probhat 0.1.1

Generalized Kernel Smoothing

Abby Spurdle

February 28, 2019

Computes nonparametric probability distributions (probability density functions, cumulative distribution functions and quantile functions) using kernel smoothing. Supports univariate, multivariate and conditional distributions, and weighted data (possibly useful mixed with fuzzy clustering or frequency data). Also, supports empirical continuous cumulative distribution functions and their inverses, and random number generation.

Pre-Intro

This package is based on function objects.

Some functions (constructors) return other functions (models), which can be evaluated.

This is equivalent but different to the d, p, q, r approach used in R's stats package.

Introduction

This package computes (constructs and evaluates) nonparametric probability distributions using kernel smoothing.

Currently, this includes:

- Nonparametric probability density functions (NPPDFs).
- Nonparametric cumulative distribution functions (NPCDFs).
- Nonparametric quantile functions (NPQFs or NPCDF inverses).

It supports univariate, multivariate and conditional distributions.

By default, univariate and conditional models, use cubic hermite splines as intermediate models. The spline approach is required if computing statistics (from probability distributions rather than from data), and is more efficient if plotting the models or evaluating many points.

NPPDFs can be used to visualize the shape of probability distributions, and compute their means and modes. NPCDFs can be used to compute probabilities, including multivariate probabilities. And NPQFs can be used to compute medians and quantiles, and for inversion sampling.

Models may be weighted, possibly useful mixed with fuzzy clustering or frequency data.

This package also supports:

- Empirical continuous (not step) cumulative distribution functions (continuous ECDFs) and their inverses (continuous EQFs or continuous ECDF inverses).
- Regression-like models.
- Random number (or random sample) generation.

However, ECDFs, EQFs and regression-like models are only discussed in appendices.

The philosophy behind nonparametric probability distributions, is to model data directly, with as few assumptions as possible.

Loading the Packages

I'm going to load (and attach) the probhat, fclust, moments and scatterplot3d packages.

```
> library (probhat)
> library (fclust)
> library (moments)
> library (scatterplot3d)
```

Preparing the Data

I'm going to use the trees data (from the datasets package) and the unemployment data (from the fclust package).

```
> data (trees)
> data (unemployment)
```

```
> #refer to datasets for a description of these variables
> #(and converting to metric)
> # -> m
> Height = 0.3048 * trees$Height
> # -> cm
> Girth = 2.54 * trees$Girth
> # -> m^3
> Volume = 0.0283168 * trees$Volume
```

```
> #refer to fclust for a description of these variables
> #note that this data set contains three variables
> #(however, I'm only using two variables)
> un.rate = unemployment$Total.Rate
> lt.rate = unemployment$LongTerm.Share
```

New matrix objects.

```
> trees.2.2 = cbind (Height, Volume)
> trees.2.3 = cbind (Height, Girth, Volume)
> unemployment.2 = cbind (un.rate, lt.rate)
```

I've provided some more information on the trees.2 and unemployment.2 data in Appendix 5 and Appendix 6.

Univariate Models (and Core Functionality)

Theory

We can evaluate univariate models using the following expressions.

$$f(x) = \frac{\sum_i g(x_i^*, \text{bw}, x)}{n}$$

$$F(x) = \frac{\sum_i G(x_i^*, \text{bw}, x)}{n}$$

Where:

$$g(L, \text{bw}, x) = \frac{2}{\text{bw}} k(\frac{2}{\text{bw}}(x - L))$$

$$G(L, \text{bw}, x) = K(\frac{2}{\text{bw}}(x - L))$$

And where $k()$ is the kernel PDF, $K()$ is the kernel CDF, bw is the bandwidth, n is the number of data points and x^* is a vector of data points.

By default, this package uses what I refer to as simplified bell curves as kernels, and a smoothness parameter of 0.65.

Note that more information about the kernel and smoothing parameters is Appendix 1 and Appendix 2.

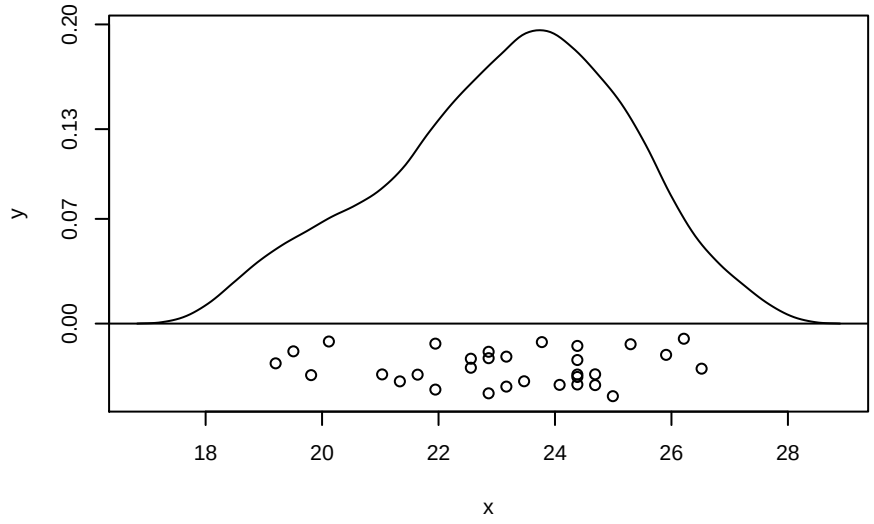
Probability Density Functions

We can use the nppdfuv() function to construct univariate NPPDFs.

```
> f = nppdfuv (Height)
```

We can plot the density with the data points.

```
> plot (f, TRUE)
```



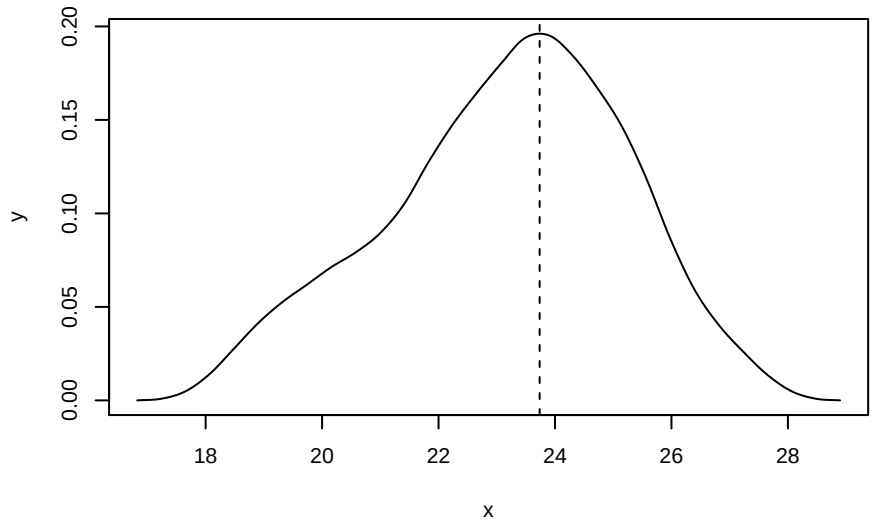
We can evaluate the object (which is a function), however, this isn't that useful.

```
> f (20)

[1] 0.06783298
```

Note that the argument can be a vector.

Also, we can compute the mean and mode and using the npmean() and npmode() functions.

```
> npmean (f)

[1] 23.16688

> mode.Height = npmode (f)
> mode.Height

[1] 23.73624

> plot (f)
> abline (v=mode.Height, lty=2)
```

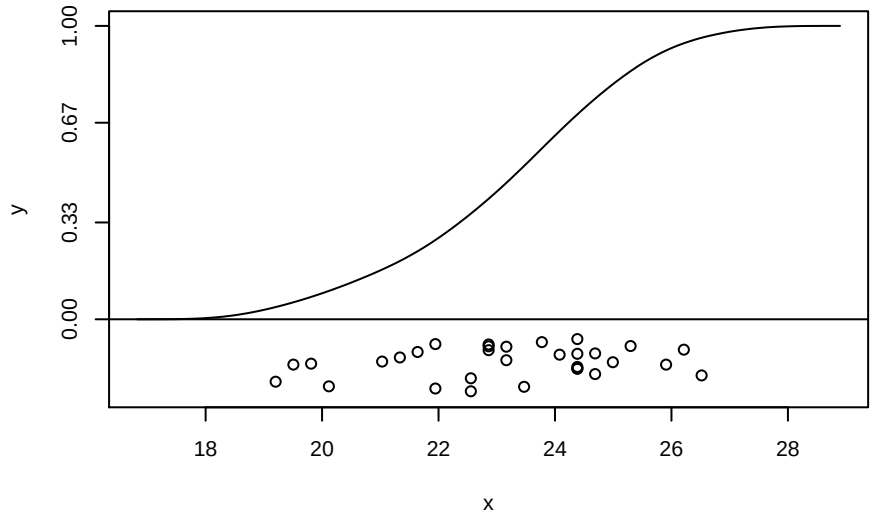


Computing the mean this way is inefficient, however, it can be used on conditional distributions too, discussed later. Note that the function used to compute modes has had limited testing and there may be unknown problems, especially if trying to compute multiple modes from multimodal distributions.

Cumulative Distribution Functions

We can use the npcdfuv() function to construct univariate NPCDFs.

```
> F = npcdfuv (Height)
> plot (F, TRUE)
```



Like nppdfuv objects we can evaluate an npcdfuv object.

Let's say that we want to compute the probability that height is between 20 and 22 meters.

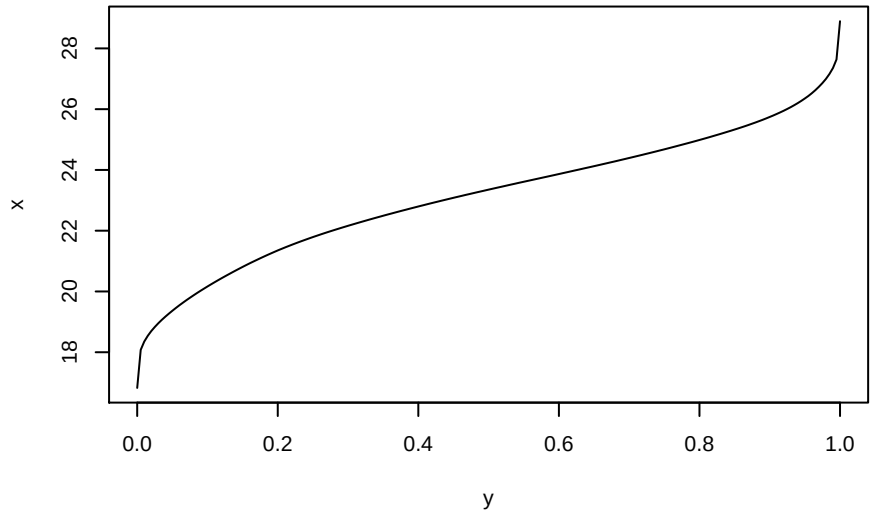
```
> F (22) - F (20)
```

```
[1] 0.1888409
```

Quantile Functions

We can compute an nonparametric quantile function by inverting the NPCDF.

```
> F.inv = npcdfuv.inverse (Height)
> plot (F.inv)
```



Like nppdfuv and npcdfuv objects we can evaluate an npcdfuv.inverse object.

Let's say we want to compute the median or other quantiles.

```
> #median
> F.inv (0.5)

[1] 23.35132

> #5 number summary
> F.inv (c (0, 0.25, 0.5, 0.75, 1) )

[1] 16.82496 21.79035 23.35132 24.67816 28.89504
```

Note that NPQFs are not the exact inverse of NPCDFs, because of the interpolation method used.

```
> F (F.inv (0.5) )

[1] 0.500011
```

Bivariate Models

We can evaluate bivariate models using the following expressions.

$$f_{X1,X2}(x_1, x_2) = \frac{\sum_i [g(x^*_{[i,1]}, \text{bw}_1, x_1) * g(x^*_{[i,2]}, \text{bw}_2, x_2)]}{n}$$

$$F_{X1,X2}(x_1, x_2) = \frac{\sum_i [G(x^*_{[i,1]}, \text{bw}_1, x_1) * G(x^*_{[i,2]}, \text{bw}_2, x_2)]}{n}$$

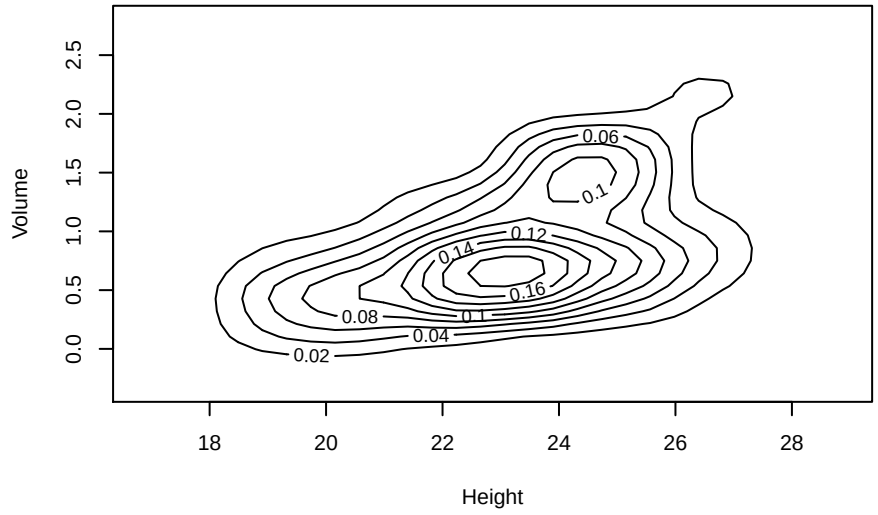
It's also possible to compute what I refer to as a chained quantile function, which evaluates the univariate NPQF for the first variable, and then evaluates the conditional NPQF for the second variable, conditional on the first variable.

We can construct bivariate NPPDFs and NPCDFs using the nppdfmv() and npcdfmv() functions.

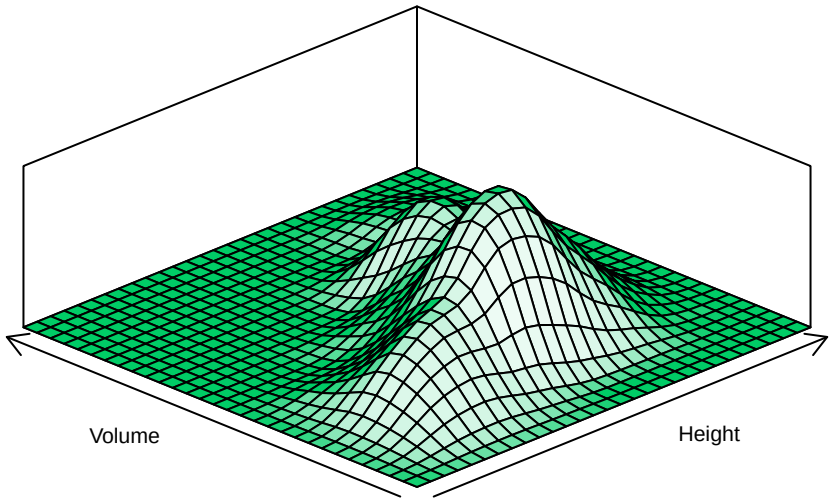
```
> bvf = nppdfmv (trees.2.2)
> bvF = npcdfmv (trees.2.2)
```

Currently, the constructors for multivariate and conditional distributions require a matrix (not data.frame) with unique column names.

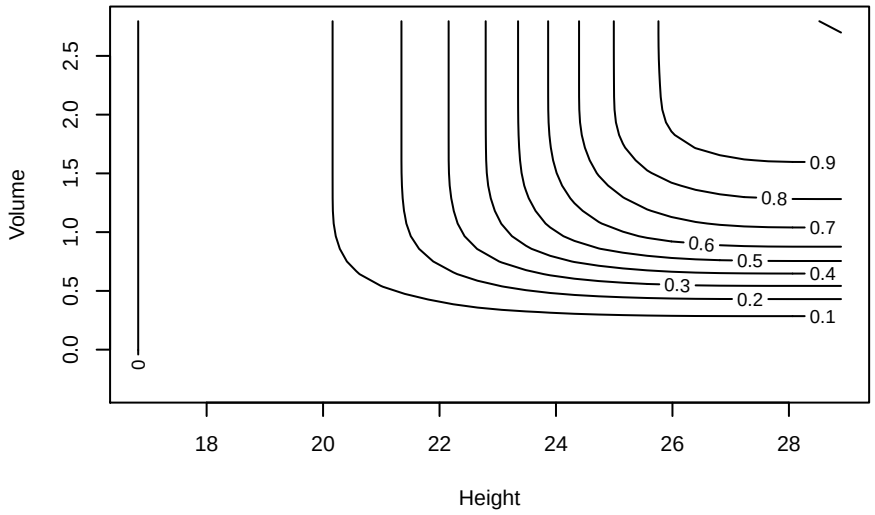
```
> plot (bvf)
```



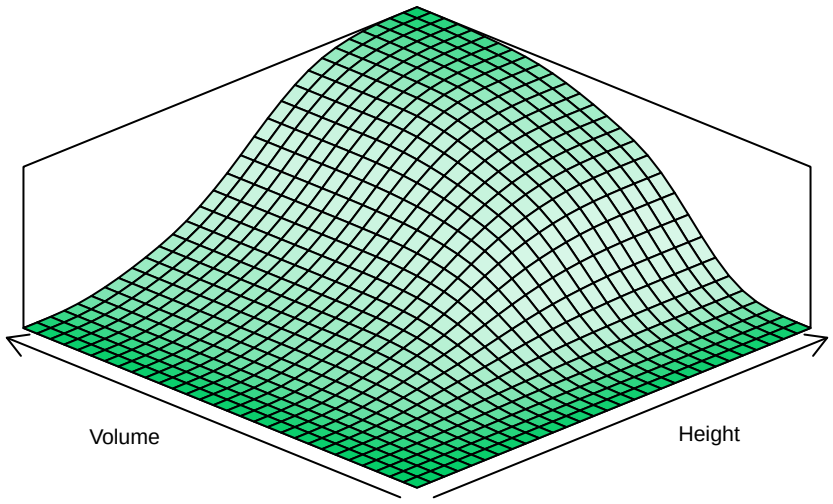

```
> plot (bvf, TRUE)
```




```
> plot (bvF)
```




```
> plot (bvF, TRUE)
```



Or alternatively:

```
> #plot (f, all=TRUE)
```

Like univariate models we can evaluate them.

```
> bvf (c (20, 0.5) )
```

```
[1] 0.0911542
```

Note that the argument can be a vector or a matrix. If it's a standard vector then it's converted to a matrix by rbind(). Each row in the matrix is one point to evaluate. Also note that the columns need to be in the same order as the data matrix.

In general, bivariate models aren't useful in themselves except for visualizing the shape of probability distributions. However, we can use bivariate models (and multivariate models generally) to compute conditional probability distributions and compute multivariate probabilities, which are discussed later.

Refer to the barsurf package for information on how to change colors, if required.

Multivariate Models

Bivariate models generalize to multivariate models (with m > 2) easily, however, they're difficult to visualize.

$$f_{X1,X2,...,Xm}(x_1, x_2, ..., x_m) = \frac{\sum_i [g(x^*_{[i,1]}, \text{bw}_1, x_1) * g(x^*_{[i,2]}, \text{bw}_2, x_2)) * ... * g(x^*_{[i,m]}, \text{bw}_m, x_m)]}{n}$$

$$F_{X1,X2,...,Xm}(x_1, x_2, ..., x_m) = \frac{\sum_i [G(x^*_{[i,1]}, \text{bw}_1, x_1) * G(x^*_{[i,2]}, \text{bw}_2, x_2)) * ... * G(x^*_{[i,m]}, \text{bw}_m, x_m)]}{n}$$

Where m is the number of variables.

Likewise, it's possible to compute the chained quantile function for three or more variables.

```
> mvf = nppdfmv (trees.2.3)
> mvF = npcdfmv (trees.2.3)
```

Currently, you can't plot multivariate models (with m > 2). However, refer to Appendix 3 on how to plot all marginal distributions.

Conditional Models

We can derive conditional models from multivariate models.

We can compute a conditional distribution (conditional on one variable) using:

$$f_{X2|X1}(x_2) = \frac{f_{X1,X2}(x_1, x_2)}{f_{X1}(x_1)}$$

$$F_{X2|X1}(x_2) = \int_{-\infty}^{x_2} \frac{f_{X1,X2}(x_1, u)}{f_{X1}(x_1)} du$$

And we can compute a conditional distribution (conditional on two variables) using:

$$f_{X3|X1,X2}(x_3) = \frac{f_{X1,X2,X3}(x_1, x_2, x_3)}{f_{X1,X2}(x_1, x_2)}$$

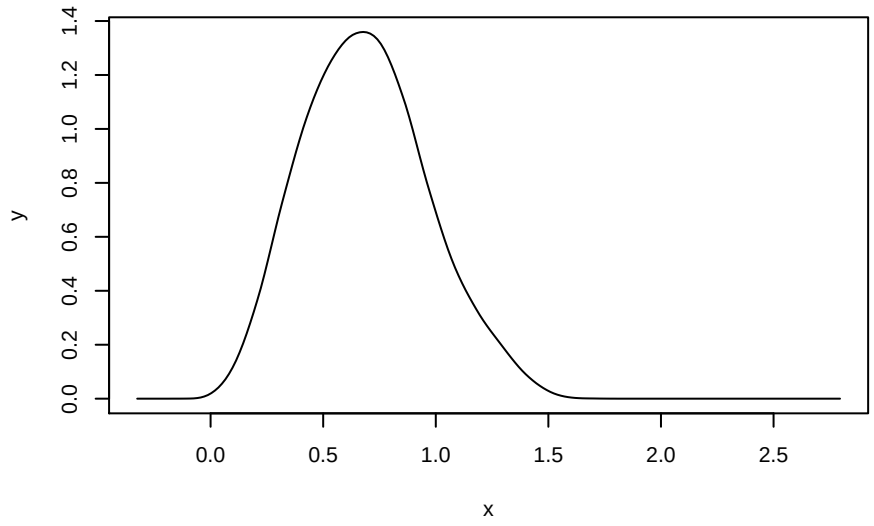
$$F_{X3|X1,X2}(x_3) = \int_{-\infty}^{x_3} \frac{f_{X1,X2,X3}(x_1, x_2, u)}{f_{X1,X2}(x_1, x_2)} du$$

And this can be generalized to m variables.

Note that it's possible for the denominator to be zero, in which case, in theory, we can't evaluate the functions. However, the R functions below returns NAs.

We can construct conditional nonparametric models using the nppdfc(), npcdfc() and npcdfc.inverse() functions.

```
> median.Height = median (Height)
> median.Girth = median (Girth)
> cf = nppdfc (Volume ~ c (Height=median.Height, Girth=median.Girth), trees.2.3)
> cF = npcdfc (Volume ~ c (Height=median.Height, Girth=median.Girth), trees.2.3)
> cF.inv = npcdfc.inverse (
    Volume ~ c (Height=median.Height, Girth=median.Girth),
    trees.2.3)

> plot (cf)
```



Fuzzy Clustering (and Weighted Data)

Fuzzy clustering computes a membership matrix, with each row representing each data point and each column representing each cluster (not variable). Each row represents the membership of that data point in each cluster as numbers in the interval (0, 1).

The following computes memberships for three clusters and then the weights for the first cluster.

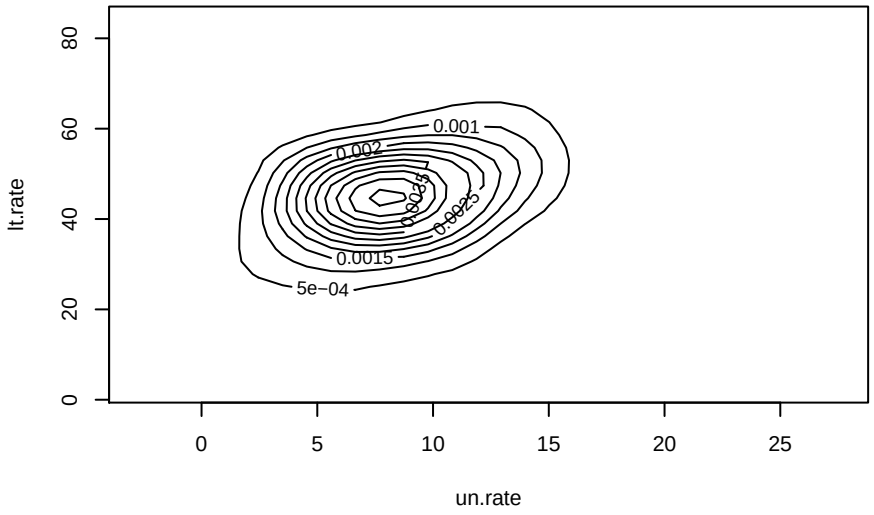
```
> w = FKM.gk (unemployment.2, k=3, seed=2)$U [,1]
> w = w / sum (w)
```

Note that a weighted scatterplot is given in Appendix 6.

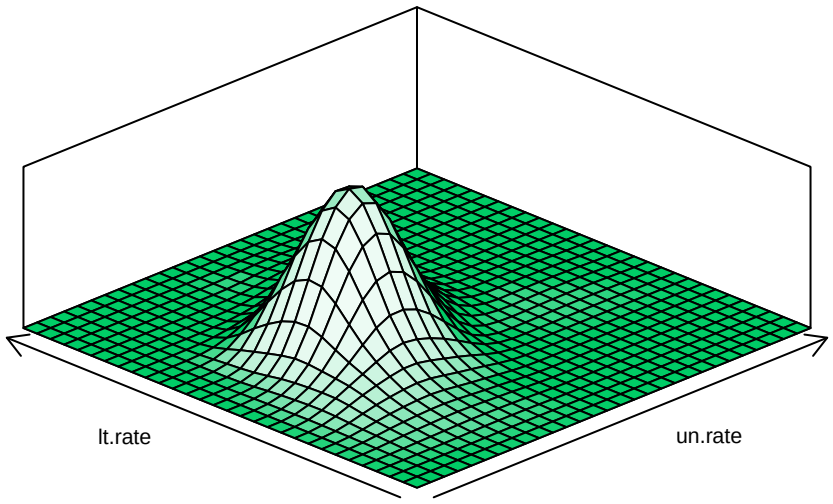
However, fuzzy clustering is limited by itself. I use the term "Membership-Weighted Data Analysis" to refer to the process of modelling the data, weighted according to a membership matrix. I'm going to focus on nonparametric probability distributions, however, there are other ways of modelling such data.

We can compute a weighted bivariate model easily.

```
> wf = nppdfmv (unemployment.2, w=w)
> plot (wf)
```




```
> plot (wf, TRUE)
```



Computing Multivariate Probabilities

As given earlier, we can compute the probability that a random variable is between a pair of values as:

$$\mathbb{P}(a \leq X \leq b) = F(b) - F(a)$$

We can compute the probability that two random variables are between two pairs of values as:

$$\mathbb{P}(a_1 \leq X_1 \leq b_1, a_2 \leq X_2 \leq b_2) = F_{X1,X2}(b_1, b_2) - [F_{X1,X2}(a_1, b_2) + F_{X1,X2}(b_1, a_2)] + F_{X1,X2}(a_1, a_2)$$

Where a is a vector of lower limits and b is a vector of upper limits.

This is the volume under the bivariate PDF.

It could be computed by computing the multiple integral of the multivariate PDF, numerically. However, it's more efficient to evaluate the multivariate CDF, combinatorically.

This can be generalized to three or more variables, however, the formulation is difficult to write.

We can use the comb.prob() function, which has three arguments.

```
> #probability that:
> #(   20  <= Height <= 30,
> #    20  <= Girth  <= 24,
> #    0.2 <= Volume <= 0.8
> #)
> a = c (20, 20, 0.2)
> b = c (30, 24, 0.8)
> comb.prob (mvF, a, b)

[1] 0.04821105
```

Note that it's possible to compute multiple regions at once by making a and b matrices with each row representing one region.

Expanding on an earlier point, a and b (or their columns, if matrices) must be in the same order as the data matrix. So, in this example, height must come first, girth must come second and volume must come last.

Random Number Generation

We can generate random numbers using the nprng() function which uses NPQF or chained quantile functions for inversion sampling.

```
> chained.inv = chained.npcdfmv.inverse (trees.2.3)
> trees.2.rs = nprng (chained.inv, 31)

> Height.rs = trees.2.rs [,"Height"]
> Girth.rs = trees.2.rs [,"Girth"]
> Volume.rs = trees.2.rs [,"Volume"]
```

The trees.2.rs data is given in Appendix 7.

Note that I've used the same sample size, however, you could use any sample size.

Conditional Probabilities and Conditional Statistics

It's possible to compute conditional densities, probabilities, medians and quantiles from conditional distributions, simply by evaluating the conditional distributions.

```
> #probability that volume is between 0.2 and 0.8 cubic meters
> #given median height and median girth
> cF (0.8) - cF (0.2)

[1] 0.6354307

> #median of volume in cubic meters
```

```
> #given median height and median girth
> cF.inv (0.5)
```

```
[1] 0.6835995
```

Likewise, it's possible to compute conditional means and modes from conditional probability density functions.

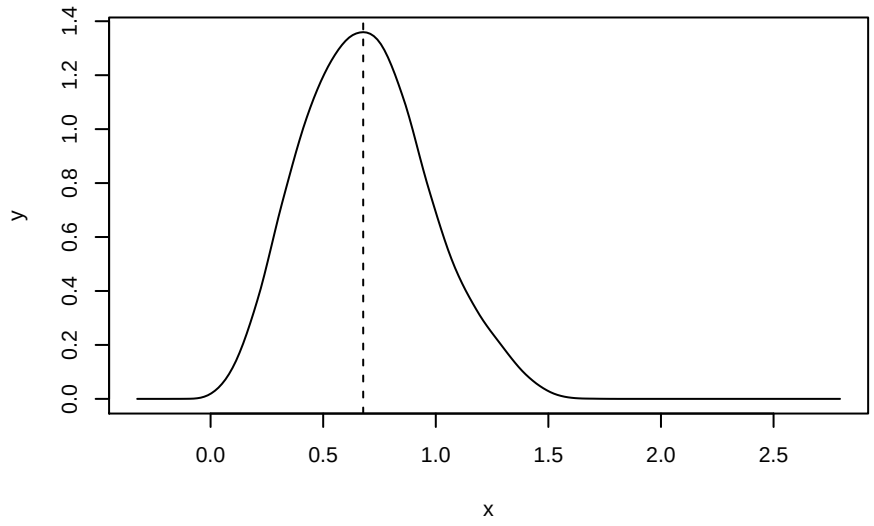
```
> #mean of volume in cubic meters
> #given median height and median girth
> npmean (cf)
```

```
[1] 0.6881425
```

```
> #mode of volume in cubic meters
> #given median height and median girth
> mode.Volume.c = npmode (cf)
> mode.Volume.c
```

```
[1] 0.6776767
```

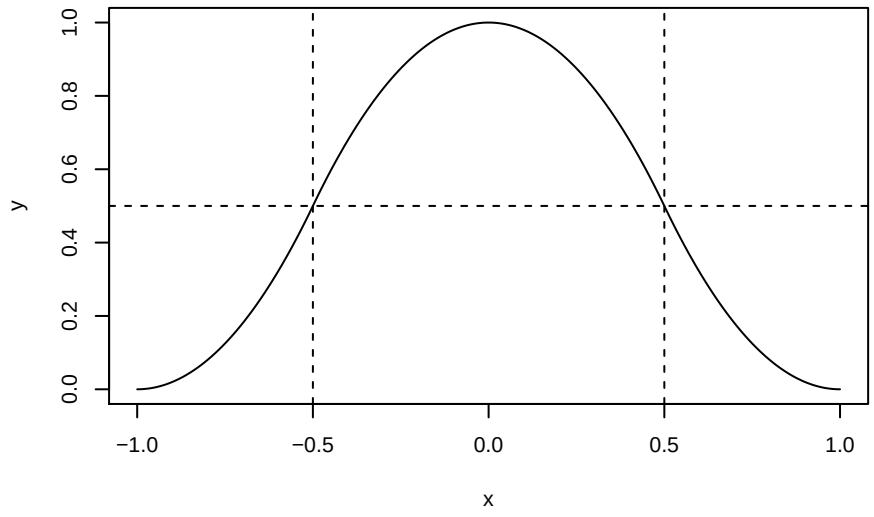
```
> plot (cf)
> abline (v=mode.Volume.c, lty=2)
```



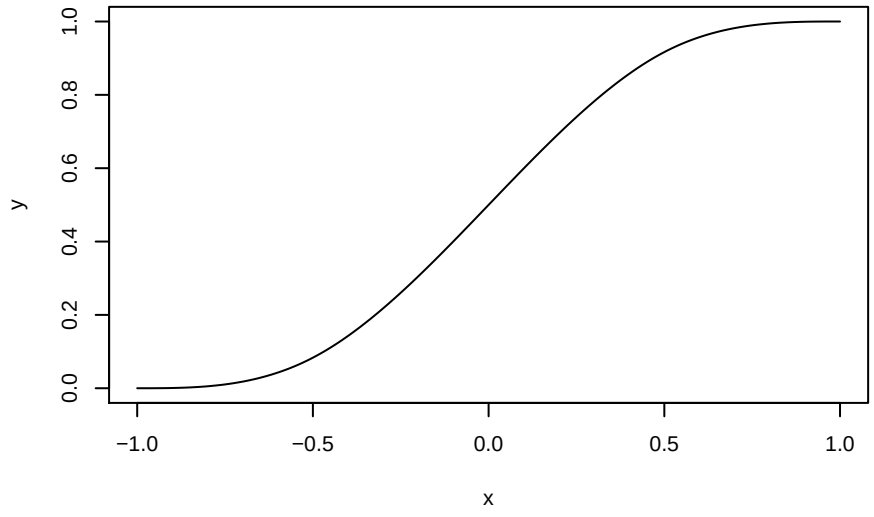
Appendix 1:
Simplified Bell Curves

```
> x = seq (-1, 1, length.out=200)
```

```
> y = sbc.pdf (x)
> plot (x, y, type="l")
> abline (v=c (-0.5, 0.5), h=c (0.5), lty=2)
```




```
> y = sbc.cdf (x)
> plot (x, y, type="l")
```



Appendix 2:
More Smoothing Information

The kernel PDF should be zero centered, symmetrical, smooth and integrate to one, with positive densities over the interval (-1, 1) and zero densities outside this interval. And the kernel CDF should differentiate to a PDF with these properties.

The smoothness parameter (which can be a scalar or a vector) is multiplied by the range for each variable, to give a (vector) bandwidth parameter.

$$\text{bw}_j = \text{smoothness}_j * [\max(x^*_{[,j]}) - \min(x^*_{[,j]})]$$

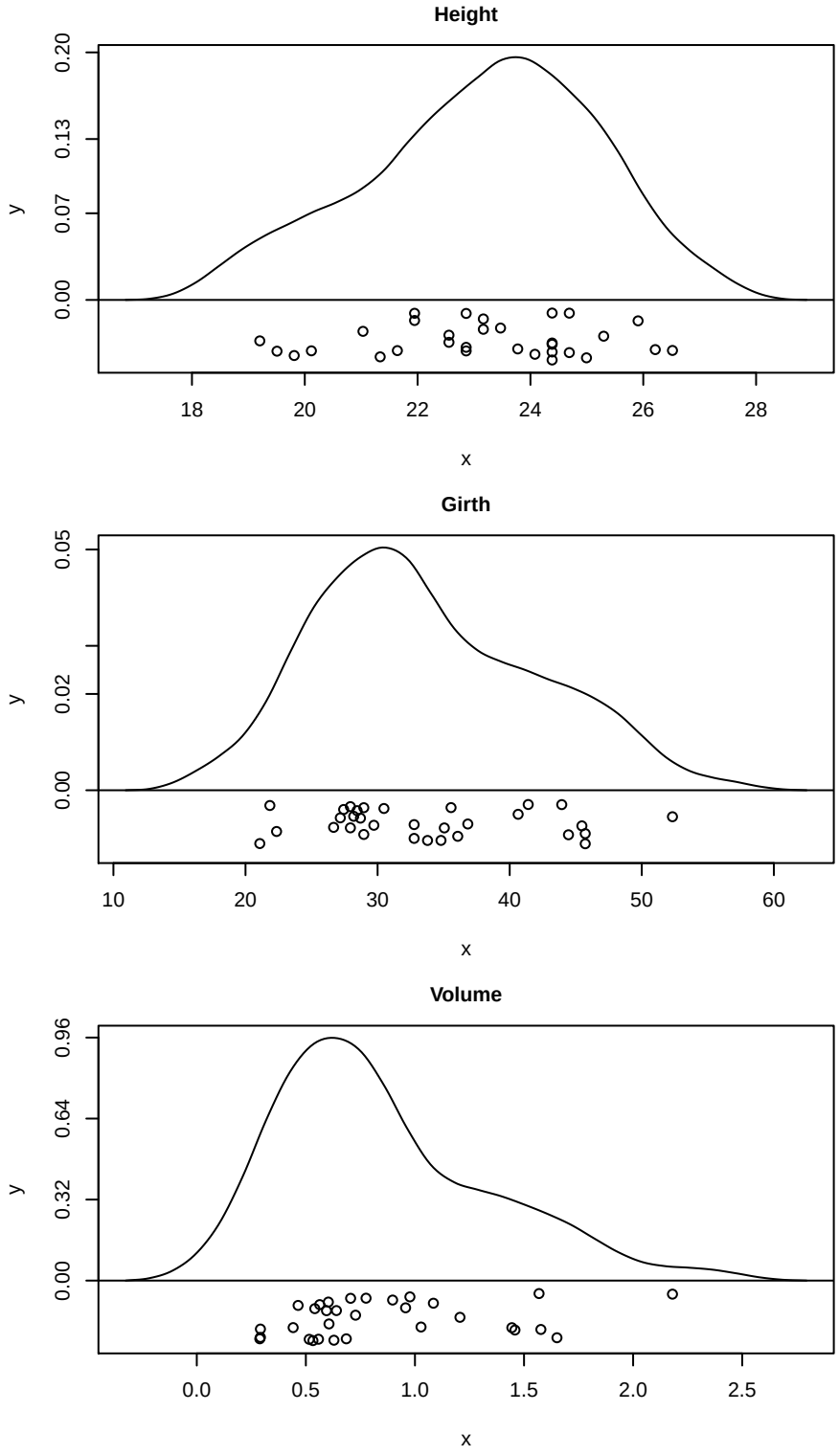
If the bandwidth parameter is provided then the smoothness parameter is ignored.

Also note that you may want to set spline=FALSE or increase the number of control points, if you decrease the bandwidth.

Marginal distributions can be used to determine (near) optimal smoothing parameters.

Appendix 3:
Plotting Marginal Distributions

```
> plot (marginal (mvf), 3, 1, TRUE)
```



Appendix 4:
Empirical Continuous Cumulative Distribution Functions and Their Inverses

Continuous ECDFs require unique x values, and a small amount of random variation is automatically added if they're not unique.

Given any x value (from the data), the continuous ECDF at that point can be computed using the following expression.

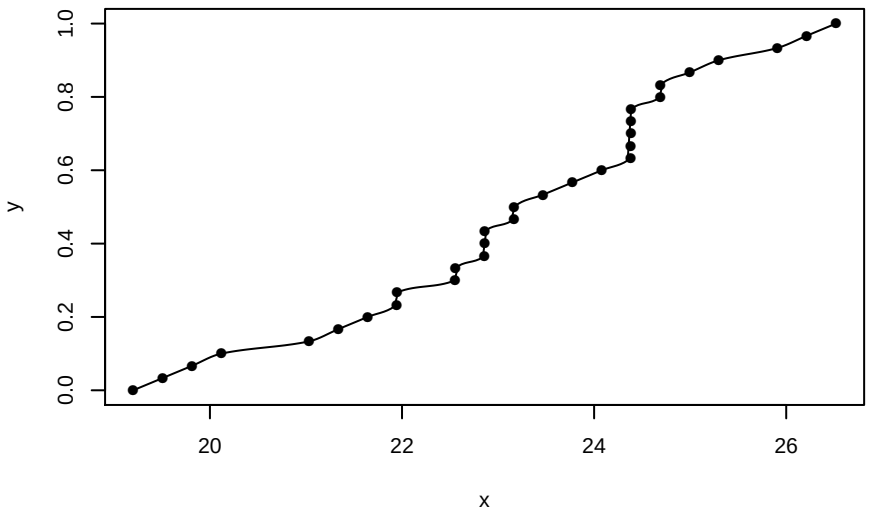
$$\mathbb{P}(X \leq x) = F(x) = \frac{\sum_i I(x_i^* \leq x) - 1}{n - 1}$$

Where $I()$ is 1 if the enclosed expression is true and is 0 if false.

The resulting points can be interpolated by a cubic hermite spline. However, the resulting functions don't necessarily appear smooth.

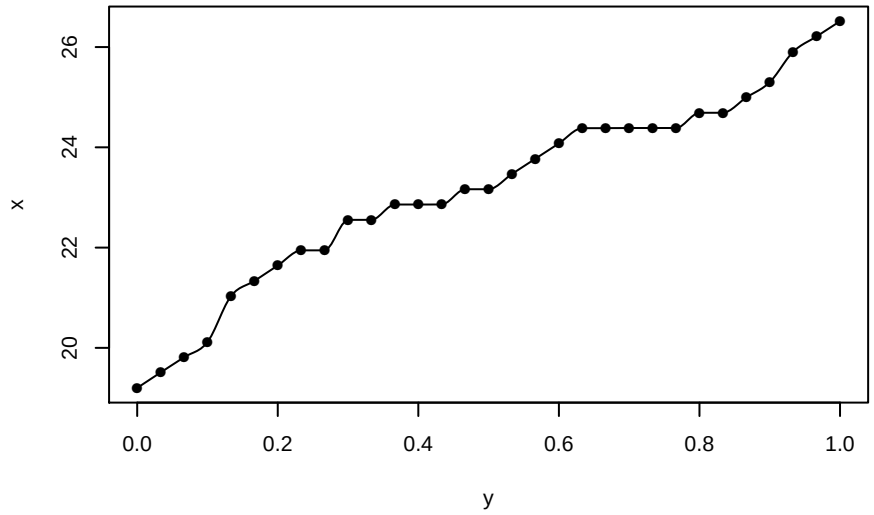
We can compute a continuous ECDF and plot it in a similar way to a univariate NPCDF.

```
> F = eccdf (Height)
> plot (F, TRUE)
```



And compute a continuous EQF and plot it in a similar way to a univariate NPQF.

```
> F.inv = eccdf.inverse (Height)
> plot (F.inv, TRUE)
```



Likewise, they can be evaluated.

```
> #median
> F.inv (0.5)

[1] 23.16439

> #5 number summary
> F.inv (c (0, 0.25, 0.5, 0.75, 1) )

[1] 19.20064 21.94504 23.16439 24.38539 26.51573
```

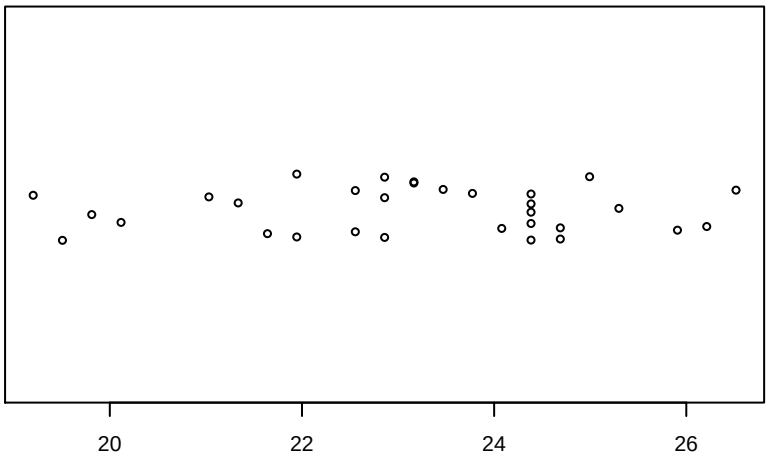
Both of these models can be weighted.

Note that like NPQFs (using kernel smoothing), continuous EQFs are not the exact inverse of continuous ECDFs. And the difference may be greater than NPQFs, because in general, they're not as smooth.

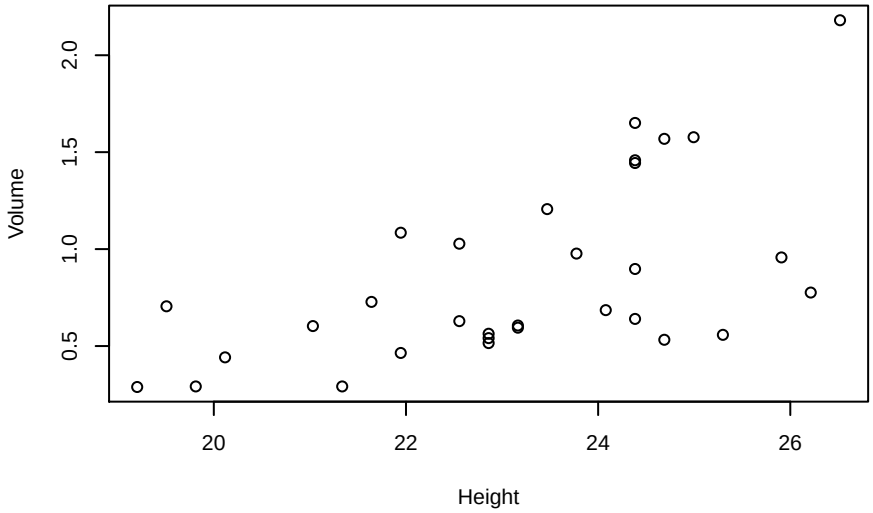
Also note that unlike the other probability distributions in this package, empirical models don't use smoothing. And ECQFs may be preferable to NPQFs if you want to compute the median or other quantiles, without any smoothing. These quantiles can be regarded as quantiles of the data itself rather than estimates derived from a model.

Appendix 5:
The trees.2 Data

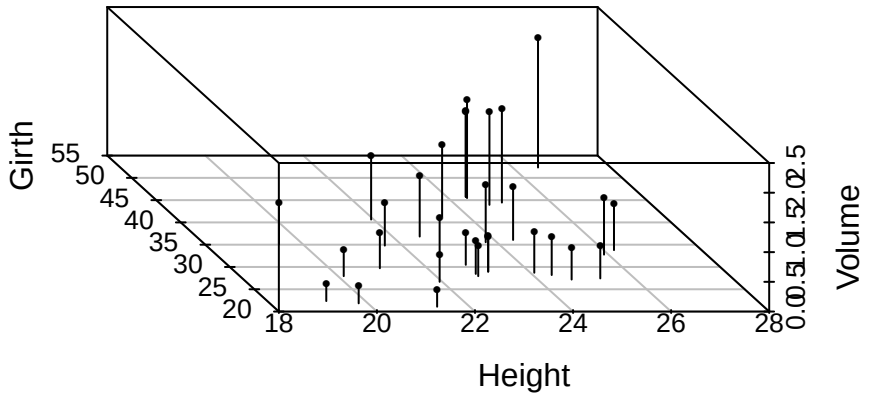
```
> head (trees.2.3, 3)

      Height  Girth    Volume
[1,] 21.3360 21.082 0.2916630
[2,] 19.8120 21.844 0.2916630
[3,] 19.2024 22.352 0.2888314

> tail (trees.2.3, 3)

       Height  Girth   Volume
[29,] 24.3840 45.720 1.458315
[30,] 24.3840 45.720 1.444157
[31,] 26.5176 52.324 2.180394

> summary (trees.2.3)

     Height          Girth           Volume
 Min.   :19.20   Min.   :21.08   Min.   :0.2888
 1st Qu.:21.95   1st Qu.:28.07   1st Qu.:0.5493
 Median :23.16   Median :32.77   Median :0.6853
 Mean   :23.16   Mean   :33.65   Mean   :0.8543
 3rd Qu.:24.38   3rd Qu.:38.73   3rd Qu.:1.0562
 Max.   :26.52   Max.   :52.32   Max.   :2.1804

> skewness (trees.2.3)

    Height      Girth     Volume
-0.3748690  0.5263163  1.0643575

> cov (trees.2.3)

         Height     Girth    Volume
Height 3.771863  8.038694 0.5408160
Girth  8.038694 63.534802 3.5881865
Volume 0.540816  3.588187 0.2166597

> stripchart (Height, "jitter", pch=1)
```




```
> plot (Height, Volume)
```

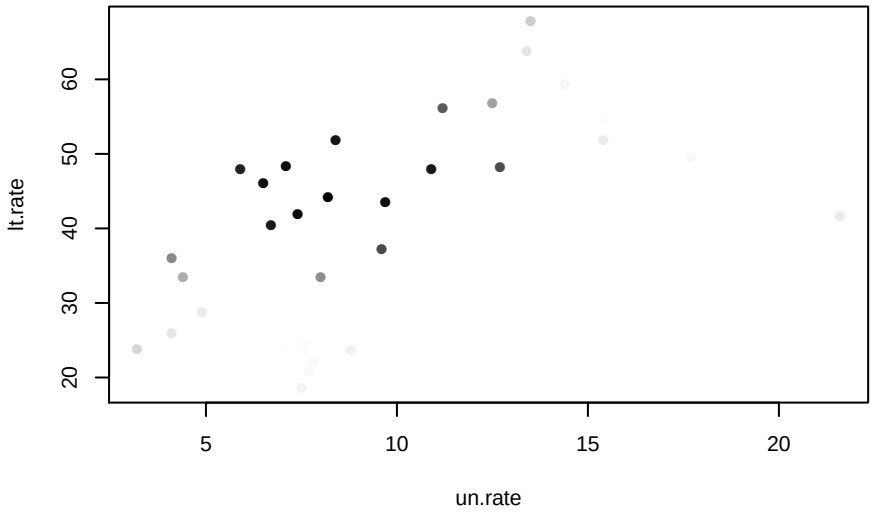



```
> scatterplot3d (Height, Girth, Volume, type="h", angle=112.5, pch=16)
```


Girth
55
50
45
40
35
30
25
20
18
20
22
24
26
28
Height
Volume

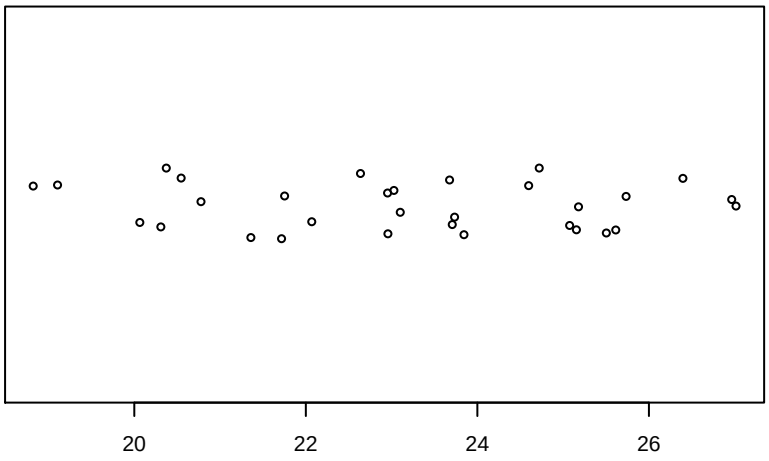
Appendix 6: The unemployment.2 Data (Membership-Weighted)

```
> head (unemployment.2, 3)

     un.rate lt.rate
[1,]     7.1    48.3
[2,]    11.2    56.2
[3,]     6.7    40.5

> tail (unemployment.2, 3)

      un.rate lt.rate
[30,]     4.1    36.0
[31,]    13.4    63.8
[32,]     8.8    23.7

> summary (unemployment.2)

    un.rate          lt.rate
 Min.   : 3.200   Min.   :18.60
 1st Qu.: 6.925   1st Qu.:28.07
 Median : 8.100   Median :42.70
 Mean   : 9.478   Mean   :41.08
 3rd Qu.:12.550   3rd Qu.:50.17
 Max.   :21.600   Max.   :67.80

> skewness (unemployment.2)

    un.rate     lt.rate
 0.87244093 -0.01797293

> cov (unemployment.2)

         un.rate   lt.rate
un.rate 18.49660  34.25571
lt.rate 34.25571 185.49706

> s = 1 - w / max (w)
> plot (unemployment.2, pch=16, col=rgb (s, s, s) )
```

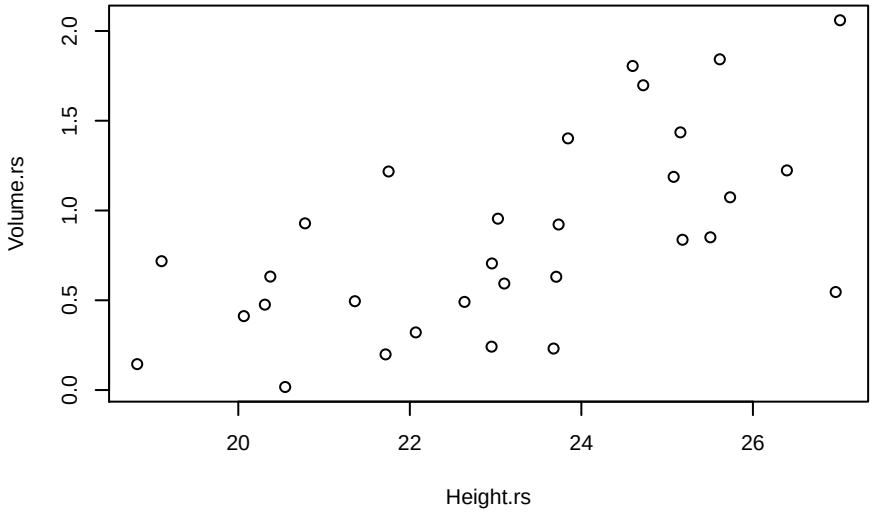

lt.rate
20
30
40
50
60
5
10
15
20
un.rate

Appendix 7:
Random Sample

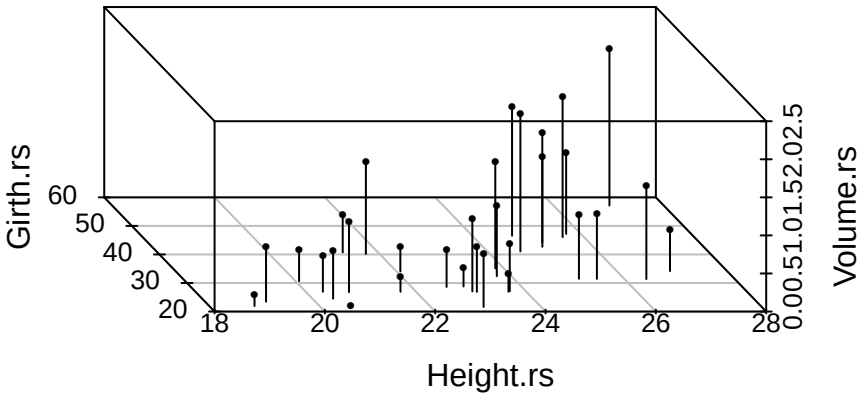
```
> head (trees.2.rs, 3)

       Height    Girth    Volume
[1,] 22.95304 28.79547 0.2416357
[2,] 20.06461 30.65187 0.4118361
[3,] 20.37315 24.48549 0.6319733

> tail (trees.2.rs, 3)

        Height    Girth    Volume
[29,] 21.71691 26.93537 0.1985119
[30,] 24.72109 46.56862 1.6976521
[31,] 23.67594 26.96112 0.2310965

> summary (trees.2.rs)

     Height          Girth           Volume
 Min.   :18.82   Min.   :21.59   Min.   :0.01697
 1st Qu.:21.54   1st Qu.:26.93   1st Qu.:0.48333
 Median :23.10   Median :31.38   Median :0.71777
 Mean   :23.18   Mean   :33.09   Mean   :0.84802
 3rd Qu.:25.12   3rd Qu.:40.46   3rd Qu.:1.20243
 Max.   :27.02   Max.   :57.14   Max.   :2.05974

> skewness (trees.2.rs)

    Height      Girth     Volume
-0.1506225  0.8170970  0.5812684

> cov (trees.2.rs)

           Height     Girth    Volume
Height  5.2555253 13.090285 0.7598902
Girth  13.0902852 79.860959 3.7643193
Volume  0.7598902  3.764319 0.2905019

> stripchart (Height.rs, "jitter", pch=1)
```




```
> plot (Height.rs, Volume.rs)
```




```
> scatterplot3d (Height.rs, Girth.rs, Volume.rs, type="h", angle=112.5, pch=16)
```


Girth.rs
60
50
40
30
20
18
20
22
24
26
28
Height.rs
0.0
0.5
1.0
1.5
2.0
2.5
Volume.rs

Appendix 8:
Regression-Like Models

Regression-like models (in this package) construct a spline by constructing a sequence of conditional distributions, and then computing a descriptive statistic (mean, mode, median or quantile) from each conditional distribution.

The constructors require a formula object with two variables. It's possible to include a data matrix with three or more variables, in which case, the conditional distributions are conditional on the medians of the other variables.

Nonparametric regression models with two variables should be equivalent to regression-like models with two variables. However, this isn't true for three or more variables.

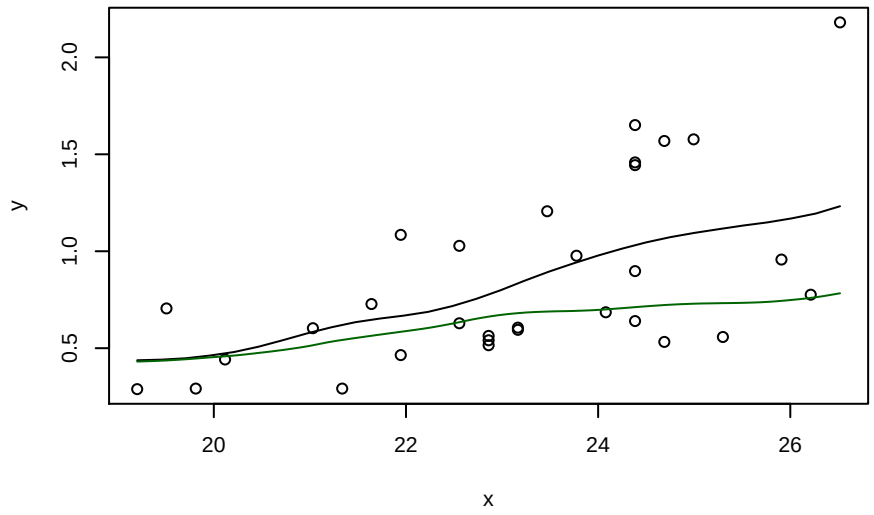
Linear (or additive) models allow you to interpret the effect of any predictor variable on the response, (mostly) independently of the other effects, unless there are interactions. In principle, regression-like models don't allow such interpretations, because the effect of any predictor variable on the response is conditional on the other predictor variables' values.

I'm assuming that if the effects are approximately additive and there's sufficient data density in the center of the multivariate distribution, then the conditioning values shouldn't make a lot of difference. However, I haven't tested this assumption. Even if true, in general, I recommend using linear (or additive) models rather than regression-like models for three or more variables.

Note that currently, the PDF of the conditioning variables must not evaluate to NA (refer to the section on conditional distributions). Also note that regression-like models with three or more variables haven't had a lot of testing.

The conditional mean and mode:

```
> reg.mean = reglike (mean (Volume) ~ Height, trees.2.2)
> reg.mode = reglike (mode (Volume) ~ Height, trees.2.2)
> plot (reg.mean, TRUE)
> lines (reg.mode, col="darkgreen")
```



Note that some of the conditional distributions are bimodal.

The conditional median and quartiles:

```
> reg.median = reglike (median (Volume) ~ Height, trees.2.2)
> reg.q.25= reglike (quantile (Volume, 0.25) ~ Height, trees.2.2)
> reg.q.75= reglike (quantile (Volume, 0.75) ~ Height, trees.2.2)
> plot (reg.median, TRUE)
> lines (reg.q.25, col="darkgreen")
> lines (reg.q.75, col="darkgreen")
```