

KC-SMART Vignette

Jorma de Ronde, Christiaan Klijn

October 22, 2008

1 Introduction

In this Vignette we demonstrate the usage of the `KCsmart` R package. This R implementation is based on the original implementation coded in Matlab and first published in January 2008 [1].

KC-SMART is a method for finding recurrent gains and losses from a set of tumor samples measured on an array Comparative Genome Hybridization (aCGH) platform. Instead of single tumor aberration calling, and subsequent minimal common region definition, KC-SMART takes an approach based on the continuous raw \log_2 values.

Briefly: we apply Gaussian locally weighted regression to the summed totals positive and negative \log_2 ratios separately. This result is corrected for uneven probe spacing as present on the given array. Peaks in the resulting normalized KC score are tested against a randomly permuted background. Regions of significant, recurrent gains and losses are determined using the resulting, multiple testing corrected, significance threshold.

2 Basic Functionality

After installing the `KCsmart` package via Bioconductor, load the `KCsmart` code into R.

```
> library(KCsmart)
```

Load some sample aCGH data. We supply an example dataset of an old 3000 probe BAC clone experiment. KC-SMART has no problem running on modern 300k+ probe datasets, but this dataset is supplied to keep the package size low. This dataset can be a simple dataframe, `DNACopy` object or a `CGHbase` object, and can not contain any missing values. If the data contains missing values, we advise the user to either impute the missing values using the mean of the two neighboring probes or to discard probes containing missing values.

```
> data(hsSampleData)
```

This data is the simplest form available to use in KCSMART. As can be observed using:

```
> str(hsSampleData)
```

```
'data.frame':      3268 obs. of  22 variables:
 $ chrom   : chr  "1" "1" "1" "1" ...
 $ maploc  : num  200000 200050 3284807 4542451 7103443 ...
 $ sample 1 : num  1.326 0.0368 0.4335 1.8879 1.2002 ...
 $ sample 2 : num  1.1155 -0.0478 0.641 0.456 0.831 ...
 $ sample 3 : num  0.4204 0.6486 -0.3638 -0.2573 0.0953 ...
 $ sample 4 : num  0.06313 -0.00108 0.13552 0.57361 -0.09144 ...
 $ sample 5 : num  0.816 0.843 0.893 1.275 0.686 ...
 $ sample 6 : num  0.471 1.262 0.4 0.987 0.725 ...
 $ sample 7 : num  1.296 0.471 0.614 1.007 1.041 ...
 $ sample 8 : num  0.835 0.522 0.643 0.572 0.953 ...
 $ sample 9 : num  1.98 -0.529 0.808 -0.165 1.668 ...
 $ sample 10: num  0.668 0.313 1.083 0.7 1.782 ...
 $ sample 11: num  0.338 -0.118 0.205 1.402 0.299 ...
 $ sample 12: num  0.688 -0.187 0.593 0.941 0.625 ...
 $ sample 13: num  0.4127 0.5349 0.9524 0.9948 0.0954 ...
 $ sample 14: num  0.938 0.254 0.599 0.615 0.466 ...
 $ sample 15: num  0.0481 -0.141 0.9038 0.5821 0.2582 ...
 $ sample 16: num  0.4 -0.306 0.198 0.652 0.589 ...
 $ sample 17: num  1.178 0.72 0.731 1.192 0.287 ...
 $ sample 18: num  0.2878 0.5926 1.3355 0.6192 -0.0581 ...
 $ sample 19: num  0.738 0.489 0.739 1.075 -0.109 ...
 $ sample 20: num  1.552 0.439 0.264 0.841 0.218 ...
```

The user can easily model their data in this format, basically only requiring a column named `$chrom` containing the chromosome on which the probe is located and a column named `$maploc` containing the chromosomal location of the probe. Alternatively, both `DNAcopy` and `CGHbase` objects can be used as direct input for the `calcSpm` function discussed below.

KC-SMART uses mirroring of the probes at the ends of chromosomes and near centromeres to prevent signal decay by the convolution. The locations of the centromeres and the lengths of the chromosomes are therefore necessary information. Hard-coded information about the human and mouse genome is supplied, but the user can easily provide the coordinates themselves in R. The `'mirrorLocs'` object is a list with vectors containing the start, centromere (optional) and end of each chromosome as the list elements. Additionally it should contain an attribute `'chromNames'` listing the chromosome names of each respective list element. The provided `'hsMirrorLocs'` can serve as an example of such a list.

To load the presupplied information use:

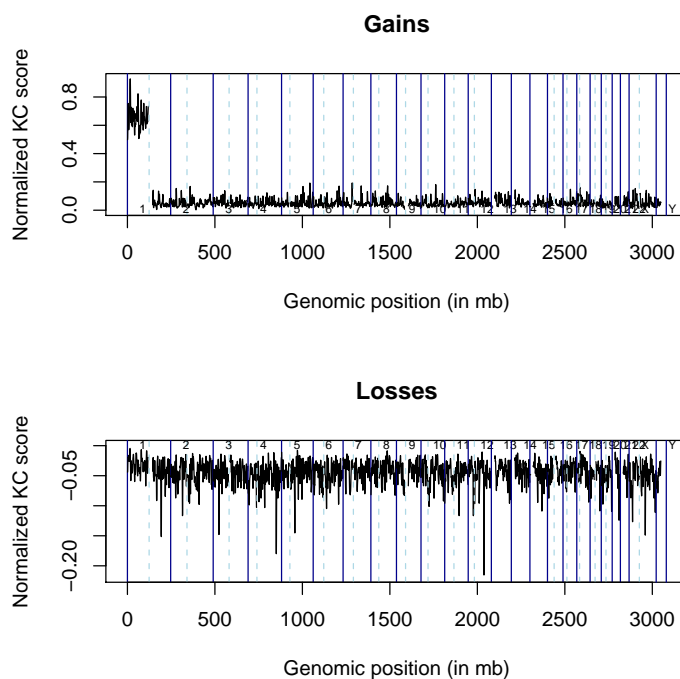
```
> data(hsMirrorLocs)
```

For an analysis in mouse `data(mmMirrorLocs)` is used. Using the `calcSpm` command the convolution is performed at the given parameters and stored in a samplepoint matrix. Here we perform the convolution at two different kernel-widths (first the default `sigma = 1Mb`, and a second `4Mb sigma`). We apply the default parameters to run the convolution.

```
> spm1mb <- calcSpm(hsSampleData, hsMirrorLocs)
> spm4mb <- calcSpm(hsSampleData, hsMirrorLocs, sigma = 4e+06)
```

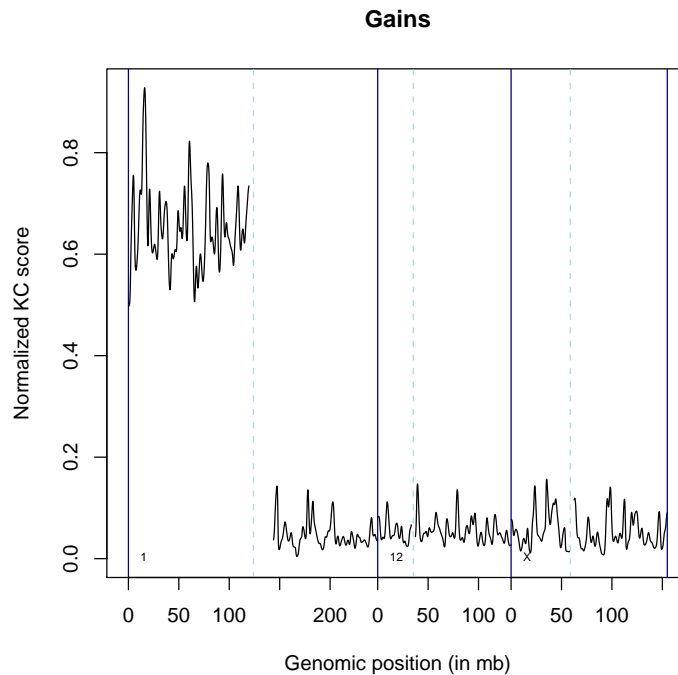
Now we plot the chromosome-wide view of the convolution for both the gains and the losses for the 1Mb kernelwidth analysis. Since the samplepoint matrix we just created is an S4 object we can just apply the plot command to it to visualize the results. To keep the size of the vignette down we use an interpolation of 10 here, resulting in every 10th point being plotted.

```
> plot(spm1mb, interpolation = 10)
```



It is possible to select only particular chromosomes in the plot command, and to only display the gains or only the losses. We will now visualize the KCsmart output for gains of chromosomes 1, 12 and X.

```
> plot(spm1mb, chromosomes = c(1, 12, "X"), type = "g")
```



As can be seen, in this particular (synthetic) dataset chromosome 1 shows a large recurrent gain compared to other chromosomes. How significant is this gain? We can determine a significance level based on permutation to determine this. In this example we will permute the data 10 times and determine a threshold at $p < 0.05$. We recommend 1000 permutations if you want to do a definitive analysis of a dataset.

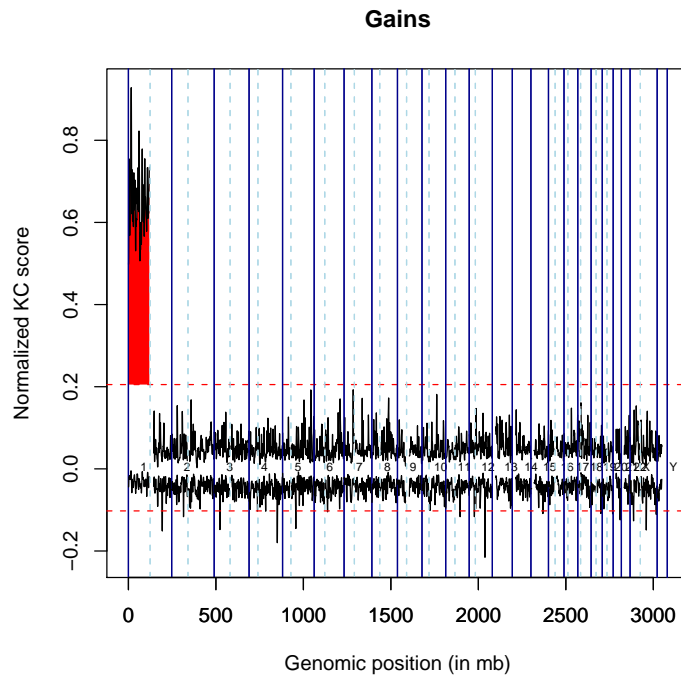
```
> sigLevel1mb <- findSigLevelTrad(hsSampleData, spm1mb, n = 3,
+   p = 0.05)
```

```
[1] "Calculating alpha = 0.05 significance cut-off"
[1] "Found 552 pos peaks and 566 neg peaks in observed sample point matrix"
[1] "Starting permutations .."
```

```
At iteration 1 of 3
At iteration 2 of 3
At iteration 3 of 3[1] "\n"
```

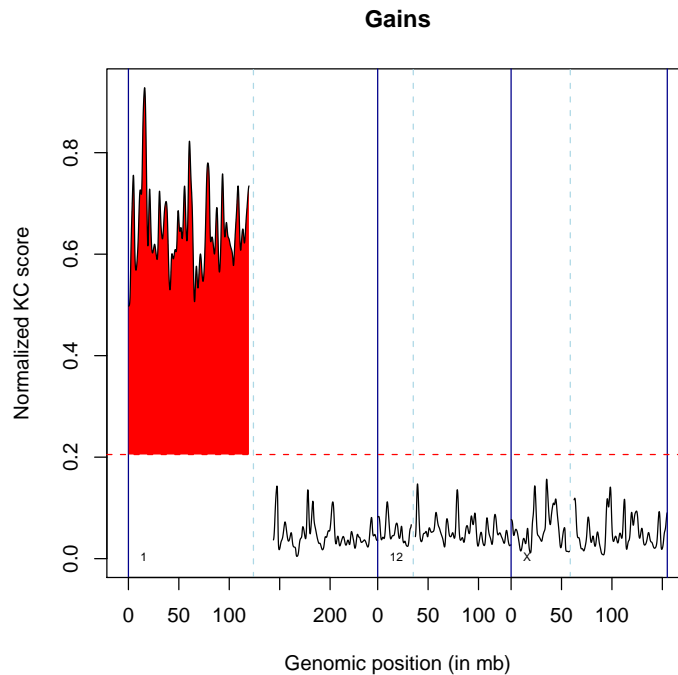
We can now plot the significance level we just calculated in the genomewide view of the KCsmart result using the `sigLevels=` parameter of the `plot` function. This time, we'll use the `type=1` parameter. This will plot the gains and losses in a single frame.

```
> plot(spm1mb, sigLevels = sigLevel1mb, type = 1, interpolation = 10)
```



Let's take a closer look at the gains of chromosome 1, 12 and X again, now with the added significance level.

```
> plot(spm1mb, chromosomes = c(1, 12, "X"), type = "g", sigLevels = sigLevel1mb)
```



We've also implemented a less stringently corrected version of the permutation algorithm that was shown above. This approach doesn't calculate the Bonferroni corrected significance level, but the False Discovery Rate (FDR). The command to find the FDR for this particular dataset would be `FDR1mb <- findSigLevelFdr(hsSampleData, spm1mb, n=10, fdrTarget = 0.05)`. In general the FDR is less conservative, and caution must be taken when using this as a threshold.

Now we have our significance level we would like to get usable information back about which regions are found to be significantly aberrant and which probes of the original dataset are contained in these regions. Using the `getSigSegments` function we can input a samplepoint matrix and our calculated significance level to get back the relevant information.

```
> sigRegions1mb <- getSigSegments(spm1mb, sigLevel1mb)
```

Now let's have a look at the information contained in `sigRegions1mb`, it is again an S4 object and will display its contents if you just type its name.

```
> sigRegions1mb
```

```
Significantly gained and lost segments
```

```
Cutoff value was 0.205192487793901 for gains and -0.102382312107292 for losses
A total of 1 gains and 18 losses were detected
```

```
Use 'write.table' to write the file to disk
```

You could write the information contained in `sigRegions1mb` to a file using `write.table(write.table(sigRegions1mb, file='sig.txt'))`. You can also

access the information in R using subsetting. For example, if you wanted to find the probe-names of the probes contained in the first significantly gained region you can get them like this:

```
> sigRegions1mb@gains[[1]]$probes

 [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[19] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
[37] 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
[55] 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
[73] 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92
[91] 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110
[109] 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
[127] 129
```

Using `str(sig.regions.1mb)` you can find out more about the way the significant regions are stored in the `sigRegions` object.

3 Multi-scale analysis

One of the powerful features of KC-SMART is the ability to perform multi-scale analysis. The algorithm can be run using different kernel widths, where small kernel widths will allow you to detect small regions of aberration and vice versa using large kernel widths large regions of aberration can be detected. Remember our convolution step? We calculated the convolution at two sigma's: 1Mb and 4Mb. We will now calculate a significance level for the 4Mb samplepoint matrix and plot the results of both kernelwidths in a scale space figure.

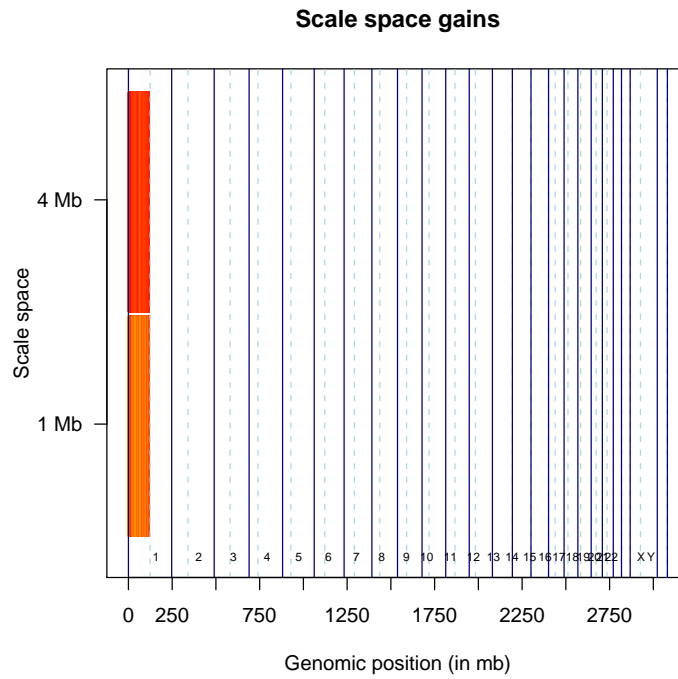
```
> sigLevel4mb <- findSigLevelTrad(hsSampleData, spm4mb, n = 3)

[1] "Calculating alpha = 0.05 significance cut-off"
[1] "Found 171 pos peaks and 180 neg peaks in observed sample point matrix"
[1] "Starting permutations .."

At iteration 1 of 3
At iteration 2 of 3
At iteration 3 of 3[1] "\n"
```

Now you can plot the scalespace of the two different kernel widths using `plotScaleSpace(list(spm1mb, spm4mb), list(sigLevel1mb, sigLevel4mb))`. A scale space can show you small and large significant regions in one plot, as well as give you insight within a single significant region. Darker red means more significant. As an option you can plot either the gains or the losses or both. When plotting both, two x11 devices will be opened. Using the 'chromosomes' parameter the chromosomes to be plotted can be set.

```
> plotScaleSpace(list(spm1mb, spm4mb), list(sigLevel1mb, sigLevel4mb),
+               type = "g")
```



4 References

1. Klijn *et. al.* Identification of cancer genes using a statistical framework for multiexperiment analysis of nondiscretized array CGH data. *Nucleic Acids Research*. **36** No. 2, e13.