Generic functions for plant epidemiology

Simulations for desease detection

Installation

To install the package, execute:

```
cred <- git2r::cred_user_pass("username", "password")
devtools::install_git(
  "http://pine--is.grid.private.cam.ac.uk:8888/gitlab/dbb31/Strategy.git",
  credentials = cred)
```

where ' "username" ' and ' "password" ' need to be replaced by correct strings.

Generate a tree population that matches the ACLUMP data:

The following script creates a tree population based on the ACLUMP data.

```
data(CLUM_Commodities_2018_v2, package = "dataACLUMP")
    sh <- CLUM_Commodities_2018_v2[CLUM_Commodities_2018_v2$Commod_dsc ==
        "avocados", ]
    set.seed(1)
    p = 0.1 + 0.9 * exp(-sh$Area_ha)
    sh$population <- 2 + rbinom(n = nrow(sh), size = round(200 *
        sh$Area_ha/p), p)
    sum(sh$population)
    sh$Avo_id <- 1:nrow(sh)
    U2 <- Generate_U(sh, .id = "Avo_id", .spatialobject = "Area_ha",
        type = "regular")
```

To visualize the data, one can use code like the following (only three fields out of 797 are selected:

```
library(Strategy)
data(Avo_fields,package="Strategy")
data(U2,package="Strategy")

U2<-U2[2:4]
#Plot the trees
Avo_fields$Source_yr<-addNA(as.factor(Avo_fields$Source_yr))

QLD<-Avo_fields$State=="Qld"

Avo_ids<-unique(Avo_fields[QLD,]$Avo_id)[c(1,3,4)]
QLD<-QLD&is.element(Avo_fields$Avo_id,Avo_ids)
QLDt<-is.element(U2$id,Avo_ids)

yearpal <- colorFactor(heat.colors(5),
                        domain = levels(Avo_fields$Source_yr),
                        na.color = "#aaff56")

leaflet(Avo_fields[QLD,]) %>%
  addProviderTiles('Esri.WorldImagery',
                   options = providerTileOptions(minZoom = 1,
                                                 maxZoom = 21,
                                                 maxNativeZoom=19)) %>%
  addProviderTiles("CartoDB.PositronOnlyLabels")%>%
  addPolylines(fillOpacity = 1, weight = 3, smoothFactor = 0.5,opacity = 1.0,
               color=~yearpal(Avo_fields[QLD,]$Source_yr),
               fillColor=~yearpal(Avo_fields[QLD,]$Source_yr))%>%
  addMarkers(lng = U2[QLDt,]$x1,
             lat = U2[QLDt,]$x2,
             clusterOptions = markerClusterOptions())
```

We have 797 avocado fields in Queensland from ACLUMP. The desease is spreading by root, so we can consider that root to root transmission is not possible for trees separated by more than the maximum avocado root reach. I read that the avocado should be planted at least 30 feet from houses, so I think it is safe to say that fields separated by more than 200 m will not contaminate each other via their roots. So we can simulate the in-field spread independantly for each of these fields, which means we can parallelize computations. We then need to compute the distances between each fields. I wrote some functions that compute distances between polygons.

The ACLUMp data uses the following projection system:

```
data(Avo_fields)
sp::proj4string(Avo_fields)
```

```
## [1] "+proj=longlat +ellps=GRS80 +no_defs"
```

The following code converts the longitude latitude values using the projection EPSG:3107 (GDA94/SA Lambert).

```
projAvo_fields<-sp::spTransform(Avo_fields,CRS("+init=epsg:3107 +units=m"))
```

The following code extracts all the projected polygons.

```
list.poly<-Strategy::extractpolygonsaslist(projAvo_fields)
```

The following code gets the list of couple of fields and their distances.

```
Australia_avo_fields_small_dist_matrix<-
  Australia_avo_fields_small_dist_matrix_f(delta=Inf)
```

From this list, one can establish a clustering of the different fields, in which simulation of root-to-root infection can be done independently.

```
Australia_connected_fields200<-Australia_connected_fields_f(delta=Inf)
```

We print the number of fields and the number of bins:

```
data(Australia_connected_fields200)
lapply(Australia_connected_fields200,function(x){length(unique(x))})
```

```
## $polygon
## [1] 797
##
## $bin
## [1] 388
```

This means we can parallelize the simulation over 388 clusters.

This is what we do when we simulate an epidemy with

```r
U2E2<-generate_Avo_Epi_1()
```

The result on a selection of fields can be seen by executing

```r
library(Strategy)
data(Avo_fields,package="Strategy")
data(U2E2,package="Strategy")

U2E2<-U2E2[c(2:4,12:112)]
names(U2E2)[1:2]<-c("long","lat")
#Plot the trees
Avo_fields$Source_yr<-addNA(as.factor(Avo_fields$Source_yr))

QLD<-Avo_fields$State=="Qld"

Avo_ids<-unique(Avo_fields[QLD,]$Avo_id)[c(1:10)]
QLD<-QLD&is.element(Avo_fields$Avo_id,Avo_ids)
QLDt<-is.element(U2$id,Avo_ids)


yearpal <- colorFactor(heat.colors(5),
                       domain = levels(Avo_fields$Source_yr),
                       na.color = "#aaff56")

leaflet(Avo_fields[QLD,]) %>%
  addProviderTiles('Esri.WorldImagery',
                   options = providerTileOptions(minZoom = 1,
                                                 maxZoom = 21,maxNativeZoom=19)) %>%
  addProviderTiles("CartoDB.PositronOnlyLabels")%>%
  addPolylines(fillOpacity = 1, weight = 3, smoothFactor = 0.5,opacity = 1.0,
               color=~yearpal(Avo_fields[QLD,]$Source_yr),
               fillColor=~yearpal(Avo_fields[QLD,]$Source_yr))%>%
 addpiechartclustermarkers(.data=U2E2[is.element(U2E2$id,Avo_ids),]
                           ,.colors=c("green","red","orange","purple","black"),
                           group="I012")
```

```r
library(Strategy)
data(Avo_fields,package="Strategy")
data(U2E2,package="Strategy")

U2E2<-U2E2[c(2:4,12:112)]
names(U2E2)[1:2]<-c("long","lat")
#Plot the trees
Avo_fields$Source_yr<-addNA(as.factor(Avo_fields$Source_yr))

QLD<-Avo_fields$State=="Qld"

Avo_ids<-unique(Avo_fields[QLD,]$Avo_id)[c(1:10)]
QLD<-QLD&is.element(Avo_fields$Avo_id,Avo_ids)
QLDt<-is.element(U2$id,Avo_ids)


yearpal <- colorFactor(heat.colors(5),domain = levels(Avo_fields$Source_yr),na.color = "#aaff56")
```

```
leaflet(Avo_fields[QLD,]) %>%
  addProviderTiles('Esri.WorldImagery',
                   options = providerTileOptions(minZoom = 1,
                                                 maxZoom = 21,maxNativeZoom=19)) %>%
  addProviderTiles("CartoDB.PositronOnlyLabels")%>%
  addPolylines(fillOpacity = 1, weight = 3, smoothFactor = 0.5,opacity = 1.0,
               color=~yearpal(Avo_fields[QLD,]$Source_yr),
               fillColor=~yearpal(Avo_fields[QLD,]$Source_yr))%>%
 addpiechartclustermarkers(.data=U2E2[is.element(U2E2$id,Avo_ids),]
                           ,.colors=c("green","red","orange","purple","black"),
                           group="I085")
```

Details and code demo

Data

I wrote a package `dataACLUMP` a contain functions (`dataACLUMP::get_data_from_web`) to download the ACLUMP data and create data files.

After installing the package dataACLUMP from gitlab, the last commodities data files can be obtained by executing

```
data(CLUM_Commodities_2018_v2, package = "dataACLUMP")
```

Plotting the epidemic

This function ‘addpiechartclustermarkers’ allows to replace cluster markers by cluster pie charts on leaflet.

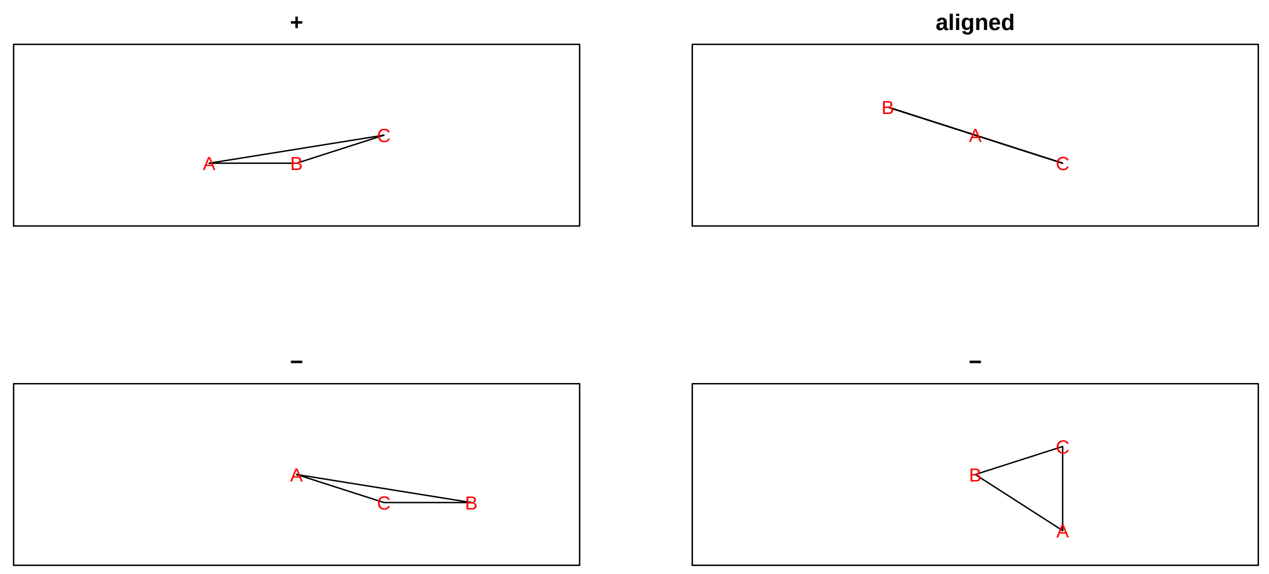
```
library(Strategy)
data("breweries91",package="leaflet")
breweries91$goodbear<-sample(as.factor(c("terrific","marvelous","culparterretaping")),
                             nrow(breweries91),
                             replace=T)
leaflet(breweries91) %>%
 addTiles() %>%
 addpiechartclustermarkers(.data=breweries91,.colors=c("red","green","blue"),group="goodbear")
```

Computing distances between polygons

The data from ACLUMP contains longitudes and latitudes of vertices of fields, given as polygons. To compute distances between two points identified with their longitudes or latitudes, one can use native R functions from the packages rgdal or . To compute the minimum distance between two fields, I had to write down my own functions. The distance between two polygons can be computed by the minimum distance separating the different edges of the two polygons. The different functions described in this section allow to compute this distance.

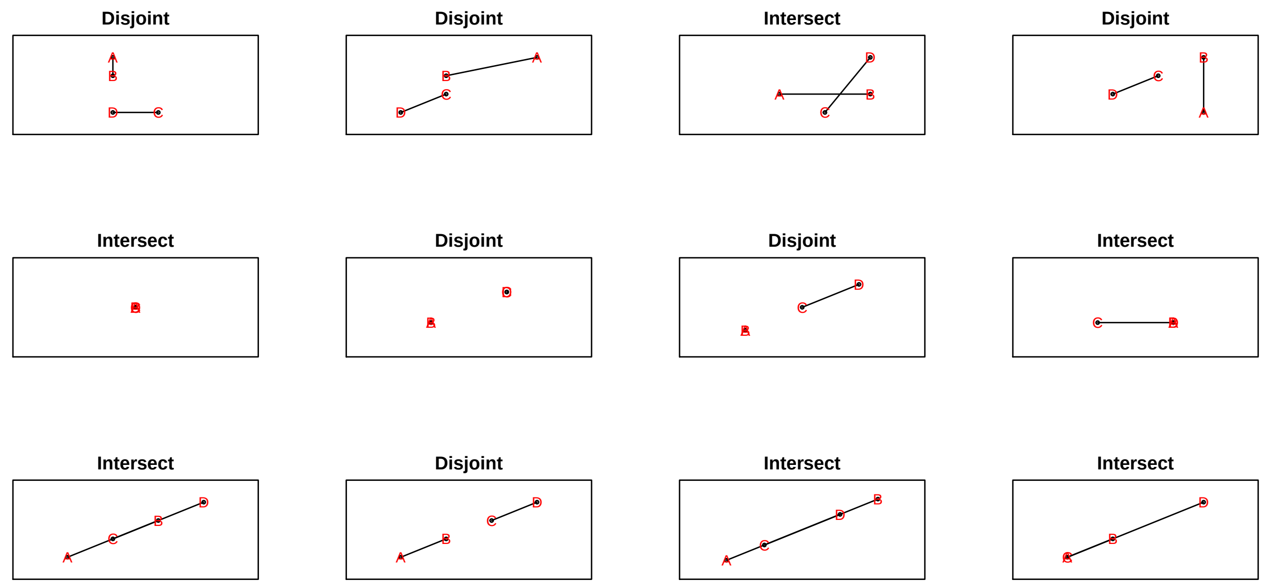
A function of two triangles have the same orientation.

```
example(triangleorientation,echo=F)
```



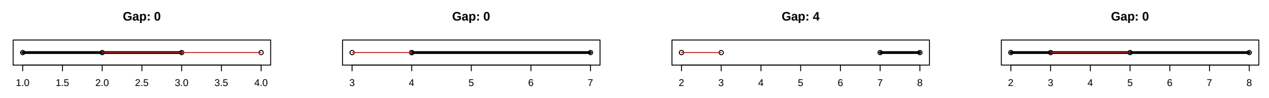
A function to tell if two segments intersect.

```r
example(segment.intersect,echo=F)
```

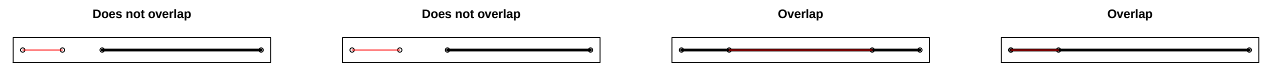


```r
#example(Generate_Discrete_Time_Epidemics,echo=F)
```

```r
example(ranges.gap,echo=F)
```



```r
example(rangesoverlap,echo=F)
```



```r
example(Generate_U,echo=F)
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/tmp/RtmpH1QUdu/Parishes_December_2011_Boundaries_EW_BFC.shp", layer: "Parishes_December_20
## with 11358 features
## It has 7 fields
## Integer64 fields read as strings:  objectid
```

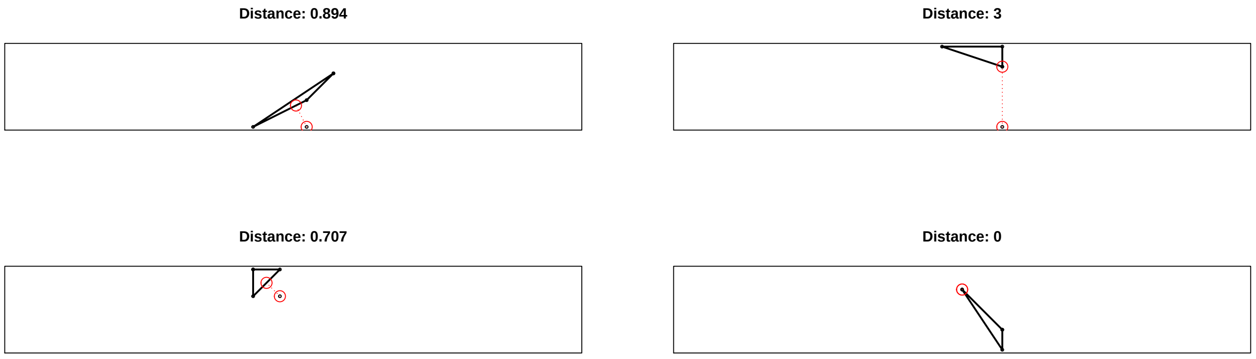
```
example(risktobeinfectedbydistancetooneinfectedunit)
```

```
##
## rsktbn> #Risk to be ingfected 2 m from the victim when the 50%risk distance is 1 m:
## rsktbn> risktobeinfectedbydistancetooneinfectedunit(2,1)
## [1] 0.05583301
```

projpointonseg_a is a function that gives the position of the closest point of a segment to a specific point on the plane. The position is the ratio of the distance of the closest point to one extremity of the segment divided by the length of the segment.

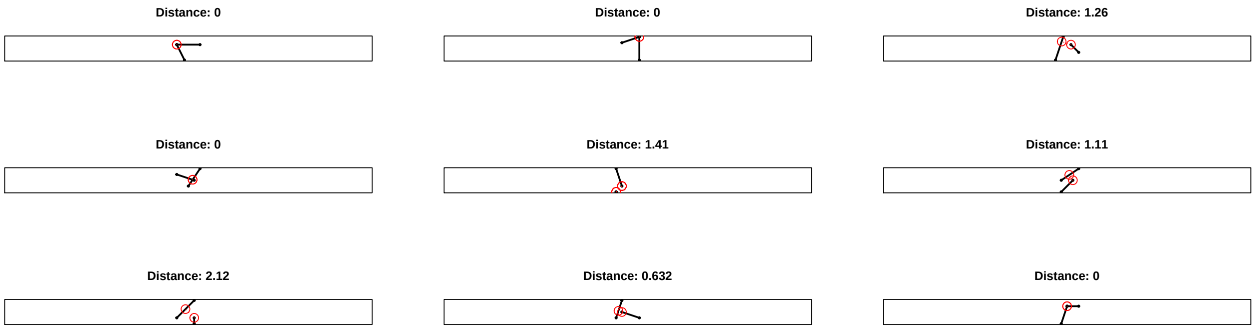
```
example(projpointonseg_a,echo=F)
```

```
example(closestpointonpolygon,echo=F)
```

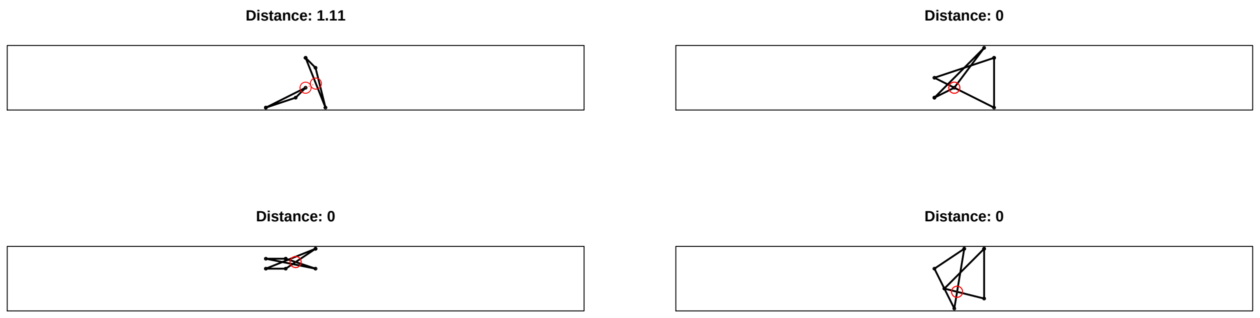


```
example(closestpointsontwosegments,echo=F)
```

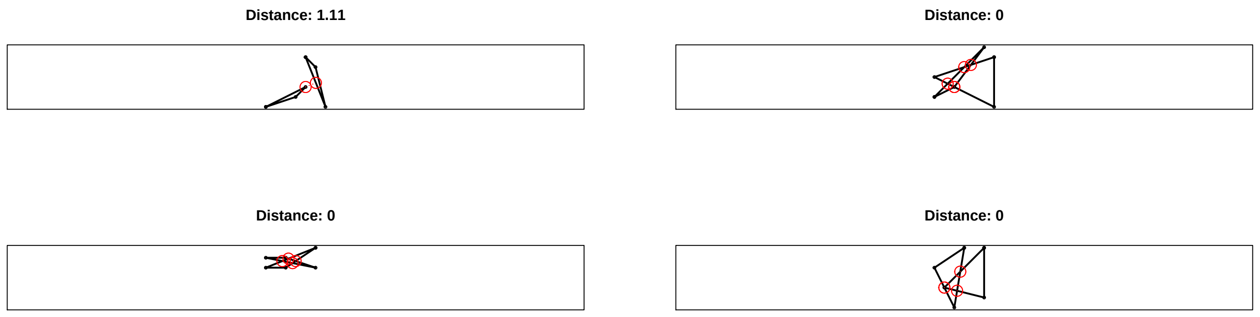
```
example(closestpointsontwosegments_n,echo=F)
```



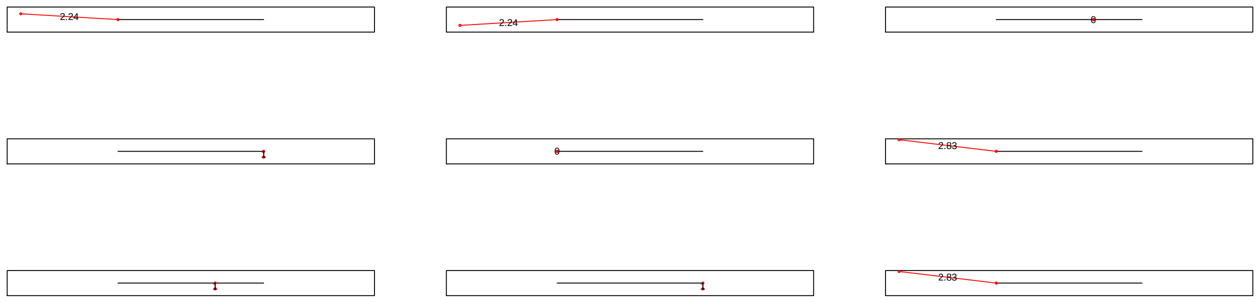
```
example(closestpointsontwopolygons,echo=F)
```



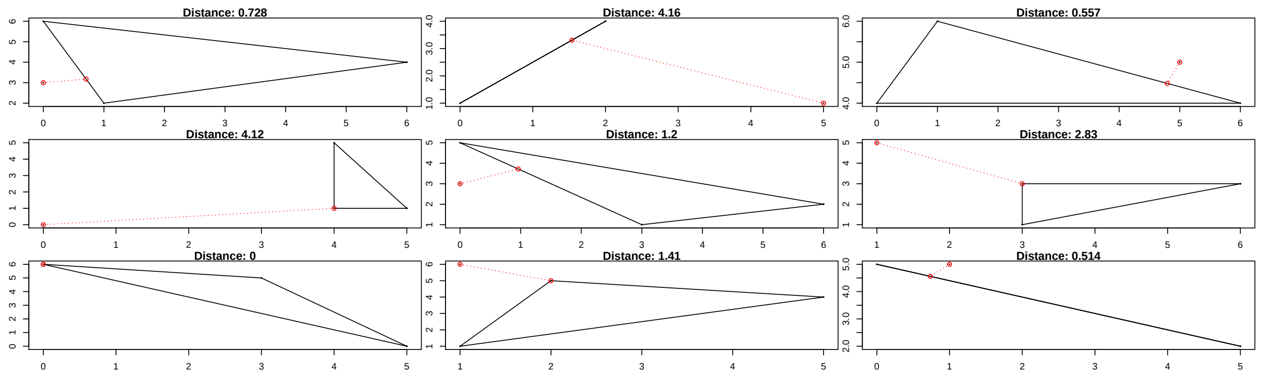
```
example(closestpointsontwopolygons_n,echo=F)
```



```
example(distpointtoseg,echo=F)
```

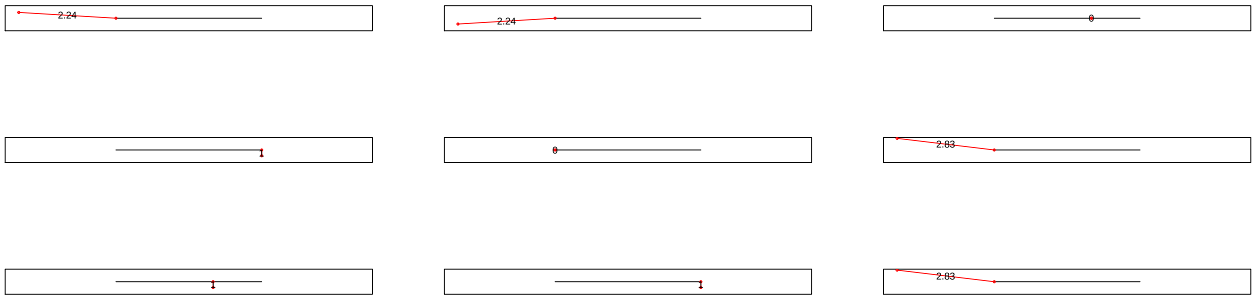


```
example(distpointtopoly,echo=F)
```

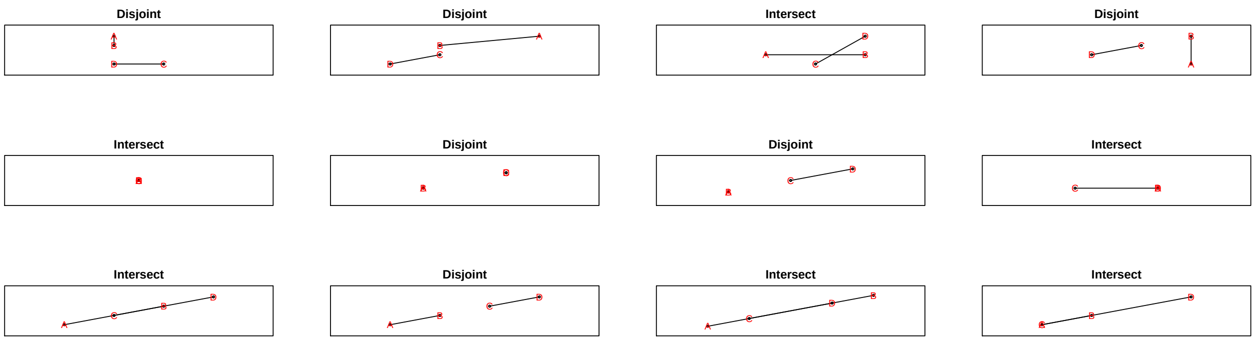


`distpointtoseg` is a function that returns the distance between a point and a segment.

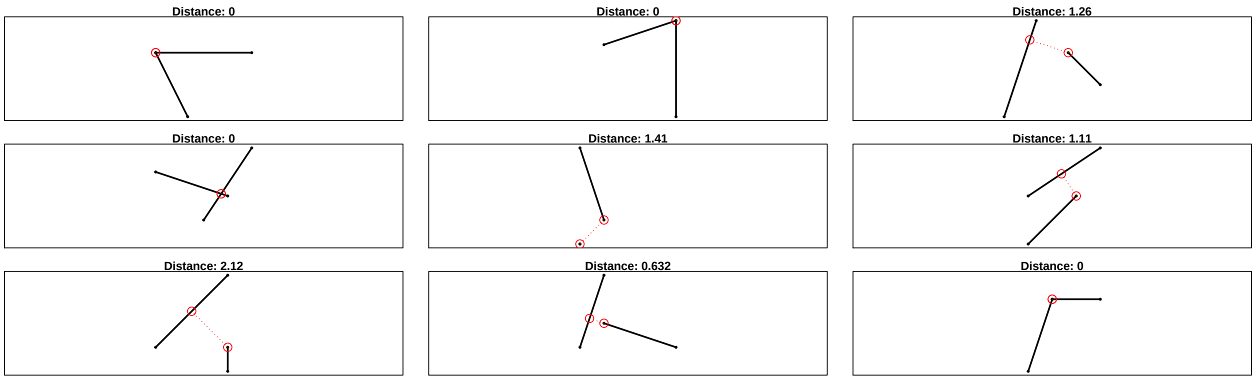
```
example(distpointtoseg,echo=F)
```



```
example(segment.intersect,echo=F)
```



```
example(distsegmenttosegment,echo=F)
```

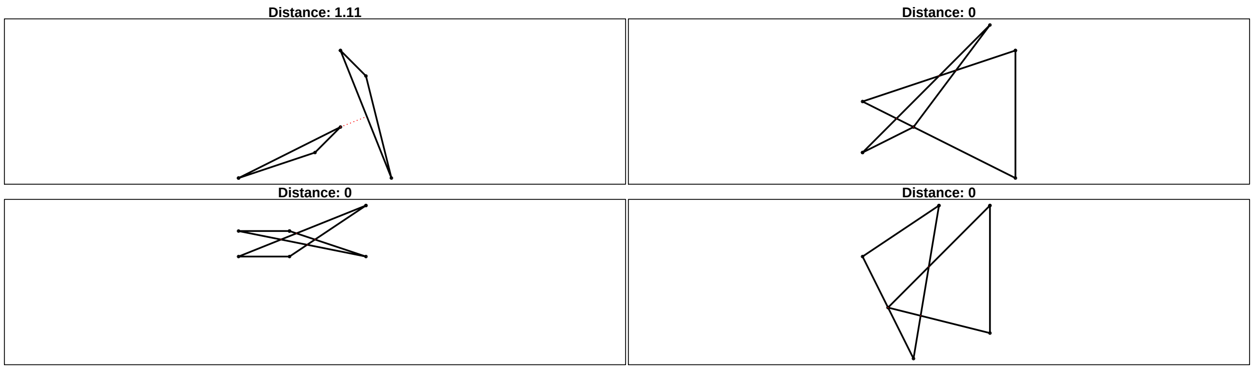


```
example(distsegmenttopoly)
```

```
##
## dstsgm> zz<-function(){
## dstsgm+ .poly<-matrix(sample(0:6,6,T),3,2)[c(1:3,1),]
## dstsgm+ s<-matrix(sample(0:6,6,T),2,2)
## dstsgm+ plot(rbind(.poly,s),xlab="",yaxt="n")
## dstsgm+ points(.poly,type="l");points(s,type="l")
## dstsgm+ points(closestpointsontwopolygons(s,.poly),lty=3,col="red")}
```

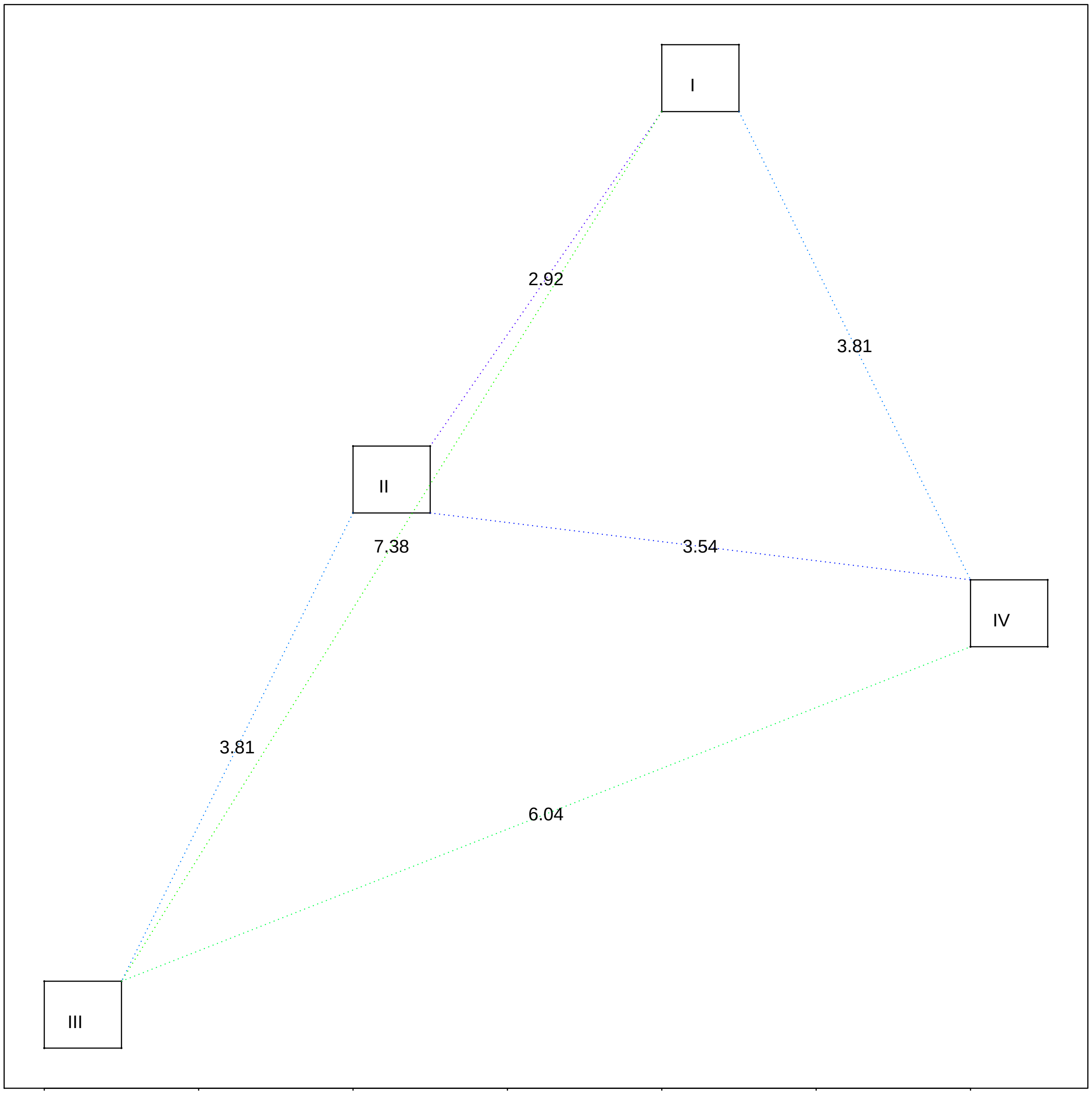
```
example(distpolytopoly,ask=F)
```

```
##
## dstply> zz<-function(){
## dstply+ poly1=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
## dstply+ poly2=matrix(sample(0:6,6,rep=T),3,2)[c(1:3,1),]
## dstply+ s<-rbind(poly1,poly2)
## dstply+ dd<-distpolytopoly(poly1,poly2)
## dstply+ plot(s,cex=.5,main=paste0("Distance: ", signif(dd,3)),asp=1,xlim=range(s),ylim=range(s),xaxt
## dstply+ points(poly1,type="l",lwd=2)
## dstply+ points(poly2,type="l",lwd=2)
## dstply+ for(cc in closestpointsontwopolygons_n(poly1,poly2)){
## dstply+ points(cc,type="l",col="red",lty=3)}}
##
## dstply> par(mfrow=c(2,2),oma=c(0,0,1,0),mar=c(0.1,0.1,1,0.1))
##
## dstply> set.seed(2);replicate(4,zz())
```

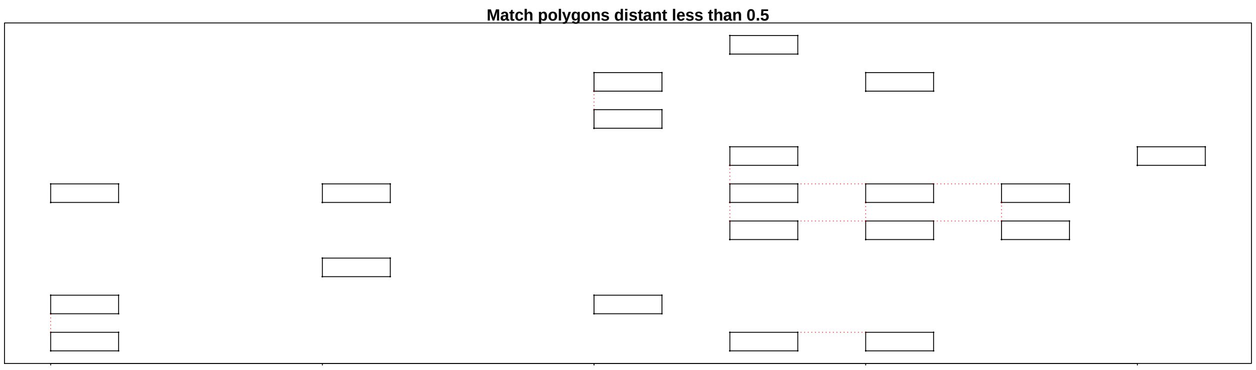


```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

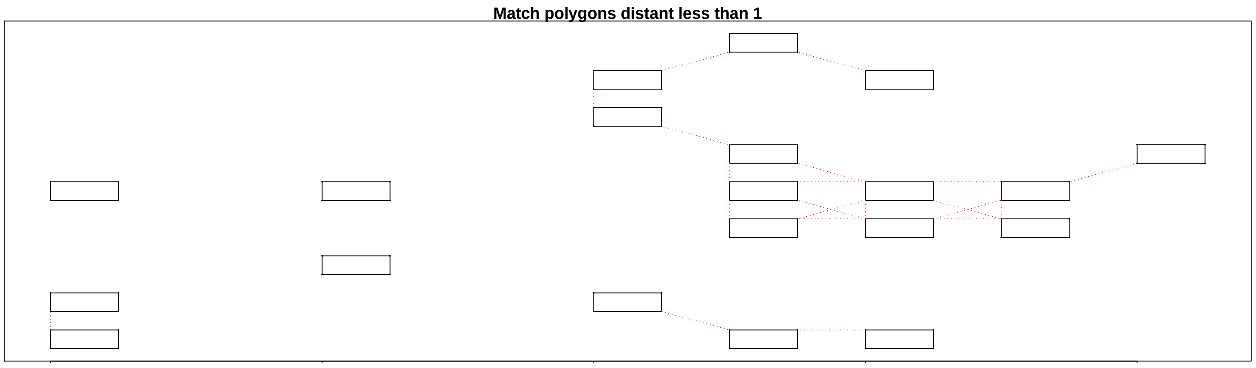
```
example(polydistmat,echo=F)
```



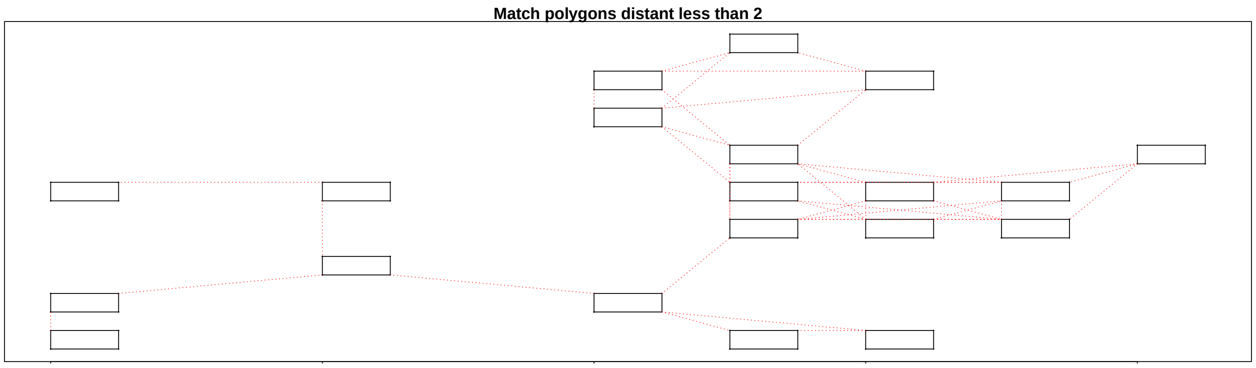
```r
example(polysmalldistmat,echo=F)
```



```
## [1] "ranges by gradient computed"
## [1] "20 polygons."
```



```
## [1] "ranges by gradient computed"
## [1] "20 polygons."
```

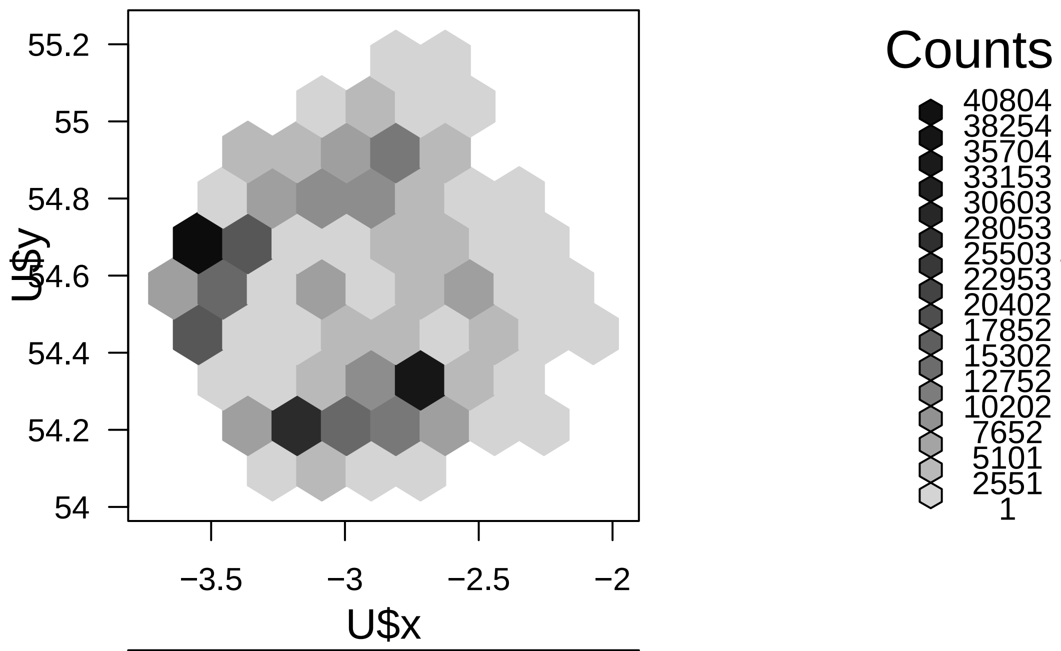


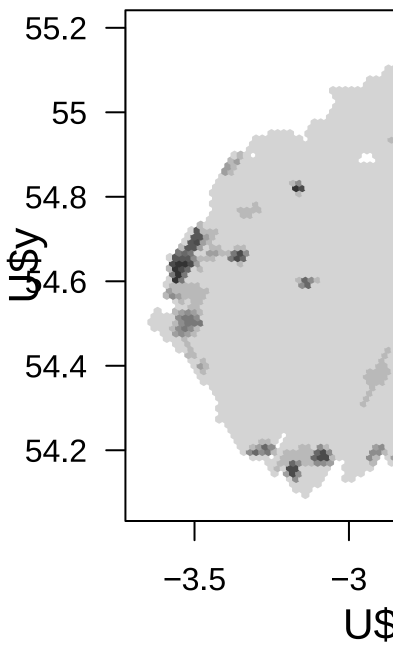
```
## [1] "ranges by gradient computed"
## [1] "20 polygons."
```

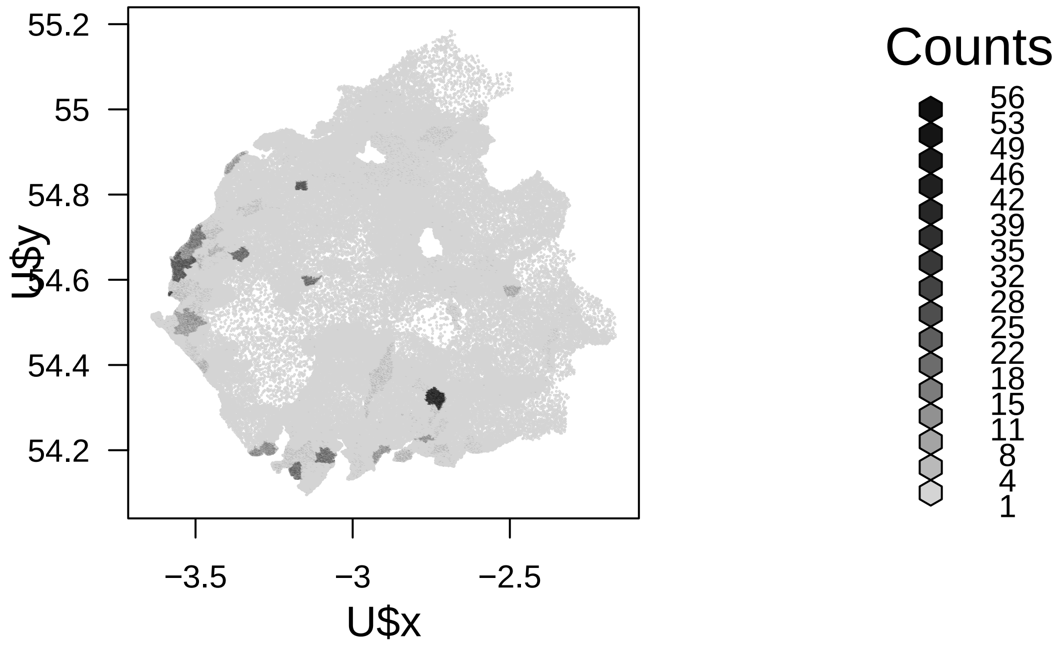
```r
example(connectedpop)
```

```r
example(extractpolygonsaslist)
```

```r
example(neighbourhoods,echo=F)
```







```r
example(dist_areas_f)
```

```
##
## dst_r_> data(U)
##
## dst_r_> dist_areas_f(U)[1:3,1:3]
##            18         67         68
## 18 0.00000000 0.03502217 0.02395041
## 67 0.03502217 0.00000000 0.01475275
## 68 0.02395041 0.01475275 0.00000000
##
## dst_r_> dist_areas_f(U,0.03)[1:3,1:3]
##             9         10         33
## 9  0.00000000 0.00000000 0.09622412
## 10 0.00000000 0.00000000 0.09622412
## 33 0.09622412 0.09622412 0.00000000
```

```r
example(newdist)
```

```
##
## newdst> data(UE,package="Strategy")
##
## newdst> delta<-.005
##
## newdst> sicks<-(1:nrow(UE))[UE$I001=="sick"]
##
## newdst> closedistances=newdist(NULL,UE,sicks)
##
## newdst> do.call(cbind,closedistances)[1:3,]
##                            ra
## [1,] 13091 13903 0.002314053
## [2,] 13118 13903 0.003925148
## [3,] 13138 13903 0.002660063
```

```r
example(updatedist)
```

```
##
## updtds> data(UE,package="Strategy")
##
## updtds> delta<-.005
##
## updtds> sicks<-(1:nrow(UE))[UE$I001=="sick"]
##
## updtds> closedistances=updatedist(NULL,UE,sicks)
##
## updtds> do.call(cbind,closedistances)[1:3,]
##                            ra
## [1,] 13091 13903 0.002314053
## [2,] 13118 13903 0.003925148
## [3,] 13138 13903 0.002660063
```