Punktwolkenrotation

H. P. Wolf

September 26, 2014, file: spin3R.rev

Die in diesen Abschnitt definierte Funktion ermöglicht dem Anwender eine Punktwolke interaktiv zu drehen und zu betrachten.

1 ⟨*start* 1⟩ ≡
⟨*definiere* spin3R 3⟩

2 ⟨*definiere Hilfe von* spin3R 2⟩ ≡

```
\name{spin3R}
\alias{spin3R}
\title{ spin3R }
\description{
  Simple spin function to rotate and to inspect
  a 3-dimensional cloud of points
}
\usage{
  spin3R(x, alpha = 1, delay = 0.015, na.rm=FALSE)
}
\arguments{
  \item{x}{ \code{(nx3)}-matrix of points }
  \item{alpha}{ angle between successive projections }
  \item{delay}{ delay in seconds between two plots }
  \item{na.rm}{ if TRUE 'NA' values are removed otherwise exchanged by mean}
}
\details{
  \code{spin3R} computes two-dimensional projections
  of \code{(nx3)}-matrix \code{x} and plots them
  on the graphics devise. The cloud of points is rotated
  step by step. The rotation is defined by a tcl/tk control
  widget. \code{spin3R} requires tcl/tk package of R.
}
\references{
   Cleveland, W. S. / McGill, M. E. (1988): Dynamic Graphics
   for Statistics. Wadsworth & Brooks/Cole, Belmont, California.
}
\author{ Peter Wolf }
\note{  version 01/2003 }
\seealso{ \code{spin} of S-Plus }
\examples{
  xyz<-matrix(rnorm(300),100,3)
  # now start:     spin3R(xyz)
}
\keyword{misc}
```

3 ⟨*definiere* spin3R 3⟩ ≡ ⊂ 1

```
spin3R <- function(x, alpha=1, delay=.015, na.rm=FALSE){
 ###################################################################
```

```
 # spin3R: simple spin function to rotate a 3-dim cloud of points#
 # pwolf 070831                                                  #
 #                                                                  #
 # arguments:                                                       #
 #                                                                  #
 #  x             (nx3)-matrix of points                            #
 #  alpha         arc of rotation                                   #
 #  delay         sleeping time between rotations                   #
 #                                                                  #
 ####################################################################
if(ncol(x)!=3) { print("Error: data matrix must have 3 columns"); return() }
 # require(tcltk) # 131104
 ⟨generiere Steuerungsfenster 4⟩
 ⟨definiere Rotationen 6⟩
 ⟨definiere Bindungen 5⟩
 ⟨initialisiere Plot 7⟩
 ⟨starte Endlosschleife 8⟩
 ⟨entferne Steuerungsfenster 9⟩
}
```

4 ⟨*generiere Steuerungsfenster* 4⟩ ≡ ⊂ 3

```
Rot <-tclVar("relax");bw <- 4
topl<-tktoplevel();   tkwm.geometry(topl,"+100+100")
f1 <- tkframe(topl);f2 <- tkframe(topl);f3 <- tkframe(topl)
f4 <- tkframe(topl);f5 <- tkframe(topl);tkpack(f1,f2,f3,f4,f5)

b12 <- tkbutton(f1, relief="ridge",  width=bw, text="up")
b21 <- tkbutton(f2, relief="ridge",  width=bw, text="left")
b22 <- tklabel(f2,  relief="flat",   width=bw)
b23 <- tkbutton(f2, relief="ridge",  width=bw, text="right")
b32 <- tkbutton(f3, relief="ridge",  width=bw, text="down")
b41 <- tkbutton(f4, relief="ridge",  width=bw, text="clock")
b42 <- tklabel(f4,  relief="flat",   width=bw)
b43 <- tkbutton(f4, relief="ridge",  width=bw, text="cclock")
b51 <- tkbutton(f5, relief="raised", width=bw, text="reset")
b52 <- tklabel(f5,  relief="flat",   width=bw)
b53 <- tkbutton(f5, relief="raised", width=bw, text="exit")
tkpack(b12,b32)
tkpack(b21,b22,b41,b42,b51,b52,side="left")
tkpack(b23,b43,b53,side="right")
```

5 ⟨*definiere Bindungen* 5⟩ ≡ ⊂ 3

```
for(type in c("12","21","23","32","41","43")){
 b<-eval(parse(text=paste("b",type,sep="")))
 tkbind(b, "<Enter>",
     eval(parse(text=paste("function()tclvalue(Rot)<-\"",type,"\"",sep=""))))
 tkbind(b, "<Leave>",function() tclvalue(Rot) <- "relax")
}
tkconfigure(b51,command=function() tclvalue(Rot) <- "reset" )
tkconfigure(b53,command=function() tclvalue(Rot) <- "exit" )
```

Für die Rotation bezüglich zwei Achsen wird nur eine 2×2-Rotationsmatrix benötigt.

6 ⟨*definiere Rotationen* 6⟩ ≡ ⊂ 3

```
alpha<-alpha/360*2*pi; ca<-cos(alpha); sa<-sin(alpha)
rot<-matrix(c(ca,-sa,sa,ca),2,2)
```

`x` hält die Daten, `x.o` die Originaldaten, `xa` die 2-dim Projektionen. Für die Anschaulichkeit wird ein

Andeutung der Achsen mitgeliefert: `A` beschreibt die Achsen, `A.o` die Originalachsen, `Aa` den darzustellenden Teil.

7 ⟨*initialisiere Plot* 7⟩ ≡ ⊂ 3

```
n <- nrow(x)
if(any(is.na(x))){
  if(na.rm){ x<-x[!apply(is.na(x),1,any),,drop=FALSE]
    print("Warning: NA elements have been removed!!")
  }else{
    xy.means<-colMeans(x,na.rm=TRUE)
    for(j in 1:ncol(x)) x[is.na(x[,j]),j]<-xy.means[j]
    print("Warning: NA elements have been exchanged by mean values!!")
  }
}
x <- x - matrix(apply(x,2,min),n,3,TRUE)
x.o<-x<-x / matrix(apply(x,2,max),n,3,TRUE) - 0.5;                xa <- x[,2:3]
A.o<-A<-0.5*matrix(c(1,0,0, 0,0,0, 0,1,0, 0,0,0, 0,0,1),5,3,TRUE);Aa <- A[,2:3]
plot(xa, xlim=.7*c(-1,1), ylim=.7*c(-1,1),
          pch=20, xlab="",ylab="",xaxt="n",yaxt="n")
lines(Aa)
```

8 ⟨*starte Endlosschleife* 8⟩ ≡ ⊂ 3

```
i <- 0              #  ; i.max<-100
cat("exit by button Exit\n")
if(delay < 0.015) delay <- 0.015
repeat{
  Sys.sleep(delay)
  choice <- tclvalue(Rot)
  if(choice=="exit"
                  #  || ((i<-i+1)>i.max)
                   ){ break }
  if(choice=="relax") next
  if(choice=="reset") {
    points(xa, pch=20, col="white"); lines(Aa, col="white")
    x <- x.o; A <- A.o; xa<-x[,2:3]; Aa<-A[,2:3]
    points(xa, pch=20, col="black"); lines(Aa, col="black")
    tclvalue(Rot)<-"relax"; next
  }
  switch(choice,
   "12" = ind<-c(1,3), "21" = ind<-c(2,1), "23" = ind<-c(1,2),
   "32" = ind<-c(3,1), "41" = ind<-c(3,2), "43" = ind<-c(2,3)
  )
  x[,ind] <- x[,ind]%*%rot; A[,ind] <- A[,ind]%*%rot
  points(xa, pch=20, col="white"); lines(Aa, col="white")
  xa<-x[,2:3]; Aa<-A[,2:3]
  points(xa, pch=20, col="black"); lines(Aa, col="black")
}
```

9 ⟨*entferne Steuerungsfenster* 9⟩ ≡ ⊂ 3

```
tkdestroy(topl)
"control widget closed"
```

Testbeispiel:

10 ⟨* 10⟩ ≡

```
x<-matrix(sample(1:333),111,3)
spin3R(x)
```

11 ⟨* 10⟩+ ≡

```
# show planes of "randu" random number generator:
random.gkg<-function(n.max,m,a,r,x){
   res<-1:n.max
   for(i in 1:n.max){res[i] <- x <- (a*x+r) %% m }; res
}
# randu:
res<-random.gkg(1000, 2^31, 65539, 0, 100000)/2^31
# define cloud of points:
xyz<-cbind(res[-c(length(res),length(res)-1)],
             res[-c(1,length(res))],res[-c(1:2)])
spin3R(xyz)
```