Playing with EBSpatCGAL

R. Drouilhet

FIGAL Team - LJK Grenoble

Plan

PoLiTe project

- **Origin:** EBSpat a companion package offering simulation and estimation tools for the nearest-neighbour point processes.
- **Now:** EBSpatCGAL (in fact, better called CGALSpat) is a complete code rewriting of EBSpat using
  1. the R package Rcpp as a replacement of my R package CqlsRCom based on the C API of R
  2. the very complete C++ library CGAL (Computational Geometry Algorithms Library) as a replacement of the code developed first in his PhD dissertation by Etienne Bertin.
- **Next:** R package PoLiTe (**Po**int and **Li**ne **Te**sselations) as a merging of EBSpatCGAL and LiTe (with Kiên Kiêu as main developer).

PoLiTe *project*

- **Origin:** EBSpat a companion package offering simulation and estimation tools for the nearest-neighbour point processes.
- **Now:** EBSpatCGAL (in fact, better called CGALSpat) is a complete code rewriting of EBSpat using
  1. the R package Rcpp as a replacement of my R package CqlsRCom based on the C API of R .
  2. the very complete C++ library CGAL (Computational Geometry Algorithms Library) as a replacement of the code developed first in his PhD dissertion by Etienne Bertin.
- **Next:** R package PoLiTe (**Po**int and **Li**ne **Te**sselations) as a merging of EBSpatCGAL and LiTe (with Kiên Kiêu as main developer).

PoLiTe *project*

- **Origin:** EBSpat a companion package offering simulation and estimation tools for the nearest-neighbour point processes.
- **Now:** EBSpatCGAL (in fact, better called CGALSpat) is a complete code rewriting of EBSpat using
  1. the R package Rcpp as a replacement of my R package CqlsRCom based on the C API of R .
  2. the very complete C++ library CGAL (Computational Geometry Algorithms Library) as a replacement of the code developed first in his PhD dissertion by Etienne Bertin.
- **Next:** R package PoLiTe (**Po**int and **Li**ne **Te**sselations) as a merging of EBSpatCGAL and LiTe (with Kiên Kiêu as main developer).

PoLiTe *project*

- **Origin:** EBSpat a companion package offering simulation and estimation tools for the nearest-neighbour point processes.
- **Now:** EBSpatCGAL (in fact, better called CGALSpat) is a complete code rewriting of EBSpat using
  1. the R package Rcpp as a replacement of my R package CqlsRCom based on the C API of R .
  2. the very complete C++ library CGAL (Computational Geometry Algorithms Library) as a replacement of the code developed first in his PhD dissertion by Etienne Bertin.
- **Next:** R package PoLiTe (**Po**int and **Li**ne **Te**sselations) as a merging of EBSpatCGAL and LiTe (with Kiên Kiêu as main developer).

PoLiTe *project*

- **Origin:** EBSpat a companion package offering simulation and estimation tools for the nearest-neighbour point processes.
- **Now:** EBSpatCGAL (in fact, better called CGALSpat) is a complete code rewriting of EBSpat using
  1. the R package Rcpp as a replacement of my R package CqlsRCom based on the C API of R .
  2. the very complete C++ library CGAL (Computational Geometry Algorithms Library) as a replacement of the code developed first in his PhD dissertion by Etienne Bertin.
- **Next:** R package PoLiTe (**Po**int and **Li**ne **Te**sselations) as a merging of EBSpatCGAL and LiTe (with Kiên Kiêu as main developer).

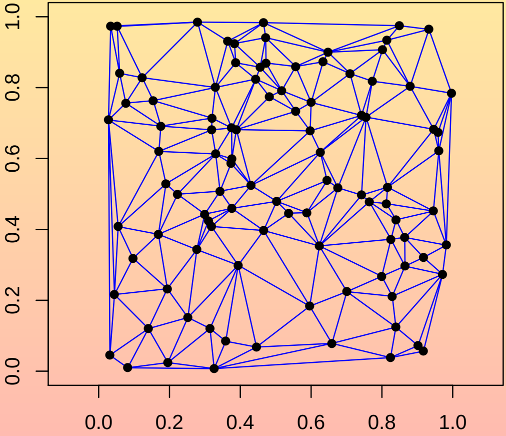
Plan

Delaunay 2D

```
> del2 <- Delaunay()
> insert(del2,x=runif(100),y=runif(100),m=rUnif(100,supp=c(1,2)))
> vertices(del2,"all")
             x           y m
1   0.37379245 0.585985234 2
2   0.31904557 0.680846427 1
3   0.25201223 0.151456890 1
4   0.38706632 0.870159549 1
5   0.76450500 0.478038448 1
 :                                    :
96  0.53653834 0.445746042 1
97  0.94627073 0.682455346 1
98  0.19538909 0.024197013 1
99  0.31864697 0.408334029 1
100 0.03392521 0.973496550 1
```

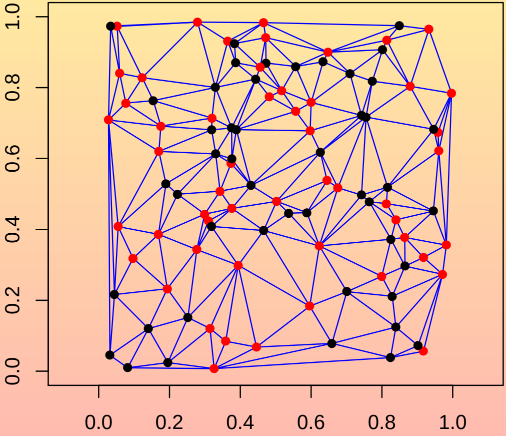
Delaunay 2D

```
> # default Delaunay plot without marks consideration
> plot(del2)
```



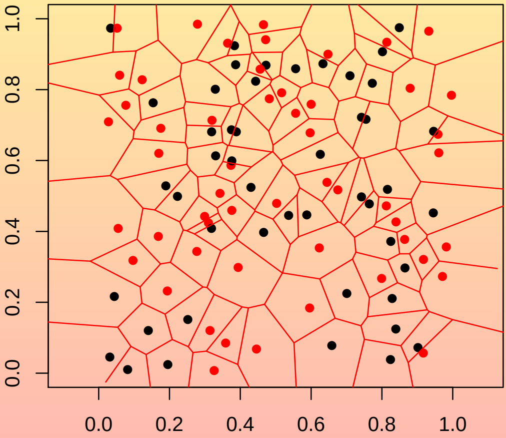
Delaunay 2D

```
> # default Delaunay plot with marks
> plot(del2,col=m)
```



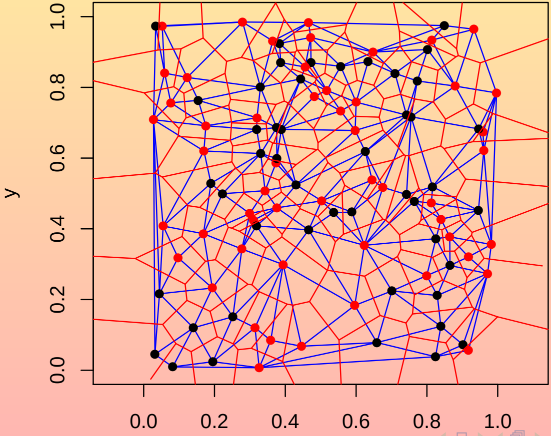
Delaunay 2D

```
> # default Voronoi plot with marks
> plot(del2,"vor",col=m)
```



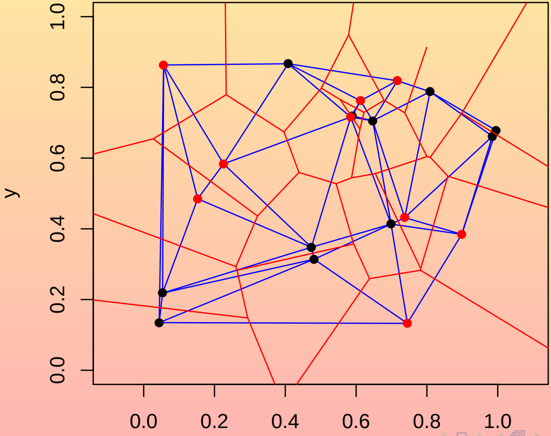
Delaunay 2D

```
> # user-defined scene
> sc <- Scene(graph=del2)
> sc %<<% window2d(xlab="x",ylab="y",main="User-defined plot!")
> sc %<<% lines(graph) %<<% points(graph,col=m) %<<% lines(graph,"vor")
> plot(sc)
```



Delaunay 2D

```
> # reuse of the previous scene
> del2bis <- Delaunay()
> insert(del2bis,x=runif(n<-20),y=runif(n),m=rUnif(n,supp=c(1,2)))
> # same scene plotted with del2bis
> plot(sc,graph=del2bis)
```

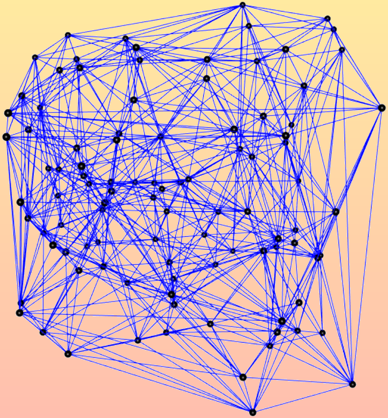


Delaunay 3D

```
> del3 <- Delaunay(3)
> insert(del3,
+    x=runif(100),y=runif(100),
+    z=runif(100),m=rUnif(100,supp=c(1,2))
+ )
> vertices(del3,"all")
             x           y          z m
1   0.9961741269 0.775198824 0.86488224 2
2   0.0520305042 0.399626697 0.34540173 2
3   0.2620854089 0.998486478 0.80300503 1
4   0.8135003112 0.149564696 0.95499060 1
5   0.0955842913 0.673535225 0.50800383 1
 :                                  :
96  0.6507950500 0.109926516 0.08267638 1
97  0.6346332736 0.045987471 0.26163885 2
98  0.1535509252 0.050600111 0.20804355 2
99  0.8973158922 0.446122207 0.78324674 2
100 0.3027747020 0.270465934 0.73376692 2
```

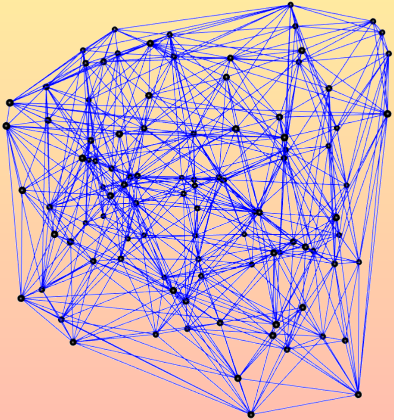
Delaunay 3D

```
> plot(del3,radius=0.01)
```



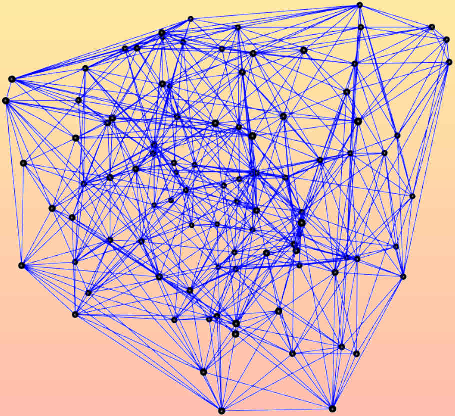
Delaunay 3D

```
> plot(del3,radius=0.01)
```



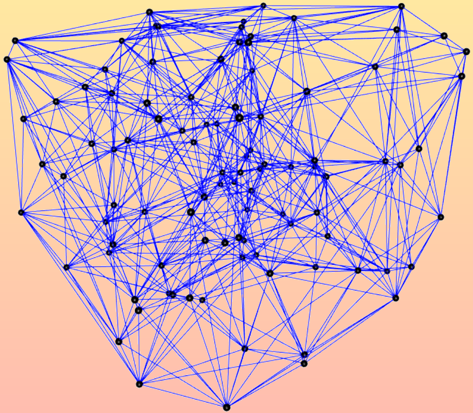
Delaunay 3D

```
> plot(del3,radius=0.01)
```



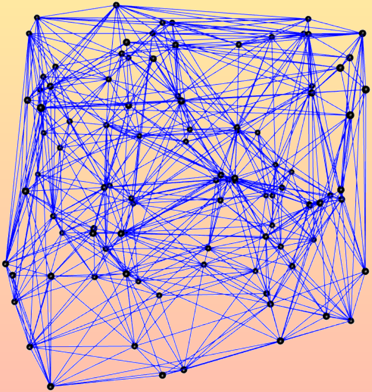
Delaunay 3D

```
> plot(del3,radius=0.01)
```



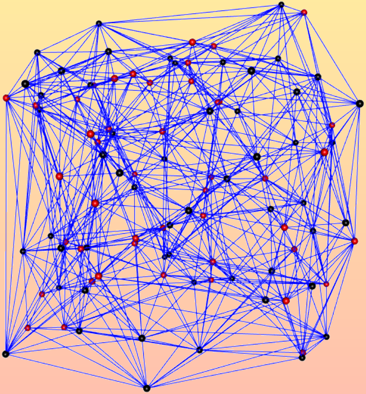
Delaunay 3D

```
> plot(del3,radius=0.01)
```



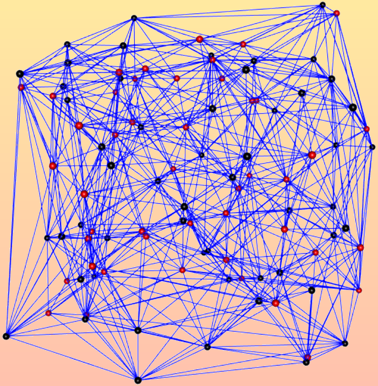
Delaunay 3D (plot with mark)

```
> plot(del3,col=m,radius=0.01)
```



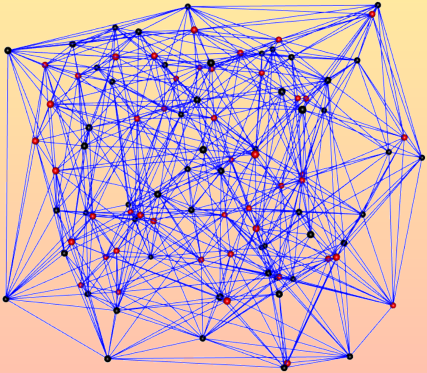
Delaunay 3D (plot with mark)

```
> plot(del3,col=m,radius=0.01)
```



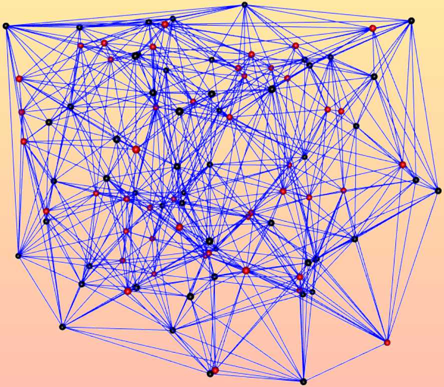
Delaunay 3D (plot with mark)

```
> plot(del3,col=m,radius=0.01)
```



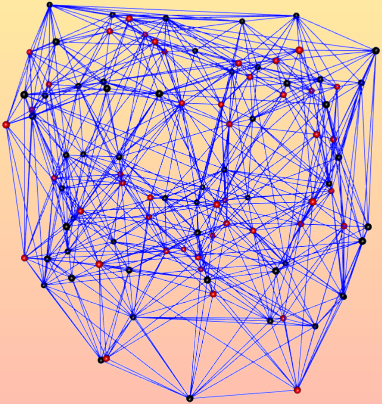
Delaunay 3D (plot with mark)

```
> plot(del3,col=m,radius=0.01)
```



Delaunay 3D (plot with mark)

```
> plot(del3,col=m,radius=0.01)
```

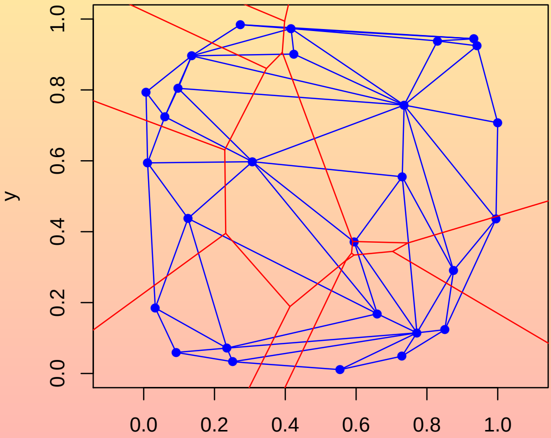


Regular graph 2D

```
> reg2 <- Regular()
> insert(reg2,x=runif(100),y=runif(100),w=runif(100))
> vertices(reg2)
             [,1]       [,2]
 [1,] 0.307228523 0.59749540
 [2,] 0.932679536 0.94436279
 [3,] 0.850657211 0.12352350
 [4,] 0.730368539 0.55508497
 [5,] 0.416309413 0.97319916
 :                                :
[24,] 0.554761716 0.01080638
[25,] 0.995206009 0.43567380
[26,] 0.273091584 0.98411324
[27,] 0.006656302 0.79359428
[28,] 0.875257040 0.29055158
```

Regular graph 2D

```
> sc <- Scene()
> sc %<<% window2d(xlab="x",ylab="y",main="Regular and dual graphs")
> sc %<<% lines(graph) %<<% points(graph) %<<% lines(graph,"vor")
> plot(sc,graph=reg2)
```

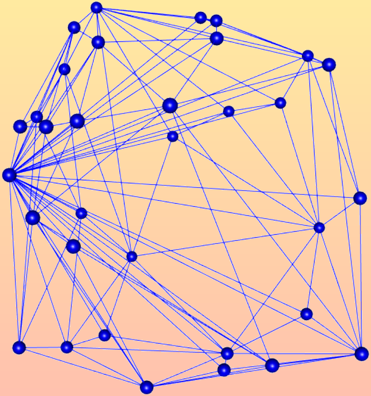


Regular graph 3D

```
> reg3 <- Regular(3)
> insert(reg3,
+     x=runif(100),y=runif(100),
+     z=runif(100),w=runif(100)
+ )
> vertices(reg3)
             [,1]         [,2]       [,3]
 [1,] 0.66739553 0.9386865757 0.06735985
 [2,] 0.08513267 0.1697152695 0.86458024
 [3,] 0.92423853 0.0159255203 0.77415013
 [4,] 0.98499313 0.0002473921 0.66801125
 [5,] 0.04380360 0.1667674046 0.38254712
 :                                    :
[28,] 0.44919900 0.1946863390 0.94983455
[29,] 0.06152015 0.0677579972 0.73685371
[30,] 0.04773288 0.7621858553 0.54797599
[31,] 0.87561551 0.9772679964 0.94096146
[32,] 0.72143524 0.3870179157 0.47274896
```

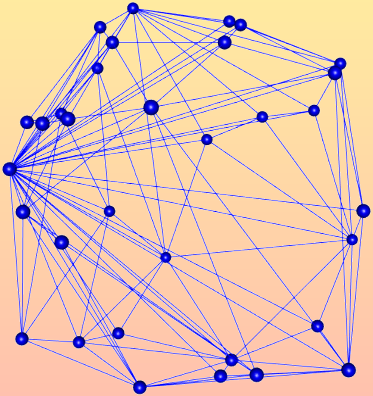
Regular graph 3D

```
> plot(sc3,gr=reg3)
```



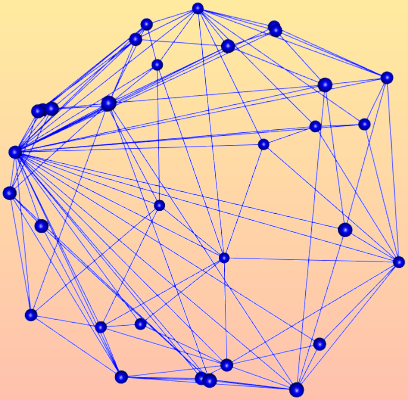
Regular graph 3D

```
> plot(sc3,gr=reg3)
```



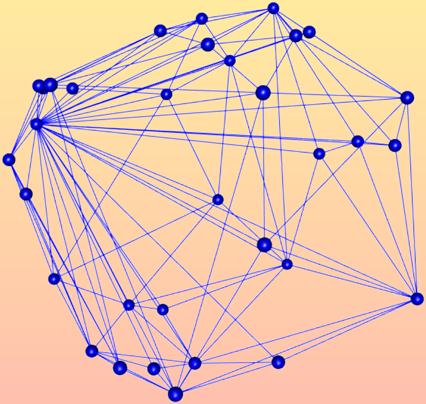
Regular graph 3D

```
> plot(sc3,gr=reg3)
```



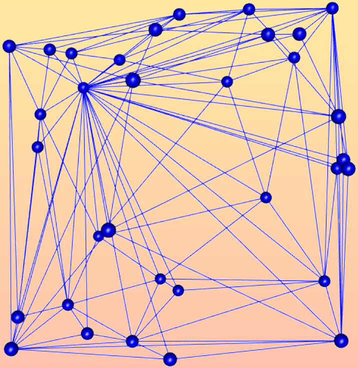
Regular graph 3D

```
> plot(sc3,gr=reg3)
```



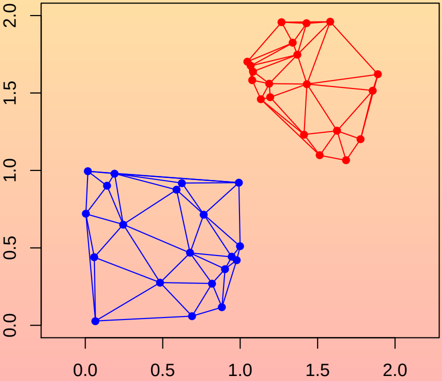
Regular graph 3D

```
> plot(sc3,gr=reg3)
```



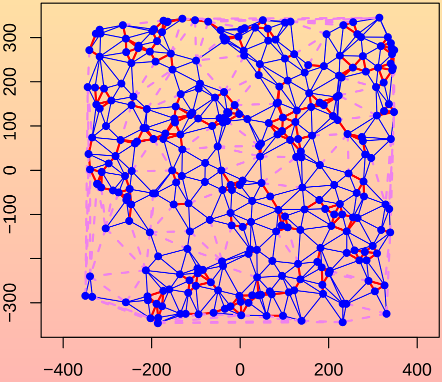
Scene with many actors

```
> del2 <- Delaunay();del2bis<-Delaunay()
> insert(del2,x=runif(n<-20),y=runif(n))
> insert(del2bis,x=runif(n,1,2),y=runif(n,1,2))
> sc2 <- Scene(gr=del2,gr2=del2bis)
> sc2 %<<% window2d(c(0,2),c(0,2),xlab="",ylab="")
> sc2 %<<% lines(gr,col="blue") %<<% points(gr,col="blue")
> sc2 %<<% lines(gr2,col="red") %<<% points(gr2,col="red");plot(sc2)
```



Scene with many colors

```
> del2 <- Delaunay()
> insert(del2,x=runif(n<-300,-350,350),y=runif(n,-350,350))
> sc2g <- Scene(gr=del2) %<<% window2d(c(-350,350),c(-350,350))
> sc2g %<<% lines(gr,when=40<length & length <= 80) %<<%
+ lines(gr,col="red",lwd=2,when= length <= 40)   %<<%
+ lines(gr,col="violet",lty=2,lwd=2,when=80<length) %<<%
+ points(gr);plot(sc2g)
```



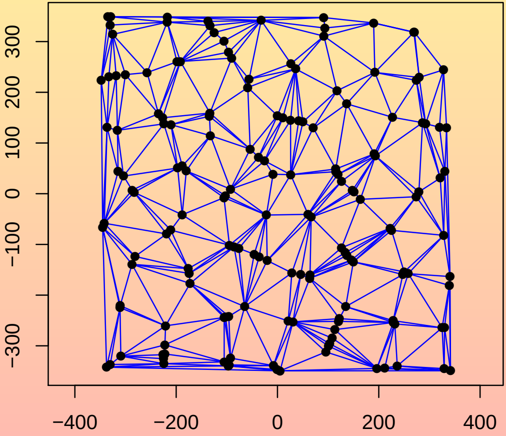
Plan

Simulation 2D

```
> # Delaunay
> del2 <- Delaunay();del2bis <- Delaunay()
> # Gibbs simulation
> gd2 <- SimGibbs(
+   del2 ~ 2 + Del2(th[1]*(l<=20)+th[2]*(20<l & l<=80),th=c(2,4)),
+   domain=Domain(c(-350,-350),c(350,350))
+ )
> # marked one
> del2m <- Delaunay()
> gd2m <- SimGibbs(
+   del2m ~ 2 + Del2(th[1]*(l<=20) + th[2]*(20<l & l<=80)
+         * abs(v[[1]]$m-v[[2]]$m), th=c(2,4))| m ~ Unif(supp=c(1,2))
+ )
```

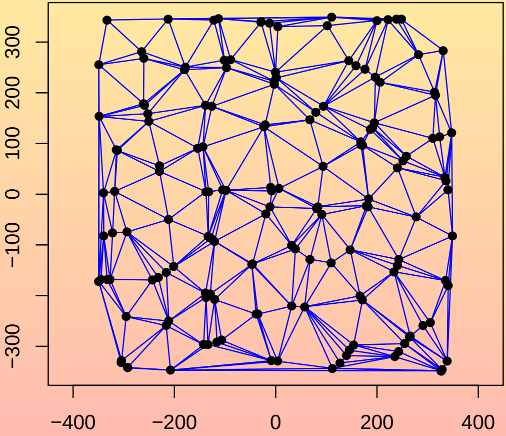
Simulation 2D

```
> # run the simulator and plot the resulted Delaunay graph
> run(gd2);plot(del2)
```



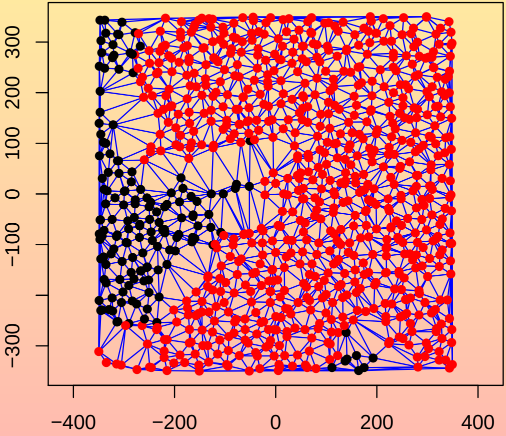
Simulation 2D

```
> # one can run the simulator with another Delaunay graph
> run(gd2,current=del2bis);plot(del2bis)
```



Simulation 2D

```
> # run the simulator with the marked Delaunay graph
> run(gd2m);plot(del2m,col=m)
```

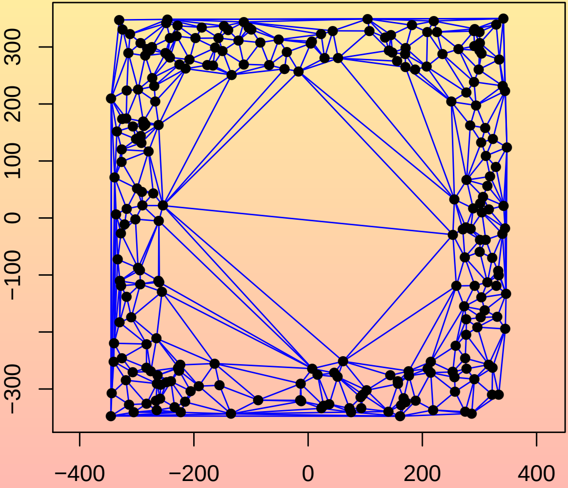


Simulation 2D

```
> # inside domain
> domIn <- Domain(c(-250,-250),c(250,250))
> #take a boundary of 1
> del2m1 <- Delaunay()
> insert(del2m1,x=runif(n<-500,-350,350),y=runif(n,-350,350),m=1)
> delete(del2m1,inside=domIn)
> #take a boundary of 2
> del2m2 <- Delaunay()
> insert(del2m2,x=runif(n<-500,-350,350),y=runif(n,-350,350),m=2)
> delete(del2m2,inside=domIn)
```

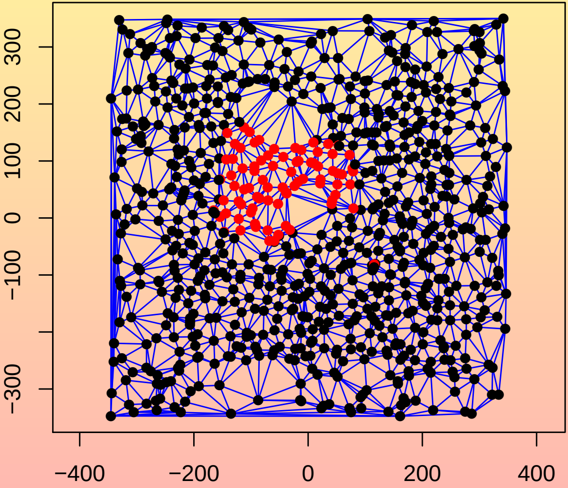
Simulation 2D

```
> plot(del2m1,col=m)
```



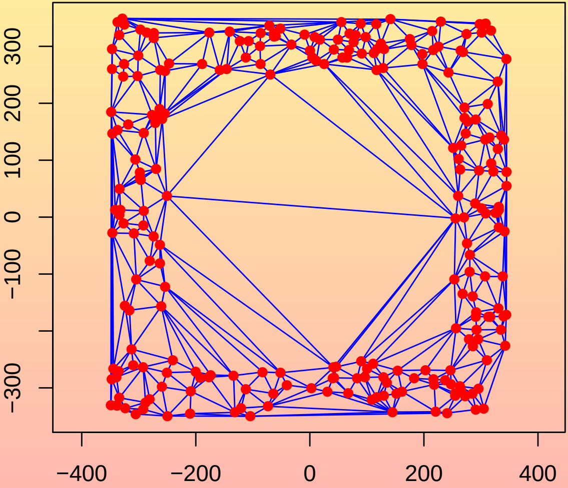
Simulation 2D

```
> run(gd2m,current=del2m1,domain=domIn);plot(del2m1,col=m)
```



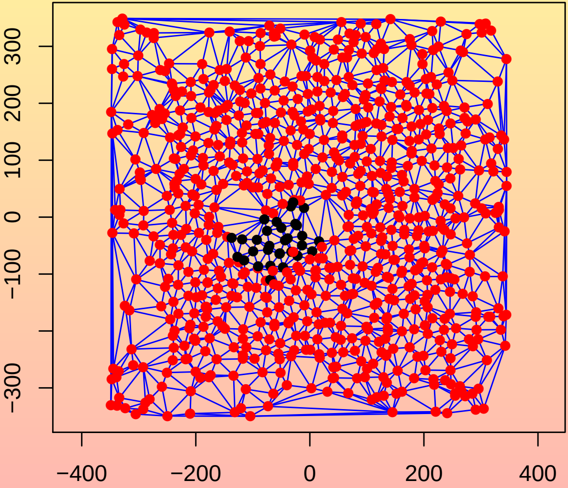
Simulation 2D

```
> plot(del2m2,col=m)
```



Simulation 2D

```
> run(gd2m,current=del2m2,domain=domIn);plot(del2m2,col=m)
```

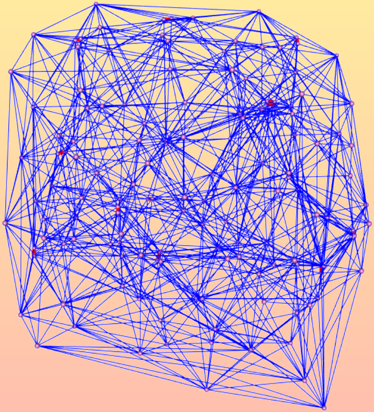


Simulation 3D (Yes! First time!)

```
> # Delaunay
> del3 <- Delaunay(3)
> insert(del3,matrix(runif(300,-350,350),ncol=3))
> # Gibbs simulation
> gd3 <- SimGibbs(
+   del3 ~ 14 + Del2(th[1]*(l<=20)+th[2]*(20<l & l<=80),th=c(-2,10)),
+   domain=Domain(c(-350,-350,-350),c(350,350,350))
+ )
> run(gd3)
> # scene 3D
> (sc3 <- Scene()) %<<%
+ window3d(gd3,windowRect=c(0,0,800,800)) %<<%
+ points(gr,col="violet",radius=5) %<<%
+ lines(gr,col="red",lwd=5,when= length <= 20)  %<<%
+ lines(gr,lwd=5,col="green",when=20<length & length <= 80) %<<%
+ lines(gr,col="blue",when=80<length)
```

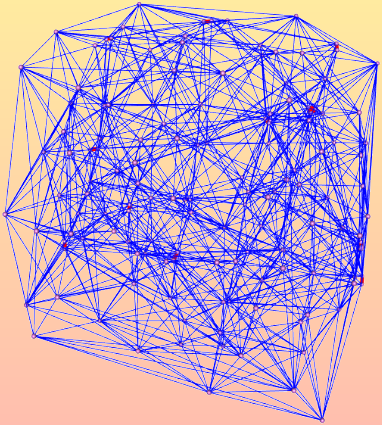
Simulation 3D

```
> plot(sc3,gr=del3)
```



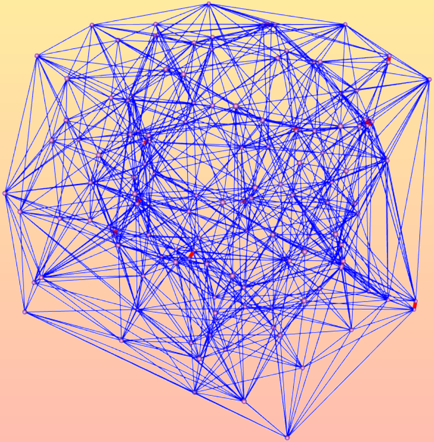
Simulation 3D

```
> plot(sc3,gr=del3)
```



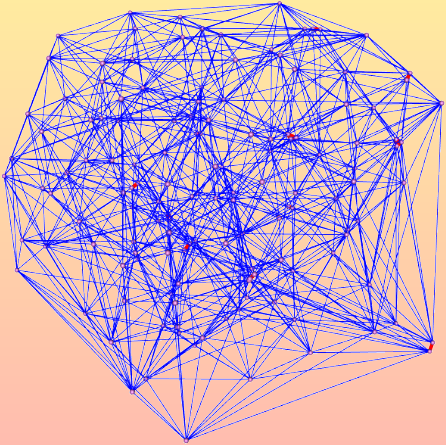
Simulation 3D

```
> plot(sc3,gr=del3)
```



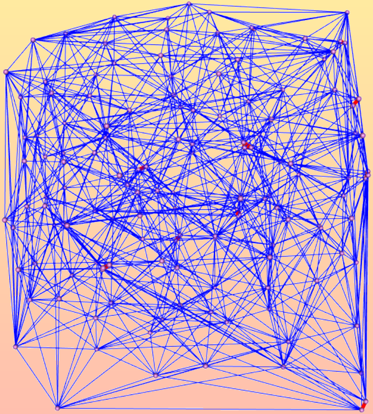
Simulation 3D

```
> plot(sc3,gr=del3)
```



Simulation 3D

```
> plot(sc3,gr=del3)
```



Plan

Innovations and Residuals

- GNZ equation: $\mathbf{E}\left(h(0,\Phi;\theta)\, e^{-V(0|\Phi;\theta^\star)}\right) = \mathbf{E}\left(h(0,\Phi\setminus 0;\theta)\right)$
- $h-$innovations:

$$\int_\Lambda h(x,\varphi;\theta^\star)\, e^{-V(x|\varphi;\theta^\star)} dx - \sum_{x\in\varphi_\Lambda} h(x,\varphi\setminus x;\theta^\star)$$

- $h-$residuals:

$$\int_\Lambda h\left(x,\varphi;\widehat{\theta}\right) e^{-V\left(x|\varphi;\widehat{\theta}\right)} dx - \sum_{x\in\varphi_\Lambda} h\left(x,\varphi\setminus x;\widehat{\theta}\right)$$

- inverse $h-$residuals:

$$\int_\Lambda h\left(x,\varphi;\widehat{\theta}\right) dx - \sum_{x\in\varphi_\Lambda} h\left(x,\varphi\setminus x;\widehat{\theta}\right) e^{V\left(x|\varphi\setminus x;\widehat{\theta}\right)}$$

Innovations and Residuals

- GNZ equation: $\mathbf{E}\left(h(0,\Phi;\theta)\, e^{-V(0|\Phi;\theta^\star)}\right) = \mathbf{E}\left(h(0,\Phi\setminus 0;\theta)\right)$
- $h-$innovations:

$$\int_\Lambda h(x,\varphi;\theta^\star)\, e^{-V(x|\varphi;\theta^\star)} dx - \sum_{x\in\varphi_\Lambda} h(x,\varphi\setminus x;\theta^\star)$$

- $h-$residuals:

$$\int_\Lambda h\left(x,\varphi;\widehat{\theta}\right) e^{-V\left(x|\varphi;\widehat{\theta}\right)} dx - \sum_{x\in\varphi_\Lambda} h\left(x,\varphi\setminus x;\widehat{\theta}\right)$$

- inverse $h-$residuals:

$$\int_\Lambda h\left(x,\varphi;\widehat{\theta}\right) dx - \sum_{x\in\varphi_\Lambda} h\left(x,\varphi\setminus x;\widehat{\theta}\right) e^{V\left(x|\varphi\setminus x;\widehat{\theta}\right)}$$

Innovations and Residuals

- GNZ equation: $\mathbf{E}\left(h\left(0,\Phi;\theta\right)e^{-V\left(0|\Phi;\theta^{\star}\right)}\right)=\mathbf{E}\left(h\left(0,\Phi\setminus 0;\theta\right)\right)$
- $h-$innovations:

$$\int_{\Lambda} h\left(x,\varphi;\theta^{\star}\right)e^{-V\left(x|\varphi;\theta^{\star}\right)}dx-\sum_{x\in\varphi_{\Lambda}} h\left(x,\varphi\setminus x;\theta^{\star}\right)$$

- $h-$residuals:

$$\int_{\Lambda} h\left(x,\varphi;\widehat{\theta}\right)e^{-V\left(x|\varphi;\widehat{\theta}\right)}dx-\sum_{x\in\varphi_{\Lambda}} h\left(x,\varphi\setminus x;\widehat{\theta}\right)$$

- inverse $h-$residuals:

$$\int_{\Lambda} h\left(x,\varphi;\widehat{\theta}\right)dx-\sum_{x\in\varphi_{\Lambda}} h\left(x,\varphi\setminus x;\widehat{\theta}\right)e^{V\left(x|\varphi\setminus x;\widehat{\theta}\right)}$$

Innovations and Residuals

- GNZ equation: $\mathbf{E}\left(h\left(0,\Phi;\theta\right)e^{-V\left(0|\Phi;\theta^{\star}\right)}\right)=\mathbf{E}\left(h\left(0,\Phi\setminus 0;\theta\right)\right)$
- $h-$innovations:

$$\int_{\Lambda} h\left(x,\varphi;\theta^{\star}\right)e^{-V\left(x|\varphi;\theta^{\star}\right)}dx-\sum_{x\in\varphi_{\Lambda}} h\left(x,\varphi\setminus x;\theta^{\star}\right)$$

- $h-$residuals:

$$\int_{\Lambda} h\left(x,\varphi;\widehat{\theta}\right)e^{-V\left(x|\varphi;\widehat{\theta}\right)}dx-\sum_{x\in\varphi_{\Lambda}} h\left(x,\varphi\setminus x;\widehat{\theta}\right)$$

- inverse $h-$residuals:

$$\int_{\Lambda} h\left(x,\varphi;\widehat{\theta}\right)dx-\sum_{x\in\varphi_{\Lambda}} h\left(x,\varphi\setminus x;\widehat{\theta}\right)e^{V\left(x|\varphi\setminus x;\widehat{\theta}\right)}$$

GNZ Cache

```
> gnz <- GNZCache(
+   del2∼Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)) ,
+   1,del2(l<=20), del2(20<l & l<=80),
+   runs=10000L,
+   domain=Domain(c(-250,-250),c(250,250))
+ )
> run(gnz,Single=2,Th=c(2,4))
Please be patient: update of caches -> done!
$first
[1]  0.0003564326  0.0005583502 -0.0001283583
$second
[1]  0.000292  0.000380 -0.000028
```

Innovations

```
> res <- Resid(
+   del2~Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)) ,
+   1,del2(l<=20), del2(20<l & l<=80),
+   runs=10000L,
+   domain=Domain(c(-250,-250),c(250,250))
+ )
> run(res,Single=2,Th=c(2,4))
Please be patient: update of caches -> done!
[1]  6.023250e-05  1.046913e-04 -3.768942e-05
```

Innovations

```
> resid <- Resid(
+   del2~Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)) ,
+   1,del2(l<=20), del2(20<l & l<=80),
+   all2(range=100|l<=20),
+   all2(range=100|20<l & l<80),
+   del3(ta),
+   runs=10000L,
+   domain=Domain(c(-250,-250),c(250,250))
+ )
> run(resid,Single=2,Th=c(2,4))
Please be patient: update of caches -> done!
[1]  4.076217e-05  8.713605e-05 -2.409239e-05  1.092030e-04
[5]  3.228932e-04 -1.649223e-02
```

Plan

Pseudo-Likelihood 2D

```
> pseudo <- Pseudo(del2~Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)),
+          runs=10000L,
+          domain=Domain(c(-250,-250),c(250,250)),
+          expo=TRUE
+ )
> run(pseudo,Single=0,Th=c(0,0))
Please be patient: update of caches -> done!
$par
  Single      Th1      Th2
1.543463 2.364175 3.864422
$value
[1] 0.001532144
$counts
function gradient
       1        1
$convergence
[1] 0
$message
NULL
$Single
[1] 1.543463
[[2]]
[[2]]$Th
[1] 2.364175 3.864422
```

Pseudo-Likelihood 3D

```
> pseudo3 <- Pseudo(del3~Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)),
+   runs=10000L,
+   domain=Domain(c(-250,-250,-250),c(250,250,250)),
+   expo=TRUE
+ )
>
NULL
> run(pseudo3,Single=0,Th=c(0,0))
Please be patient: update of caches -> done!
$par
   Single       Th1       Th2
13.992761 -1.465786 11.862158
$value
[1] 5.874762e-06
$counts
function gradient
       1        1
$convergence
[1] 0
$message
NULL
$Single
[1] 13.99276
[[2]]
[[2]]$Th
```

Takacs-Fiksel 2D (inverse)

```
> tkinv <- TKInverse(del2~Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)),
+         runs=10000L,
+         domain=Domain(c(-250,-250),c(250,250))
+ )
> run(tkinv,Single=0,Th=c(0,0))
Please be patient: update of caches -> done!
$par
   Single       Th1       Th2
-8.084966 -1.224016 11.124512
$value
[1] 3.123698
$counts
function gradient
     303      101
$convergence
[1] 1
$message
NULL
$Single
[1] -8.084966
[[2]]
[[2]]$Th
[1] -1.224016 11.124512
```

Takacs-Fiksel 3D (inverse)

```
> tkinv3 <- TKInverse(del3~Del2(Th[1]*(l<=20)+Th[2]*(20<l & l<=80)),
+   runs=10000L,
+   domain=Domain(c(-250,-250,-250),c(250,250,250))
+ )
>
NULL
> run(tkinv3,Single=0,Th=c(0,0))
Please be patient: update of caches -> done!
$par
   Single       Th1       Th2
14.846436 -2.168812 -7.611145
$value
[1] 0.6575588
$counts
function gradient
     201      101
$convergence
[1] 1
$message
NULL
$Single
[1] 14.84644
[[2]]
[[2]]$Th
[1] -2.168812 -7.611145
```

What I would like to explore with this package:

- use of innovations to check wheither the result a Gibbs Markov Chain seems to be acceptable.
- make a lot of experiments in 3D to go through the proof of existence of Gibbs Delaunay model in $\mathbb{R}^3$.
- Gibbs model based on regular graphs known as weighted Delaunay triangulations (dual of Laguerre power diagram).