SFP detection based on dip statistic

Na You

May 5, 2010

load the package.

```
> library(dipSFP)
```

load the dataset to be analyzed, for example, the estimated binding affynity values. Here, a dataset is generated for illustration.

```
> n <- 100
> m.nonSFP <- 98
> m.SFP <- 2
> dat1 <- matrix(rgamma(m.nonSFP * n, 2, 4), m.nonSFP, n)
> dat2 <- cbind(matrix(rgamma(m.SFP * n/2, 2, 4), m.SFP, n/2),
+     matrix(rgamma(m.SFP * n/2, 20, 4), m.SFP, n/2))
> dat <- rbind(dat1, dat2)
```

Determine the number of bootstrap replications and tuning parameters based on the real data, then compute the statistic values, p values and adjusted p values for every probes. The estimated null distribution of statistic is also returned.

```
> s1 <- dipSFP(dat, nboot = 20)
> str(s1)

List of 4
 $ dip.statistics   : num [1:100] 0.0223 0.0345 0.0254 0.0223 0.0273 ...
 $ null.distribution: num [1:2000] 0.023 0.0305 0.0333 0.0265 0.0212 ...
 $ p.values         : num [1:100] 0.884 0.145 0.673 0.878 0.532 ...
 $ adjusted.pvalues : num [1:100] 0.99 0.731 0.974 0.99 0.899 ...

> s2 <- dipSFP(dat, nboot = 20, outliers.ratio = 0.01)
> str(s2)

List of 4
 $ dip.statistics   : num [1:100] 0.0223 0.0345 0.0254 0.0223 0.0273 ...
 $ null.distribution: num [1:2000] 0.0248 0.0305 0.0204 0.0324 0.0199 ...
 $ p.values         : num [1:100] 0.904 0.14 0.695 0.898 0.551 ...
 $ adjusted.pvalues : num [1:100] 1 0.732 0.997 1 0.935 ...
```

If the dataset is too big, then split it into several sub-datasets, and compute the statistic values and estimated null distribution parallelly for each sub-dataset, then combine them together to calculate p values and adjusted p values.

```
> subdat1 <- dat[1:n/2, ]
> subdat2 <- dat[(n/2 + 1):n, ]
> res <- list()
> res[[1]] <- dipSFP(subdat1, nboot = 20, pvalue = FALSE)
> res[[2]] <- dipSFP(subdat2, nboot = 20, pvalue = FALSE)
> stats <- unlist(lapply(res, function(x) x$dip.statistics))
> null.dis <- unlist(lapply(res, function(x) x$null.distribution))
> pvals <- sapply(stats, function(x) mean(null.dis > x))
> padj <- p.adjust(pvals, method = "BH")
```

Given the desired number of modes and tuning parameters, the modifiled critical bandwidth and the locations of modes can be calculated as following.

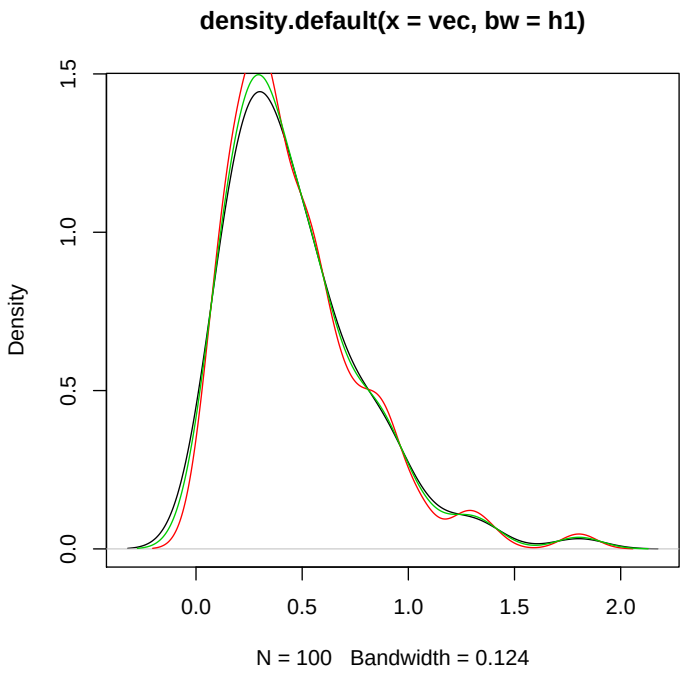
```
> vec <- dat[1, ]
> h1 <- FMbw(vec, k = 1)
> modes(vec, bw = h1)

[1] 0.3470595

> h2 <- FMbw(vec, k = 2)
> modes(vec, bw = h2)

[1] 0.3380767 1.1485827

> h20 <- FMbw(vec, k = 2, m0 = 0)
> plot(density(vec, bw = h1))
> lines(density(vec, bw = h2), col = 2)
> lines(density(vec, bw = h20), col = 3)
```


density.default(x = vec, bw = h1)
Density
0.0
0.5
1.0
1.5
0.0
0.5
1.0
1.5
2.0
N = 100 Bandwidth = 0.124