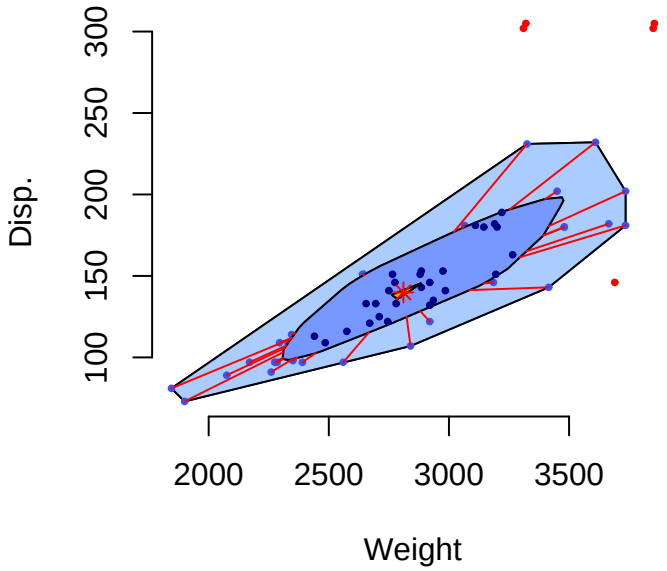
A rough R Impementation of the Bagplot, Data Peeling, Skyline Plots, and Graphical Summaries

File: bagplot.rev
in: /home/wiwi/pwolf/R/work/bagplot

Version: Dec 10 19:15:01 2012



Contents

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 ⟨*version of bagplot* 1⟩ ≡ ⊂ 32, 33, 81
```
"bagplot, version 2012/12/05, peter wolf"
```

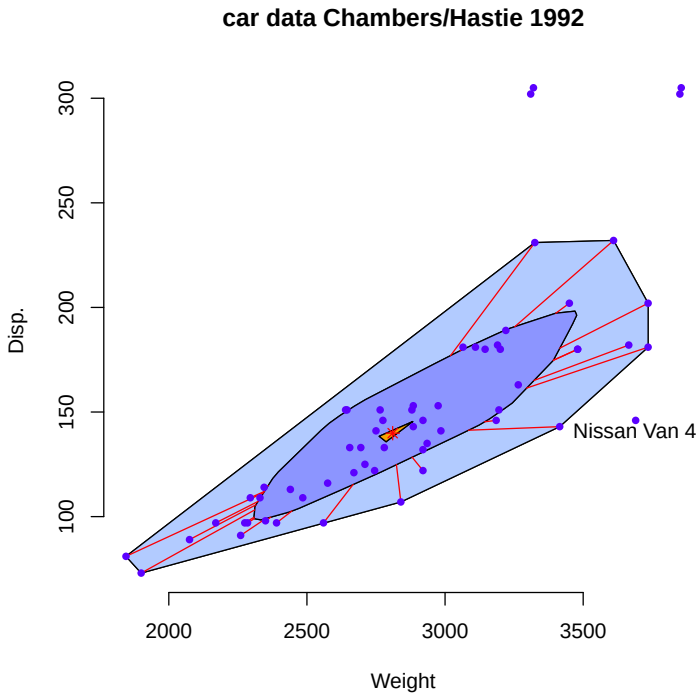
1 Examples

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 ...

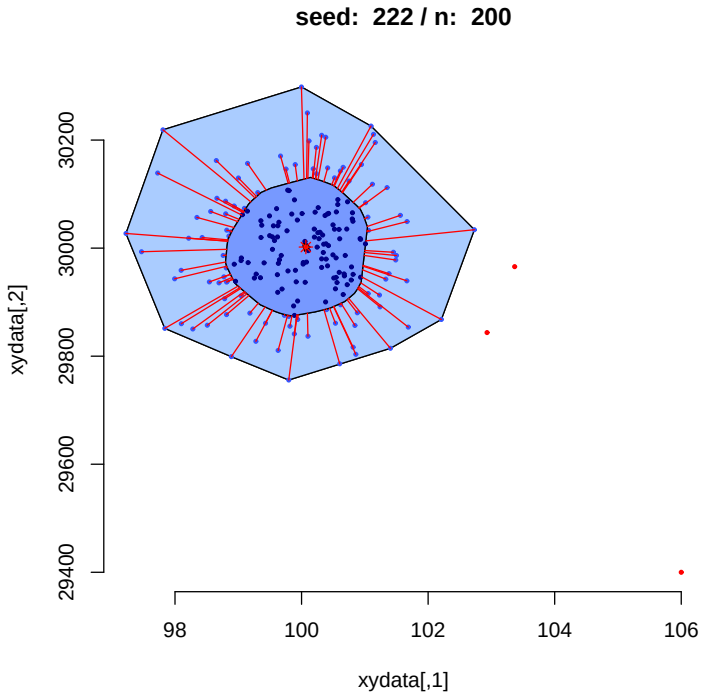
```
Splus CHAPTER bagplot.f
Splus make
Splus ...
> dyn.open("S.so"); source("bagplot.s") #; postscript("hello.ps")
> bagplot(cardata[,1],cardata[,2]) #; dev.off()
```

1.2 The normal case

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

3 ⟨*rnorm* 3⟩ ≡

```
⟨define bagplot 32⟩
seed<-222; n<-200
⟨define rnorm data data, seed: seed, size: n 118⟩
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))
```

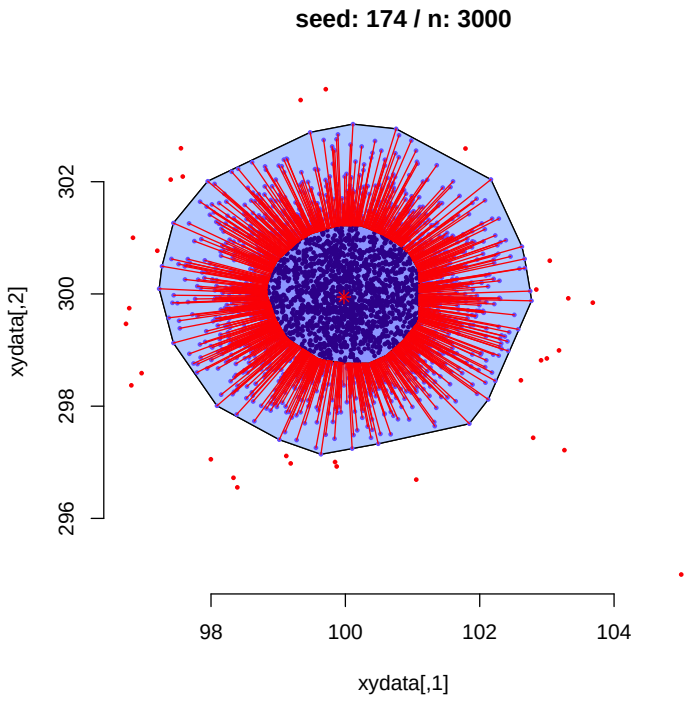


1.3 Large data sets

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 118⟩
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))
```

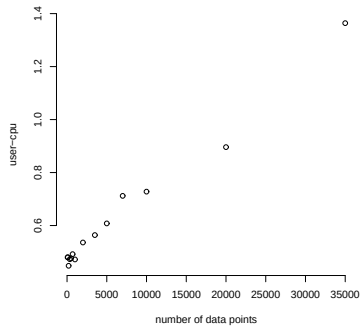


1.4 Size of data set

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⟩ ≡

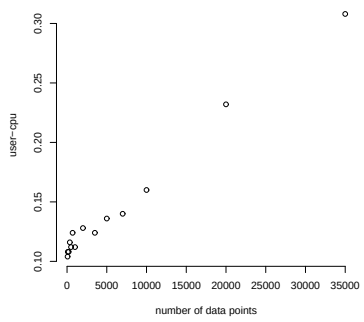
```
⟨define bagplot 32⟩
nn<-c(50,70,100,200,350); nn<-c(nn,10*nn,100*nn);nn<-nn[-1]
result<-1:length(nn)
for(j in seq(along=nn)){
  seed<-111; set.seed(seed); n<-nn[j]
  xy<-cbind(rnorm(n),rnorm(n))
  result[j]<-system.time(
    bagplot(xy,factor=3,create.plot=FALSE,approx.limit=300,
     show.outlier=TRUE,show.looppoints=TRUE,show.bagpoints=TRUE,
     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:

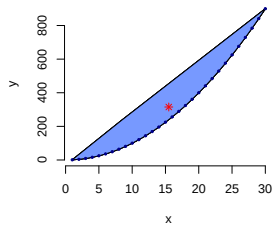
```
Wed Oct  3 11:46:05 2012
   70   100   200   350   500   700  1000  2000  3500  5000  7000 10000 20000
0.104 0.108 0.108 0.116 0.112 0.124 0.112 0.128 0.124 0.136 0.140 0.160 0.232
35000
0.308
```



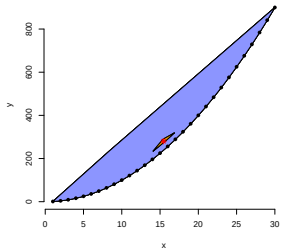
1.5 "Depth-One" data sets

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)
```

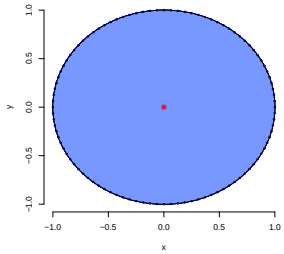


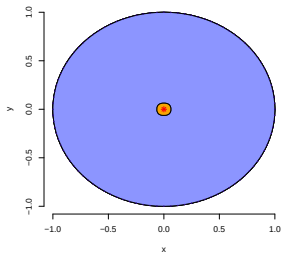
In the 2012 version the region of the median region is shown.



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





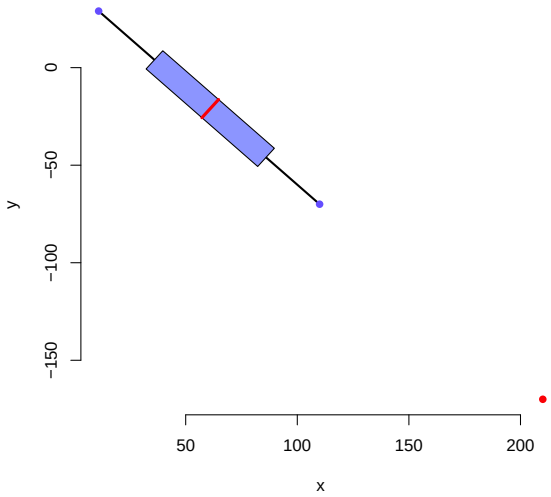
1.6 Degenerated data sets

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

8 ⟨*onedim* 8⟩ ≡

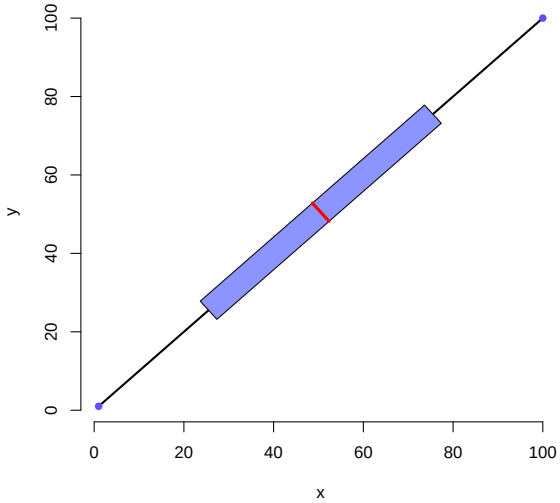
```
bagplot(x=10+c(1:100,200),y=30-c(1:100,200),verbose=FALSE)
```

Here is a second one dim data set.



9 ⟨*one dim test* 9⟩ ≡

```
bagplot(x=(1:100),y=(1:100),verbose=FALSE)
```



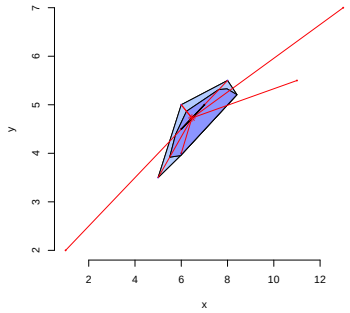
1.7 Data set from the mail of M. Maechler

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)
```

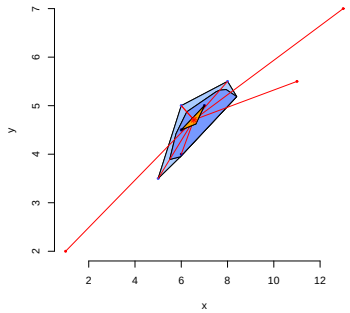
Bagplot of version 12/2012:



11 ⟨*assing data set of Martin Maechler to* x0 *and* y0 11⟩ ≡ ⊂ 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:

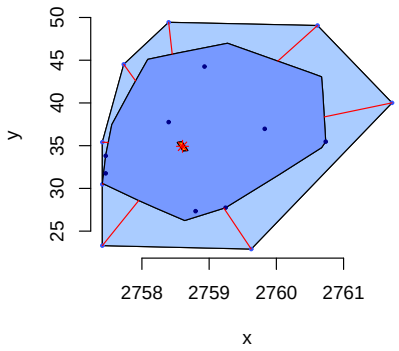


1.8 Data sets of Wouter Meuleman, running in an error with version 09/2005

An old bagplot version runs into errors with following data set. During the computation of ⟨*find* `hull.bag` 77⟩ some NaN values occured.

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)
```

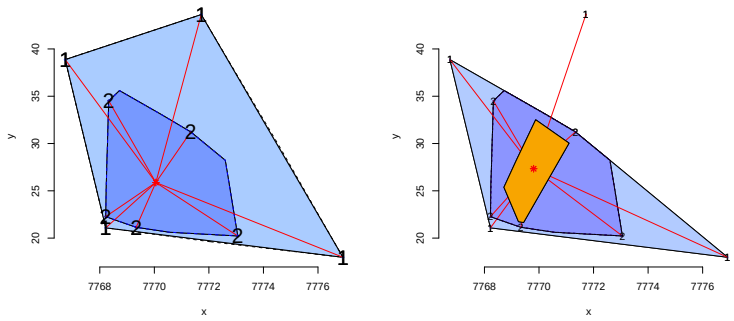


On 2006/02/17 some lines of code have been changed to remove the NaN values.

13 ⟨*data set 1 of Wouter Meuleman* 13⟩ ≡ ⊂ 121

```
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")
```

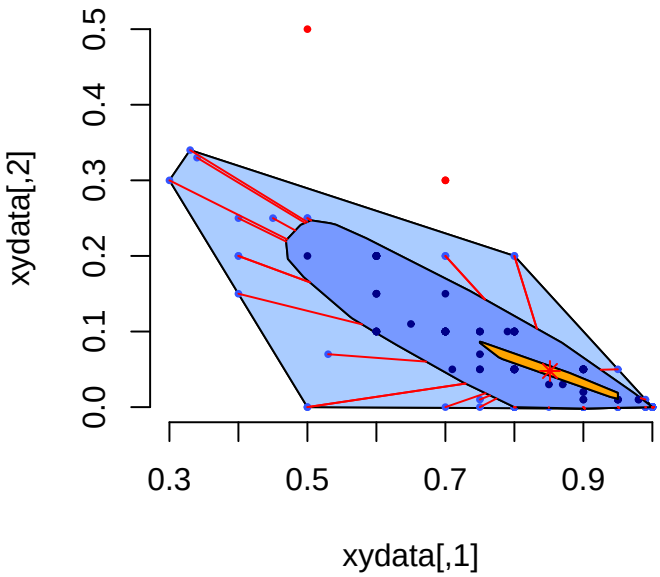
On the left we find a result of an older version of `bagplot()`. However, the central region hasn't been found – compare the following bagplot. To integrate the point at the top edge of this picture you have to enlarge the `factor` a little bit.



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,
0,0.2,0.01,0.2,0,0.1,0.01,0,0,0.33,0.1,0,0.05,0.15,0.1,0.1,0.1,0.01,
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,
0,0.2,0,0.01,0,0.05,0.2,0.05,0,0.15,0,0.05,0,0.2,0.2,0,0.2,0,0,0.2,
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,
0.05,0,0,0,0.05,0.07,0.05,0.1,0.3,0.05,0,0.1,0.1,0.2,0.02,0.05,0.2,
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,
0.01,0.05,0.1,0.03,0,0,0,0,0.1,0.05,0.1,0.05,0.1,0,0.2,0.2,0,0.01,0,0.1))
bagplot(greiner.data,precision=3)
""
```



1.9 Data sets of Amanda Joy Shaker

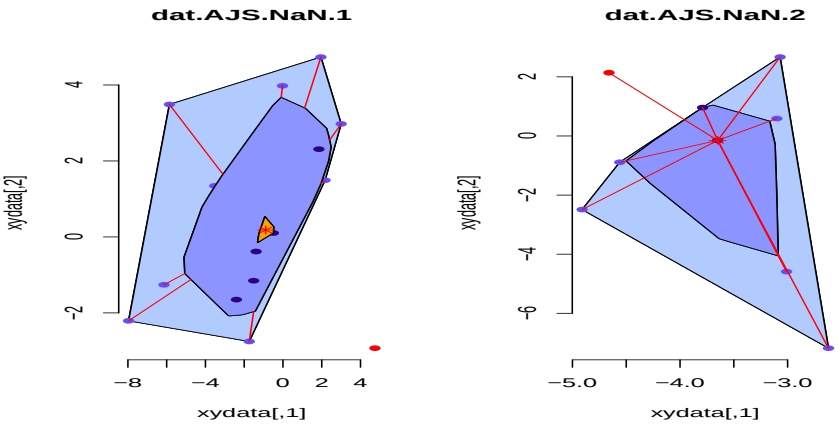
The bagplot function (version from 2011) runs into an error by using a data set of Amanda Joy Shaker (AJS):

```
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.

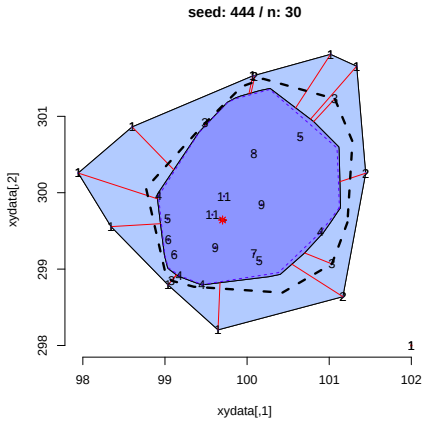
```
 [1] dat.AJS.cen.51   dat.AJS.cen.51a  dat.AJS.cen.52   dat.AJS.cen.53
 [5] dat.AJS.chull.1  dat.AJS.chull.2  dat.AJS.chull.2a dat.AJS.fence.1
 [9] dat.AJS.fence.2  dat.AJS.NA.1     dat.AJS.NA.10    dat.AJS.NA.11
[13] dat.AJS.NA.12    dat.AJS.NA.13    dat.AJS.NA.14    dat.AJS.NA.15
[17] dat.AJS.NA.16    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 ⟨*verbosetest* 16⟩ ≡

```
seed<-444; n<-30
⟨define rnorm data data, seed: seed, size: n 118⟩
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))
```

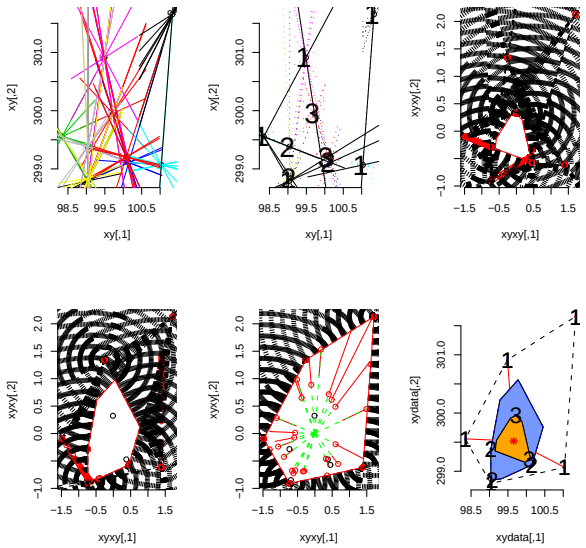


1.11 Debugging plots with additional elements

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 118⟩
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.

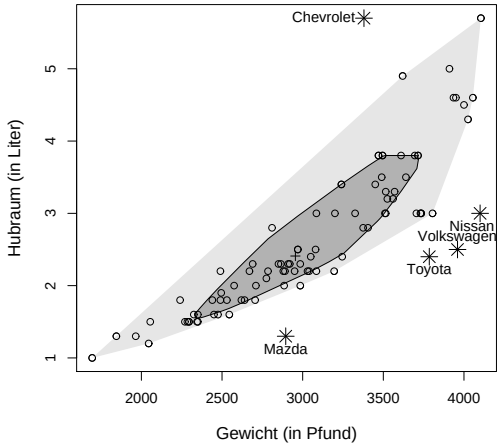
20 ⟨* 18⟩+ ≡

```
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 ).
```

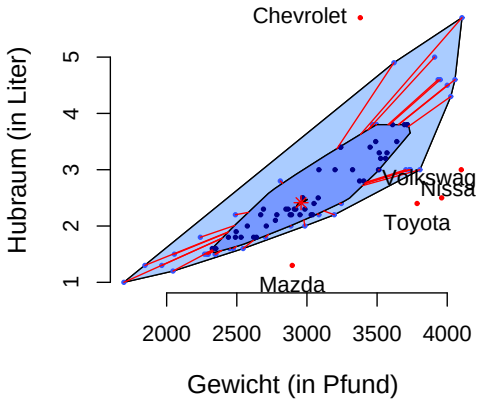
and the bagplot:



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.

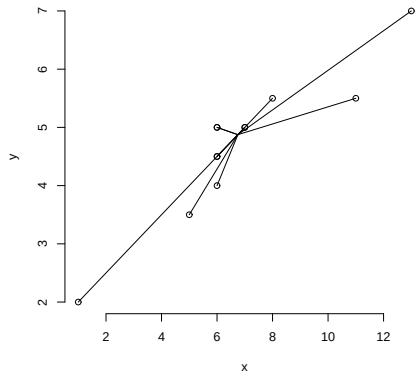
Test of data set of Martin Maechler. As a second example we check the bagplot functions by the data set of Martin Maecher.

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 ...

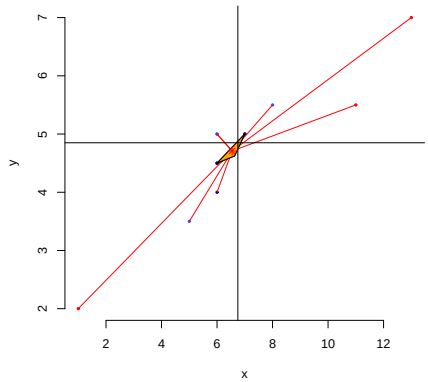
```
[1] The coordinates of the Tukey median are ( 6.75 , 4.875 ).
```

and the following plot



Our procedure will compute slightly different results:

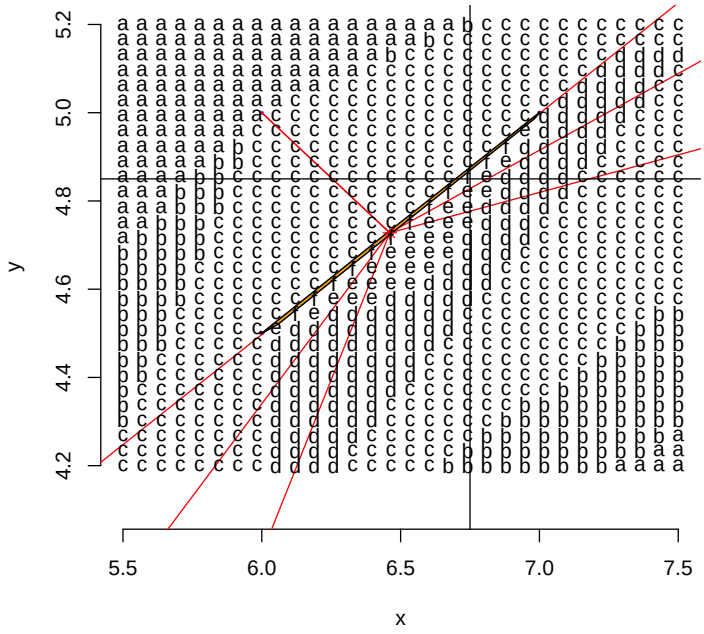
23 ⟨*bagplot of data set of Martin Maechler* 23⟩ ≡
⟨*assing data set of Martin Maechler to* `x0` *and* `y0` 11⟩
```
center<-bagplot(x0, y0, show.baghull=FALSE, show.loophull=FALSE,
                create.plot=TRUE, show.whiskers=TRUE, factor=3,
                dkmethod=2, precision=3,
                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,
                dkmethod=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:

`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

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,
       dkmethod = 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,
       dkmethod=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 plottet }
  \item{show.bagpoints}{ if TRUE bag points are plottet }
  \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⟩ ≡ ⊂ 2, 3, 4, 5, 6, 7, 10, 12, 13, 14, 15, 17, 45, 85, 86, 91, 119, 120, 121, 122, 124, 125, 126, 128, 129, 133, 136, 146, 147, 149, 150, 156, 157, 158, 177, 178

⟨*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
   dkmethod=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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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.1=exp.dk.1,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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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 `p1x,p1y,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⟩ ≡ ⊂ 36, 81

```
cut.z.pg<-function(zx,zy,p1x,p1y,p2x,p2y){
  a2<-(p2y-p1y)/(p2x-p1x); a1<-zy/zx
  sx<-(p1y-a2*p1x)/(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-p1x))
  if(!exists("verbose")) verbose<-FALSE
    if(verbose) cat("special")
    # zx is zero # 121030
    h<-0==zx
       sx<-ifelse(h,zx,sx); sy<-ifelse(h,p1y-a2*p1x,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(p1x)
       sx<-ifelse(h,p1x,sx); sy<-ifelse(h,p1y,sy)
    h<-0==(a1-a2) & sign(zx)!=sign(p1x)
       sx<-ifelse(h,p2x,sx); sy<-ifelse(h,p2y,sy)
    # line segment vertical
    #   & center NOT ON line segment
    h<-p1x==p2x & zx!=p1x & p1x!=0
       sx<-ifelse(h,p1x,sx); sy<-ifelse(h,zy*p1x/zx,sy)
    #   & center ON line segment
```

```
    h<-p1x==p2x & zx!=p1x & p1x==0
       sx<-ifelse(h,p1x,sx); sy<-ifelse(h,0,sy)
    #   & center NOT ON line segment & point on line     # 121126
    h<-p1x==p2x & zx==p1x & p1x!=0 # & sign(zy)==sign(p1y)
         sx<-ifelse(h,zx,sx); sy<-ifelse(h,zy,sy)
    #   & center ON line segment & point on line
    h<-p1x==p2x & zx==p1x & p1x==0 & sign(zy)==sign(p1y)
       sx<-ifelse(h,p1x,sx); sy<-ifelse(h,p1y,sy)
    h<-p1x==p2x & zx==p1x & p1x==0 & sign(zy)!=sign(p1y)
       sx<-ifelse(h,p1x,sx); sy<-ifelse(h,p2y,sy)
    #  points identical to end points of line segment
    h<-zx==p1x & zy==p1y; sx<-ifelse(h,p1x,sx); sy<-ifelse(h,p1y,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,p1x,p1y,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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 36

```
cut.p.sl.p.sl<-function(xy1,m1,xy2,m2){
  sx<-(xy2[2]-m2*xy2[1]-xy1[2]+m1*xy1[1])/(m1-m2)
  sy<-xy1[2]-m1*xy1[1]+m1*sx
  if(!is.nan(sy)) return( c(sx,sy) )
  if(abs(m1)==Inf) return( c(xy1[1],xy2[2]+m2*(xy1[1]-xy2[1])) )
  if(abs(m2)==Inf) return( c(xy2[1],xy1[2]+m1*(xy2[1]-xy1[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⟩ ≡ ⊂ 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.

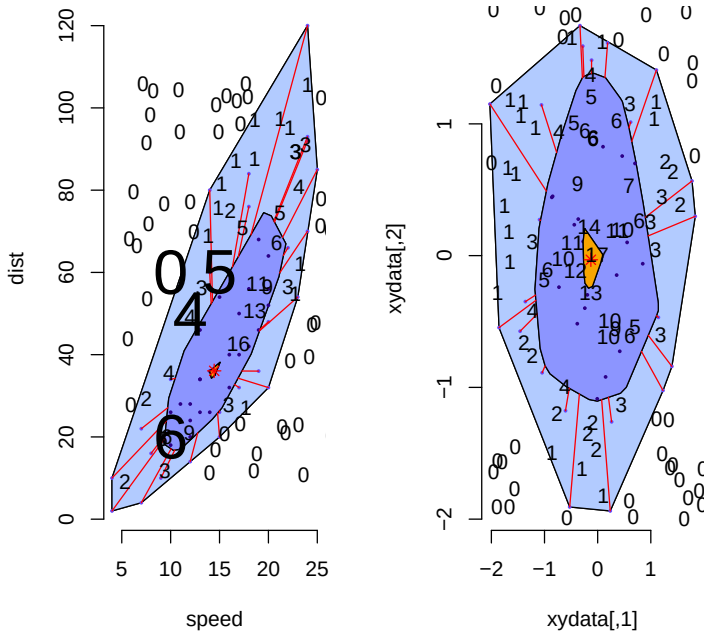
44 ⟨*define function* `hdepth()` 44⟩ ≡ ⊂ 32, 45

```
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 expandation.

46 ⟨*define function* `expand.hull` 46⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 46

```
pg.add<-0.5*(pg.new+rbind(pg.new[-1,],pg.new[1,]))
# 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 hull(d_k). `pg.new` is the output of `expand.hull`.

50 ⟨*compute hull* `pg.new` 50⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 32, 56

```
find.hdepths <- function(xy, number.of.directions=181){ # 121126
  ⟨standardize dimensions 58⟩
  ⟨loop over directions 59⟩
  hd
}
```

58 ⟨*standardize dimensions* 58⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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 substracting 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⟩ ≡ ⊂ 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))
  hdtp <- 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)
    hdtp <- pmin(hdtp,(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")
            )
  }
  hdtp
}
```

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 `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 `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 fullfilled 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⟩ ≡ ⊂ 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 boundery 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⟩ ≡ ⊂ 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))],]
  cands<-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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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 appropiate 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 68

```
# determine critical points pnew and pnewl 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]; pnew<-xyxy[ind.k,]; pnewl<-xyxy[ind.kk,]
# if(verbose) if( 1 < sum(xyt == xyt[ind.k]) )print("WARNING: some points identical")
if(debug.plots=="all") points(pnew[1],pnew[2],col="red")
# new limiting lines are defined by pnew / pnewl and slope a
# find segment of polygon that is cut by new limiting line and cut
# if(ia>200) { #<show pg pgl>#; points(pnew[1],pnew[2],col="magenta",cex=6) }

if( abs(angtan)>1e10){ if(verbose) cat("kkk",kkk,"x=c case")
    # case of vertical slope #print(pg);print(pnew);print(xyt);lines(pg,col="red",lwd=3)
    # number of points left of point pnew that limit the polygon
    pg.no<-sum(pg[,1]<pnew[1])
    if( 0 < pg.no ){
      # the polygon (segment pg.no) has to be cut at x==pnew[1]
      cutp <- c(pnew[1], pg [pg.no, 2]+pg [pg.no, 3]*(pnew [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 pnew 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]>=pnewl[1])
    if( 0 < pg.nol ){ ##??2 # 121204
      cutpl<-c(pnewl[1],pgl[pg.nol,2]+pgl[pg.nol,3]*(pnewl[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]; pnew.inter<-pnew[2]-angtan*pnew[1]
    pg.no<-sum(pg.inter<pnew.inter)
    if(is.na(pg[pg.no,3])) pg[pg.no,3] <- -Inf # 121129 NaN/Na error
    cutp<-cut.p.sl.p.sl(pnew,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]; pnew.interl<-pnewl[2]-angtan*pnewl[1]
    pg.nol<-sum(pg.interl>pnew.interl)
    if(is.na(pgl[pg.nol,3])) pgl[pg.nol,3] <- Inf # 121129 NaN/Na error
    cutpl<-cut.p.sl.p.sl(pnewl,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<<-pnew;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 aby 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 68
```
## plot(xyxy[,1:2],xlim=c(-.5,+.5),ylim=c(-.5,.50))
## 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⟩ ≡ ⊂ 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 ⟨*debug: plot ini* 75⟩ ≡ ⊂ 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 ⟨*find value of* `lambda` 76⟩ ≡ ⊂ 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 lambda * outer polygon + (1-lambda)* inner polygon.

In former versions it happened that some lines of `h` get `NaN` values because `nrow(d.k)==2` and `k.1==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⟩ ≡ ⊂ 34

```
cut.on.pdk.1<-find.cut.z.pg(exp.dk,  exp.dk.1,center=center)
# print("HALLO"); print(cut.on.pdk.1)
cut.on.pdk  <-find.cut.z.pg(exp.dk.1,exp.dk,  center=center)
# expand inner polgon exp.dk
h1<-(1-lambda)*exp.dk+lambda*cut.on.pdk.1
# shrink outer polygon exp.dk.1
h2<-(1-lambda)*cut.on.pdk+lambda*exp.dk.1
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.1,col="blue");
#   segments(cut.on.pdk[,1],cut.on.pdk[,2],exp.dk.1[,1],exp.dk.1[,2],col="red")
#   segments(cut.on.pdk.1[,1],cut.on.pdk.1[,2],exp.dk[,1],exp.dk[,2],col="blue",lwd=3)
#   points(cut.on.pdk.1,col="blue"); cat("cut.on.pdk.1"); print(cut.on.pdk.1)
#   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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 81
```
center<-hull.center<-hull.bag<-hull.loop<-pxy.bag<-pxy.outer<-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.outer` (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⟩ ≡ ⊂ 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.outer)){ # points in loop
    points(pxy.outer[,1],pxy.outer[,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.outer)){
    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.outer,hull.bag,center=center)
      segments(pxy.outer[,1],pxy.outer[,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⟩ ≡ ⊂ 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[1,2])%*%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 Pairs plot with bagplots

A simple application of bagplot is to used 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⟩ ≡ ⊂ 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⟩

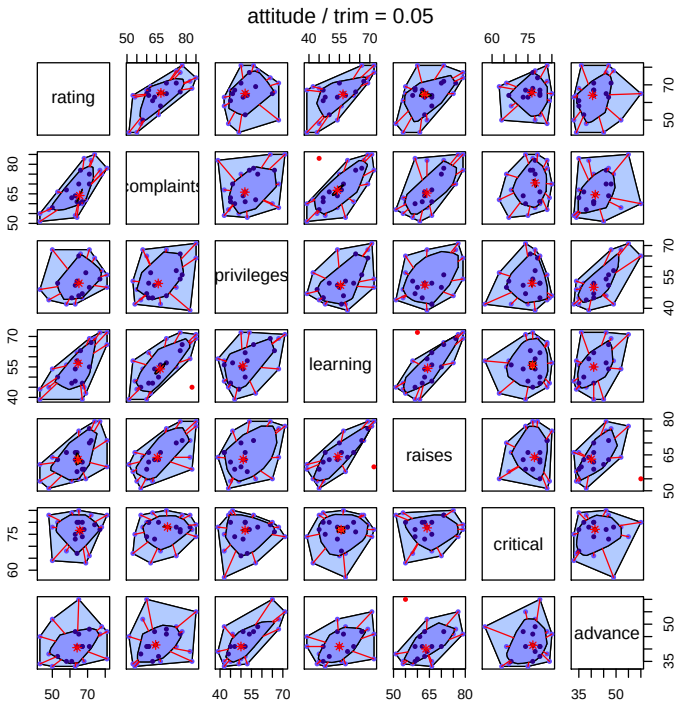
90 ⟨*define help of* `bagplot.pairs` 90⟩ ≡

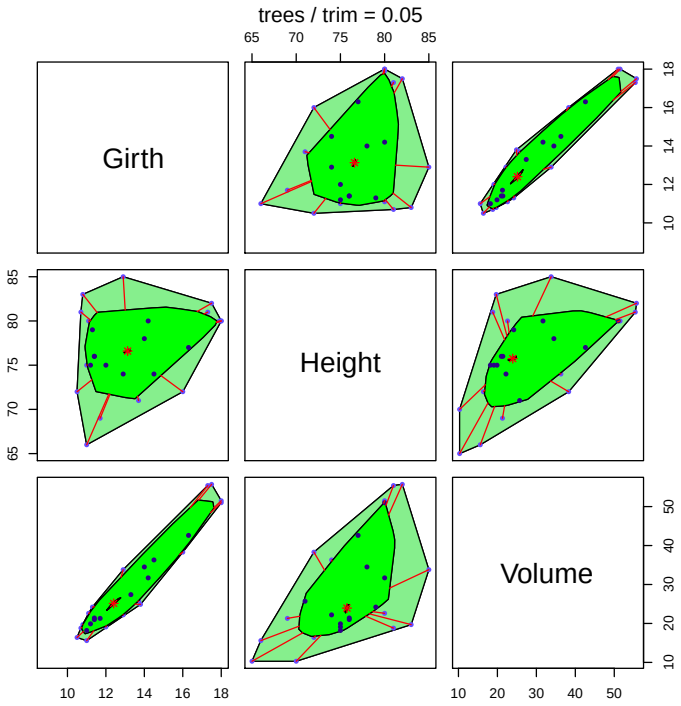
```
\name{bagplot.pairs}
\alias{bagplot.pairs}
\title{ \code{pairs} plot with bagplots }
\description{
  \code{bagplot.pairs} calls \code{pairs} and use bagplot() as panel function.
  It can be used for the inspection of data matrices.
}
\usage{
  bagplot.pairs(dm, trim = 0.0, main, numeric.only = TRUE,
```

```
                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(frabo12[,40:48],trim=0,precision=1) # a lot of variables
## bagplot.pairs(frabo12[,43:47],trim=0,precision=1) # a subset of these variables
### bagplot(frabo12[,43:47][,c(4,1)],precision=1)     ## comparison with single bagplot
### xy <- frabo12[,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(frabo12[,c(11,42,46)],trim=0.2,precision=1)[1:2,]
## bagplot.pairs(frabo12[,18:19],trim=0.2,precision=1)[1:2,]       # 0 var in both variables
## bagplot.pairs(frabo12[,c(42,43)],trim=0.25,precision=1)[1:2,]   # 0=var(Var1)&var(Var2) sm
## bagplot.pairs(frabo12[,c(1,42)],trim=0.2)                       # few mass points
## DM <- bagplot.pairs(frabo12[,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=501*c(1:100),verbose=FALSE)
  par(mfrow=c(1,1))
}
## bagplot.pairs(frabo12[,c(1,42)],trim=0.2)
### one dimensional data sets:
## bagplot.pairs(cbind(c(n[-1],300),c(n[-1],300),frabo12[,c(22,19,38)]),
##               trim=trim,precision=1)[1:2,]
## bagplot.pairs(frabo12[,c(22,38,19)],trim=0.2,precision=1)[1:2,]  # various boxplots
## bagplot.pairs(frabo12[,c(19,22,38)],trim=0.2,precision=1)[1:2,]  # various boxplots
## bagplot.pairs(frabo12[,c(19,38,22)],trim=0.2,precision=1)[1:2,]  # various boxplots
## bagplot.pairs(frabo12[,c(38,19,22)],trim=0.2,precision=1)[1:2,]  # various boxplots
## bagplot.pairs(frabo12[,c(38,22,19)],trim=0.2,precision=1)[1:2,]  # various boxplots
## DM <<- bagplot.pairs(frabo12[,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(frabo12[,11:12],trim=0.2,precision=2)[1:3,]        # check of center
## bagplot.pairs(frabo12[,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 <- frabo12[,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(frabo12[,co],trim=0.2,precision=1)[1:2,]
## bagplot.pairs(frabo12[,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)
```

6 plotsummary: A Graphical Summary of a Data 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 94, 105, 106, 154, 155

```
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, omi=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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩ ≡ ⊂ 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⟩

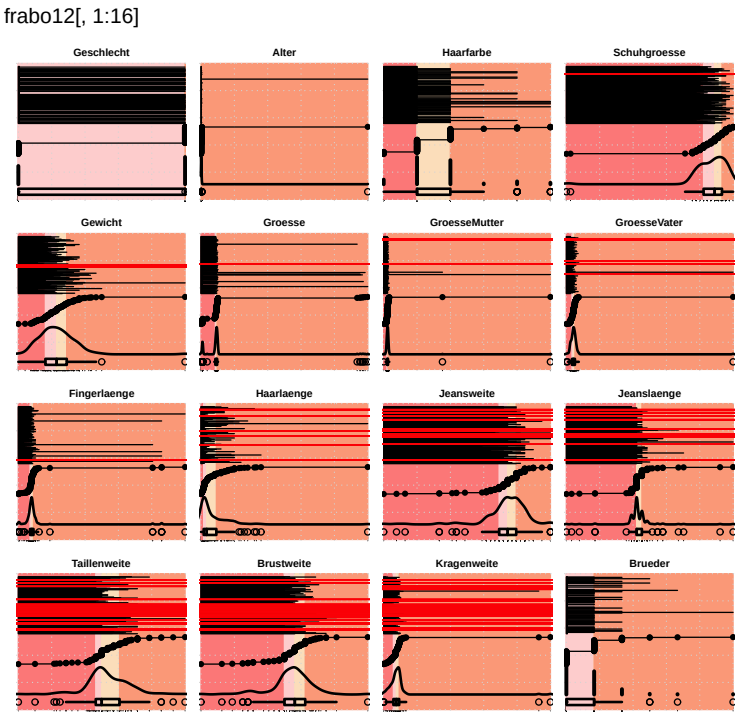
```
namelist <- ds.of.R("matrix")
# namelist <- ds.of.R("vector")
```

```
# 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 questionare 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))
```



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)
  }
```

7 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]

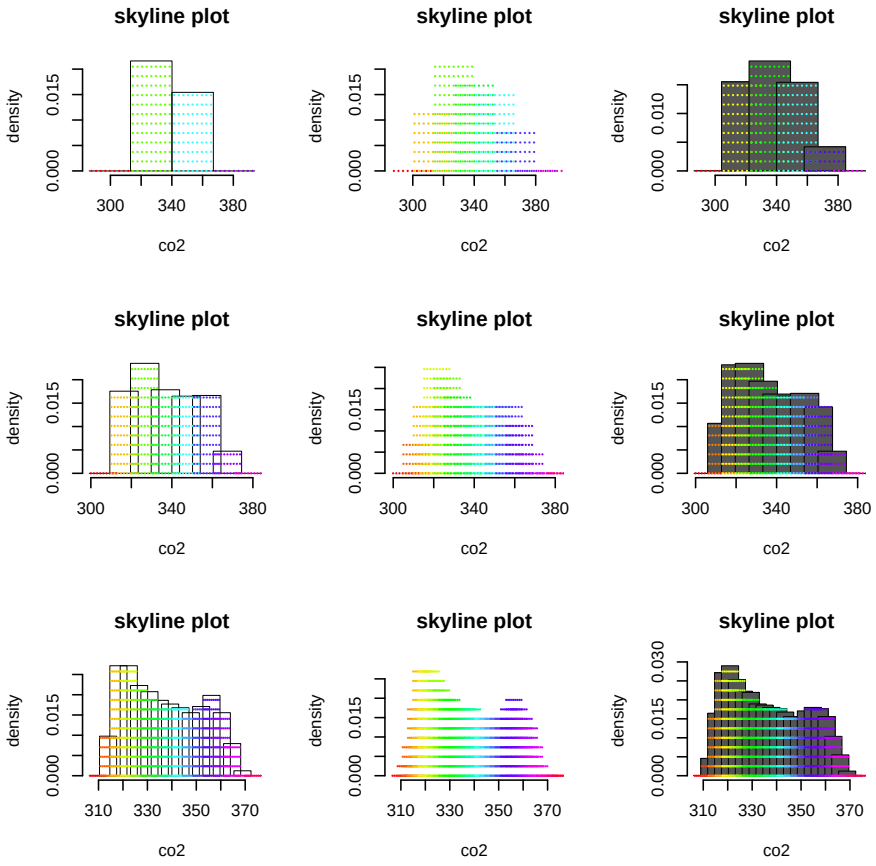
Version 1 A simple version of a skyline plot results by plotting several histograms with different starting point for first cell into the same graphics device without cleaning. If you move a window along the x-axis and represent the number of observation in the window by a rectangular box you get an 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))
 for(n.c in c(2,4,8)){  # some values for n.class
```

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

```
  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⟩ ≡ ⊂ 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)) permuation <- 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)
```

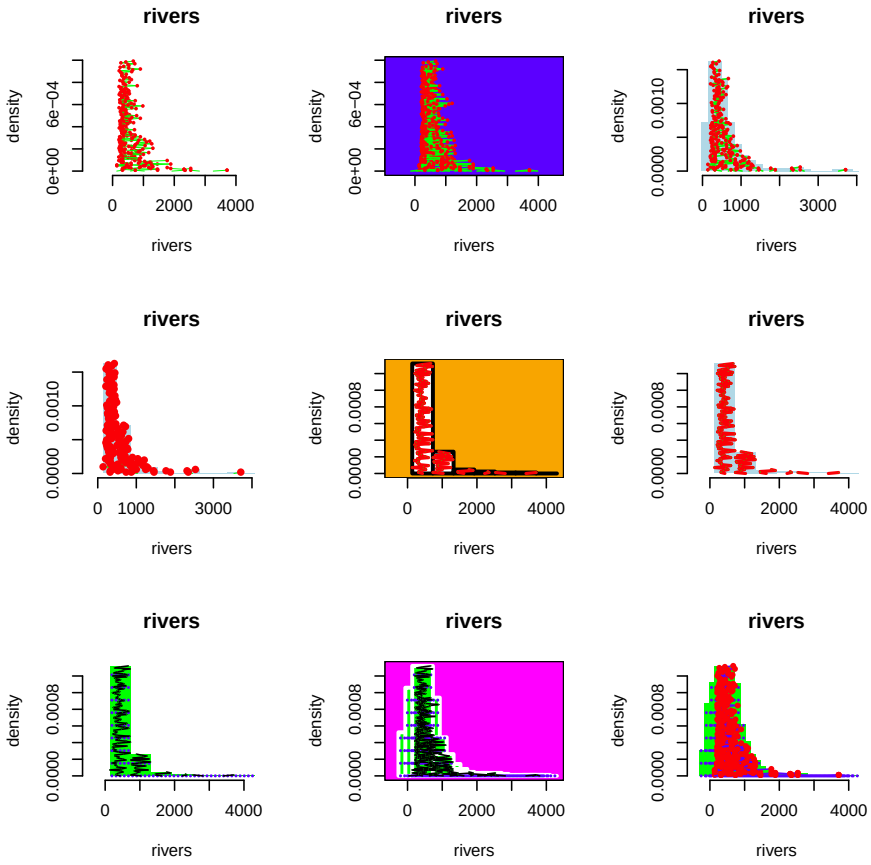
```
    plot(0, xlim=xlim, ylim=ylim, type="n", bty=bty, xlab=xlab, ylab=ylab, ...)
  title(main)
  # background
  if( night != FALSE ){
     col.night  <- if(is.logical(night)) "blue" else night
     usr <- par()$usr; rect(usr[1], usr[3], usr[2], usr[4], col = col.night)
     night <- TRUE
  }
  # add 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 Data Peeling – Plot Hulls

Sometimes data peeling is a nice way to get some insides into the structure of a bivariate data set. The idea is simple and you find some articles concerning data peeling by googling "peeling bivariate data":

Green, P.J. (1981): Peeling bivariate data.
In: Interpreting Multivariate Data, V. Barnett (ed.), pp 3-19, Wiley.

Porzio, Giovanni C., Ragozini, Giancarlo (2000): Peeling multvariate data sets: a new approach. Quanderni di Statistica, Vol. 2.

Here is a simple version to construct a peeling plot of a bivariate data set:

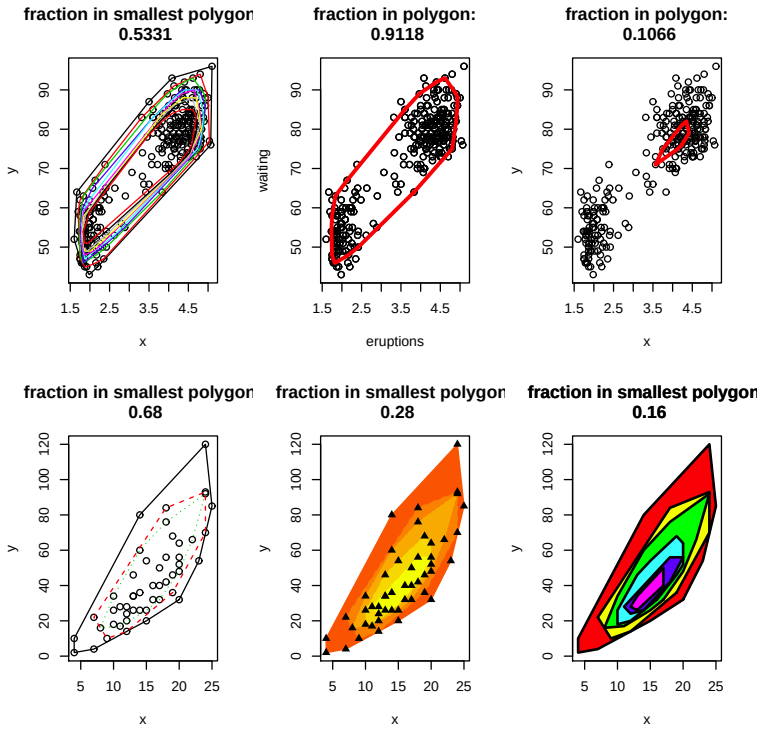
114 ⟨*define* `plothulls` 114⟩ ≡ ⊂ 117

```
plothulls <- function(x, y, fraction, n.hull = 1, main, add=FALSE,
                      col.hull, lty.hull, lwd.hull, density=0, ...){
  # function for data peeling:
  # x,y : data
  # fraction.in.inner.hull : max percentage of points within the hull to be drawn
  # n.hull : number of hulls to be plotted (if there is no fractiion argument)
  # col.hull, lty.hull, lwd.hull : style of hull line
  # add : if TRUE old plot is used
  # pw 130524
  if(ncol(x) == 2){ y <- x[,2]; x <- x[,1] }
  if(add) points(x,y,...) else plot(x,y,...)
  n <- length(x)
  if(!missing(fraction)) { # find special hull
     n.hull <- 1
     if(missing(col.hull)) col.hull <- 1
     if(missing(lty.hull)) lty.hull <- 1
     if(missing(lwd.hull)) lwd.hull <- 1
     x.old <- x; y.old <- y
     idx <- chull(x,y); x.hull <- x[idx]; y.hull <- y[idx]
     for( i in 1:(length(x)/3)){
       x <- x[-idx]; y <- y[-idx]
       if( (length(x)/n) < fraction ){
         if(missing(main))
           title(paste("fraction in polygon:\n",
                       round((length(x)+length(x.hull))/n,4)))
         polygon(x.hull, y.hull, col=col.hull,
                 lty=lty.hull, lwd=lwd.hull, density=density)
         return(cbind(x.hull,y.hull))
       }
       idx <- chull(x,y); x.hull <- x[idx]; y.hull <- y[idx];
     }
  }
  if(missing(col.hull)) col.hull <- 1:n.hull
  if(length(col.hull)) col.hull <- rep(col.hull,n.hull)
  if(missing(lty.hull)) lty.hull <- 1:n.hull
  if(length(lty.hull)) lty.hull <- rep(lty.hull,n.hull)
  if(missing(lwd.hull)) lwd.hull <- 1
  if(length(lwd.hull)) lwd.hull <- rep(lwd.hull,n.hull)
  result <- NULL
  for( i in 1:n.hull){
    idx <- chull(x,y); x.hull <- x[idx]; y.hull <- y[idx]
    result <- c(result, list( cbind(x.hull,y.hull) ))
    polygon(x[idx], y[idx], col=col.hull[i],
            lty=lty.hull[i], lwd=lwd.hull[i], density=density)
    x <- x[-idx]; y <- y[-idx]
    if(0 == length(x)) return(result)
  }
```

```
  if(missing(main))
    title(paste("fraction in smallest polygon:\n",
                round((length(x)+length(x.hull))/n,4)))
  result
} # end of definition of plothulls
```

115 ⟨*examples of* `plothulls()` 115⟩ ≡

```
xy <- faithful; #xy <- cars
#library(aplpack)
par(mfrow=c(2,3))
# 1. plot
plothulls(faithful,n.hull=10,lty.hull=1)
# 2. plot
plot(xy)
plothulls(faithful,fraction=.90,add=TRUE,col.hull="red",lwd.hull=3)
# 3. plot
plothulls(faithful,fraction=.10,col.hull="red",lwd.hull=3)
# 4. plot
n <- 3
plothulls(cars,n.hull=n,col.hull=1:n,lty.hull=1:n)
# 5. plot
n <- 5
cols <- heat.colors(9)[3:(3+n-1)]
plothulls(cars,n.hull=n,col.hull=cols,lty.hull=1:n, density=NA,col=0)
points(cars, pch=17, cex=1)
# 6. plot
n <- 6
cols <- rainbow(n)
plothulls(cars,n.hull=n,col.hull=cols,lty.hull=1:n, density=NA,col=0)
x<-plothulls(cars,n.hull=n,add=TRUE,col.hull=1,lwd.hull=2,lty=1,col=0)
par(mfrow=c(1,1))
```



116 ⟨*define help of* `plothulls` 116⟩ ≡

```
\name{plothulls}
```

```
\alias{plothulls}
\title{plothulls for data peeling }
\description{
  \code{plothulls plots convex hulls of a bivariate data set.}
}

\usage{
plothulls(x, y, fraction, n.hull = 1, main, add = FALSE, col.hull,
    lty.hull, lwd.hull, density = 0, ...)
}

\arguments{
  \item{x}{ two column matrix of the coordinates of points of x-values of a data set}
  \item{y}{ if x is one dimensional then y contains the y-values of the data set}
  \item{fraction}{ ... of points that lies inside the hull to be plotted}
  \item{n.hull}{ number of directions sequential hulls to be plotted}
  \item{main}{ title for the graphics}
  \item{add}{ if TRUE no new plot is initialized}
  \item{col.hull}{ color(s) of the hull(s)}
  \item{lty.hull}{ line type(s) of the hull(s)}
  \item{lwd.hull}{ line width(s) of the hull(s)}
  \item{density}{ density argument of polygon() that draws the hulls}
  \item{...}{ further arguments used in the call of plot() or points()}
}

\details{
The function \code{plothulls} computes hulls of a bivariate data set using the
function \code{chull}. After finding a hull the hull maybe plotted.
Then the data points of the hull will be removed and
the hull of the remaining points is computed.
The style of plotting a hull depends on the setting of
\code{col.hull}, \code{lty.hull}, \code{lwd.hull} and \code{density}.
\code{density=NA} has the effect that the regions of the hulls are filled by a color.
Using \code{fraction} you can plot a single hull.
\code{n.hull} defines the number of hull that should be drawn one after the other.
}

\value{
The points of the hull(s) are stored in matrices with two columns and
the matrices are gathered in the output object.
}
\references{
  Green, P.J. (1981): Peeling bivariate data.
  In: Interpreting Multivariate Data, V. Barnett (ed.), pp 3-19, Wiley.

  Porzio, Giovanni C., Ragozini, Giancarlo (2000):
  Peeling multvariate data sets: a new approach. Quanderni di Statistica, Vol. 2.
}
\author{ Peter Wolf }
\note{
  Version of plothulls: 10/2013 }
 \seealso{ \code{\link[aplpack]{bagplot}} }
\examples{
  # 10 hulls computed from the faithful data and plotted
  plothulls(faithful, n.hull=10, lty.hull=1)
  # plotting additionally a hull with 90 percent of points within the hull
  plot(faithful)
  plothulls(faithful, fraction=.90, add=TRUE, col.hull="red", lwd.hull=3)
  # hull with 10 percent of points within the hull
  plothulls(faithful, fraction=.10, col.hull="red", lwd.hull=3)
  # first 3 hulls of the cars data set
  n <- 3
  plothulls(cars, n.hull=n, col.hull=1:n, lty.hull=1:n)
  # 5 hulls represented by colored regions
  n <- 5
  cols <- heat.colors(9)[3:(3+n-1)]
  plothulls(cars, n.hull=n, col.hull=cols, lty.hull=1:n, density=NA, col=0)
```

```
  points(cars, pch=17, cex=1)
  # 6 hulls: regions colored and boundaries shown
  n <- 6
  cols <- rainbow(n)
  plothulls(cars, n.hull=n, col.hull=cols, lty.hull=1:n, density=NA, col=0)
  plothulls(cars, .hull=n, add=TRUE, col.hull=1, lwd.hull=2, lty=1, col=0)
}
```

117 ⟨*start* 85⟩+ ≡
⟨*define* plothulls 114⟩

9 Appendix

9.1 Some further examples – usefull for testing

118 ⟨*define rnorm data* data, *seed:* seed, *size:* n 118⟩ ≡ ⊂ 3, 4, 16, 17, 119
```
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))
```

119 ⟨* 18⟩+ ≡
```
seed<-222; n<-100
```
⟨*define rnorm data* data, *seed:* seed, *size:* n 118⟩
```
datan<-rbind(data,c(106,294)); par(mfrow=c(1,1))
datan[,2]<-datan[,2]*100
```
⟨*define* bagplot 32⟩
```
bagplot(datan,verbose=TRUE)
```

120 ⟨*quadratic-test* 120⟩ ≡
⟨*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)
```

121 ⟨*error* 121⟩ ≡
⟨*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))
```

122 ⟨* 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"
```

9.2 Data Sets of AJS

9.2.1 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.

123 ⟨*define data sets of AJS* 123⟩ ≡ ⊂ 124, 125, 126
```
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,
dat.AJS.NA.11 <- structure(c(-0.279565168686504, 0.491941365497838, -0.173602939876011,  -0.19685818172603, -0.534170026008371, -0.190080138935935, 0.504448367630096,
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,
```

```
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, 0.666905150541366,  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 artificual 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.

124 ⟨*test of amanda-data-sets* 124⟩ ≡

```
⟨define data sets of AJS 123⟩
⟨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 occured by very small data sets or one dimensional cases.

125 ⟨*test of amanda-data-sets* 124⟩+ ≡

```
⟨define data sets of AJS 123⟩
⟨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)})
}
```

126 ⟨*a verbose test with AJS data no 26* 126⟩ ≡

```
⟨define data sets of AJS 123⟩
⟨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)})
```

9.2.2 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:

127 ⟨*define function* `get.AJS.data()` 127⟩ ≡ ⊂ 128, 129, 134, 135, 137, 138, 139, 140, 141, 142, 143

```
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="")

128 ⟨* 18⟩+ ≡

```
⟨define function get.AJS.data() 127⟩
#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 -0.0858163142248452998073 2
RMS.T6  -0.4965104573284281896939  0.1028110005618869393639 9
RMS.T7  -0.4054359073563830184739 -0.1126735743928821281700 8
RMS.T8  -0.5193619418570573076366  0.0189826012344787950131
RMS.T5  -0.3986393540789945899583 -0.0562691602375199814711 8
RMS.T3  -0.4759403757463211426249 -0.0266642412103347166418 8
RMS.T10 -0.4440413505805637095492  0.0068285346810906073430 9
RMS.T11 -0.5316323986960421743575 -0.0884445049079400219538 3
"
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")
```

9.2.3 `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.

129 ⟨define data set with chull problem 129⟩ ≡ ⊂ 133

```
⟨define function get.AJS.data() 127⟩
# former name: sensitiv1
dat.AJS.chull.1<-"
  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,  0.490179090,  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,  0.1939269163,
  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,  0.0322623727,  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, -0.2068502412,  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.AJS.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,  0.2246294927,  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,  0.487757526,  0.3732180676,  0.440667092,  0.4203912650,  0.408446801, -0.1618030914,
  0.384406713, -0.0261348064,  0.328457377,  0.1821870524,  0.331531072,  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,
  0.415382189,  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, -0.0176971730,  0.427558337, -0.0231909552,
  0.432525050,  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, -0.0088666045,  0.491099424,  0.3022186063,  0.418721732,  0.1857572104,
  0.390276222,  0.3154919032,  0.399469456,  0.0498277648,  0.454754097,  0.3038404628,  0.435410153,  0.1737382460,
  0.445867263,  0.3164638891,  0.458967530,  0.1518767404,  0.335919344,  0.0823980295,  0.412763921,  0.3319087519,
  0.471049431,  0.2156398459,  0.400699273,  0.1395100302,  0.374861658,  0.2794049727,-0.138108934,  0.1904884369,
  0.280570722,  0.1983537837,  0.420120277,  0.2619778732,  0.454057069,  0.3034526292,  0.395791685,  0.0050987952,
  0.365260425, -0.0272468779,  0.484034991,  0.3569887088,  0.370863417,  0.1093298911,  0.461277753,  0.2632357520,
  0.350125223,  0.0183460251,  0.368930135,  0.2207677738,  0.531849230,  0.2197869496,  0.416717917,  0.1982293252,
  0.414321880, -0.0860104879,  0.438841302,  0.0510552491,  0.398298536,  0.0872398038,  0.306217635, -0.0149914778,
  0.469373834,  0.1709017285,  0.415929594,  0.1800595546,  0.385559535,  0.2280161638,  0.445047862,  0.1061824345,
  0.450436609,  0.1659285617,  0.105991452,  0.2547355455,  0.401561850, -0.0054767733,  0.068325138,  0.1882970253,
  0.428084182,  0.1258825829,  0.386663185,  0.2788034267,  0.364941456,  0.1938777481,  0.388006007,  0.2488389434,
  0.313331553, -0.2704348401,  0.352922806,  0.0215689704,  0.448430163, -0.0917932667,  0.431449465,  0.3508000866,
  0.432438166,  0.1241467239,  0.428358234,  0.0088965389,  0.451061147,  0.1955063904,  0.438706054,  0.4033386057,
  0.375774923,  0.1732898787,  0.519488312,  0.2018627370,  0.440325400,  0.1348568173,  0.490158077,  0.3545691684,
  0.504032360,  0.2314774746,  0.465533674,  0.3347213235,  0.351902301,  0.3282132558,  0.487045154,  0.0902528398,
  0.429961565,  0.0480713261,  0.437203684, -0.0127309372,  0.469321808,  0.2620019309,  0.405157393,  0.1902570513,
  0.464512601,  0.2275906537,  0.396079699,  0.1754275458,  0.441935381,  0.2202391505,  0.464574093,  0.2507926309,
  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,  0.506548976,  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,  0.275594490,  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.AJS.chull.1 <- get.AJS.data(dat.AJS.chull.1)
dat.AJS.chull.2 <- get.AJS.data(dat.AJS.chull.2)
bagplot(dat.AJS.chull.1,main="dat.AJS.chull.1")
points(dat.AJS.chull.2,col="green")
bagplot(dat.AJS.chull.2,main="dat.AJS.chull.2")
points(dat.AJS.chull.1,col="green")
summary(dat.AJS.chull.1-dat.AJS.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:

130 ⟨*Example where chull gives bad results* 130⟩ ≡

```
# extracted from bagplot constrution 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]
 0.4901580770000000253361 0.2446332641259137408 1629
 0.4845158797159134222987 0.1422398099101958579 5598
 0.4661984398335061174912 0.0631969378331613262 9224
 0.4841546267832931293995 0.1406809419939438476 8986
 0.4632368748395896962400 0.0504172806067827716 7916

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 polynom.

131 ⟨*check clockwise condition of polygon* 131⟩ ≡ ⊂ 133, 136
```
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:

132 ⟨*define another chull function* 132⟩ ≡ ⊂ 133, 136
```
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

133 ⟨*define experiment to check chull error and solution* 133⟩ ≡
```
⟨define another chull function 132⟩
⟨define bagplot 32⟩
⟨check clockwise condition of polygon 131⟩
⟨define data set with chull problem 129⟩
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$hull.bag), "\n")
  points(dat.AJS.chull.1,col="green")
  summary(dat.AJS.chull.1-dat.AJS.chull.2)
  ⟨define another chull function 132⟩
  assign("chull", chull, env=.GlobalEnv)
  bagplot(dat.AJS.chull.1,main="dat.AJS.chull.1 / new chull")
  rm("chull", chull, env=.GlobalEnv)
  points(dat.AJS.chull.2,col="green")
  S2new <- bagplot(dat.AJS.chull.2,main="dat.AJS.chull.2 / new chull")
  cat("\ncheck of bag:", check.sense.of.rotation(S2new$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.

9.2.4 Missing Fence on 64 Bit Machines

134 ⟨*Amanda's data set of 29Oct2012: missing fence – A* 134⟩ ≡ ⊂ 136, 144

```
⟨define function get.AJS.data() 127⟩
a<-"      [1]                                  [2]
-0.855420657131037431852 86,   2.151159411763307272025 26,-1.192086710933312643234 67,   3.667744522828495057353 85,
-0.443329944112196561167 85,  -0.213171761410803212166 70,-0.953225827162288541494 21,  -0.797583739370120503586 75,
-0.854796637251045154926 03,   3.611757123247781642305 65,-0.912684454856468541095 18,   1.279499714502437379692 88,
-1.029722420351208977962 19,   2.626343587005343760400 91,-1.065784517865945746351 03,   2.407977444938864675094 69,
-0.994621293294002839502 35,   2.194612041767711085272 435,-0.936153048643899476921 60,   3.987859028135142480664 40,
-0.876333193059151471793 20,  -0.822480443221069390524 75,-0.911608495296858944989 54,   0.712556001898259294691 23,
-0.715537672739628538565 17,  -1.126082540288954714213 79,-1.000893986086435427651 51,  -1.402879598521188152635 87,
-0.891489987093591507161 25,   1.468709941888894476846 97,-0.366056925910315800187 79,   4.160287920218812551809 15,
-0.685950892159732261710 30,  -1.851460618281322023648 27,-0.876271749842640335081 97,  -1.922219699819748495173 14,
-1.140527179800549184562 90,   1.335954624218717857827 682,-0.888335801375641498367 49,   0.964571227509928186272 25,
-0.819328572015867395705 87,  -1.651771209475026980229 02,-0.866588550882131047892 19,   0.408554335373432886413 53,
-0.736737657284542768465 49,  -0.497736872352315951051 34,-0.885541412200497535245 35,   0.869946474423871962677 88,
-0.793608844077075215039 94,   1.331977230799773925795 65,-0.751102978295256629159 84,  -1.864296852147544925060 17,
-0.886033208654472215215 49,   2.163009404129667245797 46,-0.920841159611511472782 30,   2.427774888755126703188 13,
-0.978468522920641795239 85,  -0.860814251334382052327 50,-0.802003884176517578552 53,   3.357898265637765611302 26,
-0.828180222453862913134 51,  -1.410813202810052624514 01,-0.977806248865534466396 81,  -1.968039704120108757123 83,
-0.841389413339612923795 29,  -0.854197591313215487396 35,-1.112481857782169125670 18,  -1.750472699187987979385 64,
-0.908610927001686441428 77,  -0.031545633131174144470 07,-0.912229155018748127935 69,  -0.168751756460290586181 47,
-0.741253392430125002121 60,  -1.507125527105744211555 43,-0.920810873059566992182 57,  -2.349574450974894368471 27,
-0.892653688391205957231 02,  -1.586822334259982758555 06,-0.645332490517684598252 40,  -0.639519708426754540653 29,
-0.739606990122553931140 01,   3.080523580437731112624 53,-0.276402997655908810870 07,   0.975817930300824198930 34,
-0.732981224283015264655 33,  -1.137149501658584860308 57,-0.796230112202567452683 87,   2.636660142214514390701 65,
-1.008352432303160295390 40,  -1.409453391484073447870 17,-0.874789180628348317370 07,   1.246756964718999771690 73,
-0.939468911629592340339 91,  -2.235758872857151136059 88,-1.041130283414992252488 67,  -0.433326937634337550079 97,
-0.762584415977154317367 59,   3.690764002802367826916 45,-0.578368096580188173483 35,   0.582360630116827482362 97,
-0.699774569047176497882 39,  -0.427840784203837642962 75,-0.937906214788275072180 57,  -1.239388143954733170086 25,
-0.881012640644001865375 40,   1.040364567596525846582 27,-0.895657162396526151937 34,  -1.152138994942982996860 79,
-0.880736613483969810189 93,  -0.463364300566339260001 31,-0.833541377036084107921 00,  -0.271051606559429059117 86,
-0.896225027132653173644 88,  -0.125077981234404222998 79,-1.125637526132711840887 17,  -1.621672854945781727664 67,
-0.654033248113866316764 13,  -0.250837291232049119926 73,-0.791419966818841746913 90,  -1.753753627954262928057 01,
-0.813424940799973517080 45,  -0.569448226402150314839 90,-0.635454640684833149677 50,  -1.054794890855101918702 99,
-0.967189037966913689459 15,   0.981942307234637712376 47,-0.711736331021626585925 94,   0.086622854639063454063 04,
-0.912425345337034125184 45,  -1.399574831997170365482 93,-1.097257501732441298258 89,   3.971781733248187062912 39,
-1.210522680103697013009 88,  -1.487519545276482224593 91,-0.819081080311540477012 50,  -1.271983624613456376195 59,
-1.146063062486186723276 66,  -0.444622390944514667143 75,-0.839516622346073382310 29,   1.973830809529668129999 91,
-0.692815680114740706230 24,  -1.087391869546970868397 2,-0.927027653776066477503 747,  -1.906389338373944086765 7,
-1.084759905249725209230 86,   0.540837688687349937843 86,-0.748048140368241853970 23,  -1.427252178079271294208 75,
-0.785181512355952349047 07,  -1.997285129810227211066 83,-0.860172138036781830724 9,   -1.731527477794833957958 08,
-0.996706627790669430488 89,   2.162617376234937438539 42,-1.004030223040532465716 98,   0.242562668222902927395 17,
-1.014678219003398274011 83,  -2.071165949950680928996 0,-1.002477984501642493953 00,  -0.149130700692698264786 53,
-0.964383345761325450062 88,   1.162087126459359520680 44,-0.561634121284444454147 43,   1.048163972875856719824 13,
-0.714345039382573543385 04,  -0.357838060340416430626 93,-0.745324094051362240342 94,  -1.753453387596963120032 00,
-1.009198594124325996832 43,   0.827972200345477249427 97,-0.941840021203927624249 27,   1.906076493216868117386 30,
-0.993947051835936901120 76,  -1.939643593882150574003 73,-0.872558760981079657075 88,  -2.163817527598550949363 70,
-0.626390848861929793523 22,   4.358236393710335221385 32,-0.572475736587328190901 05,   3.543782194331181134572 26,
-0.623152684615371543763 73,   0.138624880408372064266 59,-0.654198415657576637372 02,   0.923699017234647978824 57,
 0.033098166986774785192 08,  -0.135571409742803322684 42,-0.877737870836168632138 2,    2.908150954709151303489 99,
-0.690201086849552614133 07,  -0.997037896712004267918 42,-0.740956944783004445298 50,   2.107124182128131906210 91,
-0.783112709744371304765 31,   1.359436407894501597226 62,-0.721992814367621971172 18,  -1.165359756423399062441 60,
-0.739801156158444439547 85,  -2.072421153082102840414 79,-0.856317502694422305609 12,  -2.365872363566009085642 63,
-1.041277391048016642827 18,  -1.924505766386529037603 74,-0.809282984121624915374 58,   0.723920400111141670862 48,
-0.935830323554082355030 91,  -1.271671977417311083513 60,-0.951386760111438523068 73,  -2.483357683093537549012 81,
-0.695108735256396603752 19,   0.740034253628844451711 9,-0.708282927642307269522 35,  -1.740294568434479716145 80,
-0.817026369556514753256 24,  -2.056366767673182049946 94,-0.855721665954261689890 36,   2.782178248451977253807 85,
-0.867411282913804648764 26,   1.120331861847988941249 81,-0.963085751872730488898 36,   1.823445753835751581561 91,
-0.931590495648666858172 62,   3.208342291315456229483 52,-0.681519079823434625886 59,   0.182212510818523659716 92,
-0.924001779825381008848 24,   0.872702981765764995714 11,-0.975388754801635560198 75,   2.184330796690547416006 76,
-0.999191042661040085270 94,  -1.259718053207356414091 58,-0.981206497088656592886 48,  -1.544889640630292770850 93,
-0.808518277345179448012 40,  -2.315050044672998907913 10,-0.856031945420649753586 42,  -1.413215655041623941556 85,
-0.773251444669914023144 49,  -1.300923359134056100927 52,-1.010473784682569542425 77,  -1.217272029748613038080 41,
-0.385296892099502682580 29,   0.123141764822012320101 94,-0.719739817212379562327 84,  -1.567741408399109914739 71,
-0.760059024505841174246 26,   1.108371204696664724664 86,-0.610646831533551370796 43,  -1.139391752782737299289 79,
-0.768073578288774982958 39,  -0.002691276880956401836 02,-0.893177621933753274063 86,  -1.420089033998526595329 98,
-1.012544084203238270447 40,   1.644123447923203196197 05,-0.682772852885122683019 86,   0.331004198680273753474 52,
-0.815895628663555250170 70,  -1.997599048523243547847 98,-0.586586795063119947890 58,  -1.462986147739081532748 17,
-0.991748114926505703688 91,  -0.268644328633148410379 05,-0.778282502497298689370 10,   0.028583002899000294124 95,
-0.997848144923612845325 07,  -1.018016827733819518186 92,-0.517711750604285869670 21,  -1.420440927921023055446 88,
-0.884204936513267147901 73,  -0.987975602908339856078 61,-0.991058791936399008370 98,   0.309883105856517295367 31,
-0.914695960885059222711 39,   0.209331517359406221734 37,-0.610859999734173930718 84,  -0.587784807441514844938 98,
-0.960824128472530292199 85,  -0.027600557606164088930 71,-1.009083836420810786904 67,  -0.233421208304862154436 78,
-0.534951374268648360832 40,   0.671354017115542767157 88,-1.003516066012093421733 20,  1.619751518203727425060 38,
-0.861630253674875667968 54,  -1.103138135593914270771 17,-0.165955572710422522009 21,  -0.949436374983996023146 91,
-0.793406360911937613700 22,   0.851801965025307472600 96,-0.982191213474534974281 48,  -0.351579571514637034823 64,
-0.701715629900809489960 52,  -1.619870228133348134325 78,-0.853805581305348448850 62,   0.102344469645983274230 79,
-0.911199997583760912256 42,  -1.962704242119229114393 87,-0.473694104977876111834 36,  -1.245865979510560737963 9,
```

```
-1.124812089761297206891 30,  -1.429556499562588012253 17,-0.685914980291368991949 24,   0.240869703295181869506 79,
-0.761597390058189716022 24,  -2.176288224070687383240 76,-0.307741767533903309850 27,  -0.947838832204756176480 49,
-0.690094171664437205926 39,  -0.403848730235128183796 43,-1.092158564167053702576 03,   3.064840303584433376803 08,
-1.135473320401710006777 31,  -0.043041500123497111740 53,-0.947157704977940051449 54,  -1.317659176525090014564 51,
-1.075449338881940386514 20,  -1.824928984599170078695 35,-0.825693832273404315603 90,  -1.354904920904387743973 98,
-0.948371534149655248668 86,  -1.422557799651877896707 22,-1.036353411231889065646 78,   2.725926996509557387327 05,
-0.599324708796660665655 05,  -0.797643866605951656900 69,-0.703245744084900925585 66,  -0.561178868597641544724 30,
-0.797140908593619745836 63,  -0.488640625786258331331 17,-0.727799613947915813305 66,  -0.268032838349270385862 26,
-0.837543471612699264028 38,   0.560486972267733252195 40,-0.688587194571721461500 15,   2.006793159493500233736 54,
-0.826742053348380800947 84,   4.949266117518491547855 32,-0.699871600728488751919 09,  -1.997032636375854593735 9,
-0.764654482389601430900 281,  0.357731570271524190740 56,-0.862197551824676833298 61,  -1.187945972982124898109 65,
-0.754928694469573446923 50,   0.324989025185576374532 56,-0.845168744646929814834 38,  -1.794516969457027456457 7,
-0.620976535309343824486 92,  -0.589687716202551226450 04,-0.890845225056212997039 04,  -2.097698269094068646012 37,
-0.586144977559641122688 56,  -1.421168875198343783949 89,-0.782798989179400273037 05,  -0.228837181727349736615 20,
-0.721845504792575365371 48,  -1.142514917001115737349 44,-0.655484504428277214849 17,   2.804601583052321434053 06,
-0.717397389393997442219 59,  -1.554668882768101001445 87,-0.799816571141275999323 7,   -2.008283629900768474385 59,
-1.029529234245229796226 4,   0.850869611794597413556 57,-1.046537942587759673074 29,  -1.589413919328555591306 61,
-0.846686046258470990232 061, -1.347030542518165674437 11,-0.848391491779160289432 58,  -1.436660232597843922519 59,
-1.105492012378893829090 430, -0.602788276641058384797 13,-0.754687100776535002566 45,  -0.651059859804427640379 2,
-0.842563479472295262517 89,  -1.551587801201981911874 13,-1.087021822445344954033 43,  -1.175796565194322511826 29,
-1.006011349780730590453 00,  -2.401184034191671035074 48,-0.711937928042486456448 05,  -0.087707619076003565106 17,
-0.546461884120859711971 01,  -0.835376796567778190372 17, 0.288800180171868703382 64,  -0.255397574118159775835 58,
-0.782386169740001746752 97,  -2.041815364576097469040 41,-0.901252402621751325106 91,   0.629300840608310152468 2,
-0.725326624858695701014 71,  -1.147485648355581755453 16,-1.123528744533478507250 38,   2.950874523989324593734 27,
-0.818857564489966716084 43,  -0.722292575986663631937 29,-0.921100618614981381249 15,   1.455638367044691516838 38,
-0.563752045428997705656 12,  -0.722941211360848101641 35,-0.828517902335194511564 96,  -1.184351698086176574165 05,
-0.713495633745694357585 88,  -1.737868423591597544231 04,-0.749426075952371317789 07,   0.518999422176206315349 13,
-0.132495783591123100775 28,   0.299691338837304754871 84,-1.065849314845664919104 71,  -1.098091262397145273510 55,
-0.853308217953350434111 52,   1.442751794894474137720 41,-1.150131134758225481817 82,   0.031860789369762776424 01,
-0.847997178462861489478 54,   0.745140932680186329761 80,-0.950200730724008346687 32,  -1.158037455502253454399 0,
-0.752284168073079295702 14,   3.816625600191769329683 21,-0.738452736254614383604 18,  -2.634858541274204490889 59,
-0.634830411322391929829 22,  -1.967169546257666615727 400,-0.730098041144948584069 14,  -2.401749823504675251939 49,
-0.835572197888652223873 86,  0.837832349260015041281 24,-0.854735119257594488928 48,   4.797478922033581837849 105,
-1.131289427686515969284 64,  -1.936980574343863503727 90,-0.966686348735060985148 95,   0.708761971754523045774 48,
-0.780196936666323948017 52,   3.447132775281971817094 02,-0.703915823619877707778 60,   1.982358489953494506324 6,
-0.625363330395638850056 63,  4.329771999477205033588 912,-0.812221082744631339700 45,  -1.405008956862130897746 97,
-0.779022622499025874098 78,  2.895925381440024981571 92,-0.849936478889276370018 99,   0.458118201442989647631 52,
-0.814734571186332523495 82,  1.138886220523521064507 12,-0.806049665031660866887 36,  -1.360196278709920880706 68,
-1.097459821917879940755 82,  2.637788862935642519857 99,-0.726058094039651935602 82,  -0.037551689343394104025 90,
-0.352096839871934241816 83,  0.301250763796555021123 42,-0.913720485896797884888 32,  -1.553100448582752542418 699,
-0.810942577089295402181 79,  0.032829541007722715106 89,-0.998179610367300407247 63,   2.914189865090125231006 37,
-0.589428846697784170061 14,  -0.385613136552289459402 23,-0.807865030232360111514 43,   1.727869676102228302383 42,
-1.072763579188325433477 76,  -1.165861658440526360536 18,-1.072952517468948485657 17,  -2.248873285629191087764 41,
-0.850017100546459847265 62,  -1.343493956532426869545 43,-0.758448770671926841835 39,  -0.967396408335569168990 28,
-0.711019336498914578115 20,   0.471082722381293395041 98,-0.713492678603047769698 4    -0.618348272762892725736 4
"
dat.AJS.fence.1 <- get.AJS.data(a); bagplot(dat.AJS.fence.1, main="dat.AJS.fence.1")
```

135 ⟨*Amanda's data set of 29Oct2012: missing fence – B* 135⟩ ≡ ⊂ 136, 144

⟨*define function* `get.AJS.data()` 127⟩

```
a<-"
0.48856592503641266267422 3, 0.47888302302859792058953 30, 0.47051888289952842336916 2, 0.47553799030055859953236 56,
0.54082131890095819226616 0, 0.34525846127902776183660 19, 0.54388032928448393210629 8, 0.23217643862129680454309 27,
0.33469584681099928680225 7, 0.23293435920067473432482 07, 0.31000537768508301672554 2, 0.27387147647964571461898 23,
0.47209174967402439682118 1, 0.34955176103839707968390 37, 0.47045313399028776402133 2, 0.45635241475112470199704 26,
0.23943753896362157673749 5, 0.34508034385607072591639 62, 0.40760574219499509653286, 0.27706316066223635181610 541,
0.40529790993667497156496 1, 0.35640389700036867459687 07, 0.42669894690102722698501 40, 0.29597174982669323200756 685,
0.47407897495162676371194 3, 0.06321190817067089129377 42, 0.51816566945943309807631 7, -0.020802630973436875283 5110,
0.38918269919408848833697 3, 0.30936623692824516407284 82, 0.38686337916355734156681 5, 0.36391620114880268710422 23,
0.42594378693605444707807 03, -0.08092168373472555524282 019, 0.39711398779691048366302 7, 0.00089992794569759259501 583,
0.43814451552969074787924 3, 0.20933608577997889210742 24, 0.32844675169763321020965 9, 0.27513622293305844966800 327,
0.46187619248090056034428 1, 0.06685441256599625185160 029, 0.41470234144785461660731 6, 0.00732361072538633296052 92,
0.43210626066468205142001 6, 0.18796194887775066750812 41, 0.39572260448503776553153 0, 0.15314479928344676418205 95,
0.33344523904785966506025 8, 0.19652949206738282161843 76, 0.40168109396361562080102 2, 0.09468428597847232397466 40,
0.40086627817486497615817 6, 0.14719840079219478901073 61, 0.46476240020940312591690 7, 0.20098675880294772366774 23,
0.54978816087283266522689 5, 0.12781601144508958345369 36, 0.38120963976835919861940 7, 0.26687518911558766321334 70,
0.55000915281699713776930 6, 0.06689406828095119383892 35, 0.47723163268967006001020 1, 0.07412048595186608579422 52,
0.45855041895945292118597 1, 0.07762612868420031009275 56, 0.52887657797324238995173 6, 0.09491986251560149967509 97,
0.48646749211663609102629 4, 0.18453734896423878986126 97, 0.41860128767156823403894 4, 0.09916223103698484919732 44,
0.43144646527195108109964, -0.09543613921835285285322 21, 0.41079128450954551983851, -0.13448279040705146668778 63,
0.39578119982456522185643 2, 0.12286908099098242663771 43, 0.46933244296989407029485 6, 0.15082541558498141487376 64,
0.50304905538070321391330 7, 0.18506466030288989955775 2, 0.57358352906388199166087 8, 0.37072674826103207967875 62,
0.53954016066745646984514 9, 0.44115877412709952887226 45, 0.45779941102066173952422 9, 0.44064252852202029397002 01,
0.38997931025722110121023 4, 0.13275724395459670423846 89, 0.39501499405468804004826 2, 0.10975406909789155174284 05,
0.43914913163883134705756 1, 0.41936259915268647713304 52, 0.12156835558493635229826 4, 0.43398165808766698637325 717,
0.33545096688769204806845 9, 0.24734798684631367771125 10, 0.35141158854662712940353 2, 0.22462949266892417998775 50,
0.50693153066469009981887 0, 0.23896868034870433206151 57, 0.41645389077686451040349 9, 0.21507140069602379006674 65,
0.44572830666945667532630 2, 0.21025989831373559368898 90, 0.46627071254967616953379 7, 0.08438923462924155516340 36,
0.48720764578100217434553 1, 0.30591516029983006896841 86, 0.46977731826359053757258 3, 0.39578165218075062048441 74,
0.37706789679958063743470 1, 0.15901448191054662339993 80, 0.45314979079796743054185 8, 0.12159620918912614206958 20,
0.48775752613048467543046 65, 0.37321806763697967790349 71, 0.40066709214955085416320 9, 0.42039126495417139487979 441,
0.40844680062487898064560 5, -0.16180309140021292835243 12, 0.38440671255925834559619 1, -0.02613480637285424054705 45,
0.32845737739278318967350 9, 0.18218705238871563123481 68, 0.33153107187430075342149 4, 0.23244586864587656466209 542,
0.47609207608457532190371 4, 0.25885996115008946816260 50, 0.43376757742228561065545 0, 0.28359733126185493468085 50,
0.39379374150325086034740 7, -0.17737416816454604218122 88, 0.37589723796688861989778 7, -0.07032162535225656985193 87,
0.49541342514169967037318 5, 0.30068906872205564129174 60, 0.50461094457043287331288 2, 0.28878050581433828411429 059,
0.43119238716682151135728 4, 0.09471825265864763920831 14, 0.34093743371217422026447 0, 0.13295076364564803461654 21,
0.42044641132325472066000 3, 0.04832237004364259985722 54, 0.37047048208991623097219 3, 0.02778166486689441841417 82,
0.44103716719912972044070 9, 0.21076263603869058171902 40, 0.41995910778104433846991 1, 0.28249007237102907737025 09,
0.43976300695642100979654 3, 0.27782222238779274992026 56, 0.46437287541727828577009 5, 0.31773789890072307029456 06,
0.27630924706023257675369 1, -0.12025390417594530867440 03, 0.29823982189785069873974 , -0.02961426514494985492298 78,
0.39541576467462108412913 3, -0.17637458795814225931408 71, 0.34742259407057968223497 8, -0.08480764580824710907602 79,
0.33935047674050483612617 8, -0.06952997088589547847181 65, 0.41495936902914293226499 89, -0.00483851202658322225941 1,
0.45972723301224210912252 5, 0.20593844887517756547801 18, 0.49962283307694427270107 0, 0.10559386561596652287509 10,
0.44038049052310251907371 7, 0.03516229318650736232552 20, 0.48281305440869642930934 4, 0.11133879176126169896487 99,
0.43212797390399521058057 5, 0.23621510441882287434367 53, 0.49017641685692181496847 3, 0.25951525061036334607322 83,
0.47877812128352564835559 5, 0.23427182746921057510114 39, 0.46469271175863147149698 3, 0.22294300925654586542634 66,
0.39196352858228972193543 9, 0.15036854763630938536245 20, 0.41538218853134400987414 7, 0.22608719238199015411083 79,
0.46010931804708665726622 0, 0.28200223264070023443395 03, 0.41985632279074058614671 8, 0.15600308889758310315626 80,
0.43941451887736626691349 8, 0.36999702036765569342643 86, 0.48051178273634209681120 0, 0.29493921239003723089131 39,
0.43706697623219242787584 4, 0.15928601669855479983084 8, 0.47642213343593436958300 9, 0.14295397377530394411593 31,
0.27127583771773200460586 4, -0.09959914409911729826774 26, 0.35301422943846061386352 6, -0.13051504202512567709426 61,
0.35754371269310353254056 2, 0.19430965790241760959844 4, 0.45726758013655793844165 5, -0.01769717301752758362165 53,
0.42755833682937582906902 2, -0.02319095519157991780456 74, 0.43252504998150381743116 1, 0.15165618417845286303169 42,
0.45122564865290276655882 4, 0.15017010677145178898506 12, 0.36133073860381265118846 3, 0.12641746704135950474245 93,
```

```
0.435216894190775471784605, 0.10099000918971350049258O1, 0.47058116289829693412727 4, 0.18512021943859616190941 61,
0.437460030660461018925389, 0.25327169324058285 13807933, 0.41704053164639759421206 1, 0.33029833954359444435902798,
0.384555064931715928633338, -0.15733368229414618211237 10, 0.37131809268857096650862 1, -0.10937081335819148764088 03,
0.421298482269021778989782, 0.14253235548561096757147 48, 0.49573647825277300515978 1, 0.04035310046516032284058 54,
0.302958384980159090016372, -0.22062511390964753554483307, 0.27976795081793170982820 1, -0.26678562639347036533362135,
0.248455666945173819781800, 0.13726827065170665864712 87, 0.28170300606109927032966 6, 0.00358240332890765638199 08,
0.475976318661099573326112, 0.39170645090291006384930 48, 0.40198522440580075620886 2, 0.07232921244315401732016 82,
0.503963566232464188487938, 0.28498224552572276602191 44, 0.46209915894654018941878 3, 0.18650735293258808887602278,
0.451960176846561534347302, 0.37308572215474400390888 83, 0.48435462744767471265916 5, 0.10016663145269570633910 66,
0.470928061192341984586562, 0.28081999260613693669697 00, 0.35937456625988628555745 7, 0.23253650318623380305049 53,
0.494898857108408918303866, 0.07833433567488800115247 857, 0.45981145489143820936917 0, -0.04373625002992437282858 03,
0.439796406579993748309221, -0.00886660446673748908752 44, 0.49109942445622889461986 9, 0.30221860627412550082127 04,
0.418721731642468675271829, 0.18575721044079615551503 08, 0.39027622172942139089357 7, 0.31549190318710890190700 98,
0.399469455790136929174139, 0.04982776481678723605795 945, 0.45475409685418449745597 5, 0.30384046278302212495958 66,
0.435410152674876649303570, 0.17373824597518938706031 12, 0.44586726278628252551072 8, 0.31646388907845141025276 80,
0.458967529691548414838564, 0.15187674036827811252514 66, 0.33591934361936293118589 0, 0.08239802945888033414156 84,
0.412763920830580666176957, 0.33190875187133400681105 40, 0.47104943124663084530467 9, 0.21563984592788421057463 21,
0.400699272940918671537247, 0.13951003017054569577126 70, 0.37486165796151343920783 5, 0.27940497265123181899134 99,
-0.138108934234092650683579, 0.19048843685283040061406 72, 0.28057072197749421960111 2, 0.19835378368840693931819 355,
0.420120277115559659364408, 0.26197787320665581627968 560, 0.45405706873558515646038 8, 0.30345262921996657423662 21,
0.395791684619125783139992, 0.00509879515625622989349 89, 0.36526042495962551814869 8, -0.02724687786240292156487 651,
0.484034990799743880529604, 0.35698870883228223105376 74, 0.37086341651042420419770 5, 0.10932989111354837752365 65,
0.461277752786707073706651, 0.26323575200894788750360 93, 0.35012522276381036201442 5, 0.01834602513214512661821 05,
0.368930134538406506727881, 0.22076777381085041507269 30, 0.53184922975271664569163 5, 0.21978694956693037299011 45,
0.416717917131823767373788, 0.19822932524099109996562 62, 0.41432188048214907416522 1, -0.08601048791835315399101 35,
0.438841302318247195035639, 0.05105524910551846595740 95, 0.39829853624973654557860 6, 0.08723980376104591115009931,
0.306217635368917995286608, -0.01499147781128863307631 43, 0.46937383373025498567443 5, 0.17090172846258661065735 55,
0.415929594466704966038151, 0.18005955456988811805745 976, 0.38555953496746486752755 3, 0.22801616378620923053510 69,
0.445047861541811862196028, 0.10618243452919626235164 69, 0.45043660946787689169923 4, 0.16592856167698008523932 093,
0.105991451516687498757285, 0.25473554546031174083964 29, 0.40156185005382499797477 2, -0.00547677331635797414637501,
0.068325138187451131543959, 0.18829702534569908722872 36, 0.42808418165536837343543, 0.12588258290687334706017 047,
0.386663185037617151973421, 0.27880342673950786469561 08, 0.36494145571685115969629 0, 0.19387774805755139673202 34,
0.388006006739891329271330, 0.24883894335990133783731 44, 0.31333155320333472282357 0, -0.27043484005715251461410 28,
0.352922805801725303354033, 0.02156897040061528034171 62, 0.44843016328422791172414 5, -0.09179326668113296783246 82,
0.431449464825348683039152, 0.35080008658391087505989 96, 0.43243816602220463973793 6, 0.12414672385863353976098 72,
0.428358234301227214224639, 0.00889653894957224545133 5, 0.45106114693577431529192 3, 0.19550639038107886502612 43,
0.438706053898136660862406, 0.40333860571352914137222 00, 0.37577492316702632013658 6, 0.17328987872808412906699 97,
0.519488312493666267392986 2, 0.20186273695046666463 8027, 0.44032539984282087486278 1, 0.13485681734262639713128 75,
0.490158077485978449505666, 0.35456916840173324789731 17, 0.50403236043246124609140 7, 0.23147747461205947505646 92,
0.465533674286672471964721, 0.33472132354099237705469 25, 0.35190230097532826381012 7, 0.32821325580308873437006 129,
0.487045153698114741125380, 0.09025283983197275594623 22, 0.42996156459732365329173 8, 0.04807132605508801503901 85,
0.437203684433824457666873, -0.01273093721303921767307 18, 0.46932180801343648113288 5, 0.26200193094513096747633 09,
0.405157392954838768694970, 0.19025705125501030101631 27, 0.46451260077393558001190 9, 0.22759065371615591621967 67,
0.396079698974603044980825, 0.17542754576987434900203 77, 0.44193538122988845762151 3, 0.22023915051711134838186 01,
0.464574093329662873852470, 0.25079263089996584046303 95, 0.48354702542898703176987 9, 0.26722552854377690234599 64,
0.373521032078406201026866, 0.03231747289280188090154 77, 0.35030499322060881794982 4, 0.05555926939278294646173 96,
0.495803893968995323948690, 0.20237019178223408788497 30, 0.43690870678674459393064 9, 0.13973395203670774256110 58,
0.420933303726012564283110, 0.12876400200242607563261 29, 0.33357747449337538103719 7, 0.19645710498981500125026 40,
-0.008696645322344730144604, 0.49369569070503982466391 34, 0.49888218734816985877600 4, -0.04298721107277587993866 77,
0.440681193952783212264279, 0.20222399176366009920791 10, 0.41973624884786764341134, 0.18005176913460321785890 9,
0.341754055915502841234144, 0.09757325940813059428347 03, 0.35190580345905309034293 4, 0.04672743022562853010271 06,
0.510698280023458961451865, 0.23689675970048257314104 49, 0.31936792954464959839285, 0.01447495895165376925128 88,
0.387776486196835801223415, 0.14680310041905566979339 94, 0.36312725190453637447163 3, -0.01830523643800072371541 353,
0.463043824984787411036535, 0.19983364266670247677559 76, 0.48544519572485678171602 8, 0.24103540729788136554034 85,
0.351766303400639024356877, 0.12364729515992035902005 60, 0.37886087431098774969129 3, -0.20686622817809068004163 412,
0.506548976335976064433453, 0.02577625313490182479281 29, 0.46234894736977910634223 6, 0.08410910862862572279219 36,
0.197060381522191130354571, 0.06105637811321459884617 01, 0.36915973328886914073265 8, 0.19781405523128747780958 75,
0.437006678117996483212693, 0.17827060821322210371064 90, 0.35386089273990928294466 6, 0.16194738173497943978240 70,
0.378029584800037510294857, 0.11400025954515274895225 03, 0.50140376954722720714841 0, 0.11539024280644789854921 58,
0.401394224316792058715464, 0.05889357293416394578677 85, 0.27559448959114662969227 5, 0.02330068817135083951068 255,
0.463520128314569346272833, 0.08758979159317575868826 11, 0.40472914650344787856894 9, 0.29171551547207602572697 74,
0.401167105521437583615096, 0.25253940592502210904513 05, 0.42906197876573831662128 1, 0.17949373032707409958419 5,
0.353765868289411034020020, 0.21620668714340252059002 98, 0.35675533593815844657370 9, -0.26639812266781431882023 41,
0.342817446803536374222432, 0.04064854416290769600905 72, 0.40582628844438227905080, -0.12893842431398158666552 311,
0.475344946440783899799243, 0.34776117255266325578588 75, 0.51268374391017290658823 0, 0.17401335710797163902618 93,
0.388901597797993381355752, -0.16873087366509412210469 07, 0.41640499676225523062456 1 0.042245369829584211718298 5 "
dat.AJS.fence.2 <- get.AJS.data(a); bagplot(dat.AJS.fence.2, main="dat.AJS.fence.2")
```

136 ⟨*define experiment to check fence error on a 64 bit machine* 136⟩ ≡

```
⟨define another chull function 132⟩
⟨define bagplot 32⟩
⟨check clockwise condition of polygon 131⟩
⟨Amanda's data set of 29Oct2012: missing fence – A 134⟩
⟨Amanda's data set of 29Oct2012: missing fence – B 135⟩
check.fence <- function(){
  par(mfrow=c(2,2))
  ⟨Amanda's data set of 29Oct2012: missing fence – A 134⟩
  ⟨Amanda's data set of 29Oct2012: missing fence – B 135⟩
  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 chull 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")
```

9.2.5 Further Na/NaN-errors

137 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error A* 137⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
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
```

138 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error B* 138⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
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
```

139 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error C* 139⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
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
```

140 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error D* 140⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
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
```

141 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error E* 141⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
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,
 -0.3683185, 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
```

142 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error F* 142⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
a<-"
-0.4062338461342518969310,  0.1475490833699421022728 9,-0.4641202518436023383153,  0.1230059865260322016222 7,
-0.4930885230817930175995,  0.1304159915893677024190 1,-0.4908548984150178373653,  0.1904203983535431299323 2,
-0.4413398310321014483826,  0.0641505127049062234823 0,-0.3790487599768575521786, -0.0190809252041998292370 7,
-0.4525924267924343324943,  0.0238201614942648402795 1,-0.4568684921740411852831,  0.1072913728356691270837 7,
-0.3600453131384418470340,  0.2029919573216544259608 4,-0.3786537514469233700609,  0.1078285534412276730487 1,
-0.3691294180636163213549,  0.1375725482989839376646 3,-0.3634668772645677226052,  0.1377949917231504894932 2,
-0.4460341706072293299634,  0.0892734482686577679455 6,-0.3931134284545839763936 9,  0.1237593920624090459980 9,
-0.4400074650713454715856,  0.1357283530048204178708 5,-0.3687761227741535030589,  0.0897798092772694455998 6,
-0.4477093868839493451262,  0.0773254388153468558142 5,-0.5047983341975119664369,  0.1200815284855332870650 5,
-0.4912126267418248093399,  0.0655948127853584056490 3,-0.3162564550157994092139   0.1271657043662295372143 9"
dat.AJS.NAN.F <- get.AJS.data(gsub("\\[[0-9]+,\\]","",a))
bagplot(dat.AJS.NAN.F, main="dat.AJS.NAN.F") # OK
```

143 ⟨*Amanda's data set of 29Oct2012: Na/NaN-error G* 143⟩ ≡ ⊂ 145
```
⟨define function get.AJS.data() 127⟩
```

```
a<-"-0.3365320800051156413524, -0.1646563427115356748053 9,-0.3212439795751604876273, -0.0276995774662534346988 2,
-0.3131092390672871039747, -0.0214501754083063350675 4,-0.2169780789017450306488, -0.0542017942859143800338 2,
-0.2382998530537111248062, -0.1566561362638586030193 7,-0.3274251750565500551637, -0.1626951129406397356103 0,
-0.1792897122532586817734,  0.1217254804458130224009 7,-0.3644874055796215595038, -0.1101711309933256638382 6,
-0.2597330667315013164043, -0.1366762691013512498461 3,-0.1857087775577647981162, -0.1218526183175312194162 4,
-0.4410627956771460689289, -0.2451745066292911334748 8,-0.4328369531394934965896, -0.0919226879501787857984 5,
-0.3162373978549153918927, -0.0928993995067755040651 0,-0.3527203678927318675207, -0.0457599978901878784443 0,
-0.3072802538232928259987, -0.1545227645023657114808 9,-0.1900264843388008995095, -0.1932992071358595546502 49,
-0.3581798974648294220380, -0.1809258569159212237664 7,-0.4087284209462124784373, -0.0707584628051894509726 0,
-0.3143274825737536470882, -0.1009354834788323412864 1,-0.3566729290036041999379, -0.0733731805347787729987 3,
-0.3809771622860922968279, -0.1393636447984781534525 9,-0.2029868237921297058346,  0.0893478145909082227538 2,
-0.2476551957000060755210, -0.0715814226613704918111 8,-0.3710390854098435942099, -0.0178701001932438756358 8,
-0.2911476980688867954861, -0.0423282734133516372332 4,-0.2253901616470012636562, -0.2535905191526199864426 9,
-0.1347541765844346151049, -0.0437201252764197051403 6,-0.2665596373370596738894, -0.0863926955398503781946 6,
  0.5589372043228539865822, -0.0544274642468143624030 0,-0.2892559615112646476121, -0.1521445588728595754712 8,
-0.3248426515933379166157, -0.0729425388250066752959 8,-0.4280514652785712970129, -0.0423710569843330428496 7,
-0.3992184363626437582084  -0.0351016748859382249348 1
"
dat.AJS.NAN.G <- get.AJS.data(gsub("\\[[0-9]+,\\]","",a))
bagplot(dat.AJS.NAN.G, main="dat.AJS.NAN.G") # OK
```

144 ⟨*show missing fence data* 144⟩ ≡ ⊂ 145

```
par(mfrow=c(3,3))
```

⟨*Amanda's data set of 29Oct2012: missing fence – A* 134⟩
⟨*Amanda's data set of 29Oct2012: missing fence – B* 135⟩

145 ⟨*Na-NaN-error-data-test* 145⟩ ≡

```
par(mfrow=c(3,3))
```

⟨*show missing fence data* 144⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error A* 137⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error B* 138⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error C* 139⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error D* 140⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error E* 141⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error F* 142⟩
⟨*Amanda's data set of 29Oct2012: Na/NaN-error G* 143⟩

9.2.6 Identical x-values

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

146 ⟨*checke cars data set* 146⟩ ≡

```
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)
```

10 frabo12 Daten

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

147 ⟨*frabo12* 147⟩ ≡

⟨*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, 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))
```

148 ⟨*Error in 1:pg.nol : NA/NaN Argument* 148⟩ ≡

```
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, 2, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2,
1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 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, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1,
1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2,
2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2,
1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 1, 2, 2), .Dim = c(149L,
2L), .Dimnames = list(NULL, c("X", "Y")))
bagplot(XY,verbose=TRUE)
```

149 ⟨* 18⟩+ ≡
```
⟨define bagplot 32⟩
par(mfrow=3:4)
set.seed(17)
for( i in 1:11 ){
  xy <- frabo12[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 <- frabo12[1:20,10:9]
#xy <- frabo12[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.

150 ⟨doit 150⟩ ≡
```
⟨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(frabo12)
xy <- frabo12[,c("Gewicht","Groesse")]; xy <- trim.xy(xy, 0.2)
bagplot(xy,precision=1,cex=1)
xy <- frabo12[,c("GroesseMutter","GroesseVater")]; xy <- trim.xy(xy, trim = 0.01)
bagplot(xy,precision=1,cex=1)
xy <- frabo12[,c("GroesseMutter","Groesse")]; xy <- trim.xy(xy,0.2)
bagplot(xy,precision=1,cex=1)
xy <- frabo12[,c("SchulNote","EurMaterial")]; xy <- trim.xy(xy,0.01)
bagplot(xy,precision=1,cex=1)
xy <- frabo12[,c("EurMiete","EurLuxus")]; xy <- trim.xy(xy,0.01)
bagplot(xy,precision=1,cex=1)
xy <- frabo12[,c("EurHandy","EurHandyKosten")]; xy <- trim.xy(xy,0.01)
bagplot(xy,precision=1,cex=1)
```

10.1 Indices

Object Index

Code Chunk Index

⟨*old: experiment for finding bag* 163⟩ ..... p**??**
⟨*old: find bag* 166⟩ ..... p**??**
⟨*old: find points of outer polygon to be shift* 167⟩ ..... p**??**
⟨*old: find points to shift on inner polygon* 168⟩ ..... p**??**
⟨*old: find value of* `lambda` 171⟩ ..... p**??**
⟨*old: find value of* `lambda` – *old version* 170⟩ ..... p**??**
⟨*old: shift points on polygons and construct* `hull.bag` 169⟩ ..... p**??**
⟨*old version: check points on a grid to find center* 162⟩ ..... p**??**
⟨*old version: define function* `out.of.polygon` 160⟩ ..... p**??**
⟨*old version of the combination of lower and upper polygon* 159⟩ ..... p**??**
⟨*old version to find* `lambda` 164⟩ ..... p**??**
⟨*onedim* 8⟩ p8
⟨*one dim test* 9⟩ p8
⟨*output result* 35⟩ ⊂ 34 ..... p26
⟨*quadratic* 6⟩ p7
⟨*quadratic-test* 120⟩ p70
⟨*rnorm* 3⟩ p4
⟨*rnorm, different sizes* 5⟩ p6
⟨*search for points with critical hdepth* 48⟩ ⊂ 46 ..... p33
⟨*select colors* 95⟩ ⊂ 92 ..... p55
⟨*show missing fence data* 144⟩ ⊂ 145 ..... p79
⟨*show pg pgl* 70⟩ p43
⟨*some functions for generating bagplots* 86⟩ p50
⟨*standardize data and compute:* `xyxy`, `xym`, `xysd` 54⟩ ⊂ 34 ..... p36
⟨*standardize dimensions* 58⟩ ⊂ 57 ..... p37
⟨*start* 85 ∪ 89 ∪ 94 ∪ 113 ∪ 117⟩ p50
⟨*test 121126* 156 ∪ 158⟩ ..... p**??**
⟨*test: data set of Ben Greiner* 14⟩ p11
⟨*test of amanda-data-sets* 124 ∪ 125⟩ p71
⟨*Test of R data sets* 105⟩ p57
⟨*test of the examples of plotsummary* 108⟩ p60
⟨*transform data into numerical vectors* 98⟩ ⊂ 92 ..... p56
⟨*verbosetest* 16⟩ p13
⟨*version of bagplot* 1⟩ ⊂ 32, 33, 81 ..... p2