curatedBladderData

Markus Riester

November 1, 2026

Contents

1 curatedBladderData: Clinically Annotated Data for the Bladder Cancer Transcriptome . . . 1
2 Data access and caching . . . 2
3 Load data sets . . . 2
4 Load datasets based on rules . . . 3
5 Non-unique gene symbols . . . 6
A Available Clinical Characteristics . . . 7
B Summarizing the List of ExpressionSets . . . 7
C For non-R users . . . 10
D Session Info . . . 10

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 Data access and caching

As of version 1.49.1 the datasets are no longer stored inside the package. They are hosted on Zenodo and downloaded individually the first time they are requested, then cached locally with *BiocFileCache*, so later uses need no network access. The cache lives in a package-specific directory, `tools::R_user_dir("curatedBladderData", "cache")`; to reclaim disk space, delete that directory.

The legacy `data(GSE89_eset)` syntax still works, downloading through the same cache, but is deprecated in favour of the `curatedBladderData()` function shown below.

Throughout this vignette we pass `test = TRUE` (and set `test.mode <- TRUE`), which loads small offline subsets bundled with the package so that the vignette builds without downloading data. For real analyses, omit these settings.

3 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, call `curatedBladderData()` with no arguments:

```
> curatedBladderData()
```

```
 [1] "GSE13507_eset"             "GSE1827_eset"
 [3] "GSE19915.GPL3883_eset"     "GSE19915.GPL5186_eset"
 [5] "GSE31189_eset"             "GSE31684_eset"
 [7] "GSE32894_eset"             "GSE37317_eset"
 [9] "GSE5287_eset"              "GSE89_eset"
[11] "PMID17099711.GPL8300_eset" "PMID17099711.GPL91_eset"
```

Now to load a single dataset (drop `test = TRUE` to download the full dataset):

```
> GSE89_eset <- curatedBladderData("GSE89_eset", test = TRUE)
> GSE89_eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 202 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: CXCL12 CXCR4 ... ALPP (202 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.

4 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"))
> test.mode <- TRUE  # use the bundled offline subsets in this vignette
> 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"
[10] "test.mode"
```

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: 202 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: CXCL12 CXCR4 ... ALPP (202 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

$test.mode
[1] TRUE
```

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"))

2026-11-01 08:50:27.80303 INFO::Inside script createEsetList.R - inputArgs =
2026-11-01 08:50:27.824241 INFO::Loading curatedBladderData 1.49.1
2026-11-01 08:50:28.094071 INFO::Clean up the esets.
2026-11-01 08:50:28.151346 INFO::including GSE13507_eset
2026-11-01 08:50:28.158802 INFO::including GSE1827_eset
2026-11-01 08:50:28.164979 INFO::including GSE19915.GPL3883_eset
2026-11-01 08:50:28.171237 INFO::including GSE19915.GPL5186_eset
2026-11-01 08:50:28.177416 INFO::including GSE31189_eset
2026-11-01 08:50:28.183654 INFO::including GSE31684_eset
2026-11-01 08:50:28.189881 INFO::including GSE32894_eset
2026-11-01 08:50:28.195869 INFO::including GSE37317_eset
2026-11-01 08:50:28.201866 INFO::including GSE5287_eset
2026-11-01 08:50:28.207793 INFO::including GSE89_eset
2026-11-01 08:50:28.213616 INFO::including PMID17099711.GPL8300_eset
2026-11-01 08:50:28.219635 INFO::including PMID17099711.GPL91_eset
2026-11-01 08:50:28.223947 INFO::Ids with missing data: GSE1827_eset, GSE19915.GPL3883_eset, GSE19915.GPL5186
```

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"
 [3] "GSE19915.GPL3883_eset"      "GSE19915.GPL5186_eset"
 [5] "GSE31189_eset"              "GSE31684_eset"
 [7] "GSE32894_eset"              "GSE37317_eset"
 [9] "GSE5287_eset"               "GSE89_eset"
[11] "PMID17099711.GPL8300_eset" "PMID17099711.GPL91_eset"
```

5 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

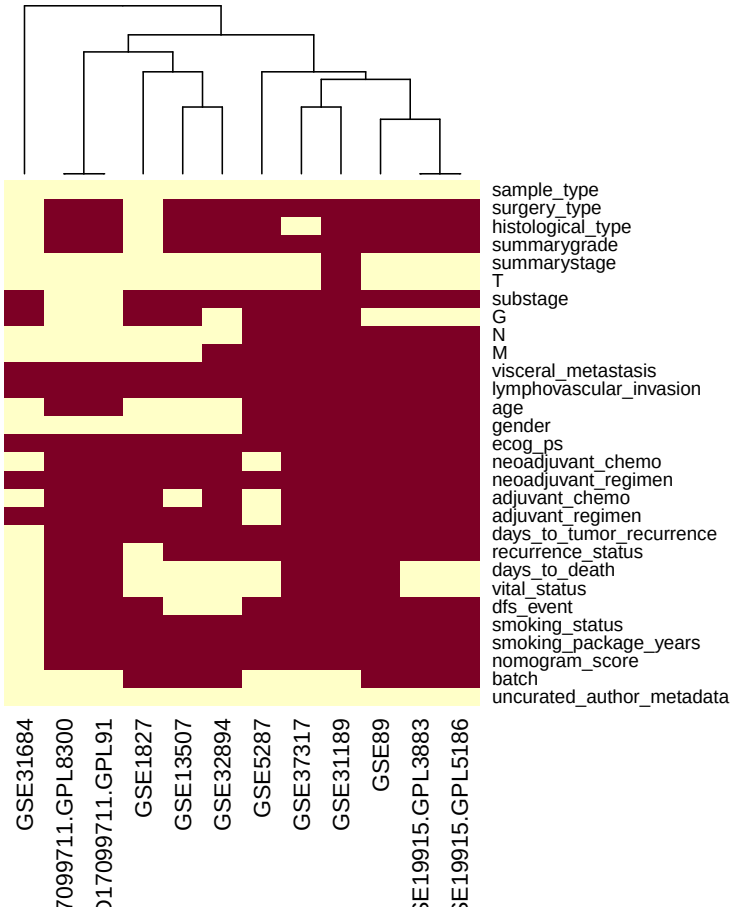


Figure 1: Available clinical annotation. This heatmap visualizes for each curated clinical characteristic (rows) the availability in each dataset (columns). Red indicates that the corresponding characteristic is available for at least one sample in the dataset. This plot is Figure 2 of the curatedBladderData 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/Rtmp0nr7pQ/file1277634de635"
```

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

	PMID	N samples	stage	histology	Platform
GSE13507	20059769	100	21/11/68	0/0/0/100	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	100	63/37/0	0/0/0/100	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.

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)
> GSE89_eset <- curatedBladderData("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))
+     assign(onedata, curatedBladderData(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 4.6.1 (2026-06-24), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `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`
- Time zone: `Etc/UTC`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 26.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.32.so` ; LAPACK version3.12.0
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: Biobase 2.73.2, BiocGenerics 0.59.12, BiocParallel 1.47.0, curatedBladderData 1.49.1, genefilter 1.95.0, generics 0.1.4, logging 0.10-111, mgcv 1.9-4, nlme 3.1-172, survival 3.8-12, sva 3.61.0, xtable 1.8-8
- Loaded via a namespace (and not attached): annotate 1.91.0, AnnotationDbi 1.75.2, BiocBaseUtils 1.15.1, BiocFileCache 3.3.1, BiocManager 1.30.27, BiocStyle 2.41.0, Biostrings 2.81.9, bit 4.6.0, bit64 4.8.6, blob 1.3.0, buildtools 1.0.0, cachem 1.1.0, cli 3.6.6, codetools 0.2-20, compiler 4.6.1, crayon 1.5.3, curl 8.1.0, DBI 1.3.0, dbplyr 2.6.0, digest 0.6.39, dplyr 1.2.1, edgeR 4.99.11, evaluate 1.0.5, fastmap 1.2.0, filelock 1.0.3, glue 1.8.1, grid 4.6.1, htmltools 0.5.9, httr 1.4.9, httr2 1.3.0,

IRanges 2.47.5, KEGGREST 1.53.6, knitr 1.52, lattice 0.23-1, lifecycle 1.0.5, limma 3.99.3, magrittr 2.0.5, maketools 1.3.2, Matrix 1.7-6, MatrixGenerics 1.25.0, matrixStats 1.5.0, memoise 2.0.1, otel 0.2.0, parallel 4.6.1, pillar 1.11.1, pkgconfig 2.0.3, png 0.1-9, R6 2.6.1, rlang 1.3.0, rmarkdown 2.32, RSQLite 3.53.3, S4Vectors 0.51.10, Seqinfo 1.3.2, splines 4.6.1, statmod 1.5.2, stats4 4.6.1, sys 3.4.3, tibble 3.3.1, tidyselect 1.2.1, tools 4.6.1, vctrs 0.7.3, xfun 0.62, XML 3.99-0.25, XVector 0.53.0, yaml 2.3.12