

Multivariate Generalized Kernel Smoothing and Related Statistical Methods

Abby Spurdle

July 14, 2020

Mass functions, density functions, distribution functions and quantile functions via continuous kernel smoothing, and to a lesser extent, discrete kernel smoothing. Also, supports categorical distributions and smooth empirical-like distributions. There are univariate, multivariate and conditional distributions, including multivariate-conditional distributions and conditional distributions with mixed input types, along with functions for plotting univariate, bivariate and trivariate distributions. Conditional categorical distributions with mixed input types can be used for statistical classification purposes. And there are extensions for computing multivariate probabilities, multivariate random numbers, moment-based statistics, robust-based statistics and mode estimates.

Introduction

This is an R package for multivariate generalized kernel smoothing, as per the title.

Kernel smoothing is generalized, by estimating:

- Both discrete and continuous probability distributions.
- Probability mass functions (PMFs), probability density functions (PDFs), cumulative distribution functions (CDFs) and quantile functions (QFs).
- In the continuous case, multivariate and conditional distributions, including multivariate-conditional distributions.

Also, there are categorical distributions (univariate and conditional), smooth empirical-like distributions (univariate), and conditional distributions with mixed input types.

Discrete kernel smoothing, could also be described as frequency smoothing.

By default, univariate continuous kernel smoothing models use cubic Hermite splines as intermediate models. Corresponding quantile functions (which are spline only) are constructed by fitting a CDF using a spline, then transposing the control points.

Categorical variables are assumed to be ordinal, however, this assumption is only relevant to the interpretation of the CDF and QF. Empirical-like models are derived from empirical cumulative distribution functions. There's a small modification to the standard formula, and the resulting points are interpolated by a cubic Hermite spline, in a similar way to

univariate continuous kernel smoothing models.

A list of models and some notes on terminology, are given in appendices A and B. Also, there are appendices with formulae, but they need some polishing.

All models can be given frequencies or weights, and there are examples of modelling fuzzy clusters with weighted multivariate kernel density estimation, in an appendix.

There are plot methods for all univariate models, and for multivariate models but only with two or three random variables.

Often the goal of kernel smoothing is simply to plot the distribution, as an exploratory tool, however, these models can be used for a variety of purposes.

Conditional categorical distributions with mixed input types, can be used for statistical classification purposes.

Also, there are extensions for computing probabilities, random number generation and parameter-like estimation, including moment-based, order-based, robust-based and mode estimates.

Notes:

- This package is based on self-referencing function objects. (Which are equivalent to the {d, p, q, r} approach used in R's stats package).
- Bivariate/trivariate plotting functions use the barsurf package, which uses the base graphics system.
- Variable names are taken from column names or list names.
- This package gives precedence to vectors and matrices, over lists and data frames. (One partial exception is categorical data which allow a list of integer/factor/character vectors).
- This vignette contains a small amount of non-visible R code, to produce multi-panel plots.

Currently, there are some limitations:

- Bandwidth selection methods are simplistic.
- Continuous models are unbounded/untruncated only.
- Parameter estimates and distribution sets, need substantial improvement.
- Expanding on the previous set of points, the input format for data, isn't as simple as it could be.

These are high priorities for future updates, which may also support categorical smoothing.

Preliminary Code

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

```
> library (probhat)      #required
> library (vectools)     #optional, used in appendix
> library (fclust)       #optional, used in appendix
> library (scatterplot3d) #optional
```

Note that the probhat package imports the intoo, barsurf and kubik packages.

I will set global options for viewing PDF documents electronically, and default colors to green:

```
> set.ph.options (rendering.style="e", theme="green")
```

And I will construct some data objects:

```
> ph.data.prep ()
```

This function emulates a script.

The script and the resulting datasets are given in the last two appendices.

Discrete Kernel Smoothing

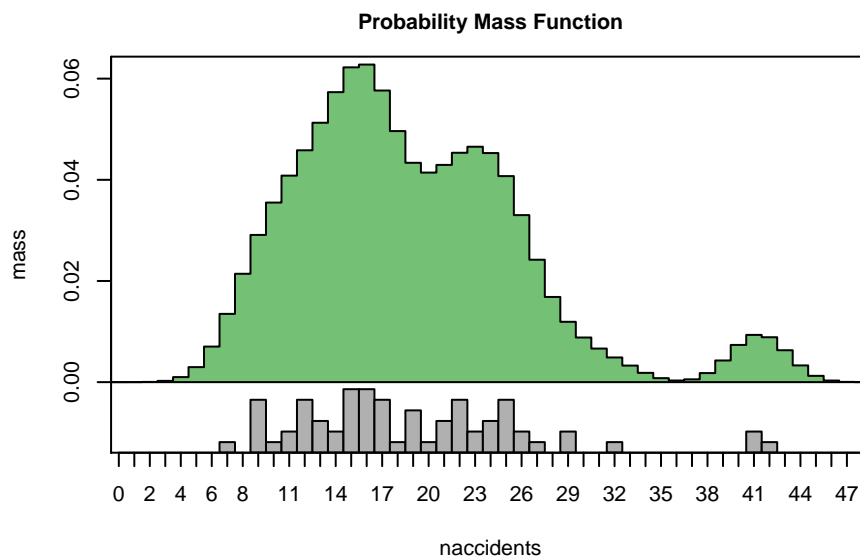
We can construct a $PMF_{(UV)} \sim DKS$ object, using the `pmfuv.dks` constructor.

Likewise, we can construct $CDF_{(UV)} \sim DKS$ and $QF_{(UV)} \sim DKS$ objects, using the `cdfuv.dks` and `qfuv.dks` constructors.

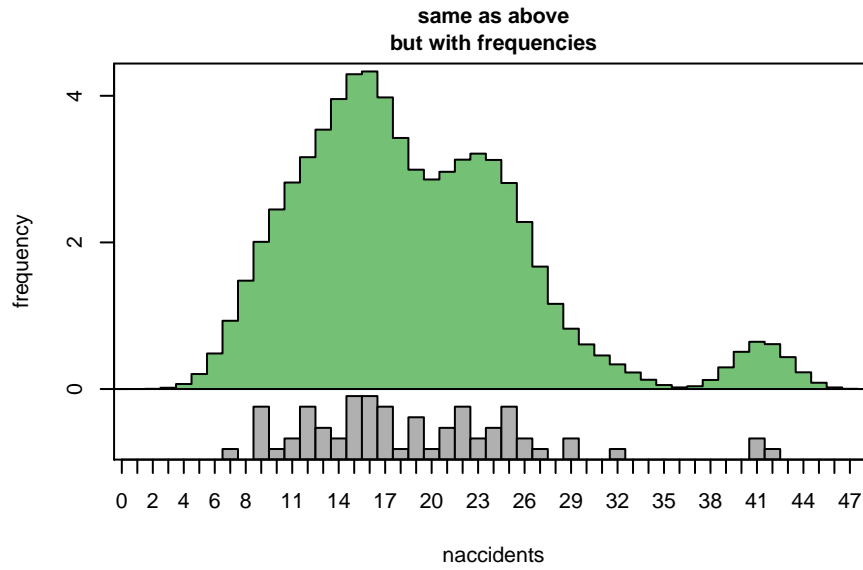
Here are examples, using traffic accident data, derived from the “Traffic” dataset in the MASS package:

```
> dfh <- pmfuv.dks (traffic.bins, traffic.freq, lower=0)
> dFh <- cdfuv.dks (traffic.bins, traffic.freq, lower=0)
> dFht <- qfuv.dks (traffic.bins, traffic.freq, lower=0)

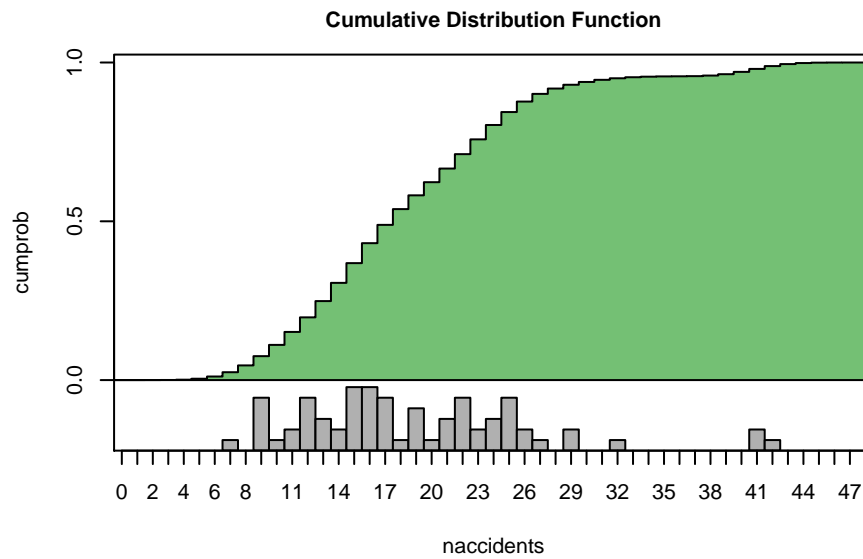
> plot (dfh, TRUE, main="Probability Mass Function")
```



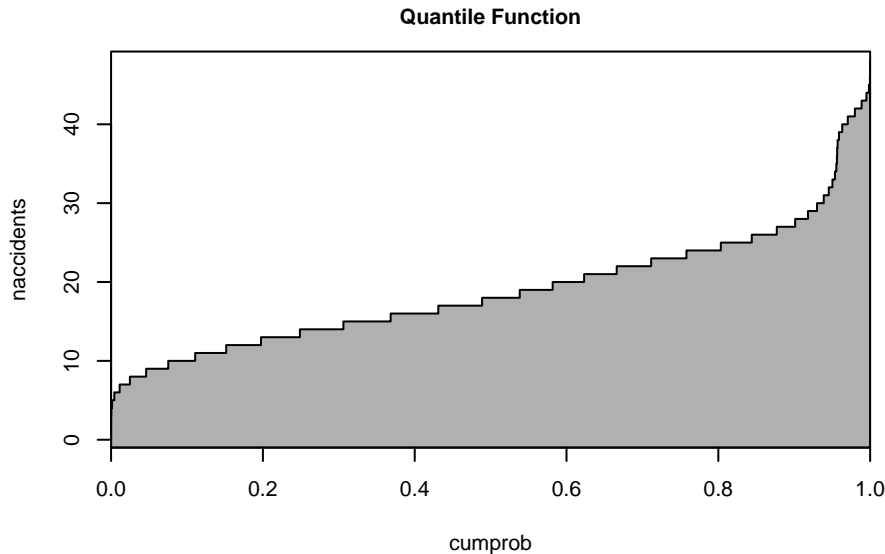
```
> plot (dfh, TRUE, freq=TRUE, main="same as above\nbut with frequencies")
```



```
> plot (dFh, TRUE, main="Cumulative Distribution Function")
```



```
> plot (dFht, main="Quantile Function")
```



Here, `traffic.bins` is a one-column matrix of integer-indexed bins, and `traffic.freq` is a vector of counts.

The resulting objects are function objects.

By default, the PMF maps an integer vector (of quantiles) to a numeric vector (of probabilities), the CDF maps quantiles to cumulative probabilities, and the QF maps probabilities to quantiles:

```
> dfh (10)
[1] 0.03549946
> dfh (10, freq=TRUE)
[1] 2.449463
> dFht (c (0.25, 0.5, 0.75) )
[1] 14 18 23
```

Note that PMFs can be constructed, plotted and evaluated with `freq=TRUE`, which results in frequencies.

Likewise, if constructed with `freq=TRUE`, they can be plotted and evaluated with `freq=FALSE`.

Also note that this dataset is not an ideal example. (Essentially, it gives frequencies of frequencies).

Continuous Kernel Smoothing: Univariate Probability Distributions

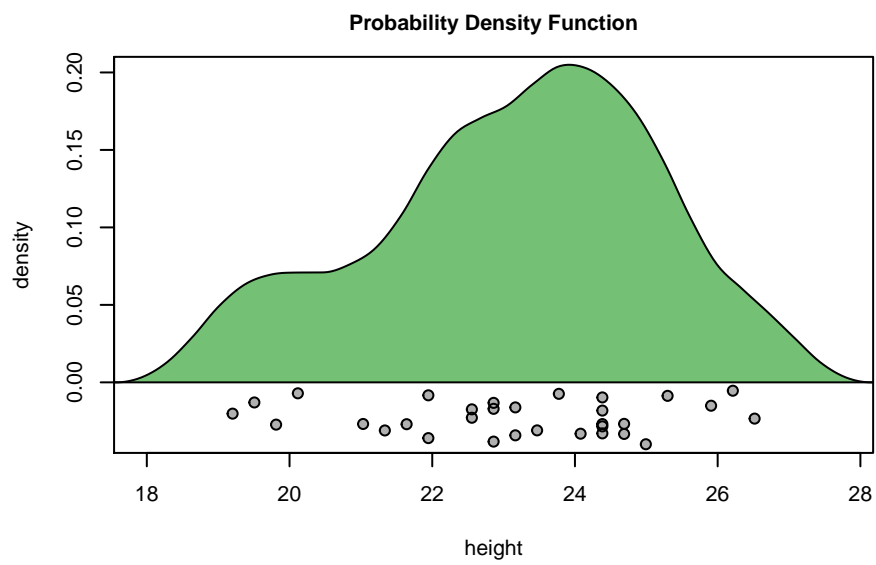
We can construct a $\text{PDF}_{(UV)} \sim \text{CKS}$ object, using the `pdfuv.cks` constructor.

Likewise, we can construct $\text{CDF}_{(UV)} \sim \text{CKS}$ and $\text{QF}_{(UV)} \sim \text{CKS}$ objects, using the `cdfuv.cks` and `qfuv.cks` constructors.

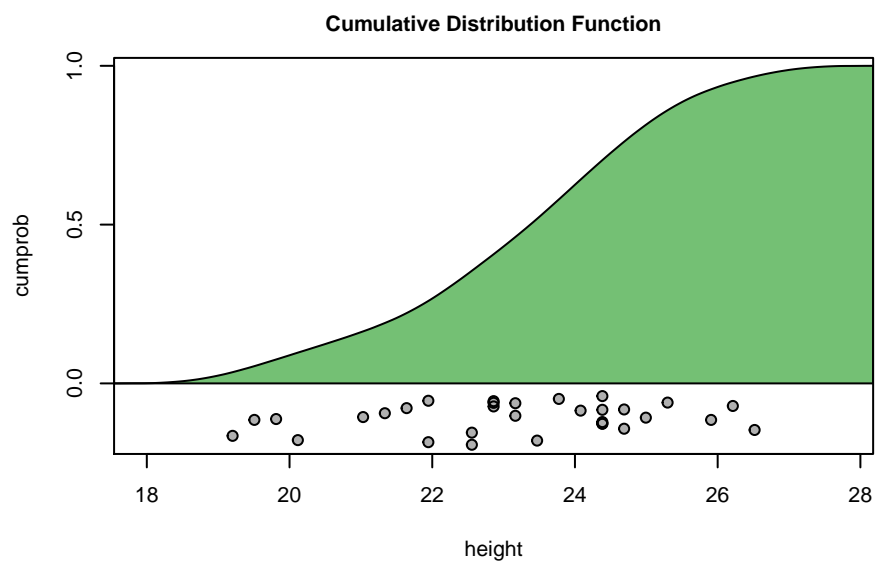
Here are examples, using height data, derived from the “trees” data in the `datasets` package:

```
> cfh <- pdfuv.cks (height)
> cFh <- cdfuv.cks (height)
> cFht <- qfuv.cks (height)
```

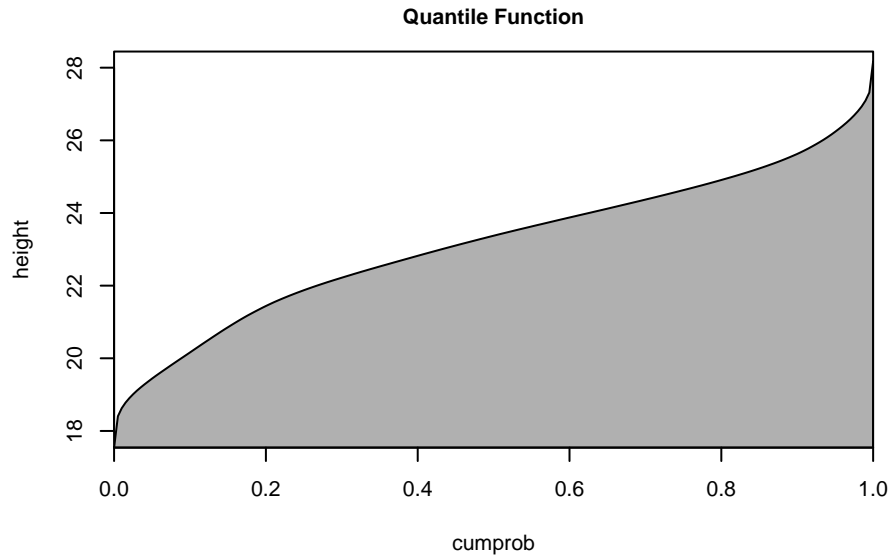
```
> plot (cfh, TRUE, main="Probability Density Function")
```



```
> plot (cFh, TRUE, main="Cumulative Distribution Function")
```



```
> plot (cFht, main="Quantile Function")
```



The resulting objects are function objects.

The PDF maps a numeric vector (of quantiles) to a numeric vector (of probabilities), the CDF maps quantiles to cumulative probabilities, and the QF maps probabilities to quantiles.

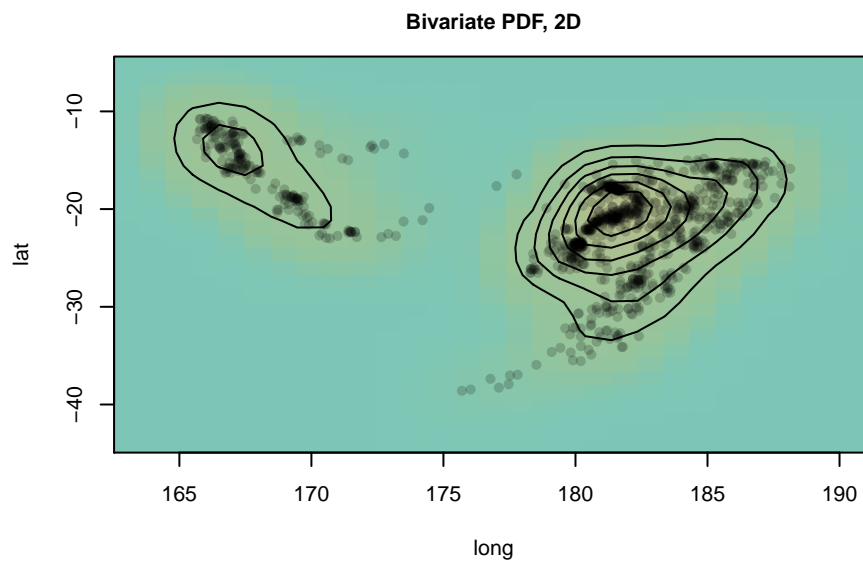
```
> cfh (22)
[1] 0.1416708
> cFht (c (0.25, 0.5, 0.75) )
[1] 21.87344 23.37409 24.62905
```

Continuous Kernel Smoothing: Multivariate Probability Distributions

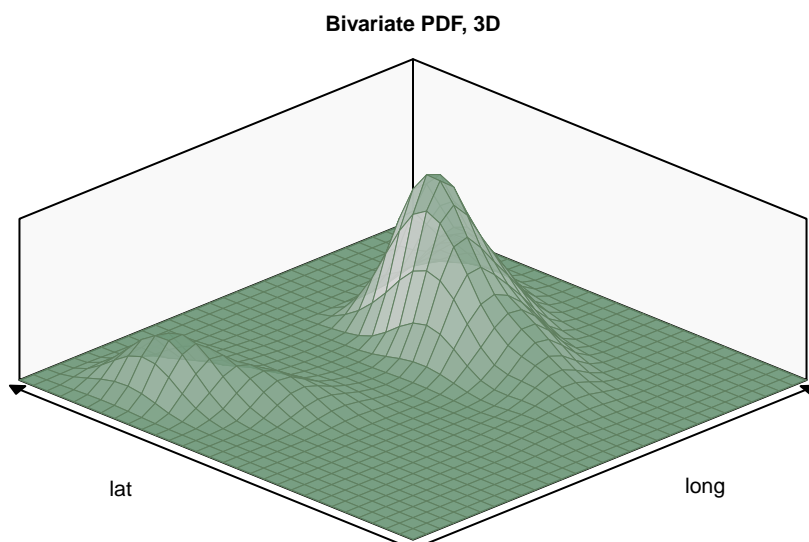
We can construct $\text{PDF}_{(MV)} \sim \text{CKS}$ and $\text{CDF}_{(MV)} \sim \text{CKS}$ objects, using the **pdfmv.cks** and **cdfuv.cks** constructors:

Here are examples, using earthquake data, derived from the “quakes” data, in the datasets package:

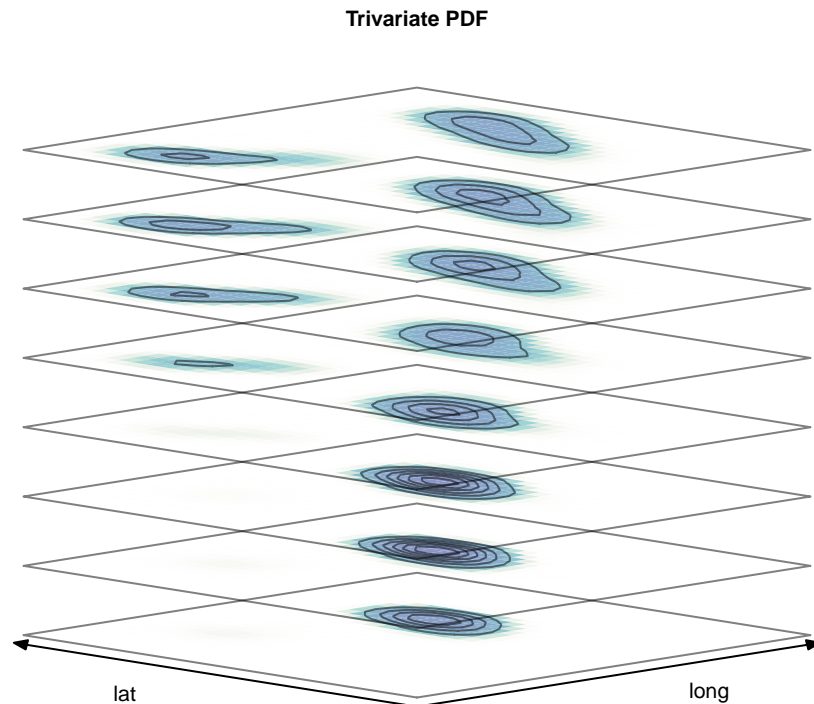
```
> cfh2 <- pdfmv.cks (quakes [,1:2], smoothness = c (0.35, 1) )
> cfh3 <- pdfmv.cks (quakes [,1:3], smoothness = c (0.35, 1, 1) )
> plot (cfh2, TRUE,
        main="Bivariate PDF, 2D")
```



```
> plot (cfh2, TRUE, main="Bivariate PDF, 3D")
```



```
> plot (cfh3, main="Trivariate PDF",  
      nslides=8, zlim = c (680, 40) )
```

The resulting objects are function objects.

The PDF maps a numeric vector or matrix (of quantiles) to a numeric vector (of probabilities) and the CDF maps quantiles to cumulative probabilities.

In PDFs and CDFs, a standard vector is equivalent to a single-row matrix.

```
> cfh2 (c (180, -20) )
[1] 0.007990662
> cfh3 (c (180, -20, 300) )
[1] 2.815987e-06
```

Note that an example of a multivariate CDF is given in the section on multivariate probabilities, later.

Also note that longitude maps to the “x” variable and latitude maps to the “y” variable. (And longitude is the first variable in the derived dataset, but is the second variable in the original dataset).

Continuous Kernel Smoothing: Conditional Probability Distributions

We can construct a $\text{PDF}_{(C)} \sim \text{CKS}$ object, using the `pdfc.cks` constructor.

Likewise, we can construct $\text{CDF}_{(C)} \sim \text{CKS}$ and $\text{QF}_{(C)} \sim \text{CKS}$ objects, using the `cdfc.cks` and `qfc.cks` constructors.

Here are examples, using the quakes data, from the previous section:

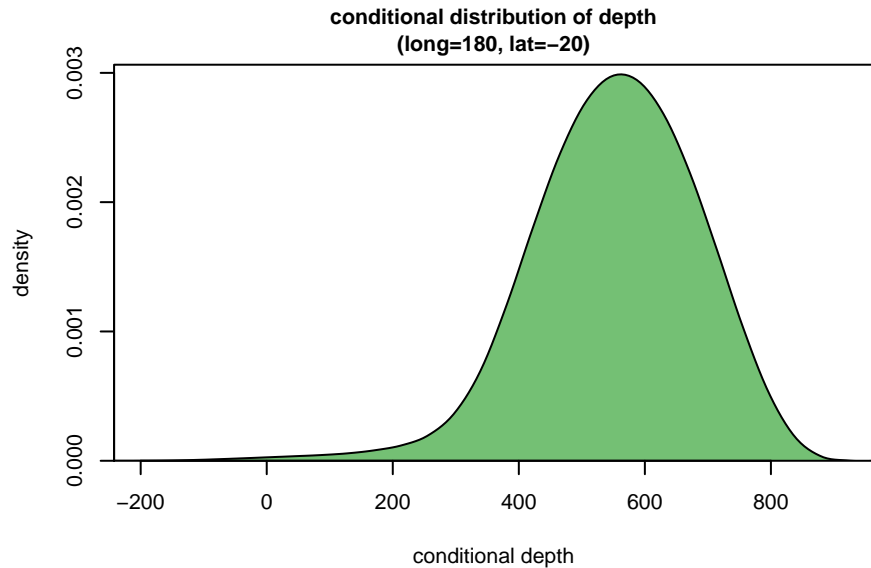
```
> conditions <- c (long=180, lat=-20)
> depth.fhc <- pdfc.cks (quakes [,4], smoothness = c (0.35, 1, 1),
```

```

conditions=conditions, preserve.range=TRUE)
> mag.fhc <- pdfc.cks (quakes [,-3], smoothness = c (0.35, 1, 1),
conditions=conditions, preserve.range=TRUE)

> plot (depth.fhc, main="conditional distribution of depth\n(long=180, lat=-20)")

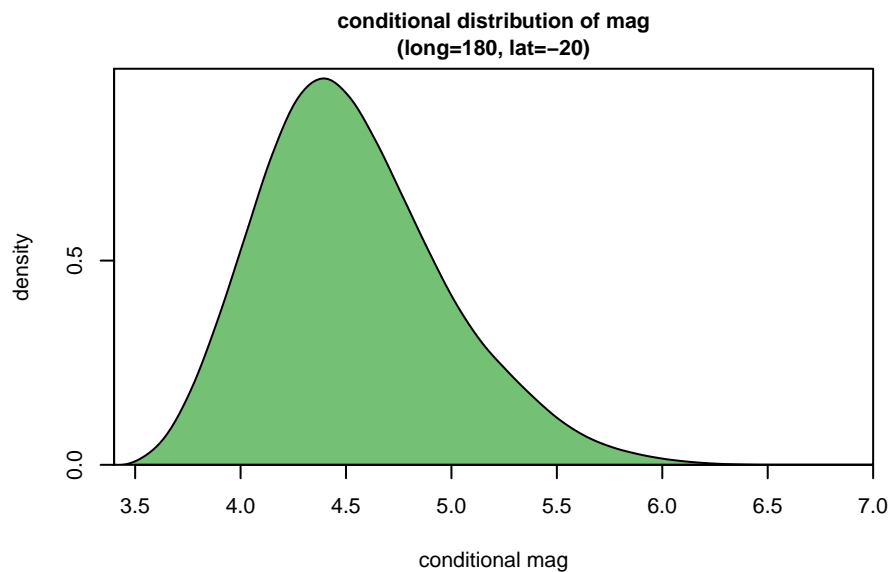
```



```

> plot (mag.fhc, main="conditional distribution of mag\n(long=180, lat=-20)")

```



The resulting objects are function objects, similar to univariate models, given earlier.

Continuous Kernel Smoothing: Multivariate-Conditional Distributions

We can construct $\text{PDF}_{(\text{MVC})\sim\text{CKS}}$ and $\text{CDF}_{(\text{MVC})\sim\text{CKS}}$ objects, using the [pdfmvc.cks](#) and [cdfmvc.cks](#) constructors.

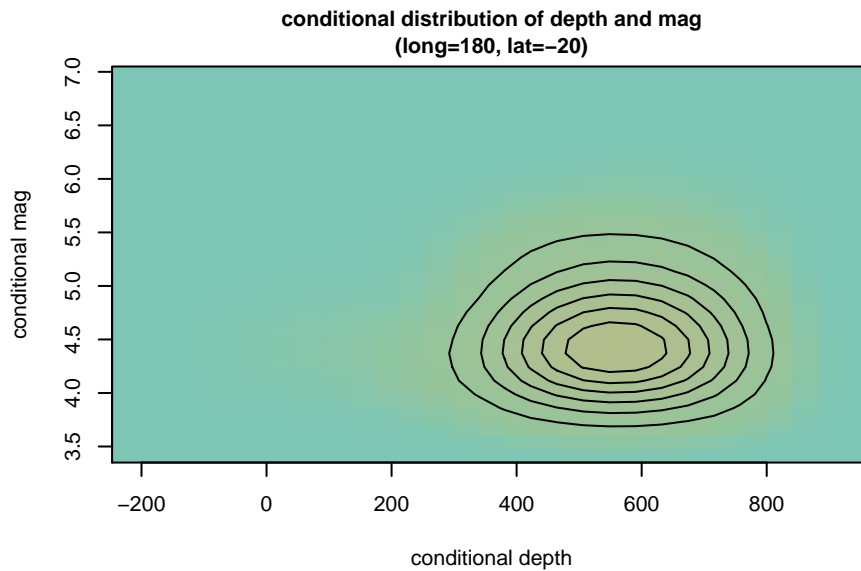
Here are examples, using the quakes data, from the previous sections:

```

> depth.mag.fhc <- pdfmvc.cks (quakes, smoothness = c (0.35, 1, 1, 1),
  conditions = c (long=180, lat=-20), preserve.range=TRUE)
> lat.long.fhc <- pdfmvc.cks (quakes [, -4], smoothness = c (0.35, 1, 1),
  conditions = c (depth=168), preserve.range=TRUE)

> plot (depth.mag.fhc,
  main="conditional distribution of depth and mag\n(long=180, lat=-20)")

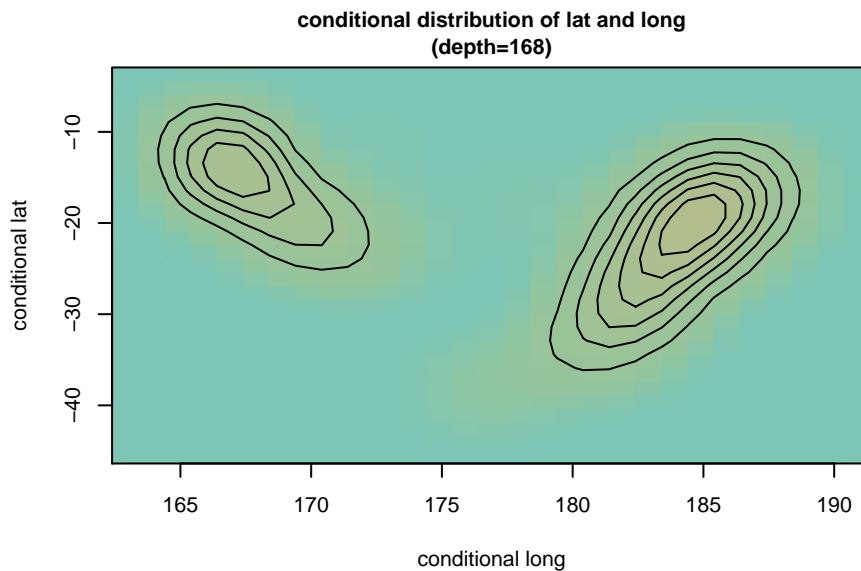
```



```

> plot (lat.long.fhc,
  main="conditional distribution of lat and long\n(depth=168)")

```



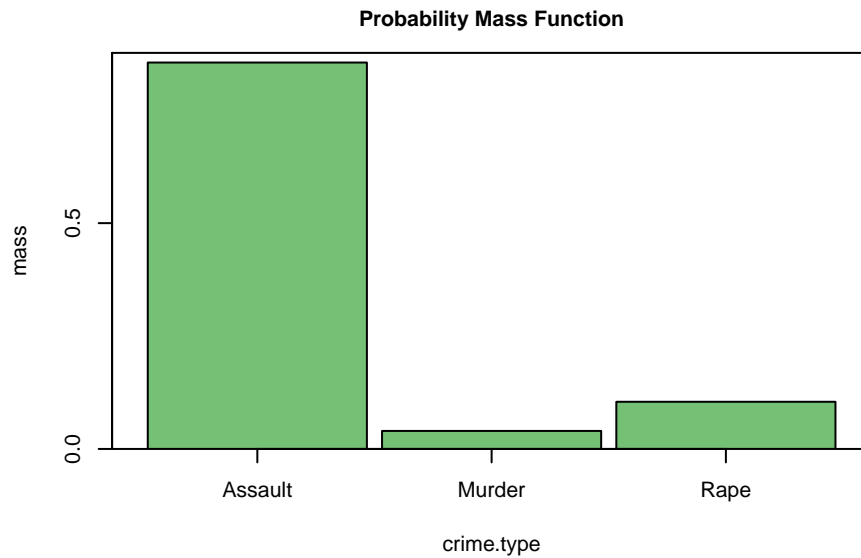
The resulting objects are function objects, similar to multivariate models, given earlier.

Categorical Distributions

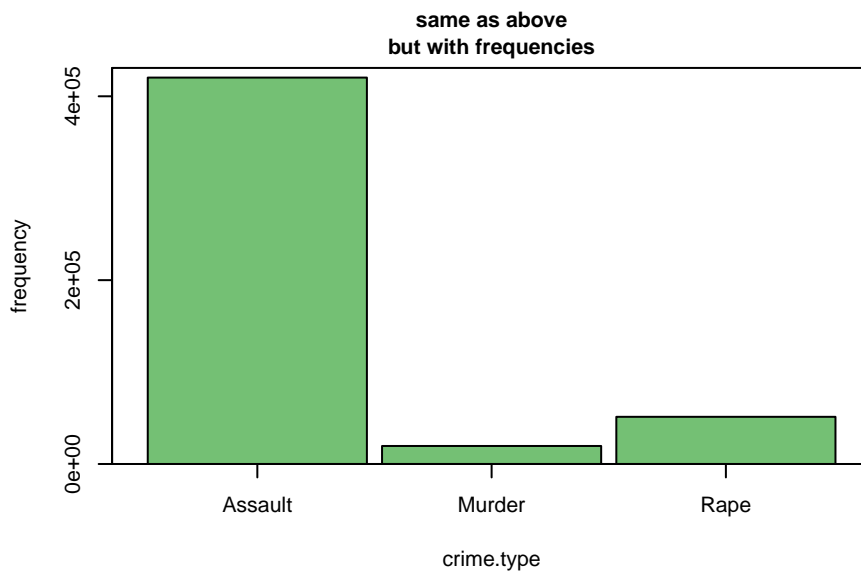
We can construct a $\text{PMF}_{(UV)} \sim \text{CAT}$ object, using the `pmfuv.cat` constructor. Likewise, we can construct $\text{CDF}_{(UV)} \sim \text{CAT}$ and $\text{QF}_{(UV)} \sim \text{CAT}$ objects, using the `cdfuv.cat` and `qfuv.cat` constructors.

Here are examples, using the number arrests per crime type, derived from the state.x77 and USArrests datasets in the datasets package:

```
> gfh <- pmfuv.cat (crime.type, n.arrests)
> plot (gfh, main="Probability Mass Function")
```



```
> plot (gfh, freq=TRUE, main="same as above\nbut with frequencies")
```



The resulting objects are function objects, similar to discrete kernel smoothing. Categorical PMFs and CDFs can be evaluated using integers, factors or characters.

```
> levels (crime.type$crime.type)
[1] "Assault" "Murder" "Rape"
> gfh (1)
[1] 0.8556499
```

```

> gfh ("Assault")
[1] 0.8556499
> gfh ("Assault", freq=TRUE)
[1] 420356

```

Note that the integer above (1) represents the index of the (first) category, and not the corresponding category name (Assault).

This is obvious here, but caution is required if a categorical distribution is constructed from integer-valued categories.

Also, it's possible to construct conditional categorical distributions from categorical data, however, I'm bypassing the examples.

Empirical-Like Distributions

We can construct a $CDF_{(UV)} \sim EL$ object, using the `cdf.el` constructor.

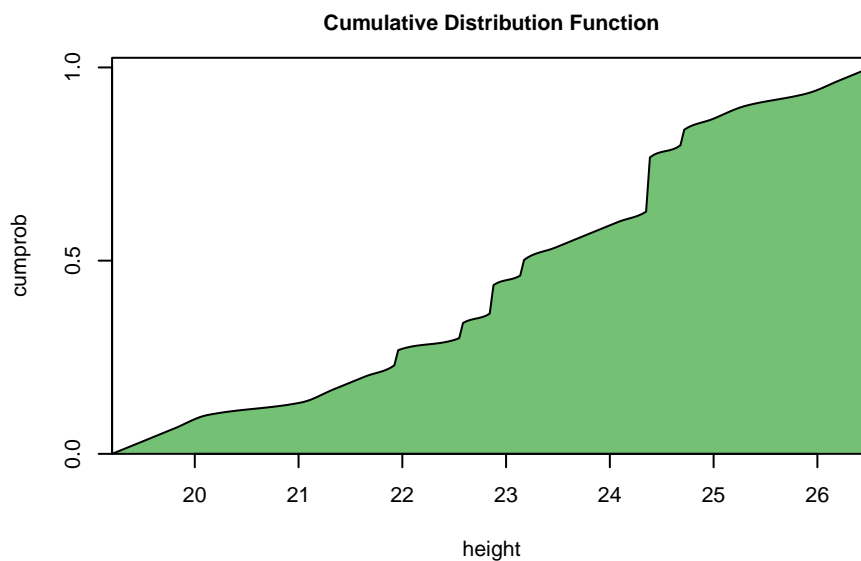
Likewise, we can construct a $QF_{(UV)} \sim EL$ object, using the `qf.el` constructor.

Here's an example, using the same height data, as earlier:

```

> eFh <- cdf.el (height)
> eFht <- qf.el (height)
> plot (eFh, main="Cumulative Distribution Function")

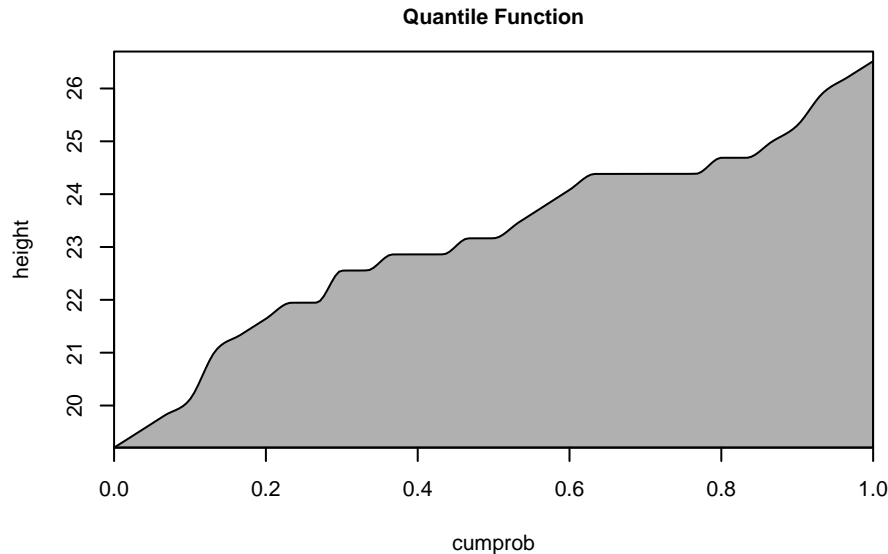
```



```

> plot (eFht, main="Quantile Function")

```



These models compute a set of points, representing cumulative probabilities, and interpolate the points with a cubic Hermite spline.

The resulting functions are smooth (the property), but don't necessarily appear smooth.

Unlike continuous kernel smoothing, empirical-like models don't smooth (the method) the model.

Empirical-like models require unique x values, and a small amount of random variation is automatically added, if they're not unique.

Conditional Distributions with Mixed Input Types (And Statistical Classification)

In addition to the conditional probability distributions listed so far, it's also possible to construct categorical and continuous distributions, conditional on both categorical and continuous variables, or conditional on the opposite type.

Currently, I refer to these as conditional distributions with mixed input types, and give them the extensions gMIXp and xMIXp.

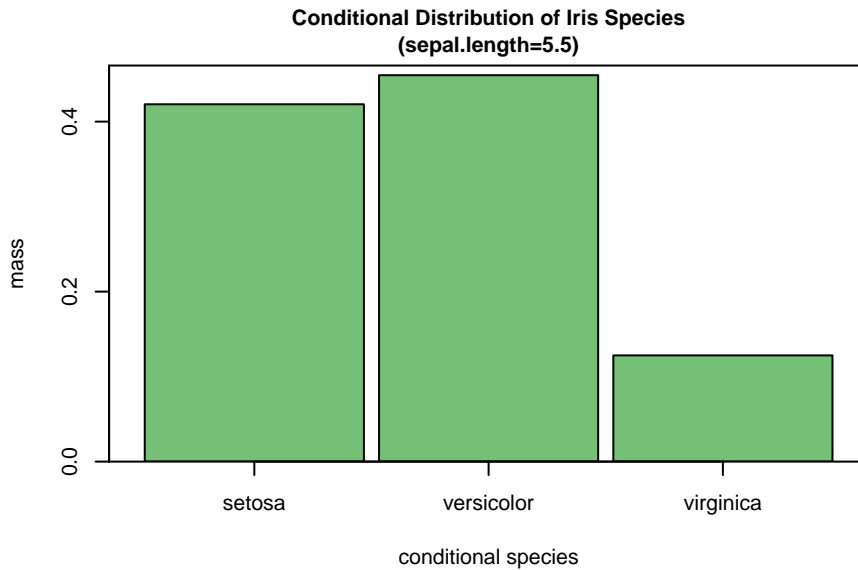
A gMIXp model, fits a categorical distribution (with a categorical conditional variable) and at least one continuous conditioning variable.

A xMIXp model, fits a continuous distribution (with a continuous conditional variable) and at least one categorical conditioning variable.

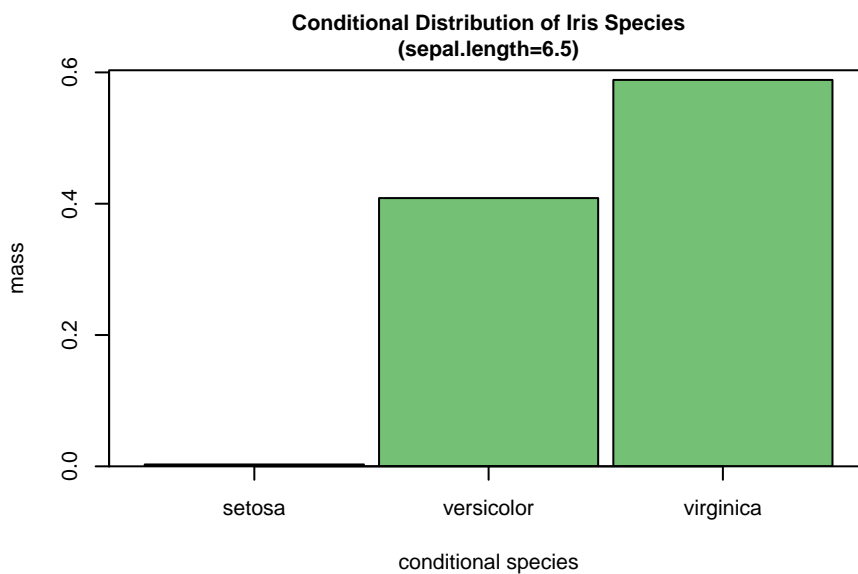
A gMIXp model is computed by fitting one model for each category, along with two non-conditional models, and then uses Bayes' Theorem, to invert the models.

Currently, xMIXp models (and gMIXp models, if there are both categorical and continuous conditions) use a subset of data, where the categorical conditions match the categorical variables, and then fit a model to that subset, however, this may change in the future.

```
> fh1a.gmix <- pmfc.gmixp (species, sepal.length,
  conditions = c (sepal.length=5.5) )
> plot (fh1a.gmix,
  main="Conditional Distribution of Iris Species\n(sepal.length=5.5)")
```

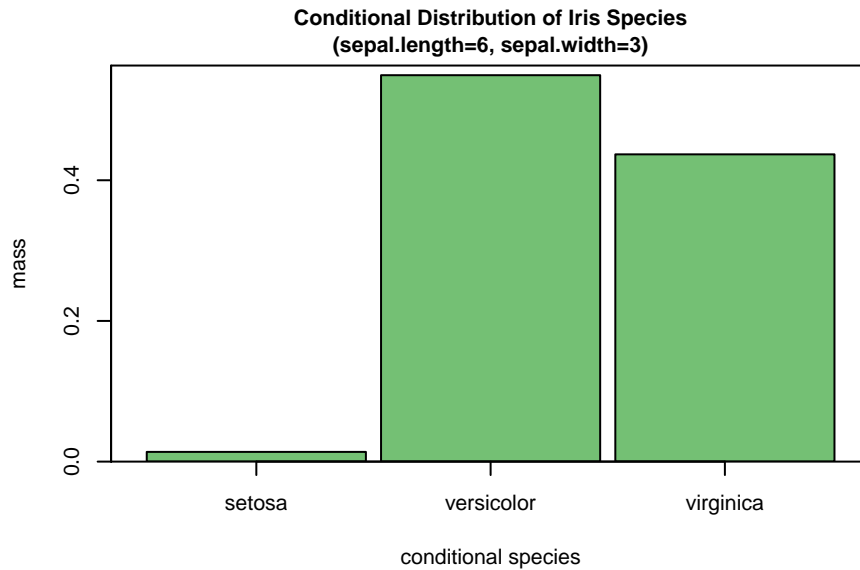


```
> fh1b.gmix <- pmfc.gmixp (species, sepal.length,
  conditions = c (sepal.length=6.5) )
> plot (fh1b.gmix,
  main="Conditional Distribution of Iris Species\n(sepal.length=6.5)")
```



We can use multiple categorical or continuous variables:
(This example uses one conditional categorical variable and two conditioning continuous variables).

```
> fh2.gmix <- pmfc.gmixp (species, cbind (sepal.length, sepal.width),
  conditions = c (sepal.length=6, sepal.width=3) )
> plot (fh2.gmix,
  main = paste (
    "Conditional Distribution of Iris Species",
    "(sepal.length=6, sepal.width=3)",
    sep="\n"
  )
)
```



Note that the next section, on distribution sets, contains a categorical set, which is equivalent to three xMIXp models.

The models above can be used for statistical classification purposes:
(Using the last model):

```
> ph.mode (fh2.gmix)
[1] 2
> gmode (fh2.gmix)
[1] "versicolor"
```

Distribution Sets

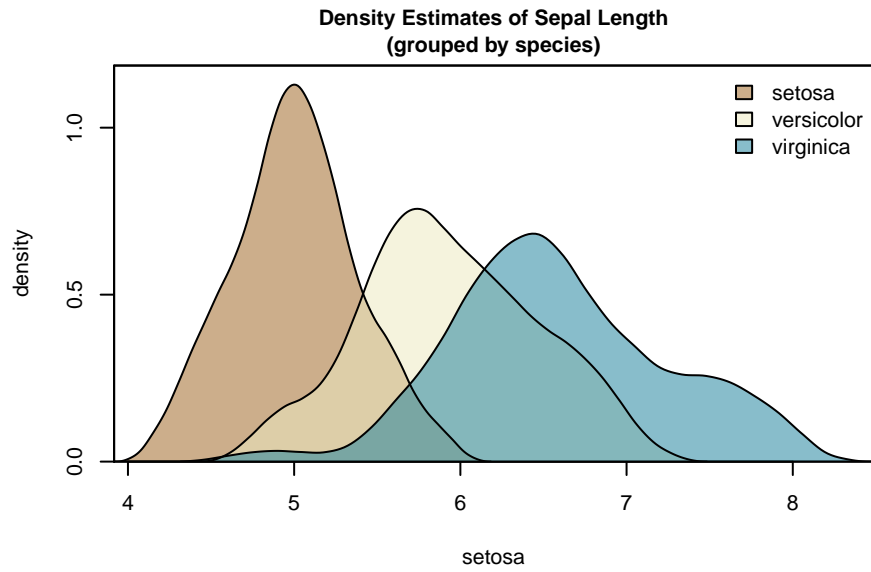
Here, a distribution set is a set of one or more probability distributions.

Currently, there are two types:

- **Categorical Set**
One univariate probability distribution for each (categorical) level, out of many (categorical) levels.
- **Marginal Set**
One univariate probability distribution for each variable, out of many variables.

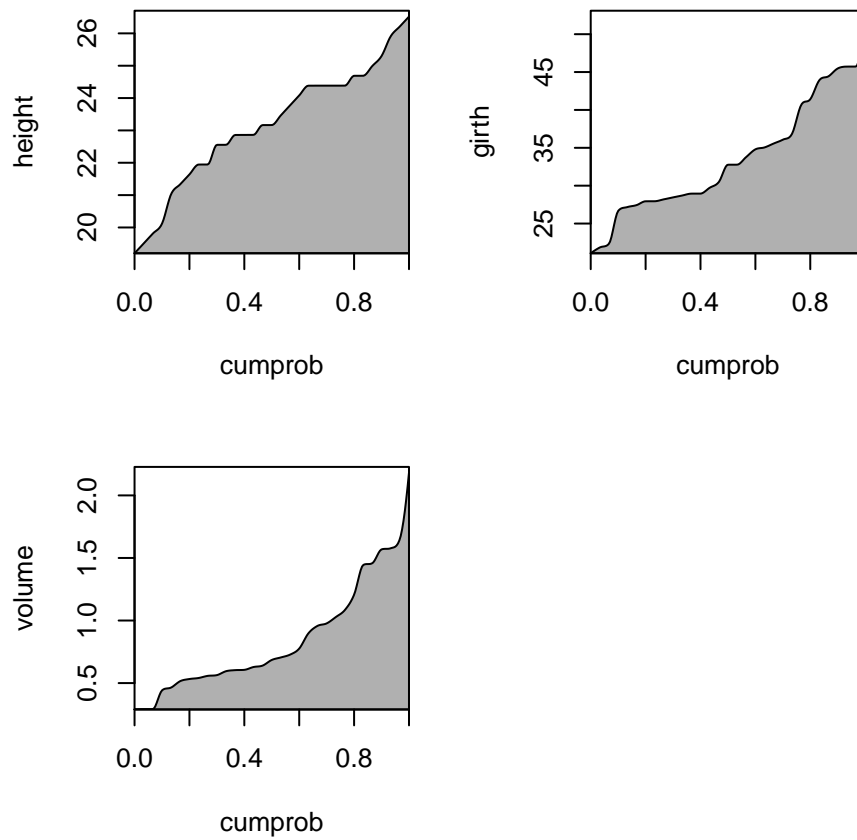
Lets construct $\text{PDF}_{(UV)} \sim \text{CKS}$ models of sepal length, grouped by species:

```
> fh.gset <- pdfuv.gset.cks (sepal.length, group.by=species)
> plot (fh.gset, main="Density Estimates of Sepal Length\n(grouped by species)")
```

Lets construct marginal $QF_{(UV)} \sim EL$ objects:

```
> Fht.mset = qfuv.mset.el (trees)
> plot (Fht.mset, nr=2, nc=2)
```



Multivariate Probabilities

Here, multivariate probability refers to the probability of observing multiple random variables between pairs of lower and upper limits. In theory, such probabilities could be computed from the multivariate PMF or PDF, however (here, at least), it's more efficient to compute them from the multivariate CDF.

Using the trees data, we can compute the probability that height, girth and volume are all between arbitrary pairs of limits.

We can use the **probmV** function, which has three arguments, the multivariate CDF, a vector of lower limits and a vector of upper limits:

```
> #multivariate cdf
> cFh3 <- cdfmv.cks (trees)

> q <- matrix (c (
  22, 24,    #height in 22 to 24
  28, 38,    #girth  in 28 to 38
  0.55, 1.05 #volume in 0.55 to 1.05
),, 2, byrow=TRUE, dimnames = list (colnames (trees), c ("a", "b") ) )
> q

      a      b
height 22.00 24.00
girth  28.00 38.00
volume  0.55  1.05

> #multivariate probability
> probmV (cFh3, q [,1], q [,2])

[1] 0.07481826
```

Note that it's possible to compute multiple regions at once by making a and b matrices with each row representing one region. Also note that currently, variables names are ignored, so they must be in the same order as the variables used to construct the CDF.

Chained Quantile Functions (And Multivariate Random Number Generation)

Standard quantile functions can be used to compute univariate random numbers via standard inversion sampling

I've created novel chained quantile functions, to compute multivariate random numbers.

It works by fitting a quantile function to the first variable's observations, and then evaluating that quantile function at the first variable's evaluation points.

Then, assuming that there are two or more variables, a sequence of conditional quantile functions are fitted to incrementing sets of variables, conditional on the previous variables' evaluations, evaluating one new variable, each time.

This is done for each evaluation point.

The convenience function, **rng**, takes two arguments, the univariate or chained quantile function, and the number of random numbers to generate, then evaluates the quantile function, using a vector or matrix of uniform random numbers.

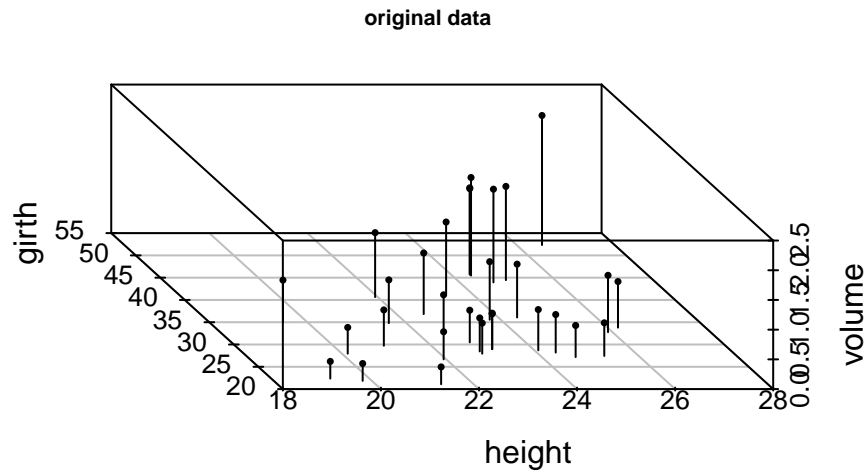
```
> chFht <- chqf.cks (trees)
> synthetic.data <- rng (chFht, 31)
```

```

> #convenience function
> plot.trees.data <- function (x, main)
{
  height <- x [,"height"]
  girth <- x [,"girth"]
  volume <- x [,"volume"]
  scatterplot3d (height, girth, volume,
                main=main, type="h", angle=112.5, pch=16)
}

> #original data
> plot.trees.data (trees, "original data")

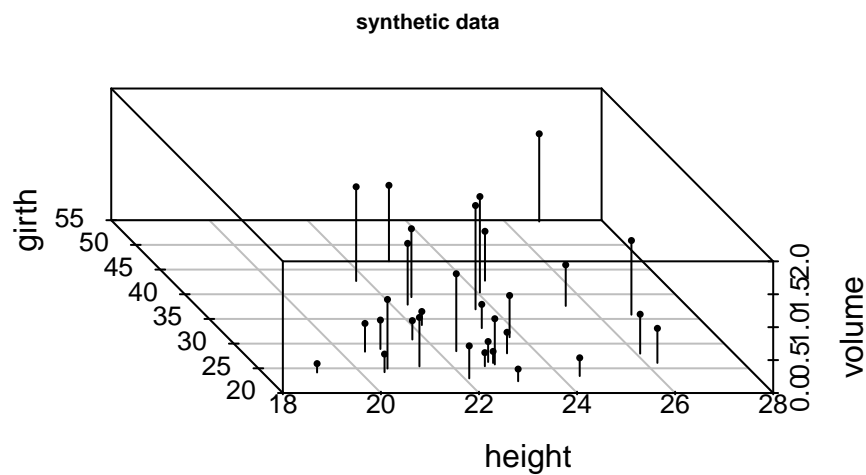
```



```

> #synthetic data
> plot.trees.data (synthetic.data, "synthetic data")

```



Note that this is computationally expensive.

For 3 variables and 31 evaluation points, the algorithm needs to fit:
 $1 + 31 * (3 - 1) = 63$ distributions.

Parameter Estimates

Overview

Univariate probability distributions (including univariate-conditional probability distributions) can be used to compute probabilities and parameter-like estimates, including the mean, standard deviation, variance, skewness and kurtosis, (arbitrary) moments, median, (arbitrary) quantiles and the mode.

Probabilities can be computed from the CDF.

(i.e. In the univariate continuous case, $\hat{F}(b) - \hat{F}(a)$, no example is given).

The mean, standard deviation, variance and higher moments, require the PMF or spline-based CDF.

The median and quantiles, require the quantile function.

The mode, requires the PMF or spline-based PDF.

In the future, I may allow automatic conversion between the PMF/PDF, CDF and QF.

Moment-Based Statistics

We can compute moment-based estimates using the high-level functions the **ph.mean**, **ph.sd**, **ph.var**, **ph.skewness** and **ph.kurtosis**.

They require a PMF or spline-based CDF.

```
> ph.mean (cFh)
```

```
[1] 23.1648
```

```
> ph.var (cFh)
```

```
[1] 4.044737
```

```
> ph.skewness (cFh)
```

```
[1] -0.3213694
```

```
> ph.kurtosis (cFh)
```

```
[1] 2.530339
```

Note that currently, standard deviation, variance and higher moments, should not be regarded as accurate because the smoothing algorithm tends to inflate their values, however, they can still be used as an exploratory tool, especially for the purpose, of comparing different conditional probability distributions.

Order-Based Summary Statistics

We can compute order-based summary statistics, using the **quartiles**, **deciles** and **ntiles** functions.

All of which require a numeric vector, quantile function, or an object than can be coerced to a numeric vector.

```
> quartiles (cFht)
```

```
      min      Q1      (Q2)      Q3      max
17.54124 21.87344 23.37409 24.62905 28.17876
```

```
> deciles (cFht)
```

```

      min      D1      D2      D3      D4      (D5)      D6      D7
17.54124 20.16155 21.43919 22.21897 22.82123 23.37409 23.87581 24.36876
      D8      D9      max
24.90875 25.62579 28.17876

> ntiles (cFht, 8)

      min      q1      q2      q3      (q4)      q5      q6      q7
17.54124 20.51468 21.87344 22.67502 23.37409 23.99723 24.62905 25.41074
      max
28.17876

> ntiles (cFht, 8, rank=FALSE)

      0      0.125      0.25      0.375      0.5      0.625      0.75      0.875
17.54124 20.51468 21.87344 22.67502 23.37409 23.99723 24.62905 25.41074
      1
28.17876

> quartiles (cFht)

      min      Q1      (Q2)      Q3      max
17.54124 21.87344 23.37409 24.62905 28.17876

> quartiles (eFht)

      min      Q1      (Q2)      Q3      max
19.20355 21.94593 23.16532 24.38424 26.51633

> quartiles (height)

      min      Q1      (Q2)      Q3      max
19.20373 21.94663 23.16443 24.38582 26.51950

```

An $QF_{(UV)} \sim EL$ object is automatically created, if a numeric vector is used, hence the last two results should be identical (for unique input values) or almost identical (for non-unique input values).

Robust-Based Statistics

We can compute robust-based summary statistics, using the `ph.median`, `ph.quantile` and `iqr` functions.

All of which require a numeric vector, quantile function, or an object than can be coerced to a numeric vector.

```

> ph.median (cFht)

[1] 23.37409

> ph.quantile (cFht, c (0.25, 0.5, 0.75) )

[1] 21.87344 23.37409 24.62905

> #inter-quartile range
> iqr (cFht)

[1] 2.755612

> #inter-quantile ranges
> iqr (cFht, 2/3)

```

```
[1] 4.046585
```

These function work in similar way to the order-based functions in the previous section, which the exception that they're less summary focused.

Note that calling these functions with a quantile function, is equivalent to evaluating the quantile function, directly.

So, the only gain is possible readability.

```
> cFht (0.5)
```

```
[1] 23.37409
```

```
> ph.median (cFht)
```

```
[1] 23.37409
```

Like the order-based functions, they can also be used on a numeric vector.

```
> ph.median (height)
```

```
[1] 23.16591
```

In theory, this should be more accurate than the median and quantile functions from the stats package, because these use smooth interpolation, rather than linear interpolation.

Mode Estimates

We can compute mode estimates using the **ph.mode** and **ph.modes** functions.

The first requires a PMF or spline-based PDF, and the second which computes multiple modes, requires a spline-based PDF only.

```
> ph.mode(cfH)
```

```
[1] 23.91958
```

```
> ph.mode(cfH, TRUE)
```

```
[1] 0.2049598
```

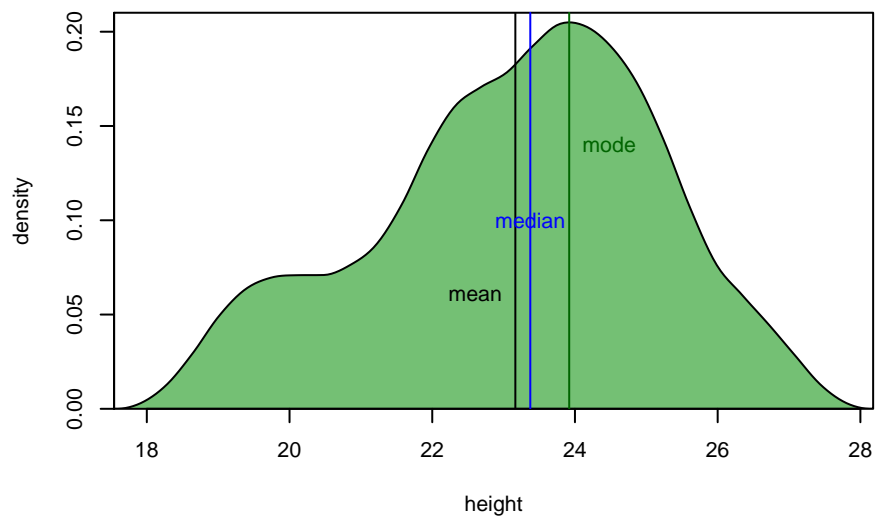
Putting it All Together

```
> height.summary <- c (
  mean = ph.mean (cFh),          #CDF
  sd = ph.sd (cFh),             #CDF
  variance = ph.var (cFh),      #CDF
  skewness = ph.skewness (cFh), #CDF
  kurtosis = ph.kurtosis (cFh), #CDF
  #
  median = ph.median (cFht),     #QF
  mode = ph.mode (cfh) )        #PDF

> str <- c (c ("mean", "median", "mode") )
> x <- height.summary [strs]
> y <- c (0.06, 0.1, 0.14)
> colors <- c ("black", "blue", "darkgreen")
> adjv <- c (1.25, 0.5, -0.25)

> plot (cfh)
> abline (v=x, col=colors)
```

```
> for (i in 1:3)
  text (x [i], y [i], strs [i], adj = adjv [i], col = colors [i])
```



```
> height.summary
```

mean	sd	variance	skewness	kurtosis	median	mode
23.1647993	2.0111531	4.0447368	-0.3213694	2.5303394	23.3740852	23.9195751

References

My main sources are:

[R's stats Package](#)

[Wikipedia](#)

Also, this package uses the following R packages, or has been influenced by them:

intoo: Minimal Language-Like Extensions

Spurdle, A.

barsurf: Heatmap-Related Plots and Smooth Multiband Color Interpolation

Spurdle, A.

bivariate: Bivariate Probability Distributions

Spurdle, A.

mvtnorm: Multivariate Normal and t Distributions

Genz, A., Bretz, F., Miwa, T., Mi, X. & Hothorn, T.

kubik: Cubic Hermite Splines and Related Optimization Methods

Spurdle, A.

KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995)

Wand, M. & Ripley, B.

mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation

Wood, S.

fclust: Fuzzy Clustering

Giordani, P., Ferraro, M.B. & Serafini, A.

scatterplot3d: 3D Scatter Plot

Ligges, U., Maechler, M. & Schnackenberg, S.

MASS: Support Functions and Datasets for Venables and Ripley's MASS

Ripley, B.

Appendix A: List of Probability Distributions

Discrete kernel smoothing models (DKS):

- Univariate probability mass function ($\text{PMF}_{(UV)} \sim \text{DKS}$).
- Univariate cumulative distribution function ($\text{CDF}_{(UV)} \sim \text{DKS}$).
- Univariate quantile function ($\text{QF}_{(UV)} \sim \text{DKS}$).

Continuous kernel smoothing models (CKS):

- Univariate probability density function ($\text{PDF}_{(UV)} \sim \text{CKS}$).
- Univariate cumulative distribution function ($\text{CDF}_{(UV)} \sim \text{CKS}$).
- Univariate quantile function ($\text{QF}_{(UV)} \sim \text{CKS}$).
- Multivariate probability density function ($\text{PDF}_{(MV)} \sim \text{CKS}$).
- Multivariate cumulative distribution function ($\text{CDF}_{(MV)} \sim \text{CKS}$).
- Conditional probability density function ($\text{PDF}_{(C)} \sim \text{CKS}$).
- Conditional cumulative distribution function ($\text{CDF}_{(C)} \sim \text{CKS}$).
- Conditional quantile function ($\text{QF}_{(C)} \sim \text{CKS}$).
- Multivariate-conditional probability density function ($\text{PDF}_{(MVC)} \sim \text{CKS}$).
- Multivariate-conditional cumulative distribution function ($\text{CDF}_{(MVC)} \sim \text{CKS}$).
- Chained quantile function ($\text{ChQF} \sim \text{CKS}$).

Categorical models (CAT):

- Univariate probability mass function ($\text{PMF}_{(UV)} \sim \text{CAT}$).
- Univariate cumulative distribution function ($\text{CDF}_{(UV)} \sim \text{CAT}$).
- Univariate quantile function ($\text{QF}_{(UV)} \sim \text{CAT}$).
- Conditional probability mass function ($\text{PMF}_{(C)} \sim \text{CAT}$).
- Conditional cumulative distribution function ($\text{CDF}_{(C)} \sim \text{CAT}$).
- Conditional quantile function ($\text{QF}_{(C)} \sim \text{CAT}$).

Empirical-like models (EL):

- Univariate cumulative distribution function ($\text{CDF}_{(UV)} \sim \text{EL}$).
- Univariate quantile function ($\text{QF}_{(UV)} \sim \text{EL}$).

Conditional categorical distributions with mixed input types (gMIXp):

- Conditional probability mass function ($\text{PMF}_{(C)} \sim \text{gMIXp}$).
- Conditional cumulative distribution function ($\text{CDF}_{(C)} \sim \text{gMIXp}$).
- Conditional quantile function ($\text{QF}_{(C)} \sim \text{gMIXp}$).

Conditional continuous distributions with mixed input types (xMIXp):

- Conditional probability density function ($\text{PDF}_{(C)} \sim \text{xMIXp}$).
- Conditional cumulative distribution function ($\text{CDF}_{(C)} \sim \text{xMIXp}$).

- Conditional quantile function ($QF_{(C)} \sim xMIXp$).

Distribution sets:

- Marginal sets.
- Categorical sets.

Appendix B: Notation and Terminology Notes

When used as a prefix, the letters “d” and “c” mean discrete and continuous, respectively. Likewise, DPD and CPD mean discrete probability distribution and continuous probability distribution.

PMF, PDF, CDF and QF refer to probability mass function (or just mass function), probability density function (or just density function), cumulative distribution function (or just distribution function) and quantile function, respectively.

Note that I sometimes use the notation, `Fht` or `Fh.inv`, to describe the quantile function. However, quantile functions are not necessarily the exact inverse of CDFs. Firstly, CDFs may have level (non-increasing) sections which are non-invertible, and secondly, the algorithm for constructing quantile splines, only transposes the control points, it does not compute an exact inverse.

When used as a suffix, the letters “uv”, “mv”, “c” and “mvc”, mean univariate, multivariate, conditional and multivariate-conditional.

In publicly visible constructors (e.g. `pmfuv.dks`):

Univariate and multivariate distributions, refer to non-conditional univariate and multivariate distributions only, and conditional distributions refer to univariate conditional distributions only.

This also applies to the corresponding class names, and to most of the section headings in this vignette.

In other contexts, univariate and multivariate distributions, can be conditional or non-conditional, and conditional distributions can be univariate or multivariate, unless stated otherwise.

Currently, conditional distributions need at least one condition.
(i.e. A model with no conditions is not regarded as conditional).

In this package, a “bounded” distribution is a fitted distribution with constraints on the range of the random variable, where the smoothing is truncated at the limits.

In the context of plotting functions, limits refer to the min/max values of axes.
In other contexts, limits refer the (allowable) min/max values of random variables.
When evaluating CDFs, such limits become the corresponding limits of integration.

Also, they define the default values for sequence and plotting functions.

Appendix C: Multivariate Probabilities

We can compute the probability that a single (continuous) random variable is between a pair of values as:

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

Where a is the lower limit and b is the upper limit.

This is the area under a univariate PDF.

Likewise, we can compute the probability that two (continuous) random variables are between two pairs of values as:

$$\begin{aligned} \mathbb{P}(a_1 \leq X_1 \leq b_1, \quad a_2 \leq X_2 \leq b_2) &= \sum P^{\binom{2}{2}} - \sum P^{\binom{2}{1}} + \sum P^{\binom{2}{0}} \\ &= F_{(X_1, X_2)}(b_1, b_2) \\ &\quad - [F_{(X_1, X_2)}(a_1, b_2) + F_{(X_1, X_2)}(b_1, a_2)] \\ &\quad + F_{(X_1, X_2)}(a_1, a_2) \end{aligned}$$

Where $\sum P^{\binom{m}{k}}$ is shorthand for the sum of the m -variate CDF evaluated with each possible combination of k b -terms and $(m - k)$ a -terms.

And where a is a vector of lower limits and b is a vector of upper limits.

This is the volume under the bivariate PDF.

For three and four variables we have:

$$\begin{aligned} \mathbb{P}(a_1 \leq X_1 \leq b_1, \quad a_2 \leq X_2 \leq b_2, \quad a_3 \leq X_3 \leq b_3) \\ &= \sum P^{\binom{3}{3}} - \sum P^{\binom{3}{2}} + \sum P^{\binom{3}{1}} - \sum P^{\binom{3}{0}} \\ &= F_{(X_1, X_2, X_3)}(b_1, b_2, b_3) \\ &\quad - [F_{(X_1, X_2, X_3)}(a_1, b_2, b_3) + F_{(X_1, X_2, X_3)}(b_1, a_2, b_3) + F_{(X_1, X_2, X_3)}(b_1, b_2, a_3)] \\ &\quad + [F_{(X_1, X_2, X_3)}(a_1, a_2, b_3) + F_{(X_1, X_2, X_3)}(a_1, b_2, a_3) + F_{(X_1, X_2, X_3)}(b_1, a_2, a_3)] \\ &\quad - F_{(X_1, X_2, X_3)}(a_1, a_2, a_3) \end{aligned}$$

$$\begin{aligned} \mathbb{P}(a_1 \leq X_1 \leq b_1, \quad a_2 \leq X_2 \leq b_2, \quad a_3 \leq X_3 \leq b_3, \quad a_4 \leq X_4 \leq b_4) \\ &= \sum P^{\binom{4}{4}} - \sum P^{\binom{4}{3}} + \sum P^{\binom{4}{2}} - \sum P^{\binom{4}{1}} + \sum P^{\binom{4}{0}} \end{aligned}$$

More generally (given a continuous multivariate CDF, $F_{(X_1, X_2, \dots, X_m)}$, for m random variables), we have:

$$\mathbb{P}(a_1 \leq X_1 \leq b_1, \quad a_2 \leq X_2 \leq b_2, \quad \dots, \quad a_m \leq X_m \leq b_m) = \sum_{k \in [0, m]} \left((-1)^{m-k} \sum P^{\binom{m}{k}} \right)$$

Appendix D: Conditional Formulae

We can compute univariate-conditional (continuous) distributions, with one random variable conditional on one other variable, using:

$$\begin{aligned} f_Y(y) &= f_{(Y|X=x)}(y) \\ &= \frac{f_{(X_1, X_2)}(x_1 = x, x_2 = y)}{f_{X_1}(x_1 = x)} \\ \\ F_Y(y) &= F_{(Y|X=x)}(y) \\ &= \int_{-\infty}^y \frac{f_{(X_1, X_2)}(x_1 = x, x_2 = u)}{f_{X_1}(x_1 = x)} du \end{aligned}$$

We can compute univariate-conditional (continuous) distributions, with one random variable conditional on multiple other variables, using:

$$\begin{aligned} f_Y(y) &= f_{(Y|X_1=x_1, X_2=x_2, \dots, X_{[m-1]}=x_{[m-1]})}(y) \\ &= \frac{f_{(X_1, X_2, \dots, X_m)}(\$X, \$Y_{uv})}{f_{(X_1, X_2, \dots, X_{[ncon]})}(\$X)} \\ &= \frac{f_{(X_1, X_2, \dots, X_m)}(x_1 = x_1, x_2 = x_2, \dots, x_{[ncon]} = x_{[ncon]}, x_m = y)}{f_{(X_1, X_2, \dots, X_{[ncon]})}(x_1 = x_1, x_2 = x_2, \dots, x_{[ncon]} = x_{[ncon]})} \\ &= \frac{f_{(X_1, X_2, \dots, X_m)}(x_1 = x_1, x_2 = x_2, \dots, x_{[m-1]} = x_{[m-1]}, x_m = y)}{f_{(X_1, X_2, \dots, X_{[m-1]})}(x_1 = x_1, x_2 = x_2, \dots, x_{[m-1]} = x_{[m-1]})} \\ \\ F_Y(y) &= F_{(Y|X_1=x_1, X_2=x_2, \dots, X_{[m-1]}=x_{[m-1]})}(y) \\ &= \int_{-\infty}^y \frac{f_{(X_1, X_2, \dots, X_m)}(\$X, \$U_{uv})}{f_{(X_1, X_2, \dots, X_{[ncon]})}(\$X)} du \\ &= \int_{-\infty}^y \frac{f_{(X_1, X_2, \dots, X_m)}(x_1 = x_1, x_2 = x_2, \dots, x_{[ncon]} = x_{[ncon]}, x_m = u)}{f_{(X_1, X_2, \dots, X_{[ncon]})}(x_1 = x_1, x_2 = x_2, \dots, x_{[ncon]} = x_{[ncon]})} du \\ &= \int_{-\infty}^y \frac{f_{(X_1, X_2, \dots, X_m)}(x_1 = x_1, x_2 = x_2, \dots, x_{[m-1]} = x_{[m-1]}, x_m = u)}{f_{(X_1, X_2, \dots, X_{[m-1]})}(x_1 = x_1, x_2 = x_2, \dots, x_{[m-1]} = x_{[m-1]})} du \end{aligned}$$

Note that the convention in this package, is that conditioning variables are enumerated first, and the conditional variables last.

In the univariate-conditional case, ncon (the number of conditions) is equal to m (the total number of variables) minus one.

This can be further generalized to compute multivariate-conditional (continuous) distributions, with M random variables conditional on multiple other variables, using:

$$\begin{aligned} f_{Y_1, Y_2, \dots, Y_M}(y_1, y_2, \dots, y_M) &= f_{(Y_1, Y_2, \dots, Y_M|X_1=x_1, X_2=x_2, \dots, X_{[ncon]}=x_{[ncon]})}(y_1, y_2, \dots, y_M) \\ &= \frac{f_{(X_1, X_2, \dots, X_m)}(\$X, \$Y_{mv})}{f_{(X_1, X_2, \dots, X_{[ncon]})}(\$X)} \\ \\ F_{Y_1, Y_2, \dots, Y_M}(y_1, y_2, \dots, y_M) &= F_{(Y_1, Y_2, \dots, Y_M|X_1=x_1, X_2=x_2, \dots, X_{[ncon]}=x_{[ncon]})}(y_1, y_2, \dots, y_M) \\ &= \int_{-\infty}^{y_1} \int_{-\infty}^{y_2} \dots \int_{-\infty}^{y_M} \frac{f_{(X_1, X_2, \dots, X_m)}(\$X, \$U_{mv})}{f_{(X_1, X_2, \dots, X_{[ncon]})}(\$X)} du_M, \dots, du_2, du_1 \end{aligned}$$

Where the subexpressions expand as follows:

$$\$X : \{x_1 = x_1, x_2 = x_2, \dots, x_{[\text{ncon}]} = x_{[\text{ncon}]}\}$$

$$\$Y_{\text{mv}} : \{x_{[\text{ncon}+1]} = y_1, x_{[\text{ncon}+2]} = y_2, \dots, x_{[\text{ncon}+M]} = y_M\}$$

$$\$U_{\text{mv}} : \{x_{[\text{ncon}+1]} = u_1, x_{[\text{ncon}+2]} = u_2, \dots, x_{[\text{ncon}+M]} = u_M\}$$

These formulae do not use all the data. A conditional window is computed, and observations outside the window are discarded. There needs to be at least one observation within the conditional window, otherwise, the denominator is undefined.

It's not necessary to compute all of the expression, in each evaluation of the PDF or CDF. Rather, the denominator (which is a multivariate PDF) and the first part of the numerator (given later), can be computed when the object is constructed.

And it's not necessary to integrate the expression, as such.

The algorithms for computing the multivariate PDFs and CDFs via kernel smoothing (also, given later), can be combined.

Appendix E: Discrete Kernel Smoothing Formulae

Note that discrete kernels (but not a discrete version of kernel smoothing) may be removed in the future.

Discrete kernels (without standardized intervals), take the form:

$$\begin{aligned} k(x; \text{bw}) &= \dots \\ K(x; \text{bw}) &= \dots \end{aligned}$$

$$\text{hbw} = \frac{\text{bw} - 1}{2}$$

Where k and K are the kernel's PMF and CDF, respectively.
And where bw is the (odd positive) bandwidth parameter.

Unstandardized discrete kernels have zero mass outside the interval $[-\text{hbw}, +\text{hbw}]$.

We can define additive component-distributions as:

$$\begin{aligned} k^*(x; k, x_i^*, \text{bw}) &= k(x - x_i^*; \text{bw}) \\ K^*(x; K, x_i^*, \text{bw}) &= K(x - x_i^*; \text{bw}) \end{aligned}$$

Where x_i^* is the (integer-valued) center of the of each additive component-distribution, and x is the (integer-valued) point on the x-axis, where the function is evaluated.

Unbounded PMFs and CDFs can be computed, as follows:

$$\begin{aligned} \hat{f}_X(x; k, \text{bw}, n, \mathbf{x}^*, \mathbf{w}) &= \sum_i w_i k^*(x; k, x_i^*, \text{bw}) \\ \hat{F}_X(x; K, \text{bw}, n, \mathbf{x}^*, \mathbf{w}) &= \sum_i w_i K^*(x; K, x_i^*, \text{bw}) \end{aligned}$$

Where:

$$w_i = \frac{h_i}{\sum_i h_i}$$

And where \mathbf{x}^* is a vector of (integer-valued) bins and \mathbf{h} is a vector of frequencies, both of which, are of length n , and $i \in [1, n]$.

In general, frequencies are integer-valued, however, there's no requirement for this.

Bounded PMFs and CDFs can be computed by modifying the expressions above, such that mass estimates are truncated at the lower or upper boundaries. The x and h values are reflected about the boundaries prior to smoothing, otherwise, estimates near the boundaries tend to be too small.

Appendix F: Continuous Kernel Smoothing Formulae

Continuous kernels (with standardized intervals), take the form:

$$\begin{aligned} k(x) &= \dots \\ K(x) &= \dots \end{aligned}$$

Where k and K are the kernel's PDF and CDF, respectively.

Standardized continuous kernels have zero density outside the interval $[-1, 1]$.

We can define additive component-distributions using:

$$\begin{aligned} k^*(x; k, x_i^*, \text{bw}) &= \frac{2}{\text{bw}} k\left(\frac{2}{\text{bw}}(x - x_i^*)\right) \\ K^*(x; K, x_i^*, \text{bw}) &= K\left(\frac{2}{\text{bw}}(x - x_i^*)\right) \end{aligned}$$

Where bw is the bandwidth, x_i^* is the center of each additive component-distribution, and x is a point on the x-axis, where the function is evaluated.

Univariate PDFs and CDFs can be computed, as follows:

$$\begin{aligned} \hat{f}_X(x; k, \text{bw}, n, \mathbf{x}^*) &= \frac{\sum_i k^*(x; k, \text{bw}, x_i^*)}{n} \\ \hat{F}_X(x; K, \text{bw}, n, \mathbf{x}^*) &= \frac{\sum_i K^*(x; K, \text{bw}, x_i^*)}{n} \end{aligned}$$

Where \mathbf{x}^* is a vector of length n , and $i \in [1, n]$.

Multivariate PDFs and CDFs can be computed, as follows:

$$\begin{aligned} \hat{f}_{\mathbf{X}}(\mathbf{x}; k, \mathbf{bw}, n, m, \mathbf{x}^*) &= \frac{\sum_i (\$f_1 \times \$f_2 \times \dots \times \$f_m)}{n} \\ &= \frac{\sum_i \left(k^*(x_1; k, \text{bw}_1, x_{[i,1]}^*) \times k^*(x_2; k, \text{bw}_2, x_{[i,2]}^*) \times \dots \times k^*(x_m; k, \text{bw}_m, x_{[i,m]}^*) \right)}{n} \\ \hat{F}_{\mathbf{X}}(\mathbf{x}; K, \mathbf{bw}, n, m, \mathbf{x}^*) &= \frac{\sum_i (\$F_1 \times \$F_2 \times \dots \times \$F_m)}{n} \\ &= \frac{\sum_i \left(K^*(x_1; K, \text{bw}_1, x_{[i,1]}^*) \times K^*(x_2; K, \text{bw}_2, x_{[i,2]}^*) \times \dots \times K^*(x_m; K, \text{bw}_m, x_{[i,m]}^*) \right)}{n} \end{aligned}$$

Where \mathbf{bw} is a bandwidth vector, \mathbf{x}^* is a matrix with n rows (observations) and m columns (variables), and \mathbf{x} is a vector of points on the x-plane, where the function is evaluated.

Weighted versions of these formulae are created by substituting:

$$\frac{\sum_i (\$SUB-EXPRESSION)}{n}$$

With:

$$\sum_i w_i (\$SUB-EXPRESSION)$$

Subject to:

$$\sum_i w_i = 1$$

Appendix G: Empirical-Like Formulae

An empirical cumulative distribution function, which is a step function, can be computed by:

$$\mathbb{P}(X \leq x) = \hat{F}_X(x; n, \mathbf{x}^*) = \frac{\sum_i I(x_i^* \leq x)}{n}$$

Where I is an indicator function, which equals 1, if the enclosed logical expression is true, and equals 0, if false.

A proto-empirical-like distribution, which is also a step function, can be computed by modifying the formula above, to give:

$$\mathbb{P}(X \leq x) = \hat{G}_X(x; n, \mathbf{x}^*) = \frac{(\sum_i I(x_i^* \leq x)) - 1}{n - 1}$$

This function can be used to generate a sequence of points:

$$\{(x_1^*, \hat{G}_X(x_1^*; n, \mathbf{x}^*)), (x_2^*, \hat{G}_X(x_2^*; n, \mathbf{x}^*)), \dots, (x_n^*, \hat{G}_X(x_n^*; n, \mathbf{x}^*))\}$$

An empirical-like distribution, which is a continuous function, can be computed by using a cubic Hermite spline to interpolate this sequence.

Appendix H: Fuzzy Clustering (And Weighted Multivariate Kernel Smoothing)

Fuzzy clustering computes a membership matrix, from some data.

The values in the membership matrix represent the membership of each data point in each cluster, with each row representing one data point and each column representing one cluster.

(Note that rows, not columns, sum to one).

In some situations, it may be of interest to identify the clusters, only. In other situations, it may be of interest to identify the clusters, and model the properties of one or more of those clusters.

It's possible to model each cluster using weighted kernel smoothing.

The following computes the membership matrix for three clusters:

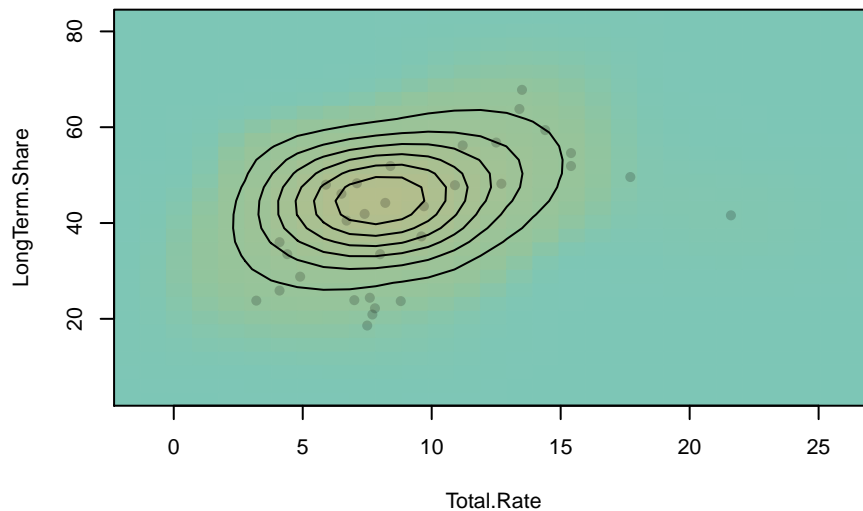
```
> membership <- FKM.gk (unemployment, k=3, seed=2)$U
```

I'm going to extract the weights of the first cluster, and transform them, so that they sum to one:

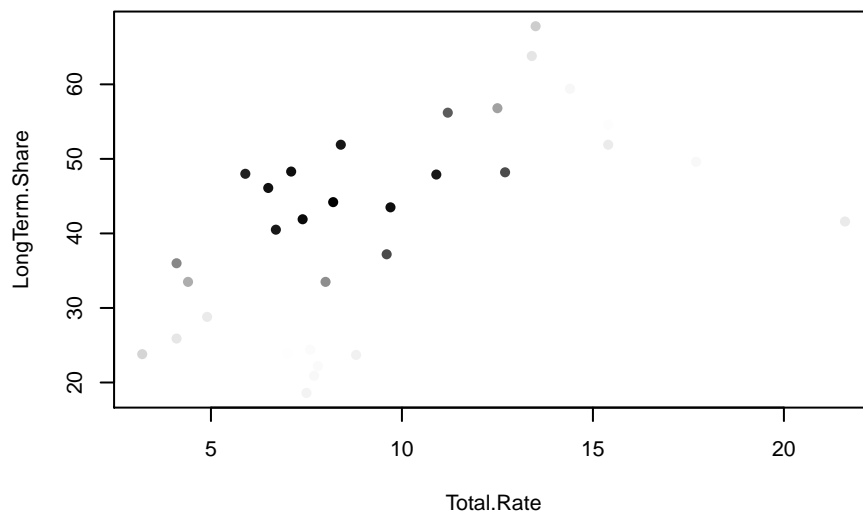
```
> w <- membership [,1]
> w <- w / sum (w)
```

And a weighted model:

```
> wfh.1 <- pdfmv.cks (unemployment, w=w)
> plot (wfh.1, , TRUE)
```



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

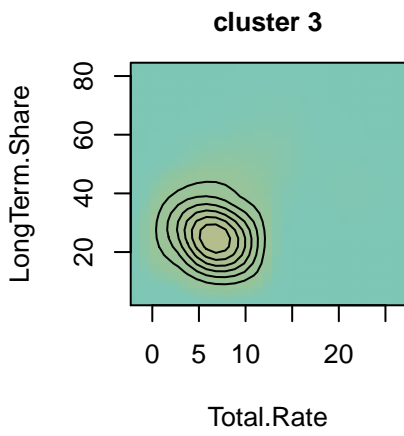
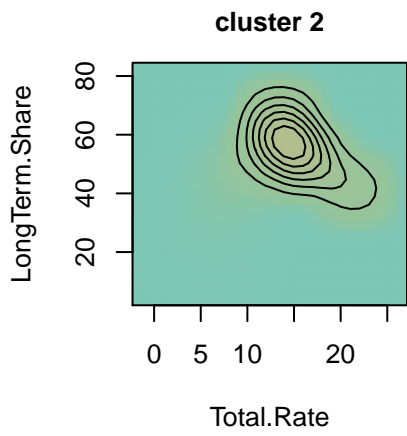
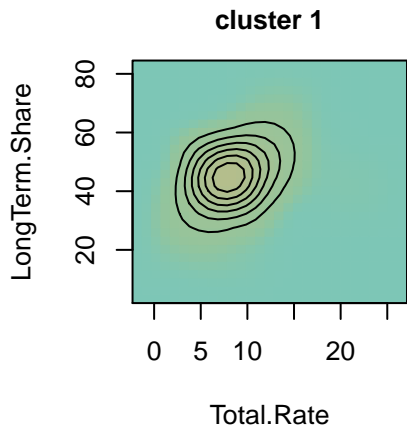


And for the other two clusters:

```
> w <- membership [,2]
> wfh.2 = pdfmv.cks (unemployment, w = w / sum (w) )
> w <- membership [,3]
> wfh.3 = pdfmv.cks (unemployment, w = w / sum (w) )
```

All three:

```
> plot (wfh.1, main="cluster 1")
> plot (wfh.2, main="cluster 2")
> plot (wfh.3, main="cluster 3")
```



Appendix I: Data Preparation

```
> ph.data.prep (eval=FALSE, echo=TRUE)

data (Traffic, package="MASS")
traffic.table <- table (Traffic$y [Traffic$limit=="yes"])
traffic.bins <- as.integer (names (traffic.table) )
traffic.bins <- cbind (naccidents=traffic.bins)
traffic.freq <- as.vector (traffic.table)

trees <- as.matrix (datasets::trees)[,c (2, 1, 3)]
colnames (trees) <- tolower (colnames (trees) )
#Height (-> m)
trees [,"height"] <- 0.3048 * trees [,"height"]
#Girth (-> cm)
trees [,"girth"] <- 2.54 * trees [,"girth"]
#Volume (-> m ^ 3)
trees [,"volume"] <- 0.0283168 * trees [,"volume"]

height <- trees [,"height", drop=FALSE]

quakes <- as.matrix (datasets::quakes)[,c (2, 1, 3:4)]

crimes <- cbind (state.x77 [,1, drop=FALSE] * 1e3, as.matrix (USArrests [, -3]) / 1e5)

crime.type <- as.factor (rep (colnames (crimes)[-1], each=50) )
crime.type <- list (crime.type=crime.type)
n.arrests <- as.vector (round (crimes [,1] * crimes [, -1]) )

species <- list (species=iris$Species)
sepal.length <- cbind (sepal.length = iris$Sepal.Length)
sepal.width <- cbind (sepal.width = iris$Sepal.Width)

data (unemployment, package="fclust")
unemployment <- as.matrix (unemployment)[,-2]
```

Appendix J: Datasets

```
> headt (cbind (traffic.bins, freq=traffic.freq) )
```

```
      naccidents freq
[1,]    7         1
[2,]    9         5
[3,]   10         1
     ##         ##
[22,]  32         1
[23,]  41         2
[24,]  42         1
```

```
> headt (trees)
```

```
      height    girth    volume
[1,] 21.3360000 21.0820000 0.2916630
[2,] 19.8120000 21.8440000 0.2916630
[3,] 19.2024000 22.3520000 0.2888314
     #####
[29,] 24.3840000 45.7200000 1.4583152
[30,] 24.3840000 45.7200000 1.4441568
[31,] 26.5176000 52.3240000 2.1803936
```

```
> headt (quakes)
```

```
      long  lat  depth mag
[1,] 181.62 -20.42 562.00 4.80
[2,] 181.03 -20.62 650.00 4.20
[3,] 184.10 -26.00 42.00 5.40
     #####
[998,] 184.20 -20.13 244.00 4.50
[999,] 187.80 -17.40 40.00 4.50
[1000,] 170.56 -21.59 165.00 6.00
```

```
> headt (crimes)
```

```
      Population Murder  Assault  Rape
Alabama  3.615e+06 1.320e-04 2.360e-03 2.120e-04
Alaska   3.650e+05 1.000e-04 2.630e-03 4.450e-04
Arizona  2.212e+06 8.100e-05 2.940e-03 3.100e-04
     #####
West Virginia 1.799e+06 5.700e-05 8.100e-04 9.300e-05
Wisconsin  4.589e+06 2.600e-05 5.300e-04 1.080e-04
Wyoming   3.760e+05 6.800e-05 1.610e-03 1.560e-04
```

```
> headt (data.frame (crime.type=crime.type$crime.type, n.arrests) )
```

```
      crime.type n.arrests
1  Murder      477
2  Murder      36
3  Murder     179
     #####
148 Rape      167
149 Rape     496
150 Rape      59
```

```
> headt (data.frame (species, sepal.length, sepal.width) )
```

```
   species  sepal.length sepal.width
1  setosa    5.1         3.5
2  setosa    4.9         3.0
3  setosa    4.7         3.2
#####
148 virginica 6.5         3.0
149 virginica 6.2         3.4
150 virginica 5.9         3.0
```

```
> headt (unemployment)
```

```
          Total.Rate LongTerm.Share
BELGIUM          7.1         48.3
BULGARIA         11.2         56.2
CZECHREPUBLIC    6.7         40.5
#####
SWITZERLAND       4.1         36.0
CROATIA           13.4        63.8
TURKEY            8.8         23.7
```