empirical 0.1.0

Empirical Probability Density Functions and Empirical Cumulative Distribution Functions

Abby Spurdle

September 11, 2018

Implements empirical probability density functions (continuous functions) and empirical cumulative distribution functions (step functions or continuous). Currently, univariate only.

Introduction

This package implements what I refer to as empirical probability distributions (empirical probability density functions and empirical cumulative distribution functions).

Empirical probability density functions (EPDFs) are continuous functions, interpolated by a cubic hermite spline.

Empirical cumulative distributions functions (ECDFs) are either step functions or continuous functions, interpolated by a cubic hermite spline.

Note that continuous functions are smooth, in that they're continuous and have a continuous first derivative. However, they don't necessarily appear smooth.

I'm planning to add multivariate and conditional probability distributions in the near future.

Loading The empirical Package

First we need to load the empirical package.

```
> library (empirical)
```

Empirical Probability Density Functions

We can compute an EPDF by computing a continuous ECDF and then computing difference quotients from finite differences, subject to a smoothing parameter that determines the size of the intervals.

I don't think that the current EPDFs integrate to one. And reasonable models require large data. So the current method requires some improvement.

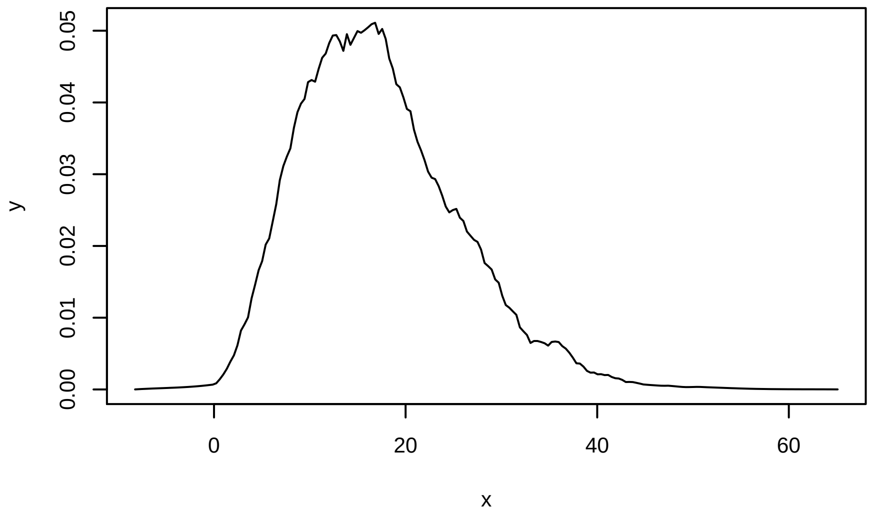
We can use the euvpdf() function. I recommend using the ebind function first to add two additional data points:

```
> x = rnorm (2000, 4) ^ 2
```

```
> ebx = ebind (x)
> f = euvpdf (ebx)
> f
```

```
function (x)
{
    .euvpdf.eval(x)
}
attr(,"class")
[1] "euvpdf"
attr(,"smoothness")
[1] 0.04469902
attr(,"n")
[1] 2002
note that some attributes not printed
```

```
> plot (f)
```



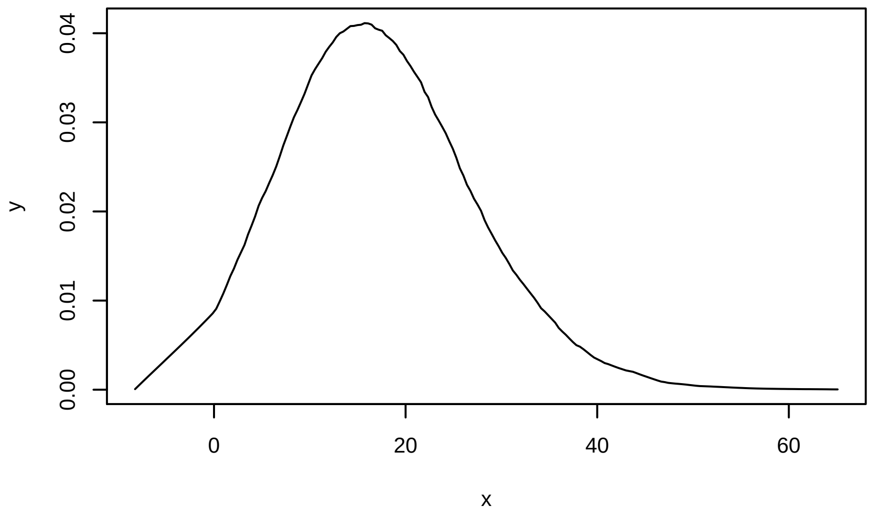
The object returned is a function so we can evaluate it:

```
> f (16)
```

```
[1] 0.04973911
```

It's possible to specify a smoothing parameter. A value of 0.25 indicates that an interval equal to 0.25*diff(range(x)). Higher values produce smoother models but are likely to over smooth.

```
> f = euvpdf (ebx, 0.25)
> plot (f)
```



Empirical Cumulative Distribution Functions

We can compute a step function ECDF function using the following expression:

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

Where I() is 1 if the enclosed expression is true and 0 if false, n is the number of observations and x^* is a vector of observations.

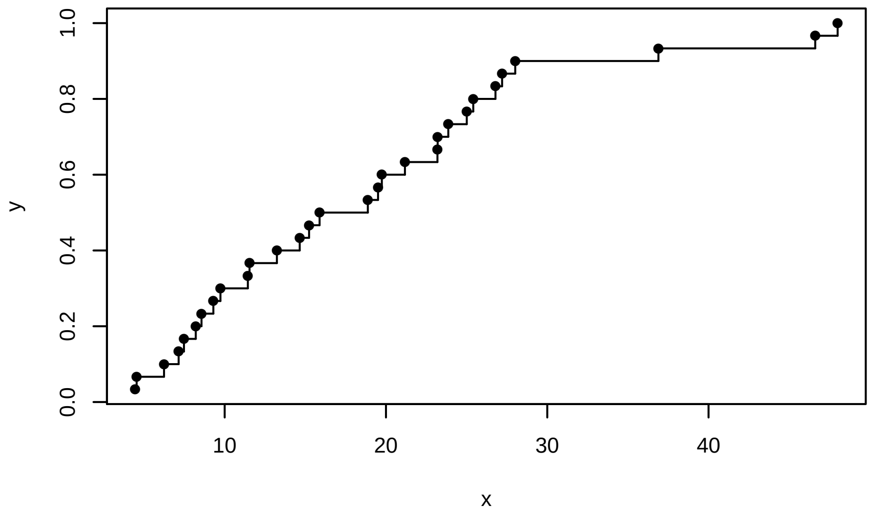
We can used the euvcdf() function:

```
> x = rnorm (30, 4) ^ 2
```

```
> F = euvcdf (x)
> F
```

```
function (x)
{
    .euvcdf.step.eval(x)
}
attr(,"class")
[1] "euvcdf"
attr(,"continuous")
[1] FALSE
attr(,"n")
[1] 30
attr(,"x")
 [1]  4.442193  4.542436  6.240843  7.145910  7.475982  8.206847  8.562720
 [8]  9.298018  9.736920 11.436204 11.541768 13.237064 14.652331 15.233389
[15] 15.884623 18.875957 19.508687 19.744361 21.175783 23.190651 23.205773
[22] 23.864912 25.009825 25.411059 26.789728 27.201172 28.013776 36.891845
[29] 46.613617 48.005241
```

```
> plot (F)
```



The object returned is a function so we can evaluate it:

```
> F (16)
```

```
[1] 0.5
```

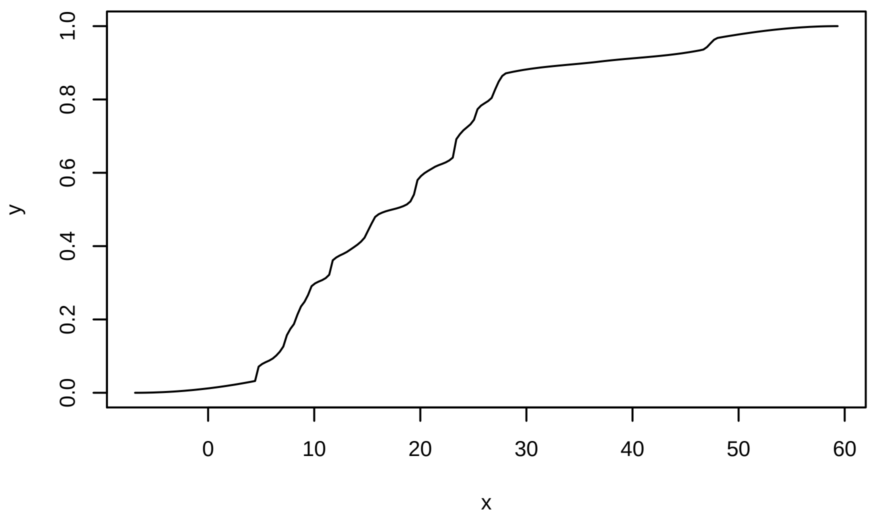
We can compute a continuous ECDF by computing two vertices:

$$\mathrm{F}_v(a) = \frac{\sum_i \mathrm{I}(x_i^* \leq a) - 1}{n - 1}, \mathrm{F}_v(b) = \frac{\sum_i \mathrm{I}(x_i^* \leq b) - 1}{n - 1}$$

Where a and b are the values of x^* adjacent to x. Then interpolating between them.

We can using the euvcdf() function with TRUE as the second argument. I recommend using the ebind() function first to add two additional data points.

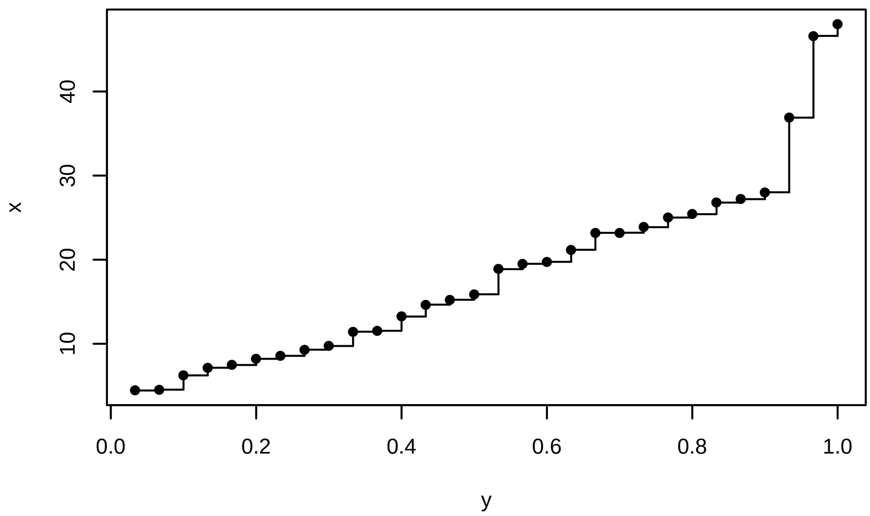
```
> ebx = ebind (x)
> F = euvcdf (ebx, TRUE)
> plot (F)
```



Inverse Empirical Cumulative Distribution Functions

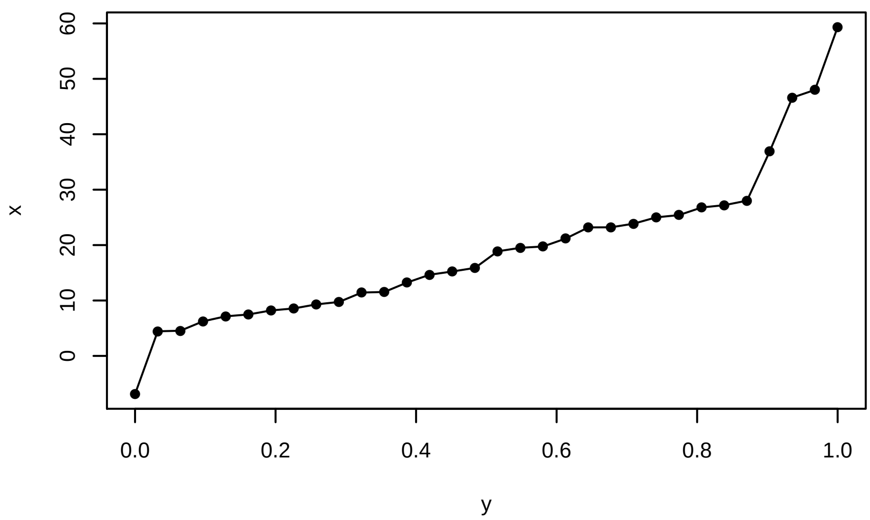
We can construct an inverse ECDF using the euvcdf.inverse() function:

```
> F.inverse = euvcdf.inverse (x)
> plot (F.inverse)
```



Or a continuous version:

```
> F.inverse = euvcdf.inverse (ebx, TRUE)
> plot (F.inverse)
```



Currently, this function uses linear interpolation rather than cubic hermite splines.

Multivariate Empirical Cumulative Distribution Functions

We can compute a step function bivariate ECDF using the following expression:

$$\mathbb{P}(X_1 \leq x_1, X_2 \leq x_2) = \mathrm{F}(x_1, x_2) = \frac{\sum_i \mathrm{I}(x^*_{[i][1]} \leq x_1 \wedge x^*_{[i][2]} \leq x_2)}{n_r}$$

Where n_r is the number of observations.

This can be generalized for more variables.

However, there are some issues with this expression. So I'm considering alternatives.