How to forge a BSgenome data package

Hervé Pagès
Gentleman Lab
Fred Hutchinson Cancer Research Center
Seattle, WA

February 8, 2024

Contents

1 Introduction

This vignette describes the process of forging a *BSgenome data package*. It is intended for Bioconductor users who want to make a new *BSgenome data package*, not for regular users of these packages.

IMPORTANT NOTE: Starting with Bioconductor 3.17, a new package, the BSgenomeForge package, provides more user-friendly tools for creating a BSgenome data package from an NCBI assembly or UCSC genome. However the process described in this vignette still applies if your assembly or genome is not from NCBI or UCSC.

Start R (make sure you are using the latest release version), load the BSgenome package, and use the `available.genomes` function to get the list of *BSgenome data packages* available in the current release version of Bioconductor. So you confirm that none of those genomes suits your needs? And you want to make your own package? If you answer yes to these 2 questions, then you've come to the right place.

Requirements:

- Some basic knowledge of the Unix/Linux command line is required. The commands that you will most likely need are: `cd`, `mkdir`, `mv`, `rmdir`, `tar`, `gunzip`, `unzip`, `ftp` and `wget`. Also you will need to create and edit some text files.
- You need access to a good Unix/Linux build machine with a decent amount of RAM (>= 4GB), especially if your genome is big. For smaller genomes, 2GB or even 1GB of RAM might be enough.
- You need the latest release versions of R plus the Biostrings and BSgenome packages installed on the build machine. To check your installation, start R and try to load the BSgenome package.
- Finally, you need to obtain the *source data files* of the genome that you want to build a package for. There are 2 groups of *source data files*: (1) the files containing the sequence data (those files are required), and (2) the files containing the mask data (those files are optional). For most organisms, these files have been made publicly available on the internet by genome providers like UCSC, NCBI, FlyBase, TAIR, etc. The next sections of this vignette explain how to obtain and prepare these files.

Refer to the *R Installation and Administration* manual [1] if you need to install R or upgrade your R

[1] `https://cran.r-project.org/doc/manuals/R-admin.html`

version, and to the *Bioconductor - Install* page [2] on the Bioconductor website if you need to install or update the Biostrings or BSgenome packages.

Questions, comments or bug reports about this vignette or about the BSgenomeForge functions are welcome. Please use the Bioconductor support site [3] if you have questions or need help to forge a *BSgenome data package*, or about anything related to the BSgenome package. For bug reports or feature requests, please open an issue on GitHub at `https://github.com/Bioconductor/BSgenome/issues`.

In the next section ("How to forge a BSgenome data package with bare sequences"), we describe how to forge a *BSgenome data package* with sequences that have no masks on them. If this is what you're after, then you'll be done at the end of the section. Only if you need to forge a package with masked sequences, you'll have to keep reading but you'll have to forge a *BSgenome data package* with bare sequences first.

In this vignette, we call *target packages* the *BSgenome data packages* that we're going to forge (one target package with bare sequences and possibly a 2nd target package with masked sequences).

2 How to forge a BSgenome data package with bare sequences

2.1 Obtain and prepare the sequence data

2.1.1 What do I need?

The sequence data must be in a single twoBit file (e.g. *musFur1.2bit*) or in a collection of FASTA files (possibly gzip-compressed).

If the latter, then you need 1 FASTA file per sequence that you want to put in the *target package*. In that case the name of each FASTA file must be of the form <*prefix*><*seqname*><*suffix*> where <*seqname*> is the name of the sequence in it and <*prefix*> and <*suffix*> are a prefix and a suffix (possibly empty) that are the same for all the FASTA files.

2.1.2 Examples

UCSC provides the genome sequences for Stickleback (gasAcu1) in various forms: (a) as a twoBit file (*gasAcu1.2bit* located in `https://hgdownload.cse.ucsc.edu/goldenPath/gasAcu1/bigZips/`, (b) as a big compressed tarball containing one file per chromosome (*chromFa.tar.gz* located in the same folder), and (c) as a collection of compressed FASTA files (`chrI.fa.gz`, `chrII.fa.gz`, `chrIII.fa.gz`, ..., `chrXXI.fa.gz`, `chrM.fa.gz` and `chrUn.fa.gz` located in `https://hgdownload.cse.ucsc.edu/goldenPath/gasAcu1/chromosomes/`). To forge a *BSgenome data package* for gasAcu1, the easiest would be to use (a) (the twoBit file). However it would also be possible to use (c) (the collection of compressed FASTA files). In that case the suffix would need to be set to `.fa.gz`. Note that the prefix here is empty and not `chr`, because `chr` is considered to be part of the sequence names (a commonly used chromosome naming convention at UCSC). Alternatively, the big compressed tarball (`chromFa.tar.gz`) could be used. After downloading and extracting it, we should end up with the same files as with (c).

You can use the `fasta.seqlengths` function from the Biostrings package to get the lengths of the sequences in a FASTA file:

[2] `https://bioconductor.org/install/`
[3] `https://support.bioconductor.org/`

```
> library(Biostrings)
> file <- system.file("extdata", "ce2chrM.fa.gz", package="BSgenome")
> fasta.seqlengths(file)

 chrM
13794
```

For the BSgenome.Mfuro.UCSC.musFur1 package, the twoBit file was used and downloaded with:

```
wget https://hgdownload.soe.ucsc.edu/goldenPath/musFur1/bigZips/musFur1.2bit
```

For the BSgenome.Rnorvegicus.UCSC.rn4 package, the big compressed tarball was used. It was downloaded and extracted with:

```
wget https://hgdownload.cse.ucsc.edu/goldenPath/rn4/bigZips/chromFa.tar.gz
tar zxf chromFa.tar.gz
```

2.1.3 The *<seqs_srcdir>* folder

So we assume that you've downloaded the *sequence data files* and that they are now located in a folder on your machine[4]. From now on, we'll refer to this folder as the *<seqs_srcdir>* folder.

Depending on what data files you downloaded (see previous sub-subsection), you might also need to extract and possibly rename the files. Note that all the *sequence data files* should be located directly in the *<seqs_srcdir>* folder, not in subfolders of this folder. For example, depending on the genome, UCSC provides either a big `chromFa.tar.gz` or `chromFa.zip` file that contains the sequence data for all the chromosomes. But it could be that, after extraction of this big file, the individual FASTA files for each chromosome end up being located one level down the *<seqs_srcdir>* folder (granted that you were in this folder when you extracted the file). If this is the case, then you will need to move them one level up (use `mv -i */*.fa .` for this, then remove all the empty subfolders with `rmdir *`).

2.2 Prepare the BSgenome data package seed file

2.2.1 Overview

The *BSgenome data package seed file* will contain all the information needed by the `forgeBSgenomeDataPkg` function to forge the *target package*.

The format of this file is DCF (Debian Control File), which is also the format used for the `DESCRIPTION` file of any R package. The valid fields of a *seed file* are divided in 3 categories:

1. Standard `DESCRIPTION` fields. These fields are actually the mandatory fields found in any `DESCRIPTION` file. They will be copied to the `DESCRIPTION` file of the *target package*.
2. Non-standard `DESCRIPTION` fields. These fields are specific to *seed files* and they will also be copied to the `DESCRIPTION` file of the *target package*. In addition, the values of these fields will be stored in the

[4]Note that checking the md5sums after download is always a good idea.

BSgenome object that will be contained in the *target package*. This means that the users of the *target package* will be able to retrieve these values via the accessor methods defined for *BSgenome* objects. See the man page for the *BSgenome* class (`?'BSgenome-class'`) for a description of these methods.

3. Additional fields that don't fall in the first 2 categories.

The 3 following sub-subsections give an extensive description of all the valid fields of a *seed file*.

Alternatively, the reader in a hurry can go directly to the last sub-subsection of this subsection for an example of a *seed file*.

2.2.2 Standard `DESCRIPTION` fields

- `Package`: Name to give to the *target package*. The convention used for the packages built by the Bioconductor project is to use a name made of 4 parts separated by a dot. Part 1 is always `BSgenome`. Part 2 is the abbreviated name of the organism (when the name of the organism is made of 2 words, we put together the first letter of the first word in upper case followed by the entire second word in lower case e.g. `Rnorvegicus`). Part 3 is the name of the organisation who provided the genome (e.g. `UCSC`). Part 4 is the release string or number used by this organisation to identify this version of the genome (e.g. `rn4`).

- `Title`: The title of the *target package*. E.g. `Full genome sequences for Rattus norvegicus (UCSC version rn4)`.

- `Description`, `Version`, `Author`, `Maintainer`, `License`: Like the 2 previous fields, these are mandatory fields found in any `DESCRIPTION` file. Please refer to the *The DESCRIPTION file* subsection of the *Writing R Extensions* manual [5] for more information about these fields. If you plan to distribute the *target package* that you are about to forge, please pickup the license carefully and make sure that it is compatible with the license of the *sequence data files* if any.

- `Suggests`: [OPTIONAL] If you're going to add examples to the `Examples` section of the *target package* (see `PkgExamples` field below for how to do this), use this field to list the packages used in these examples.

2.2.3 Non-standard `DESCRIPTION` fields

- `organism`: The scientific name of the organism in the format `Genus species` (e.g. `Rattus norvegicus`, `Homo sapiens`) or `Genus species subspecies` (e.g. `Homo sapiens neanderthalensis`).

- `common_name`: The common name of the organism (e.g. `Rat` or `Human`). For organisms that have more than one commmon name (e.g. `Bovine` or `Cow` for Bos taurus), choose one. Note that for the packages built by the Bioconductor project from a UCSC genome, this field corresponds to the `SPECIES` column of the *List of UCSC genome releases* table [6].

- `genome`: The name of the genome. Typically the name of an NCBI assembly (e.g. `GRCh38.p12`, `WBcel235`, `TAIR10.1`, `ARS-UCD1.2`, etc...) or UCSC genome (e.g. `hg38`, `bosTau9`, `galGal6`, `ce11`, etc...). Should preferably match part 4 of the package name (field `Package`). For the packages built by the Bioconductor project from a UCSC genome, this field corresponds to the `UCSC VERSION` field of the *List of UCSC genome releases* table.

[5] `https://cran.r-project.org/doc/manuals/R-exts.html#The-DESCRIPTION-file`
[6] `https://genome.ucsc.edu/FAQ/FAQreleases#release1`

- `provider`: The provider of the *sequence data files* e.g. `UCSC`, `NCBI`, `BDGP`, `FlyBase`, etc... Should preferably match part 3 of the package name (field `Package`).
- `release_date`: When this assembly of the genome was released. For the packages built by the Bioconductor project from a UCSC genome, this field corresponds to the `RELEASE DATE` field of the *List of UCSC genome releases* table.
- `source_url`: The permanent URL where the *sequence data files* used to forge the *target package* can be found. If the *target package* is for an NCBI assembly, use the link to the NCBI landing page for the assembly e.g. `https://www.ncbi.nlm.nih.gov/assembly/GCF_003254395.2/`.
- `organism_biocview`: The official biocViews term for this organism. This is generally the same as the `organism` field except that spaces should be replaced with underscores. The value of this field matters only if the *target package* is going to be added to a Bioconductor repository, because it will determine where the package will show up in the biocViews tree [7]. Note that this is the only field in this category that won't get stored in the *BSgenome* object that will get wrapped in the *target package*.

2.2.4 Other fields

- `BSgenomeObjname`: Should match part 2 of the package name (see `Package` field above).
- `seqnames`: [OPTIONAL] Needed only if you are using a collection of *sequence data files*. In that case `seqnames` must be an R expression returning the names of the single sequences to forge (in a character vector). E.g. `paste("chr", c(1:20, "X", "M", "Un", paste(c(1:20, "X", "Un"), "_random", sep="")), sep="")`.
- `circ_seqs`: Not needed if your NCBI assembly or UCSC genome is registered in the GenomeInfoDb package. An R expression returning the names of the circular sequences (in a character vector). If the `seqnames` field is specified, then `circ_seqs` must be a subset of it. E.g. `"chrM"` for rn4 or `c("chrM", "2micron")` for the sacCer2 genome (Yeast) from UCSC. If the assembly or genome has no circular sequence, set `circ_seqs` to `character(0)`.
- `mseqnames`: [OPTIONAL] An R expression returning the names of the multiple sequences to forge (in a character vector). E.g. `c("random", "chrUn")` for `rn5`. The default value for `mseqnames` is `NULL` (no multiple sequence).
- `PkgDetails`: [OPTIONAL] Some arbitrary text that will be copied to the `Details` section of the man page of the *target package*.
- `SrcDataFiles`: [OPTIONAL] Some arbitrary text that will be copied to the `Note` section of the man pages of the *target package*. `SrcDataFiles` should describe briefly where the *sequence data files* are coming from. Will typically contain URLs (permanent URLs are a must).
- `PkgExamples`: [OPTIONAL] Some R code (possibly with comments) that will be added to the `Examples` section of the man page of the *target package*. Don't forget to list the packages used in these examples in the `Suggests` field.
- `seqs_srcdir`: The absolute path to the folder containing the *sequence data files*.
- `seqfile_name`: Required if the *sequence data files* is a single twoBit file (e.g. `musFur1.2bit`). Ignored otherwise.

[7] `https://bioconductor.org/packages/release/BiocViews.html#___Organism`

- `seqfiles_prefix`, `seqfiles_suffix`: [OPTIONAL] Needed only if you are using a collection of *sequence data files*. The common prefix and suffix that need to be added to all the sequence names (fields `seqnames` and `mseqnames`) to get the name of the corresponding FASTA file. Default values are the empty prefix for `seqfiles_prefix` and `.fa` for `seqfiles_suffix`.
- `ondisk_seq_format`: [OPTIONAL] Specifies how the single sequences should be stored in the *target package*. Can be `2bit`, `rda`, `fa.rz`, or `fa`. If `2bit` (the default), then all the single sequences are stored in a single twoBit file. If `rda`, then each single sequence is stored in a separated serialized *XString* object (one per single sequence). If `fa.rz` or `fa`, then all the single sequences are stored in a single FASTA file (compressed in the RAZip format if `fa.rz`).

2.2.5 An example

The *seed files* used for the packages forged by the Bioconductor project are included in the BSgenome package:

```
> library(BSgenome)
> seed_files <- system.file("extdata", "GentlemanLab", package="BSgenome")
> tail(list.files(seed_files, pattern="-seed$"))
```

```
[1] "BSgenome.Sscrofa.UCSC.susScr3-seed"
[2] "BSgenome.Sscrofa.UCSC.susScr3.masked-seed"
[3] "BSgenome.Tguttata.UCSC.taeGut1-seed"
[4] "BSgenome.Tguttata.UCSC.taeGut1.masked-seed"
[5] "BSgenome.Tguttata.UCSC.taeGut2-seed"
[6] "BSgenome.influenza.NCBI.20100628-seed"
```

```
> ## Display seed file for musFur1:
> musFur1_seed <- list.files(seed_files, pattern="\\.musFur1-seed$", full.names=TRUE)
> cat(readLines(musFur1_seed), sep="\n")
```

```
Package: BSgenome.Mfuro.UCSC.musFur1
Title: Full genomic sequences for Mustela putorius furo (UCSC version musFur1)
Description: Full genomic sequences for Mustela putorius furo (Ferret) as provided by UCSC (musFur1, A
Version: 1.4.2
organism: Mustela putorius furo
common_name: Ferret
provider: UCSC
provider_version: musFur1
release_date: Apr. 2011
release_name: Ferret Genome Sequencing Consortium MusPutFur1.0
source_url: http://hgdownload.soe.ucsc.edu/goldenPath/musFur1/bigZips/
organism_biocview: Mustela_furo
BSgenomeObjname: Mfuro
SrcDataFiles: musFur1.2bit from http://hgdownload.soe.ucsc.edu/goldenPath/musFur1/bigZips/
PkgExamples: bsg$GL896898  # same as bsg[["GL896898"]]
seqs_srcdir: /fh/fast/morgan_m/BioC/BSgenomeForge/srcdata/BSgenome.Mfuro.UCSC.musFur1/seqs
seqfile_name: musFur1.2bit
```

```
> ## Display seed file for rn4:
> rn4_seed <- list.files(seed_files, pattern="\\.rn4-seed$", full.names=TRUE)
> cat(readLines(rn4_seed), sep="\n")


Package: BSgenome.Rnorvegicus.UCSC.rn4
Title: Full genomic sequences for Rattus norvegicus (UCSC version rn4)
Description: Full genomic sequences for Rattus norvegicus (Rat) as provided by UCSC (rn4, Nov. 2004) a
Version: 1.4.2
Suggests: TxDb.Rnorvegicus.UCSC.rn4.ensGene
organism: Rattus norvegicus
common_name: Rat
provider: UCSC
provider_version: rn4
release_date: Nov. 2004
release_name: Baylor College of Medicine HGSC v3.4
source_url: http://hgdownload.cse.ucsc.edu/goldenPath/rn4/bigZips/
organism_biocview: Rattus_norvegicus
BSgenomeObjname: Rnorvegicus
seqnames: paste("chr", c(1:20, "X", "M", "Un", paste(c(1:20, "X", "Un"), "_random", sep="")), sep="")
circ_seqs: "chrM"
SrcDataFiles: chromFa.tar.gz from http://hgdownload.cse.ucsc.edu/goldenPath/rn4/bigZips/
PkgExamples: bsg$chr1  # same as bsg[["chr1"]]
        .
        ## ---------------------------------------------------------------------
        ## Upstream sequences
        ## ---------------------------------------------------------------------
        ## Starting with BioC 3.0, the upstream1000, upstream2000, and
        ## upstream5000 sequences for rn4 are not included in the BSgenome data
        ## package anymore. However they can easily be extracted from the full
        ## genomic sequences with something like:
        .
        library(TxDb.Rnorvegicus.UCSC.rn4.ensGene)
        txdb <- TxDb.Rnorvegicus.UCSC.rn4.ensGene
        gn <- sort(genes(txdb))
        up1000 <- flank(gn, width=1000)
        up1000seqs <- getSeq(bsg, up1000)
        .
        ## IMPORTANT: Make sure you use a TxDb package (or TxDb object),
        ## that contains a gene model based on the exact same reference genome
        ## as the BSgenome object you pass to getSeq(). Note that you can make
        ## your own custom TxDb object from various annotation resources.
        ## See the makeTxDbFromUCSC(), makeTxDbFromBiomart(),
        ## and makeTxDbFromGFF() functions in the GenomicFeatures
        ## package.
seqs_srcdir: /fh/fast/morgan_m/BioC/BSgenomeForge/srcdata/BSgenome.Rnorvegicus.UCSC.rn4/seqs
```

From now we assume that you have managed to prepare the *seed file* for your *target package*.

2.3 Forge the *target package*

To forge the *target package*, start R, load the BSgenome package, and call the `forgeBSgenomeDataPkg` function on your *seed file*. For example, if the path to your *seed file* is `"path/to/my/seed"`, do:

```
> library(BSgenome)
> forgeBSgenomeDataPkg("path/to/my/seed")
```

Depending on the size of the genome and your hardware, this can take between 2 minutes and 1 or 2 hours. By default `forgeBSgenomeDataPkg` will create the source tree of the *target package* in the current directory.

Once `forgeBSgenomeDataPkg` is done, ignore the warnings (if any), quit R, and build the source package (tarball) with

```
R CMD build <pkgdir>
```

where `<pkgdir>` is the path to the source tree of the package.

Then check the package with

```
R CMD check <tarball>
```

where `<tarball>` is the path to the tarball produced by `R CMD build`.

Finally install the package with

```
R CMD INSTALL <tarball>
```

and use it!

3 How to forge a BSgenome data package with masked sequences

3.1 Obtain and prepare the mask data

3.1.1 What do I need?

The mask data are not available for all organisms. What you download exactly depends of course on what's available and also on what built-in masks you want to have in the *2nd target package*. 4 kinds of built-in masks are currently supported by BSgenomeForge:

- the masks of assembly gaps, aka "the AGAPS masks";
- the masks of intra-contig ambiguities, aka "the AMB masks";
- the masks of repeat regions that were determined by the RepeatMasker software, aka "the RM masks";

- the masks of repeat regions that were determined by the Tandem Repeats Finder software (where only repeats with period less than or equal to 12 were kept), aka "the TRF masks".

For the AGAPS masks, you need UCSC "gap" or NCBI "agp" files. It can be one file per chromosome or a single big file containing the assembly gap information for all the chromosomes together. In the former case, the name of each file must be of the form *<prefix><seqname><suffix>*. Like for the FASTA files in group 1, *<seqname>* must be the name of the sequence (sequence names for FASTA files and AGAPS masks must match) and *<prefix>* and *<suffix>* must be a prefix and a suffix (possibly empty) that are the same for all the files (this prefix/suffix doesn't need to be, and typically is not, the same as for the FASTA files in group 1).

You don't need any file for the AMB masks.

For the RM masks, you need RepeatMasker `.out` files. Like for the AGAPS masks, it can be one file per chromosome or a single big file containing the RepeatMasker information for all the chromosomes together. In the former case, the name of each file must also be of the form *<prefix><seqname><suffix>*. Same comments apply as for the AGAPS masks above.

For the TRF masks, you need Tandem Repeats Finder `.bed` files. Again, it can be one file per chromosome or a single big file. In the former case, the name of each file must also be of the form *<prefix><seqname><suffix>*). Same comments apply as for the AGAPS masks above.

Again, for some organisms none of the masks above are available or only some of them are.

3.1.2 An example

Here is how the *mask data files* for the BSgenome.Rnorvegicus.UCSC.rn4 package were obtained and prepared:

- AGAPS masks: all the `chr*_gap.txt.gz` files (UCSC "gap" files) were downloaded from the UCSC *database* folder [8] for `rn4`. This was done with the standard Unix/Linux `ftp` command:

```
ftp hgdownload.cse.ucsc.edu # login as "anonymous"
cd goldenPath/rn4/database
prompt
mget chr*_gap.txt.gz
```

  Then all the downloaded files were uncompressed with:

```
for file in chr*_gap.txt.gz; do gunzip $file ; done
```

- RM masks: file `chromOut.tar.gz` was downloaded from the UCSC *bigZips* folder and extracted with:

```
tar zxf chromOut.tar.gz
```

- TRF masks: file `chromTrf.tar.gz` was downloaded from the UCSC *bigZips* folder and extracted with:

```
tar zxf chromTrf.tar.gz
```

[8] `https://hgdownload.cse.ucsc.edu/goldenPath/rn4/database/`

3.1.3 The *<masks_srcdir>* folder

From now we assume that you've downloaded (checking the md5sums is always a good idea) and possibly extracted all the *mask data files*, and that they are located in the *<masks_srcdir>* folder.

Note that all the *mask data files* should be located directly in the *<masks_srcdir>* folder, not in subfolders of this folder.

3.2 Prepare the seed file for the masked BSgenome data package

3.2.1 Overview

The *seed file* for the *BSgenome data package* with masked sequences (i.e. *2nd target package*) is similar to the *seed file* for the *BSgenome data package* containing the corresponding bare sequences (a.k.a. the *reference target package* i.e. the package you forged in the previous section). It will contain all the information needed by the `forgeMaskedBSgenomeDataPkg` function to forge the *2nd target package*.

The fields of this *seed file* are described in the 3 following sub-subsections.

Alternatively, the reader in a hurry can go directly to the last sub-subsection of this subsection for an example of *seed file*.

3.2.2 Standard `DESCRIPTION` fields

- `Package`: Name to give to the *2nd target package*. The convention used for the packages built by the Bioconductor project is to use the same name as the *reference target package* with the `.masked` suffix added to it.
- `Title`: The title of the package. E.g. `Full masked genome sequences for Rattus norvegicus (UCSC version rn4)`.
- `Description`, `Version`, `Author`, `Maintainer`, `License`: See previous section.

3.2.3 Non-standard `DESCRIPTION` fields

- `organism_biocview`: Same as for the *reference target package*.
- `source_url`: The permanent URL where the *mask data files* used to forge this package can be found.

3.2.4 Other fields

- `RefPkgname`: The name of the *reference target package*.
- `nmask_per_seq`: The number of masks per sequence (1 to 4).
- `PkgDetails`, `PkgExamples`: See previous section.
- `SrcDataFiles`: [OPTIONAL] Some arbitrary text that will be copied to the `Note` section of the man pages of the *target package*. `SrcDataFiles` should describe briefly where the *mask data files* are coming from. Will typically contain URLs (permanent URLs are a must).

- `masks_srcdir`: The path to the *<masks_srcdir>* folder.
- `AGAPSfiles_type`: [OPTIONAL] Must be `gap` (the default) if the *source data files* containing the AGAPS masks information are UCSC "gap" files, or `agp` if they are NCBI "agp" files.
- `AGAPSfiles_name`: [OPTIONAL] Omit this field if you have one *source data file* per single sequence for the AGAPS masks and use the `AGAPSfiles_prefix` and `AGAPSfiles_suffix` fields below instead. Otherwise, use this field to specify the name of the single big file.
- `AGAPSfiles_prefix`, `AGAPSfiles_suffix`: [OPTIONAL] Omit these fields if you have one single big *source data file* for all the AGAPS masks and use the `AGAPSfiles_name` field above instead. Otherwise, use these fields to specify the common prefix and suffix that need to be added to all the single sequence names (field `seqnames`) to get the name of the file that contains the corresponding AGAPS mask information. Default values are the empty prefix for `AGAPSfiles_prefix` and `_gap.txt` for `AGAPSfiles_suffix`.
- `RMfiles_name`, `RMfiles_prefix`, `RMfiles_suffix`: [OPTIONAL] Those fields work like the `AGAPSfiles*` fields above but for the RM masks. Default values are the empty prefix for `RMfiles_prefix` and `.fa.out` for `RMfiles_suffix`.
- `TRFfiles_name`, `TRFfiles_prefix`, `TRFfiles_suffix`: [OPTIONAL] Those fields work like the `AGAPSfiles*` fields above but for the TRF masks. Default values are the empty prefix for `TRFfiles_prefix` and `.bed` for `TRFfiles_suffix`.

3.2.5 An example

The *seed files* used for the packages forged by the Bioconductor project are included in the BSgenome package:

```
> library(BSgenome)
> seed_files <- system.file("extdata", "GentlemanLab", package="BSgenome")
> tail(list.files(seed_files, pattern="\\.masked-seed$"))
```

```
[1] "BSgenome.Ptroglodytes.UCSC.panTro2.masked-seed"
[2] "BSgenome.Ptroglodytes.UCSC.panTro3.masked-seed"
[3] "BSgenome.Rnorvegicus.UCSC.rn4.masked-seed"
[4] "BSgenome.Rnorvegicus.UCSC.rn5.masked-seed"
[5] "BSgenome.Sscrofa.UCSC.susScr3.masked-seed"
[6] "BSgenome.Tguttata.UCSC.taeGut1.masked-seed"
```

```
> rn4_masked_seed <- list.files(seed_files, pattern="\\.rn4\\.masked-seed$", full.names=TRUE)
> cat(readLines(rn4_masked_seed), sep="\n")
```

```
Package: BSgenome.Rnorvegicus.UCSC.rn4.masked
Title: Full masked genomic sequences for Rattus norvegicus (UCSC version rn4)
Description: Full genomic sequences for Rattus norvegicus (Rat) as provided by UCSC (rn4, Nov. 2004) a
Version: 1.3.99
RefPkgname: BSgenome.Rnorvegicus.UCSC.rn4
source_url: http://hgdownload.cse.ucsc.edu/goldenPath/rn4/bigZips/
organism_biocview: Rattus_norvegicus
nmask_per_seq: 4
```

```
SrcDataFiles: AGAPS masks: all the chr*_gap.txt.gz files from ftp://hgdownload.cse.ucsc.edu/goldenPath
        RM masks: chromOut.tar.gz from http://hgdownload.cse.ucsc.edu/goldenPath/rn4/bigZips/
        TRF masks: chromTrf.tar.gz from http://hgdownload.cse.ucsc.edu/goldenPath/rn4/bigZips/
PkgExamples: mbsg$chr1  # a MaskedDNAString object!
        ## To get rid of the masks altogether:
        unmasked(mbsg$chr1)  # same as BSgenome.Rnorvegicus.UCSC.rn4$chr1
masks_srcdir: /fh/fast/morgan_m/BioC/BSgenomeForge/srcdata/BSgenome.Rnorvegicus.UCSC.rn4/masks
```

From now we assume that you have managed to prepare the *seed file* for the *2nd target package*.

3.3 Forge the *2nd target package*

To forge the package, start R, make sure the *reference target package* is installed (try to load it), load the BSgenome package, and call the `forgeMaskedBSgenomeDataPkg` function on the *seed file* you made for the *2nd target package*. For example, if the path to your *seed file* is `"path/to/my/seed"`, do:

```
> library(BSgenome)
> forgeMaskedBSgenomeDataPkg("path/to/my/seed")
```

Depending on the size of the genome and your hardware, this can take between 2 minutes and 1 or 2 hours. By default `forgeMaskedBSgenomeDataPkg` will create the source tree of the *2nd target package* in the current directory.

Once `forgeMaskedBSgenomeDataPkg` is done, ignore the warnings (if any), quit R, and build the source package (tarball) with

```
R CMD build <pkgdir>
```

where `<pkgdir>` is the path to the source tree of the package.

Then check the package with

```
R CMD check <tarball>
```

where `<tarball>` is the path to the tarball produced by `R CMD build`.

Finally install the package with

```
R CMD INSTALL <tarball>
```

and use it!

If you want to distribute this package, you need to distribute it with the *reference target package* since it depends on it.

4 Session information

The output in this vignette was produced under the following conditions:

```
> sessionInfo()

R version 4.3.2 Patched (2023-11-13 r85521)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.3 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.18-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] BSgenome_1.70.2      rtracklayer_1.62.0   BiocIO_1.12.0
 [4] GenomicRanges_1.54.1 Biostrings_2.70.2    GenomeInfoDb_1.38.6
 [7] XVector_0.42.0       IRanges_2.36.0       S4Vectors_0.40.2
[10] BiocGenerics_0.48.1

loaded via a namespace (and not attached):
 [1] crayon_1.5.2                DelayedArray_0.28.0
 [3] SummarizedExperiment_1.32.0 GenomicAlignments_1.38.2
 [5] rjson_0.2.21                RCurl_1.98-1.14
 [7] XML_3.99-0.16.1             MatrixGenerics_1.14.0
 [9] Biobase_2.62.0              grid_4.3.2
[11] restfulr_0.0.15             abind_1.4-5
[13] bitops_1.0-7                yaml_2.3.8
[15] compiler_4.3.2              codetools_0.2-19
[17] BiocParallel_1.36.0         lattice_0.22-5
[19] SparseArray_1.2.3           parallel_4.3.2
[21] GenomeInfoDbData_1.2.11     Matrix_1.6-5
[23] tools_4.3.2                 matrixStats_1.2.0
[25] Rsamtools_2.18.0            zlibbioc_1.48.0
[27] S4Arrays_1.2.0
```