Chernoff Faces and Spline Interpolation

H. P. Wolf

file: faces.rev, 08/Jan/2009, 03/Nov/2011, printed: October 30, 2013

Contents

1 Definition of `faces`

Chernoff Faces are funny repesentations of multi dimensional data sets. For there is no `faces` function in R we present a simple version in this paper. The construction of the faces is quiet clear and the reader is invited to modify `faces` as she / he likes it.

The smooth curves for drawing the elements of a face (eyes, ears, etc.) are computed by fitting spline functions. At first we define a function to be able to plot smooth curves. Then the data matrix is checked and the standardized according to the input parameters `which.row`, `fill`, `scale` and `byrow`. In the initialisation part of the function characteristic points of a standard face and some graphics parameter are fixed. Within a loop along the rows of the data matrix faces are constructed in three steps:

1. the coordinates of the standard face are transformed,
2. the transformed points are arranged to organised to sets which represent elements of a face
3. smooth curves fitted to the sets of points are plotted on the graphics device

Finally a title is placed above the faces.

rbind(1:3,5:3,3:5,5:7)

1 ⟨*define* `faces` 1⟩ ≡ ⊂ 25, 26, 27, 28, 31, 32, 34

```
faces<-function(xy,which.row,fill=FALSE,face.type=1,
                nrow.plot,ncol.plot,scale=TRUE,byrow=FALSE,main,
                labels,print.info = TRUE,na.rm = FALSE,
                ⟨define some colors 19⟩
                plot.faces=TRUE){  # 070831 pwolf
  if((demo<-missing(xy))){
    xy<-rbind(
              c(1,3,5),c(3,5,7),
              c(1,5,3),c(3,7,5),
              c(3,1,5),c(5,3,7),
              c(3,5,1),c(5,7,3),
              c(5,1,3),c(7,3,5),
              c(5,3,1),c(7,5,3),
              c(1,1,1),c(4,4,4),c(7,7,7)
    )
    labels<-apply(xy,1,function(x) paste(x,collapse="-"))
  }
  ⟨define spline 35⟩
  ⟨standardize input 5⟩
  ⟨define characteristic points of standard face 6⟩
  ⟨define graphics parameter 3⟩
  ⟨loop over faces 7⟩
  ⟨finish plot 4⟩
  ⟨info about the effects of the variables 9⟩
  ⟨output face.list 2⟩
}
```

2 ⟨*output face.list* 2⟩ ≡ ⊂ 1

```
names(face.list)<-xnames
out<-list(faces=face.list,info=info,xy=t(xy))
class(out)<-"faces"
invisible(out)
```

Let's start with some simple tasks. The graphics device needs to be prepared. If parameters `nrow.plot` and `ncol.plot` are found they will be used for the splitting of the graphics device. Otherwise a chess board design is used.

3 ⟨*define graphics parameter* 3⟩ ≡ ⊂ 1

```
nr<-n^0.5; nc<-n^0.5
if(!missing(nrow.plot)) nr<-nrow.plot
if(!missing(ncol.plot)) nc<-ncol.plot
if(plot.faces){
  opar<-par(mfrow=c(ceiling(c(nr,nc))),oma=rep(6,4), mar=rep(.7,4))
  on.exit(par(opar))
}
```

If a title is given it has to be placed at the top of the page.

4 ⟨*finish plot* 4⟩ ≡ ⊂ 1

```
if(plot.faces&&!missing(main)){
  par(opar);par(mfrow=c(1,1))
  mtext(main, 3, 3, TRUE, 0.5)
  title(main)
}
```

The data have to be structured as matrix. `byrow=TRUE` results in a transposition of the input data `xy`. `which.row` allows to permute the order of the rows. In effect the representation of data attribute will be changed. `labels` can be used to name the faces. If `fill=TRUE` and `n.c` equals the number of columns of the data matrix the first `n.c` items of the standard face will be modified whereas all the rest will be unchanged. If `scale=FALSE` the variables of the input are not standardized to the intervall [-1,1]. However, the data are rounded to the interval.

5 ⟨*standardize input* 5⟩ ≡ ⊂ 1

```
n.char<-15
xy<-rbind(xy)
if(byrow) xy<-t(xy)
if(any(is.na(xy))){
  if(na.rm){
    xy<-xy[!apply(is.na(xy),1,any),,drop=FALSE]
    if(nrow(xy)<3) {print("not enough data points"); return()}
    print("Warning: NA elements have been removed!!")
   }else{
    xy.means<-colMeans(xy,na.rm=TRUE)
    for(j in 1:length(xy[1,])) xy[is.na(xy[,j]),j]<-xy.means[j]
    print("Warning: NA elements have been exchanged by mean values!!")
  }
}
if(!missing(which.row)&& all(  !is.na(match(which.row,1:dim(xy)[2]))  ))
       xy<-xy[,which.row,drop=FALSE]
mm<-dim(xy)[2];  n<-dim(xy)[1]
xnames<-dimnames(xy)[[1]]
if(is.null(xnames)) xnames<-as.character(1:n)
if(!missing(labels)) xnames<-labels
if(scale){
   xy<-apply(xy,2,function(x){
           x<-x-min(x); x<-if(max(x)>0) 2*x/max(x)-1 else x })
} else xy[]<-pmin(pmax(-1,xy),1)
xy<-rbind(xy);n.c<-dim(xy)[2]
# expand input matrix xy by replication of cols
xy<-xy[,(rows.orig<-h<-rep(1:mm,ceiling(n.char/mm))),drop=FALSE]
if(fill) xy[,-(1:n.c)]<-0
```

We have to define some characteristic points of a standard face. For we want to produce symmetrical faces we save points of the right half only. A face is considered as a set of objects: lips, eyes, nose, ears, hair, contour / shape. Each object is defined by a 2 column matrix which rows represent coordinate points. To be able to draw an element of a face very easy the points have been sorted in the correct order. The six objects are assembed into the list `face.orig`. If we want to plot the complete face we have to add the points of the left half of the face. For the vertical centerline of a face is $x = 0$ the coordinates of points of the left part are given by changing the sign of the associeted right points. However we have to keep in mind the points to be reflected and their order. For this we use some vectors of indices `*refl.ind`. The vectors `*.notnull` tell us which points are not on the centertral line.

6 ⟨*define characteristic points of standard face* 6⟩ ≡ ⊂ 1

```
face.orig<-list(
      eye  =rbind(c(12,0),c(19,8),c(30,8),c(37,0),c(30,-8),c(19,-8),c(12,0))
     ,iris =rbind(c(20,0),c(24,4),c(29,0),c(24,-5),c(20,0))
     ,lipso=rbind(c(0,-47),c( 7,-49),lipsiend=c( 16,-53),c( 7,-60),c(0,-62))
```

```
      ,lipsi=rbind(c(7,-54),c(0,-54))                  # add lipsiend
      ,nose =rbind(c(0,-6),c(3,-16),c(6,-30),c(0,-31))
      ,shape =rbind(c(0,44),c(29,40),c(51,22),hairend=c(54,11),earsta=c(52,-4),
                   earend=c(46,-36),c(38,-61),c(25,-83),c(0,-89))
      ,ear  =rbind(c(60,-11),c(57,-30))               # add earsta,earend
      ,hair =rbind(hair1=c(72,12),hair2=c(64,50),c(36,74),c(0,79)) # add hairend
)
lipso.refl.ind<-4:1
lipsi.refl.ind<-1
nose.refl.ind<-3:1
hair.refl.ind<-3:1
shape.refl.ind<-8:1
shape.xnotnull<-2:8
nose.xnotnull<-2:3
```

A specific face is created by three steps:

step 1 : modify the characteristic points of the standard face

step 2 : define polygons of the objects of the modified points

step 3 : plot spline approximations of the polynoms

One important question is how the data effects on the variation of the standard face. S-Plus offers the following fatures: 1-area of face, 2-shape of face, 3-length of nose, 4-location of mouth, 5-curve of smile, 6-width of mouth, 7 .. 11 location, separation, angle, shape, width of eyes, 12-location of pupil, 13 .. 15 location angle and width of eyebrow.

Our features are: 1-height of face, 2-width of face, 3-shape of face, 4-height of mouth, 5-width of mouth, 6-curve of smile, 7-height of eyes, 8-width of eyes, 9-height of hair, 10-width of hair, 11-styling of hair, 12-height of nose, 13-width of nose, 14-width of ears, 15-height of ears. The modification are performed one after the other. Then a face can be constructed and plotted.

7 ⟨*loop over faces* 7⟩ ≡ ⊂ 1

```
face.list<-list()
for(ind in 1:n){
  ⟨initialize new face 8⟩
  ⟨modify lips 11⟩
  ⟨modify eye 12⟩
  ⟨modify hair 13⟩
  ⟨modify nose 14⟩
  ⟨modify ear 15⟩
  ⟨modify shape 10⟩
  ⟨construct specific face 16⟩
  ⟨plot specific face 17⟩
}
```

As an initialisation the standard face is copied to `face` and the values of the variables are stored in `factors`.

8 ⟨*initialize new face* 8⟩ ≡ ⊂ 7

```
factors<-xy[ind,]
face<-face.orig
```

Now we have to transform the face according to the data.

9 ⟨*info about the effects of the variables* 9⟩ ≡ ⊂ 1

```
info<-c(
```

```
  "var1"="height of face   ",
  "var2"="width of face    ",
  "var3"="structure of face",
  "var4"="height of mouth  ",
  "var5"="width of mouth   ",
  "var6"="smiling          ",
  "var7"="height of eyes   ",
  "var8"="width of eyes    ",
  "var9"="height of hair   ",
  "var10"="width of hair   ",
  "var11"="style of hair   ",
  "var12"="height of nose  ",
  "var13"="width of nose   ",
  "var14"="width of ear    ",
  "var15"="height of ear   ")
var.names<-dimnames(xy)[[2]]
if(0==length(var.names)) var.names<-paste("Var",rows.orig,sep="")
info<-cbind("modified item"=info,"Var"=var.names[1:length(info)])
rownames(info)<-rep("",15)
if(print.info){
  cat("effect of variables:\n")
  print(info)
}
if(demo&&plot.faces) {
  plot(1:15,1:15,type="n",axes=FALSE,bty="n")
  text(rep(1,15),15:1,adj=0,apply(info,1,function(x)
      paste(x,collapse="   -   ")),cex=0.7)
}
```

Height, width and structure of the faces is changed by `factors[1:3]`. In the actual version `factor[1:3]` have an overall scaling effect. The comment lines show how the effect can be reduced to the contour line of the face.

10 ⟨*modify shape* 10⟩ ≡ ⊂ 7

```
face<-lapply(face,function(x){ x[,2]<-x[,2]*(1+0.2*factors[1]);x})
face<-lapply(face,function(x){ x[,1]<-x[,1]*(1+0.2*factors[2]);x})
face<-lapply(face,function(x){ x[,1]<-ifelse(x[,1]>0,
                                        ifelse(x[,2] > -30, x[,1],
                  pmax(0,x[,1]+(x[,2]+50)*0.2*sin(1.5*(-factors[3])))),0);x})
#face$shape[,2]<-face$shape[,2]*(1+0.2*factors[1])
#face$shape[,1]<-face$shape[,1]*(1+0.2*factors[2])
#face$shape[,1]<-face$shape[,1]<-ifelse(face$shape[,1]>0,
#   ifelse(face$shape[,2] > -30, face$shape[,1],
#      pmax(0,face$shape[,1]+(face$shape[,2]+50)*0.2*sin(1.5*(-factors[3])))),0)
```

Factor 4 and 5 have a scaling effect on the mouth. Factor 6 changes the smiling.

11 ⟨*modify lips* 11⟩ ≡ ⊂ 7

```
m<-mean(face$lipso[,2])
face$lipso[,2]<-m+(face$lipso[,2]-m)*(1+0.7*factors[4])
face$lipsi[,2]<-m+(face$lipsi[,2]-m)*(1+0.7*factors[4])
face$lipso[,1]<-face$lipso[,1]*(1+0.7*factors[5])
face$lipsi[,1]<-face$lipsi[,1]*(1+0.7*factors[5])
face$lipso["lipsiend",2]<-face$lipso["lipsiend",2]+20*factors[6]
```

Factor 7 and 8 define scaling effects on the eyes.

12 $\langle$*modify eye* 12$\rangle \equiv$ $\subset$ 7

```
m<-mean(face$eye[,2])
face$eye[,2] <-m+(face$eye[,2] -m)*(1+0.7*factors[7])
face$iris[,2]<-m+(face$iris[,2]-m)*(1+0.7*factors[7])
m<-mean(face$eye[,1])
face$eye[,1] <-m+(face$eye[,1] -m)*(1+0.7*factors[8])
face$iris[,1]<-m+(face$iris[,1]-m)*(1+0.7*factors[8])
```

The hair is changed by factor 9, 10 and 11.

13 $\langle$*modify hair* 13$\rangle \equiv$ $\subset$ 7

```
m<-min(face$hair[,2])
face$hair[,2]<-m+(face$hair[,2]-m)*(1+0.2*factors[9])
m<-0
face$hair[,1]<-m+(face$hair[,1]-m)*(1+0.2*factors[10])
m<-0
face$hair[c("hair1","hair2"),2]<-face$hair[c("hair1","hair2"),2]+50*factors[11]
```

The nose scaling factors are 12 and 13 and ...

14 $\langle$*modify nose* 14$\rangle \equiv$ $\subset$ 7

```
m<-mean(face$nose[,2])
face$nose[,2]<-m+(face$nose[,2]-m)*(1+0.7*factors[12])
face$nose[nose.xnotnull,1]<-face$nose[nose.xnotnull,1]*(1+factors[13])
```

... for the ears factors 14 and 15 matters.

15 $\langle$*modify ear* 15$\rangle \equiv$ $\subset$ 7

```
m<-mean(face$shape[c("earsta","earend"),1])
face$ear[,1]<-m+(face$ear[,1]-m)* (1+0.7*factors[14])
m<-min(face$ear[,2])
face$ear[,2]<-m+(face$ear[,2]-m)* (1+0.7*factors[15])
```

After transforming the standard face elements of the specific face are completed and collected in a list (`face.obj`).

16 $\langle$*construct specific face* 16$\rangle \equiv$ $\subset$ 7

```
invert<-function(x) cbind(-x[,1],x[,2])
face.obj<-list(
      eyer=face$eye
     ,eyel=invert(face$eye)
     ,irisr=face$iris
     ,irisl=invert(face$iris)
     ,lipso=rbind(face$lipso,invert(face$lipso[lipso.refl.ind,]))
     ,lipsi=rbind(face$lipso["lipsiend",],face$lipsi,
                  invert(face$lipsi[lipsi.refl.ind,,drop=FALSE]),
                  invert(face$lipso["lipsiend",,drop=FALSE]))
     ,earr=rbind(face$shape["earsta",],face$ear,face$shape["earend",])
     ,earl=invert(rbind(face$shape["earsta",],face$ear,face$shape["earend",]))
     ,nose=rbind(face$nose,invert(face$nose[nose.refl.ind,]))
     ,hair=rbind(face$shape["hairend",],face$hair,invert(face$hair[hair.refl.ind,]),
                 invert(face$shape["hairend",,drop=FALSE]))
     ,shape=rbind(face$shape,invert(face$shape[shape.refl.ind,]))
)
face.list<-c(face.list,list(face.obj))
```

Now we are ready to compose the specific faces by drawing smooth curves fitted to the polygons. The following code chunk uses a chunk that is also referenced in the function `plot.face`. This function allows the user to place faces anywhere in a plot. Therefore, transformations are needed and are done by the functions `xtrans` and `ytrans`. Here these two functions do not change the input values. For plainting we need to map the data values (`factors`) on colors. `f` is the corresponding index of the palettes.

17 ⟨*plot specific face* 17⟩ ≡ ⊂ 7

```
if(plot.faces){
  plot(1,type="n",xlim=c(-105,105)*1.1, axes=FALSE,
       ylab="",ylim=c(-105,105)*1.3)
  title(xnames[ind])
  f<-1+(ncolors-1)*(factors+1)/2 # translate factors into color numbers
  xtrans<-function(x){x};  ytrans<-function(y){y}
  for(obj.ind in seq(face.obj)[c(10:11,1:9)]) {
    x <-face.obj[[obj.ind]][,1]; y<-face.obj[[obj.ind]][,2]
    xx<-spline(1:length(x),x,40,FALSE)[,2]
    yy<-spline(1:length(y),y,40,FALSE)[,2]
    if(plot.faces){
      lines(xx,yy)
      if(face.type>0){
        ⟨paint elements of a face 18⟩
      }
    }
  }
}
```

For painting parts of the faces the order of painting has to been recognized. The colors are found by averaging over the color indices of the relevant variables.

18 ⟨*paint elements of a face* 18⟩ ≡ ⊂ 17, 20

```
if(obj.ind==10)
  polygon(xtrans(xx),ytrans(yy),col=col.hair[ceiling(mean(f[9:11]))],xpd=NA) # hair
if(obj.ind==11){
  polygon(xtrans(xx),ytrans(yy),col=col.face[ceiling(mean(f[1:2 ]))],xpd=NA) # face
  ⟨paint christmas like if 2==face.type 21⟩
}
xx<-xtrans(xx); yy<-ytrans(yy)
if(obj.ind %in% 1:2) polygon(xx,yy,col="#eeeeee") # eyes without iris
if(obj.ind %in% 3:4) polygon(xx,yy,col=col.eyes[ceiling(mean(f[7:8 ]))],xpd=NA) # eyes:iris
if(obj.ind %in% 9)   polygon(xx,yy,col=col.nose[ceiling(mean(f[12:13]))],xpd=NA)# nose
if(obj.ind %in% 5:6) polygon(xx,yy,col=col.lips[ceiling(mean(f[1:3]))],xpd=NA)  # lips
if(obj.ind %in% 7:8) polygon(xx,yy,col=col.ears[ceiling(mean(f[14:15]))],xpd=NA)# ears
```

For painted faces it is nice to have suitable colors. Different elements of a face are painted with colors of different color palettes.

19 ⟨*define some colors* 19⟩ ≡ ⊂ 1, 20

```
ncolors=20,
col.nose=rainbow(ncolors),                   # nose
col.eyes=rainbow(ncolors,start=0.6,end=0.85),# eyes
col.hair=terrain.colors(ncolors),            # hair
col.face=heat.colors(ncolors),               # face
col.lips=rainbow(ncolors,start=0.0,end=0.2), # lips
col.ears=rainbow(ncolors,start=0.0,end=0.2), # ears
```

That's it for usual applications. Sometimes it is nice to draw a face at a certain position of an existing plot. For this the function `plot.faces` will do the job. This function takes a face object and reconstructs faces either at positions fixed by the user or the faces we be drawn on a chess board.

20 ⟨*define* `plot.faces` 20⟩ ≡ ⊂ 25, 26, 34

```
plot.faces<-function(x,x.pos,y.pos,face.type = 1,
                     width=1,height=1,labels,
                     ⟨define some colors 19⟩
                     ...){
  if(missing(x)) return("no face.list object in call")
  face.list<-x$faces; face.data<-x$xy
  if(class(face.list)!="faces") {
      if(!is.list(face.list) || !any(names(face.list[[1]])=="lipso") )
        return("input not of class faces")
  }
  ⟨define spline 35⟩
  n<-length(face.list)
  if(missing(x.pos)){
     co<-ro<-ceiling(n^0.5)
     plot((0:ro)+.5,(0:co)+.5,type="n",xlab="",ylab="",axes=FALSE)
     m<-matrix(1,ro,co); x.pos<-col(m); y.pos<-(1+ro)-row(m)
  }
  if(!missing(labels)) names(face.list)<-labels
  fac.x<-width/1.1/210; fac.y<-height/1.3/210
  xtrans<-function(x){x.pos[j]+fac.x*x};  ytrans<-function(y){y.pos[j]+fac.y*y}
  for(j in seq(face.list)){
    face.obj<-face.list[[j]]; factors<-face.data[,j]
    f<-1+(ncolors-1)*(factors+1)/2 # translate factors into color numbers
    for(obj.ind in seq(face.obj)[c(10:11,1:9)]) {
       x <-face.obj[[obj.ind]][,1]; y<-face.obj[[obj.ind]][,2]
       xx<-spline(1:length(x),x,40,FALSE)[,2]
       yy<-spline(1:length(y),y,40,FALSE)[,2]
       lines(xtrans(xx),ytrans(yy),...)
       if(face.type>0){ ⟨paint elements of a face 18⟩ }
    }
    lab<-names(face.list)[j]
    text(x.pos[j],y.pos[j]-0.5*height,lab,xpd=NA)
  }
}
```

The expressions for computing the christmas version are extracted to increase the readability.

21 ⟨*paint christmas like if* `2==face.type` 21⟩ ≡ ⊂ 18

```
if(face.type==2){
    # beard
    for(zzz in seq(hhh<-max(face.obj[[8]][,1]),-hhh,length=30)){
      hrx<-rnorm(8,zzz,2); hry<-0:7*-3*rnorm(1,3)+abs(hrx)^2/150
      hry<-min(face.obj[[9]][,2])+hry
      lines(xtrans(hrx),ytrans(hry),lwd=5,col="#eeeeee",xpd=NA)
    }
    ind<-which.max(xx); wx<-xx[ind]; ind<-which.max(yy); wy<-yy[ind]
    # edge of hat
    wxh<-wx<-seq(-wx,wx,length=20); wyh<-wy<-wy-(wx-mean(wx))^2/250+runif(20)*3
    lines(xtrans(wxh),ytrans(wyh)); wx<-c(wx,rev(wx)); wy<-c(wy-10,rev(wy)+20)
    wmxy1<-wmxy0<-c(min(wx),min(wy)+20)
```

```
    wmxy2<-wmxy3<-c(runif(1,wmxy0[1],-wmxy0[1]), wy[1]+100)
    wmxy1[2]<-0.5*(wmxy0[2]+wmxy3[2]); wmxy2[1]<-0.5*(wmxy2[1]+wmxy0[1])
    npxy<-20; pxy<-seq(0,1,length=npxy)
    gew<-outer(pxy,0:3,"^")*outer(1-pxy,3:0,"^")*
         matrix(c(1,3,3,1),npxy,4,byrow=TRUE)
    wxl<-wmxy0[1]*gew[,1]+wmxy1[1]*gew[,2]+wmxy2[1]*gew[,3]+wmxy3[1]*gew[,4]
    wyl<-wmxy0[2]*gew[,1]+wmxy1[2]*gew[,2]+wmxy2[2]*gew[,3]+wmxy3[2]*gew[,4]
    lines(xtrans(wxl),ytrans(wyl),col="green")
    wmxy1[1]<- wmxy0[1]<- -wmxy0[1]
    wmxy1[2]<-0.5*(wmxy0[2]+wmxy3[2]); wmxy2[1]<-0.5*(wmxy2[1]+wmxy0[1])
    wxr<-wmxy0[1]*gew[,1]+wmxy1[1]*gew[,2]+wmxy2[1]*gew[,3]+wmxy3[1]*gew[,4]
    wyr<-wmxy0[2]*gew[,1]+wmxy1[2]*gew[,2]+wmxy2[2]*gew[,3]+wmxy3[2]*gew[,4]
    points(xtrans(wmxy3[1]),ytrans(wmxy3[2]),pch=19,cex=2,col="#ffffff",xpd=NA)
    points(xtrans(wmxy3[1]),ytrans(wmxy3[2]),pch=11,cex=2.53,col="red",xpd=NA)
    polygon(xtrans(c(wxl,rev(wxr))),ytrans(c(wyl,rev(wyr))),col="red",xpd=NA) # hat
    polygon(xtrans(wx),ytrans(wy),col="#ffffff",xpd=NA) # edge of hat
}
```

2 Code extraction

22 ⟨*extract code* 22⟩ ≡

```
tangleR("faces",expand.roots="define [[faces]] and [[faces.plot]]")
```

23 ⟨*** 23⟩ ≡

```
tangleR("faces",expand.roots="christmas")
```

24 ⟨*start* 24⟩ ≡

```
"relax"
```

3 Some tests

Some tests may be useful. Here is a chunk that will trigger the definition of the functions.

25 ⟨*define* `faces` *and* `faces.plot` 25⟩ ≡

```
⟨define faces 1⟩
⟨define plot.faces 20⟩
faces(face.type=2); ""
args(plot.faces)
```

Here are the arguments of `plot.faces` –

```
Wed Jan  7 15:08:19 2009
function (x, x.pos, y.pos, face.type = 1, width = 1, height = 1, labels,
        ncolors = 20, col.nose = rainbow(ncolors), col.eyes = rainbow(ncolors,
        start = 0.6, end = 0.85), col.hair = terrain.colors(ncolors),
        col.face = heat.colors(ncolors), col.lips = rainbow(ncolors,
        start = 0, end = 0.2), col.ears = rainbow(ncolors, start = 0,
        end = 0.2), ...)
```

This code chunk shows different ways of constructing faces.

26 ⟨*test1* 26⟩ ≡

```
⟨define faces 1⟩
⟨define plot.faces 20⟩
a<-faces(rbind(1:3,5:3,3:5,5:7),plot=!FALSE,face.type=2)
plot.faces(a,face.type=0)
plot(1:4,type="n");plot.faces(a,x.pos=1:4,y.pos=1:4,width=0.5,height=1,face.type=1)
""
```

27 ⟨*test2* 27⟩ ≡

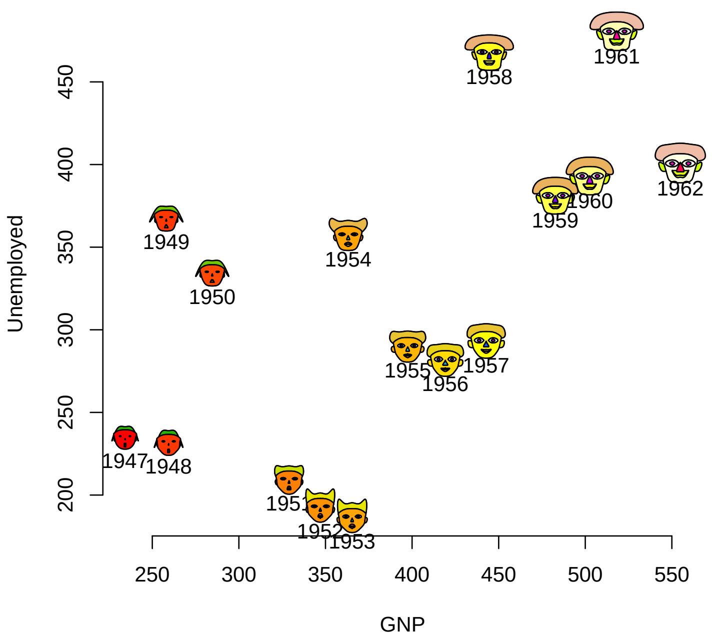
```
⟨define faces 1⟩
faces(rbind(rep(1,3),rep(5,3),c(1,1,5),c(1,5,1),c(1,5,5),c(5,5,1),
             c(5,1,5),c(3,3,3),c(1,5,1)))
""
```

28 ⟨*test3* 28⟩ ≡

```
⟨define faces 1⟩
data(longley)
faces(longley[1:9,])
```

29 ⟨* 23⟩+ ≡

```
a<-faces(longley[1:16,],plot=FALSE)
plot(longley[1:16,2:3],bty="n")
plot.faces(a,longley[1:16,2],longley[1:16,3],width=35,height=30)
```



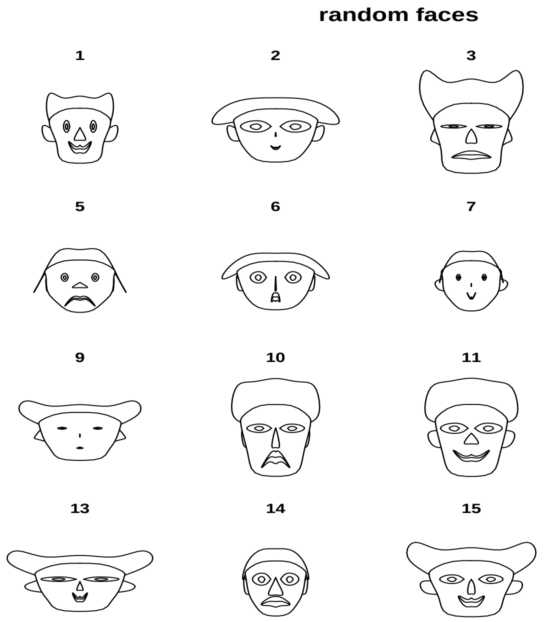
30 ⟨* 23⟩+ ≡

```
dim(longley)
```

31 ⟨*test4* 31⟩ ≡

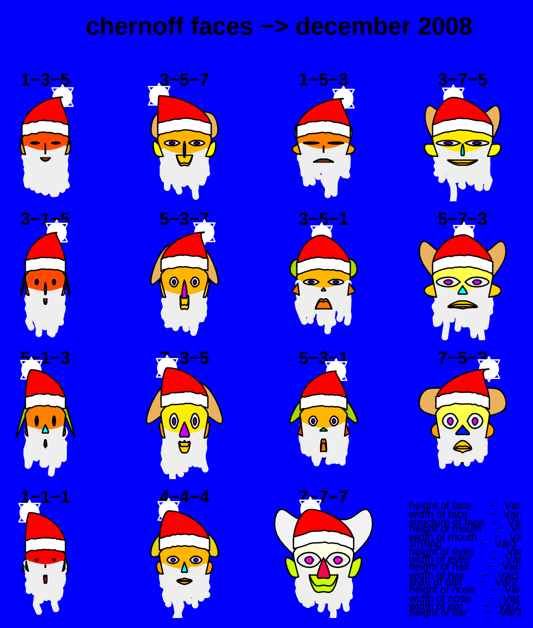
```
⟨define faces 1⟩
set.seed(17)
faces(matrix(sample(1:1000,128,),16,8),main="random faces")
```

Here is the result:



32 ⟨*christmas* 32⟩ ≡

```
christmas<-function(){
delay<-.1
⟨define faces 1⟩
par(bg="white")
faces(face.type=0)
title("the old  chernoff faces")
Sys.sleep(delay)
faces()
title("new chernoff faces")
Sys.sleep(delay)
par(bg="blue")
faces(face.type=2)
title("chernoff faces -> december 2008")
}
christmas()
```



4 Rd-file

In the chapter "Documenting functions" of the manual "Writing R Extensions" we find that there is a special style for the usage section concerning S3 method functions. Ignoring this style results in a note and later perhaps in an error situation. Therefore, we now use the method style

33 ⟨*define* `faces` *help* 33⟩ ≡

```
\name{faces}
\alias{faces}
\alias{plot.faces}
\title{    Chernoff Faces    }
\description{
     \code{faces} represent the rows of a data matrix by faces.
     \code{plot.faces} plots faces into a scatterplot.
}
\usage{
faces(xy, which.row, fill = FALSE, face.type = 1, nrow.plot, ncol.plot,
    scale = TRUE, byrow = FALSE, main, labels, print.info = TRUE,
    na.rm = FALSE, ncolors = 20, col.nose = rainbow(ncolors),
    col.eyes = rainbow(ncolors, start = 0.6, end = 0.85),
    col.hair = terrain.colors(ncolors), col.face = heat.colors(ncolors),
    col.lips = rainbow(ncolors, start = 0, end = 0.2),
    col.ears = rainbow(ncolors, start = 0, end = 0.2), plot.faces = TRUE)
\method{plot}{faces}(x, x.pos, y.pos, face.type = 1, width = 1, height = 1, labels,
        ncolors = 20, col.nose = rainbow(ncolors), col.eyes = rainbow(ncolors,
        start = 0.6, end = 0.85), col.hair = terrain.colors(ncolors),
        col.face = heat.colors(ncolors), col.lips = rainbow(ncolors,
        start = 0, end = 0.2), col.ears = rainbow(ncolors, start = 0,
```

```
        end = 0.2), \ldots)
}
\arguments{
  \item{xy}{   \code{xy} data matrix, rows represent individuals and columns variables  }
  \item{which.row}{   defines a permutation of the rows of the input matrix   }
  \item{fill}{   \code{if(fill==TRUE)}, only the first \code{nc} attributes of the faces are
                  transformed, \code{nc} is the number of columns of \code{xy}   }
  \item{face.type}{  an integer between 0 and 2 with the meanings:
                     0 = line drawing faces,
                     1 = the elements of the faces are painted,
                     2 = Santa Claus faces are drawn }
  \item{nrow.plot}{   number of columns of faces on graphics device   }
  \item{ncol.plot}{   number of rows of faces   }
  \item{scale}{   \code{if(scale==TRUE)}, variables will be normalized   }
  \item{byrow}{   \code{if(byrow==TRUE)}, \code{xy} will be transposed   }
  \item{main}{   title   }
  \item{labels}{   character strings to use as names for the faces   }
  \item{print.info}{   if TRUE information about usage of variables for face elements are pr
  \item{na.rm}{ if TRUE 'NA' values are removed otherwise exchanged by mean of data}
  \item{plot.faces}{   if \code{FALSE} no face is plotted  }
  \item{x}{ an object of class \code{faces} computed by \code{faces}   }
  \item{x.pos}{ x coordinates of positions of faces  }
  \item{y.pos}{ y coordinates of positions of faces  }
  \item{width}{ width of the faces }
  \item{height}{ height of the faces }
  \item{ncolors}{ number of colors in the palettes for painting the elements of the faces }
  \item{col.nose}{ palette of colors for painting the nose }
  \item{col.eyes}{ palette of colors for painting the eyes }
  \item{col.hair}{ palette of colors for painting the hair }
  \item{col.face}{ palette of colors for painting the face }
  \item{col.lips}{ palette of colors for painting the lips }
  \item{col.ears}{ palette of colors for painting the ears }
  \item{...}{ additional graphical arguments }
}
\details{
Explanation of parameters:
1-height of face,
2-width of face,
3-shape of face,
4-height of mouth,
5-width of mouth,
6-curve of smile,
7-height of eyes,
8-width of eyes,
9-height of hair,
10-width of hair,
11-styling of hair,
12-height of nose,
13-width of nose,
14-width of ears,
15-height of ears.

For painting elements of a face the colors of are found by
averaging of sets of variables:
```

```
(7,8)-eyes:iris, (1,2,3)-lips,
(14,15)-ears, (12,13)-nose, (9,10,11)-hair, (1,2)-face.

Further details can be found in the literate program of \code{faces}.
}
\value{
  list of two elements: The first element \code{out$faces}
  is a list of standardized faces of \code{class faces},
  this object could be plotted by plot.faces;
  a plot of faces is created on the graphics device if
  \code{plot.faces=TRUE}.
  The second list is short description of the effects of the variables.
}
\references{  Chernoff, H. (1973): The use of faces to represent statistiscal assoziation,
JASA, 68, pp 361--368.
The smooth curves are computed by an algorithm found in
Ralston, A. and Rabinowitz, P. (1985):
A first course in numerical analysis, McGraw-Hill, pp 76ff.
\url{http://www.wiwi.uni-bielefeld.de/mitarbeiter/wolf/} :
S/R - functions : faces
   }
\author{   H. P. Wolf    }
\note{ version 01/2009   }

\seealso{   ---    }
\examples{

faces()
faces(face.type=1)

faces(rbind(1:3,5:3,3:5,5:7))

data(longley)
faces(longley[1:9,],face.type=0)
faces(longley[1:9,],face.type=1)

plot(longley[1:16,2:3],bty="n")
a<-faces(longley[1:16,],plot=FALSE)
plot.faces(a,longley[1:16,2],longley[1:16,3],width=35,height=30)

set.seed(17)
faces(matrix(sample(1:1000,128,),16,8),main="random faces")

a<-faces(rbind(1:3,5:3,3:5,5:7),plot.faces=FALSE)
plot(0:5,0:5,type="n")
plot(a,x.pos=1:4,y.pos=1:4,1.5,0.7)
# during Christmastime
faces(face.type=2)
}
\keyword{misc}
```

Wrong description of aplpack version 1.2.7 :

```
For painting elements of a face the colors of are found by
```

```
averaging of sets of variables:
(7,8)-eyes, (12,13)-iris, (1,2,3)-lips,
(14,15)-ears, (9)-nose, (10)-hair, (11)-face.
```

Chunk to test the help examples.

34 ⟨* 23⟩+ ≡

```
⟨define faces 1⟩
⟨define plot.faces 20⟩
a<-faces(rbind(1:3,5:3,3:5,5:7),plot.faces=FALSE)
plot(0:5,0:5,type="n")
plot(a,x.pos=1:4,y.pos=1:4,1.5,0.7)
```

5 Definition of a spline function

35 ⟨*define* spline 35⟩ ≡ ⊂ 1, 20, 42

```
spline<-function(a,y,m=200,plot=FALSE){
    n<-length(a)
  h<-diff(a)
  dy<-diff(y)
  sigma<-dy/h
  lambda<-h[-1]/(hh<-h[-1]+h[-length(h)])
  mu<-1-lambda
  d<-6*diff(sigma)/hh
  tri.mat<-2*diag(n-2)
  tri.mat[2+  (0:(n-4))*(n-1)] <-mu[-1]
  tri.mat[    (1:(n-3))*(n-1)] <-lambda[-(n-2)]
  M<-c(0,solve(tri.mat)%*%d,0)
  x<-seq(from=a[1],to=a[n],length=m)
  anz.kl <- hist(x,breaks=a,plot=FALSE)$counts
  adj<-function(i) i-1
  i<-rep(1:(n-1),anz.kl)+1
  S.x<-  M[i-1]*(a[i]-x          )^3 / (6*h[adj(i)])  +
         M[i]  *(x        -a[i-1])^3 / (6*h[adj(i)])  +
  (y[i-1] - M[i-1]*h[adj(i)]^2 /6) * (a[i]-x)/ h[adj(i)] +
        (y[i]   - M[i]  *h[adj(i)]^2 /6) * (x-a[i-1]) / h[adj(i)]
  if(plot){ plot(x,S.x,type="l"); points(a,y)    }
  return(cbind(x,S.x))
}
```

Test of spline function:

36 ⟨*spline-test* 36⟩ ≡

```
a<-c(.25,.30,.39,.45,.53); y<-c(.5,.5477,.6245,.6708,runif(1)) # .7280)
spline(a,y,,T)
#6*(.8533-.954)/sum(h[1:2])
x<-runif(10); y<-runif(10)
xx<-spline(1:length(x),x,100,FALSE)[,2]
yy<-spline(1:length(y),y,100,FALSE)[,2]
plot(xx,yy,type="l"); points(xx,yy)
```

6 Literatur

Chernoff, H. (1973): The use of faces to represent statistiscal assoziation, JASA, 68, pp 361–368.

Ralston, A. and Rabinowitz, P. (1985): A first course in numerical analysis, McGraw-Hill, pp 76ff.

7 Appendix: The way to `faces`

7.1 Definition of characteristical points

The first step is to draw a face on a transparent sheet. Then we fix this slide in front of a monitor and read some characteristical points using the `locator` function, see following code chunk. To check the points of face we plot them on the graphics device. Only the right half of the face has been digitized. The point between the eyes is defined as `c(0,0)`.

37 ⟨*get points* 37⟩ ≡

```
plot(1,type="n",xlim=c(-100,100),ylim=c(-100,100))
abline(v=0)
result<-NULL
for(i in 1:50) {
  xy<-locator(1); text(xy$x,xy$y,i)
  result<-rbind(result,round(c(xy$x,xy$y)))
}
result<-rbind(result,cbind(-result[,1],result[,2]))
points(result[,1],result[,2],pch=".")
result
```

7.2 Saving of the characteristical points

The first 8 points lie on the line $x = 0$. Points 9 to 36 have an x-value greater 0. To show the face the left half of the face has to be added. To be able to do some corrections the points are stored in a file.

38 ⟨*save points* 38⟩ ≡

```
result[1:8,1]<-0; result1<-result[1:36,]
result1<-rbind(result1,cbind(-result1[9:36,1],result1[9:36,2]))
plot(1,type="n",xlim=c(-100,100),ylim=c(-100,100))
abline(v=0)
points(result1[,1],result1[,2],pch="*")
text(result1[,1],result1[,2],1:length(result1[,1]))
result.save<-paste(result1[,1]," ",result1[,2])
cat(file="facecoord",result.save[1:36],sep="\n")
```

7.3 Redrawing of the face

For looking at the standard face we need a chunk that reads the data from file and replots the points. The plot shows the positions of points and adds indices.

39 ⟨*show points* 39⟩ ≡

```
r<-scan(file="facecoord") # r<-rr
r<-matrix(r,ncol=2,byrow=TRUE)
```

```
r<-rbind(r,cbind(-r[9:36,1],r[9:36,2]))
plot(1,type="n",xlim=c(-100,100),ylim=c(-100,100))
abline(v=0)
points(r[,1],r[,2],pch="*")
text(r[,1],r[,2],1:length(r[,1]))
```

The next step is in defining elements of the face by sets of points. The polygons of the sets are plotted. Data input: file.

40 ⟨*show face polygon* 40⟩ ≡

```
r<-scan(file="facecoord")
result1<-r<-matrix(r,ncol=2,byrow=TRUE)
facecoor.orig<-r<-rbind(r,cbind(-r[9:36,1],r[9:36,2]))
facecoor<-list(
                      face=r[c(8:15,2,43:37,8),]
                     ,eyer=r[c(27:32,27),]
                     ,eyel=r[c(55:60,55),]
                     ,irisr=r[c(33:36,33),]
                     ,irisl=r[c(61:64,61),]
                     ,lipso=r[c(23,21,5,49,51,50,7,22,23),]
                     ,lipsi=r[c(51,52,6,24,23),]
                     ,nose=r[c(3,53,54,4,26,25,3),]
                     ,hair=r[c(41,46,45,44,1,16,17,18,13),]
                     ,nose=r[c(3,53,54,4,26,25,3),]
                     ,earr=r[c(11,20,19,12),]
                     ,earl=r[c(39,48,47,40),]
                )
plot(1,type="n",xlim=c(-100,100),ylim=c(-100,100))
for(i in seq(facecoor)) lines(facecoor[[i]][,1],facecoor[[i]][,2])
```

To get a nicer face we construct smooth lines to connect the points of the elements. The curve are found by computing spline functions. Data input: variable `facecoor`.

41 ⟨*show face smooth* 41⟩ ≡

```
plot(1,type="n",xlim=c(-100,100),ylim=c(-100,100))
for(i in seq(facecoor)) {
  x <-facecoor[[i]][,1]; y<-facecoor[[i]][,2]
  xx<-spline(1:length(x),x,100,FALSE)[,2]
  yy<-spline(1:length(y),y,100,FALSE)[,2]
  lines(xx,yy)
}
```

Now a first version of `faces` can be designed. What's to be done?

1. define `spline`
2. check input
3. fix points of standard face
4. draw a face for each row of data in a loop:
   (a) initialize face
   (b) modify points of the face according the data values
   (c) define elements of the face
   (d) plot the face

42 ⟨*define first version of face* 42⟩ ≡

```
faces1<-function(xy){
     ⟨define spline 35⟩
# standardize input
  xy<-rbind(xy); mm<-dim(xy)[2];  n<-dim(xy)[1]
  xnames<-dimnames(xy)[[1]]
  if(is.null(xnames)) xnames<-as.character(1:n)
  xy<-apply(xy,2,function(x){
                  x<-x-min(x)
                  x<-if(max(x)>0) 2*x/max(x)-1 else x+0.5
               })
  xy<-xy[,rep(1:mm,ceiling(14/mm))]
# definie points of standard face
  r<-c(0,79,0,44,0,-6,0,-31,0,-47,0,-54,0,-62,0,-89,
         25,-83,38,-61,46,-36,52,-4,54,11,
         51,22,29,40,36,74,64,50,72,12,60,
         -11,57,-30,7,-49,7,-60,16,-53,
         7,-54,3,-16,6,-30,12,0,19,8,30,
         8,37,0,30,-8,19,-8,20,0,24,4,29,0,24,-5)
  r<-matrix(r,ncol=2,byrow=TRUE)
  facecoor.orig<-rbind(r,cbind(-r[9:36,1],r[9:36,2]))
# loop over elements
  for(ind in 1:n){
   # initialize face for element ind
    factors<-xy[ind,]
    face <- facecoor.orig
   # modify face characteristics
    # head
       face[,2]<-face[,2] * ((5+factors[1])/5)
       face[,1]<-face[,1] * ((5+factors[2])/5)
       face[9:15,1]<-face[9:15,1] +
                     (face[ 9:15,2]+40)/5 * (-factors[3] )
       face[37:43,1]<-face[37:43,1] +
                     (face[37:43,2]+40)/5 * (factors[3] )
    # lips
       face[c(21:24,49:52,5:7),2]<-face[c(21:24,49:52,5:7),2] +
               ( face[c(21:24,49:52,5:7),2]+53 )* factors[4]
       face[c(21:24,49:52,5:7),1]<-face[c(21:24,49:52,5:7),1] +
               ( face[c(21:24,49:52,5:7),1] )* factors[5]/2
       face[c(23,51),2]<-face[c(23,51),2] +
               ( face[c(23,51),2]-53 )* factors[6]/15
     # eyes
        face[c(27:36,55:64),2]<-face[c(27:36,55:64),2] +
               ( face[c(27:36,55:64),2]-1)* (factors[7]) /2
        face[c(27:36),1]<-face[c(27:36),1] +
               ( face[c(27:36),1]-25)* (factors[8]) /2
        face[c(55:64),1]<-face[c(55:64),1] +
               ( face[c(55:64),1]+25)* (factors[8]) /2
      ##  face[c(27:36,55:64),1]<-face[c(27:36,55:64),1] +
      ##        (factors[??]) *5  # shift
     # hair
         face[c(16:18,44:46,1),2]<-face[c(16:18,44:46,1),2] +
               ( face[c(16:18,44:46,1),2]-50)* (factors[9])
         face[c(16:18,44:46,1),1]<-face[c(16:18,44:46,1),1] +
```

```
            ( face[c(16:18,44:46,1),1])* (factors[10])/3
   # nose
      face[c(25,26,53,54,3,4),2]<-face[c(25,26,53,54,3,4),2] +
            ( face[c(25,26,53,54,3,4),2]+25)* (factors[11])/2
      face[c(25,26,53,54),1]<-face[c(25,26,53,54),1] +
            ( face[c(25,26,53,54),1])* (factors[12])/2
   # ears
     face[c(19,20,47:48),2]<-face[c(19,20,47:48),2] +
            ( face[c(19,20,47:48),2]+20)* (factors[13])*.5
        # construct face
     face[c(20,48),1]<-face[c(20,48),1] +
          (1+factors[14])*c(1,-1)*5
           r<-face;facecoor<-list(
                    face=r[c(8:15,2,43:37,8),]
                   ,eyer=r[c(27:32,27),]
                   ,eyel=r[c(55:60,55),]
                   ,irisr=r[c(33:36,33),]
                   ,irisl=r[c(61:64,61),]
                   ,lipso=r[c(23,21,5,49,51,50,7,22,23),]
                   ,lipsi=r[c(51,52,6,24,23),]
                   ,nose=r[c(3,53,54,4,26,25,3),]
                   ,hair=r[c(41,46,45,44,1,16,17,18,13),]
                   ,nose=r[c(3,53,54,4,26,25,3),]
                   ,earr=r[c(11,20,19,12),]
                   ,earl=r[c(39,48,47,40),]
             )
      # initialize plot
  plot(1,type="n",xlim=c(-100,100)*1.1,axes=FALSE,
       ylab="",xlab=xnames[ind],ylim=c(-100,100)*1.3)
 # plot elements of the face
  for(i in seq(facecoor)) {
    x <-facecoor[[i]][,1]; y<-facecoor[[i]][,2]
    xx<-spline(1:length(x),x,20,FALSE)[,2]
    yy<-spline(1:length(y),y,20,FALSE)[,2]
    lines(xx,yy)
  }
}
xy
}
```

Some tests are necessary to experiment with the parameters of the transformations.

43 ⟨*test first version* 43⟩ ≡

```
faces(rbind(1:3,5:3,3:5,5:7))
```

A second test show the results for a data set.

44 ⟨*test first version II* 44⟩ ≡

```
data(longley)
par(mfrow=c(3,3))
faces(longley[1:9,])
par(mfrow=c(1,1))
title("longley")
longley[1:9,]
```

Now we know what to do and we can rewrite `faces` in a literate style. See p 1 ff.

45 $\langle *\ 23\rangle+\equiv$

```
data(longley)
```

46 $\langle *\ 23\rangle+\equiv$

```
print
```