Parent-specific copy-number segmentation using Paired PSCBS

Henrik Bengtsson

June 27, 2017

Abstract

The Paired Parent-Specific Circular Binary Segmentation (Paired PSCBS) method partitions a tumor genome into segments of constant parent-specific copy numbers (PSCNs) based on SNP DNA microarray data from a matched tumor-normal pair. The method also calls segments with run of homozygosity (ROH), segments in allelic balance (AB) and segments with loss of heterozygosity (LOH). Paired PSCBS was designed to work with data from any SNP microarray technology and generation, including Affymetrix and Illumina.

This document shows how to use the *PSCBS* package to run Paired PSCBS on a tumor-normal pair.

Keywords: copy numbers, allele specific, parent specific, genomic aberrations

This vignette is distributed as part of the PSCBS package, which is available on CRAN (`https://cran.r-project.org/`). The authors very much appreciate feedback on this document.

Contents

1 Background

We will here use a small example data set to illustrate how to setup the data in a format suitable for Paired PSCBS, how to identify segments, how to call them, and how to plot and export the segmentation results. The statistical model and the algorithm behind Paired PSCBS is explained in detail in Olshen *et al.* (2011).

2 Preparing data to be segmented

The Paired PSCBS (Olshen *et al.*, 2011) method requires tumor-normal paired parent-specific copy-number (PSCN) signals. More precisely, it requires total copy-number (TCN) estimates for the tumor relative to the matched normal (C_T), allele B fractions (BAFs) for the tumor (β_T) and BAFs for the matched normal (β_N). The genomic locations of the loci in form of chromosome and physical position are also required.

2.1 Locus-level SNP copy-number signals

In this example we will use a small example data set part of the *PSCBS* package. It can be loaded as:

```
> data <- PSCBS::exampleData("paired.chr01")
> str(data)
'data.frame': 73346 obs. of  6 variables:
 $ chromosome: int  1 1 1 1 1 1 1 1 1 1 ...
 $ x         : int  1145994 2224111 2319424 2543484 2926730 2941694 3084986 3155127..
 $ CT        : num  1.625 1.071 1.406 1.18 0.856 ...
 $ betaT     : num  0.757 0.771 0.834 0.778 0.229 ...
 $ CN        : num  2.36 2.13 2.59 1.93 1.71 ...
 $ betaN     : num  0.827 0.875 0.887 0.884 0.103 ...
```

In additional to the mandatory fields (`chromosome`, `x`, `CT`, `betaT`, and `betaN`), this data set also contains TCNs for normal (`CN`) relative to a large pool of normal samples. The latter will not be used here.

2.2 Dropping TCN outliers

There may be some outliers among the TCNs. In CBS (Olshen *et al.*, 2004; Venkatraman and Olshen, 2007), the authors propose a method for identifying outliers and then to shrink such values toward their neighbors ("smooth") before performing segmentation. At the time CBS was developed it made sense to not just to drop outliers because the resolution was low and every datapoint was valuable. With modern technologies the resolution is much higher and we can afford dropping such outliers, which can be done by:

```
> data <- dropSegmentationOutliers(data)
```

Dropping TCN outliers is optional.

3 Paired PSCBS segmentation

3.1 Skipping centromeres and other large gaps

Like the CBS method, Paired PSCBS does not take the physical locations (in units of nucleotides) of the loci in to account when segmenting the data, only their relative ordering along the genome.

This means that after having ordered the loci along genome, it will treat two "neighboring" loci that are on both sides of the centromere equally to two neighboring loci that are only a few hundred bases apart. This may introduce erroneous change points that appear to be inside the centromere. The same issues occur for other large gaps of the genome where there are no observed signals.

To avoid this, although not mandatory, we will locate all gaps of the genome where there are no observered loci. As a threshold we will consider a region to be a "gap" if the distance between the two closest loci is greater than 1Mb.

```
> gaps <- findLargeGaps(data, minLength = 1e+06)
> gaps
  chromosome     start       end   length
1          1 120992604 141510002 20517398
```

which shows that there is a 20.5Mb long gap between 121.0Mb and 141.5Mb on Chromosome 1. This is the centromere of Chromosome 1. It is not possible to specify "gaps" to the segmentation function. Instead they need to be given as part of a set of "known" segments, which is done as:

```
> knownSegments <- gapsToSegments(gaps)
> knownSegments
  chromosome    start      end   length
1          1     -Inf 1.21e+08      Inf
2          1 1.21e+08 1.42e+08 20517398
3          1 1.42e+08      Inf      Inf
```

Below, we will use this to tell Paired PSCBS to segment Chromosome 1 into three independent segments, where the first segment is from the beginning of the chromosome (hence '-Inf') to 120.1Mb, the second one from 120.1-141.5Mb (the above gap), and the third one is from 141.5Mb to the end of the chromosome (hence '+Inf'). Just as Paired PSCBS segments chromosomes independently of each other, it also segments priorly known segments independently of each other. Specifying known segments is optional.

3.2 Identifying PSCN segments

We are now ready to segment the locus-level PSCN signals. This is done by[1]:

```
> fit <- segmentByPairedPSCBS(data, knownSegments = knownSegments,
+     preserveScale = FALSE, seed = 48879, verbose = -10)
```

Note that this may take several minutes when applied to whole-genome data. The above call will also normalize the tumor BAFs using the TumorBoost normalization method (Bengtsson *et al.*, 2010) (without preserving the relative scale for homozygous and heterozygous BAFs; in a future version, this will be the default). If this has already been done or the tumor signals have been normalized by other means, the TumorBoost step can be skipped by setting argument `tbn=FALSE`.

The result of the segmentation is a set of segments identified to have the same underlying PSCN levels. In this particular case, 11 PSCN segments were found:

```
> getSegments(fit, simplify = TRUE)
   chromosome tcnId dhId    start      end tcnNbrOfLoci tcnMean tcnNbrOfSNPs
1           1     1    1 5.54e+05 4.28e+06          687    1.40          169
2           1     1    2 4.28e+06 4.27e+07        11406    1.38         3260
3           1     1    3 4.27e+07 1.20e+08        25159    1.38         6657
```

[1] We fix the random seed in order for the results of this vignette to be numerically reproducible.

```
4           1     1    4 1.20e+08 1.20e+08          72    1.47            8
5           1     1    5 1.20e+08 1.21e+08         171    1.44           52
6           1     2    1 1.21e+08 1.42e+08           0      NA            0
7           1     3    1 1.42e+08 1.86e+08       13434    2.07         3770
8           1     4    1 1.86e+08 1.99e+08        4018    2.71         1271
9           1     5    1 1.99e+08 2.07e+08        2755    2.59          784
10          1     6    1 2.07e+08 2.07e+08          14    3.87            9
11          1     7    1 2.07e+08 2.47e+08       15581    2.64         4492
   tcnNbrOfHets dhNbrOfLoci dhMean c1Mean c2Mean
1           169         169 0.4145  0.411  0.992
2          3260        3260 0.4944  0.348  1.030
3          6657        6657 0.5205  0.332  1.052
4             8           8 0.0767  0.679  0.792
5            52          52 0.5123  0.351  1.089
6             0          NA     NA     NA     NA
7          3770        3770 0.0943  0.935  1.130
8          1271        1271 0.2563  1.007  1.701
9           784         784 0.2197  1.009  1.577
10            9           9 0.2769  1.400  2.472
11         4492        4492 0.2290  1.017  1.621
```

Note that Segment #6 has no mean-level estimates. It is because it corresponds to the centromere (the gap) that was identified above. Paired PSCBS did indeed try to segment it, but since there are no data points, all estimates are missing values. Similarly, for Segment # the DH and minor and major CNs mean estimates are all missing values. This is because, Paired PSCBS identified that segment by first segmenting the TCN signals by themselves, and after which it tried to segment the DH signals within that segment. Since there are no heterozygous SNPs in the segment, there are no DH signals, and hence there is no DH mean estimate.

3.3 Displaying genomic PSCN profiles

To plot the PSCN segmentation results, do:

```
plotTracks(fit)
```

which by default displays three panels containing TCN, decrease of heterozygosity (DH), and minor and major CNs as in Figure 1. To plot only one panel with TCN and minor and major CNs and zoom in on a particular region, do (not shown):

```
plotTracks(fit, tracks="tcn,c1,c2", xlim=c(120,244)*1e6)
```

4 Calling segments

The calling algorithms for allelic balance (AB) and loss of heterozygosity (LOH) are based on quantile estimates of the different mean levels. These estimates are obtained using non-parametric bootstrap techniques. For more details, see Olshen *et al.* (2011). After the Paired PSCBS method was published, we have added methods for calling run of homozygosity (ROH) and neutral total copy number (NTCN). *The ROH and NTCN calling methods should be considered under development until further notice, meaning they and their results may change without notice.*

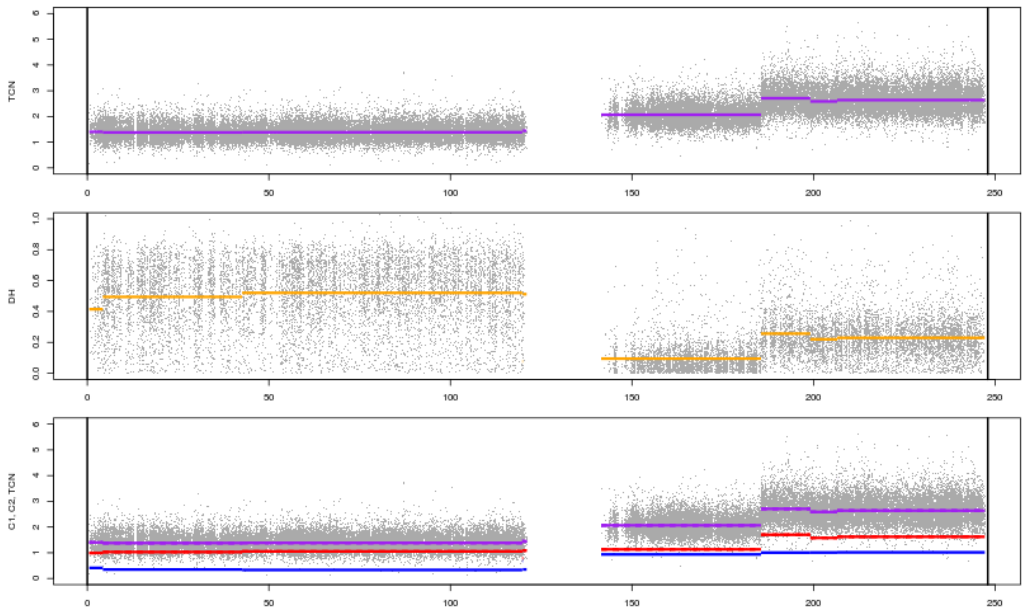

Figure 1: PSCN segments identified by Paired PSCBS. **Top**: The TCN signals with the TCN mean levels (purple). **Middle**: The DH signals with the DH mean levels (orange). **Bottom**: The TCN signals with the minor CN (C_1; blue), the major CN (C_2; red) and the TCN ($C = C_1 + C_2$; purple) mean levels.

4.1 Calling segments with run of homozygosity (ROH)*

Please note that this method is under development. This means that it may change without further notice.

A region has a run of homozygotes (ROH) if all of its SNPs are homozygous (in the normal). Since such a region has no heterozyous SNPs, its decrease in heterozygosity (DH) is undefined. Likewise, the minor and major copy numbers are unknown. However, if there are genotyping errors within an ROH region, we will obtain a non-missing DH mean level and hence also finite minor and major CNs. In order to adjust for these faulty estimates, we test if the identified segments are ROHs or not by:

```
> fit <- callROH(fit, verbose = -10)
```

This will also set the corresponding DH and minor and major CN mean levels to NA. The total CN mean levels are not affected by the ROH call.

4.1.1 Tuning parameters

For each segment, the test for ROH calculates the fraction of SNPs that are called heterozygous. If the fraction of heterozygotes is smaller than a threshold, the segmented is called ROH, otherwise not. The default threshold is 1/12. To use a different threshold, set argument `delta` to a scalar in $[0, 1]$. For example, using `callROH(fit, delta=1/30)` makes the ROH caller more conservative.[2]

[2]It is our plan to replace this basic test with a bionomial test.

4.2 Calling segments in allelic balance (AB)

The AB caller tests whether the DH level of a segment is small enough to be considered zero, which means that the minor and the major CNs are equal, i.e. the segment is in allelic balance. To call the AB state of all segments, do:

```
> fit <- callAB(fit, verbose = -10)
```

Because the caller utilizes bootstrapping techniques, calling AB may take some time if there is a large number of segments. Segments already called ROH will not be called for AB, and their AB statuses will have a missing value (`NA`).

4.2.1 Tuning parameters

The AB caller tests whether a segment's DH is zero or not, by comparing its DH level (or more precisely, the 5% quantile of its bootstrapped DH mean level) to a threshold. This threshold will be a function of the noise level, because the noiser the BAF signals (and hence the DH signals), the greater the bias of the DH mean level for segments in AB will be. Because of this, the threshold is choosen from data by estimating the noise level of the DH signals near zero. Further rationales and details are given in Olshen *et al.* (2011). The AB threshold can be estimated explicitly and used in the caller as

```
deltaAB <- estimateDeltaAB(fit, scale=1)
fit <- callAB(fit, delta=deltaAB)
```

By decreasing argument `scale` (a positive scalar), a smaller threshold will be obtained resulting in a more conservative AB caller.

4.3 Calling segments with loss of heterozygosity (LOH)

The LOH caller tests whether the minor CN level of a segment is large enough to be considered non-zero, which means that the segment is *not* in LOH. To call the LOH state of all segments, do:

```
> fit <- callLOH(fit, verbose = -10)
```

Note that in order to call LOH, one has to call allelic balance first. Since the bootstrapping was already done in the AB caller, it is not repeated here, which is why calling LOH is faster than calling AB. Analogously to the AB caller, segments already called ROH will not be called for LOH, and their LOH statuses will have a missing value (`NA`). Segments already called AB will be called non-LOH, because AB and LOH are exclusively mutual states.

4.3.1 Tuning parameters

The LOH caller tests whether a segment's minor CN is non-zero or not, by comparing its minor CN level (or more precisely, the 95% quantile of its bootstrapped minor CN mean level) to a threshold. This threshold will, among other components, be a function of normal contamination, i.e. the greater the fraction of normal cells is the greater the threshold needs to be in order to call LOH in the tumor cells. Because of this, the threshold is choosen from data as the midpoint of the estimated levels of minor CNs zero and one, which in turn is a function of the normal contamination. For details on the definition of LOH and details on the LOH caller, see Olshen *et al.* (2011). The LOH threshold can be estimated explicitly and used as:

```
deltaLOH <- estimateDeltaLOH(fit, midpoint=1/2)
fit <- callLOH(fit, delta=deltaLOH)
```

By decreasing argument `midpoint` in $[0, 1]$, a smaller threshold will be obtained resulting in a more conservative LOH caller.

4.4 Calling segments with neutral total copy number (NTCN)*

Please note that this method is under development. This means that it may change without further notice.

The neutral total copy number (NTCN) caller tests whether the total CN level of a segment is neutral (e.g. diploid) or not. To call the NTCN state of all segments, do:

```
> fit <- callNTCN(fit, verbose = -10)
```

4.4.1 Tuning parameters

The NTCN caller identifies segments which TCN mean levels (or more precisely, which 95% confidence intervals) are within a given "acceptance" region. This acceptance region is determined by the 95% confidence interval of an initial set of AB segments identified to be copy neutral and then expanded by half a TCN unit length. The true length of half a total copy number unit is specified by the argument `delta` to `callNTCN()`. Its length should be a function of the overall background signals (which includes normal contamination and more), such that the width of the acceptance region becomes smaller when the background increases. The background signal (`kappa`) and `delta` can be estimated explicitly as:

```
kappa <- estimateKappa(fit)
deltaCN <- estimateDeltaCN(fit, scale=1, kappa=kappa)
fit <- callNTCN(fit, delta=deltaCN, verbose=-10)
```

By decreasing the tuning parameter `scale` (a positive scalar), a smaller acceptance region will be obtained, which results in a more conservative CN caller.

4.5 Results from calling ROH, AB, LOH and NTCN

All calls are appended to the segmentation results as logical columns:

```
> getSegments(fit, simplify = TRUE)
   chromosome tcnId dhId    start      end tcnNbrOfLoci tcnMean tcnNbrOfSNPs
1           1     1    1 5.54e+05 4.28e+06          687    1.40          169
2           1     1    2 4.28e+06 4.27e+07        11406    1.38         3260
3           1     1    3 4.27e+07 1.20e+08        25159    1.38         6657
4           1     1    4 1.20e+08 1.20e+08           72    1.47            8
5           1     1    5 1.20e+08 1.21e+08          171    1.44           52
6           1     2    1 1.21e+08 1.42e+08            0      NA            0
7           1     3    1 1.42e+08 1.86e+08        13434    2.07         3770
8           1     4    1 1.86e+08 1.99e+08         4018    2.71         1271
9           1     5    1 1.99e+08 2.07e+08         2755    2.59          784
10          1     6    1 2.07e+08 2.07e+08           14    3.87            9
11          1     7    1 2.07e+08 2.47e+08        15581    2.64         4492
   tcnNbrOfHets dhNbrOfLoci dhMean c1Mean c2Mean rohCall abCall lohCall ntcnCall
1           169         169 0.4145  0.411  0.992   FALSE  FALSE    TRUE    FALSE
2          3260        3260 0.4944  0.348  1.030   FALSE  FALSE    TRUE    FALSE
3          6657        6657 0.5205  0.332  1.052   FALSE  FALSE    TRUE    FALSE
4             8           8     NA     NA     NA    TRUE     NA      NA    FALSE
5            52          52 0.5123  0.351  1.089   FALSE  FALSE    TRUE    FALSE
6             0          NA     NA     NA     NA      NA     NA      NA       NA
7          3770        3770 0.0943  0.935  1.130   FALSE   TRUE   FALSE     TRUE
8          1271        1271 0.2563  1.007  1.701   FALSE  FALSE   FALSE    FALSE
9           784         784 0.2197  1.009  1.577   FALSE  FALSE   FALSE    FALSE
```

```
10            9           9 0.2769  1.400  2.472   FALSE  FALSE   FALSE    FALSE
11         4492        4492 0.2290  1.017  1.621   FALSE  FALSE   FALSE    FALSE
```

5 Saving results

5.1 Writing segments to a tab-delimited text file

To write the PSCN segmentation results to file, do:

```
pathname <- writeSegments(fit, name="MySample", simplify=TRUE)
```

With `simplify=FALSE` (default) quantile estimates of the different mean levels will also be written, which roughly doubles the file size.

6 Experimental

In this section we illustrate some of the ongoing and future work that will be contained in the PSCBS package. Please be aware that these methods are very much under construction, possibly incomplete and in the worst case, even incorrect.

6.1 Less biased Decrease of Heterozygosity (DH) estimates

The DH mean levels of the segments are estimated as the sample mean of the DH signals within each segment. Since DH signals are by definition truncated at zero, the mean level estimates will always be greater than zero even if the true underlying DH is exactly zero. An additional bias is introduced because the distribution of the DH signals is skewed for small DHs, causing the sample mean estimate to be biased. A less biased estimate can be obtained by using the median estimator. To use the median estimator for DH mean levels, specify argument `avgDH="median"` to the `segmentByPairedPSCBS()` call. Because the DH mean levels will be less biased, the (C_1, C_2) mean level estimates will also be less biased, e.g. they will be closer to each other for segments that are in allelic balance.

6.2 Pruning segmentation profile

By applying hierarchical clustering on the segment means, it is possible to prune the PSCN profile such that change points with very small absolute changes are dropped. If change points are dropped this way, this results in a smaller number of segments, which are hence longer on average. To prune Paired PSCBS segmentation results this way, do:

```
> fitP <- pruneByHClust(fit, h = 0.25, verbose = -10)
```

The result of this is shown in Figure 2. In the current implementation, any segment calls and quantile mean levels estimates previously are dropped when pruning.

6.3 Report generation

A multipage PDF report that contains both whole-genome and per-chromosome summaries and figures can be generated by:

```
> report(fit, sampleName="PairedPSCBS", studyName="PSCBS-Ex", verbose=-10)
```

By default, the reports are written to directory `reports/<studyName>/` under the current working directory. In addition to the PDF, that directory also contains subdirectory `figures/` holding all generated figure files (e.g. PNGs and PDFs) for easy inclusion elsewhere.

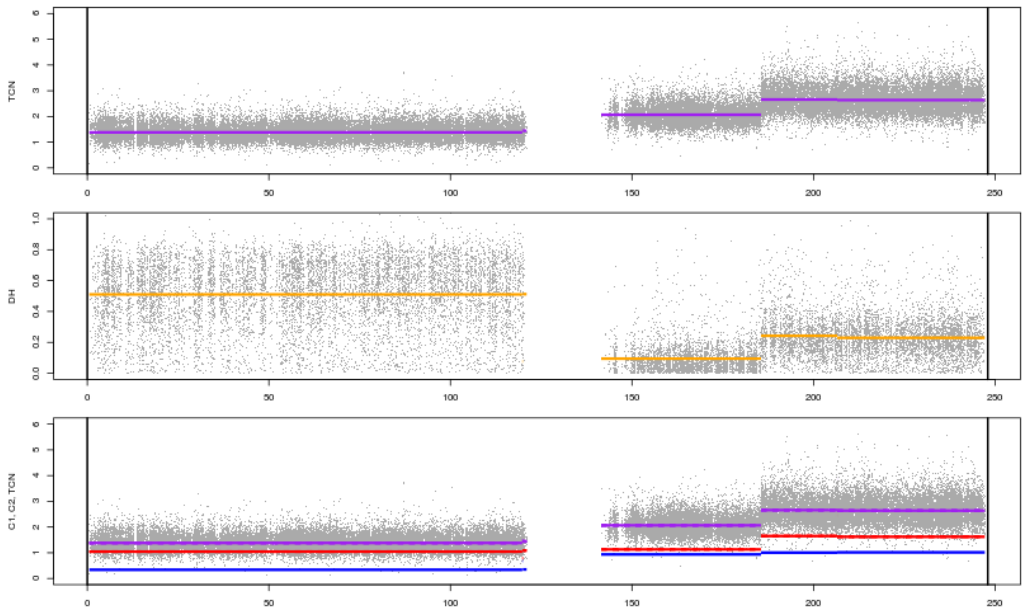

Figure 2: Pruned PSCN segments plotted as in Figure 1.

References

Bengtsson, H., Neuvial, P., and Speed, T. P. (2010). TumorBoost: Normalization of allele-specific tumor copy numbers from a single pair of tumor-normal genotyping microarrays. *BMC Bioinformatics*, **11**(1), 245.

Olshen, A. B., Venkatraman, E. S., Lucito, R., and Wigler, M. (2004). Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, **5**(4), 557–572.

Olshen, A. B., Bengtsson, H., Neuvial, P., Spellman, P., Olshen, R. A., and Seshan, V. E. (2011). Parent-specific copy number in paired tumor-normal studies using circular binary segmentation. *Bioinformatics*, **27**(15), 2038–2046.

Venkatraman, E. S. and Olshen, A. B. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, **23**(6), 657–663.

Appendix

Session information

- R version 3.4.0 (2017-04-21), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.2 LTS`
- Matrix products: default
- BLAS: `/usr/lib/atlas-base/atlas/libblas.so.3.0`
- LAPACK: `/usr/lib/atlas-base/atlas/liblapack.so.3.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: DNAcopy 1.50.1, PSCBS 0.63.0, R.devices 2.15.1-9000, R.methodsS3 1.7.1, R.oo 1.21.0, R.utils 2.5.0-9000
- Loaded via a namespace (and not attached): Cairo 1.5-9, Formula 1.2-1, Hmisc 4.0-3, Matrix 1.2-10, R.cache 0.12.0, R.rsp 0.41.0-9000, RColorBrewer 1.1-2, Rcpp 0.12.11, acepack 1.4.1, aroma.light 3.6.0, backports 1.1.0, base64enc 0.1-4, checkmate 1.8.2, cluster 2.0.6, codetools 0.2-15, colorspace 1.3-2, compiler 3.4.0, data.table 1.10.4, digest 0.6.12, foreign 0.8-69, future 1.5.0, ggplot2 2.2.1, globals 0.10.0, grid 3.4.0, gridExtra 2.2.1, gtable 0.2.0, htmlTable 1.9, htmltools 0.3.6, htmlwidgets 0.8, knitr 1.16, lattice 0.20-35, latticeExtra 0.6-28, lazyeval 0.2.0, listenv 0.6.0, magrittr 1.5, matrixStats 0.52.2-9000, munsell 0.4.3, nnet 7.3-12, parallel 3.4.0, plyr 1.8.4, rlang 0.1.1.9000, rpart 4.1-11, scales 0.4.1, splines 3.4.0, stringi 1.1.5, stringr 1.2.0, survival 2.41-3, tibble 1.3.3, tools 3.4.0

This report was automatically generated using `rfile()` of the R.rsp package. Total processing time after RSP-to-R translation was 30.27 secs.