`gridR`: Grid Builder and Spline Knot Selector for R

Florian Oswald

May 2, 2013

https://github.com/floswald/gridR

1 Intro

`gridR` is a small utility package that helps with an ubiquitous task in **state space modelling** or **function approximation** in general:

> making it easy to select different grids on which to evaluate the model/function.

It is often the case that an analyst starts out the first draft by setting a certain space to a *uniformly spaced grid* as in `seq(from,to,length)`, with the intention of assessing the robustness of the results with respect to this choice later on.

But, alas, this is seldom done. It is worth mentioning at this point that approximations are extremely sensitive to where the approximation is measured. The obvious response is the rule of thumb that *the more points, the better* – which is certainly true – but frequently the strategy of filling out `seq(from,to,length=N)` with many points `N` is not practical because it implies great computational cost.

In such a sitution `gridR` facilitates an easy way to experiment with different grids. There are currently 7 different spacing rules available via `grid.maker()` which transform the uniform grid. Extensions are easy to make by any user, just extend on of the existing makers with your mapping.

The analyst can use some prior knowledge about where in the state space a greater number of points is required whn choosing the appropriate rule. Each `grid.maker()` has a plotting method to help making this choice.

Relatedly, the package contains a function `knot.select()` to construct *knot vectors* for use with `splines::splineDesign()`. The usefulness of this may not be immediately obvious, given that functions like `splines::bs()` and `splines::ns()` take care of this task internally. Both functions evaluate the polynomial basis function of the spline at suitably chosen quantiles of the supplied data. The need to select a knot vector in a more flexible kind of way and to actually *see* the knot vector and use it together with `splines::splineDesign()` arises again in function approximation, more so than in, say, regression analysis with splines. Let it be said that `splines::bs()` is close to this, but it lacks some flexiblity. For instance, one cannot compute the derivative of the basis function. This is important for example in applications where the user needs to supply an analytic gradient of some function to a numeric optimization routine. A judicious choice of knot placement can have great influence on the quality of approximation and indeed the value of the solution to the optimization problem itself.

I'll now demonstrate the two main functions of the package.

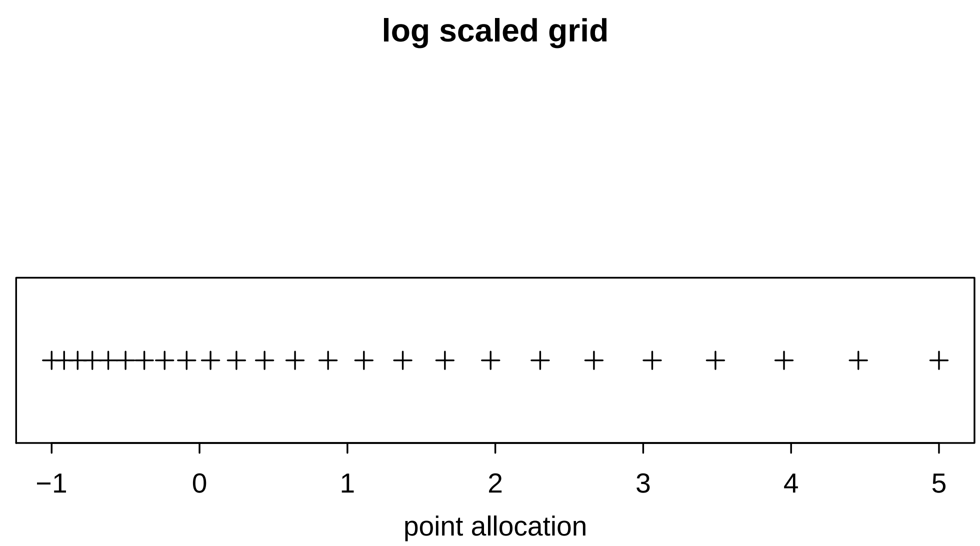
2 Some `grid.maker()` sample usage

Suppose we are interested in finding an approximating to the unknown function $f(x) : \mathbb{R} \mapsto \mathbb{R}$, and let's call it $\hat{f}$. One way to get $\hat{f}$ is to compute $f(x)$ at a finite number of points $X = \{x_i\}_{i=1}^N$ and establish a rule to *connect the dots.* `gridR` is useful for trying out different X's with little effort.

2.1 Log Scaling

Log scaling the uniform grid places more points towards the lower bound.

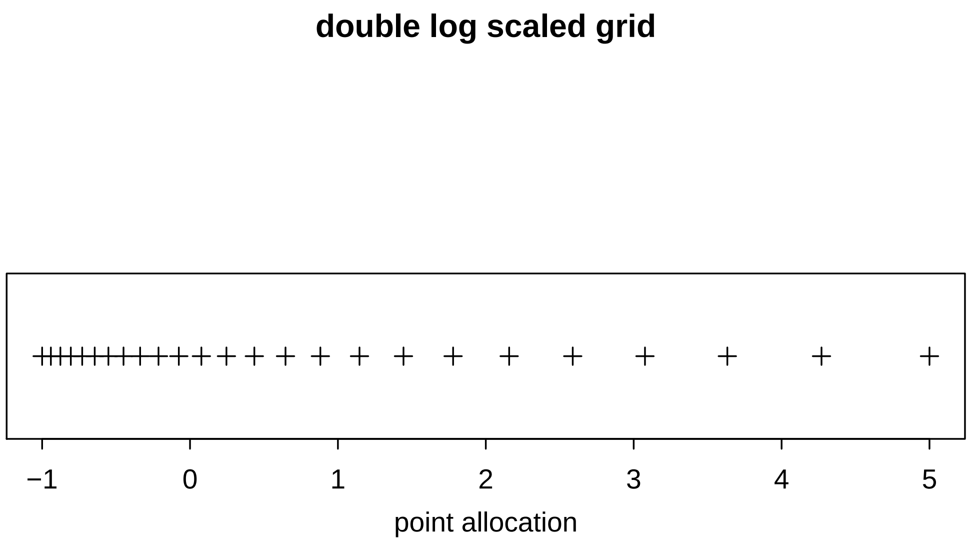
```
library(gridR)

## Loading required package:  evd

b <- c(-1, 5)
n <- 25
x <- grid.maker(bounds = b, num.points = n, spacing = "log.g", plotit = TRUE)
```



2.2 Double Log Scaling

Same idea, but apply the log transformation twice.

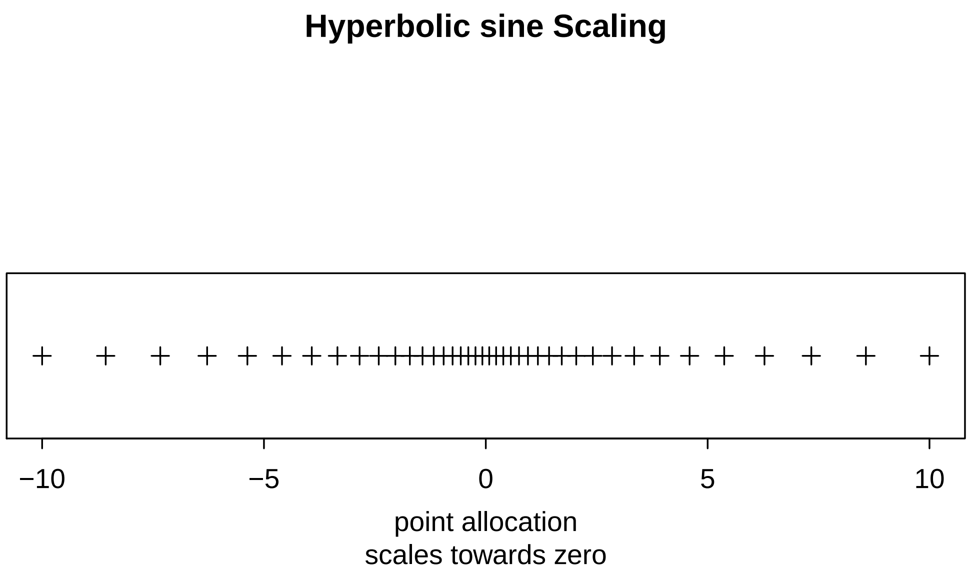
```
b <- c(-1, 5)
n <- 25
x <- grid.maker(bounds = b, num.points = n, spacing = "log.g2", plotit = TRUE)
```



2.3 Hyperbolic Sine Scaling

This transformation increases the point density around zero in a symmetric kind of way.

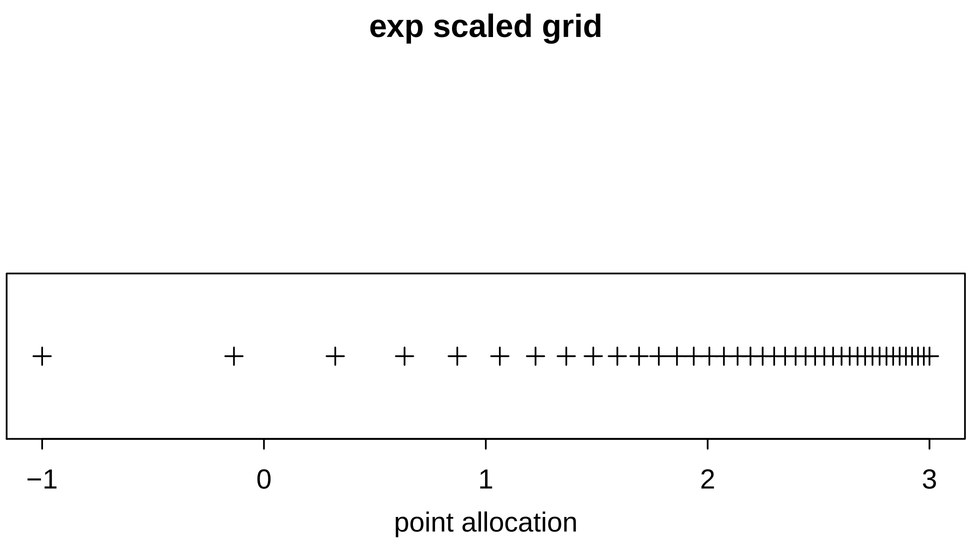
```
b <- c(-10, 10)
n <- 40
x <- grid.maker(bounds = b, num.points = n, spacing = "hyp.sine", plotit = TRUE)
```



2.4 Exponential Scaling

The exponential transformation bunches the points at the upper bound. Quickly takes off as the upper bound gets large.

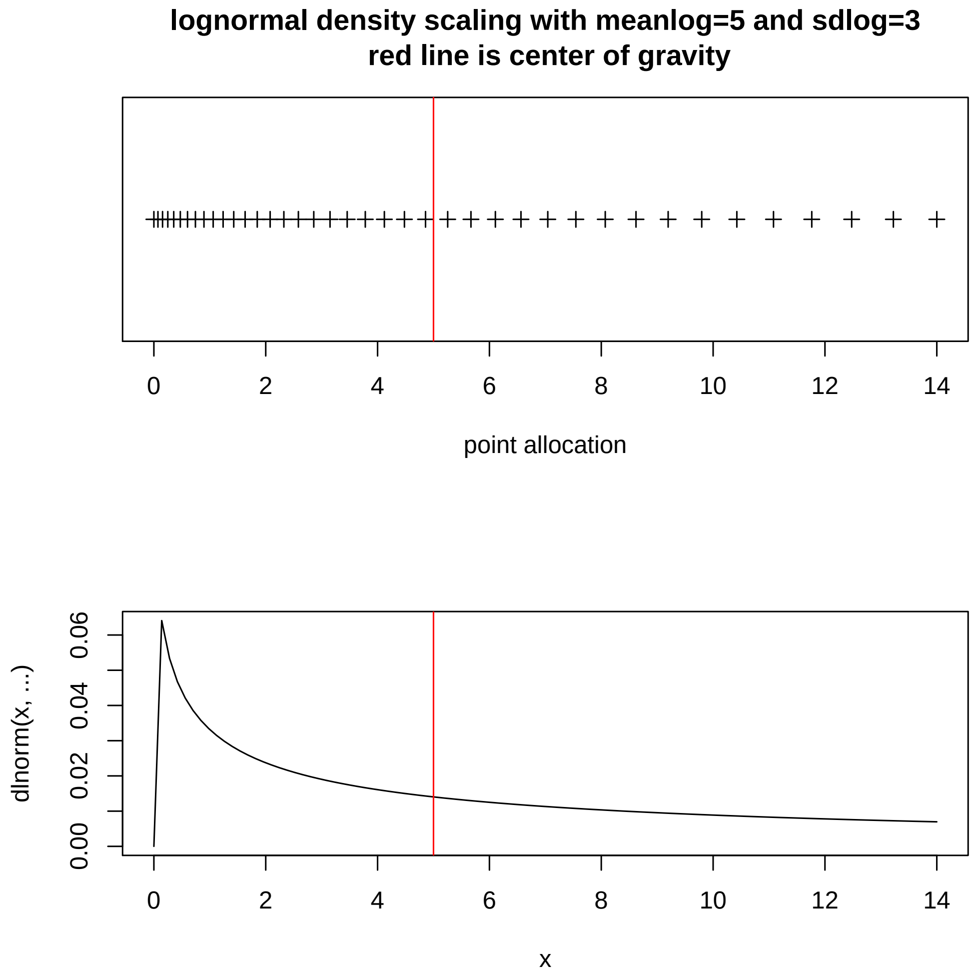
```
b <- c(-1, 3)
n <- 40
x <- grid.maker(bounds = b, num.points = n, spacing = "exp.grid", plotit = TRUE)
```



2.5 Log normal scaling

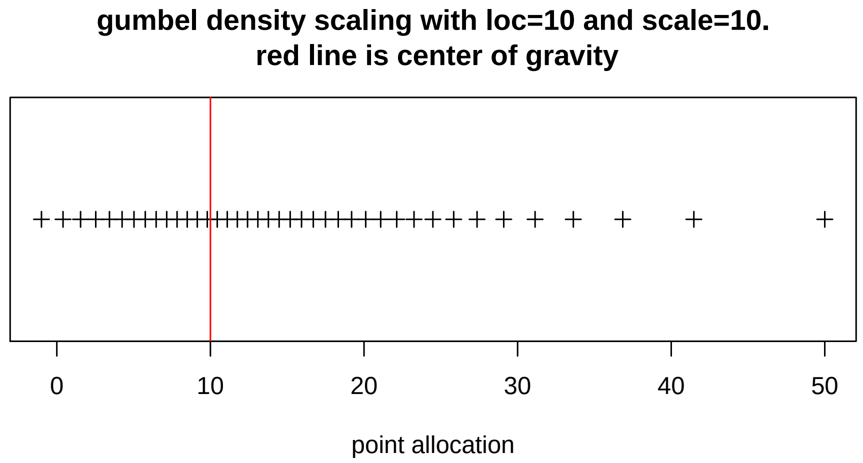
This is the first in a series of transformations that can be flexibly changed by passing additional parameters. Can be useful if you need a relatively long tail.

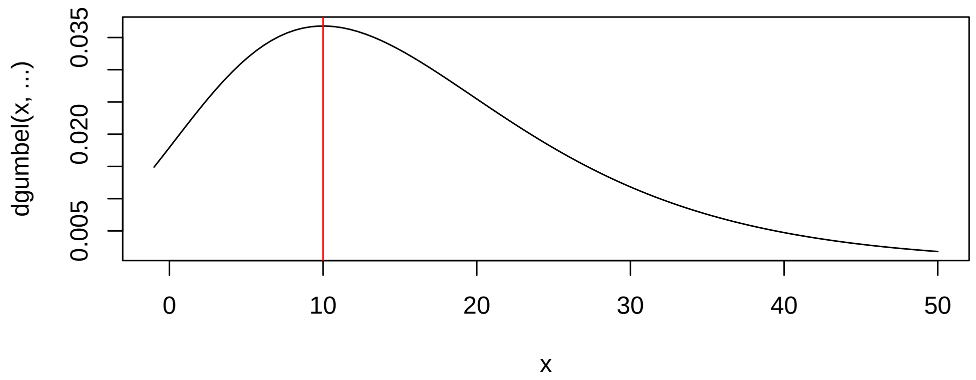
```
b <- c(0, 14)
n <- 40
x <- grid.maker(bounds = b, num.points = n, spacing = "lognorm.grid", plotit = TRUE,
    meanlog = 5, sdlog = 3)
```



2.6 Gumbel density scaling

```
n <- 40
x <- grid.maker(bounds = c(-1, 50), num.points = n, spacing = "gumbel.grid",
    plotit = TRUE, loc = 10, scale = 10)
```

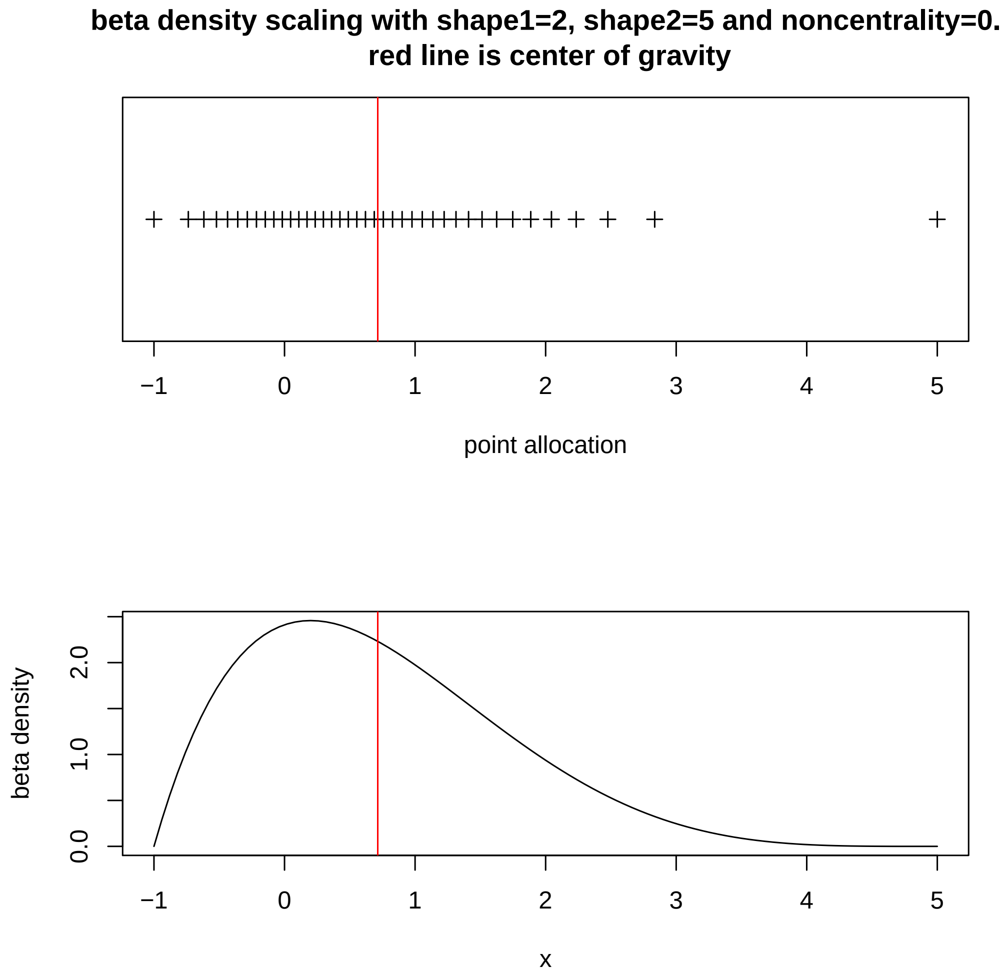




2.7 Beta density scaling

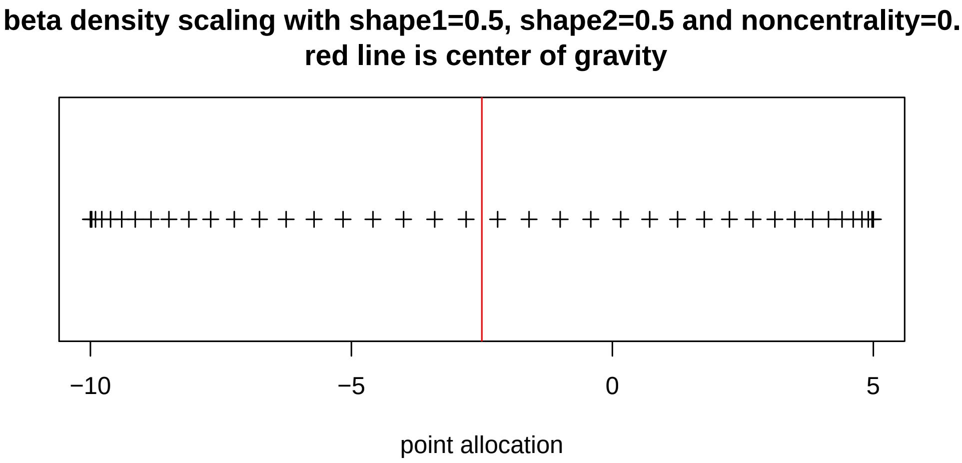
The beta is well known to be extremely flexible. Playing around with the shape parameters will give you any kind of point allocation you may desire.

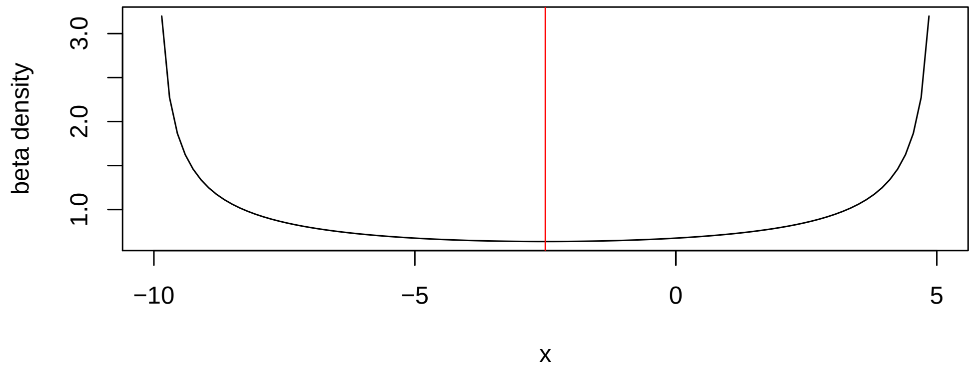
```
n <- 40
x <- grid.maker(bounds = c(-1, 5), num.points = n, spacing = "beta.grid", plotit = TRUE,
    shape1 = 2, shape2 = 5, ncp = 0)
```



To illustrate, let's change those parameters:

```
n <- 40
x <- grid.maker(bounds = c(-10, 5), num.points = n, spacing = "beta.grid", plotit = TRUE,
    shape1 = 0.5, shape2 = 0.5, ncp = 0)
```





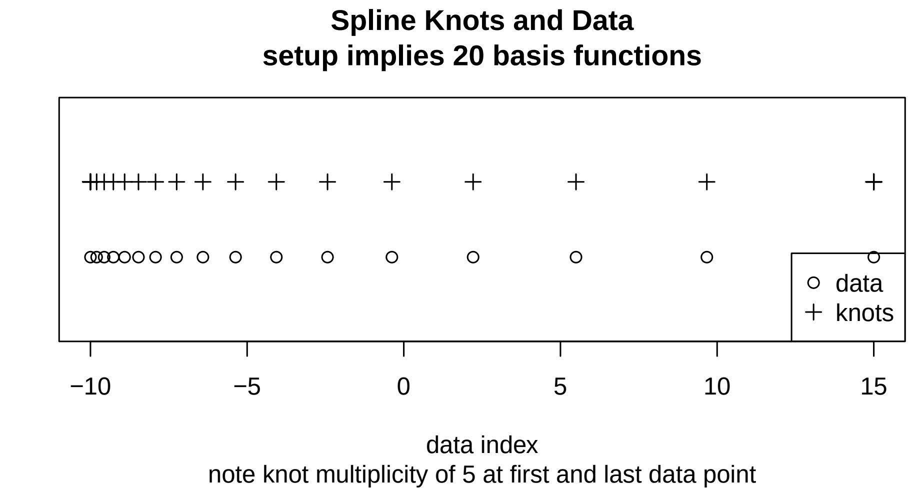
3 `Knot.Select()`

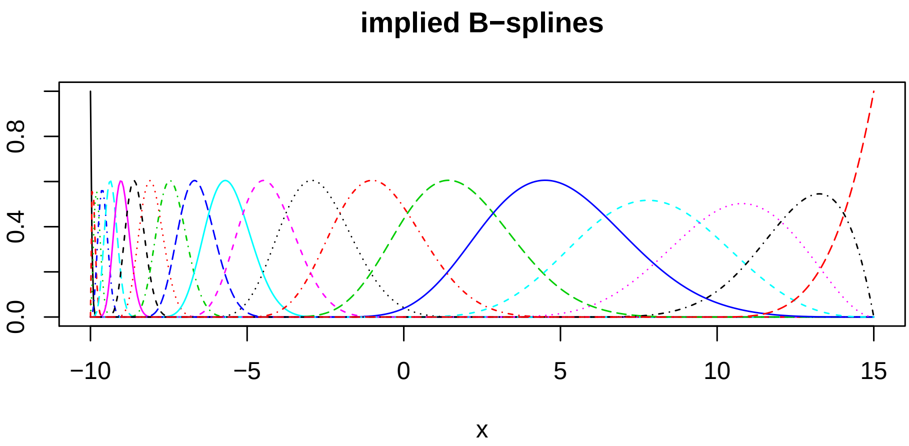
3.1 Case 1: strong prior about optimal knot location

We'll construct a grid x and then use all except the first and last grid point as interior knots. By setting `num.basis=NULL`, we let the algorithm figure out how many basis functions we'll need. This is useful in cases where you have a very strong prior on where your interior knots should be (i.e. you think they should be *exactly* on x).

```
n <- 17  # I want n-2 = 15 interior knots
deg <- 4  # I want spline degree 4
# I will get 15 + deg + 1 = 20 basis functions
x <- grid.maker(c(-10, 15), num.points = n, spacing = "log.g2")
```

```r
k <- knot.select(degree = deg, x = x, num.basis = NULL, plotit = TRUE)
```





3.2 Case 2: fix `num.basis` and place knots at quantiles of data

In this case we fix the number of basis functions we want and let the algorithm place the knots at equispaced quantiles of the data. Depending on the number of available interior knots (available after we constructed the multiplicities at both ends of the knot vector), the quantiles are

# interior knots	quantiles
1	0.5
2	$\frac{1}{3}, \frac{2}{3}$
3	$\frac{1}{4}, \frac{1}{2}, \frac{3}{4}$
4	$\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}$
...	...

```
nb <- 8  # I want 8 basis functions
# I supply the same grid x
k <- knot.select(degree = deg, x = x, num.basis = nb, plotit = TRUE)
```