Lab 11: Multitype summary functions and models

This session is concerned with summary statistics and Gibbs models for multitype point patterns.
The lecturer's R script is available here (right click and save).

Exercise 1

The `amacrine` dataset contains the locations of cells of two types ("on" and "off" detectors) in a layer of the retina.

1. Compute and plot the bivariate L function for the amacrine data.

```
Lam <- alltypes(amacrine, "L")
```

2. plot estimates of the bivariate pair correlation functions by

```
plot(alltypes(amacrine, pcfcross))
```

3. What is the overall interpretation of these summary functions?

Exercise 2

Continuing with the `amacrine` data,

1. Use `alltypes` to plot the bivariate G-functions G_{ij} for each pair of types i, j in the amacrine data.
2. Use `alltypes` to plot the functions $G_{i\bullet}$ (`Gdot` in `spatstat`) for each type i in the amacrine data.
3. What is the overall interpretation of the G-functions?

Exercise 3

The dataset `bramblecanes` gives the locations and ages of bramble cane plants in a study region. Age is a categorical variable, with three levels. We will conduct a randomisation test of the Random Labelling Property.

1. Read the help for the command `rlabel`.
2. We will use the bivariate K-function $K_{2,0}$ as our summary statistic. Compute this for the data using `Kcross(bramblecanes, "2", "0")` and plot it.
3. Read the help for `Kcross`. Find the names of the second and third arguments to the function.
4. Generate the simulation envelopes as follows

```
shuffle <- expression(rlabel(bramblecanes))
E <- envelope(bramblecanes, Kcross, nsim=19, simulate=shuffle, i="2", j="0")
plot(E)
```

   Note that the named arguments `i` and `j` are not recognised by the `envelope` command (as we can check from the help file for `envelope`), so they are passed to the command `Kcross` as we intended.

5. Generate the corresponding simulation envelopes of the bivariate L-function, either by replacing `Kcross` by `Lcross` in the code above, or by

```
plot(E, sqrt(./pi) ~ r)
```

Exercise 4

We want to fit a Gibbs process model to the `betacells` data.

1. Access the `betacells` data and plot the pattern.
2. Save the data as a point pattern `X` and save only the mark `type`

```
X <- betacells
marks(X) <- marks(betacells)$type
```

3. Plot the bivariate K functions. 1. Does it appear that cells of the same type interact? If so, guess at a suitable interaction distance. 2. Does it appear that cells of different types interact? If so, guess at a suitable interaction distance.
4. Fit a multitype Strauss model using the selected interactions. For example if your answer to question i was "yes, at 20 microns" and your answer to question ii was "yes, at 30 microns",

```
rad <- matrix(c(20,30,30,20), 2, 2)
ppm(X ~ marks, MultiStrauss(typ,rad))
```

   while if your answer to question i was "no" and your answer to question ii was "yes, at 60 microns",

```
rad <- matrix(c(NA,60,60,NA), 2, 2)
ppm(X ~ marks, MultiStrauss(typ,rad))
```

   Interpret the fitted model. Plot the array of fitted pairwise interactions using `plot(fitin(fit))` where `fit` is the fitted model. What is the fitted strength of the interaction?

5. For comparison purposes, fit the following models, interpret them, and compare the results:

```
fitU <- ppm(X ~ marks, Strauss(60))
rad <- matrix(60, 2, 2)
fitE <- ppm(X ~ marks, MultiStrauss(rad))
```

Exercise 5

Here we will use profile pseudolikelihood to estimate the interaction distances for the multitype Strauss model in Question 4. We'll assume that points of different types do not interact, and that points of the same type interact at a distance R which is the same for each type.

1. Create a vector of values of R to search over:

```
rval <- data.frame(R=seq(50,100,by=5))
```

   This will become the argument `s` of `profilepl`.

2. We need the argument `f` of `profilepl`, and this should be a function that takes the value R and produces a multitype Strauss interaction. So define

```
MS <- function(R){ MultiStrauss(diag(c(R,R))) }
```

   Try typing `MS(50)` to check that this is what you expect.

3. Then we can use maximum profile pseudolikelihood:

```
profilepl(rval, MS, X, ~marks)
```