Boxplots von Projektionen

H. P. Wolf

Version: 11.04.06, formatiert: April 13, 2006, file: sc.rev

1 Gegenstand

In diesem Papier wird die Funktion `boxplot2D` definiert. Mit dieser läßt sich in einen Scatterplot eines zweidimensionalen Datensatzes zu einer beliebigen Projektion der Daten ein Boxplots erstellen und geeignet im Scatterplot der Daten anbringen.

2 Definition von `boxplot2D`

Die Syntax der Funktion ergibt sich aus Kopfzeile wie aus den ersten Kommentarzeilen der Funktion:

1 ⟨*start* 1⟩≡

```
boxplot2D<-function( xy, add.to.plot=TRUE, box.size=10, box.shift=0, angle=0,
                angle.type="0", tukey.style=TRUE, coef.out=1.5, coef.h.out=3, design="sl",... ){
##############################################################################
# boxplot for scatterplots, pw 03/05                                         #
# xy:          2-col matrix                                                  #
# add.to.plot: if TRUE => plot(xy,...)                                       #
# box.size:    height of box in mm                                           #
# box.shift:   shift of boxplot in mm                                        #
# angle:       direction of projection in units defined by angle.type        #
# angle.type:  "0"=     angle    , # angle in (0,2*pi)                       #
#              "1"=2*pi*angle/12,  # clock-like                            #
#              "2"=2*pi*angle/360, # angle in (0,360)                       #
#              "3"=arctan(angle)   # by fraction: delta.y/delta.x           #
# tukey.style: if TRUE outliers are defined as described in Tukey (1977): EDA
# coef.out=1.5  # outliers are defined outside coef.out*boxwidth
# coef.h.out=3   # heavy outliers are defined outside coef.h.out*boxwidth
# design:      if "sl" then parallelogram else box                           #
##############################################################################
⟨zeichne ggf. Scatterplot 3⟩
⟨konstruiere Rotationsmatrix TM mittels angle und angle.type 4⟩
⟨ermittle Quantile der gedrehten Daten und retransformiere: xy.q 5⟩
⟨berechne für Box Vektor für Boxhöhe: real.step 6⟩
⟨realisiere gewünschte Verschiebung 9⟩
⟨erstelle Boxplot 10⟩
} # end of boxplot2D
```

2 ⟨*definiere Hilfe von* boxplot2D 2⟩≡

```
\name{boxplot2D}
\alias{boxplot2D}
\title{ Boxplot of projection of two dimensional data }
\description{
  boxplot2D computes summary statistics of a one dimensional
  projection of a two dimensional data set and plots a sloped
  boxplot of the statistics into the scatterplot of the two
  dimensional data set.
}
\usage{
    boxplot2D(xy, add.to.plot = TRUE, box.size = 10, box.shift = 0,
    angle = 0, angle.type = "0", tukey.style = TRUE, coef.out = 1.5,
    coef.h.out = 3, design = "sl", ...)
}
\arguments{
  \item{xy}{ \code{(nx2)}-matrix, two dimensional data set  }
  \item{add.to.plot}{ if \code{TRUE} the boxplot is added to
      the actual plot of the graphics device }
  \item{box.size}{ height of the box (of the boxplot) }
  \item{box.shift}{ shift of boxplot perpendicular to the
  projection direction }
  \item{angle}{ direction of projection in units defined by
  angle.type }
  \item{angle.type}{
    \code{"0"}: angle in (0,2*pi),
    \code{"1"}: clock-like: angle.typ.0==2*pi*angle.typ.1/12,
    \code{"2"}: degrees: angle.typ.0==2*pi*angle.typ.2/360,
    \code{"3"}: by fraction: delta.y/delta.x
 }
  \item{tukey.style}{ if \code{TRUE} outliers are defined as described
    in Tukey (1977)
 }
  \item{coef.out}{ outliers are values that are more than
     \code{coef.out*boxwidth} away from the box, default:
     \code{coef.out=1.5}  }
  \item{coef.h.out}{ heavy outliers are values that are more
      than \code{coef.h.out*boxwidth} away from the box,
      default: \code{coef.h.out=3}   }
  \item{design}{ if \code{sl} then parallelogram else box  }
  \item{\dots}{ additional graphical parameters }
}
\references{
    Tukey, J.
    \emph{Exploratory Data Analysis.}
    Addison-Wesley, 1977.
    }
\author{Peter Wolf }
\note{ version 08/2003 }
\seealso{ \code{\link[graphics]{boxplot}}}
\examples{
 xy<-cbind(1:100, (1:100)+rnorm(100,,5))
 plot(xy,xlim=c(-50,150),ylim=c(-50,150))
 boxplot2D(xy,box.shift=-30,angle=3,angle.typ=1)
 boxplot2D(xy,box.shift=20,angle=1,angle.typ=1)
 boxplot2D(xy,box.shift=50,angle=5,angle.typ=1)
}
\keyword{misc}
```

Ohne Kommentar.

3 ⟨*zeichne ggf. Scatterplot* 3⟩≡

```
if(!add.to.plot) plot(xy,...)
```

Benötigt wird eine (2,2)-Rotationsmatrix – `TM`.

4 ⟨*konstruiere Rotationsmatrix* `TM` *mittels* `angle` *und* `angle.type` 4⟩≡
```
if(is.numeric(angle.type)){
  angle.type<-as.character(angle.type)
  if(all(angle.type!=c("0","1","2","3"))) angle.type<-"0"
}
w <- switch(angle.type, "0"=     angle, "1"=2*pi*(3-angle)/12,
                        "2"=2*pi*angle/360, "3"=atan(angle) )
TM <- matrix(c(cos(w),sin(w),-sin(w),cos(w)),2,2)
```

Als Standardposition wird der Rerotierte Median der rotierten y-Werte verwendet. Nach Tukey: Die Rolle der Quartile übernehmen die Angeln (hinges), ihr Abstand heißt Angelabstand (H-Spread).
Ein Schritt (step) ist das Produkt H-Spread $*$ factor z.B. mit factor=`coef.default==1.5`
inner fence: [lower hinge − step, upper hinge +step]
Ausreißer: (upper hinge + step, upper hinge + 2step]= (upper hinge + 1.5∗H-Spread, upper hinge + 3∗H-Spread] =(fence, fence + 1.5∗H-Spread] sowie entsprechend links.
heavy outliers: (fence+1.5H-Spread , unendlich) sowie entsprechend links.

5 ⟨*ermittle Quantile der gedrehten Daten und retransformiere:* `xy.q` 5⟩≡
```
xyt<- xy %*% TM
##ermittle empirsche Quantile##=
z  <- xyt[,1]
if(tukey.style){
  z.stats1 <- boxplot.stats(z,coef=coef.out)
  z.stats2 <- boxplot.stats(z,coef=coef.h.out)
  z<-c(min(z),z.stats1$stats,max(z))
  names(z) <- c("min","fence","hinge","median","hinge","fence","max")
  outlier.heavy <- z.stats2$out
  outlier <- z.stats1$out
  outlier <- outlier[! outlier %in% outlier.heavy ]
}else{
  z  <-     c(min(z),quantile(z,c(0.10, 0.25, 0.5, 0.75, 0.90)),max(z))
  names(z)<-c("min",              ".1", ".25",".5",".75",".9", "max")
}
xy.q <- cbind(z,median(xyt[,2])) %*% t(TM)
if(tukey.style){
  xy.out<-if(0<length(outlier)) cbind(outlier,median(xyt[,2])) %*% t(TM) else numeric(0)
  xy.out.heavy<-if(0<length(outlier.heavy))
                                      cbind(outlier.heavy,median(xyt[,2])) %*% t(TM) else numeric(0)
}
```

Die Ermittlung eines Höhenvektors für die Box ist gar nicht so einfach. Außerdem werden zwei Designs verfolgt. Soll sich das Erscheinungsbild der Box, wie auch immer die Weltkoordinaten sind und welche Größenverhältnisse der graphische Device aufweist, durch ein Rechteck auszeichnen? Dieses wird erreicht durch `design!="sl"`. Angemessener für viele Probleme werden jedoch parallelogrammförmige *Boxen* sind, die auch bei einer Veränderung des Device sich entsprechend mitändern und im Weltkoordinatensystem immer Rechtecke sind. Das Parallelgramm soll dann mit jenem übereinstimmen, daß sich einstellt, wenn bei einem quadratischen Device und gleichen Bereichen für x und y der Device gestreckt wird. Für die Berechnung des Boxhöhenvektors sind auf jeden Fall erforderlich: Spannweiten der Wertebereiche der Weltkoordinaten `d.xy`, Spannweiten des Viewports in Weltkoordinaten `d.usr` und Maße der Viewports in mm `d.pmm`. `d.` für `delta` deutet auf Differenzen zwischen maximalen und minimalen Werten hin.

6 ⟨*berechne für Box Vektor für Boxhöhe:* `real.step` 6⟩≡
```
# d. steht fuer delta.:
d.xy <-apply(xy.q[c(1,7),],           2,diff)
d.usr<-apply(matrix(par()$usr,2,2),2,diff)
d.pmm<-par()$pin*25.4
if(design=="sl"){
    ⟨ermittle Boxhöhenvektor real.step für schräges Design 7⟩
}else{
    ⟨ermittle Boxhöhenvektor real.step für Box-Design 8⟩
}
```

In der schrägen Form wird im Weltkoordinatensystem ein Boxplot erstellt und in die geeignete Position gebracht, was in der Regel zu schrägen Erscheinungsbildern führen wird. Im Weltkoordinatensystem liegt jedoch ein Rechteck vor. Zu beachten ist, daß die gewünschte Boxhöhe in mm angegeben wird. Was ist zu tun?
Zunächst wird zu einem Vektor, der entlang des Boxplots zeigt, ein in Weltkoordinaten senkrechter Vektor `d.xy.s.mm` konstruiert und im mm-Koordinaten überführt. Dann wird ein Einheitsvektor senkrecht zu Verlauf des Boxplots in mm-Koordinaten `d.xy.0.s` erstellt. `d.xy.s.mm` zeigt in die gewünschte Richtung und muß proportional zu dem Wunsch `box.size` verlängert werden. Vorher ist jedoch `d.xy.s.mm` so zu strecken, daß seine Projektion auf `d.xy.0.s` einen Einheitsvektor ergibt, und deshalb durch die Länge der Projektion zu teilen: `d.xy.s.mm.proj1`. Zum Schluß ist eine Retransformation in Weltkoordinaten notwendig.

7 ⟨*ermittle Boxhöhenvektor* `real.step` *für schräges Design* 7⟩≡
```
# d.xy.s.mm: bzgl. Welt-coor senkrechten Vektor in mm-coor
d.xy.s.mm <- (c(-1,1)*rev(d.xy))/d.usr*d.pmm
# d.xy.0.s: bzgl. mm-coor senkrechten Einheits-Vektor
d.xy.mm.s <- c(-1,1)*rev( d.xy/d.usr*d.pmm )
d.xy.0.s  <- d.xy.mm.s/(d.xy.mm.s%*%d.xy.mm.s)^0.5
# Streckung von d.xy.s.mm so, dass Projektion auf d.xy.0.s von Laenge 1
d.xy.s.mm.proj1 <- d.xy.s.mm / (d.xy.s.mm %*% d.xy.0.s)
# Boxsize: in mm
real.step.mm <- box.size * d.xy.s.mm.proj1
# Darstellung in Welt-coor
real.step <- real.step.mm/d.pmm*d.usr
```

Die Idee für das Box-Design ist etwas einfacher. Nehme den Vektor, der längs des Boxplots zeigt (`d.xy`), transformiere diesen in mm-Koordinaten und ermittle den zugehörigen senkrechten Vektor. Verlängere diesen Boxhöhenvektor gemäß der gewünschten `box.size`. Retransformiere Höhenvektor in Weltkoordinaten.

8 ⟨*ermittle Boxhöhenvektor* `real.step` *für Box-Design* 8⟩≡
```
# Einheitsvektor zu d.xy in mm-coor
d.xy.mm   <- d.xy/d.usr*d.pmm
d.xy.mm.ev<- d.xy.mm/(d.xy.mm%*%d.xy.mm)^0.5
# Konstruktion eines Boxhoehenvektors der Laenge box.size in mm
mm.step   <- box.size*c(-1,1)*rev(d.xy.mm.ev)
# Darstellung in Welt-coor
real.step <- mm.step/d.pmm*d.usr # real.step fuer Boxdickenvektor
```

Der gewünschte Shift verhält sich zur gewünschten Boxdicke wie der realisierte Shift zur realisierten Boxdicke. Folglich ergibt sich der zu realisierende Shift durch Multiplikation der gewünschten Shiftes mit dem Quotienten aus realisiertem `real.step` und gewünschter Dicke.

9 ⟨*realisiere gewünschte Verschiebung* 9⟩≡
```
# Boxshift: in mm
real.shift<-box.shift*real.step/box.size
xy.q.orig <- xy.q
xy.q      <-cbind(xy.q[,1]+real.shift[1], xy.q[,2]+real.shift[2])
if(tukey.style){
  if(0<length(xy.out)) xy.out<-cbind(xy.out[,1]+real.shift[1], xy.out[,2]+real.shift[2])
  if(0<length(xy.out.heavy))
            xy.out.heavy<-cbind(xy.out.heavy[,1]+real.shift[1], xy.out.heavy[,2]+real.shift[2])
}
```

Mit Hilfe der transformierten Quantile `xy.q` und des halben vertikalen Boxhöhen-Vektors `hs` – ermittelt aus `real.step` – ist die Erstellung der Boxplot kein großes Problem mehr. Die Extrema werden durch Punkte und nicht durch Striche dargestellt, da sie andernfalls aufgrund perspektivischer Verzerrungen eher zu Fehlinterpretationen verleiten würden. Der Code für Abschlußstriche ist jedoch in Kommentarzeilen vermerkt worden.

10 ⟨*erstelle Boxplot* 10⟩≡

```
# Boxplotkonstruktion
hs<-real.step/2
xyxy <-c(
          #min:xy.q[1,1]-hs[1],xy.q[1,2]-hs[2],xy.q[1,1]+hs[1],xy.q[1,2]+hs[2],
          xy.q[2,1]      ,xy.q[2,2]      ,xy.q[3,1]      ,xy.q[3,2]      ,
          xy.q[3,1]-hs[1],xy.q[3,2]-hs[2],xy.q[5,1]-hs[1],xy.q[5,2]-hs[2],
          xy.q[3,1]-hs[1],xy.q[3,2]-hs[2],xy.q[3,1]+hs[1],xy.q[3,2]+hs[2],
          xy.q[4,1]-hs[1],xy.q[4,2]-hs[2],xy.q[4,1]+hs[1],xy.q[4,2]+hs[2],
          xy.q[5,1]-hs[1],xy.q[5,2]-hs[2],xy.q[5,1]+hs[1],xy.q[5,2]+hs[2],
          xy.q[3,1]+hs[1],xy.q[3,2]+hs[2],xy.q[5,1]+hs[1],xy.q[5,2]+hs[2],
          xy.q[6,1]      ,xy.q[6,2]      ,xy.q[5,1]      ,xy.q[5,2]
          #max ,xy.q[7,1]-hs[1],xy.q[7,2]-hs[2],xy.q[7,1]+hs[1],xy.q[7,2]+hs[2]
        )
xyxy<-matrix(xyxy,length(xyxy)/4,4,TRUE)
segments(xyxy[,1],xyxy[,2],xyxy[,3],xyxy[,4])
if(tukey.style){
  if(0<length(xy.out)) points(xy.out[,1],xy.out[,2],pch=1,cex=1.5)
  if(0<length(xy.out)) points(xy.out[,1],xy.out[,2],pch=18)
  if(0<length(xy.out.heavy)) points(xy.out.heavy[,1],xy.out.heavy[,2],pch=19,cex=1.5)
}else{
  points(xy.q[c(1,7),1],xy.q[c(1,7),2],pch=19)
}
```

3 Einige Demo-Aufrufe

11 ⟨* 11⟩≡

```
x<-y<-rcauchy(50); plot(x,y);boxplot2D(cbind(x,y),angle=0,angle.type="0")
```

12 ⟨*demos* 12⟩≡

```
data(co2)
plot(co2,xlim=c(1940,2030),ylim=c(300,400))
h<-lsfit(time(co2),co2)
abline(h)
boxplot2D(cbind(time(co2),co2),angle=h$coef[2],angle.type="3",
     box.shift=5,box.size=10,design=0)
cat("RETURN!\n");readline()

data(co2)
plot(co2,xlim=c(1940,2030),ylim=c(300,400))
h<-lsfit(time(co2),co2)
abline(h)
boxplot2D(cbind(time(co2),co2),angle=h$coef[2],angle.type="3",
     box.shift=10,box.size=10,design="sl")

cat("RETURN!\n");readline()
xx<-(1:50);   yy<-10*xx+sort(rnorm(50,,30))
bs<-10;bsh<-20;ang<-9;at<-"3"

plot(xx,yy,xlim=c(-50,100),ylim=c(-200,700))
boxplot2D(cbind(xx,yy),angle=ang,angle.type=at,box.size=bs,design=0)
cat("RETURN!\n");readline()
boxplot2D(cbind(xx,yy),angle=ang,angle.type=at,box.size=bs,box.shift=bsh,design=0)
cat("RETURN!\n");readline()
boxplot2D(cbind(xx,yy),angle=-ang,angle.type=at,box.size=bs)
cat("RETURN!\n");readline()
boxplot2D(cbind(xx,yy),angle=-ang,angle.type=at,box.size=bs,box.shift=-bsh)
cat("RETURN!\n");readline()
xxx<-c(-80,-30,xx,95,140)
plot(xxx,xxx,xlim=c(-200,200),ylim=c(-200,200))
boxplot2D(cbind(xxx,xxx),angle=ang,angle.type=at,box.size=bs,box.shift=-bsh)

#boxplot2D(cbind(xx,yy),angle=60,angle.type=at,box.size=30)
#boxplot2D(cbind(xx,yy),angle=30,angle.type=at,box.shift=bsh,box.size=30)
```