Minimal Empirical Density Estimation

Matthew Leonawicz

December 24, 2014

1 Use case 2: Vegetation

The motivation and the context for vegetation area and age distributions from SNAP's ALFRESCO model ouput are the same as with use case one. However, the data are different. Through preliminary investigation, I can make changes to my approach, such as requiring a greater initial sample size or some other change that will maintain accuracy, which conceivably will differ for different random variables, data sets, and forms of variability and uncertainty involved. In this case I do something very similar to above.

As in the case of climate, there is a similar function for estimating densities storing only minimal information. However, with this data there were more things to look out for, such as entirely missing data or all data consisting of a single unique value (e.g., all modeled vegetation ages in one subregion are the same age in the ALFRESCO model), and how these situations are to be handled.

Furthermore, depending on how they are handled, which to some extent depends on how I eventually plan to use these estimated distributions, there are different potential consequences arising from other internal forms of sample. For instance, if all cells in a map layer of a given vegetation class are of the same age, do I want to sample with some kernal located on that value, knowing that these ages are not all the same in reality? On the other hand, I could return nothing but `NA` values if I am not satisfied with certain circumstances, as shown in the subsequent version of `denFun` which is applied to vegetated area rather than age.

```
denFun <- function(x, n, min.zero = TRUE, diversify = FALSE) {
    x <- x[!is.na(x)]
    lx <- length(x)
    if (diversify && length(unique(x)) == 1)
        x <- rnorm(max(10, lx), mean = x[1])  # diversify constant values
    if (lx == 1)
        x <- x + c(-1:1)  #single pixel of veg type, add and subtract one age year to make procedure pos
    dif <- diff(range(x))
    z <- density(x, adjust = 2, n = n, from = min(x) - max(1, 0.05 * dif), to = max(x) +
        max(1, 0.05 * dif))
    if (min.zero && any(z$x < 0))
        z <- density(x, adjust = 2, n = n, from = 0, to = max(x) + max(1, 0.05 *
            dif))
    as.numeric(c(z$x, z$y))
}
```

```
denFun <- function(x, n = 20, min.zero = TRUE, diversify = FALSE, missing.veg.NA = TRUE,
    fire = FALSE) {
    if (all(is.na(x)))
        return(rep(NA, 2 * n))
    x <- x[!is.na(x)]
    lx <- length(x)
    if (sum(x) == 0 & missing.veg.NA & !fire)
        return(rep(NA, 2 * n))
    if (diversify && length(unique(x)) == 1)
        x <- rnorm(max(10, lx), mean = x[1])  # diversify constant values
    if (lx == 1)
        x <- x + c(-1:1)  #single pixel of veg type, add and subtract one age year to make procedure pos
    dif <- diff(range(x))
    z <- density(x, adjust = 2, n = n, from = min(x) - max(1, 0.05 * dif), to = max(x) +
        max(1, 0.05 * dif))
    if (min.zero && any(z$x < 0))
        z <- density(x, adjust = 2, n = n, from = 0, to = max(x) + max(1, 0.05 *
            dif))
    as.numeric(c(z$x, z$y))
}
```

Again, there is another case of bootstrap resampling from the estimated distributions to arrive at new simulated, representative draws from the original distribution.

```
btfun <- function(p, n.samples = length(p)/2, n.boot = 10000, interp = FALSE,
    n.interp = 1000, ...) {
    if (!length(p))
        return(p)
    if (all(is.na(p)))
        return(rep(NA, n.boot))
    p <- list(x = p[1:n.samples], y = p[(n.samples + 1):(2 * n.samples)])
    if (interp && length(unique(p[1:n.samples])) > 1)
        p <- approx(p$x, p$y, n = n.interp)
    p <- round(sample(p$x, n.boot, prob = p$y, rep = T), ...)
    p
}
```

Think of it like teleporting to your destination instead of walking, specifically when you have a heavy pack. As with use case 1, it is helpful to see the documentation for the ALFRESCO output extraction and related projects to which these functions belong.