Creating Annotation Packages for RnBeads

Yassen Assenov, Fabian Müller, Pavlo Lutsik

Contact: rnbeads@mpi-inf.mpg.de

Package version: 0.99.0

March 10, 2018

RnBeads is an R package for comprehensive analysis of genome-wide DNA methylation data with single-CpG resolution. It relies on annotation packages – one for every supported genome assembly. RnBeadsAnnotationCreator facilitates the generation of annotation packages for RnBeads.

This vignette is a valuable resource for researchers who wish to apply RnBeads to currently unsupported genomes. It describes the overall structure of annotation packages for RnBeads and how RnBeadsAnnotationCreator (also referred to as *annotation creator*) automatizes the process of creating and validating a new annotation package.

Contents

1 Anatomy of an Annotation Package

The RnBeads analysis pipeline depends on an annotation package, dedicated to a specific genome assembly. The name of every annotation package consists of the prefix *RnBeads.*, followed by an assembly code in lower case. For example, RnBeads needs the package RnBeads.hg19 in order to provide analysis of methylome data on the human genome hg19.

Every annotation package contains exclusively R data structures of three types, along with their documentation. As an example, Table 1 shows all RData files in the data subdirectory of the package RnBeads.hg19, as well as the objects they contain. The different data types are described in details in the following paragraphs.

1.1 Site Annotation Tables

A site annotation table lists all genomic sites that are theoretically targeted by a platform. For example, every annotation package contains the CpG site annotation table which lists all CpG dinucleotides in the dedicated genome assembly[1]. This table is in the form of a GRangesList object, containing one GRanges instance per chromosome. Genomic coordinates are 1-based and sites are sorted based on their genomic position. Every site annotation table contains the following metadata columns:

CpG
: Number of CpG dinucleotides in a window of length 100 base pairs, centered at the site's location. This is an integer value between 0 and 50.

GC
: Number of C and G nucleotides in a window of length 100 base pairs, centered at the site's location. This is an integer value between 0 and 100.

CGI Relation
: Relationship of the respective site to (its closest) CpG island. This is a factor with values among the following levels: "Open Sea", "Shelf", "Shore" and "Island".

Annotation tables could contain additional metadata. For example, the CpG site annotation table in RnBeads.hg19 contains also the following metadata columns:

SNPs
: Identifiers of all prefiltered dbSNP records that overlap with the respective site. This is a character value containing a comma-separated list of identifiers, or NA if no dbSNP records overlap with the respective site.

HumanMethylation27 (human assemblies only)
: Index (within the same chromosome) of the probe in the Infinium HumanMethylation27 Bead Chip assay that targets the respective CpG dinucleotide. This is a positive integer value, or NA if the CpG dinucleotide is not covered by this platform.

HumanMethylation450 (human assemblies only)
: Index (within the same chromosome) of the probe in the Infinium HumanMethylation450

[1] More precisely, in the predefined list of supported chromosomes for the dedicated assembly.

File	Object	Type	Description
hg19.sites	sites$sites	GRangesList	CpG site annotation table.
	sites$mappings	list	Mappings from built-in regions to sites.
hg19.probes27	sites$sites	GRangesList	Probe annotation table.
	sites$mappings	list	Mappings from built-in regions to probes.
	sites$controls27	data.frame	Control probe annotation table.
hg19.probes450	sites$sites	GRangesList	Probe annotation table.
	sites$mappings	list	Mappings from built-in regions to probes.
	sites$controls450	data.frame	Control probe annotation table.
hg19.regions	GENOME	character	Bioconductor package with genomic sequence.
	CHROMOSOMES	character	Names of all supported chromosomes.
	regions$tiling	GRangesList	Tiling region annotation table.
	regions$genes	GRangesList	Gene body annotation table.
	regions$promoters	GRangesList	Gene promoter annotation table.
	regions$cpgislands	GRangesList	CpG island annotation table.
small.example.object	rnb.set.example	RnBeadRawSet	Example Infinium 450k dataset.

Table 1: Metadata accompanying the HumanMethylation450 probe definitions.

No.	Column Name	Description
1	Design	Probe design type.
2	Color	Color channel.
3	Context	Probe context.
4	Random	Flag indicating if the probeś location was randomly chosen.
5	HumanMethylation27	Flag indicating if the probe is also covered by HumanMethylation27k assay.
6	Mismatches A	Number of base mismatches between the provided and expected probe sequence.
7	Mismatches B	Number of base mismatches between the provided and expected probe sequence.
8	CGI Relation	Relation to a CpG island.
9	CpG	Number of CpG dinucleotides in the sequence neighborhood of the target.
10	GC	Percentage of C and G bases in the sequence neighborhood of the target.
11	SNPs 3	Number of SNPs that overlap with the last 3 bases of a probe's target sequence.
12	SNPs 5	Number of SNPs that overlap with the last 5 bases of a probe's target sequence.
13	SNPs Full	Number of SNPs that overlap with the probe's sequence.

Table 2: Metadata accompanying the HumanMethylation450 probe definitions.

Bead Chip assay that targets the respective CpG dinucleotide. This is a positive `integer` value, or `NA` if the CpG dinucleotide is not covered by this platform.

Probe annotation tables are a special case of site annotation tables. They define Infinium probes targeting individual CpGs, and define some special features that are used in loading data, as well as by normalization and filtering procedures. Table 2 gives an overview of the Infinium 450k probes for the hg19 genome assembly.

Finally, control probe annotation tables are `data.frame` objects that list all control probes in a given assay, along with their characteristics. In a given genomic assembly, every probe annotation table is accompanied by a corresponding control probe annotation table. This allows different control probe definitions for different assemblies, although control probes are generally unrelated to the targeted genomic sequence. Table 3 lists the columns of the Infinium 450k control probe annotation table for hg19.

1.2 Region Annotation Tables

A region annotation table lists all regions in the targeted genome characterized by a specific annotation. For example, the gene body annotation table contains all Ensembl gene records, defined as genomic regions between the transcription start site and the transcription termination site of a gene. Similarly to the annotation of sites, every region annotation is a single `GRangesList` object, containing one `GRanges` instance per chromosome. Regions are sorted

No.	Column Name	Description
1	ID	Probe identifier.
2	Target	Probe category.
3	Color	Probe color.
4	Description	Probe description, abbreviated.
5	AVG	...
6	Evaluate Green	If probe is used in the evaluation of the green channel.
7	Evaluate Red	If probe is used in the evaluation of the red channel.
8	Expected Intensity	Expected intensity of the probe (background, low, medium or high).
9	Sample-dependent	Flag indicating if the intensity of the probe is sample-dependent.
10	Index	Index of the probe in its category.

Table 3: Columns in the HumanMethylation450 control probe annotation table.

based on their first genomic positions.[2] Also, the following two columns are always present in the metadata of the object:

CpG
Number of CpG dinucleotides in the region. This is a non-negative `integer` value.

GC
Number of C and G nucleotides in the region. This is a non-negative `integer` value.

Successfully contructed `RnBeads` annotation packages should contain the following 4 built-in region annotation tables: tiling regions, gene bodies, gene promoters and CpG islands. More details about these annotations are provided in the following sections.

Note that regions in an annotation table are not necessarily targeted by a methylation assay, or by methylation in general. For example, most of the genomic tiling regions in hg19 are not targeted by Infinium 450k; a significant fraction of these regions contain no CpG dinucleotides at all.

1.3 Mappings

Mapping structures link regions to site annotations and are implemented as `lists` of `IRanges` instances, one per chromosome. For example, a mapping from gene promoters to Infinium 27k probes in hg19 is a `list` of 24 `IRanges` objects. Only promoters that are targeted by Infinium 27k appear in the mapping structure. For every such promoter, the corresponding `IRanges` instance stores the range of all probes as indices (on the same chromosome) that lie in the promoter region. Note that sites in an annotation table are always sorted based on their genomic coordinates, therefore, all sites that overlap a given region have consecutive indices in the `GRanges` object of their annotation table.

[2]For a region defined on the complementary strand, its first position is the end of the region.

Mappings are extensively used in RnBeads. They enable summarizing methylation levels at pre-defined genomic regions, defining methylation profiles based on regions and the identification of differentially methylated regions between sample groups. As shown in Table 1, an .RData file that stores a site annotation table also contains the mapping structures from all built-in regions to the respective sites.

1.4 Managing Annotations

All currently loaded annotation structures are located in a dedicated environment managed internally by RnBeads. In order to achieve parsimonious use of the system resources, data provided from an annotation package is loaded dynamically, i.e. on demand. Initial request for a genome assembly support triggers loading the chromosome names and the built-in region annotation tables. An example for such event is the request to obtain a list of supported chromosome names, or a list of supported site or region annotations. A site annotation table (along with its associated mapping structures) is loaded only when it is needed for analysis, e.g. a dataset of the corresponding type is loaded, or when the user explicitly requests annotation information.

Note that RnBeads annotation packages do not export any R functions or classes. In fact, an annotation package is expected to contain no R code at all.

2 How the Annotation Creator Works

RnBeadsAnnotationCreator contains routines that initialize a new annotation package and create the expected data structures. The general workflow consists of 4 major steps, described below.

1. Create a new R source package structure, including the package's base directory, the files DESCRIPTION and NAMESPACE, subdirectories data, inst, man and R, as well as some additional files and/or directories.

2. Download and process genomic annotation from public repositories, such as:
   - SNP records from dbSNP.
   - Gene definitions from Ensembl.
   - CpG island definitions from the UCSC Genome Browser.
   - HumanMethylation27 and HumanMethylation450 probe assay annotation from the Gene Expression Omnibus.

3. Construct annotation tables for the targeted genome's CpG dinucleotides and Infinium probes (site annotation tables), as well as for four different region types - genome tiling regions, gene bodies, gene promoters and CpG islands (region annotation tables).

4. Contruct mapping structures between every pair of region type and site type. A mapping stores how many and which sites are contained in each individual region.

Creating a new annotation package is achieved by calling the only exported function by the annotation creator. For example, the following code attempts to create a new annotation package for the Zv9 assembly (zebrafish genome) that is currently not natively supported by RnBeads:

```
createAnnotationPackage("zv9", dest="/path/to/package")
```

Note, that this command will not work until the corresponding functionality has been implemented in the `RnBeadsAnnotationCreator` package. The following section provides an example on how this can be done. The function above creates the new package directory named `RnBeads.zv9` in `path/to/package` and initializes the annotation tables and mappings. If the directory already exists, the function assumes that the package is partially created and resumes processing from the first step that was not completed. All downloaded and preprocessed intermediate resources are saved in the `temp` subdirectory of the package, which is (by default) removed after all annotation structures are successfully initialized and saved to the subdirectory `data`.

Note that some processing steps are computationally and memory intensive because they operate on large tables. Examples for such steps include merging and sorting dbSNP records, calculating CpG density and GC content at the neighborhood of every CpG dinucleotide, and others. The creation of an hg19 annotation package may take several days on a single processing core and have a peak memory usage of over 80 GB.

The following section in this documentation gives on overview of the steps needed to add support for a new genome assembly.

3 Adding Support for a Genome Assembly

This section lists the steps to add support for the zebrafish genome to the annotation creator. This involves downloading and unpacking the source version of the `RnBeadsAnnotationCreator` package. The archive can be obtained from the RnBeads website. After modifying the source code of the package according to the instructions presented here, the annotation creator will be able to construct a package `RnBeads.zv9`.

3.1 Prerequisites

There are several prerequisites to adding support for a new genome assembly to the annotation creator, and thus to `RnBeads`.

1. **The genomic sequence is known, and a corresponding package is available in Bioconductor.**
   The annotation creator relies on an object of type `BSgenome` in order to extract all genomic CpG dinucleotides, as well as to compute CpG density and GC content values.

2. **Gene definitions are available in an Ensembl biomart.**
   The annotation creator uses the `biomaRt` package to download gene definition tables from Ensembl.

3. **CpG island definitions are available in the UCSC Genome Browser.**
   The annotation creator downloads the list of CpG islands to make it available as a built-in region type, and also to enrich the site annotation tables.

The following sections guide `RnBeadsAnnotationCreator` to the corresponding Bioconductor package, Ensembl table and URL for the zebrafish genome.

3.2 Adding a New File

In `RnBeadsAnnotationCreator`, support for a given assembly is implemented in a function with a name consisting of the prefix `createAnnotationPackage.*`, where `*` resembles the assembly identifier. Every such function is implemented in a separate R file. Notice the file names in the `R` subdirectory of the `RnBeadsAnnotationCreator` package source.

For the zebrafish genome, we need a function named `createAnnotationPackage.zv9`. An easy approach to implement it is to use an already existing function as a template. Please create a copy of the file `createAnnotationPackage.mm10.R` and save it as `createAnnotationPackage.zv9.R` in the same directory.

3.3 Adjusting the R Code

Now let us open the newly created file. As a first step, we need to rename the function being defined to `createAnnotationPackage.zv9`. Feel free to adjust the documentation in the header of the file, as well as the Roxygen-style documentation of the function.

3.3.1 Genome Assembly Package and Chromosomes

The Zebrafish genome contains 25 chromosomes, named `"chr1"` to `"chr25"`. The Bioconductor package defining the genomic sequence of zv9 is `BSgenome.Drerio.UCSC.danRer7`. Therefore, we need to install this package (follow the instructions on its Bioconductor web page) and modify the first section of the package creation function to:

```
suppressPackageStartupMessages(library(BSgenome.Drerio.UCSC.danRer7))

## Genomic sequence and supported chromosomes
GENOME <- 'BSgenome.Drerio.UCSC.danRer7'
assign('GENOME', GENOME, .globals)
CHROMOSOMES <- as.character(1:25)
names(CHROMOSOMES) <- paste0("chr", CHROMOSOMES)
assign('CHROMOSOMES', CHROMOSOMES, .globals)
rm(GENOME)
```

The environment `.globals` is extensively used during the annotation package initialization. The code above sets the variables `GENOME` and `CHROMOSOMES` in this environment, which are later read by other functions of the annotation creator. Once initialized, every site and region annotation table are also saved as variables in the `.globals` environment. This is the reason why they cannot be seen as variables defined in the function `createAnnotationPackage.zv9`.

3.3.2 SNP Definitions

The dbSNP database contains records for small variations in zebrafish. The annotation creator includes functions for downloading and processing VCF files from dbSNP[3]. Similarly to the mm10 assembly, there is one file per zebrafish chromosome in dbSNP. The only R code we need to modify in this section is the initialization of the `vcf.files` variable:

[3]The FTP server we use here can be found at ftp://ftp.ncbi.nih.gov/snp/organisms/

```
vcf.files <- gsub("^chr(.+)$", "vcf_chr_\\1.vcf.gz", names(CHROMOSOMES))
vcf.files <- paste0(DBSNP.FTP.BASE, "zebrafish_7955/VCF/", vcf.files)
```

The annotation creator currently uses a hard-coded mapping from RefSeq assembly identifiers (used in dbSNP) to the assembly codes used in `RnBeads` and other Bioconductor packages. When including SNP support for a genome assembly, you might need to update the variable `REFERENCE2ASSEMBLY` in `globals.R`.

It is important to note that the availability of SNP definitions is not a requirement for the construction of an annotation package. `RnBeads.mm9`, for example, does not incorporate SNP data in its site annotation table.

3.3.3 Region Annotations

In order to construct region annotation tables, the annotation creator needs the specific parameters to access the gene definition table using biomaRt. These parameters are summarized in the variable `biomart.params`. In addition, we will overwrite the default URL for the CpG island definition table from the UCSC Genome Browser. The reason to use a non-default path is that the zv9 assembly is referred to as `danRer7` by the Genome Browser. The section on creating region annotations should look like:

```
## Define genomic regions
biomart.parameters <- list(
  database.name = "ensembl",
  dataset.name = "drerio_gene_ensembl",
  required.columns = c(
    "id" = "ensembl_gene_id",
    "chromosome" = "chromosome_name",
    "start" = "start_position",
    "end" = "end_position",
    "strand" = "strand",
    "symbol" = "zfin_symbol",
    "entrezID" = "entrezgene"))
cgi.url <- paste0(UCSC.FTP.BASE, "danRer7/database/cpgIslandExt.txt.gz")
logger.start("Region Annotation")
update.annot("regions", "region annotation", rnb.update.region.annotation,
  biomart.parameters = biomart.parameters, cgi.download.url = cgi.url)
rm(biomart.parameters, cgi.url)
logger.completed()
```

3.3.4 Sites and Mappings

We can keep the remaining code in the function unchanged. Now all that is left to do is to build and install the modified `RnBeadsAnnotationCreator` package (e.g. using `R CMD build` and `R CMD INSTALL`), and test the code using the command introduced earlier:

```
createAnnotationPackage("zv9", dest="/path/to/package")
```

Upon completion, this function creates a directory named `RnBeads.zv9` in the specified path. You should look at the `DESCRIPTION` file, include additional fields if you wish to, and maybe also create a file `inst/NEWS`. More information on the directory structure and files in an R package is available in the tutorial Writing R Extensions. The last step is to build and install the newly created package. You can validate that the zebrafish genome is now supported by `RnBeads` by starting a new R session and typing the following commands:

```
suppressPackageStartupMessages(library(RnBeads))
rnb.get.assemblies()
```

4 Tips and Tricks

This section contains practical advices on approaches for testing R code in `RnBeadsAnnotationCreator` and investigating its functionality in details.

4.1 Testing and Production Phases

Creating an annotation package can be time- and memory-consuming. Here, we present some suggestions for debugging this process.

- **Use logging.**
  By default, the function `createAnnotationPackage` enables logging to the console. Keeping logs to a file (see the function `logger.start` in `RnBeads`) might help recognizing issues and measuring efficiency.
- **Disable cleaning up.**
  The last parameter of the function `createAnnotationPackage` is a flag specifying if the temporary directory should be removed at the end of the annotation package creation. Setting this to `FALSE` for debugging purposes might save a lot of time in re-runs.
- **Limit supported chromosomes.**
  Making a test run of a package creation process using only two chromosomes as supported ones could validate that the code behaves as expected before starting the function for a 'full' annotation package.

4.2 Data Tables and Columns in Ensembl

Adjusting the biomaRt parameters for downloading gene definition tables might not be trivial, especially when the targeted assembly is not a commonly used one. Here, we present the approach we used to find these parameters for the zebrafish genome.

First, we need to find the dataset name that corresponds to the assembly of interest. The following code snippet extracts a table with all datasets available from the biomaRt service.

```
suppressPackageStartupMessages(library(biomaRt))
database.name <- "ensembl"
mart <- useMart(database.name)
datasets <- listMarts(mart)
head(datasets)
```

The variable `datasets` is a `data.frame` with three columns: `"dataset"`, `"description"` and `"version"`. The first column contains the dataset name we should use. After manual inspection of the table, we find that the dataset for Zebrafish is `"drerio_gene_ensembl"`. Carefully also verify that the gene annotation fits the genome assembly you used. The next step is to extract all possible attributes (table columns) and select the ones related to gene body definitions.

```
dataset.name <- "drerio_gene_ensembl"
mart <- useMart(database.name, dataset.name)
data.attributes <- unique(listAttributes(mart))
head(data.attributes)
```

The variable `data.attributes` is a large `data.frame` with two columns: `"name"` and `"description"`. One way to pinpoint relevant attribute names is by searching for keywords. For example, the following code shows the attributes related to gene symbols.

```
data.attributes[grep("symbol", data.attributes[, 1]), ]
```

Appendices

Appendix A

Full R code of the function `createAnnotationPackage.zv9`.

```
createAnnotationPackage.zv9 <- function() {

  suppressPackageStartupMessages(library(BSgenome.Drerio.UCSC.danRer7))

  ## Genomic sequence and supported chromosomes
  GENOME <- 'BSgenome.Drerio.UCSC.danRer7'
  assign('GENOME', GENOME, .globals)
  CHROMOSOMES <- as.character(1:25)
  names(CHROMOSOMES) <- paste0("chr", CHROMOSOMES)
  assign('CHROMOSOMES', CHROMOSOMES, .globals)
  rm(GENOME)

  ## Download SNP annotation
  logger.start("SNP Annotation")
  vcf.files <- gsub("^chr(.+)$", "vcf_chr_\\1.vcf.gz", names(CHROMOSOMES))
  vcf.files <- paste0(DBSNP.FTP.BASE, "zebrafish_7955/VCF/", vcf.files)
  update.annot("snps", "polymorphism information", rnb.update.dbsnp,
    ftp.files = vcf.files)
```

```
  logger.info(paste("Using:", attr(.globals[['snps']], "version")))
  rm(vcf.files)
  logger.completed()

  ## Define genomic regions
  biomart.parameters <- list(
    database.name = "ensembl",
    dataset.name = "drerio_gene_ensembl",
    required.columns = c(
      "id" = "ensembl_gene_id",
      "chromosome" = "chromosome_name",
      "start" = "start_position",
      "end" = "end_position",
      "strand" = "strand",
      "symbol" = "zfin_symbol",
      "entrezID" = "entrezgene"))
  cgi.url <- paste0(UCSC.FTP.BASE, "danRer7/database/cpgIslandExt.txt.gz")
  logger.start("Region Annotation")
  update.annot("regions", "region annotation", rnb.update.region.annotation,
    biomart.parameters = biomart.parameters, cgi.download.url = cgi.url)
  rm(biomart.parameters, cgi.url)
  logger.completed()

  ## Define genomic sites
  logger.start("Genomic Sites")
  update.annot("sites", "CpG annotation", rnb.update.sites)
  logger.completed()

  ## Create all possible mappings from regions to sites
  logger.start("Mappings")
  update.annot("mappings", "mappings", rnb.create.mappings)
  logger.completed()

  ## Export the annotation tables
  rnb.export.annotations.to.data.files()
}
```

Appendix B

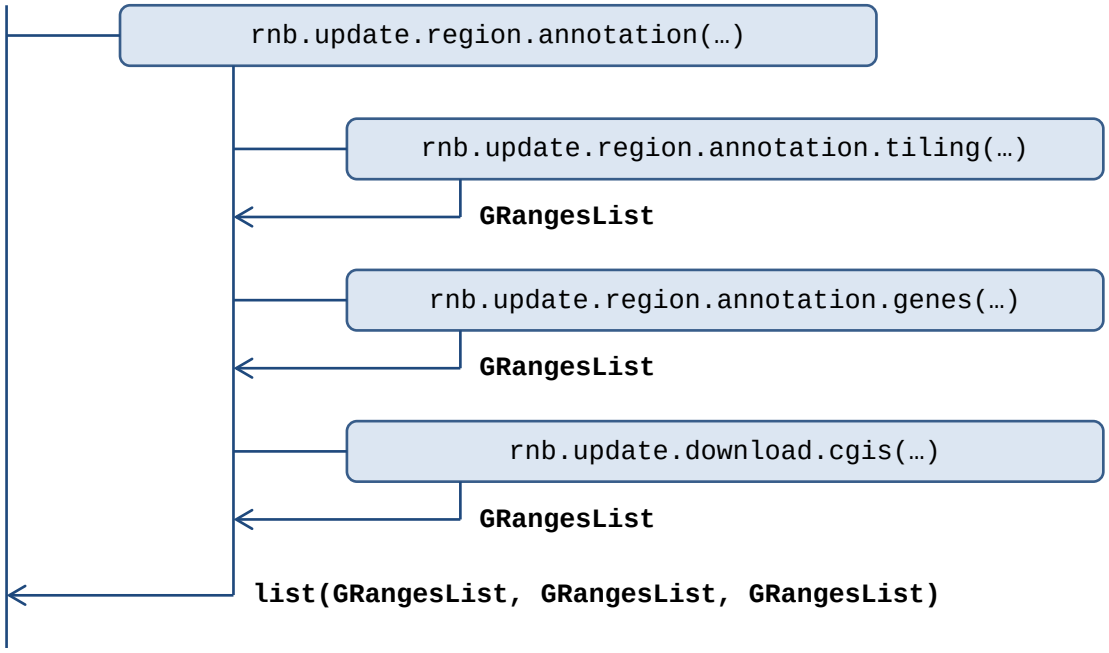

Figure 1: Workflow of initializating region annotation tables. The responsible function is `rnb.update.region.annotation`.

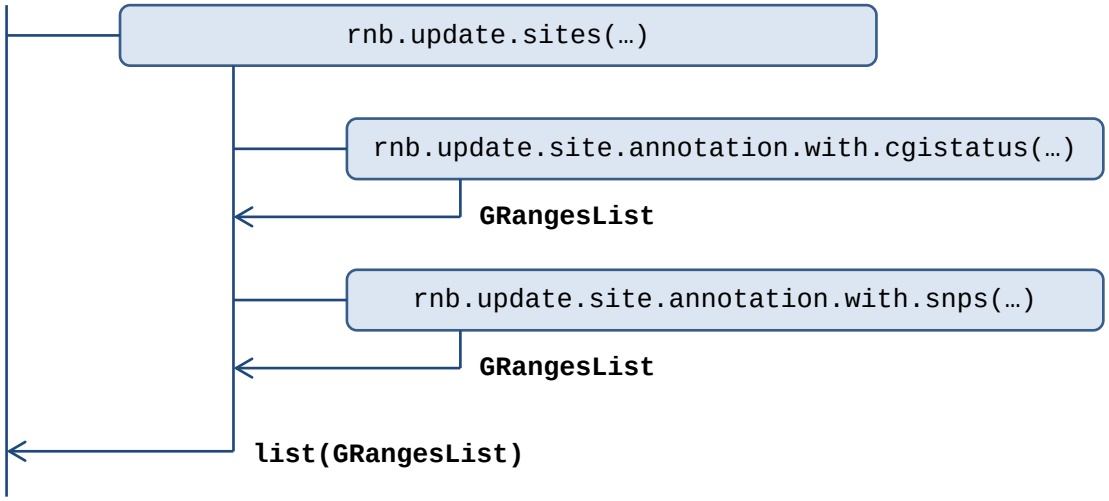

Figure 2: Workflow of initializating site annotation tables. The responsible function is `rnb.update.sites`.

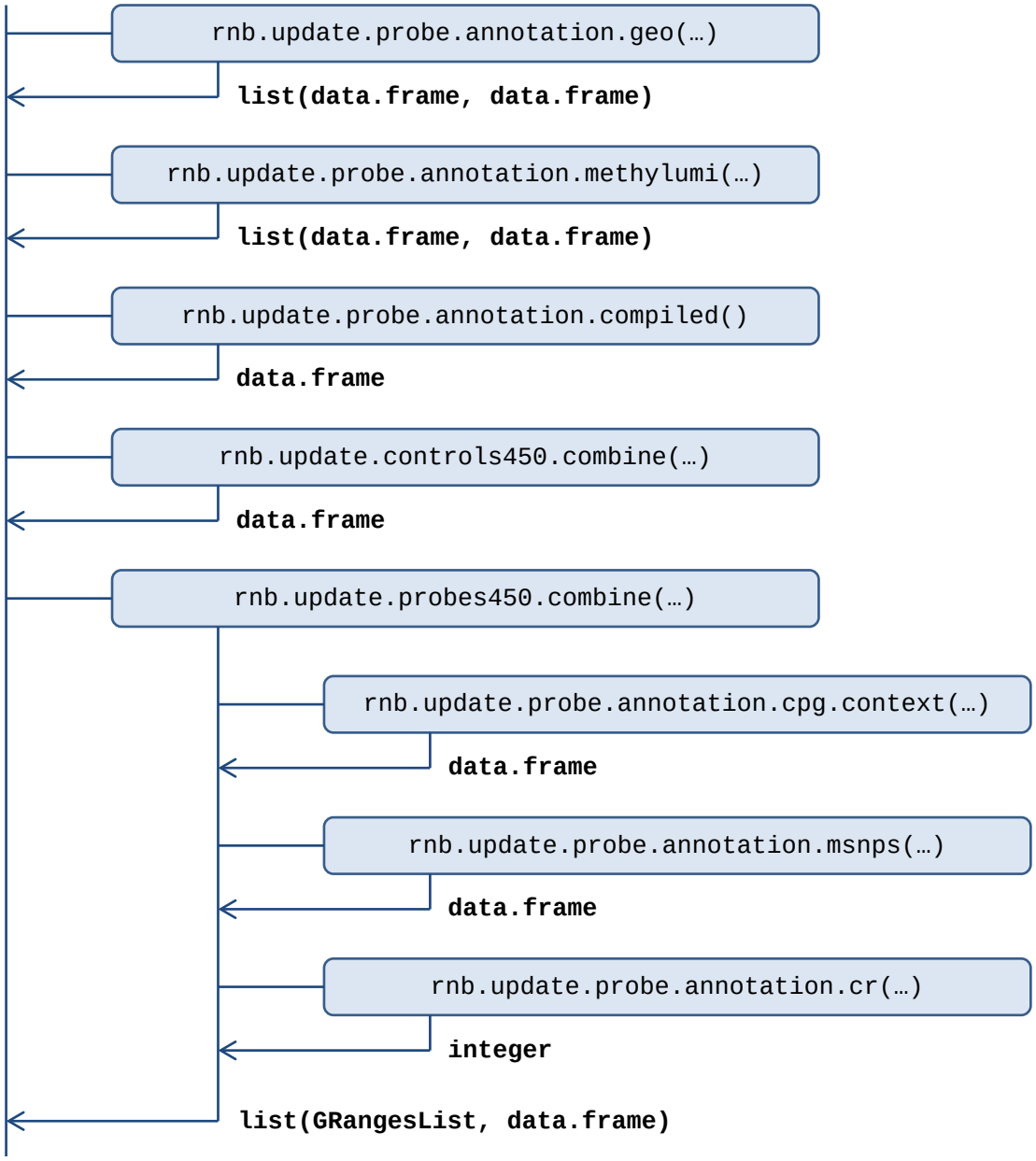

Figure 3: Workflow of initializating an Infinium 450k probe annotation table. The responsible function is `rnb.update.probe450k.annotation`.

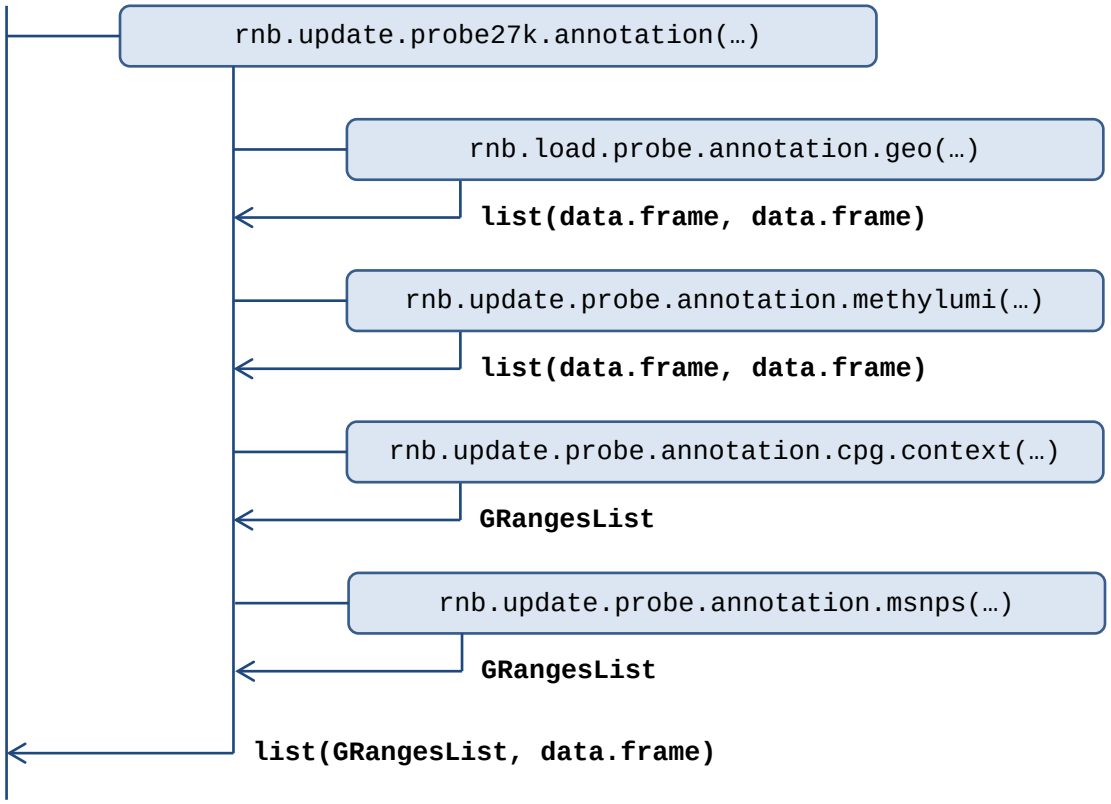

Figure 4: Workflow of initializating an Infinium 27k probe annotation table. The responsible function is `rnb.update.probe27k.annotation`.