Adding new resources to AnnotationHub.

Marc Carlson

April 11, 2014

1 Overview of the process

If you are reading this it is (hopefully) because you intend to write some code that will allow the processing of online resources into R objects that are to be made available via that the *AnnotationHub* package. In order to do this you will have to do three basic steps (outlined below). These steps will have you writing two functions and then calling a third function to do some automatic set up for you. The 1st function will contain instructions on how to process data that is stored online into metadata for describing your new R resources for the AnnotationHub. And the 2nd function is for describing how to take these online resources and transform them into an R object that is useful to end users.

2 Introducing `AnnotationHubMetadata` and `AnnotationHubRecipe` Objects

The *AnnotationHubData* package is a complementary package to the *AnnotationHub* package that provides a place where we can store code that processes online resources into R objects suitable for access through the *AnnotationHub* package. But before you can understand the requirements for this package it is important that you 1st learn about a pair of objects that are used as intermediaries between the hub and its web based repository behind the scenes. The 1st object you need to know about is the `AnnotationHubMetadata` object. These objects store the metadata that describes an online resource. And if you want to see a set of online resources added to the repository and maintained, then it will be necessary to become familiar with the `AnnotationHubMetadata` constructor. For each online resource that you want to process into the AnnotationHub, you will have to be able to

construct an AnnotationHubMetadata object that describes it in detail and that specifies where the recipe function lives.

The second type of object you need to know about is the AnnotationHubRecipe object. This object is actually created from an AnnotationHubMetadata object, so you don't need to be able to make one. But it offers a few conveniences for accessing certain fields while hiding some other things so you will have to know about this when writing your recipe function. In particular the inputFiles and outputFiles methods allow for convenient extraction of relevant filenames needed to process a recipe function into it's AnnotationHub based representation.

3 Step 1: Writing your AnnotationHubMetadata generating function

The 1st function you need to provide is one that processes some online resources into AnnotationHubMetadata objects. This function MUST return a list of AnnotationHubMetadata object. It can rely on other helper functions, but ultimately it (and it's helpers need to know how to find resources and how to process those resources into AnnotationHubMetadata objects on it's own.

The following example function takes GTF files from Ensembl and processes them into AnnotationHubMetadata objects using Map. The calling of the Map function is really the important part of this function, as it shows the function creating a series of AnnotationHubMetadata objects. Prior to that, the function was just calling out to other helper functions in order to process the metadata so that it could be passed to the AnnotationHubMetadata constructor using Map. Notice how one of the fields specified by this function is the Recipe, which indicates both the name and location of the recipe function. We expect most people will want to submit their recipe to the same package as they are submitting their metadata processing function.

```
> makeEnsemblGTFsToAHMs <- function(){
+     baseUrl <- .ensemblBaseUrl
+     sourceUrl <- .ensemblGtfSourceUrls(.ensemblBaseUrl)
+
+     sourceFile <- .ensemblSourcePathFromUrl(baseUrl, sourceUrl)
+     meta <- .ensemblMetadataFromUrl(sourceUrl)
+     rdata <- sub(".gz$", ".RData", sourceFile)
+     description <- paste("Gene Annotation for", meta$species)
```

```
+
+     Map(AnnotationHubMetadata,
+         AnnotationHubRoot=meta$annotationHubRoot,
+         Description=description, Genome=meta$genome,
+         SourceFile=sourceFile, SourceUrl=sourceUrl,
+         SourceVersion=meta$sourceVersion, Species=meta$species,
+         TaxonomyId=meta$taxonomyId, Title=meta$title,
+         MoreArgs=list(
+           Coordinate_1_based = TRUE,
+           DataProvider = "ftp.ensembl.org",
+           Maintainer = "Martin Morgan <mtmorgan@fhcrc.org>",
+           RDataClass = "GRanges",
+           RDataDateAdded = Sys.time(),
+           RDataVersion = "0.0.1",
+           Recipe = c("ensemblGtfToGRangesRecipe", package="AnnotationHubData"),
+           Tags = c("GTF", "ensembl", "Gene", "Transcript", "Annotation")))
+ }
```

The typical case when writing a `AnnotationHubMetadata` generating function like the one above, is to not have it take no arguments. However, if you need this function to take arguments, you can still do so, but you will have to pass them in separately to the helper function described in step 3.

4 Step 2: Writing your recipe

The 2nd kind of function you need to write is called a recipe function. It always must take an single argument called recipe, which is an `AnnotationHubRecipe` object. This object allows for some conveniences by letting you access some of the data in the original AnnotationHubMetadata object that was created for this resource by the function above. Below is a recipe function that takes an Ensembl GTF file and then processes it into a GRanges object by using the import method from the rtracklayer package. Along the way, the `inputFiles` and `outputFile` accessors are used to extract the files/filenames that are needed from the metadata in the `AnnotationHubRecipe` object.

```
> ensemblGTFToGRangesRecipe <- function(recipe){
+      require(rtracklayer)
+     gz.inputFile <- inputFiles(recipe)[1]
+     con <- gzfile(gz.inputFile)
```

```
+     on.exit(close(con))
+     gr <- import(con, "gtf", asRangedData=FALSE)
+     save(gr, file=outputFile(recipe))
+     outputFile(recipe)
+ }
```

5 Step 3: Calling the `makeAnnotationHubResource` helper

Finally you will need to call the `makeAnnotationHubResource` function to do some setup. This function only has two required arguments. The 1st is basically the name of a class that describes the kind of resource you are writing code to import. It just needs to be a unique name. The 2nd argument is the name of your metadata processing function from step one. Once you have finished this, the only step left is to export the class name in the NAMESPACE (this is that string you are providing as your 1st argument), and then add this code to the *AnnotationHubData* repository. We are going to set up a bridge to github so that you can give us a pull request. If you have arguments that you need to get passed down to the RobjectAnnotationHubMetadata generating function that you defined in step 1, you can pass those in after the 1st two arguemnts.

```
> makeAnnotationHubResource("EnsemblGtfImportPreparer",
+                           makeEnsemblGTFsToAHMs)
>
```

6 Session Information

```
R version 3.1.0 RC (2014-04-02 r65358)
Platform: x86_64-apple-darwin10.8.0 (64-bit)

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

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

other attached packages:
[1] GenomicRanges_1.16.0 GenomeInfoDb_1.0.0   AnnotationHub_1.4.0
```

```
[4] IRanges_1.21.45      BiocGenerics_0.10.0

loaded via a namespace (and not attached):
 [1] annotate_1.42.0          AnnotationDbi_1.26.0     Biobase_2.24.0
 [4] BiocInstaller_1.14.0     bitops_1.0-6             Category_2.30.0
 [7] caTools_1.16             colorspace_1.2-4         DBI_0.2-7
[10] dichromat_2.0-0          digest_0.6.4             genefilter_1.46.0
[13] ggplot2_0.9.3.1          graph_1.42.0             grid_3.1.0
[16] gridSVG_1.4-0            GSEABase_1.26.0          gtable_0.1.2
[19] httpuv_1.3.0             httr_0.3                 interactiveDisplay_1.2.0
[22] labeling_0.2             lattice_0.20-29          MASS_7.3-31
[25] Matrix_1.1-3             munsell_0.4.2            plyr_1.8.1
[28] proto_0.3-10             RBGL_1.40.0              RColorBrewer_1.0-5
[31] Rcpp_0.11.1              RCurl_1.95-4.1           reshape2_1.2.2
[34] rjson_0.2.13             RJSONIO_1.0-3            RSQLite_0.11.4
[37] scales_0.2.3             shiny_0.9.1              splines_3.1.0
[40] stats4_3.1.0             stringr_0.6.2            survival_2.37-7
[43] tools_3.1.0              XML_3.98-1.1             xtable_1.7-3
[46] XVector_0.4.0
```