

ChemmineR-V2: Analysis of Small Molecule and Screening Data

Yiqun Cao, Tyler Backman, Yan Wang, Thomas Girke

February 19, 2011

Introduction

ChemmineR is an R package for analyzing small molecule and screening data. It contains functions for processing SDFs (structure data files), structural similarity searching, clustering/diversity analyses of compound libraries with a wide spectrum of algorithms and utilities for managing complex data sets from high-throughput compound bio-assays (Carhart et al., 1985; Chen and Reynolds, 2002). In addition, it offers visualization functions for compound clustering results and chemical structures. The integration of chemoinformatic tools with the R programming environment has many advantages, such as easy access to a wide spectrum of statistical methods, machine learning algorithms and graphic utilities. The first version of this package was published in Cao et al. (2008). Since then many additional utilities have been added to the package and many more are under development for future releases.

Getting Started

Installation

The R software for running ChemmineR can be downloaded from CRAN (<http://cran.at.r-project.org/>). The ChemmineR package can be installed from R using the `bioLite` install command.

```
> source("http://bioconductor.org/biocLite.R")
> biocLite(ChemmineR)
```

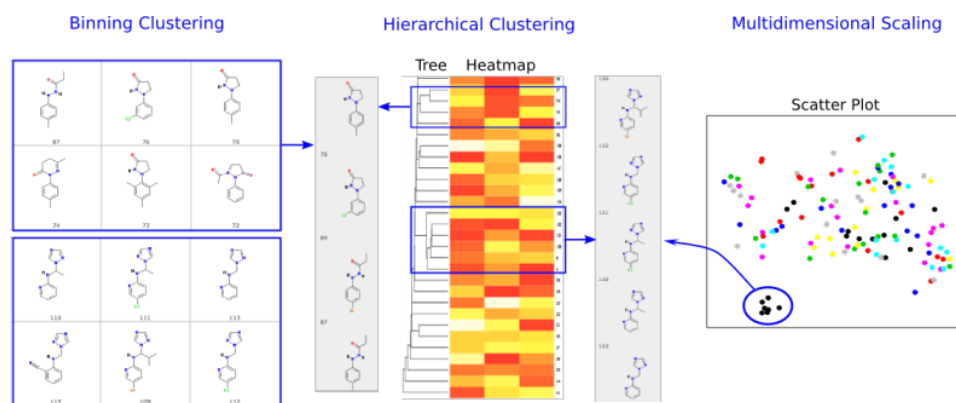


Figure 1: Overview of functions provided in the *ChemmineR* package.

Loading the Package and Documentation

```
> library(ChemmineR) # Loads the package
> library(help=ChemmineR) # Lists all functions and classes
> vignette("ChemmineR") # Opens this PDF manual from R
```

One Minute Tutorial

The following code gives an interactive overview of the most important functionalities provided by ChemmineR. It creates an instance of the `SDFset` class using as sample data set the first 100 compounds from PubChem's SD file "Compound_00650001_00675000.sdf.gz" (<ftp://ftp.ncbi.nih.gov/pubchem/Compound/CURRENT-Full/SDF/>).

```
> data(sdfsampl); sdfset <- sdfsampl
> sdfset
> sdfset[[1]] # To view summary of many SDFs use: view(sdfset[1:4])
```

The SD files can be imported with the `read.SDFset` function.

```
> sdfset <- read.SDFset("http://faculty.ucr.edu/~tgirke/Documents/
+                      R_BioCond/Samples/sdfsampl.sdf")
> ## Miscellaneous accessor methods for SDFset container
> header(sdfset[1:4])
> atomblock(sdfset[1:4])
> atomcount(sdfset[1:4])
```

```

> bondblock(sdfset[1:4])
> datablock(sdfset[1:4])
> ## Assigning compound IDs and keeping them unique
> cid(sdfset); sdfid(sdfset)
> unique_ids <- makeUnique(sdfid(sdfset))
> cid(sdfset) <- unique_ids
> ## Converting the data blocks in SDFset to matrix
> blockmatrix <- datablock2ma(datablocklist=datablock(sdfset)) # Converts data block
> numchar <- splitNumChar(blockmatrix=blockmatrix) # Splits to numeric and character
> numchar[[1]][1:4,]; numchar[[2]][1:4,]
> ## Compute atom frequency matrix, molecular weight and formula
> propma <- data.frame(MF=MF(sdfset), MW=MW(sdfset), atomcountMA(sdfset))

```

Single Compound Import from SDF

The `cmp.parse1` function will parse an SDF for a single compound. Similarly as before, the only argument required is the path (or URL) to the SDF.

Descriptor Database Content

The descriptors of compounds are stored as numeric vectors in a list object along with available annotation information about the database. You may skip this section if you are not interested in internals of descriptor database.

The `cmp.parse1` function parses the SDF of a single compound, generates the descriptors and stores them in a numeric vector. Each entry of the vector is a descriptor for this compound.

In contrast to this, the `cmp.parse` function generates a list object with four components.

The `descdb` component is a list. Each entry of the list is a vector of descriptors of one compound.

The `db.explain` function returns the descriptors in a human readable format. A single descriptor can be returned like this: The same is possible for multiple descriptors at once.

Removing Duplicated Compounds

The `cmp.duplicated` function can be used to quickly identify and remove duplicated compounds in imported compound databases. It takes a de-

scriptor database as the only required argument and returns the duplication information as a logical vector.

To demo this feature on the imported sample data set, one can create a duplication with the following command. In the next step the duplication is identified with the `cmp.duplicated` function. The `TRUE` entry in the returned logical vector indicates the duplication. It can be easily removed with the standard R subsetting syntax. In a real example one also needs to remove the duplications from the other database components.

Pairwise Compound Comparisons

The `cmp.similarity` computes the atom pair similarity between two compounds using the Tanimoto coefficient as similarity measure.

With the `cmp.similarity` function one can easily design custom search subroutines similar to the one introduced in the next section.

Similarity Searching

The `cmp.search` function searches an atom pair database for compounds that are similar to a query compound.

The function returns a data frame where the rows are sorted by similarity score (best to worst). The first column contains the indices of the matching compounds in the database. The argument `cutoff` can be a similarity cutoff, meaning only compounds with a similarity value larger than this cutoff will be returned; or it can be an integer value restricting how many compounds will be returned. If the argument `return.score` is set to `FALSE`, then the function will return a vector of indices rather than a data frame. When supplying a cutoff of 0, the function will return the similarity values for every compound in the database.

The `cmp.search` function allows to visualize the chemical structure images for the search results. A similar but more flexible chemical structure rendering function (`sdf.visualize`) is described later in this manual. By setting the `visualize` argument in `cmp.search` to `TRUE`, the matching compounds and their scores can be visualized with a standard web browser on the online ChemMine interface. Depending on the `visualize.browse` argument, an URL will be printed or a webpage will be opened showing the structures of the matching compounds along with their scores. Setting the `visualize.browse` argument to `TRUE` will automatically open the webpage in the default browser.

The query structure can also be displayed on the visualization webpage by supplying the SDF of the query in a character string or providing its file name or URL. For example, This will read the SDF provided by `query.url`, and display it as a “reference compound” at the top of the page. Part of the screenshot of the resulting output is shown in Fig. 2. A live demo is also available and linked from the online version of this manual (<http://bioweb.ucr.edu/ChemMineV2/chemminer/tutorial>).

The screenshot displays the ChemMine web interface. At the top, there are logos for UCR, IIGB, CEPCEB, ChemMine, and ChemmineR. Below these are navigation tabs: Systemics Network, GCD, Expression, POND, CWN, BAP DB, ChemMine, and Links. A left sidebar contains a menu with items like Home, Readme, 2010 Project, Protocols, CMP Sources, Search Database, Annotation, Structure, Screen Data, Workbench, Manage CMPs, Descriptors, Clustering, Clusters, Software, ChemmineR (highlighted), Links, and Login. The main content area shows three compound entries:

- Reference Compound (ka-01834)**: Includes a chemical structure and a button to "View SDF".
- (ChemmineR_Unnamed_Compound_3)**: Includes a chemical structure, a "View SDF" button, and a similarity score of 0.550335570469799.
- (ChemmineR_Unnamed_Compound_1)**: Includes a chemical structure, a "View SDF" button, and a similarity score of 0.484662576687117.

Figure 2: `cmp.search` can automatically upload the structures and scores of matching compounds to ChemMine for visualization.

Any information uploaded to *ChemMine* by *ChemmineR* is kept private

and secure using a highly randomized URL. The visualization pages can be shared with colleagues by providing the corresponding URLs.

Rendering Chemical Structure Images

Internally, the similarity search function uses the `sdf.visualize` function to send compounds to ChemMine for structure visualization. The same function can be used to send any custom combination of compounds for visualization on ChemMine along with complex annotation and activity information. The function accepts a database and a vector of compound indices. The following example performs first a similarity search to obtain a vector of indices.

The URL stored in the `url` object points to a webpage that shows the structures of the compounds. If the `browse` argument is set to `TRUE`, then the default browser will open automatically.

In addition, one can display other information next to the structures using the `extra` argument. In the following example, a vector of character strings is assigned to `extra`, and its entries are displayed next to corresponding chemical structures.

The function also allows to list a reference compound at the top of the page. The user supplies the SDF of this reference compound in form of a character string or a file. Annotation information can also be displayed next to the reference structure.

It is also possible to display more complex tabular data next to each compound by providing a list of data frames. To demonstrate this utility, the following example creates such a list of data frames via a similarity search. Each data frame is then displayed next to the corresponding compound. The screenshot of the resulting output is shown in Fig. 3.

To generate this output, first a similarity is performed using a cutoff of 0 to obtain the similarity values between the query compound and each of the compounds in the database.

The resulting data frame will be used as annotation table for the query compound. To provide a table name, one has to embed it into a list. If a table name is not required, then there is no need to generate the list object

For each of the top 10 hits in the search result, we perform the same search to obtain the similarity values between the hit and all compounds in the database. This information will then be displayed next to the structures on the visualization page. The following step displays the complex sample data set on ChemMine.

UCR :: IIGB :: CEPCEB

ChemMine **ChemmineR**

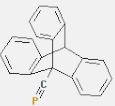
Systemics Network GCD Expression POND CWN BAP DB ChemMine Links

Home
Readme
2010 Project
Protocols
CMP Sources
Search Database
Annotation
Structure
Screen Data
Workbench
Manage CMPs
Descriptors
Clustering
Clusters
Software
ChemmineR
Links
Login

[View Previously Accessed Compounds >>>](#) Width of information table:

Reference Compound (ka-01834)

[View SDF](#) [Structure Search](#) [Add to Selection](#)

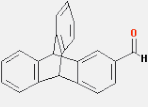


Similarities With All

ids	scores
3	0.550335570
43	0.484662577
42	0.484662577
1	0.484662577
4	0.480122324
2	0.480122324
44	0.356097561
46	0.312500000
11	0.311653117
35	0.287719298

(ChemmineR_Unnamed_Compound_3)

[View SDF](#) [Structure Search](#) [Add to Selection](#)

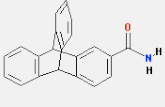


Similarities With All

ids	scores
3	1.000000000
43	0.785977860
42	0.785977860
1	0.785977860
2	0.766423358
4	0.646258503
44	0.561797753
11	0.415204678
35	0.390151515
46	0.368539326

(ChemmineR_Unnamed_Compound_43)

[View SDF](#) [Structure Search](#) [Add to Selection](#)



Similarities With All

ids	scores
43	1.000000000
42	0.840000000
1	0.840000000
3	0.785977860
2	0.709459459
4	0.611464968
44	0.601108033
11	0.390109890
46	0.339702760
35	0.323120252

Figure 3: Visualization webpage created by calling `sdf.visualize`. This page shows the table information properly rendered and displayed next to the compound structures.

Note: the `sdf.visualize` function depends on the original SDF file from which the descriptor database has been generated. If the SDF file has been moved or altered then this step cannot be used.

Any information uploaded to *ChemMine* by *ChemmineR* is kept private and secure using a highly randomized URL. The visualization pages can be shared with colleagues by providing the corresponding URLs.

Subsetting SDF Batch Files

After identifying a subset of interesting compounds, one can generate an SDF for this subset of compounds using the `sdf.subset` function.

For example, one can perform a similarity search, and use the top 10 results for subsetting. With the corresponding indices one can generate a custom SDF batch data set and store it in an external file.

One may also create a sub-database from a descriptor database using the related `db.subset` function.

Note: the `sdf.visualize` function depends on the original SDF file from which the descriptor database has been generated.

Binning Clustering

Compound libraries can be clustered into discrete similarity groups with the binning clustering function `cmp.cluster`. The function requires as input a descriptor database as well as a similarity threshold. The binning clustering result is returned in form of a data frame. Single linkage is used for cluster joining. The function calculates the required compound-to-compound distance information on the fly, while a memory-intensive distance matrix is only created upon user request via the `save.distances` argument (see below).

The previous step clusters the compounds stored in `db` with a similarity cutoff of 0.65. In other words, if two compounds share a similarity of 0.65 or above, then they will be joined into the same cluster. The first 10 rows of the result data frame are shown here:

The first column contains the compound IDs, the second the cluster size and third the cluster ID. The compound in cluster ID 1 can be returned with the following command: Similarly as above, one can visualize the chemical structures for a compound cluster of interest with the `sdf.visualize` function.

Binning Clustering with Multiple Cutoffs

Because an optimum similarity threshold is often not known, the `cmp.cluster` function can calculate cluster results for multiple cutoffs in one step with almost the same speed as for a single cutoff. The clustering results for the different cutoffs will be stored in one data frame.

One may force the `cmp.cluster` function to calculate and store the distance matrix by supplying a file name to the `save.distances` argument. The generated distance matrix can be loaded and passed on to many other clustering methods available in R, such as the hierarchical clustering function `hclust` (see below).

If a distance matrix is available, it may also be supplied to `cmp.cluster` via the `use.distances` argument. This is useful when one has a pre-computed distance matrix either from a previous call to `cmp.cluster` or from other distance calculation subroutines.

Multi-Dimensional Scaling (MDS)

To visualize and compare clustering results, the `cluster.visualize` function can be used. The function performs Multi-Dimensional Scaling (MDS) and visualizes the results in form of a scatter plot. It requires as input a descriptor database, a clustering result from `cmp.cluster`, and a cutoff for the minimum cluster size to consider in the plot. To help determining a proper cutoff size, the `cluster.sizestat` function is provided to generate cluster size statistics.

The following example uses the clustering result obtained above using cutoff values 0.65 and 0.5. By default, the `cluster.sizestat` uses the first cutoff value:

Based on this size statistics, clusters of size 3 or larger will be included in the following MDS visualization step.

By default `cluster.visualize` will draw the scatter plot in the R plotting device, and the user can interactively click a point to retrieve more information on the corresponding compounds. In the non-interactive mode (`non.interactive`), it will save the plot to a specified file in EPS or PDF format.

A 3D MDS plot can be created with the following sequence of commands.

Clustering with Other Packages

ChemmineR allows the user to take advantage of the wide spectrum of clustering utilities available in R. An example on how to perform hierarchical clustering with the `hclust` function is given below. The `cmp.cluster` function is used with the `save.distances="distmat.rda"` argument to generate a distance matrix. The matrix is saved to a file named `'distmat.rda'` and it needs to be loaded into R with the `load` function. This matrix can be directly passed on to `hclust`.

Format Interconversions between SMILES and SDF

This option will be provided in the future. At this point, SMILES strings can be imported into *ChemmineR* only indirectly by converting them into SDFs via ChemMine’s online WorkBench (<http://bioweb.ucr.edu/ChemMineV2/work/smiles/>).

Calculation of Physicochemical Descriptors

Several functions will be available in the near future for calculating physicochemical descriptors directly in *ChemmineR*. Currently, users can calculate 40 common physicochemical descriptors with the online descriptor prediction tool available on ChemMine’s WorkBench (<http://bioweb.ucr.edu/ChemMineV2/work/sdf/>).

References

- Y Cao, A Charisi, L C Cheng, T Jiang, and T Girke. *ChemmineR*: a compound mining framework for R. *Bioinformatics*, 24(15):1733–1734, Aug 2008. doi: 10.1093/bioinformatics/btn307. URL <http://www.hubmed.org/display.cgi?uids=18596077>.
- R.E. Carhart, D.H. Smith, and R. Venkataraghavan. Atom pairs as molecular features in structure-activity studies: definition and applications. *Journal of Chemical Information and Computer Sciences*, 25(2):64–73, 1985.
- X. Chen and C.H. Reynolds. Performance of Similarity Measures in 2D Fragment-Based Similarity Searching: Comparison of Structural Descriptors and Similarity Coefficients. *Journal of Chemical Information and Computer Sciences*, 42(6):1407–1414, 2002.