Density estimation in R

Henry Deng and Hadley Wickham

September 2011

Abstract

Density estimation is an important statistical tool, and within R there are over 20 packages that implement it: so many that it is often difficult to know which to use. This paper presents a brief outline of the theory underlying each package, as well as an overview of the code and comparison of speed and accuracy. We focus on univariate methods, but include pointers to other more specialised packages.

Overall, we found **ASH** and **KernSmooth** to be excellent: they are both fast, accurate, and well-maintained.

1 Motivation

There are over 20 packages that perform density estimation in R, varying in both theoretical approach and computational performance. Users and developers who require density estimation tools have different needs, and some methods of density estimation may be more appropriate than others. This paper aims to summarise the existing approaches to make it easier to pick the right package for the job.

We begin in Section 2 with a brief review of the underlying theory behind the main approaches for density estimation, providing links to the relevant literature. In Section 3, we describe the R packages that implement each approach, highlighting the basic code needed to run their density estimation function and listing differences in features (dimensionality, bounds, bandwidth selection, etc).

Section 4 compares the performance of each package with calculation speed, looking at density estimation computations from 10 to 10 million observations. The accuracy of the density estimations generated is also important. Section 5 compares the accuracy of the density estimates using three distributions with varying degrees of challenge: the uniform, normal and claw distributions. Section 6 investigates the relationship between calculation time and accuracy, and we conclude in Section 7 with our findings and recommendations.

2 Theoretical approaches

Density estimation builds an estimate of some underlying probability density function using an observed data sample. Density estimation can either be parametric, where the

data is from a known family, or nonparametric, which attempts to flexibly estimate an unknown distribution. We begin with a brief overview of the underlying theory, focusing on nonparametric methods because of their generality. Common methods include histograms, Section 2.1, kernel methods, Section 2.2, and penalized approaches, Section 2.3. We attempt to give the flavor of edge method, without going into too much detail. For a more in-depth treatment, we recommend Scott (1992a) and Silverman (1986).

We will assume that we have n iid data points, $X_1, X_2, ..., X_n$, and we are interested in an estimate, $\hat{f}(x)$, of the true density, $f(x)$, at new location x.

2.1 Histogram

The histogram (Silverman, 1986) is the oldest (dating to the 1840's (Friendly, 2005)) and least sophisticated method of density estimation. The main advantages are its extreme simplicity and speed of computation. A histogram is piecewise constant (hence not at all smooth) and can be extremely sensitive to the choice of bin origin.

A simple enhancement to the histogram is the average shifted histogram (ASH): it is smoother than the histogram and avoids sensitivity to the choice of origin, but is still computationally efficient. The premise of this approach (Scott, 1992b) is to take m histograms, $\hat{f}_1, \hat{f}_2, ..., \hat{f}_m$, of bin width h with origins of $t_o = 0, \frac{h}{m}, \frac{2h}{m}, ..., \frac{(m-1)h}{m}$. As the name suggests, the naïve ASH is simply

$$\widehat{f}_{ash}(x) = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_i(x)$$

There are $k = 1 \ldots m \cdot n$ bins across all histograms, each spanning $[k\frac{h}{m}, (k+1)\frac{h}{m}]$ with center $(k+0.5)\frac{h}{m}$. The ASH can be made somewhat more general by using all bins to estimate the density at each point, weighting bins closer to the data more highly. The general form of the weighted ASH is:

$$\widehat{f}_{ash}(x) = \frac{1}{m} \sum_{k=1}^{m \cdot n} w(l_k - x)\hat{c}_k(x)$$

where w is a weighting function, l_k is the center of bin k, and $\hat{c}_k$ is the number of points in that bin.

Because the univariate ASH is piecewise constant, it can be computed by taking a histogram with $m \cdot n$ bins and computing a rolling sum over m adjacent bins. This makes the ASH extremely fast to compute.

2.2 Kernel density estimation

The kernel density estimation approach overcomes the discreteness of the histogram approaches by centering a smooth kernel function at each data point then summing to get a density estimate. The basic kernel estimator can be expressed as

$$\widehat{f}_{kde}(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where K is the *kernel* and h is the *bandwidth* (Scott, 1992b). The kernel is a symmetric, usually positive function that integrates to one. Common kernel functions are uniform, triangle, Epanechnikov, quartic (biweight), tricube (triweight), Gaussian (normal), and cosine. The bandwidth, h, is a smoothing parameter: large bandwidths produce very smooth estimates, small values produce wiggly estimates. It influences estimates much more than the kernel, and so choosing a good bandwidth is critical to get a good estimate.

Typically the bandwidth is selected by minimising L^2 risk, aka the mean integrated square error:

$$\text{MISE}(\widehat{f}) = \text{E}\left[\int \left(\widehat{f}(x) - f(x)\right)^2 dx\right]$$

This optimisation problem is not easy because $f(x)$ is unknown, and leads to many theoretical approaches. General consensus is that plug-in selectors and cross validation selectors are the most useful over a wide range of inputs.

The main challenge to the kde approach is varying data density: regions of high data density could have small bandwidths, but regions with sparse data need large bandwidths. Extensions to the basic kde approach overcome this problem by allowing the bandwidth to vary (Terrell and Scott, 1992)

2.3 Penalized likelihood approaches

An alternative approach is to estimate the density as a mixture of m "basis" functions (densities), compromising between quality of fit and complexity. This has the advantage of allowing the density to be wigglier where the underlying data supports it. In general, the penalized approach (Kauermann and Schellhase, 2009) approximates the density of x as a mixture of m densities:

$$\widehat{f}_{pen}(x) = \sum_{i=1}^{m} c_i \, \phi_i(x)$$

where ϕ_k is a density and c_i are weights picked to ensure that $\widehat{f}_{pen}$ integrates to one. Typically the basis functions are equally weighed and differ only by a location parameter, μ_i, so we can simplify the definition to more closely resemble the kernel approach:

$$\widehat{f}_{pen}(x) = \frac{1}{m} \sum_{i=1}^{m} K\left(\frac{x - \mu_i}{h}\right)$$

Compared to the KDE, the bases/densities are no longer constrained to be centered on the data points.

The μ_i are often called knots, and the key problem of penalized density estimation is finding the most appropriate number and location of knots. Typically this is done by placing a large number of knots at equally spaced locations along the domain of the data, and then minimizing a penalized likelihood to remove knots that contribute little to the overall quality.

? TODO Should we include this section (including the part that's commented out?

The importance of λ lies in its role in controlling the amount of smoothness in the density estimation. A conventional method of selecting this penalty parameter is by the Akaike Information Criterion (AIC), in which we minimize:

$$\text{AIC}(\lambda) = -l(\widehat{\beta}) + df(\lambda)$$

where $df(\lambda) = tr(J_p^{-1}(\widehat{\beta}, \lambda)J_p(\widehat{\beta}, \lambda = 0))$. However, the penalized spline smoothing approach illustrated by Kauermann and Schellhase uses mixed models through a Bayesian viewpoint.

3 Density estimation packages

We now shift from our overview of broad theoretical approaches to specific R packages. Table 1 summarizes the 15 density estimation packages that we reviewed, giving basic information on what they do, their theoretical foundation, and the basic R code to use them. The remainder of this section describes each package in more detail. For each package, we summarize the input, output, and special features.

3.1 Histograms

3.1.1 graphics

`graphics::hist` allows users to generate a histogram (Venables and Ripley, 2002) of the data `x`. The `breaks` argument specifies the desired number of bins, or the borders of each bin (for irregular binning), or a function that estimates the number of bins automatically. By default, the function creates a plot, but this behavior can be suppressed with `plot = F`. The function returns a list giving bin boundaries, midpoints, counts and densities (standardized by bin width).

3.1.2 ash

The **ash** package (Scott, 2009) estimates ASHs using two functions: `bin1` and `ash1`. `bin1` takes the data vector `x`, the interval for the estimation, `a` to `b`, and the desired number of bins `nbin`, and counts the number of points in each bin. `ash1` takes the output from `bin1` and computes the generalized ASH, with weights defined by `kopt`. It produces an equally spaced mesh of predictions, centered on each bin.

ash also provides 2d average shifted histograms with `bin2` and `ash2` functions.

3.2 Kernel density estimation

CRAN packages **GenKern**, **kerdiest**, **KernSmooth**, **ks**, **np**, **plugdensity**, and **sm** all use the kernel density approach, as does `stats::density`. They differ primarily in their means of selecting bandwidth.

Package	Function cal	Max Dim.	Arbitrary Grid	Predicted Density	Approach
ASH	`ash1(bin1(x, ab = c(min(x), max(x)), nbin = 512))`	2	No	`d$y`	ASH
ftnonpar	`pmden(x)`	1	No	`d$y`	Taut Strings
GenKern	`KernSec(x, 512, range.x = c(min(x), max(x)))`	2	No	`d$yden/100`	Kernel
gss	`dssden(ssden(~x), seq(min(x), max(x), length = 512))`	2	Yes	`d`	Penalized
kerdiest	`kerdiest::kde(vec_data = x, y = xgrid)`	1	Yes	`d$Estimated_values`	Kernel
KernSmooth	`bkde(x = x, gridsize = 512L, range.x(min(x), max(x)))`	2d	No	`d$y`	Kernel
ks	`kde(x = x, hpi(x), eval.points = xgrid)`	6	Yes	`d$estimate`	Kernel
locfit	`density.lf(x, ev = xgrid)`	1	Yes	`d$y`	Local Likelihood
logspline	`dlogspline(xgrid, logspline(x))`	1	Yes	`d`	Penalized
MASS	`hist(x, 512)`	1	Yes	`d$density`	Histogram
np	`npudens(~ x, edat = xgrid)`	1	Yes	`d$dens`	Kernel
pendensity	`pendensity(x ~ 1)`	1	No	`d$results$fitted`	Penalized
plugdensity	`plugin.density(x, xout = xgrid)`	1	Yes	`d$y`	Kernel
stat	`density(x, n = 512)`	1	No	`d$y`	Kernel
sm	`sm.density(x, display = "none", eval.points = xgrid)`	3	Yes	`d$estimate`	Kernel

Table 1: Packages we investigated. We assume that the estimate output is `d`, the input data is `x`, and the desired evaluation grid is `xgrid`, which sequences x into 512 evaluation points.

3.2.1 GenKern

The function `KernSec` in **GenKern** (Lucy and Aykroyd, 2010) performs univariate density estimation with Gaussian kernels on input `x`. It supports multiple bandwidths, i.e.:

$$\widehat{f}_{GK}(x) \propto \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_i}\right)$$

defined by `xbandwidth`. The default bandwidth is a single number selected using the plugin rule defined by Sheather and Jones (1991). Output is a grid of `xgridsize` predictions over `range.x`.

The function `KernSur` works similarly for 2 dimensions.

3.2.2 kerdiest

The `kde` function in **kerdiest** estimates a kernel density on input `vec_data`, with four types of kernels, `type_kernel`. It provides three different automatic bandwidth selection methods. The default bandwidth is the plug-in method by Polansky and Baker (2000), `PBbw`; users can also choose the plug-in method by Altman and Léger (1995), `ALbW`, and the cross-validation method by Bowman and Azzalini (1997), `CVbw`. Output defaults to a grid of 100 evenly spaced points over the range of the samples, but this can be overridden by supplying a vector of locations, `y`.

3.2.3 KernSmooth

`bkde` (Wand and Ripley, 2010) implements binned kernel density estimates using linear binning (Wand, 1994). Linear binning is a slight tweak to regular binning where if the grid points x and z surround the data point y, then $(z-y)/(z-x)$ mass is distributed to the grid point at x, and $(y-x)/(z-x)$ to the point at z. Density estimation proceeds in a straightforward manner, using weighted bin counts rather than the individual data points. Five kernels (normal, uniform, Epanechnikov, biweight and triweight), can be selected with `kernel`. The default bandwidth is the "oversmoothed bandwidth selector" (Wand and Jones, 1995, pg. 61). Output is the density at `gridsize` evenly spaced points over `range.x`.

`bkde2D` implements a binned 2d density estimate, and `bkfe` provides kernel functionals: the integral of the product of the density estimate and its derivative of order `drv`. `locpoly` implements local polynomial regression for a non-parametric estimate of $\mathrm{E}(Y|X)$.

3.2.4 ks

The **ks** package (Duong, 2011) can perform density estimation computation for up to 6 dimensions. **ks** provides a range of bandwidth selection routines: exact MISE and asymptotic MISE estimates `Hamise.mixt` and `Hmise.mixt`, biased cross-validated

`Hbcv`, least-squares cross-validation `Hlscv`, plug-in estimator `Hpi`, and smoothed cross-validation `Hscv`. All bandwidth estimators also have a variant that estimates only the diagonal (i.e. it assumes the dimensions are uncorrelated).

`kde` estimates densities of 1-6d data `x`, using bandwidth `H` (or `h` if diagonal). Estimates use binned input with bin size `bgridsize` if `binned` is `TRUE`, and can use weights specified by `w`. Output is predicted densities over a grid of `gridsize` points from `xmin` to `xmax`, or alternatively at locations `eval.points`.

ks also provides pdf `dkde`, cdf `pkde`, inverse cdf `qkde`, and random number generation `prkde` functions for the densities it generates.

3.2.5 np

The **np** package (Hayfield and Racine, 2008) uses a three-step approach: computing the bandwidth, `npudensbw`; estimating the density, `npudens`; then extracting predicted densities values and standard errors, `predict`. **np** focuses on "generalized product kernels" (Racine and Li, 2004), and data-driven bandwidth selection methods that the authors warn can be computationally demanding. For continuous data, the three classes of kernel estimators in the function are fixed, adaptive nearest-neighbor, and generalized nearest neighbor.

np also supports semi-paramteric and partially linear models, kernel quantile regression and smooth coefficient kernel regression. It works with any type of data: numeric, factors and ordered factors.

3.2.6 plugdensity

This package's sole function, `plugin.density`, uses the iterative plug-in approach to select bandwidth in kernel density estimation Gasser *et al.* (1991). Output is a grid of estimated densities at `nout` evenly spaced points, or, if specified, locations `xout`.

3.2.7 sm

The **sm** package (Bowman and Azzalini, 2010, 1997) can perform kernel density estimation from 1d to 3d. The function `sm.density` takes input `x`, bandwidths `h`, and frequency weights `h.weights`. If no bandwidth is specified, `h.select` uses a normal optimal smoothing estimate. Output is predicted density, with standard error, at `eval.points` locations. The default graphical output can be suppressed with `display = "none"`.

sm also provides tools for non-parametric ANCOVA, analysis of autoregression, regression (including logistic, Poisson and autocorrelated), and for density estimation on a sphere.

3.2.8 stats

stats::density performs univariate density estimate with normal (Gaussian), rectangular, triangular, and cosine `kernel`s. Bandwidth is specified by parameter `bw`, which defaults to Silverman's rule of thumb, `bw.nrd0`. Other bandwidth estimators are `bw.nrd`,

Scott's variant, `bw.bcv` and `bw.ucv`, biased and unbiased cross-validation, and `bw.SJ`, the Sheather and Jones estimate. Weights can be supplied with `weights`. Output is `n` evenly spaced points from `from` to `to`. By default `from` and `to` are set `cut` bandwidths away from the range of the data.

3.3 Penalized approaches

These packages provided density estimation based on penalised approaches and tend to have far fewer tuning parameters. The underlying methodology is considerably more complex than for kernel density estimates, and we recommend users familiarise themselves with the underlying literature before using these packages.

3.3.1 gss

gss (Gu, 2011) uses a penalized likelihood technique for nonparametric density estimation. `ssden` has a `formula` interface, working on vectors in the global environment, or variables in a data frame supplied with the `data` argument. The function returns an `ssden` object, and predictions can be obtained at arbitrary locations with `dssden`. By default, the number of knots and their locations are picked using cross-validation, but they can be supplied with the `nbasis` and `id.basis` arguments.

gss also provides cdf `pssden` and inverse cdf `qssden` functions for the densities it generates; in addition to 1d density estimation, it also performs conditional density estimation, smoothing spline ANOVA, hazard models, and log-linear models.

3.3.2 locfit

The **locfit** package (Loader, 2010) utilizes a local likelihood approach to density estimation. The `locfit` function has a flexible formula interface, with `~ lp(x)` performing 1d density estimation. The `lp` function controls the local polynomial model: `nn` specifies the nearest-neighbour bandwidth, `h` the constant bandwidth, and `deg` the degree of the polynomial. Weights can be supplied with the `weights` argument, and bounds can be supplied with `xlim`. The function returns an object of class `locfit`, and predictions at arbitrary locations can be produced with the `predict` method.

locfit also performs local regression in robust and censored variants, and can provide standard errors for predictions.

3.3.3 pendensity

The eponymous `pendensity` function (Schellhase, 2010) estimates densities with penalized splines. It has a formula interface where `x ~ 1` specifies a 1d density and `base` is used to select the basis function (either bspline or gaussian). The function returns an object of class `pendensity`, and predicted densities at the locations of the original data points are stored in `$results$fitted`.

3.3.4 logspline

logspline (Kooperberg, 2009) uses a maximum likelihood approach. The `logspline` function takes numeric data in argument `x`, and if known, bounds can be supplied with `lbound` and `ubound`. The function returns an object of class `logspline`, and predicted densities at arbitrary locations can be extracted with `dlogspline`.

logspline also provides cdf `plogspline`, inverse cdf `qlogspline` and random number generation, `rlogspline`, functions for the densities it generates.

3.4 Taut strings approach

3.4.1 ftnonpar

For one-dimensional data `x`, the `pmden` function in **ftnonpar** returns density estimates through piecewise monotone density estimation. The function returns a list with element `y` giving density estimates at the locations of the original data points.

ftnonpar also provides piecewise monotone logistic regression, regression, and spectral density approximation.

3.5 Other packages

Other packages compute more specialized density estimates. We list them below, with a brief description of each package's use case and pointers to the relevant literature.

- Package **bayesm** (Rossi., 2011): a Bayesian approach to density estimation.
- Package **delt** (Klemela, 2009): uses adaptive histograms.
- Package **feature** (Duong and Wand, 2011): identifies significant features of kernel estimates of 1- to 4- dimensional data.
- Package **fdrtool** (Strimmer., 2011): monotone density estimates.
- Package **hdrcde** (Hyndman, 2010): conditional kernel density estimate.
- Package **ICE** (Braun): kernel estimates for interval-censored data.
- Package **latticeDensity** (Barry, 2011): lattice based density estimator for 2d regions with irregular boundaries and holes.
- Package **logcondens** (D'umbgen and Rufibach, 2011): fitting of log-concave densities
- Package **MKLE** (Jaki, 2009): maximum kernel likelihood estimator for a given dataset and bandwidth.
- Package **pencopula** (Schellhase, 2011): penalized hierarchical B-splines estimation of copula densities.
- `MASS::kde2d`, bivariate kernel density estimates

4 Density estimation computation speed

We compared the computation time for the default method in each package using normally distributed data from 10 to 10,000,000 samples. To measure calculation time, we used the **microbenchmark** package (Mersmann, 2011). Figure 1 shows the results. Note that both axes are log scaled.

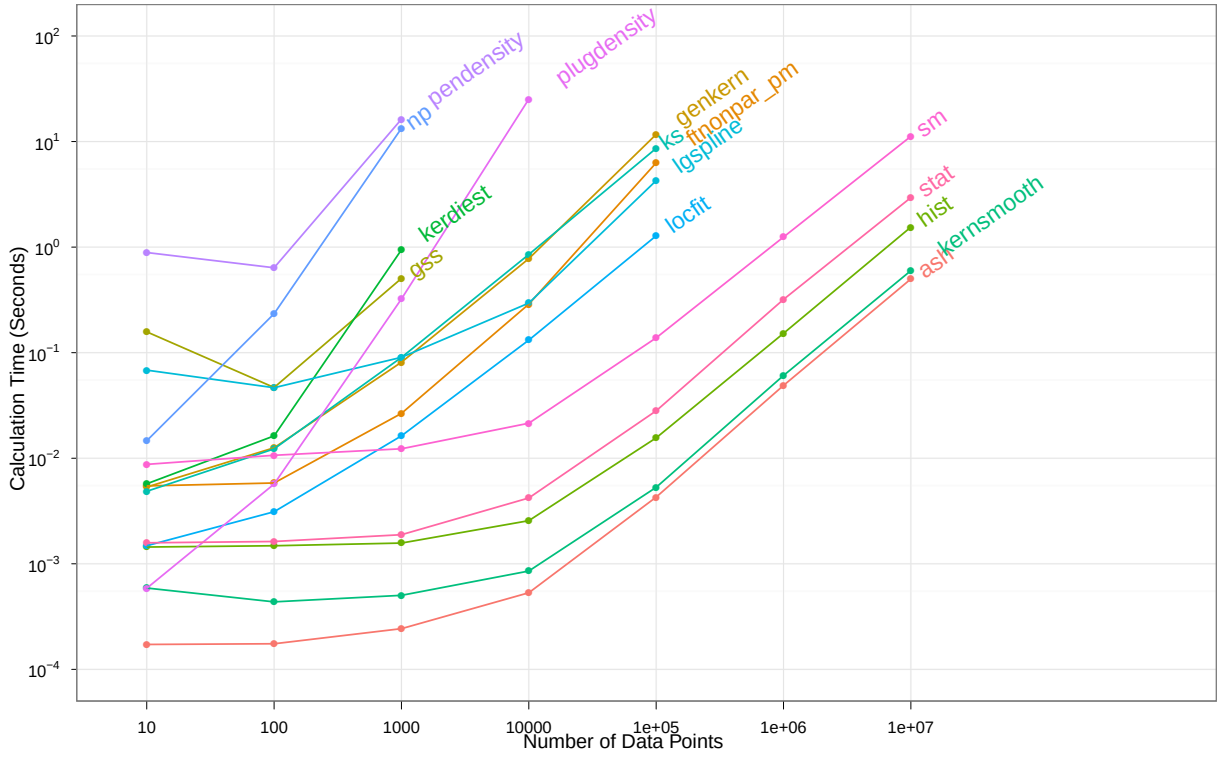

Figure 1: Calculation time required for density estimation as the number of data points in the calculation increased. Graph is shown on a log scale.

The **ASH** and **KernSmooth** packages are exceptionally fast, computing density estimates for 10 million points in under a second. **ASH** is fast because it uses a simple binning algorithm implemented in Fortran, and **KernSmooth** is fast because it uses a binned approximation for the density estimate. Core functions `hist` and `density` (labeled stat) are close runners up, taking under 10 seconds for 10 million points. `hist` is fast because it is a simple algorithm and is implemented in C, while `density` is fast because it takes advantage of a fast Fourier transform.

On the other end, the **np** package implements data-driven bandwidth calculation, and as we can see from the results, computation time may be extremely slow. Data sets with more than 10^3 points are computationally infeasible. The developers do warn that this package's approach may be computationally demanding and that the cross-validation run times are of order n^2. Similarly, it appears that **pendensity**'s penalized splines method requires significant computational time, more so than **gss**'s penalized

likelihood technique. Note that these packages do considerably more than univariate density estimation: generalisation comes with a performance cost for simple cases.

5 Accuracy of density estimates

Speed of computation in a package is insufficient in assessing performance, as the quality of the results must be considered as well. To do so, we use the uniform, normal, and claw distributions for varying degrees of challenge. For each distribution, we take increasing sizes of data points (n), estimate the density using the default parameters for each package and compare results to the true underlying distribution by calculating the mean absolute error. Results are shown in Figure 2. These are admittedly crude challenges, but should serve to give a general sense of the accuracy of each package.

Due to memory and computational constraints, some trials could only test certain numbers of data points before calculations became unmanageable. This is reflected in the plots, where not all packages test up to the same n.

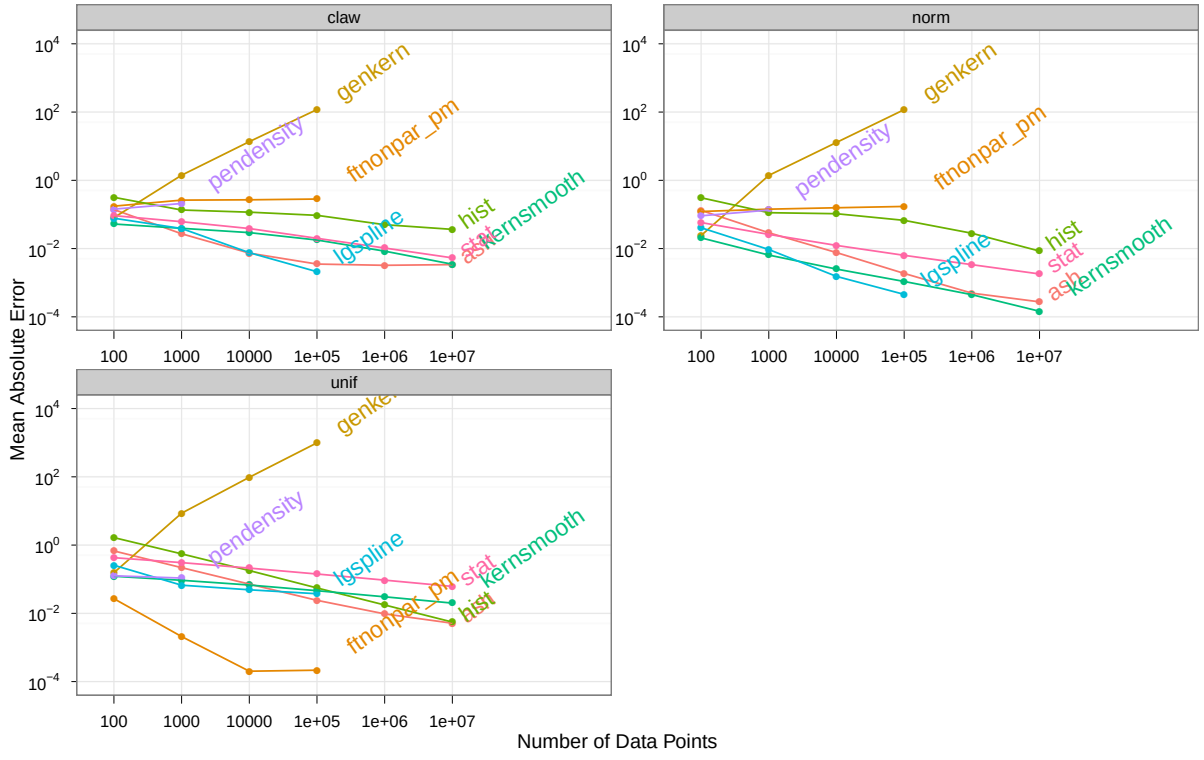

Figure 2: Mean absolute error using each package with different true underlying distributions. To reduce cluttering, only select packages are displayed.

We expect that as the number of data points in the estimate increases, the amount of error should decrease, in accordance with the law of large numbers. In the plot from our trials, all the packages exhibit this behavior except for **GenKern**, which

disturbingly has greater error with more data points. From the uniform distribution density estimates, it appears that **ftnonpar** and **kerdiest** yield lower levels of error than the other packages. For **ftnonpar**, the results partially support the claim of Kovac (2007) that the taut strings approach should allow good estimates for uniform and claw distributions. Overall, the packages that yielded the most accurate estimates were **logspline**, **ASH**, **KernSmooth**, and the `density` function.

6 Speed vs. accuracy

We initially expected that calculation time and level of error would be correlated. The computation time results showed that faster packages included those taking the histogram approach or binning in kernel density estimation, and we expect these to have a higher degree of error. Meanwhile, more time spent on computation should hopefully lead to better results.

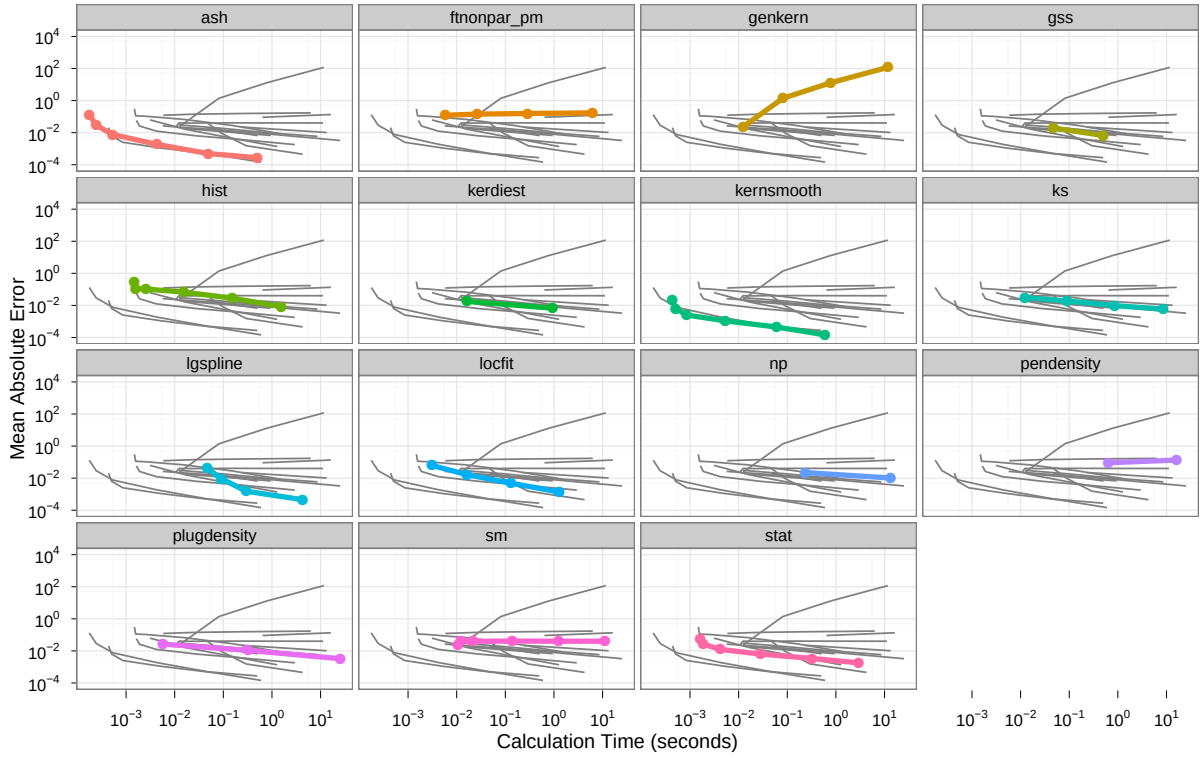

Figure 3: Estimation speed (x-axis) and mean absolute error (y-axis) for each package when fit to the normal distribution. All paths are shown in the background of each panel to support easier comparison. The best methods lie in the lower left quadrant: they are both fast and accurate.

Figure 3 shows the tradeoff between speed and accuracy. In this plot, the most desirable position is in the lower left corner, where both computation time and level of error are low. We do not see what we expect: there is no clear negative correlation

between speed and accuracy. In fact **ASH** and **KernSmooth** packages are both fast and accurate compared to the other packages.

We wondered if a better causal explanation might be that some packages are better written, hence are both faster and more accurate. A useful proxy for package quality might be the number of times the package has been updated. More frequent updating could suggest that the maintainer is making constant improvements, resulting in higher package quality; however, this trend might also imply that those packages are the ones in need of modifications while others already operate effectively. By inspecting the CRAN archive we determined the release dates of all versions of the packages we investigated. Figure 4 summarises this graphically, and Table 2 highlights the most frequently updated packages.

Both **KernSmooth** and **ASH** have a long history of regular updates.

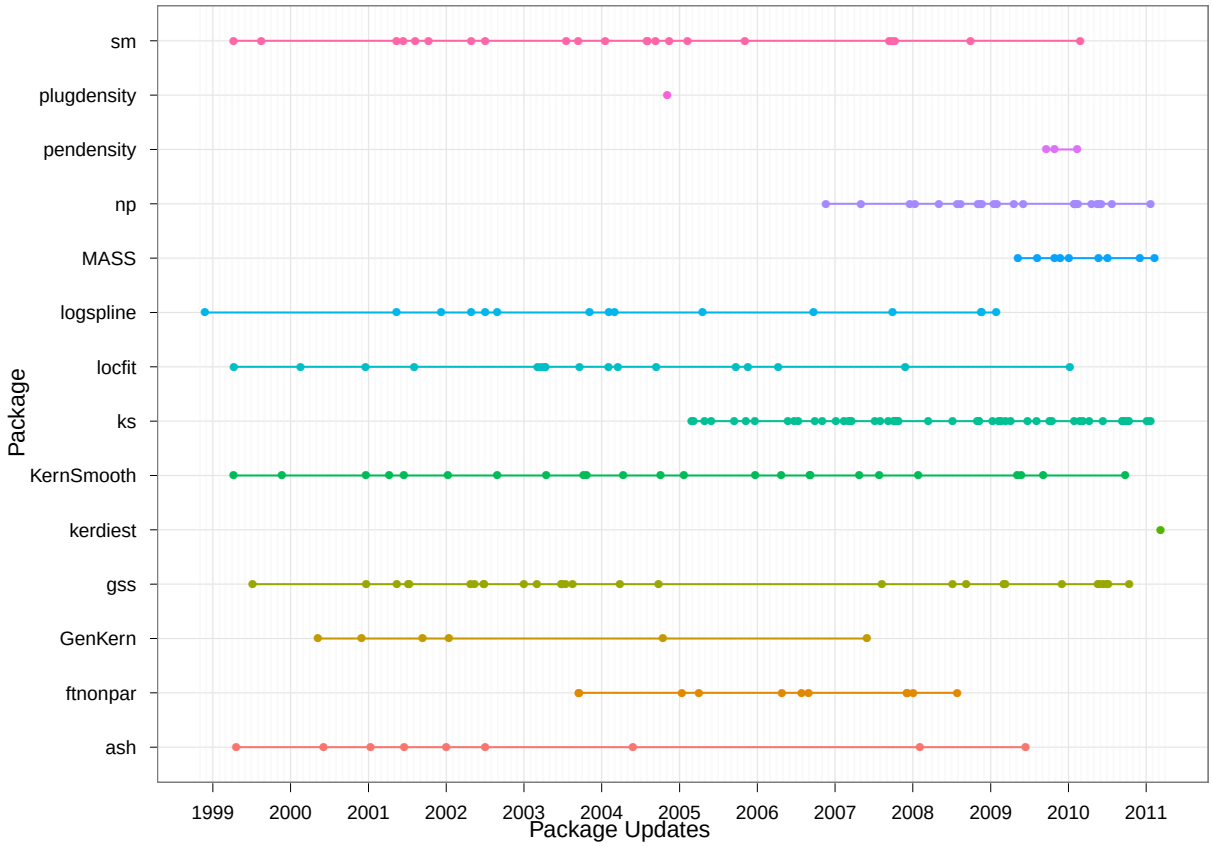

Figure 4: The update dates for each package examined in this paper.

7 Conclusion

Our findings suggest that the best packages did exhibit a tradeoff between speed and accuracy but were fast, low in error, and regularly updated. We recommend begin-

Package	Current Version	Number of Updates
ks	1.8.1	52
gss	1.1-7	30
np	0.40-4	24
KernSmooth	2.23-4	24
sm	2.2-4.1	22
locfit	1.5-6	17
logspline	2.1.3	15
ftnonpar	0.1-84	11
ASH	1.0-12	9
MASS	7.3-13	9
GenKern	1.1-10	6
pendensity	0.2.3	3
kerdiest	1	1
plugdensity	0.8-2	1

Table 2: This table shows version numbers and updates for packages used in both computation time and calculation error testing.

ning with the **ASH** or **KernSmooth** packages. Both ranked high for performance, and the package updates information also showed that they are two of the oldest density estimation packages, with regular updates.

It is important to note that our recommendations were based on the series of tests we were able to perform and that it may be more apropos to utilize other packages depending on user needs. For example, our density estimation accuracy benchmark distributions were uniform, normal, and claw, and we observed that the package **ftnonpar** performed better than the other packages for uniform but not for normal or claw. In short, certain theoretical approaches may result in better performance for different underlying distributions or data sets.

We hope that our review of density estimation packages in R will make it easier for others to match problems to solutions in the future.

References

Altman N, Léger C (1995). "Bandwidth selection for kernel distribution function estimation." *Journal of Statistical Planning and Inference*, **46**(2), 195–214.

Barry R (2011). *latticeDensity: Density estimation and nonparametric regression on irregular regions*. R package version 1.0.6, URL `http://CRAN.R-project.org/package=latticeDensity`.

Bowman A, Azzalini A (1997). *Applied smoothing techniques for data analysis: The kernel approach with S-Plus illustrations*. Oxford University Press, USA.

Bowman AW, Azzalini A (2010). *R package sm: nonparametric smoothing methods (version 2.2-4)*. University of Glasgow, UK and Università di Padova, Italia. URL URLhttp://www.stats.gla.ac.uk/~adrian/sm,http://azzalini.stat.unipd.it/Book_sm.

Braun WJ (????). *ICE: Iterated Conditional Expectation*. R package version 0.61.

D'umbgen L, Rufibach K (2011). "logcondens: Computations Related to Univariate Log-Concave Density Estimation." *Journal of Statistical Software*, **39**(6), 1–28. URL http://www.jstatsoft.org/v39/i06/.

Duong T (2011). *ks: Kernel smoothing*. R package version 1.8.1, URL http://CRAN.R-project.org/package=ks.

Duong T, Wand M (2011). *feature: Feature significance for multivariate kernel density estimation*. R package version 1.2.7, URL http://CRAN.R-project.org/package=feature.

Friendly M (2005). "Milestones in the history of data visualization: A case study in statistical historiography." *Classification—the Ubiquitous Challenge*, pp. 34–52.

Gasser T, Kneip A, Kohler W (1991). "A flexible and fast method for automatic smoothing." *Journal of the American Statistical Association*, **86**(415), 643–652.

Gu C (2011). *gss: General Smoothing Splines*. R package version 1.1-7, URL http://CRAN.R-project.org/package=gss.

Hayfield T, Racine JS (2008). "Nonparametric Econometrics: The np Package." *Journal of Statistical Software*, **27**(5). URL http://www.jstatsoft.org/v27/i05/.

Hyndman RJ (2010). *hdrcde: Highest density regions and conditional density estimation*. R package version 2.15, URL http://CRAN.R-project.org/package=hdrcde.

Jaki T (2009). *MKLE: Maximum kernel likelihood estimation*. R package version 0.05.

Kauermann G, Schellhase C (2009). "Density Estimation with a Penalized Mixture Approach."

Klemela J (2009). *delt: Estimation of multivariate densities with adaptive histograms*. R package version 0.8.0, URL http://CRAN.R-project.org/package=delt.

Kooperberg C (2009). *logspline: Logspline density estimation routines*. R package version 2.1.3, URL http://CRAN.R-project.org/package=logspline.

Kovac A (2007). "Smooth functions and local extreme values." *Computational Statistics & Data Analysis*, **51**(10), 5155–5171.

Loader C (2010). *locfit: Local Regression, Likelihood and Density Estimation*. R package version 1.5-6, URL http://CRAN.R-project.org/package=locfit.

Lucy D, Aykroyd R (2010). *GenKern: Functions for generating and manipulating binned kernel density estimates*. R package version 1.1-10, URL `http://CRAN.R-project.org/package=GenKern`.

Mersmann O (2011). *microbenchmark: Sub microsecond accurate timing functions*. R package version 1.1-3, URL `http://CRAN.R-project.org/package=microbenchmark`.

Polansky A, Baker E (2000). "Multistage plug-in bandwidth selection for kernel distribution function estimates." *Journal of statistical computation and simulation*, **65**(1), 63–80.

Racine J, Li Q (2004). "Nonparametric estimation of regression functions with both categorical and continuous data." *Journal of Econometrics*, **119**(1), 99–130.

Rossi P (2011). *bayesm: Bayesian Inference for Marketing/Micro-econometrics*. R package version 2.2-4, URL `http://CRAN.R-project.org/package=bayesm`.

Schellhase C (2010). *pendensity: Density Estimation with a Penalized Mixture Approach*. R package version 0.2.3, URL `http://CRAN.R-project.org/package=pendensity`.

Schellhase C (2011). *pencopula: Flexible Copula Density Estimation with Penalized Hierarchical B-Splines*. R package version 0.2.2, URL `http://CRAN.R-project.org/package=pencopula`.

Scott D (1992a). *Multivariate density estimation*. Wiley Online Library.

Scott D (1992b). *Multivariate density estimation*, volume 139. Wiley Online Library.

Scott D (2009). *ash: David Scott's ASH routines*. R package version 1.0-12, URL `http://CRAN.R-project.org/package=ash`.

Sheather SJ, Jones MC (1991). "A reliable data-based bandwidth selection method for kernel density estimation." *Journal of the Royal Statistical Society, Series B*, **53**, 683–690.

Silverman B (1986). "Density estimation for statistics and data analysis. 1986." *Monographs on Statistics and Applied Probability. Chapman and Hall, New York*.

Strimmer K (2011). *fdrtool: Estimation and Control of (Local) False Discovery Rates*. R package version 1.2.7, URL `http://CRAN.R-project.org/package=fdrtool`.

Terrell G, Scott D (1992). "Variable kernel density estimation." *The Annals of Statistics*, **20**(3), 1236–1265.

Venables W, Ripley B (2002). *Modern applied statistics with S*. Springer verlag.

Wand M (1994). "Fast computation of multivariate kernel estimators." *Journal of Computational and Graphical Statistics*, **3**(4), 433–445.

Wand M, Jones M (1995). *Kernel smoothing*, volume 60. Chapman & Hall/CRC.

Wand M, Ripley B (2010). *KernSmooth: Functions for kernel smoothing for Wand & Jones (1995)*. R package version 2.23-4, URL `http://CRAN.R-project.org/package=KernSmooth`.