Spatial Statistics

Adrian Baddeley
www.maths.uwa.edu.au/˜adrian
CSIRO and University of Western Australia

ASC Workshop
Computing with R
2008

Types of spatial data

Types of spatial data

Three basic types of spatial data:

Types of spatial data

Three basic types of spatial data:

- geostatistical

Types of spatial data

Three basic types of spatial data:

- geostatistical
- regional

Types of spatial data

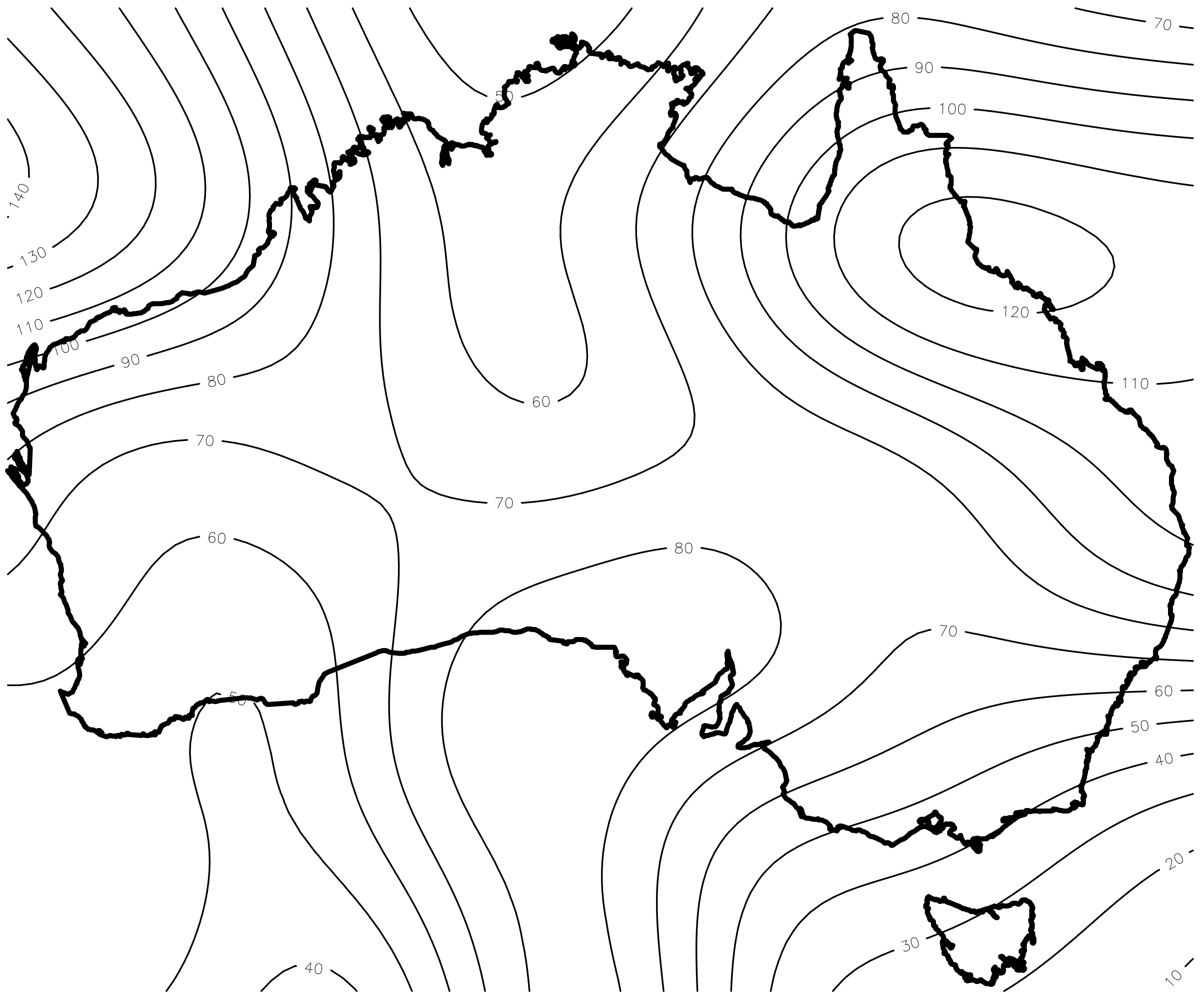
Three basic types of spatial data:

- geostatistical
- regional
- point pattern

Geostatistical data

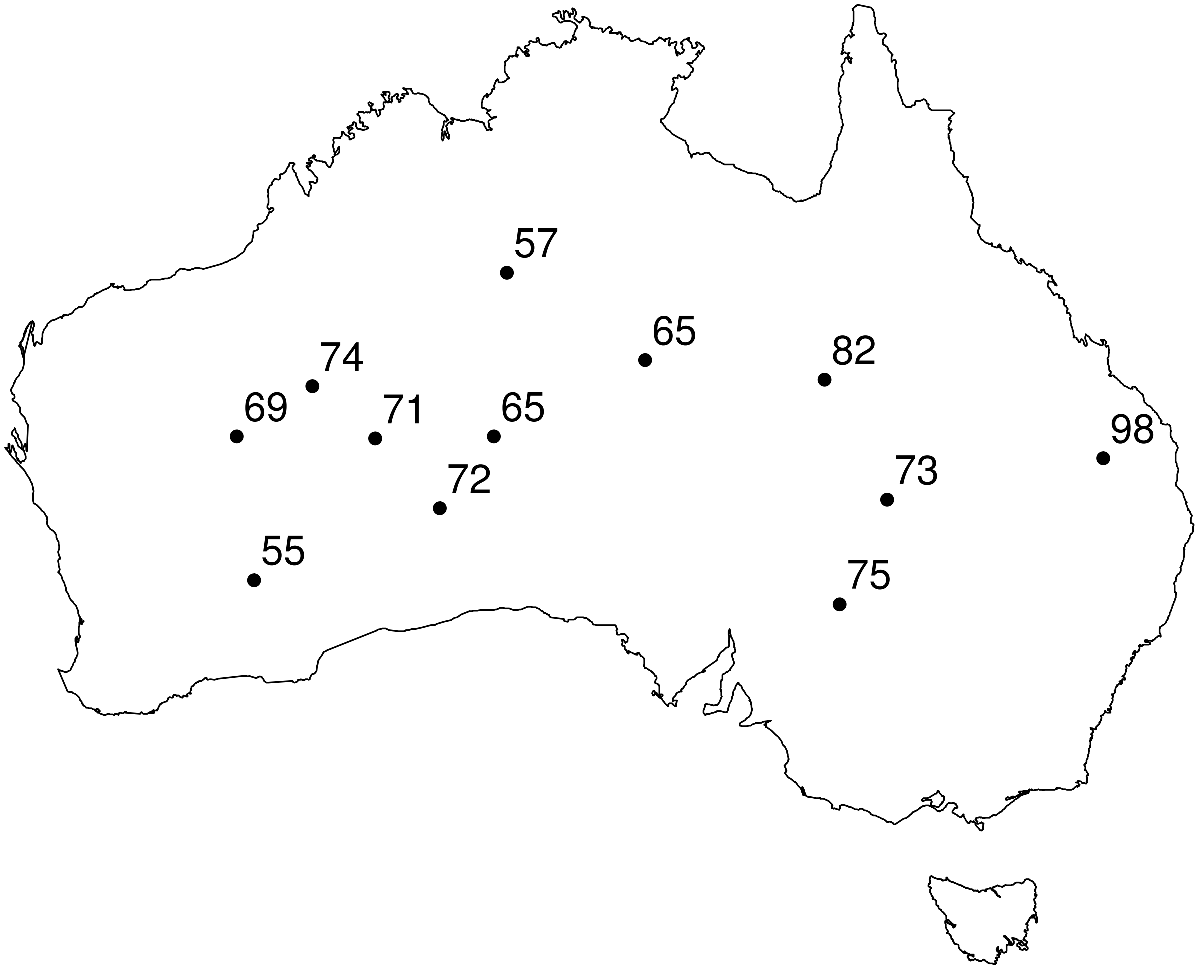
GEOSTATISTICAL DATA:

The quantity of interest has a value at any location, . . .



Geostatistical data

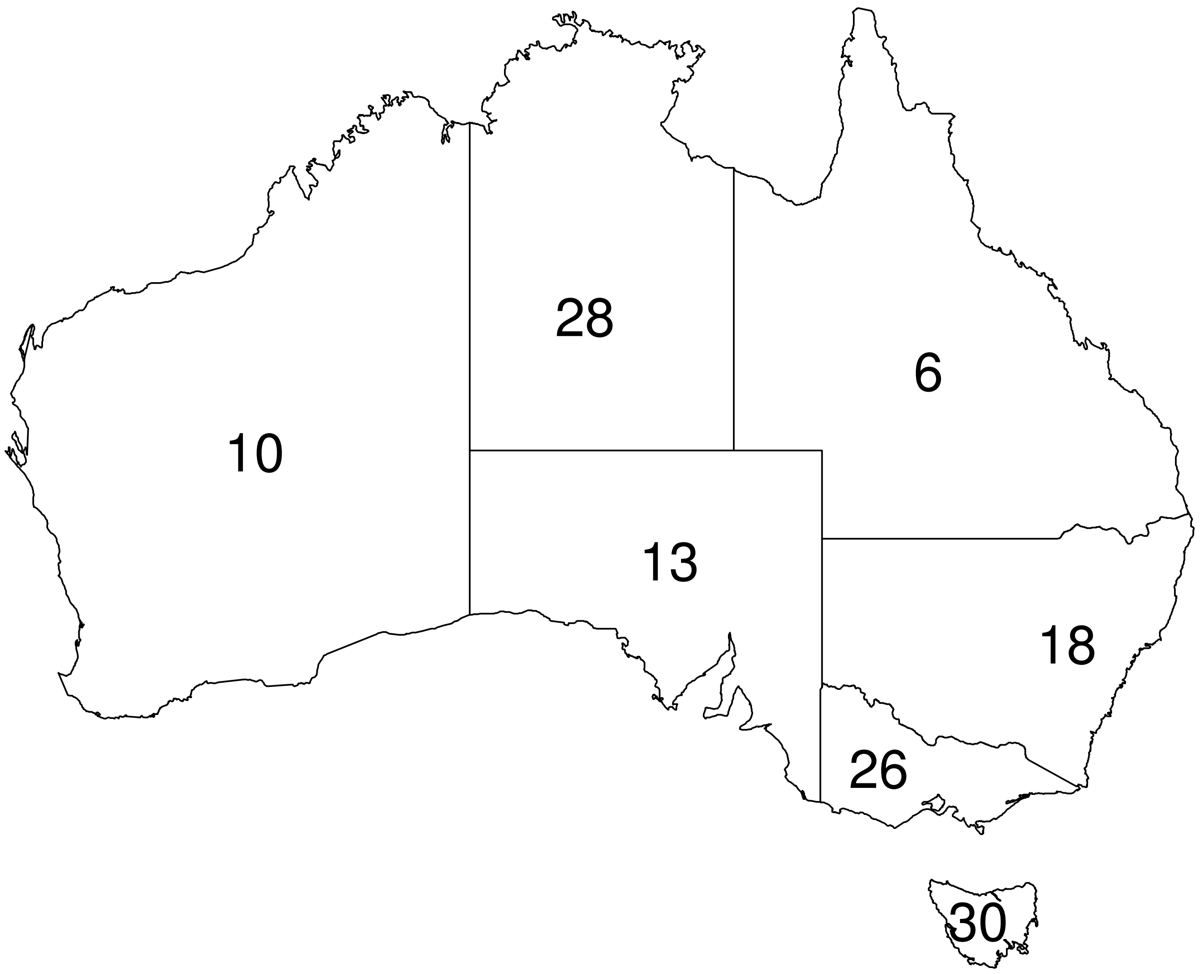
. . . but we only measure the quantity at certain sites. These values are our data.



Regional data

REGIONAL DATA:

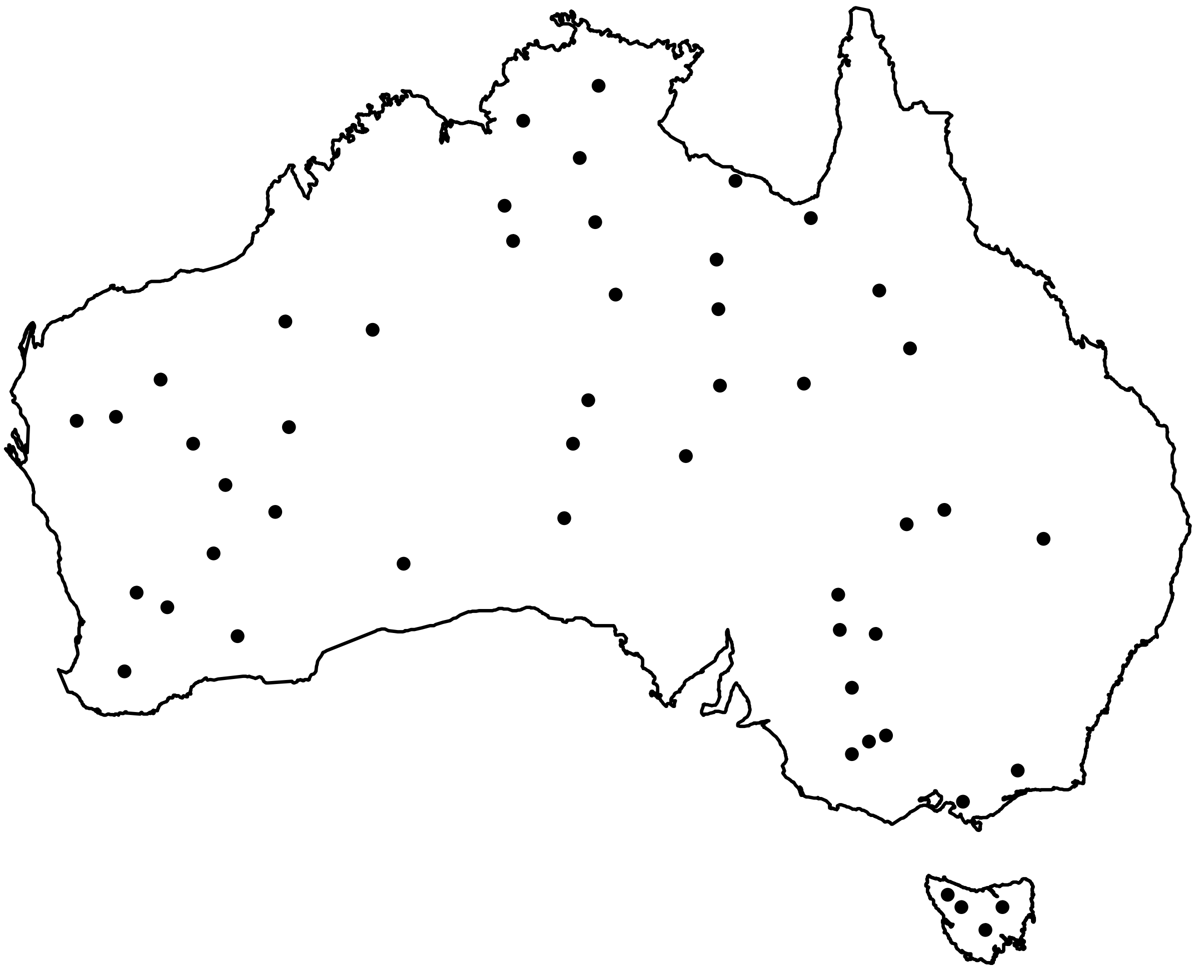
The quantity of interest is only defined for regions. It is measured/reported for certain *fixed* regions.



Point pattern data

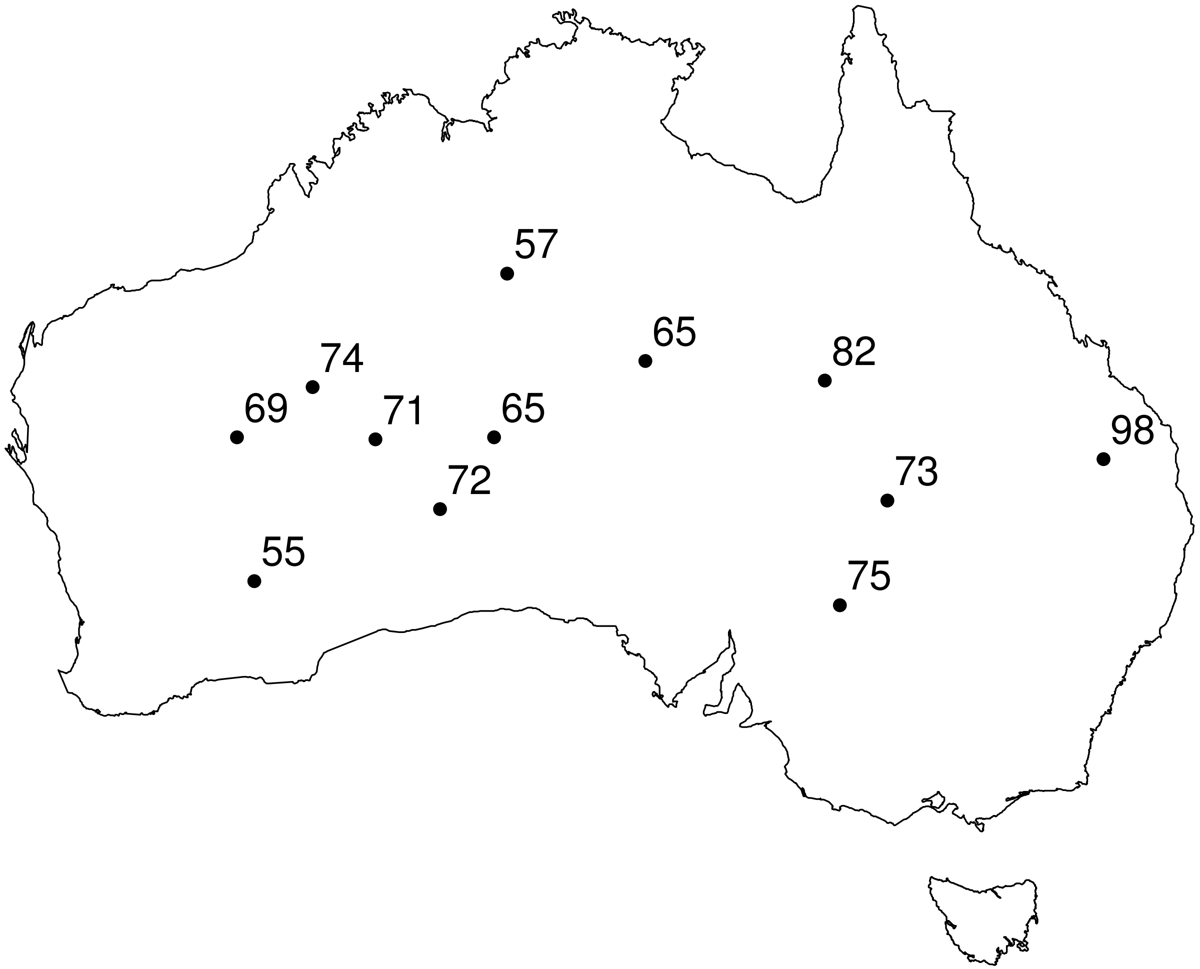
POINT PATTERN DATA:

The main interest is in the *locations* of all occurrences of some event (e.g. tree deaths, meteorite impacts, robberies). Exact locations are recorded.

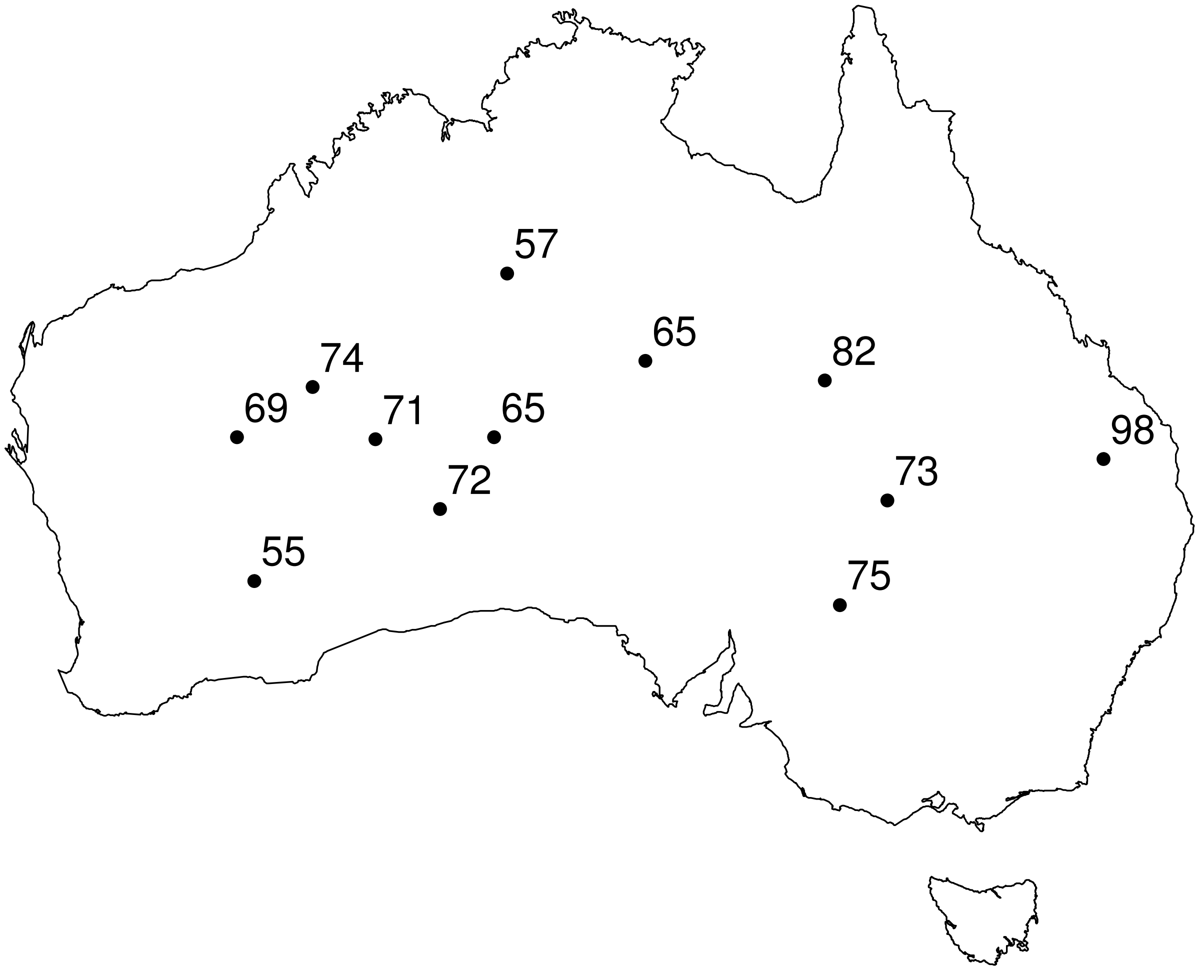


Points with marks

Points may also carry data (e.g. tree heights, meteorite composition)



Point pattern or geostatistical data?



Explanatory vs. response variables

Response variable: the quantity that we want to “predict” or “explain”

Explanatory variable: quantity that can be used to “predict” or “explain” the response.

Point pattern or geostatistical data?

Geostatistics treats the spatial locations as explanatory variables and the values attached to them as response variables.

Spatial point pattern statistics treats the spatial locations, and the values attached to them, as the response.

Point pattern or geostatistical data?

“Temperature is increasing as we move from South to North” — **geostatistics**
“Trees become less abundant as we move from South to North” — **point pattern statistics**

Software Overview

Software overview

For information on spatial statistics software:

For information on spatial statistics software:

- go to `cran.r-project.org`

Software overview

For information on spatial statistics software:

- go to `cran.r-project.org`
- find `Task Views`

Software overview

For information on spatial statistics software:

- go to `cran.r-project.org`
- find `Task Views --- Spatial`

GIS software

GIS = Geographical Information System

GIS = Geographical Information System

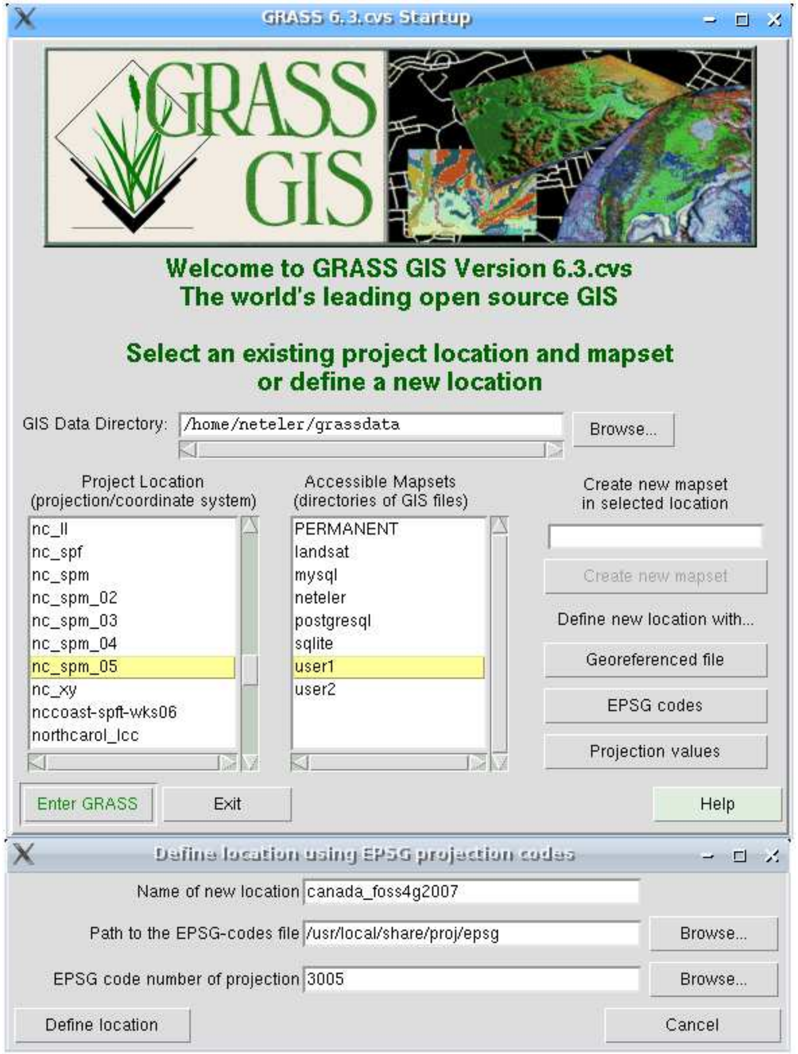
ArcInfo

GIS = Geographical Information System

ArcInfo proprietary `esri.com`

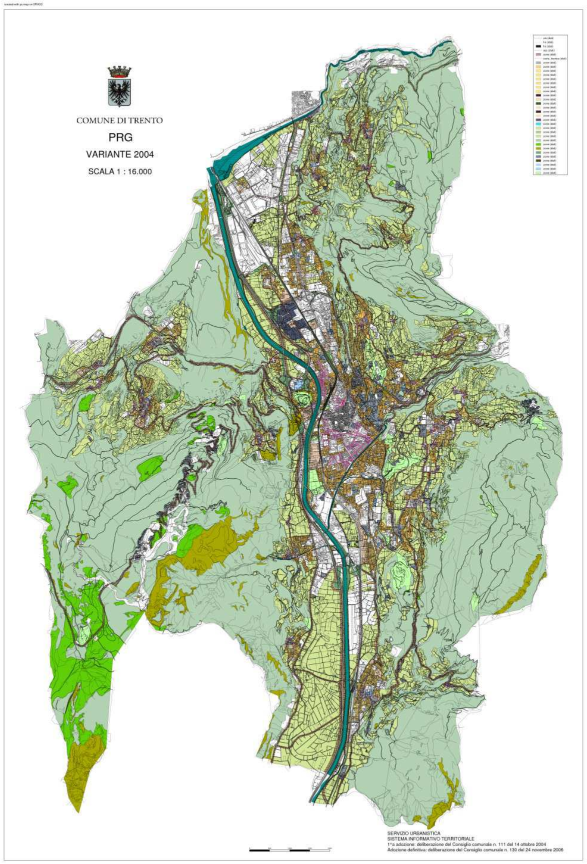
GIS = Geographical Information System

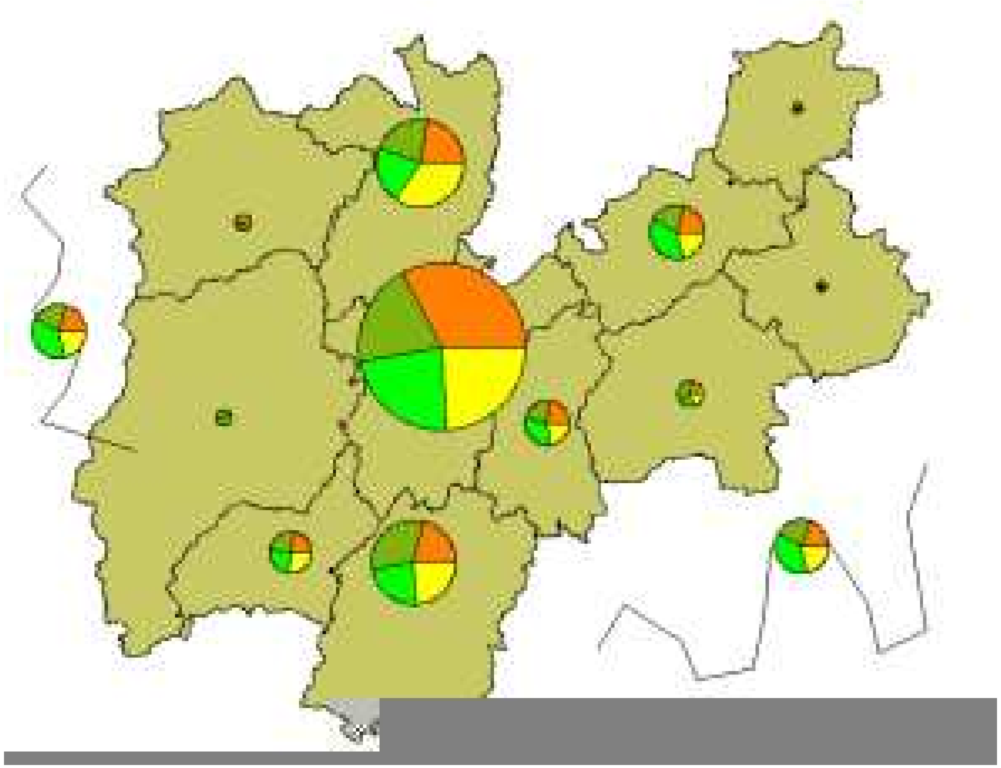
ArcInfo	proprietary	`esri.com`
GRASS	open source	`grass.osgeo.org`

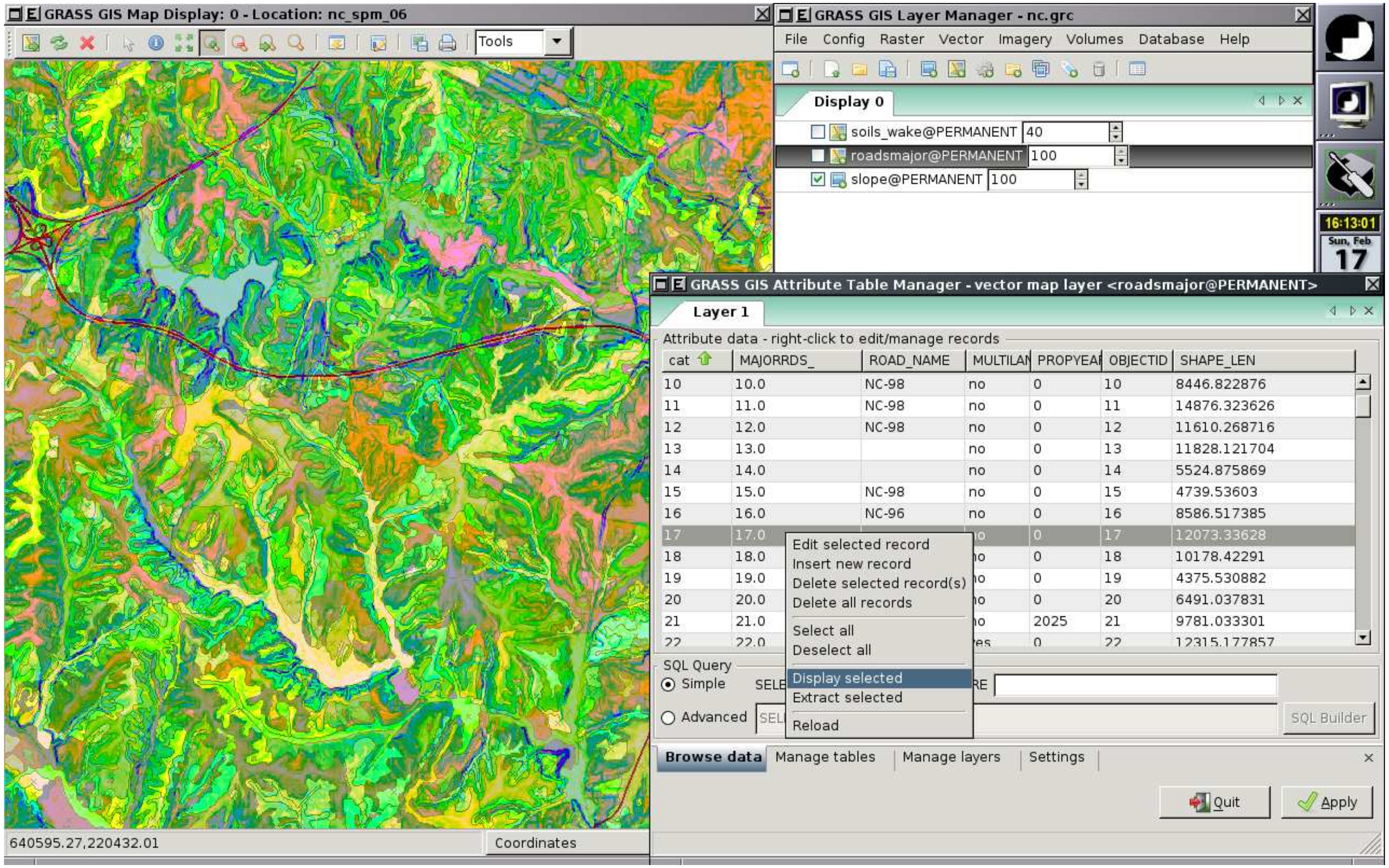

GRASS 6.3.cvs Startup
GRASS GIS
Welcome to GRASS GIS Version 6.3.cvs
The world's leading open source GIS
Select an existing project location and mapset
or define a new location
GIS Data Directory: /home/neteler/grassdata
Browse...
Project Location
(projection/coordinate system)
nc_ll
nc_spf
nc_spm
nc_spm_02
nc_spm_03
nc_spm_04
nc_spm_05
nc_xy
nccoast-spft-wks06
northcarol_lcc
Accessible Mapsets
(directories of GIS files)
PERMANENT
landsat
mysql
neteler
postgresql
sqlite
user1
user2
Create new mapset
in selected location
Create new mapset
Define new location with...
Georeferenced file
EPSG codes
Projection values
Enter GRASS
Exit
Help
Define location using EPSG projection codes
Name of new location canada_foss4g2007
Path to the EPSG-codes file /usr/local/share/proj/epsg
Browse...
EPSG code number of projection 3005
Browse...
Define location
Cancel

GRASS: Image data

GRASS: Vector data

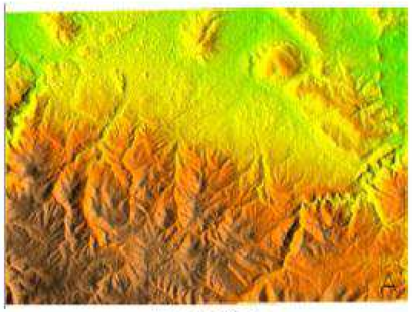




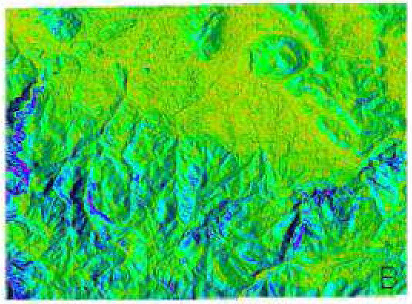


cat	MAJORRDS_	ROAD_NAME	MULTILAN	PROPYEAR	OBJECTID	SHAPE_LEN
10	10.0	NC-98	no	0	10	8446.822876
11	11.0	NC-98	no	0	11	14876.323626
12	12.0	NC-98	no	0	12	11610.268716
13	13.0		no	0	13	11828.121704
14	14.0		no	0	14	5524.875869
15	15.0	NC-98	no	0	15	4739.53603
16	16.0	NC-96	no	0	16	8586.517385
17	17.0		no	0	17	12073.33628
18	18.0		no	0	18	10178.42291
19	19.0		no	0	19	4375.530882
20	20.0		no	0	20	6491.037831
21	21.0		no	2025	21	9781.033301
22	22.0		yes	0	22	12315.177857

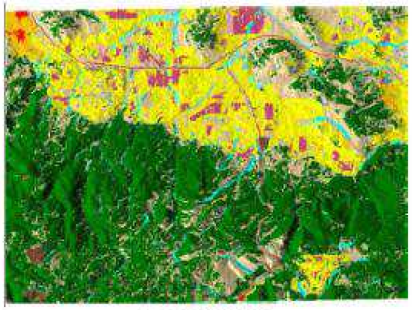
GRASS: Multiple data layers



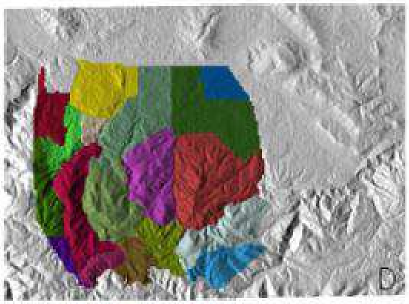
Elevation



Slope

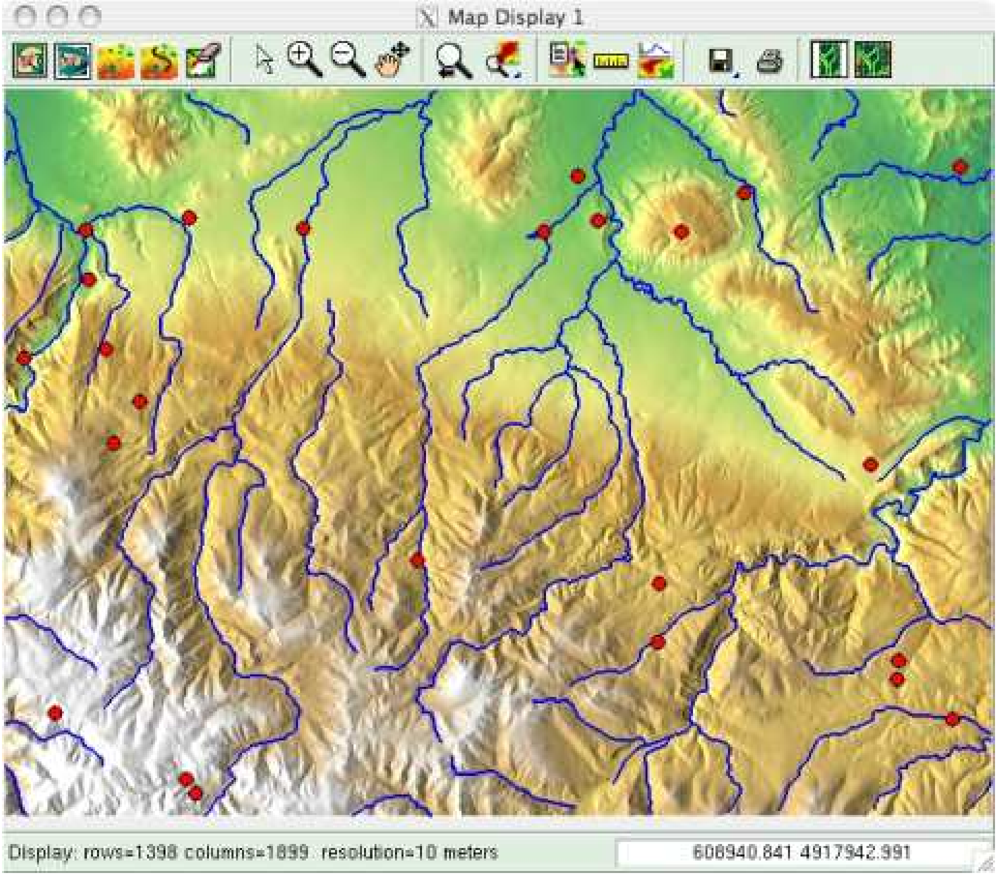


Landcover

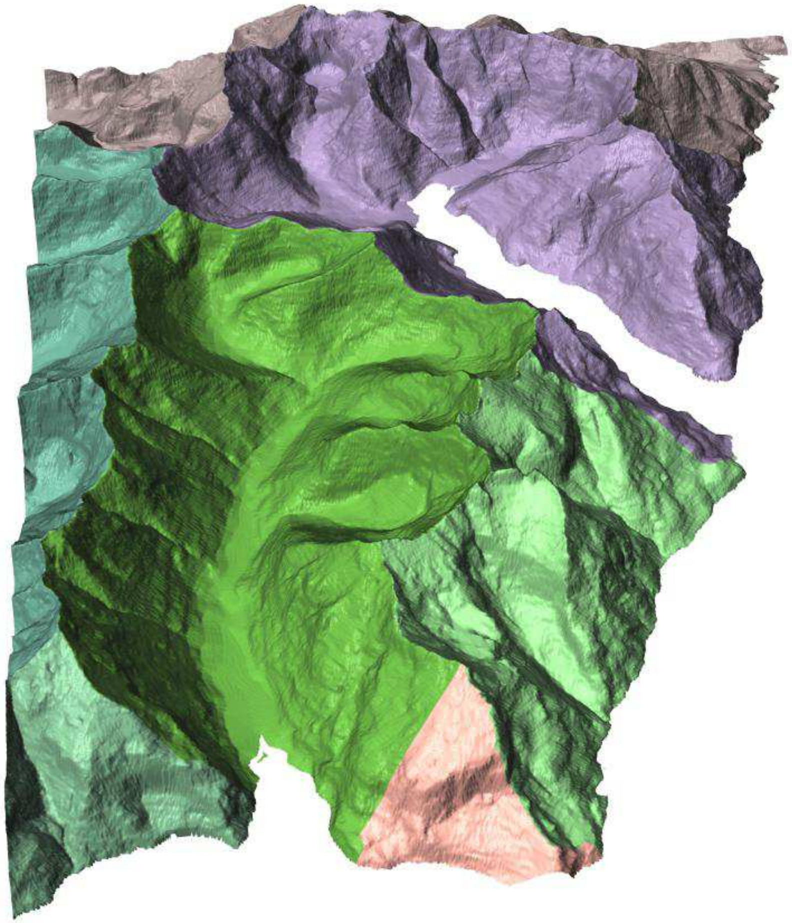


Training Areas

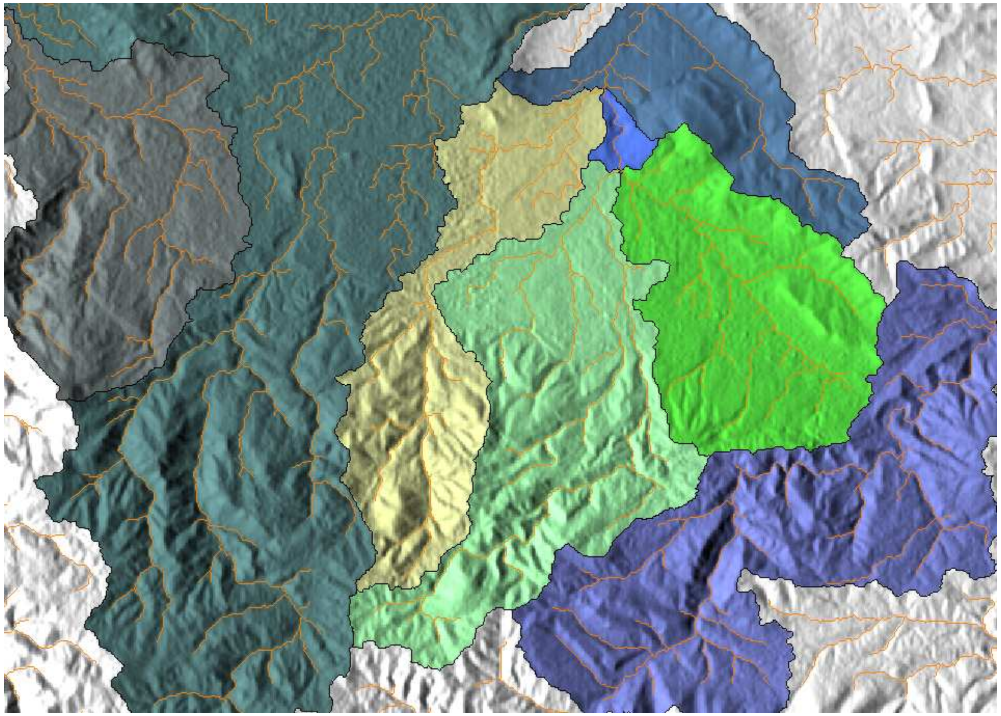
GRASS: Mixed layers



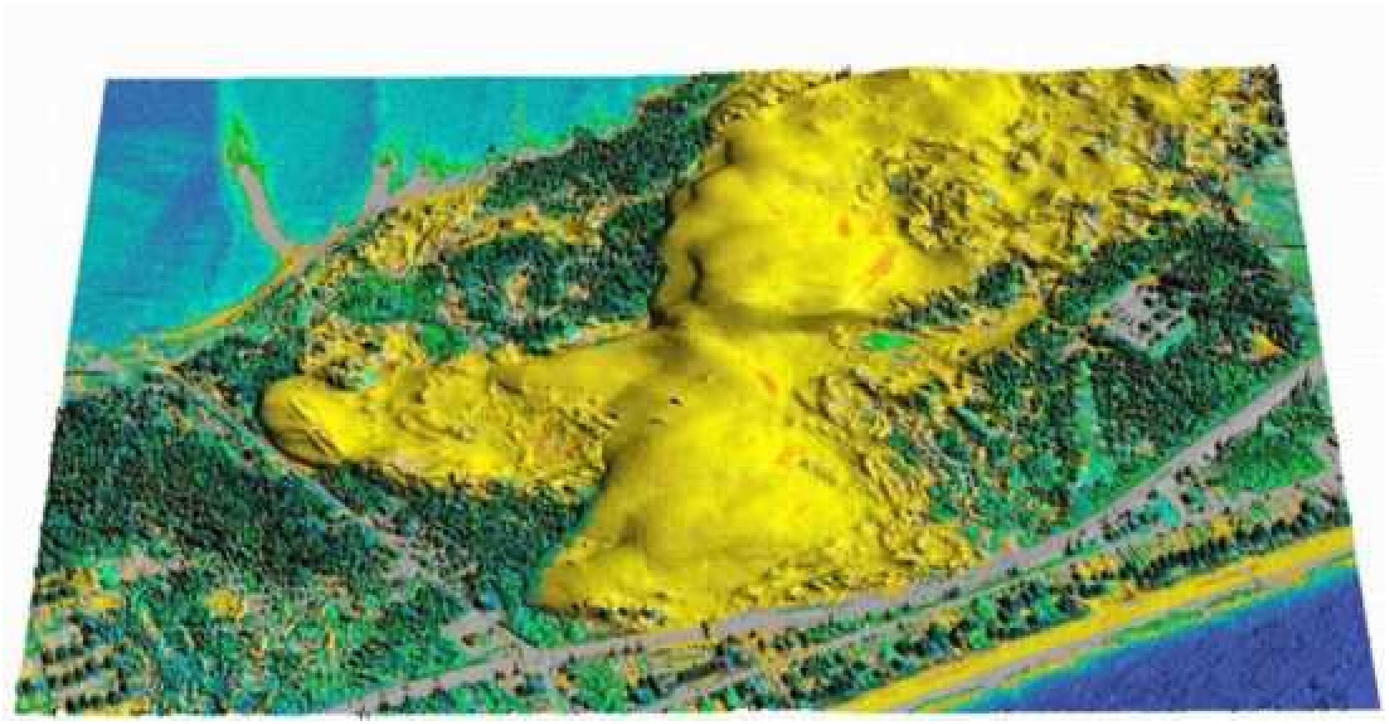
GRASS: visualisation



GRASS: Data integration

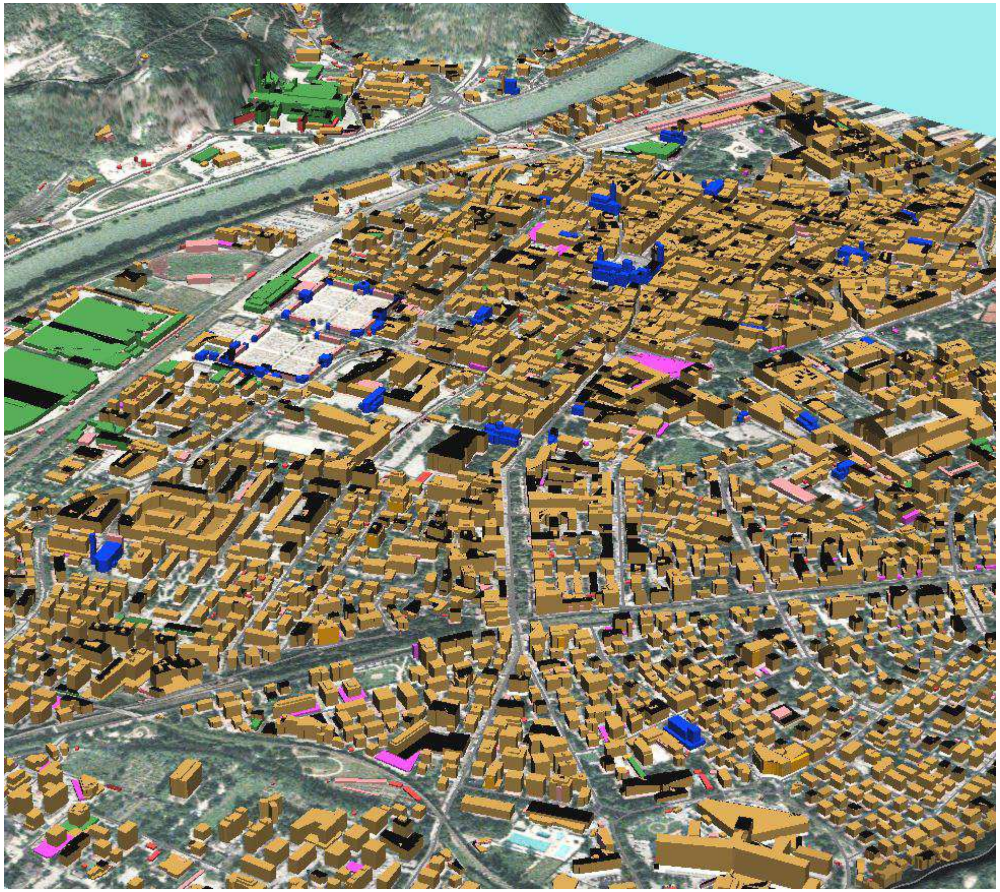


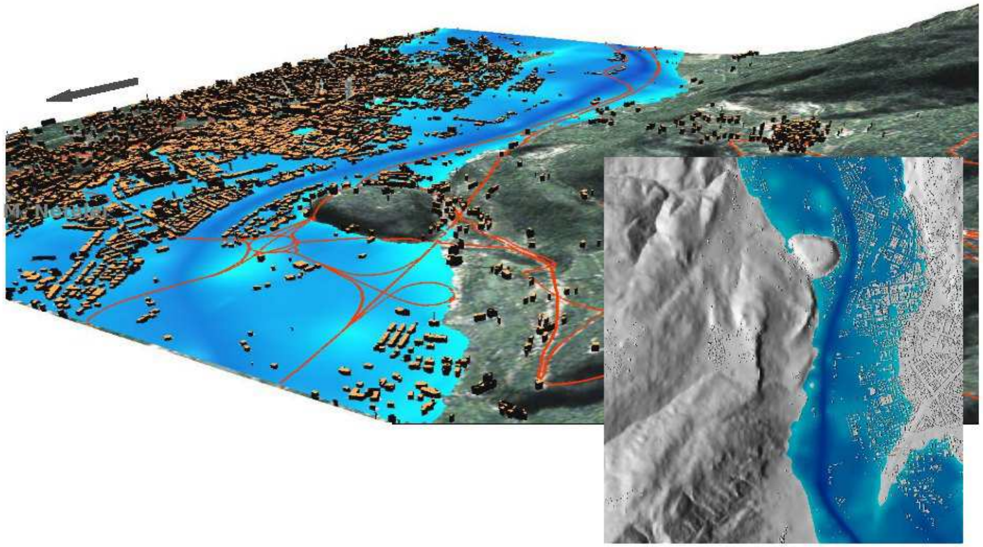
GRASS: I mean really integrated



GRASS: did I mention data integration

GRASS: unbelievably well integrated





GRASS: runs on anything

Recommendations:

Recommendations:

For visualisation of spatial data, especially for presentation graphics, use a GIS.

Recommendations:

For visualisation of spatial data, especially for presentation graphics, use a GIS.

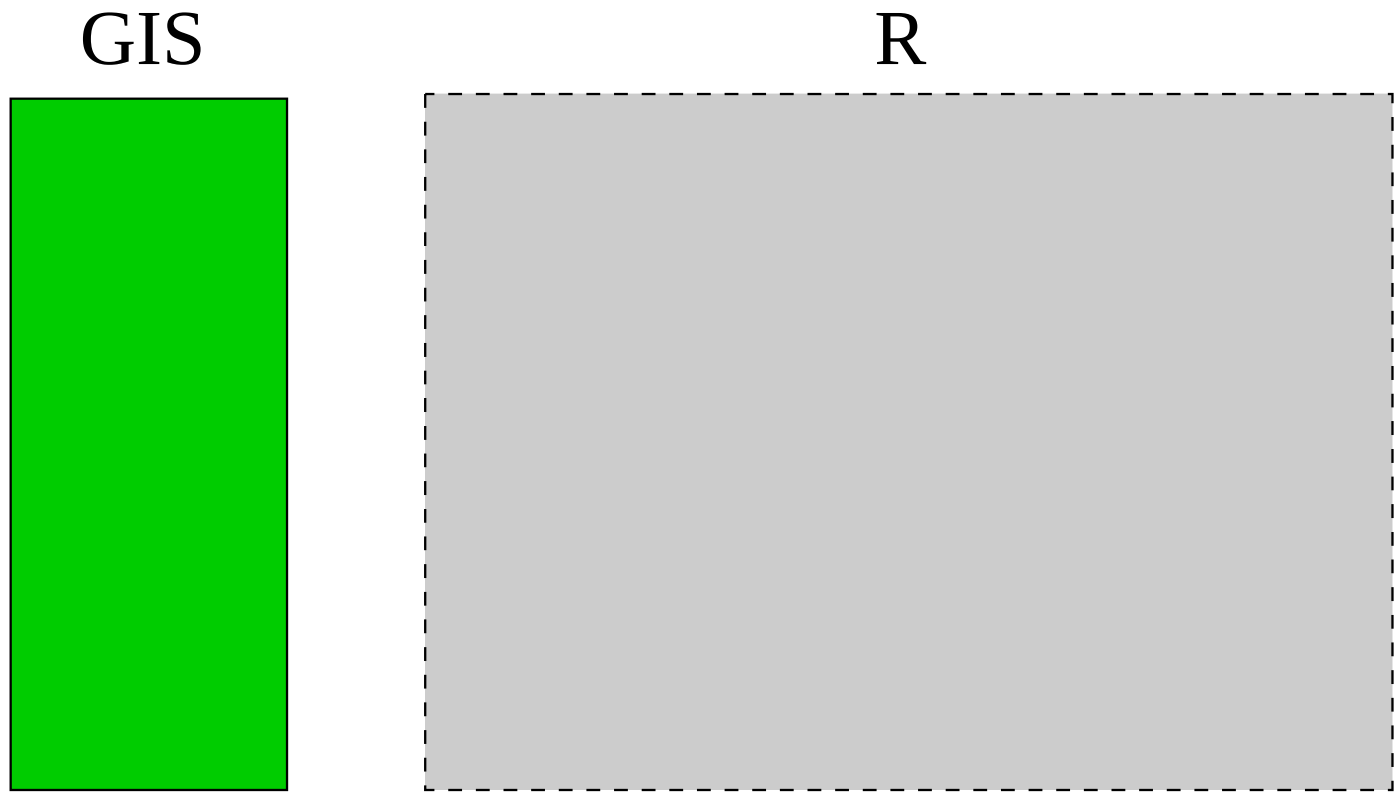
For statistical analysis of spatial data, use R.

Recommendations:

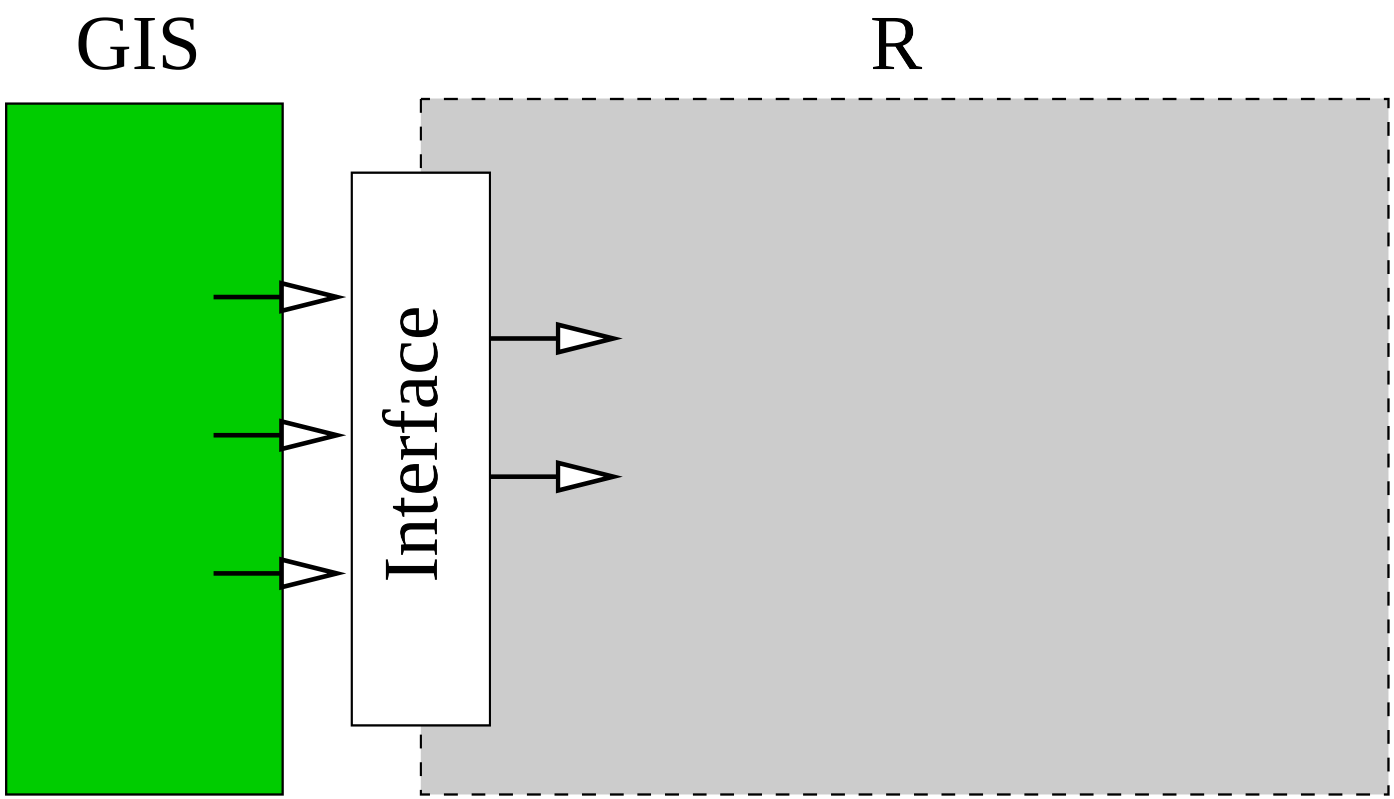
For visualisation of spatial data, especially for presentation graphics, use a GIS.

For statistical analysis of spatial data, use R.

Establish two-way communication between GIS and R, either through a direct software interface, or by reading/writing files in mutually acceptable format.


GIS
R

Putting the pieces together



Interface between R and GIS (online or offline)

Direct interfaces between R *and GIS:*

`spgrass6`	interface to GRASS 6
`RArcInfo`	interface to ArcInfo

Direct interfaces between R *and GIS:*

`spgrass6`	interface to GRASS 6
`RArcInfo`	interface to ArcInfo

Start R and GRASS independently; then start `library(spgrass6)` to establish communication

Dominant formats for data files:

ESRI "shapefiles"	ArcInfo software	`esri.com`
NetCDF	Unidata GIS standard	`unidata.ucar.edu`

GIS data files

Dominant formats for data files:

ESRI “shapefiles”	ArcInfo software	`esri.com`
NetCDF	Unidata GIS standard	`unidata.ucar.edu`

Libraries for reading/writing formats, etc:

GDAL	geospatial data	`gdal.org`
PROJ.4	map projections	`remotesensing.org`

GIS data files

Dominant formats for data files:

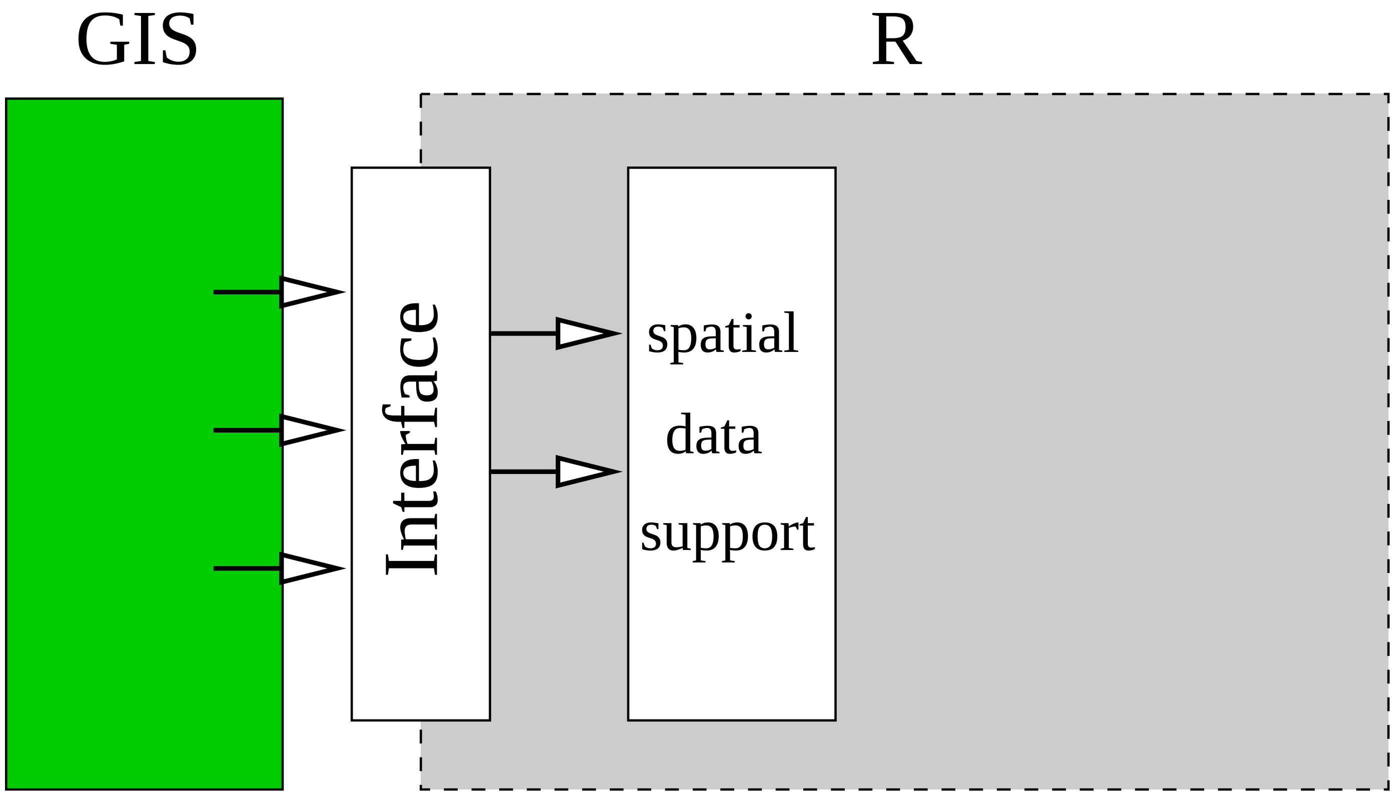
ESRI "shapefiles"	ArcInfo software	`esri.com`
NetCDF	Unidata GIS standard	`unidata.ucar.edu`

Libraries for reading/writing formats, etc:

GDAL	geospatial data	`gdal.org`
PROJ.4	map projections	`remotesensing.org`

R *packages handling GIS data files:*

`rgdal`	shapefiles, GDAL, PROJ.4
`maps` + `mapproj`	map databases

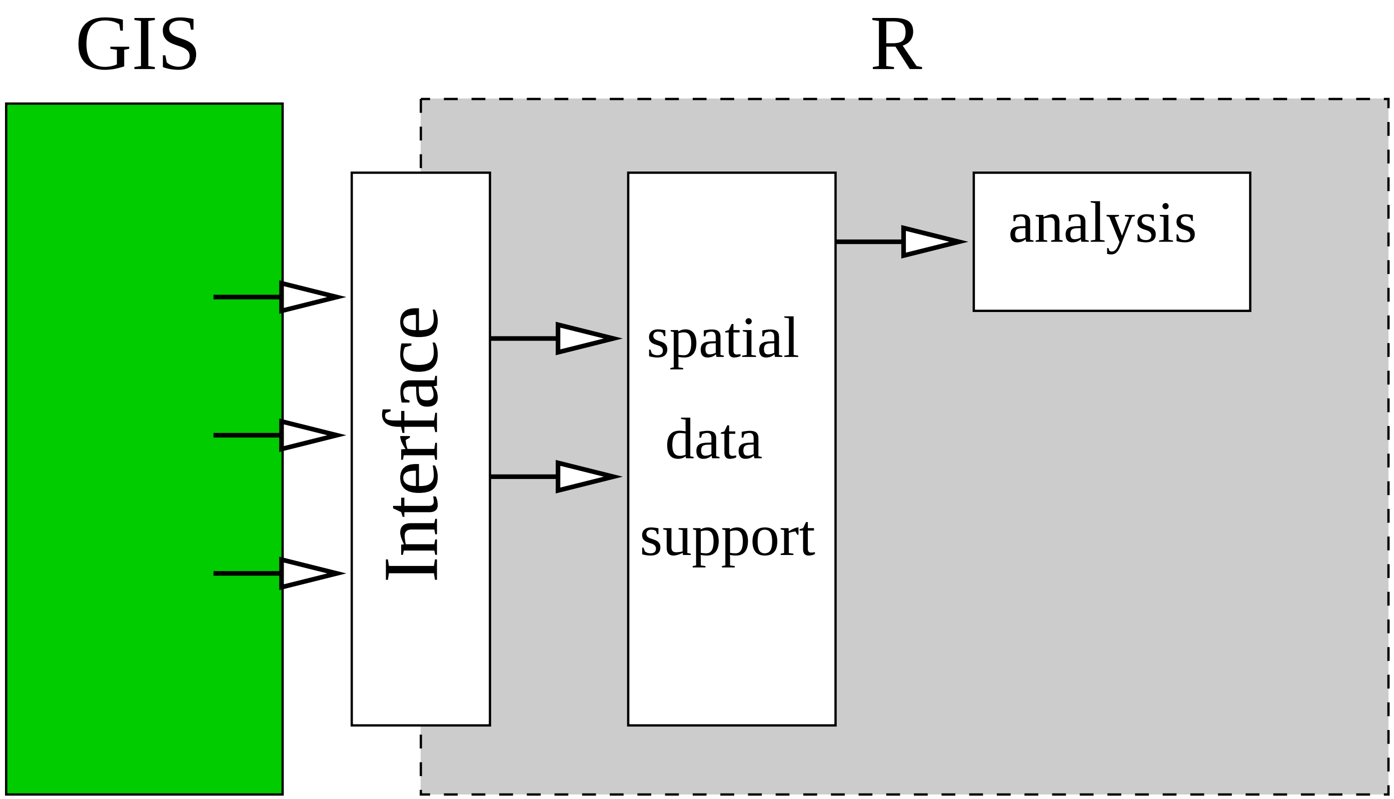


Support for spatial data: data structures, classes, methods

R packages supporting spatial data

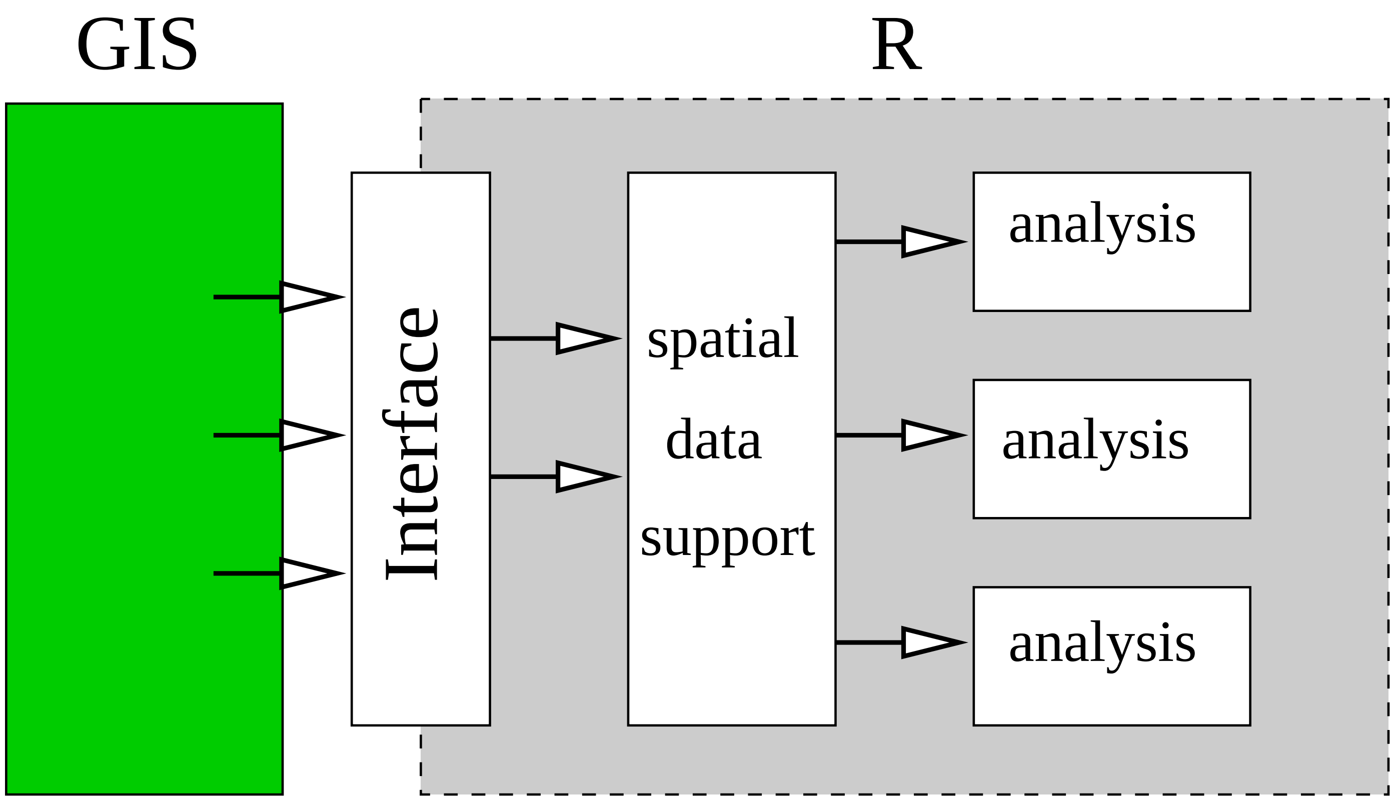
R *packages supporting spatial data classes:*	
`sp`	generic
`maps`	polygon maps
`spatstat`	point patterns

Putting the pieces together



Capabilities for statistical analysis

Putting the pieces together



Multiple packages for different analyses

Statistical functionality

R *packages for geostatistical data*

`gstat`	classical geostatistics
`geoR`	model-based geostatistics
`RandomFields`	stochastic processes
`akima`	interpolation

Statistical functionality

R *packages for geostatistical data*

`gstat`	classical geostatistics
`geoR`	model-based geostatistics
`RandomFields`	stochastic processes
`akima`	interpolation

R *packages for regional data*

`spdep`	spatial dependence
`spgwr`	geographically weighted regression

Statistical functionality

R *packages for geostatistical data*

`gstat`	classical geostatistics
`geoR`	model-based geostatistics
`RandomFields`	stochastic processes
`akima`	interpolation

R *packages for regional data*

`spdep`	spatial dependence
`spgwr`	geographically weighted regression

R *packages for point patterns*

`spatstat`	parametric modelling, diagnostics
`splancs`	nonparametric, space-time

Geostatistical data

The R package `gstat` does classical geostatistics: kriging, variograms etc.

Maas River data

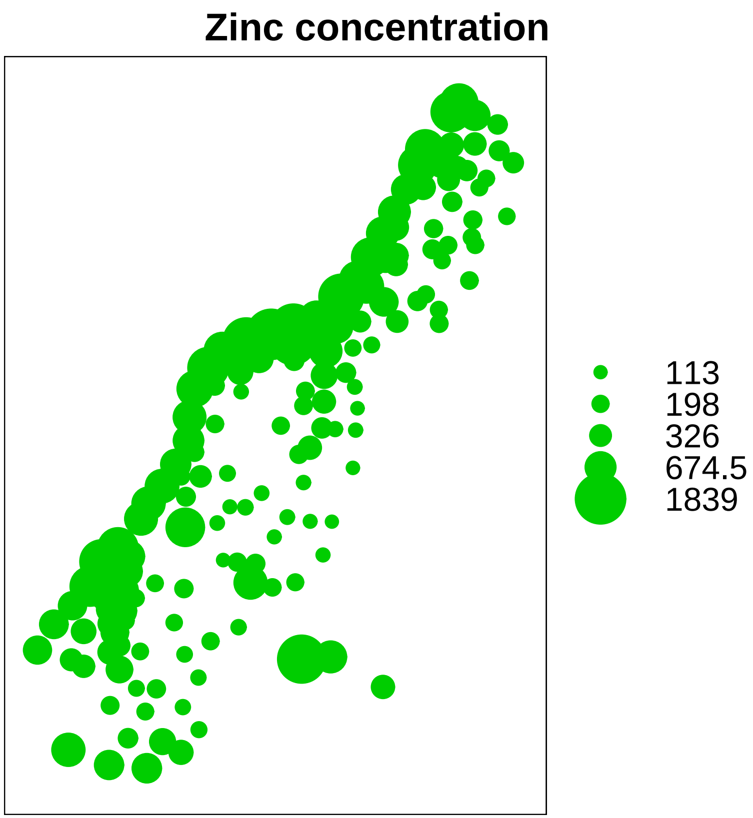
```
 > library(gstat)
Loading required package: sp
 > data(meuse)
 > class(meuse)
[1] "data.frame"
> names(meuse)
 [1] "x"         "y"         "cadmium" "copper"  "lead"    "zinc"    "elev"
 [8] "dist"      "om"        "ffreq"   "soil"    "lime"    "landuse" "dist.m"
```

Convert raw data to spatial class

```
> coordinates(meuse) = ~x+y
> class(meuse)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
```

Bubble plot

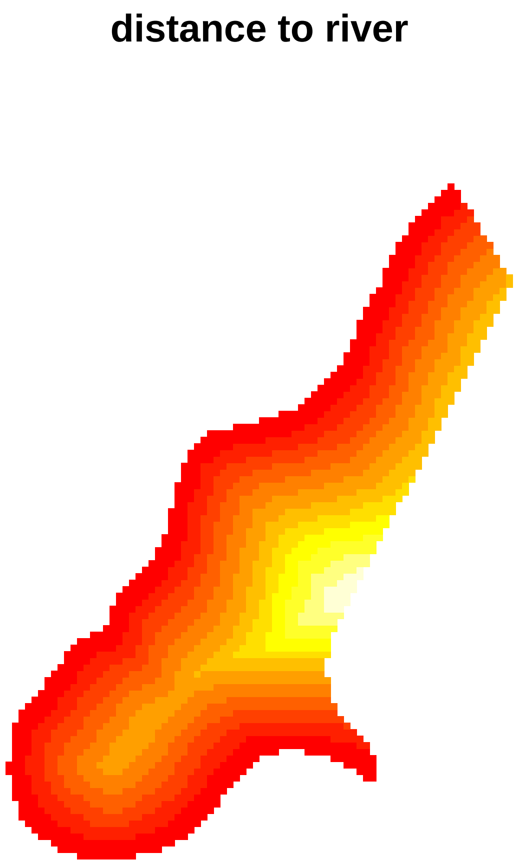
```
> bubble(meuse, "zinc",
         main="Zinc concentration (ppm)")
```



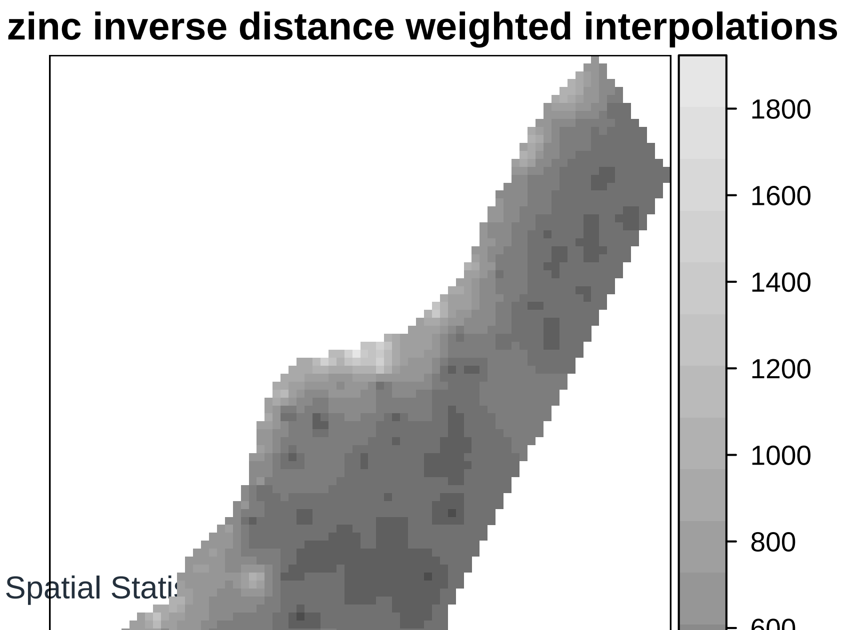
Pixel image

```
> data(meuse.grid)
> coordinates(meuse.grid) = ~x+y
> gridded(meuse.grid) = TRUE
> class(meuse.grid)
[1] "SpatialPixelsDataFrame"
attr(,"package")
[1] "sp"
```

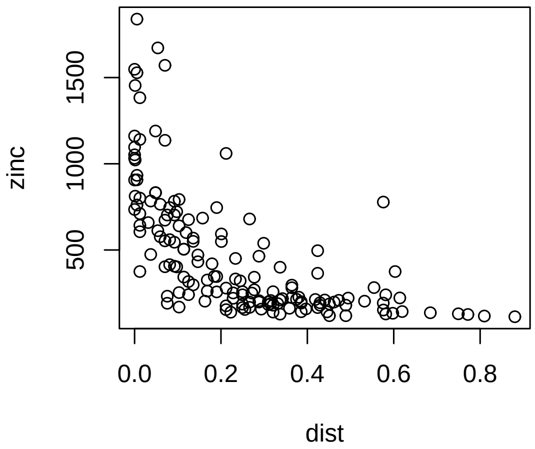
```
> image(meuse.grid["dist"])
> title("distance to river")
```



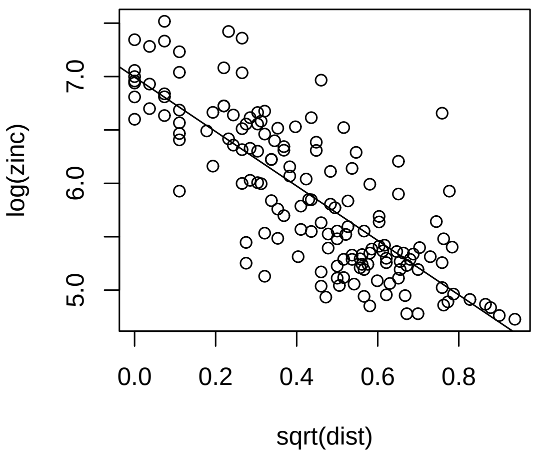
```
> zinc.idw = krige(zinc~1, meuse, meuse.grid)
[inverse distance weighted interpolation]
> class(zinc.idw)
[1] "SpatialPixelsDataFrame"
attr(,"package")
[1] "sp"
> spplot(zinc.idw["var1.pred"],
    main = "Inverse distance weighted interpolations")
```



```
> plot(zinc ~ dist, meuse)
```

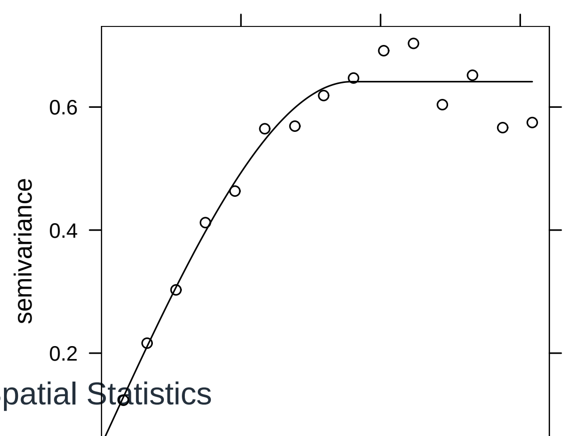


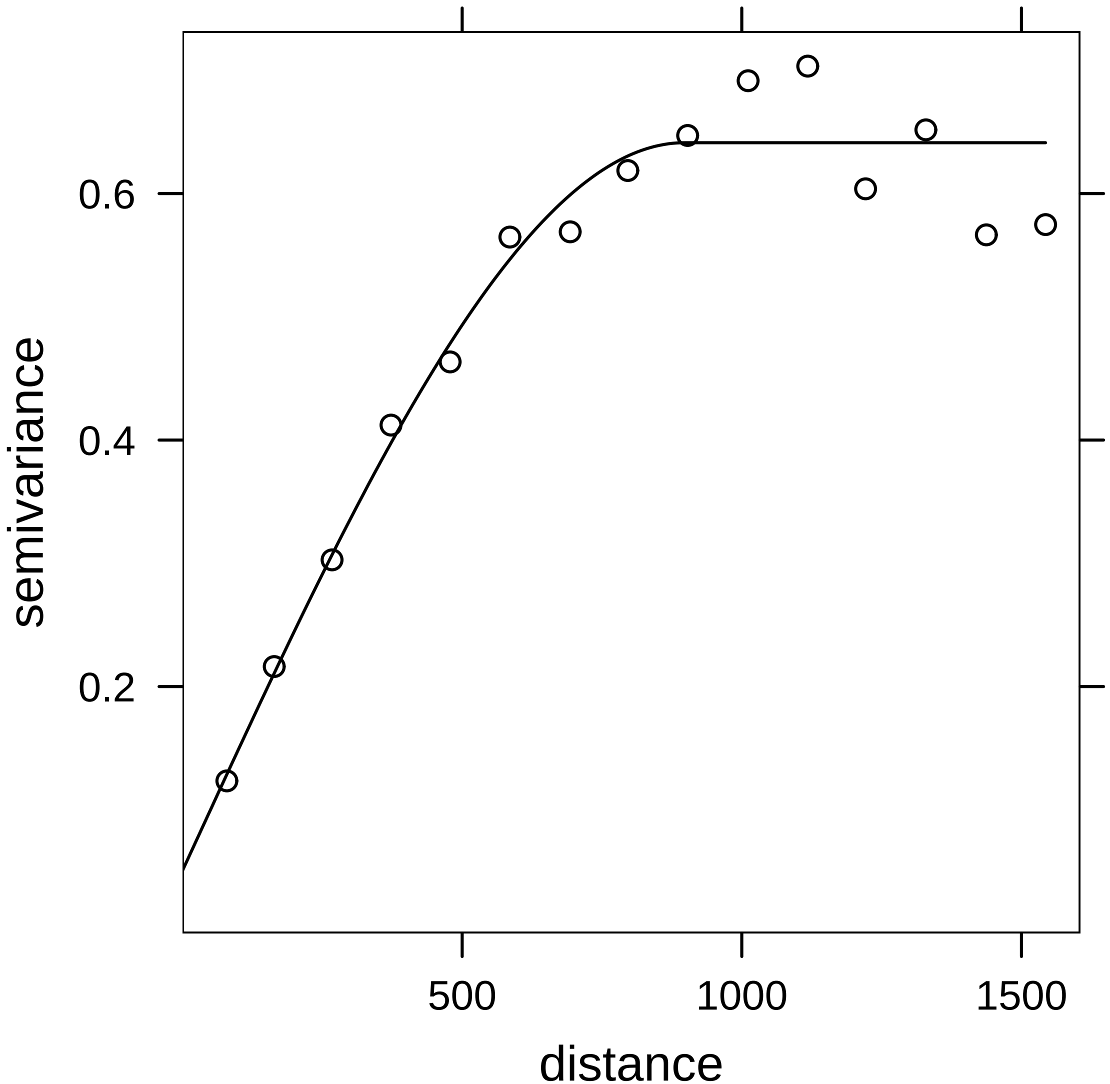
```
> plot(log(zinc) ~ sqrt(dist), meuse)
> abline(lm(log(zinc) ~ sqrt(dist), meuse))
```



Variogram assuming constant mean:

```
> lzn.vgm = variogram(log(zinc)~1, meuse)
> head(lzn.vgm)
   np      dist     gamma dir.hor dir.ver   id
1  57  79.29244 0.1234479       0       0 var1
2 299 163.97367 0.2162185       0       0 var1
3 419 267.36483 0.3027859       0       0 var1
> lzn.fit = fit.variogram(lzn.vgm, model = vgm(1, "Sph", 900, 1))
> lzn.fit
  model      psill    range
1   Nug 0.05066243   0.0000
2   Sph 0.59060780 897.0209
> plot(lzn.vgm, lzn.fit)
```

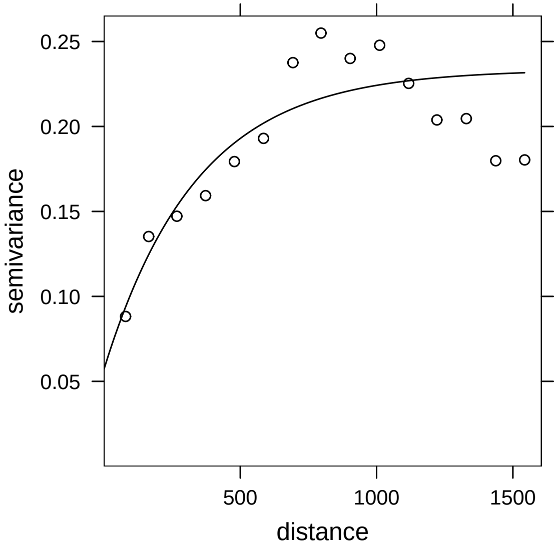


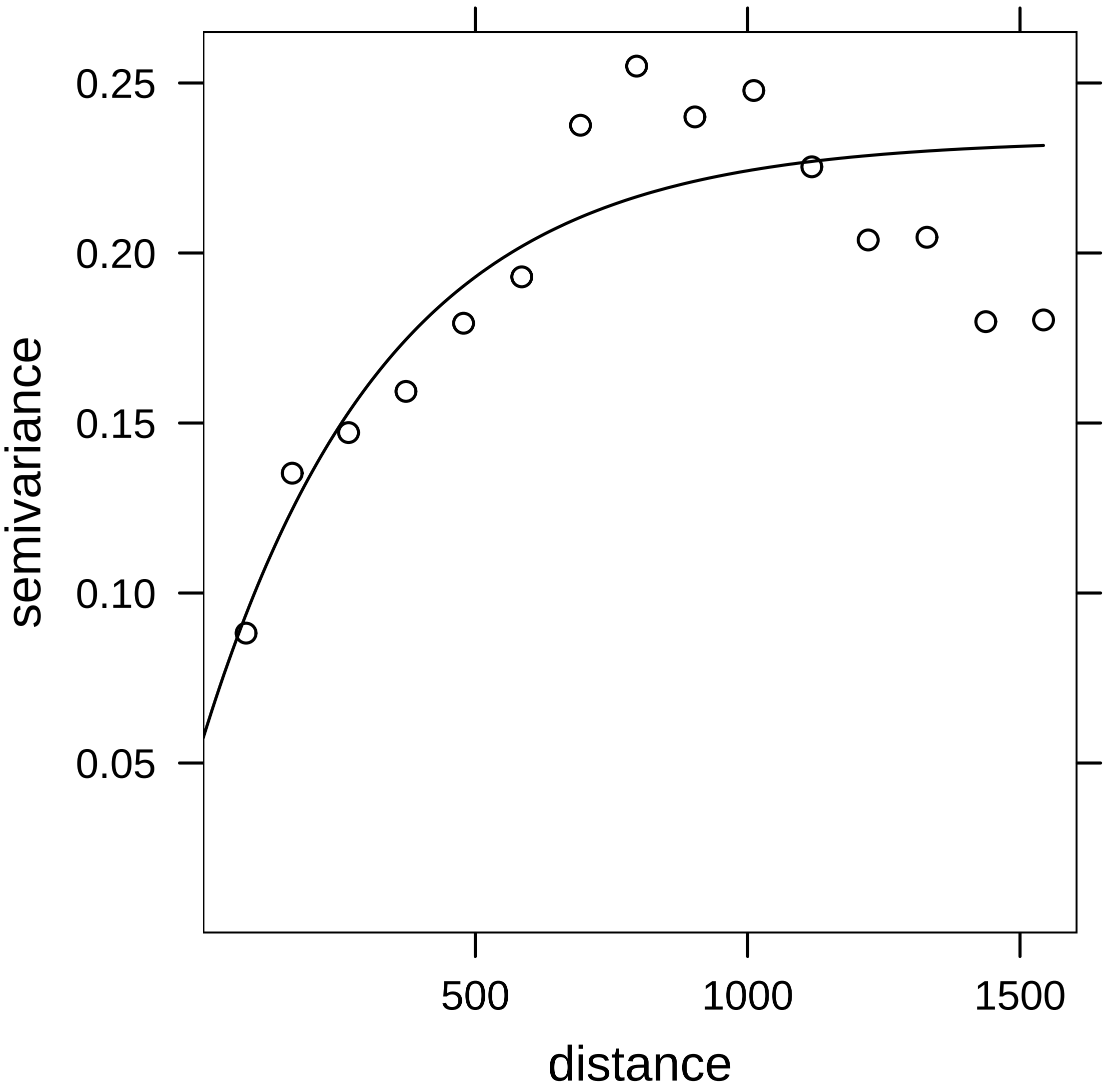


0.6
0.4
0.2
semivariance
500
1000
1500
distance

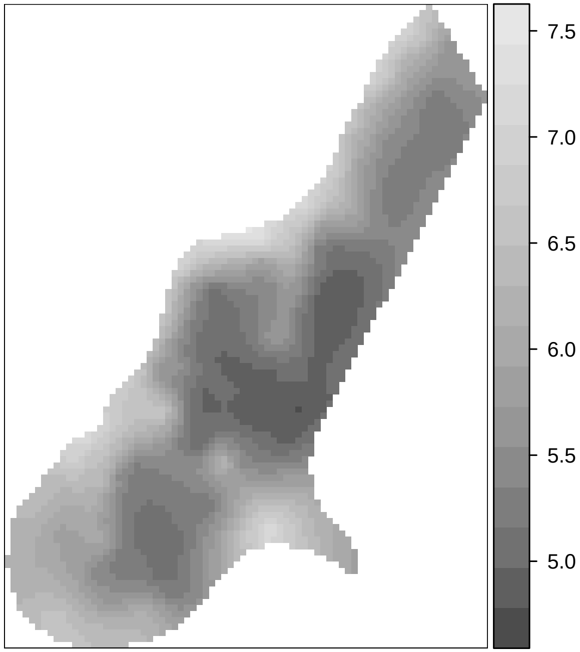
Variogram of residuals from a fitted spatial trend:

```
> lznr.vgm = variogram(log(zinc)~sqrt(dist), meuse)
> lznr.fit = fit.variogram(lznr.vgm, model = vgm(1, "Exp", 300, 1))
> lznr.fit
  model      psill    range
1   Nug 0.05712231   0.0000
2   Exp 0.17641559 340.3201
> plot(lznr.vgm, lznr.fit)
```



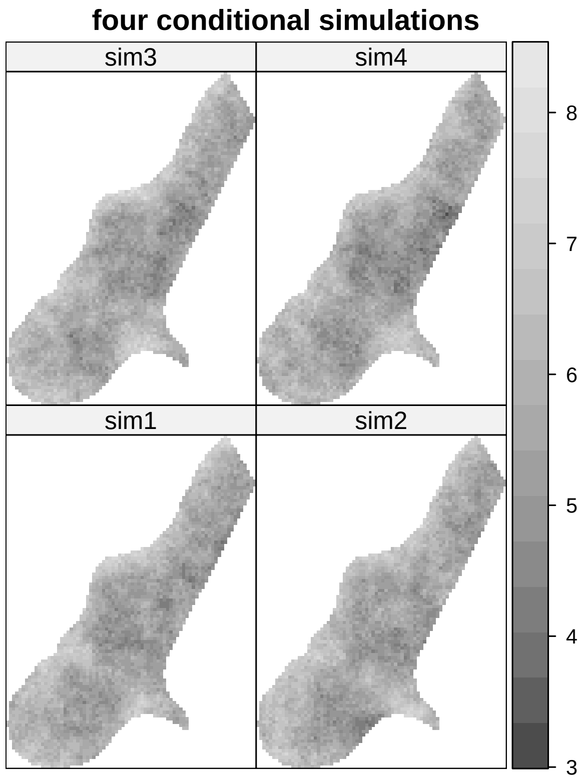

0.25
0.20
0.15
0.10
0.05
semivariance
500
1000
1500
distance

```
lzn.kriged = krige(log(zinc)~1, meuse, meuse.grid, model = lzn.fit)
spplot(lzn.kriged["var1.pred"])
```



Conditional simulation

```
lzn.condsim = krige(log(zinc)~1, meuse, meuse.grid, model = lzn.fit,
    nmax = 30, nsim = 4)
spplot(lzn.condsim, main = "four conditional simulations")
```



Regional data

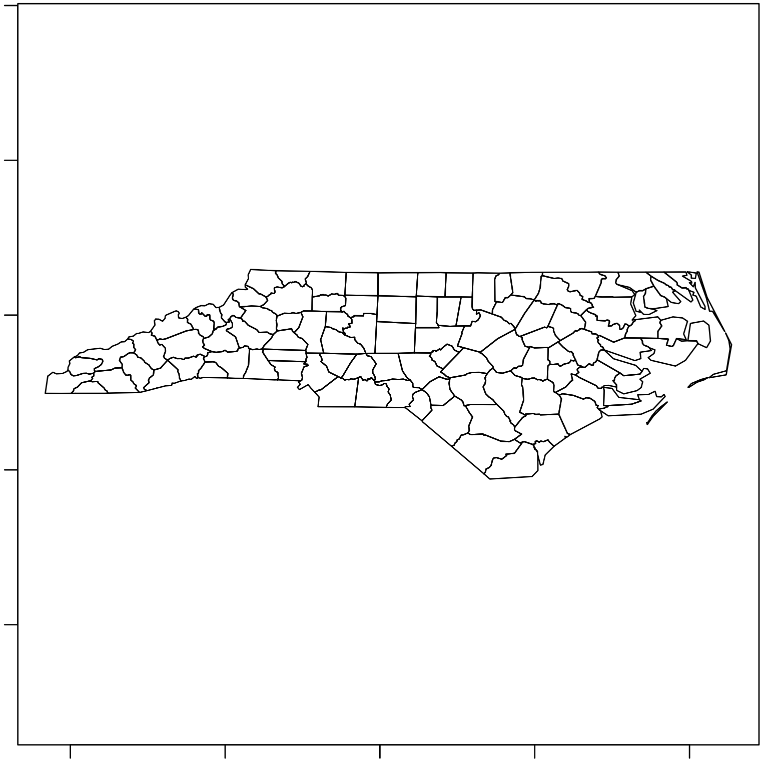
The R package `spdep` analyses regional data using neighbourhood dependence statistics.

Software

```
> library(spdep)
Loading required package: sp
Loading required package: tripack
Loading required package: maptools
Loading required package: foreign
Loading required package: SparseM
Loading required package: boot
```

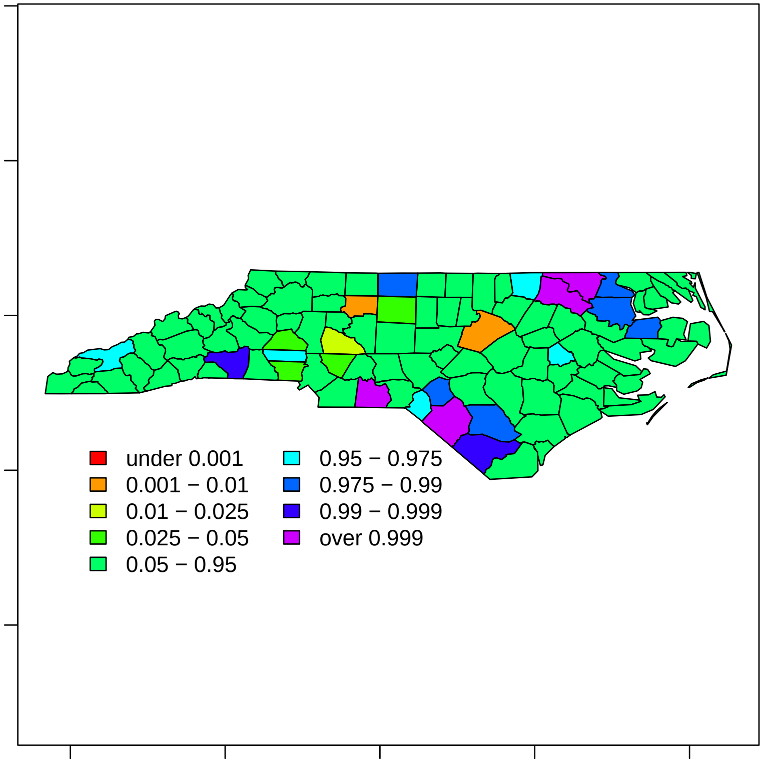
North Carolina SIDS data

```
> data(nc.sids)
> plot(sidspolys, forcefill=FALSE)
```



p-value for each region

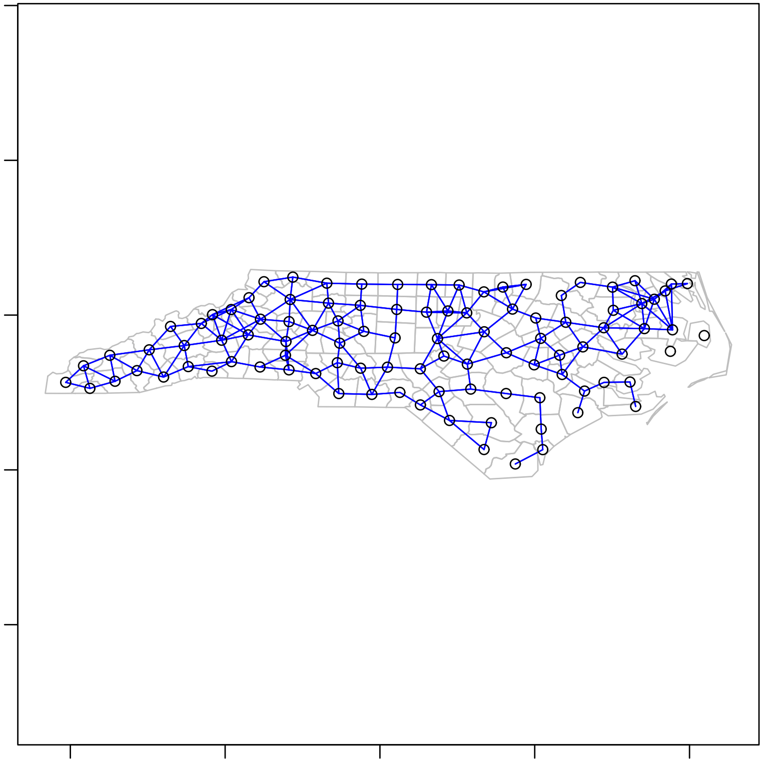
```
pmap <- probmap(nc.sids$SID74, nc.sids$BIR74)
brks <- c(0,0.001,0.01,0.025,0.05,0.95,0.975,0.99,0.999,1)
cols <- rainbow(length(brks))
plot(sidspolys, col=cols[findInterval(pmap$pmap, brks)], forcefill=FALSE)
```



Neighbours

Define which regions are immediate neighbours according to some criterion.

```
coords <- nc.sids[, c("east", "north")]
gg <- gabrielneigh(coords)
nb <- graph2nb(gg)
```



Moran's I

An index of spatial autocorrelation:

$$I = \frac{n \sum_i \sum_j w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{(\sum_i \sum_j w_{ij})(\sum_i (y_i - \bar{y})^2)}$$

where $w_{ij} = 1$ if sites i and j are neighbours, and 0 otherwise.

Convert neighbourhood relations to weights w_{ij} between each pair of regions i, j.

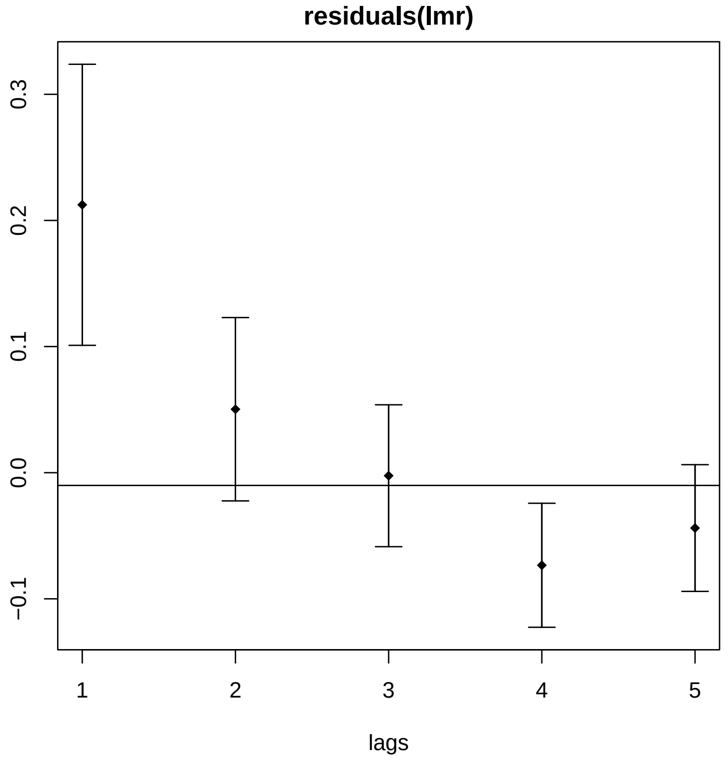
```
lw <- nb2listw(nb)
```

(Non-binary weights are possible too.)

```
> rates <- with(nc.sids, SID74/BIR74)
> moran.test(rates, listw=lw)

        Moran's I test under randomisation

data:  rates
weights: lw

Moran I statistic standard deviate = 4.1051, p-value = 2.021e-05
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.222612195      -0.010101010       0.003213686
```

Spatial correlogram

```
> lmr <- lm(rates ~ 1, data=nc.sids, weights=BIR74)
> res <- sp.correlogram(nb, residuals(lmr), order=5, method="I")
> plot(res)
```

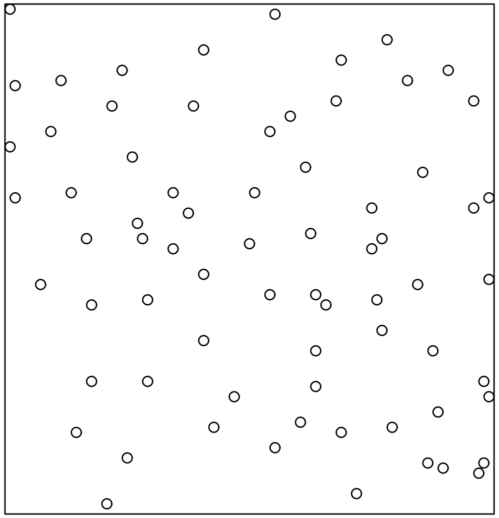


Spatial point patterns

The R package `spatstat` supports statistical analysis for spatial point patterns.

Point patterns

A **point pattern** dataset gives the locations of objects/events occurring in a study region.



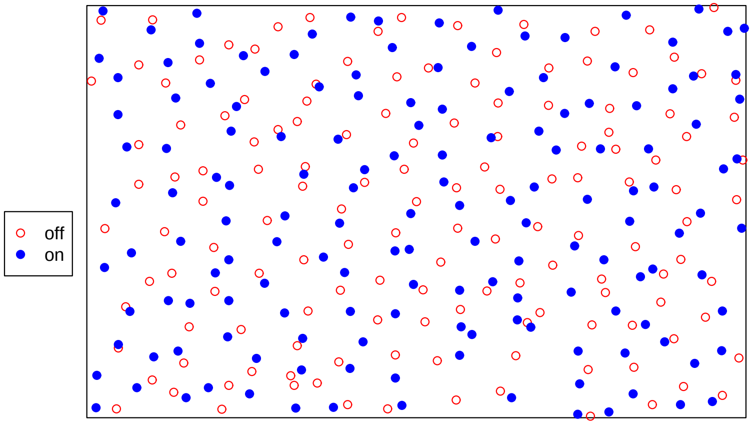
The points could represent trees, animal nests, earthquake epicentres, petty crimes, domiciles of new cases of influenza, galaxies, etc.

Marks

The points may have extra information called **marks** attached to them. The mark represents an "attribute" of the point.

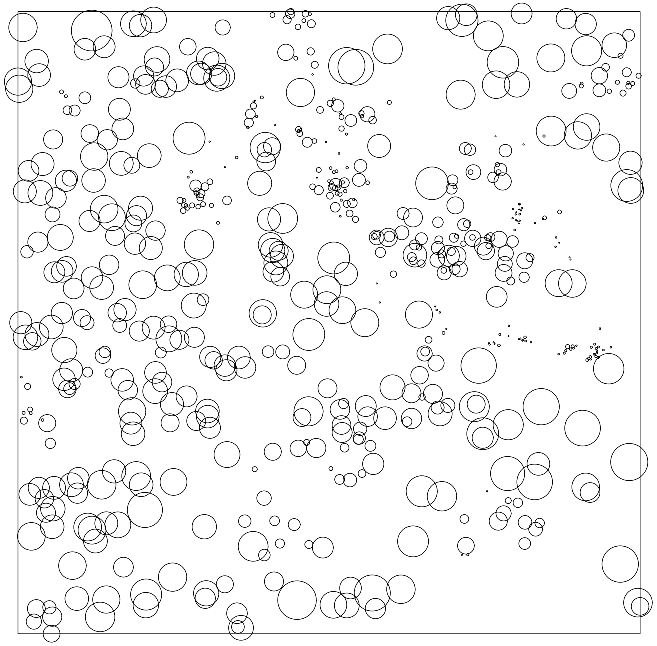
The points may have extra information called **marks** attached to them. The mark represents an "attribute" of the point.

The mark variable could be *categorical*, e.g. species or disease status:



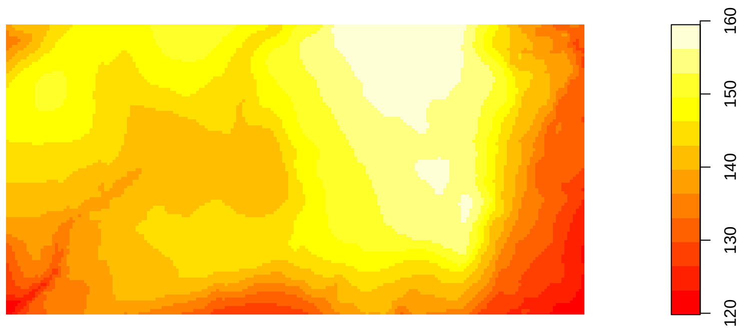
Continuous marks

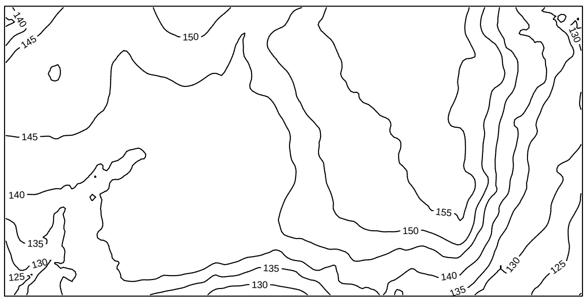
The mark variable could be *continuous*, e.g. tree diameter:



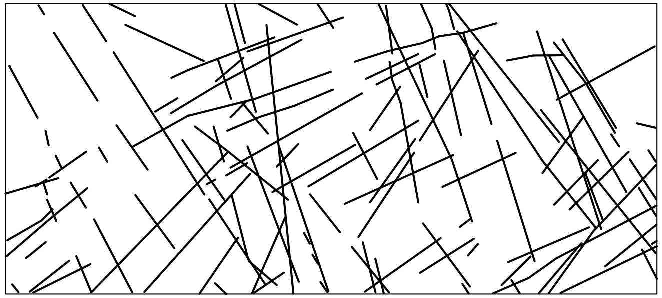
Our dataset may also include **covariates** — any data that we treat as explanatory, rather than as part of the 'response'.

Covariate data may be a *spatial function* $Z(u)$ defined at all spatial locations u, e.g. altitude, soil pH, displayed as a pixel image or a contour plot:





Covariate data may be another *spatial pattern* such as another point pattern, or a line segment pattern, e.g. a map of geological faults:

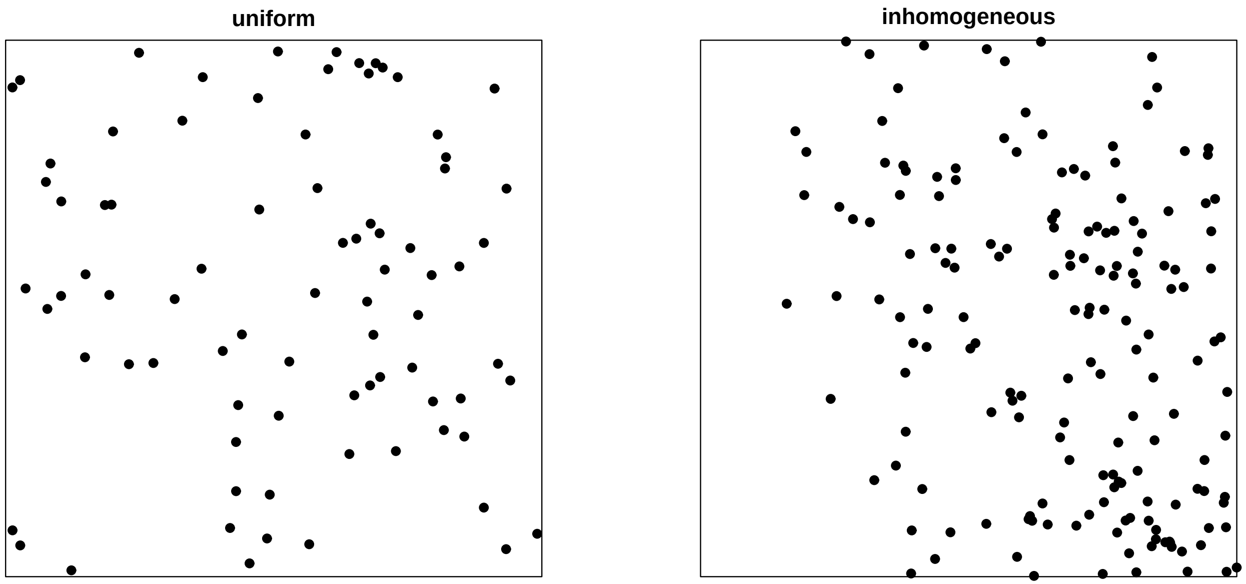


Intensity

Intensity

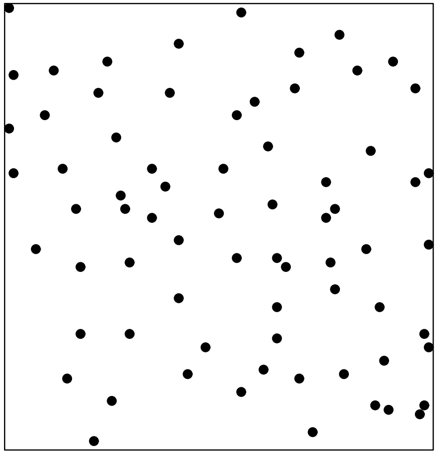
'Intensity' is the average density of points (expected number of points per unit area).

Intensity may be constant ('uniform') or may vary from location to location ('non-uniform' or 'inhomogeneous').



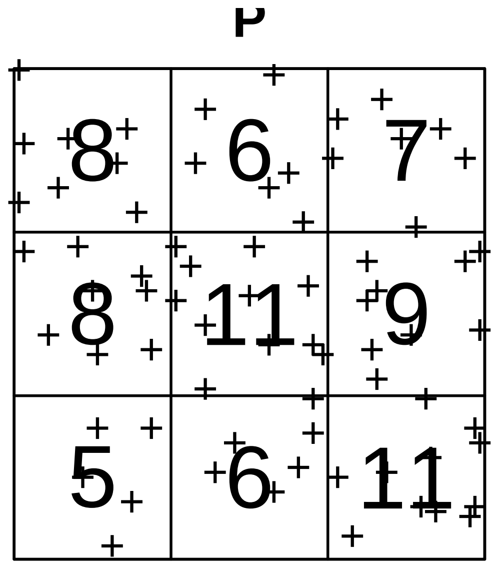
Swedish Pines data

```
> data(swedishpines)
> P <- swedishpines
> plot(P)
```



Divide study region into rectangles ('quadrats') of equal size, and count points in each rectangle.

```
Q <- quadratcount(P, nx=3, ny=3)
Q
plot(Q, add=TRUE)
```



χ^2 test of uniformity

If the points have uniform intensity, and are completely random, then the quadrat counts should be Poisson random numbers with constant mean.

χ^2 test of uniformity

If the points have uniform intensity, and are completely random, then the quadrat counts should be Poisson random numbers with constant mean.

Use the χ^2 goodness-of-fit test statistic

$$X^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

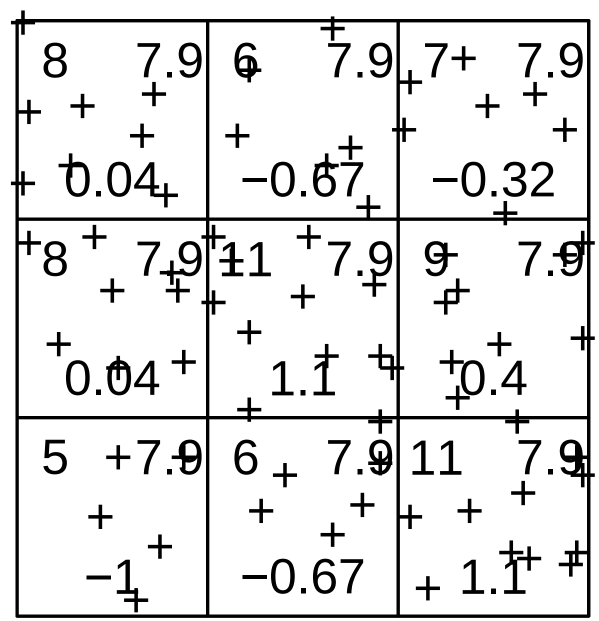
χ^2 test of uniformity

If the points have uniform intensity, and are completely random, then the quadrat counts should be Poisson random numbers with constant mean.
Use the χ^2 goodness-of-fit test statistic

$$X^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

```
> quadrat.test(P, nx=3, ny=3)

        Chi-squared test of CSR using quadrat counts

data:  P
X-squared = 4.6761, df = 8, p-value = 0.7916
```

```
> QT <- quadrat.test(P, nx=3, ny=3)
> plot(P)
> plot(QT, add=TRUE)
```



Kernel smoothing

Kernel smoothed intensity

$$\widetilde{\lambda}(u) = \sum_{i=1}^{n} \kappa(u - x_i)$$

where $\kappa(u)$ is the kernel function and $x_1, \ldots, x_n$ are the data points.

Kernel smoothing

Kernel smoothed intensity

$$\widetilde{\lambda}(u) = \sum_{i=1}^{n} \kappa(u - x_i)$$

where $\kappa(u)$ is the kernel function and $x_1, \ldots, x_n$ are the data points.

1. replace each data point by a square of chocolate

Kernel smoothing

Kernel smoothed intensity

$$\widetilde{\lambda}(u) = \sum_{i=1}^{n} \kappa(u - x_i)$$

where $\kappa(u)$ is the kernel function and $x_1, \ldots, x_n$ are the data points.

1. replace each data point by a square of chocolate
2. melt chocolate with hair dryer

Kernel smoothing

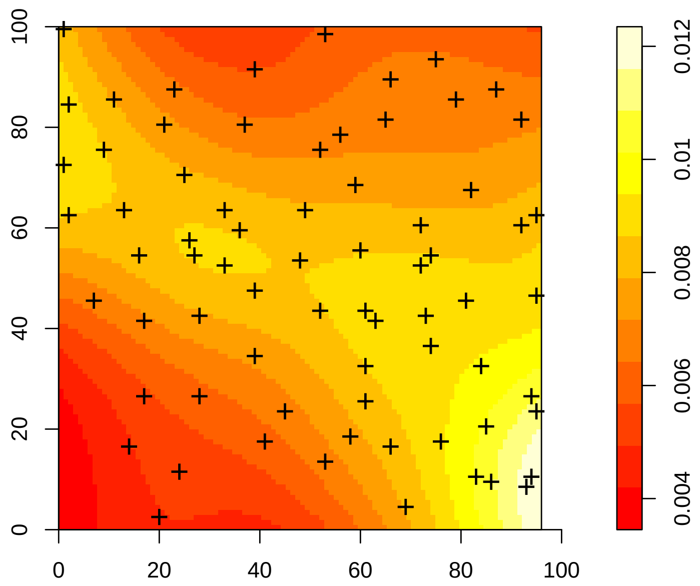
Kernel smoothed intensity

$$\widetilde{\lambda}(u) = \sum_{i=1}^{n} \kappa(u - x_i)$$

where $\kappa(u)$ is the kernel function and $x_1, \ldots, x_n$ are the data points.

1. replace each data point by a square of chocolate
2. melt chocolate with hair dryer
3. resulting landscape is a *kernel smoothed estimate of intensity function*

```
den <- density(P, sigma=15)
plot(den)
plot(P, add=TRUE)
```



A more searching analysis involves fitting *models* that describe how the point pattern intensity $\lambda(u)$ depends on spatial location u or on spatial covariates $Z(u)$.

A more searching analysis involves fitting *models* that describe how the point pattern intensity $\lambda(u)$ depends on spatial location u or on spatial covariates $Z(u)$.

Intensity is modelled using a “log link”.

COMMAND	INTENSITY
`ppm(P, ~1)`	$\log \lambda(u) = \beta_0$

Modelling intensity

COMMAND	INTENSITY
`ppm(P, ~1)`	$\log \lambda(u) = \beta_0$

$\beta_0, \beta_1, \ldots$ denote parameters to be estimated.

COMMAND	INTENSITY
`ppm(P, ~1)`	$\log \lambda(u) = \beta_0$
`ppm(P, ~x)`	$\log \lambda((x, y)) = \beta_0 + \beta_1 x$

$\beta_0, \beta_1, \ldots$ denote parameters to be estimated.

Modelling intensity

COMMAND	INTENSITY
`ppm(P, ~1)`	$\log \lambda(u) = \beta_0$
`ppm(P, ~x)`	$\log \lambda((x, y)) = \beta_0 + \beta_1 x$
`ppm(P, ~x + y)`	$\log \lambda((x, y)) = \beta_0 + \beta_1 x + \beta_2 y$

$\beta_0, \beta_1, \ldots$ denote parameters to be estimated.

```
> ppm(P, ~1)
Stationary Poisson process
Uniform intensity:       0.007
```

```
> ppm(P, ~x+y)
Nonstationary Poisson process
Trend formula: ~x + y
Fitted coefficients for trend formula:
  (Intercept)         x         y
    -5.1237    0.00461   -0.00025
```

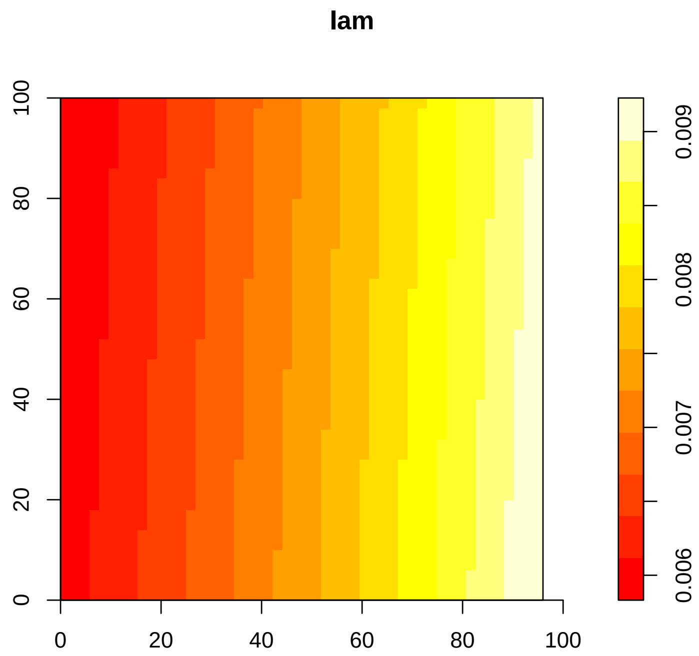
COMMAND	INTENSITY
`ppm(P, ~polynom(x,y,3))`	3rd order polynomial in x and y

Modelling intensity

COMMAND	INTENSITY
`ppm(P, ~polynom(x,y,3))`	3rd order polynomial in x and y
`ppm(P, ~I(y > 18))`	different constants above and below the line $y = 18$

```
fit <- ppm(P, ~x+y)
lam <- predict(fit)
plot(lam)
```

The `predict` method computes fitted values of intensity function $\lambda(u)$ at a grid of locations.



Likelihood ratio test

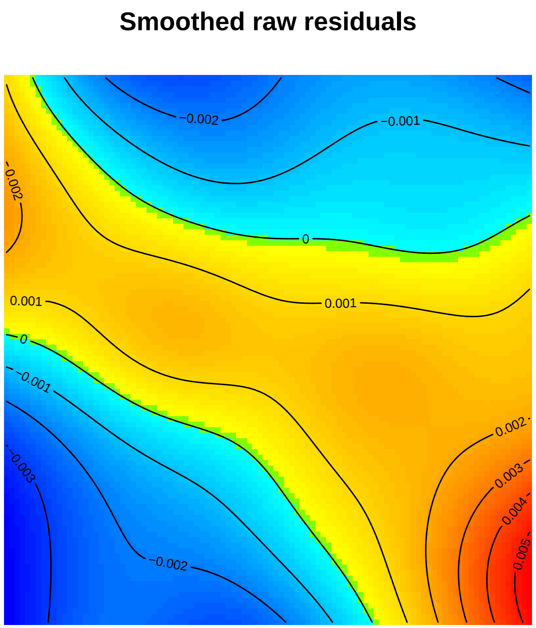
```
fit0 <- ppm(P, ~1)
fit1  <- ppm(P, ~polynom(x,y,2))
anova(fit0, fit1, test="Chi")
```

```
  fit0 <- ppm(P, ~1)
  fit1  <- ppm(P, ~polynom(x,y,2))
  anova(fit0, fit1, test="Chi")
Analysis of Deviance Table

Model 1: .mpl.Y ~ 1
Model 2: .mpl.Y ~ polynom(x, y, 5)
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       699     408.10
2       694     400.62   5     7.48      0.19
```

```
fit0 <- ppm(P, ~1)
fit1  <- ppm(P, ~polynom(x,y,2))
anova(fit0, fit1, test="Chi")
Analysis of Deviance Table

Model 1: .mpl.Y ~ 1
Model 2: .mpl.Y ~ polynom(x, y, 5)
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       699     408.10
2       694     400.62   5     7.48      0.19
```

The p-value 0.19 exceeds 0.05 so the log-quadratic spatial trend is *not significant*.

`diagnose.ppm(fit0, which="smooth")`



Spatial covariates

Spatial covariates

A *spatial covariate* is a function $Z(u)$ of spatial location.

Spatial covariates

A *spatial covariate* is a function $Z(u)$ of spatial location.

- geographical coordinates

Spatial covariates

A *spatial covariate* is a function $Z(u)$ of spatial location.

- geographical coordinates
- terrain altitude

Spatial covariates

A *spatial covariate* is a function $Z(u)$ of spatial location.

- geographical coordinates
- terrain altitude
- soil pH

Spatial covariates

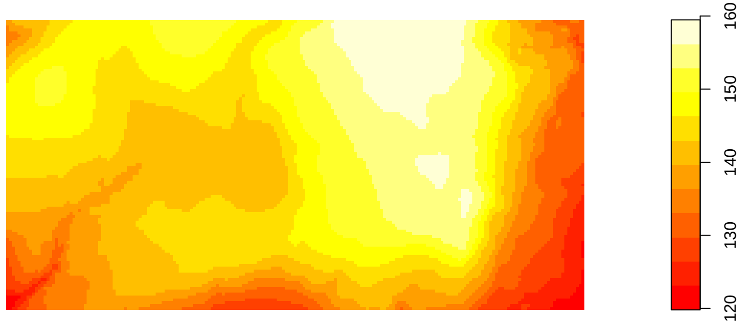
A *spatial covariate* is a function $Z(u)$ of spatial location.

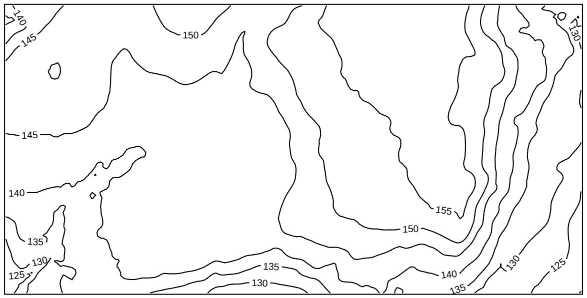
- geographical coordinates
- terrain altitude
- soil pH
- distance from location u to another feature

Spatial covariates

A *spatial covariate* is a function $Z(u)$ of spatial location.

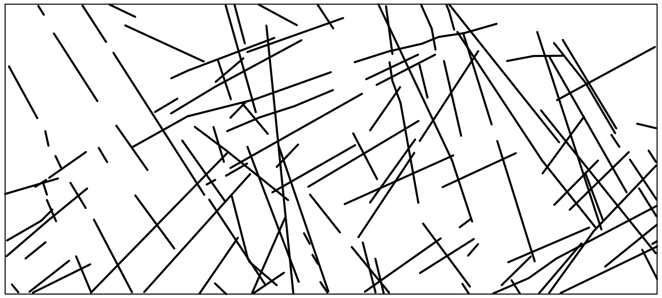
- geographical coordinates
- terrain altitude
- soil pH
- distance from location u to another feature





Covariates

Covariate data may be another *spatial pattern* such as another point pattern, or a line segment pattern:



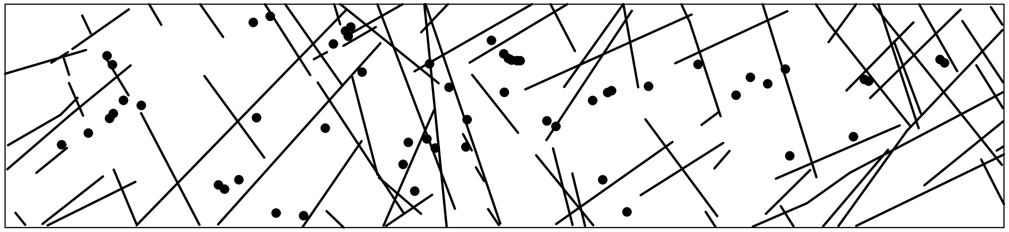
Covariate effects

For a point pattern dataset with covariate data, we typically

- investigate whether the intensity depends on the covariates
- allow for covariate effects on intensity before studying dependence between points

Example: Queensland copper data

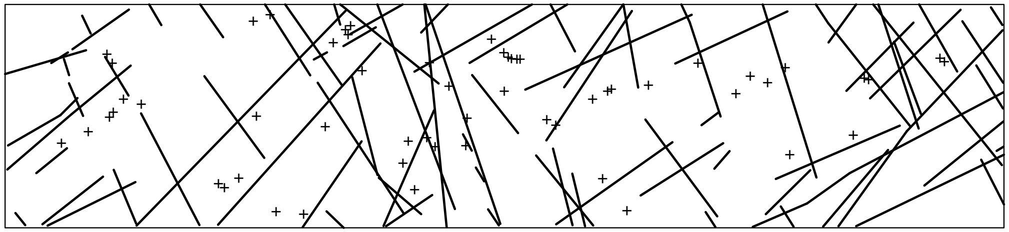
A intensive mineralogical survey yields a map of copper deposits (essentially pointlike at this scale) and geological faults (straight lines). The faults can easily be observed from satellites, but the copper deposits are hard to find.



Main question: whether the faults are 'predictive' for copper deposits (e.g. copper less/more likely to be found near faults).

Copper data

```
data(copper)
P <- copper$SouthPoints
Y <- copper$SouthLines
plot(P)
plot(Y, add=TRUE)
```



Copper data

For analysis, we need a value $Z(u)$ defined at each location u.

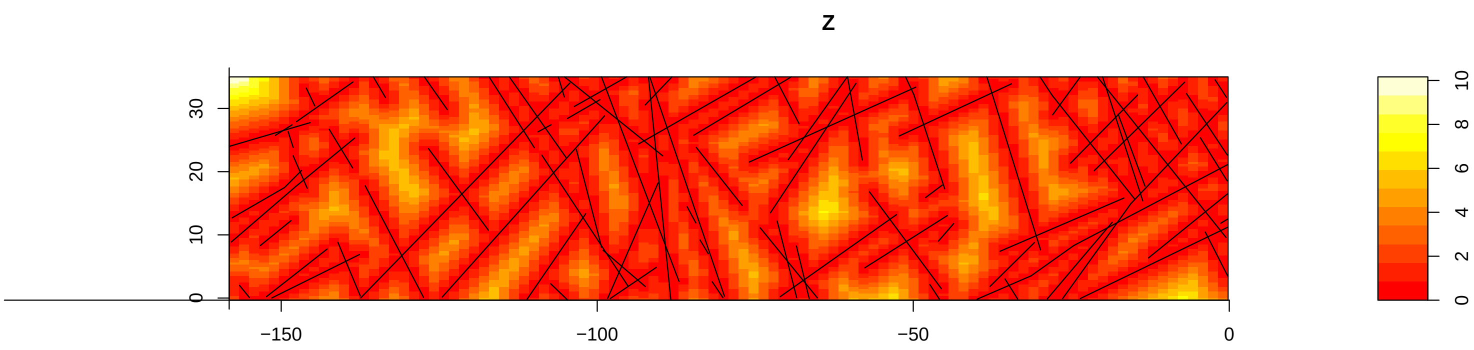
Copper data

For analysis, we need a value $Z(u)$ defined at each location u.
Example: $Z(u) =$ distance from u to nearest line.

For analysis, we need a value $Z(u)$ defined at each location u.

Example: $Z(u) =$ distance from u to nearest line.

```
D <- distmap(Y)
plot(D)
```



Lurking variable plot

We want to determine whether intensity depends on a spatial covariate Z.

Lurking variable plot

We want to determine whether intensity depends on a spatial covariate Z.
Plot $C(z)$ against z, where $C(z)$ = fraction of data points x_i for which $Z(x_i) \leq z$.

Lurking variable plot

We want to determine whether intensity depends on a spatial covariate Z.
Plot $C(z)$ against z, where $C(z)$ = fraction of data points x_i for which $Z(x_i) \leq z$.
Also plot $C_0(z)$ against z, where $C_0(z) =$ fraction of area of study region where $Z(u) \leq z$.

Lurking variable plot

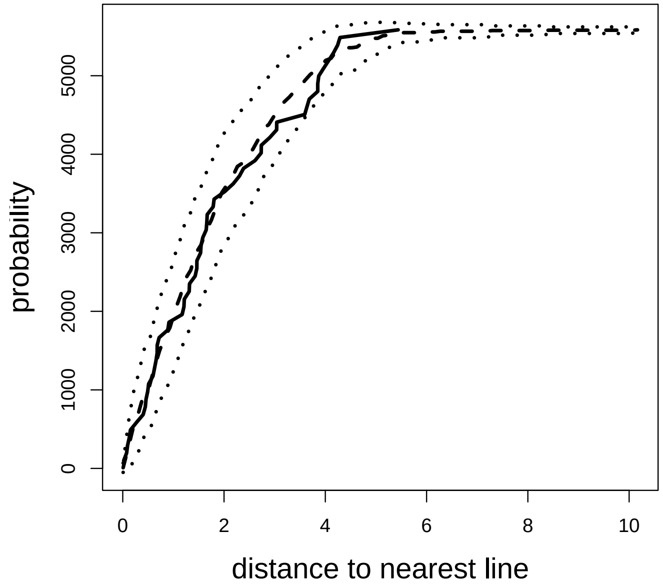
We want to determine whether intensity depends on a spatial covariate Z.
Plot $C(z)$ against z, where $C(z)$ = fraction of data points x_i for which $Z(x_i) \leq z$.
Also plot $C_0(z)$ against z, where $C_0(z) =$ fraction of area of study region where $Z(u) \leq z$.

```
lurking(ppm(P), Z)
```

Lurking variable plot

We want to determine whether intensity depends on a spatial covariate Z.
Plot $C(z)$ against z, where $C(z)$ = fraction of data points x_i for which $Z(x_i) \leq z$.
Also plot $C_0(z)$ against z, where $C_0(z) =$ fraction of area of study region where $Z(u) \leq z$.

```
lurking(ppm(P), Z)
```



Kolmogorov-Smirnov test

Formal test of agreement between $C(z)$ and $C_0(z)$.

Kolmogorov-Smirnov test

Formal test of agreement between $C(z)$ and $C_0(z)$.

```
> kstest(P, Z)

        Spatial Kolmogorov-Smirnov test of CSR

data:  covariate 'Z' evaluated at points of 'P'
         and transformed to uniform distribution under CSR
D = 0.1163, p-value = 0.3939
alternative hypothesis: two-sided
```

```
D <- distmap(Y)
ppm(P, ~Z, covariates=list(Z=D))
```

Fits the model

$$\log \lambda(u) = \beta_0 + \beta_1 Z(u)$$

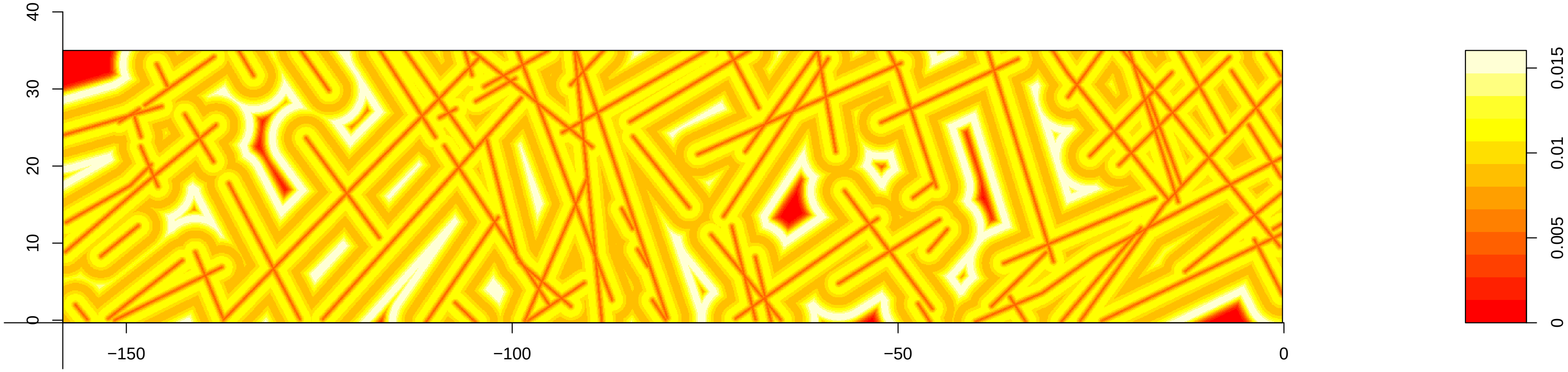
where $Z(u)$ is the distance from u to the nearest line segment.

```
D <- distmap(Y)
ppm(P, ~polynom(Z,5), covariates=list(Z=D))
```

fits a model in which $\log \lambda(u)$ is a 5th order polynomial function of $Z(u)$.

Copper data

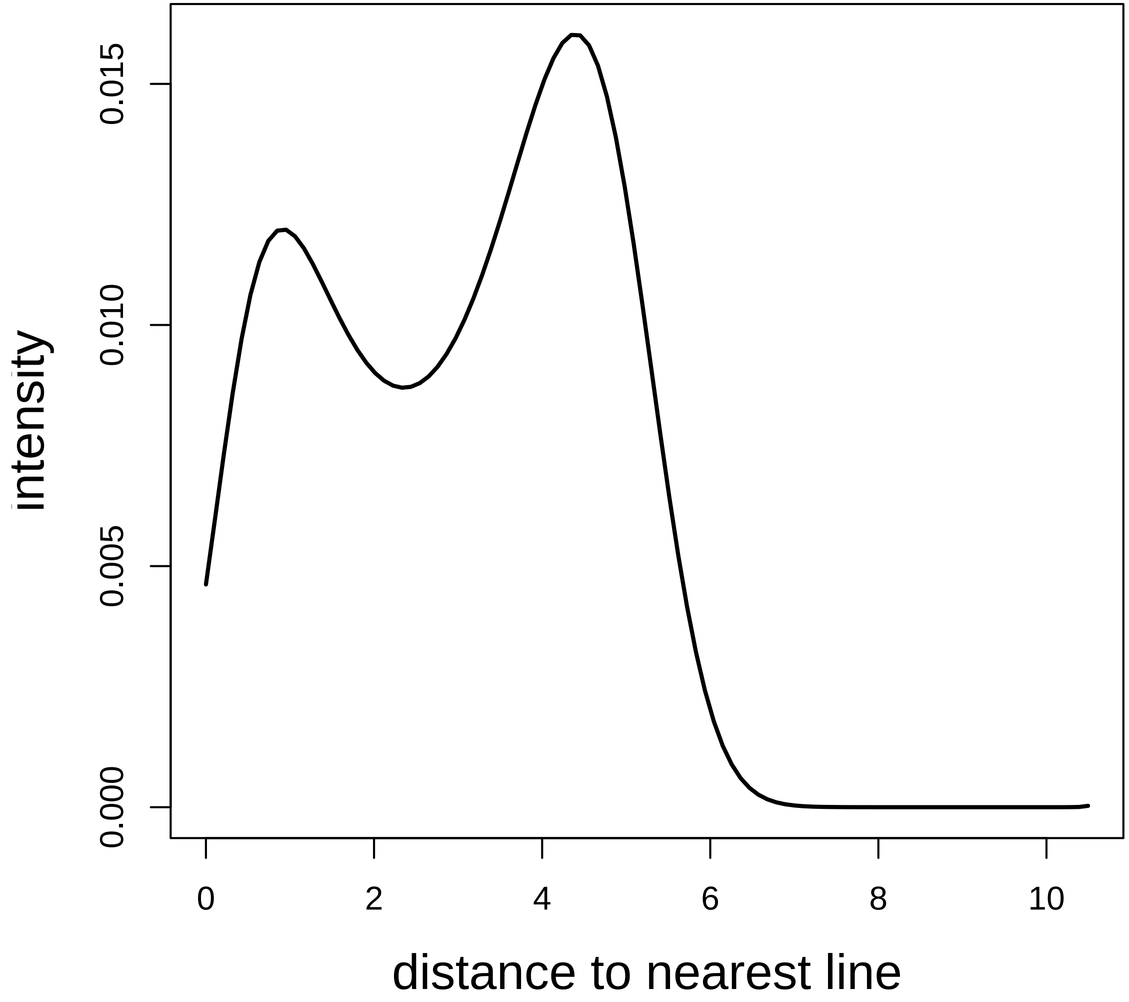
```
fit <- ppm(P, ~polynom(Z,5), covariates=list(Z=D))
plot(predict(fit))
```



```
Dr <- summary(D)$range
Dvalues <- seq(Dr[1], Dr[2], length=100)
fakeZ <- data.frame(Z=Dvalues)
fakexy <- data.frame(x=rep(0,100), y=rep(0,100))
lambda <- predict(fit, locations=fakexy, covariates=fakeZ)
plot(Dvalues, lambda, type="l")
```

plots fitted curve of λ against Z.

Copper data



intensity
0.000
0.005
0.010
0.015
0
2
4
6
8
10
distance to nearest line

Likelihood ratio test

```
fit0 <- ppm(P, ~1)
fit1  <- ppm(P, ~polynom(Z,5), covariates=list(Z=D))
anova(fit0, fit1, test="Chi")
```

Likelihood ratio test

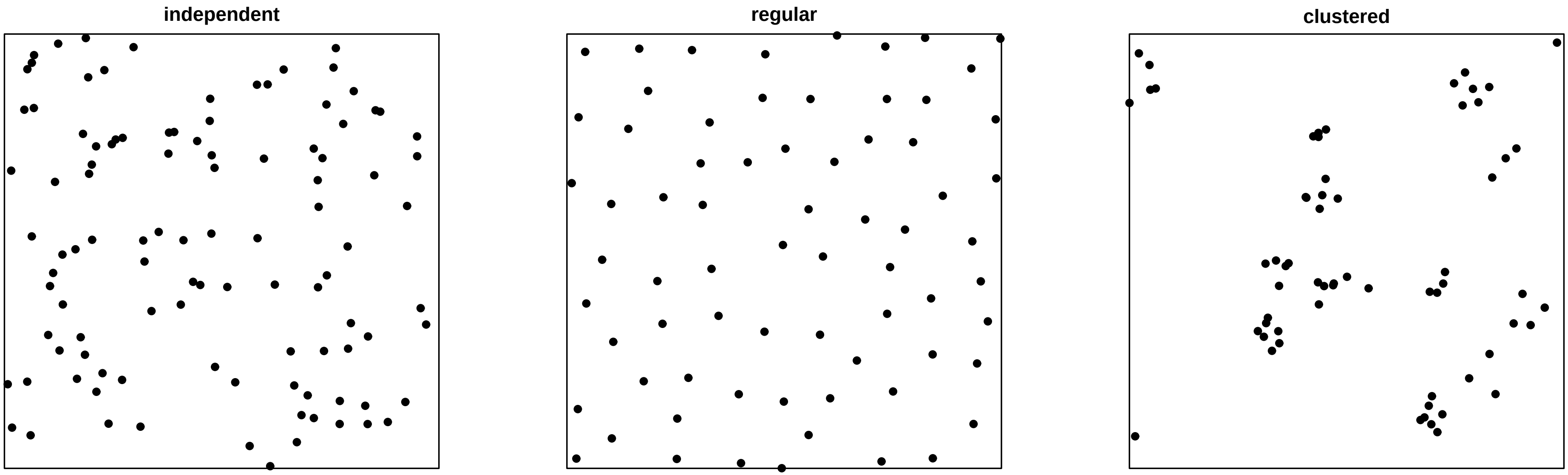
```
fit0 <- ppm(P, ~1)
fit1  <- ppm(P, ~polynom(Z,5), covariates=list(Z=D))
anova(fit0, fit1, test="Chi")
Analysis of Deviance Table

Model 1: .mpl.Y ~ 1
Model 2: .mpl.Y ~ polynom(Z, 5)
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       682     372.32
2       677     370.04   5     2.28      0.81
```

```
fit0 <- ppm(P, ~1)
fit1  <- ppm(P, ~polynom(Z,5), covariates=list(Z=D))
anova(fit0, fit1, test="Chi")
Analysis of Deviance Table

Model 1: .mpl.Y ~ 1
Model 2: .mpl.Y ~ polynom(Z, 5)
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       682     372.32
2       677     370.04   5     2.28      0.81
```

The p-value 0.81 exceeds 0.05 so the 5th order polynomial is *not significant*.

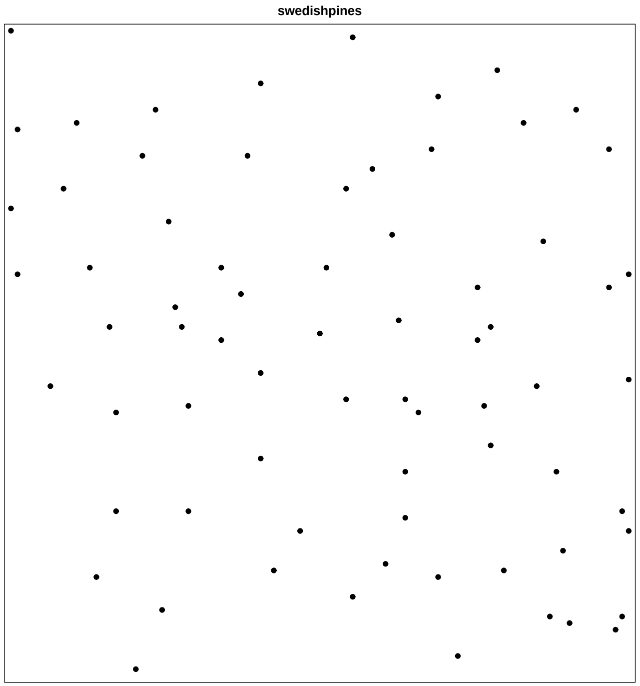
Interaction

‘Interpoint interaction’ is stochastic dependence between the points in a point pattern. Usually we expect dependence to be strongest between points that are close to one another.



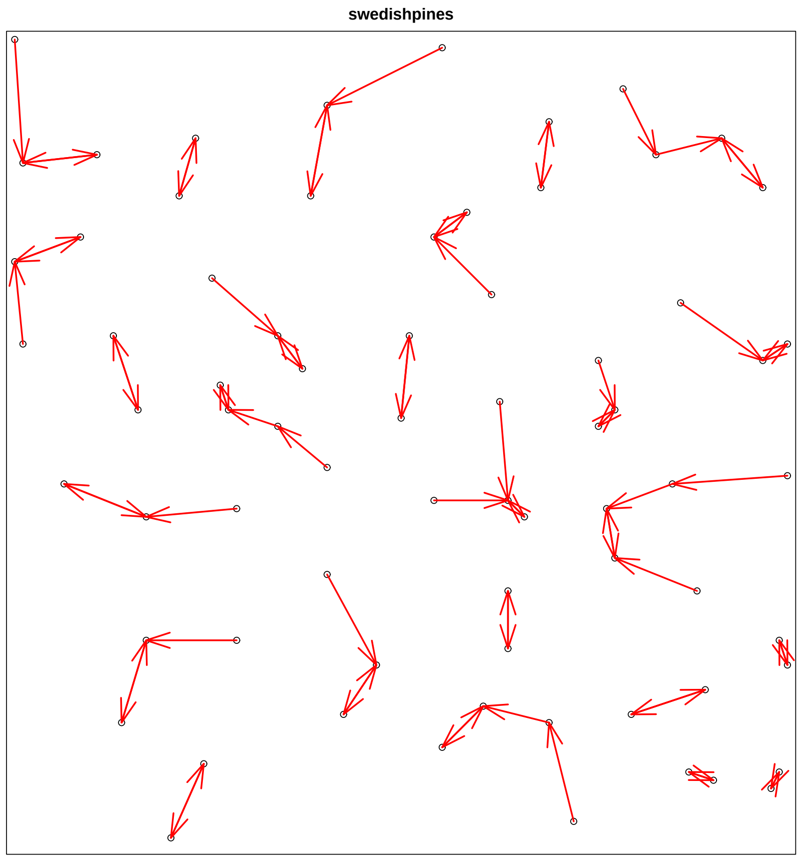
Example

Example: spacing between points in Swedish Pines data



Example

nearest neighbour distance = distance from a given point to the nearest other point



Example

Summary approach:

Example

Summary approach:

1. calculate average nearest-neighbour distance

Example

Summary approach:

1. calculate average nearest-neighbour distance
2. divide by the value expected for a completely random pattern.

Clark & Evans (1954)

Example

Summary approach:

1. calculate average nearest-neighbour distance
2. divide by the value expected for a completely random pattern.

Clark & Evans (1954)

```
> mean(nndist(swedishpines))
[1] 7.90754
> clarkevans(swedishpines)
   naive Donnelly      cdf
1.360082 1.291069 1.322862
```

Example

Summary approach:

1. calculate average nearest-neighbour distance
2. divide by the value expected for a completely random pattern.

Clark & Evans (1954)

```
> mean(nndist(swedishpines))
[1] 7.90754
> clarkevans(swedishpines)
   naive Donnelly      cdf
1.360082 1.291069 1.322862
```

Value greater than 1 suggests a regular pattern.

Example

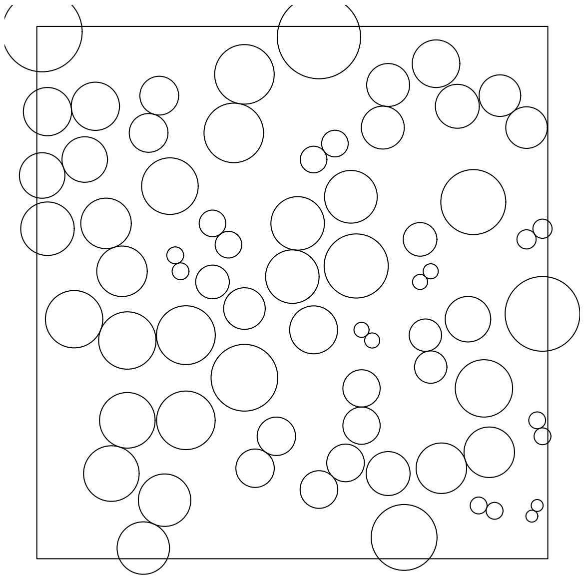
Exploratory approach:

Example

Exploratory approach:

- plot NND for each point

```
P <- swedishpines
marks(P) <- nndist(P)
plot(P, markscale=0.5)
```



Example

Exploratory approach:

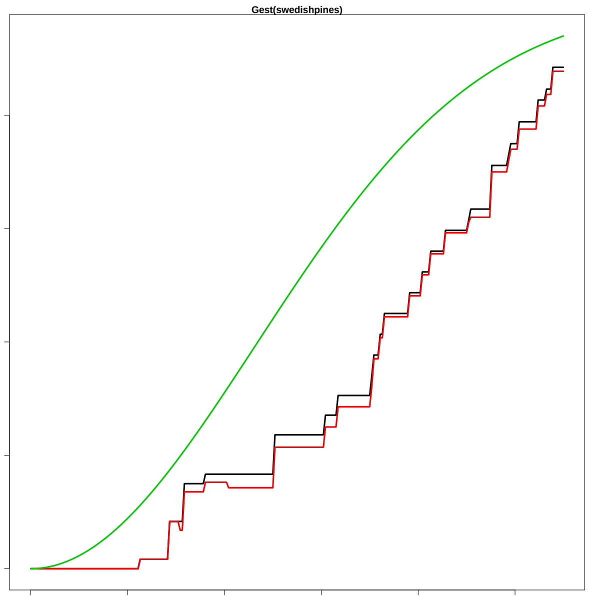
- plot NND for each point

Example

Exploratory approach:

- plot NND for each point
- look at empirical distribution of NND's

```
plot(Gest(swedishpines))
```



Example

Modelling approach:

Example

Modelling approach:

- Fit a stochastic model to the point pattern, with likelihood based on the NND's.

Example

Modelling approach:

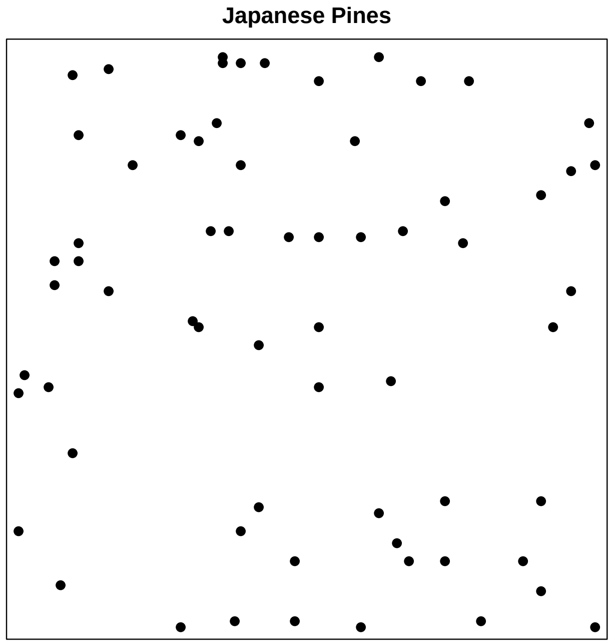
- Fit a stochastic model to the point pattern, with likelihood based on the NND's.

```
> ppm(P, ~1, Geyer(4,1))

Stationary Geyer saturation process
First order term:
      beta
0.00971209
Fitted interaction parameter gamma: 0.6335
```

Example: Japanese pines

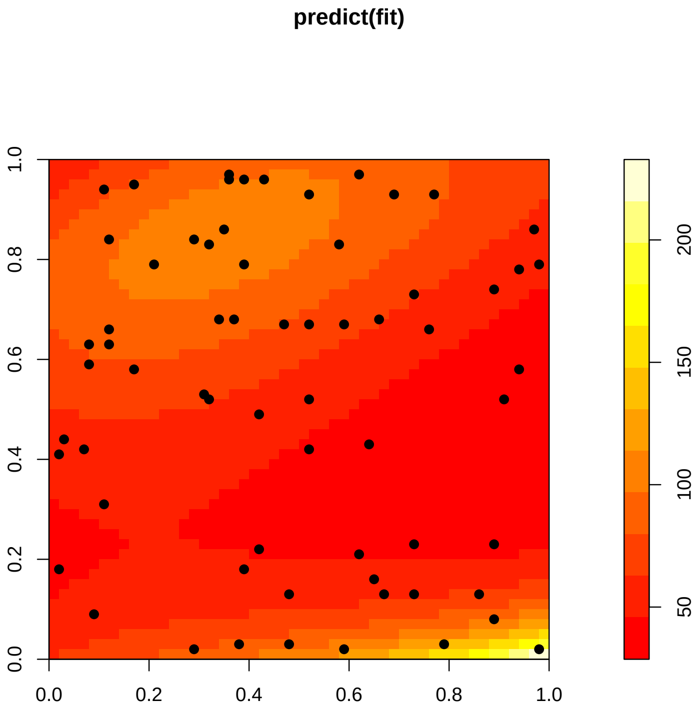
Locations of 65 saplings of Japanese pine in a 5.7×5.7 metre square sampling region in a natural stand.

```
data(japanesepines)
J <- japanesepines
plot(J)
```



Japanese Pines

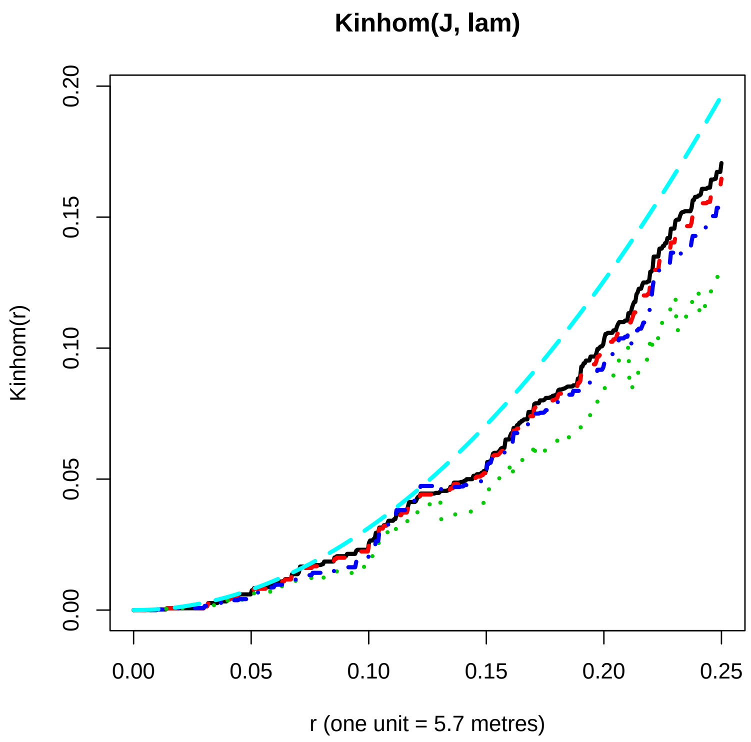
```
fit <- ppm(J, ~polynom(x,y,3))
plot(predict(fit))
plot(J, add=TRUE)
```



Adjusting for inhomogeneity

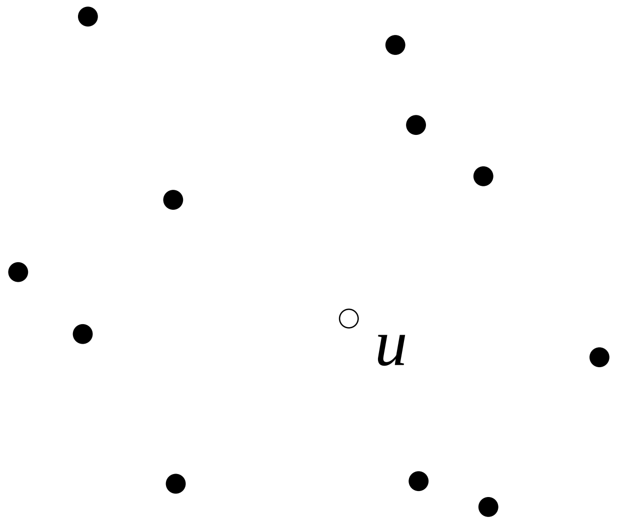
If the intensity function $\lambda(u)$ is known, or estimated from data, then some statistics can be adjusted by counting each data point x_i with a weight $w_i = 1/\lambda(x_i)$.

```
lam <- predict(fit)
plot(Kinhom(J, lam))
```



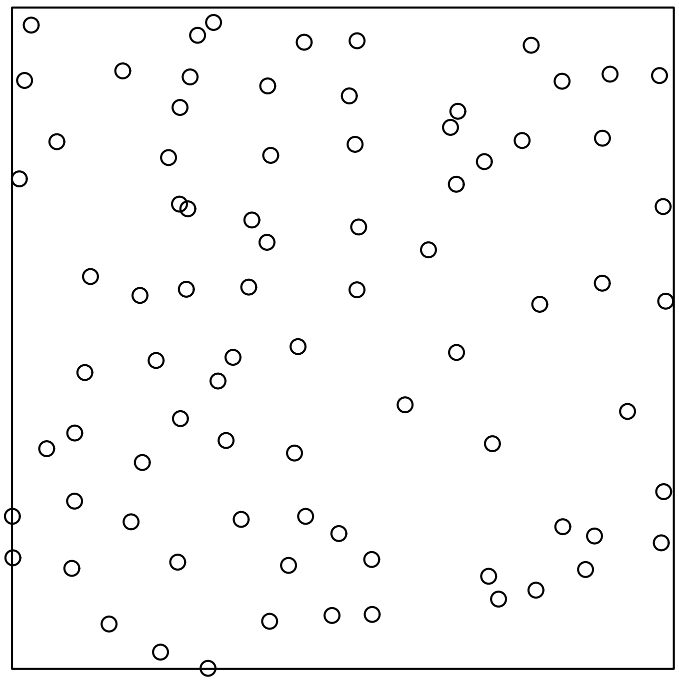
Conditional intensity

A point process model can also be defined through its *conditional intensity* $\lambda(u \mid \mathbf{x})$.
This is essentially the conditional probability of finding a point of the process at the location u, given complete information about the rest of the process $\mathbf{x}$.

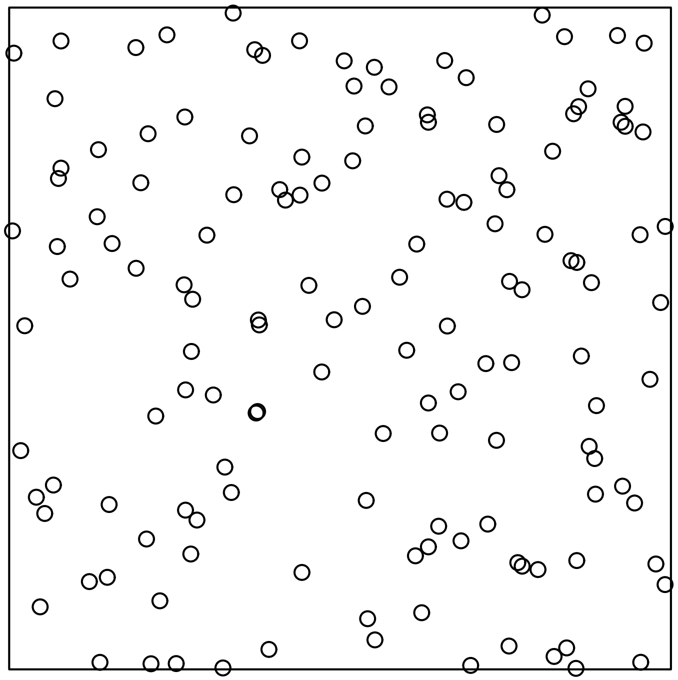


Strauss process

Strauss($\gamma = 0.2$)



Strauss($\gamma = 0.7$)



Fitting Gibbs models

The command `ppm` will also fit Gibbs models, using the technique of ‘maximum pseudolikelihood’.

Fitting Gibbs models

The command `ppm` will also fit Gibbs models, using the technique of 'maximum pseudolikelihood'.

```
data(swedishpines)
ppm(swedishpines, ~1, Strauss(r=7))
```

Fitting Gibbs models

The command `ppm` will also fit Gibbs models, using the technique of ‘maximum pseudolikelihood’.

```
data(swedishpines)
ppm(swedishpines, ~1, Strauss(r=7))

Stationary Strauss process

First order term:
      beta
0.02583902

Interaction: Strauss process
interaction distance:   7
Fitted interaction parameter gamma:     0.1841
```

Fitting Gibbs models

The model can include both spatial trend and interpoint interaction.

Fitting Gibbs models

The model can include both spatial trend and interpoint interaction.

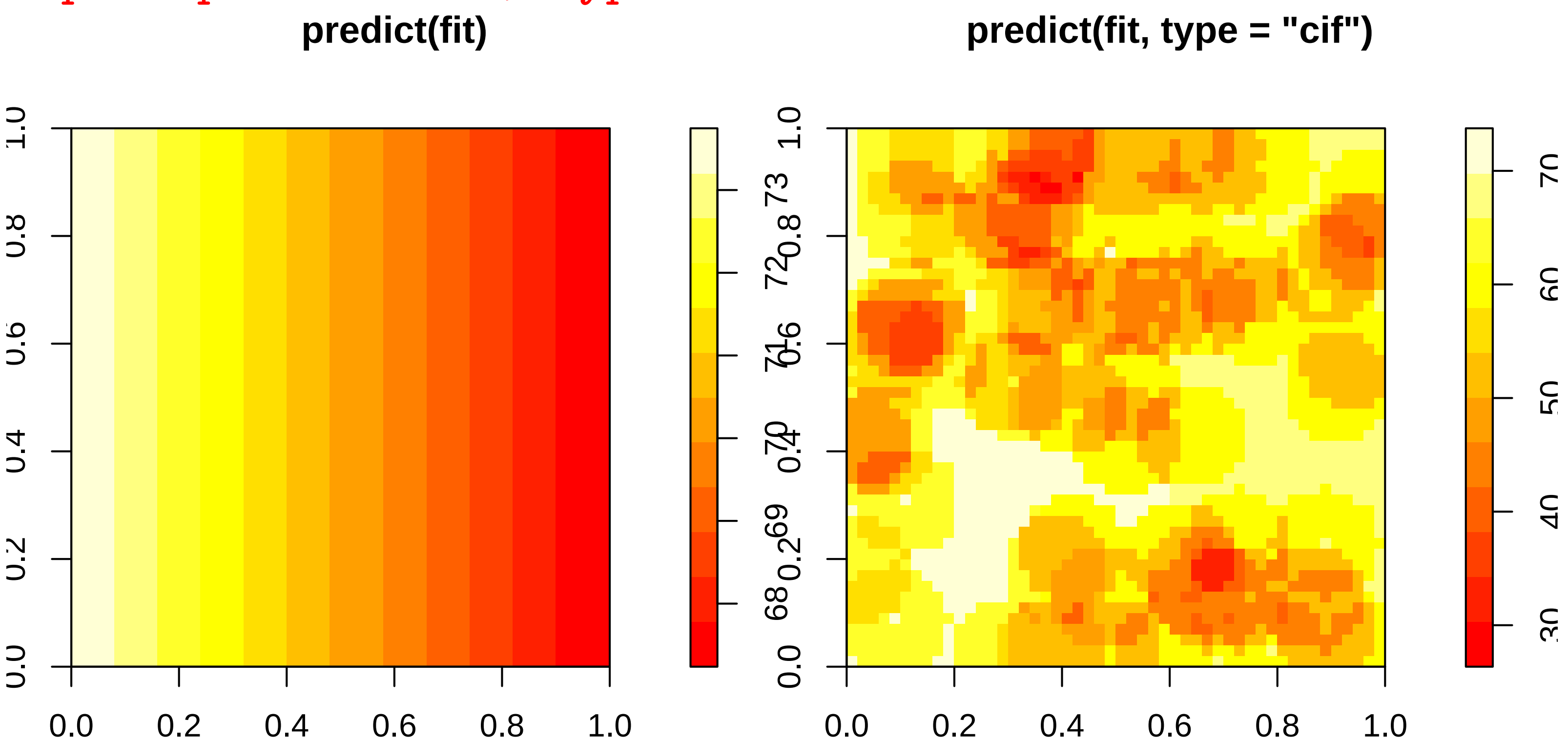
```
data(japanesepines)
ppm(japanesepines, ~polynom(x,y,3), Strauss(r=0.07))
```

The model can include both spatial trend and interpoint interaction.

```
data(japanesepines)
ppm(japanesepines, ~polynom(x,y,3), Strauss(r=0.07))
Nonstationary Strauss process
Trend formula: ~polynom(x, y, 3)
Fitted coefficients for trend formula:
          (Intercept)     polynom(x, y, 3)[x]     polynom(x, y, 3)[y]
            0.4925368              22.0485400              -9.1889134
  polynom(x, y, 3)[x^2]   polynom(x, y, 3)[x.y]   polynom(x, y, 3)[y^2]
          -14.6524958             -41.0222232              50.2099917
  polynom(x, y, 3)[x^3] polynom(x, y, 3)[x^2.y] polynom(x, y, 3)[x.y^2]
            3.4935300               5.4524828              23.9209323
  polynom(x, y, 3)[y^3]
          -38.3946389
Interaction: Strauss process
interaction distance:   0.1
Fitted interaction parameter gamma:     0.5323
```

When we `plot` or `predict` a fitted Gibbs model, the first order trend $\beta(u)$ and/or the conditional intensity $\lambda(u \mid \mathbf{x})$ are plotted.

```
fit <- ppm(japanesepines, ~x, Strauss(r=0.1))
plot(predict(fit))
plot(predict(fit, type="cif"))
```



Simulating the fitted model

A fitted Gibbs model can be simulated automatically using the Metropolis-Hastings algorithm (which only requires the conditional intensity).

Simulating the fitted model

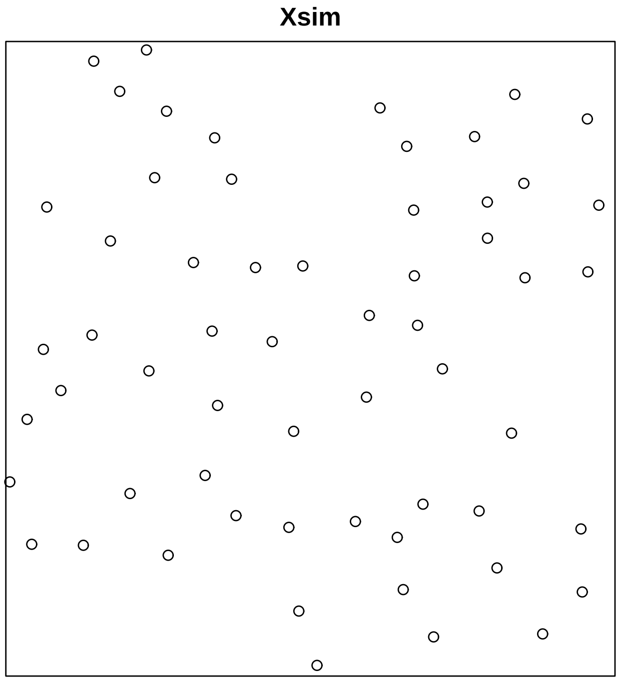
A fitted Gibbs model can be simulated automatically using the Metropolis-Hastings algorithm (which only requires the conditional intensity).

```
fit <- ppm(swedishpines, ~1, Strauss(r=7))
Xsim <- rmh(fit)
plot(Xsim)
```

Simulating the fitted model

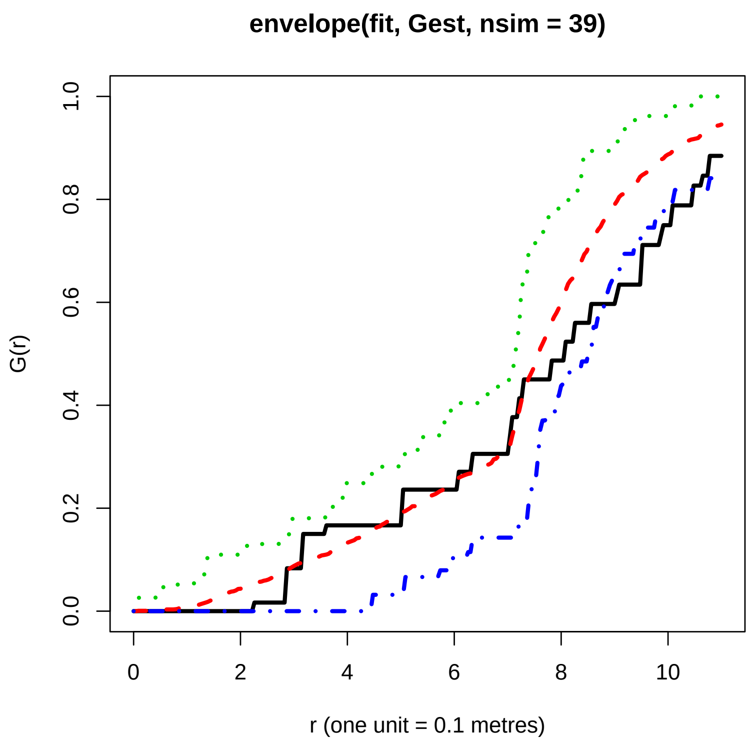
A fitted Gibbs model can be simulated automatically using the Metropolis-Hastings algorithm (which only requires the conditional intensity).

```
fit <- ppm(swedishpines, ~1, Strauss(r=7))
Xsim <- rmh(fit)
plot(Xsim)
```



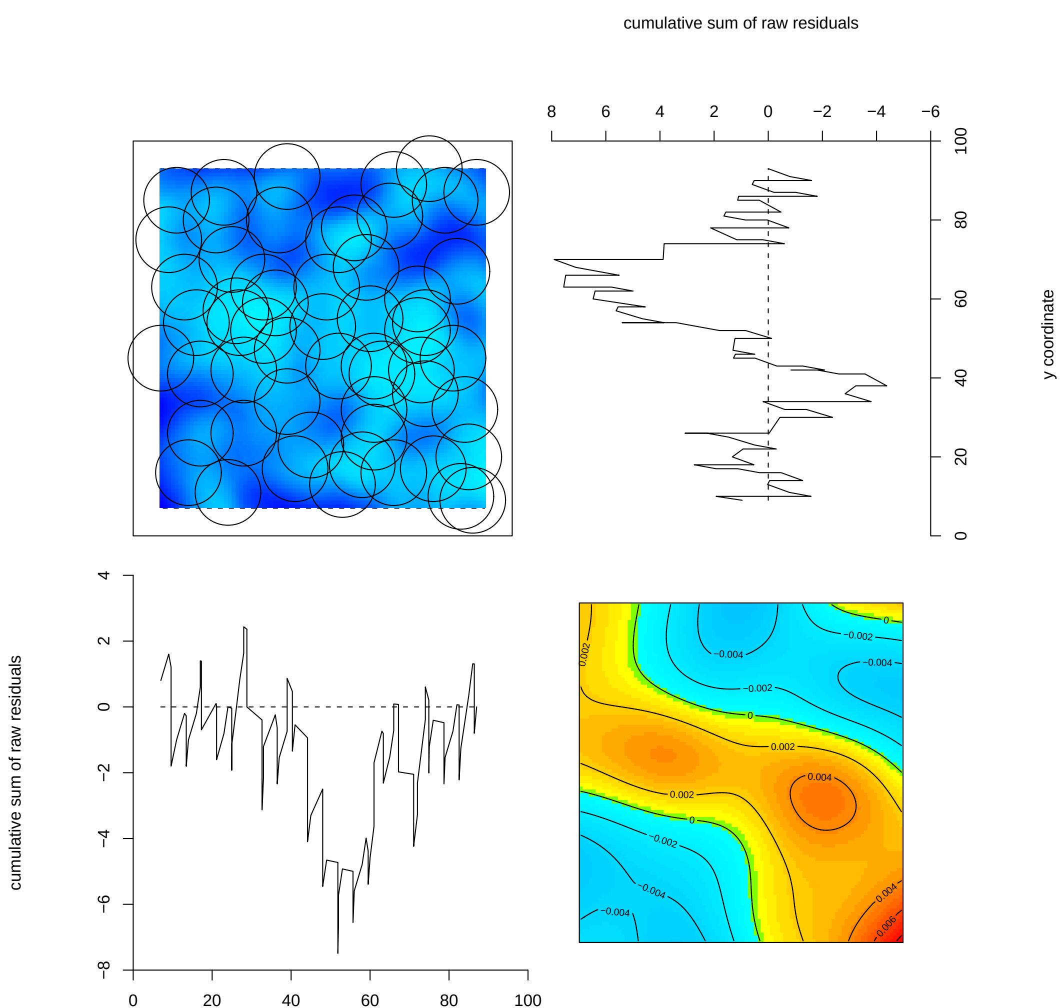
Tests of goodness-of-fit can be performed by simulating from the fitted model.

```
plot(envelope(fit, Gest, nsim=19))
```



More powerful diagnostics are available.

```
diagnose.ppm(fit)
```



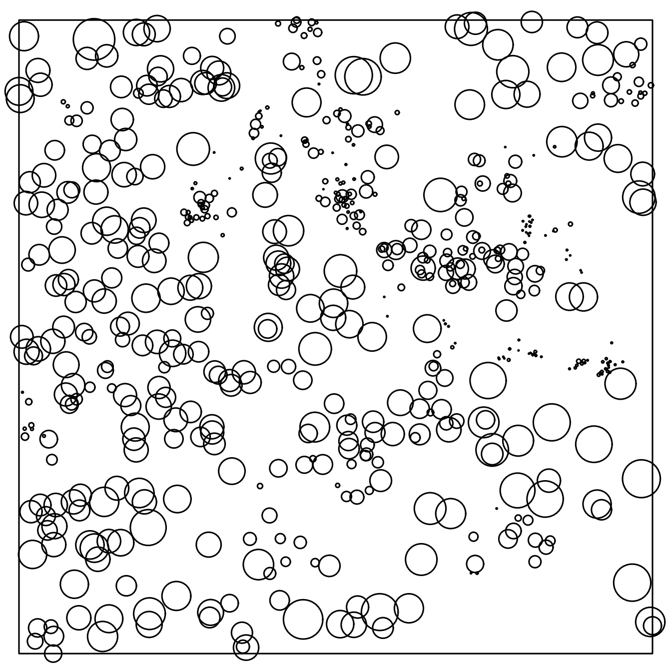
Marks

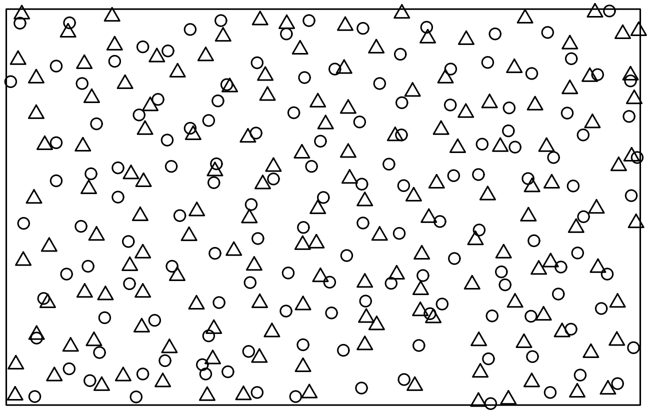
Marks

Each point in a spatial point pattern may carry additional information called a 'mark'. It may be

a continuous variate: tree diameter, tree height

a categorical variate: label classifying the points into two or more different types (on/off, case/control, species, colour)



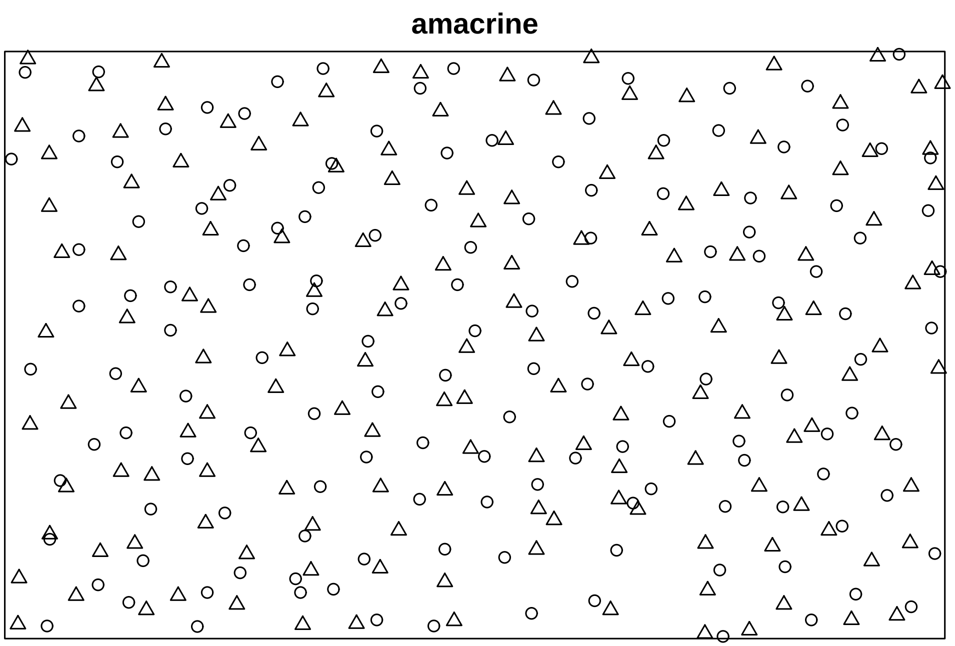


In `spatstat` version 1, the mark attached to each point must be a *single* value.

Categorical marks

A point pattern with categorical marks is usually called “multi-type”.

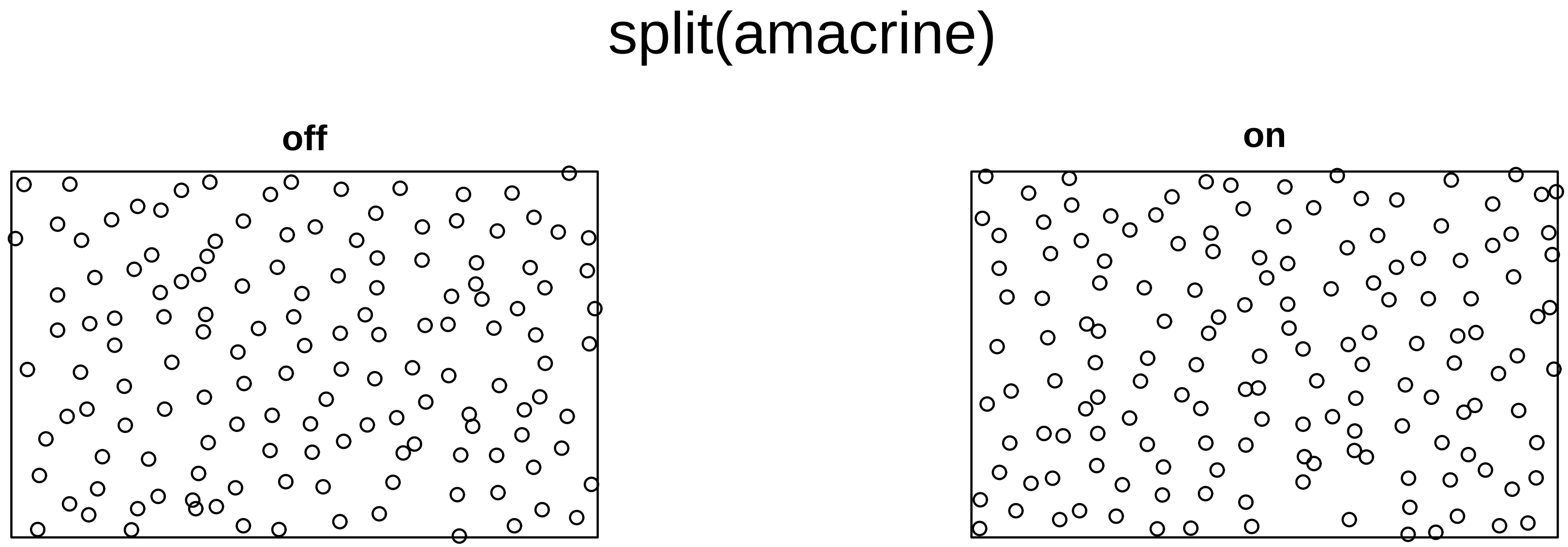
```
> data(amacrine)
> amacrine
marked planar point pattern: 294 points
multitype, with levels = off    on
window: rectangle = [0, 1.6012] x [0, 1] units (one unit = 662 microns)
> plot(amacrine)
```



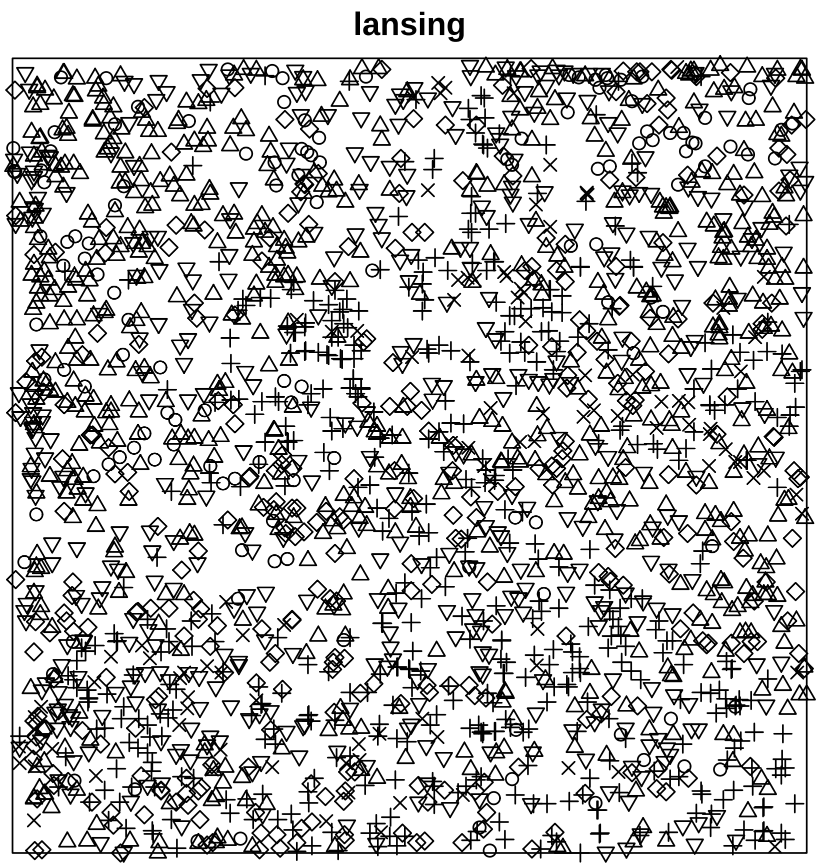
```
summary(amacrine)
```

```
summary(amacrine)

Marked planar point pattern: 294 points
Average intensity 184 points per square unit (one unit = 662 microns)
Multitype:
    frequency proportion intensity
off       142      0.483      88.7
on        152      0.517      94.9

Window: rectangle = [0, 1.6012] x [0, 1] units
Window area =  1.60121 square units
Unit of length: 662 microns
```

`plot(split(amacrine))`

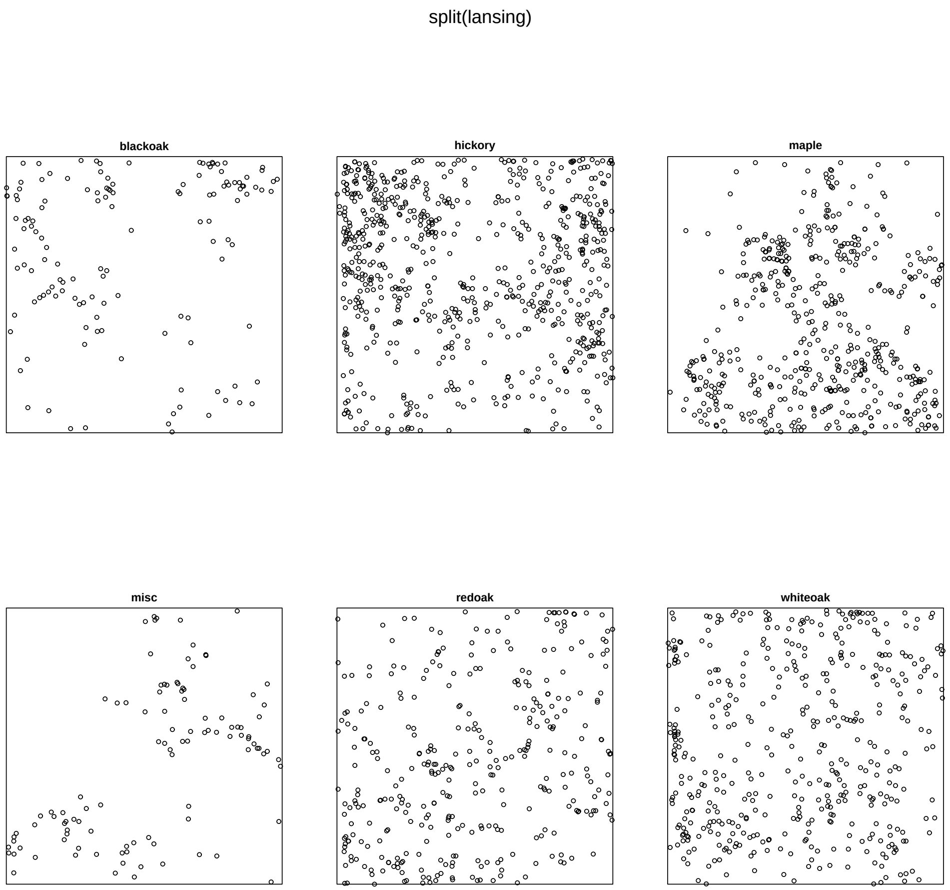


```
data(lansing)
summary(lansing)
plot(lansing)
```



“Segregation” occurs when the intensity depends on the mark (i.e. on the type of point).

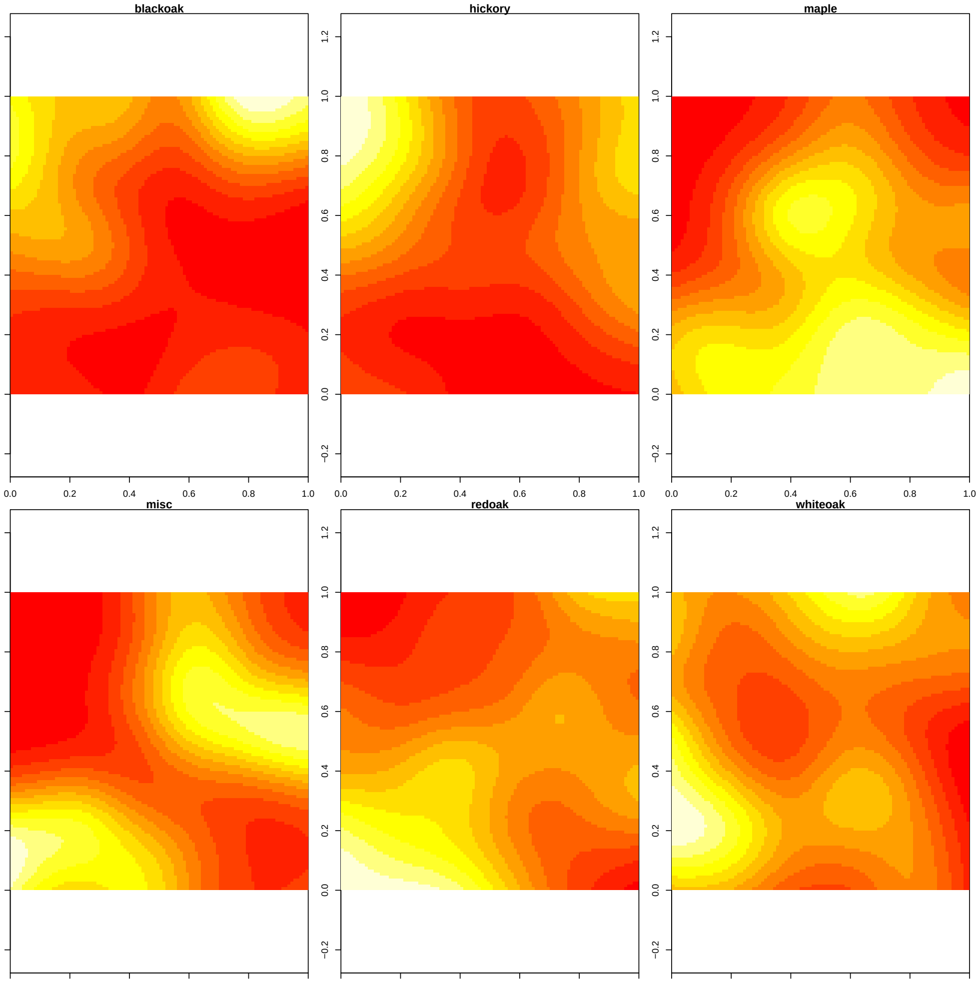
```
plot(split(lansing))
```



Intensity of multitype patterns

Let $\lambda(u, m)$ be the intensity function for points of type m at location u. This can be estimated by kernel smoothing the data points of type m.

```
plot(density(split(lansing)))
```



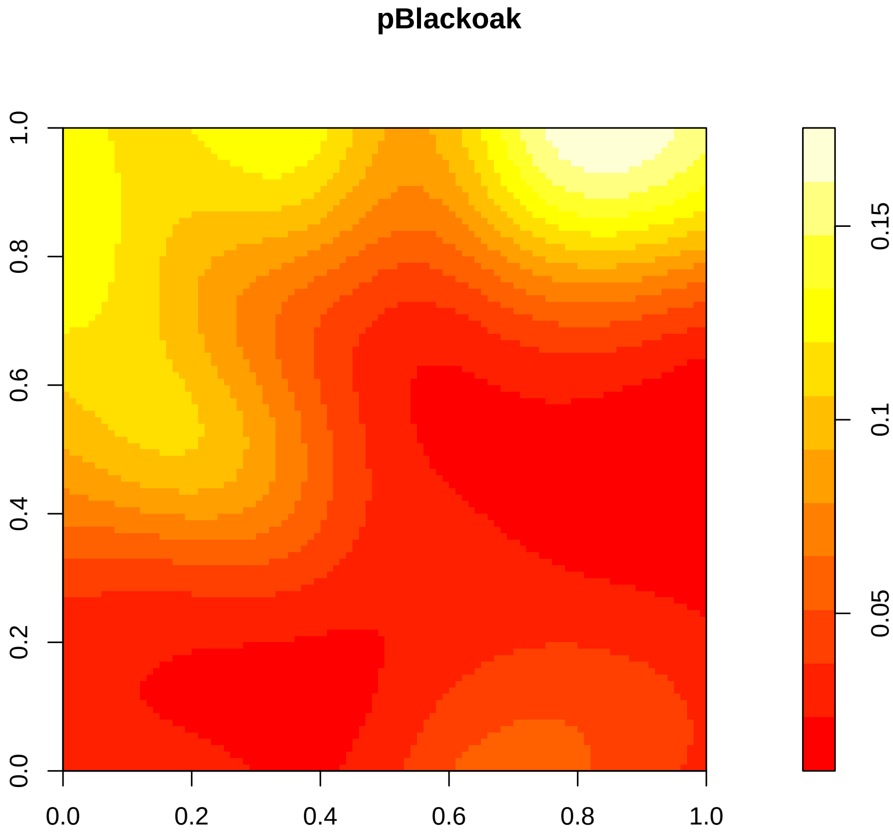
Segregation

The probability that a point at location u has mark m is

$$p(m \mid u) = \frac{\lambda(u, m)}{\lambda(u)}$$

where $\lambda(u) = \sum_m \lambda(u, m)$ is the intensity function of points of all types.

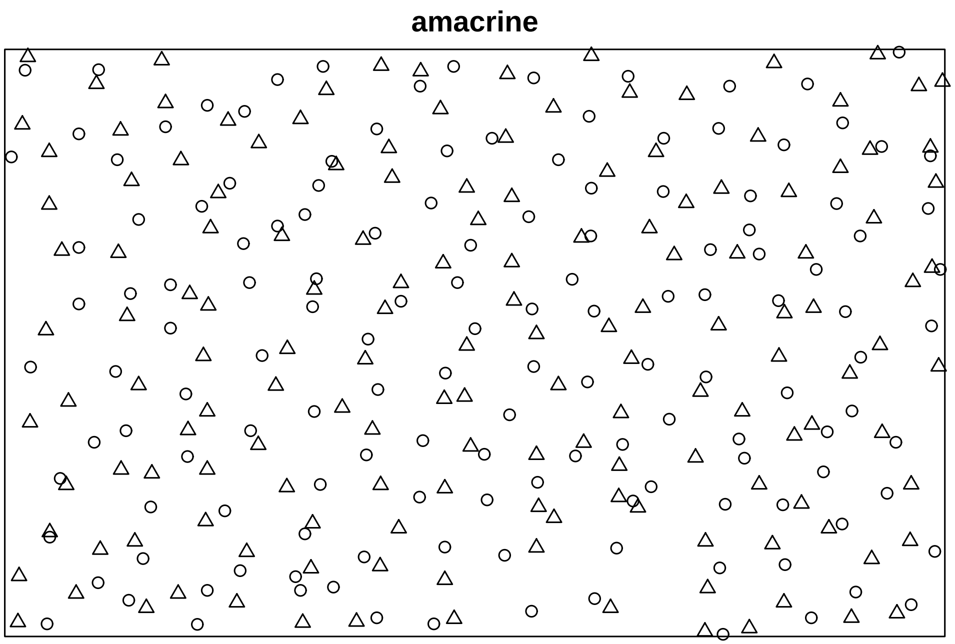
```
D <- density(lansing)
Y <- density(split(lansing))
Dblackoak <- Y$blackoak
pBlackoak <- eval.im(Dblackoak/D)
plot(pBlackoak)
```



Interaction between types

Interaction between types

In a multitype point pattern, there may be interaction between the points of *different* types, or between points of the *same* type.



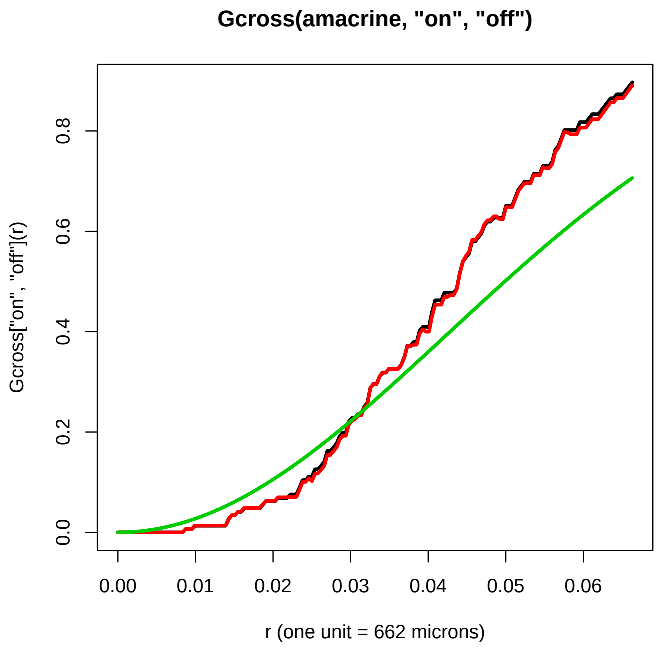
Bivariate G-function

Assume the points of type i have uniform intensity λ_i, for all i.
For two given types i and j, the bivariate G-function G_{ij} is

$$G_{ij}(r) = P(R_{ij} \leq r)$$

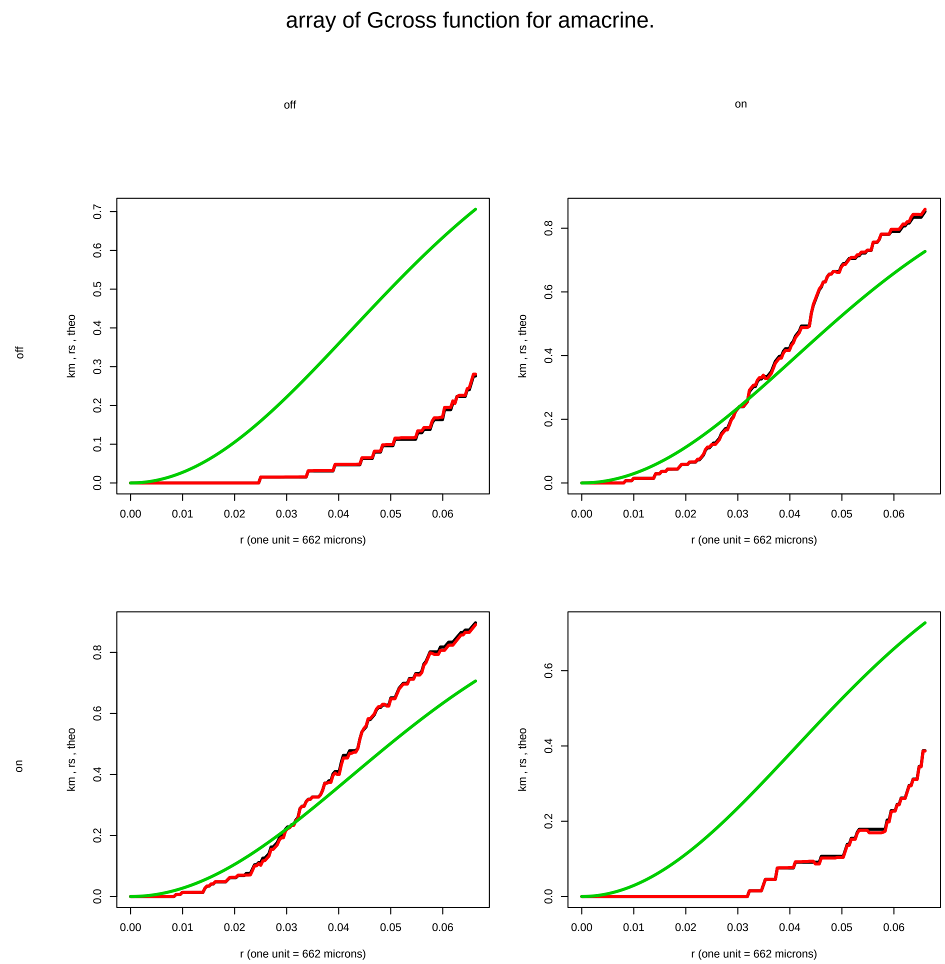
where R_{ij} is the distance from a typical point of type i to the nearest point of type j.

```
plot(Gcross(amacrine, "on", "off"))
```



Bivariate G-function

```
plot(alltypes(amacrine, Gcross))
```



Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant.

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x, y), m) = \beta_m + \alpha x$

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x, y), m) = \beta_m + \alpha x$ Common spatial trend

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x, y), m) = \beta_m + \alpha x$ Common spatial trend Different overall intensity for each type.

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x, y), m) = \beta_m + \alpha x$ Common spatial trend Different overall intensity for each type.
`ppm(X, ~marks + x + marks:x)`	

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u,m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u,m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x,y),m) = \beta_m + \alpha x$ Common spatial trend Different overall intensity for each type.
`ppm(X, ~marks + x + marks:x)` `ppm(X, ~marks * x)`	equivalent to

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x, y), m) = \beta_m + \alpha x$ Common spatial trend Different overall intensity for each type.
`ppm(X, ~marks + x + marks:x)` `ppm(X, ~marks * x)`	equivalent to $\log \lambda((x, y), m) = \beta_m + \alpha_m x$

Fitting Poisson models

For a *multitype* point pattern:

COMMAND	INTERPRETATION
`ppm(X, ~1)`	$\log \lambda(u, m) = \beta$ constant. Equal intensity for points of each type.
`ppm(X, ~marks)`	$\log \lambda(u, m) = \beta_m$ Different constant intensity for points of each type.
`ppm(X, ~marks + x)`	$\log \lambda((x, y), m) = \beta_m + \alpha x$ Common spatial trend Different overall intensity for each type.
`ppm(X, ~marks + x + marks:x)` `ppm(X, ~marks * x)`	equivalent to $\log \lambda((x, y), m) = \beta_m + \alpha_m x$ Different spatial trends for each type

Segregation test

Likelihood ratio test of segregation in Lansing Woods data:

Segregation test

Likelihood ratio test of segregation in Lansing Woods data:

```
fit0 <- ppm(lansing, ~marks + polynom(x,y,3))
fit1 <- ppm(lansing, ~marks * polynom(x,y,3))
anova(fit0, fit1, test="Chi")
```

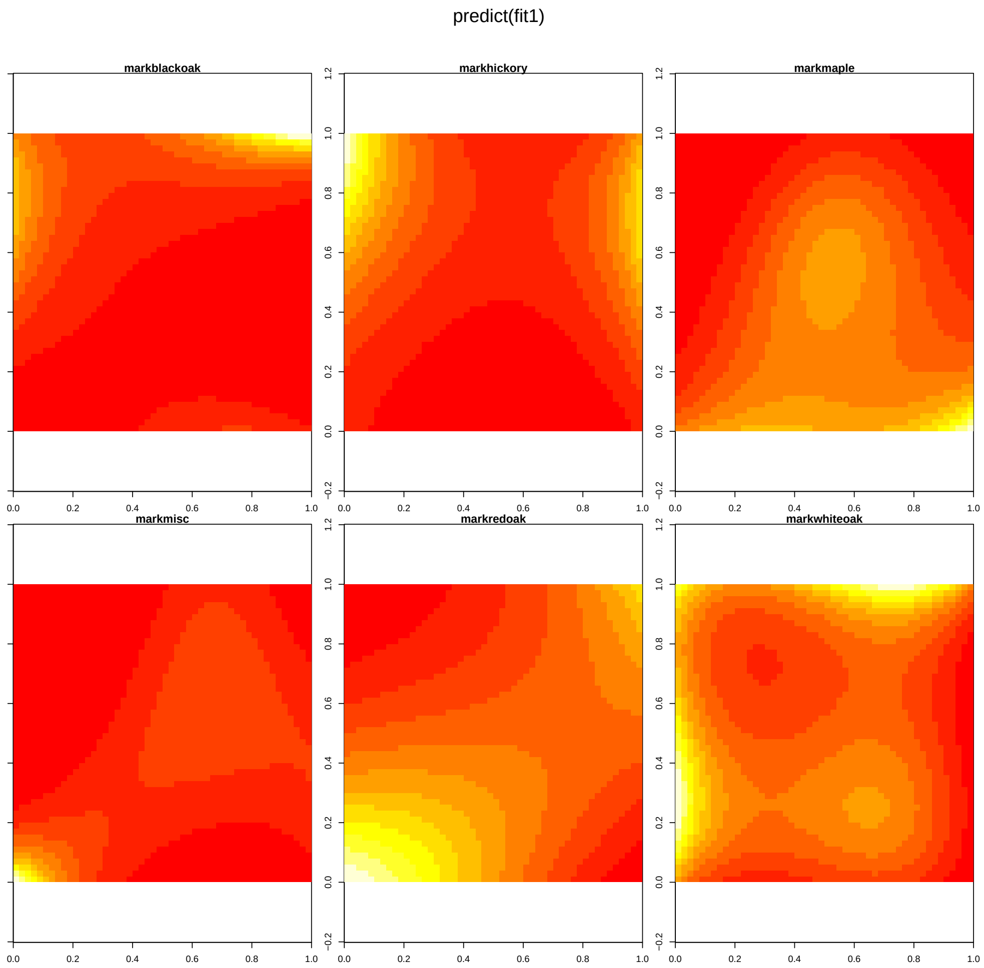
Likelihood ratio test of segregation in Lansing Woods data:

```
fit0 <- ppm(lansing, ~marks + polynom(x,y,3))
fit1 <- ppm(lansing, ~marks * polynom(x,y,3))
anova(fit0, fit1, test="Chi")
```

```
Analysis of Deviance Table

Model 1: .mpl.Y ~ marks + polynom(x, y, 3)
Model 2: .mpl.Y ~ marks * polynom(x, y, 3)
  Resid. Df Resid. Dev    Df Deviance  P(>|Chi|)
1     73515    17485.0
2     73470    16872.4    45    612.6 1.226e-100
```

Fitted intensity

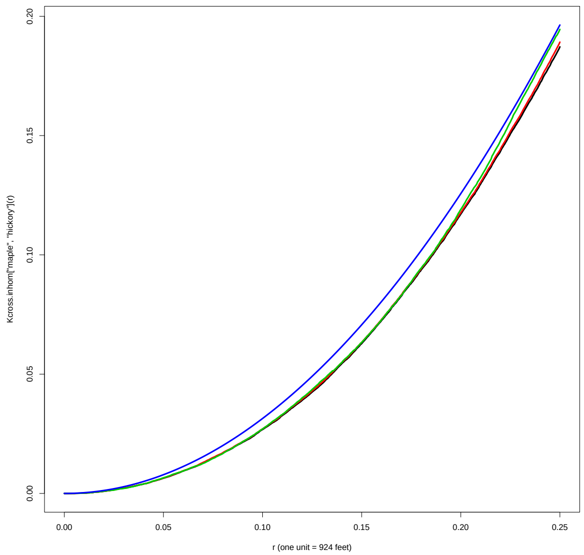
```
fit1 <- ppm(lansing, ~marks * polynom(x,y,3))
plot(predict(fit1))
```



Inhomogeneous multitype K function

Inhomogeneous K function can be generalised to inhomogeneous multitype K function.

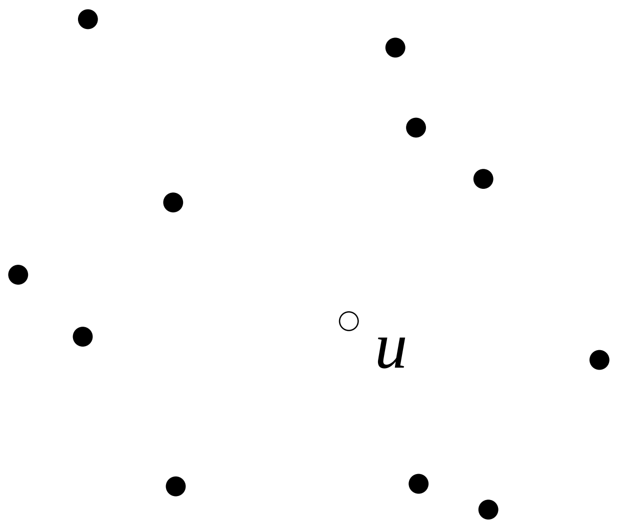
```
fit1 <- ppm(lansing, ~marks * polynom(x,y,3))
lamb <- predict(fit1)
plot(Kcross.inhom(lansing, "maple","hickory",
      lamb$markmaple, lamb$markhickory))
```



Multitype Gibbs models

Conditional intensity

The conditional intensity $\lambda(u, m \mid \mathbf{x})$ is essentially the conditional probability of finding a point of type m at location u, given complete information about the rest of the process $\mathbf{x}$.



Multitype Strauss process

```
>    ppm(amacrine, ~marks, Strauss(r=0.04))

Stationary Strauss process

First order terms:
beta_off  beta_on
156.0724 162.1160

Interaction: Strauss process
interaction distance:   0.04
Fitted interaction parameter gamma:     0.4464
```

```
> rad <- matrix(c(0.03, 0.04, 0.04, 0.02), 2, 2)
> ppm(amacrine, ~marks,
        MultiStrauss(radii=rad,types=c("off", "on")))

Stationary Multitype Strauss process

First order terms:
beta_off  beta_on
120.2312 108.8413

Interaction radii:
     off   on
off 0.03 0.04
on  0.04 0.02
Fitted interaction parameters gamma_ij:
       off     on
off 0.0619 0.8786
on  0.8786 0.0000
```

www.spatstat.org