Retrieving sample attributes

Martin Morgan, Robert Gentleman

Created: 12 April 2008

Affymetrix provides mechanisms for specifying sample attributes, including attributes associated with sample processing. Affymetrix provides two types of sample attribute files. `ARR` files are newer, and are produced by the GeneGchip® Command Console (AGCC). `MAGE` files are created by the older GCOS software. This vignette describes how sample attributes can be retrieved from appropriate Affymetrix files. Additional functionality in this package facilitates navigation of Affymetrix NetAffx chip annotation files; parsing CHP and certain other Affymetrix files is available in affxparser.

```
> library(AffyCompatible)
```

1 Reading sample attributes from `ARR` files

`ARR` files are produced by AGCC, with one file per sample. Examples are include in this package.

```
> arrDir <- system.file("extdata", "ARR", package = "AffyCompatible")
> arrFiles <- list.files(arrDir, pattern = ".*ARR", full = TRUE)
> basename(arrFiles)
```

```
[1] "TisMap_Brain_01_v1_WTGene1.ARR"  "TisMap_Breast_01_v1_WTGene1.ARR"
[3] "TisMap_Heart_01_v1_WTGene1.ARR"  "TisMap_Kidney_01_v1_WTGene1.ARR"
```

Use `readArr` to read a single file, or `sapply` to read several:

```
> arr <- readArr(arrFiles[[1]])
> arrs <- sapply(arrFiles, readArr)
> arrs[[1]]
```

```
type: affymetrix-calvin-arraysetfile
createdStep: Other
createdBy:
createdDateTime:
originalProjectName:
guid: 72f5f761-187b-44c9-81d9-b150b118909d
version: 1.0
physicalArrays: PhysicalArrays(1)
userAttributes: UserAttributes(1)
```

The result is an object or list of objects of the auto-generated class *ArraySetFile*. These objects contain information extracted from the `ARR` file, e.g., the type of the file, and the globally unique identifier; some attributes, e.g., creation date in this example, are not defined in this particular file. Access the values of these attributes with the *accessor* implied by the label at the start of the line, e.g.

```
> guid(arr)

[1] "72f5f761-187b-44c9-81d9-b150b118909d"

> version(arr)

[1] "1.0"
```

Simple accessors return their content. Some object slots have multiple values. These are indicated using notation like `PhayiscalArrays(1)` to indicate that the content is a vector (in this case of length 1) of *PhysicalArrays* objects. Access the vector and its elements in the usual way, e.g.,

```
> physicalArrays(arr)

PhysicalArrays(1)

> pas <- physicalArrays(arr)[[1]]
> pas

physicalArray: PhysicalArray(1)
```

In this case, the first `physicalArrays` element is itself a vector of a slightly different object, *PhysicalArray* (no 's' at the end!). We can further navigate the structure to find out information on the arrays used in this sample, e.g.,

```
> physicalArray(pas)[[1]]

type: affymetrix-calvin-array
affxComment: Beta3
createdStep: Other
createdBy:
createdDateTime: 1/1/0001 12:00:00 AM
patassignmentMethod: None
masterFileGUID: HuGene-1_0-st-v1_MASTER
masterFileName: HuGene-1_0-st-v1.MASTER
libraryPackageName: Universal
mediaFileGUID: HuGene-1_0-st-v1_MEDIA
mediaFileName: HuGene-1_0-st-v1.MEDIA
mediaCol: 0
mediaRow: 0
mediaType: Cartridge
affyBarcode:
arrayName: TisMap_Brain_01_v1_WTGene1
guid: 4e65113a-928a-447d-84fb-13fd52df7cbd
arrayAttribute: ArrayAttribute(1)
```

User attributes associated with the sample can be recovered in a similar way:

```
> ua <- userAttribute(userAttributes(arr)[[1]])
> ua

UserAttribute(16)
```

A useful paradigm for retrieving either all information, or a specific attribute from several elements, e.g., the first 4, is

```
> lapply(ua[1:4], force)

[[1]]
name: Age
namespace:
defaultValue:
required: false
type: String
userAttributeValue: UserAttributeValue(1)

[[2]]
name: Assay Type
namespace:
defaultValue:
required: false
type: String
userAttributeValue: UserAttributeValue(1)

[[3]]
name: Automation Flag
namespace:
defaultValue:
required: false
type: String
userAttributeValue: UserAttributeValue(1)

[[4]]
name: Cause of Death / Other
namespace:
defaultValue:
required: false
type: String
userAttributeValue: UserAttributeValue(1)

> sapply(ua[1:4], name)

[1] "Age"                    "Assay Type"             "Automation Flag"
[4] "Cause of Death / Other"
```

A second approach to navigating ARR files is to process the files as XML. For instance, one can read the first file as an xml document.

```
> xml <- readXml(arrFiles[[1]])
```

R objects can be extracted from this document using the xclass function with the *xpath* to the element. The xpath is implied by the navigation scheme outlined above.

```
> xclass(xml, "/ArraySetFile")
```

```
[[1]]
type: affymetrix-calvin-arraysetfile
createdStep: Other
createdBy:
createdDateTime:
originalProjectName:
guid: 72f5f761-187b-44c9-81d9-b150b118909d
version: 1.0
physicalArrays: PhysicalArrays(1)
userAttributes: UserAttributes(1)
```

```
> xclass(xml, "/ArraySetFile/UserAttributes/UserAttribute[4]")
```

```
[[1]]
name: Cause of Death / Other
namespace:
defaultValue:
required: false
type: String
userAttributeValue: UserAttributeValue(1)
```

Notice that the return value from xclass is an instance of an R class. For many elements, it is possible to abbreviate the full xpath

```
> xclass(xml, "//UserAttribute[4]")
```

```
[[1]]
name: Cause of Death / Other
namespace:
defaultValue:
required: false
type: String
userAttributeValue: UserAttributeValue(1)
```

```
> sapply(xclass(xml, "//UserAttribute"), name)[1:4]
```

```
[1] "Age"                    "Assay Type"             "Automation Flag"
[4] "Cause of Death / Other"
```

For advanced use, content is available through the interface provided by the XML package. This is faciliated by understanding the conventions used to map between XML element and attribute names, and R class and slot names. XML attribute names starting with upper-case letters have been replaced by lower-case slot names in R. Certain slot names have been replaced by the prefix `affx`. The list of reserve words and an example of a direct XML query are:

```
> AffyCompatible:::.xreserved()
```

```
 [1] "class"     "comment"   "dim"       "dimnames"  "names"     "row.names"
 [7] "tsp"       "order"     "row"       "array"     "sequence"  "url"
[13] "file"      "title"     "text"      "person"    "image"     "start"
[19] "end"       "date"      "category"
```

```
> unlist(xpathApply(xml, "//UserAttribute/@Name", xmlValue))
```

```
 [1] "Age"                    "Assay Type"             "Automation Flag"
 [4] "Cause of Death / Other" "Cel"                    "Experiment Date"
 [7] "Gender"                 "Lab Technician"         "Lot#"
[10] "P/N*"                   "Race"                   "Sample Date"
[13] "Sample Name"            "Sample#"                "Tissue"
[16] "Vendor"
```

2 Reading sample attributes from MAGE files

MAGE files are produced by GCOS, with one file per sample. Examples are included in this package

```
> mageDir <- system.file("extdata", "DTT", package = "AffyCompatible")
> mageFiles <- list.files(mageDir, pattern = ".*xml", full = TRUE)
> basename(mageFiles)
```

```
[1] "MPRO_0hr_A.xml"                "Sample_1_Wash_Stain_Param.xml"
[3] "Sample_2_4C_Scan.xml"
```

Use `readMage` to read a single file, or `sapply` to read several:

```
> mage <- readMage(mageFiles[[1]])
> mages <- sapply(mageFiles, readMage)
> mages[[1]]
```

```
identifier: Affymetrix.com
name:
propertySets_assnlist: PropertySets_assnlist(1)
auditAndSecurity_package: AuditAndSecurity_package(1)
protocol_package: Protocol_package(1)
bioMaterial_package: BioMaterial_package(1)
arrayDesign_package: ArrayDesign_package(1)
```

```
array_package: Array_package(1)
bioAssay_package: BioAssay_package(1)
bioAssayData_package: BioAssayData_package(1)
experiment_package: Experiment_package(1)
```

These objects can be navigated following the same paradigm as for ARR files, using accessors to arrive at desired end points.

```
> ba <- bioAssay_assnlist(bioAssay_package(mage)[[1]])[[1]]
> ba

physicalBioAssay: PhysicalBioAssay(2)
derivedBioAssay: DerivedBioAssay(1)
measuredBioAssay: MeasuredBioAssay(1)

> measuredBioAssay(ba)[[1]]

identifier: DJCV4V01:MeasuredBioAssay:MPRO_0hr_A
name: MPRO_0hr_A
propertySets_assnlist: PropertySets_assnlist(1)
auditTrail_assnlist: AuditTrail_assnlist(1)
featureExtraction_assn: FeatureExtraction_assn(1)
measuredBioAssayData_assnreflist: MeasuredBioAssayData_assnreflist(1)
```

The objects can also be queried with `xclass`, and more directly with `xpathApply`.

```
> xml <- readXml(mageFiles[[1]])
> xclass(xml, "//MeasuredBioAssay")[[1]]

identifier: DJCV4V01:MeasuredBioAssay:MPRO_0hr_A
name: MPRO_0hr_A
propertySets_assnlist: PropertySets_assnlist(1)
auditTrail_assnlist: AuditTrail_assnlist(1)
featureExtraction_assn: FeatureExtraction_assn(1)
measuredBioAssayData_assnreflist: MeasuredBioAssayData_assnreflist(1)

> sapply(xclass(xml, "//Protocol/*/Parameter"), name)[1:10]

 [1] "Operator"         "Scan Date"        "Scan Comments"    "Scanner Type"
 [5] "Scanner ID"       "Scan Filter"      "Scan Temperature" "Scan Pixel Size"
 [9] "Scan Line Length" "Scan # of Lines"

> xpathApply(xml, "//MeasuredBioAssay/@name", xmlValue)

[[1]]
[1] "MPRO_0hr_A"
```

Note that MAGE attribute names are lower-case, in contrast to ARR attribute names. A document describing XPaths to important MAGE attributes is referenced below. With this in hand, one can obtain, for instance, a named list of all hardware parameters

```
> hq <- "//Hybridization/*/ProtocolApplication/*/HardwareApplication/*/ParameterValue"
> hval <- xpathApply(xml, paste(hq, "/@value"), xmlValue)
> names(hval) <- xpathApply(xml, paste(hq, "//@identifier"), xmlValue)
> hval

$`Affymetrix.com:Parameter:FluidicsStationNumber`
[1] "0"

$`Affymetrix.com:Parameter:FluidicsStationPosition`
[1] "0"

$`Affymetrix.com:Parameter:FluidicsStationID`
[1] "1819113518"

$`Affymetrix.com:Parameter:FluidicsStationStage`
[1] "Completed"
```

3 Additional resources

The package includes the document type definition used for automatic class generation.

```
> dtdDir <- system.file("extdata", package = "AffyCompatible")
> list.files(dtdDir, pattern = ".*dtd")

[1] "ArraySetAndTemplateFile.dtd" "MAGE-ML.dtd"
[3] "NetAffxAnnotFileList.dtd"
```

XPath information for MAGE attributes used by Affymetrix are defined at `http://www.affymetrix.com/support/developer/dtt_sdk/index.affx?terms=no`.