having regularly spaced breaks (i.e., equi-width bins) and irregularly spaced ones. The choice is done accordingly with the penalization method described in histogram. “regular” returns equi-width binned histograms, “irregular” returns a histogram without equi-width histograms.
qua	a positive integer to provide if algo="FixedQuantiles" is chosen. Default=10.
breaks	a vector of values to provide if algo="ManualBreaks" is chosen.
epsilon	a number between 0 and 1 to provide if algo="PolyLine" is chosen. Default=0.01.

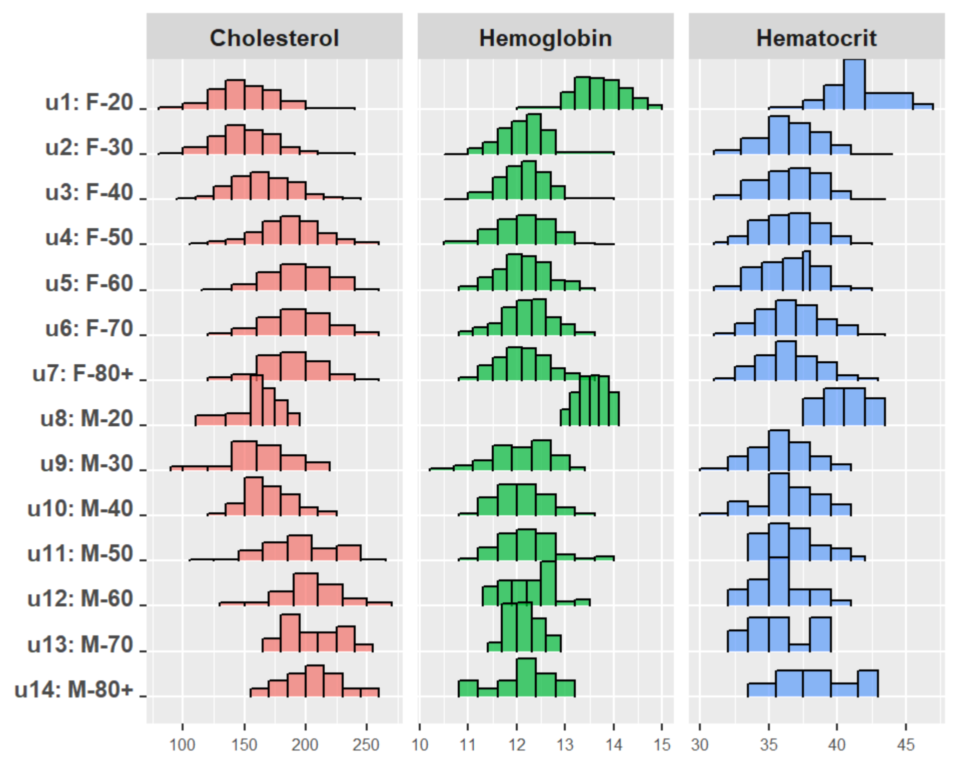
Output : A distributionH object, i.e. a distribution.

The `MatH`class: a histogram-valued data table

A data table of HD: the MatH object

A MatH object is a table (a matrix): each row is an individual each column is a histogram variable.

```
a matrix of distributions
 3  variables  14  rows
 each distibution in the cell is represented by the mean and the standard deviation
                Cholesterol                 Hemoglobin                 Hematocrit
u1: F-20  [m= 150.1  ,s= 26.336 ]  [m= 13.695  ,s= 0.55031 ] [m= 41.526  ,s= 2.1968 ]
u2: F-30 [m= 150.71  ,s= 25.284 ]  [m= 12.158  ,s= 0.52834 ] [m= 36.497  ,s= 2.1225 ]
u3: F-40 [m= 164.96  ,s= 25.334 ]  [m= 12.134  ,s= 0.50739 ] [m= 36.549  ,s= 2.2299 ]
u4: F-50 [m= 186.51  ,s= 26.655 ]  [m= 12.133  ,s= 0.58514 ]  [m= 36.48  ,s= 2.1985 ]
u5: F-60 [m= 194.03  ,s= 25.215 ]  [m= 12.145  ,s= 0.52031 ] [m= 36.341  ,s= 2.0979 ]
u6: F-70  [m= 193.2  ,s= 26.561 ]  [m= 12.205  ,s= 0.52258 ] [m= 36.703  ,s= 2.1818 ]
u7: F-80+ [m= 187.14  ,s= 24.592 ]  [m= 12.141  ,s= 0.55247 ] [m= 36.503  ,s= 2.1911 ]
u8: M-20 [m= 159.62  ,s= 19.844 ]  [m= 13.557  ,s= 0.29974 ]  [m= 40.5  ,s= 1.6358 ]
u9: M-30 [m= 164.43  ,s= 26.486 ]  [m= 12.088  ,s= 0.62237 ] [m= 35.914  ,s= 2.1141 ]
u10: M-40 [m= 170.06  ,s= 20.011 ]  [m= 12.092  ,s= 0.52656 ] [m= 36.456  ,s= 2.2476 ]
u11: M-50 [m= 194.22  ,s= 30.165 ]  [m= 12.214  ,s= 0.59708 ]  [m= 36.72  ,s= 2.0024 ]
u12: M-60 [m= 203.36  ,s= 26.223 ]  [m= 12.245  ,s= 0.50862 ] [m= 35.814  ,s= 2.0083 ]
u13: M-70 [m= 205.66  ,s= 22.499 ]  [m= 12.15  ,s= 0.33425 ]   [m= 35.75  ,s= 2.1651 ]
u14: M-80+ [m= 205.48  ,s= 23.537 ]   [m= 12.12  ,s= 0.6163 ]   [m= 38.45  ,s= 2.6158 ]
```



The `MatH` class and its initialization

- `x` is a list of `distributionH` objects
- `nrows` is the number of rows (the individuals)
- `ncols` is the number of columns (the variables)
- `rownames` is a vector of strings with the labels of the individuals
- `varnames` is a vector of strings with the labels of variables
- `by.row` indicates if the matrix must be filled by row (TRUE) or by column (FALSE this is the default)

An example of creation of a new MatH object

```r
##---- create a list of six distributionH objects
ListOfDist<-vector("list",6)
ListOfDist[[1]]<-distributionH(c(1,2,3),c(0, 0.4, 1))
ListOfDist[[2]]<-distributionH(c(7,8,10,15),c(0, 0.2, 0.7, 1))
ListOfDist[[3]]<-distributionH(c(9,11,20),c(0, 0.5, 1))
ListOfDist[[4]]<-distributionH(c(2,5,8),c(0, 0.3, 1))
ListOfDist[[5]]<-distributionH(c(8,10,15),c(0,  0.75, 1))
ListOfDist[[6]]<-distributionH(c(20,22,24),c(0, 0.12, 1))

## create a MatH object filling it by columns
MyMAT=MatH(x=ListOfDist,nrows=3,ncols=2,
  rownames=c("I1","I2","I3"), varnames=c("Var1","Var2"),by.row=FALSE)

#bulding an empty 10 by 4 matrix of histograms
empty.MAT=MatH(nrows=10,ncols=4)
```

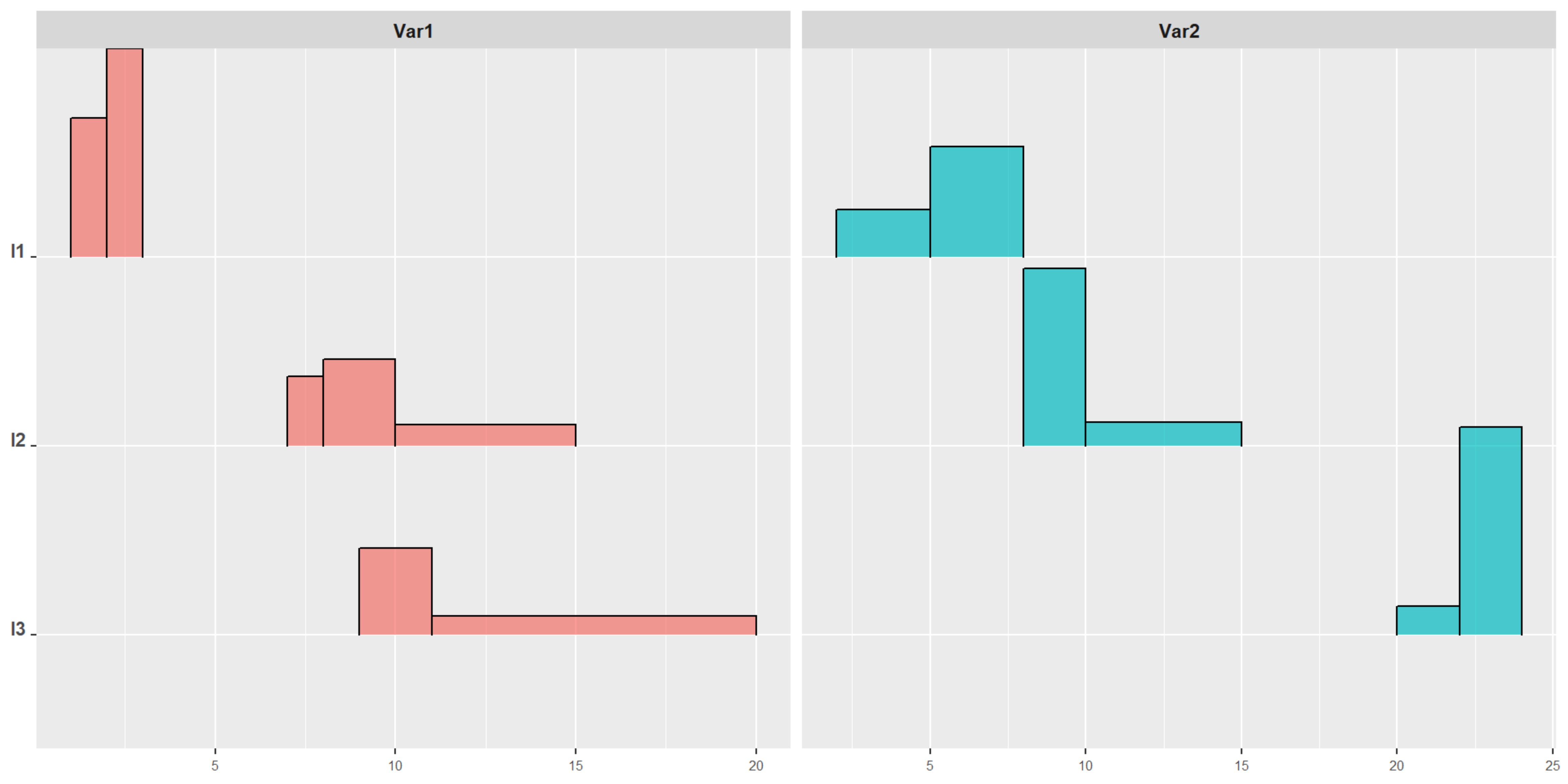
show method

```r
show(MyMAT) #or simply type MyMAT
```

```
a matrix of distributions
 2  variables  3  rows
 each distibution in the cell is represented by the mean and the standard deviation
              Var1                     Var2
I1  [m= 2.1  ,s= 0.56862 ]   [m= 5.6  ,s= 1.6248 ]
I2  [m= 9.75  ,s= 2.0907 ]  [m= 9.875  ,s= 1.7515 ]
I3 [m= 12.75  ,s= 3.3323 ]  [m= 22.76  ,s= 0.86933 ]
```

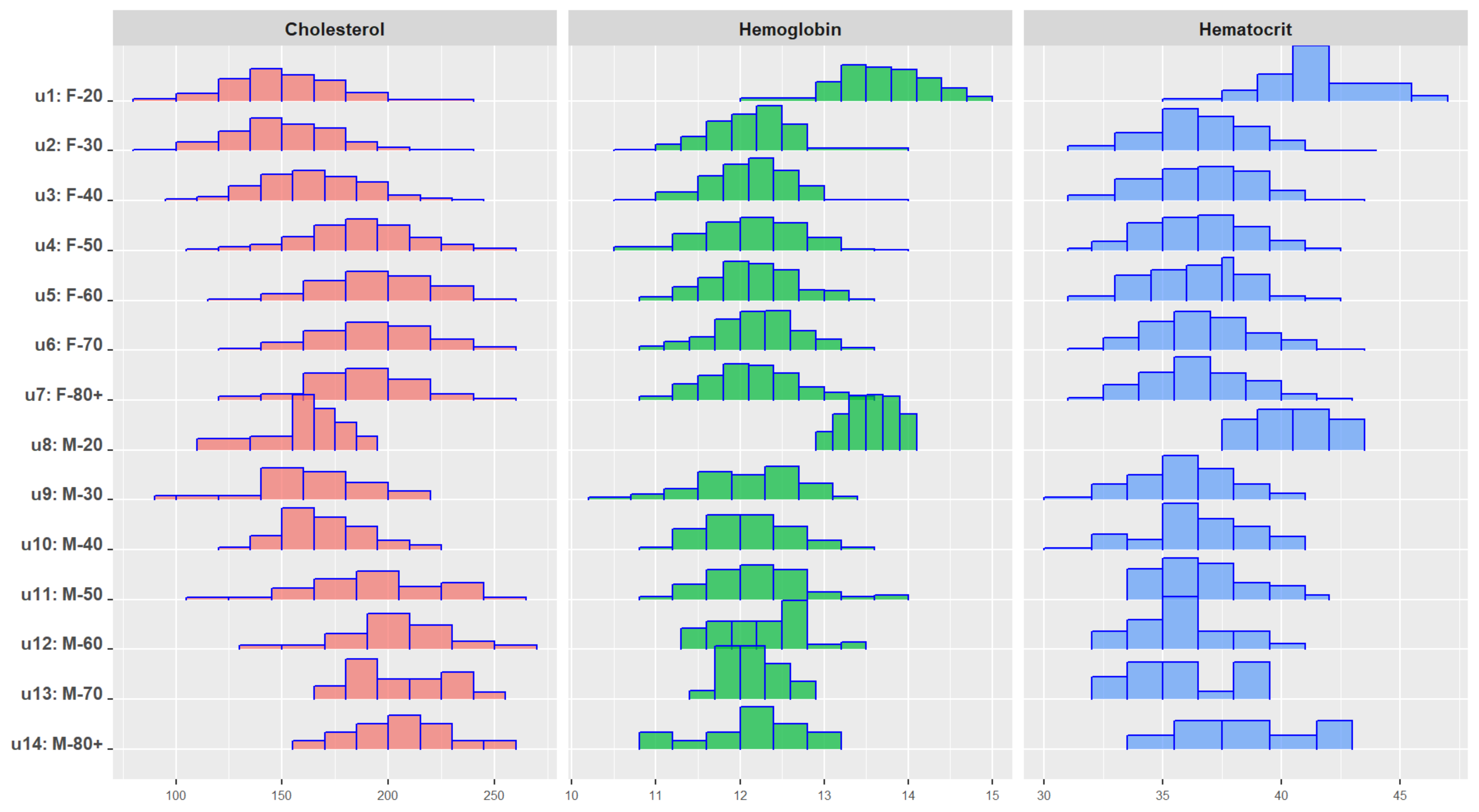
plot method

```r
plot(MyMAT)
```



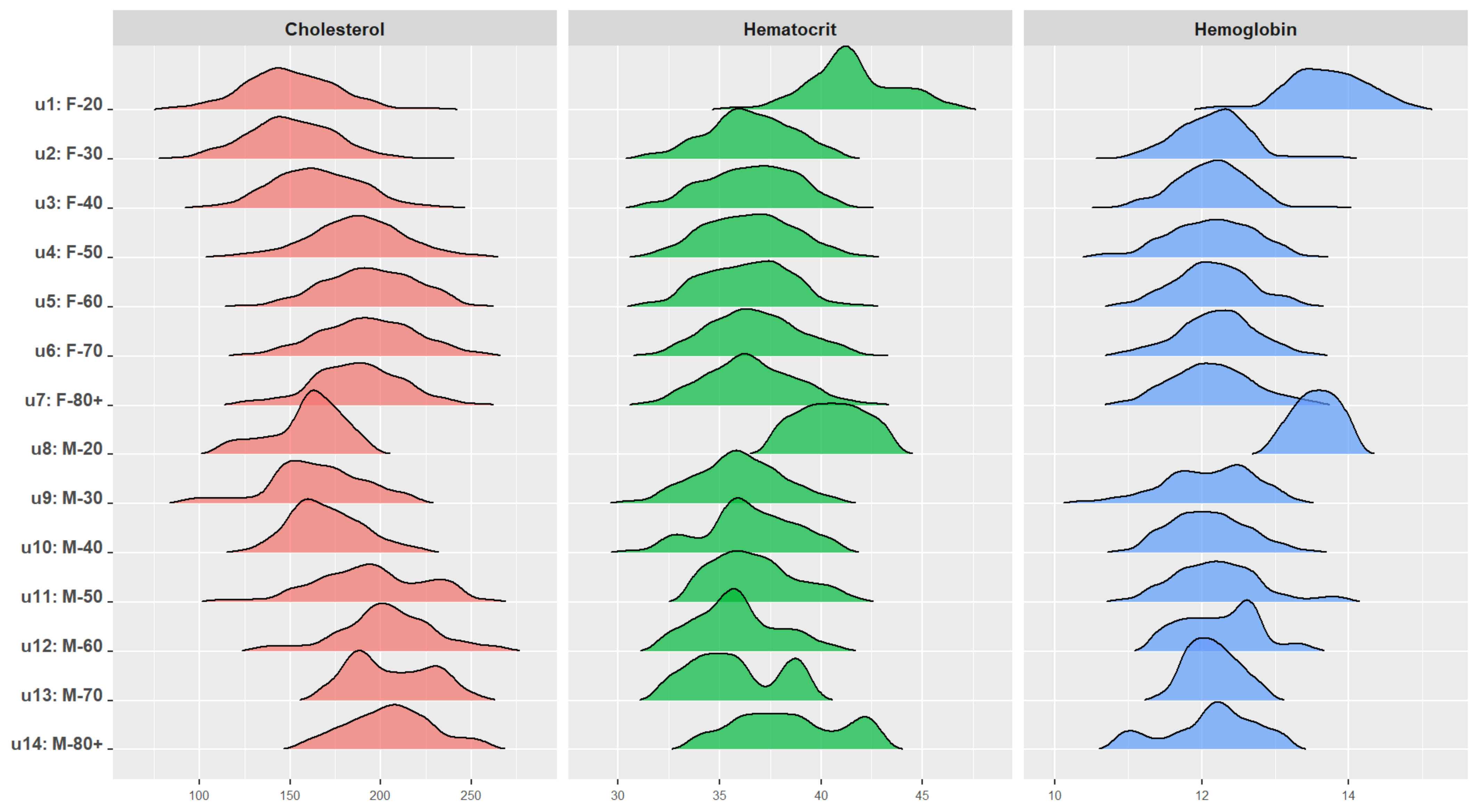
More on ploting a MatH (1/3): basic plot

```
plot(BLOOD, type="HISTO",  border="blue") #plots a matrix of histograms
```



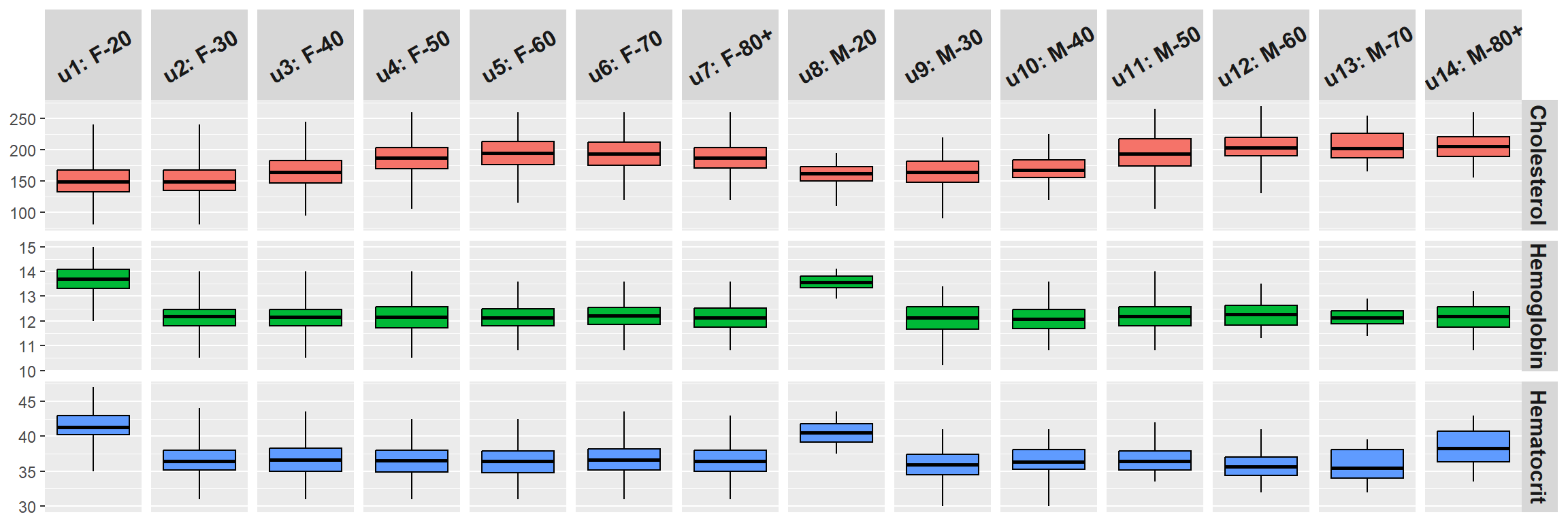
More on ploting a MatH (2/3): density plot

```
plot(BLOOD, type="DENS",  border="blue") #plots a matrix of densities
```



More on ploting a MatH (3/3): boxplots

```
plot(BLOOD, type="BOXPLOT") #plots a  boxplots
```



Main methods for the `MatH` class

For accessing to some basic information

- `get.MatH.ncols` returns the number of columns
- `get.MatH.nrows` returns the number of rows
- `get.MatH.rownames` returns the list of the row-labels
- `get.MatH.varnames` return the list of labels of thevariables (the columns)
- `get.MatH.stats` return a matrix of a basic statistic computed for each cell of the matrix (some examples follow).

An example of get.MatH.stats method

The get.MatH.stats returns a list containing the name of the basic statistics computed and a $mat containing a matrix of numbers with the same size of the MatH object

```
get.MatH.stats(BLOOD) # the means of the distributions in BLOOD dataset
get.MatH.stats(BLOOD,stat='median') # the medians of the distributions
get.MatH.stats(BLOOD,stat='quantile', prob=0.5) #the same as median
get.MatH.stats(BLOOD,stat='min') # minima of the distributions
get.MatH.stats(BLOOD,stat='quantile', prob=0) #the same as min
get.MatH.stats(BLOOD,stat='max') # maxima of the distributions
get.MatH.stats(BLOOD,stat='quantile', prob=1) #the same as max
get.MatH.stats(BLOOD,stat='std') # standard deviations
get.MatH.stats(BLOOD,stat='skewness') #skewness indices
get.MatH.stats(BLOOD,stat='kurtosis') #kurtosis indices
get.MatH.stats(BLOOD,stat='quantile',prob=0.05)
```

Functions useful for manipulating

Useful for the histogram trick

- `registerMH(MyMat)`, returns a new `MatH` object with all the distributions transformed. All the distributions have the same number of bins each one containing the same mass (It is useful for computing exactly the basic statistics based on L_2 Wasserstein metric)

Fuctions for subsetting of for binding `MatH` objects

- `WH.bind(MAT1, MAT2, byrow=TRUE)` attaches MAT2 on the right of MAT1. The two `MatH` objects must have the same number of rows;
- `WH.bind(MAT1, MAT2, byrow=FALSE)` attaches MAT2 under MAT1. The two `MatH` objects must have the same number of columns;
- `[,]` an overloaded method for sub-setting a matrix. It returns a new `MatH` object.

Subsetting example

```
BLOOD[10:14,1:2]
## a matrix of distributions
##  2  variables  5  rows
##  each distibution in the cell is represented by the mean and the standard deviation
##                   Cholesterol               Hemoglobin
## u10: M-40 [m= 170.06  ,s= 20.011 ]  [m= 12.092  ,s= 0.52656 ]
## u11: M-50 [m= 194.22  ,s= 30.165 ]  [m= 12.214  ,s= 0.59708 ]
## u12: M-60 [m= 203.36  ,s= 26.223 ]  [m= 12.245  ,s= 0.50862 ]
## u13: M-70 [m= 205.66  ,s= 22.499 ]  [m= 12.15  ,s= 0.33425 ]
## u14: M-80+ [m= 205.48  ,s= 23.537 ]   [m= 12.12  ,s= 0.6163 ]

BLOOD[2,3] #ATTENTION: returns a 1x1 MatH, and not a distributionH
## a matrix of distributions
##  1  variables  1  rows
##  each distibution in the cell is represented by the mean and the standard deviation
##                  Hematocrit
## u2: F-30 [m= 36.497  ,s= 2.1225 ]
```

Extracting a `distributionH` from a `MatH`: how to do it?

```
# instead of BLOOD[2,3]
# you must use

BLOOD@M[2,3][[1]] #Not attractive, it needs improvements!
##               X             p
## Bin_1   [ 31 ; 33 )         0.046
## Bin_2   [ 33 ; 35 )         0.171
## Bin_3 [ 35 ; 36.5 )         0.295
## Bin_4 [ 36.5 ; 38 )         0.243
## Bin_5 [ 38 ; 39.5 )          0.17
## Bin_6 [ 39.5 ; 41 )         0.072
## Bin_7   [ 41 ; 44 ]         0.003
##
##  mean =  36.497   std   =  2.12245714522903
##
```

Matrix operation between `MatH` objects

Methods based on L_2 Wasserstein norm for summing and multiplying matrices

- `WH.mat.sum` performs a classic cell by cell sum. In particular, the result is a new `MatH` object having in each cell a distribution associated with the quantile function resulting from the sum of the corresponding quantile functions (It is a sum consistent with the Wasserstein metric). As usual, the matrix must have the same dimensions.

```
MAT.sum=WH.mat.sum(MyMAT1,MyMAT2)
```

- `WH.mat.prod` performs the matrix multiplication of two `MatH` objects. It returns a **matrix of numbers** according to the dot product defined for two distributions and associated with the L_2 Wasserstein metric. It is possible also to consider trasposition of matrices.

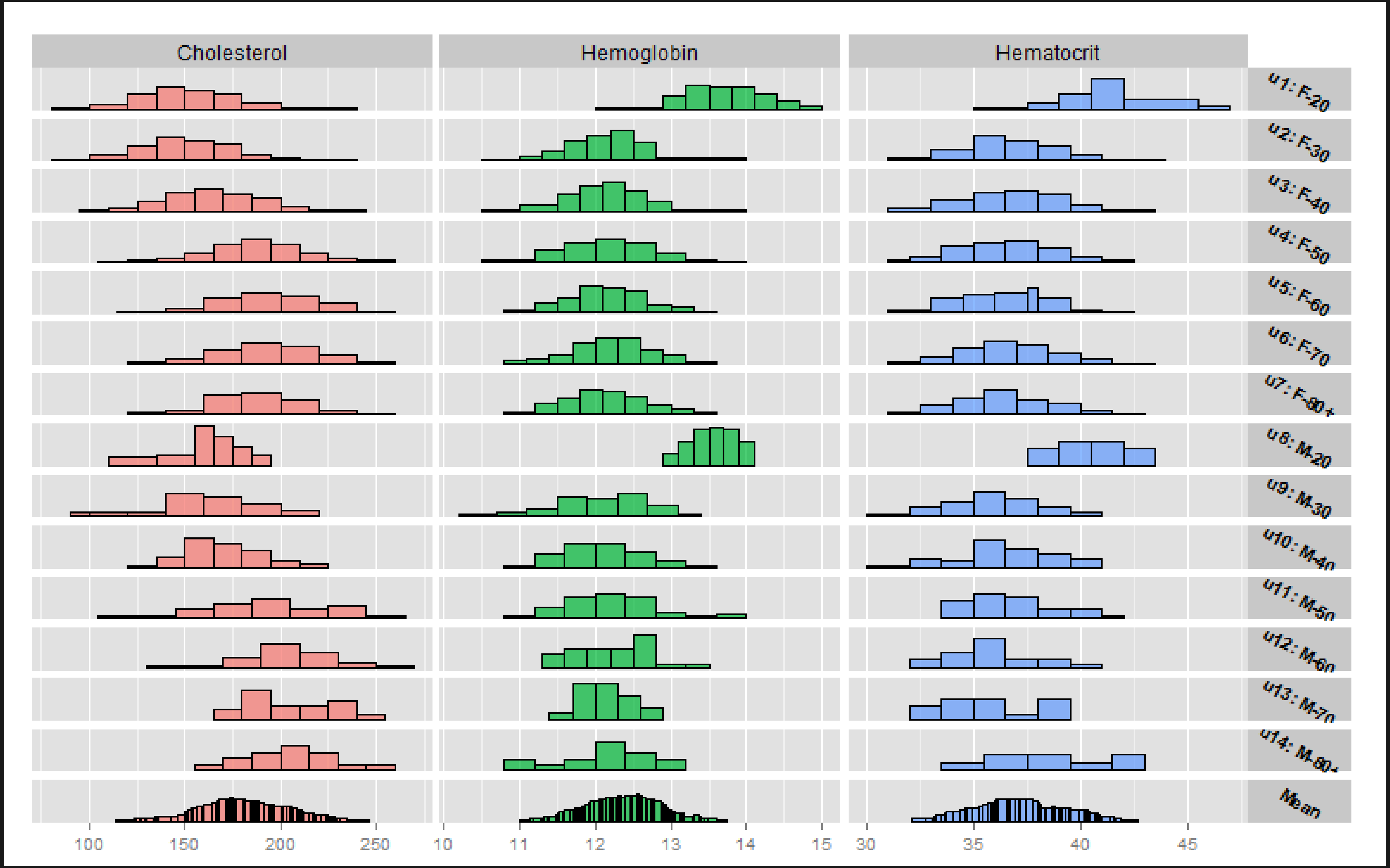
```
MAT.prod=WH.mat.prod(MyMAT1,MyMAT2, traspose1=FALSE, traspose2=FALSE)
MAT.prod=WH.mat.prod(MyMAT1,MyMAT2, traspose1=TRUE, traspose2=FALSE)
MAT.prod=WH.mat.prod(MyMAT1,MyMAT2, traspose1=FALSE, traspose2=TRUE)
```

Methods for univariate and bivariate statistics of histogram variables: the mean distribution

- `WH.vec.mean` computes a `distributionH` that is the mean distribution of a vector or a matrix of distributions (a `MatH` object). The mean distribution is computed accordingly to the sum based on the L_2 Wasserstein distance, namely, it is the distribution associated with the average *quantile function* of the *quantile functions* of the vector of distributions. It is possible also to assign weights to the elements of vetor in order to obtain a weighted mean.
- `WH.vec.sum` same as `WH.vec.mean`, but computes only the sum.

```r
WH.vec.mean(BLOOD[,1]) #returns the average distribution of
# the first variable
WH.vec.mean(BLOOD[,1], w = runif(get.MatH.nrows(MyMAT)))
# returns a random weighted average
```

A graphic example of `WH.vec.mean`



Methods for univariate and bivariate statistics of histogram variables: the variability and the association indexes

These methods returns a matrix of numbers accordingly to the number of the compared variables. The formulas are those presented in (R. Irpino A.and Verde 2015).

Sum of squares

- `WH.SSQ(MyMAT, w=w)` computes the wheighted Sum of Squares and the Sum of products of the deviations from the means of a `MatH` object. The weights are associated to the rows, if missing, the weights are considered equal. The result is square matrix of dimension equal to the number of variables of the `MatH` object.
- `WH.SSQ2(MyMAT1, MyMAT2, w=w)` computes the wheighted Sum of products of the deviations from the means of two `MatH` objects. The two matrices must have the same number of rows. The result is rectangular matrix with rows equal to the number of variables of the first `MatH` object and columns the number of variables of the first `MatH` object.

Using these it is possible to compute covariances and correlations.

Covariance and correlation indexes

Variance, covariance and correlation

- `WH.var.covar(MyMAT, w=w)` computes the variance-covariance matrix of a `MatH` object.
- `WH.var.covar2(MyMAT1, MyMAT2, w=w)` computes a covariance matrix between two `MatH` objects.
- `WH.correlation(MyMAT, w=w)` computes the correlation matrix of a `MatH` object.
- `WH.correlation2(MyMAT1, MyMAT2, w=w)` computes the correlation matrix of two `MatH` objects.

```
WH.var.covar(BLOOD)
##             Cholesterol Hemoglobin  Hematocrit
## Cholesterol  388.137633 -5.0005134 -14.9202378
## Hemoglobin    -5.000513  0.2802215   0.8266558
## Hematocrit   -14.920238  0.8266558   2.9779786
WH.correlation(BLOOD)
##             Cholesterol Hemoglobin Hematocrit
## Cholesterol   1.0000000 -0.4794806 -0.4388560
## Hemoglobin   -0.4794806  1.0000000  0.9049264
## Hematocrit   -0.4388560  0.9049264  1.0000000
```

Examples of how to compute basic statistics of variability for a single variable

```
# the variance of a histogram variable
VAR.Choresterol=WH.var.covar(BLOOD[,1])
as.numeric(VAR.Choresterol)
## [1] 388.1376
# the standard deviation
STD.Choresterol=sqrt(VAR.Choresterol)
as.numeric(STD.Choresterol)
## [1] 19.70121
```

The `TdistributionH` class: a class for a histogram-valued data observed along time

`TdistributionH` class (Not completely developed!!)

It is essentially a specilized class of `distributionH` equipped with a time-related information. Being a distribution, we considered to specialize the class such that it can contains

- a `tstamp`, a numeric value that represents a time stamp, or;
- a `period`, a list containing two slots `start` and `end`, two numbers (for now, a time/date dormat will be chosen in the future), indicating the initial and the final period of observation. Indeed, we imagine that a histogram (or a distribution) is the set of values observed during a time period.

```
My.Time.histo  = new('TdistributionH',tstamp=1,
                      x=dist1@x, p=dist1@p)
My.Period.histo= new('TdistributionH',period=list(start=1,end=3),
                      x=dist1@x, p=dist1@p)
```

The HTS class: a class for histogram time series

A HTS is a list of TdistributionH

Initializing a HTS

The list of TdistributionH objects are organized into a vector of dimension equal to epoc. The slot containing the vector is called data. If we need to see what is in the HTS we need to call elements from @data slot.

```
# RetHTS is a HTS of the returns five-minutes returns of DOLL/YEN changes
# rates observed along 108 days (it is one of the dataset available in the pkg)
RetHTS@data[2]
## [[1]]
##                   X                 p
## Bin_1  [ -0.1 ; -0.06 )           0.03833
## Bin_2 [ -0.06 ; -0.03 )            0.1394
## Bin_3  [ -0.03 ; 0.03 )            0.7038
## Bin_4   [ 0.03 ; 0.06 )           0.09059
## Bin_5   [ 0.06 ; 0.12 ]           0.02787
##
##  mean =  -0.00275259208362382   std  =  0.0342206481885284
##
```

Main methods

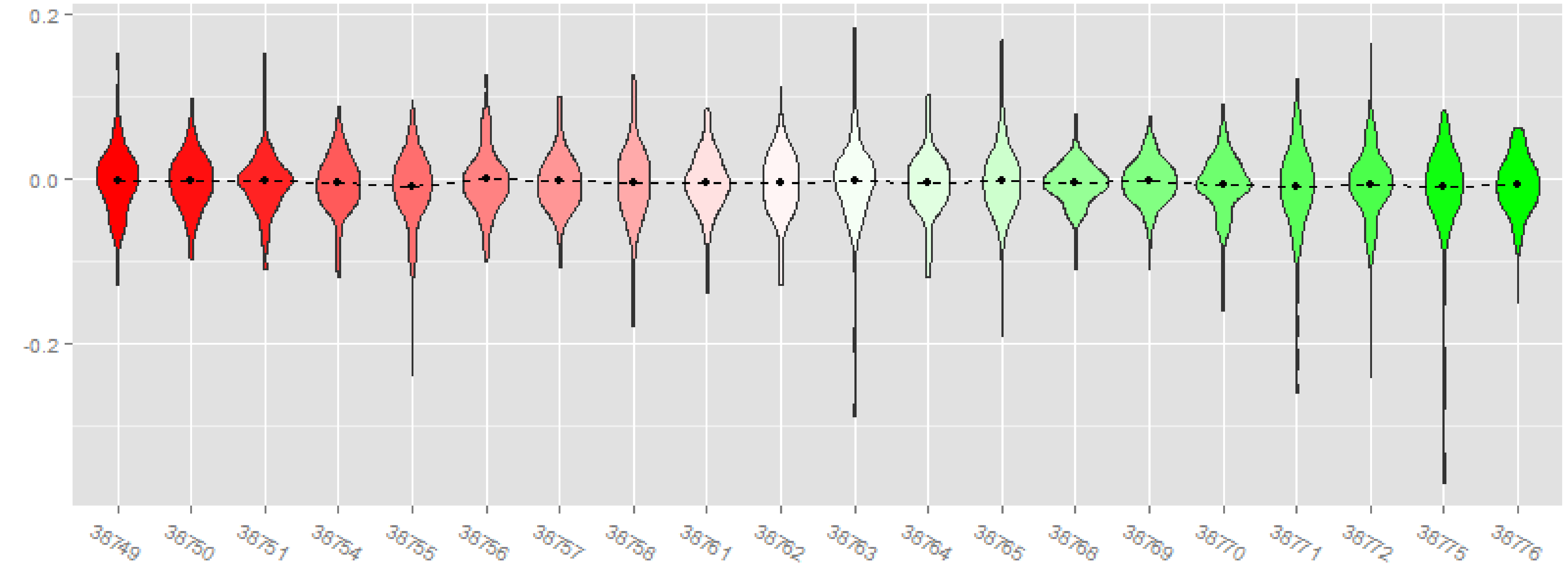
- `subsetHTS(HTS, from, to)` extracts a piece (in the time sense) of HTS
- `show(HTS)` returns a list of valuesand basic statistics related to the distributions in the HTS (wee see the results in the next slide)
- `plot(HTS,...)` shows a plot of the HTS, (we see examples in the next slides)

The show method

```
##      Tstamp T_period_start T_period_end          mean        std         skew
## 1     38749          38749        38749 -0.0019860297 0.04224402  0.696353637
## 2     38750          38750        38750 -0.0027525921 0.03422065  0.161710205
## 3     38751          38751        38751 -0.0022545180 0.04252923  1.054658950
## 4     38754          38754        38754 -0.0056181618 0.03660866 -0.287735490
## 5     38755          38755        38755 -0.0108187759 0.04469958 -0.691931506
## 6     38756          38756        38756 -0.0015679186 0.03932478  0.676302020
## 7     38757          38757        38757 -0.0038849961 0.03521140  0.474190871
## 8     38758          38758        38758 -0.0057090553 0.05180512 -0.108275213
## 9     38761          38761        38761 -0.0048545204 0.03741751 -0.264327106
## 10    38762          38762        38762 -0.0060801071 0.04140493  0.276796508
## 11    38763          38763        38763 -0.0018814832 0.05313136 -0.151476128
## 12    38764          38764        38764 -0.0060526111 0.03971651 -0.041866347
## 13    38765          38765        38765 -0.0019454181 0.05084949  0.354931260
## 14    38768          38768        38768 -0.0051052438 0.02793509  0.009895520
## 15    38769          38769        38769 -0.0034561197 0.02806190  0.028340867
## 16    38770          38770        38770 -0.0069337719 0.03726113 -0.602640682
## 17    38771          38771        38771 -0.0106445600 0.05501497 -0.995706447
## 18    38772          38772        38772 -0.0073090483 0.05280985 -0.254510473
## 19    38775          38775        38775 -0.0092402357 0.04994473 -2.456202932
## 20    38776          38776        38776 -0.0070208812 0.03425427 -0.170867642
```

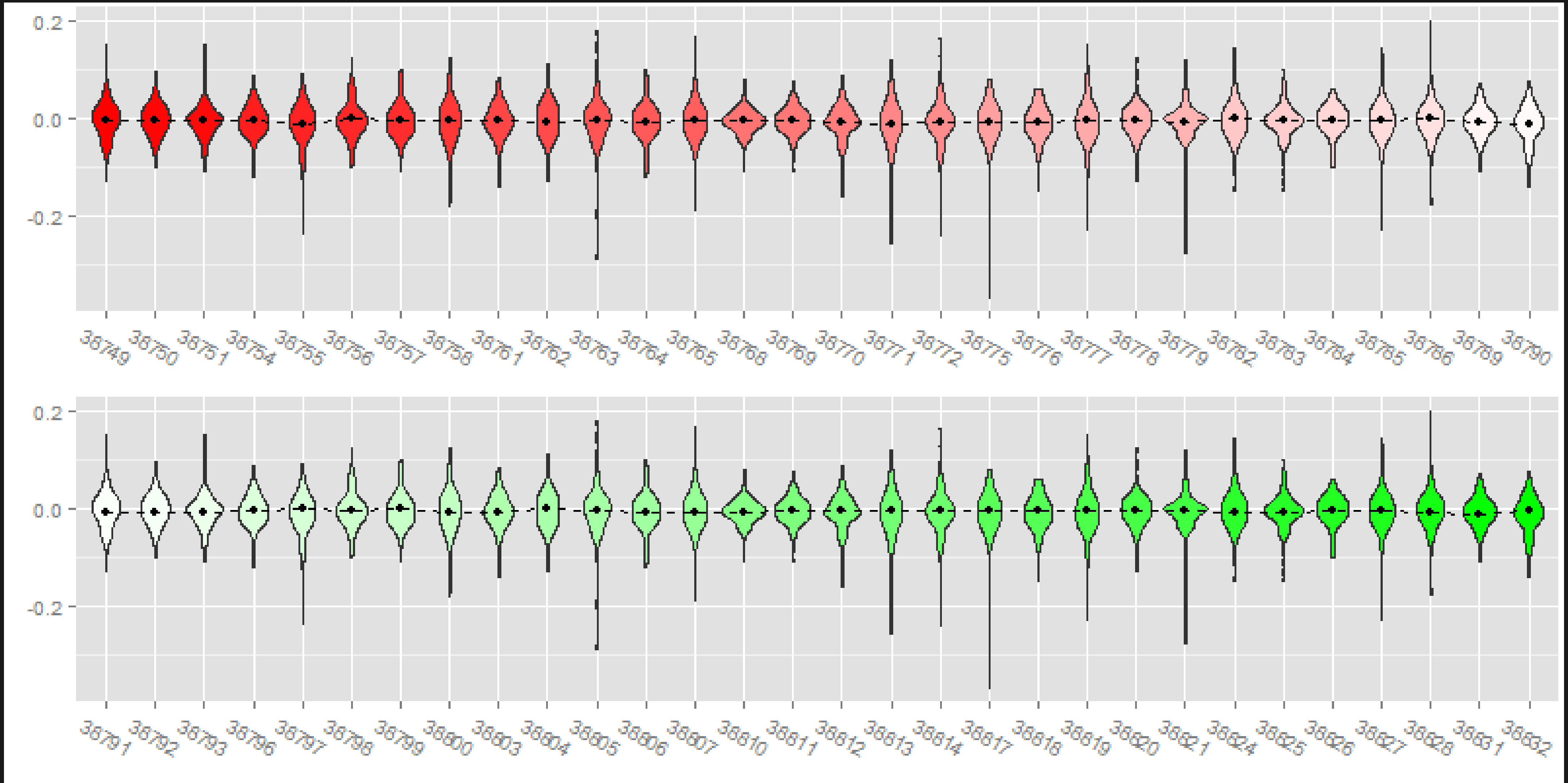
The plot method (1/4)

```
#plots the first 20 elements RetHTS dataset
plot(subsetHTS(RetHTS,from=1,to=20))
```



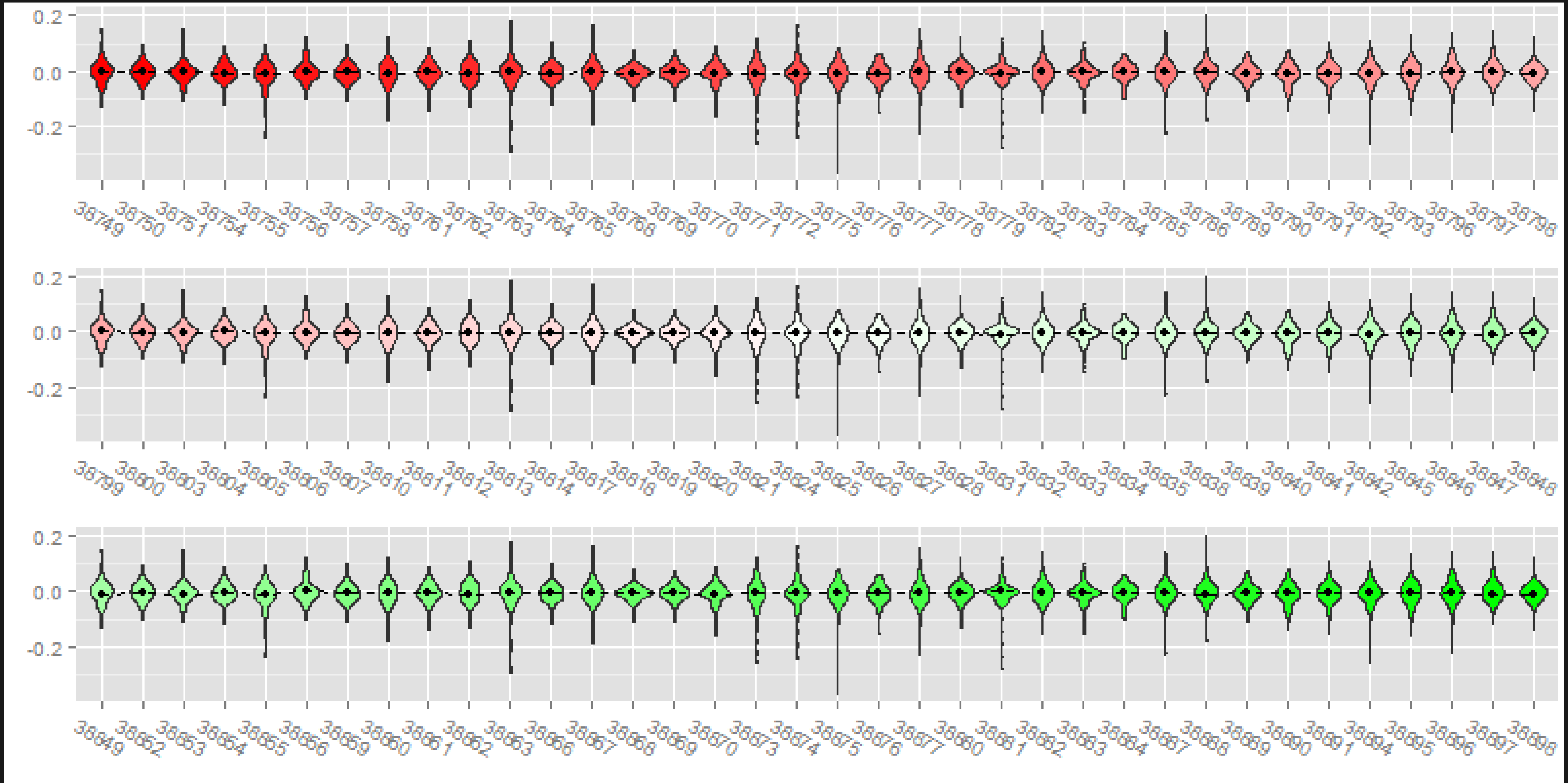
The plot method (2/4)

```
# plots the first 60 elements RetHTS dataset divided in
# pieces of 30 epocs
plot(subsetHTS(RetHTS,from=1,to=60), maxno.perplot=30)
```



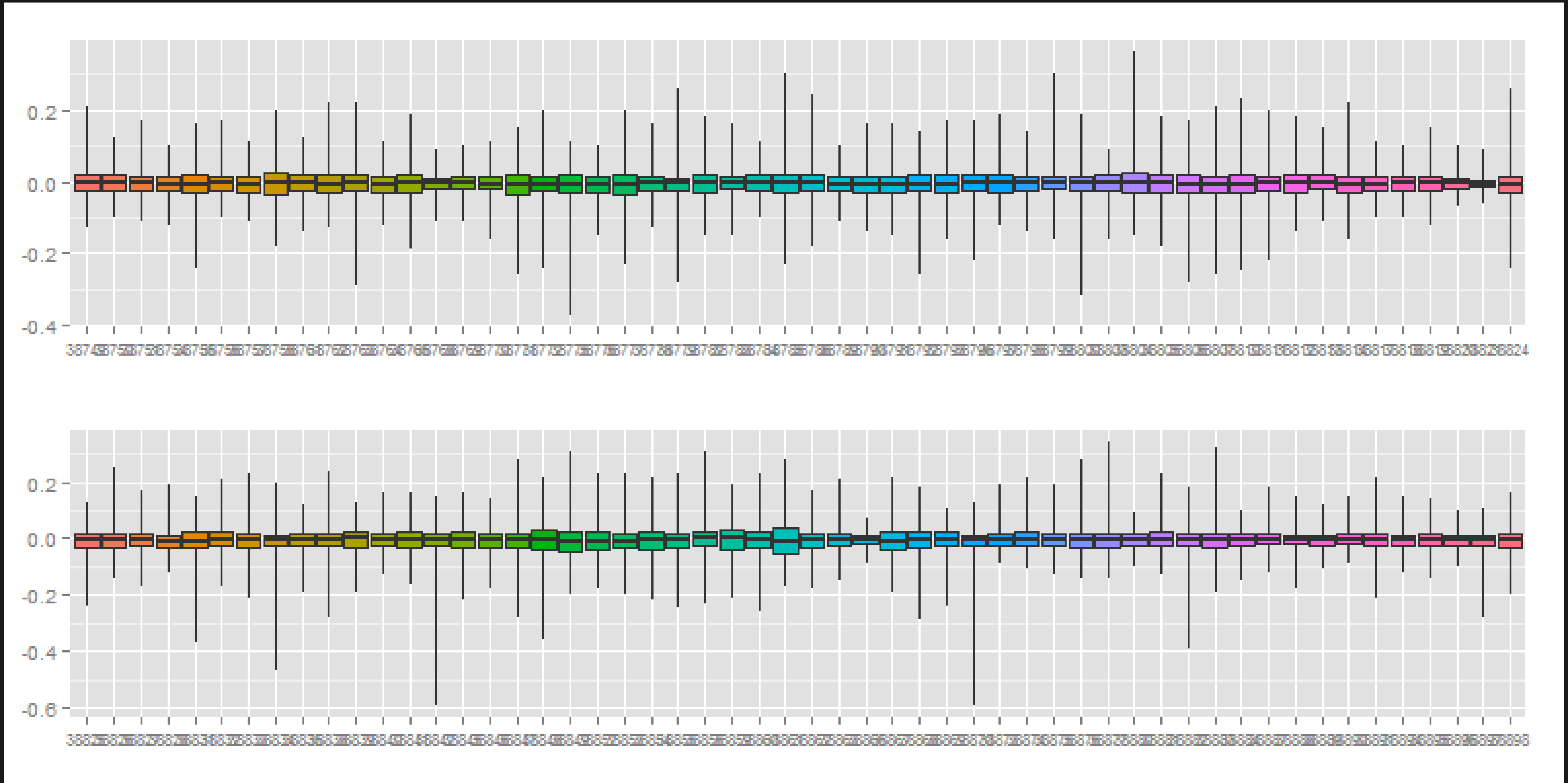
The plot method (3/4)

```
# plots HTS elements RetHTS dataset divided in
# pieces of 36 epocs
plot(RetHTS, maxno.perplot=36)
```



The plot method (4/4): plotting using boxplots

```r
# plots HTS elements RetHTS dataset divided in
# pieces of 54 epocs
plot(RetHTS, type="BOXPLOT", maxno.perplot=54)
```



In the next presentations

Implementation of methods for data analysis of histogram-valued data tables

- Principal component analysis
  - of a single Histogram variable
  - of several histogram variables
- Regression
  - Two components regression analysis

- Clustering methods
  - Dynamic clustering (a generalization of k-means algorithm)
  - Adaptive distances-based dynamic clustering
  - Hierarchical clustering
  - Kohonen Self Organizing Maps
  - Fuzzy c-means
  - Adaptive distances-based Fuzzy c-means

References

Gilchrist, W. 2000. *Statistical Modelling with Quantile Functions*. Abingdon: CRC Press.

Irpino, Antonio, and Elvira Romano. 2007. “Optimal Histogram Representation of Large Data Sets: Fisher Vs Piecewise Linear Approximation.” In *EGC*, edited by Monique Noirhomme-Fraiture and Gilles Venturini, RNTI-E-9:99–110. Revue Des Nouvelles Technologies de l’information. Cépaduès-Éditions.

Irpino, R., A.and Verde. 2015. “Basic Statistics for Distributional Symbolic Variables: A New Metric-Based Approach.” *Advances in Data Analysis and Classification* 9 (2): 143–75. https://doi.org/10.1007/s11634-014-0176-4.