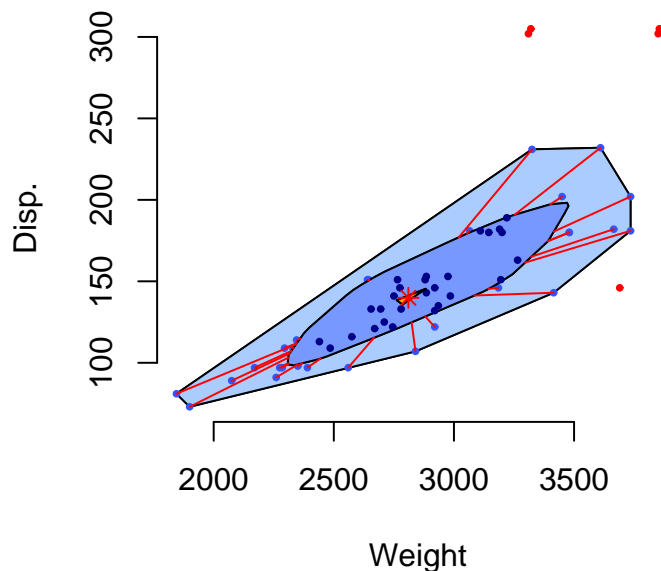


A rough R Impementation of the Bagplot

File: bagplot.rev
in: /home/wiwi/pwolf/R/work/bagplot
Version: Dec 10 19:15:01 2012



Contents

1	Examples	3
1.1	Example: car data (Chambers / Hastie 1992)	3
1.2	The normal case	4
1.3	Large data sets	5
1.4	Size of data set	6
1.5	"Depth-One" data sets	7
1.6	Degenerated data sets	8
1.7	Data set from the mail of M. Maechler	9
1.8	Data sets of Wouter Meuleman, running in an error with version 09/2005	10
1.9	Data sets of Amanda Joy Shaker	12
1.10	Bagplot with additional graphical supplements	13
1.11	Debugging plots with additional elements	14
2	Bagplots by an alternative approach, proposed by Rousseeuw, Ruts and Tukey	15

3	Arguments and output of bagplot, the help page and some links	20
4	The definition of bagplot	24
4.1	The body of <code>compute.bagplot</code>	25
4.2	Output of <code>bagplot</code>	26
4.3	Initilization of <code>bagplot</code>	26
4.4	Some local functions to find intersection points	28
4.5	A function to compute the h-depths of data points	31
4.5.1	A user function to compute h-depths	31
4.6	A function to expand the hull	33
4.7	A function to find the position of points respectively to a polygon	34
4.8	Check if data set is one dimensional	35
4.9	Standardize data and compute h-depths of points	36
4.10	Find the center of the data set	39
4.11	Finding of the bag	45
4.12	Computation of the loop	46
4.13	The definition of <code>plot.bagplot</code>	47
4.14	Some technical leftovers	50
4.14.1	Definition of <code>bagplot</code> on start	50
4.14.2	Extracting the R code file <code>bagplot.R</code>	50
5	Inspection tools	51
5.1	Pairs plot with bagplots	51
5.2	Function for summarizing variables of a matrix	54
6	Skyline-Plots	60
7	Version 1	60
8	Appendix	67
8.1	Some further examples – usefull for testing	67
8.2	Data Sets of AJS	67
8.2.1	NA/NaN/Inf errors	67
8.2.2	Vanishing center polygons	70
8.2.3	<code>chull</code> problem	71
8.2.4	Missing Fence on 64 Bit Machines	75
8.2.5	Further Na/NaN-errors	78
8.2.6	Identical x-values	80
9	frabo12 Daten	81
9.1	Indices	83

In this paper we describe a rough implementation of the bagplot. The first section shows some examples. Section 2 compares our bagplot function to the solution of Rousseeuw, Ruts, and Tukey (1999). Then the arguments, the help page of the function `bagplot` and some links are listed. In section 4 you find the definition of the function. In the appendix further examples for testing are given and some old code chunks are listed.

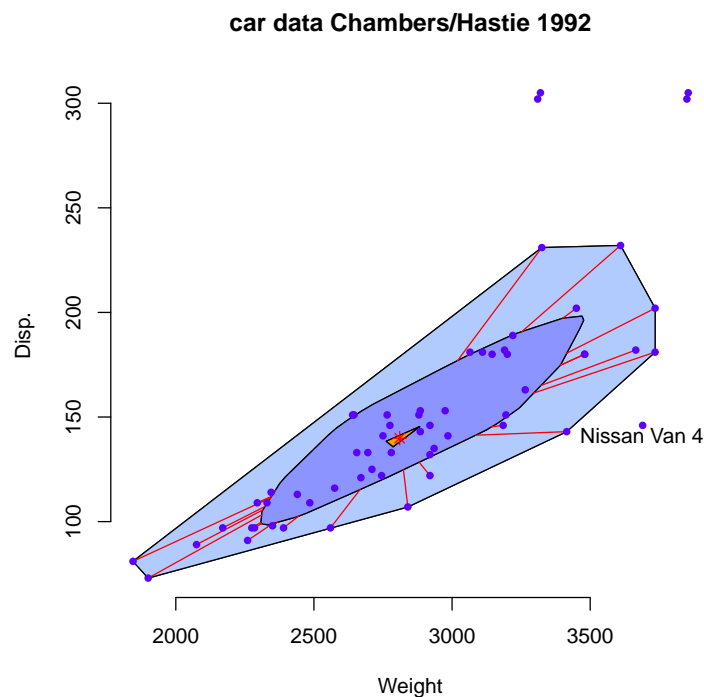
1 $\langle \text{version of bagplot} \rangle \equiv \subset 32, 33, 81$
"bagplot, version 2012/12/05, peter wolf"

1 Examples

1.1 Example: car data (Chambers / Hastie 1992)

The first example is a bagplot of the famous car data of Chambers and Hastie. In the code chunk the data set is assigned to `cardata` and `bagplot()` is called with some parameters that are described later in this paper.

```
2 <cardata 2> ≡
  library(rpart); cardata<-car.test.frame[,6:7]; par(mfrow=c(1,1))
  <define bagplot 32>
  bagplot(cardata,verbose=FALSE,factor=2.5,show.baghull=TRUE,dkmethod=2,
    show.loophull=TRUE,precision=1)$center
  points(cardata,pch=16,cex=.8,col="blue")
  title("car data Chambers/Hastie 1992")
  text(cardata[60,] + c(0,-5), rownames(cardata)[60]) # Nissan Van
  # h <- which(250 < cardata[,2])
  # text(cardata[h,]+cbind(250*c(-1.2,-.91,-.3,-.3),8*c(-1,.7,-1,1)),
  #       rownames(cardata)[h],xpd=NA)
```



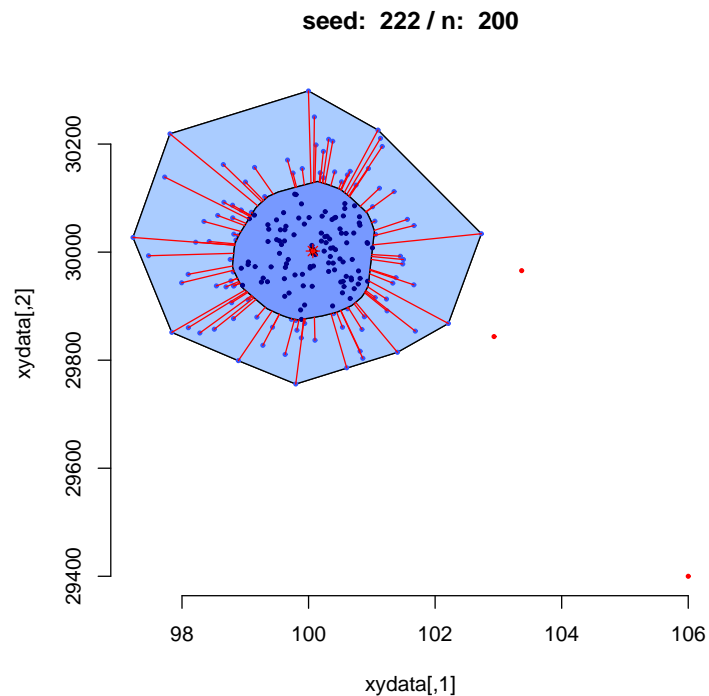
To integrate the data point of Nissan Van into the light blue region the argument `factor` has to be set to 3. The Tukey median of our bagplot function is (2810.431, 139.879). Splus computes a slightly different point: (2806.63, 139.513). In difference to Rousseeuw et al. our bagplot as well as the bagplot of Splus classified the data point of Nissan Van 4 as outlier. To get the Splus results you have to download `bagplot*`, the car data and ...

```
Splus CHAPTER bagplot.f
Splus make
Splus ...
> dyn.open("S.so"); source("bagplot.s") #; postscript("hello.ps")
> bagplot(cardata[,1],cardata[,2]) #; dev.off()
```

1.2 The normal case

A bagplot of an `rnorm` sample with one heavy outlier is shown by the following code chunk.

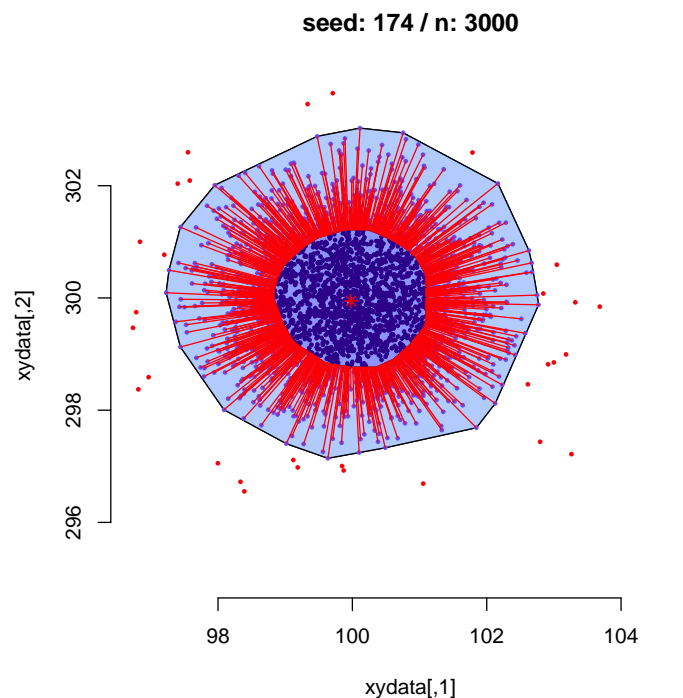
```
3 <norm 3> ≡  
<define bagplot 32>  
seed<-222; n<-200  
<define rnorm data data, seed: seed, size: n 114>  
datan<-rbind(data,c(106,294)); datan[,2]<-datan[,2]*100  
h <- bagplot(datan,factor=3,create.plot=TRUE,approx.limit=300,  
  show.outlier=TRUE,show.looppoints=TRUE,show.bagpoints=TRUE,  
  show.whiskers=TRUE,show.loophull=TRUE,show.baghull=TRUE,  
  verbose=FALSE, cex=0.6)  
title(paste("seed: ",seed,"/ n: ",n))
```



1.3 Large data sets

What about very large data sets? The algorithm computes some of the quantities of the bagplot on base of a sample if there are more then `approx.limit` data points.

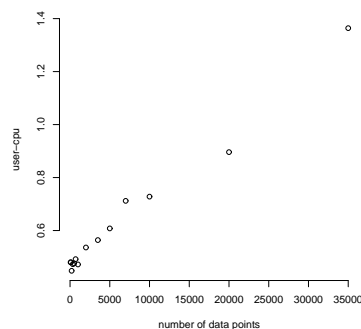
```
4 <large 4> ≡  
  seed<-174; n<-3000  
  <define bagplot 32>  
  <define rnorm data data, seed: seed, size: n 114>  
  datan<-rbind(data,c(105,295))  
  bagplot(datan,factor=2.5,create.plot=TRUE,approx.limit=1000,  
    cex=0.5,show.outlier=TRUE,show.looppoints=TRUE,  
    show.bagpoints=TRUE,dkmethod=2,show.loophull=TRUE,  
    show.baghull=TRUE,verbose=FALSE,debug.plots="no")  
  title(paste("seed:",seed," / n:",n))
```



1.4 Size of data set

The time for computation increases with the number of observations. To illustrate the run times we measure the times necessary for rnorm data sets of different sizes and plot the results.

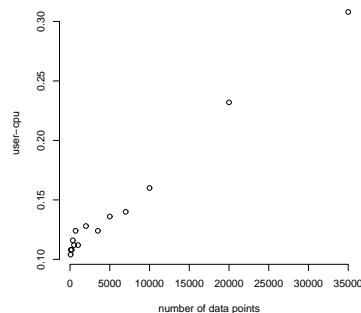
```
5 <rnorm, different sizes 5> ≡
  <define bagplot 32>
  nn<-c(50,70,100,200,350); nn<-c(nn,10*nn,100*nn);nn<-nn[-1]
  result<-1:length(nn)
  for(j in seq(along=nn)){
    seed<-111; set.seed(seed); n<-nn[j]
    xy<-cbind(rnorm(n),rnorm(n))
    result[j]<-system.time(
      bagplot(xy,factor=3,create.plot=FALSE,approx.limit=300,
        show.outlier=TRUE,show.looppoints=TRUE,show.bagpoints=TRUE,
        show.whiskers=TRUE,show.loophull=TRUE,show.baghull=TRUE,
        verbose=FALSE)
    )[1]
  }
  plot(nn,result,bty="n",ylab="user-cpu",xlab="number of data points")
  names(result)<-nn; result
```



```
Wed Aug 29 11:29:35 2007
 70 100 200 350 500 700 1000 2000 3500 5000 7000 10000 20000 35000
0.480 0.480 0.448 0.473 0.476 0.492 0.472 0.536 0.564 0.608 0.712 0.728 0.896 1.364
```

Rerunning gives the following results:

```
Wed Oct 3 11:46:05 2012
 70 100 200 350 500 700 1000 2000 3500 5000 7000 10000 20000
0.104 0.108 0.108 0.116 0.112 0.124 0.112 0.128 0.124 0.136 0.140 0.160 0.232
35000
0.308
```

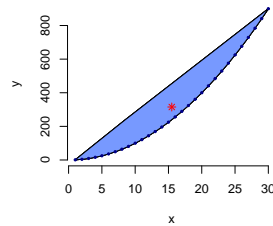


1.5 "Depth-One" data sets

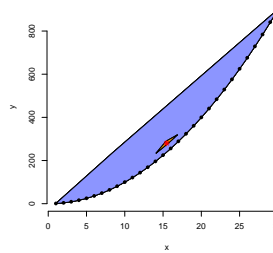
It is very interesting to test extrem cases. What happens if the depths of all points are one?

6

```
<quadratic 6> ≡
<define bagplot 32>
bagplot(x=1:30,y=(1:30)^2,verbose=FALSE,dkmethod=2)
points(x=1:30,y=(1:30)^2,col="black",pch=16)
```



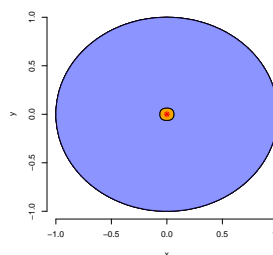
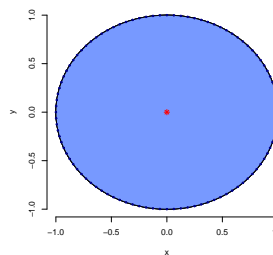
In the 2012 version the region of the median region is shown.



7

```
<circle 7> ≡
<define bagplot 32>
n<-100;bagplot(x=cos((1:n)/n*2*pi),y=sin((1:n)/n*2*pi),
precision=1,verbose=FALSE,dkmethod=2,debug.plot=FALSE)$center
```

On the right side we find the version 12/2012

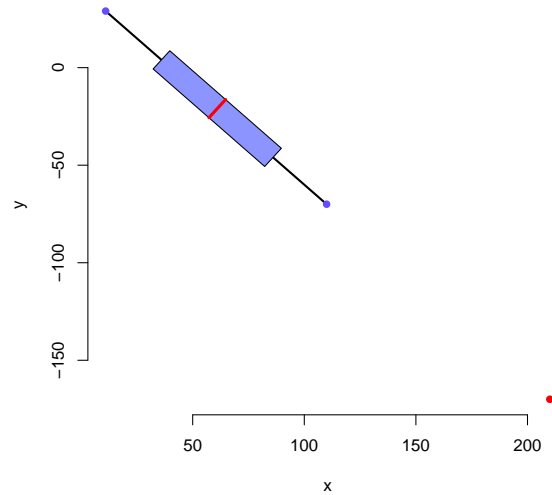


1.6 Degenerated data sets

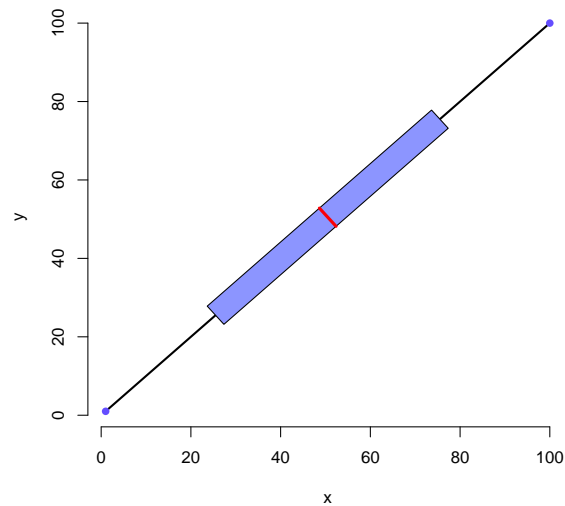
What happens if all the data points lie in a one dimensional subspace?

8 $\langle \text{onedim } 8 \rangle \equiv$
`bagplot(x=10+c(1:100,200),y=30-c(1:100,200),verbose=FALSE)`

Here is a second one dim data set.



9 $\langle \text{one dim test } 9 \rangle \equiv$
`bagplot(x=(1:100),y=(1:100),verbose=FALSE)`

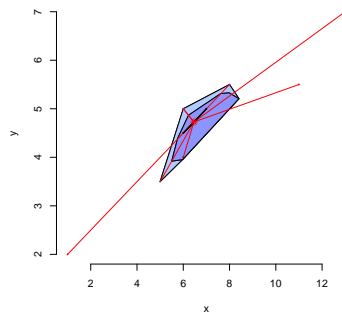


1.7 Data set from the mail of M. Maechler

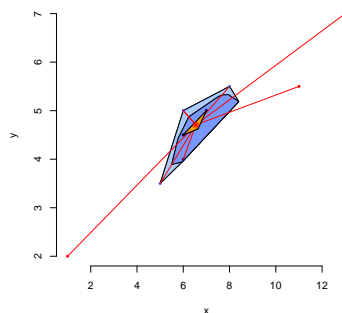
The data set of M. Maechler is discussed within R-help. Decide of yourself if our bagplot is acceptable. Maybe this doesn't matter because mostly a data set is *in regular position* (Rousseeuw, Ruts 1998) and there are no identical coordinates. But it may happen, e.g. in the car data set there are two points that are identical. M. Maechler wrote in a reply concerning a bagplot question that the correct Tukey median is (6.75 , 4.875) and not (6.544480, 4.708483) that is computed by our bagplot procedure.

10 `<data set of Martin Maechler 10> ≡
 <define bagplot 32>
 <assing data set of Martin Maechler to x0 and y0 11>
 bagplot(x0,y0,show.baghull=TRUE,show.loophull=TRUE,
 create.plot=TRUE,show.whiskers=TRUE,factor=3,
 debug.plots="notall",dkmethod=2,verbose=FALSE,
 precision=1)$center
 #abline(h=4.85,v=6.75)`

Bagplot of version 12/2012:



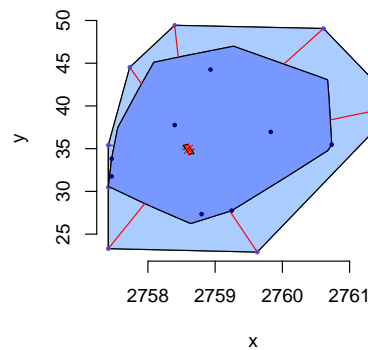
11 `<assing data set of Martin Maechler to x0 and y0 11> ≡ c(10, 22, 23, 25)
 x0<-c(1,5, 6,6, 6, 6,6,7,7,8, 11, 13) #; x0 <- c(x0, 8)
 y0<-c(2,3.5,4,4.5,4.5,5,5,5,5,5.5,5.5, 7) #; y0 <- c(y0, 7)`
 Here is the result of older bagplot versions:



1.8 Data sets of Wouter Meuleman, running in an error with version 09/2005

An old bagplot version runs into errors with following data set. During the computation of `<find hull . bag 77>` some NaN values occurred.

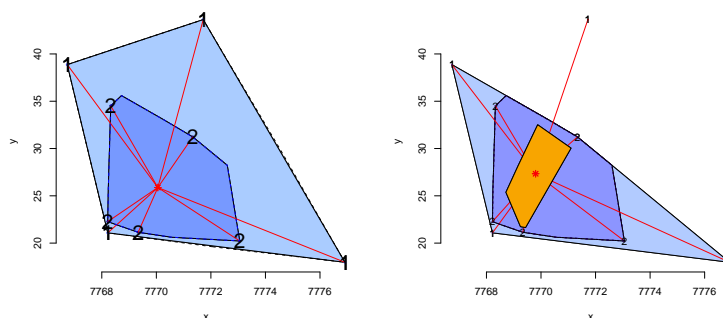
```
12 <data set 2 of Wouter Meuleman 12> ≡
a<-gsub("\n"," ",c("3 2759.626 22.90411 6 2757.461 31.75789 13 2758.931 44.25797
15 2757.411 30.47785 16 2761.720 40.01067 18 2759.827 36.97118 19 2758.398 49.43611
21 2757.411 23.30404 26 2757.461 33.81379 27 2758.398 37.75841 28 2759.244 27.74002
32 2757.411 35.40853 34 2760.734 35.47206 38 2760.612 49.05950 39 2757.730 44.51406
40 2758.798 27.33595"))
a<-unlist(strsplit(paste(a,collapse=" "), " "))
a<-as.numeric(a[a!=""]); a<-matrix(a,ncol=3,byrow=TRUE)
<define bagplot 32>
bagplot(a[,2],a[,3],verbose=FALSE,precision=1)
```



On 2006/02/17 some lines of code have been changed to remove the NaN values.

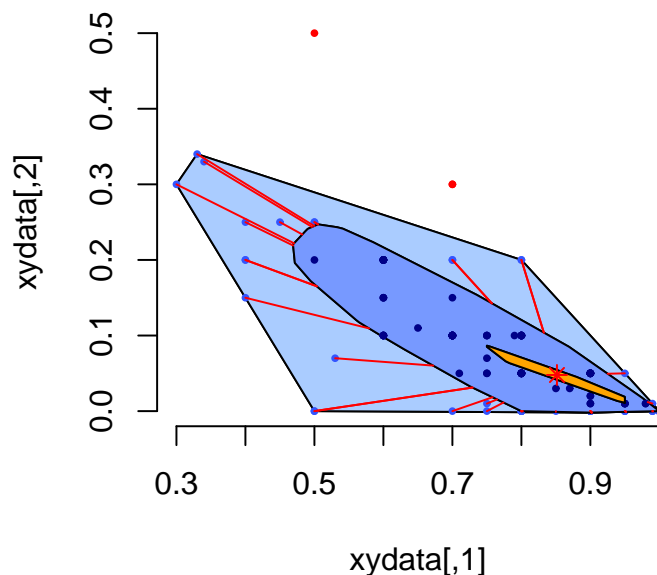
```
13 <data set 1 of Wouter Meuleman 13> ≡ c 117
a<-gsub("\n"," ",c("1 7766.734 38.86814 2 7768.329 34.50661 3 7769.335
21.14797 4 7768.221 21.08619 5 7776.913 17.97344 6 7768.221 22.27727 8
7771.719 43.62978 9 7773.056 20.22909 12 7771.334 31.22399"))
a<-unlist(strsplit(paste(a,collapse=" "), " "))
a<-as.numeric(a[a!=""]); a<-matrix(a,ncol=3,byrow=TRUE)
<define bagplot 32>
bagplot(a[,2],a[,3],verbose=TRUE,dkmethod=2,precision=1,factor=3)
# points(a[,2:3],pch=1,cex=2.5,col="red")
```

On the left we find a result of an older version of `bagplot()`. However, the central region hasn't been found – compare the following bagplot. To integrate the point at the top edge of this picture you have to enlarge the factor a little bit.



The following data set was proposed by Ben Greiner in January 2007. In the version of November 2012 the central region is quite narrow and not a long yellow field as shown in the next figure.

14 `<test: data set of Ben Greiner 14> ≡`
`<define bagplot 32>`
`greiner.data<-cbind(c(1,1,1,0.7,0.8,0.98,0.9,0.85,1,1,0.7,1,0.65,`
`0.8,0.5,0.7,0.95,0.7,0.8,0.8,0.75,1,0.95,0.7,0.95,0.8,0.75,0.7,0.85,`
`0.8,0.8,1,0.5,0.9,0.7,0.8,0.6,0.9,0.98,1,0.5,0.45,0.95,1,0.9,0.9,`
`0.7,1,1,0.7,1,0.4,0.9,0.85,0.75,1,0.5,0.9,0.4,0.95,0.8,0.95,0.99,`
`1,0.34,0.6,1,0.9,0.6,0.7,0.8,0.7,0.95,1,0.6,0.99,0.85,0.78,0.8,1,`
`0.4,1,0.33,0.99,0.6,0.8,0.85,0.75,0.9,0.9,1,0.9,1,0.8,1,0.9,1,0.71,`
`0.4,0.8,1,0.7,1,0.8,1,0.6,0.6,1,0.6,1,1,0.7,0.85,1,0.8,1,0.95,0.8,`
`0.9,0.8,0.6,0.85,1,0.9,0.9,0.8,1,1,0.6,0.9,1,1,0.5,0.75,0.53,0.8,`
`0.7,0.3,0.8,0.9,0.7,0.8,0.6,0.9,0.8,0.8,0.6,1,0.6,1,1,0.9,0.8,0.7,`
`0.6,0.8,1,0.5,0.85,1,0.75,1,0.8,1,0.85,1,0.75,0.8,0.7,0.87,1,1,1,`
`0.7,0.79,0.8,0.6,0.9,0.6,0.8,0.6,0.7,0.8,0.99,0.9,0.75),`
`c(0,0,0,0.1,0,0.01,0,0.05,0,0,0.1,0,0.11,0.1,0,0.1,0,0.3,0,0,0.07,`
`0,0.01,0.1,0,0.05,0.05,0.3,0.05,0.1,0,0,0.25,0,0.1,0.05,0.2,0.05,`
`0.01,0,0.25,0.25,0.05,0,0.05,0.02,0.1,0,0,0.1,0,0.25,0.03,0.05,0.1,`
`0,0.2,0.01,0.2,0,0.1,0.01,0,0,0.33,0.1,0,0.05,0.15,0.1,0.1,0.01,`
`0,0.1,0,0.05,0.07,0.1,0,0.15,0,0.34,0,0.15,0.1,0.03,0,0,0.05,0,0.05,`
`0,0.2,0,0.01,0,0.05,0.2,0.05,0,0.15,0,0.05,0,0.2,0.2,0,0,0.2,`
`0.05,0,0.05,0,0.01,0,0.05,0.05,0.1,0.05,0,0.05,0.05,0.05,0,0,0.15,`
`0.05,0,0,0.05,0.07,0.05,0.1,0.3,0.05,0,0.1,0.1,0.2,0.02,0.05,0.2,`
`0.2,0,0.1,0,0,0.05,0.05,0.1,0.2,0.1,0,0.5,0,0,0.1,0,0.05,0,0.05,0,`
`0.01,0.05,0.1,0.03,0,0,0,0.1,0.05,0.1,0.05,0.1,0,0.2,0.2,0,0.01,0,0.1))`
`bagplot(greiner.data,precision=3)`
`" "`



1.9 Data sets of Amanda Joy Shaker

The bagplot function (version from 2011) runs into an error by using a data set of Amanda Joy Shaker (AJS):

```
Error in chull(pg[, 1], pg[, 2]) :
  NA/NaN/Inf in externem Funktionsaufruf (arg 2)
```

This problem was caused by:

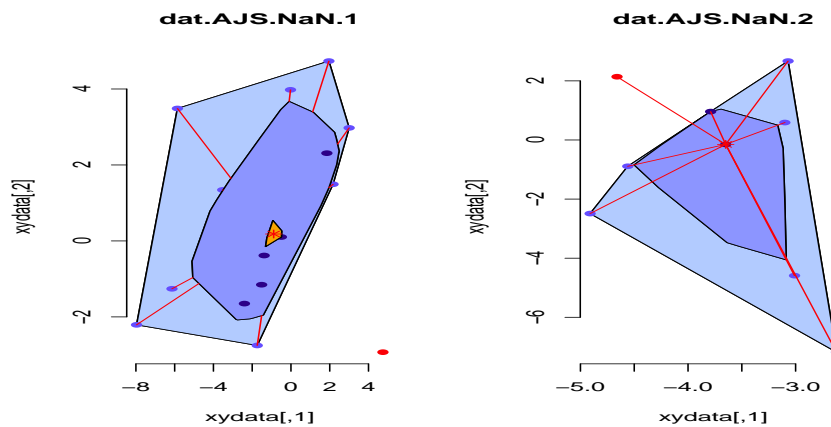
```
pgl<-pgl[c(TRUE,(abs(diff(pgl[,1]))>limit)|(abs(diff(pgl[,2]))>limit)),]
instead of the correct statement
```

```
pgl<-pgl[c(      (abs(diff(pgl[,1]))>limit)|(abs(diff(pgl[,2]))>limit),TRUE),]
```

Date of correction: Wed Oct 3 12:15:01 2012 In the right figure the most upper point joins the loop only if the precision up to 4. Otherwise the point will be isolated plotted.

15

```
<data set of Amanda Joy Shaker 15> ≡
<define bagplot 32>
par(mfrow=c(1,2))
dat.AJS.NaN.1 <- cbind(c(1.852, -1.740, 2.170, 4.745, -5.858, -3.521, 3.001, -6.139,
-1.513, -1.384, 1.957, -0.023, -0.495, -2.401, -7.966),
c(2.310, -2.752, 1.492, -2.929, 3.487, 1.343, 2.976, -1.260,
-1.153, -0.384, 4.735, 3.978, 0.103, -1.650, -2.209)
)
bagplot(dat.AJS.NaN.1,cex=1,main="dat.AJS.NaN.1")
dat.AJS.NaN.2 <- cbind(c(-4.66, -2.62, -3.65, -3.07, -4.91, -4.56, -3.79, -3.10, -3.01),
c(2.14, -7.18, -0.15, 2.67, -2.49, -0.89, 0.96, 0.59, -4.59))
bagplot(dat.AJS.NaN.2,cex=1,main="dat.AJS.NaN.2",precision=1)
```



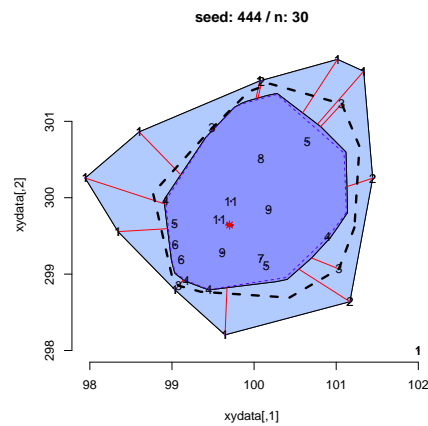
Further data sets which run into difficulties can be found in the appendix. Here is the list of the data sets used for discussion with AJS.

[1]	dat.AJS.cen.51	dat.AJS.cen.51a	dat.AJS.cen.52	dat.AJS.cen.53
[5]	dat.AJS.chull.1	dat.AJS.chull.2	dat.AJS.chull.2a	dat.AJS.fence.1
[9]	dat.AJS.fence.2	dat.AJS.NA.1	dat.AJS.NA.10	dat.AJS.NA.11
[13]	dat.AJS.NA.12	dat.AJS.NA.13	dat.AJS.NA.14	dat.AJS.NA.15
[17]	dat.AJS.NA.16	dat.AJS.NA.17	dat.AJS.NA.18	dat.AJS.NA.19
[21]	dat.AJS.NA.2	dat.AJS.NA.20	dat.AJS.NA.21	dat.AJS.NA.22
[25]	dat.AJS.NA.23	dat.AJS.NA.24	dat.AJS.NA.25	dat.AJS.NA.26
[29]	dat.AJS.NA.3	dat.AJS.NA.4	dat.AJS.NA.5	dat.AJS.NA.6
[33]	dat.AJS.NA.7	dat.AJS.NA.8	dat.AJS.NA.9	dat.AJS.NaN.1
[37]	dat.AJS.NaN.2	dat.AJS.NAN.A	dat.AJS.NAN.B	dat.AJS.NAN.C
[41]	dat.AJS.NAN.D	dat.AJS.NAN.E	dat.AJS.NAN.F	dat.AJS.NAN.G

1.10 Bagplot with additional graphical supplements

Verbose computation of bagplot of a sample of 100 *rnorm* points and an outlier is performed by the following code chunk. With the verbose option the h-depths of the data points are shown in the plot and some of the intermediate results are printed during the the computation.

```
16 <verboetest 16> ≡  
  seed<-444; n<-30  
  <define rnorm data data, seed: seed, size: n 114>  
  datan<-rbind(data,c(102,298))  
  bagplot(datan,factor=2.5,create.plot=TRUE,approx.limit=300,  
    show.outlier=TRUE,show.looppoints=TRUE,show.bagpoints=TRUE,dkmethod=2,  
    show.whiskers=TRUE,show.loophull=TRUE,show.baghull=TRUE,verbose=TRUE)  
  title(paste("seed:",seed,"/ n:",n))
```

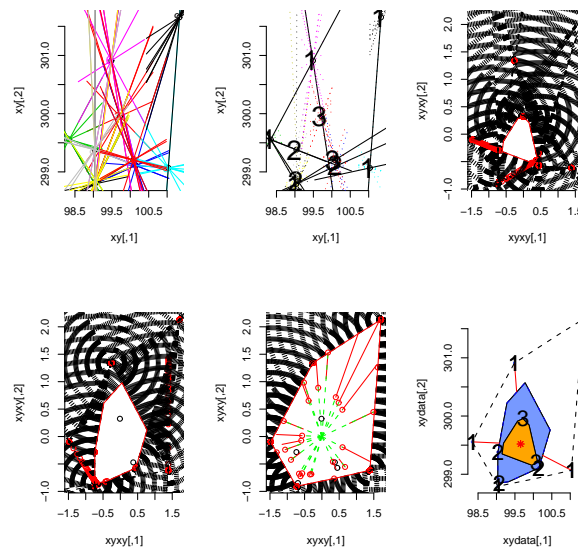


1.11 Debugging plots with additional elements

Here is an example of plots generated with option `debug.plots="all"`. This option has been helpful during debugging and now the plots can be classified as R art.

17

```
<debugplot 17> ≡
seed<-444; n<-30
<define bagplot 32>
<define rnorm data data, seed: seed, size: n 114>
datan<-data[1:10,] #datan<-cbind(c(1:100,200),c(1:100,200))
par(mfrow=c(2,3))
bagplot(datan,factor=2.5,create.plot=TRUE,approx.limit=300,
  show.outlier=TRUE,show.looppoints=TRUE,show.bagpoints=TRUE,
  show.whiskers=TRUE,show.loophull=FALSE,show.baghull=TRUE,dkmethod=2,
  debug.plots="all", precision=.4, verbose=TRUE); par(mfrow=c(1,1))
```



Remark: Setting a higher precision will discover a central region of points with a higher depth.

2 Bagplots by an alternative approach, proposed by Rousseeuw, Ruts and Tukey

As mentioned above there is a solution using a fortran procedure for generating bagplots, see:

<http://www.statistik.tuwien.ac.at/public/filz/students/edavis/ws0607/skriptum/page134.html>.

To get the procedure work you have to perform the following steps:

- fetch the fortran code by downloading

```
$ get ftp://ftp.win.ua.ac.be/pub/software/agoras/newfiles/bagplot.tar.gz
```

– this link has been found on the web page: <http://www.agoras.ua.ac.be/Locdept.htm>
- unzip and unpack the tar.gz-file

```
$ gunzip bagplot.tar.gz; tar -xvf bagplot.tar
```
- translate the fortran program `bagplot.f` and generate the object file `bagplot.so`

```
$ R CMD SHLIB -o bagplot.so bagplot.f
```
- download bagplot-R-function

```
$ get http://www.statistik.tuwien.ac.at/public/filz/students/edavis/ws0607/skriptum/bagplot.R
```
- start R and load so-file

```
18 < * 18 > ≡  
    dyn.load("Tukey/bagplot.so")
```

- source bagplot function; to avoid conflicts in the names we change the name of the bagplot function of Rousseeuw, Ruts, and Tukey to BAGPLOT.

```
19 < * 18 > + ≡  
    BAGPLOT<-readLines("Tukey/BAGPLOT.R")  
    eval(parse(text=sub("^bagplot", "\"BAGPLOT", BAGPLOT))); "ok"  
    args(BAGPLOT)
```

Here are the arguments of BAGPLOT():

Wed Aug 29 15:19:43 2007

```
function (x, y, plotinbag = T, plotoutbag = T, ident = T, drawfence = F,  
    drawloop = T, truncxmin = NULL, truncxmax = NULL, truncymin = NULL,  
    truncymax = NULL, xlab = "x", ylab = "y", ...)
```

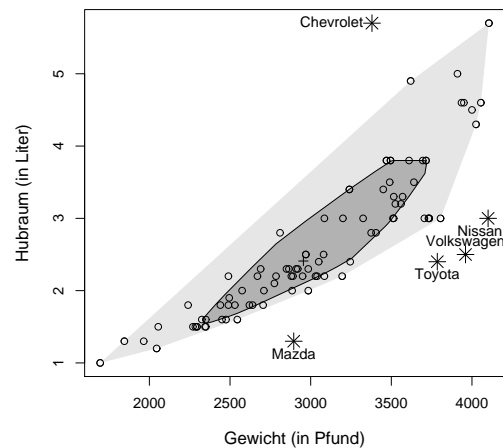
- compute an example bagplot.

```
20 < * 18 > + ≡  
    library(MASS)  
    data(Cars93); x<-Cars93[, "Weight"]; y<-Cars93[, "EngineSize"]  
    names(x)<-dimnames(Cars93)[[1]]; names(y)<-dimnames(Cars93)[[1]]  
    # par(mar=c(4,4,1,1))  
    BAGPLOT(x,y,xlab="Gewicht (in Pfund)",ylab="Hubraum (in Liter)",  
    ##      main="Press Mouse BUTTON!",  
           cex.lab=1.2)  
    text(3380,5.7,"Chevrolet",pos=2); text(2895,1.3,"Mazda",pos=1)  
    text(4050,3.0,"Nissan",pos=1);   text(3785,2.4,"Toyota",pos=1)  
    text(3960,2.5,"Volkswagen",pos=3)
```

Here is the numerical result

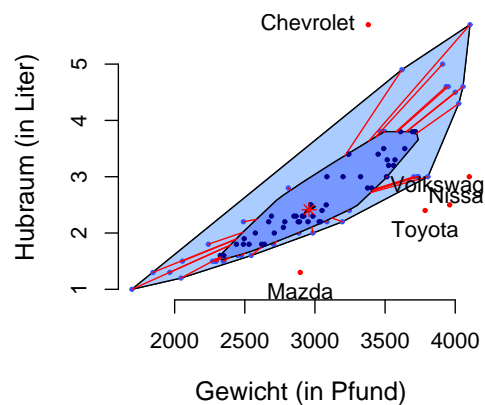
[1] The coordinates of the Tukey median are (2954.84 , 2.40962).

and the bagplot:



A reconstruction of this plot can be done by our bagplot function. For a suitable loop you have to set factor=2.8.

```
21 (*18)+ ≡
library(MASS)
data(Cars93); x<-Cars93[,"Weight"]; y<-Cars93[,"EngineSize"]
names(x)<-dimnames(Cars93)[[1]]; names(y)<-dimnames(Cars93)[[1]]
center<-bagplot(x,y,factor=2.8,xlab="Gewicht (in Pfund)",precision=3,
               ylab="Hubraum (in Liter)",cex.lab=1.2,cex=0.7)$center
text(3380,5.7,"Chevrolet",pos=2); text(2895,1.3,"Mazda",pos=1)
text(4050,3.0,"Nissan",pos=1); text(3785,2.4,"Toyota",pos=1)
text(3960,2.5,"Volkswagen",pos=3)
center
```



We find the center:

Tue Dec 4 08:30:45 2012
 [1] 2955.184896 2.410503

The difference may be caused by numerical difficulties.

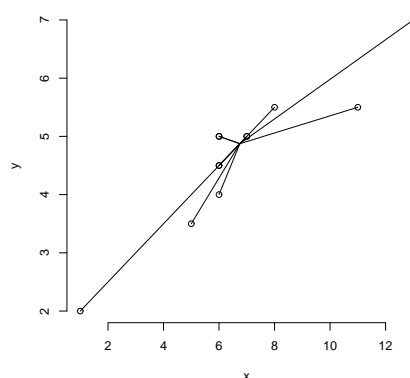
Test of data set of Martin Maechler. As a second example we check the bagplot functions by the data set of Martin Maechler.

22 `<BAGPLOT of data set of Martin Maechler 22> ≡`
`<assing data set of Martin Maechler to x0 and y0 11>`
`BAGPLOT(x0,y0)`

We get the numerical result ...

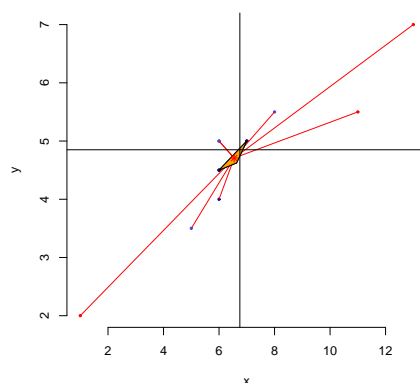
[1] The coordinates of the Tukey median are (6.75 , 4.875).

and the following plot



Our procedure will compute slightly different results:

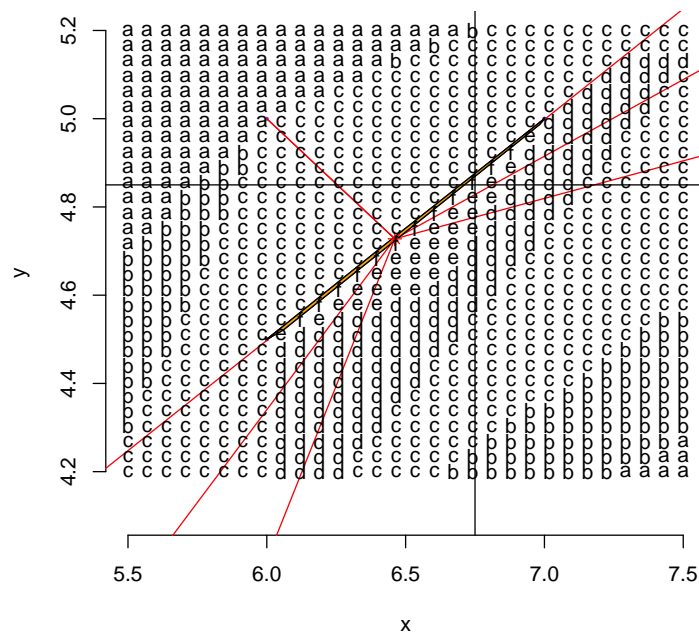
23 `<bagplot of data set of Martin Maechler 23> ≡`
`<assing data set of Martin Maechler to x0 and y0 11>`
`center<-bagplot(x0, y0, show.baghull=FALSE, show.loophull=FALSE,`
`create.plot=TRUE, show.whiskers=TRUE, factor=3,`
`dkmethod=2, precision=3,`
`xlim=c(5.5,7.5), ylim=c(4.1,5.2))$center`
`abline(h=4.85,v=6.75); center`



The two lines mark the Tukey median computed by BAGPLOT. Our median is (precision=3):

```
[1] 6.483859 4.741716
```

```
24  (*18)+ ≡
    TP <- cbind(as.vector(matrix(seq(5.5,7.5,length=30),30,30)),
                as.vector(matrix(seq(4.2,5.2,length=30),30,30,TRUE)))
    TPD <- hdepth(TP,cbind(x0,y0))
    points(TP,pch=c(letters,LETTERS)[TPD+1])
```



Now we check the stability of the functions by exchanging the variables.

```
25  (BAGPLOT of data set of Martin Maechler, exchanged variables 25) ≡
    (assing data set of Martin Maechler to x0 and y0 11)
    BAGPLOT(y0,x0)
```

```
[1] The coordinates of the Tukey median are ( 4.84231 , 6.68461 ).
```

There is a difference! The relative difference is:

```
26  (BAGPLOT of data set of Martin Maechler, relative difference 26) ≡
    abs((c(6.483859, 4.741716)-rev(c(4.84231,6.68461)))/c(6.483859, 4.741716))
```

```
[1] 0.03096165 0.02121468
```

How will our function master the test?

```
27  (bagplot of data set of Martin Maechler, exchanged variables 27) ≡
    center.ex<-bagplot(y0, x0, show.baghull=FALSE, show.loophull=FALSE,
```

```

        create.plot=TRUE, show.whiskers=TRUE, factor=3,
        dkmethode=2, precision=3)$center
cat("center original data:", center )
cat("center exchanged data:", rev(center.ex))

```

```

center original data: 6.483859 4.741716
center exchanged data: 6.483912 4.741755

```

The relative difference is approximately 0.07%. If we increase the precision by `precision=5` the difference is reduced as we like it:

```

28 <bagplot of data set of Martin Maechler, difference if precision is increased 28> ≡
center      <-bagplot(x0,y0,create.plot=FALSE,factor=3,precision=5)$center
center.ex   <-bagplot(y0,x0,create.plot=FALSE,factor=3,precision=5)$center
cat("center original data:", center )
cat("center exchanged data:", rev(center.ex))
abs(center-center.ex[2:1])/center

```

The results seems to be identical for we get:

```

center original data: 6.447704 4.723065
center exchanged data: 6.44778 4.723117
Tue Dec 4 08:41:05 2012
[1] 1.182812e-05 1.094571e-05

```

By analyzing the scatterplot we find that the area of the points with h-depth 4 build a triangle. The corners of this triangle are: (6, 4.5), (7,5) and (6.625, 14.125). The center of its gravity is equal to the mean of the three points and we get the Tukey median (6.541666, 4.7083333). Our bagplot function finds a result that is very near to the one computed by hand.

Memory faults. There are some other problems with the implementation via the fortran procedure because we got some memory faults during testing BAGPLOT. These errors killed the R process and some of the computed results got lost. But it was not difficult to reconstruct them ... by relax.

3 Arguments and output of bagplot, the help page and some links

A summary of the arguments can be found by `args()`.

29 `<args 29> ≡`
`args(bagplot)`

```
function (x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
  show.outlier = TRUE, show.whiskers = TRUE, show.looppoints = TRUE,
  show.bagpoints = TRUE, show.loophull = TRUE, show.baghull = TRUE,
  create.plot = TRUE, add = FALSE, pch = 16, cex = 0.4, dkmethod = 2,
  precision = 1, verbose = FALSE, debug.plots = "no",
  col.loophull = "#aaccff", col.looppoints = "#3355ff",
  col.baghull = "#7799ff", col.bagpoints = "#000088",
  transparency = FALSE, ...)
```

The output of `bagplot` is a list of the relevant quantities of the constructed bagplot. To identify singular points, use `identify()`. Here is a short description of the return values:

<code>center</code>	Tukey median
<code>hull.loop</code>	set of points of polygon that defines the loop
<code>hull.bag</code>	set of points of polygon that defines the bag
<code>hull.center</code>	region of points with maximal ldepth
<code>pxy.outlier</code>	outlier
<code>pxy.outer</code>	outer points
<code>pxy.bag</code>	points in bag
<code>hdepth</code>	location depth of data points in xy
<code>is.one.dim</code>	is TRUE if data set is one dimensional
<code>prdata</code>	result of PCA
<code>xydata</code>	data set
<code>xy</code>	sample of data set

30 `<define help of hdepth 30> ≡`

```
\name{hdepth}
\alias{hdepth}
\title{ hdepth of points }
\description{
  \code{hdepth()} computes the h-depths of points.
}

\usage{
hdepth(tp, data, number.of.directions=181)
}

\arguments{
  \item{tp}{ two column matrix of the coordinates of points which h-depths are needed }
  \item{data}{ two column matrix of the coordinates of the points of a data set}
  \item{number.of.directions}{ number of directions to be checked }
}

\details{
The function \code{hdepth} computes the h-depths of the points
\code{tp} relative to data set \code{data}. If \code{data} is
missing \code{tp} will also be taken as data set.
}
```

```

\value{
the h-depths of the test points
}

\author{ Peter Wolf }
\note{
  Version of bagplot: 12/2012 }
\seealso{ \code{\link[aplpack]{bagplot}} }
\examples{
  # computation of h-depths
  data <- cbind(rnorm(40), rnorm(40))
  xy <- cbind(runif(50,-2,2),runif(50,-2,2))
  bagplot(data); text(xy, as.character(hdepth(xy,data)))
}

```

The help page is defined as a code chunk.

```

31 <define help of bagplot 31> ≡
  \name{bagplot}
  \alias{bagplot}
  \alias{compute.bagplot}
  \alias{plot.bagplot}
  \title{ bagplot, a bivariate boxplot }
  \description{
    \code{compute.bagplot()} computes an object describing a bagplot
    of a bivariate data set. \code{plot.bagplot()} plots a bagplot object.
    \code{bagplot()} computes and plots a bagplot.
  }
  \usage{
bagplot(x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
  show.outlier = TRUE, show.whiskers = TRUE,
  show.looppoints = TRUE, show.bagpoints = TRUE,
  show.loophull = TRUE, show.baghull = TRUE,
  create.plot = TRUE, add = FALSE, pch = 16, cex = 0.4,
  dkmethode = 2, precision = 1, verbose = FALSE,
  debug.plots = "no", col.loophull="#aaccff",
  col.looppoints="#3355ff", col.baghull="#7799ff",
  col.bagpoints="#000088", transparency=FALSE, ...
)

compute.bagplot(x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
  dkmethode=2,precision=1,verbose=FALSE,debug.plots="no")
\method{plot}{bagplot}(x,
  show.outlier = TRUE, show.whiskers = TRUE,
  show.looppoints = TRUE, show.bagpoints = TRUE,
  show.loophull = TRUE, show.baghull = TRUE,
  add = FALSE, pch = 16, cex = 0.4, verbose = FALSE,
  col.loophull="#aaccff", col.looppoints="#3355ff",
  col.baghull="#7799ff", col.bagpoints="#000088",
  transparency=FALSE,\ldots)
}

\arguments{
  \item{x}{ x values of a data set;
    in \code{bagplot}: an object of class \code{bagplot}
    computed by \code{compute.bagplot} }
  \item{y}{ y values of the data set }
  \item{factor}{ factor defining the loop }

```

```

\item{na.rm}{ if TRUE 'NA' values are removed otherwise exchanged by median}
\item{approx.limit}{ if the number of data points exceeds
  \code{approx.limit} a sample is used to compute
  some of the quantities; default: 300 }
\item{show.outlier}{ if TRUE outlier are shown }
\item{show.whiskers}{ if TRUE whiskers are shown }
\item{show.looppoints}{ if TRUE loop points are plotted }
\item{show.bagpoints}{ if TRUE bag points are plotted }
\item{show.loophull}{ if TRUE the loop is plotted }
\item{show.baghull}{ if TRUE the bag is plotted }
\item{create.plot}{ if FALSE no plot is created }
\item{add}{ if TRUE the bagplot is added to an existing plot }
\item{pch}{ sets the plotting character }
\item{cex}{ sets characters size }
\item{dkmethod}{ 1 or 2, there are two method of
  approximating the bag, method 1 is very rough (only based on observations )
\item{precision}{ precision of approximation, default: 1 }
\item{verbose}{ automatic commenting of calculations }
\item{debug.plots}{ if TRUE additional plots describing
  intermediate results are constructed }
\item{col.loophull}{ color of loop hull }
\item{col.looppoints}{ color of the points of the loop }
\item{col.baghull}{ color of bag hull }
\item{col.bagpoints}{ color of the points of the bag }
\item{transparency}{ see section details }
\item{\dots}{ additional graphical parameters }
}
\details{
A bagplot is a bivariate generalization of the well known
boxplot. It has been proposed by Rousseeuw, Ruts, and Tukey.
In the bivariate case the box of the boxplot changes to a
convex polygon, the bag of bagplot. In the bag are 50 percent
of all points. The fence separates points within the fence from
points outside. It is computed by increasing the
the bag. The loop is defined as the convex hull containing
all points inside the fence.
If all points are on a straight line you get a classical
boxplot.
\code{bagplot()} plots bagplots that are very similar
to the one described in Rousseeuw et al.
Remarks:
The two dimensional median is approximated.
For large data sets the error will be very small.
On the other hand it is not very wise to make a (graphical)
summary of e.g. 10 bivariate data points.

In case you want to plot multiple (overlapping) bagplots,
you may want plots that are semi-transparent. For this
you can use the \code{transparency} flag.
If \code{transparency==TRUE} the alpha layer is set to '99' (hex).
This causes the bagplots to appear semi-transparent,
but ONLY if the output device is PDF and opened using:
\code{pdf(file="filename.pdf", version="1.4")}.
For this reason, the default is \code{transparency==FALSE}.
This feature as well as the arguments
to specify different colors has been proposed by Wouter Meuleman.
}
\value{

```

```

\code{compute.bagplot} returns an object of class
\code{bagplot} that could be plotted by
\code{plot.bagplot()}.
An object of the bagplot class is a list with the following
elements: \code{center} is a two dimensional vector with
the coordinates of the center. \code{hull.center} is a
two column matrix, the rows are the coordinates of the
corners of the center region. \code{hull.bag} and
\code{hull.loop} contain the coordinates of the hull of the bag
and the hull of the loop. \code{pxy.bag} shows you the
coordinates of the points of the bag. \code{pxy.outer} is
the two column matrix of the points that are within the
fence. \code{pxy.outlier} represent the outliers. The vector
\code{hdepths} shows the depths of data points. \code{is.one.dim}
is \code{TRUE} if the data set is (nearly) one dimensional.
The dimensionality is decided by analysing the result of \code{prcomp}
which is stored in the element \code{prdata}. \code{xy} shows you
the data that are used for the bagplot. In the case of very large
data sets subsets of the data are used for constructing the
bagplot. A data set is very large if there are more data points
than \code{approx.limit}. \code{xydata} are the input data structured
in a two column matrix.
}
\references{ P. J. Rousseeuw, I. Ruts, J. W. Tukey (1999):
  The bagplot: a bivariate boxplot, The American
  Statistician, vol. 53, no. 4, 382--387 }
\author{ Peter Wolf }
\note{
  Version of bagplot: 10/2012 }
\seealso{ \code{\link[graphics]{boxplot}} }
\examples{
  # example: 100 random points and one outlier
  dat<-cbind(rnorm(100)+100,rnorm(100)+300)
  dat<-rbind(dat,c(105,295))
  bagplot(dat,factor=2.5,create.plot=TRUE,approx.limit=300,
    show.outlier=TRUE,show.looppoints=TRUE,
    show.bagpoints=TRUE,dkmethod=2,
    show.whiskers=TRUE,show.loophull=TRUE,
    show.baghull=TRUE,verbose=FALSE)
  # example of Rousseeuw et al., see R-package rpart
  cardata <- structure(as.integer( c(2560,2345,1845,2260,2440,
    2285, 2275, 2350, 2295, 1900, 2390, 2075, 2330, 3320, 2885,
    3310, 2695, 2170, 2710, 2775, 2840, 2485, 2670, 2640, 2655,
    3065, 2750, 2920, 2780, 2745, 3110, 2920, 2645, 2575, 2935,
    2920, 2985, 3265, 2880, 2975, 3450, 3145, 3190, 3610, 2885,
    3480, 3200, 2765, 3220, 3480, 3325, 3855, 3850, 3195, 3735,
    3665, 3735, 3415, 3185, 3690, 97, 114, 81, 91, 113, 97, 97,
    98, 109, 73, 97, 89, 109, 305, 153, 302, 133, 97, 125, 146,
    107, 109, 121, 151, 133, 181, 141, 132, 133, 122, 181, 146,
    151, 116, 135, 122, 141, 163, 151, 153, 202, 180, 182, 232,
    143, 180, 180, 151, 189, 180, 231, 305, 302, 151, 202, 182,
    181, 143, 146, 146)), .Dim = as.integer(c(60, 2)),
    .Dimnames = list(NULL, c("Weight", "Disp.")))
  bagplot(cardata,factor=3,show.baghull=TRUE,
    show.loophull=TRUE,precision=1,dkmethod=2)
  title("car data Chambers/Hastie 1992")
  # points of y=x*x
  bagplot(x=1:30,y=(1:30)^2,verbose=FALSE,dkmethod=2)

```

```
# one dimensional subspace
bagplot(x=1:100,y=1:100)
}
\keyword{ misc }
\keyword{ hplot }
```

Here are some important links:

```
http://www.cim.mcgill.ca/~lsimard/Pattern/TheBag.htm
http://www.math.yorku.ca/SCS/Gallery/bright-ideas.html
http://maven.smith.edu/~streinu/Research/LocDepth/algorithm.html
http://www.agoras.ua.ac.be/abstract/Bagbiv97.htm
http://www.agoras.ua.ac.be/Locdept.htm
http://article.gmane.org/gmane.comp.lang.r.general/25235
http://finzi.psych.upenn.edu/R/Rhelp02a/archive/45106.html
http://delivery.acm.org/10.1145/370000/365565/
  p690-miller.pdf?key1=365565&key2=9093786211&coll=GUIDE&
  dl=GUIDE&CFID=53086693&CFTOKEN=38519152
http://www.cs.tufts.edu/research/geometry/half_space/
http://www.statistik.tuwien.ac.at/public/filz/students/edavis/
  ws0607/skriptum/page134.html
```

4 The definition of bagplot

The function `bagplot` is a container that calls the two functions `compute.bagplot` and `plot.bagplot`. The first one generates an object of class `bagplot` and the second one is called by the generic `plot` function.

```
32 <define bagplot 32> ≡  C 2, 3, 4, 5, 6, 7, 10, 12, 13, 14, 15, 17, 45, 85, 86, 91, 115, 116, 117, 118, 120, 121, 122,
124, 125, 129, 132, 142, 143, 145, 146, 152, 153, 154, 173, 174
  <define compute.bagplot 33>
  <define plot.bagplot 81>
  <define find.hdepths 57>
  <define find.hdepths.tp 60>
  <define function hdepth() 44>
  bagplot<-function(x,y,
    factor=3, # expanding factor for bag to get the loop
    na.rm=FALSE, # should 'NAs' values be removed or exchanged
    approx.limit=300, # limit
    show.outlier=TRUE, # if TRUE outlier are shown
    show.whiskers=TRUE, # if TRUE whiskers are shown
    show.looppoints=TRUE, # if TRUE points in loop are shown
    show.bagpoints=TRUE, # if TRUE points in bag are shown
    show.loophull=TRUE, # if TRUE loop is shown
    show.baghull=TRUE, # if TRUE bag is shown
    create.plot=TRUE, # if TRUE a plot is created
    add=FALSE, # if TRUE graphical elements are added to actual plot
    pch=16, cex=.4, # some graphical parameters
    dkmeth=2, # in 1:2; there are two methods for approximating the bag
    precision=1, # controls precision of computation
    verbose=FALSE, debug.plots="no", # tools for debugging
    col.loophull="#aaccff", # Alternatives: #ccffaa, #ffaacc
    col.looppoints="#3355ff", # Alternatives: #55ff33, #ff3355
    col.baghull="#7799ff", # Alternatives: #99ff77, #ff7799
    col.bagpoints="#000088", # Alternatives: #008800, #880000
    transparency=FALSE, ... # to define further parameters of plot
  ){
    if(missing(x)) return(<version of bagplot 1>)
```



```

bo<-compute.bagplot(x=x,y=y,factor=factor,na.rm=na.rm,
  approx.limit=approx.limit,dkmethod=dkmethod,
  precision=precision,verbose=verbose,debug.plots=debug.plots)
if(create.plot){
  plot(bo,
    show.outlier=show.outlier,
    show.whiskers=show.whiskers,
    show.looppoints=show.looppoints,
    show.bagpoints=show.bagpoints,
    show.loophull=show.loophull,
    show.baghull=show.baghull,
    add=add,pch=pch,cex=cex,
    verbose=verbose,
    col.loophull=col.loophull,
    col.looppoints=col.looppoints,
    col.baghull=col.baghull,
    col.bagpoints=col.bagpoints,
    transparency=transparency, ...
  )
}
invisible(bo)
}

```

compute.bagplot computes the relevant values to allow plot.bagplot to draw the bagplot.

```

33 <define compute.bagplot 33> ≡ C 32
compute.bagplot<-function(x,y,
  factor=3, # expanding factor for bag to get the loop
  na.rm=FALSE, # should NAs removed or exchanged
  approx.limit=300, # limit
  dkmethod=2, # in 1:2; method 2 is recommended
  precision=1, # controls precision of computation
  verbose=FALSE,debug.plots="no" # tools for debugging
){
  <version of bagplot 1>
  <body of compute.bagplot 34>
}

```

4.1 The body of compute.bagplot

Here you see the main steps of the construction of a bagplot.

```

34 <body of compute.bagplot 34> ≡ C 33
  <init 36>
  <check and handle linear case 52>
  <handle three or four data points 53>
  <standardize data and compute: xyxy, xym, xysd 54>
  <compute angles between points 55>
  <compute hdepths 56>
  <find k 61>
  <compute hdepths of test points to find center 62>
  if(dkmethod==1){
    <method one: find hulls of  $D_k$  and  $D_{k-1}$  66>
  }else{
    <method two: find hulls of  $D_k$  and  $D_{k-1}$  67>
  }
  <find value of lambda 76>
  <find hull.bag 77>

```

<find hull.loop 78>
<find points outside of bag but inside loop 79>
<find hull of loop 80>
<output result 35>

4.2 Output of bagplot

The following table of output values of bagplot is copy from section 2:

center	Tukey median
hull.loop	set of points of polygon that defines the loop
hull.bag	set of points of polygon that defines the bag
hull.center	region of points with maximal ldepth
pxy.outlier	outlier
pxy.outer	outer points
pxy.bag	points in bag
hdepth	location depth of data points in xy
is.one.dim	is TRUE if data set is one dimensional
prdata	result of PCA
xydata	data set
xy	sample of data set

These elements are return as a list.

```

35  <output result 35> ≡  C 34
    res<-list(
      center=center,
      hull.center=hull.center,
      hull.bag=hull.bag,
      hull.loop=hull.loop,
      pxy.bag=pxy.bag,
      pxy.outer=if(length(pxy.outer)>0) pxy.outer else NULL,
      pxy.outlier=if(length(pxy.outlier)>0) pxy.outlier else NULL,
      hdepths=hdepth,
      is.one.dim=is.one.dim,
      prdata=prdata,
      ## random.seed=random.seed,  #SEED
      xy=xy,xydata=xydata
    )
    if(verbose) res<-c(res,list(exp.dk=exp.dk,exp.dk.l=exp.dk.l,hdepth=hdepth))
    class(res)<-"bagplot"
    return(res)

```

4.3 Initilization of bagplot

Points with identical coordinates may run in numerical problem. Therefore, some noise may be added to the data – for this feature the comment signs have to be deleted.

To do not disturb simulation studies in very large sets we replace the use of random samples and use systematically drawn samples now.

```

36  <init 36> ≡  C 34
    # define some functions
    <define function win 37>
    <define function out.of.polygon 38>
    <define function cut.z.pg 40>

```

```

<define function find.cut.z.pg 41>
<define function hdepth.of.points 43>
<define function expand.hull 46>
<define function cut.p.sl.p.sl 42>
<define function pos.to.pg 51>
<define find.polygon.center 65>
# check input
xydata<-if(missing(y)) x else cbind(x,y)
if(is.data.frame(xydata)) xydata<-as.matrix(xydata)
if(any(is.na(xydata))){
  if(na.rm){ xydata<-xydata[!apply(is.na(xydata),1,any),,drop=FALSE]
    print("Warning: NA elements have been removed!!")
  }else{ #121129
    xy.medians<-apply(xydata,2,function(x) median(x, na.rm=TRUE))
    # colMeans(xydata,na.rm=TRUE)
    for(j in 1:ncol(xydata)) xydata[is.na(xydata[,j]),j]<-xy.medians[j]
    print("Warning: NA elements have been exchanged by median values!!")
  }
}
# if(nrow(xydata)<3) {print("not enough data points"); return()} ## 121008
if(length(xydata)<4) {print("not enough data points"); return()}
if((length(xydata)%2)==1) {print("number of values isn't even"); return()}
if(!is.matrix(xydata)) xydata<-matrix(xydata,ncol=2,byrow=TRUE)
# select sample in case of a very large data set
very.large.data.set<-nrow(xydata) > approx.limit
# use of random number generator may disturb simulation
# therefore we now use a systematical part of the data 20120930
### OLD: set.seed(random.seed<-13) ### SEED
if(very.large.data.set){
  ## OLD: ind<-sample(seq(nrow(xydata)),size=approx.limit)
  step<-(n<-nrow(xydata))/approx.limit; ind <- round(seq(1,n,by=step))
  xy<-xydata[ind,]
} else xy<-xydata
n<-nrow(xy)
points.in.bag<-floor(n/2)
# if jittering is needed
# the following two lines can be activated
#xy<-xy+cbind(rnorm(n,0,.0001*sd(xy[,1])),
#             rnorm(n,0,.0001*sd(xy[,2])))
if(verbose) cat("end of initialization")

```

Yuankun Shi asked how the proportion of data points of the bag could be changed. Although this may result in misinterpretations we show a way to implement a modified bagplot:

```

# make copy of compute.bagplot and change halve number of points in the
bag,
# for example by "n*myproportion":
mycompute.bagplot<-
  eval(parse(text=sub("n/2","n*myproportion",deparse(compute.bagplot))))
# define your own bagplot version which calls "mycompute.bagplot":
mybagplot<-eval(parse(text=
  sub("compute.bagplot","mycompute.bagplot",deparse(bagplot))))
# example application:
myproportion<-0.2; set.seed(13); mybagplot(cbind(rnorm(100),rnorm(100)))

```

4.4 Some local functions to find intersection points

win: After a lot of experiments the function `atan2` is found to compute the gradient in fastest way.

```
37 <define function win 37> ≡  C 36, 44, 81
    win<-function(dx,dy){  atan2(y=dy,x=dx) }
```

out.of.polygon: The function `out.of.polygon` checks if the points of `xy` are within the polygon `pg` (return value FALSE) or not (return value TRUE).

```
38 <define function out.of.polygon 38> ≡  C 36
    out.of.polygon<-function(xy,pg){  # 121026
      xy<-matrix(xy,ncol=2)
      # check trivial case
      if(nrow(pg)==1)  return(xy[,1]==pg[1] & xy[,2]==pg[2])
      # store number of points of xy and polygon
      m<-nrow(xy); n<-nrow(pg)
      # find small value relative to polygon
      limit <- -abs(1E-10*diff(range(pg)))
      # find vectors that are orthogonal to segments of polygon
      pgn<-cbind(diff(c(pg[,2],pg[1,2])), -diff(c(pg[,1],pg[1,1])))
      # find center of gravity of xy
      S<-colMeans(xy)
      # compute negative distances of polygon to center of gravity of xy
      dxy<-cbind(S[1]-pg[,1],S[2]-pg[,2])
      # unused: S.in.pg<-all(limit<apply(dxy*pgn,1,sum))
      if( !all( limit < apply(dxy*pgn,1,sum) ) ){
        pg<-pg[n:1,]; pgn<--pgn[n:1,]
      }
      # initialize result
      in.pg<-rep(TRUE,m)
      for(j in 1:n){
        dxy<-xy-matrix(pg[j,],m,2,byrow=TRUE)
        in.pg<-in.pg & limit<(dxy%*%pgn[j,])
      }
      return(!in.pg)
    }
```

Here is a chunk with some statements for checking the function and maybe as a starting point of the next version.

```
39 <chunk for checking function out.of.polygon 39> ≡
    out.of.polygon<-function(xy,pg){
      xy<-matrix(xy,ncol=2)
      # check trivial case
      if(nrow(pg)==1)  return(xy[,1]==pg[1] & xy[,2]==pg[2])
      # store number of points of xy and polygon
      m<-nrow(xy); n<-nrow(pg)
      # find small value relative to polygon
      limit<--abs(1E-10*diff(range(pg)))
      # find vectors that are orthogonal to segments of polygon
      pgn<-cbind(diff(c(pg[,2],pg[1,2])), -diff(c(pg[,1],pg[1,1])))
      segments(hx<-(pg[-1,1]+pg[-n,1])/2,hy<-(pg[-1,2]+pg[-n,2])/2,
        hx+pgn[-n,1],hy+pgn[-n,2],col="blue")
      # find center of gravity of xy
      S<-matrix(colMeans(xy),1,2)
      # compute negative distances of polygon to center of gravity of xy
      dxy<-cbind(S[1]-pg[,1],S[2]-pg[,2])
```

```

      S.in.pg<-                                     # unused
      all(limit<apply(dxy*pgn,1,sum)) # unused
if(!all(limit<apply(dxy*pgn,1,sum))) {
  cat("INIF")
  pg<-pg[n:1,]; pgn<--pgn[n:1,]
}
# initialize result
in.pg<-rep(TRUE,m)
for(j in 1:n){
  dxy<-xy-matrix(pg[j,],m,2,byrow=TRUE)
  in.pg<-in.pg & limit<(dxy%*%pgn[j,])
}
return(!in.pg)
}
# if PG, XY are available:
plot(PG,type="l",xlim=c(-.5,1),ylim=c(-.5,1)); points(XY)
out.of.polygon(XY[1:6,],PG)
points(XY[1:6,],col="red"); points(PG*1.5,col="red",pch=letters)

```

This version of `out.of.polygon` is based on the following algorithm:

1. compute the orthogonal vectors of the sides of the polygon pointing to the interior
2. compute the vectors which starts in the corners of the polygon and ends in a point to be tested
3. check if all the angles between the pairs of associated vectors lie between $-\pi/2$ and $\pi/2$ which is equivalent to get positive signs of the inner products of the associated vectors only.

For more than 2000 test points the new version is 100 times faster than the old one.

cut.z.pg: `cut.z.pg` finds the intersection points of lines defined by `plx,ply,p2x,p2y` and lines that contains `zx,zy` and origin. Data set `cars[6:10,]` induced `zx` values of 0. Therefore, the set of special case had to be expanded.

```

40 <define function cut.z.pg 40> ≡ C 36, 81
cut.z.pg<-function(zx,zy,plx,ply,p2x,p2y){
  a2<-(p2y-ply)/(p2x-plx); a1<-zy/zx
  sx<-(ply-a2*plx)/(a1-a2); sy<-a1*sx
  sxy<-cbind(sx,sy)
  h<-any(is.nan(sxy))||any(is.na(sxy))||any(Inf==abs(sxy))
  if(h){ # print("NAN found"); print(cbind(a1,a2,zx,zy,sxy,p2x-plx))
  if(!exists("verbose")) verbose<-FALSE
    if(verbose) cat("special")
    # zx is zero # 121030
    h<-0==zx
    sx<-ifelse(h,zx,sx); sy<-ifelse(h,ply-a2*plx,sy)
    # points on line defined by line segment
    a1 <- ifelse( abs(a1) == Inf, sign(a1)*123456789*1E10, a1) # 121030
    a2 <- ifelse( abs(a2) == Inf, sign(a2)*123456789*1E10, a2)
    # points on line defined by line segment
    h<-0==(a1-a2) & sign(zx)==sign(plx)
    sx<-ifelse(h,plx,sx); sy<-ifelse(h,ply,sy)
    h<-0==(a1-a2) & sign(zx)!=sign(plx)
    sx<-ifelse(h,p2x,sx); sy<-ifelse(h,p2y,sy)
    # line segment vertical
    # & center NOT ON line segment
    h<-plx==p2x & zx!=plx & plx!=0
    sx<-ifelse(h,plx,sx); sy<-ifelse(h,zy*plx/zx,sy)
    # & center ON line segment

```

```

h<-plx==p2x & zx!=plx & plx==0
  sx<-ifelse(h,plx,sx); sy<-ifelse(h,0,sy)
#   & center NOT ON line segment & point on line      # 121126
h<-plx==p2x & zx==plx & plx!=0 # & sign(zy)==sign(ply)
  sx<-ifelse(h,zx,sx); sy<-ifelse(h,zy,sy)
#   & center ON line segment & point on line
h<-plx==p2x & zx==plx & plx==0 & sign(zy)==sign(ply)
  sx<-ifelse(h,plx,sx); sy<-ifelse(h,ply,sy)
h<-plx==p2x & zx==plx & plx==0 & sign(zy)!=sign(ply)
  sx<-ifelse(h,plx,sx); sy<-ifelse(h,p2y,sy)
#   points identical to end points of line segment
h<-zx==plx & zy==ply; sx<-ifelse(h,plx,sx); sy<-ifelse(h,ply,sy)
h<-zx==p2x & zy==p2y; sx<-ifelse(h,p2x,sx); sy<-ifelse(h,p2y,sy)
#   point of z is center
h<-zx==0 & zy==0; sx<-ifelse(h,0,sx); sy<-ifelse(h,0,sy)
sxy<-cbind(sx,sy)
} # end of special cases
#if(verbose){ print(rbind(a1,a2));print(cbind(zx,zy,plx,ply,p2x,p2y,sxy))}
if(!exists("debug.plots")) debug.plots<-"no"
if(debug.plots=="all"){
  segments(sxy[,1],sxy[,2],zx,zy,col="red")
  segments(0,0,sxy[,1],sxy[,2],col="green",lty=2) ###!
  points(sxy,col="red")
}
return(sxy)
}

```

find.cut.z.pg: find.cut.z.pg finds the intersection points of the lines defined by z and center center and polygon pg.

```

41 <define function find.cut.z.pg 41> ≡ C 36,81
find.cut.z.pg<-function(z,pg,center=c(0,0),debug.plots="no"){
  if(!is.matrix(z)) z<-rbind(z)
  if(1==nrow(pg)) return(matrix(center,nrow(z),2,TRUE))
  n.pg<-nrow(pg); n.z<-nrow(z)
  z<-cbind(z[,1]-center[1],z[,2]-center[2])
  pgo<-pg; pg<-cbind(pg[,1]-center[1],pg[,2]-center[2])
  if(!exists("debug.plots")) debug.plots<-"no"
  if(debug.plots=="all"){
    plot(rbind(z,pg,0),bty="n"); points(z,pch="p")
    lines(c(pg[,1],pg[1,1]),c(pg[,2],pg[1,2]))}
  # find angles of pg und z
  apg<-win(pg[,1],pg[,2])
  apg[is.nan(apg)]<-0; a<-order(apg); apg<-apg[a]; pg<-pg[a,]
  az<-win(z[,1],z[,2])
  # find line segments
  segm.no<-apply((outer(apg,az,"<")),2,sum)
  segm.no<-ifelse(segm.no==0,n.pg,segm.no)
  next.no<-1+(segm.no %% length(apg))
  # compute cut points
  cuts<-cut.z.pg(z[,1],z[,2],pg[segm.no,1],pg[segm.no,2],
                pg[next.no,1],pg[next.no,2])
  # rescale
  cuts<-cbind(cuts[,1]+center[1],cuts[,2]+center[2])
  return(cuts)
}
## find.cut.z.pg(EX, EX1,center=CE)

```

cut.p.sl.p.sl: cut.p.sl.p.sl finds the intersection point of two lines. Both lines are described by a point and its slope. Remember:

$$y = y_1 + m_1(x - x_1)$$

If both slopes are identical an inf-value will be returned.

```
42 <define function cut.p.sl.p.sl 42> ≡ C 36
cut.p.sl.p.sl<-function(xyl,m1,xy2,m2){
  sx<-(xy2[2]-m2*xy2[1]-xyl[2]+m1*xyl[1])/(m1-m2)
  sy<-xyl[2]-m1*xyl[1]+m1*sx
  if(!is.nan(sy)) return( c(sx,sy) )
  if(abs(m1)==Inf) return( c(xyl[1],xy2[2]+m2*(xyl[1]-xy2[1])) )
  if(abs(m2)==Inf) return( c(xy2[1],xyl[2]+m1*(xy2[1]-xyl[1])) )
}
```

4.5 A function to compute the h-depths of data points

hdepth.of.points: hdepth.of.points computes the h-depths of test points tp. local variable ident stores the number of identical points. Algorithmical aspects of finding the h-depth are later discussed in: *<find kkk-hull: pg 68>* The test points are delivered by argument tp. Additionally the matrix of data points xy is used.

```
43 <define function hdepth.of.points 43> ≡ C 36
hdepth.of.points<-function(tp){
  # 121030 second parameter n has been removed
  # if(!exists("precision")) precision <- 1 # 121203
  # return(find.hdepths.tp(tp, xy, 181*precision)) # 121202
  n.tp<-nrow(tp)
  tphdepth<-rep(0,n.tp); dpi<-2*pi-0.000001
  for(j in 1:n.tp) {
    dx<-tp[j,1]-xy[,1]; dy<-tp[j,2]-xy[,2]
    a<-win(dx,dy)+pi; h<-a<10; a<-a[h]; ident<-sum(!h)
    init<-sum(a < pi); a.shift<-(a+pi) %% dpi
    minusplus<-c(rep(-1,length(a)),rep(1,length(a))) ## 070824
    h<-cumsum(minusplus[order(c(a,a.shift))])
    tphdepth[j]<-init+min(h)+1 # +1 because of the point itself!!
    # tphdepth[j]<-init+min(h)+ident; cat("SUMME",ident)
  }
  tphdepth
}
```

4.5.1 A user function to compute h-depths

It may be helpful to have a function for computing the h-depths of test points relative to a data set.

```
44 <define function hdepth() 44> ≡ C 32, 45
hdepth<-function(xy,data){
  # function to compute the h-depths of points
  <define function win 37>
  if(missing(data)) data <- xy
  tp <- xy; xy <- data
  n.tp<-nrow(tp); n <- length(xy[,1])
```

```

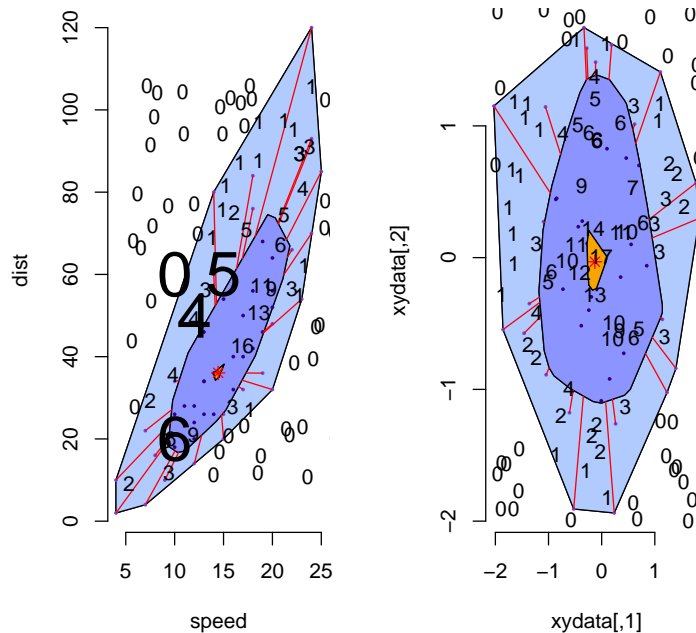
tphdepth<-rep(0,n.tp); dpi<-2*pi-0.000001
for(j in 1:n.tp) {
  # compute difference of coordinates of tp j and data
  dx<-tp[j,1]-xy[,1]; dy<-tp[j,2]-xy[,2]
  # remove data points that are identical to tp j
  h <- tp[j,1] != xy[,1] & tp[j,2] != xy[,2]
  dx <- dx[h]; dy <- dy[h]; n <- length(dx)
  minusplus<-c(rep(-1,n),rep(1,n)) ## 070824
  # compute angles of slopes of lines through tp j and data
  a<-win(dx,dy)+pi; h<-a<10; a<-a[h]; ident<-sum(!h)
  # count number of angles that are lower than pi == points above tp j
  init<-sum(a < pi); a.shift<-(a+pi) %% dpi
  # count points relative to the tp j in halve planes
  h<-cumsum(minusplus[order(c(a,a.shift))])
  # find minimum number of points in a halve plane
  tphdepth[j]<-init+min(h)+1 # +1 because of the point itself!!
  # tphdepth[j]<-init+min(h)+ident; cat("SUMME",ident)
}
tphdepth
}
hdepth<-find.hdepths.tp #121202

```

```

45  <check of function hdepth() 45> ≡
    <define bagplot 32>
    <define function hdepth() 44>
    par(mfrow=c(1,2))
    TP <- rbind(c(10,20),c(15,60),c(10,60),c(12,50)); data <- cars
    bagplot(data)
    #text(TP, as.character(hdepth(TP,data)),cex=3)
    set.seed(13); tp <- cbind(runif(100,5,35),runif(100,9,110))
    text(tp, as.character(hdepth(tp,data)),cex=1)
    set.seed(13); n<-40; data <- cbind(rnorm(n), rnorm(n))
    tp <- cbind(runif(100,-2,2),runif(100,-2,2))
    bagplot(data)
    text(tp, as.character(hdepth(tp,data)))

```

4.6 A function to expand the hull

expand.hull: `expand.hull` expands polygon `pk` without changing the depth of its points. `k` is the depth and `resolution` is the number of points to be checked during expansion.

```
46 <define function expand.hull 46> ≡ C 36
  expand.hull<-function(pg,k){
    if( 1 >= nrow(pg) ) return(pg) ## 121026 ## 121123 <= statt ==
    <find end points of line segments: center → pg → pg0 47>
    <search for points with critical hdepth 48>
    <find additional points between the line segments 49>
    <compute hull pg.new 50>
  }
```

At first we search the intersection points of the hull of the data set with the lines beginning in the center and running through the points of `pg`. Then test points on the segments defined by these intersection points and the points of `pg` will be generated by using a vector `lam`.

```
47 <find end points of line segments: center → pg → pg0 47> ≡ C 46
  resolution<-floor( 20*precision)
  pg0<-xy[hdepth==1,]
  pg0<-pg0[chull(pg0[,1],pg0[,2]),]
  end.points<-find.cut.z.pg(pg,pg0,center=center,debug.plots=debug.plots)
  lam<-((0:resolution)^1)/resolution^1
```

The test is performed in two stages. In the interval from start point to end point `resolution` test points are tested concerning their h-depth. The critical interval is divided again to find a better limit.

```
48 <search for points with critical hdepth 48> ≡ C 46
  pg.new<-pg
  for(i in 1:nrow(pg)){
    tp<-cbind(pg[i,1]+lam*(end.points[i,1]-pg[i,1]),
```

```

      pg[i,2]+lam*(end.points[i,2]-pg[i,2]))
# hd.tp<-hdepth.of.points(tp)
hd.tp<-find.hdepths.tp(tp,xy)
ind<-max(sum(hd.tp>=k),1)
if(ind<length(hd.tp)){ # hd.tp[ind]>k &&
  tp<-cbind(tp[ind,1]+lam*(tp[ind+1,1]-tp[ind,1]),
            tp[ind,2]+lam*(tp[ind+1,2]-tp[ind,2]))
  # hd.tp<-hdepth.of.points(tp)
  hp.tp<-find.hdepths.tp(tp,xy)
  ind<-max(sum(hd.tp>=k),1)
}
pg.new[i,]<-tp[ind,]
}
pg.new<-pg.new[chull(pg.new[,1],pg.new[,2]),]
# cat("depth pg.new", hdepth.of.points(pg.new))
# cat("depth pg.new", find.hdepths.tp(pg.new,xy))

```

Between the spurts we interpolated additional directions and find additional limits by the same procedure.

```

49 <find additional points between the line segments 49> ≡ C 46
pg.add<-0.5*(pg.new+rbind(pg.new[,1],pg.new[,2]))
# end.points<-find.cut.z.pg(pg,pg0,center=center)
end.points<-find.cut.z.pg(pg.add,pg0,center=center) ## 070824
for(i in 1:nrow(pg.add)){
  tp<-cbind(pg.add[i,1]+lam*(end.points[i,1]-pg.add[i,1]),
            pg.add[i,2]+lam*(end.points[i,2]-pg.add[i,2]))
  # hd.tp<-hdepth.of.points(tp)
  hd.tp<-find.hdepths.tp(tp,xy)
  ind<-max(sum(hd.tp>=k),1)
  if(ind<length(hd.tp)){ # hd.tp[ind]>k &&
    tp<-cbind(tp[ind,1]+lam*(tp[ind+1,1]-tp[ind,1]),
              tp[ind,2]+lam*(tp[ind+1,2]-tp[ind,2]))
    # hd.tp<-hdepth.of.points(tp)
    hd.tp<-find.hdepths.tp(tp,xy)
    ind<-max(sum(hd.tp>=k),1)
  }
  pg.add[i,]<-tp[ind,]
}
# cat("depth pg.add", hdepth.of.points(pg.add))

```

Finally the hull of the limits is computed and our numerical solution of $\text{hull}(d_k)$. `pg.new` is the output of `expand.hull`.

```

50 <compute hull pg.new 50> ≡ C 46
pg.new<-rbind(pg.new,pg.add)
pg.new<-pg.new[chull(pg.new[,1],pg.new[,2]),]

```

4.7 A function to find the position of points respectively to a polygon

pos.to.pg: `pos.to.pg` finds the position of points `z` relative to a polygon `pg`. If a point is below the polygon "lower" is returned otherwise "upper".

```

51 <define function pos.to.pg 51> ≡ C 36
pos.to.pg<-function(z,pg,reverse=FALSE){
  if(reverse){
    int.no<-apply(outer(pg[,1],z[,1],">="),2,sum)
    zy.on.pg<-pg[int.no,2]+pg[int.no,3]*(z[,1]-pg[int.no,1])
  }
}

```

```

    }else{
      int.no<-apply(outer(pg[,1],z[,1], "<="), 2, sum)
      zy.on.pg<-pg[int.no,2]+pg[int.no,3]*(z[,1]-pg[int.no,1])
    }
    ### ifelse(z[,2]<zy.on.pg, "lower", "higher") ### 121004
    result <- ifelse(z[,2]<zy.on.pg, "lower", "higher") ###
    return(result)

    if( all(result=="lower") ){
      result <- ifelse(((z[,2] - zy.on.pg)/max(z[,2] - zy.on.pg)+1e-10) < 0,
        "lower", "higher")
    }
    if( all(result=="higher") ){
      result <- ifelse(((z[,2] - zy.on.pg)/max(z[,2] - zy.on.pg)-1e-10) < 0,
        "lower", "higher")
    }
    print(result)
    return(result)
  }
}

```

4.8 Check if data set is one dimensional

Now the local functions are ready for usage. To detect a data set being one dimensional we apply `prcomp()`. In the one dimensional case we construct a boxplot by hand.

```

52  <check and handle linear case 52> ≡  C 34
    prdata<-prcomp(xydata)
    is.one.dim<-(0 == max(prdata[[1]])) || (min(prdata[[1]])/max(prdata[[1]]))<0.00001 # 121129
    if(is.one.dim){
      if(verbose) cat("data set one dimensional")
      center<-colMeans(xydata)
      res<-list(xy=xy, xydata=xydata, prdata=prdata,
        is.one.dim=is.one.dim, center=center)
      class(res)<- "bagplot"
      return(res)
    }
    if(verbose) cat("data not linear")

53  <handle three or four data points 53> ≡  C 34
    if(nrow(xydata)<=4) {
      if(verbose) cat("only three or four data points")
      center<-colMeans(xydata)
      res<-list(xy=xy, xydata=xydata, prdata=prdata, hdepths=rep(1,n), hdepth=rep(1,n),
        is.one.dim=is.one.dim, center=center, hull.center=NULL,
        hull.bag=NULL, hull.loop=NULL, pxy.bag=NULL, pxy.outer=xydata,
        pxy.outlier=NULL, exp.dk=xydata)
      class(res)<- "bagplot"
      return(res)
    }
  }

```

4.9 Standardize data and compute h-depths of points

For numerical reasons we standardize the data set: `xyxy`. Some computations takes place on the standardized copy `xyxy` of `xy`.

```
54 <standardize data and compute: xyxy, xym, xysd 54> ≡ C 34
    xym<-apply(xy,2,mean); xysd<-apply(xy,2,sd)
    xyxy<-cbind((xy[,1]-xym[1])/xysd[1],(xy[,2]-xym[2])/xysd[2])
```

For each data point we compute the directions to all the points and determine the angles of the directions. This information helps us to find the h-depths of the points. For friends of complexity: the angles between all pair of points are computed in $O(n^2 \log n)$ time because of sorting the columns of a $(n \times n)$ -matrix. The angle between identical points is coded by entry 1000.

```
55 <compute angles between points 55> ≡ C 34
    dx<-(outer(xy[,1],xy[,1],"-"))
    dy<-(outer(xy[,2],xy[,2],"-"))
    alpha<-atan2(y=dy,x=dx); diag(alpha)<-1000
    for(j in 1:n) alpha[,j]<-sort(alpha[,j])
    alpha<-alpha[-n,] ; m<-n-1
    ## quick look inside, just for check
    if(debug.plots=="all"){
      plot(xy,bty="n"); xdelta<-abs(diff(range(xy[,1]))); dx<-xdelta*.3
      for(j in 1:n) {
        p<-xy[j,]; dy<-dx*tan(alpha[,j])
        segments(p[1]-dx,p[2]-dy,p[1]+dx,p[2]+dy,col=j)
        text(p[1]-xdelta*.02,p[2],j,col=j)
      }
    }
    if(verbose) print("end of computation of angles")
```

We compute the h-depths in $O(n^2 \log(n))$. The NaN angles are extracted because they indicate points with identical coordinates. For each point we find the h-depth by the following algorithm: At first we count the number of angles of the actual point within interval $[0, \pi)$. This is equivalent to the number of points above the actual point. Then we rotate the $y = 0$ -line counterclockwise and increment the initial counter if an additional point emerges and we decrement the counter if a point / angle leaves the halve plain.

The median is defined as the gravity center of all points with maximal h-depth.

As usually some problems were induced by equality of angles. One reaction was to add some shift to compare with slightly modified π -values.

```
56 <compute hdepths 56> ≡ C 34
    hdepth<-rep(0,n); dpi<-2*pi-0.000001; mypi<-pi-0.000001
    minusplus<-c(rep(-1,m),rep(1,m))
    if(FALSE){
      for(j in 1:n) {
        a<-alpha[,j]+pi; h<-a<10; a<-a[h]; init<-sum(a < mypi) # hallo
        a.shift<-(a+pi) %% dpi
        minusplus<-c(rep(-1,length(a)),rep(1,length(a))) ## 070824
        h<-cumsum(minusplus[order(c(a,a.shift))])
        hdepth[j]<-init+min(h)+1 # or do we have to count identical points?
        # hdepth[j]<-init+min(h)+sum(xy[j,1]==xy[,1] & xy[j,2]==xy[,2])
      }
    }
    <define find.hdepths 57>
    hdepth <- find.hdepths(xy,181*precision)
    if(verbose){print("end of computation of hdepth:"); print(hdepth)}
    ## quick look inside, just for a check
    if(debug.plots=="all"){
      plot(xy,bty="n")
```

```

xdelta<-abs(diff(range(xy[,1]))); dx<-xdelta*.1
for(j in 1:n) {
  a<-alpha[,j]+pi; a<-a[a<10]; init<-sum(a < pi)
  a.shift<-(a+pi) %% dpi
  minusplus<-c(rep(-1,length(a)),rep(1,length(a))) ## 070824
  h<-cumsum(minusplus[ao<-(order(c(a,a.shift)))])
  no<-which((init+min(h)) == (init+h))[1]
  p<-xy[,j]; dy<-dx*tan(alpha[,j])
  segments(p[1]-dx,p[2]-dy,p[1]+dx,p[2]+dy,col=j,lty=3)
  dy<-dx*tan(c(sort(a),sort(a))[no])
  segments(p[1]-5*dx,p[2]-5*dy,p[1]+5*dx,p[2]+5*dy,col="black")
  text(p[1]-xdelta*.02,p[2],hdepth[j],col=1) # cex=2.5 assumes suitable fonts
}
}

```

Because of numerical problem we introduce a alternative way to compute the H-depths by function `find.hdepths`.

```

57 <define find.hdepths 57> ≡ C 32,56
  find.hdepths <- function(xy, number.of.directions=181){ # 121126
    <standardize dimensions 58>
    <loop over directions 59>
    hd
  }

```

```

58 <standardize dimensions 58> ≡ C 57
  xy <- as.matrix(xy)
  for( j in 1:2) {
    xy[,j] <- xy[,j] - min(xy[,j])
    if( 0 < (h <- max(xy[,j]))) xy[,j] <- xy[,j] / max(xy[,j])
  }

```

It is not faster to change the for loop in the following chunk by a vector solution. Maybe the management of the storage costs a lot of time. Besides the loop construction has the advantage of clarity.

```

59 <loop over directions 59> ≡ C 57
  phi <- c(seq(0,180,length=number.of.directions)[-1]*(2*pi/360))
  sinphi <- c(sin(phi),1); cosphi <- c(cos(phi),0)
  RM1 <- round(digits=6,rbind(cosphi,sinphi))
  hd <- rep(h<-length(xy[,1]),h)
  for( j in seq(along=sinphi)){
    xyt <- xy %*% RM1[,j]
    hd <- pmin(hd,rank(xyt,ties.method="min"), rank(-xyt,ties.method="min"))
  }
  # xyt <- xy %*% RM1
  # hd2 <- cbind(apply(xyt, 2, rank, ties.method="min"),
  #             apply(-xyt,2, rank, ties.method="min"))
  # hd2 <- apply(hd2, 1, min)

```

To find the h-depths of some test points `tp` we use the same idea. After some standardizations we compute the projection of the data points and the testpoints onto several linear lines. Then we look at the ranks of the vector found by combining the mappings of `xy` and `tp` and correct the ranks of `tpt` by subtracting its ranks computed without `xyt`. The results shows us a one sided depth information. To find the desired h-depths we have to repeat the process by looking from the other side. This is performed by taking the negative vectors of `xyt` and `tpt`. A p-minimization of all vectors of all the directions finishes the computation.

```

60 <define find.hdepths.tp 60> ≡ C 32

```

```

find.hdepths.tp <- function(tp, data, number.of.directions=181){ # 121130
  ## standardize dimensions ##
  xy <- as.matrix(data); tp <- as.matrix(rbind(tp)); n.tp <- dim(tp)[1]
  for( j in 1:2) {
    xy[,j] <- xy[,j] - (h <- min(xy[,j], na.rm=TRUE))
    tp[,j] <- tp[,j] - h
    if( 0 < (h <- max(xy[,j], na.rm=TRUE))) {
      xy[,j] <- xy[,j]/h; tp[,j] <- tp[,j]/h
    }
  }
  ##loop over directions##
  phi <- c(seq(0,180,length=number.of.directions)[-1]*(2*pi/360))
  sinphi <- c(sin(phi),1); cosphi <- c(cos(phi),0)
  RM1 <- round(digits=6,rbind(cosphi,sinphi))
  hntp <- rep(length(xy[,1]),length(tp[,1]))
  for( j in seq(along=sinphi)){ #print(j)
    xyt <- xy %*% RM1[,j]; tpt <- (tp %*% RM1[,j])[]
    xyt <- xyt[!is.na(xyt)] #; tpt <- sort(tpt)
    hntp <- pmin(hntp,(rank( c(tpt,xyt), ties.method="min"))[1:n.tp]
                  -rank( tpt,ties.method="min")
                  ,rank(-c(tpt,xyt), ties.method="min")[1:n.tp]
                  -rank(-tpt,ties.method="min")
                )
  }
  hntp
}

```

We determine the h-depth k so that the following condition holds:

(the number of points of h-depth greater or equal k is lower or equal to the number of data points staying in the bag) and (the number of points of h-depth greater equal $k - 1$ is greater $n/2$):

$$\#D_{k-1} > \text{points in the bag}; n/2 \geq \#D_k.$$

We assume that in the bag are $\text{floor}(n/2)$ points.

In a bagplot we have usually to interpolated between two regions of depths. But, what should we do if there are no points in the inner region? a) interpolate with the center as a symbolic in polygon, b) choose the outer polygon In boxplots we can get more than $\text{floor}(n/2)$ points in the box (including the edge) if there are ties in the region of the quartiles: Ties will be ignored. Therefore, in the strange case that D_k with maximal k is greater than $n/2$ the condition cannot be fulfilled and we take the maximal k to define the bag. $k.1$ is the row of $d.k$ with depth $d.k[k.1, 2]$ that contains $d.k[k.1, 2]$ points that are more than the bag should contain. The polygon with depth $k=k.1+1$ will contain points.in.bag or fewer points.

```

61 <find k 61> ≡ C 34
hd.table<-table(sort(hdepth))
d.k<-cbind(dk=rev(cumsum(rev(hd.table))),
           k =as.numeric(names(hd.table)))
k.1<-sum( points.in.bag < d.k[,1] )
# if(nrow(d.k)>1){ # version 09/2005, error in data set 1 of Meuleman
# instead of >2 now >k.1 # 070827
# if(nrow(d.k)>k.1){ k<-d.k[k.1+1,2] } else { k<-d.k[k.1,2] }
# this statement will not have an effect because of the next one:
k<-d.k[k.1,2]+1 # 121004 increment depth by one not by looking for next depth
if(verbose){cat("numbers of members of dk:"); print(hd.table); print(d.k)}
if(verbose){cat("end of computation of k, k=",k,"k.1:",k.1)}
# D.K<-d.k; K.1<-k.1; EX<-exp.dk; EX.1<-exp.dk.1; PDK<-pdk; HDEPTH<-hdepth

```

4.10 Find the center of the data set

The two dimensional median is the center of gravity of the polygon of the points (not data points) with maximal h-depths.

We check some inner test points to find the maximal h-depth. Then we look for the boundary of the area of points with this h-depth.

This procedure has been tested with the Ben Greiner data using: *(test: data set of Ben Greiner 14)*

```
62 <compute hdepths of test points to find center 62> ≡ C 34
center<-apply(xy[which(hdepth==max(hdepth)),,drop=FALSE],2,mean)
hull.center<-NULL
if(3<nrow(xy)&&length(hd.table)>0){
  n.p<-floor(1.5*c(32,16,8)[1+(n>50)+(n>200)]*precision)
  # limit.hdepth.to.check <- sort(hdepth, decreasing = TRUE)[min(nrow(xy),6)]
  # 121126
  h <- unique(sort(hdepth, decreasing = TRUE))
  limit.hdepth.to.check <- sort(h)[min(length(h),3)]
  h<-cands<-xy[limit.hdepth.to.check <= hdepth,,drop=FALSE]
  # h<-cands<-xy[rev(order(hdepth))[1:(min(nrow(xy),6))],,]
  cand<-cands[chull(cands[,1],cands[,2]),,]; n.c<-nrow(cands)
  if(is.null(n.c))cands<-h
  <check some points to find the maximal h-depth 63>
  # if max of testpoint is smaller than max depth of points take that max!
  if(verbose){ cat("depth of testpoints"); print(summary(tphdepth)) } # 121126
  tphdepth<-max(tphdepth,d.k[,2]) # 121004
  <find the polygon of points of maximal h-depth 64>
  if(verbose){cat("hull.center",hull.center); print(table(tphdepth)) }
}
# if(verbose) cat("center depth:",hdepth.of.points(rbind(center))-1)
if(verbose) cat("center depth:",find.hdepths.tp(rbind(center),xy)-1)
if(verbose){print("end of computation of center"); print(center)}
```

We check points of a rectangular field and compute their depths. If these points should be plotted for debugging issues remove the comment sign of the last line of the following chunk and activate the statement `points(TP,pch = letters[TPD])` after `bagplot()` has finished.

```
63 <check some points to find the maximal h-depth 63> ≡ C 62
xyextr<-rbind(apply(cands,2,min),apply(cands,2,max))
## xydel<-2*(xyextr[2,]-xyextr[1,])/n.p # unused
if( (xyextr[2,1]-xyextr[1,1]) < 0.2*(h <- diff(range(xy[,1])))){
  xyextr[1:2,1] <- mean(xyextr[,1]) + c(-.1,.1) * h } ## 121203
if( (xyextr[2,2]-xyextr[1,2]) < 0.2*(h <- diff(range(xy[,2])))){
  xyextr[1:2,2] <- mean(xyextr[,2]) + c(-.1,.1) * h } ## 121203
if(verbose){cat("xyextr: looking for maximal depth"); print(xyextr) }
h1<-seq(xyextr[1,1],xyextr[2,1],length=n.p)
h2<-seq(xyextr[1,2],xyextr[2,2],length=n.p)
tp<-cbind(as.vector(matrix(h1,n.p,n.p)), # [1:n.p^2],
          as.vector(matrix(h2,n.p,n.p,TRUE))) # [1:n.p^2])
# tphdepth<-max(hdepth.of.points(tp))-1
tphdepth<-max(find.hdepths.tp(tp,xy))
# if(verbose) { TP<-tp; TPD<-find.hdepths.tp(tp,xy) }
if(verbose) cat("points(TP,pch=c(letters,LETTERS)[TPD+1])")
```

For finding the area of maximal h-depth we use an algorithm that has been implemented for finding the bag, see below. *(find kkk-hull: pg 68)* Due to computational problems it may be happen that there are NAs in `hull.center`. Then will take the center computed before as center polygon.

```
64 <find the polygon of points of maximal h-depth 64> ≡ C 62
<initialize angles, ang 72>
```

```

kkk<-tphdepth
if(verbose){cat("max-hdepth found:"); print(kkk)}
if(verbose) cat("find polygon with max depth")
<find kkk-hull: pg 68>
hull.center<-cbind(pg[,1]*xysd[1]+xym[1],pg[,2]*xysd[2]+xym[2])
if(!any(is.na(hull.center))) center<-find.polygon.center(hull.center) else
      hull.center <- rbind(center)          # 121126
if(verbose){ cat("CENTER"); print(center) }

```

A function to compute the center of gravity of a polygon. The function `find.polygon.center` determines the center of gravity of a polygon.

```

65 <define find.polygon.center 65> ≡ C 36
find.polygon.center<-function(xy){
  ## if(missing(xy)){n<-50;x<-rnorm(n);y<-rnorm(n); xy<-cbind(x,y)}
  ## xy<-xy[chull(xy),]
  if(length(xy)==2) return(xy[1:2])
  if(nrow(xy)==2) return(colMeans(xy)) ## 121009
  ## partition polygon into triangles
  n<-length(xy[,1]); mxy<-colMeans(xy)
  xy2<-rbind(xy[-1,],xy[1,]); xy3<-cbind(rep(mxy[1],n),mxy[2])
  ## determine areas and centers of gravity of triangles
  S<-(xy+xy2+xy3)/3
  F2<-abs((xy[,1]-xy3[,1])*(xy2[,2]-xy3[,2])-
          (xy[,2]-xy3[,2])*(xy2[,1]-xy3[,1]))
  ## compute center of gravity of polygon
  lambda<-F2/sum(F2)
  SP<-colSums(cbind(S[,1]*lambda,S[,2]*lambda))
  return(SP)
}

```

We compute the convex hull of D_k : polygon `pdk` and the hull of D_{k-1} : polygon `pdk.1`. `pdk` represents inner polygon and `pdk.1` outer one.

Then polygon `pdk` and `pdk.1` are enlarged without changing its h-depth: `exp.dk`, `exp.dk.1`-

```

66 <method one: find hulls of  $D_k$  and  $D_{k-1}$  66> ≡ C 34
# inner hull of bag
xyi<-xy[hdepth>=k,,drop=FALSE] # cat("dim XYI", dim(xyi))
# 121028 some corrections for strange k situations
if(0 < length(xyi)) pdk<-xyi[chull(xyi[,1],xyi[,2]),,drop=FALSE]
# outer hull of bag
if( k > 1 ){
  xyo<-xy[hdepth>=(k-1),,drop=FALSE]
  pdk.1<-xyo[chull(xyo[,1],xyo[,2]),,drop=FALSE]
} else pdk.1 <- pdk
if(0 == length(xyi)) pdk <- pdk.1
if(verbose)cat("hull computed: pdk, pdk.1:")
if(verbose){print(pdk); print(pdk.1) }
if(debug.plots=="all"){
  plot(xy,bty="n")
  h<-rbind(pdk,pdk[1,]); lines(h,col="red",lty=2)
  h<-rbind(pdk.1,pdk.1[1,]);lines(h,col="blue",lty=3)
  points(center[1],center[2],pch=8,col="red")
}
exp.dk<-expand.hull(pdk,k)
exp.dk.1<-expand.hull(exp.dk,k-1) # pdk.1,k-1,20)

```

The new approach to find the hull works as follows:

For a given k we move lines with different slopes from outside of the cloud to the center and stop if k points are crossed. To keep things simple we rotate the data points so that we have only move a vertical line.

1. define directions / angles for hdepth search
2. standardize data set to get appropriate directions
3. computation of D_k polygon and restandardization
4. computation of D_{k-1} polygon and restandardization

We determine the hulls on the base of the standardized data `xyxy`.

```
67 <method two: find hulls of  $D_k$  and  $D_{k-1}$  67> ≡ C 34
  <initialize angles, ang 72>
  # standardization of data set xyxy is used
  kkk<-k
  if(verbose) print("find polygon with depth something higher than that of the bag")
  if( kkk <= max(d.k[,2]) ){ # inner one # 121030
    <find kkk-hull: pg 68>
    exp.dk<-cbind(pg[,1]*xysd[1]+xym[1],pg[,2]*xysd[2]+xym[2])
  } else {
    exp.dk <- NULL
  }
  if( 1 < kkk ) kkk<-kkk-1 # outer one
  if(verbose) print("find polygon with depth a little bit lower than that of the bag")
  <find kkk-hull: pg 68>
  exp.dk.1<-cbind(pg[,1]*xysd[1]+xym[1],pg[,2]*xysd[2]+xym[2])
  if(is.null(exp.dk)) exp.dk <- exp.dk.1
  # EX.1 <- exp.dk.1; EX <- exp.dk
  if(verbose) print("End of find hulls, method two")
```

The polygon for h-depth kkk is constructed in a loop. In each step we consider one direction / angle.

```
68 <find kkk-hull: pg 68> ≡ C 64, 67
  <initialize loop of directions 71>
  if(verbose) cat("start of computation of the directions: ", "kkk=", kkk) # 121030
  for(ia in seq(angles)[-1]){
    <body of loop of directions 69>
  }
  # if(verbose) PG <- pg; PGL <- pgl
  if(2<nrow(pg) && 2<nrow(pgl)){
    <combination of lower and upper polygon 73>
  } else {
    if(2<nrow(pgl)){ #121204
      pg <- rbind(pg[2,1:2],pgl[-c(1,length(pgl[,1])),1:2])
    } else {
      pg <- rbind(pg [-c(1,length(pg [,1])),1:2],pgl[2,1:2])
      # rbind(pgl[2,1:2],pg[2,1:2])
    }
  }
  if(verbose) cat("END of computation of the directions")
```

At first we search the limiting points for every direction by rotating the data set and then we determine the quantiles $x_{k/n}$ and $x_{(n+1-k)/n}$. With this points we construct a upper polygon `pg` and a lower one `pgl` that limit the hull we are looking for. To update a polygon we have to find the line segments of the polygon that are cut by the lines of slope a through the limiting points as well as the intersection points.

69

```

<body of loop of directions 69> ≡ C 68
# determine critical points pnw and pnwl of direction a
# if(verbose) cat("ia",ia,angles[ia])
# 121030
a<-angles[ia]; angtan<-ang[ia]; xyt<-xyxy%*%c(cos(a),-sin(a)); xyto<-order(xyt)
ind.k <-xyto[kkk]; ind.kk<-xyto[n+1-kkk]; pnw<-xyxy[ind.k,]; pnwl<-xyxy[ind.kk,]
# if(verbose) if( 1 < sum(xyt == xyt[ind.k]) )print("WARNING: some points identical")
if(debug.plots=="all") points(pnw[1],pnw[2],col="red")
# new limiting lines are defined by pnw / pnwl and slope a
# find segment of polygon that is cut by new limiting line and cut
# if(ia>200) { #<show pg pgl>#; points(pnw[1],pnw[2],col="magenta",cex=6) }

if( abs(angtan)>1e10){ if(verbose) cat("kkk",kkk,"x=c case")
# case of vertical slope #print(pg);print(pnw);print(xyt);lines(pg,col="red",lwd=3)
# number of points left of point pnw that limit the polygon
pg.no<-sum(pg[,1]<pnw[1])
if( 0 < pg.no ){
# the polygon (segment pg.no) has to be cut at x==pnw[1]
cutp <- c(pnw[1], pg [pg.no, 2]+pg [pg.no, 3]*(pnw [1]-pg [pg.no ,1]))
pg<- rbind(pg[1:pg.no,], c(cutp,angtan), c(cutp[1]+dxy, cutp[2] +angtan*dxy,NA))
} else {
if(verbose) cat("!!! case degenerated UPPER polygon: pg.no==0")
# the limiting point pnw is above the beginning of the polygon
# therefore, the polygon reduces to line
pg <- rbind(pg[1,], c(pg[2,1:2],NA))
}
pg.nol<-sum(pgl[,1]>=pnwl[1])
if( 0 < pg.nol ){ ##??2 # 121204
cutpl<-c(pnwl[1],pgl[pg.nol,2]+pgl[pg.nol,3]*(pnwl[1]-pgl[pg.nol,1]))
pgl<-rbind(pgl[1:pg.nol,],c(cutpl,angtan),c(cutpl[1]-dxy, cutpl[2]-angtan*dxy,NA))
} else {
if(verbose) cat("!!! case degenerated LOWER polygon: pgl.no==0")
pgl <- rbind(pgl[1,], c(pgl[2,1:2],NA))
}
}else{ # if(verbose) cat("kkk",kkk,"normal case")
# normal case upper polygon
pg.inter<-pg[,2]-angtan*pg[,1]; pnw.inter<-pnw[2]-angtan*pnw[1]
pg.no<-sum(pg.inter<pnw.inter)
if(is.na(pg[pg.no,3])) pg[pg.no,3] <- -Inf # 121129 NaN/Na error
cutp<-cut.p.sl.p.sl(pnw,ang[ia],pg[pg.no,1:2],pg[pg.no,3])
pg<- rbind(pg[1:pg.no,], c(cutp,angtan), c(cutp[1]+dxy, cutp[2] +angtan*dxy,NA))
# normal case lower polygon
pg.interl<-pgl[,2]-angtan*pgl[,1]; pnw.interl<-pnwl[2]-angtan*pnwl[1]
pg.nol<-sum(pgl.interl>pnw.interl)
if(is.na(pgl[pg.nol,3])) pgl[pg.nol,3] <- Inf # 121129 NaN/Na error
cutpl<-cut.p.sl.p.sl(pnwl,angtan,pgl[pg.nol,1:2],pgl[pg.nol,3])
pgl<-rbind(pgl[1:pg.nol,],c(cutpl,angtan),c(cutpl[1]-dxy, cutpl[2]-angtan*dxy,NA))
}
## if(kkk==KKK && ia == 51) { cat("ENDE: pgl"); print(pgl) }
# update pg, pgl completed
# PG<-pg; PG.NO<-pg.no; CUTP<-cutp; DXY<-dxy; PNEW<-pnw; PGL<-pgl; PG.NOL<-pg.nol
<debug: plot within for loop 74>
##show pg pgl##

```

For debugging it may be helpful to plot the polygons pg and pgl. Also the slope stored in the third column of matrix pg and pgl. The code chunk can be placed at any point of interest.

```

70 <show pg pgl 70> ≡
plot(rbind(pg,pgl)[,-3],xlab="",ylab="",xlim=2*c(-1,1),ylim=3*c(-1,1),type="n")
lines(pg[,-3],col="orange",type="b"); lines(pgl[,-3],col="lightblue",type="b")
nnn <- length(pg[,1]); points(pg[, -3],col="red", pch=as.character(1:nnn),cex=2)
nnn <- length(pgl[,1]); points(pgl[, -3],col="blue",pch=as.character(1:nnn),cex=2)
title(as.character(ia),sub=paste("kkk=",kkk))
segments(pg[,1], pg[,2], pg[,1]+1,pg[,2]+pg[,3],lty=3,col="red",lwd=3)
segments(pgl[,1],pgl[,2],pgl[,1]-1,pgl[,2]-pgl[,3],lty=3,col="blue",lwd=3)

```

To initialize the loop we construct the first polygons (upper one: pg, lower one: pgl) by vertical lines. dxdy is a step that is larger than the range of the standardized data set.

```

71 <initialize loop of directions 71> ≡ C 68
ia<-1; a<-angles[ia]; xyt<-xyxy%%c(cos(a),-sin(a)); xyto<-order(xyt)
# initial for upper part
ind.k<-xyto[kkk]; cutp<-c(xyxy[ind.k,1],-10)
dxy<-diff(range(xyxy))
pg<-rbind(c(cutp[1],-dxy,Inf),c(cutp[1],dxy,NA))
# initial for lower part
ind.kk<-xyto[n+1-kkk]; cutpl<-c(xyxy[ind.kk,1],10)
# pgl<-rbind(c(cutpl[1],dxy,Inf),c(cutpl[1],-dxy,NA))
pgl<-rbind(c(cutpl[1],dxy,-Inf),c(cutpl[1],-dxy,NA))
# the sign of inf doesn't matter
<debug: plot ini 75>

```

It is necessary to initialize the angles of the directions. If the data set is very large we will check fewer directions than in case of a small data set. If the data set is small the choice of the angles may be improved by using the observed angles defined by the slopes of lines running through the pairs of the points.

```

72 <initialize angles, ang 72> ≡ C 64, 67
# define direction for hdepth search
num<-floor(2*c(417,351,171,85,67,43)[sum(n>c(1,50,100,150,200,250))]*precision)
num.h<-floor(num/2); angles<-seq(0,pi,length=num.h)
ang<-tan(pi/2-angles)

```

The combination of the polygons is a little bit complicated because sometimes at the right and at left margin an additional intersection point has to be computed and integrated. l in front of a variable name indicates the left margin whereas the right one is coded by r. Letter l (u) at the end of a name is short for lower and upper.

AJS found an error that was caused by a wrong reduction of pgl. Because of the orientation of pgl which is the other way round we have to copy the last point and not the first one.

```

73 <combination of lower and upper polygon 73> ≡ C 68
## plot(xyxy[,1:2],xlim=c(-.5,+.5),ylim=c(-.5,.5))
## lines(pg,type="b",col="red"); lines(pgl,type="b",col="blue")
## remove first and last points and multiple points #<show pg pgl>#
limit<-1e-10
## pg <-pg [c(TRUE,(abs(diff(pg[,1]))>limit)|(abs(diff(pg[,2]))>limit)),] old#
idx <- c(TRUE,(abs(diff(pg[,1]))>limit)|(abs(diff(pg[,2]))>limit)) # 121008
if(any(idx==FALSE)){
  pg <-pg[idx,]; pg[,3] <- c(diff(pg[,2])/diff(pg[,1]), NA)
}

# old reduction which caused some errors:
## pgl<-pgl[c(TRUE,(abs(diff(pgl[,1]))>limit)|(abs(diff(pgl[,2]))>limit)),] error##
## pgl<-pgl[c(      (abs(diff(pgl[,1]))>limit)|(abs(diff(pgl[,2]))>limit),TRUE),] old#
idx <- c(      (abs(diff(pgl[,1]))>limit)|(abs(diff(pgl[,2]))>limit),TRUE)#121008
if(any(idx==FALSE)){
  pgl<-pgl[idx,]; pgl[,3] <- c(diff(pgl[,2])/diff(pgl[,1]), NA)
}

```

```

}
## add some tolerance in course of numerical problems
pgl[,2]<-pgl[,2] - .00001 ## 121004
## show pg pgl>>
pg<- pg [-nrow(pg ),][-1,,drop=FALSE]
pgl<-pgl[-nrow(pgl),][-1,,drop=FALSE]

# determine position according to the other polygon
# cat("relative position: lower polygon")
indl<-pos.to.pg(round(pgl,digits=10),round(pg,digits=10)) # 121126
# cat("relative position: upper polygon")
indu<-pos.to.pg(round(pg,digits=10),round(pgl,digits=10),TRUE)
sr<-sl<-NULL # ; ##show pg pgl>>

# right region
if(indu[(npg<-nrow(pg))]=="lower" & indl[1]=="higher"){
  # cat("in if of right region: the upper polynom is somewhere lower")
  # checking from the right: last point of lower polygon that is NOT ok
  rnuml<-which(indl=="lower")[1]-1
  # checking from the left: last point of upper polygon that is ok
  rnumu<-npg+1-which(rev(indu=="higher"))[1]
  # special case all points of lower polygon are upper
  if(is.na(rnuml)) rnuml<-sum(pg[rnumu,1]<pgl[,1])
  # special case all points of upper polygon are lower
  if(is.na(rnumu)) rnumu<-sum(pg[,1]<pgl[rnuml,1])
  xyl<-pgl[rnuml,]; xyu<-pg[rnumu,]
  # cat("right"); print(rnuml); print(xyl)
  # cat("right"); print(rnumu); print(xyu)
  sr<-cut.p.sl.p.sl(xyl[1:2],xyl[3],xyu[1:2],xyu[3])
}
# left region
if(indl[(npgl<-nrow(pgl))]=="higher"&indu[1]=="lower"){
  # cat("in if of left region: the upper polynom is somewhere lower")
  # checking from the right: last point of lower polygon that is ok
  lnuml<-npgl+1-which(rev(indl=="lower"))[1]
  # checking from the left: last point of upper polygon that is NOT ok
  lnumu<-which(indu=="higher")[1]-1
  # special case all points of lower polygon are upper
  if(is.na(lnuml)) lnuml<-sum(pg[lnumu,1]<pgl[,1])
  # special case all points of upper polygon are lower
  if(is.na(lnumu)) lnumu<-sum(pg[,1]<pgl[lnuml,1])
  xyl<-pgl[lnuml,]; xyu<-pg[lnumu,]
  # cat("left"); print(lnuml); print(xyl)
  # cat("left"); print(lnumu); print(xyu)
  sl<-cut.p.sl.p.sl(xyl[1:2],xyl[3],xyu[1:2],xyu[3])
}
# if(kkk==2){ ##show pg pgl##; INDU<-indu; INDL<-indl; PGL<-pgl; PGU<-pg}

pg<-rbind(pg [indu=="higher",1:2,drop=FALSE],sr,
          pgl[indl=="lower", 1:2,drop=FALSE],sl)

if(debug.plots=="all") lines(rbind(pg,pg[1,]),col="red")
if(!any(is.na(pg))) pg<-pg[chull(pg[,1],pg[,2]),]
# if(kkk==7){ PG <- pg }

```

74 <debug: plot within for loop 74> ≡ C 69

#####

```
#### cat("angtan", angtan, "pg.no", pg.no, "pkt:", pnew)
# if(ia==stopp) lines(pg, type="b", col="green")
if(debug.plots=="all"){
  points(pnew[1], pnew[2], col="red")
  hx<-xyxy[ind.k, c(1,1)]; hy<-xyxy[ind.k, c(2,2)]
  segments(hx, hy, c(10, -10), hy+ang[ia]*(c(10, -10)-hx), lty=2)
# text(hx+rnorm(1, .1), hy+rnorm(1, .1), ia)
# print(pg)
# if(ia==stopp) lines(pgl, type="b", col="green")
points(cutpl[1], cutpl[2], col="red")
hx<-xyxy[ind.kk, c(1,1)]; hy<-xyxy[ind.kk, c(2,2)]
segments(hx, hy, c(10, -10), hy+ang[ia]*(c(10, -10)-hx), lty=2)
# text(hx+rnorm(1, .1), hy+rnorm(1, .1), ia)
# print(pgl)
}
```

75 $\langle \text{debug: plot ini 75} \rangle \equiv \subset 71$

```
if(debug.plots=="all"){ plot(xyxy, type="p", bty="n")
  text(xy, 1:n, col="blue")
  hx<-xy[ind.k, c(1,1)]; hy<-xy[ind.k, c(2,2)]
  segments(hx, hy, c(10, -10), hy+ang[ia]*(c(10, -10)-hx), lty=2)
  text(hx+rnorm(1, .1), hy+rnorm(1, .1), ia)
}
```

4.11 Finding of the bag

To find the bag the function `expand.hull` computes not an exact solution but a numerical approximation. `k.1` indicates the polygon (`exp.dk.1`) with h-depth $(k-1)$. `k.1+1` will usually point to h-depth k (polygon: `exp.dk`), to the inner polygon.

In computing λ we follow Miller et al. (1999). They define λ as the relative distance from the bag to the inner contour and they compute it by $\lambda = (50 - J) / (L - J)$, where D_k contains $J\%$ of the original points and D_{k-1} contains $L\%$ of the original points:

$$\lambda = \frac{50 - J}{L - J} = \frac{n/2 - \#D_k}{\#D_{k-1} - \#D_k} = \frac{\text{number in bag} - \text{number in inner contour}}{\text{number in outer contour} - \text{number in inner contour}}$$

If bag and inner contour is identical then $\lambda \leftarrow 0$.

`k.1` is the number of the rows of `d.k` that represent points within the bag / h-depths greater $n/2$.

76 $\langle \text{find value of lambda 76} \rangle \equiv \subset 34$

```
# if(max(d.k[,2])==k.1 || nrow(d.k)==1) lambda<-0 else { # 121027
if(nrow(d.k)==k.1 || nrow(d.k)==1) lambda<-0 else { # 121126
  ind <- sum(d.k[,2] <= k.1) # complicated, may be wrong in case of missing depths
  ind <- k.1 # 121123
  ndk.1 <- d.k[ind, 1]
  ndk <- d.k[ind+1, 1] # number inner
  # (halve - number inner)/(number outer - number inner)
  lambda <- (n/2 - ndk) / (ndk.1 - ndk)
  # lambda <- (n/2 - d.k[k.1+1,1]) / (d.k[k.1,1] - d.k[k.1+1,1]) # old
  # cat(n/2, ndk, ndk.1, "k.1", k.1, "ind", ind)
}
if(verbose) cat("lambda", lambda)
```

The bag is constructed by $\text{lambda} * \text{outer polygon} + (1 - \text{lambda}) * \text{inner polygon}$.

In former versions it happened that some lines of `h` get NaN values because `nrow(d.k)==2` and `k.l==2` (e.g. example of Wouter Meuleman). This problem doesn't occur no longer but to be sure we have an additional look at `h`.

```
77 <find hull.bag 77> ≡ C 34
cut.on.pdk.l<-find.cut.z.pg(exp.dk, exp.dk.l,center=center)
# print("HALLO"); print(cut.on.pdk.l)
cut.on.pdk <-find.cut.z.pg(exp.dk.l,exp.dk, center=center)
# expand inner polgon exp.dk
h1<-(1-lambda)*exp.dk+lambda*cut.on.pdk.l
# shrink outer polygon exp.dk.l
h2<-(1-lambda)*cut.on.pdk+lambda*exp.dk.l
h<-rbind(h1,h2);
h<-h[!is.nan(h[,1])&!is.nan(h[,2]),]
hull.bag<-h[chull(h[,1],h[,2]),]
# if(verbose){
#   plot(xy); lines(exp.dk,col="red"); lines(exp.dk.l,col="blue");
#   segments(cut.on.pdk[,1],cut.on.pdk[,2],exp.dk.l[,1],exp.dk.l[,2],col="red")
#   segments(cut.on.pdk.l[,1],cut.on.pdk.l[,2],exp.dk[,1],exp.dk[,2],col="blue",lwd=3)
#   points(cut.on.pdk.l,col="blue"); cat("cut.on.pdk.l"); print(cut.on.pdk.l)
#   points(cut.on.pdk,col="red"); cat("cut.on.pdk"); print(cut.on.pdk)
#   lines(hull.bag,col="green")
# }
if(verbose)cat("bag completed:")
#if(verbose) print(hull.bag)
if(debug.plots=="all"){ lines(hull.bag,col="red") }
```

4.12 Computation of the loop

The loop is found by expanding `hull.bag` by factor factor.

```
78 <find hull.loop 78> ≡ C 34
hull.loop<-cbind(hull.bag[,1]-center[1],hull.bag[,2]-center[2])
hull.loop<-factor*hull.loop
hull.loop<-cbind(hull.loop[,1]+center[1],hull.loop[,2]+center[2])
if(verbose) cat("loop computed")
```

Now we identify the points of the bag, the outliers and the outer points. Remark: There may be some points of h -depth $(k-1)$ that are members of the bag. If the data set is very large we will not check whether the h -depth $(k-1)$ points are in the bag.

```
79 <find points outside of bag but inside loop 79> ≡ C 34
if(!very.large.data.set){
  pxy.bag <-xydata[hdepth>= k ,,drop=FALSE]
  pkt.cand <-xydata[hdepth==(k-1),,drop=FALSE]
  pkt.not.bag<-xydata[hdepth< (k-1),,drop=FALSE]
  if( 0 < length(pkt.cand) && 0 < length(hull.bag) ){
    outside<-out.of.polygon(pkt.cand,hull.bag)
    if(sum(!outside)>0)
      pxy.bag <-rbind(pxy.bag, pkt.cand[!outside,])
    if(sum( outside)>0)
      pkt.not.bag<-rbind(pkt.not.bag, pkt.cand[ outside,])
  }
}else {
  extr<-out.of.polygon(xydata,hull.bag)
  pxy.bag <-xydata[!extr,]
  pkt.not.bag<-xydata[extr,,drop=FALSE]
}
```

```

if(length(pkt.not.bag)>0){
  extr<-out.of.polygon(pkt.not.bag,hull.loop)
  pxy.outlier<-pkt.not.bag[extr,,drop=FALSE]
  if(0==length(pxy.outlier)) pxy.outlier<-NULL
  pxy.outer<-pkt.not.bag[!extr,,drop=FALSE]
}else{
  pxy.outer<-pxy.outlier<-NULL
}
if(verbose) cat("points of bag, outer points and outlier identified")

```

The points of the hull of the loop are stored in `hull.loop`.

```

80 <find hull of loop 80> ≡ C 34
    hull.loop<-rbind(pxy.outer,hull.bag)
    hull.loop<-hull.loop[chull(hull.loop[,1],hull.loop[,2]),]
    if(verbose) cat("end of computation of loop")

```

4.13 The definition of `plot.bagplot`

Finally we have to draw the bagplot. This job is managed by a new plot method.

```

81 <define plot.bagplot 81> ≡ C 32
    plot.bagplot<-function(x,
      show.outlier=TRUE, # if TRUE outlier are shown
      show.whiskers=TRUE, # if TRUE whiskers are shown
      show.looppoints=TRUE, # if TRUE points in loop are shown
      show.bagpoints=TRUE, # if TRUE points in bag are shown
      show.loophull=TRUE, # if TRUE loop is shown
      show.baghull=TRUE, # if TRUE bag is shown
      add=FALSE, # if TRUE graphical elements are added to actual plot
      pch=16,cex=.4, # to define further parameters of plot
      verbose=FALSE, # tools for debugging
      col.loophull="#aaccff", # Alternatives: #ccffaa, #ffaacc
      col.looppoints="#3355ff", # Alternatives: #55ff33, #ff3355
      col.baghull="#7799ff", # Alternatives: #99ff77, #ff7799
      col.bagpoints="#000088", # Alternatives: #008800, #880000
      transparency=FALSE,...
    ){
      if(missing(x)) return(<version of bagplot 1>)
      # transparency flag and color flags have been proposed by wouter
      if (transparency==TRUE) {
        col.loophull = paste(col.loophull, "99", sep="")
        col.baghull = paste(col.baghull, "99", sep="")
      }
      <define function win 37>
      <define function cut.z.pg 40>
      <define function find.cut.z.pg 41>
      <initialize some variable 82> #090216
      bagplotobj<-x
      for(i in seq(along=bagplotobj))
        eval(parse(text=paste(names(bagplotobj)[i],"<-bagplotobj[[",i,"]]")))
      if(is.one.dim){
        <construct plot for one dimensional case and return 84>
      } else {
        <construct bagplot as usual 83>
      }
    }
}

```

To prevent "no visible binding" messages during the package building we initialize all variable

that may be referenced. The following list shows the elements of a bagplot object (copied from `compute.bagplot`).

```
82 <initialize some variable 82> ≡ C 81
  center<-hull.center<-hull.bag<-hull.loop<-pxy.bag<-pxy.outlier<-NULL
  # random.seed <-
  hdepths<-is.one.dim<-prdata<-xy<-xydata<-exp.dk<-exp.dk.1<-hdepth<-NULL
  tphdepth<-tp<-NULL
```

The following elements allows us to draw the bagplot: `xydata` (data set), `xy` (sample of data set), `hdepth` (location depth of data points in `xy`), `hull.loop` (points of polygon that define the loop), `hull.bag` (points of polygon that define the bag), `hull.center` (region of points with maximal ldepth), `pxy.outlier` (outlier), `pxy.outier` (outer points), `pxy.bag` (points in bag), `center` (Tukey median), `is.one.dim` is TRUE if data set is one dimensional, `prdata` result of PCA

```
83 <construct bagplot as usual 83> ≡ C 81
  if(!add) plot(xydata,type="n",pch=pch,cex=cex,bty="n",...)
  if(verbose) text(xy[,1],xy[,2],paste(as.character(hdepth))) # cex=2 needs fonts
  # loop: -----
  if(show.loophull){ # fill loop
    h<-rbind(hull.loop,hull.loop[1,]); lines(h[,1],h[,2],lty=1)
    polygon(hull.loop[,1],hull.loop[,2],col=col.loophull)
  }
  if(show.looppoints && 0 < length(pxy.outier)){ # points in loop
    points(pxy.outier[,1],pxy.outier[,2],col=col.looppoints,pch=pch,cex=cex)
  }
  # bag: -----
  if(show.baghull && 0 < length(hull.bag)){ # fill bag
    h<-rbind(hull.bag,hull.bag[1,]); lines(h[,1],h[,2],lty=1)
    polygon(hull.bag[,1],hull.bag[,2],col=col.baghull)
  }
  if(show.bagpoints && 0 < length(pxy.bag)){ # points in bag
    points(pxy.bag[,1],pxy.bag[,2],col=col.bagpoints,pch=pch,cex=cex)
  }
  # whiskers
  if(show.whiskers && 0 < length(pxy.outier)){
    debug.plots<-"not"
    if((n<-length(xy[,1]))<15){
      segments(xy[,1],xy[,2],rep(center[1],n),rep(center[2],n),
        col="red")
    }else{
      pkt.cut<-find.cut.z.pg(pxy.outier,hull.bag,center=center)
      segments(pxy.outier[,1],pxy.outier[,2],pkt.cut[,1],pkt.cut[,2],
        col="red")
    }
  }
  # outlier: -----
  if(show.outlier && 0 < length(pxy.outlier)){ # points in loop
    points(pxy.outlier[,1],pxy.outlier[,2],col="red",pch=pch,cex=cex)
  }
  # center:
  if(exists("hull.center") && 2 < length(hull.center)){
    h<-rbind(hull.center,hull.center[1,]); lines(h[,1],h[,2],lty=1)
    polygon(hull.center[,1],hull.center[,2],col="orange")
  }
  if(!is.one.dim) points(center[1],center[2],pch=8,col="red")
  if(verbose && 0 < length(exp.dk.1) ){
    h<-rbind(exp.dk,exp.dk[1,]); lines(h,col="blue",lty=2)
    h<-rbind(exp.dk.1,exp.dk.1[1,]); lines(h,col="black",lty=2, lwd=3)
    if(exists("tphdepth") && 0<length(tphdepth))
```



```

      text(tp[,1],tp[,2],as.character(tphdepth),col="green")
      text(xy[,1],xy[,2],paste(as.character(hdepth))) # cex=2 needs special fonts
      points(center[1],center[2],pch=8,col="red")
    }
    "bagplot plottet"

```

```

84  <construct plot for one dimensional case and return 84> ≡ C 81
      if(!verbose) cat("data set one dimensional") # 121202
      ROT<-round(prdata[[2]],digits=5); IROT<-round(solve(ROT),digits=5)
      if(!add){ ## 121008 ## 121130
        plot(xydata,type="n",bty="n",pch=16,cex=1, ...) # xlim=xlim, ylim=ylim, ...
      }
      # find five points for box and whiskers
      usr <- par()$usr; xlim <- usr[1:2]; ylim <- usr[3:4]
      mins <- usr[c(1,3)]; ranges <- usr[c(2,4)] - mins
      if(ROT[1,1]==0){ # cat("FALL senkrecht")
        xydata <- cbind( mean(usr[1:2]) ,xydata[,2])
        boxplotres<-boxplot(xydata[,2],plot=FALSE)
        five<-cbind(mean(usr[1:2]),boxplotres$stat)
        dx <- 0.1*(xlim[2]-xlim[1]); dy <- 0
        idx.out <- if(0<length(boxplotres$out)) match(boxplotres$out, xydata[,2] ) else NULL
      }
      if(ROT[1,2]==0){ # cat("FALL waagerecht")
        xydata <- cbind( xydata[,1], mean(usr[3:4]))
        boxplotres<-boxplot(xydata[,1],plot=FALSE)
        five<-cbind(boxplotres$stat,mean(usr[3:4]))
        dx <- 0; dy <- 0.1*(ylim[2]-ylim[1]) # 1/5 of del.y
        idx.out <- if(0<length(boxplotres$out)) match(boxplotres$out, xydata[,1] ) else NULL
      }
      if(ROT[1,2]!=0 && ROT[1,1]!=0){
        xytr<-xydata%*%ROT
        boxplotres<-boxplot(xytr[,1],plot=FALSE)
        five<-cbind(boxplotres$stat,xytr[,2])%*%IROT
        # find small vector for box height
        vec <- five[5,] - five[1,]
        vec.ortho <- c(vec[2],-vec[1]) * ranges / par()$pin
        xy.delta <- vec.ortho * par()$pin[2:1] * ranges # plot region inches
        xy.delta <- xy.delta / sqrt( sum(xy.delta * xy.delta) )
        xy.delta <- xy.delta * .15 / ( sqrt(sum(abs(par()$pin*xy.delta/ranges)^2) ))
        dx <- xy.delta[1]; dy <- xy.delta[2]
        idx.out <- if(0<length(boxplotres$out)) match(boxplotres$out, xytr ) else NULL
      }
      # construct segments
      # whiskers
      segments(five[h<-c(1,5),1],five[h,2],five[h<-c(2,4),1],five[h,2], # col=col.looppoints,
        lwd=2)
      points(five[c(1,5),], cex=1, col=col.looppoints,pch=16)
      # box
      #segments(five[h<-2:4,1] + dx, five[h,2] + dy, five[h,1] - dx, five[h,2] - dy,
      #          col=col.bagpoints,lwd=2)
      #segments(five[2,1] + (h<-c(-1,1))*dx, five[2,2] + h*dy,
      #          five[4,1] + h*dx, five[4,2] + h*dy,
      #          col=col.bagpoints,lwd=2)
      polygon(five[c(2,4,4,2,2),1] + c(dx,dx,-dx,-dx,dx),
        five[c(2,4,4,2,2),2] + c(dy,dy,-dy,-dy,dy),
        col=col.baghull,lwd=1)
      # median

```

```

segments(five[h<-3 ,1] + dx, five[h,2] + dy,
         five[h,1] - dx, five[h,2] - dy,col="red",lwd=3)
# Outlier
if(0 < length(idx.out) && !is.na(idx.out[1])){
  points(xydata[idx.out,,drop=FALSE], cex=1, pch=16,col="red")
}
# segments(five[3,1],five[3,2],five[3,1]+1*vec.ortho[1],
#          five[3,2]+100*vec.ortho[2],col="green",lwd=5)
# segments(five[3,1],five[3,2],five[3,1]+1*vec1[1],
#          five[3,2]+1*vec1[2],col="red",lwd=5)
# points(five,cex=2,col="green")
return("one dimensional boxplot plottet")

```

4.14 Some technical leftovers

4.14.1 Definition of bagplot on start

85 *<start 85> ≡*
<define bagplot 32>

4.14.2 Extracting the R code file bagplot.R

86 *<some functions for generating bagplots 86> ≡*
<define bagplot 32>

87 *<call tangleR to extract tangle function bagplot() 87> ≡*
 tangleR("bagplot.rev",expand.roots="some functions for generating bagplots",
 expand.root.start=FALSE)

5 Inspection tools

5.1 Pairs plot with bagplots

A simple application of bagplot is to use it within `pairs()` for the inspection of data matrices. In this section we present the high level routine `pairs.bagplot()` that allows some trimming of the data, too.

```
88 <define bagplot.pairs 88> ≡ C 89,91
    bagplot.pairs<- function(dm, trim = 0.0, main, numeric.only = TRUE,
                             factor = 3, approx.limit = 300, pch = 16,
                             cex = 0.8, precision = 1, col.loophull = "#aaccff",
                             col.looppoints = "#3355ff", col.baghull = "#7799ff",
                             col.bagpoints = "#000088", ...){
      if(missing(main)) main <- paste(deparse(substitute(dm)), "/ trim =", round(trim,3))
      if(length(trim) == 1) trim <- rep(trim, ncol(dm))
      if(numeric.only){
        dm <- dm[, idx <- sapply(1:ncol(dm), function(x) is.numeric(dm[,x]))]
        trim <- trim[idx]
      }
      for(j in 1:ncol(dm)){
        x <- dm[,j]
        if(!is.numeric(x)) x <- as.numeric(x)
        if( trim[j] > 0) {
          na.idx <- is.na(x)
          xlim <- quantile(x[!na.idx], c(trim[j] , 1-trim[j]))
          x[ na.idx | x < xlim[1] | xlim[2] < x ] <- NA
        }
        dm[,j] <- x
      }
      # DM0 <- dm
      h.fn <- function(x,y){
        idx <- !is.na(x) & !is.na(y)
        x <- x[ idx ]; y <- y[ idx ]
        BP <- bagplot(x,y,add=TRUE,factor = factor, approx.limit = approx.limit, pch = pch,
                      cex = cex, precision = precision, col.loophull = col.loophull,
                      col.looppoints = col.looppoints, col.baghull = col.baghull,
                      col.bagpoints = col.bagpoints, verbose=FALSE)
        # BP <- BP # for debugging
      }
      par(mfrow=c(1,1))
      pairs(dm, panel = h.fn, ...)
      mtext(main, line=2.5)
      dm
    }

89 <start 85>+ ≡
    <define bagplot.pairs 88>

90 <define help of bagplot.pairs 90> ≡
    \name{bagplot.pairs}
    \alias{bagplot.pairs}
    \title{ \code{pairs} plot with bagplots }
    \description{
      \code{bagplot.pairs} calls \code{pairs} and use bagplot() as panel function.
      It can be used for the inspection of data matrices.
    }
  }
```

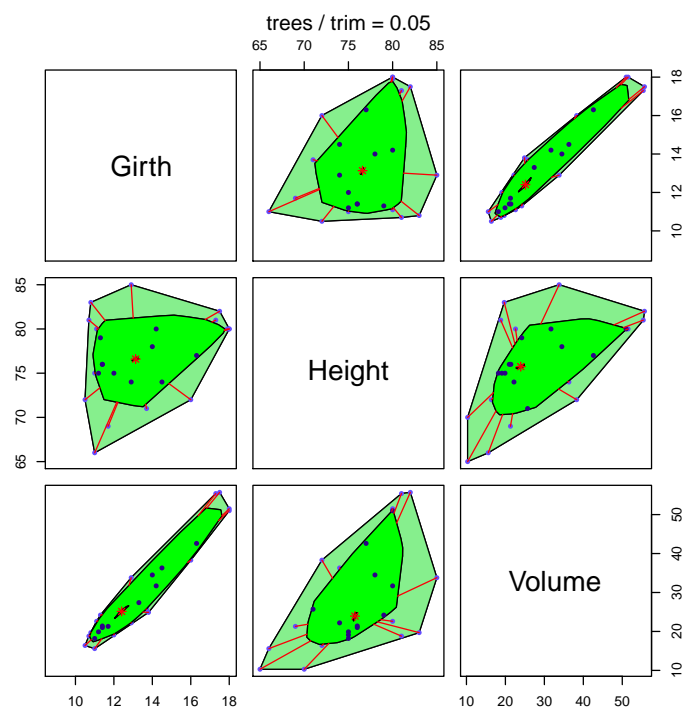
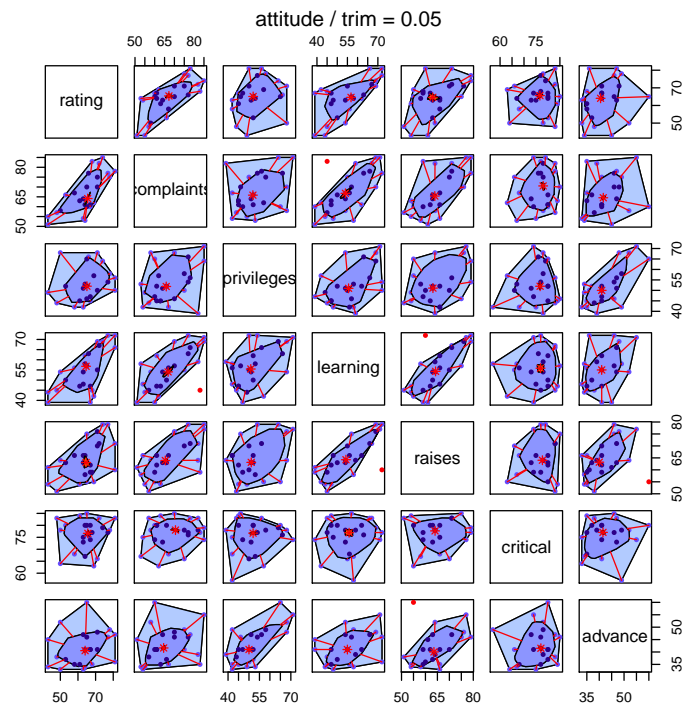
```

\usage{
  bagplot.pairs(dm, trim = 0.0, main, numeric.only = TRUE,
               factor = 3, approx.limit = 300, pch = 16,
               cex = 0.8, precision = 1, col.loophull = "#aaccff",
               col.looppoints = "#3355ff", col.baghull = "#7799ff",
               col.bagpoints = "#000088", ...)
}
\arguments{
  \item{dm}{ datamatrix, columns contain values of the variables }
  \item{trim}{ fraction or vector of fractions of data points
               that should be removed from the variables before computing }
  \item{main}{ title of the plot }
  \item{numeric.only}{ if TRUE only numerical variables will be used. Otherwise an
                       transformation to numeric will be performed.}
  \item{factor}{ see help of bagplot }
  \item{approx.limit}{ see help of bagplot }
  \item{pch}{ see help of bagplot }
  \item{cex}{ see help of bagplot }
  \item{precision}{ see help of bagplot }
  \item{col.loophull}{ see help of bagplot }
  \item{col.looppoints}{ see help of bagplot }
  \item{col.baghull}{ see help of bagplot }
  \item{col.bagpoints}{ see help of bagplot }
  \item{\dots}{ further arguments to be passed to \code{pairs} }
}
\details{
  \code{bagplot.pairs} is a cover function which calls \code{pairs} and uses
  \code{bagplot} to display the data.
}
\value{
  The data which has been used for the plot.
}
\author{Peter Wolf }
\note{
  Feel free to have a look inside of bagplot.pairs and
  to improve it according to your ideas.
}

\seealso{ \code{\link{bagplot}}, \code{\link{pairs}} }
\examples{
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##-- or do help(data=index) for the standard data sets.
# bagplot.pairs(attitude)
# bagplot.pairs(freeny)
# bagplot.pairs(trees,col.baghull="green", col.loophull="lightgreen")
}
% Add one or more standard keywords, see file 'KEYWORDS' in the
% R documentation directory.
\keyword{ misc }
\keyword{ hplot }

```

Here are some results of the examples:



Some tests of function `bagplot.pairs`.

```
91 <do bagplot.pairs test 91> ≡
  # remove "#" in next line to load frabo12 data
  # source("http://www.wiwi.uni-bielefeld.de/fileadmin/stat/wolf/data/frabo12.R")
  <define bagplot 32>
```

```

(define bagplot.pairs 88)
## bagplot.pairs(frabol2[,40:48],trim=0,precision=1) # a lot of variables
## bagplot.pairs(frabol2[,43:47],trim=0,precision=1) # a subset of these variables
### bagplot(frabol2[,43:47][,c(4,1)],precision=1)      ## comparison with single bagplot
### xy <- frabol2[,c(46,41)]; xy<-xy[!is.na(xy[,1]) & !is.na(xy[,2]), ]; xy<-xy[1:151,]
### bagplot(xy,verbose=TRUE,precision=precision,debug.plots="not all")
## bagplot.pairs(frabol2[,c(11,42,46)],trim=0.2,precision=1)[1:2,]
## bagplot.pairs(frabol2[,18:19],trim=0.2,precision=1)[1:2,]      # 0 var in both variables
## bagplot.pairs(frabol2[,c(42,43)],trim=0.25,precision=1)[1:2,] # 0=var(Var1)&var(Var2) sm
## bagplot.pairs(frabol2[,c(1,42)],trim=0.2)                    # few mass points
## DM <- bagplot.pairs(frabol2[,c(39,35)],trim=0.05,precision=1.5) #
if(FALSE){ ## some one dimensional material
  par(mfrow=c(2,2))
  bagplot(x=(rnorm(101)*0-50+c(1:100,200))*100,y=(-50+1*c(1:100,200))*100,verbose=FALSE)
  bagplot(x=-50+c(1:100,190),y=50-1*c(1:100,190),verbose=FALSE)
  bagplot(x=-50+c(1:100),y=50-1*c(1:100),verbose=FALSE)
  bagplot(x=50+c(1:100),y=50+1*c(1:100),verbose=FALSE)
  par(mfrow=c(1,1))
}
## bagplot.pairs(frabol2[,c(1,42)],trim=0.2)
### one dimensional data sets:
## bagplot.pairs(cbind(c(n[-1],300),c(n[-1],300),frabol2[,c(22,19,38)]),
##               trim=trim,precision=1)[1:2,]
## bagplot.pairs(frabol2[,c(22,38,19)],trim=0.2,precision=1)[1:2,] # various boxplots
## bagplot.pairs(frabol2[,c(19,22,38)],trim=0.2,precision=1)[1:2,] # various boxplots
## bagplot.pairs(frabol2[,c(19,38,22)],trim=0.2,precision=1)[1:2,] # various boxplots
## bagplot.pairs(frabol2[,c(38,19,22)],trim=0.2,precision=1)[1:2,] # various boxplots
## bagplot.pairs(frabol2[,c(38,22,19)],trim=0.2,precision=1)[1:2,] # various boxplots
## DM <- bagplot.pairs(frabol2[,c(40,43,45,47)],trim=0,precision=2)# check of center
### points(TP,pch=c(letters,LETTERS)[TPD+1]) ### show h-depths of center candidates
## bagplot.pairs(frabol2[,11:12],trim=0.2,precision=2)[1:3,]      # check of center
## bagplot.pairs(frabol2[,c("Jeanslaenge","SchulNote","Jeansweite")],trim=0.2,precision=1)
## check by construction of bagplots:
if(FALSE){
  par(mfrow=c(2,2)); if(!exists("DM0")) DM0 <- frabol2[,11:12]
  bagplot(DM0[10:40,1:2],precision=1,verbose=TRUE,debug.plots="ll",cex=.2) # check center
  bagplot(DM0[10:40,2:1],precision=1,verbose=TRUE)                        # check center
}
## random selection of variables
# set.seed(13); print(co <- sample(1:49,5));
# bagplot.pairs(frabol2[,co],trim=0.2,precision=1)[1:2,]
## bagplot.pairs(frabol2[,c(20,25)],trim=0.1,precision=02)[1:2,]
## bagplot.pairs(DM0[,2:1],trim=0)
## idx<-!is.na(DM0[,2]);DM1<-DM0[idx,];bagplot.pairs(DM1[,2:1],trim=0)

```

5.2 Function for summarizing variables of a matrix

The function `plotsummary()` computes graphical summary of the variables of a data set. Each variable will be represented by some small types of graphics which are stacked in a summary figure.

At first we define a function for summarizing a vector of data: `plot_single_summary`

```

92 (define plot_single_summary 92) ≡ C 93
plot_single_summary <-
  function(x, trim = 0.0, types=c("stripes","ecdf","density","boxplot"), y.sizes = 1,
          main="", mycols = c("#bbff77","#ffffbb","#bbffff","#77ffbb"), set.par=TRUE){
  (select colors 95)

```

```

    <fix main title 97>
    <transform data into numerical vectors 98>
    <find limits for trimming, normalize data and store visibility: idx.visible 99>
    <initialize graphics and find ymin 100>
    for(i in seq(along=types)){
      type <- types[i]
      <compute stripes 101>
      <compute density trace 102>
      <compute boxplot 103>
      <compute ecdf 104>
    }
    return()
  }
}

```

Now we can use the function for a summarizing function for several variables.

```

93 <define plotsummary 93> ≡ C 94, 105, 106, 150, 151
    plotsummary <-
      function(data, trim = 0.0, types=c("stripes","ecdf","density","boxplot"), y.sizes = 4:1,
        design = "chessboard", main, mycols="RB"){
        <define plot_single_summary 92>
        data.name <- deparse(substitute(data)); if(missing(main)) main <- data.name
        if(is.list(data)){ cols <- length(data); Names <- names(data)} else
          if(is.matrix(data)){ cols <- ncol(data); Names <- colnames(data)} else {
            data <- cbind(as.vector(unlist(data))); cols <- Names <- 1 }
            # if(is.vector(data)||is.ts(data)){
            #   cols <- 1; data <- cbind(data); Names <- " "
            # } else return("Warning: data set not compatible")
        if(0 == length(Names)) Names <- as.character(1:cols)
        plot(1,type="n",xlab="",ylab="",axes=FALSE); oldpar <- par(no.readonly = TRUE)
        if( design != "chessboard" ) mfrow <- c(1,cols) else{
          mfrow <- ceiling(rep(sqrt(cols),2)); mfrow[1] <- ceiling(cols/mfrow[2])
        }
        par(mfrow=mfrow, oma=c(0.1,0.1,.5,0.1), mai=c(.1,.1,.15,0))
        for(j in seq(cols)){
          dat <- if( is.matrix(data) ) dat <- data[,j] else dat <- unlist(data[[j]])
          try(plot_single_summary(dat, trim = trim, types=types, y.sizes=y.sizes,
            main=Names[j], mycols=mycols, set.par=FALSE))
        }
        mtext(main,line=1,adj=0,outer=TRUE)
        par(oldpar)
        NULL
      }
}

```

```

94 <start 85>+ ≡
    <define plotsummary 93>

```

The user should be able to choose between some palettes of data. Here we make a simple proposal:

```

95 <select colors 95> ≡ C 92
    if("#" != substring(mycols[1],1,1)){
      mycols <- if(is.numeric(mycols)) c("RG","RB","GB","GR","BR","BG")[mycols] else
        substring(mycols[1],1,2)
    #   h <- cbind("ff",c("77","bb","ff","bb"), c("bb","ff","bb","77"))
    h <- cbind("ff",c("77","cc","bb","77"), c("77","cc","dd","99"))
    switch( mycols,
      "RG" = mycols <- h[,c(1,2,3)], "RB" = mycols <- h[,c(1,3,2)],
      "GB" = mycols <- h[,c(3,1,2)], "GR" = mycols <- h[,c(2,1,3)],

```

```

        "BR" = mycols <- h[,c(2,3,1)], "BG" = mycols <- h[,c(3,2,1)]
    )
    mycols <- paste("#",mycols[,1],mycols[,2],mycols[,3],sep="")
}

```

Some color schemes, maybe usefull for further experiments:

```

96  (*18)+ ≡
    #               mycols = c("#9999ff", "#ccccff", "#ccffcc", "#99ff99"),
    #               mycols = c("#9999ff", "#ccccff", "#ffee99", "#ffcc55"),
    #               mycols = c("#ff5533", "#ff9977", "#ffee99", "#ffcc55"),
    #               mycols = c("#ffcc55", "#ffee99", "#ff9977", "#ff5533"),
    #               mycols = c("#55ff33", "#99ff77", "#77ff99", "#33ff55"),

```

If a title is delivered it will be used. Otherwise the argument of the call will be plotted.

```

97  <fix main title 97> ≡   C 92
    if("" == main) main <- deparse(substitute(x))

```

To keep things simple we transform data in a way that we can handle them as numerical data.

```

98  <transform data into numerical vectors 98> ≡   C 92
    if(is.ts(x)) x <- as.vector(x)
    if(is.character(x)) x <- as.factor(x); if(is.factor(x)) x <- as.numeric(x)

```

Because of the effects of outliers we trim the data. For defining the limits the R function `quantile()` is used. As a consequence there must not be data points that lie exactly on the limits and there will usually a (small) region at the limits without any point.

```

99  <find limits for trimming, normalize data and store visibility: idx.visible 99> ≡   C 92
    idx.na <- is.na(x)
    if(0 < trim) xlim <- quantile(x[!idx.na],c(trim,1-trim)) else xlim <- range(x[!idx.na])
    if(xlim[1] < xlim[2]) x <- (x - xlim[1])/(xlim[2] - xlim[1]) else x <- x*0 +.5
    xlim <- 0:1
    idx.visible <- (!idx.na) & (0 <= x) & (x <= 1) # index of visible data points

```

```

100 <initialize graphics and find ymin 100> ≡   C 92
    if(set.par) oldpar <- par(mfrow=c(1,1),omi=c(0,0,0,0),mar=c(2,2,1,1))
    plot(1,xlim=xlim,bty="n",type="n",main="",axes=FALSE,ylim=c(0,1))
    par(usr=c(-0.01,1.01,-0.01,1.01)) # ; axis(1)
    title(main,cex.main=0.75)
    rug(x[idx.visible])
    fn <- fivenum(x[idx.visible])
    rect(fn[1:4], 0, fn[2:5], 1, col=mycols[1:4],border=mycols[1:4],xpd=NA) # color
    grid() # ; axis(1)
    n.types <- length(types); if(length(y.sizes) == 1) y.sizes <- rep(y.sizes, n.types)
    y.mins <- 1 - cumsum(y.sizes <- y.sizes/sum(y.sizes))

```

```

101 <compute stripes 101> ≡   C 92
    if(type == "stripes" || "s" == substring(type,1,1) ){
        n <- length(x)
        y0 <- 0.95*seq(x)/n; y0 <- y0*y.sizes[i] + y.mins[i] # print(x); print(xlim)
        segments(rep(xlim[1],n),y0,pmin(xlim[2],pmax(xlim[1],x)),y0,col=i) # stripes
        x [idx.na] <- 0.5
        if(any( ind <- x<xlim[1] )){ points( cbind(-0.01, y0[ ind ]), pch=18, xpd=NA ) }
        if(any( ind <- xlim[2]<x )){ points( cbind( 1.01, y0[ ind ]), pch=18, xpd=NA ) }
        if( 0 < sum(idx.na)){
            segments(rep(xlim[1]-.02,length(idx.na)),y0[idx.na],
                    rep(xlim[2]+.02,length(idx.na)),y0[idx.na],col="red") # red NA-stripes
        }
    }
}

```



```

102  <compute density trace 102> ≡  C 92
      if(type == "density" || "d" == substring(type,1,1) ){
        if( 10 < length( tb <- table(x[idx.visible])) ){
          dx <- density(x[!idx.na]); y0 <- dx$y; x0 <- dx$x
          y0 <- 0.9*y0 / max(y0); y0 <- y0*y.sizes[i] + y.mins[i]
          lines(x0, y0, lwd=2)                                     # density
        } else {
          y0 <- tb; x0 <- as.numeric(names(tb))
          y0 <- 0.9*y0 / max(y0); y0 <- y0*y.sizes[i] + y.mins[i]
          segments( x0, y.mins[i], x0, y0, lwd=3)
        }
      }

103  <compute boxplot 103> ≡  C 92
      if(type == "boxplot" || "b" == substring(type,1,1) ){
        result <- boxplot(x[idx.visible],plot=FALSE); fn <- result$stats; out <- result$out
        y0 <- y.mins[i] + c(0.3,0.7)*y.sizes[i]
        rect(fn[2:3],y0[1],fn[3:4],y0[2],lwd=2)                   # box
        y0 <- y.mins[i] + 0.5*y.sizes[i]
        segments(fn[c(1,5)],y0,fn[c(2,4)],y0,lwd=2)              #,col=mycols[c(1,4)]      # whisker
        points(cbind(out,y0))
        if(any( ind <- x < xlim[1] )){ points( -0.01, y0, pch=18, xpd=NA, cex=2 ) }
        if(any( ind <- xlim[2] < x )){ points(  1.01, y0, pch=18, xpd=NA, cex=2 ) }
      }

104  <compute ecdf 104> ≡  C 92
      if(type == "ecdf" || "e" == substring(type,1,1) ){
        xx <- sort(x[idx.visible])
        yy <- 0.9*seq(along=xx)/length(xx)
        y0 <- yy*y.sizes[i] + y.mins[i]
        points(xx, y0, pch=16); points(xx, y0, type="s"); n.xx <- length(xx)
        segments(-0.02,y.mins[i],xx[1],y.mins[i]); segments(1.02,y0[n.xx],xx[n.xx],y0[n.xx])
        if(any( ind <- x < xlim[1] )){points(-0.01, y.mins[i], pch=18, xpd=NA ) }
        if(any( ind <- xlim[2] < x )){points( 1.01, y.mins[i] + .9*y.sizes[i], pch=18, xpd=NA)}
      }

```

How can get some test data? A way is to look for data sets and check them whether they produce good results.

```

105  <Test of R data sets 105> ≡
      ds.of.R<-function(type="vector"){
        dat<-ls(pos=grep("datasets",search()))
        dat.type<-unlist(lapply(dat,function(x) {
          num<-mode(x<-eval(parse(text=x)))
          num<-ifelse(is.array(x),"array",num)
          num<-ifelse(is.list(x),"list",num)
          num<-ifelse(is.matrix(x),"matrix",num)
          num<-ifelse(is.data.frame(x),"matrix",num)
          num<-ifelse(num=="numeric","vector",num)
          num })))
        return(dat[dat.type==type])
      }
      <define plotsummary 93>
      namelist <- ds.of.R("matrix")
      # namelist <- ds.of.R("vector")
      # namelist <- ds.of.R("list")

```

```

for(i in seq(along=namelist)){
  print(i); print(namelist[i])
  xy <- get(namelist[i])
  if(ncol(xy) < 8 && 2 < ncol(xy) && all(sapply(xy,function(x) !is.factor(x))))
    plotsummary(xy,y.sizes=4:1,trim=.05,main=namelist[i])
  Sys.sleep(1)
}

```

Some special tests.

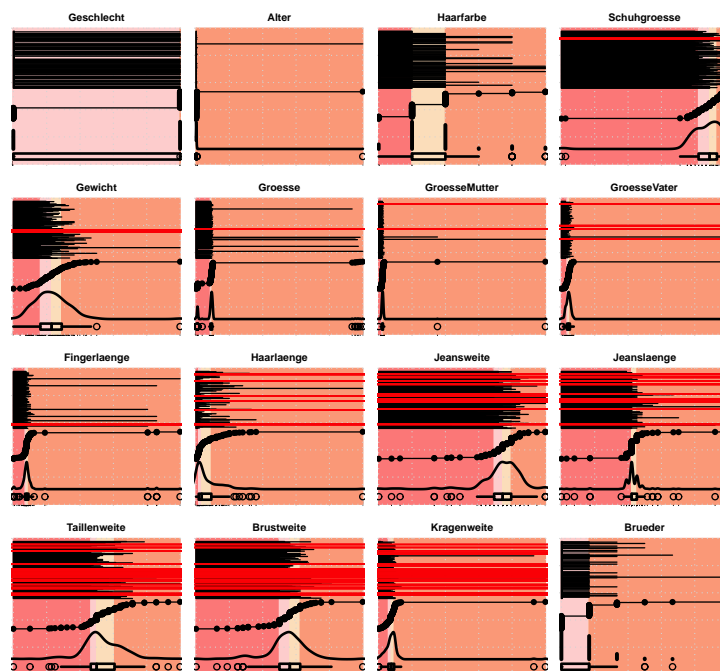
For testing the data matrix of an questionnaire which could be found on our server has been used. The data set `frabo12` can be loaded by sourcing it. An advantage is that there are a lot of missing values and values which are completely wrong.

```

106 < *18) + ≡
# source("http://www.wiwi.uni-bielefeld.de/fileadmin/stat/wolf/data/frabo12.R")
< define plotsummary 93)
plotsummary(frabo12[,1:16],y.sizes = c(8,4,4,2))

```

frabo12[, 1:16]



```

107 < define help of plotsummary 107) ≡
  \name{plotsummary}
  \alias{plotsummary}
  \title{ graphical summaries of variables of a data set }
  \description{
    \code{plotsummary} shows some important characteristics of the variables of a data set.
    For each variable a plot is computed consisting of a barplot, an ecdf,
    a density trace and a boxplot. }
  \usage{
    plotsummary(data, trim = 0, types = c("stripes", "ecdf", "density", "boxplot"),
      y.sizes = 4:1, design = "chessboard", main, mycols = "RB")
  }
  \arguments{

```

```

\item{data}{ Data set for computing a graphical summary. }
\item{trim}{ \code{trim} defines the fraction of observation for trimming on both
ends of the data. }
\item{types}{ vector of types of representation of the data set.
The elements of the vector will induce small plots which are stacked
in vertical order. The first letter of the types is sufficient for
defining a type. }
\item{y.sizes}{defines the relative sizes of the small plots.
The values are divided by their sum to get percentages. }
\item{design}{ if \code{design} is \code{chessboard} the graphics device
is fragmented into rows and cols. Otherwise the images of a variable
build vertical stripes. }
\item{main}{ defines a title for the graphics. }
\item{mycols}{ allows to define some colors for the showing the regions separated
by the quartils. }
}
\details{ \code{plotsummary} can be use for a quick and dirty inspection
of a data matrix or a list of variables.
Without further specification some representation of each of the
variables is built and stacked into a plot.
The sizes of the types of representation can be set as well as the
layout design of the graphics device. It is helpful to trim the data
before processing because outliers will often hide
the interesting characteristics. }
\author{
Peter Wolf, pwolf@wiwi.uni-bielefeld.de
}
\seealso{
\code{\link{pairs}}, \code{\link{summary}}, \code{\link{str}}
}
\examples{
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--\tor do help(data=index) for the standard data sets.
plotsummary(cars)
plotsummary(cars, types=c("ecdf", "density", "boxplot"),
y.sizes = c(1,1,1), design = "stripes")
plotsummary(c(list(rivers=rivers, co2=co2), cars), y.sizes=c(10,3,3,1), mycols=3)
plotsummary(cars, design="chessboard")
# find all matrices in your R
ds.of.R <- function(type="vector"){
dat <- ls(pos=grep("datasets",search()))
dat.type <- unlist(lapply(dat,function(x) {
num <- mode(x<-eval(parse(text=x)))
num <- ifelse(is.array(x),"array",num)
num <- ifelse(is.list(x),"list",num)
num <- ifelse(is.matrix(x),"matrix",num)
num <- ifelse(is.data.frame(x),"matrix",num)
num <- ifelse(num=="numeric","vector",num)
num })))
return(dat[dat.type==type])
}
namelist <- ds.of.R("matrix")
# inspect the matrices one after the other
for(i in seq(along=namelist)){
print(i); print(namelist[i])
xy <- get(namelist[i])
# plotsummary(xy,y.sizes=4:1,trim=.05,main=namelist[i])
}

```

```

    # Sys.sleep(1)
  }
}
\keyword{ hplot }
\keyword{ manip }% __ONLY ONE__ keyword per line

```

Test of the examples.

```

108 <test of the examples of plotsummary 108> ≡
plotsummary(list(rivers,co2),y.sizes=c(10,3,3,1),mycols=3)
plotsummary(c(list(rivers=rivers, co2=co2), cars), y.sizes=c(10,3,3,1),mycols=3)
plotsummary(cars)
plotsummary(cars, types=c("ecdf", "density", "boxplot"), y.sizes = c(1,1,1), design ="stripe")
plotsummary(c(list(rivers=rivers, co2=co2), cars), y.sizes=c(10,3,3,1), mycols=3)
plotsummary(cars, design="chessboard")
# find all matrices in your R
ds.of.R <- function(type="vector"){
  dat <- ls(pos=grep("datasets",search()))
  dat.type <- unlist(lapply(dat,function(x) {
    num <- mode(x<-eval(parse(text=x)))
    num <- ifelse(is.array(x),"array",num)
    num <- ifelse(is.list(x),"list",num)
    num <- ifelse(is.matrix(x),"matrix",num)
    num <- ifelse(is.data.frame(x),"matrix",num)
    num <- ifelse(num=="numeric","vector",num)
    num }))
  return(dat[dat.type==type])
}
namelist <- ds.of.R("matrix")
# inspect the matrices one after the other
for(i in seq(along=namelist)){
  print(i); print(namelist[i])
  xy <- get(namelist[i])
  plotsummary(xy,y.sizes=4:1,trim=.05,main=namelist[i])
  Sys.sleep(1)
}

```

6 Skyline-Plots

Histograms are often used for visualization of the empirical distribution of data sets. However, you get different results depending on the choice of the number of cells and the positions of the first cell. The skyline plot tries to get rid of the unsteadinesses caused by the starting position. Skyline plots are similar to shaded histograms, proposed in [Young et al., 2006]¹

7 Version 1

A simple version of a skyline plot results by plotting several histograms with different starting point for first cell into the same graphics device without cleaning. If you move a window along the x -axis and represent the number of observation in the window by a rectangular box you get an equivalent plot. This procedure is related to the computation of density traces. But for skyline plots we sketch the rectangular boxes and don't estimate the density at single points.

```

109 <*18>+ ≡
<define skyline.hist 111>
par(mfrow=c(3,3))

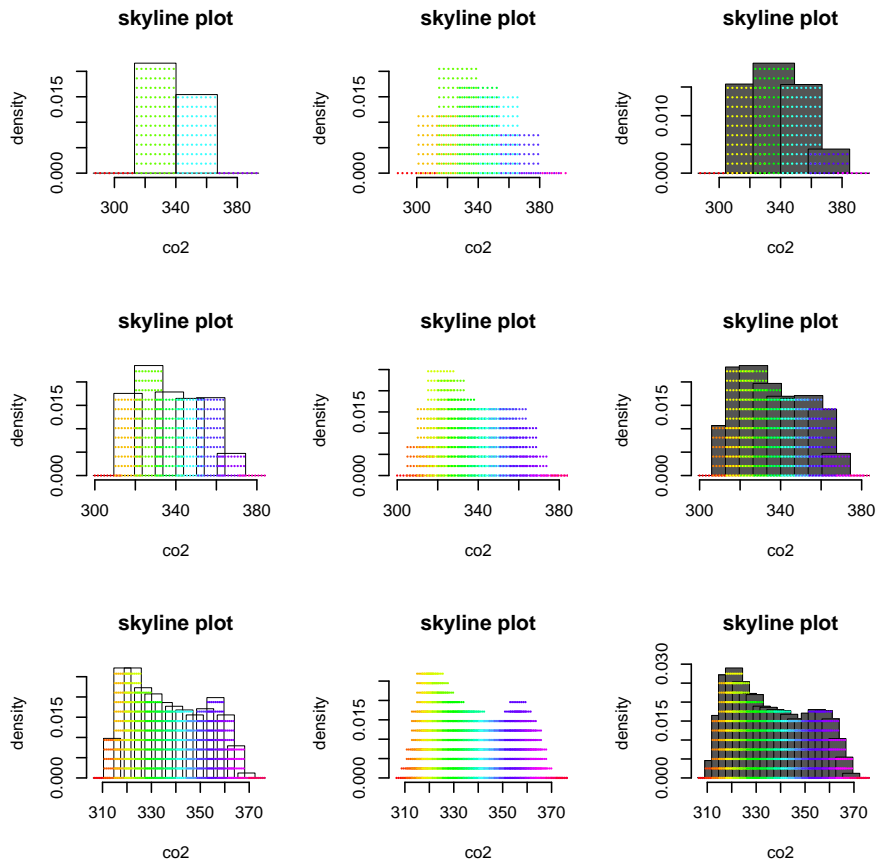
```

¹F.W. Young, R.M. Valero-Mora, M. Friendly (2006): Visual Statistics. Wiley, p207–208.

```

for(n.c in c(2,4,8)){ # some values for n.class
  for(n.h in c(2,4,3)){ # some values for number of n.hist
    n.s <- 9           # value for number of vertical lines
    skyline.hist(co2, n.shading=n.s, n.hist=n.h ,n.class=n.c,
                 night=n.h==3, col.border=n.h!=4)
  }
}

```



```

110 <define help of skyline.hist 110> ≡
    \name{skyline.hist}
    \alias{skyline.hist}
    %- Also NEED an '\alias' for EACH other topic documented here.
    \title{
      \code{skyline.hist} computes a skyline plot which is special histogram.
    }
    \description{
      The function \code{skyline.hist} draws several histograms in one plot. The
      resulting image may look like a skyline.
    }
    \usage{
      skyline.hist(x, n.class, n.hist = 1, main, ylab="density",
        night = FALSE, col.bars = NA, col.border = 4, lwd.border = 2.5,
        n.shading = 6, lwd.shading = 2, col.shading = NA, lty.shading = 3,
        pcol.data = "green", cex.data = 0.3, pch.data = 16, col.data = 1,
        lwd.data = .2, permutation = FALSE,

```

```

        xlab, xlim, ylim, new.plot=TRUE, bty="n", ...)
}
\arguments{
  \item{x}{ one dimensional data set.}
  \item{n.class}{
    number of classes that should be used to find the width of the bars
    of the histogram(s).}
  \item{n.hist}{
    number of histograms that should be plotted.}
  \item{main}{
    used for call of \code{title}.}
  \item{ylab}{
    text for y axis.}
  \item{night}{
    If \code{TRUE} the background will be colored blue.
    If \code{FALSE} there will be no colored background. Otherwise
    \code{night} is used as background color.}
  \item{col.bars}{
    defines the color of the bars. If \code{is.na(col.bars)} and
    \code{night==TRUE} the bars will be colored gray. }
  \item{col.border}{
    color of the borders of the bars.}
  \item{lwd.border}{
    line width of the borders of the bars.}
  \item{n.shading}{
    number of vertical lines for filling the bars of the histograms.}
  \item{lwd.shading}{
    line width of the vertical lines for shading the bars. }
  \item{col.shading}{
    color for the vertical lines for shading. If \code{NA} heat colors are used.}
  \item{lty.shading}{
    line type for the vertical lines for shading.}
  \item{pcol.data}{
    color of data points.}
  \item{cex.data}{
    character size of plotting character.}
  \item{pch.data}{
    plotting character of data points.}
  \item{lwd.data}{
    line width for segments between data points.}
  \item{col.data}{
    color for segments between data points.}
  \item{permutation}{
    if not \code{FALSE} a permutation of the data set is erformed.}
  \item{xlab}{
    text for y axis. }
  \item{xlim}{
    range of x. }
  \item{ylim}{
    range of y. }
  \item{new.plot}{
    logical. If \code{TRUE} a new plot is constructed.}
  \item{bty}{
    box type, used by \code{plot}. }
  \item{\dots}{
    further graphical parameters passed to plot. }
}
\details{
  \code{skyline.hist} computes several histograms and plots them one upon
  the other. The histograms differ in the positions of the first cells,
  but all cells have the same width. The parameters \code{n.class} and
  \code{n.hist} have the greatest effect on the design of the result.

  \code{col.border} allows to color the border of the rectangular boxes of the
  histogram bars. \code{col.bars} defines the fill color of the bars.
  \code{n.shading} defines the number of vertical lines of type
  \code{lty.shading} and width \code{lwd.shading} that are drawn within the boxes.

  Another feature of \code{skyline.hist} is to represent the data points.
  The data points of a cell are plotted according their x-values and
  their ranks (within the points of the cell). The resulting points are connected
  by line segments and you will see a time series running from bottom to top

```

in each cell. The points and lines can be specified by `\code{pcol.data}`, `\code{cex.data}`, `\code{pch.data}`, `\code{lwd.data}`, `\code{col.data}`. To get rid of the original order of the data you can permutated them (`\code{permutation=1}`).

The "skyline" of the plot may be similar to the skyline of a town and the vertical lines may look like small windows of buildings.

In Young et. al. you find "shaded histograms". These histograms have triggered the idea of `\code{skyline.hist}` and the representation of a one dimensional data set by laying histograms on top of otheroverlied histograms.

```

}
\value{
  The result of a call of hist is returned.
}
\references{
  F.W. Young, R.M. Valero-Mora, M. Friendly (2006): Visual Statistics.
  Wiley, p207--208.
}
\author{
  Peter Wolf, pwolf@wiwi.uni-bielefeld.de
}

\seealso{
  \code{\link{hist}}, \code{\link{density}}
}
\examples{
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##-- or do help(data=index) for the standard data sets.
par(mfrow=c(3,3))
for(n.c in c(2,4,8)){ # some values for n.class
  for(n.h in c(2,4,3)){ # some values for number of n.hist
    n.s <- 9           # value for number of vertical lines
    skyline.hist(co2, n.shading=n.s, n.hist=n.h ,n.class=n.c,
                 night=n.h==3, col.border=n.h!=4)
  }
}
skyline.hist(x=rivers, n.class=4, n.hist=2, n.shading=0, main="rivers",
             cex.data=.5, lwd.data = .2, col.data = "green", pcol.data = "red",
             col.border=NA, night=FALSE, ylab="density")
skyline.hist(x=rivers, n.class=4, n.hist=5, n.shading=0, main="rivers",
             cex.data=.5, lwd.data = 1, col.data = "green", pcol.data = "red",
             col.border=NA, night="blue" , ylab="density", col.bars =NA)
skyline.hist(x=rivers, n.class=10, n.hist=2, n.shading=0, main="rivers",
             cex.data=.5, lwd.data = 1, col.data = "green", pcol.data = "red",
             col.border=NA, night=FALSE , ylab="density", col.bars = "lightblue")
skyline.hist(x=rivers, n.class=10, n.hist=1, n.shading=0, main="rivers",
             cex.data=1, lwd.data = 0, col.data = "green", pcol.data = "red",
             col.border=NA, night=FALSE , ylab="density", col.bars = "lightblue" )
skyline.hist(x=rivers, n.class=6, n.hist=1, n.shading=0, main="rivers",
             cex.data=0.1, lwd.data = 2, col.data = "red", pcol.data = "green",
             night="orange" , ylab="density", col.bars = "white", col.border=1 )
skyline.hist(x=rivers, n.class=6, n.hist=1, n.shading=0, main="rivers",
             cex.data=0.1, lwd.data = 2, col.data = "red", pcol.data = "green",
             col.border=NA, night=FALSE , ylab="density", col.bars = "lightblue")
skyline.hist(x=rivers, n.class=6, n.hist=1, n.shading=5, col.shading = "blue",
             main="rivers",
             cex.data=0.1, lwd.data = 1, col.data = "black", pcol.data = "green",

```

```

        col.border=NA, night=FALSE , ylab="density", col.bars = "green")
skyline.hist(x=rivers, n.class=6, n.hist=3, n.shading=5, col.shading = "blue",
            main="rivers", col.bars = "green",
            cex.data=0.1, lwd.data = 1, col.data = "black", pcol.data = "green",
            col.border="white", night="magenta" , ylab="density")
skyline.hist(x=rivers, n.class=6, n.hist=4, n.shading=5, col.shading = "blue",
            main="rivers",
            cex.data=0.8, lwd.data = 1, col.data = "blue", pcol.data = "red",
            col.border=NA, night=FALSE , ylab="density", col.bars = "green")
}
% Add one or more standard keywords, see file 'KEYWORDS' in the
% R documentation directory.
\keyword{ misc }
\keyword{ hplot }

```

Besides the idea of overlaying several histograms we propose different types of representation of the relative frequencies of the cells.

```

111 <define skyline.hist 111> ≡ C 109, 113
    skyline.hist <- function(
        x, n.class, n.hist = 1, main, ylab="density",
        night = FALSE, col.bars = NA, col.border = 4, lwd.border = 2.5,
        n.shading = 6, lwd.shading = 2, col.shading = NA, lty.shading = 3,
        pcol.data = "green", cex.data = 0.3, pch.data = 16, col.data = 1,
        lwd.data = .2, permutation = FALSE,
        xlab, xlim, ylim, new.plot=TRUE, bty="n", ...){
# initialization of variables
if(missing(main)) main <- paste("skyline plot")
if(missing(xlab)) xlab <- deparse(substitute(x))
# find the width of a cell
hist.res <- hist(x, plot=FALSE)
if(!missing(n.class)) width.bars <- diff(range(x))/n.class else {
    width.bars <- diff(hist.res$breaks[1:2])
    n.class <- ceiling(diff(rank(x))/width.bars)
}
# find range for x-axis
x.lim <- (x.range<-range(x)) + c(-1,1) * width.bars
# find size of the bars
starts.bars <- x.range[1] - width.bars*(n.hist-1)/n.hist # start of bar one
starts.bars <- starts.bars + width.bars*(0:(n.hist-1))/n.hist # all starts
starts.bars <- sapply(starts.bars, function(x) x + width.bars*(0:(n.class+1)))
starts.bars <- starts.bars[starts.bars <= max(x)]
idx <- x==x.range[1]; if(any(idx)) x[idx] <- x[idx]+0.001*diff(x.range)
n.bars <- length(starts.bars)
counts.bars <- rep(0,n.bars)
for(i in 1:n.bars){
    counts.bars[i]<-sum( starts.bars[i] < x & x <= (starts.bars[i] + width.bars))
}
n <- length(x)
heights.bars <- counts.bars/n/width.bars
if( 0 < permutation ){
    if(!is.numeric(permutation)) permutation <- 1
    idx <- rep((permutation + 13*7654321)%%9573, n)
    for(i in 2:n) idx[i] <- (idx[i-1] * 1234567 + 87654321) %% 9573
    idx <- order(idx); x <- x[idx]
}
# prepare plot
if(missing(xlim)) xlim <- x.lim
if(missing(ylim)) ylim <- c(0, max(heights.bars))

```



```

if(new.plot)
  plot(0, xlim=xlim, ylim=ylim, type="n", bty="n", xlab=xlab, ylab=ylab, ...)
title(main)
# background
if( night != FALSE ){
  col.night <- if(is.logical(night)) "blue" else night
  usr <- par()$usr; rect(usr[1], usr[3], usr[2], usr[4], col = col.night)
  night <- TRUE
}
# add histogram like borders # if(broder) hist(x,add=TRUE,prob=TRUE)
if(missing(col.bars)) col.bars <- if(night) "#555555" else "white"
rect(starts.bars, heights.bars * 0, starts.bars + width.bars,
      heights.bars, lwd = lwd.border, border = col.border,
      col = col.bars)
# add pattern for filling the bars small windows
if(0 < n.shading){
  dx <- width.bars * ((1:n.shading) - 0.5)/n.shading
  if(is.na(col.shading)) col.shading <- rainbow(n.bars)
  if(length(col.shading) < n.bars) col.shading <- rep(col.shading,n.bars)
  for(i in 1:n.bars){
    segments(starts.bars[i] + dx, rep(0,n.shading),
             starts.bars[i] + dx, rep(heights.bars[i],n.shading),
             lty=lty.shading, lwd=lwd.shading, col=col.shading[i])
  }
}
# representation of the data
for(i in 1:n.bars){
  idx <- starts.bars[i] < x & x <= (starts.bars[i] + width.bars)
  if((count <- sum(idx)) == 0) next
  x.i <- c(starts.bars[i] + 0.5 * width.bars, x[idx])
  n.i <- count + 1
  max.h <- count/n/width.bars
  if(missing(col.data)) col.data <- rainbow(count,start=.0,end=.15)
  y.i <- (0:count)/count * max.h
  segments( x.i[-n.i], y.i[-n.i], x.i[-1], y.i[-1], col = col.data, lwd = lwd.data)
  points(col = pcol.data, pch = pch.data, cex=cex.data, x.i[-1], y.i[-1])
}
# return info of hist
invisible(hist.res)
}

```

```

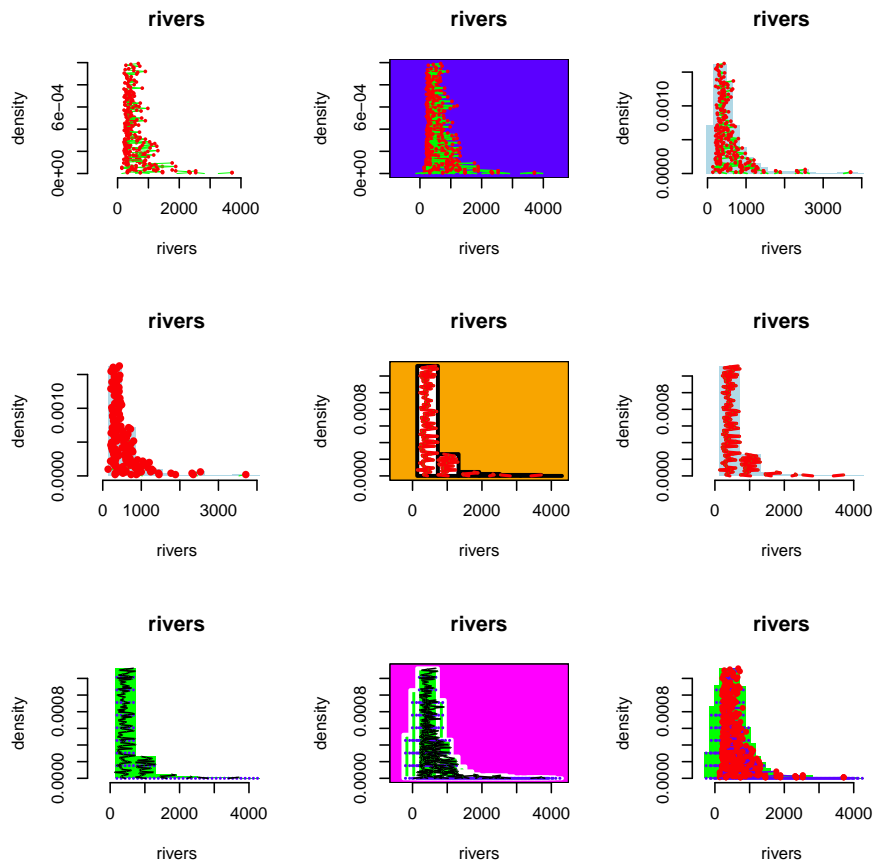
112 (*18)+≡
par(mfrow=c(3,3))
skyline.hist(x=rivers, n.class=4, n.hist=2, n.shading=0, main="rivers",
             cex.data=.5, lwd.data = .2, col.data = "green", pcol.data = "red",
             col.border=NA, night=FALSE, ylab="density"
)
skyline.hist(x=rivers, n.class=4, n.hist=5, n.shading=0, main="rivers",
             cex.data=.5, lwd.data = 1, col.data = "green", pcol.data = "red",
             col.border=NA, night="blue" , ylab="density", col.bars =NA
)
skyline.hist(x=rivers, n.class=10, n.hist=2, n.shading=0, main="rivers",
             cex.data=.5, lwd.data = 1, col.data = "green", pcol.data = "red",
             col.border=NA, night=FALSE , ylab="density", col.bars = "lightblue"
)
skyline.hist(x=rivers, n.class=10, n.hist=1, n.shading=0, main="rivers",
             cex.data=1, lwd.data = 0, col.data = "green", pcol.data = "red",
             col.border=NA, night=FALSE , ylab="density", col.bars = "lightblue"
)

```

```

)
skyline.hist(x=rivers, n.class=6, n.hist=1, n.shading=0, main="rivers", col.border=1,
             cex.data=0.1, lwd.data = 2, col.data = "red", pcol.data = "green",
             night="orange" , ylab="density", col.bars = "white"
)
skyline.hist(x=rivers, n.class=6, n.hist=1, n.shading=0, main="rivers",
             cex.data=0.1, lwd.data = 2, col.data = "red", pcol.data = "green",
             col.border=NA, night=FALSE , ylab="density", col.bars = "lightblue"
)
skyline.hist(x=rivers, n.class=6, n.hist=1, n.shading=5, col.shading = "blue",
             main="rivers",
             cex.data=0.1, lwd.data = 1, col.data = "black", pcol.data = "green",
             col.border=NA, night=FALSE , ylab="density", col.bars = "green",
)
skyline.hist(x=rivers, n.class=6, n.hist=3, n.shading=5, col.shading = "blue",
             main="rivers",
             cex.data=0.1, lwd.data = 1, col.data = "black", pcol.data = "green",
             col.border="white", night="magenta" , ylab="density", col.bars = "green"
)
skyline.hist(x=rivers, n.class=6, n.hist=4, n.shading=5, col.shading = "blue",
             main="rivers",
             cex.data=0.8, lwd.data = 1, col.data = "blue", pcol.data = "red",
             col.border=NA, night=FALSE , ylab="density", col.bars = "green",
)

```



```
113 <(start 85)+≡
      <(define skyline.hist 111)>
```

8 Appendix

8.1 Some further examples – usefull for testing

```
114 <(define rnorm data data, seed: seed, size: n 114)>≡  C 3, 4, 16, 17, 115
      if(!exists("seed")) seed<-75 # 267 81 115
      set.seed(seed)
      #data<-matrix(sample(1:10000,size=2000),1000,2)
      #data<-matrix(sample(1:10000,size=300),50,2)
      if(!exists("n")) n<-100
      data<-cbind(rnorm(n)+100,rnorm(n)+300)
      par(mfrow=c(1,1))
```

```
115 <(* 18)+≡
      seed<-222; n<-100
      <(define rnorm data data, seed: seed, size: n 114)>
      datan<-rbind(data,c(106,294)); par(mfrow=c(1,1))
      datan[,2]<-datan[,2]*100
      <(define bagplot 32)>
      bagplot(datan,verbose=TRUE)
```

```
116 <(quadratic-test 116)>≡
      <(define bagplot 32)>
      dkmethod <- 2; n <- 7 # ; n <- 30
      b <- bagplot(x=1:n,y=(1:n)^2,verbose=TRUE,dkmethod=dkmethod,
                   show.loophull = FALSE)
      lines(b$hull.bag,col="green",lwd=3)
      lines(b$hull.loop,col="red")
      lines(b$exp.dk,col="blue",lwd=4)
      lines(b$exp.dk.1,col="magenta",lwd=4)
```

```
117 <(error 117)>≡
      <(define bagplot 32)>
      <(data set 1 of Wouter Meuleman 13)>
      par(mfrow=2:3)
      bagplot(a[,2],a[,3],verbose=TRUE,debug.plots="all",dkmethod=2)
      par(mfrow=c(1,1))
```

```
118 <(* 18)+≡
      par(mfrow=2:3)
      <(define bagplot 32)>
      bagplot(x=cos((1:100)/100*2*pi),y=-sin((1:100)/100*2*pi),
              precision=1,verbose=TRUE,dkmethod=2,debug.plot="all"); "ok"
```

8.2 Data Sets of AJS

8.2.1 NA/NaN/Inf errors

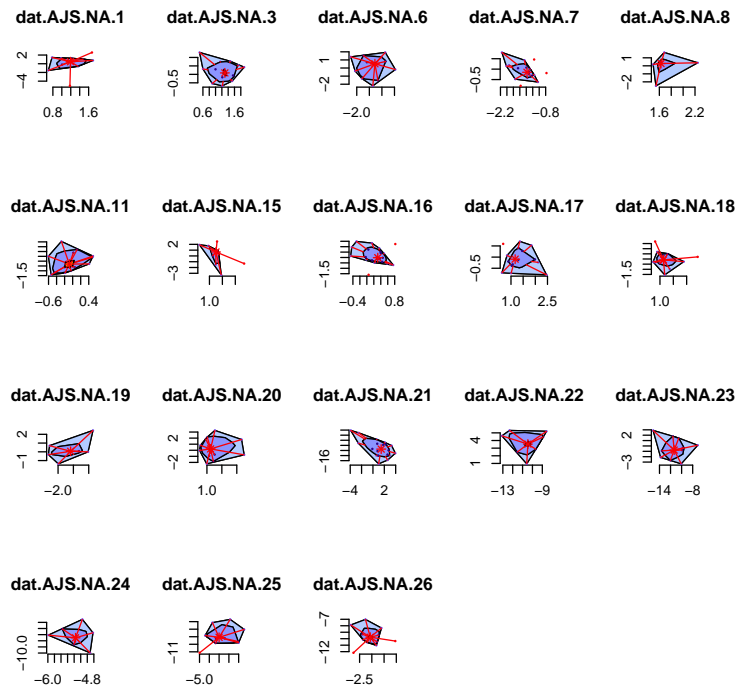
During the work at her thesis Amanda Joy Shaker (AJS) used a lot of different data sets. Some of them generated situations with NA/NaN/Inf errors. These data sets can be used for tests.

```
119 <(define data sets of AJS 119)>≡  C 120, 121, 122
      dat.AJS.NA.1 <- structure(c(0.700748406122636, 0.782097051744013, 1.19494957848218, 1.47659354666956, 1.22724791287169, 1.00382641917022, 0.983568811815667, 1.1814708
      dat.AJS.NA.2 <- structure(c(1, 2, -2, 2, 4, 6), .Dim = c(3L, 2L))
      dat.AJS.NA.3 <- structure(c(1.21436270375692, 1.55386184789978, 1.32729428163828, 1.47680716244686, 1.72291418554236, 1.90528102063964, 0.998303018606679, 1.214690807
      dat.AJS.NA.4 <- structure(c(0.180068704657341, 0.156415127323711, -0.18231911106215, 0.268752871104897, -0.259677941828763, 0.447044306081937, 0.139654173358102, 0.19
      dat.AJS.NA.5 <- structure(c(1.03264386624424, 1.43935200964145, 1.06340967405687, 1.18205321492478, 1.29045381764465, 1.07099564122555, 1.00528855057742, 1.0225907790
      dat.AJS.NA.6 <- structure(c(-1.79145914492076, -1.15875086612266, -1.02145627379982, -0.854737209607734, -2.0630180644543, -1.26431160321251, -0.467161238550217, -0.8
      dat.AJS.NA.7 <- structure(c(-1.3684237236938, -1.65516109096198, -1.37767155061742, -2.15270482106296, -2.01673577099271, -1.45717405769748, -1.31497367263747, -1.396
      dat.AJS.NA.8 <- structure(c(1.54326030862933, 1.68242734109213, 1.62021301730016, 1.49573128869219, 2.2448544609516, -2.49339265803509, 1.66309449850748, 0.3655547175
      dat.AJS.NA.9 <- structure(c(-1.05030572979914, -1.32609971632578, -1.47004504895321, -1.17533756611546, -0.825699561965908, -1.38368827499804, -1.07843553552209, -1.0
      dat.AJS.NA.10 <- structure(c(-0.257855299240321, 0.241495840089441, 0.151683453816352, -0.00715826699300276, -0.276300525209444, 0.299087905248054, 0.21222362760509, 0.
      dat.AJS.NA.11 <- structure(c(-0.279565168686504, 0.491941365497838, -0.173602939876011, -0.19685818172603, -0.534170026008371, -0.190080138935935, 0.504448367630096, 1.
      dat.AJS.NA.12 <- structure(c(0.0831725143122457, 0.191529474941752, 0.251507046865818, -0.0192221009951303, -0.0896881582990316, -0.105353605325231, 0.304107284467248
      dat.AJS.NA.13 <- structure(c(-1.20590819190744, -1.22382974805989, -1.58080060817588, -1.28952446673604, -1.20767921371048, -1.53188318008957, -0.991589312444463, -1.
      dat.AJS.NA.14 <- structure(c(0.160318166551109, 0.263218455998661, -0.138069776534857, 0.0252208468543689, 0.277932015874982, 0.0136710656708733, 0.000647962159711579, 0.
      dat.AJS.NA.15 <- structure(c(1.24249286106101, 1.35303446092109, 1.2529486837982, 1.13531325855934, 0.619887008265003, 1.2940460006555, 2.34074761919455, 1.4542155094
      dat.AJS.NA.16 <- structure(c(-0.460755058165587, 0.8172066720615, 0.666590150541366, 0.456814047248765, 0.142898941396717, 0.580196552111308, 0.392025059314243, 0.220
      dat.AJS.NA.17 <- structure(c(1.36577751048761, 1.84110637374626, 1.46273044580332, 0.890603292480744, 1.00380665882013, 2.11373708423167, 1.98538481298915, 0.83977234
      dat.AJS.NA.18 <- structure(c(0.820055739423539, 0.825044475185051, 1.12973233989894, 1.16326332610937, 1.89013571843305, 2.41249951079619, 0.745546924669453, 0.998440
      dat.AJS.NA.19 <- structure(c(-2.274721736105, -2.31360040700328, -1.0228510082017, -1.99059049801102, -1.31737496414061, -1.63644528759106, -0.862315543463694, 0.7026
      dat.AJS.NA.20 <- structure(c(2.24578438678207, 1.07306586553063, 0.781878198073328, 2.15808651855139, 1.26756253939963, 1.01232594432679, 1.25772827124044, 1.15596430
      dat.AJS.NA.21 <- structure(c(0.997158538921801, 2.58062932297289, 0.614863866430638, 2.55548896378729, 0.806425741060454, 0.662335146824545, 2.22967678970452, 1.43557
      dat.AJS.NA.22 <- structure(c(-13.088254941555, -10.5800843933805, -8.57828767593912, -11.5248721239085, -8.91249596585892, -10.4305909452994, -9.66201715251068, -12.3
      dat.AJS.NA.23 <- structure(c(-13.7879657469664, -15.1422601555544, -10.5183697346654, -7.10147218821867, -11.377301968002, -9.91384432411673, -12.0331310337032, -9.78
      dat.AJS.NA.24 <- structure(c(-5.15758611039991, -4.95032693793226, -5.5575052975057, -4.71606152028894, -4.62746507630567, -5.20432300389483, -5.98242652228454, -8.75
      dat.AJS.NA.25 <- structure(c(-4.15942747285471, -4.99475833541678, -2.72104114362262, -3.4351147069493, -3.96812138733031, -3.70752204695723, -3.00411365069314, -4.01
      dat.AJS.NA.26 <- structure(c(-2.18662147519799, -2.7505614660519, -1.60754752457797, -2.26027306824472, -2.38650537221318, -2.07568459110461, -1.04086205240606, -1.78
```

At first the bagplot function could not deal with data sets consisting only of three points (dat2). Data set 17 runs in an NA/NaN/Inf error because after removing some artificial points during construction wrong slopes were used. Data sets with numbers 24, 22, 19, 15, 11 showed a wrong center.hull. Data sets 4, 5, 9, 10, 12, 13, 14 are near one dimensional and they produce figures similar to a boxplot.

120

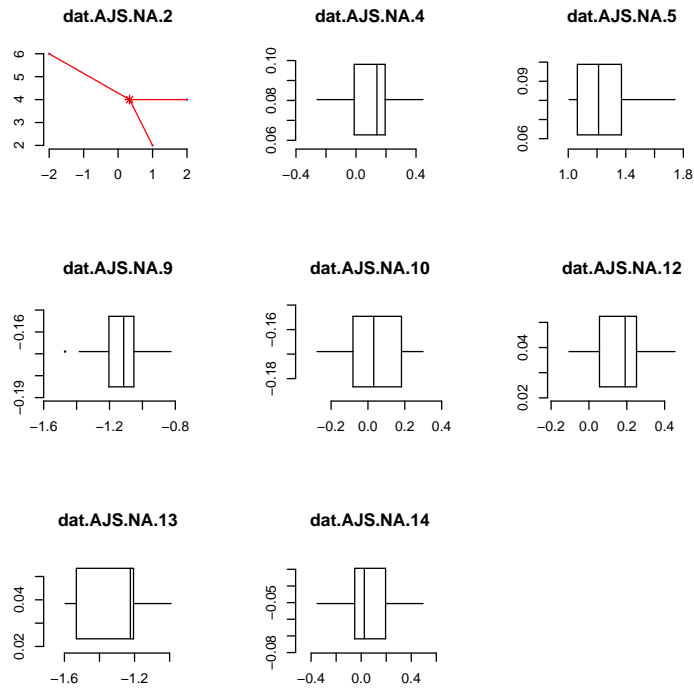
```
<test of amanda-data-sets 120> ≡
<define data sets of AJS 119>
<define bagplot 32>
par(mfrow=c(4,5))
for(nr in c(1,3,6,7,8,11,15,16,17:26)){
  dat.name <- paste("dat.AJS.NA.",nr,sep=" ")
  dat <- eval(parse(text=dat.name))
  try({bagplot(dat,xlab=" ",ylab=" ",main=dat.name)})
}
```



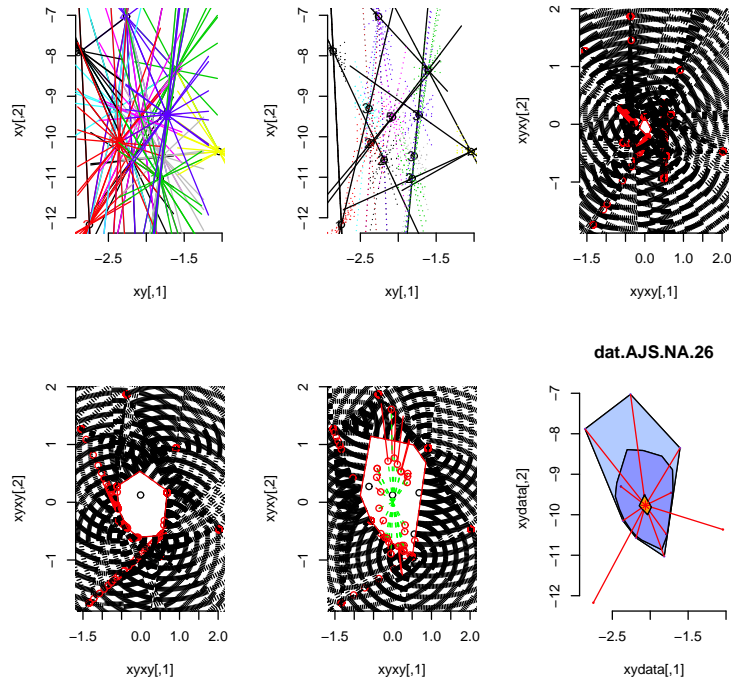
Other errors of these data sets occurred by very small data sets or one dimensional cases.

121

```
<test of amanda-data-sets 120> + ≡
<define data sets of AJS 119>
<define bagplot 32>
par(mfrow=c(3,3))
for(nr in c(2,4,5,9,10,12,13,14)){
  dat.name <- paste("dat.AJS.NA.",nr,sep=" ")
  dat <- eval(parse(text=dat.name))
  try({bagplot(dat,xlab=" ",ylab=" ",main=dat.name)})
}
```



122 *<a verbose test with AJS data no 26 122> ≡*
<define data sets of AJS 119>
<define bagplot 32>
par(mfrow=c(2,3))
nr <- 26; dat.name <- paste("dat.AJS.NA.",nr,sep="")
dat <- eval(parse(text=dat.name))
try({bagplot(dat,debug.plots="all",verbose=!TRUE, main=dat.name)})



8.2.2 Vanishing center polygons

Errors concerning vanishing center polygons. Data sets have been proposed by AJS.

To simplify the extraction of the data from a text representation we use the following function:

```
123 (define function get.AJS.data() 123) ≡ C 124, 125, 130, 131, 133, 134, 135, 136, 137, 138, 139
get.AJS.data <- function(a){
  a <- gsub("RMS.T[0-9]*", "", a); a <- gsub("[^0-9.-]", "", a)
  a <- gsub(" ", "", a); a <- gsub("^(.*)", "", a); a <- gsub(" ", "", a)
  a <- gsub(" ", "", a); a <- paste("dat <- c(", a, ")", sep="")
  a <- eval(parse(text=a)); matrix(a, ncol=2, byrow=2)
}

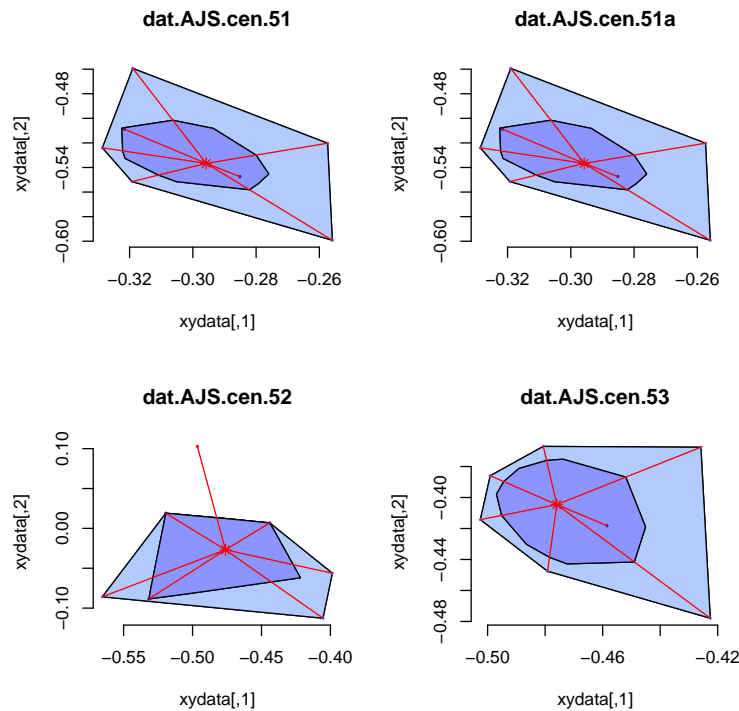
dat.name <- paste("dat.AJS.cen.", nr, sep="")

124 (* 18) + ≡
(define function get.AJS.data() 123)
#Original data that causes the error (Bagploterror51)
# Dir1 Dir2
a<-"RMS.T2 -0.3216039446622190323133 -0.5088442253636682455209
RMS.T6 -0.3189769467668709390651 -0.4593876621644119584431
RMS.T7 -0.3285935044091578549619 -0.5239886751320974589419
RMS.T8 -0.2559059546946075291984 -0.5992055256395006912484
RMS.T5 -0.2957622695531119672019 -0.5366769866153198176306
RMS.T3 -0.2574085563863388603778 -0.5202555871190320813113
RMS.T10 -0.2851383909737250643701 -0.5473508555970837408111
RMS.T11 -0.3192201201256392173455 -0.5514560438800543140658"
dat.AJS.cen.51 <- get.AJS.data(a)
# Data after it has been saved and imported back into R they no longer causes bagplot errors
# V1 V2
a<-"
-0.3216039446622189768021 -0.5088442253636680234763
-0.3189769467668709945762 -0.4593876621644120139543
-0.3285935044091580214953 -0.5239886751320970148527
-0.2559059546946079732876 -0.5992055256395010243153
-0.2957622695531120227130 -0.5366769866153200396752
-0.2574085563863390269113 -0.5202555871190319702890
-0.2851383909737250088590 -0.5473508555970839628557
-0.3192201201256389953009 -0.5514560438800539809989
"
dat.AJS.cen.51a <- get.AJS.data(a)
# Bagploterror52 (original data)
a<-"
RMS.T2 -0.5653783183505530773871 -0.08581631422484529980732
RMS.T6 -0.4965104573284281896939 0.10281100056188693936399
RMS.T7 -0.4054359073563830184739 -0.11267357439288212817008
RMS.T8 -0.5193619418570573076366 0.01898260123444787950131
RMS.T5 -0.3986393540789945899583 -0.05626916023751998147118
RMS.T3 -0.4759403757463211426249 -0.02666424121033471664188
RMS.T10 -0.4440413505805637095492 0.00682853468109060734309
RMS.T11 -0.5316323986960421743575 -0.08844450490794002195383
"
dat.AJS.cen.52 <- get.AJS.data(a)
#Bagploterror53 (original data)
# Dir1 Dir2
a<-"
RMS.T2 -0.4584929840556748459335 -0.4181620733398946354598
```

```

RMS.T6 -0.4806368903226097555326 -0.3670104855925014897267
RMS.T7 -0.4990824552495897736826 -0.3859610888430542452454
RMS.T8 -0.4225857340800127093239 -0.4779159058790974357045
RMS.T5 -0.4759390192662101282117 -0.4045462910525126631889
RMS.T3 -0.4258465780390215393858 -0.3675420112969287611548
RMS.T10 -0.4791654818900950574267 -0.4475245142534178266480
RMS.T11 -0.5025219139233927378996 -0.4142742860227112688953
"
dat.AJS.cen.53 <- get.AJS.data(a)
par(mfrow=c(2,2))
(define bagplot 32)
bagplot(dat.AJS.cen.51,main="dat.AJS.cen.51")
bagplot(dat.AJS.cen.51a,main="dat.AJS.cen.51a")
bagplot(dat.AJS.cen.52,main="dat.AJS.cen.52")
bagplot(dat.AJS.cen.53,main="dat.AJS.cen.53")

```



8.2.3 chull problem

The data from "12.10.2012 05:25" show some strange behavior due to computational problems. On 64 bit machines there are situations (using R 2.10.1 as well as on MacLion systems with newer R versions) showing numerical problems of the `chull()` procedure. This problems resulted in an incorrect order of the points of the returned hull. The problems could be solved by a `chull` function that used some jittering of the input data.

```

125 (define data set with chull problem 125) ≡ C 129
      (define function get.AJS.data() 123)
      # former name: sensitivl
      dat.AJS.chull.l<- "
0.488588355, 0.4788661146, 0.470541460, 0.4755196116, 0.540839656, 0.3452228415, 0.543894166, 0.2321341635,
0.334708444, 0.2329099471, 0.310019325, 0.2738524661, 0.472109253, 0.3495264831, 0.470475506, 0.4563289060,
0.239453165, 0.3450766932, 0.407620177, 0.2770376896, 0.405310593, 0.3564185094, 0.426707657, 0.2959932986,
0.474084923, 0.0631711688, 0.518166032, -0.0208338748, 0.389197562, 0.3093495788, 0.386880009, 0.3639057016,
0.425945405, -0.0809788937, 0.397119599, 0.0008433205, 0.438148342, 0.2093625146, 0.328455105, 0.2751572311,
0.461878996, 0.0668381157, 0.414704143, 0.0072959685, 0.432113535, 0.1879563919, 0.395729535, 0.1531312207,
0.333457931, 0.1964919005, 0.401687453, 0.0946544297, 0.400877654, 0.1471504997, 0.464775289, 0.2009438452,
0.549797862, 0.1277673373, 0.381226770, 0.2668278562, 0.550015218, 0.0668506110, 0.477237805, 0.0740817598,
0.458555746, 0.0775954771, 0.528883119, 0.0948835096, 0.486477230, 0.1845105884, 0.418608439, 0.0991275028,
0.431439496, -0.0954497603, 0.410785196, -0.1345022514, 0.395787730, 0.1228479500, 0.469339289, 0.1508086005,
0.503063303, 0.1850048726, 0.573604375, 0.3706804547, 0.539558323, 0.4411575294, 0.457817697, 0.4406435958,
0.389985667, 0.1327410145, 0.395020633, 0.1097348779, 0.439170368, 0.4193358276, 0.121591061, 0.4339624810,
0.335464949, 0.2473185283, 0.351423426, 0.2246070027, 0.506942440, 0.2389513899, 0.416466941, 0.2150341745,
0.445740068, 0.2102291025, 0.466276802, 0.0843550997, 0.487222847, 0.3058908093, 0.469796584, 0.3957596935,
0.377076327, 0.1589928857, 0.453156085, 0.1215739645, 0.487773276, 0.3732129409, 0.440684358, 0.4203934197,
0.408433299, -0.1617784087, 0.384397653, -0.0260943984, 0.328457468, 0.1822356067, 0.331533173, 0.2324971255,
0.476101470, 0.2588617367, 0.433777557, 0.2836051641, 0.393777809, -0.1773367157, 0.375883187, -0.0702600981,
0.495420045, 0.3007244720, 0.504617623, 0.2888109972, 0.431188119, 0.0947638338, 0.340934937, 0.1330004897,
0.420439561, 0.0483710223, 0.370461663, 0.0278395062, 0.441039662, 0.2107990385, 0.419965287, 0.2825255013,
0.439766224, 0.2778765905, 0.464379420, 0.3177810056, 0.276301115, -0.1202481539, 0.298233219, -0.0295892006,
0.395396220, -0.1763108102, 0.347408037, -0.0847462807, 0.339337876, -0.0694771178, 0.414951665, -0.0048018001,
0.459726406, 0.2059963736, 0.499616132, 0.1056579095, 0.440375533, 0.0351919354, 0.482810983, 0.1113721218,
0.432128546, 0.2362745203, 0.490179909, 0.2595651429, 0.478780067, 0.2343187346, 0.464693584, 0.2229943353,
0.391961926, 0.1504157125, 0.415383010, 0.2261420094, 0.460120520, 0.2819996100, 0.419861026, 0.1560055650,
0.439423034, 0.3700449651, 0.480516957, 0.2949836759, 0.437066927, 0.1593231841, 0.476421460, 0.1429883832,

```

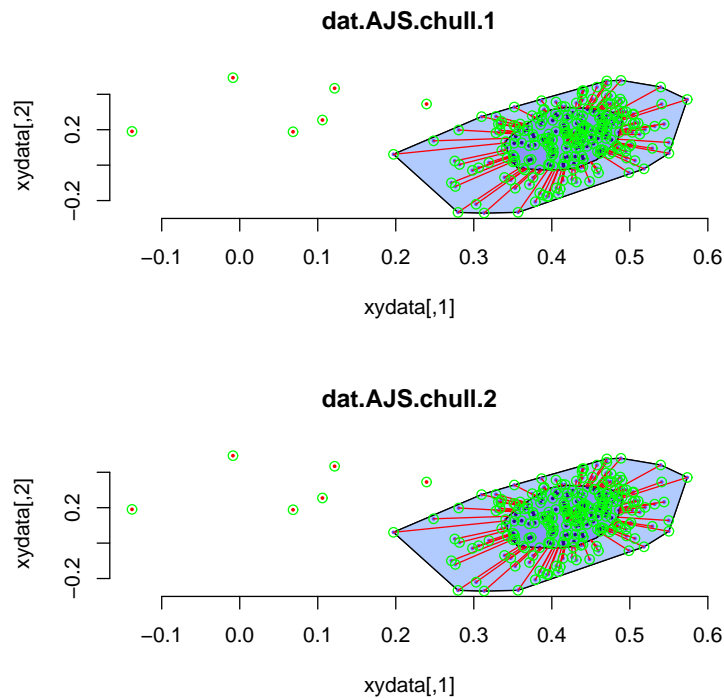
```

0.271264449, -0.0995625736, 0.353000692, -0.1304770232, 0.357545439, 0.1943493893, 0.457257215, -0.0176474562,
0.427548344, -0.0231445876, 0.432522398, 0.1517096819, 0.451225294, 0.1502057497, 0.361329739, 0.1264532926,
0.435214532, 0.1010238420, 0.470582949, 0.1851516685, 0.437463975, 0.2533124438, 0.417048013, 0.3303409805,
0.384538442, -0.1572839601, 0.371303473, -0.1093184580, 0.421293546, 0.1425996465, 0.495727796, 0.0404091001,
0.302936071, -0.2205527851, 0.279745580, -0.2667278459, 0.248452778, 0.1373261412, 0.281694737, 0.0036316128,
0.475994216, 0.3916927014, 0.401989717, 0.0723050968, 0.503976003, 0.2849699244, 0.462111313, 0.1864648537,
0.451977235, 0.3730726001, 0.484357984, 0.1001568997, 0.470939953, 0.2808115329, 0.359385679, 0.2325216403,
0.494904270, 0.0783018081, 0.459803689, -0.0437143636, 0.439798527, -0.0089031661, 0.491112867, 0.3022055065,
0.418731981, 0.1857299860, 0.390291752, 0.3154725004, 0.399472537, 0.0498061800, 0.454767812, 0.3038274596,
0.435419817, 0.1737104118, 0.445882290, 0.3164461434, 0.458978707, 0.1518294804, 0.335919123, 0.0824141186,
0.412775582, 0.3319219809, 0.471062889, 0.2155976166, 0.400704784, 0.1395017856, 0.374875509, 0.2793858787,
-0.138096988, 0.1904736671, 0.280579335, 0.1983484280, 0.420128402, 0.2619921934, 0.454072497, 0.3034271682,
0.395793576, 0.0050705166, 0.365257809, -0.0272525317, 0.484044950, 0.3570199113, 0.370868043, 0.1093188543,
0.461285268, 0.2632532003, 0.350116296, 0.0184022352, 0.368931907, 0.2208158238, 0.531850140, 0.2198341291,
0.416720210, 0.1982638678, 0.414312034, -0.0859862833, 0.438835285, 0.0510980754, 0.398296976, 0.0872646406,
0.306207783, -0.0149382671, 0.469371414, 0.1709586638, 0.415926279, 0.1801283181, 0.385563049, 0.2280534621,
0.445041509, 0.1062464264, 0.450434171, 0.1659846969, 0.105995009, 0.2547933195, 0.401549400, -0.0054054975,
0.068326754, 0.1883474596, 0.428085276, 0.1259003630, 0.386667519, 0.2788522754, 0.364941811, 0.193929163,
0.388007330, 0.2488991171, 0.313310938, -0.2703923641, 0.352913354, 0.0216299592, 0.448419498, -0.0917665640,
0.431455476, 0.3508598056, 0.432434083, 0.1242010665, 0.428347577, 0.0089587276, 0.451061960, 0.1955492468,
0.438723171, 0.4033360370, 0.375784983, 0.1732615065, 0.519498731, 0.2018356726, 0.440336331, 0.1348062337,
0.490173805, 0.3545576781, 0.504040802, 0.2314755193, 0.465547217, 0.3347197651, 0.351917086, 0.3282052014,
0.487052697, 0.0902093812, 0.429959415, 0.0480855954, 0.437206119, -0.0127709904, 0.469334595, 0.2619805975,
0.405168014, 0.1902292618, 0.464521889, 0.2275828903, 0.396088688, 0.1754067895, 0.441944151, 0.2202335286,
0.464586609, 0.2507695841, 0.483560544, 0.2672001335, 0.373528072, 0.032623727, 0.350304411, 0.0555681148,
0.495812155, 0.2023598468, 0.436919679, 0.1396848934, 0.420938210, 0.1287555760, 0.333588506, 0.1964314573,
-0.008673143, 0.4936967464, 0.498878462, -0.0429958219, 0.440685091, 0.2022473549, 0.419744785, 0.1800348923,
0.341760660, 0.0975451159, 0.351905442, 0.0467315695, 0.510702253, 0.2369286285, 0.319368430, 0.0144631138,
0.387779601, 0.1468151850, 0.363116992, 0.0182525890, 0.463043764, 0.1998838022, 0.485449316, 0.2410686817,
0.351764496, 0.1236883913, 0.378846594, -0.2068052412, 0.506541313, 0.0258194329, 0.462346785, 0.0841345518,
0.197055593, 0.0611038043, 0.369159816, 0.1978663678, 0.437002216, 0.1783461539, 0.353863511, 0.1619696742,
0.378026956, 0.1140428654, 0.501398826, 0.1154449124, 0.401387838, 0.0589433175, 0.275583568, 0.0233760805,
0.463515049, 0.0876374263, 0.404737777, 0.2917370814, 0.401168387, 0.2526006111, 0.429059789, 0.1795592816,
0.353763986, 0.2162801483, 0.356734605, -0.2663552210, 0.342808121, 0.0407156242, 0.405815773, -0.1289238452,
0.475349726, 0.3478269079, 0.512681955, 0.1740650182, 0.388882734, -0.1686691003, 0.416400594, 0.0422744471"

dat.<A>JS.chull.2<-"
0.488565925, 0.4788830230,
0.470518883, 0.4755379903, 0.540821319, 0.3452584613, 0.543880329, 0.2321764386, 0.334695847, 0.2329343592,
0.310005378, 0.2738714765, 0.472091750, 0.3495517610, 0.470453134, 0.4563524148, 0.239437539, 0.3450803439,
0.407605742, 0.2770631607, 0.405297910, 0.3564038970, 0.426698947, 0.2959717498, 0.474078975, 0.0632119082,
0.518165669, -0.0208026310, 0.389182699, 0.3093662369, 0.386863379, 0.3639162011, 0.425943787, -0.0809216837,
0.397113988, 0.0008999279, 0.438144516, 0.2093360858, 0.328446752, 0.2751362229, 0.461876192, 0.0668544126,
0.414702341, 0.0073236107, 0.432106261, 0.1879619489, 0.395722604, 0.1531447993, 0.333445239, 0.1965294921,
0.401681094, 0.0946842860, 0.400866278, 0.1471984008, 0.464762400, 0.2009867588, 0.549788161, 0.1278160114,
0.381209640, 0.2668751891, 0.550009153, 0.0668940683, 0.477231633, 0.0741204860, 0.458550419, 0.0776261287,
0.528876578, 0.0949198625, 0.486467492, 0.1845373490, 0.418601288, 0.0991622310, 0.431444647, -0.0954361392,
0.410791285, -0.1344827904, 0.395781200, 0.1228690810, 0.469332443, 0.1508254156, 0.503049055, 0.1850646603,
0.573583529, 0.3707267483, 0.539540161, 0.4411587741, 0.457799411, 0.4406425283, 0.389979310, 0.1327572440,
0.395014994, 0.1097540691, 0.439149132, 0.4193625992, 0.121568356, 0.4339816581, 0.335450967, 0.2473479868,
0.351411589, 0.224629427, 0.506931531, 0.2389686803, 0.416453891, 0.2150714007, 0.445728307, 0.2102598983,
0.466270713, 0.0843892346, 0.487207646, 0.3059151603, 0.469777318, 0.3957816522, 0.377067897, 0.1590144819,
0.453149791, 0.1215962092, 0.4757757526, 0.3732180676, 0.440667092, 0.4203912650, 0.408446801, -0.1618030914,
0.384406713, -0.0261348064, 0.328457377, 0.1821870524, 0.31531072, 0.2324458686, 0.476092076, 0.2588599612,
0.433767577, 0.2835973313, 0.393793742, -0.1773741682, 0.375897238, -0.0703216254, 0.495413425, 0.3006890687,
0.504610945, 0.2887805058, 0.431192387, 0.0947182527, 0.340937434, 0.1329507636, 0.420446411, 0.0483223700,
0.370470482, 0.0277816649, 0.441037167, 0.2107626360, 0.419959108, 0.2824900724, 0.439763007, 0.2778222224,
0.464372875, 0.3177378989, 0.276309247, -0.1202539042, 0.298239822, -0.0296142651, 0.395415765, -0.1763745880,
0.347422594, -0.0848076458, 0.339350477, -0.0695299709, 0.414959369, -0.0048385120, 0.459727233, 0.2059384489,
0.499622833, 0.1055938656, 0.440380491, 0.0351622932, 0.482813054, 0.1113387918, 0.432127974, 0.2362151044,
0.490176417, 0.2595152506, 0.478778121, 0.2342718275, 0.464692712, 0.2229430093, 0.391963529, 0.1503685476,
0.31582189, 0.2260871924, 0.460109318, 0.2820022326, 0.419856323, 0.1560030889, 0.439414519, 0.3699970204,
0.480511783, 0.2949392124, 0.437066976, 0.1592860167, 0.476422133, 0.1429539738, 0.271275838, -0.0995991441,
0.353014229, -0.1305150420, 0.357543713, 0.1943096579, 0.457267580, 0.176971730, 0.427558337, -0.0231909552,
0.432520500, 0.1516561842, 0.451225649, 0.1501701068, 0.361330739, 0.1264174670, 0.435216894, 0.1009900092,
0.470581163, 0.1851202194, 0.437460031, 0.2532716932, 0.417040532, 0.3302983395, 0.384555065, -0.1573336823,
0.371318093, -0.1093708134, 0.421298482, 0.1425323555, 0.495736478, 0.0403531005, 0.302958385, -0.2206251139,
0.279767951, -0.2667856264, 0.248455667, 0.1372682707, 0.281703006, 0.0035824033, 0.475976319, 0.3917064509,
0.401985224, 0.0723292124, 0.503963566, 0.2849822455, 0.462099159, 0.1865073529, 0.451960177, 0.3730857222,
0.484354627, 0.1001666315, 0.470928061, 0.2808199926, 0.359374566, 0.2325365032, 0.494898857, 0.0783343357,
0.459811455, -0.0437362500, 0.439796407, -0.0088566045, 0.491099424, 0.3022186063, 0.438721732, 0.1857572104,
0.390276222, 0.3154919032, 0.399469456, 0.0489677648, 0.454754097, 0.3038404628, 0.435410153, 0.1737382460,
0.445867263, 0.3164638891, 0.456967530, 0.458967530, 0.1518767404, 0.335919344, 0.0821980295, 0.432763921, 0.3319087519,
0.471049431, 0.2156338459, 0.406992737, 0.1395100302, 0.374861658, 0.2794049727, -0.138108934, 0.1904884369,
0.280570722, 0.198237877, 0.420120277, 0.2619778732, 0.454057069, 0.3034526292, 0.395791685, 0.0050987952,
0.365260425, 0.0272468779, 0.484034491, 0.3569887088, 0.370863417, 0.1093298911, 0.461277753, 0.2632357520,
0.350125223, 0.0183460251, 0.368930135, 0.2207677738, 0.531849230, 0.197869496, 0.436717917, 0.1982293252,
0.414321880, -0.086010479, 0.438841302, 0.0510552491, 0.398298536, 0.0872398038, 0.306217635, -0.0149914778,
0.469373834, 0.1709017285, 0.415929594, 0.1800595546, 0.385559536, 0.2280161638, 0.445047862, 0.1061824345,
0.450436609, 0.1659285617, 0.105991452, 0.386663185, 0.2547355455, 0.364941456, -0.0054767733, 0.068325138, 0.1882970253,
0.428084182, 0.1258825829, 0.386663185, 0.2788034267, 0.448430163, -0.0917932667, 0.388006007, 0.3488389434,
0.313331553, -0.2704348401, 0.352922806, 0.0215689704, 0.451061147, 0.195063904, 0.431449465, 0.350800866,
0.432438166, 0.1241467239, 0.428358234, 0.0088965389, 0.440325400, 0.1348568173, 0.437706577, 0.4903386057,
0.375774923, 0.1247747466, 0.519488312, 0.2018627370, 0.351902301, 0.3282132558, 0.487045154, 0.4002528398,
0.504032360, 0.2314774746, 0.465533674, 0.3347213235, 0.469321808, 0.2620019309, 0.405157393, 0.1902570513,
0.429961565, 0.4800713261, 0.437203684, -0.0127309372, 0.469321808, 0.2202391505, 0.464574093, 0.2507926309,
0.464512601, 0.2275906537, 0.396079699, 0.1754275458, 0.441935381, 0.0555592694, 0.495803894, 0.2023701918,
0.483547025, 0.2672255285, 0.373521032, 0.0323174729, 0.350304993, 0.0555592694, 0.495803894, 0.2023701918,
0.436908707, 0.1397339520, 0.420933304, 0.1287640020, 0.333577474, 0.1964571050, -0.008696645, 0.4936956907,
0.498882187, -0.0429872111, 0.440681194, 0.2022239918, 0.419736249, 0.1800517691, 0.341754056, 0.0975732594,
0.351905803, 0.0467274302, 0.510698280, 0.2368967597, 0.319367930, 0.0144749590, 0.387776486, 0.1468031004,
0.363127252, -0.0183052364, 0.463043825, 0.1998336427, 0.485445196, 0.2410354073, 0.351766303, 0.1236472952,
0.378860874, -0.2068662282, 0.505488976, 0.0257762531, 0.462348947, 0.0841091086, 0.197060382, 0.0610563781,
0.369159733, 0.1978140552, 0.437006678, 0.1782706082, 0.353860893, 0.1619473817, 0.378029585, 0.1140002595,
0.501403770, 0.1153902428, 0.401394224, 0.0588935729, 0.275594400, 0.0233006882, 0.463520128, 0.0875897916,
0.404729147, 0.2917155155, 0.401167106, 0.2525394059, 0.429061979, 0.1794993730, 0.353765868, 0.2162066871,
0.356755336, -0.2663981227, 0.342817447, 0.0406485442, 0.405826288, -0.1289384243, 0.475344946, 0.3477611726,
0.512683744, 0.1740133571, 0.388901598, -0.1687308737, 0.416404997, 0.0422453698"

par(mfrow=c(2,1))
(define bagplot 32)
dat.<A>JS.chull.1 <- get.<A>JS.data(dat.<A>JS.chull.1)
dat.<A>JS.chull.2 <- get.<A>JS.data(dat.<A>JS.chull.2)
bagplot(dat.<A>JS.chull.1,main="dat.<A>JS.chull.1")
points(dat.<A>JS.chull.2,col="green")
bagplot(dat.<A>JS.chull.2,main="dat.<A>JS.chull.2")
points(dat.<A>JS.chull.1,col="green")
summary(dat.<A>JS.chull.1-dat.<A>JS.chull.2)

```

A time consuming analysis showed that on 64 bit machines chull will not work correct in some rare data situations. Here is an example:

```
126 (Example where chull gives bad results 126) ≡
# extracted from bagplot construction of data set [[dat.AJS.chull.2]]
# hull <- hull.bag[chull(hull.bag[1:5,1], hull.bag[1:5,2]),]
# print(H1 <- H<-hull.bag[c(1:5),,])
options(digits=22)

# DATA
HH<- structure(c(0.490158077, 0.484515879715913, 0.466198439833506,
0.484154626783293, 0.46323687483959, 0.244633264125914, 0.142239809910196,
0.0631969378331613, 0.140680941993944, 0.0504172806067828), .Dim = c(5L,
2L), .Dimnames = list(c("", "", "", "", ""), NULL))
HHdelta <- structure(c(0, 4.44089209850063e-16, 1.11022302462516e-16,
1.11022302462516e-16, -2.77555756156289e-16, -2.4980018054066e-16,
-1.38777878078145e-16, 2.77555756156289e-17, -1.38777878078145e-16,
-2.77555756156289e-17), .Dim = c(5L, 2L),
.Dimnames = list(c("", "", "", "", ""), NULL))
dat.AJS.chull.2a <- H <- HH + HHdelta
print(H)
print(diff(H[,2])/diff(H[,1]))
# PROBLEM
plot(H,type="n",xlim=c(0.46,0.49),)
points(H[,1]-0.0015,H[,2],pch=as.character(1:5))
no <- grDevices::chull(H[,1],H[,2])
hull <- H[no,]
points(hull,type="b",pch=LETTERS,col="red")
print(no)

# SOLUTION experimental
HK <- H+1E-10*sin(H+(1:length(H)))
no <- grDevices::chull(HK[,1],HK[,2])
hull <- H[no,]
points(hull[,1]-0.0005,hull[,2],type="b",pch=letters,col="blue")
print(no)
```

Output of 64 bits R:

```
      [,1]      [,2]
0.4901580770000000 0.24463326412591374
0.4845158797159134 0.14223980991019586
0.4661984398335061 0.06319693783316133
0.4841546267832931 0.14068094199394385
0.4632368748395897 0.05041728060678277

18.147797579590311  4.315170274037586  4.315170274037585  4.315170274037590
[1] 2 3 4 5 1
[1] 2 5 1
```

Output 32 bits R

```

                                [,1]                                [,2]
0.49015807700000000253361 0.24463326412591374081629
0.4845158797159134222987 0.14223980991019585795598
0.4661984398335061174912 0.06319693783316132629224
0.4841546267832931293995 0.14068094199394384768986
0.4632368748395896962400 0.05041728060678277167916

18.147797579590310590447  4.315170274037585684823  4.315170274037584796645
 4.315170274037590125715
[1] 2 4 5 1
[1] 2 5 1

```

The result of `chull` should be the indices of points in a clockwise order. To check this condition we propose here a short algorithm. We build triangles consisting of the gravity center point and the end points of the segments of the polygons and check the senses of rotation of these triangles. For the check of a triangle we construct an orthogonal vector to a side of the triangle and multiply this with the vector of a segment of the polygon.

```

127 (check clockwise condition of polygon 127) ≡ C 129,132
check.sense.of.rotation <- function(pg, debug=FALSE){
  # find segments of polygon
  xy <- rbind(pg, pg[1,]); xy <- cbind(diff(xy[,1]),diff(xy[,2]))
  # center of gravity must be within the polygon
  S <- colMeans(pg)
  # compute line segments from S to polygon points
  S.pg <- cbind(pg[,1] - S[1], pg[,2] - S[2])
  # compute orthogonal vector to S.pg
  S.pg.ortho <- c(S.pg[,2], -S.pg[,1])
  # check sign of products of S.pg.ortho and segments of polygon
  product <- sign(rowSums(S.pg.ortho * xy))
  if(debug) print(product)
  result <- "undefined"
  if(all(product == 1)) result <- "clockwise"
  if(all(product == -1)) result <- "counterclockwise"
  result
}
# check.sense.of.rotation(H[5:1,])
# check.sense.of.rotation(rbind(0,c(1,0),1))

```

By a little modification of `chull` the `bagplot` function has not to be touched:

```

128 (define another chull function 128) ≡ C 129,132
chull <- function(x, y){ # 121018
  x <- x + (diff(range(x))*1E-10)*sin(x+(1:length(x)))
  y <- y + (diff(range(y))*1E-10)*sin(y+(1:length(y)))
  grDevices::chull(x,y)
}

```

In the experiment

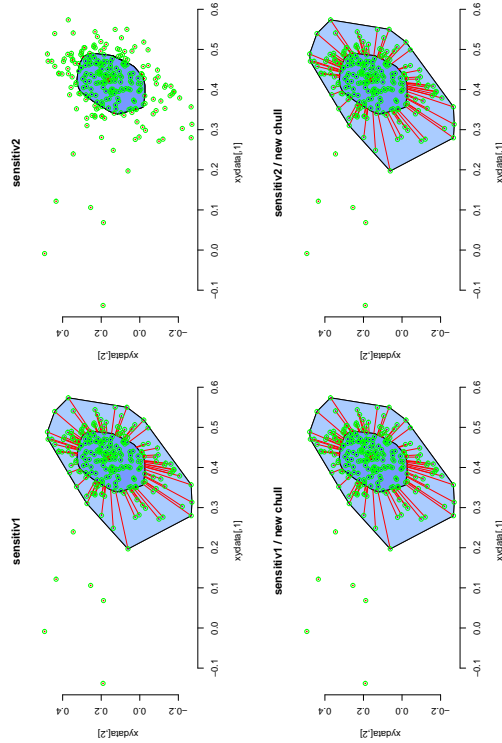
```

129 (define experiment to check chull error and solution 129) ≡
(define another chull function 128)
(define bagplot 32)
(check clockwise condition of polygon 127)
(define data set with chull problem 125)
check.sen <- function(){
  par(mfrow=c(2,2))
  bagplot(dat.AJS.chull.1,main="dat.AJS.chull.1")
  points(dat.AJS.chull.2,col="green")
  S2 <- bagplot(dat.AJS.chull.2,main="dat.AJS.chull.2")
  cat("\ncheck of bag:", check.sense.of.rotation(S2$hull.bag), "\n")
  points(dat.AJS.chull.1,col="green")
  summary(dat.AJS.chull.1-dat.AJS.chull.2)
  (define another chull function 128)
  assign("chull", chull, env=.GlobalEnv)
  bagplot(dat.AJS.chull.1,main="dat.AJS.chull.1 / new chull")
  rm("chull", chull, env=.GlobalEnv)
  points(dat.AJS.chull.2,col="green")
  S2new <- bagplot(dat.AJS.chull.2,main="dat.AJS.chull.2 / new chull")
  cat("\ncheck of bag:", check.sense.of.rotation(S2new$hull.bag), "\n")
  points(dat.AJS.chull.1,col="green")
  summary(dat.AJS.chull.1-dat.AJS.chull.2)
} # check.sen()
dump( c("bagplot", "check.sen", "compute.bagplot", "dat.AJS.chull.1",
        "check.sense.of.rotation", "dat.AJS.chull.2", "plot.bagplot" ),
      file="check.sen")

```

To start the experiment you have to source the file `check.sen` into R and afterwards you have to call function `check.sen()`. Maybe you have to open / to close a graphics device.

Here is an result on an 64 Bit linux machine working with R-2.10.1:



8.2.4 Missing Fence on 64 Bit Machines

130

(Amanda's data set of 29Oct2012: missing fence - A 130) \equiv C 132, 140

```
(define function get_A130.data() 123)
a<- " [1] [2]
-0.85542065713103743185286, 2.15115941176330727202526, -1.19208671093331264323467, 3.66774452282849505735385,
-0.44332994411219656116785, -0.21317176141080321216670, -0.95322582716228854149421, -0.79758373937012050358675,
-0.85479663725104515492603, 3.61175712324778164230565, -0.91268445485646854109518, 1.27949971450243737969288,
-1.02972242035120897796219, 2.62634358700534376040991, -1.06578451786594574635103, 2.40797744493886467509469,
-0.99462129329400283950235, 2.19461204176771085272435, -0.93615304864389947692160, 3.98785902813514248066440,
-0.87633319305915147179320, -0.82248044322106939052475, -0.91160849529685894498954, 0.71255600189825929469123,
-0.71553767273962853856517, -1.12608254028895471421379, -1.00089398608643542765151, -1.4028795852118815362587,
-0.89148998709359150716125, 1.46870994188889447684687, -0.36605692591031580018779, 4.16028792021881255180915,
-0.68595089215973226171030, -1.85146061828132202364827, -0.87627174984264033508197, -1.92221969981974849517314,
-1.14052717980054918456290, 1.33595462421871857827682, -0.88833580137564149836749, 0.96457122750992818627225,
-0.81932857201586739570587, -1.65177120947502698022902, -0.86658855088213104789219, 0.40855433537343288641353,
-0.73673765728454276846549, -0.49773687235231595105134, -0.88554141220049753524535, 0.86994647442387196267788,
-0.79360884407707521503994, 1.33197723079977392579565, -0.75110297829525662915984, -1.86429685214754492506017,
-0.88603320865447221521549, 2.16300940412966724579746, -0.92084115961151147278230, 2.42777488875512670318813,
-0.97846852292064179223985, -0.86081425133438205232750, -0.80200388417651757855253, 3.35789826563776561130226,
-0.82818022245386291313451, -1.41081320281005262451401, -0.97780624886553446639681, -1.96803970412010875712383,
-0.84138941333961292379229, -0.85419759131321548739635, -1.11248185778216912567018, -1.75047269918798797938564,
-0.90861092700168644142877, -0.03154563313117414447007, -0.91222915501874812793659, -0.16875175646029058618147,
-0.74125339243012500212160, -1.50712552710574421155343, -0.92081087305956699218257, -2.34957445097489436847127,
-0.8926536839120595723102, -1.58682233425998275855306, -0.64533249051768498252440, -0.63951970842675454065329,
-0.73960699012255393114401, 3.080523580437773111262453, -0.27640299765590881087007, 0.97581793030082419893034,
-0.73298122428301526465333, -1.13714950165858486030857, -0.79623011220256745268387, 2.63666014221451439070165,
-1.00835243220316029539040, -1.40945339148407344787017, -0.87478918062834831737007, 1.24675696471899977169073,
-0.93946891162959234033991, -2.23575887285715113605988, -1.04113028341499225248867, -0.43332693763433755007597,
-0.76258441597715431736759, 3.69076400280236782691645, -0.5783680958018817348335, 0.58236063011682748236097,
-0.69977456904717649788239, -0.42784078420383764296275, -0.93790621478827507218057, -1.23938814395473317908625,
-0.88101264064400186537540, 1.04036456759652584658227, -0.89565716239652615193734, -1.15213899494229899686079,
-0.88073661348396980108993, -0.46336430056633926000131, -0.83354137703608410792100, -0.27105160655942905911786,
-0.89622502713265317364488, -0.12507798123440422299879, -1.12563752613271184088717, -1.62167285494578172766467,
-0.65403324811386631676413, -0.25083729123204911992673, -0.79141996681884174691390, -1.75375362795426292805701,
-0.81342494079997351708045, -0.56944822640215031483990, -0.63545464068483314967750, -1.05479489085510191870299,
-0.96718903796691368945915, 0.98194230723463771237647, -0.71173633102162658925494, 0.08662285463906345406304,
-0.91245224633703412158445, -1.39957483199717036548293, -1.09725750173244129825889, 3.97178173324818706291239,
-1.21052268010369701300988, -1.48751954527648222453931, -0.81908108031154047701250, -1.2719836246134563719559,
-1.14606306248618672327666, -0.44462230904451466714375, -0.83951662234607338231029, 1.97383080095296681299999,
-0.69281568011474070623024, -1.08739186954697086839872, -0.92702765377606477503747, -1.906389338739440867857,
-1.08475990524972520923086, 0.54083768868734993784386, -0.74804814036824185397023, -1.42725217807927129420875,
-0.78518151235595234940707, -1.9972851298102271106683, -0.86017213803678183037249, -1.7315274779483395795808,
-0.9967062779066943048889, 2.16261737623493743853942, -1.00403022304053246571698, 0.24256266822290292739517,
-1.01467821900339827401183, -2.0711659495068092899260, -1.00247798450164249395300, -0.14913070069269826478653,
-0.71434503938257354338504, 1.16208712645395952068044, -0.56163412128444445414743, -1.04816397287585671982413,
-1.00919859412432599832243, -0.35783806034041643062693, -0.74532409405136224034294, -1.75345338759696312003200,
-0.99394705183593690112076, 0.82797220034547724942797, -0.94184002120392762424927, 1.90670649321686811738630,
-0.73298122428301526465333, -1.93964359388215057400373, -0.87255876098107965077588, -2.1638175275985094936370,
-0.62639084886192797352322, 4.3582363937103522138532, -0.57247573658732819890105, 3.54378219433118113457226,
-0.62315268461537154376373, 0.13862488040837206426659, -0.65419841565757663737202, 0.92369901723464797882457,
-0.03309816698677478519208, -1.355714097428032268442, -0.8777378708361686321382, 2.90815095470915130348999,
-0.69020108684955261413307, -0.99703789671200426791842, -0.74095694478300444529850, 2.10712418212813190621091,
-0.78311270974437130476531, 1.35943640789450159722662, -0.721928143672197117218, -1.16535975642339906244160,
-0.73980115615844443945785, -0.27422115308210284041479, -0.85631750269442230560912, -2.3658723635660908564263,
-1.04127739104801664282718, -1.92450576638652903760374, -0.80928298412162491537458, 0.72392040011114167086248,
-0.93583032355408235503091, -1.27167197741731108351360, -0.95138676011143852306873, -2.4835768309353754901281,
```

```

-0.69510873525639660375219, 0.74003425362884445171119, -0.70828292764230726952235, -1.74029456843447971614580,
-0.8170263695561475325624, -2.05636676767318204994694, -0.85572166595426168989036, 2.78217824845197725380785,
-0.86741128291380464876426, 1.12033186184798894124981, -0.96308575187273048889836, 1.82344575383575158156191,
-0.93159049564866858172962, 3.20834229131545622948352, -0.68151907982343462588659, 0.18221251081852365971692,
-0.92400177982538100884824, 0.87270298176576499571411, -0.97538875480163556019875, 2.18433079669054741600576,
-0.99919104266104008527094, -1.25971805320735641409158, -0.98120649708865659288648, -1.54488964063029277085093,
-0.80851827734517944801240, -2.31505004467299890791310, -0.85603194542064975358642, -1.41321565504162394155685,
-0.7732514446991402314449, -1.30092335913405610092752, -1.01047378468256954242577, -1.2172720297486130380041,
-0.38529689209950268580229, 0.12314176482201232010194, -0.71973981721237956232784, -1.56774140839910991473971,
-0.76005902450584117424626, 1.10837120469666472466486, -0.61064683153355137079643, -1.13939175278273729929879,
-0.76807357828877498295839, -0.00269127688095640183602, -0.89317762193375327406386, -1.42008903399852659532598,
-1.01254408420323827044740, 1.64412344792320319619705, -0.68277285288512268301986, 0.33100419868027375347452,
-0.81589562866355525017070, -1.99759904852324354784798, -0.58658679506311994789058, -1.46298614773908153274817,
-0.99174871126505703688991, -0.26864432863314841037905, -0.77828250249729868937010, 0.02858300289900294124945,
-0.99784814492361284532507, -1.01801682773381951818692, -0.51771175060428586967021, -1.42044092792102305544688,
-0.88420493651326714790173, -0.98797560290833985607861, -0.99105879193639900837098, 0.30988310585651729356371,
-0.91469596088505922271139, 0.20933151735940622173437, -0.6108599973417393071884, -0.58778480744151484493898,
-0.9608241284723209219985, -0.02760055760616408893071, -1.00908383642081078690467, -0.233421208304862154443678,
-0.53495137426864836083240, 0.67135401711554276715788, -1.00351606601209342173320, 1.61975151820372742506038,
-0.86163025367487566796854, -1.10313813559391427077117, -0.16595557271042252200921, -0.94943637498399602314691,
-0.79340636091193761370022, 0.85180196502530747260096, -0.98219121347453497428148, -0.351579571514637034823624,
-0.70171562990080948996052, -1.61987022813334813342578, -0.85380558130534844885062, 0.10234446964598327423079,
-0.91119999758376091225642, -1.96270424211922911439387, -0.47369410497787611813436, -1.24568659795105607379639,
-1.12481208976129720689130, -1.42955649956258801225317, -0.68591498029136899149424, 0.240869703295181866950679,
-0.76159739005818971602224, -0.40384873023512818379643, -1.09215856416705370257603, -0.947838832204756176480409,
-0.69009417166443720592639, -0.04304150012349711174053, -0.94715770497794005144954, 0.3648403035844337860308,
-1.13547332040171000677731, -1.82492894859917007869535, -0.82569383227340431563590, -1.31765917652509001456451,
-1.0754933888194038651420, -1.42255779965187789670722, -1.03635341123188906564678, -1.35490492090438774397398,
-0.94837153414965524866886, -0.79764386660595165690069, -0.70324574408490092558566, 2.72592699650955738732705,
-0.5993247087966066565505, -0.4886406257862583313117, -0.72779961394791581330566, -0.56117886859764154472430,
-0.79714090059361974583663, 0.56048697226773325219540, -0.68858719457172146150015, -0.26803283834927038586926,
-0.837543471621692962402838, 4.94926611751849154785532, -0.69987160072848875191909, 2.0067931594935002373654,
-0.82674205334838080094784, 0.35773157027152419074056, -0.86219755182467683239861, -1.9970326367585459733259,
-0.7646548238960143090281, 0.32498902518557637453256, -0.86168744646692981483438, -1.18794597298212489810965,
-0.75492869446957344692350, -0.58968771620255122645204, -0.89084522505621299703904, -1.79451696945702754426577,
-0.62097653530934382448692, -1.42116887519834378394989, -0.78279898917940027303075, -2.09769826909406864601237,
-0.58614497755964112268856, -1.14251491700111573734944, -0.65548450442827721484917, -2.28887318172734973661520,
-0.72184550479257536537148, -1.55466888276810100144587, -0.79986165711412759993237, 2.80460158305232143405306,
-0.71739738939399744221959, 0.85086961179459741355657, -1.04653794258775967307429, -2.00828362990076847438559,
-1.02952932442522979622964, -1.34703054251816567443711, -0.84839149177916028943258, -1.58941391932855591306861,
-0.84668604625847099232061, -0.60278827664105838479713, -0.7546871007765300256645, -1.43666023259784392251959,
-1.054924017237889382909430, -1.55158780120198191987413, -1.08702182244534495403343, -0.65105985980144276403792,
-0.842563497347229526251789, -2.40118403419167103507448, -0.71193792804248645644805, -1.17579565519432251826629,
-1.00601134978073059045300, -0.8353767965677819037217, -0.28880018017186870338264, -0.087706197600356510617,
-0.54646188412085971997101, -2.04181536457609746904041, -0.90125240262175132510691, -0.25539757411815977583558,
-0.78238616974000174675297, -1.1474856483555817545316, -1.12352874453347850725038, 0.62930084060831015246862,
-0.72532662485869570101471, -0.7229257598666363193729, -0.92110061861498138124915, 2.9508745239893245974237,
-0.8188575448996671608143, -0.7229412113608401064135, -0.8285179023519451156496, 1.45563836704469151683838,
-0.56375204548899770565612, -1.73786842359159754423104, -0.74942607595237131778987, -1.18435169808617657416505,
-0.7134956374569435758588, 0.29969133883730475487184, -1.06584931484566491910471, 0.51899942217620631534913,
-0.13249578359112310077528, 1.44275179489447413772041, -1.15013113475822548181782, -1.09809126239714527315055,
-0.85330821795335043411512, 0.74514093268018632976180, -0.95020073072400834668372, 0.03186078936976277642401,
-0.84799717846286148947854, 3.816625600197176932968321, -0.73845273625461438360418, -1.15803745550225345439799,
-0.75228416807307929570214, -1.96716954625766615727400, -0.73009804114494858406914, -2.63485854127420449088959,
-0.6348300413942239192982922, 0.83783234926001504128124, -0.85473511925759448892848, 2.40174982350467525193949,
-0.83557219788865222387386, -1.93698057434386350372790, -0.96668634873506098514895, 4.79747892203358183849105,
-0.713128942768651596928464, 3.44713277528197181709402, -0.70391582361987770077860, 0.70876197175452304577448,
-0.78019639366632394801752, 4.32977199947720503558912, -0.81222108274463133970045, 1.98235848995349450696324,
-0.6253633303956388505663, 2.89592538144002498157192, -0.84993647888927637001899, -1.40500895686213089774697,
-0.7790226224902587409878, 1.1388862205232106450712, -0.80604966503166086688736, 0.45811820144298964763152,
-0.81473457118633252349582, 2.63778886293564251985799, -0.72605809403965193560282, -1.36019627870992088070068,
-1.09745982191787994075582, 0.30125076379655502112342, -0.9137204858967978848382, -0.03755168934339410402590,
-0.35209683987193424181683, 0.03282954100772271510689, -0.99817961036730040724763, -1.55310044858275242418699,
-0.81094257708929540218179, -0.38561313655228945940223, -0.8078650323236011511443, 2.91418986509012523100637,
-0.5894288466778417006114, -1.16586165844052636053618, -1.072952517468944848565717, 1.72786967610222830238342,
-0.71276357918832543347776, -1.34349395653242686954343, -0.75844877067192684183539, -2.24887328562919108776441,
-0.850017100546454984726562, 0.47108272238129339504198, -0.71349267860304776966984, -0.96739640833556916899028,
-0.71101933649891457811520, 0.47108272238129339504198, -0.71349267860304776966984, -0.61834827276289272557364,

```

```

dat.AJS.fence.l <- get.AJS.data(a) %>% bagplot(dat.AJS.fence.l, main="dat.AJS.fence.l")

```

131 (Amanda's data set of 29Oct2012: missing fence – B 131) ≡ 132, 140

```

(define function get.AJS.data() 123)
a<-
0.488565925036412662674223, 0.4788830230285979205895330, 0.470518882899528423369162, 0.4755379903005585995323656,
0.540821318900958192266160, 0.3452584612790277618366019, 0.543880329284483932106298, 0.2321764386212968045430927,
0.334695846810999286802257, 0.2329343592006747343248207, 0.31005377685083016725542, 0.2738714764796457146189826,
0.472091749674024396821181, 0.3495517610383970796839037, 0.470453133990287764021332, 0.4563524147511247019970452,
0.239437538963621576737495, 0.3450803438560707259163962, 0.407605742194999509653286, 0.2770631606622363518610541,
0.405297909936674971564941, 0.356403897000368745968707, 0.426698946910272269850140, 0.2959717498266932200756685,
0.474078974951626763711943, 0.0632119081706708912937742, 0.518165669459433098076317, -0.0208026309734368752835110,
0.389182699194088488336973, 0.3093662369282451640728482, 0.386863379163557341566815, 0.3639162011488026871042223,
0.425943786936054447078703, -0.0809216837347255524282019, 0.397113987796910483663027, 0.0008999279456975259501583,
0.438144515529690747879243, 0.2093360857799788921074224, 0.328446751697633210209659, 0.2751362229330584496800327,
0.461876192480900560344281, 0.0668544125659962518160029, 0.414702341447854616607316, 0.007323610725386329605292,
0.432106260664682051420016, 0.1879619488777506675081241, 0.395722604485037765531530, 0.1531447992834467641820595,
0.333445239047859665060258, 0.1965294920673828216184376, 0.401681093963615620801022, 0.094684285978472339746640,
0.400866278174864976158176, 0.1471984007921947890107361, 0.464762400209403125916907, 0.2009867588029477236677423,
0.54978816087283665226895, 0.1278160114450895834536936, 0.381209639768359198619407, 0.2668751891155876632133470,
0.550009152816997137769306, 0.0668940682809511938389235, 0.477231632689670060010201, 0.0741204859518660857942152,
0.485550418959452921185971, 0.0776261286842003100927556, 0.528876577973242389951736, 0.0949198625156014996750997,
0.486467492116636091026294, 0.1845373489642387898612697, 0.418601287671568234038944, 0.0991622310369848491973244,
0.43144464527195108109964, -0.0954361392183528528532221, 0.410791284509544551983851, -0.1344827904070514668077863,
0.39578126984565221856432, 0.1228690809909824266377143, 0.469332442969894070294856, 0.1508254155849814148737664,
0.503409405380703213913307, 0.185064660302889895457752, 0.573583529063881991660878, 0.3707267482610320796787052,
0.539540160667456469845149, 0.4411587741270995288726645, 0.457799411020661739524229, 0.446425282522002939700261,
0.389979310257221101210234, 0.1327572439545967042384689, 0.395014994054688040048262, 0.1097540690978915517428405,
0.43914913163883134705761, 0.4193625991526864771330452, 0.12156835584936352298264, 0.4339816580876698637325717,
0.33545096887692048068459, 0.2473479868463136777112510, 0.351411588546627129403532, 0.2246294926689241799877550,
0.506931530664690099818870, 0.2389686803487043320615157, 0.416453890776864510403499, 0.2150714006960237900667465,
0.44572830669456675326302, 0.21025989831373535936889890, 0.466270712549676169533797, 0.0843892346292415551634036,
0.487207645108102174345531, 0.3059151602998300689684186, 0.469777318263590537572583, 0.3957816521807506204844174,
0.377067896799580637434701, 0.1590144819105466233999380, 0.453149790799679430541858, 0.1215962091891261420695258,
0.487757526130484675430665, 0.373218067636979779034971, 0.440667092149550854163209, 0.4203912649541739487979441,
0.408446800624878980645605, -0.1618030914002129283524312, 0.384406712559258345596191, -0.0261348063728542405470545,

```

```

0.328457377392783189673509, 0.1821870523887156312348168, 0.331531071874300753421494, 0.2324458686458765466209542,
0.476092076084575321903714, 0.2588599611500894681626050, 0.433767577422285610655450, 0.2835973312618549346808550,
0.393793741503250860347407, -0.1773741681645460421812288, 0.375897237966888619897787, -0.0703216253522565698519387,
0.495413425141699670373185, 0.3006890687220556412917460, 0.504610944570432873312882, 0.2887805058143382841429059,
0.431192387166821511357284, 0.0947182526586476392083114, 0.340937433712147242026447, 0.1329507636456480346165421,
0.420446411323254720660003, 0.0483223700436425988572254, 0.370470482089916230972193, 0.0277816648686944184141782,
0.441037167199129720440709, 0.2107626360386905817190240, 0.419959107781044338469911, 0.2824900723710290773027509,
0.439763006956421009796543, 0.277822223877927499202656, 0.464372875417278285770095, 0.3177378989007230702945606,
0.276309247060232576753691, -0.1202539041759453086744003, 0.298239821897850698739774, -0.0296142651449498549298678,
0.395415764674621084129313, -0.1763745879581422593140871, 0.347422594070579682234978, -0.0848076458082471090760279,
0.339350476740504836126178, -0.0695299708858954784718165, 0.414959369029142932649989, -0.0048385120265832422259411,
0.459727233012242109122525, 0.2059384488751775654780118, 0.499622833076944272701070, 0.1055938665159665228750910,
0.440380490523102519073717, 0.0351622931865073623525220, 0.482813054408696429309344, 0.1113387917612616989648799,
0.432127973903995210580575, 0.2362151044188228743436753, 0.490176416856921814968473, 0.2595152506103633460732283,
0.478778121283525648355095, 0.2342718274692105751011439, 0.464692711758631471496983, 0.2229430092565458654263466,
0.391963528582289721935439, 0.1503685476363093853624520, 0.415382188531344009874147, 0.2260871923819901541108379,
0.460109318047086657266220, 0.2820022326407002344339503, 0.419856322790740588146718, 0.1560030888975831031562080,
0.439414518877366266913498, 0.3699970203676556934624386, 0.480511782736342096811200, 0.2949392132900372308931139,
0.437066976232192427875844, 0.1592860166985547983084848, 0.476422133435934369583009, 0.1429539737753039441159331,
0.271275837177732004605864, -0.0995991440991172982677426, 0.353014229438460613863526, -0.1305150420251256770942661,
0.357543712693103532540562, 0.1943096579024176095984444, 0.457267580136557938441655, -0.0176971730175275836216553,
0.427558336829375820969022, -0.0231909551915799178045674, 0.432525049981503817431161, 0.1515661841784528630316942,
0.451225648652902766588824, 0.1501701067714517889850612, 0.361330738603812651188463, 0.1264174670413595047424581,
0.43521689409707547178405, 0.1009900091897135044925801, 0.470581162898296934127274, 0.1851202194385961619094161,
0.437460030660461018925389, 0.2532716932405828513807933, 0.417040531646397594212061, 0.3302983395435944435902798,
0.38455064931715928633338, -0.1573336822941461821123710, 0.371318092688570966508621, -0.1093708133581914876408803,
0.4212984862269021778989582, 0.1425323554856109675714748, 0.495736478252773005159781, 0.0403531004651603228405854,
0.302958384980159090016372, -0.2206251139096475355483307, 0.279767950817931709828201, -0.2667856263934703653362135,
0.248455666945173819781800, 0.137268270651706586471287, 0.281703006061099270329606, 0.0035824033289076563819908,
0.475976318661099573326112, 0.3917064509029100638493048, 0.401985224405800756208862, 0.072329212443154017302682,
0.503965946523464188487938, 0.2849822455257227660219144, 0.462099158946540189418783, 0.18650735293258088867602278,
0.451960176846561534347302, 0.37308572261547440039088883, 0.484354627447674712659165, 0.1001666314526957063391066,
0.470928061192341984586562, 0.28081999260613693669700, 0.359374566259886285557457, 0.2325365031862380305049953,
0.4949885710840891830366, 0.0783343356748880115247857, 0.459811454891438209369170, -0.0437362500299243728285603,
0.43976406579993748309221, -0.008866044667374890875244, 0.491099424456228894619869, 0.3022186062741255008212704,
0.41872173164468675271829, 0.1857572104407961555150308, 0.390276221729421390893577, 0.3154919031871089019070098,
0.39946945290136929174139, 0.0498277648167823605795945, 0.454754096854184497455975, 0.30384046278302212495595866,
0.435410152674876649303570, 0.1737382459751893870603112, 0.44586726278628525510728, 0.3164638890784514102527680,
0.4589675299975148414838564, 0.1518767403682781125251466, 0.335919343619362931185890, 0.0823980294588803341475884,
0.41276392083080666176957, 0.3319087518713340068110540, 0.471049431246630845304679, 0.2156398495278842105746321,
0.400699272940918671537247, 0.1395100301705456957712670, 0.374861657961513439207835, 0.417994049726512318199134194,
-0.138108934234092650683579, 0.1904884368528304006140672, 0.280570721977494219601112, 0.1983537836884063931819355,
0.42012027711559659364408, 0.2619778732066581627968560, 0.454057068735585156460388, 0.3034526292199665742366221,
0.395791684619125783139992, 0.0050987951562562298934989, 0.365260424959625518148698, -0.0272468778624029156487651,
0.484034990799743880529604, 0.3569887088322822310537674, 0.37086316510424204197705, 0.1093298911135483775236695,
0.461277752786707073706651, 0.2632357520089478875036093, 0.350125222763810362014425, 0.0183460251321451266182105,
0.368930134538406506727881, 0.270077738108504150726930, 0.531849229752716645691635, 0.21978694956693037299011405,
0.416717917131823767373788, 0.1982293252409910999656262, 0.414321880482149074165221, -0.0860104879183531539910135,
0.438841302318247195035639, 0.0510552491055184659574095, 0.398298536249736545578060, 0.0872398037610459115009931,
0.306217635368917995286608, -0.0149914778112886330763143, 0.469373833730254985674435, 0.1709017284625866106573265,
0.41592594466704966038151, 0.1800595545698811805745976, 0.385559534967464867527553, 0.2280161637862092305315699,
0.44504786154811862196028, 0.1861824345291962623516469, 0.450436609467876891699234, 0.1659285616769800852933093,
0.105991451516687498757285, 0.2547355454603117408396429, 0.401561850053824997974772, -0.0054767733163579414637051,
0.068325138187451131543959, 0.1882970253456990872287236, 0.428084181655836837343543, 0.1258825829068733470617047,
0.386663185037617151973421, 0.2788034267395078646956108, 0.364941455716851159696290, 0.1938777480575513967320234,
0.388006006739891329271330, 0.2488389433599013378373144, 0.313331553203334722823570, -0.270434840057152514641028,
0.35292280801725303354033, 0.0215689704506152803417162, 0.448430163284227911724145, -0.0917932666811329678324682,
0.431449464825348683039152, 0.350800086583910870598996, 0.432438166022204639737936, 0.1241467238586335397609872,
0.428358234301227214224639, 0.0088965389495472245451335, 0.451061146935774315291923, 0.1955063903810788650261543,
0.43870605389136660862046, 0.403386057135294143722200, 0.375774923167026320136586, 0.1732898787280841290669997,
0.519488312493662673929862, 0.2018627369504666646538027, 0.440325399842820874862781, 0.1348568173426236971312875,
0.490158077485978449505666, 0.3545691684017332478973117, 0.504032360432461246091407, 0.2314774746120594750564692,
0.46533574286672471964721, 0.3347213235409923770546925, 0.351902300975328263810127, 0.32821325580308874370066129,
0.48704515369814741125380, 0.090252839819727559462322, 0.429961564597323653291738, 0.0480713260550880150390185,
0.437203684433824457666873, -0.0127309372130392176730718, 0.469321808013436481132885, 0.2620019309451309674763309,
0.40517573929584838768694970, 0.1902570512550103010163127, 0.464512600773935580011909, 0.2275906537161559162196767,
0.396079698974603044980825, 0.1754275457698743490020377, 0.441935381229888457621513, 0.2202391505171113483818601,
0.464574093329662873852470, 0.2507926308999658404630395, 0.483547025428987031769879, 0.2672255285437769023459964,
0.373521032078406201026866, 0.0323174728928018809015477, 0.350304983220608617948824, 0.055592693927829464617396,
0.495803893669895323948690, 0.202370191782240878849730, 0.436908706786744593930649, 0.1397339520367077425611058,
0.420933303726012564283110, 0.1287640020024260755326129, 0.33357747449375381037197, 0.1964571049898150012502640,
-0.008696645322344730144604, 0.4936956907050396246639134, 0.498882187348169858776004, -0.042987211072758799386677,
0.4046811939578312264297, 0.2022239917636600992079110, 0.419736248847867643441134, 0.1800517691346032178589809,
0.34175405915502841234144, 0.0975732594081305842834703, 0.3519058043459053090342934, 0.0467274302256285301027106,
0.510698280023458961451865, 0.236967537004825731410449, 0.319367929544649598039285, 0.0144749589516537692512888,
0.387776485196835801223415, 0.1468031004190566979339344, 0.363127251904536374471633, -0.0183052364380072371541353,
0.463043824984787411036535, 0.1398346426667024767755976, 0.485445195724856781716028, 0.2410354072978813655403485,
0.351766303400639024356877, 0.1236472951599203590200560, 0.378860874310987749691293, -0.2068662281780968004163412,
0.50654897635976064433453, 0.0257762531349018247928129, 0.462348947369779106342236, 0.0841091086286257227921936,
0.1970603815222191130354571, 0.0610563781132145988461701, 0.369159733288869140732658, 0.1978140552312874778095875,
0.437006678117996483212693, 0.17827006582132221037106490, 0.353860892739909282944666, 0.1619473817349794397824070,
0.370839258480037510294857, 0.118400290451527489522903, 0.50140376954722707148410, 0.1153902428064478985492158,
0.463520128314569346272833, 0.0589935729341639457867785, 0.275594489591146629692275, 0.023306881735083951068255,
0.401167105521437583615096, 0.2525394059250221090451305, 0.429061978765738316621281, 0.1794993730327074099584195,
0.353765868289411034020020, 0.2162066871434025205900298, 0.356755335938158446573709, -0.2663981226678143188202344,
0.342817464803536374222432, 0.040648544162907696090572, 0.40582628844388227905080, -0.1289384243139815866552311,
0.475344946440783899799243, 0.3477611725526632557858875, 0.512683743910172906588230, 0.1740133571079716390261893,
0.388901597797993381355752, -0.1687308736650941221046907, 0.416404996762255230624561, 0.0422453698295842117182985 *
dat.AJS.fence.2 <- get.AJS.data(a) / bagplot(dat.AJS.fence.2, main="dat.AJS.fence.2")

```

```

132 (define experiment to check fence error on a 64 bit machine 132) ==
      (define another chull function 128)
      (define bagplot 32)
      (check clockwise condition of polygon 127)
      (Amanda's data set of 29Oct2012: missing fence - A 130)
      (Amanda's data set of 29Oct2012: missing fence - B 131)
      check.fence <- function(){
        par(mfrow=c(2,2))
        (Amanda's data set of 29Oct2012: missing fence - A 130)
        (Amanda's data set of 29Oct2012: missing fence - B 131)
        R1 <- bagplot(dat.AJS.fence.1,main="dat.AJS.fence.1")
        cat("\ncheck of bag:", check.sense.of.rotation(R1$hull.bag), "\n")
        points(dat.AJS.fence.1,col="green")

```

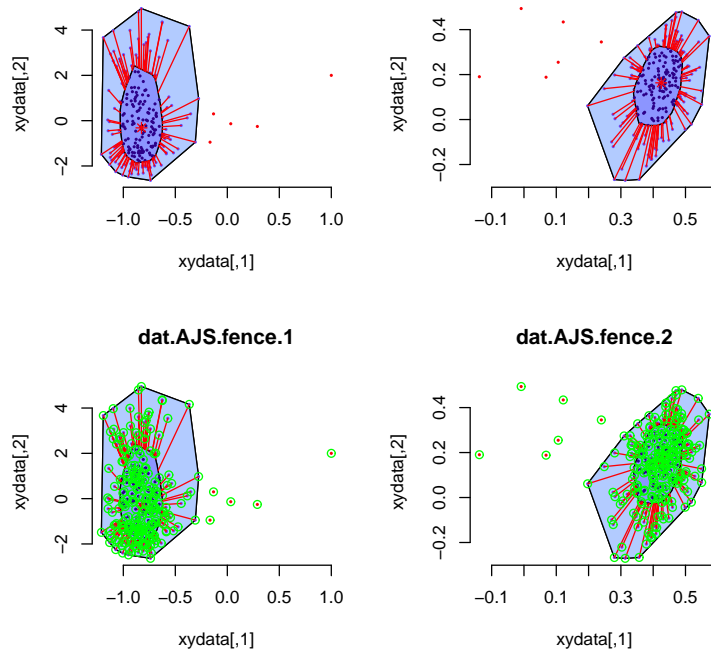
```

R2 <- bagplot(dat.AJS.fence.2,main="dat.AJS.fence.2")
cat("\ncheck of bag:", check.sense.of.rotation(R2$hull.bag), "\n")
points(dat.AJS.fence.2,col="green")

## summary(dat.sensitiv1-dat.sensitiv2)
##define another hull function>>
##assign("chull", chull, env=.GlobalEnv)
##bagplot(dat.sensitiv1,main="sensitiv1 / new chull")
##rm("chull", chull, env=.GlobalEnv)
##points(dat.sensitiv2,col="green")
##S2new <- bagplot(dat.sensitiv2,main="sensitiv2 / new chull")
##cat("\ncheck of bag:", check.sense.of.rotation(S2new$hull.bag),"\n")
##points(dat.sensitiv1,col="green")
##summary(dat.sensitiv1-dat.sensitiv2)
} # ; check.fence()
dump( c("bagplot", "check.fence", "compute.bagplot",
        "check.sense.of.rotation", "plot.bagplot" ),
      file="check.fence.R")

```

Here is the desired output of the chunk.



8.2.5 Further Na/NaN-errors

```

133 (Amanda's data set of 29Oct2012: Na/NaN-error A 133) ≡ C 141
      (define function get.AJS.data() 123)
      a<- "
      -0.5287511, 0.29140334, -0.5491688, -0.05484604, -0.3919317, -0.11957324, -0.2342807, 0.02022132,
      -0.3921559, 0.29079326, -0.4158044, 0.18658202, -0.2871524, 0.05062737, -0.3617926, 0.29075199,
      -0.2892480, 0.05695156, -0.4405352, 0.12075804, -0.3838563, 0.03978882
      "
      dat.AJS.NAN.A <- get.AJS.data(a)
      bagplot(dat.AJS.NAN.A, main="dat.AJS.NAN.A") # OK

134 (Amanda's data set of 29Oct2012: Na/NaN-error B 134) ≡ C 141
      (define function get.AJS.data() 123)
      a<- "
      0.33817, 0.02979,0.32204, -0.08801,0.29517, -0.25250,0.22258, -0.08154,0.26135, -0.08324,0.34717, -0.11121,
      0.35933, -0.19076,0.38608, -0.14942,0.45037, -0.12851,0.04225, -0.16722,0.29133, -0.04328,0.33022, -0.09896,
      0.20654, -0.08464,0.28204, -0.04508,0.29815, 0.14699,0.30624, 0.12807,0.15477, 0.03248,0.32912, -0.07302,
      0.30138, -0.08916,0.18917, 0.17526,0.23798, 0.02009,0.23924, -0.14238,0.17265 -0.11008"
      dat.AJS.NAN.B <- get.AJS.data(a)
      bagplot(dat.AJS.NAN.B, main="dat.AJS.NAN.B") # OK

135 (Amanda's data set of 29Oct2012: Na/NaN-error C 135) ≡ C 141
      (define function get.AJS.data() 123)
      a<- "[1,] 0.3786439 0.097738688 [2,] 0.5362123 0.339630360 [3,] 0.5193582 0.157707944
      [4,] 0.3338111 -0.115373436 [5,] 0.4582097 -0.043625237 [6,] 0.4921440 0.074836745 [7,] 0.5218369 0.102194532
      [8,] 0.5172927 0.137981797 [9,] 0.5433263 0.211029679 [10,] 0.4827819 0.015513368 [11,] 0.3460246 0.080163464
      [12,] 0.4494423 0.166911911 [13,] 0.5122454 0.065798073 [14,] 0.4136952 0.162839099 [15,] 0.4979262 0.083749028
      [16,] 0.5058579 0.057943131 [17,] 0.3867839 0.175866756 [18,] 0.5063583 0.065406168 [19,] 0.4807745 0.201581462

```

```

[20,] 0.4599295 0.177264912[21,] 0.4500260 0.105345954[22,] 0.4336654 0.197944371[23,] 0.4406115 -0.060430011
[24,] 0.4492211 0.109822612[25,] 0.5160831 0.015950560[26,] 0.5043602 0.066780298[27,] 0.5465611 0.145911903
[28,] 0.5541721 0.134060064[29,] 0.4727388 0.006861838[30,] 0.5613212 0.077141184[31,] 0.5369902 -0.024873891
[32,] 0.4378382 0.122520785[33,] 0.4252500 0.093480024[34,] 0.5039986 0.075154013[35,] 0.5216457 0.196010331
[36,] 0.4117548 0.035776717[37,] 0.5040175 0.143153339[38,] 0.4845090 0.136095252[39,] 0.2075045 0.154560697
[40,] 0.4061658 0.250111568[41,] 0.4942135 0.178913967[42,] 0.5407099 0.008868236[43,] 0.5317494 0.095625935
[44,] 0.4559603 0.135323999[45,] 0.4634133 0.180307104[46,] 0.5268978 -0.041798329[47,] 0.5831482 0.127251165
[48,] 0.4120890 -0.010194722[49,] 0.5693523 0.057730316[50,] 0.4794066 0.129579450[51,] 0.5242518 0.134452331
[52,] 0.4402205 0.097278607[53,] 0.4071411 0.115581565[54,] 0.5605076 0.094539922[55,] 0.4889528 0.155507716
[56,] 0.4570601 0.046582559[57,] 0.4699092 0.163047744[58,] 0.5025529 -0.019894103[59,] 0.5441058 0.218039267
[60,] 0.4621933 0.095914941[61,] 0.4115889 0.154296680[62,] 0.3124742 0.118646650[63,] 0.3777633 0.177376132
[64,] 0.4529856 0.316883371[65,] 0.5279446 0.085011958[66,] 0.5126856 0.142196509[67,] 0.3278320 0.199462024"
dat.AJS.NAN.C <- get.AJS.data(gsub("\\[[0-9]+\\|", "", a))
bagplot(dat.AJS.NAN.C, main="dat.AJS.NAN.C") # OK

136 (Amanda's data set of 29Oct2012: Na/NaN-error D 136) ≡ C 141
(define function get.AJS.data() 123)
a<-" 0.4015263, -0.26233503, 0.4317795, -0.20336274, 0.3630288, -0.17533677, 0.4665716, -0.43077830,
0.5019500, -0.16576077, 0.4465282, -0.22522654, 0.4564547, -0.19561030, 0.5397231, -0.05622721,
0.3168710, -0.18490364, 0.4637404, -0.31444370, 0.2084304, -0.18378842, 0.5039862, -0.27860019,
0.4686724, -0.42776110, 0.4710615, -0.40642947, 0.4341304, -0.23033476, 0.5164032, -0.18469189,
0.4881164, -0.14362839, 0.4331307, -0.32284998, 0.3839510, -0.01538800, 0.4679067, -0.42500528,
0.4221514, -0.26589452, 0.4611424, -0.26542121, 0.4804025, -0.51445482, 0.5115250, -0.45830031,
0.4571291, -0.32162648, 0.3930053, -0.39147648, 0.4628674, -0.07719377, 0.4653781, -0.27503555,
0.4393759, -0.22965847, 0.4964249, -0.49612750, 0.2991952, -0.19227342, 0.4852176, -0.41668100,
0.3462163, -0.18303605, 0.4071043, -0.15098413, 0.4252945, -0.38060330, 0.5436867, -0.33408535,
0.4869311, -0.39235895, 0.4441376, -0.52603457, 0.2704323, -0.30335612, 0.3871155, -0.36949791,
0.3883343, -0.35457602, 0.4627015, -0.10874423, 0.4425959, -0.48179784, 0.3449825, -0.29517753,
0.4584851, -0.02167414, 0.4887845, -0.25134534, 0.4651791, -0.29775759, 0.4767561, -0.29090887,
0.4342507, -0.03061359, 0.4245398, -0.49428349, 0.5156233, -0.21271206, 0.4240498, -0.45404050,
0.3868688, -0.27232066, 0.4491924, -0.41249646, 0.6022689, -0.45291968, 0.4623975, -0.33215390,
0.4475317, -0.27044472, 0.4851270, -0.41659817, 0.5006870, -0.47004304, 0.4542369, -0.31246866,
0.4392972, -0.35501689, 0.4987533, -0.17762899, 0.4711685, -0.32507225, 0.3651114, -0.12669620,
0.3379918, -0.34953533, 0.4950965, -0.34470014, 0.3751116 -0.33303722 "
dat.AJS.NAN.D <- get.AJS.data(gsub("\\[[0-9]+\\|", "", a))
bagplot(dat.AJS.NAN.D, main="dat.AJS.NAN.D") # OK

137 (Amanda's data set of 29Oct2012: Na/NaN-error E 137) ≡ C 141
(define function get.AJS.data() 123)
a<-"
-0.4430864, 0.4296289, -0.5519971, 0.4087558, -0.2946034, 0.5730177, -0.4017741, 0.3810218, -0.2382924, 0.5786751,
-0.4904851, 0.5480627, -0.4942082, 0.4333985, -0.5649365, 0.5723637, -0.4574408, 0.5466591, -0.2514772, 0.5538926,
-0.2580041, 0.5321808, -0.4294454, 0.3233148, -0.5088054, 0.4833570, -0.4557215, 0.4735297, -0.2792391, 0.5351567,
-0.3330553, 0.5655698, -0.3485064, 0.4550155, -0.1443746, 0.5873422, -0.4588404, 0.4143949, -0.3619520, 0.5101329,
-0.5700096, 0.4911063, -0.4136021, 0.4748494, -0.3843874, 0.5453594, -0.4944427, 0.4202887, -0.4823245, 0.3963911,
-0.3683131, 0.5664984, -0.3274923, 0.4623527, -0.2061843, 0.6071268, -0.2989706, 0.5654522, -0.4170459, 0.3988646,
-0.5705104, 0.4594555, -0.4143036, 0.5280171, -0.4130142, 0.5165166, -0.3615246 0.4938318 "
dat.AJS.NAN.E <- get.AJS.data(gsub("\\[[0-9]+\\|", "", a))
bagplot(dat.AJS.NAN.E, main="dat.AJS.NAN.E") # OK

138 (Amanda's data set of 29Oct2012: Na/NaN-error F 138) ≡ C 141
(define function get.AJS.data() 123)
a<-"
-0.4062338461342518969310, 0.14754908336994210227289, -0.4641202518436023383153, 0.12300598652603220162227,
-0.4930885230817930175995, 0.13041599158936770241901, -0.4908548984150178373653, 0.1904203983535431299323,
-0.4413398310320104483826, 0.06415051270490622348230, -0.3790487599768575521786, -0.01908092520419982923707,
-0.4525924267924343324943, 0.02382016149426484027951, -0.4568684921740411852831, 0.10729137283566912708377,
-0.3600453131384418470340, 0.20299195732165442596084, -0.3786537514469233700609, 0.10782855344122767304871,
-0.3691294180636163213549, 0.13757254829898393766463, -0.3634668772645677226052, 0.13779499172315048949322,
-0.4460341706072293299634, 0.08927344826865776794556, -0.3931134284548397639369, 0.12375939206240904599809,
-0.4400074650713454715856, 0.13572835300482041787085, -0.3687761227741535030589, 0.08977980927726944559986,
-0.4477093868839493451262, 0.07732543881534685581425, -0.5047983314975119664369, 0.12008152848553328706505,
-0.4912126267418248093399, 0.06559481278535840564903, -0.3162564550157994092139 0.12716570436622953721439"
dat.AJS.NAN.F <- get.AJS.data(gsub("\\[[0-9]+\\|", "", a))
bagplot(dat.AJS.NAN.F, main="dat.AJS.NAN.F") # OK

139 (Amanda's data set of 29Oct2012: Na/NaN-error G 139) ≡ C 141
(define function get.AJS.data() 123)
a<-"-0.3365320800051156413524, -0.16465634271153567480539, -0.3212439795751604876273, -0.02769957746625343469882,
-0.3131092390672871039747, -0.02145017540830633506754, -0.2169780789017450306488, -0.05420179428591438003382,
-0.2382998530537111248062, -0.15665613626385860301937, -0.3274251750565500551637, -0.16269511294063973561030,
-0.1792897122532586817734, 0.12172548044581302240097, -0.3644874055796215595038, -0.1101711309932566383826,
-0.2597330667315013164043, -0.13667626910135124984613, -0.1857087775577647981162, -0.12185261831753121941624,
-0.4410627956771460689289, -0.24517450662929113347488, -0.4328369531394934965896, -0.09192268795018787579845,
-0.3162373978549153918927, -0.09289939950677550406510, -0.3527203678927318675207, -0.04575999789018787844430,
-0.3072802538232928259987, -0.15452276450236571148089, -0.1900264843388008995095, -0.19329920713585954650249,
-0.3581798974648294220380, -0.18092585691592122376647, -0.4087284209462124784373, -0.07075846280518945097260,
-0.3143274825737536470882, -0.10093548347883234128641, -0.3566729290036041999379, -0.07337318053477877299873,
-0.3809771622860922968279, -0.13936364479847815345259, -0.2029868237921297058346, 0.08934781459090822275382,
-0.247655195700060755210, -0.07158142266137049181118, -0.3710390854098435942099, -0.01787010019324387563588,
-0.2911476980688867954861, -0.04232827341335163723324, -0.2253901616470012636562, -0.25359051915261998644269,
-0.1347541765844346151049, -0.04372012527641970514036, -0.266559637370596738894, -0.08639269553985037819466,
0.5589372043228539865822, -0.05442746424681436240300, -0.2892559615112646476121, -0.15214455887285957547128,
-0.3248426515933379166157, -0.07294253882500667529598, -0.4280514652785712970129, -0.04237105698433304284967,
-0.3992184363626437582084 -0.035101674888593822493481
"
dat.AJS.NAN.G <- get.AJS.data(gsub("\\[[0-9]+\\|", "", a))
bagplot(dat.AJS.NAN.G, main="dat.AJS.NAN.G") # OK

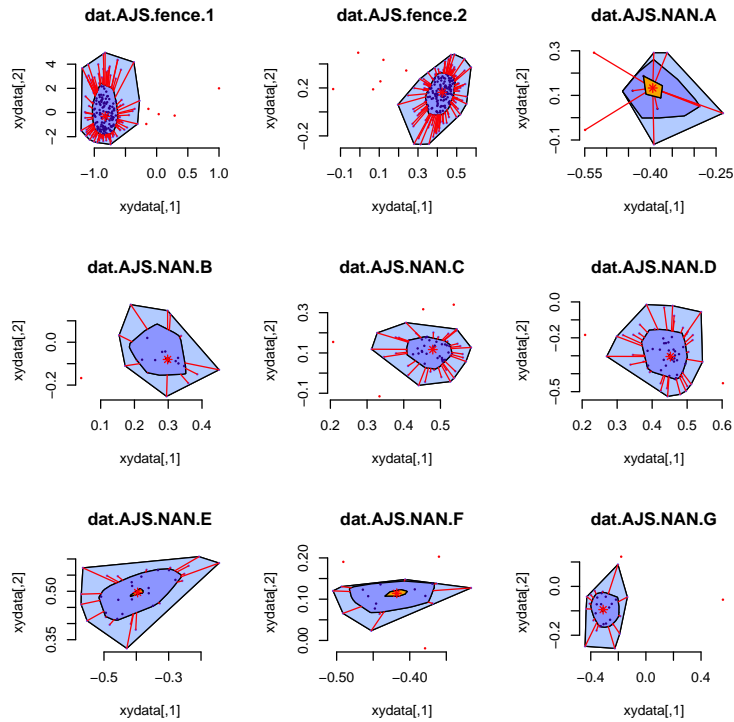
140 (show missing fence data 140) ≡ C 141
par(mfrow=c(3,3))
(Amanda's data set of 29Oct2012: missing fence - A 130)
(Amanda's data set of 29Oct2012: missing fence - B 131)

141 (Na-NaN-error-data-test 141) ≡
par(mfrow=c(3,3))
(show missing fence data 140)
(Amanda's data set of 29Oct2012: Na/NaN-error A 133)

```

(Amanda's data set of 29Oct2012: Na/NaN-error B 134)
 (Amanda's data set of 29Oct2012: Na/NaN-error C 135)
 (Amanda's data set of 29Oct2012: Na/NaN-error D 136)
 (Amanda's data set of 29Oct2012: Na/NaN-error E 137)
 (Amanda's data set of 29Oct2012: Na/NaN-error F 138)
 (Amanda's data set of 29Oct2012: Na/NaN-error G 139)

Here are the result of a 32 bit machine:



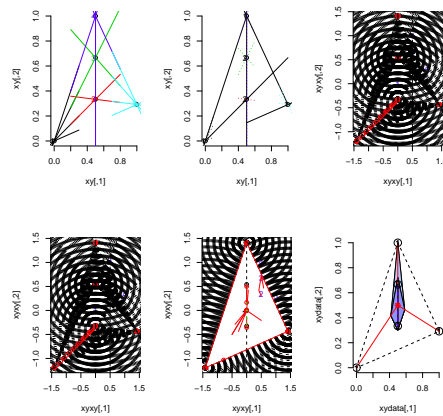
8.2.6 Identical x-values

The cars data set has some points with identical x-values.

```

142 (check cars data set 142) ≡
par(mfrow=c(2,3)) # 121030
(define bagplot 32)
# normalising cars data
c2 <- cars[6:10,]; c2 <- c2 - matrix(apply(c2,2,min),5,2,byrow=TRUE)
c2 <- c2/matrix(apply(c2,2,max),5,2,byrow=TRUE)
# rotation of the data
a<-4*pi/2; c2 <- as.matrix(c2) %*% matrix(c(cos(a),-sin(a),sin(a),cos(a)),2,2)
aa<-bagplot(c2,debug.plots="all",verbose=TRUE);
points(c2,cex=2)

```

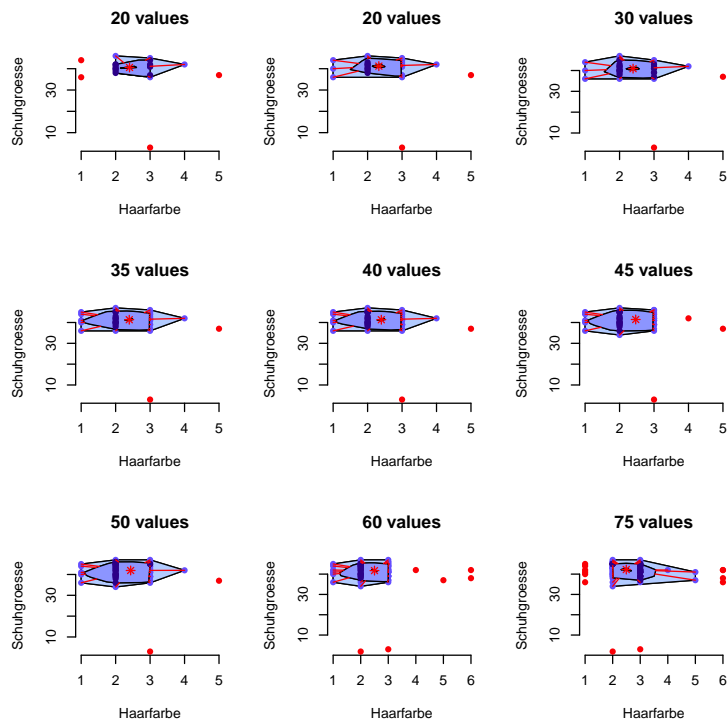


9 frabo12 Daten

The frabo12 data set has some variables with a lot of ties and helped to discover some problems of `bagplot()`.

```
143 (frabo12 143) ≡
(define bagplot 32)
# source("http://www.wiwi.uni-bielefeld.de/fileadmin/stat/wolf/data/frabo12.R"); xy <- frabo12[,3:4]
xy <- structure(list(Haarfarbe = c(2, 2, 1, 3, 3, 3, 2, 3, 1, 5, 3,
3, 3, 2, 3, 2, 2, 4, 2, 3, 2, 2, 2, 1, 2, 3, 2, 3, 2, 3, 1,
1, 3, 3, 2, 2, 3, 2, 3, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 6, 1, 3,
2, 3, 2, 2, 3, 3, 6, 3, 3, 6, 2, 3, 2, 6, 2, 2, 3, 2, 3, 2, 3,
5), Schuhgroesse = c(42, 42, 44, 41, 41, 42, 40, 37, 36, 37,
44, 45, 36, 38, 3, 46, 41, 40, 42, 39, 43, 45, 45, 43, 40, 40,
39, 36, 39, 47, 44, 41, 45, 46, 42, 45, 45, 39, 39, 39, 41, 46,
46, 34, 42, 44, 47, 44, 45, 43, 42, 42, 37, 44, 44, 1.88, 45,
41, 41, 38, 42, 45, 42, 43, 41, 41, 36, 45, 43, 45, 43, 38, 46,
39, 41)), .Names = c("Haarfarbe", "Schuhgroesse"), row.names = c(NA,
75L), class = "data.frame")
par(mfrow=c(3,3))
bagplot(xy[1:20,],dkmethod=2,cex=1); title("20 values")
bagplot(xy[1:25,],dkmethod=2,cex=1); title("20 values")
bagplot(xy[1:30,],dkmethod=2,cex=1); title("30 values")
bagplot(xy[1:35,],dkmethod=2,cex=1); title("35 values")
bagplot(xy[1:40,],dkmethod=2,cex=1); title("40 values")
bagplot(xy[1:45,],dkmethod=2,cex=1); title("45 values")
bagplot(xy[1:50,],dkmethod=2,cex=1); title("50 values")
bagplot(xy[1:60,],dkmethod=2,cex=1); title("60 values")
bagplot(xy,dkmethod=2,cex=1); title("75 values")
par(mfrow=c(1,1))
```

```
144 (Error in 1:pg.nol : NA/NaN Argument 144) ≡
XY <- structure(c(25, 20, 5, 40, 20, 20, 50, 10, 12, 30, 30, 39, 10,
30, 20, 15, 15, 15, 5, 30, 7, 35, 21.66, 25, 40, 20, 4, 15, 20,
25, 50, 3.3, 10, 15, 30, 15, 10, 10, 48, 24, 3, 15, 40, 35, 20,
42, 30, 20, 35, 5, 25, 25, 20, 10, 40, 10, 10, 12, 35, 35, 5,
15, 20, 45, 20, 30, 20, 40, 25, 40, 20, 40, 35, 30, 36, 39, 32,
30, 17.5, 10, 10, 10, 41, 7, 25, 31, 13, 34, 5, 40, 3.9, 30,
20, 12, 25, 40, 40, 15, 5, 19.9, 25, 39.95, 50, 18, 25, 20, 10,
35, 15, 20, 30, 10, 15, 10, 7, 10, 15, 20, 15, 15, 20, 6, 19.95,
30, 15.95, 24.95, 22, 35, 15, 20, 8, 15, 30, 18, 50, 5, 30, 5,
17, 9.9, 30, 22.5, 50, 50, 35, 15, 3, 30, 25, 2, 1, 1, 1, 1,
1, 2, 2, 1, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2,
1, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 1, 1,
1, 1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2,
2, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1,
1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2,
2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2,
2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 2,
1L), .Dimnames = list(NULL, c("X", "Y"))))
bagplot(XY,verbose=TRUE)
```



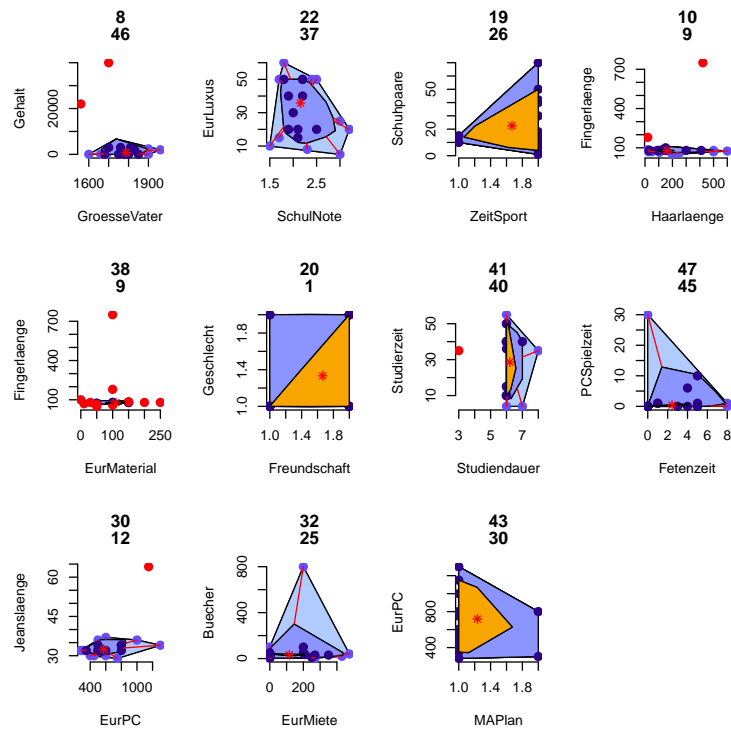
```
145 (* 18) + ≡
(define bagplot 32)
par(mfrow=3:4)
```

```

set.seed(17)
for( i in 1:11 ){
  xy <- frabol2[1:20,IJ <- sample((1:49)[-c(23,28,49)],2)]: print(IJ)
  bp <- bagplot(xy,dkmethod=2,main=as.character(IJ),precision=1.1,cex=1.5)
}
xy <- frabol2[1:20,10:9]
#xy <- frabol2[0,10:9]
#plot(xy)
#bp <- bagplot(xy,precision=2,main="correct center\nby precision 2\n10,9: ",cex=1.5)

```

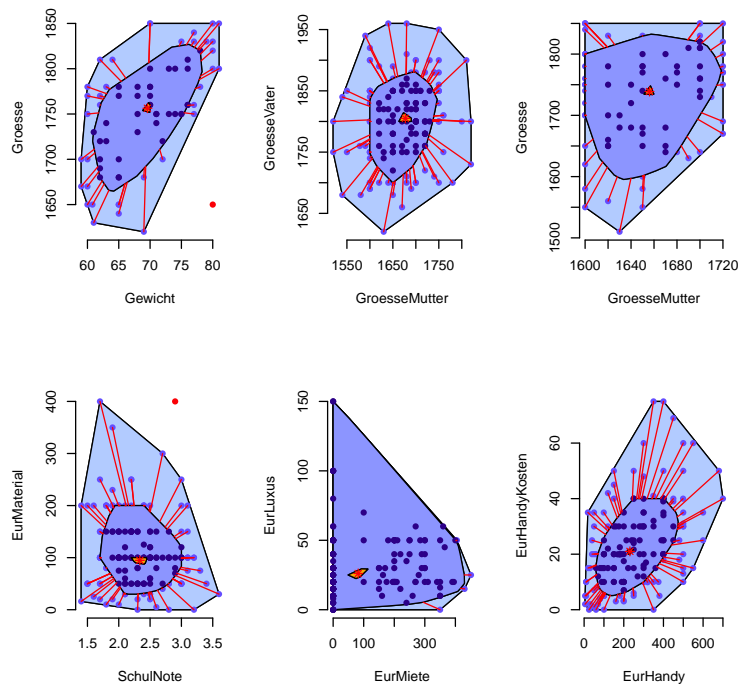
Setting precision to 1.0 will result in a center field in plot four that is a little bit too big. To get rid of this effect the precision has been set to 1.1.



```

146 (doit 146) ≡
(define bagplot 32)
trim.xy <- function(xy,trim = 0.2){
  xy <- as.matrix(xy)
  xy <- xy[ !is.na(xy[,1]) & !is.na(xy[,2]), ]
  xlimits <- quantile(xy[,1], c(trim, 1-trim))
  ylimits <- quantile(xy[,2], c(trim, 1-trim))
  xy <- xy[ xlimits[1] <= xy[,1] & xlimits[2] >= xy[,1] &
    ylimits[1] <= xy[,2] & ylimits[2] >= xy[,2] ,]
  return(xy)
}
par(mfrow=2:3); names(frabol2)
xy <- frabol2[,c("Gewicht","Groesse")]: xy <- trim.xy(xy, 0.2)
bagplot(xy,precision=1,cex=1)
xy <- frabol2[,c("GroesseMutter","GroesseVater")]: xy <- trim.xy(xy, trim = 0.01)
bagplot(xy,precision=1,cex=1)
xy <- frabol2[,c("GroesseMutter","Groesse")]: xy <- trim.xy(xy,0.2)
bagplot(xy,precision=1,cex=1)
xy <- frabol2[,c("SchulNote","EurMaterial")]: xy <- trim.xy(xy,0.01)
bagplot(xy,precision=1,cex=1)
xy <- frabol2[,c("EurMiete","EurLuxus")]: xy <- trim.xy(xy,0.01)
bagplot(xy,precision=1,cex=1)
xy <- frabol2[,c("EurHandy","EurHandyKosten")]: xy <- trim.xy(xy,0.01)
bagplot(xy,precision=1,cex=1)

```



9.1 Indices

Object Index

a1 ∈ 40
 a2 ∈ 40
 aa ∈ 142
 ai ∈ 161, 163
 alpha ∈ 31, 55, 56, 156
 ang ∈ 64, 67, 69, 72, 74, 75
 angles ∈ 34, 41, 44, 55, 64, 67, 68, 69, 71, 72
 angtan ∈ 69, 74
 ao ∈ 56, 161, 164
 a.pdk ∈ 159
 apg ∈ 41
 a.shift ∈ 43, 44, 56, 159
 az ∈ 41
 bagplot ∈ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 21, 23, 27, 28, 29, 30, 31, 32, 33, 35, 45, 52, 53, 81, 83, 85, 86, 88, 90, 91, 115, 116, 117, 118, 120, 121, 122, 124, 125, 126, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 142, 143, 144, 145, 146, 152, 153, 154, 169, 170, 171, 172, 173, 174, 175, 177
 BAGPLOT ∈ 19, 20, 22, 25
 bagplotobj ∈ 81
 bagplot.pairs ∈ 88, 89, 90, 91, 177
 bo ∈ 32
 boxplotres ∈ 84, 168
 bp ∈ 145
 BP ∈ 88
 breaks ∈ 111, 147, 148, 149
 c2 ∈ 142
 candS ∈ 62, 63, 157, 158
 cardata ∈ 2, 31
 center ∈ 2, 7, 10, 21, 23, 27, 28, 31, 34, 35, 38, 39, 40, 41, 46, 47, 49, 52, 53, 62, 64, 65, 66, 77, 78, 82, 83, 91, 127, 145, 153, 157, 158, 163, 166, 167, 170
 center.ex ∈ 27, 28
 check.fence ∈ 132
 check.sen ∈ 129
 check.sense.of.rotation ∈ 127, 129, 132

hull ∈ 47, 48, 50, 62, 65, 66, 73, 77, 80, 126, 128, 129, 132, 155, 157, 158, 165
 col.night ∈ 111
 colors ∈ 31, 92, 107, 110, 147, 148
 cols ∈ 93
 compute.bagplot ∈ 31, 32, 33, 129, 132, 177
 cosphi ∈ 59, 60
 counts.bars ∈ 111, 148
 critical.angles.of.points ∈ 159
 cut.on.pdk ∈ 77
 cut.on.pdk.1 ∈ 77
 cutp ∈ 69, 71
 cut.pkt ∈ 166
 cutpl ∈ 69, 71, 74
 cut.p.sl.p.sl ∈ 36, 42, 69, 73, 155
 cuts ∈ 41, 166
 cut.z.pg ∈ 36, 40, 41, 81
 dat ∈ 31, 93, 105, 107, 108, 120, 121, 122, 123, 173
 data ∈ 2, 3, 4, 5, 10, 16, 17, 20, 21, 22, 23, 25, 27, 28, 30, 31, 34, 36, 44, 45, 52, 53, 60, 61, 67, 84, 90, 91, 92, 93, 99, 106, 107, 110, 111, 114, 115, 117, 120, 121, 122, 124, 126, 129, 132, 140, 141, 142, 143, 160, 168, 169, 170, 173
 dat.AJS.cen.51 ∈ 124
 dat.AJS.cen.51a ∈ 124
 dat.AJS.cen.52 ∈ 124
 dat.AJS.cen.53 ∈ 124
 dat.AJS.chull.1 ∈ 125, 129, 175
 dat.AJS.chull.2 ∈ 125, 126, 129
 dat.AJS.chull.2a ∈ 126
 dat.AJS.fence.1 ∈ 130, 132
 dat.AJS.fence.2 ∈ 131, 132
 dat.AJS.NA.1 ∈ 119
 dat.AJS.NA.10 ∈ 119
 dat.AJS.NA.11 ∈ 119
 dat.AJS.NA.12 ∈ 119
 dat.AJS.NA.13 ∈ 119
 dat.AJS.NA.14 ∈ 119
 dat.AJS.NA.15 ∈ 119
 dat.AJS.NA.16 ∈ 119
 dat.AJS.NA.17 ∈ 119
 dat.AJS.NA.18 ∈ 119
 dat.AJS.NA.19 ∈ 119
 dat.AJS.NA.2 ∈ 119
 dat.AJS.NA.20 ∈ 119
 dat.AJS.NA.21 ∈ 119
 dat.AJS.NA.22 ∈ 119
 dat.AJS.NA.23 ∈ 119
 dat.AJS.NA.24 ∈ 119
 dat.AJS.NA.25 ∈ 119
 dat.AJS.NA.26 ∈ 119
 dat.AJS.NA.3 ∈ 119
 dat.AJS.NA.4 ∈ 119
 dat.AJS.NA.5 ∈ 119
 dat.AJS.NA.6 ∈ 119
 dat.AJS.NA.7 ∈ 119
 dat.AJS.NA.8 ∈ 119
 dat.AJS.NA.9 ∈ 119
 dat.AJS.NaN.1 ∈ 15
 dat.AJS.NaN.2 ∈ 15
 dat.AJS.NAN.A ∈ 133
 dat.AJS.NAN.B ∈ 134
 dat.AJS.NAN.C ∈ 135
 dat.AJS.NAN.D ∈ 136
 dat.AJS.NAN.E ∈ 137
 dat.AJS.NAN.F ∈ 138
 dat.AJS.NAN.G ∈ 139, 152, 154
 datan ∈ 3, 4, 16, 17, 115
 data.name ∈ 93
 dat.name ∈ 120, 121, 122
 dat.type ∈ 105, 107, 108

debug.plots ∈ 4, 10, 17, 31, 32, 33, 40, 41, 47, 55, 56, 66, 69, 73, 74, 75, 77, 83, 91, 117, 122, 142, 155, 159
 d.k ∈ 61, 62, 67, 76, 160, 166
 dkmethod ∈ 2, 4, 6, 7, 10, 13, 16, 17, 23, 27, 31, 32, 33, 34, 116, 117, 118, 143, 145, 152, 154
 dm ∈ 88, 90
 DM1 ∈ 91
 dpi ∈ 43, 44, 56, 159
 ds.of.R ∈ 105, 107, 108
 dx ∈ 37, 43, 44, 55, 56, 84, 102, 111, 148, 159
 dxlim ∈ 168
 dxy ∈ 38, 39, 69, 71
 dy ∈ 37, 43, 44, 55, 56, 84, 159, 168
 dylim ∈ 168
 end.points ∈ 40, 46, 47, 48, 49
 expand.hull ∈ 36, 46, 66
 exp.dk ∈ 35, 53, 61, 66, 67, 77, 82, 83, 116, 167
 exp.dk.1 ∈ 35, 61, 66, 67, 77, 82, 83, 116, 167
 extr ∈ 79, 156, 168
 F2 ∈ 65
 find.cut.z.pg ∈ 36, 41, 47, 49, 77, 81, 83, 166, 167
 find.hdepths ∈ 32, 56, 57
 find.hdepths.tp ∈ 32, 43, 44, 48, 49, 60, 62, 63
 find.polygon.center ∈ 36, 64, 65
 five ∈ 84
 fn ∈ 100, 103
 found ∈ 40, 64, 163
 get.AJS.data ∈ 123, 124, 125, 130, 131, 133, 134, 135, 136, 137, 138, 139
 greiner.data ∈ 14
 h1 ∈ 63, 77, 158, 163, 164
 h2 ∈ 63, 77, 158, 163, 164
 hd ∈ 57, 59
 hdepth ∈ 24, 30, 32, 35, 44, 45, 46, 47, 53, 56, 61, 62, 64, 66, 72, 79, 82, 83, 159
 hdepth.of.points ∈ 30, 36, 43, 48, 49, 62, 63, 157, 158
 hdepths ∈ 31, 34, 35, 53, 82
 hd.table ∈ 61, 62
 hdtp ∈ 60
 hd.tp ∈ 48, 49
 heights.bars ∈ 111, 148
 h.fn ∈ 88
 HH ∈ 126
 HHdelta ∈ 126
 hist.res ∈ 111, 147, 148, 149
 HK ∈ 126
 hp.tp ∈ 48
 hull ∈ 31, 34, 46, 64, 66, 67, 126
 hull.bag ∈ 31, 34, 35, 53, 77, 78, 79, 80, 82, 83, 116, 126, 129, 132, 162, 165
 hull.center ∈ 31, 35, 53, 62, 64, 82, 83, 157, 158
 hull.loop ∈ 31, 34, 35, 53, 78, 79, 80, 82, 83, 116
 hx ∈ 39, 74, 75
 hy ∈ 39, 74, 75
 ia ∈ 68, 69, 70, 71, 74, 75
 ident ∈ 43, 44
 idx ∈ 73, 88, 91, 111
 idx.na ∈ 99, 101, 102
 idx.out ∈ 84
 idx.visible ∈ 92, 99, 100, 102, 103, 104
 ind ∈ 36, 48, 49, 76, 101, 103, 104
 ind1 ∈ 163, 164
 ind2 ∈ 163, 164
 ind.i.points.to.shift ∈ 164, 165
 ind.k ∈ 69, 71, 74, 75
 ind.kk ∈ 69, 71, 74
 indl ∈ 73, 155
 ind.o.points.to.shift ∈ 163, 165
 ind.pdk.1 ∈ 163
 indu ∈ 73, 155
 init ∈ 34, 43, 44, 56
 inner.shift.points ∈ 164, 165

in.pg ∈ 38, 39
int.no ∈ 51
IROT ∈ 84
is.one.dim ∈ 31, 35, 52, 53, 81, 82, 83
jitter.hist ∈ 147, 149
k.1 ∈ 34, 61, 66, 76, 79, 160, 166
kkk ∈ 64, 67, 68, 69, 70, 71, 73
lam ∈ 47, 48, 49, 157, 163, 164
lambda ∈ 34, 65, 76, 77, 160, 163, 164, 166, 167
lambda.i ∈ 166, 167
limit ∈ 32, 33, 38, 39, 69, 73
limit.hdepth.to.check ∈ 62
lnuml ∈ 73, 155
lnumu ∈ 73, 155
lv ∈ 166, 167
lv.1 ∈ 166, 167
lz ∈ 166, 167
max.h ∈ 111, 147
mfrow ∈ 2, 15, 17, 45, 88, 91, 93, 100, 109, 110, 112, 114, 115, 117, 118, 120, 121, 122, 124, 125, 129, 132, 140, 141, 142, 143, 145, 146, 152, 153, 154, 170, 173
mids ∈ 147
mins ∈ 84
minusplus ∈ 43, 44, 56, 159
mxy ∈ 65
mycols ∈ 92, 93, 95, 96, 100, 103, 107, 108, 150
mypi ∈ 56
na.idx ∈ 88
name ∈ 30, 31, 90, 107, 110, 125, 173
namelist ∈ 105, 107, 108
Names ∈ 93
n.bars ∈ 111, 148
n.c ∈ 62, 72, 109, 110, 157
n.cells ∈ 147
n.class ∈ 109, 110, 111, 112, 148
ndk ∈ 76
ndk.1 ∈ 76
next.no ∈ 41
n.i ∈ 111, 147
night ∈ 109, 110, 111, 112, 148
nn ∈ 5
nnn ∈ 70
no ∈ 4, 31, 32, 33, 40, 41, 56, 110, 124, 126, 159
noise ∈ 147
n.p ∈ 62, 63, 157, 158
n.p.beta ∈ 157
npg ∈ 73, 155
n.pg ∈ 41
npgl ∈ 73, 155
nr ∈ 120, 121, 122, 173
n.s ∈ 109, 110
n.tp ∈ 43, 44, 60, 159
n.types ∈ 100
num ∈ 72, 105, 107, 108
num.h ∈ 72
n.xx ∈ 104
n.z ∈ 41
oldpar ∈ 93, 100
out ∈ 84, 103
outer.shift.points ∈ 163, 165
out.of.polygon ∈ 36, 38, 39, 79, 156
outside ∈ 34, 79
pcenter ∈ 161
pdk ∈ 61, 66, 159, 161, 163, 164, 165, 166
pdk.1 ∈ 66, 161, 163, 164, 165, 166
pg ∈ 38, 39, 41, 46, 47, 48, 49, 51, 64, 67, 68, 69, 70, 71, 73, 74, 127, 155, 156
pg0 ∈ 46, 47, 49
pg.add ∈ 49, 50

pgcenter ∈ 156
 pg.inter ∈ 69
 pg.interl ∈ 69
 pgl ∈ 68, 69, 70, 71, 73, 74, 155
 pgn ∈ 38, 39
 pg.new ∈ 46, 48, 49, 50
 pg.no ∈ 69, 74
 pg.nol ∈ 69
 pgo ∈ 41, 166
 phi ∈ 59, 60
 pkt.cand ∈ 79
 pkt.cut ∈ 83
 pkt.not.bag ∈ 79
 plot.bagplot ∈ 31, 32, 81, 129, 132, 177
 plot_single_summary ∈ 92, 93, 150
 plotsummary ∈ 93, 94, 105, 106, 107, 108, 150, 151, 177
 pnew ∈ 69, 74
 pnew.inter ∈ 69
 pnew.interl ∈ 69
 pnewl ∈ 69
 points.in.bag ∈ 32, 36, 61, 81, 83, 160, 166
 pos.to.pg ∈ 36, 51, 73, 155
 prdata ∈ 31, 35, 52, 53, 82, 84, 168
 product ∈ 127
 pxy.bag ∈ 31, 35, 53, 79, 82, 83
 pxy.outer ∈ 31, 35, 53, 79, 80, 82, 83
 pxy.outlier ∈ 31, 35, 53, 79, 82, 83
 R1 ∈ 132
 R2 ∈ 132
 range.bars ∈ 148
 ranges ∈ 84
 res ∈ 35, 52, 53
 resolution ∈ 47
 result ∈ 5, 31, 34, 38, 39, 51, 103, 110, 127, 159
 RM1 ∈ 59, 60, 176
 rnuml ∈ 73, 155
 rnumu ∈ 73, 155
 ro ∈ 159
 ROT ∈ 84
 S2 ∈ 129
 S2new ∈ 129, 132
 seed ∈ 3, 4, 5, 16, 17, 114, 115
 segm ∈ 168
 segm.no ∈ 41, 166
 segtr ∈ 168
 shift ∈ 149, 162
 S.in.pg ∈ 38, 39
 sinphi ∈ 59, 60
 skyline.hist ∈ 109, 110, 111, 112, 113, 148
 sl ∈ 73, 155
 SP ∈ 65
 S.pg ∈ 127
 S.pg.ortho ∈ 127
 sr ∈ 73, 155
 starts.bars ∈ 111, 148
 step ∈ 36
 sx ∈ 40, 42
 sxy ∈ 40
 sy ∈ 40, 42
 tp ∈ 30, 43, 44, 45, 48, 49, 60, 63, 82, 83, 157, 158, 159
 TP ∈ 24, 45, 63, 91, 152, 154
 TPD ∈ 24, 63, 91, 152, 154
 tphdepth ∈ 43, 44, 62, 63, 64, 82, 83, 157, 158, 159
 tpt ∈ 60
 trdata ∈ 168
 trim ∈ 88, 90, 91, 92, 93, 99, 105, 107, 108, 146, 150, 151
 trim.xy ∈ 146

type ∈ 39, 70, 73, 74, 75, 83, 84, 92, 93, 100, 101, 102, 103, 104, 105, 107, 108, 110, 111, 126, 147, 148, 152, 159, 168
 union.points ∈ 163
 usr ∈ 84, 100, 111, 150, 168
 vec ∈ 84
 vec.ortho ∈ 84
 very.large.data.set ∈ 36, 79
 vt ∈ 166, 167
 vt.1 ∈ 166, 167
 width.bars ∈ 111, 148
 win ∈ 36, 37, 41, 43, 44, 81, 156, 159, 161, 163, 164
 x0 ∈ 10, 11, 22, 23, 24, 25, 27, 28, 102, 170
 x1 ∈ 170
 x2 ∈ 170
 x3 ∈ 170
 xdelta ∈ 55, 56
 x.i ∈ 66, 111, 147
 xlim ∈ 23, 39, 70, 73, 84, 88, 99, 100, 101, 103, 104, 110, 111, 126, 148, 168
 x.lim ∈ 111
 xlimits ∈ 146
 x.split ∈ 147
 xx ∈ 104
 xy ∈ 5, 30, 31, 35, 36, 38, 39, 43, 44, 47, 48, 49, 52, 53, 54, 55, 56, 57, 58, 59, 60, 62, 63, 65, 66, 75, 77, 82, 83, 91, 105, 107, 108, 127, 143, 145, 146, 150, 152, 154, 156, 159, 166, 167, 176
 XY ∈ 39, 144
 xy0 ∈ 163, 164
 xy1 ∈ 42, 163, 164
 xy2 ∈ 42, 65, 163, 164
 xy3 ∈ 65
 xy.cut ∈ 163, 164
 xydata ∈ 31, 35, 36, 52, 53, 79, 82, 83, 84, 168
 xydel ∈ 63, 158
 xy.delta ∈ 84
 xyextr ∈ 63, 158
 xyi ∈ 66
 xyl ∈ 73, 155
 xym ∈ 34, 54, 64, 67
 xy.medians ∈ 36
 xyo ∈ 66
 xysd ∈ 34, 54, 64, 67
 xyt ∈ 59, 60, 69, 71
 xyto ∈ 69, 71
 xytr ∈ 84
 xyu ∈ 73, 155
 xyxy ∈ 34, 54, 67, 69, 71, 73, 74, 75
 y0 ∈ 10, 11, 22, 23, 24, 25, 27, 28, 101, 102, 103, 104, 170
 y1 ∈ 170
 y2 ∈ 170
 y3 ∈ 170
 y.i ∈ 111
 ylim ∈ 23, 39, 70, 73, 84, 100, 110, 111, 148, 168
 ylimits ∈ 146
 y.mins ∈ 100, 101, 102, 103, 104
 ytr ∈ 168
 yy ∈ 104
 zy.on.pg ∈ 51

Code Chunk Index

$\langle *18 \cup 19 \cup 20 \cup 21 \cup 24 \cup 96 \cup 106 \cup 109 \cup 112 \cup 115 \cup 118 \cup 124 \cup 145 \cup 147 \cup 149 \cup 150 \cup 151 \cup 153 \cup 173 \cup 174 \cup 175 \cup 176 \cup 177 \rangle$	p15
$\langle \textit{Amanda's data set of 29Oct2012: missing fence} - A \ 130 \rangle$	⊂ 132, 140	p75
$\langle \textit{Amanda's data set of 29Oct2012: missing fence} - B \ 131 \rangle$	⊂ 132, 140	p76
$\langle \textit{Amanda's data set of 29Oct2012: Na/NaN-error A} \ 133 \rangle$	⊂ 141	p78
$\langle \textit{Amanda's data set of 29Oct2012: Na/NaN-error B} \ 134 \rangle$	⊂ 141	p78
$\langle \textit{Amanda's data set of 29Oct2012: Na/NaN-error C} \ 135 \rangle$	⊂ 141	p78
$\langle \textit{Amanda's data set of 29Oct2012: Na/NaN-error D} \ 136 \rangle$	⊂ 141	p79

<code><Amanda's data set of 29Oct2012: Na/NaN-error E 137></code>	<code>⊂ 141</code>	p79
<code><Amanda's data set of 29Oct2012: Na/NaN-error F 138></code>	<code>⊂ 141</code>	p79
<code><Amanda's data set of 29Oct2012: Na/NaN-error G 139></code>	<code>⊂ 141</code>	p79
<code><another data set 1 169></code>		p??
<code><another data set 2 170></code>		p??
<code><another data set 3 171 ∪ 172></code>		p??
<code><args 29></code>		p20
<code><assing data set of Martin Maechler to x0 and y0 11></code>	<code>⊂ 10, 22, 23, 25</code>	p9
<code><a verbose test with AJS data no 26 122></code>		p69
<code><BAGPLOT of data set of Martin Maechler 22></code>		p17
<code><bagplot of data set of Martin Maechler 23></code>		p17
<code><bagplot of data set of Martin Maechler, difference if precision is increased 28></code>		p19
<code><BAGPLOT of data set of Martin Maechler, exchanged variables 25></code>		p18
<code><bagplot of data set of Martin Maechler, exchanged variables 27></code>		p18
<code><BAGPLOT of data set of Martin Maechler, relative difference 26></code>		p18
<code><beta 157></code>		p??
<code><body of loop of directions 69></code>	<code>⊂ 68</code>	p42
<code><body of compute.bagplot 34></code>	<code>⊂ 33</code>	p25
<code><call tangleR to extract tangle function bagplot() 87></code>		p50
<code><cardata 2></code>		p3
<code><check and handle linear case 52></code>	<code>⊂ 34</code>	p35
<code><check clockwise condition of polygon 127></code>	<code>⊂ 129, 132</code>	p74
<code><checke cars data set 142></code>		p80
<code><check of function hdepth() 45></code>		p32
<code><check some points to find the maximal h-depth 63></code>	<code>⊂ 62</code>	p39
<code><chunk for checking function out.of.polygon 39></code>		p28
<code><circle 7></code>		p7
<code><combination of lower and upper polygon 73></code>	<code>⊂ 68</code>	p43
<code><compute angles between points 55></code>	<code>⊂ 34</code>	p36
<code><compute boxplot 103></code>	<code>⊂ 92</code>	p57
<code><compute density trace 102></code>	<code>⊂ 92</code>	p57
<code><compute ecdf 104></code>	<code>⊂ 92</code>	p57
<code><compute hdepths 56></code>	<code>⊂ 34</code>	p36
<code><compute hdepths of test points to find center 62></code>	<code>⊂ 34</code>	p39
<code><compute hull pg.new 50></code>	<code>⊂ 46</code>	p34
<code><compute stripes 101></code>	<code>⊂ 92</code>	p56
<code><construct bagplot as usual 83></code>	<code>⊂ 81</code>	p48
<code><construct plot for one dimensional case and return 84></code>	<code>⊂ 81</code>	p49
<code><data set 1 of Wouter Meuleman 13></code>	<code>⊂ 117</code>	p10
<code><data set 2 of Wouter Meuleman 12></code>		p10
<code><data set of Amanda Joy Shaker 15></code>		p12
<code><data set of Martin Maechler 10></code>		p9
<code><debugplot 17></code>		p14
<code><debug: plot ini 75></code>	<code>⊂ 71</code>	p45
<code><debug: plot within for loop 74></code>	<code>⊂ 69</code>	p44
<code><define another hull function 128></code>	<code>⊂ 129, 132</code>	p74
<code><define data sets of AJS 119></code>	<code>⊂ 120, 121, 122</code>	p67
<code><define data set with hull problem 125></code>	<code>⊂ 129</code>	p71
<code><define experiment to check hull error and solution 129></code>		p74
<code><define experiment to check fence error on a 64 bit machine 132></code>		p77
<code><define function cut.p.sl.p.sl 42></code>	<code>⊂ 36</code>	p31
<code><define function cut.z.pg 40></code>	<code>⊂ 36, 81</code>	p29
<code><define function expand.hull 46></code>	<code>⊂ 36</code>	p33
<code><define function find.cut.z.pg 41></code>	<code>⊂ 36, 81</code>	p30
<code><define function get.AJS.data() 123></code>	<code>⊂ 124, 125, 130, 131, 133, 134, 135, 136, 137, 138, 139</code>	p70
<code><define function hdepth() 44></code>	<code>⊂ 32, 45</code>	p31
<code><define function hdepth.of.points 43></code>	<code>⊂ 36</code>	p31
<code><define function out.of.polygon 38></code>	<code>⊂ 36</code>	p28
<code><define function pos.to.pg 51></code>	<code>⊂ 36</code>	p34
<code><define function win 37></code>	<code>⊂ 36, 44, 81</code>	p28
<code><define help of bagplot 31></code>		p21

<code><define help of bagplot.pairs 90></code>	p51
<code><define help of hdepth 30></code>	p20
<code><define help of plotsummary 107></code>	p58
<code><define help of skyline.hist 110></code>	p61
<code><define rnorm data data, seed: seed, size: n 114></code>	C 3, 4, 16, 17, 115 p67
<code><define bagplot 32></code>	C 2, 3, 4, 5, 6, 7, 10, 12, 13, 14, 15, 17, 45, 85, 86, 91, 115, 116, 117, 118, 120, 121, 122, 124, 125, 129, 132, 142, 143, 145, 146, 152, 153, 154, 173, 174 p24
<code><define bagplot.pairs 88></code>	C 89, 91 p51
<code><define compute.bagplot 33></code>	C 32 p25
<code><define find.hdepths 57></code>	C 32, 56 p37
<code><define find.hdepths.tp 60></code>	C 32 p37
<code><define find.polygon.center 65></code>	C 36 p40
<code><define plot.bagplot 81></code>	C 32 p47
<code><define plot_single_summary 92></code>	C 93 p54
<code><define plotsummary 93></code>	C 94, 105, 106, 150, 151 p55
<code><define skyline.hist 111></code>	C 109, 113 p64
<code><doit 146></code>	p82
<code><do bagplot.pairs test 91></code>	p53
<code><error 117></code>	p67
<code><Error in 1:pg.nol : NA/NaN Argument 144></code>	p81
<code><Example where hull gives bad results 126></code>	p73
<code><find additional points between the line segments 49></code>	C 46 p34
<code><find end points of line segments: center → pg → pg0 47></code>	C 46 p33
<code><find hull of loop 80></code>	C 34 p47
<code><find kkk-hull: pg 68></code>	C 64, 67 p41
<code><find limits for trimming, normalize data and store visibility: idx.visible 99></code>	C 92 p56
<code><find points outside of bag but inside loop 79></code>	C 34 p46
<code><find the polygon of points of maximal h-depth 64></code>	C 62 p39
<code><find hull.bag 77></code>	C 34 p46
<code><find hull.loop 78></code>	C 34 p46
<code><find k 61></code>	C 34 p38
<code><find value of lambda 76></code>	C 34 p45
<code><fix main title 97></code>	C 92 p56
<code><frabo12 143></code>	p81
<code><handle three or four data points 53></code>	C 34 p35
<code><init 36></code>	C 34 p26
<code><initialize graphics and find ymin 100></code>	C 92 p56
<code><initialize loop of directions 71></code>	C 68 p43
<code><initialize some variable 82></code>	C 81 p48
<code><initialize angles, ang 72></code>	C 64, 67 p43
<code><large 4></code>	p5
<code><loop over directions 59></code>	C 57 p37
<code><method one: find hulls of D_k and D_{k-1} 66></code>	C 34 p40
<code><method two: find hulls of D_k and D_{k-1} 67></code>	C 34 p41
<code><Na-NaN-error-data-test 141></code>	p79
<code><old 161></code>	p??
<code><OLD: construct plot for one dimensional case and return 168></code>	p??
<code><old: define skyline.hist 148></code>	p??
<code><old: experiment for finding bag 159></code>	p??
<code><old: find bag 162></code>	p??
<code><old: find points of outer polygon to be shift 163></code>	p??
<code><old: find points to shift on inner polygon 164></code>	p??
<code><old: find value of lambda 167></code>	p??
<code><old: find value of lambda – old version 166></code>	p??
<code><old: shift points on polygons and construct hull.bag 165></code>	p??
<code><old version: check points on a grid to find center 158></code>	p??
<code><old version: define function out.of.polygon 156></code>	p??
<code><old version of the combination of lower and upper polygon 155></code>	p??
<code><old version to find lambda 160></code>	p??
<code><onedim 8></code>	p8
<code><one dim test 9></code>	p8

<code><output result 35></code>	<code>⊂ 34</code>	p26
<code><quadratic 6></code>		p7
<code><quadratic-test 116></code>		p67
<code><rnorm 3></code>		p4
<code><rnorm, different sizes 5></code>		p6
<code><search for points with critical hdepth 48></code>	<code>⊂ 46</code>	p33
<code><select colors 95></code>	<code>⊂ 92</code>	p55
<code><show missing fence data 140></code>	<code>⊂ 141</code>	p79
<code><show pg pgl 70></code>		p43
<code><some functions for generating bagplots 86></code>		p50
<code><standardize data and compute: xyxy, xym, xysd 54></code>	<code>⊂ 34</code>	p36
<code><standardize dimensions 58></code>	<code>⊂ 57</code>	p37
<code><start 85 ∪ 89 ∪ 94 ∪ 113></code>		p50
<code><test 121126 152 ∪ 154></code>		p??
<code><test: data set of Ben Greiner 14></code>		p11
<code><test of amanda-data-sets 120 ∪ 121></code>		p68
<code><Test of R data sets 105></code>		p57
<code><test of the examples of plotsummary 108></code>		p60
<code><transform data into numerical vectors 98></code>	<code>⊂ 92</code>	p56
<code><verbosetest 16></code>		p13
<code><version of bagplot 1></code>	<code>⊂ 32, 33, 81</code>	p2