

# MIGSA: Getting pbcmc datasets

**Juan C Rodriguez**

CONICET

Universidad Católica de Córdoba

Universidad Nacional de Córdoba

**Cristóbal Fresno**

Instituto Nacional de Medicina Genómica

**Andrea S Llera**

CONICET

Fundación Instituto Leloir

**Elmer A Fernández**

CONICET

Universidad Católica de Córdoba

Universidad Nacional de Córdoba

---

## Abstract

In this vignette we are going to show how we got the RData *pbcmcData.RData* which can be loaded via the **MIGSAdata** package using `data(pbcmcData)`.

*Keywords:* singular enrichment analysis, over representation analysis, gene set enrichment analysis, functional class scoring, big omics data.

---

## 1. Getting the data

Following we give the used code to download this data and their PAM50 subtypes.

```
> library(limma);
> library(pbcmc);
> # datasets included in BioConductor repository
> libNames <- c("mainz", "nki", "transbig", "unt", "upp", "vdx");
> # let's load them!
> pbcmcData <- lapply(libNames, function(actLibName) {
+   print(actLibName);
+
+   # the pbcmc package provides an easy way to download and classify them
+   actLib <- loadBCDataset(Class=PAM50, libname=actLibName, verbose=FALSE);
+   actLibFilt <- filtrate(actLib, verbose=FALSE);
+   actLibFilt <- classify(actLibFilt, std="none", verbose=FALSE);
+   actSubtypes <- classification(actLibFilt)$subtype;
+
+   # get the expression matrix and the annotation
+   actExprs <- exprs(actLib);
+   actAnnot <- annotation(actLib);
+ })
```

```

+   # we recommend working allways with Entrez IDs, let's match them with
+   # expression matrix rownames (and modify them)
+   if (all(actAnnot$probe == rownames(actExprs))) {
+       actExprs <- actExprs[!is.na(actAnnot$EntrezGene.ID),];
+       actAnnot <- actAnnot[!is.na(actAnnot$EntrezGene.ID),];
+       rownames(actExprs) <- as.character(actAnnot$EntrezGene.ID);
+   } else {
+       matchedEntrez <- match(rownames(actExprs), actAnnot$probe);
+       # all(rownames(actExprs) %in% actAnnot$probe == !is.na(matchedEntrez));
+
+       stopifnot(all(
+           actAnnot$probe[!is.na(matchedEntrez)] ==
+           rownames(actExprs)[!is.na(matchedEntrez)]));
+
+       actExprs <- actExprs[!is.na(matchedEntrez),];
+       actAnnot <- actAnnot[!is.na(matchedEntrez),];
+       stopifnot(all(actAnnot$probe == rownames(actExprs)));
+       actExprs <- actExprs[!is.na(actAnnot$EntrezGene.ID),];
+       actAnnot <- actAnnot[!is.na(actAnnot$EntrezGene.ID),];
+       rownames(actExprs) <- as.character(actAnnot$EntrezGene.ID);
+   }
+
+   # average repeated genes expression
+   actExprs <- avereps(actExprs);
+
+   stopifnot(all(colnames(actExprs) == names(actSubtypes)));
+   # filtrate only these two conditions
+   actExprs <- actExprs[, actSubtypes %in% c("Basal", "LumA")];
+   actSubtypes <- as.character(
+       actSubtypes[actSubtypes %in% c("Basal", "LumA")]);
+
+   return(list(geneExpr=actExprs, subtypes=actSubtypes));
+ })
> names(pbcmcData) <- libNames;

```

And let's check it is the same data.

```

> # save the just created pbcmcData to newPbcmcData
> newPbcmcData <- pbcmcData;
> library(MIGSAdata);
> # and load the MIGSAdata one.
> data(pbcmcData);
> all.equal(newPbcmcData, pbcmcData);

```

## Session Info

```
> sessionInfo()
```

```
R version 4.1.0 RC (2021-05-10 r80283)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server x64 (build 17763)
```

```
Matrix products: default
```

```
locale:
```

```
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252
```

```
attached base packages:
```

```
[1] stats4      parallel  stats      graphics  grDevices  utils      datasets
[8] methods     base
```

```
other attached packages:
```

```
[1] edgeR_3.34.0      MIGSAdat_1.15.0    MIGSA_1.16.0
[4] mGSZ_1.0          ismev_1.42         mgcv_1.8-35
[7] nlme_3.1-152      MASS_7.3-54        limma_3.48.0
[10] GSA_1.03.1        BiocParallel_1.26.0 GSEABase_1.54.0
[13] graph_1.70.0      annotate_1.70.0     XML_3.99-0.6
[16] AnnotationDbi_1.54.0 IRanges_2.26.0     S4Vectors_0.30.0
[19] Biobase_2.52.0    BiocGenerics_0.38.0
```

```
loaded via a namespace (and not attached):
```

```
[1] Category_2.58.0    bitops_1.0-7        matrixStats_0.58.0
[4] bit64_4.0.5        httr_1.4.2          GenomeInfoDb_1.28.0
[7] Rgraphviz_2.36.0   tools_4.1.0         utf8_1.2.1
[10] R6_2.5.0           vegan_2.5-7         DBI_1.1.1
[13] colorspace_2.0-1   permute_0.9-5       tidyselect_1.1.1
[16] bit_4.0.4          compiler_4.1.0      formatR_1.9
[19] gg dendro_0.1.22    labeling_0.4.2      scales_1.1.1
[22] genefilter_1.74.0  RBGL_1.68.0         digest_0.6.27
[25] stringr_1.4.0      AnnotationForge_1.34.0 XVector_0.32.0
[28] pkgconfig_2.0.3    fastmap_1.1.0       rlang_0.4.11
[31] rstudioapi_0.13    RSQLite_2.2.7       farver_2.1.0
[34] G0stats_2.58.0     generics_0.1.0      jsonlite_1.7.2
[37] dplyr_1.0.6        RCurl_1.98-1.3      magrittr_2.0.1
[40] G0.db_3.13.0       GenomeInfoDbData_1.2.6 futile.logger_1.4.3
[43] Matrix_1.3-3       Rcpp_1.0.6          munsell_0.5.0
[46] fansi_0.4.2        lifecycle_1.0.0     stringi_1.6.2
[49] zlibbioc_1.38.0    org.Hs.eg.db_3.13.0 plyr_1.8.6
[52] grid_4.1.0         blob_1.2.1          crayon_1.4.1
```

[55]	lattice_0.20-44	Biostrings_2.60.0	splines_4.1.0
[58]	KEGGREST_1.32.0	locfit_1.5-9.4	pillar_1.6.1
[61]	reshape2_1.4.4	futile.options_1.0.1	glue_1.4.2
[64]	lambda.r_1.2.4	data.table_1.14.0	png_0.1-7
[67]	vctrs_0.3.8	gtable_0.3.0	purrr_0.3.4
[70]	assertthat_0.2.1	cachem_1.0.5	ggplot2_3.3.3
[73]	xtable_1.8-4	survival_3.2-11	snow_0.4-3
[76]	tibble_3.1.2	memoise_2.0.0	cluster_2.1.2
[79]	ellipsis_0.3.2		

**Affiliation:**

Juan C Rodriguez & Elmer A Fernández

Bioscience Data Mining Group

Facultad de Ingeniería

Universidad Católica de Córdoba - CONICET

X5016DHK Córdoba, Argentina

E-mail: [jcrodriguez@bdmg.com.ar](mailto:jcrodriguez@bdmg.com.ar), [efernandez@bdmg.com.ar](mailto:efernandez@bdmg.com.ar)

URL: <http://www.bdmg.com.ar/>