# GMD Vignette:

Generic histogram construction, generic distance measure, cluster analysis with evaluation and visualization

Xiaobei Zhao* Albin Sandelin*

February 6, 2012

## (Version: 0.3.1.1)

**Abstract**

The purpose of this GMD Vignette is to show how to get started with the R package GMD. *GMD* denotes **Generalized Minimum Distance between distributions**, which is a true metric that measures the distance between the shapes of any two discrete numerical distributions (*e.g.* histograms).

The vignette includes a brief introduction, an example to illustrate the concepts and the implementation of GMD and case studies that were carried out using classical data sets (*e.g. iris*) and high-throughput sequencing data (*e.g. ChIP-seq*) from biology experiments. The appendix on page 15 contains an overview of package functionality, and examples using primary functions in histogram construction (the `ghist` function), how to measure distance between distributions (the `gdist` function), cluster analysis with evaluation (the ``elbow'' method in the `css` function) and visualization (the `heatmap.3` function).

**Keywords:** histogram, distance, metric, non-parametric, cluster analysis, hierarchical clustering, sum-of-squares, heatmap.3

## Contents



*Bioinformatics Centre, University of Copenhagen, Department of Biology and Biotech Research and Innovation Centre, Ole Maaløes Vej 5, DK-2200, Copenhagen, Denmark. Email: xiaobei@binf.ku.dk ; albin@binf.ku.dk

# 1 Introduction

Similar to the Earth Mover's distance, `Generalized Minimum distance of Distributions (GMD)` (based on MDPA - Minimum Difference of Pair Assignment [3]) is a true metric of the similarity between the shapes of two histograms[1]. Considering two normalized histograms $A$ and $B$, GMD measures their similarity by counting the necessary "shifts" of elements between the bins that have to be performed to transform distribution $A$ into distribution $B$.

The `GMD` package provides classes and methods for computing `GMD` in R [5]. The algorithm has been implemented in C to interface with R for efficient computation. The package also includes downstream cluster analysis in function `css` (A.4 on page 24) that use a pairwise distance matrix to make partitions given variant criteria, including the "elbow" rule as discussed in [7] or desired number of clusters. In addition, the function `heatmap.3` (A.5 on page 27) integrates the visualization of the hierarchical clustering in dendrogram, the distance measure

[1] In statistics (and many other fields), histogram refers to a graphical representation of category frequencies in the data. Here, we use this term in a more mathematical sense, defined as a function that counts categorical data or a result returned by such a function.

in heatmap and graphical representations of summary statistics of the resulting clusters or the overall partition. For more flexibility, the function `heatmap.3` can also accept plug-in functions defined by end-users for custom summary statistics.

The motivation to write this package was born with the project [7] on characterizing Transcription Start Site (TSS) landscapes using high-throughput sequencing data, where a non-parametric distance measure was developed to assess the similarity among distributions of high-throughput sequencing reads from biological experiments. However, it is possible to use the method for any empirical distributions of categorical data.

The package is available on CRAN. The source code is available at http://cran.r-project.org/web/packages/GMD/ under GPL license.

## 2 Minimal Example: "Hello, GMD!"

hello-GMD.R

```
##' Check GMD's sanity and start up
##' @name hello-GMD

## GMD at CRAN, for source code download and installation
## http://cran.r-project.org/web/packages/GMD/index.html

## load GMD
library(GMD)

## version of GMD and description
packageVersion("GMD")
packageDescription("GMD")

## view GMD vignette
vignette("GMD-vignette",package="GMD")

## list the available data sets in GMD
data(package="GMD")

## list all the objects in the GMD
ls("package:GMD")

## help info on GMD
help(package="GMD")

## run a demo
demo("GMD-demo")

## cite GMD in publications
citation(package="GMD")

```

**Figure 1:** Source code of "hello-GMD.R".

`hello-GMD.R` (fig. 1) is a minimal example to load and check of that your `GMD` installation works. It also

includes code for viewing the package information and this "Vignette", checking data sets provided by GMD, starting a demo and listing the citation of GMD.

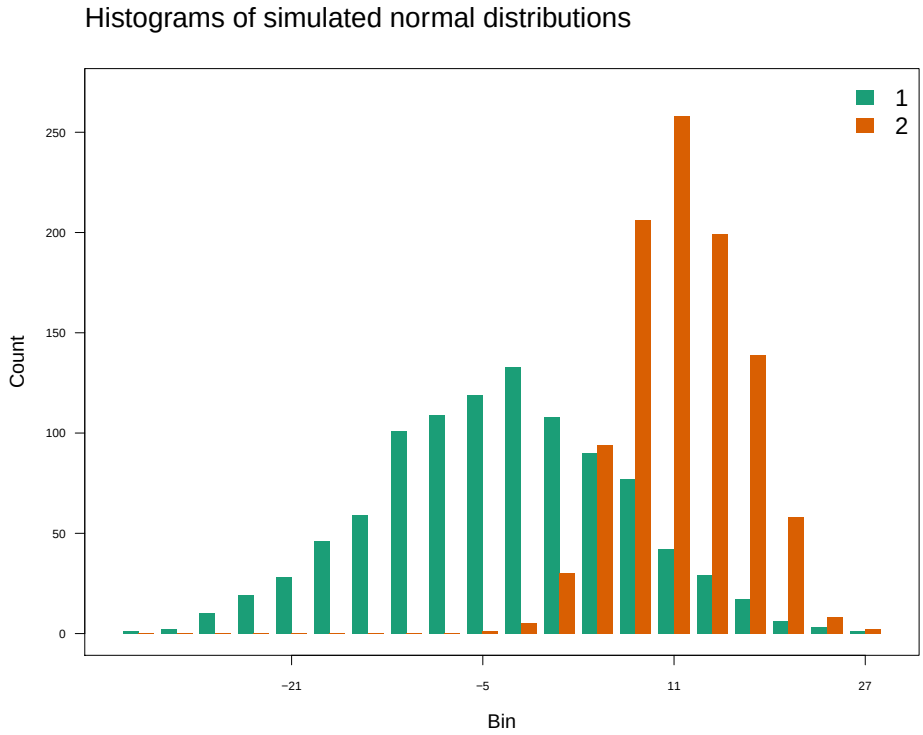
# 3 An example to understand GMD

This example, based on simulated data, is designed to illustrate the concepts and the implementation of GMD by stepping through the computations in detail.

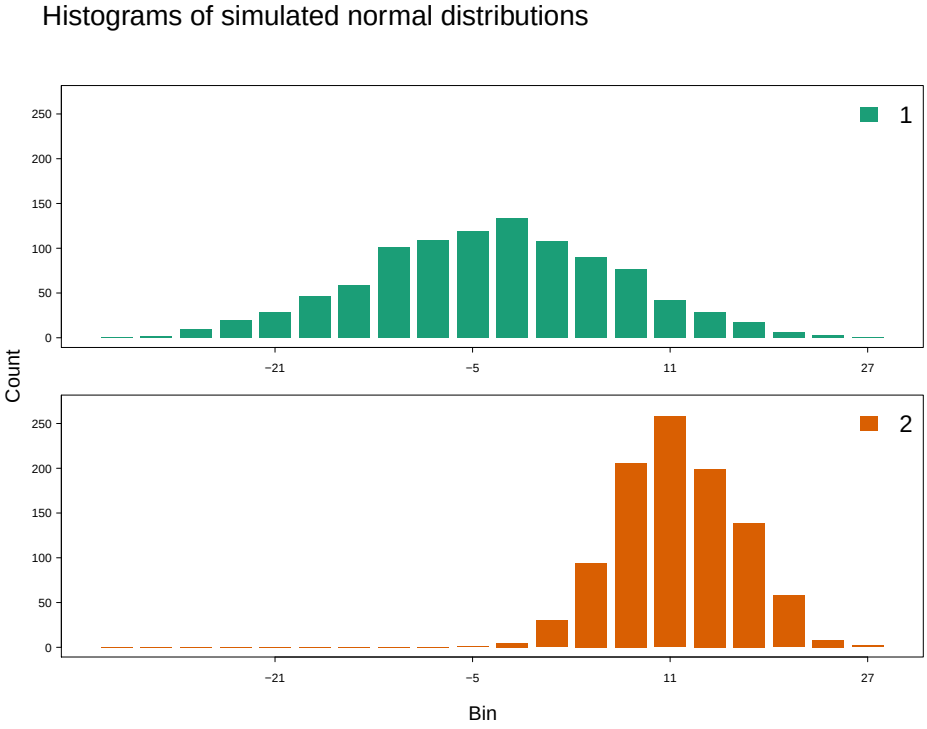
## 3.1 Histogram: construction and visualization

```
> ## ------------------------------------------------------------------------
> ## chunk1: Load library and simulate data
> ## ------------------------------------------------------------------------
> require("GMD") # load library
> ## create two normally-distributed samples
> ## with unequal means and unequal variances
> set.seed(2012)
> x1 <- rnorm(1000,mean=-5, sd=10)
> x2 <- rnorm(1000,mean=10, sd=5)


> ## ------------------------------------------------------------------------
> ## chunk2: Construct histograms
> ## ------------------------------------------------------------------------
> ## create common bins
> n <- 20 # desired number of bins
> breaks <- gbreaks(c(x1,x2),n) # bin boundaries
> ## make two histograms
> v1 <- ghist(x1,breaks=breaks,digits=0)
> v2 <- ghist(x2,breaks=breaks,digits=0)


> ## ------------------------------------------------------------------------
> ## chunk3: Save histograms as multiple-histogram (`mhist') object
> ## ------------------------------------------------------------------------
> x <- list(v1,v2)
> mhist.obj <- as.mhist(x)
```

```
> ## ------------------------------------------------------------------------
> ## chunk4: Visualize an `mhist' object
> ## ------------------------------------------------------------------------
> ## plot histograms side-by-side
> plot(mhist.obj,mar=c(1.5,1,1,0),main="Histograms of simulated normal distributions")
```



```
> ## plot histograms as subplots, with corresponding bins aligned
> plot(mhist.obj,beside=FALSE,mar=c(1.5,1,1,0),
+      main="Histograms of simulated normal distributions")
```



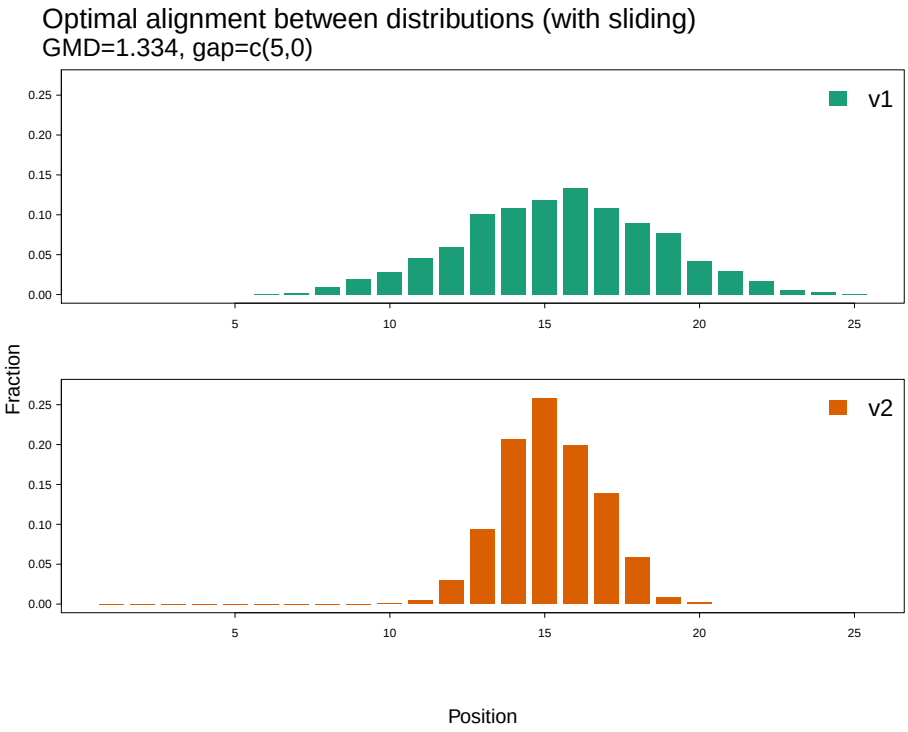
## 3.2 Histogram: distance measure and alignment

Here we measure the *GMD* distance between shapes of two histograms with option *sliding* on.

```
> ## ------------------------------------------------------------------------
> ## chunk5: Measure the pairwise distance between two histograms by GMD
> ## ------------------------------------------------------------------------
> gmdp.obj <- gmdp(v1,v2,sliding=TRUE)
> print(gmdp.obj)                       # print a brief version by default


[1] 1.334


> print(gmdp.obj,mode="detailed") # print a detailed version


GM-Distance: 1.334

Sliding: TRUE

Number of hits: 1

Gap:
       v1        v2
Hit1        5        0


Resolution: 1


> print(gmdp.obj,mode="full")     # print a full version


Distribution of v1:
1 2 10 19 28 46 59 101 109 119 133 108 90 77 42 29 17 6 3 1
(After normalization)
0.001 0.002 0.01 0.019 0.028 0.046 0.059 0.101 0.109 0.119 0.133 0.108 0.09 0.077 0.042 0.029 0.017 0.006 0.003 0.001

Distribution of v2:
0 0 0 0 0 0 0 0 0 1 5 30 94 206 258 199 139 58 8 2
(After normalization)
0 0 0 0 0 0 0 0 0 0.001 0.005 0.03 0.094 0.206 0.258 0.199 0.139 0.058 0.008 0.002

GM-Distance: 1.334

Sliding: TRUE

Number of hits: 1

Gap:
       v1        v2
Hit1        5        0


Resolution: 1
```

Now, let's have a look at the alignment by `GMD`, with a distance 1.334 and a "shift" of 5 in the $1^{st}$ distribution. It is important to note that the specific features (the values in this case) of the original bins in the histograms are ignored with *sliding* on. To keep original bin-to-bin correspondence, please set *sliding* to `FALSE` (see examples in section 4.2 on page 12).

```
> ## ------------------------------------------------------------------------
> ## chunk6: Show alignment
> ## ------------------------------------------------------------------------
> plot(gmdp.obj,beside=FALSE)
```


Optimal alignment between distributions (with sliding)
GMD=1.334, gap=c(5,0)
v1
v2
Fraction
Position
0.25
0.20
0.15
0.10
0.05
0.00
5
10
15
20
25

# 4 Case study

## 4.1 CAGE: measuring the dissimilarities among TSSDs

Studies have demonstrated that the spatial distributions of read-based sequencing data from different platforms often indicate functional properties and expression profiles (reviewed in [6] and [8]). Analyzing the distributions of DNA reads is therefore often meaningful. To do this systematically, a measure of similarity between distributions is necessary. Such measures should ideally be true metrics, have few parameters as possible, be computationally efficient and also make biological sense to end-users. Case studies were made in section 4.1 and 4.2 to demonstrate the applications of `GMD` using distributions of CAGE and ChIP-seq reads.

In this section we demonstrate how `GMD` is applied to measure the dissimilarities among TSSDs, histograms of transcription start site (TSS) that are made of CAGE tags, with option *sliding* on. The spatial properties of TSSDs vary widely between promoters and have biological implications in both regulation and function. The raw data were produced by CAGE and downloaded from FANTOM3 ([2]) and CAGE sequence reads were preprocessed as did in [7]. The following codes `case-cage.R` (fig. 2) are sufficient to perform both pairwise GMD calculation by function `gmdp` and to construct a GMD distance matrix by function `gmdm`. A handful of options are available for control and flexibility, particularly, the option `sliding` is enabled by default to allow partial alignment.

case-cage.R

```
require("GMD") # load library
data(cage)     # load data

## measure pairwise distance
x <- gmdp(cage[["Pfkfb3 (T02R00AEC2D8)"]],cage[["Csf1 (T03R0672174D)"]])
print(x)                      # print a brief version by default
print(x, mode="full")  # print a full version by default

## show alignment
plot(x,labels=c("Pfkfb3","Csf1"),beside=FALSE)

## show another alignment
plot(gmdp(cage[["Hig1 (T09R0743763C)"]],cage[["Cd72 (T04R028B8BC9)"]]),
     labels=c("Hig1 (T09R0743763C)","Cd72 (T04R028B8BC9)"),
     beside=FALSE)

## construct a distance matrix and visualize it
short.labels <- gsub("(.+) \\(.+","\\1",names(cage)) # get short labels
x <- gmdm(cage[1:6],labels=short.labels[1:6])
plot(x)
```

**Figure 2:** Source code of "case-cage.R".

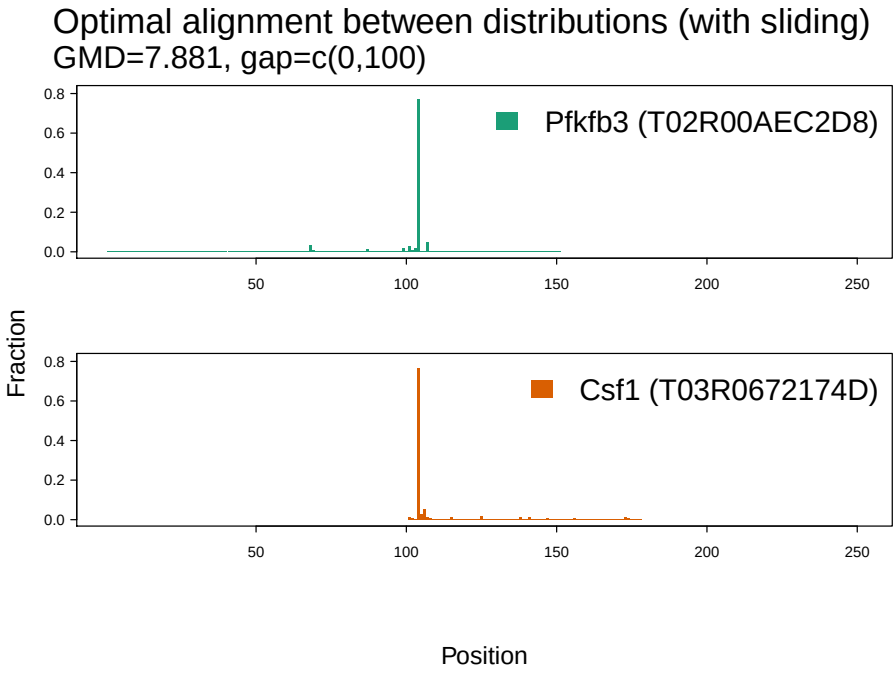



**Figure 3:** Graphical output 1 of source code "case-cage.R".

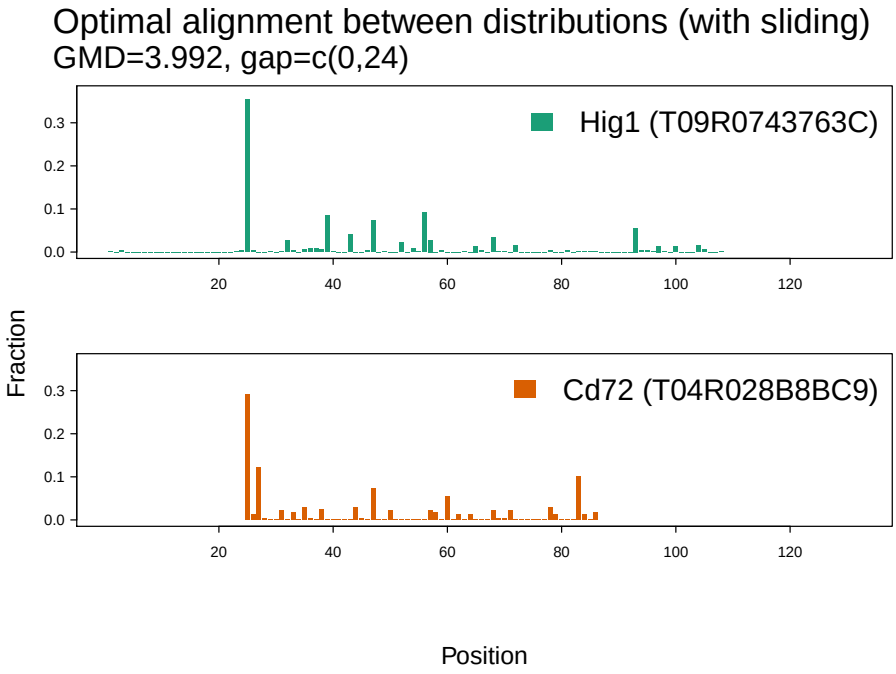



**Figure 4:** Graphical output 2 of source code "case-cage.R".

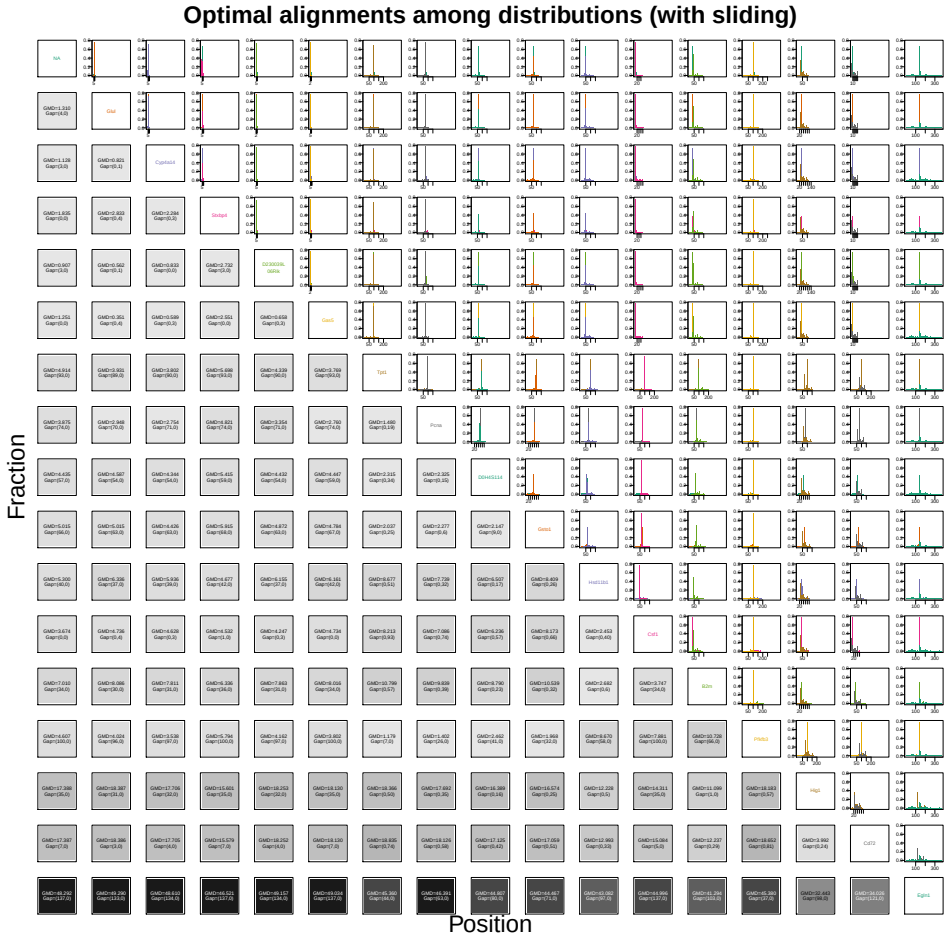



**Figure 5:** Graphical output 1 of source code "case-cage.R".

## 4.2 ChIP-seq: measuring the similarities among histone modification patterns

In this section we demonstrate how `GMD` is applied to measure the dissimilarities between histone modifications represented by ChIP-seq reads. Distinctive patterns of chromatin modifications around the TSS are associated with transcription regulation and expression variation of genes. Comparing the chromatin modification profiles (originally produced by [1] and [4], and preprocessed by [7]), the `sliding` option is disabled for fixed alignments at the TSSs and the flanking regions. The GMD measure indicates how well profiles are co-related to each other. In addition, the downstream cluster analysis is visualized with function `heatmap.3` that use GMD distance matrix to generate clustering dendrograms and make partitions given variant criteria, including the `‘‘elbow’’` rule (discussed in [7]) or desired number of clusters.

case-chipseq.R

```
require("GMD")      # load library
data(chipseq_mES)   # load data
data(chipseq_hCD4T) # load data

## pairwise distance and alignment based on GMD metric
plot(gmdm(chipseq_mES,sliding=FALSE))

## clustering on spatial distributions of histone modifications
x <- gmdm(chipseq_hCD4T,sliding=FALSE,resolution=10)
heatmap.3(x,revC=TRUE)

## Determine the number of clusters by "Elbow" criterion
main <- "Heatmap of ChIP-seq data (human CD4+ T cells)"
dist.obj <- gmdm2dist(x)
css.multi.obj <- css.hclust(dist.obj,hclust(dist.obj))
elbow.obj <- elbow.batch(css.multi.obj,ev.thres=0.90,inc.thres=0.05)
heatmap.3(dist.obj, main=main, revC=TRUE, kr=elbow.obj$k, kc=elbow.obj$k)

## more strict threshold
elbow.obj <- elbow.batch(css.multi.obj,ev.thres=0.75,inc.thres=0.1)
heatmap.3(dist.obj, main=main, revC=TRUE, kr=elbow.obj$k, kc=elbow.obj$k)

## side plots
normalizeVector <- function(v){v/sum(v)} # a function to normalize a vector
dev.new(width=12,height=8)

## summary of row clusters
expr1 <- list(quote(op <- par(mar = par("mar")*2)),
              quote(plot(mhist.summary(as.mhist(i.x)),if.plot.new=FALSE)),
              quote(par(op))
              )

## summary of row clustering
expr2 <- list(quote(tmp.clusters <- cutree(hclust(dist.row),k=kr)),
              quote(tmp.css <- css(dist.row,tmp.clusters)),
              quote(print(tmp.css)),
              quote(tmp.wev <- tmp.css$wss/tmp.css$tss),
              quote(names(tmp.wev) <- as.character(unique(tmp.clusters))),
              quote(tmp.wev <- tmp.wev[order(unique(tmp.clusters))]),
              quote(barplot(tmp.wev,main="Cluster Explained Variance", xlab="Cluster",
                            ylab="EV",col="white",border="black",
                            ylim=c(0,max(tmp.wev)*1.1),cex.main=1)))
expr3 <- list(quote(op <- par(mar = par("mar")*2)),
              quote(plot.elbow(css.multi.obj,elbow.obj,if.plot.new=FALSE)),
              quote(par(op))
              )

heatmap.3(dist.obj, main=main, cex.main=1.25, revC=TRUE, kr=elbow.obj$k, kc=elbow.obj$k,
          keysize=1,mapsize=4.5,row.data=lapply(chipseq_hCD4T,normalizeVector),
          plot.row.clusters=TRUE,plot.row.clusters.list=list(expr1),
          plot.row.clustering=TRUE,plot.row.clustering.list=list(expr2,expr3))

```

**Figure 6:** Source code of "case-chipseq.R".

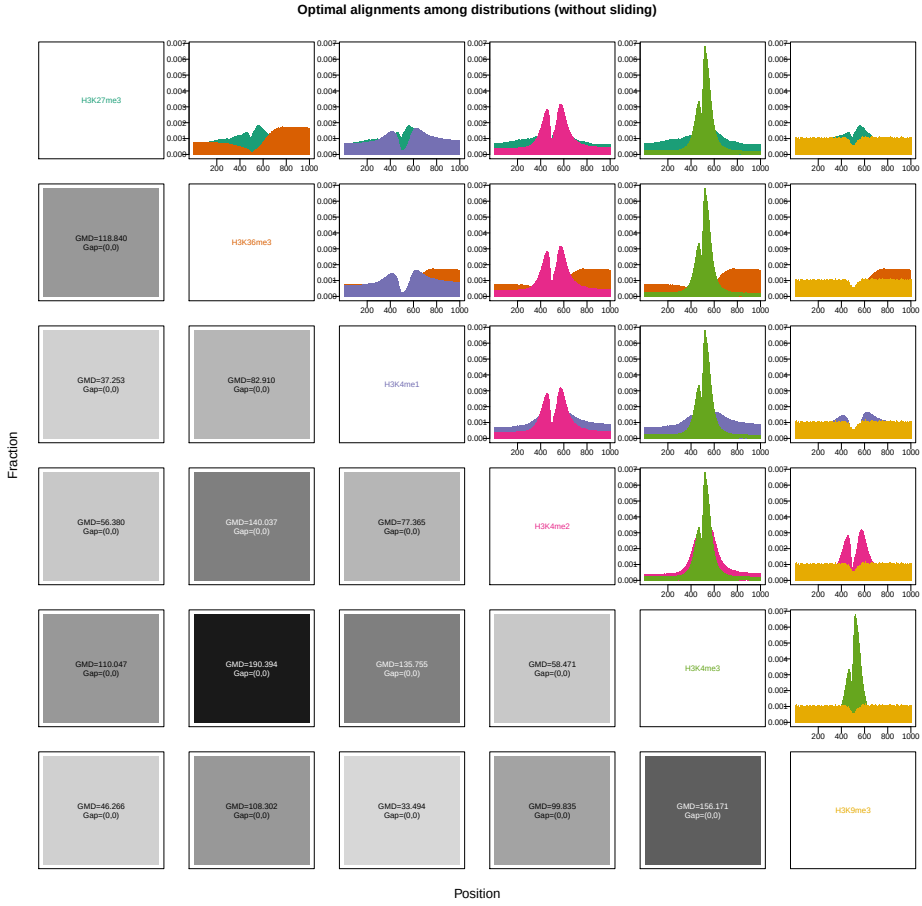



**Figure 7:** Graphical output 1 of source code "case-chipseq.R".

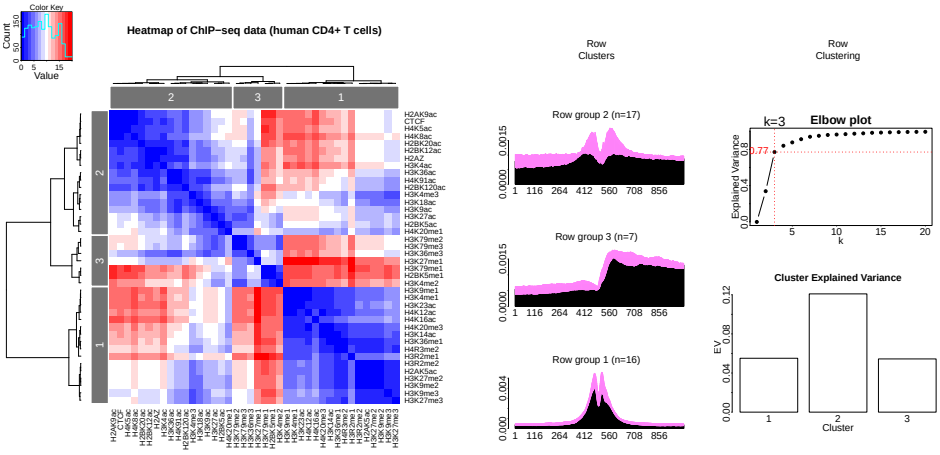



**Figure 8:** Graphical output 1 of source code "case-chipseq.R".

# A Functionality

## A.1 An overview

**Table 1.** Functions of the `GMD` R package

| Function | Description |
|---|---|
| `ghist` | Generalized Histogram Computation and Visualization |
| `gdist` | Generalized Distance Matrix Computation |
| `css` | Computing Clustering Sum-of-Squares and evaluating the clustering by the ‘‘`elbow`’’ method |
| `heatmap.3` | Enhanced Heatmap Representation with Dendrogram and Partition |
| `gmdp` | Computation of GMD on a pair of histograms |
| `gmdm` | Computation of GMD Matrix on a set of histograms |

## A.2 ghist: Generic construction and visualization of histograms

### A.2.1 Examples using simulated data

example-ghist.R (fig. 9) is an example on how to construct a histogram object from raw data and make a visualization based on this.

example-ghist.R

```
## load library
require("GMD")

## create two normally-distributed samples
## with unequal means and unequal variances
set.seed(2012)
v1 <- rnorm(1000,mean=-5, sd=10)
v2 <- rnorm(1000,mean=10, sd=5)

## create common bins
n <- 20 # desired number of bins
breaks <- gbreaks(c(v1,v2),n) # bin boundaries
x <-
  list(ghist(v1,breaks=breaks,digits=0),
       ghist(v2,breaks=breaks,digits=0))
mhist.obj <- as.mhist(x)

## plot histograms side-by-side
plot(mhist.obj,mar=c(1.5,1,1,0),main="Histograms of simulated normal distributions")

## plot histograms as subplots,
## with corresponding bins aligned
plot(mhist.obj,beside=FALSE,mar=c(1.5,1,1,0),
     main="Histograms of simulated normal distributions")
```

**Figure 9:** Source code of "example-ghist.R".

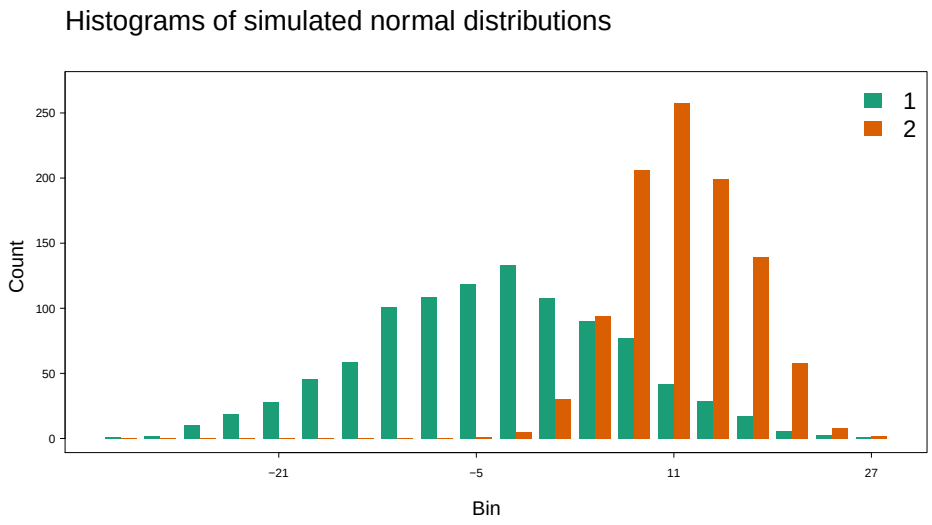



**Figure 10:** Graphical output 1 of source code "example-ghist.R".

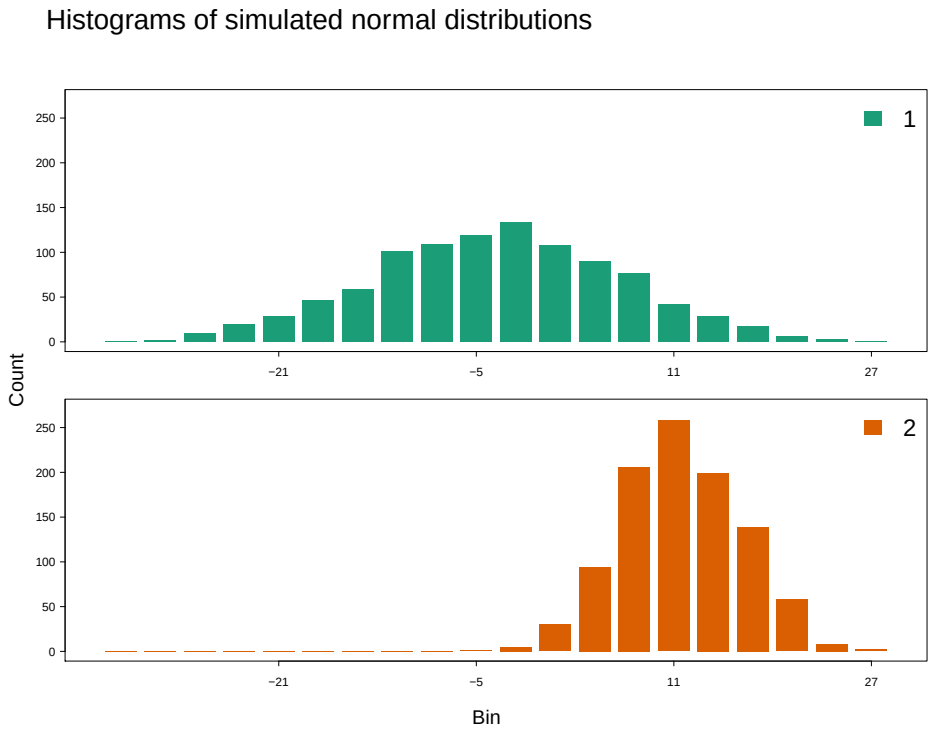



**Figure 11:** Graphical output 1 of source code "example-ghist.R".

### A.2.2 Examples using `iris` data

`case-iris.R` (fig. 12) is a study on how to obtain and visualize histograms, using Fisher's *iris* data set.

case-iris.R

```
## load library
require("GMD")

## load data
data(iris)

## create common bins
n <- 30                                      # the number of bins
breaks <- gbreaks(iris[,"Sepal.Length"],n) # the boundary of bins

## create a list of histograms
Sepal.Length <-
  list(setosa=ghist(iris[iris$Species=="setosa","Sepal.Length"],breaks=breaks),
       versicolor=ghist(iris[iris$Species=="versicolor","Sepal.Length"],breaks=breaks),
       virginica=ghist(iris[iris$Species=="virginica","Sepal.Length"],breaks=breaks)
       )

## convert to a `hist' object
x <- as.mhist(Sepal.Length)

## get bin-wise summary statistics
summary(x)

## visualize the histograms
plot(x,beside=FALSE,
     main="Histogram of Sepal.Length of iris",xlab="Sepal.Length (cm)")
```

**Figure 12:** Source code of "case-iris.R".

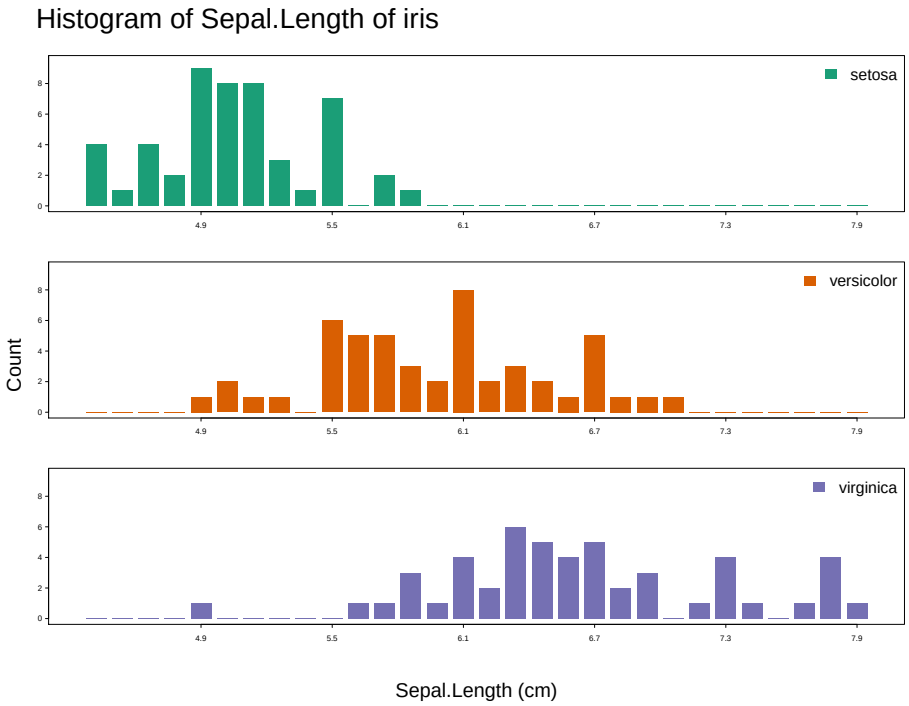



**Figure 13:** Graphical output 1 of source code "case-iris.R".

### A.2.3 Examples using `nottem` data

`case-nottem.R` (fig. 14) is a study on how to draw histograms side-by-side and to compute and visualize a bin-wise summary plot, using air temperature data at Nottingham Castle.

case-nottem.R

```
## load library
require("GMD")

## load data
data(nottem)

class(nottem)      # a time-series (ts) object
x <- ts2df(nottem) # convert ts to data.frame
mhist1 <- as.mhist(x[1:3,])

## plot multiple discrete distributions side-by-side
plot(mhist1,xlab="Month",ylab="Degree Fahrenheit",
     main="Air temperatures at Nottingham Castle")

## make summary statistics for each bin
mhist2 <- as.mhist(x)
ms <- mhist.summary(mhist2)
print(ms)

## plot bin-wise summary statistics with
## confidence intervals over the bars
plot(ms, main="Mean air temperatures at Nottingham Castle (1920-1939)",
     xlab="Month", ylab="Degree Fahrenheit")
```

**Figure 14:** Source code of "case-nottem.R".

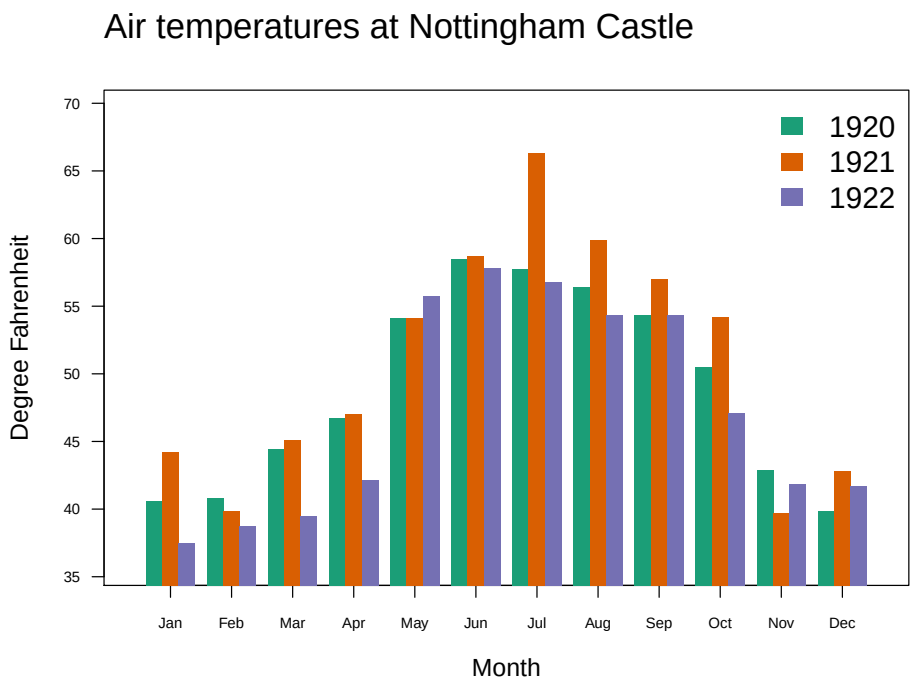



**Figure 15:** Graphical output 1 of source code "case-nottem.R".

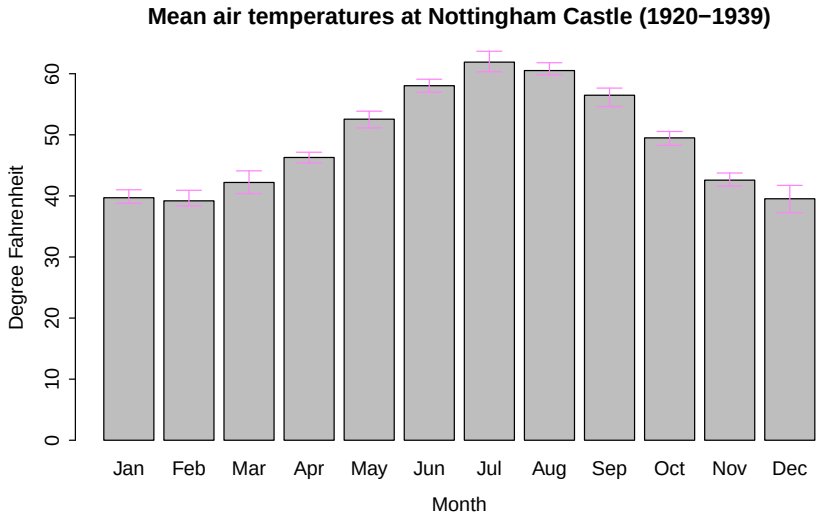



**Figure 16:** Graphical output 2 of source code "case-nottem.R".

## A.3 gdist: Generic construction and visualization of distances

example-gdist.R (fig. 17) is an example on how to measure distances using a user-defined metric, such as *correlation distance* and *GMD*.

example-gdist.R

```
## load library
require("GMD")
require(cluster)

## compute distance using Euclidean metric (default)
data(ruspini)
x <- gdist(ruspini)

## see a dendrogram result by hierarchical clustering
dev.new(width=12, height=6)
plot(hclust(x),
     main="Cluster Dendrogram of Ruspini data",
     xlab="Observations")

## convert to a distance matrix
m <- as.matrix(x)

## convert from a distance matrix
d <- as.dist(m)
stopifnot(d == x)

## Use correlations between variables "as distance"
data(USJudgeRatings)
dd <- gdist(x=USJudgeRatings,method="correlation.of.variables")
dev.new(width=12, height=6)
plot(hclust(dd),
     main="Cluster Dendrogram of USJudgeRatings data",
     xlab="Variables")
```

**Figure 17:** Source code of "example-gdist.R".

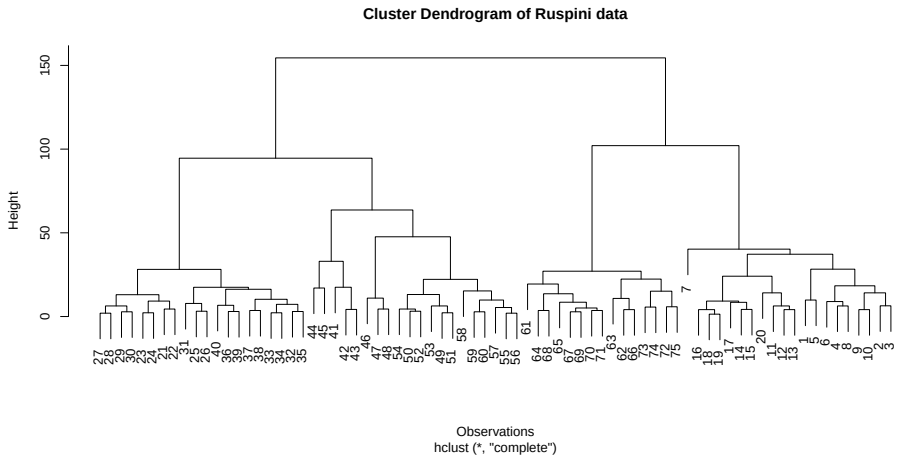



**Figure 18:** Graphical output 1 of source code "example-gdist.R".

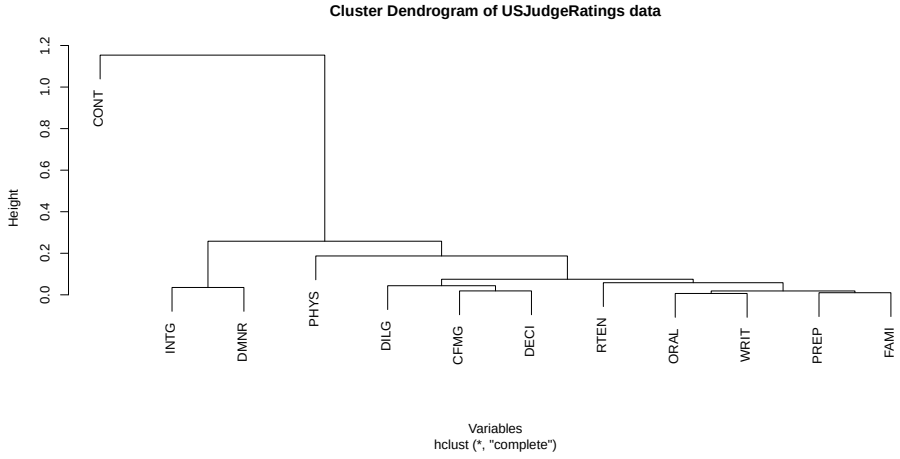



**Figure 19:** Graphical output 2 of source code "example-gdist.R".

## A.4 `css`: Clustering Sum-of-Squares and the ‘‘`elbow`’’ plot: determining the number of clusters in a data set

A good clustering yields clusters where the total within-cluster sum-of-squares (WSSs) is small (i.e. cluster cohesion, measuring how closely related are objects in a cluster) and the total between-cluster sum-of-squares (BSSs) is high (i.e. cluster separation, measuring how distinct or well-separated one cluster is from the other).

`example-css.R` (fig. 20) is an example on how to make correct choice of $k$ using **“elbow criterion”**. A good $k$ is selected according a) how much of the total variance in the whole data that the clusters can explain, and b) how large gain in explained variance we obtain by using these many clusters compared to one less or one more, the so-called “elbow” criterion.

The optimal choice of $k$ will strike a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster. More important, an ideal $k$ should also be relevant in terms of what it reveals about the data, which typically cannot be measured by a metric but by a human expert. Here we present a way to measure such performance of a clustering model, using squared Euclidean distances. The evaluation is based on pairwise distance matrix and therefore more generic in a way that doesn't involve computating the “centers” of the clusters in the raw data, which are often not available or hard to obtain.

example-css.R

```
## load library
require("GMD")

## simulate data around 12 points in Euclidean space
pointv <- data.frame(x=c(1,2,2,4,4,5,5,6,7,8,9,9),y=c(1,2,8,2,4,4,5,9,9,8,1,9))
set.seed(2012)
mydata <- c()
for (i in 1:nrow(pointv)){
  mydata <- rbind(mydata,cbind(rnorm(10,pointv[i,1],0.1),rnorm(10,pointv[i,2],0.1)))
}
mydata <- data.frame(mydata); colnames(mydata) <- c("x","y")
plot(mydata,type="p",pch=21, main="Simulated data")

## determine a "good" k using elbow
dist.obj <- dist(mydata[,1:2])
hclust.obj <- hclust(dist.obj)
css.obj <- css.hclust(dist.obj,hclust.obj)
elbow.obj <- elbow.batch(css.obj)
print(elbow.obj)

## make partition given the "good" k
k <- elbow.obj$k; cutree.obj <- cutree(hclust.obj,k=k)
mydata$cluster <- cutree.obj

## draw a elbow plot and label the data
dev.new(width=12, height=6)
par(mfcol=c(1,2),mar=c(4,5,3,3),omi=c(0.75,0,0,0))
plot(mydata$x,mydata$y,pch=as.character(mydata$cluster),col=mydata$cluster,cex=0.75,
     main="Clusters of simulated data")
plot.elbow(css.obj,elbow.obj,if.plot.new=FALSE)

## clustering with more relaxed thresholds (, resulting a smaller "good" k)
elbow.obj2 <- elbow.batch(css.obj,ev.thres=0.90,inc.thres=0.05)
mydata$cluster2 <- cutree(hclust.obj,k=elbow.obj2$k)

dev.new(width=12, height=6)
par(mfcol=c(1,2), mar=c(4,5,3,3),omi=c(0.75,0,0,0))
plot(mydata$x,mydata$y,pch=as.character(mydata$cluster2),col=mydata$cluster2,cex=0.75,
     main="Clusters of simulated data")
plot.elbow(css.obj,elbow.obj2,if.plot.new=FALSE)
```

**Figure 20:** Source code of "example-css.R".

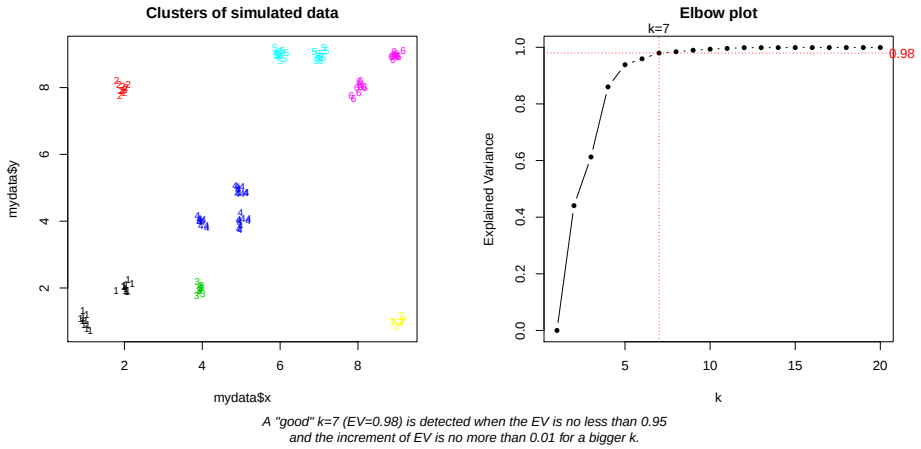



**Figure 21:** Graphical output 1 of source code "example-css.R".

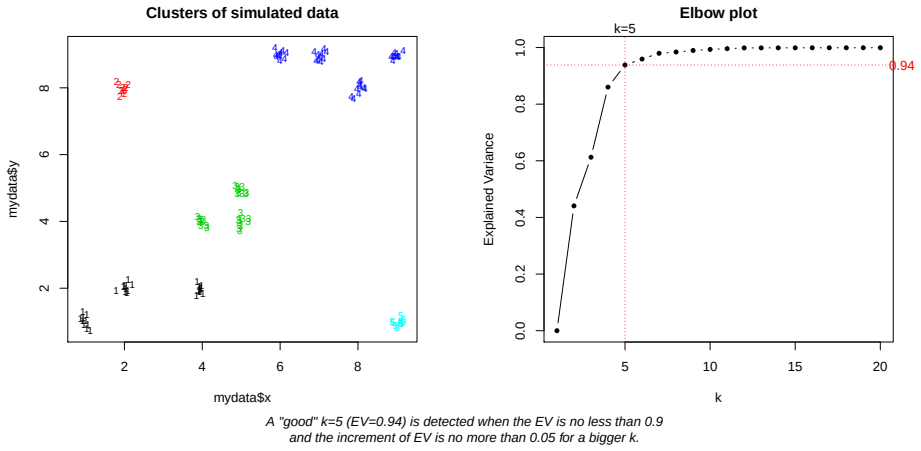



**Figure 22:** Graphical output 2 of source code "example-css.R".

## A.5 `heatmap.3`: Visualization in cluster analysis, with evaluation

### A.5.1 Examples using `mtcars` data

`example-heatmap3a.R` (fig. 23) is an example on how to make a heatmap with summary visualization of observations.

example-heatmap3a.R

```
require("GMD")          # load library
data(mtcars)            # load data
x  <- as.matrix(mtcars) # data as a matrix

dev.new(width=10,height=8)
heatmap.3(x)                                # default, with reordering and dendrogram
heatmap.3(x, Rowv=FALSE, Colv=FALSE)        # no reordering and no dendrogram
heatmap.3(x, dendrogram="none")             # reordering without dendrogram
heatmap.3(x, dendrogram="row")              # row dendrogram with row (and col) reordering
heatmap.3(x, dendrogram="row", Colv=FALSE) # row dendrogram with only row reordering
heatmap.3(x, dendrogram="col")              # col dendrogram
heatmap.3(x, dendrogram="col", Rowv=FALSE) # col dendrogram with only col reordering
heatmapOut <-
  heatmap.3(x, scale="column")              # sacled by column
names(heatmapOut)                           # view the list that is returned
heatmap.3(x, scale="column", x.center=0)   # colors centered around 0
heatmap.3(x, scale="column",trace="column")# trun "trace" on

## coloring cars (row observations) by brand
brands <- sapply(rownames(x), function(e) strsplit(e," ")[[1]][1])
names(brands) <- c()
brands.index <- as.numeric(as.factor(brands))
RowIndividualColors <- rainbow(max(brands.index))[brands.index]
heatmap.3(x, scale="column", RowIndividualColors=RowIndividualColors)

## coloring attributes (column features) randomly (just for a test :)
heatmap.3(x, scale="column", ColIndividualColors=rainbow(ncol(x)))

## add a single plot for all row individuals
dev.new(width=12,height=8)
expr1 <- list(quote(plot(row.data[rowInd,"hp"],1:nrow(row.data),xlab="hp",ylab="",
                         main="Gross horsepower",yaxt="n")),
              quote(axis(2,1:nrow(row.data),rownames(row.data)[rowInd],las=2)))
heatmap.3(x, scale="column", plot.row.individuals=TRUE, row.data=x,
          plot.row.individuals.list=list(expr1))

## add a single plot for all col individuals
dev.new(width=12,height=8)
expr2 <- list(quote(plot(colMeans(col.data)[colInd], xlab="", ylab="Mean",xaxt="n",
                         main="Mean features",cex=1,pch=19)),
              quote(axis(1,1:ncol(col.data),colnames(col.data)[colInd],las=2)))
heatmap.3(x, scale="column", plot.col.individuals=TRUE, col.data=x,
          plot.col.individuals.list=list(expr2))

## add another single plot for all col individuals
dev.new(width=12,height=8)
expr3 <- list(quote(op <- par(mar = par("mar")*1.5)),
              quote(mytmp.data <- apply(col.data,2,function(e) e/sum(e))),
              quote(boxplot(mytmp.data[,colInd], xlab="", ylab="Value",
                         main="Boxplot of normalized column features")),
              quote(par(op)))
heatmap.3(x, scale="column", plot.col.individuals=TRUE, col.data=x,
          plot.col.individuals.list=list(expr3))
```



**Figure 23:** Source code of "example-heatmap3a.R".

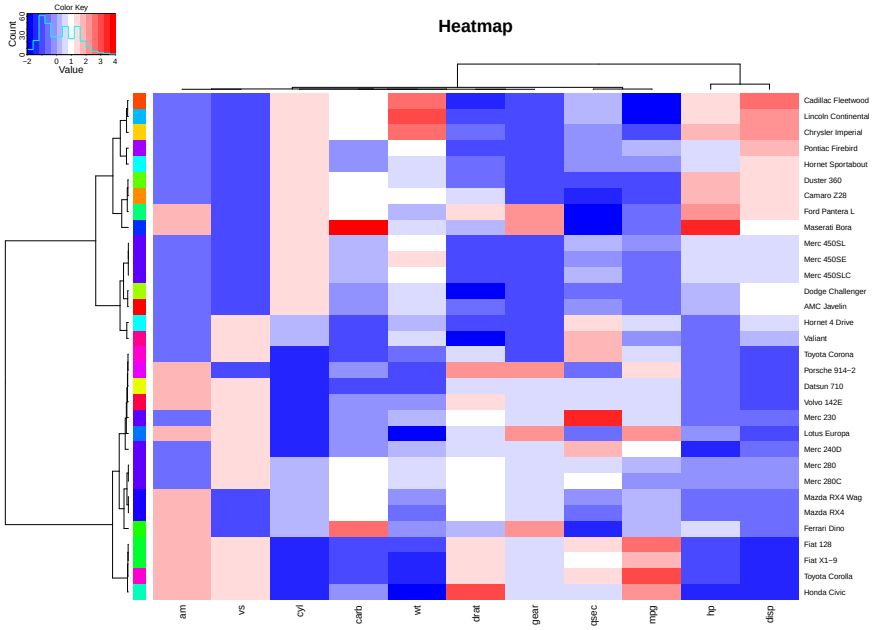



**Figure 24:** Graphical output 1 of source code "example-heatmap3a.R".

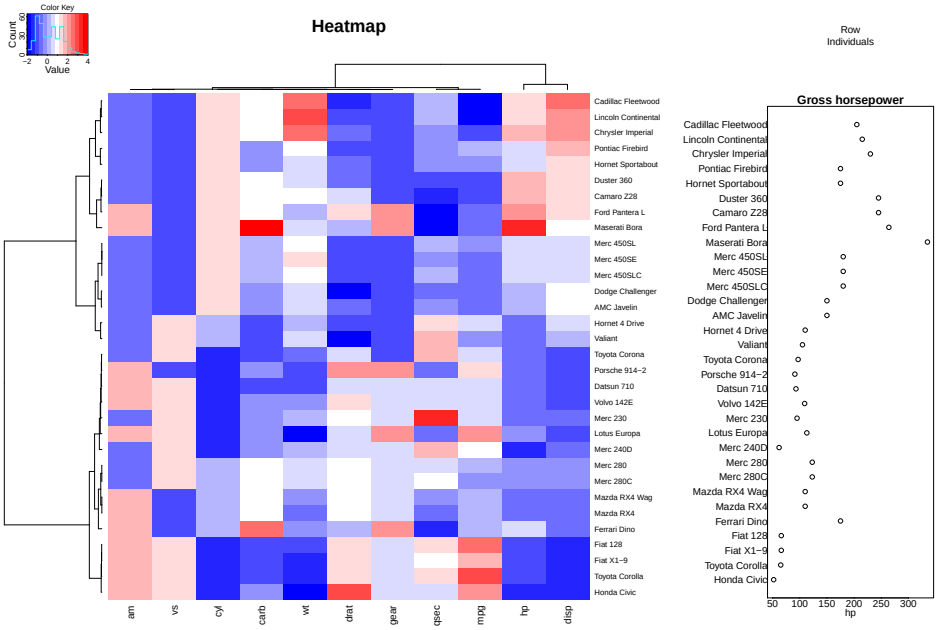



**Figure 25:** Graphical output 2 of source code "example-heatmap3a.R".

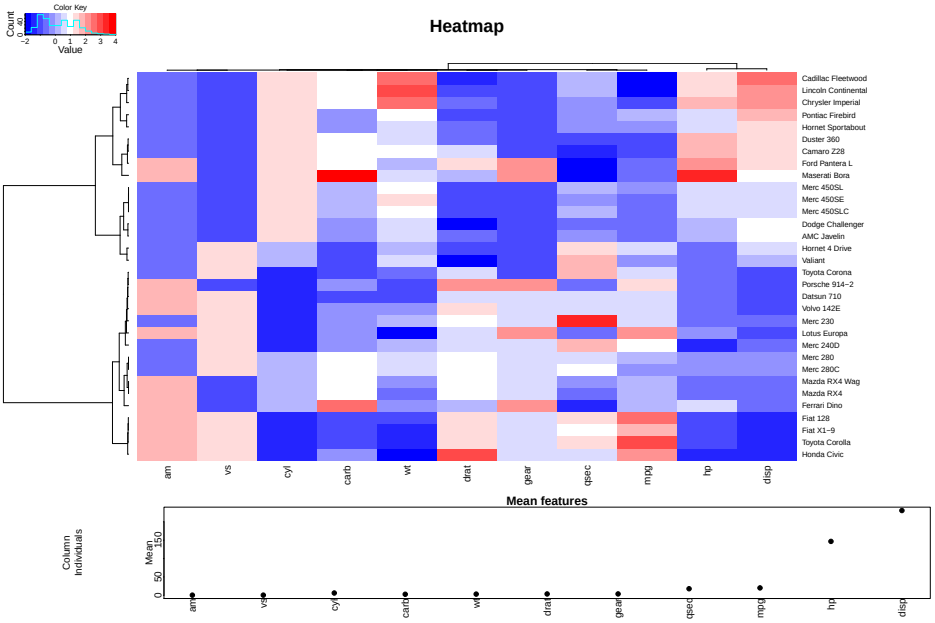



**Figure 26:** Graphical output 3 of source code "example-heatmap3a.R".

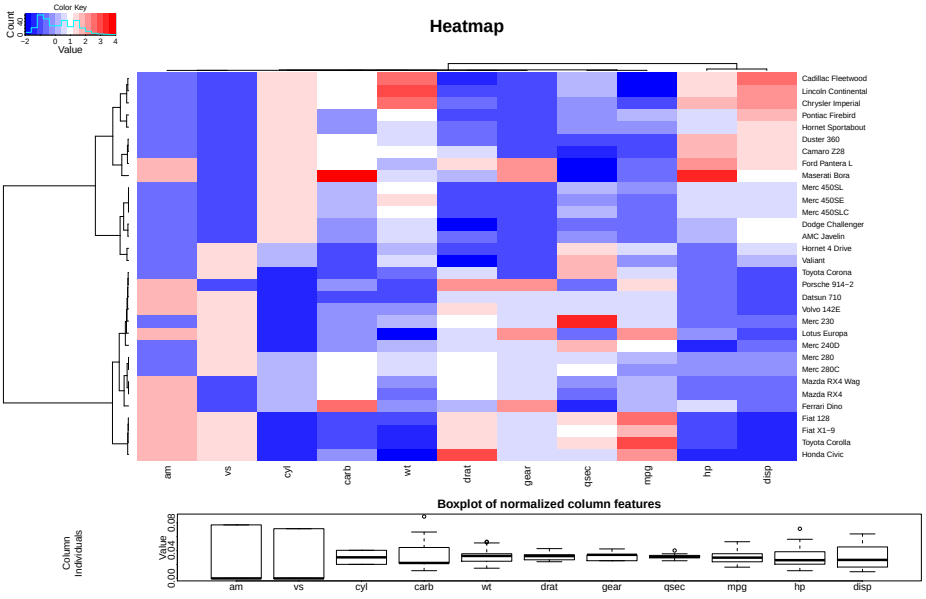



**Figure 27:** Graphical output 4 of source code "example-heatmap3a.R".

### A.5.2 Examples using `ruspini` data

`example-heatmap3b.R` (fig. 28) is an example on how to make a heatmap with summary visualization of clusters.

example-heatmap3b.R

```
## load library
require("GMD")
require(cluster)

## load data
data(ruspini)

## heatmap on a `dist' object
x <- gdist(ruspini)
main <- "Heatmap of Ruspini data"
dev.new(width=10,height=10)
heatmap.3(x, main=main, mapratio=1) # default with a title and a map in square!
heatmap.3(x, main=main, revC=TRUE)  # reverse column for a symmetric look
heatmap.3(x, main=main, kr=2, kc=2) # show partition by predefined number of clusters

## show partition by elbow
css.multi.obj <- css.hclust(x,hclust(x))
elbow.obj <- elbow.batch(css.multi.obj,ev.thres=0.90,inc.thres=0.05)
heatmap.3(x, main=main, revC=TRUE, kr=elbow.obj$k, kc=elbow.obj$k)
heatmap.3(x, main=main, sub=sub("\n"," ",attr(elbow.obj,"description")), cex.sub=1.25,
          revC=TRUE,kr=elbow.obj$k, kc=elbow.obj$k) # show elbow info as subtitle

## side plot for every row clusters
dev.new(width=10,height=10)
expr1 <- list(quote(plot(do.call(rbind,i.x),xlab="x",ylab="y",
                         xlim=range(ruspini$x),ylim=range(ruspini$y),)))
heatmap.3(x, main=main, revC=TRUE, kr=elbow.obj$k, kc=elbow.obj$k, trace="none",
          row.data=as.list(data.frame(t(ruspini))),
          plot.row.clusters=TRUE,plot.row.clusters.list=list(expr1))

## side plot for every col clusters
dev.new(width=10,height=10)
expr2 <- list(quote(plot(do.call(rbind,i.x),xlab="x",ylab="y",
                         xlim=range(ruspini$x),ylim=range(ruspini$y),)))
heatmap.3(x, main=main, revC=TRUE, kr=elbow.obj$k, kc=elbow.obj$k, trace="none",
          col.data=as.list(data.frame(t(ruspini))),
          plot.col.clusters=TRUE,plot.col.clusters.list=list(expr2))

```

**Figure 28:** Source code of "example-heatmap3b.R".

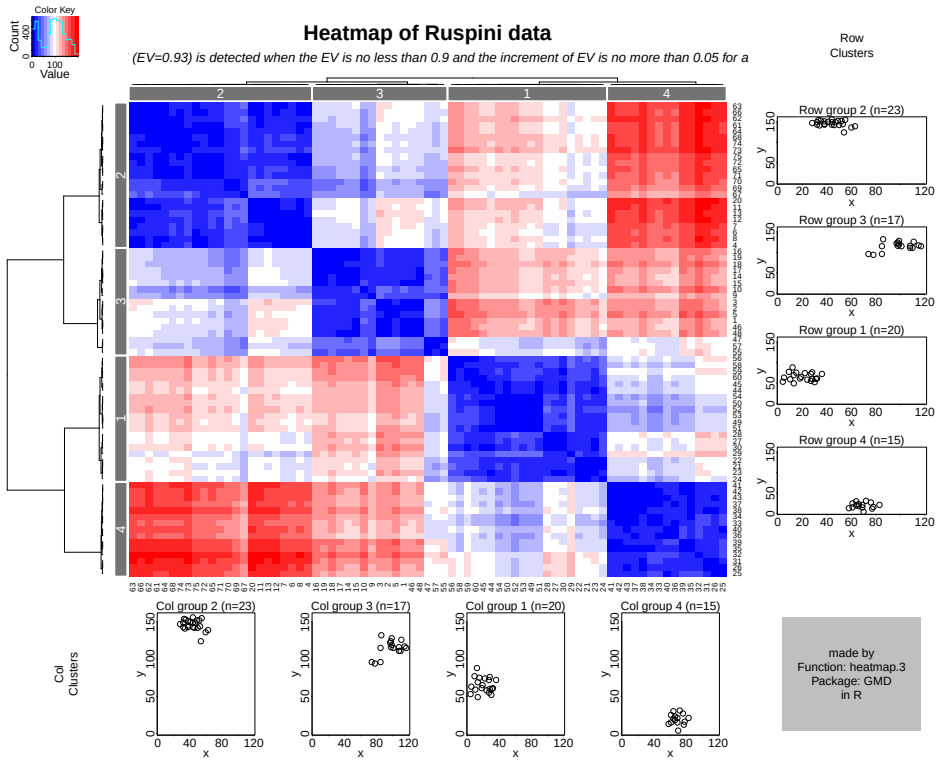



**Figure 29:** Graphical output 1 of source code "example-heatmap3b.R".

# B Data

## B.1 GMD dataset overview

```
> data(package="GMD")


Data sets in package 'GMD':

cage                    CAGE Data
cagel                   CAGE Data
chipseq_hCD4T           ChIP-seq Data
chipseq_mES             ChIP-seq Data
```

## B.2 CAGE data: cage and cagel

```
> help(cage)


> data(cage)
> class(cage)


[1] "list"


> length(cage)


[1] 20


> names(cage)


 [1] "NA (T01F029805F8)"             "Glul (T01F092C2995)"
 [3] "Cyp4a14 (T04R06C91673)"        "Stxbp4 (T11R05607FD4)"
 [5] "D230039L06Rik (T01F0AA465EB)" "Gas5 (T01F09995479)"
 [7] "Rab5c (T11R05FBC6C4)"          "BC003940 (T11R072A6CB0)"
 [9] "Tpt1 (T14F04079189)"           "Pcna (T02R07DE319B)"
[11] "D0H4S114 (T18R020553F0)"       "Gsto1 (T19F02D03566)"
[13] "Hsd11b1 (T01R0B8305BD)"        "Csf1 (T03R0672174D)"
[15] "B2m (T02F0743FF05)"            "Alox5ap (T05F08BCF2C4)"
[17] "Pfkfb3 (T02R00AEC2D8)"         "Hig1 (T09R0743763C)"
[19] "Cd72 (T04R028B8BC9)"           "Egln1 (T08R0769239F)"


> data(cagel)
> names(cagel)


 [1] "NA (T17F05912B83)"             "Tpt1 (T14F04079189)"
 [3] "B2m (T02F0743FF05)"            "Grn (T11F0615F289)"
 [5] "2600001A11Rik (T12R043A2595)" "Rbbp7 (T0XF091A7ACA)"
 [7] "Rpl41 (T10R07AB7138)"          "H2afy (T13R034ACF47)"
 [9] "Dscr1l1 (T17F02802885)"        "Ckap4 (T10R0504CE97)"
[11] "Rab5c (T11R05FBC6C4)"          "Pfkfb3 (T02R00AEC2D8)"
[13] "Csf1 (T03R0672174D)"           "Ctsb (T14F0348EDBA)"
[15] "Crim1 (T17F04928998)"          "3930401E15Rik (T18R02CDD141)"
[17] "Rai17 (T14F014BF473)"          "Hig1 (T09R0743763C)"
[19] "Apbb2 (T05R03E329C8)"          "Ptn (T06R0230806E)"
[21] "Tmeff2 (T01F030EC757)"         "Mrps6 (T16F0583C906)"
[23] "Hsd11b1 (T01R0B8305BD)"        "D0H4S114 (T18R020553F0)"
[25] "4930430J02Rik (T09F036E80C6)" "Phtf2 (T05R0125E896)"
[27] "Trpv2 (T11F03B4EBD8)"          "Slco3a1 (T07R03AC06B9)"
[29] "Scd1 (T19R029B5186)"           "Ctxn (T08R0040864A)"
[31] "5730596K20Rik (T19F006DFC1A)" "Arbp (T05F06BBE13B)"
[33] "Klhl5 (T05F03CCA673)"          "Gsto1 (T19F02D03566)"
[35] "NA (T02R07EF5EDA)"             "Srpk1 (T17R019F4A41)"
[37] "Nudt7 (T08F06C3B651)"          "Tnfrsf10b (T14F03AB1306)"
[39] "Egln1 (T08R0769239F)"          "9630050M13Rik (T02F002EC972)"
[41] "BC003940 (T11R072A6CB0)"       "Ppia (T11F00604AFF)"
[43] "Alox5ap (T05F08BCF2C4)"        "Pcna (T02R07DE319B)"
[45] "Gch1 (T14R02602138)"           "Yap1 (T09R0079F3FF)"
[47] "Vrk1 (T12F06010C9B)"           "Cd72 (T04R028B8BC9)"
[49] "Wdtc1 (T04R07DAFEDC)"          "Centg2 (T01F055392D1)"
```

## B.3 ChIP-seq data: chipseq_mES and chipseq_hCD4T

```
> help(chipseq)
```

```
> data(chipseq_mES)
> class(chipseq_mES)
```

```
[1] "list"
```

```
> length(chipseq_mES)
```

```
[1] 6
```

```
> names(chipseq_mES)
```

```
[1] "H3K27me3" "H3K36me3" "H3K4me1"  "H3K4me2"  "H3K4me3"  "H3K9me3"
```

```
> data(chipseq_hCD4T)
> names(chipseq_hCD4T)
```

```
 [1] "CTCF"      "H2AK5ac"   "H2AK9ac"   "H2AZ"      "H2BK120ac" "H2BK12ac"
 [7] "H2BK20ac"  "H2BK5ac"   "H2BK5me1"  "H3K14ac"   "H3K18ac"   "H3K23ac"
[13] "H3K27ac"   "H3K27me1"  "H3K27me2"  "H3K27me3"  "H3K36ac"   "H3K36me1"
[19] "H3K36me3"  "H3K4ac"    "H3K4me1"   "H3K4me2"   "H3K4me3"   "H3K79me1"
[25] "H3K79me2"  "H3K79me3"  "H3K9ac"    "H3K9me1"   "H3K9me2"   "H3K9me3"
[31] "H3R2me1"   "H3R2me2"   "H4K12ac"   "H4K16ac"   "H4K20me1"  "H4K20me3"
[37] "H4K5ac"    "H4K8ac"    "H4K91ac"   "H4R3me2"
```

# References


[1] Artem Barski, Suresh Cuddapah, Kairong Cui, Tae-Young Roh, Dustin E Schones, Zhibin Wang, Gang Wei, Iouri Chepelev, and Keji Zhao. High-resolution profiling of histone methylations in the human genome. *Cell*, 129(4):823–837, May 2007.

[2] Piero Carninci, Albin Sandelin, Boris Lenhard, Shintaro Katayama, Kazuro Shimokawa, Jasmina Ponjavic, Colin A M Semple, Martin S Taylor, Pr G Engstrm, Martin C Frith, Alistair R R Forrest, Wynand B Alkema, Sin Lam Tan, Charles Plessy, Rimantas Kodzius, Timothy Ravasi, Takeya Kasukawa, Shiro Fukuda, Mutsumi Kanamori-Katayama, Yayoi Kitazume, Hideya Kawaji, Chikatoshi Kai, Mari Nakamura, Hideaki Konno, Kenji Nakano, Salim Mottagui-Tabar, Peter Arner, Alessandra Chesi, Stefano Gustincich, Francesca Persichetti, Harukazu Suzuki, Sean M Grimmond, Christine A Wells, Valerio Orlando, Claes Wahlestedt, Edison T Liu, Matthias Harbers, Jun Kawai, Vladimir B Bajic, David A Hume, and Yoshihide Hayashizaki. Genome-wide analysis of mammalian promoter architecture and evolution. *Nat Genet*, 38(6):626–635, Jun 2006.

[3] Sung-Hyuk Cha and Sargur N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355–1370, 2002.

[4] Tarjei S Mikkelsen, Manching Ku, David B Jaffe, Biju Issac, Erez Lieberman, Georgia Giannoukos, Pablo Alvarez, William Brockman, Tae-Kyung Kim, Richard P Koche, William Lee, Eric Mendenhall, Aisling O'Donovan, Aviva Presser, Carsten Russ, Xiaohui Xie, Alexander Meissner, Marius Wernig, Rudolf Jaenisch, Chad Nusbaum, Eric S Lander, and Bradley E Bernstein. Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, 448(7153):553–560, Aug 2007.

[5] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[6] Albin Sandelin, Piero Carninci, Boris Lenhard, Jasmina Ponjavic, Yoshihide Hayashizaki, and David A Hume. Mammalian rna polymerase ii core promoters: insights from genome-wide studies. *Nat Rev Genet*, 8(6):424–436, Jun 2007.

[7] Xiaobei Zhao, Eivind Valen, Brian J Parker, and Albin Sandelin. Systematic clustering of transcription start site landscapes. *PLoS ONE*, 6(8):e23409, August 2011.

[8] Vicky W Zhou, Alon Goren, and Bradley E Bernstein. Charting histone modifications and the functional organization of mammalian genomes. *Nat Rev Genet*, 12(1):7–18, Jan 2011.