Software Manual
Institute of Bioinformatics, Johannes Kepler University Linz

cn.FARMS: a latent variable model to detect copy number variations in microarray data with a low false discovery rate

— *Manual for the* `cn.farms` *package* —

Djork-Arné Clevert and Andreas Mitterecker

Institute of Bioinformatics, Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
okko@clevert.de and mitterecker@bioinf.jku.at

Version 1.51.0, April 30, 2024

Contents

1 Introduction

The `cn.farms` package provides a novel copy number variation (CNV) detection method, called "cn.FARMS", which is based on our FARMS ("factor analysis for robust microarray summarization" (Hochreiter *et al.*, 2006)) algorithm. FARMS is since 2006 the leading summarization method of the international "affycomp" competition if sensitivity and specificity are considered simultaneously. We extended FARMS to cn.FARMS (Clevert *et al.*, 2011) for detecting CNVs by moving from mRNA copy numbers to DNA copy numbers.
In the following section we will briefly describe the algorithm and provide a quick start guide. For further information regarding the algorithm and its assessment see the `cn.farms` homepage at `http://www.bioinf.jku.at/software/cnfarms/cnfarms.html`.

2 cn.FARMS: FARMS for CNV Detection

cn.FARMS is described "in a nutshell" by the preprocessing pipeline depicted in Figure 1: **(1) Normalization** is performed at two levels. It has as *input* the raw probe intensity values and as *output* intensity values at chromosome locations which are leveled between arrays and are allele independent. At the *first level* normalization methods remove technical variations between arrays arising from differences in sample preparation or labeling, array production (e.g. batch effects), or scanning differences. The goal of the first level is to correct for array-wide effects. At the *second level* alleles are combined to one intensity value at a chromosome location and a correction for cross-hybridization between allele A and allele B probes is performed. Cross-hybridization arise due to close sequence similarity between the probes of different alleles, therefore a probe of one allele picks up a signal of the other allele. The optional corrections for differences in PCR yield can be performed at this step or after "single-locus modeling". We propose sparse overcomplete representation in the two-dimensional space of allele A and B intensity to correct for cross-hybridization

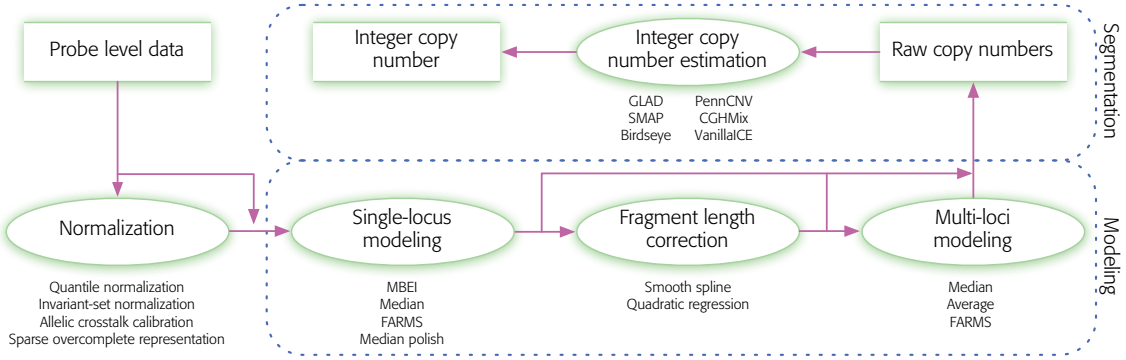

Figure 1: Copy number analysis for (Affymetrix) DNA genotyping arrays as a three-step pipeline: (1) Normalization, (2) Modeling, and (3) Segmentation. Modeling is divided into "single-locus modeling" and "multi-loci modeling" with "fragment length correction" as an optional intermediate step. The cn.FARMS pipeline is: normalization by sparse overcomplete representation, single-locus modeling by FARMS, fragment length correction, and multi-loci modeling by FARMS.

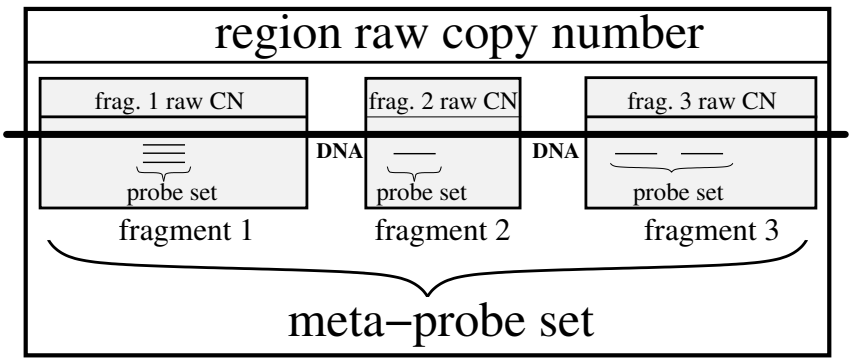

Figure 2: The copy number hierarchy probes-fragment-region. Fragment copy numbers serve as meta-probes used for "multi-loci modeling" which yields region copy numbers. Inner boxes: The probes which target a fragment (often at a SNP position) are single-locus summarized to a raw copy number of this fragment. Note, that instead of fragments a DNA probe loci can be summarized. Outer box: The raw fragment copy numbers are the meta-probes for a DNA region and are multi-loci summarized to a raw region copy number.

between allele A and allele B probes. Therefore we do not only estimate the AA and the BB cross-hybridization like CRMA (Bengtsson *et al.*, 2008) but also the AB cross-hybridization. The latter takes into account that hybridization and cross-hybridization may be different for the AB genotype, where for both allele probes target fragments are available and compete for hybridization. After allele correction, we follow CRMA and normalize by scaling the probes to a pre-specified mean intensity value. CNV probes which have only one allele are scaled in the same way. **(2) Modeling** is also performed at two levels. The *input* is the probe intensity values which independently measure the copy number of a specific target fragment or DNA probe locus. The *output* is an estimate for the region copy number. At the *first level*, "single-locus modeling" the probes which measure the same fragment are combined to a raw fragment copy number ("raw" means that the copy number is still a continuous values) —see Figure 2. These raw fragment copy numbers are estimated by FARMS. The original FARMS was designed to summarize probes which target the same mRNA. This can readily transfered to CNV analysis where FARMS now summarizes probes which target the same DNA fragment. Either both strands can be summarized together or separately where our default is the former.Nannya *et al.* (2005) suggested considering fragment characteristics like sequence patterns and the length because they affect PCR amplification. For example, PCR is usually less efficient for longer fragments, which lead to fewer copies to hybridize and result in weaker probe intensities. Following these suggestions cn.FARMS performs an optional intermediate level to correct for the fragment length and sequence features to make raw fragment copy numbers comparable along the chromosome. At the *second level*, "multi-loci modeling", the raw copy numbers of neighboring fragments or neighboring DNA probe loci are combined to a "meta-probe set" which targets a DNA region. The raw fragment copy numbers from single-locus modeling are now themselves probes for a DNA region as depicted in Figure 2. Again we use FARMS to summarize meta-probes and to estimate a raw copy number for the region. This modeling across samples is novel as previous methods only model along the chromosome. Multi-loci modeling considerably reduces the false discovery rates, because raw copy numbers of neighboring fragments or neighboring DNA probe loci must agree to each other on

the copy number, which reduces the likelihood of a discovery by chance. However, low FDR is traded against high resolution by the window size for multi-loci modeling, i.e. by how many raw copy numbers of neighboring fragments or neighboring DNA probe loci are combined. The more loci are combined, the more the FDR is reduced, because more meta-probes must mutually agree on the region's copy number. The window size for multi-loci modeling is a hyperparameter which trades off low FDR against high resolution. We recommend a window size of 5 as default, 3 for high resolution, and 10 for low FDR. Alternatively to a fixed number of CNV or SNP sites, the cn.FARMS software allows defining a window in terms of base pairs. In this case, multi-loci modeling may use a different number of meta-probes at different DNA locations, in particular for less than two meta-probes multi-loci modeling is skipped. Note, however that controlling the FDR is more difficult because a minimal number of meta-probes cannot be assured for each window and modeling with few meta-probes is prone to false discoveries. FARMS supplies an informative/non-informative (I/NI) call (Talloen *et al.*, 2007, 2010) which is used to detect CNVs. Additionally, the I/NI value gives the signal-to-noise-ratio of the estimated raw copy number. **(3) Segmentation** can afterwards be performed by `fastseg` or `DNAcopy`.

3 Getting Started: cn.FARMS

A very small subset of the HapMap data set on Affymetrix SNP 6.0 array - included in the R package `hapmapsnp6` - is used to show how the `cn.farms` is utilized.

3.1 Quick start : Process SNP 6.0 array

After loading the `cn.farms` it is sufficient to state the CEL files you want to process and run the function `cn.farms` to gain first results. Be aware that `cn.farms` will create result files in your current working directory.

```
> require(cn.farms)
> require("hapmapsnp6")
> celDir <- system.file("celFiles", package = "hapmapsnp6")
> filenames <- dir(path = celDir, full.names = TRUE)
> results <- cn.farms(filenames)
```

In this function only the copy number probes (and no SNP probes) are used for copy number estimation. Segmentation of the the gained results is advised as an additional step.

For more sophisticated settings - like different normalization methods, multicore support, finer parameter adjustment - we refer to Subsection 3.2.

3.2 Process SNP 6.0 array step by step

As usual, it is necessary to load the `cn.farms` package:

```
require(cn.farms)
```

The hapmapsnp6 package is loaded for testing purpose.

```
> require("hapmapsnp6")
> celDir <- system.file("celFiles", package="hapmapsnp6")
> filenames <- dir(path=celDir, full.names=TRUE)
```

Next, the user specifies a working directory on the harddisk where to save the results.

```
> workDir <- tempdir()
> dir.create(workDir, showWarnings=FALSE, recursive=TRUE)
> setwd(workDir)
```

For reasons of computational time and memory consumption `cn.farms` supports high-performance computation. The parameter `cores` specifies number of CPUs requested for the cluster and the parameter `runtype` indicates how the data matrix should be stored. `runtype="ff"` creates a transient flat-file which will not be saved automatically. Whereas `runtype="bm"` creates a persistent flat-file which can be saved permanently.

```
> cores <- 2
> runtype <- "ff"
```

Next, the user specifies a subdirectory whereto save the flat-files.

```
> dir.create("ffObjects/ff", showWarnings=FALSE, recursive=TRUE)
> ldPath(file.path(getwd(), "ffObjects"))
> options(fftempdir=file.path(ldPath(), "ff"))
```

The directory (`celDir="where/are/my/cel-files"`) which contain the cel-files has to be specified.

```
> celDir <- system.file("celFiles", package="hapmapsnp6")
> filenames <- dir(path=celDir, full.names=TRUE)
```

The following step will create the annotation file.

```
> if(exists("annotDir")) {
>        createAnnotation(filenames=filenames, annotDir=annotDir)
> } else {
>         createAnnotation(filenames=filenames)
> }
```

Afterwards, the data will be corrected for cross-hybridization and normalized.

```
> normMethod <- "SOR"

> ## normalization of SNP data
> if(exists("annotDir")) {
>         normData <- normalizeCels(filenames, method=normMethod, cores,
>                         alleles=TRUE, annotDir=annotDir, runtype=runtype)
> } else {
>         normData <- normalizeCels(filenames, method=normMethod, cores,
>           alleles=TRUE, runtype=runtype)
> }
```

Now, the normalized data will be summarized at DNA probe loci. `summaryMethod <- "Variational"` indicates which FARMS approach should be used and `summaryParam$cyc <- c(10, 10)` specifies the number of iterations of the EM-algorithm. The parameter `summaryWindow` indicates whether DNA probe loci on the same DNA fragments are summarized together (`summaryWindow="fragment"`) or if the DNA probe loci are summarized separately (`summaryWindow="std"` is the default setting).

```
> summaryMethod <- "Variational"
> summaryParam <- list()
> summaryParam$cyc <- c(10)
> callParam <- list(cores=cores, runtype=runtype)
> slData <- slSummarization(normData,
+                 summaryMethod=summaryMethod,
+                 summaryParam=summaryParam,
+                 callParam=callParam,
+                 summaryWindow="std")

2024-04-30 15:41:41.041195 |   Starting summarization
2024-04-30 15:41:41.046208 |   Computations will take some time, please be patient
2024-04-30 15:41:41.107563 |   Summarizing ...
2024-04-30 15:41:42.384308 |   Summarization done
Time difference of 1.417093 secs

> show(slData)

ExpressionSet (storageMode: list)
assayData: 34 features, 268 samples
  element names: intensity, L_z, IC, lapla
protocolData: none
phenoData
  rowNames: NA10846 NA12146 ... NA19238 (268 total)
  varLabels: filenames gender
  varMetadata: labelDescription
featureData
  featureNames: 532152 532153 ... 745206 (34 total)
```

```
  fvarLabels: chrom start ... man_fsetid (5 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: pd.genomewidesnp.6
```

```
> assayData(slData)$intensity[1:6, 1:5] ## intensity values
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 10.342002 10.803840 10.887975 10.755269 11.078718
[2,]  9.163161  9.996646  9.981950  9.850188  9.457437
[3,] 10.135801 11.210505 10.767266 10.908054 11.130400
[4,]  9.033439  9.977772  9.458606  9.710699  9.713734
[5,] 12.300862 12.224501 11.972838 12.146508 12.440324
[6,] 11.704262 11.692252 11.557435 11.985427 11.662011
```

```
> assayData(slData)$L_z[1:6, 1:5] ## relative values
```

```
            [,1]        [,2]        [,3]         [,4]         [,5]
[1,] -0.68146434 -0.21962721 -0.13549212 -0.268197864  0.055251714
[2,] -0.52203358  0.31145168  0.29675558  0.164992996 -0.227757886
[3,] -1.03323856  0.04146504 -0.40177357 -0.260985394 -0.038639342
[4,] -0.67243501  0.27189881 -0.24726799  0.004825231  0.007860522
[5,]  0.06263520 -0.01372537 -0.26538810 -0.091718464  0.202097403
[6,]  0.07681114  0.06480030 -0.07001661  0.357976062  0.034559524
```

Now, the intensity values of the non-polymorphic probes (CN-probes) were normalized.

```
> if (exists("annotDir")) {
        npData <- normalizeNpData(filenames, cores, annotDir=annotDir)
 } else {
        npData <- normalizeNpData(filenames, cores, runtype=runtype)
 }
```

This step combines non-polymorphic probes and single-locus summarized SNP-probes.

```
> combData <- combineData(slData, npData, runtype=runtype)
> show(combData)
```

```
ExpressionSet (storageMode: list)
assayData: 188 features, 268 samples
  element names: intensity
protocolData: none
phenoData
  rowNames: NA10846 NA12146 ... NA19238 (268 total)
  varLabels: filenames gender
```

```
  varMetadata: labelDescription
featureData
  featureNames: 70792 532141 ... 570013 (188 total)
  fvarLabels: chrom start end man_fsetid
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

In this final step intensity values of non-polymorphic probes and single-locus summarized SNP-probes are multi-locus summarized with a windows size of 5 probes (`windowParam$windowSize <- 5`). The window size for multi-loci modeling is a hyperparameter which trades off low FDR against high resolution. We recommend a window size of 5 as default, 3 for high resolution, and 7 for low FDR. Setting `windowParam$overlap <- TRUE` inidicates that the multi-locus summariaztion is done by step-wise moving the window over the genome. Alternatively to a fixed number of CNV or SNP sites, the cn.FARMS software allows defining a window in terms of base pairs. To make use of this option set `windowMethod <- "bps"`. In this case, multi-loci modeling may use a different number of meta-probes at different DNA locations, in particular for less than two meta-probes multi-loci modeling is skipped. Note, however that controlling the FDR is more difficult because a minimal number of meta-probes cannot be assured for each window and modeling with few meta-probes is prone to false discoveries.

```
> windowMethod <- "std"
> windowParam <- list()
> windowParam$windowSize <- 5
> windowParam$overlap <- TRUE
> summaryMethod <- "Variational"
> summaryParam <- list()
> summaryParam$cyc <- c(20)
> callParam <- list(cores=cores, runtype=runtype)
> mlData <- mlSummarization(slData,
+                 windowMethod =windowMethod,
+                 windowParam  =windowParam,
+                 summaryMethod=summaryMethod,
+                 summaryParam =summaryParam,
+                 callParam    =callParam)

2024-04-30 15:41:42.559103 |   Starting summarization
2024-04-30 15:41:42.559435 |   Computations will take some time, please be patient
2024-04-30 15:41:42.568591 |   Summarizing ...
2024-04-30 15:41:43.106781 |   Summarization done

> names(assayData(mlData))

[1] "intensity" "L_z"       "IC"        "lapla"

> assayData(mlData)$intensity[1:6, 1:5]
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 10.51830 11.02014 10.91918 10.91897 11.01864
[2,] 10.61055 11.37205 11.10092 11.18820 11.12730
[3,] 11.22623 11.17878 11.12928 11.10760 11.17453
[4,] 11.16531 11.12207 11.05674 11.06176 11.13546
[5,] 11.56645 11.52526 11.45202 11.46987 11.54787
[6,] 11.42628 11.35723 11.49604 11.53227 11.43893
```

```
> assayData(mlData)$L_z[1:6, 1:5]
```

```
            [,1]          [,2]        [,3]        [,4]         [,5]
[1,] -0.51098199 -0.0091432086 -0.11010793 -0.11030978 -0.010645099
[2,] -0.56870234  0.1928008487 -0.07833413  0.00894956 -0.051951631
[3,]  0.04697137 -0.0004764791 -0.04997542 -0.07165292 -0.004726317
[4,]  0.04258345 -0.0006536843 -0.06598514 -0.06096191  0.012738223
[5,]  0.04043301 -0.0007622544 -0.07399726 -0.05614699  0.021851713
[6,]  0.01170655 -0.0573518778  0.08146067  0.11768897  0.024348169
```

Next, the summarized data will be segmented in order to identify break points. Therefore we provide a parallelized version of `DNAcopy`.

```
> colnames(assayData(mlData)$L_z) <- sampleNames(mlData)
> segments <- dnaCopySf(
+                 x        =assayData(mlData)$L_z[, 1:10],
+                 chrom    =fData(mlData)$chrom,
+                 maploc   =fData(mlData)$start,
+                 cores    =cores,
+                 smoothing=FALSE)
```

```
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Analyzing: Sample.1
Time difference of 0.1633282 secs
```

```
> head(fData(segments))
```

```
  chrom    start      end num.mark seg.mean individual
1    21 23655900 31642387        7  -0.1324    NA10846
```

```
2     23 47882970 47893260        9  -0.0594    NA10846
3     23 47895066 47942931       14  -0.1885    NA10846
4     21 23655900 31642387        7   0.0072    NA12146
5     23 47882970 47942931       23  -0.1209    NA12146
6     21 23655900 31642387        7  -0.0272    NA12239
```

To get further information, e.g. how to process Affymetrix 500K arrays, please check the followings demos.

```
> demo(package="cn.farms")

Demos in package 'cn.farms':

demo01Snp6            Demo for an Affymetrix SNP6 data set
demo02Snp5            Demo for an Affymetrix SNP5 data set
demo03Snp500k         Demo for an Affymetrix 500K data set
demo04Snp250k         Demo for an Affymetrix 250K NSP data set
demo05Testing         Run the examples
```

The most recent `cn.farms` version can be found at http://www.bioinf.jku.at/software/cnfarms/cnfarms.html.

4 Segmentation

This shows the segmentation with fastseg:

```
require(cn.farms)
require(parallel)
require(fastseg)

## set cores
myCores <- 8

## load the expression-set object for the segmentation
## e.g.: mlData
str(mlData)

for (chrom in 22:1) {
  print(chrom)
  combDataTmp <- mlData[which(fData(mlData)$chrom == chrom), ]
  z2 <- assayData(combDataTmp)$intensity

  cl <- makeCluster(getOption("cl.cores", myCores))
  clusterEvalQ(cl, { require(fastseg) })
  system.time(segRes <- parCapply(cl, as.matrix(z2), fastseg, type=1, alpha=50, minSeg=3))
  stopCluster(cl)
```

```
  nbrOfSamples <- length(sampleNames(mlData))
  resList <- list()
  for (sampIdx in seq_len(nbrOfSamples)) {
    res <- segRes[[sampIdx]]
    seqlevels(res) <- as.character(chrom)
    end(res) <- fData(combDataTmp)$start[end(res)]
    start(res) <- fData(combDataTmp)$start[start(res)]
    values(res)$ID <- sampleNames(mlData)[sampIdx]
    resList[[sampIdx]] <- as.data.frame(res)
  }

  phInf <- fData(combDataTmp)
  save(segRes, phInf, resList, file=paste("segRes_chr", chrom, ".RData", sep=""))
}
```

Alternatively you can use the function "dnaCopySf" from the cn.farms package.

5 Annotation and supported platforms

The `cn.farms` package works with per default works with 250K/500K and SNP6 arrays from Affymetrix. Anyway the package also works with the most recent CytoscanHD array, where you can find the annotation file on our homepage `http://www.bioinf.jku.at/software/cnfarms/cnfarms.html`.

6 Use case: SNP6 data

```
> require(cn.farms)
>
> ## load test data
> require("hapmap500knsp")
/-----------------------------------------\
|         SAMPLE HAPMAP 500K NSP          |
|-----------------------------------------|
| Data obtained from http://www.hapmap.org  |
| This package is meant to be used only for |
| demonstration of BioConductor packages.   |
| Access http://www.hapmap.org for details. |
|-----------------------------------------|
| The contents of this package are provided |
| in good faith and the maintainer does not |
| warrant their accuracy.                   |
\-----------------------------------------/
> require("hapmap500ksty")
/-----------------------------------------\
```

```
|         SAMPLE HAPMAP 500K STY            |
|-------------------------------------------|
| Data obtained from http://www.hapmap.org  |
| This package is meant to be used only for |
| demonstration of BioConductor packages.   |
| Access http://www.hapmap.org for details. |
|-------------------------------------------|
| The contents of this package are provided |
| in good faith and the maintainer does not |
| warrant their accuracy.                   |
\-------------------------------------------/
> celDirNsp <- system.file("celFiles", package="hapmap500knsp")
> celDirSty <- system.file("celFiles", package="hapmap500ksty")
> celFiles <- data.frame(
+         NSP=dir(celDirNsp, full.names=TRUE),
+         STY=dir(celDirSty, full.names=TRUE),
+         stringsAsFactors=FALSE)
> workDir <- tmpdir()
Error: could not find function "tmpdir"
> workDir <- tmpDir()
Error: could not find function "tmpDir"
> workDir <- tempdir()
> dir.create(workDir, showWarnings=FALSE, recursive=TRUE)
> setwd(workDir)
> cores <- 2
> runtype <- "bm"
>
> dir.create("ffObjects/ff", showWarnings=FALSE, recursive=TRUE)
> ldPath(file.path(getwd(), "ffObjects"))
> options(fftempdir=file.path(ldPath(), "ff"))
> createAnnotation(filenames=celFiles$NSP)
Loading required package: DBI
========================================================================
Welcome to oligo version 1.22.0
========================================================================
09:56:45 |   Reading annotation from package pd.mapping250k.nsp 1.8.0
09:56:45 |   Annotation will be saved in /tmp/RtmpdDq8c4/annotation/pd.mapping250k.nsp/1.8.
09:58:13 |   SNP information done
09:58:13 |   Non polymorphic information done
09:58:28 |   Annotation processed
> createAnnotation(filenames=celFiles$STY)
09:58:29 |   Reading annotation from package pd.mapping250k.sty 1.8.0
09:58:29 |   Annotation will be saved in /tmp/RtmpdDq8c4/annotation/pd.mapping250k.sty/1.8.
09:59:36 |   SNP information done
09:59:36 |   Non polymorphic information done
09:59:51 |   Annotation processed
```

```
>
>
> ## normalize the data
> normMethod <- "SOR"
> normDataNsp <- normalizeCels(filenames=celFiles$NSP,
+         method=normMethod, cores=cores, runtype=runtype)
09:59:51 |   Annotation directory:  /tmp/RtmpdDq8c4/annotation/pd.mapping250k.nsp/1.8.0
R Version:  R version 2.15.2 (2012-10-26)

10:00:04 |   Starting normalization
10:00:17 |   Normalization done

Stopping cluster

10:00:18 |   Saving normalized data
> normDataSty <- normalizeCels(filenames=celFiles$STY,
+         method=normMethod, cores=cores, runtype=runtype)
10:00:19 |   Annotation directory:  /tmp/RtmpdDq8c4/annotation/pd.mapping250k.sty/1.8.0
10:00:31 |   Starting normalization
10:00:44 |   Normalization done

Stopping cluster

10:00:45 |   Saving normalized data
>
>
> ## run single-locus FARMS algorithm
> summaryMethod <- "Variational"
> summaryParam <- list()
> summaryParam$cyc <- c(10)
> callParam <- list(cores=cores, runtype=runtype)
>
> slDataNsp <- slSummarization(normDataNsp,
+         summaryMethod=summaryMethod,
+         summaryParam=summaryParam,
+         callParam=callParam,
+         summaryWindow="std")
10:00:47 |   Starting summarization
10:00:47 |   Computations will take some time, please be patient

Library cn.farms loaded.
Library cn.farms loaded in cluster.

10:00:52 |   Summarizing batch 1 ...

Stopping cluster
```

```
10:13:45 |   Summarization done
Time difference of 12.97855 mins
10:13:45 |   Saving data
> slDataSty <- slSummarization(normDataSty,
+         summaryMethod=summaryMethod,
+         summaryParam=summaryParam,
+         callParam=callParam,
+         summaryWindow="std")
10:13:47 |   Starting summarization
10:13:47 |   Computations will take some time, please be patient

Library cn.farms loaded.
Library cn.farms loaded in cluster.

10:13:52 |   Summarizing batch 1 ...

Stopping cluster

10:25:39 |   Summarization done
Time difference of 11.87153 mins
10:25:39 |   Saving data
>
>
> ## combine NSP and STY arrays
> combData <- combineData(slDataNsp, slDataSty, runtype=runtype)
10:25:41 |   Saving normalized data
> fData(combData)[1:10, ]
        chrom   start     end    man_fsetid
962431      1  752566  752566 SNP_A-1909444
682661      1  779322  779322 SNP_A-4303947
56638       1  785989  785989 SNP_A-1886933
2102261     1  792480  792480 SNP_A-2236359
1708871     1  799463  799463 SNP_A-2205441
1331411     1 1003629 1003629 SNP_A-2116190
1796451     1 1097335 1097335 SNP_A-4291020
136779      1 1130727 1130727 SNP_A-1902458
152430      1 1156131 1156131 SNP_A-2131660
761810      1 1158631 1158631 SNP_A-2109914
>
>
> ## multi-loci FARMS
> windowMethod <- "std"
> windowParam <- list()
> windowParam$windowSize <- 5
> windowParam$overlap <- TRUE
```

```
> summaryMethod <- "Variational"
> summaryParam <- list()
> summaryParam$cyc <- c(20)
> callParam <- list(cores=cores, runtype=runtype)
> mlData <- mlSummarization(combData,
+         windowMethod=windowMethod,
+         windowParam=windowParam,
+         summaryMethod=summaryMethod,
+         summaryParam=summaryParam,
+         callParam=callParam)
Slot intensity of assayData is used
10:25:42 |   Starting summarization
10:25:42 |   Computations will take some time, please be patient

Library cn.farms loaded.
Library cn.farms loaded in cluster.

10:25:48 |   Summarizing batch 1 ...

Stopping cluster

10:50:34 |   Summarization done
10:50:34 |   Saving data
>
> head(fData(mlData))
  chrom   start     end    man_fsetid
1     1  752566  799463 SNP_A-2205441
2     1  779322 1003629 SNP_A-2116190
3     1  785989 1097335 SNP_A-4291020
4     1  792480 1130727 SNP_A-1902458
5     1  799463 1156131 SNP_A-2131660
6     1 1003629 1158631 SNP_A-2109914
```

7 Use case: 250K/500K arrays

```
> require(cn.farms)
>
> ## load test data
> require("hapmap500knsp")
/-----------------------------------------\
|         SAMPLE HAPMAP 500K NSP          |
|-----------------------------------------|
| Data obtained from http://www.hapmap.org  |
| This package is meant to be used only for |
| demonstration of BioConductor packages.   |
```

```
| Access http://www.hapmap.org for details. |
|-------------------------------------------|
| The contents of this package are provided |
| in good faith and the maintainer does not |
| warrant their accuracy.                   |
\-------------------------------------------/
> require("hapmap500ksty")
/-------------------------------------------\
|         SAMPLE HAPMAP 500K STY            |
|-------------------------------------------|
| Data obtained from http://www.hapmap.org  |
| This package is meant to be used only for |
| demonstration of BioConductor packages.   |
| Access http://www.hapmap.org for details. |
|-------------------------------------------|
| The contents of this package are provided |
| in good faith and the maintainer does not |
| warrant their accuracy.                   |
\-------------------------------------------/
> celDirNsp <- system.file("celFiles", package="hapmap500knsp")
> celDirSty <- system.file("celFiles", package="hapmap500ksty")
> celFiles <- data.frame(
+         NSP=dir(celDirNsp, full.names=TRUE),
+         STY=dir(celDirSty, full.names=TRUE),
+         stringsAsFactors=FALSE)
> workDir <- tmpdir()
Error: could not find function "tmpdir"
> workDir <- tmpDir()
Error: could not find function "tmpDir"
> workDir <- tempdir()
> dir.create(workDir, showWarnings=FALSE, recursive=TRUE)
> setwd(workDir)
> cores <- 2
> runtype <- "bm"
>
> dir.create("ffObjects/ff", showWarnings=FALSE, recursive=TRUE)
> ldPath(file.path(getwd(), "ffObjects"))
> options(fftempdir=file.path(ldPath(), "ff"))
> createAnnotation(filenames=celpaste("Please load the mlData object!")Files$NSP)
Loading required package: DBI
========================================================================
Welcome to oligo version 1.22.0
========================================================================
09:56:45 |   Reading annotation from package pd.mapping250k.nsp 1.8.0
09:56:45 |   Annotation will be saved in /tmp/RtmpdDq8c4/annotation/pd.mapping250k.nsp/1.8.
09:58:13 |   SNP information done
```

```
09:58:13 |   Non polymorphic information done
09:58:28 |   Annotation processed
> createAnnotation(filenames=celFiles$STY)
09:58:29 |   Reading annotation from package pd.mapping250k.sty 1.8.0
09:58:29 |   Annotation will be saved in /tmp/RtmpdDq8c4/annotation/pd.mapping250k.sty/1.8.
09:59:36 |   SNP information done
09:59:36 |   Non polymorphic information done
09:59:51 |   Annotation processed
>
>
> ## normalize the data
> normMethod <- "SOR"
> normDataNsp <- normalizeCels(filenames=celFiles$NSP,
+         method=normMethod, cores=cores, runtype=runtype)
09:59:51 |   Annotation directory:  /tmp/RtmpdDq8c4/annotation/pd.mapping250k.nsp/1.8.0
R Version:  R version 2.15.2 (2012-10-26)

snowfall 1.84 initialized (using snow 0.3-10): parallel execution on 2 CPUs.

Library cn.farms loaded.
Library cn.farms loaded in cluster.

Library affxparser loaded.
Library affxparser loaded in cluster.

Library oligo loaded.
Library oligo loaded in cluster.

10:00:04 |   Starting normalization
10:00:17 |   Normalization done

Stopping cluster

10:00:18 |   Saving normalized data
> normDataSty <- normalizeCels(filenames=celFiles$STY,
+         method=normMethod, cores=cores, runtype=runtype)
10:00:19 |   Annotation directory:  /tmp/RtmpdDq8c4/annotation/pd.mapping250k.sty/1.8.0
snowfall 1.84 initialized (using snow 0.3-10): parallel execution on 2 CPUs.

Library cn.farms loaded.
Library cn.farms loaded in cluster.

Library affxparser loaded.
Library affxparser loaded in cluster.

Library oligo loaded.
```

```
Library oligo loaded in cluster.

10:00:31 |   Starting normalization
10:00:44 |   Normalization done

Stopping cluster

10:00:45 |   Saving normalized data
>
>
> ## run single-locus FARMS algorithm
> summaryMethod <- "Variational"
> summaryParam <- list()
> summaryParam$cyc <- c(10)
> callParam <- list(cores=cores, runtype=runtype)
>
> slDataNsp <- slSummarization(normDataNsp,
+         summaryMethod=summaryMethod,
+         summaryParam=summaryParam,
+         callParam=callParam,
+         summaryWindow="std")
10:00:47 |   Starting summarization
10:00:47 |   Computations will take some time, please be patient
snowfall 1.84 initialized (using snow 0.3-10): parallel execution on 2 CPUs.

Library cn.farms loaded.
Library cn.farms loaded in cluster.

10:00:52 |   Summarizing batch 1 ...

Stopping cluster

10:13:45 |   Summarization done
Time difference of 12.97855 mins
10:13:45 |   Saving data
> slDataSty <- slSummarization(normDataSty,
+         summaryMethod=summaryMethod,
+         summaryParam=summaryParam,
+         callParam=callParam,
+         summaryWindow="std")
10:13:47 |   Starting summarization
10:13:47 |   Computations will take some time, please be patient
snowfall 1.84 initialized (using snow 0.3-10): parallel execution on 2 CPUs.

Library cn.farms loaded.
Library cn.farms loaded in cluster.
```

```
10:13:52 |   Summarizing batch 1 ...

Stopping cluster

10:25:39 |   Summarization done
Time difference of 11.87153 mins
10:25:39 |   Saving data
>
>
> ## combine NSP and STY arrays
> combData <- combineData(slDataNsp, slDataSty, runtype=runtype)
10:25:41 |   Saving normalized data
> fData(combData)[1:10, ]
        chrom   start     end    man_fsetid
962431      1  752566  752566 SNP_A-1909444
682661      1  779322  779322 SNP_A-4303947
56638       1  785989  785989 SNP_A-1886933
2102261     1  792480  792480 SNP_A-2236359
1708871     1  799463  799463 SNP_A-2205441
1331411     1 1003629 1003629 SNP_A-2116190
1796451     1 1097335 1097335 SNP_A-4291020
136779      1 1130727 1130727 SNP_A-1902458
152430      1 1156131 1156131 SNP_A-2131660
761810      1 1158631 1158631 SNP_A-2109914
>
>
> ## multi-loci FARMS
> windowMethod <- "std"
> windowParam <- list()
> windowParam$windowSize <- 5
> windowParam$overlap <- TRUE
> summaryMethod <- "Variational"
> summaryParam <- list()
> summaryParam$cyc <- c(20)
> callParam <- list(cores=cores, runtype=runtype)
> mlData <- mlSummarization(combData,
+         windowMethod=windowMethod,
+         windowParam=windowParam,
+         summaryMethod=summaryMethod,
+         summaryParam=summaryParam,
+         callParam=callParam)
Slot intensity of assayData is used
10:25:42 |   Starting summarization
10:25:42 |   Computations will take some time, please be patient
snowfall 1.84 initialized (using snow 0.3-10): parallel execution on 2 CPUs.
```

```
Library cn.farms loaded.
Library cn.farms loaded in cluster.

10:25:48 |   Summarizing ...

Stopping cluster

10:50:34 |   Summarization done
10:50:34 |   Saving data
>
> head(fData(mlData))
  chrom   start     end    man_fsetid
1     1  752566  799463 SNP_A-2205441
2     1  779322 1003629 SNP_A-2116190
3     1  785989 1097335 SNP_A-4291020
4     1  792480 1130727 SNP_A-1902458
5     1  799463 1156131 SNP_A-2131660
6     1 1003629 1158631 SNP_A-2109914
```

8 Setup

This vignette was built on:

```
> sessionInfo()

R version 4.4.0 Patched (2024-04-24 r86482)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.6.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] DNAcopy_1.77.0      cn.farms_1.51.0     snow_0.4-4
```

```
[4] oligoClasses_1.65.0 ff_4.0.12           bit_4.0.5
[7] Biobase_2.63.1      BiocGenerics_0.49.1

loaded via a namespace (and not attached):
 [1] Matrix_1.7-0                preprocessCore_1.65.0
 [3] jsonlite_1.8.8              compiler_4.4.0
 [5] BiocManager_1.30.22         crayon_1.5.2
 [7] SummarizedExperiment_1.33.3 blob_1.2.4
 [9] affxparser_1.75.2           GenomicRanges_1.55.4
[11] Biostrings_2.71.6           parallel_4.4.0
[13] splines_4.4.0               IRanges_2.37.1
[15] fastmap_1.1.1               lattice_0.22-6
[17] R6_2.5.1                    XVector_0.43.1
[19] S4Arrays_1.3.7              GenomeInfoDb_1.39.14
[21] iterators_1.0.14            DelayedArray_0.29.9
[23] MatrixGenerics_1.15.1       GenomeInfoDbData_1.2.12
[25] DBI_1.2.2                   rlang_1.1.3
[27] affyio_1.73.0               cachem_1.0.8
[29] bit64_4.0.5                 memoise_2.0.1
[31] cli_3.6.2                   RSQLite_2.3.6
[33] SparseArray_1.3.7           zlibbioc_1.49.3
[35] foreach_1.5.2               grid_4.4.0
[37] vctrs_0.6.5                 S4Vectors_0.41.7
[39] codetools_0.2-20            abind_1.4-5
[41] stats4_4.4.0                httr_1.4.7
[43] oligo_1.67.0                matrixStats_1.3.0
[45] tools_4.4.0                 UCSC.utils_0.99.7
```

References

Bengtsson, H., Irizarry, R., Carvalho, B., and Speed, T. P. (2008). Estimation and assessment of raw copy numbers at the single locus level. *Bioinformatics*, **24**(6), 759–767.

Clevert, D.-A., Mitterecker, A., Mayr, A., Klambauer, G., Tuefferd, M., Bondt, A. D., Talloen, W., Göhlmann, H., and Hochreiter, S. (2011). cn.FARMS: a latent variable model to detect copy number variations in microarray data with a low false discovery rate. *Nucleic Acids Research*.

Hochreiter, S., Clevert, D.-A., and Obermayer, K. (2006). A new summarization method for Affymetrix probe level data. *Bioinformatics*, **22**(8), 943–949.

Nannya, Y., Sanada, M., Nakazaki, K., Hosoya, N., Wang, L., Hangaishi, A., Kurokawa, M., Chiba, S., Bailey, D. K., Kennedy, G. C., *et al.* (2005). A robust algorithm for copy number detection using high-density oligonucleotide single nucleotide polymorphism genotyping arrays. *Cancer Research*, **65**(14), 6071–6079.

Talloen, W., Clevert, D.-A., Hochreiter, S., Amaratunga, D., Bijnens, L., Kass, S., and Göhlmann, H. W. H. (2007). I/NI-calls for the exclusion of non-informative genes: a highly effective feature filtering tool for microarray data. *Bioinformatics*, **23**(21), 2897–2902.

Talloen, W., Hochreiter, S., Bijnens, L., Kasim, A., Shkedy, Z., and Amaratunga, D. (2010). Filtering data from high-throughput experiments based on measurement reliability. *Proc. Natl. Acad. Sci. U S A*, **107**(46), 173–174.