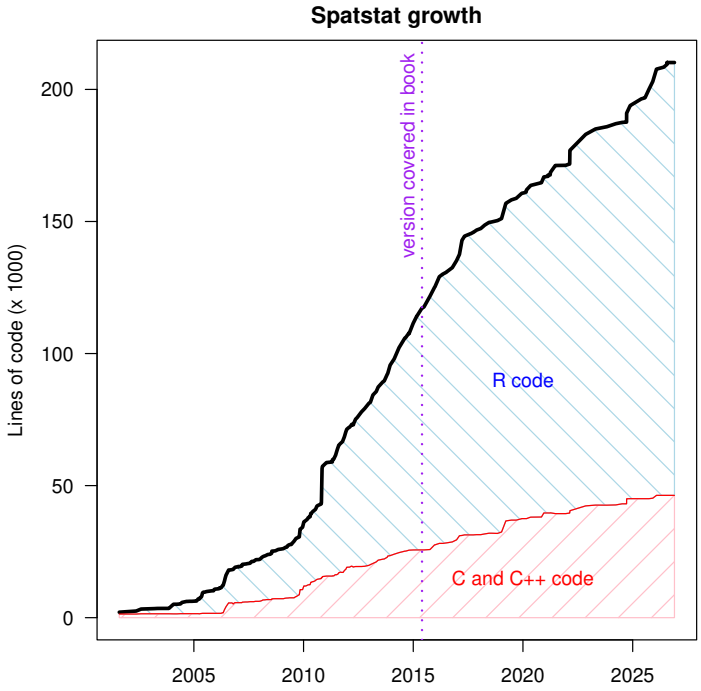
Summary of recent updates to `spatstat`

Adrian Baddeley, Rolf Turner and Ege Rubak

`spatstat` version 3.6-4, released on 2026-11-27

This is a summary of changes to the `spatstat` package that have occurred since the publication of the book [2] in 2015. Since then, the `spatstat` family has grown by 79%, including 1503 new functions and 10 new datasets, and now contains more than 210,000 lines of code. This document summarises the most important changes.



Contents

1 Version information **2**

2 Package structure **2**

3 Précis of all changes **3**

4 New datasets **13**

5 New classes **13**

6 New Functions **14**

7 Alphabetical list of changes **26**

1 Version information

The book [2], published in December 2015, covered `spatstat` version `1.42-0`, released in May 2015.

The current versions of the `spatstat` family of packages (used to produce this document) are:

date	package	version
2026-11-06	spatstat.utils	3.2-6
2025-09-12	spatstat.data	3.1-9
2026-05-21	spatstat.sparse	3.2-0
2026-11-24	spatstat.univar	3.2-1
2026-11-25	spatstat.geom	3.8-4
2026-09-21	spatstat.random	3.5-2
2026-11-26	spatstat.explore	3.8-5
2026-07-27	spatstat.model	3.7-2
2026-09-25	spatstat.linnet	3.5-4
2026-11-27	spatstat	3.6-4
2024-07-11	spatstat.local	5.1-0
2026-05-21	spatstat.Knet	3.1-3
2024-07-16	spatstat.gui	3.1-0
2026-09-10	spatstat.convert	1.0-1

2 Package structure

The original `spatstat` package grew to be very large. It has now been split into a family of packages, to satisfy the requirements of CRAN.

This should not affect the user: existing code will continue to work in the same way.

Typing `library(spatstat)` will load the familiar `spatstat` package which can be used as before.

2.1 Sub-packages

Currently there are ten sub-packages, called `spatstat.utils`, `spatstat.data`, `spatstat.univar`, `spatstat.sparse`, `spatstat.geom`, `spatstat.random`, `spatstat.explore`, `spatstat.model`, `spatstat.linnet`, and `spatstat`.

- The `spatstat` package now contains only documentation and introductory material. It provides beginner's introductions, latest news, vignettes, interactive demonstration scripts, and a few help files summarising the package.
- The `spatstat.data` package now contains all the datasets for `spatstat`.
- The `spatstat.utils` package contains basic utility functions for `spatstat`.
- The `spatstat.univar` package contains functions for estimating and manipulating probability distributions of one-dimensional random variables.
- The `spatstat.sparse` package contains functions for manipulating sparse arrays and performing linear algebra.
- The `spatstat.geom` package contains definitions of spatial objects (such as point patterns, windows and pixel images) and code which performs geometrical operations.
- The `spatstat.random` package contains functions for random generation of spatial patterns and random simulation of models.
- The `spatstat.explore` package contains the code for exploratory data analysis and nonparametric analysis of spatial data.
- The `spatstat.model` package contains the code for model-fitting, model diagnostics, and formal inference.
- The `spatstat.linnet` package defines spatial data on a linear network, and performs geometrical operations and statistical analysis on such data.

Installing: If you install `spatstat`, then the system will install all the other sub-packages listed above.
Running: If you type `library(spatstat)` in an R session, the system will automatically load `spatstat.data`, `spatstat.univar`, `spatstat.geom`, `spatstat.random`, `spatstat.explore`, `spatstat.model` and `spatstat.linnet`. It will also silently **"import"** `spatstat.utils` and `spatstat.sparse`. To access the functions in `spatstat.utils` directly, you would need to type `library(spatstat.utils)`. Similarly for `spatstat.sparse`.

2.2 Extension packages

There are also extension packages which provide additional capabilities and must be loaded explicitly when you need them. Currently there are four extension packages, with a fifth in development:

- `spatstat.local` for local model-fitting ('geographically weighted regression');
- `spatstat.Knet` which provides additional code for analysing point patterns on a network;
- `spatstat.gui` which contains interactive graphics functions;
- `spatstat.convert` which provides code for converting spatial data between the formats defined in `spatstat` and the formats defined in the packages `sp`, `sf` and `terra`;
- `spatstat.sphere` for analysing point patterns on a sphere (under development!)

3 Précis of all changes

Here is the text from the 'overview' sections of the News and Release Notes for each update.

- Bootstrap aggregation method for estimating point process intensity.
- Modified Jones-Foster correction for pair correlation functions.
- New option for boundary correction at zero in one-dimensional density estimation.
- Adaptive kernel estimation of relative risk.
- Transect of a spatial function.
- Transect of an image on a network.
- Transect of a function on a network.
- `relrisk.ppp` handles multiple columns of marks.
- Extensions to `relrisk.lpp`
- More utilities for handling indices.
- New package `spatstat.convert`.
- Variance-covariance matrix for determinantal models.
- More diagnostics for determinantal models.
- Added variable plots for all kinds of models.
- Adaptive kernel smoothing of mark values.
- Random Dirichlet tessellation avoiding edge effects.
- More capabilities for three-dimensional point patterns.
- More methods for `intensity`
- More methods for repulsion index `repul`.
- Changed rule for converting bandwidths in pair correlation.
- Improved handling of boundary pixels in `density()`, `relrisk()`, `Smooth()`
- Improved handling of boundary pixels in model predictions.
- Improvements to rose diagrams.
- Fiksel's bandwidth rule.
- Faster calculation of Ripley's isotropic edge correction.
- Random diffusion of points.

- Random unrounding.
- Truncated Poisson distribution.
- Improved handling of boundary pixels when discretising a polygonal window.
- Improved handling of 'rejected' points.
- Simulate a Markov chain with sparse transition matrix.
- Weighted sum of outer products in a 3D array or sparse array.
- Distribution of a weighted sum of chi-squared random variables.
- New generic `Lmodel` computes the theoretical L-function of a fitted model or theoretical model.
- `envelope` method for determinantal point process models.
- Specify the theoretical value of the summary function in `envelope`.
- Important changes to default settings for pair correlation functions.
- More flexible interpretation of arguments `lambda = <ppm>` or `lambda = <lppm>` in inhomogeneous summary statistics
- New options for multitype pair correlation functions.
- More options for `simulate` argument in `envelope` methods.
- Extended capability of `envelope.pp3`
- Method of T.F. Cox for identifying dense and sparse regions of points.
- New class of (theoretical) cluster process models
- Inhomogeneous Poisson process in 3D.
- Non-uniform random points in 3D.
- Simple Sequential Inhibition in 3D.
- Support function, Feret diameter, polar body, Voronoi flower.
- Merge tiles in a tessellation.
- Return squared nearest neighbour distances.
- Bug fix in Minkowski sum.
- Summary method for tessellations.
- Identify threads in a network
- Improved algorithm for integrating a function over a network.
- Summary functions have argument `rmax`.
- Diagnostic functions have argument `rmax`.
- More support for long vectors (vectors with more than 2^{31} entries).
- Waagepetersen's composite likelihood for cluster processes.
- Important changes to mark correlation function.
- Convert a recursively partitioned point process model to a spatial tessellation.
- Signed distance transform and signed distance function.
- Spatial persistence index and cluster strength index.
- Improvements to `profilepl` and `effectfun`.
- Improvements to perspective plot of a function on a network.

- Replicated point patterns on a linear network.
- Diagnostics for a fitted point process model on a linear network.
- `lppm` recognises local coordinates in the model formula.
- Major extension of code for ROC curves.
- ROC curves for determinantal point process models.
- Extended facilities for spatial logistic regression.
- Level sets calculated using analytic geometry, for some functions.
- Easier control over transparency of image plots.
- Scramble the colours when plotting a point pattern or line segment pattern.
- Different discretisation rules supported in `slrm`.
- Leave-one-pixel-out prediction in `slrm`.
- Generate Gaussian random fields.
- Lurking variable plot on a network.
- Partial residual plot on a network.
- Quadrat counting and quadrat tests on a linear network.
- More geometrical operations on linear network objects.
- Pixelwise arithmetic on a linear network.
- Partial residuals for cluster process and Cox process models.
- Extended facilities for spatial logistic regression.
- Improvements to perspective views of images.
- Missing or unavailable (NA) spatial objects are accepted by `solist` and `hyperframe`.
- Missing or unavailable (NA) spatial objects are handled by `solapply`, `with.hyperframe` and by many low-level functions.
- Summary functions handle NA objects.
- Simulation of a fitted model in a different window is better supported.
- More options when simulating a fixed number of random points.
- `unnormdensity` returns the smoothing parameters used.
- Find centroids of tiles of a tessellation.
- More options for converting a tessellation to an image.
- Youden index.
- Shrinkage estimator in `Smooth.ppp`.
- Backgrounds for plots of point patterns.
- Connected components of polygonal windows.
- More options for disc windows.
- Interactively identify tiles of a tessellation.
- Interactively identify segments of a network.
- Interactively identify tiles of a tessellation on a network.
- `spatstatLocator` supports 'snap' behaviour (rounding the clicked location to the nearest grid point).

- `clickpoly` allows vertices to be 'snapped' (rounded to the nearest grid point).
- Corrected unit of length in two datasets.
- New options for defining nearest neighbours.
- `im.apply` handles vector-valued functions.
- Remove small holes from a window.
- Extension to weighted median and weighted quantiles.
- Shrinkage estimator of relative risk.
- Normalised relative risk.
- Extensions to cross-validated bandwidth selection for smoothing.
- Conditional simulation (fixed number of points) in log-Gaussian Cox processes and Neyman-Scott cluster processes.
- Nonlinear colour maps and symbol maps.
- Backgrounds for image plots.
- `quadratcount.ppp` supports left-closed, right-open rectangular tiles.
- More support for interpolated CDF objects
- Improvements to plotting of images, and arrays of images.
- More control over plotting of colour maps, symbol maps and texture maps.
- Hyperframes handle a `Surv` object as a single column.
- New dataset `meningitis`.
- New dataset `shelling`.
- Methods for functions which are constant on each tile of a tessellation.
- New estimator of pair correlation function.
- New methods for bandwidth selection for pair correlation estimation.
- Helper functions for measuring estimator performance.
- Conditional simulation in `rLGCP`, `rThomas`, `rMatClust`, `rCauchy`, `rVarGamma`.
- Recognise powers of an integer.
- Prime factorisation computation improved.
- More support for `interpolatedCDF` objects.
- Nonlinear colour maps and symbol maps.
- Simulation of `zclustermodel` objects.
- Shortest path between two points on a network.
- Boundary-corrected kernel density estimation on the positive half-line.
- Relative risk estimation using diffusion.
- Smoothing using diffusion.
- Tessellations can have any kind of marks.
- Bandwidth selection by non-random bootstrap.
- More control over default colours.
- Perspective plot of spatial point pattern with numerical marks.
- Quantiles using linear approximation.

- Extract the knots (jump points) of a weighted CDF.
- List the history of all changes made to a function in spatstat.
- More efficient prime factorisation.
- Digits in the decimal representation of a number.
- Easier control over quadrature schemes.
- Some geometry code accelerated.
- Corrected format of `gorillas` dataset.
- The `spatstat` family no longer depends on the packages `maptools`, `sp` and `RandomFields`.
- geometry code accelerated.
- Spatially weighted median and quantile of mark values.
- Boyce index.
- code for fitting and simulating log-Gaussian Cox models has changed.
- New vignette on function objects (class `"fv"` and `"envelope"`)
- Vignette on shapefiles temporarily removed.
- Integration of functions.
- `clarkevans.test` modified.
- Improvements to `envelope` methods.
- Conditional simulation for Matern cluster process.
- Improvements to `runifpoint` and `rpoispp`.
- Extension of distance transform algorithm.
- Improvement to progress reports.
- Suppress annoying warnings.
- Changed the calculation of standard errors in `density.ppp` and `relrisk.ppp`.
- Inline arithmetic for function tables (class `"fv"`) and arrays (class `"fasp"`)
- Standard error calculation for `Smooth.ppp` (experimental)
- multitype pair correlation functions can save numerator and denominator.
- multitype inhomogeneous J functions.
- More support for automatic bandwidth selection.
- Standard errors are now available for `ppm` models fitted using `gam`.
- `linearKinhom` and `linearpcfinhom` now automatically estimate the intensity.
- `density.lpp` accepts bandwidth selection rules, and has a simple default bandwidth.
- Pair correlation functions allow more control over smoothing parameters.
- Extension to support for one-dimensional smoothing kernels.
- Improvements to `update` methods for point process models.
- New `update` methods for classes `dppm` and `rppm`.
- Generate truncated Poisson random variables.
- reciprocal moment of Poisson variable conditioned to be positive.
- Methods for `[[` and `[[<-` for hyperframes.

- Colour map for `pH` values.
- Restrict a colour map to a narrower range of values.
- Integral of a one-dimensional density estimate.
- `kppm` has been accelerated when `method="palm"` or `"clik2"`.
- `kppm` can save the history of the optimisation algorithm.
- Faster algorithms for simulating cluster processes.
- Penalised model-fitting for Neyman-Scott cluster process models.
- Index of the strength of clustering in a Neyman-Scott cluster process model.
- Probability of having any siblings.
- More information is printed about Neyman-Scott cluster process models.
- Palm intensity diagnostic plot.
- Convert several factors or factor-valued images to a common set of levels.
- Extension to `rjitter`
- Alternative to `rjitter`
- Quantile function as a function
- Periodic edge correction for K function.
- Changed denominator in K function and pair correlation function.
- Bandwidth selection for adaptive kernel estimation of intensity.
- U-shaped and inverted-U-shaped curves in `rhohat`.
- Radial cumulative integral of an image.
- New dataset `stonetools`.
- Regularized model-fitting in `ppm` and `kppm`.
- Residuals for recursively-partitioned models.
- Residuals for any observed point pattern and an estimate of its intensity.
- Weighted measures and weighted integrals.
- Improved approximation of intensity of Gibbs models.
- Experimental code to represent (theoretical) point process models
- Extract more information about a point process model.
- kernel smoothing on a linear network.
- linear network K function and pair correlation function based on Euclidean distance.
- inhomogeneous linear network J function.
- Terminal vertices of a network.
- A point pattern on a network can be plotted as cross-ticks.
- The interactive graphics functions `iplot` and `istat` have been removed from `spatstat` into a new extension package `spatstat.gui`.
- The packages `tcltk` and `rpanel` are no longer Suggested by `spatstat`.
- `spatstat` now Imports the package `spatstat.sparse`.
- `spatstat` now Imports the package `spatstat.utils`.

- `spatstat` now requires the package `spatstat.data` which contains the datasets.
- `spatstat` now suggests the package `fftwtools`.
- Conditional simulation in `kppm`.
- More diagnostics for spatial logistic regression models.
- Increased numerical stability in `kppm`.
- Simulation of the product shot noise Cox process.
- Information criteria for model selection in `kppm`.
- Estimation of the spatial covariance function of a pixel image
- Modified handling of covariates in `slrm`
- New options for `weighted.quantile`
- Buffer tessellation
- New function for jittering point patterns on a network.
- Extensions to `rhohat`
- `densityfun.ppp` handles query points outside original window
- Extension to `discretise`.
- Improvement to `densityEqualSplit`.
- summary method for spatial logistic regression models
- New options for `distmap.psp`
- Improved output in `summary.mppm`
- Increased speed for large datasets.
- Variance calculations handle larger datasets.
- Relative risk estimation on a network.
- Leave-one-out density estimation on a network.
- Add new vertices to a linear network.
- More support for multi-dimensional patterns.
- `predict.mppm` now works for multitype point process models.
- Improved handling of `newdata` in `predict.mppm`
- New datasets `concrete` and `btb`.
- Changed default value of `stringsAsFactors`.
- Function `lengths.psp` has been renamed `lengths_psp`.
- Tessellations on a linear network can now have marks.
- More functions for manipulating tessellations on a linear network.
- New functions for simulating point processes on a linear network.
- Nearest Neighbour Index function can now return mark values.
- Index of repulsion strength for determinantal point process models.
- Nearest neighbours between two point patterns in any number of dimensions.
- More options for handling bad simulation outcomes in `envelope`.
- `mppm` accepts case weights.

- Bandwidth selectors warn about extreme values of bandwidth.
- Fast kernel estimation on a linear network using 2D kernels.
- Extension of Scott's rule for bandwidth selection.
- Cross-validated bandwidth selection on a linear network.
- Random thinning and random labelling of spatial patterns extended to different types of pattern.
- Confidence intervals for multitype K function.
- Envelopes for balanced two-stage test
- Extensions to adaptive intensity estimators
- 'Dartboard' tessellation using polar coordinates.
- Standard error calculation for inverse-distance weighting.
- Kernel estimate of intensity as a `function(x,y)`.
- Extract discrete and continuous components of a measure.
- Improvements and extensions to leverage and influence code.
- Plot a line segment pattern using line widths.
- Find connected components of each tile in a tessellation.
- Geometrical operations on `distfun` objects.
- Join vertices in a linear network.
- Distance map and contact distribution for rectangular structuring element.
- Lurking variable plot for models fitted to several point patterns.
- New dataset `cetaceans`.
- Gamma correction for colour maps and image plots.
- Class `units` has been renamed `unitname` to avoid package collision.
- More support for tessellations.
- Fixed longstanding bug in leverage and influence diagnostics.
- Improvements and bug fixes for leverage and influence diagnostics.
- Tighter bounding box for `psp`, `lpp`, `linnet` objects.
- Improved layout in `plot.solist`
- Tools to increase colour saturation.
- Connected components of a 3D point pattern.
- Accelerated computations on linear networks.
- Accelerated simulation of determinantal point processes.
- Improved printing of 3D point patterns.
- Minor corrections to handling of unitnames.
- Improvements to `ppm` and `update.ppm`.
- Correction to `lohboot`
- Numerous bug fixes for linear networks code.
- Now handles disconnected linear networks.
- Effect function is now available for all types of fitted model.

- Geometric-mean smoothing.
- A model can be fitted or re-fitted to a sub-region of data.
- New fast algorithm for kernel smoothing on a linear network.
- Leverage and influence diagnostics extended to Poisson/Gibbs models fitted by logistic composite likelihood.
- Two-stage Monte Carlo test.
- Dirichlet/Voronoi tessellation on a linear network.
- Thinning of point patterns on a linear network.
- More support for functions and tessellations on a linear network.
- Bandwidth selection for pair correlation function.
- Pooling operations improved.
- Operations on signed measures.
- Operations on lists of pixel images.
- Improved pixellation of point patterns.
- Stieltjes integral extended.
- Subset operators extended.
- Greatly accelerated `rmh` when using `nsave`
- Sufficient Dimension Reduction for point processes.
- Alternating Gibbs Sampler for point process simulation.
- New class of spatially sampled functions.
- ROC and AUC extended to other types of point patterns and models.
- More support for linear networks.
- More support for infinite straight lines.
- `spatstat` now depends on the packages `nlme` and `rpart`.
- Important bug fix in `linearK`, `linearpcf`
- Changed internal format of `linnet` and `lpp` objects.
- Faster computation in linear networks.
- Bias correction techniques.
- Bounding circle of a spatial object.
- Option to plot marked points as arrows.
- Kernel smoothing accelerated.
- Workaround for bug in some graphics drivers affecting image orientation.
- Non-Gaussian smoothing kernels.
- Improvements to inhomogeneous multitype K and L functions.
- Variance approximation for pair correlation function.
- Leverage and influence for multitype point process models.
- Functions for extracting components of vector-valued objects.
- Recursive-partition point process models.
- Minkowski sum, morphological dilation and erosion with any shape.

- Minkowski sum also applicable to point patterns and line segment patterns.
- Important bug fix in Smooth.ppp
- Important bug fix in spatial CDF tests.
- More bug fixes for replicated patterns.
- Simulate a model fitted to replicated point patterns.
- Inhomogeneous multitype F and G functions.
- Summary functions recognise `correction="all"`
- Leverage and influence code handles bigger datasets.
- More support for pixel images.
- Improved progress reports.
- New dataset `redwood3`
- Fixed namespace problems arising when spatstat is not loaded.
- Important bug fix in leverage/influence diagnostics for Gibbs models.
- Surgery with linear networks.
- Tessellations on a linear network.
- Laslett's Transform.
- Colour maps for point patterns with continuous marks are easier to define.
- Pair correlation function estimates can be pooled.
- Stipulate a particular version of a package.
- More support for replicated point patterns.
- More support for tessellations.
- More support for multidimensional point patterns and point processes.
- More options for one-sided envelopes.
- More support for model comparison.
- Convexifying operation.
- Subdivide a linear network.
- Penttinen process can be simulated (by Metropolis-Hastings or CFTP).
- Calculate the predicted variance of number of points.
- Accelerated algorithms for linear networks.
- Quadrat counting accelerated, in some cases.
- Simulation algorithms have been accelerated; simulation outcomes are *not* identical to those obtained from previous versions of `spatstat`.
- Determinantal point process models.
- Random-effects and mixed-effects models for replicated patterns.
- Dao-Genton test, and corresponding simulation envelopes.
- Simulated annealing and simulated tempering.
- spatstat colour tools now handle transparent colours.
- Improvements to `[` and `subset` methods

- Extensions to kernel smoothing on a linear network.
- Support for one-dimensional smoothing kernels.
- Mark correlation function may include weights.
- Cross-correlation version of the mark correlation function.
- Penttinen pairwise interaction model.
- Improvements to simulation of Neyman-Scott processes.
- Improvements to fitting of Neyman-Scott models.
- Extended functionality for pixel images.
- Fitted intensity on linear network
- Triangulation of windows.
- Corrected an edge correction.

4 New datasets

The following new datasets have been added. These are now provided in the sub-package `spatstat.data`.

- `austates`: The states and large mainland territories of Australia represented as polygonal regions forming a tessellation.
- `redwood3`: a more accurate version of the `redwood` data.
- `cetaceans`: point patterns of whale and dolphin sightings.
- `concrete`: air bubbles in concrete.
- `btb`: bovine tuberculosis occurrences.
- `stonetools`: palaeolithic stone tools and bone fragments.
- `shelling`: artillery impacts in Ukraine.
- `meningitis`: meningitis cases in Germany.

5 New classes

The following new classes of objects may be of use.

- `clusterprocess`: Neyman-Scott-Cox cluster process model with specified values of the parameters (whereas an object of class `kppm` represents such a model fitted to data).
- `NAobject`: a spatial object that is missing or was not observed.
- `traj`: Trajectory (history of function evaluations) in a model that was fitted by optimisation.
- `metric`: Class of distance metrics. An object of class `metric` represents a distance metric between points in two-dimensional space. See `help(metric.object)`.
- `ssf`: Class of spatially sampled functions. An object of class `"ssf"` represents a spatial function which has been evaluated or sampled at an irregular set of points. See `help(ssf)`.

There are also two experimental classes which will be removed:

- `zclustermodel`: Experimental, deprecated. An object of class `zclustermodel` represents a Neyman-Scott cluster point process model with specified parameter values.
- `zgibbsmodel`: Experimental, deprecated. An object of class `zgibbsmodel` represents a Gibbs point process model with specified parameter values.

6 New Functions

Following is a list of all the functions that have been added, starting with the most recent additions.

- `relriskAdaptiveKernel`: New generic.
- `relriskAdaptiveKernel.ppp`: Variable-bandwidth kernel estimation of spatial relative risk or spatially-varying probability.
- `transect`: New generic.
- `transect.funxy`: Linear transect for a spatial function (object of class `"funxy"`).
- `transect.linim`: Transect of an image on a linear network (object of class `"linim"`) along a path in the network.
- `transect.linfun`: Transect of a function on a linear network (object of class `"linfun"`) along a path in the network.
- `positiveIndex` (package `spatstat.utils`): Convert any kind of index to a vector of positive integers.
- `logicalIndex` (package `spatstat.utils`): Convert any kind of index to a logical vector.
- `auc.dppm`: Area under the ROC curve for a fitted determinantal point process model. Method for `auc` for objects of class `dppm`.
- `parres.dppm`: Partial residuals for a fitted determinantal point process model. Method for `parres` for objects of class `dppm`.
- `vcov.dppm`: Variance-covariance matrix for the trend parameters in a fitted determinantal point process model.
- `SmoothAdaptiveKernel.ppp`: Adaptive kernel smoothing of mark values of a point pattern.
- `rpoisDirichletTess`: Dirichlet tessellation induced by a Poisson process on the infinite two-dimensional plane, avoiding edge effects.
- `distbdry`: New generic.
- `distbdry.ppp`: distance from each point in a 2D point pattern to the boundary of the observation window. Equivalent to `bdist.points`.
- `distbdry.pp3`: distance from each point in a 3D point pattern to the boundary of the containing box.
- `distbdry.lpp`: distance from each data point on a linear network to the nearest terminal vertex.
- `rjitter.pp3`: randomly displace the points in a 3D point pattern.
- `intensity.kppm`: Intensity of a fitted cluster process or Cox process model.
- `intensity.lppm`: Intensity of a fitted point process model on a linear network.
- `intensity.mppm`: Intensity of a point process model fitted to multiple point patterns.
- `repul.detpointprocfamily`: Repulsion index for determinantal point process model.
- `repul.kppm`, `repul.clusterprocess`: Repulsion index for cluster process models and Cox models (gives negative value).
- `repul.slrm`, `repul.rppm`: Repulsion index for spatial logistic regression models and recursive partition models (gives zero value)
- `repul.ppm`, `repul.mppm`: Repulsion index for fitted Poisson or Gibbs models using Poisson-saddlepoint approximation.
- `bw.fiksel`: Fiksel's (1988) rule for smoothing bandwidth for estimating the pair correlation function.
- `rdiffuse`: Randomly perturb each point in a point pattern by diffusion.
- `rUnround`, `rUnround.ppp` Random un-rounding of a discretised point pattern.
- `dpoisnonzero`, `ppoisnonzero`, `qpoisnonzero`: Density, cumulative probability and quantile function for a Poisson random variable conditioned to be nonzero.

- `dpoistrunc`, `ppoistrunc`, `qpoistrunc` Density, cumulative probability and quantile function for the truncated Poisson distribution.
- `runSparseMarkovChain` (in `spatstat.sparse`): Generate simulated realisations of a Markov chain using a sparse matrix representation of the transition probability matrix.
- `sumsymouter` (in `spatstat.sparse`): Weighted sum of outer products in a 3D array or sparse array.
- `length.sparse3Darray`: method for `length` for sparse 3D arrays.
- `dsocs`, `psocs`, `qsocs`, `rsocs`: Probability density function, cumulative probability distribution function, quantile function and random generator for a random variable which is a weighted Sum Of Chi-Squared random variables.
- `farebro`: Farebrother's Algorithm for the cumulative probability of a weighted sum of non-central chi-squared variables.
- `envelope.dppm`: Method for `envelope` for fitted models of class `"dppm"`
- `Lmodel`, `Lmodel.kppm`, `Lmodel.dppm`, `Lmodel.ppm`, `Lmodel.detpointprocfamily`: Given a point process model, return a function in the R language which computes the theoretical L-function of the model.
- `Kmodel.clustermodel`, `pcfmodel.clustermodel`, `Lmodel.clustermodel`: Calculate the theoretical K-function, L-function or pair correlation function for a model of class `"clustermodel"`.
- `coxmap`: Given a point pattern, divide the spatial region into areas where the pattern is dense, sparse or neither, using the method of T.F. Cox (1979).
- `rpoint3`: Random points in three dimensions with any probability density
- `clusterprocess`: Create an object of class `clusterprocess`.
- `clusterradius.clusterprocess`: Radius of the support of the offspring density of a cluster process model.
- `predict.clusterprocess`: Calculate intensity of a cluster process model
- `print.clusterprocess`: Print a cluster process model
- `reach.clusterprocess`: Distance beyond which the pair correlation is effectively 1.
- `simulate.clusterprocess` Generate simulated realisations of a cluster process model.
- `mergeTiles`: Merge tiles in a tessellation.
- `mergeTiles.lintess`: Merge tiles in a tessellation on a network.
- `FeretDiamFun`: Create a function that computes the Feret diameter of a domain at any angle.
- `FeretBox`: Compute the tightest rectangle enclosing a domain at any angle.
- `minFeretDiam`, `maxFeretDiam`: Minimum or maximum Feret diameter of a domain
- `SupportFun` Create a function that computes the support function of a domain.
- `polarbody` Compute the polar body of a domain.
- `voronoiFlower` Compute the Voronoi flower of a domain.
- `summary.tess`, `print.summary.tess` Method for `summary` for tessellations.
- `capitalise`: make the first letter of each word a capital letter.
- `default.colourmap`: default colour map for a vector or factor.
- `levelset`: new generic
- `levelset.im`: level set of a pixel image
- `levelset.distfun`: level set of a distance function.
- `levelset.tessfun`: level set of a function created by `as.function.tess`.
- `difflong`: lagged difference of long vector.

- `persist`: spatial persistence index for a cluster process model.
- `clusterstrength`: cluster strength index for a cluster process or Cox process model.
- `as.tess.rppm`: convert a recursively partitioned point process model to a spatial tessellation.
- `formula.rppm`: extract the model formula of a recursively partitioned point process model.
- `parres`: new generic function for partial residual plots.
- `parres.ppm`: the original function `parres` has become the method `parres.ppm`.
- `parres.kppm`: partial residual plot for fitted cluster process or Cox process models.
- `diagnose`: new generic function for model diagnostics.
- `diagnose.lppm`: diagnostic plots for point process model on a linear network.
- `parres.lppm`: partial residual plot for point process model on a linear network.
- `lurking.lppm`: Lurking variable plot for point process model on a linear network.
- `lurking.lpp`: Lurking variable plot for point pattern on a linear network.
- `residuals.lppm`: residual measure for point process model on a linear network.
- `eem.lppm`: Stoyan-Grabarnik exponential energy weights diagnostic for point process model on a linear network.
- `flipxy.linnet`: Swap x and y coordinates of a linear network
- `flipxy.lpp`: Swap x and y coordinates of a point pattern on a linear network
- `as.lintess`: Convert other kinds of data to a tessellation on a network.
- `quadratcount.lpp` Method for quadrat counting on a linear network.
- `quadrat.test.lpp`, `quadrat.test.lppm` Methods for quadrat test on a linear network.
- `harmonise.linim`: Convert several images on a linear network to a common pixel grid.
- `linim.apply`: Apply a function pixelwise to a list of images on a network.
- `roc.dppm`: ROC curve for determinantal point process model.
- `markequal`: Mark equality function $e(r)$ for multitype point patterns.
- `nnequal`, `nncount`: Summary functions for multitype point patterns based on the types of nearest neighbours.
- `tolcon`: Tolerance contours for spatial relative risk.
- `rGRFexpo`, `rGRFgauss`, `rGRFmatern`, `rGRFstable`, `rGRFgencauchy`: generate a Gaussian random field.
- `tile.centroids`: Find the centroids of the tiles of a tessellation and return them as a point pattern.
- `addROC`: Calculate partial ROC curve for adding a new covariate to a model.
- `dropROC`: Calculate partial ROC curve for removing a covariate from a model.
- `dropply`, `addapply`: Consider all single-variable additions or deletions in a model.
- `roc.rhohat`, `roc.cdftest`, `roc.bermantest`: calculate ROC curve from other kinds of data.
- `roc.im`: calculate ROC curve from a pixel image such as an estimate of intensity.
- `youden`: Youden statistic for ROC curves.
- `fillholes.owin`: remove small holes from a window.
- `NAobject`: create a missing object of a particular class.
- `is.NAobject`: recognise whether an object is a missing object.
- `is.na.hyperframe`: determine which entries in a hyperframe are missing (NA values or NA objects).

- as.ppp.NAobject, as.owin.NAobject, as.im.NAobject, domain.NAobject, Frame.NAobject, Window.NAobject, marks.NAobject, marks<-.NAobject: methods for the generics as.ppp, as.owin etc, which return an NAobject.
- npoints.NAobject, nobjects.NAobject: methods for the generics npoints and nobjects, which return an NA integer value.
- c.layered: concatenate layered objects, retaining their plot arguments.
- identify.tess: interactively identify tiles of a tessellation.
- identify.linnet: interactively identify segments of a network.
- identify.lintess: interactively identify tiles of a tessellation on a network.
- as.linfun.linnet: create a function on a linear network which maps each network segment to a specified value.
- plot.interpolatedCDF, print.interpolatedCDF: methods for plotting and printing objects of class "interpolatedCDF".
- simulate.zclustermodel: method for simulating objects of class zclustermodel.
- integral.tessfun: Integral of a function which is constant on each tile of a tessellation.
- print.tessfun, plot.tessfun, as.tess.tessfun: Methods for the class "tessfun" of functions which are constant on each tile of a tessellation.
- bw.bdh: Adjust bandwidth of kernel estimate of pair correlation function, to account for inhomogeneity.
- bw.pcfinhom: Cross-validation rule for selecting bandwidth of kernel estimate of (inhomogeneous) pair correlation function.
- ptwise.envelope, bias.envelope, RMSE.envelope: Calculate pointwise statistics of the simulated function values in an envelope object.
- ISE.envelope, ISB.envelope, IV.envelope: Integrated squared error, integrated squared bias, integrated variance of the simulated function values in an envelope object.
- MISE.envelope: Mean integrated squared error of the simulated function values in an envelope object.
- rev.colourmap: Reverses the sequence of colour values in a colour map. A method for the generic rev.
- default.image.colours, reset.default.image.colours: control the default colours used for plotting images in spatstat.
- shortestpath: Find the shortest path between two specified points on a network, and return it as a line segment pattern.
- densityBC: An extension of stats::density.default that includes boundary corrections for truncation of the density to the positive half line.
- densityAdaptiveKernel.default: Variable-bandwidth boundary-corrected kernel density estimation.
- relriskHeat, relriskHeat.ppp: Relative risk estimation using diffusion.
- bw.relriskHeatppp: Bandwidth selection for relriskHeat.ppp
- SmoothHeat, SmoothHeat.ppp: Smoothing numerical values observed at points, using diffusion.
- blurHeat, blurHeat.im: Image smoothing using diffusion.
- bw.taylor: Bandwidth selection for kernel density estimation using Taylor's non-random bootstrap
- latest.changes: Lists the history of all changes that have been made to a particular function in the spatstat family of packages.
- persp.ppp: For a spatial point pattern with numeric marks, generate a perspective plot in which each data point is shown as a vertical spike, with height proportional to the mark value.
- knots.ewcdf: Method for generic knots for extracting the jump points of a weighted cumulative distribution function.
- firstdigit, lastdigit, ndigits: digits in the decimal representation of a number.

- `bw.abram.default`: Abramson adaptive bandwidths. Default method for `bw.abram`, applicable to numerical vectors.
- `default.symbolmap.ppp`: Algorithm for determining the graphical symbol map used by `plot.ppp`.
- `summary.symbolmap`: Method for `summary` for symbol maps.
- `SpatialMedian.ppp`, `SpatialQuantile.ppp`: spatially weighted median and quantile of mark values of a point pattern.
- `boyce`: Boyce index and continuous Boyce index.
- `densityAdaptiveKernel.splitppp`: A method for `densityAdaptiveKernel` for split point patterns.
- `integral.fv`: Compute the integral of a function object.
- `compileCDF`: Low level utility for calculating cumulative distribution function of distance variable.
- `Math.fv`, `Complex.fv`, `Summary.fv`, `Ops.fv`: Methods for arithmetic operations for function tables (class `"fv"`)
- `Math.fasp`, `Complex.fasp`, `Summary.fasp`, `Ops.fasp`: Methods for arithmetic operations for function arrays (class `"fasp"`)
- `Gcross.inhom`, `Gdot.inhom`: Multitype G functions for inhomogeneous point processes.
- `Jcross.inhom`, `Jdot.inhom`, `Jmulti.inhom`: Multitype J functions for inhomogeneous point processes.
- `summary.bw.optim`, `print.summary.bw.optim`: Method for `summary` of optimised bandwidth objects (class `bw.optim`). These are the objects produced by the bandwidth selection functions such as `bw.diggle`, `bw.scott`, `bw.pcf`
- `psp2mask`: Function `as.mask.psp` has been renamed `psp2mask`. The old function `as.mask.psp` still exists but will soon be deprecated and later removed.
- `update.dppm`: Update method for determinantal point process models.
- `update.rppm`: Update method for recursively partitioned point process models.
- `[[.hyperframe`, `[[<-.hyperframe`: Methods for `[[` and `[[<-` for hyperframes.
- `pHcolourmap`, `pHcolour`: Colour map for values of pH
- `restrict.colourmap`: Restrict a colourmap to a narrower range of values.
- `integral.density`: Compute the integral of a one-dimensional kernel density estimate.
- `as.colourmap`: Extract colour information from an object.
- `panysib`: Probability that a point in a cluster process has *any* siblings.
- `is.poissonclusterprocess`: Detects whether a given model is a Poisson cluster process (which includes Neyman-Scott processes).
- `traj`, `print.traj`, `plot.traj`, `lines.traj`: Extract, print and plot the trajectory of function evaluations.
- `rpoisnonzero`: Generate Poisson random variables conditioned to be positive.
- `rpoistrunc`: Generate 'truncated' Poisson random variables, conditioned to be greater than or equal to a specified minimum value.
- `recipEnzpois`: Calculate the first reciprocal moment of nonzero Poisson variable.
- `rclusterBKBC`: (Advanced use) Internal algorithm to simulate any Neyman-Scott cluster process using either the naive, Brix-Kendall, or Baddeley-Chang algorithm.
- `palmdiagnose`, `plot.palmdiag`: Palm intensity diagnostic plot for cluster process models proposed by Tanaka, Ogata and Stoyan.
- `harmoniseLevels`: Given several factors or factor-valued pixel images, convert them all to have the same set of factor levels.
- `rexplode`: "Explode" a point pattern by randomly displacing each group of duplicated points to make a circular pattern around the original location. An alternative to `rjitter`.

- `quantilefun`: Return a function that computes any quantiles of a given dataset.
- `bw.CvL.adaptive`: Bandwidth selection for adaptive kernel estimation of intensity.
- `radcumint`: Radial cumulative integral of an image.
- `Smooth.lpp`: kernel smoothing on a linear network.
- `residuals.rppm`: Residual measure for a recursively-partitioned point process model.
- `residualMeasure`: Residual measure for any observed point pattern and any estimate of its intensity.
- `linearKEuclid, linearpcfEuclid, linearKEuclidInhom, linearpcfEuclidInhom`: Linear network K function and pair correlation function based on Euclidean distances.
- `linearJinhom`: Inhomogeneous J function on a linear network.
- `terminalvertices`: Extract the terminal vertices of a linear network.
- `bw.relrisk.lpp`: This function replaces `bw.relrisklpp` and is a method for the generic `bw.relrisk`.
- `measureWeighted`: weighted version of a measure.
- `harmonicmean, harmonicsum`: The harmonic mean of a set of numbers, calculated robustly.
- `which.min.fair, which.max.fair` (in `spatstat.utils`): Find the location of the minimum or maximum entry in a vector; if there are multiple minima or maxima, choose one of them at random.
- `hardcoredist`: Extract the hard core distance of a point process model.
- `interactionorder`: Extract the order of interpoint interaction of a point process model.
- `zgibbsmodel`: Experimental. Create an object of class `zgibbsmodel`.
- `print.zgibbsmodel`: Experimental. Print an object of class `zgibbsmodel`.
- `is.poisson.zgibbsmodel`, `is.stationary.zgibbsmodel`: Experimental. Methods for class `zgibbsmodel`.
- `indefinteg`: Numerically computes the indefinite integral of a function
- `framedist.pixels`: Computes distance from each pixel to the enclosing rectangle.
- `lurking.slrm`: Lurking variable plot for spatial logistic regression models.
- `eem.slrm`: Exponential energy marks for spatial logistic regression models.
- `eem.ppm`: Exponential energy marks for Gibbs and Poisson point process models (this function was previously called `eem`).
- `transformquantiles`: Transform the quantiles of a vector, matrix, array or pixel image.
- `convexmetric`: Distance metric based on a convex set.

  `invoke.metric`: Low level function to perform a desired operation using a given metric.
- `mean.ecdf, mean.ewcdf` Calculate the mean of an empirical cumulative distribution function.
- `rjitter.ppp`:
  - This function was previously called `rjitter`. It is now a method for the new generic function `rjitter`.
  - New argument `adjust` allows the default radius to be adjusted.
  - The resulting point pattern now has attribute `radius`.
  - If `retry=TRUE`, the resulting point pattern now has attribute `tries` which counts the number of trials that were required.
- `bufftess`: Distance buffer tessellation
- `ic`: Information criteria for model selection in ppm and kppm. Kindly contributed by Achmad Choiruddin, Jean-Francois Coeurjolly and Rasmus Waagepetersen.
- `rPSNCP`: Generate simulated realisations of the product shot noise Cox process. Contributed by Abdollah Jalilian, Yongtao Guan and Rasmus Waagepetersen.

- `spatcov`: Estimate the spatial covariance function of a pixel image.
- `summary.slrm`, `print.summary.slrm` Summary method for spatial logistic regression models
- `coef.summary.slrm`: Print the fitted coefficients, confidence interval and p-values for a spatial logistic regression model.
- `pairMean`: Compute the mean of a specified function of interpoint distance between random points in a window.
- `rjitterlpp`: Apply random displacements to the points on a linear network.
- `intersect.boxx`: Compute intersection of boxes in multi-dimensional space
- `scale.boxx`, `scale.ppx`: Methods for `scale` for boxes and patterns in multi-dimensional space
- `shift.boxx`, `shift.ppx`: Methods for `shift` for boxes and patterns in multi-dimensional space
- `is.boxx`: Determine whether an object is a multidimensional box
- `relrisk.lpp`: nonparametric estimation of relative risk on a network.
- `bw.relrisklpp`: Bandwidth selection for relative risk estimation on a network.
- `bw.lppl`: Bandwidth selection for kernel density estimation of point patterns on a linear network, using likelihood cross-validation.
- `densityfun.lpp`: a method for `densityfun` for point patterns on a linear network.
- `addVertices`: Add new vertices to a network, at locations outside the existing network.
- `lengths_psp`: this is the new name of the function `lengths.psp`, which had to be changed because of a conflict with the generic `lengths`.
- `densityEqualSplit`: The equal-split algorithm for kernel density estimation on a network is now visible as a separate function.
- `densityHeat`: The heat-equation algorithm for kernel density estimation on a network is now visible as a separate function. It has also been extended to computing leave-one-out density estimates at the data points.
- `hotrod`: Compute the heat kernel $\kappa(u, v)$ on a one-dimensional line segment.
- `heatkernelapprox`: Calculate an approximation to the value of the heat kernel on a network evaluated at the source point, $\kappa(u, u)$.
- `is.linim`: test whether an object is a pixel image on a linear network (class `"linim"`).
- `rcelllpp`: Simulate the cell point process on a linear network.
- `rSwitzerlpp`: Simulate the Switzer-type point process on a linear network.
- `intersect.lintess`: Form the intersection of two tessellations on a linear network.
- `chop.linnet`: Divide a linear network into tiles using infinite lines.
- `repairNetwork`: Detect and repair inconsistencies in internal data in a `linnet` or `lpp` object.
- `marks<-.lintess`, `unmark.lintess`: Assign marks to the tiles of a tessellation on a linear network.
- `marks.lintess`: Extract the marks of the tiles of a tessellation on a linear network.
- `tilenames.lintess`: Extract the names of the tiles in a tessellation on a linear network
- `tilenames<-.lintess`: Change the names of the tiles in a tessellation on a linear network
- `nobjects.lintess`: Count the number of tiles in a tessellation on a linear network
- `as.data.frame.lintess`: Convert a tessellation on a linear network into a data frame.
- `repul`: Repulsiveness index for a determinantal point process model.
- `reach.kppm`: Reach (interaction distance) for a Cox or cluster point process model.
- `summary.dppm`, `print.summary.dppm`: Summary method for determinantal point process models.

- `nncross.ppx`: Nearest neighbours between two point patterns in any number of dimensions.
- `rthinclumps`: Divide a spatial region into clumps and randomly delete some of them.
- `densityQuick.lpp`: Fast kernel estimator of point process intensity on a network using 2D smoothing kernel.
- `data.lppm`: Extract the original point pattern dataset (on a linear network) to which the model was fitted.
- `bw.scott.iso`: Isotropic version of Scott's rule (for point patterns in any dimension).
- `bits.envelope`: Global simulation envelope corresponding to `bits.test`, the balanced independent two-stage Monte Carlo test.
- `extrapolate.psp`: Extrapolate line segments to obtain infinite lines.
- `uniquemap`: Map duplicate points to unique representatives. Generic with methods for `ppp`, `lpp`, `ppx`
- `uniquemap.data.frame`, `uniquemap.matrix`: Map duplicate rows to unique representatives
- `localKcross`, `localLcross`, `localKdot`, `localLdot`, `localKcross.inhom`, `localLcross.inhom`: Multitype local K functions.
- `polartess`: tessellation using polar coordinates.
- `densityVoronoi`: adaptive estimate of point process intensity using tessellation methods.
- `densityAdaptiveKernel`: adaptive estimate of point process intensity using variable kernel methods.
- `bw.abram`: compute adaptive smoothing bandwidths using Abramson's rule.
- `coords.quad`: method for `coords`, to extract the coordinates of the points in a quadrature scheme.
- `lineartileindex`: low-level function to classify points on a linear network according to which tile of a tessellation they fall inside.
- `markmarkscatter`: Mark–mark scatterplot.
- `bw.CvL`: Cronie-van Lieshout bandwidth selection for density estimation.
- `subset.psp`: subset method for line segment patterns.
- `densityfun`, `densityfun.ppp`: Compute a kernel estimate of intensity of a point pattern and return it as a function of spatial location.
- `as.im.densityfun`: Convert `function(x,y)` to a pixel image.
- `measureDiscrete`, `measureContinuous`: Extract the discrete and continuous components of a measure.
- `connected.tess`: Find connected components of each tile in a tessellation and make a new tessellation composed of these pieces.
- `dffit.ppm`: Effect change diagnostic `DFFIT` for spatial point process models.
- `shift.distfun`, `rotate.distfun`, `reflect.distfun`, `flipxy.distfun`, `affine.distfun`, `scalardilate.distfun`: Methods for geometrical operations on `distfun` objects.
- `rescale.distfun`: Change the unit of length in a `distfun` object.
- `plot.indicfun`: Plot method for indicator functions created by `as.function.owin`.
- `Smooth.leverage.ppm`, `Smooth.influence.ppm`: Smooth a leverage function or an influence measure.
- `integral.leverage.ppm`, `integral.influence.ppm`: Compute the integral of a leverage function or an influence measure.
- `mean.leverage.ppm`: Compute the mean value of a leverage function.
- `rectdistmap`: Distance map using rectangular metric.
- `rectcontact`: Contact distribution function using rectangular structuring element.
- `joinVertices`: Join specified vertices in a linear network.

- `summary.ssf`: Summary method for a spatially sampled function (class `ssf`).
- `unstack.tess`: Given a tessellation with multiple columns of marks, take the columns one at a time, and return a list of tessellations, each carrying only one of the original columns of marks.
- `contour.leverage.ppm`: Method for `contour` for leverage functions of class `leverage.ppm`
- `lurking`: New generic function for lurking variable plots.
- `lurking.ppp`, `lurking.ppm`: These are equivalent to the original function `lurking`. They are now methods for the new generic `lurking`.
- `lurking.mppm`: New method for class `mppm`. Lurking variable plot for models fitted to several point patterns.
- `print.lurk`: Prints information about the object returned by the function `lurking` representing a lurking variable plot.
- `model.matrix.mppm`: Method for `model.matrix` for models of class `mppm`.
- `test.crossing.psp`, `test.selfcrossing.psp`: Previously undocumented functions for testing whether segments cross.
- `to.saturated`: Convert a colour value to the corresponding fully-saturated colour.
- `intensity.psp`: Compute the average total length of segments per unit area.
- `boundingbox.psp`: Bounding box for line segment patterns. This produces a tighter bounding box than the previous default behaviour.
- `boundingbox.lpp`: Bounding box for point patterns on a linear network. This produces a tighter bounding box than the previous default behaviour.
- `boundingbox.linnet`: Bounding box for a linear network. This produces a tighter bounding box than the previous default behaviour.
- `"Frame<-.default"`: New default method for assigning bounding frame to a spatial object.
- `connected.pp3`: Connected components of a 3D point pattern.
- `colouroutputs`, `"colouroutputs<-"`: Extract or assign colour values in a colour map. (Documented a previously-existing function)
- `fitin.profilepl`: Extract the fitted interaction from a model fitted by profile likelihood.
- `[<-.linim`: Subset assignment method for pixel images on a linear network.
- `nnfromvertex`: Given a point pattern on a linear network, find the nearest data point from each vertex of the network.
- `tile.lengths`: Calculate the length of each tile in a tessellation on a network.
- `text.ppp`, `text.lpp`, `text.psp`: Methods for `text` for spatial patterns.
- `as.data.frame.envelope`: Extract function data from an envelope object, including the functions for the simulated data ('simfuns') if they were saved.
- `is.connected`, `is.connected.default`, `is.connected.linnet`: Determines whether a spatial object consists of one topologically connected piece, or several pieces.
- `is.connected.ppp`: Determines whether a point pattern is connected after all pairs of points closer than distance R are joined.
- `hist.funxy`: Histogram of values of a spatial function.
- `model.matrix.ippm`: Method for `model.matrix` which allows computation of regular and irregular score components.
- `harmonise.msr`: Convert several measures (objects of class `msr`) to a common quadrature scheme.
- `bits.test`: Balanced Independent Two-Stage Monte Carlo test, an improvement on the Dao-Genton test.
- `lineardirichlet`: Computes the Dirichlet-Voronoi tessellation associated with a point pattern on a linear network.

- `domain.lintess`, `domain.linfun`: Extract the linear network from a `lintess` or `linfun` object.
- `summary.lintess`: Summary of a tessellation on a linear network.
- `clicklpp`: Interactively add points on a linear network.
- `envelopeArray`: Generate an array of envelopes using a function that returns `fasp` objects.
- `bw.pcf`: Bandwidth selection for pair correlation function.
- `grow.box3`: Expand a three-dimensional box.
- `hexagon`, `regularpolygon`: Create regular polygons.
- `Ops.msr`: Arithmetic operations for measures.
- `Math.imlist`, `Ops.imlist`, `Summary.imlist`, `Complex.imlist`: Arithmetic operations for lists of pixel images.
- `measurePositive`, `measureNegative`, `measureVariation`, `totalVariation`: Positive and negative parts of a measure, and variation of a measure.
- `as.function.owin`: Convert a spatial window to a `function(x,y)`, the indicator function.
- `as.function.ssf`: Convert an object of class `ssf` to a `function(x,y)`
- `as.function.leverage.ppm` Convert an object of class `leverage.ppm` to a `function(x,y)`
- `sdr`, `dimhat`: Sufficient Dimension Reduction for point processes.
- `simulate.rhohat`: Simulate a Poisson point process with the intensity estimated by `rhohat`.
- `rlpp`: Random points on a linear network with a specified probability density.
- `cut.lpp`: Method for `cut` for point patterns on a linear network.
- `has.close`: Faster way to check whether a point has a close neighbour.
- `psib`: Sibling probability (index of clustering strength in a cluster process).
- `rags`, `ragsAreaInter`, `ragsMultiHard`: Alternating Gibbs Sampler for point processes.
- `bugfixes`: List all bug fixes in recent versions of a package.
- `ssf`: Create a spatially sampled function
- `print.ssf`, `plot.ssf`, `contour.ssf`, `image.ssf`: Display a spatially sampled function
- `as.im.ssf`, `as.ppp.ssf`, `marks.ssf`, `marks<-.ssf`, `unmark.ssf`, `[.ssf`, `with.ssf`: Manipulate data in a spatially sampled function
- `Smooth.ssf`: Smooth a spatially sampled function
- `integral.ssf`: Approximate integral of spatially sampled function
- `roc.kppm`, `roc.lppm`, `roc.lpp`: Methods for `roc` for fitted models of class `"kppm"` and `"lppm"` and point patterns of class `"lpp"`
- `auc.kppm`, `auc.lppm`, `auc.lpp`: Methods for `auc` for fitted models of class `"kppm"` and `"lppm"` and point patterns of class `"lpp"`
- `timeTaken`: Extract the timing data from a `"timed"` object or objects.
- `rotate.infline`, `shift.infline`, `reflect.infline`, `flipxy.infline`: Geometrical transformations for infinite straight lines.
- `whichhalfplane`: Determine which side of an infinite line a point lies on.
- `matrixpower`, `matrixsqrt`, `matrixinvsqrt`: Raise a matrix to any power.
- `points.lpp`: Method for `points` for point patterns on a linear network.
- `pairs.linim`: Pairs plot for images on a linear network.

- `closetriples`: Find close triples of points.
- `anyNA.im`: Method for `anyNA` for pixel images.
- `bc`: Bias correction (Newton-Raphson) for fitted model parameters.
- `rex`: Richardson extrapolation for numerical integrals and statistical model parameter estimates.
- `boundingcircle`, `boundingcentre`: Find the smallest circle enclosing a window or point pattern.
- `[.linim` : Subset operator for pixel images on a linear network.
- `mean.linim`, `median.linim`, `quantile.linim`: The mean, median, or quantiles of pixel values in a pixel image on a linear network.
- `weighted.median`, `weighted.quantile`: Median or quantile of numerical data with associated weights.
- `"[.linim"`: Subset operator for pixel images on a linear network.
- `mean.linim`, `median.linim`, `quantile.linim`: The mean, median, or quantiles of pixel values in a pixel image on a linear network.
- `boundingcircle`, `boundingcentre`: Smallest circle enclosing a spatial object.
- `split.msr`: Decompose a measure into parts.
- `unstack.msr`: Decompose a vector-valued measure into its component measures.
- `unstack.ppp`, `unstack.psp`, `unstack.lpp`: Given a spatial pattern with several columns of marks, separate the columns and return a list of spatial patterns, each having only one column of marks.
- `kernel.squint`: Integral of squared kernel, for the kernels used in density estimation.
- `as.im.data.frame`: Build a pixel image from a data frame of coordinates and pixel values.
- `covering`: Cover a window using discs of a given radius.
- `dilationAny`, `erosionAny`, `%(-)%` : Morphological dilation and erosion by any shape.
- `FmultiInhom`, `GmultiInhom` Inhomogeneous multitype/marked versions of the summary functions `Fest`, `Gest`.
- `kernel.moment` Moment or incomplete moment of smoothing kernel.
- `MinkowskiSum`, `%(+)%`: Minkowski sum of two windows: `A %(+)% B`, or `MinkowskiSum(A,B)`
- `nobjects`: New generic function for counting the number of 'things' in a dataset. There are methods for `ppp`, `ppx`, `psp`, `tess`.
- `parameters.interact`, `parameters.fii`: Extract parameters from interpoint interactions. (These existing functions are now documented.)
- `ppmInfluence`: Calculate `leverage.ppm`, `influence.ppm` and `dfbetas.ppm` efficiently.
- `rppm`, `plot.rppm`, `predict.rppm`, `prune.rppm`: Recursive-partition point process models.
- `simulate.mppm` Simulate a point process model fitted to replicated point patterns.
- `update.interact`: Update the parameters of an interpoint interaction. [This existing function is now documented.]
- `where.max`, `where.min` Find the spatial location(s) where a pixel image achieves its maximum or minimum value.
- `compileK`, `compilepcf`: make a K function or pair correlation function given the pairwise distances and their weights. [These existing internal functions are now documented.]
- `laslett`: Laslett's Transform.
- `lintess`: Tessellation on a linear network.
- `divide.linnet`: Divide a linear network into pieces demarcated by a point pattern.
- `insertVertices`: Insert new vertices in a linear network.
- `thinNetwork`: Remove vertices and/or segments from a linear network etc.

- `connected.linnet`: Find connected components of a linear network.
- `nvertices`, `nvertices.linnet`, `nvertices.owin`: Count the number of vertices in a linear network or vertices of the boundary of a window.
- `as.data.frame.linim`, `as.data.frame.linfun`: Extract a data frame of spatial locations and function values from an object of class `linim` or `linfun`.
- `as.linfun`, `as.linfun.linim`, `as.linfun.lintess`: Convert other kinds of data to a `linfun` object.
- `requireversion`: Require a particular version of a package (for use in stand-alone R scripts).
- `as.function.tess`: Convert a tessellation to a `function(x,y)`. The function value indicates which tile of the tessellation contains the point (x, y).
- `tileindex`: Determine which tile of a tessellation contains a given point (x, y).
- `persp.leverage.ppm`: Method for persp plots for objects of class `leverage.ppm`
- `AIC.mppm`, `extractAIC.mppm`: AIC for point process models fitted to replicated point patterns.
- `nobs.mppm`, `terms.mppm`, `getCall.mppm`: Methods for point process models fitted to replicated point patterns.
- `rPenttinen`: Simulate the Penttinen process using perfect simulation.
- `varcount`: Given a point process model, compute the predicted variance of the number of points falling in a window.
- `inside.boxx`: Test whether multidimensional points lie inside a specified multidimensional box.
- `lixellate`: Divide each segment of a linear network into smaller segments.
- `nsegments.linnet`, `nsegments.lpp`: Count the number of line segments in a linear network.
- `grow.boxx`: Expand a multidimensional box.
- `deviance.ppm`, `deviance.lppm`: Deviance for a fitted point process model.
- `pseudoR2`: Pseudo-R-squared for a fitted point process model.
- `tiles.empty` Checks whether each tile of a tessellation is empty or nonempty.
- `summary.linim`: Summary for a pixel image on a linear network.
- Determinantal Point Process models:
  - `dppm`: Fit a determinantal point process model.
  - `fitted.dppm`, `predict.dppm`, `intensity.dppm`: prediction for a fitted determinantal point process model.
  - `Kmodel.dppm`, `pcfmodel.dppm`: Second moments of a determinantal point process model.
  - `rdpp`, `simulate.dppm`: Simulation of a determinantal point process model.
  - `logLik.dppm`, `AIC.dppm`, `extractAIC.dppm`, `nobs.dppm`: Likelihood and AIC for a fitted determinantal point process model.
  - `print.dppm`, `reach.dppm`, `valid.dppm`: Basic information about a `dpp` model.
  - `coef.dppm`, `formula.dppm`, `print.dppm`, `terms.dppm`, `labels.dppm`, `model.frame.dppm`, `model.matrix.dppm`, `model.images.dppm`, `is.stationary.dppm`, `reach.dppm`, `unitname.dppm`, `unitname<-.dppm`, `Window.dppm`: Various methods for `dppm` objects.
  - `parameters.dppm`: Extract meaningful list of model parameters.
  - `objsurf.dppm`: Objective function surface of a `dppm` object.
  - `residuals.dppm`: Residual measure for a `dppm` object.
- Determinantal Point Process model families:
  - `dppBessel`, `dppCauchy`, `dppGauss`, `dppMatern`, `dppPowerExp`: Determinantal Point Process family functions.
  - `detpointprocfamilyfun`: Create a family function.
  - `update.detpointprocfamily`: Set parameter values in a determinantal point process model family.
  - `simulate.dppm`: Simulation.

  - is.stationary.detpointprocfamily, intensity.detpointprocfamily, Kmodel.detpointprocfamily, pcfmodel.det Moments.
  - dim.detpointprocfamily, dppapproxkernel, dppapproxpcf, dppeigen, dppkernel, dppparbounds, dppspecdenrange, dppspecden: Helper functions.

- dg.envelope: Simulation envelopes corresponding to Dao-Genton test.
- dg.progress: Progress plot (envelope representation) for the Dao-Genton test.
- dg.sigtrace: significance trace for the Dao-Genton test.
- markcrosscorr: Mark cross-correlation function for point patterns with several columns of marks.
- rtemper: Simulated annealing or simulated tempering.
- rgb2hsva: Convert RGB to HSV data, like rgb2hsv, but preserving transparency.
- superimpose.ppplist, superimpose.splitppp: New methods for 'superimpose' for lists of point patterns.
- dkernel, pkernel, qkernel, rkernel: Probability density, cumulative probability, quantiles and random generation from distributions used in basic one-dimensional kernel smoothing.
- kernel.factor: Auxiliary calculations for one-dimensional kernel smoothing.
- spatdim: Spatial dimension of any object in the spatstat package.
- as.boxx: Convert data to a multi-dimensional box.
- intensity.ppx: Method for intensity for multi-dimensional space-time point patterns.
- fourierbasis: Evaluate Fourier basis functions in any number of dimensions.
- valid: New generic function, with methods valid.ppm, valid.lppm, valid.dppm.
- emend, emend.ppm, emend.lppm: New generic function with methods for ppm and lppm. emend.ppm is equivalent to project.ppm.
- Penttinen: New pairwise interaction model.
- quantile.density: Calculates quantiles from kernel density estimates.
- CDF.density: Calculates cumulative distribution function from kernel density estimates.
- triangulate.owin: decompose a spatial window into triangles.
- fitted.lppm: fitted intensity values for a point process on a linear network.
- parameters: Extract all parameters from a fitted model.

7 Alphabetical list of changes

Here is a list of all changes made to existing functions, listed alphabetically.

- adaptive.density: This function can now perform adaptive estimation by three methods: tessellation-based methods, variable-bandwidth kernel estimation, and nearest-neighbour intensity estimation. The calculations are performed by densityVoronoi, densityAdaptiveKernel or nndensity.
- addvar:
  - Now accepts point process models of class ppm, kppm, dppm, rppm, lppm.
  - For non-Poisson models, significance bands are computed by simulation.
- affine.owin: Allows transformation matrix to be singular, if the window is polygonal.
- alltypes: If envelope=TRUE and the envelope computation reaches the maximum permitted number of errors (maxnerr) in evaluating the summary function for the simulated patterns, then instead of triggering a fatal error, the envelope limits will be set to NA.
- anova.mppm:

  - Now handles Gibbs models, and performs the adjusted composite likelihood ratio test.
  - New argument `fine`.
  - Issues a warning when applied to random-effects models (models fitted using the argument `random`).

- `anyDuplicated.ppp`:
  - New argument `rule` supports different levels of accuracy consistent with the options for `duplicated.ppp`.
  - Accelerated.

- `append.psp`: arguments may be `NULL`.

- `applynbd`: Now works for point patterns in three dimensions (class `"pp3"`) and point patterns on a network (class `"lpp"`).

- `as.function.tess`: New argument `values` specifies the function values.

- `as.fv.kppm`: Improved arguments for empirical estimation of pair correlation function when the model was fitted with a non-default value of `method`.

- `as.im`:
  - Many methods for `as.im` now have argument `rule.pix` passed to `owin2mask`. This makes it possible to ensure that boundary pixels which partially intersect the window are included in the resulting pixel image, by setting `rule.pix="cover"`.
  - Many methods for `as.im` now have argument `rule.eps`.

- `as.im.distfun`: New argument `approx` specifies the choice of algorithm.

- `as.im.tess`:
  - New argument `rule.tile` specifies how to determine, for each pixel, the tile of the tessellation that covers the pixel.
  - New argument `values`.

- `as.im.function`:
  - New argument `strict`.
  - New argument `stringsAsFactors`.
  - The formal default value of `stringsAsFactors` has been changed to `NULL` to conform to changes in R. (The actual default value is `TRUE` for `R < 4.1.0` and `FALSE` for `R >= 4.1.0`).

- `as.im.leverage.ppm`: New argument `what`.

- `as.im.nnfun`: New argument `approx` chooses between a fast, approximate algorithm and a slow, exact algorithm.

- `as.im.smoothfun`: New argument `approx` chooses between a fast, approximate algorithm and a slow, exact algorithm.

- `as.layered`: Default method now handles a (vanilla) list of spatial objects.

- `as.layered.ppp`: Now handles rejected points in `attr(x, "rejects")` and retrieves them as a layer.

- `as.linfun.lintess`:
  - New argument `values` specifies the function value for each tile.
  - The default `values` are the marks, if present.
  - New argument `navalue`.
  - Computation accelerated.

- `as.linim.default`:
  - New arguments `delta` and `nd` control spacing of sample points in internal data.
  - New argument `rule.eps` passed to `as.mask`.

- `as.linim.linfun`:
  - New argument `rule.eps` passed to `as.mask`.

- as.linnet.linnet: New argument maxsize.
- as.linnet.psp:
  - New arguments chop and fuse.
  - If the line segment pattern has marks, then the resulting linear network also carries these marks in the $lines component.
  - Computation accelerated.
  - The resulting network has attribute "camefrom" indicating the provenance of each line segment in the network.
- as.lpp: accepts more data formats:
  - Now handles the case where coordinates seg and tp are given but x and y are missing.
  - Now handles the case where x is a data frame with columns named x,y,seg,tp or x,y or seg,tp.
- as.mask: New argument rule.eps specifies what to do when the desired pixel size is not a divisor of the frame size.
- as.owin.default:
  - Now refuses to convert a box3 to a two-dimensional window.
  - Now accepts a structure with entries named xmin,xmax, ymin, ymax in any order. This handles objects of class bbox in the sf package.
  - Now detects objects of class SpatialPolygons and issues a more helpful error message.
- as.owin.data.frame: New argument step
- as.polygonal:
  - Can now repair errors in polygon data, if repair=TRUE.
  - Accelerated when w is a pixel mask.
- as.psp: now permits a data frame of marks to have only one column, instead of coercing it to a vector.
- as.rectangle: accelerated in many cases.
- as.solist: The argument x can now be a spatial object; as.solist(cells) is the same as solist(cells).
- bdist.pixels: Accelerated for polygonal windows. New argument method.
- bdist.points: Accelerated for polygonal windows.
- beachcolours:
  - Improved positioning of the yellow colour band.
  - If sealevel lies outside srange, then srange will be extended to include it (without a warning).
- beachcolourmap: Improved positioning of the yellow colour band.
- bilinearform: This function has been moved to the sub-package spatstat.sparse.
- bind.fv:
  - Additional arguments may be functions in the R language.
  - New argument clip.
- blur: New argument kernel.
- bw.abram:
  - This function is now generic, with a method for class ppp.
  - Default method added.
  - New argument smoother determines how the pilot estimate is computed.
  - Formal arguments rearranged.
- bw.diggle, bw.ppl, bw.relrisk, bw.smoothppp:

  - These functions now extract and store the name of the unit of length from the point pattern dataset. When the bandwidth selection criterion is plotted, the name of the unit of length is shown on the x-axis.
  - A warning is issued if the optimal value of the cross-validation criterion occurs at an endpoint of the search interval. New argument `warn`.

- `bw.ppl`:
  - New argument `varcov1` for anisotropic bandwidth selection.
  - New arguments `weights` and `sigma`.
  - New argument `shortcut` allows faster computation.
  - Argument `shortcut` now defaults to `TRUE`.
  - Additional arguments `...` are now passed to `density.ppp`.
- `bw.relrisk`: This function is now generic, with methods for class `"ppp"` and `"lpp"`.
- `bw.relrisk.lpp`: When `method="likelihood"`, the cross-validation criterion is now defined as the *negative* likelihood. This is consistent with `bw.relrisk.ppp`, and ensures that the optimum bandwidth is always found by minimising the cross-validation criterion.
- `bw.relrisk.ppp`: Additional arguments `...` are now passed to `density.ppp`.
- `bw.scott`:
  - the two bandwidth values in the result now have names `sigma.x` and `sigma.y`.
  - Now handles point patterns of any dimension.
  - New arguments `isotropic` and `d`.
- `bw.smoothppp`:
  - New arguments `train` and `test` for cross-validation.
  - New argument `varcov1` for anisotropic bandwidth selection.
- `bw.stoyan`:
  - The rule has been modified so that, if the pattern is empty, it is now treated as if it contained 1 point, so that a finite bandwidth value is returned.
  - New argument `extrapolate` supports a modification of Stoyan's rule.
- `cbind.fv`:
  - Additional arguments may be functions in the R language.
- `cbind.hyperframe`:
  - The result now retains the `row.names` of the original arguments.
  -
- `cdf.test`:
  - Calculations are more robust against numerical rounding effects.
  - The methods for classes `ppp`, `ppm`, `lpp`, `lppm`, `slrm` have a new argument `interpolate`.
  - Monte Carlo test runs much faster.
  - More jittering is applied when `jitter=TRUE`. Warnings about tied values should not occur any more.
- `cdf.test.ppm`:
  - Recognises argument `rule.eps` passed to `as.mask`.
- `cdf.test.mppm`:
  - Now handles Gibbs models.
  - Now recognises `covariate="x"` or `"y"`.
- `circdensity`: Improved output of `print` method.

- `clarkevans`: The argument `correction="all"` is now recognised: it selects all the available options. [This is also the default.]
- `clarkevans.test`:
  - The asymptotic test is now available for any choice of edge correction.
  - New argument `method` determines whether to use the asymptotic test or Monte Carlo test. The default has changed to `method="asymptotic"`.
  - Default edge correction has changed, to avoid bias.
- `clickpoly`:
  - The polygon is now drawn progressively as the user clicks new vertices.
  - Allows user-selected locations of vertices to be snapped (rounded) to the nearest grid points. New arguments `snap.step` and `snap.origin`.
- `closepairs.ppp`: New argument `periodic`.
- `closepairs.ppp`, `closepairs.pp3`:
  - New arguments `distinct` and `neat` allow more options.
  - Argument `ordered` has been replaced by `twice` (but `ordered` is still accepted, with a warning).
  - Performance improved (computation time and memory requirements reduced.) This should improve the performance of many functions in `spatstat`.
- `closepairs.pp3`: Argument `what` can take the value `"ijd"`
- `clusterset`: Improved behaviour.
- `clusterfit`:
  - New argument `algorithm` specifies the choice of optimisation algorithm.
  - Changed precedence rule for handling the algorithm parameters in the minimum contrast algorithm. Individually-named arguments `q,p,rmax,rmin` now take precedence over entries with the same names in the list `ctrl`.
  - New argument `verbose`.
- `clusterstrength`: Now accepts an object of class `"clustermodel"`.
- `colourmap`:
  - New arguments `compress`, `decompress` for nonlinear colour maps.
  - argument `col` can have length 1, representing a trivial colour map in which all data values are mapped to the same colour.
  - New arguments `compress`, `decompress` for nonlinear colour maps.
- `collapse.fv`:
  - This is now treated as a method for the `nlme` generic `collapse`. Its syntax has been adjusted slightly.
  - Recognises the abbreviations used by `fvnames()`.
- `connected.im`: Now handles a logical-valued image properly. Arguments `...` now determine pixel resolution.
- `connected.owin`:
  - New argument `polygonal` allows calculation of connected components using polygonal geometry instead of pixel-based algorithm.
  - Arguments `...` now determine pixel resolution.
- `contour.im`:
  - New argument `col` specifies the colour of the contour lines. If `col` is a colour map, then the contours are drawn in different colours.
  - New argument `log` specifies whether the contour lines should be equally spaced on a logarithmic scale.
- `convolve.im`: the name of the unit of length is preserved.

- `cor.im`, `cov.im`: Default changed to `use="complete.obs"`.
- `crossdist.lpp`:
  - Now handles much larger networks, using the sparse representation of the network.
  - New argument `check`.
- `crossing.psp`: New argument `details` gives more information about the intersections between the segments.
- `crosspairs.ppp`:
  - New argument `periodic` specifies whether to use periodic (toroidal) distances.
  - New arguments `iX`, `iY` make it possible to eliminate pairs in which the two points are identical.
- `crosspairs.pp3`: Argument `what` can take the value `"ijd"`
- `cut.ppp`: Argument `z` can be `"x"` or `"y"` indicating one of the spatial coordinates.
- `dclf.test, mad.test, dclf.progress, mad.progress, dclf.sigtrace, mad.sigtrace, dg.progress, dg.sigtrace`:
  - New argument `clamp` determines the test statistic for one-sided tests.
  - New argument `rmin` determines the left endpoint of the test interval.
  - New argument `leaveout` specifies how to calculate discrepancy between observed and simulated function values.
  - New argument `scale` allows summary function values to be rescaled before the comparison is performed.
  - New argument `interpolate` supports interpolation of p-value.
  - Function values which are infinite, `NaN` or `NA` are now ignored in the calculation (with a warning) instead of causing an error. Warning messages are more detailed.
- `default.rmhcontrol, default.rmhexpand`: New argument `w`.
- `default.symbolmap.ppp`:
  - For a multitype point pattern, `cols` can be a `function(n)`
  - New argument `scramble.cols`
  - Argument `marktransform` renamed `transform` for consistency.
- `densityAdaptiveKernel.default`:
  - New option `zerocor="stableJF"` for improved numerical stability.
- `densityBC`:
  - New option `zerocor="stableJF"` for improved numerical stability.
  - accelerated
  - new argument `xout`
- `densityfun.ppp`: The resulting function can now handle query points which lie outside the window of the original data, and has argument `drop=TRUE` which specifies how to handle them.
- `densityEqualSplit`: New arguments `at` and `leaveoneout` for consistency with other functions.
- `densityHeat`:
  - default behaviour has changed slightly.
  - new argument `finespacing`.
- `density.lpp`:
  - Argument `weights` can be logical valued.
  - Argument `sigma` can now be a function in the R language, assumed to provide a bandwidth selection rule. This function will be applied to the point pattern `x` to compute the bandwidth.
  - Argument `sigma=NULL` is now accepted. The default value is one-eighth of the length of the shortest side of the bounding box of `x`.
  - New fast algorithm (up to 1000 times faster) for the default case where `kernel="gaussian"` and `continuous=TRUE`. Generously contributed by Greg McSwiggan.

- Fast algorithm has been further accelerated.
- Further accelerated when the point pattern contains duplicated points.
- New argument `kernel` specifies the smoothing kernel. Any of the standard one-dimensional smoothing kernels can be used.
- Now supports both the 'equal-split continuous' and 'equal-split discontinuous' smoothers. New argument `continuous` determines the choice of smoother.
- New arguments `weights` and `old`.
- New argument `distance` offers a choice of different kernel methods.
- Infinite bandwidth (`sigma=Inf`) is now permitted, and results in a density estimate that is constant over all locations.

- `density.ppp`:
  - Now accepts argument `rule.pix` passed to `owin2mask`.
  - Argument `weights` can be logical-valued.
  - A non-Gaussian kernel can now be specified using the argument `kernel`.
  - Standard error calculation is now available with any smoothing kernel.
  - The interpretation of `weights` in the calculation of standard error has changed. New argument `wtype` controls this interpretation.
  - Argument `weights` can now be a pixel image.
  - Infinite bandwidth `sigma=Inf` is supported.
  - Accelerated by about 30% when `at="pixels"`.
  - Accelerated by about 15% in the case where `at="points"` and `kernel="gaussian"`.
  - Accelerated in the cases where weights are given or `diggle=TRUE`.
  - New argument `verbose`.
- `densityQuick.lpp`: Argument `X` changed to `x` for consistency.
- `densityVoronoi.ppp`:
  - Now supports "bagging" (bootstrap aggregation) estimator.
  - New arguments `bagging`, `radius`, `adaptive`, `adjust`, `alpha`.
  - Behaviour improved in cases where `f` is extremely close to 0 or 1.
- `density.psp`:
  - Now accepts argument `rule.pix` passed to `owin2mask`.
  - New argument `method`.
  - Accelerated by 1 to 2 orders of magnitude.
- `density.splitppp`: New argument `weights`.
- `dfbetas.ppm`:
  - For Gibbs models, memory usage has been dramatically reduced, so the code can handle larger datasets and finer quadrature schemes.
  - Increased the default resolution of the pixel images. Spatial resolution can now be controlled by the arguments `dimyx`, `eps`.
  - Recognises argument `rule.eps` passed to `as.mask`.
- diagnostic functions:
  - The following functions have new formal argument `rmax`:

    `Gcom`, `Kcom`, `psstA`, `psstG`, `psst`
  - The following functions accept argument `rmax`:

    `Gres`, `Kres`
  - The following functions accept `correction="all"`:

Gcom, Gres, Kcom, Kres

- diagnose.ppm:
  - This is now a method for the generic diagnose.
  - Infinite values of rbord are now ignored and treated as zero. This ensures that diagnose.ppm has a sensible default when the fitted model has infinite reach.
  - Accelerated, when type="inverse", for models without a hard core.
- diagnose.ppm, plot.diagppm:
  - New arguments col.neg, col.smooth control the colour maps.
  - Accelerated, when type="inverse", for models without a hard core.
- diameter.owin: accelerated when the window is a rectangle.
- dilation.ppp: Improved geometrical accuracy. Now accepts arguments to control resolution of polygonal approximation.
- dirichletEdges: New argument clip.
- discretise:
  - New argument move.points determines whether the point coordinates are also discretised.
  - New argument rule.eps
- disc: New argument type allows the user to specify an inscribed polygon or a circumscribed polygon.
- discs:
  - Now accepts a single numeric value for radii.
  - New argument npoly.
  - Accelerated in some cases.
- distcdf:
  - Arguments which are NULL will be treated as missing.
  - New argument savedenom.
- distfun:
  - When the user calls a distance function that was created by distfun, the user may now give a ppp or lpp object for the argument x, instead of giving two coordinate vectors x and y.
  - New argument rule.eps
- distfun.lpp:
  - New argument k allows computation of k-th nearest point.
  - Computation accelerated.
- distfun.owin: New argument signed.
- distmap.owin:
  - New argument signed.
  - New argument connect.
  - Behaviour has been altered so that, when X is a binary mask, the results of distmap(X, invert=TRUE) and distmap(complement.owin(X)) are identical. This affects a few pixels close to the edge of the frame.
- distmap.ppp:
  - New option squared=TRUE returns the squared distances, reducing computation time.
  - New option extras=FALSE avoids returning additional attributes, reducing computation time.
  - New argument clip.
- distmap.psp: New arguments extras and clip.

- dppm: Changed precedence rule for handling the algorithm parameters in the minimum contrast algorithm. Individually-named arguments q,p,rmax,rmin now take precedence over entries with the same names in the list ctrl.
- duplicated.ppp: accelerated.
- edge.Trans: New argument gW for efficiency.
- edge.Ripley:
  - New argument bdistX for efficiency.
  - Responds more quickly to user interrupt signal.
- eem: The function eem is now generic, with methods for ppm and slrm. The function previously named eem is now called eem.ppm.
- effectfun:
  - Improved defaults for a recursive partition model (class "rppm"). The user only needs to provide values for those covariates which are used in the decision tree. If the entire tree involves only one variable, then covname is not needed.
  - If a spatial covariate is constant, its value does not need to be provided in the call to effectfun.
  - Now works for ppm, kppm, lppm, dppm, rppm and profilepl objects.
  - New argument nvalues.
  - Standard error calculation (se.fit=TRUE) now works for models fitted with use.gam=TRUE.
- Emark: The result now has an attribute smooth.args containing the smoothing parameters used.
- envelope:
  - Argument simulate can be a point process model of class "ppm", "kppm", "dppm", "slrm", "clusterprocess" or "detpointprocfamily".
  - New argument theoryfun can be used to provide the theoretical value of the summary function.
  - All methods for envelope now accept a summary function in which the function argument is not named r. This includes functions such as transect.im and roc.
  - New argument clamp gives greater control over one-sided envelopes.
  - New argument funargs
  - New argument scale allows global envelopes to have width proportional to a specified function of r, rather than constant width.
  - New argument funYargs contains arguments to the summary function when applied to the data pattern only.
  - The argument simulate can now be a function (such as rlabel). The function will be applied repeatedly to the original data pattern.
  - rejectNA and silent.
- envelope.lpp, envelope.lppm:
  - New argument theoryfun can be used to provide the theoretical value of the summary function.
  - New arguments fix.n and fix.marks allow envelopes to be computed using simulations conditional on the observed number of points.
  - New arguments maxnerr, rejectNA and silent.
- envelope.pp3:
  - Marked point patterns are accepted.
  - New arguments fix.n and fix.marks determine whether the simulated patterns have a fixed or random number of points.
- erosion.owin: when shrink.frame=TRUE (the default), the frame of the resulting window is always the erosion of the original frame by distance r, as promised in the help file.
- eval.im: New argument warn.
- eval.linim: New argument warn.

- `eval.fasp`: automatically generated labels have been improved.
- `ewcdf`:
  - Argument `weights` can now be `NULL`.
  - New arguments `normalise` and `adjust`.
  - Computation accelerated.
  - The result does not inherit class `"ecdf"` if `normalise=FALSE`.
- `Fest`:
  - New argument `rmax`.
  - Additional checks for errors in input data.
- `Finhom`:
  - New argument `rmax`.
  - A warning is issued if bias is likely to occur because of undersmoothing.
  - New arguments `warn.bias` and `savelambda`.
- `fastFindInterval`: New argument `left.open` controls whether intervals are left-open and right-closed, or left-closed and right-open.
- `fitted.lppm`: New argument `leaveoneout` allows leave-one-out computation of fitted value.
- `fitted.ppm`:
  - New option, `type="link"`.
  - New argument `ignore.hardcore`.
- `fitted.slrm`:
  - Argument `leaveoneout` is no longer ignored when `dataonly=FALSE`; it now supports "leave-one-pixel-out" prediction.
  - New argument `type` allows calculation of fitted probabilities, intensities or link function values.
  - New arguments `dataonly` and `leaveoneout` allow calculation of fitted values at the data points only, using leave-one-out calculation if desired.
- `FmultiInhom`: New argument `rmax`.
- `funxy`:
  - When the user calls a function that was created by `funxy`, the user may now give a `ppp` or `lpp` object for the argument `x`, instead of giving two coordinate vectors `x` and `y`.
  - Functions of class `"funxy"` can now be applied to quadrature schemes.
  - The result now has a `unitname`, inherited from the argument `W`.
- `Gcross`:
  - New argument `rmax`.
  - Function labels (shown on the plot legend) have been improved when `i = j`.
- `Geyer`: The saturation parameter `sat` can now be less than 1.
- `Ginhom`:
  - A warning is issued if bias is likely to occur because of undersmoothing.
  - New arguments `warn.bias` and `savelambda`.
- `grow.rectangle`: New argument `fraction`.
- `harmonise.im`:
  - Accelerated in the common case where all rasters are identical.
  - The result belongs to classes `solist` and `imlist` so that it can be plotted.

- `Hest`:
  - Argument `X` can now be a pixel image with logical values.
  - New argument `W`. [Based on code by Kassel Hingee.]
  - Additional checks for errors in input data.
- `hist.im`: New argument `xname`.
- `hyperframe`:
  - An object of class `Surv` from the `survival` package is now treated as a single column of data (mimicking the behaviour of `data.frame`)
  - The formal default value of `stringsAsFactors` has been changed to `NULL` to conform to changes in R. (The actual default value is `TRUE` for `R < 4.1.0` and `FALSE` for `R >= 4.1.0`).
  - Removed slight inconsistencies in the internal format produced in different cases.
- `identify.ppp`: Automatically starts a new plot device if none is open.
- `identify.psp`:
  - New arguments `paint` and `paint.args`.
  - Identified segments are highlighted.
  - Automatically starts a new plot device if none is open.
  - Improved placement of labels.
  - Arguments can be passed to `text.default` to control the plotting of labels.
- `idw`: Standard errors can now be calculated by setting `se=TRUE`.
- `imcov`: the name of the unit of length is preserved.
- `im.apply`:
  - Now handles vector-valued functions.
  - Computation accelerated
  - New argument `fun.handles.na`
  - New argument `check`
  - Improved algorithm avoids violating memory limits, handles larger datasets.
- `influence.ppm`: For Gibbs models, memory usage has been dramatically reduced, so the code can handle larger datasets and finer quadrature schemes.
- `integral.im`:
  - New argument `weight` specifies a weight function for the integration.
  - Accelerated in the case where `domain` is a tessellation.
- `integral.linfun`:
  - New argument `exact`. If `exact=TRUE`, a more accurate algorithm is used.
  - Argument `domain` can be a tessellation on a network (class `lintess`).
  - New argument `weight` specifies a weight function for the integration.
  - New argument `delta` controls step length of approximation to integral.
  - New argument `nd` controls approximate number of sample points used to calculate integral.
  - Argument `domain` can be a tessellation.
  - Now handles complex-valued functions.
- `integral.linim`:
  - Argument `domain` can be a tessellation on a network (class `lintess`).
  - New argument `weight` specifies a weight function for the integration.
  - Argument `domain` can be a tessellation.

  - Now handles complex-valued functions.

- `integral.msr`: New argument `weight` specifies a weight (integrand) for the integration.
- `integral.ssf`: Argument `domain` can be a tessellation.
- `intensity.ppp`: Argument `weights` can be logical-valued.
- `intensity.ppm`:
  - Intensity approximation is now implemented for area-interaction model, and Geyer saturation model.
  - Can now calculate the Coeurjolly-Lavancier DPP approximation of intensity. [Code kindly contributed by Frederic Lavancier]
  - New argument `approx` specifies the choice of approximation.
- `interp.im`: New argument `bilinear`.
- `intersect.lintess`: Can also compute the intersection between a two-dimensional tessellation and a linear network (yielding a tessellation on the network).
- `intersect.owin`: New argument `rescue`.
- `intersect.tess`:
  - Now handles marks of any kind (vector, list, data frame or hyperframe).
  - New argument `keepempty`.
- `invoke.symbolmap`: new argument `angleref`.
- `iplot`, `iplot.ppp`, `iplot.layered`, `iplot.linnet`, `iplot.default`: These interactive plotting functions have been removed from `spatstat` into a new package `spatstat.gui`.
- `ippm`:
  - Accelerated.
  - The internal format of the result has been extended slightly.
  - Improved defaults for numerical algorithm parameters.
- `istat`: This interactive analysis function has been removed from `spatstat` into a new package `spatstat.gui`.
- `Jcross`: Function labels (shown on the plot legend) have been improved when `i = j`.
- `Jfox`: new argument `warn.trim`.
- `Jinhom`:
  - A warning is issued if bias is likely to occur because of undersmoothing.
  - New arguments `warn.bias` and `savelambda`.
- `Kcross`:
  - Function labels (shown on the plot legend) have been improved when `i = j`.
  - Now accepts the option `correction="periodic"` to compute the periodic (toroidal) edge correction estimate.
- `Kcross.inhom`, `Kdot.inhom`, `Kmulti.inhom`:
  - Now recognise argument `ratio`. If `ratio=TRUE`, the numerator and denominator of each estimate are saved, for use in analysing replicated point patterns.
  - Interpretation of arguments `lambdaI`, `lambdaJ`, `lambdaX`, `lambdadot` has changed when they are fitted point process models. If the model is a multitype point process then it is assumed to be a model for the entire point pattern `X`, while if the model is an unmarked point process then it is assumed to be a model for the relevant subset of the data.
  - These functions now allow intensity values to be given by a fitted point process model.
  - New arguments `update`, `leaveoneout`, `lambdaX`.
  - Leave-one-out calculation is now implemented when `lambbdaX` is a fitted model of class `"dppm"`.
- `Kdot`: Now accepts the option `correction="periodic"` to compute the periodic (toroidal) edge correction estimate.

- `Kest`
  - Accelerated computation (for translation and rigid corrections) when window is an irregular shape.
  - Calculation of isotropic edge correction for polygonal windows has changed slightly. Results are believed to be more accurate. Computation has been accelerated by about 20 percent in typical cases.
  - Now accepts the option `correction="periodic"` to compute the periodic (toroidal) edge correction estimate.
- `Kest.fft`: Now has `...` arguments allowing control of spatial resolution.
- `Kinhom`:
  - New argument `ratio`.
  - Stops gracefully if `lambda` contains any zero values.
  - Leave-one-out calculation is implemented when `lambda` is a fitted model of class `"dppm"`.
- `Kmulti`: Now accepts the option `correction="periodic"` to compute the periodic (toroidal) edge correction estimate.
- `kernel.moment`:
  - New arguments `mean` and `sd`.
  - Computation accelerated for `kernel='cosine'` or `'optcosine'`.
  - All cases are now computed using analytic expressions, for `m=0,1,2`.
- `kppm`:
  - New option: `method="waag"` fits the model by maximising Waagepetersen's (2007) second order composite likelihood
  - The code for fitting log-Gaussian Cox process models (`clusters="LGCP"`) has been re-implemented without using the package `RandomFields`. The current code supports the `"exponential"`, `"gauss"`, `"stable"`, `"gencauchy"` and `"matern"` covariance models.
  - Computation accelerated when `method="palm"` or `method="clik2"`. [Kindly contributed by Bethany Macdonald.]
  - New argument `trajectory` specifies whether to save the history of function evaluations performed by the optimization algorithm.
  - New argument `penalised` supports penalised model-fitting with a penalty against extremely large or small values of the cluster scale.
  - New arguments `ppm.improve.type` and `ppm.improve.args`.
  - The first order trend is fitted using a regularized fitting algorithm when `ppm.improve.type="enet"`.
  - New default settings ensure greater numerical stability of the optimization algorithm against the effects of the scale of the spatial coordinates. New argument `stabilize` specifies whether the optimization algorithm should be numerically stabilized.
  - Fitting a model with `clusters="LGCP"` no longer requires the package `RandomFields` to be loaded explicitly.
  - New argument `algorithm` specifies the choice of optimisation algorithm.
  - Left hand side of formula can now involve entries in the list `data`.
  - refuses to fit a log-Gaussian Cox model with anisotropic covariance.
  - A warning about infinite values of the summary function no longer occurs when the default settings are used. Also affects `mincontrast`, `cauchy.estpcf`, `lgcp.estpcf`, `matclust.estpcf`, `thomas.estpcf`, `vargamma.estpcf`.
  - Changed precedence rule for handling the algorithm parameters in the minimum contrast algorithm. Individually-named arguments `q,p,rmax,rmin` now take precedence over entries with the same names in the list `ctrl`.
  - Improved printed output.
  - Improved numerical robustness.
- `latest.news`: Now prints news documentation for the current major version, by default. New argument `major`.
- `layout.boxes`: Argument `aspect` can be `NA` or `Inf` indicating that the aspect ratio of the boxes is unconstrained.
- `Lcross`: Now accepts the option `correction="periodic"` to compute the periodic (toroidal) edge correction estimate.

- `Ldot`: Now accepts the option `correction="periodic"` to compute the periodic (toroidal) edge correction estimate.
- `Lcross.inhom`, `Ldot.inhom`: These functions now allow intensity values to be given by a fitted point process model. New arguments `update`, `leaveoneout`, `lambdaX`.
- `lengths.psp`:
  - New argument `squared`.
  - This function will soon be Deprecated in favour of the new name `lengths_psp`.
- `Lest`, `Linhom`, `Ldot`, `Lcross`, `Ldot.inhom`, `Lcross.inhom`: These summary functions now have explicit argument `"correction"`.
- `levelset`:
  - This function is now generic and can be applied to pixel images or functions.
  - For some kinds of functions, the level set is calculated more accurately using analytic geometry and returned as a polygonal window. This includes
    * Functions of class `"distfun"` created bt `distfun` where the level set is a dilation of the original object
    * Functions of class `tessfun` created by `as.function.tess` where the level set is a union of tiles of the tessellation.
- `leverage.ppm`:
  - For Gibbs models, memory usage has been dramatically reduced, so the code can handle larger datasets and finer quadrature schemes.
  - Increased the default resolution of the pixel images. Spatial resolution can now be controlled by the arguments `dimyx`, `eps`.
  - Recognises argument `rule.eps` passed to `as.mask`.
- `leverage.ppm`, `influence.ppm`, `dfbetas.ppm`:
  - These methods now work for models that were fitted by logistic composite likelihood (`method='logi'`).
  - Computation has been vastly accelerated for models with Geyer interaction fitted using isotropic or translation edge corrections.
  - Faster computation in many cases.
  - Virtually all models and edge corrections are now supported, using a "brute force" algorithm. This can be slow in some cases.
- `lgcp.estK`, `lgcp.estpcf`: This code for fitting log-Gaussian Cox process models has been re-implemented without using the package `RandomFields`. The current code supports the `"exponential"`, `"gauss"`, `"stable"`, `"gencauchy"` and `"matern"` covariance models.
- `lineardisc`:
  - New argument `add`.
  - Default plotting behaviour has changed.
- `linearK`, `linearpcf` and relatives:
  - substantially accelerated.
  - ratio calculations are now supported.
  - new argument `ratio`.
- `linearKEuclidInhom`, `linearpcfEuclidInhom`: Argument `lambda=NULL` is now interpreted to mean that the intensity should be estimated by kernel smoothing. A warning is issued that this is different from the previous behaviour.
- `linearKinhom`: new argument `normpower`.
- `linearKdot.inhom`, `linearpcfdot.inhom`:
  - When argument `lambdaI` or `lambdadot` is a point process model, the interpretation now depends on whether the model is multitype or unmarked. A multitype point process model will be interpreted as providing the intensity for each possible type of point. An unmarked point process model will be interpreted as providing the intensity only for the relevant type of point (type `i` only, or total intensity for all types of points).

  - Argument `lambdaI=NULL` or `lambdadot=NULL` is now interpreted to mean that the intensity should be estimated by kernel smoothing.

- `linearKcross.inhom`, `linearpcfcross.inhom`
  - When argument `lambdaI`, `lambdaJ` or `lambdadot` is a point process model, the interpretation now depends on whether the model is multitype or unmarked. A multitype point process model will be interpreted as providing the intensity for each possible type of point. An unmarked point process model will be interpreted as providing the intensity only for the relevant type of point (type `i` only, type `j` only, or total intensity for all types of points).
  - Argument `lambdaI=NULL` or `lambdaJ=NULL` is now interpreted to mean that the intensity should be estimated by kernel smoothing.

- `linearKinhom`, `linearpcfinhom`:
  - Argument `lambda=NULL` is now interpreted to mean that the intensity should be estimated by kernel smoothing. A warning is issued that this is different from the previous behaviour.
  - Changed behaviour when `lambda` is a fitted model.
  - New arguments `update` and `leaveoneout`.

- `linearpcf`: new argument `normpower`.

- `linearpcfinhom`: New arguments `adjust.sigma`, `bw` and `adjust.bw`.

- `linearpcfcross.inhom`, `linearpcfdot.inhom`: New arguments `adjust.sigma`, `bw` and `adjust.bw`.

- `linearpcfEuclidInhom`: New arguments `adjust.sigma`, `bw` and `adjust.bw`.

- `linequad`: Argument `Y` can be a linear network.

- `linim`:
  - The image `Z` is now automatically restricted to the network.
  - New argument `restrict`.

- `linnet`:
  - The internal format of a `linnet` (linear network) object has been changed. Existing datasets of class `linnet` are still supported. However, computation will be faster if they are converted to the new format. To convert a linnet object `L` to the new format, use `L <- as.linnet(L)`.
  - If the argument `edges` is given, then this argument now determines the ordering of the sequence of line segments. For example, the `i`-th row of `edges` specifies the `i`-th line segment in `as.psp(L)`.
  - New argument `warn`.
  - When argument `edges` is specified, the code now checks whether any edges are duplicated.

- `lintess`:
  - Argument `df` can be missing or `NULL`, resulting in a tesellation with only one tile.
  - Tessellations can now have marks. New argument `marks`.

- `localpcf`: New argument `rvalue`.

- `localpcfinhom`:
  - New arguments `update`, `leaveoneout`, `rvalue`.

- `logLik.ppm`:
  - New argument `absolute`.
  - The warning about pseudolikelihood ('log likelihood not available') is given only once, and is not repeated in subsequent calls, within a spatstat session.

- `logLik.mppm`: new argument `warn`.

- `lohboot`:
  - Algorithm has been corrected and extended thanks to Christophe Biscio and Rasmus Waagepetersen.

  - New arguments `block`, `basicboot`, `Vcorrection`.
  - Accelerated when the window is a rectangle.
  - Now works for multitype K functions `Kcross`, `Kdot`, `Lcross`, `Ldot`, `Kcross.inhom`, `Lcross.inhom`
  - Confidence bands for `Lest`, `Linhom`, `Lcross`, `Ldot`, `Lcross.inhom` are now computed differently. First a confidence band is computed for the corresponding K function `Kest`, `Kinhom`, `Kcross`, `Kdot`, `Kcross.inhom` respectively. Then this is transformed to a confidence band for the L function by applying the square root transformation.

- `lpp`:
  - The internal format of an `lpp` object has been changed. Existing datasets of class `lpp` are still supported. However, computation will be faster if they are converted to the new format. To convert an `lpp` object `X` to the new format, use `X <- as.lpp(X)`.
  - `X` can be missing or `NULL`, resulting in an empty point pattern.
  - Now handles the case where coordinates `seg` and `tp` are given but `x` and `y` are missing.

- `lppm`:
  - The model formula may involve the local coordinates `seg` and `tp`.
  - Covariates can be objects of class `lintess`.
  - New argument `random` controls placement of dummy points.
  - Computation accelerated.

- `lurking.lppm`:
  - Argument `covariate` can be a `function(x,y)` or one of the strings `"x"` or `"y"`.
  - When the covariate is a cartesian coordinate or a distance function, the unit of length is stated on the horizontal axis label of the plot.

- `lurking.ppm`:
  - Argument `covariate` can be a `function(x,y)` or one of the strings `"x"` or `"y"`.
  - When the covariate is a cartesian coordinate or a distance function, the unit of length is stated on the horizontal axis label of the plot.
  - accelerated.

- `lurking.slrm`:
  - Argument `covariate` can be a `function(x,y)` or one of the strings `"x"` or `"y"`.
  - When the covariate is a cartesian coordinate or a distance function, the unit of length is stated on the horizontal axis label of the plot.

- `lut`:
  - argument `outputs` may have length 1, representing a lookup table in which all data values are mapped to the same output value.
  - New arguments `compress`, `decompress` for nonlinear lookup tables.

- `markconnect`:
  - The result now has an attribute `smooth.args` containing the smoothing parameters used.
  - Accepts the argument `weights` which is passed to `markcorr`.

- `markcorr`:
  - Argument `weights` can be logical-valued.
  - The result now has an attribute `smooth.args` containing the smoothing parameters used.
  - New argument `fadjust`.
  - New argument `weights` allows computation of the weighted version of the mark correlation function. Weights can be an expression to be evaluated, or a function, or a pixel image, or a numeric vector. (NOTE: interpretation of `weights` has changed in `spatstat.explore` version `3.6-1`)

  - Now allows negative mark values, when `normalise=FALSE`.

- `markcrosscorr`:
  - Argument `weights` can be logical-valued.
  - New argument `weights`.
  - Now allows negative mark values, when `normalise=FALSE`.
- `marks<-.tess`: A tessellation can now have any kind of marks (vector, list, data frame or hyperframe).
- `markstat`: Now works for point patterns in three dimensions (class `"pp3"`) and point patterns on a network (class `"lpp"`).
- `marktable`: Now works for point patterns in three dimensions (class `"pp3"`) and point patterns on a network (class `"lpp"`).
- `markvario`:
  - The result now has an attribute `smooth.args` containing the smoothing parameters used.
  - Accepts the argument `weights` which is passed to `markcorr`.
- `mincontrast`: New argument `action.bad.values` specifies what action is taken when the summary function produces `NA` or `NaN` or infinite values.
- `minnndist`, `maxnndist`: New argument `by` makes it possible to find the minimum or maximum nearest neighbour distance between each pair of possible types in a multitype pattern.
- `model.images.ppm`: Now recognises arguments passed to `as.mask` to control the pixel raster for the images.
- `mppm`:
  - The responses can be point patterns on a linear network (objects of class `"lpp"`).
  - Recognises argument `quad.args`.
  - Now handles models with a random effect component. (This is covered in [2, Chap. 16].)
  - New argument `random` is a formula specifying the random effect. (This is covered in [2, Chap. 16].)
  - Performs more checks for consistency of the input data.
  - New arguments `gcontrol` and `reltol.pql` control the fitting algorithm.
  - New argument `weights` specifies case weights for each row of data.
- `msr`: Infinite and `NA` values are now detected (if `check=TRUE`) and are reset to zero, with a warning.
- `multiplicity`: Methods for `multiplicity` can now handle objects of larger size.
- `nbfires`:
  - the unit of length for the coordinates is now specified in this dataset.
  - This dataset now includes information about the different land and sea borders of New Brunswick.
- `nncorr,nnmean,nnvario`: New argument `na.action`.
- `nncross.default`:
  - New option `squared=TRUE` returns the squared nearest neighbour distances, reducing computation time.
- `nncross.lpp`:
  - New argument `k` allows computation of k-th nearest point.
  - Computation accelerated.
- `nncross.ppp`:
  - New option `squared=TRUE` returns the squared nearest neighbour distances, reducing computation time.
  - slightly accelerated.
  - When `X` is a point pattern and `Y` is a line segment pattern, higher order neighbours (`k > 1`) are now supported.

- `nndist.default`: New option `squared=TRUE` returns the squared nearest neighbour distances, reducing computation time.
- `nndist.ppp`:
  - New option `squared=TRUE` returns the squared nearest neighbour distances, reducing computation time.
  - New argument `proper`.
- `nndist.pp3`: New argument `by` allows computation of the nearest distance to each group of points.
- `nndist.ppx`: New argument `by` allows computation of the nearest distance to each group of points.
- `nndist.lpp`:
  - New argument `k` allows computation of k-th nearest point.
  - new argument `by` allows computation of the nearest distance to each group of points.
  - Computation accelerated.
- `nnmap.ppp`: New option `squared=TRUE` returns the squared distances, reducing computation time.
- `nnwhich.lpp`:
  - New argument `k` allows computation of k-th nearest point.
  - new argument `by` allows computation of the nearest distance to each group of points.
  - Computation accelerated.
- `nnfun`: new argument `rule.eps`.

  `nnfun.lpp`:
  - New argument `k`.
  - New argument `value` specifies whether to return the index of the nearest neighbour or the mark value of the nearest neighbour.

  `nnfun.ppp`:
  - New argument `value` specifies whether to return the index of the nearest neighbour or the mark value of the nearest neighbour.

  `nnfun.psp`:
  - New argument `value` specifies whether to return the index of the nearest neighbour or the mark value of the nearest neighbour.
- `objsurf` methods: Arguments `xlim,ylim` renamed `par1lim,par2lim`.
- `owin`:
  - accelerated in many cases.
  - If argument `mask` is a logical matrix, `NA` entries will be accepted, and converted to `FALSE`.
- `owin2mask`:
  - Arguments `'...'` that are not recognised by `as.mask` are now silently ignored, rather than causing an error.
  - New options `op="majority"` and `op="minority"`. If `op="majority"`, a pixel belongs to the resulting mask if at least half of the pixel area is covered by the window.
  - Changed formal arguments. First argument is now called `w`. Formal argument `op` renamed `rule.pix`. All arguments must be given by name except for the first argument.
- `padimage`: New argument `W` allows an image to be padded out to fill any window.
- `pairdist.lpp`: Now handles much larger networks, using the sparse representation of the network.
- `pairorient`: Default edge corrections now include `"bord.modif"`.
- `pairs.im`: new argument `drop`.
- `palmdiagnose`:

  - Now accepts objects of class `dppm` as well as `kppm`.
  - New argument `delta` as an alternative to `breaks`.

- `parres`: the argument `covariate` is allowed to be missing if the model only depends on one covariate.

- `pcf.ppp`:
  - New option `zerocor="stableJF"` is a modification of `zerocor="JonesFoster"` designed to be more stable to numerical error in small samples.
  - In cases where the default value of `zerocor` was `"JonesFoster"`, the new default is `zerocor="stableJF"`.
  - The default value of `divisor` has temporarily reverted to `"r"` pending further investigation.
  - Changed rule for converting bandwidths when `divisor="a"`.
  - The default value of `divisor` has changed to `"a"`, so that the new area-based estimator is the default.
  - The default value of `zerocor` is now `"JonesFoster"` for small patterns and `"convolution"` for larger patterns.
  - New argument `nsmall` determines the threshold for large patterns.
  - New option: `divisor="a"` calculates a new estimator of the pcf with improved performance.
  - New argument `ratio` allows several estimates of pcf to be pooled.
  - Now calculates an analytic approximation to the variance of the estimate of the pair correlation function (when `var.approx=TRUE`).
  - Now returns the smoothing bandwidth used, as an attribute of the result.
  - New argument `close` for advanced use.
  - Now accepts `correction="none"`.

- `pcfcross`, `pcfdot`:
  - In cases where the default value of `zerocor` was `"JonesFoster"`, the new default is `zerocor="stableJF"`.
  - Shortened list of formal arguments (other arguments pass to `pcfmulti`).

- `pcfcross`, `pcfdot`, `pcfmulti`:
  - The default value of `divisor` has temporarily reverted to `"r"` pending further investigation.
  - The default estimation method has changed. The new default is based on squared distances (`divisor="a"`).
  - New argument `ratio` makes it possible to save the numerator and denominator of the function estimates, so that estimates can be pooled.

- `pcfcross.inhom`, `pcfdot.inhom`:
  - In cases where the default value of `zerocor` was `"JonesFoster"`, the new default is `zerocor="stableJF"`.
  - Now recognise argument `ratio`. If `ratio=TRUE`, the numerator and denominator of each estimate are saved, for use in analysing replicated point patterns.
  - Shortened list of formal arguments (other arguments pass to `pcfmulti.inhom`).
  - The default value of `divisor` has temporarily reverted to `"r"` pending further investigation.
  - The default estimation method has changed. The new default is based on squared distances (`divisor="a"`).
  - New arguments `adjust.sigma` and `adjust.bw` allow separate adjustment of the one-dimensional smoothing bandwidth `bw` and the spatial smoothing bandwidth `sigma`.

- `pcfinhom`:
  - New option `zerocor="stableJF"` is a modification of `zerocor="JonesFoster"` designed to be more stable to numerical error in small samples.
  - In cases where the default value of `zerocor` was `"JonesFoster"`, the new default is `zerocor="stableJF"`.
  - The default value of `divisor` has temporarily reverted to `"r"` pending further investigation.
  - The default value of `divisor` has changed to `"a"`, so that the new area-based estimator is the default.
  - The default value of `zerocor` is now `"JonesFoster"` for small patterns and `"convolution"` for larger patterns.
  - New argument `nsmall` determines the threshold for large patterns.
  - Default value of `normpower` has changed to 2.
  - New option: `divisor="a"` calculates a new estimator of the pcf with improved performance.

- New arguments `adjust.sigma` and `adjust.bw` allow separate adjustment of the one-dimensional smoothing bandwidth `bw` and the spatial smoothing bandwidth `sigma`.
- New argument `close` for advanced use.
- Default behaviour is changed when `lambda` is a fitted model. The default is now to re-fit the model to the data before computing pcf. New arguments `update` and `leaveoneout` control this.
- New argument `close` for advanced use.
- Now handles `correction="good"`
- Leave-one-out calculation is implemented when `lambda` is a fitted model of class `"dppm"`.

- `pcfmulti`:
  - New option `zerocor="stableJF"` is a modification of `zerocor="JonesFoster"` designed to be more stable to numerical error in small samples.
  - In cases where the default value of `zerocor` was `"JonesFoster"`, the new default is `zerocor="stableJF"`.
  - New arguments `h`, `bw.args`, `adjust`, `zerocor`, `nsmall`, `gref`, `tau`, `close`.
- `pcfmulti`:
  - Interpretation of arguments `lambdaI`, `lambdaJ`, `lambdaX`, `lambdadot` has changed when they are fitted point process models. If the model is a multitype point process then it is assumed to be a model for the entire point pattern `X`, while if the model is an unmarked point process then it is assumed to be a model for the relevant subset of the data.
  - New arguments `h`, `bw.args`, `adjust`, `zerocor`, `nsmall`, `gref`, `tau`, `close`.
- `pcfmulti.inhom`:
  - New option `zerocor="stableJF"` is a modification of `zerocor="JonesFoster"` designed to be more stable to numerical error in small samples.
  - In cases where the default value of `zerocor` was `"JonesFoster"`, the new default is `zerocor="stableJF"`.
- `persist`: Now accepts an object of class `"clustermodel"`.
- `persp.funxy`: Improved z-axis label.
- `persp.im`:
  - The pixel values of `colin` can be colour values. New argument `valuesAreColours`.
  - Now recognises argument `adj.main` controlling the position of main title.
  - Improved appearance when `apron=TRUE` and the image domain is not a rectangle.
- `persp.linim`:
  - Improved behaviour close to vertices of the network.
  - New arguments `extrapolate` and `zadjust`.
  - Argument `col.base` can be a pixel image, allowing a colour image to be plotted on the horizontal plane at height zero.
- `persp.ppp`:
  - Now plots the `zlab` label alongside the vertical scale bar.
  - Argument `col.base` can be a pixel image, which will be rendered as a colour image in perspective view on the horizontal plane.
  - New argument `show.window`.
  - Now recognises argument `adj.main` controlling the position of main title.
- `pixellate.owin`:
  - Now accepts the argument `rule.pix` passed to `owin2mask`. This makes it possible to ensure that boundary pixels which partially intersect the window are included in the resulting pixel image, by setting `rule.pix="cover"`.
- `pixellate.ppp`:

  - Now accepts the argument `rule.pix` passed to `owin2mask`. This makes it possible to ensure that boundary pixels which partially intersect the window are included in the resulting pixel image, by setting `rule.pix="cover"`.
  - If the pattern is empty, the result is an integer-valued image (by default) for consistency with the results for non-empty patterns.
  - Accelerated in the case where weights are given.
  - New arguments `fractional` and `preserve` for more accurate discretisation.
  - New argument `savemap`.
- `pixellate.psp`:
  - Now accepts the argument `rule.pix` passed to `owin2mask`. This makes it possible to ensure that boundary pixels which partially intersect the window are included in the resulting pixel image, by setting `rule.pix="cover"`.
- `pixelquad`: Now accepts arguments passed to `as.mask` to control the pixel resolution.
- `plot.anylist`:
  - If a list entry `x[[i]]` belongs to class `"anylist"`, it will be expanded so that each entry `x[[i]][[j]]` will be plotted as a separate panel.
  - New arguments `panel.begin.args`, `panel.end.args`
  - Result is now an (invisible) list containing the result from executing the plot of each panel.
- `plot.bermantest`: Improved layout for plots of Berman's `Z2` test.
- `plot.colourmap`:
  - New argument `do.plot`
  - New formal argument `side`.
  - Now handles a colour map for a zero-length interval [a,a]
  - New argument `increasing` specifies whether the colours are displayed in order left-to-right/bottom-to-top.
  - Changed default behaviour for discrete colour maps when `vertical=FALSE`.
  - New argument `nticks` controls the number of axis tick marks when the colourmap is defined on a continuous range of numerical values.
  - New argument `box` controls whether a box will be drawn around the colours.
  - New argument `at` determines the position of tick marks on the axis.
- `plot.fv`:
  - New argument `do.plot`.
  - New argument `clip.xlim`.
- `plot.im`:
  - New arguments `ribxlim`, `ribylim` control the position of the colour ribbon, overriding the default rule.
  - Improved rule for layout of text labels.
  - New argument `opacity` controls semi-transparency of colours.
  - The position of text next to the colour ribbon can now be controlled by `riblab$side`.
  - Placement of main title improved in some cases.
  - New arguments `background` and `clip.background` allow the user to specify an object that will be plotted before the image `x`, and will therefore appear underneath it.
  - New argument `drop.ribbon` determines whether a ribbon will be displayed in the case where the pixel values are all equal. Default behaviour has changed.
  - New argument `reverse.col` allows the sequence of colours to be reversed.
  - New argument `addcontour` specifies that contour lines should be drawn over the image plot.
  - Now handles complex-valued images.
  - New argument `workaround` to avoid a bug in some MacOS device drivers that causes the image to be displayed in the wrong spatial orientation.
  - The number of tick marks in the colour ribbon can now be controlled using the argument `nint` in `ribargs`.

  - Improved behaviour when all pixel values are `NA`.
  - Improved handling of tickmarks on colour ribbon.
  - Improved behaviour when the image values are almost constant.
  - New argument `riblab`.
  - Axes are prevented from extending outside the image rectangle.
  - New argument `zap`.
  - Some warnings are suppressed when `do.plot=FALSE`.
  - Return value has attribute `"at"` giving the position of the tick marks on the axis next to the colour ribbon.
- `plot.imlist`:
  - New argument `equal.scales`.
  - If `equal.ribbon=TRUE` and `equal.scales=TRUE`, the colour ribbon is now neatly aligned with the plotted images.
  - Result is now an (invisible) list containing the results from executing the plot of each panel.
- `plot.influence.ppm`: New argument `multiplot`.
- `plot.kppm`:
  - For models fitted by Palm likelihood or composite likelihood, the plot selected by `what="statistic"` displays the pair correlation function.
  - New arguments `pause` and `xname`.
  - The argument `what="all"` is now recognised: it selects all the available options. [This is also the default.]
- `plot.leverage.ppm`:
  - New arguments `multiplot` and `what`.
  - A contour line showing the average value of leverage is now drawn on the colour ribbon, as well as on the main image. New argument `args.contour`.
- `plot.linfun`:
  - Now passes arguments to the function being plotted.
  - A scale bar is now plotted when `style="width"`.
  - New argument `legend`.
  - The return value has a different format.
- `plot.linim`:
  - The return value has a different format.
  - New argument `fatten` improves visual appearance when `style="colour"`.
  - A scale bar is now plotted when `style="width"`.
  - When `style="width"`, negative values are plotted in red (by default). New argument `negative.args` controls this.
  - New argument `zlim` specifies the range of values to be mapped.
  - New explicit argument `box` determines whether to plot a bounding box; default is `FALSE` in all cases.
- `plot.linnet`:
  - New argument `adj.main`.
- `plot.lintess`:
  - Improved plot method, with more options.
  - Modified to display the marks attached to the tiles.
  - Options: `style=c("colour", "width", "image")`.
- `plot.lpp`:
  - New argument `adj.main`.

  - New argument `show.network`.
  - For a point pattern with continuous marks ("real numbers") the colour arguments `cols`, `fg`, `bg` can now be vectors of colour values, and will be used to determine the default colour map for the marks.
  - If `shape="crosssticks"`, the points will be drawn as short line segments perpendicular to the network.

- `plot.mppm`:
  - New argument `main`.
  - New argument `se`.

- `plot.msr`:
  - Now handles multitype measures.
  - New argument `multiplot`.
  - New argument `massthresh`.
  - New arguments `equal.markscale` and `equal.ribbon`.

- `plot.onearrow`: Graphical parameters, specified when the object was created, are now taken as the defaults for graphical parameters to the plot.

- `plot.owin`:
  - New argument `use.polypath` controls how to plot a filled polygon when it has holes.
  - New argument `adj.main` controls the justification of the text in the main title.
  - New argument `background` specifies an object (or a colour) that will be plotted before `x` is plotted, and will therefore appear underneath it.

- `plot.profilepl`: This function has now been documented, and the graphics improved.

- `plot.psp`:
  - New argument `scramble.cols`
  - Segments can be plotted with widths proportional to their mark values.
  - New argument `style`.
  - New argument `col` gives control over the colour map representing the values of marks attached to the segments.
  - The code for `style="width"` has been completely rewritten, so that it no longer depends on `plot.linim`, and is much more efficient.
  - The formal argument list has been extended.
  - New argument `background` specifies an object (or a colour) that will be plotted before `x` is plotted, and will therefore appear underneath it.

- `plot.pp3`: New arguments `box.front`, `box.back` control plotting of the box.

- `plot.ppp`:
  - New argument `show.rejects`.
  - New argument `scramble.cols`
  - New arguments `background` and `clip.background` allow the user to specify an object (or a colour value) that will be plotted before the point pattern `x`, and will therefore appear underneath it.
  - For multitype point patterns, a warning is issued if the plot legend does not represent every possible type of point due to space restrictions.
  - The default colour for the points is now a transparent grey, if this is supported by the plot device.
  - For a point pattern with continuous marks ("real numbers") the colour arguments `cols`, `fg`, `bg` can now be vectors of colour values, and will be used to determine the default colour map for the marks.
  - Now recognises graphics parameters for text, such as `family` and `srt`
  - When `clipwin` is given, any parts of the boundary of the window of `x` that lie inside `clipwin` will also be plotted.
  - Improved placement of symbol map legend when argument `symap` is given.

- `plot.tess`:

  - This plot method can now fill each tile with a different colour.
  - New arguments `do.col`, `values`, `col` and `ribargs`. Old argument `col` has been renamed `border` for consistency.
  - Now generates a separate plot panel for each column of marks, if `do.col=TRUE`.
  - New argument `multiplot`.
  - Changed the default values for `do.col` and `do.labels`.

- `plot.palmdiag`:
  - Improved calculation of y axis limits.
  - Improved rule for automatic placement of legend.

- `plot.profilepl` ,`plot.quadratcount`, `plot.quadrattest`, `plot.tess`: Now recognise graphics parameters for text, such as `family` and `srt`

- `plot.solist`:
  - Arguments `adorn.left`, `adorn.right`, `adorn.bottom`, `adorn.top` may now be objects of class `colourmap` or `symbolmap`.
  - New argument `adorn.args`.
  - When `equal.ribbon=TRUE`, the images may now be factor-valued or character-valued. Character-valued images will be converted to factor-valued images. The common colour map will combine the levels of all the factor images.
  - New arguments `panel.begin.args`, `panel.end.args`
  - Result is now an (invisible) list containing the result from executing the plot of each panel.

- `plot.studpermutest`: This existing function now has a help file.

- `plot.symbolmap`:
  - New argument `do.plot`.
  - New formal argument `side`.
  - New argument `colour.only` makes it possible to display only the colour map information in a symbol map.
  - New argument `warn`.
  - Issues a warning if the plot of a discrete symbol map does not represent every possible input value, due to space restrictions.
  - New argument `nsymbols` controls the number of symbols plotted.

- `plot.texturemap`: new formal argument `side`.

- `plot.yardstick`:
  - new argument `style` allows different styles of plotting a scale bar, including a zebra pattern (`style = "zebra"`).
  - New arguments `zebra.step`, `zebra.width`, `zebra.col`.

- `ponderosa`: In this installed dataset, the function `ponderosa.extra$plotit` has changed slightly (to accommodate the dependence on the package `spatstat.utils`).

- `polynom`: This function now has a help file.

- `pool.fv`:
  - The default plot of the pooled function no longer includes the variance curves.
  - New arguments `relabel` and `variance`.

- `pool.rat`: New arguments `weights`, `relabel` and `variance`.

- `ppm`:
  - Now supports regularized model-fitting.
  - Huang-Ogata approximate maximum likelihood can be applied to logistic fits.
  - New argument `improve.type`.
    * Option `method="ho"` is replaced by `improve.type="ho"`.

    - Regularized model-fitting is performed when `improve.type="enet"`.
    - Huang-Ogata approximate maximum likelihood can be applied to logistic fits by setting `method="logi"` and `improve.type="ho"`.
  - Argument `interaction` can now be a function that makes an interaction, such as `Poisson`, `Hardcore`, `MultiHard`.
  - Argument `subset` can now be a window (class `"owin"`) specifying the sub-region of data to which the model should be fitted.

- `ppm.ppp, ppm.quad`:
  - New argument `emend`, equivalent to `project`.
  - New arguments `subset` and `clipwin`.
  - New argument `quad.args` is a list of arguments passed to `quadscheme` to control the construction of the quadrature scheme.

- `ppmInfluence`: The result now belongs to class `ppmInfluence`, for which there are methods for `leverage`, `influence`, `dfbetas` which extract the desired component.

- `ppp`:
  - New argument `checkdup`.
  - If the coordinate vectors `x` and `y` contain `NA`, `NaN` or infinite values, these points are deleted with a warning, instead of causing a fatal error.

- `pp3`: New argument `marks`.

- `predict.kppm, residuals.kppm` Now issues a warning when the calculation ignores the cluster/Cox component and treats the model as if it were Poisson. (This currently happens in predict.kppm when se=TRUE or interval != "none", and in residuals.kppm when type != "raw").

- `predict.lppm`: Argument `locations` can now be an `lpp` object.

- `predict.mppm`:
  - The argument `type="all"` is now recognised: it selects all the available options. [This is also the default.]
  - Now supports multitype point process models.
  - Improved handling of argument `newdata`.

- `predict.ppm`:
  - Now recognises the arguments `dimyx` and `eps` for specifying the resolution of the grid of prediction points.
  - New argument `ignore.hardcore`.
  - Accelerated for models fitted with `method="VBlogi"`
  - Standard error calculation (`se=TRUE`) now works for models fitted with `use.gam=TRUE`.

- `predict.rhohat`: New argument `what` determines which value should be calculated: the function estimate, the upper/lower confidence limits, or the standard error.

- `predict.slrm`:
  - New argument `leaveoneout` supports "leave-one-pixel-out" prediction.
  - New argument `fast` determines algorithm used when `leaveoneout=TRUE`.

- `print.kppm`: Additional characteristics of the fitted model are reported, including the cluster strength `phi` and the sibling probability.

- `print.linim`: More information is printed.

- `print.lintess`:
  - Output improved.
  - Output includes information about marks.

- `print.lppm`: The name of the original point pattern dataset (to which the model was fitted) is now printed.

- `print.quad`: More information is printed.

- `print.rmhmodel`: More information is printed.
- `print.slrm`: Prints information about the pixel raster.
- `print.summary.owin`: Output can now be abbreviated by increasing the value of `spatstat.options('terse')`
- `print.summary.psp`: Output can now be abbreviated by increasing the value of `spatstat.options('terse')`
- `progressreport`
  - The estimated time of completion is also printed, if the remaining time is longer than 10 minutes.
  - Behaviour improved.
  - New arguments `state`, `tick`, `showtime`.
  - New option: `style="tk"`
  - New argument `formula` controls the calculation of estimated time remaining.
  - New argument `savehistory` specifies whether to save the elapsed times when the function was called.
- `projectsegment`: Accelerated by 20 percent.
- `projectset`: Supports projection to a polygonal window; new argument `polygonal`.
- `pseudoR2.ppm`, `pseudoR2.lppm`:
  - The null model now includes any offset terms, by default.
  - New argument `keepoffset`.
- `psib`: Now accepts an object of class `"clustermodel"`.
- `quadform`: This function has been moved to the sub-package `spatstat.sparse`.
- `quadratcount.ppp`:
  - Computation accelerated in some cases.
  - New argument `left.open` controls the treatment of data points which lie on the boundary between two rectangular quadrats.
- `quadrat.test.ppm`: Computation accelerated in some cases.
- `quantess`:
  - The covariate `Z` can now be `"rad"` or `"ang"` representing polar coordinates.
  - New argument `origin` specifies the origin of polar coordinates.
  - New argument `eps` controls the accuracy of the calculation.
- `quantile.ecdf`:
  - Now supports `type=4` (linear interpolation).
- `quantile.ewcdf`:
  - Now supports `type=4` (linear interpolation).
  - The function is now normalised to the range `[0,1]` before the quantiles are computed. This can be suppressed by setting `normalise=FALSE`.
- `qqplot.ppm` Argument `expr` can now be a list of point patterns, or an envelope object containing a list of point patterns.
- Most random generators: now accept `nsim=0` and return a zero-length list.
- `rbind.hyperframe`: The result now retains the `row.names` of the original arguments.
- `rcellnumber`: New argument `mu`.
- `rebound.owin`: Now preserves unitnames of the objects.
- `relrisk.lpp`:
  - Now handles a point pattern with multiple columns of marks.

  - New argument `weights`
  - New argument `normalise`

- `relrisk.ppp`:
  - Now handles a point pattern with multiple columns of marks.
  - Argument `weights` can be logical valued.
  - Now accepts argument `rule.pix` passed to `owin2mask`.
  - New argument `shrink` supports Bithell's shrinkage estimator.
  - New argument `normalise` controls whether the estimate will be normalised so that constant relative risk corresponds to the value 1.

- `rescale.owin`, `rescale.ppp`, `rescale.psp`: The geometrical type of the window is now preserved in all cases. (Previously if the window was polygonal but was equivalent to a rectangle, the rescaled window was a rectangle.)

- `reload.or.compute`: New argument `exclude` specifies which objects should not be saved.

- `residualMeasure`: More options for argument `lambda`.

- `relrisk.ppp`:
  - If `se=TRUE` and `at="pixels"`, the result belongs to class `solist`.
  - The arguments `adjust`, `edge`, `diggle` are now explicit formal arguments.
  - New argument `weights`.
  - Ratios which are close to $0/0$ are handled more effectively, reducing the likelihood of strange-looking plots when `sigma` is very small.
  - Issues a warning if numerical underflow is detected.
  - The interpretation of `weights` in the calculation of standard error has changed. New argument `wtype` controls this interpretation.
  - New argument `fudge` specifies a constant numeric value that will be added to each estimate of point process intensity before calculation of relative risk.

- `rgbim, hsvim`: New argument `A` controls the alpha (transparency) channel.

- `rgb2hex, col2hex, paletteindex, is.colour, samecolour, complementarycolour, is.grey, to.grey` These colour tools now handle transparent colours.

- `rgb2hex`: New argument `maxColorValue`

- `rhohat`:
  - Methods for `rhohat` now recognise the argument `delta` as an alternative to argument `breaks` when `smoother="piecewise`
  - The result now includes the "average" intensity $\overline{\rho}$.
  - New options `smoother="piecewise"` computes a piecewise-constant estimate of $\rho(z)$.
  - Nonparametric maximum likelihood estimation is now supported, assuming the intensity is a monotone function of the covariate.
  - New options `smoother="increasing"` and `smoother="decreasing"` for estimating a monotone increasing or monotone decreasing curve.
  - New options `smoother="mountain"` and `smoother="valley"` for estimating a unimodal function (U-shaped or inverted-U-shaped curve).
  - New argument `subset` allows computation for a subset of the data.
  - New argument `positiveCI` specifies whether confidence limits should always be positive.
  - If the covariate is a `distfun`, the name of the unit of length is saved and displayed on the plot.
  - New argument `rule.eps` passed to `as.mask`.

- `rhohat.lpp`:
  - New argument `random` controls placement of dummy points.
  - New argument `rule.eps` passed to `as.mask`.

- `rhohat.lppm`:
  - New argument `rule.eps` passed to `as.mask`.
- `rhohat.ppm`: New argument `rule.eps` passed to `as.mask`.
- `ripras`: argument `shape` is partially matched.
- `rjitter.ppp`: If `trim=TRUE`, the displacement radius will be constrained to be less than or equal to the distance from the data point to the window boundary. This guarantees that all displaced points fall inside the window, and accelerates the computation.
- `rlabel`:
  - New argument `group` specifies that the points are divided into several groups, and that relabelling is applied within each group.
  - New arguments `nsim` and `drop`.
  - `X` can now be a point pattern of any type (`ppp`, `lpp`, `pp3`, `ppx`) or a line segment pattern (`psp`).
- `rlabel.ppp`: New argument `group` specifies that the points are divided into several groups, and that relabelling is applied within each group.
- `rLGCP`:
  - New arguments `n.cond` and `w.cond` specify conditional simulation. Every realisation will have exactly `n.cond` points in total, or exactly `n.cond` points inside window `w.cond` if it is given.
  - This function has been completely re-implemented so that it no longer requires the package `RandomFields`, which is defunct (and sadly missed).
  - The current implementation supports only the `"exponential"`, `"gauss"`, `"stable"`, `"gencauchy"` and `"matern"` covariance functions.
  - Now recognises argument `rule.eps` passed to `as.mask`.
  - New arguments `n.cond` and `w.cond` specify conditional simulation. Every realisation will have exactly `n = n.cond` points, or exactly `n.cond` points inside window `w.cond` if it is given.
- `rMaternI, rMaternII`: These functions can now generate random patterns in three dimensions and higher dimensions, when the argument `win` is of class `box3` or `boxx`.
- `rMatClust`:
  - Can now perform conditional simulation given a fixed number of points.
  - New arguments `n.cond` and `w.cond`.
- `rmh`:
  - Accelerated, in the case where multiple patterns are saved using `nsave`.
  - The printed output of the debugger (invoked by `snoop=TRUE`) has been improved.
- `rmhmodel.ppm`, `simulate.ppm`: New argument `newdata` provides support for simulating a fitted model in a different window.
- `rmh.ppm, rmhmodel.ppm, simulate.ppm`: A model fitted using the `Penttinen` interaction can now be simulated.
- `rmh.default, rmhmodel.default`:
  - These functions now recognise `cif='penttinen'` for the Penttinen interaction.
  - New arguments `nsim`, `saveinfo`.
  - The printed output of the debugger (invoked by `snoop=TRUE`) has been improved.
- `rmhcontrol`:
  - New parameter `pstage` determines when to generate random proposal points.
  - The parameter `nsave` can now be a vector of integers.
- `rmpoint`: New argument `fail.action` specifies what to do if the required number of random points cannot be generated.

- `rNeymanScott`:
  - Argument `lmax` has been replaced by `kappamax`.
  - New argument 'mumax'.
- `roc.ppm`, `roc.kppm`, `roc.slrm`:
  - New argument `baseline` allows calculation of ROC relative to a baseline.
  - New argument `method` determines the estimation method. New options include a kernel smoothing estimate and a monotone estimate.
  - New arguments `CI`, `alpha` for calculating confidence intervals.
  - New argument `subset` specifies a sub-region of the spatial domain in which the ROC should be calculated.
  - New argument `leaveoneout` specifies whether to use leave-one-out estimates of intensity at the data points.
- `roc.ppp`:
  - New argument `baseline` allows calculation of ROC relative to a baseline.
  - New argument `weights` for numerical weights on the data points.
  - New argument `observations` specifies whether to use the exact point coordinates or pixel presence-absence indicators.
  - New argument `method` determines the estimation method. New options include a kernel smoothing estimate and a monotone estimate.
  - New arguments `CI`, `alpha` for calculating confidence intervals.
  - New argument `subset` specifies a sub-region of the spatial domain in which the ROC should be calculated.
- `roc.lpp`:
  - New argument `baseline` allows calculation of ROC relative to a baseline.
  - New argument `weights` for numerical weights on the data points.
  - New argument `method` determines the estimation method. New options include a kernel smoothing estimate and a monotone estimate.
  - New arguments `CI`, `alpha` for calculating confidence intervals.
  - New argument `subset` specifies a sub-region of the spatial domain in which the ROC should be calculated.
- `roc.lppm`:
  - New argument `baseline` allows calculation of ROC relative to a baseline.
  - New argument `method` determines the estimation method. New options include a kernel smoothing estimate and a monotone estimate.
  - New arguments `CI`, `alpha` for calculating confidence intervals.
  - New argument `subset` specifies a sub-region of the spatial domain in which the ROC should be calculated.
  - New argument `leaveoneout` specifies whether to use leave-one-out estimates of intensity at the data points.
- `roc.slrm`:
  - Changed interpretation of argument `leaveoneout=TRUE`. This now performs "leave-one-pixel-out" calculation (previously it performed "leave-one-point-out" calculation).
- `rose`
  - Rose diagrams can now be computed and plotted for data expressed in any units with any period (such as hours in a 24-hour day).
  - More options for argument `unit`.
  - New argument `fullcircle`.
  - New arguments `start` and `clockwise` specify the convention for measuring and plotting angles.
- `rose.default`
  - Polygons are now filled in light grey (by default)
  - New argument `col` specifies fill colour

  - New argument `do.ticks` controls whether ticks are drawn
  - Concentric rings are now drawn inside the circle, for scale.
  - New argument `do.rings`.
  - The return value now has attribute `R` giving the radius of the outer circle of the diagram, and attribute `rings` giving the radii of the concentric rings.
  - New argument `root` specifies whether to apply a square root transformation.
  - New argument `weights`.

- `rose.density`
  - New argument `do.ticks` controls whether ticks are drawn
  - Concentric rings are now drawn inside the circle, for scale.
  - New argument `do.rings`.
  - The return value now has attribute `R` giving the radius of the outer circle of the diagram, and attribute `rings` giving the radii of the concentric rings.
  - New argument `root` specifies whether to apply a square root transformation.

- `rose.fv`
  - New argument `do.ticks` controls whether ticks are drawn
  - Concentric rings are now drawn inside the circle, for scale.
  - New argument `do.rings`.
  - The return value now has attribute `R` giving the radius of the outer circle of the diagram, and attribute `rings` giving the radii of the concentric rings.
  - Now plots multiple curves if there are multiple columns of values.
  - New argument `fmla` controls which columns are plotted.
  - Now handles envelopes (objects of class `"envelope"`) as well.
  - New argument `root` specifies whether to apply a square root transformation.

- `rose.histogram`
  - Polygons are now filled in light grey (by default)
  - New argument `col` specifies fill colour
  - New argument `do.ticks` controls whether ticks are drawn
  - Concentric rings are now drawn inside the circle, for scale.
  - New argument `do.rings`.
  - The return value now has attribute `R` giving the radius of the outer circle of the diagram, and attribute `rings` giving the radii of the concentric rings.
  - New argument `root` specifies whether to apply a square root transformation.

- `rotate.infline`: New argument `centre` determines the centre of rotation.

- `rotmean`:
  - New argument `padzero`.
  - Default behaviour has changed.
  - Improved algorithm stability.
  - The result now has the same `unitname` as the input object.
  - New argument `adjust` controls the smoothing bandwidth.

- `rpoint`:
  - New argument `forcewin` forces the code to use the window `win` when `f` is a pixel image.
  - New argument `fail.action` specifies what to do if the required number of random points cannot be generated.

- `rpoispp`:

  - Improved algorithm in the case where the intensity `lambda` is a function of class `"tessfun"` representing a function which is constant on each tile of a tessellation.
  - Accelerated, when `lambda` is a pixel image.

- `rpoispp3`:
  - New argument `ex` (example point pattern, to be emulated)
  - Argument `lambda` may now be a `function(x,y,z)` so that an inhomogeneous Poisson process may be simulated.
  - New argument `lmax`.
- `rpoisppx`: New argument `drop`.
- `rpoisline`: Also returns information about the original infinite random lines.
- `rpoislpp`: If `lambda` is a list of `"linim"` or `"linfun"` objects, then the argument `L` can be omitted.
- `rPoissonCluster`: Argument `lmax` has been replaced by `kappamax`.
- `rshift.ppp`, `rshift.splitppp`: new argument `nsim`.
- `rSSI`:
  - Now supports three-dimensional random point patterns. Argument `win` can be a window (class `owin`) or a three-dimensional box (class `box3`).
  - Accelerated.
  - New argument `verbose` specifies whether to print progress reports when `nsim > 1`.
- `rStrauss, rHardcore, rStraussHard, rDiggleGratton, rDGS, rPenttinen:` New argument `drop`.
- `rtemper:` new argument `track`.
- `rthin`
  - Accelerated, when `P` is a single number.
  - `X` can now be a point pattern of any type (`ppp`, `lpp`, `pp3`, `ppx`) or a line segment pattern (`psp`).
- `rThomas, rMatClust, rCauchy, rVarGamma`:
  - New arguments `n.cond` and `w.cond` specify conditional simulation. Every realisation will have exactly `n.cond` points in total, or exactly `n.cond` points inside window `w.cond` if it is given.
  - These algorithms have been accelerated by several orders of magnitude in the case where the cluster radius is large.
  - These functions now offer a choice of simulation algorithms.
  - Formal arguments have changed.
  - When the model is approximately Poisson, it is simulated using rpoispp. This avoids computations which would require huge amounts of memory. New argument `poisthresh` controls this behaviour.
  - New argument `saveparents`.
  - New arguments `n.cond` and `w.cond` specify conditional simulation. Every realisation will have exactly `n = n.cond` points, or exactly `n.cond` points inside window `w.cond` if it is given.
- `runiflpp`, `rpoislpp`: The simulation parameters can be determined from an example point pattern, given as the argument `ex`.
- `runifpoint`: New argument `fail.action` specifies what to do if the required number of random points cannot be generated.
- `runifpointOnLines`, `rpoisppOnLines`: New argument `drop`.
- `runifpoint3`: New argument `ex` (example point pattern, to be emulated)
- `runifpointx`: New argument `drop`.
- `selfcut.psp`:
  - Computation accelerated.

  - The result now has an attribute `"camefrom"` indicating the provenance of each segment in the result.
  - No longer checks for validity of the resulting segments.

- `sessionInfo`: Output now includes a list of packages that are imported but not loaded.
- `sessionLibs`: Package names are now sorted alphabetically
- `setcov`: the name of the unit of length is preserved.
- `setminus.owin`: new argument `rescue`.
- `shapley`: In this installed dataset, the function `shapley.extra$plotit` has changed slightly (to accommodate the dependence on the package `spatstat.utils`).
- `shift.im`, `shift.owin`, `shift.ppp`, `shift.psp`: More options for the argument `origin`.
- Simulation: Several basic simulation algorithms have been accelerated. Consequently, simulation outcomes are not identical to those obtained with previous versions of `spatstat`, even when the same random seed is used. To ensure compatibility with previous versions of spatstat, revert to the slower code by setting `spatstat.options(fastthin=FALSE, fastpois=FALSE)`.
- `simulate.kppm`:
  - For log-Gaussian Cox process models (`clusters="LGCP"`) the simulation algorithm has been completely re-implemented without using the package `RandomFields`. The current code supports the `"exponential"`, `"gauss"`, `"stable"`, `"gencauchy"` and `"matern"` covariance models.
  - Conditional simulation of the model, given a fixed number of points, is now supported using the new arguments `n.cond` and `w.cond`.
  - Additional arguments `...` are now passed to the function that performs the simulation.
- `simulate.ppm`:
  - New argument `newdata` provides support for simulating a fitted model in a different window.
  - New argument `w` controls the window of the simulated patterns.
  - New argument `verbose`.
  - Now recognises the argument `window` as an alternative to `w`.
- `slrm`:
  - New argument `ruleAtPoints` controls discretisation rule at data points.
  - Arguments `"..."` are now passed to `owin2mask` to determine the polygon-to-raster discretisation rule for a covariate of class `owin`.
  - In the default case (where `dataAtPoints` is not given) all spatial covariates, including the spatial coordinates `x` and `y`, are now evaluated at the centre of each pixel. This improves consistency with other implementations of spatial logistic regression.
  - Silently ignores any arguments `'...'` that are not recognised by `as.mask`
- `Smooth.ppp`:
  - Now accepts argument `rule.pix` passed to `owin2mask`.
  - Shrinkage estimate implemented. New arguments `shrink` and `shrinktype`.
  - Slight change to algorithms for handling very small bandwidths.
  - A non-Gaussian kernel can now be specified using the argument `kernel`.
  - Argument `weights` can now be a pixel image, a function, a numeric vector or an expression to be evaluated.
  - Infinite bandwidth `sigma=Inf` is supported.
  - Accelerated by about 30% in the case where `at="pixels"`.
  - Accelerated by about 15% in the case where `at="points"` and `kernel="gaussian"`.
  - Now exits gracefully if any mark values are `NA`, `NaN` or `Inf`.
  - New argument `geometric` supports geometric-mean smoothing.
  - The arguments `adjust`, `edge`, `diggle` and `kernel` are now explicit formal arguments.
  - Standard error calculation is now supported (Experimental).

- solapply: If an entry of the list is an NAobject, the result for that entry will be an NAobject.
- solist: New argument .NameBase
- spatialcdf:
  - Computation accelerated.
  - The result does not inherit class "ecdf" if normalise=FALSE.
- spatstat.family: The list of extension packages now includes spatstat.convert.
- spatstat.options New options fastthin and fastpois enable fast simulation algorithms. Set these options to FALSE to reproduce results obtained with previous versions of spatstat.
- split.ppp, split.ppx: The splitting variable f can now be a logical vector.
- split<-.ppp: The default for argument un in split<-.ppp now agrees with the default for the same argument in split.ppp.
- square: Handles a common error in the format of the arguments.
- step: now works for models of class "mppm".
- stieltjes: Argument M can be a stepfun object (such as an empirical CDF).
- subset.ppp, subset.lpp, subset.pp3, subset.ppx: The argument subset can now be any argument acceptable to the "[" method.
- Summary.linim (methods for the operations range, max, min etc): Recognises the argument finite so that range(x, finite=TRUE) works for a linim object x.
- summary functions: The argument correction="all" is now recognised: it selects all the available options.

  This applies to Fest, F3est, Gest, Gcross, Gdot, Gmulti, G3est, Gfox, Hest, Jest, Jmulti, Jcross, Jdot, Jfox, Kest, Kinhom, Kmulti, Kcross, Kdot, Kmulti.inhom, Kcross.inhom, Kdot.inhom, Kscaled, Ksector, Kmark, K3est, Lscaled, markcorr, markcrosscorr, nnorient, pairorient, pcfinhom, pcfcross.inhom, pcfcross, pcf, Tstat.
- summary functions: The following functions have a new argument rmax:

  Fest, Finhom, FmultiInhom, Gcross, Gdot, Ginhom, Gmulti, GmultiInhom, Jest, Jcross, Jdot, Jinhom, Jmulti, Kcross, Kdot, Kinhom, Kmulti, Kcross.inhom, Kdot.inhom, Kmulti.inhom, Kmark, Ksector, markconnect, markcorr, markcorrint, markcrosscorr, markequal, markvario, pcf.ppp, pcfcross, pcfdot, pcfmulti, pcfinhom pcfcross.inhom, pcfcross.
- summary.distfun, summary.funxy:
  - More information is printed.
  - Pixel resolution can now be controlled.
- summary.im: Output improved when the image is empty (i.e. when all pixel values are undefined).
- summary.kppm: prints more information about algorithm convergence.
- summary.lintess: prints information about marks.
- summary.lppm: The name of the original point pattern dataset (to which the model was fitted) is now printed.
- summary.mppm: Improved summary of the dependence of the interpoint interaction on the covariates.
- summary.ppm: New argument fine selects the algorithm for variance estimation.
- summary.slrm: Now prints information about the pixel raster.
- summary.owin, summary.im: The fraction of frame area that is occupied by the window/image is now reported.
- sumouter:
  - New argument y allows computation of asymmetric outer products.
  - This function has now been moved to the sub-package spatstat.sparse

- `symbolmap`:
  - Now accepts a vector of colour values for the arguments `col`, `cols`, `fg`, `bg` if the argument `range` is given.
  - New option: `shape="arrows"`.
  - New arguments `transform`, `compress`, `decompress` for nonlinear symbol maps.
- `tess`:
  - A tessellation can now have any kind of marks (vector, list, data frame or hyperframe).
  - Argument `window` is ignored when xgrid, ygrid are given.
- `texturemap`: Argument `textures` can be missing or NULL.
- `textureplot`: Argument `x` can now be something acceptable to `as.im`.
- `thinNetwork`: `X` can be a pixel image on a network.
- `tileindex`: New arguments `close.gaps` and `all.inside`.
- `tilenames`, `tilenames<-`: These functions are now generic, with methods for `tess` and `lintess`.
- `to.grey` New argument `transparent`.
- `transect.im`:
  - This function is now a method for the generic `transect()`.
  - new argument `nsample`.
- `union.owin`:
  - New argument `rescue`.
  - Improved behaviour when there are more than 2 windows.
- `uniquemap`: handles larger objects.
- `unnormdensity`:
  - Argument `bw` may be `NULL` (treated as equivalent to being missing).
  - The result has an attribute `smooth.args` containing the smoothing parameters that were used in the calculation.
  - Suppress annoying warning messages from `density.default`. This affects many functions in the `spatstat` family of packages.
  - Argument `weights` may have length 1.
  - New argument `defaults`.
  - Computation accelerated.
- `unstack.lintess`: now handles marks.
- `update`: now works for models of class `"mppm"`.
- `update.kppm`:
  - New argument `evaluate`.
  - Now handles additional arguments in any order, with or without names.
  - Changed arguments.
  - Improved behaviour.
- `update.ppm`: For the case `update(model, X)` where `X` is a point pattern, if the window of `X` is different from the original window, then the model is re-fitted from scratch (i.e. `use.internal=FALSE`).
- `valid.ppm` This is now a method for the generic function `valid`.
- `varcount`:
  - Now accepts an object of class `"clustermodel"`.
  - New argument `relative` supports calculation of the overdispersion index.

- `vcov.mppm`:
  - Now handles models with Gibbs interactions.
  - New argument `nacoef.action` specifies what to do if some of the fitted coefficients are `NA`, `NaN` or `Inf`.
- `vcov.ppm`:
  - Performance slightly improved, for Gibbs models.
  - Variance calculations now handle larger datasets because they use sparse arrays, by default.
  - New argument `nacoef.action` specifies what to do if some of the fitted model coefficients are `NA`, `NaN` or infinite.
- `Vmark`: The result now has an attribute `smooth.args` containing the smoothing parameters used.
- `with.hyperframe`: If a column in the hyperframe contains `NAobject` entries, and if the name of that column is included in the expression being evaluated, then the result of evaluation is an `NA` or `NAobject` in the relevant row.
- `[<-.hyperframe`:
  - In an assignment like `x[i,j] <- NA`, if column `j` contains spatial objects, the `NA` value will be coerced to an `NAobject` of the appropriate class.
  - Improved error message when the format of the index is not supported.
- `[<-.im`
  - Accepts an array for `value`.
  - The subset index `i` can now be a linear network. Then the result of `x[i, drop=FALSE]` is a pixel image of class `linim`.
  - New argument `drop` controls behaviour when indices are missing as in `x[] <- value`
- `[.im` New argument `rule.pix` controls the handling of boundary pixels in computations like `x[W]` where `W` is a polygonal window.
- `[.layered`:
  - Subset index `i` can now be an `owin` object.
  - Additional arguments `...` are now passed to other methods.
- `[.leverage.ppm`: New argument `update`.
- `[.linnet`:
  - New argument `snip` determines what to do with segments of the network that cross the boundary of the window. Default behaviour has changed.
  - More robust against artefacts when the subset index is a pixel mask.
- `[.linim`:
  - More robust against artefacts.
  - Accelerated.
- `[.lpp`: New argument `snip` determines what to do with segments of the network that cross the boundary of the window. Default behaviour has changed.
- `[.ppx`:
  - The subset index `i` may now be a spatial domain of class `boxx` or `box3`.
  - New argument `clip`.
- `[.ppp`:
  - New argument `clip` determines whether the window is clipped.
  - The previously-unused argument `drop` now determines whether to remove unused levels of a factor.
- `[.pp3`, `[.lpp`, `[.ppx`, `subset.ppp`, `subset.pp3`, `subset.lpp`, `subset.ppx`: These methods now have an argument `drop` which determines whether to remove unused levels of a factor.
- `[.psp`:

  - accelerated.
  - New argument `fragments` specifies whether to keep fragments of line segments that are cut by the new window, or only to retain segments that lie entirely inside the window.

- `[.solist`: Subset index `i` can now be an `owin` object.

References

[1] A. Baddeley. Analysing spatial point patterns in R. Technical report, CSIRO, 2010. Version 4. URL `https://research.csiro.au/software/r-workshop-notes/`

[2] A. Baddeley, E. Rubak, and R. Turner. *Spatial Point Patterns: Methodology and Applications with R.* Chapman & Hall/CRC Press, 2015.