curatedBladderData

Markus Riester

April 18, 2015

Contents

1 curatedBladderData: Clinically Annotated Data for the Bladder Cancer Transcriptome

This package represents a manually curated data collection for gene expression meta-analysis of patients with bladder cancer. This resource provides uniformly prepared microarray data with curated and documented clinical metadata. It allows a computational user to efficiently identify studies and patient subgroups of interest for analysis and to run such analyses immediately without the challenges posed by harmonizing heterogeneous microarray technologies, study designs, expression data processing methods, and clinical data formats.

In this vignette, we give a short tour of the package and will show how to use it efficiently.

2 Load data sets

Loading a single dataset is very easy. First we load the package:

```
> library(curatedBladderData)
```

To get a listing of all the datasets, use the `data` function:

```
> data(package="curatedBladderData")
```

Now to load a single dataset, we use the `data` function again:

```
> data(GSE89_eset)
> GSE89_eset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 5466 features, 40 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM2505 GSM2506 ... GSM2544 (40 total)
  varLabels: alt_sample_name unique_patient_ID ... uncurated_author_metadata (31
    total)
  varMetadata: labelDescription
featureData
  featureNames: A2M AADAC ... ZYX (5466 total)
  fvarLabels: probeset gene
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
  pubMedIds: 12469123
Annotation: hu6800
```

The datasets are provided as Bioconductor `ExpressionSet` objects and we refer to the Bioconductor documentation for users unfamiliar with this data structure.

3 Load datasets based on rules

For a meta-analysis, we typically want to filter datasets and patients to get a population of patients we are interested in. We provide a short but powerful R script that does the filtering and provides the data as a list of `ExpressionSet` objects. One can use this script within R by first sourcing a config file which specifies the filters, like the minimum numbers of patients in each dataset. It is also possible to filter samples by annotation, for example to remove early stage and normal samples.

```
> source(system.file("extdata",
+ "patientselection_all.config",package="curatedBladderData"))
> ls()

[1] "GSE89_eset"          "keep.common.only"    "meta.required"
[4] "min.number.of.events" "min.sample.size"     "package.name"
[7] "quantile.cutoff"      "rescale"             "strict.checking"
```

See what the values of these variables we have loaded are. The variable names are fairly descriptive, but note that "rule.1" is a character vector of length 2, where the first entry is the name of a clinical data variable, and the second entry is a Regular Expression providing a requirement for that variable. Any number of rules can be added, with increasing identifiers, e.g. "rule.2", "rule.3", etc.

Here strict.checking is FALSE, meaning that samples not annotated for the variables in these rules are allowed to pass the filter. If strict.checking == TRUE, samples missing this annotation will be removed.

```
> sapply(ls(), get)

$GSE89_eset
ExpressionSet (storageMode: lockedEnvironment)
assayData: 5466 features, 40 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM2505 GSM2506 ... GSM2544 (40 total)
```

```
  varLabels: alt_sample_name unique_patient_ID ... uncurated_author_metadata (31
    total)
  varMetadata: labelDescription
featureData
  featureNames: A2M AADAC ... ZYX (5466 total)
  fvarLabels: probeset gene
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
  pubMedIds: 12469123
Annotation: hu6800


$keep.common.only
[1] FALSE


$meta.required
NULL


$min.number.of.events
[1] 0


$min.sample.size
[1] 1


$package.name
[1] "curatedBladderData"


$quantile.cutoff
[1] 0


$rescale
[1] FALSE


$strict.checking
[1] FALSE
```

Now that we have defined the sample filter, we create a list of ExpressionSets by sourcing the `createEsetList.R` file:

```
> source(system.file("extdata", "createEsetList.R", package =
+ "curatedBladderData"))

2015-04-18 09:05:52 INFO::Inside script createEsetList.R - inputArgs =
2015-04-18 09:05:52 INFO::
2015-04-18 09:05:52 INFO::Loading curatedBladderData 1.4.0
2015-04-18 09:06:21 INFO::Clean up the esets.
2015-04-18 09:06:21 INFO::including GSE13507_eset
2015-04-18 09:06:21 INFO::including GSE1827_eset
2015-04-18 09:06:22 INFO::including GSE19915.GPL3883_eset
2015-04-18 09:06:22 INFO::including GSE19915.GPL5186_eset
2015-04-18 09:06:22 INFO::including GSE31189_eset
2015-04-18 09:06:22 INFO::including GSE31684_eset
2015-04-18 09:06:22 INFO::including GSE32894_eset
2015-04-18 09:06:23 INFO::including GSE37317_eset
2015-04-18 09:06:23 INFO::including GSE5287_eset
2015-04-18 09:06:23 INFO::including GSE89_eset
2015-04-18 09:06:23 INFO::including PMID17099711.GPL8300_eset
```

```
2015-04-18 09:06:23 INFO::including PMID17099711.GPL91_eset
2015-04-18 09:06:23 INFO::Ids with missing data: GSE1827_eset, GSE19915.GPL3883_eset, GSE19915.GPL5186_eset
```

It is also possible to run the script from the command line and then load the R data file within R:

```
R --vanilla "--args patientselection.config ovarian.eset.rda tmp.log"  < createEsetList.R
```

Now we have 12 datasets with samples that passed our filter in a list of ExpressionSets called esets:

```
> names(esets)
 [1] "GSE13507_eset"         "GSE1827_eset"          "GSE19915.GPL3883_eset"
 [4] "GSE19915.GPL5186_eset" "GSE31189_eset"         "GSE31684_eset"
 [7] "GSE32894_eset"         "GSE37317_eset"         "GSE5287_eset"
[10] "GSE89_eset"            "PMID17099711.GPL8300_eset" "PMID17099711.GPL91_eset"
```

4   Non-unique gene symbols

In the standard version of curatedBladderData (the version available on Bioconductor), we collapse manufacturer probesets to official HGNC symbols using the Biomart database. Some probesets are mapped to multiple HGNC symbols in this database. For these probesets, we provide all the symbols. For example 220159_at maps to *ABCA11P* and *ZNF721* and we provide ABCA11P///ZNF721 as probeset name. If you have an array of gene symbols for which you want to access the expression data, "ABCA11P" would not be found in curatedBladderData in this example. The following function will create a new ExpressionSet in which both *ZNF721* and *ABCA11P* are features with identical expression data:

```
> expandProbesets <- function (eset, sep = "///")
+ {
+     x <- lapply(featureNames(eset), function(x) strsplit(x, sep)[[1]])
+     eset <- eset[order(sapply(x, length)), ]
+     x <- lapply(featureNames(eset), function(x) strsplit(x, sep)[[1]])
+     idx <- unlist(sapply(1:length(x), function(i) rep(i, length(x[[i]]))))
+     xx <- !duplicated(unlist(x))
+     idx <- idx[xx]
+     x <- unlist(x)[xx]
+     eset <- eset[idx, ]
+     featureNames(eset) <- x
+     eset
+ }
> X <- GSE89_eset[head(grep("///", featureNames(GSE89_eset))),]
> exprs(X)[,1:3]
                          GSM2505    GSM2506    GSM2507
ACOT1///ACOT2            11.324354 11.149031 10.847654
ACTBP11///ACTB          15.257080 15.123444 15.008748
ACY1///ABHD14A-ACY1     10.733768 12.309362 11.906228
ADH1A///ADH1B///ADH1C    8.899369  9.618607 10.684844
AGAP11///ADIRF          11.769862 15.264290 14.838765
AK4P3///AK4P1///AK4     10.273964  8.519981  8.572399

> exprs(expandProbesets(X))[,1:3]

                  GSM2505    GSM2506    GSM2507
ACOT1           11.324354 11.149031 10.847654
ACOT2           11.324354 11.149031 10.847654
ACTBP11         15.257080 15.123444 15.008748
ACTB            15.257080 15.123444 15.008748
ACY1            10.733768 12.309362 11.906228
ABHD14A-ACY1    10.733768 12.309362 11.906228
AGAP11          11.769862 15.264290 14.838765
ADIRF           11.769862 15.264290 14.838765
ADH1A            8.899369  9.618607 10.684844
ADH1B            8.899369  9.618607 10.684844
ADH1C            8.899369  9.618607 10.684844
AK4P3           10.273964  8.519981  8.572399
AK4P1           10.273964  8.519981  8.572399
AK4             10.273964  8.519981  8.572399
```
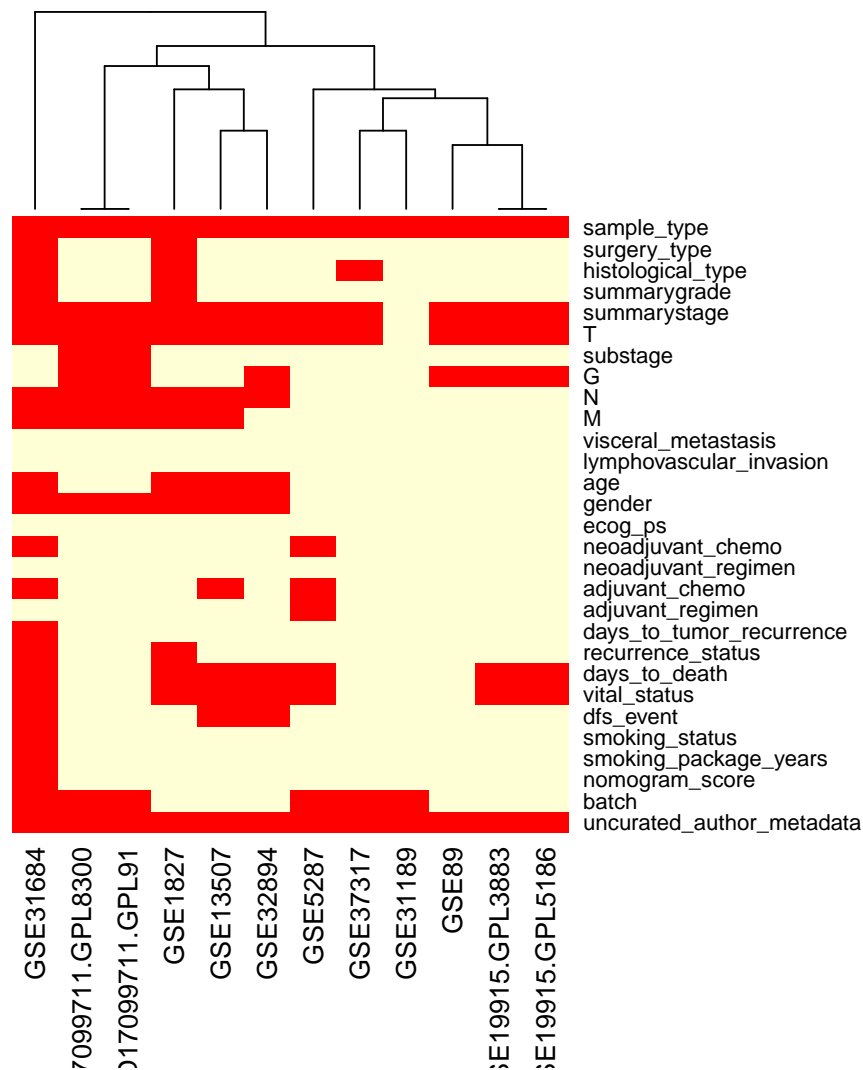
A   Available Clinical Characteristics

Figure 1: Available clinical annotation. This heatmap visualizes for each curated clinical characteristic (rows) the availability in each dataset (columns). Red indicates that the corresponding characteristic is available for at least one sample in the dataset. This plot is Figure 2 of the curatedBladderData manuscript.

B   Summarizing the List of ExpressionSets

This example provides a table summarizing the datasets being used, and is useful when publishing analyses based on curatedBladderData. First, define some useful functions for this purpose:

```
> source(system.file("extdata", "summarizeEsets.R", package =
+     "curatedBladderData"))
```

Optionally write this table to file, for example (replace myfile <- tempfile() with something like myfile <- "nicetable.csv")

```
> (myfile <- tempfile())
```

```
[1] "/tmp/RtmpgQj56M/file503441423ae6"

> write.table(summary.table, file=myfile, row.names=FALSE, quote=TRUE, sep=",")
```

C   For non-R users

If you are not doing your analysis in R, and just want to get some data you have identified from the curatedBladderData manual, here is a simple way to do it. For one dataset:

```
> library(curatedBladderData)
> library(affy)
> data(GSE89_eset)
> write.csv(exprs(GSE89_eset), file="GSE89_eset_exprs.csv")
> write.csv(pData(GSE89_eset), file="GSE89_eset_clindata.csv")
```

Or for several datasets:

```
> data.to.fetch <- c("GSE89_eset", "GSE37317_eset")
> for (onedata in data.to.fetch){
+     print(paste("Fetching", onedata))
+     data(list=onedata)
+     write.csv(exprs(get(onedata)), file=paste(onedata, "_exprs.csv", sep=""))
+     write.csv(pData(get(onedata)), file=paste(onedata, "_clindata.csv", sep=""))
+ }
```

D   Session Info

- R version 3.2.0 (2015-04-16), x86_64-unknown-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
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.28.0, BiocGenerics 0.14.0, affy 1.46.0, curatedBladderData 1.4.0, genefilter 1.50.0, logging 0.7-103, mgcv 1.8-6, nlme 3.1-120, survival 2.38-1, sva 3.14.0, xtable 1.7-4
- Loaded via a namespace (and not attached): AnnotationDbi 1.30.0, BiocInstaller 1.18.1, BiocStyle 1.6.0, DBI 0.3.1, GenomeInfoDb 1.4.0, IRanges 2.2.0, Matrix 1.2-0, RSQLite 1.0.0, S4Vectors 0.6.0, XML 3.98-1.1, affyio 1.36.0, annotate 1.46.0, grid 3.2.0, lattice 0.20-31, preprocessCore 1.30.0, splines 3.2.0, stats4 3.2.0, tools 3.2.0, zlibbioc 1.14.0

	PMID	N samples	stage	histology	Platform
GSE13507	20059769	255	103/62/90	0/0/0/255	Illumina human-6 v2.0
GSE1827	15930339	80	27/53/0	74/0/6/0	JAKE
GSE19915.GPL3883	20406976	84	73/2/9	0/0/0/84	Swegene Human 27K
GSE19915.GPL5186	20406976	98	43/45/10	0/0/0/98	SWEGENE H-v3.0.1 35K
GSE31189	23097579	92	0/0/92	0/0/0/92	Affymetrix HG-U133Plus2
GSE31684	22228636	93	15/78/0	86/2/5/0	Affymetrix HG-U133Plus2
GSE32894	22553347	308	213/93/2	0/0/0/308	Illumina HumanHT-12 V3.0
GSE37317	22586063	19	8/11/0	18/0/1/0	Affymetrix HG-U133A
GSE5287	17671123	30	0/30/0	0/0/0/30	Affymetrix HG-U133A
GSE89	12469123	40	31/9/0	0/0/0/40	Affymetrix HuGeneFL
PMID17099711.GPL8300	17099711	30	16/14/0	0/0/0/30	Affymetrix U95Av2
PMID17099711.GPL91	17099711	31	9/17/5	0/0/0/31	Affymetrix U95A

Table 1: Datasets provided by curatedBladderData.