# Using the inSilicoDb package

Jonatan Taminau*

CoMo, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium

## 1 The inSilico database

With more than 500,000 genomic profiles freely available in the public domain, there is a huge amount of information accessible for computational biologists or bioinformaticians. However, accessibility to these data requires complex computational steps. Manual parsing of annotations and keywords, which is in most cases a necessary evil, tends to be time-consuming and is known to be error-prone. Also the wide variety of normalization and preprocessing methods makes the comparability of different existing studies hard, or even impossible. The inSilico initiative (`http://insilico.ulb.ac.be`) provides an answer to those problems with its freely available web-based database tool: the inSilico database[1]. Starting with all public available human Affymetrix studies from Gene Expression Omnibus (GEO) [1] it provides those studies in a consistent and well curated form. With a direct connection to GenePattern [3] and the ability to export the data to different formats, the inSilico database is an efficient mean to re-analyse public datasets and improve reproducibility in genome-wide research.

To further ease the use of this vast amount of genomic data the inSilicoDb R package was developed. This package can be seen as a different front-end to the core inSilico database and, although it provides only limited functionality compared to the web-based tool, it can become very valuable for R programmers or anyone who is interested in large scale analysis using automated scripting.

Similar packages to retrieve gene expression data in R exist [5, 4], but the added value and strength of this package is tightly connected to the innovative concept of the inSilico database and will therefore circumvent common obstacles like incompatibility and missing or malformed annotations.

*`jtaminau@vub.ac.be`

[1] Manuscript in preparation

# 2 Getting started using inSilicoDb

As this section will show, accessing data from the inSilico database is surprisingly easy and straightforward.

## 2.1 Simple access

Suppose one is interested in a number of publicly available gene expression studies which he found while browsing Gene Expression Omnibus (GEO) or the inSilico database. Using only the GSE identifier, a completely annotated and formatted dataset can be downloaded in just seconds, without any need for further manual parsing:

```
> library("inSilicoDb");
> res = getDatasets("GSE4635");
> eset = res[[1]];
```

The result of `getDataset` is a list, containing a Bioconductors ExpressionSet (`eset`) for every platform that exists for this dataset (in the example there is only one platform). An alternative approach to obtain the same data is to specify the platform. In this case no list but the expression set is directly returned:

```
> eset = getDataset("GSE4635", "GPL96");
```

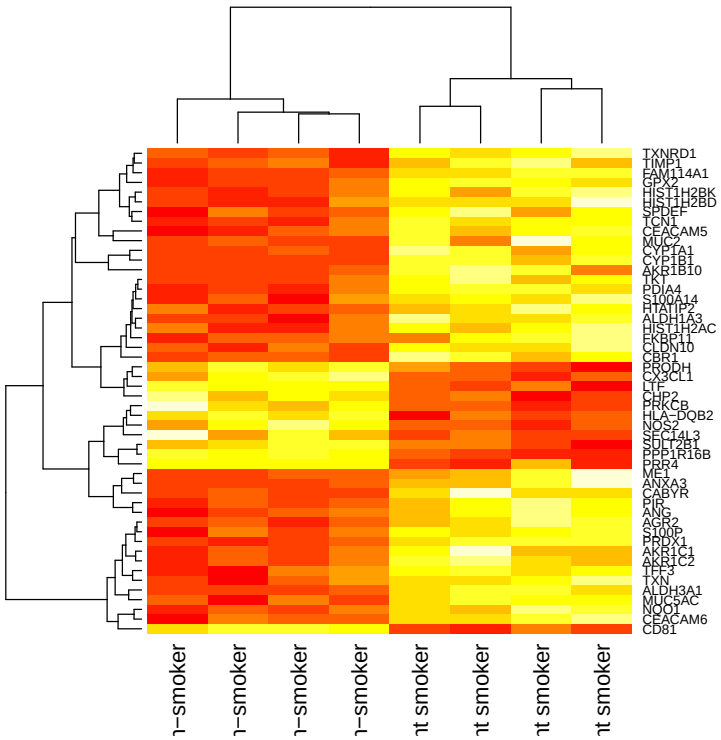
And in case the platform is unknown, the auxiliary function `getPlatforms` is provided:

```
> platforms = getPlatforms("GSE4635");
> print(platforms);

[1] "GPL96"
```

Once an expression set is retrieved, all available Bioconductor packages can be applied for further analysis, as the following code illustrates:

```
> #eset = getDataset("GSE4635", "GPL96",  features = "gene");
> #heatmap(exprs(eset)[1:100,]);
> library("limma")
> eset = getDataset("GSE4635", "GPL96",
+                   norm="FRMA", features = "gene");
> # Find 50 most discriminating genes
> f = pData(eset)[ ,"Smoker"]
> design = model.matrix(~f);
> fit = eBayes(lmFit(eset,design));
> t = topTable(fit, coef=2, number=50);
> selected = is.element(rownames(exprs(eset)),
+                           t[ ,"SYMBOL"])
> eset = eset[selected, ];
> labels = pData(eset)[,"Smoker"];
> heatmap(exprs(eset), labCol=labels);
```




*(Only the first 100 genes are printed for simplicity).*

In case only the annotation information is needed and there is no need for the numerical data, the `getAnnotations` function also exists for convenience:

```
> annot = getAnnotations("GSE4635", "GPL96");
> pData(annot);
```

```
          Age    Sex Anatomical Site                 Cell Type         Smoker
GSM15729  N/A    N/A  bronchus, lung bronchial epithelial cell     non-smoker
GSM104072 N/A    N/A  bronchus, lung bronchial epithelial cell current smoker
GSM104074 N/A    N/A  bronchus, lung bronchial epithelial cell current smoker
GSM104075  45 female  bronchus, lung                       N/A current smoker
GSM104076 N/A    N/A  bronchus, lung bronchial epithelial cell current smoker
GSM104078 N/A    N/A  bronchus, lung bronchial epithelial cell     non-smoker
GSM104080 N/A    N/A  bronchus, lung bronchial epithelial cell     non-smoker
GSM104082 N/A    N/A  bronchus, lung bronchial epithelial cell     non-smoker
                      Ethnicity Cigarette Consumption (pack/years) platform
GSM15729                    N/A                                N/A    GPL96
GSM104072                   N/A                                N/A    GPL96
GSM104074                   N/A                                N/A    GPL96
GSM104075 african-american (AFA)                                 14    GPL96
GSM104076                   N/A                                N/A    GPL96
```

```
GSM104078                N/A                              N/A    GPL96
GSM104080                N/A                              N/A    GPL96
GSM104082                N/A                              N/A    GPL96
```

## 2.2 More options

By default all numerical data is retrieved the same way the original authors have submitted the data to GEO and can therefore have been processed by a wide variety of preprocessing methods. However, when combining different studies a consistent preprocessing is required and therefore all studies for which there are CEL files available, are also precomputed by applying the FRMA preprocessing method [2]. The user can retrieve those studies as fast and easy as the original ones, simply by using the optional `norm` parameter.

```
> eset = getDataset("GSE4635", "GPL96", norm="FRMA");
```

All gene expression matrices contain probes as features, although it is also possible to retrieve the genes instead. This probe to gene mapping is precomputed for every dataset and can be selected using the `genes` parameter. By default probes are selected, as this is how the data was submitted to GEO.

```
> eset = getDataset("GSE4635", "GPL96");
> print(nrow(eset));

Features
   22215

> eset = getDataset("GSE4635", "GPL96", features="gene");
> print(nrow(eset));

Features
   12698
```

## 2.3 Create your own loop...

One of the advantages of retrieving data through R is the possibility to develop a whole automated workflow in just a few lines of code. The following example illustrates the many opportunities researchers can have using this tool.

In the example code below, we iterate over a list of series GSE identifiers and try to retrieve every dataset from the database. Once retrieved some basic analysis is performed (printing the number of annotations and missing values). Note that the `getDataset` function can throw an error (e.g. no internet connection, dataset is not available, etc.) which is best caught in a try-catch loop, as is shown in the example.

```
> lst = list("GSE4635", "GSExxx", "GSE781");
> gpl = "GPL96";
```

```
> for(gse in lst)
+ {
+   catn = function(...) { cat(...,"\n"); }
+   catn("Processing",gse);
+   catn("======================");
+   eset = tryCatch({getDataset(gse, gpl);},
+                   error = function(x) { print(as.character(x)); NULL; });
+   if(is.null(eset)) { next; }
+   catn("Number of annotations:");
+   catn(ncol(pData(eset)));
+   catn("Number of missing values:");
+   catn(sum(is.na(exprs(eset))));
+ }

Processing GSE4635
======================
Number of annotations:
8
Number of missing values:
0
Processing GSExxx
======================
[1] "Error: Stopped because of previous errors\n"
Processing GSE781
======================
Number of annotations:
24
Number of missing values:
0
```

# 3 Conclusion

This package is built in addition to a very powerful web-based database tool for genomic analysis. Despite its simplicity, it captures many of the benefits of this tool and provides the typical R users efficient means of performing large scale genomic analysis using automated scripting.

# 4 Session Info

```
> sessionInfo()

R version 3.1.0 (2014-04-10)
Platform: x86_64-apple-darwin13.1.0 (64-bit)

locale:
```

```
[1] C/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] limma_3.20.1        inSilicoDb_2.0.1    RCurl_1.95-4.1
[4] bitops_1.0-6        Biobase_2.24.0      BiocGenerics_0.10.0
[7] rjson_0.2.13

loaded via a namespace (and not attached):
[1] tools_3.1.0
```

# References


[1] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res*, 30(1):207–10, Jan 2002.

[2] Matthew N McCall, Benjamin M Bolstad, and Rafael A Irizarry. Frozen robust multiarray analysis (fRMA). *Biostatistics*, 11(2):242–53, Apr 2010.

[3] Michael Reich, Ted Liefeld, Joshua Gould, Jim Lerner, Pablo Tamayo, and Jill P Mesirov. Genepattern 2.0. *Nat Genet*, 38(5):500–1, May 2006.

[4] Davis Sean and Paul S Meltzer. GEOquery: a bridge between the gene expression omnibus (GEO) and bioconductor. *Bioinformatics*, 23(14):1846–7, Jul 2007.

[5] Yuelin Zhu, Sean Davis, Robert Stephens, Paul S Meltzer, and Yidong Chen. GEOmetadb: powerful alternative search engine for the gene expression omnibus. *Bioinformatics*, 24(23):2798–800, Dec 2008.