Cheminformatics Basics
- Structure Formats
- Similarity Searching
- Physicochemical Properties
- Clustering

Hands-on Section
- Compound Import/Export
- Object Classes
- Compound Structure Depictions
- Compound Properties
- Compound Similarity Searching
- Compound Clustering
Outline

Cheminformatics Basics
- Structure Formats
- Similarity Searching
- Physicochemical Properties
- Clustering

Hands-on Section
- Compound Import/Export
- Object Classes
- Compound Structure Depictions
- Compound Properties
- Compound Similarity Searching
- Compound Clustering
Outline

Cheminformatics Basics
Structure Formats
Similarity Searching
Physicochemical Properties
Clustering

Hands-on Section
Compound Import/Export
Object Classes
Compound Structure Depictions
Compound Properties
Compound Similarity Searching
Compound Clustering
Computations on Small Molecule Structures

Requirements
- Computer readable representations of chemical structures

Challenges
- Compounds
  - Several connection types, many branch points and/or ring closures
- DNA/protein sequences
  - Linear strings, one connection type, usually no branch points or ring closures
Utility of Structure Formats

- Nomenclature to uniquely represent chemicals
- Computer representation and manipulation
- Format interconversions
- Representation of stereochemistry and 3D formats
Most Commonly Used Structure Formats

- **Chemical nomenclature**
  - Trivial names: aspirin, acetylsalicylic acid
  - IUPAC: 2-acetoxybenzoic acid
  - InChI: `1.12Beta/C9H8O4/c1-6(10)13-8-5-3-2-4-7(8)9(11)12/h1H3,2-5H,(H,11,12)`

- **Line notations**
  - SMILES: `CC(=O)Oc1cccccc1C(=O)O`
  - Other: WLN, ROSDAL, SLN, etc.

- **Connection tables hold 3D & annotation information**
  - SDF (structure definition file)
  - MDL Molfile
  - Other: PDB, CML, etc.
Connection Table Formats: SDF and Mol

Molfile: header block and connection table (a, b)
SDfile: extension of Molfile (a, b, c)

(a) Header block
   (a1) CMP name or blank line
   (a2) software, date, 2/3D, ...
   (a3) blank line

(b) Connection table (CT)
   (b1) counts line: n atoms, n bonds, chiral, ...
   (b2) atom block: x,y,z coordinates, atoms, mass diff., charge, ...
       2D representation when z coordinates all zero
   (b3) bond block: atom 1, atom 2, bond type, stereo specs, ...
   (b4) CT delimiter

(c) Annotation data
   (c1) <data header>
   (c2) data
   (c3) blank line
   (c4) continues like c1-3
   (c5) SDF delimiter ($$$$

Cheminformatics of Drug-like Small Molecules
Cheminformatics Basics
Structure Formats
Slide 8/69
## Example: SDF Format

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>NSC85228 ethanol 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>APtc1serve02230600142D 0 0.00000 0.00000NCI NS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b1</td>
<td>9 8 0 0 0 0 0 0 0 0 0999 V2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>2.8660 -0.250 0.0000 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>3.7321 0.2500 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>4.5981 -0.250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>2.3291 0.0600 0.0000 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>4.1306 0.7249 0.0000 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>3.3335 0.7249 0.0000 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>4.2881 -0.786 0.0000 H 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>5.1350 -0.560 0.0000 H 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2</td>
<td>4.9081 0.2869 0.0000 H 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>1 2 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>2 3 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>1 4 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>2 5 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>2 6 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>3 7 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>3 8 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b3</td>
<td>3 9 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b4</td>
<td>M END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cheminformatics of Drug-like Small Molecules**  
**Cheminformatics Basics**  
**Structure Formats**  
Slide 9/69
SMILES: Simplified Molecular Input Line Entry System

- Online rendering: http://www.daylight.com/daycgi/depict
- Non-canonical SMILES for manual entry
- Canonical SMILES needs to be computer generated
- Canonicalization: single (‘correct’) representation of several possibilities
  - OCC - ethanol
  - CCO - ethanol
- Canonical format important for databases
Outline

Cheminformatics Basics
  Structure Formats
  Similarity Searching
  Physicochemical Properties
  Clustering

Hands-on Section
  Compound Import/Export
  Object Classes
  Compound Structure Depictions
  Compound Properties
  Compound Similarity Searching
  Compound Clustering
Small Molecule Similarity Concepts

How to define similarities between compounds?

- Identical structure search
- Substructure and superstructure searches
- 2D fragment similarity searching
- 3D similarity searches (e.g., pharmacophore searching)
- Graph-based approaches (e.g., maximum common substructure: MCS)
- Many additional methods
2D Fragment Similarity Search Methods

Involve two major steps

- Encode structural descriptors from compounds
  - e.g. structural keys, fingerprints, atom pairs
- Similarity measure for encoded descriptors
  - e.g. Tanimoto coefficient, Euclidean
Structural Keys

- Structural descriptors are based on lookup library of known "functional" substructures.
- Pre-compute presence of relevant substructures up front and encode them in bit-vector.
- Example of structural keys:
  - Presence of atoms (C, N, O, S, Cl, Br, etc.)
  - Ring systems
  - Aromatic, Phenol, Alcohol, Amine, Acid, Ester, ...
- Disadvantages:
  - Lookup library tends to be incomplete.
  - Sparsely populated vectors.
Fingerprints

- Fingerprints are generated directly from the molecule itself and not from a reference set of substructures.
- The algorithm examines each molecule and generates the following patterns:
  - One for each atom.
  - One representing each atom and its nearest neighbors (plus the bonds that join them).
  - One representing each group of atoms and bonds connected by paths up to 2, 3, 4, ... bonds long.
  - For example, the molecule OC=CN would generate the following patterns:
    - 0-bond paths: C, O, N
    - 1-bond paths: OC, C=C, CN
    - 2-bond paths: OC=C, C=CN
    - 3-bond paths: OC=CN,
Fingerprints

- Patterns are often encoded into fixed length (binary) vectors for fast similarity searching.
- Abstract, hard to traceback meaning of individual bits.

Database of binary fingerprints
Atom Pair and Atom Sequence Similarity Searching

- Like fingerprints, atom pairs are generated directly from the molecule itself and not from a reference set of substructures (Chen and Reynolds, 2002).

- Atom pairs are defined by:
  - the length of the shortest bond path between two atoms,
  - while the terminal atoms in this path are described by:
    - their element type
    - their number of pi electrons
    - their number of non-hydrogen neighbors

  - Example: C12N03_06

- Atom sequences:
  - similar to atom pairs, but all atoms in the bond path are described.
  - Example: C12C13C13C02C02N03

- Conversion of atom pairs/sequences to binary vectors of constant length is usually not performed, but would be possible.
Similarity Coefficients

1. Euclidean

\[ \sqrt{\frac{c + d}{a + b + c + d}} \]  

(1)

2. Tanimoto coefficient

\[ \frac{c}{a + b + c} \]  

(2)

3. Simpson coefficient

\[ \frac{c}{\min((a + c), (b + c))} \]  

(3)

4. Tversky index

\[ \frac{c}{\alpha \ast a + \beta \ast b + c} \]  

(4)

5. Many more similarity coefficients (Holliday et al., 2003)

Legend for variables:
- a: count of features in CMP A but not in CMP B
- b: count of features in CMP B but not in CMP A
- c: count of features in both CMP A and CMP B
- d: count of features absent in CMP A and CMP B
- \( \alpha \) and \( \beta \): weighting variables
MCS-based Similarity Concepts

- Graph-based algorithms that find maximum common substructure (MCS) shared among two molecules
- Flexible MCS matching algorithm implemented in \textit{fmcsR} allows bond and atom mismatches.
- Major advantage: identification of local similarities

![Chemical structures with MCS highlighted]
Conformer Predictions

Prediction of the most stable conformers in 3D space.

3D Searches

Uses shape and topological indices to query a 3D conformer database.

3D Substructure searches

Related to pharmacophore searches

Docking

Computational modeling of the possible binding modes of a ligand to a target site.
Important Compound Databases

- PubChem
- DrugBank
- ChemBank
- ChEMBL
- Many more
Outline

Cheminformatics Basics
  Structure Formats
  Similarity Searching
  Physicochemical Properties
  Clustering

Hands-on Section
  Compound Import/Export
  Object Classes
  Compound Structure Depictions
  Compound Properties
  Compound Similarity Searching
  Compound Clustering
Compound Descriptors

Structural descriptors
- Atom pairs, fingerprints
- many others

Property descriptors
- Formula
- Molecular weight
- Octanol/Water partition coefficient (logP)
- Hydrogen Bond Acceptors
- Hydrogen Bond Donors
- Acidic groups
- Rotatable bonds
- over 300-3000 additional ones
Drug-likeliness Filters

Lipinski Rules
In a selection of 2245 compounds from the World Drug Index Lipinski identified four property cutoffs that were common in 90% of these drugs (Lipinski et al, 1997, Adv Drug Deliv Rev: 23, 3-25). These property filters are known as the "Rule of Five" (all multiple of 5):
- MW < 500g/mol
- lipophilicity: logP < 5
- n H-bond donors < 5 (e.g. OH and NH)
- n H-bond acceptors < 10 (e.g. N and O)

Extended Lipinski Rules
- n rotatable bonds < 10

ADMET Rules
- Criteria for predicting adsorption, distribution, metabolism, excretion and toxicity (ADMET) more important for pharmaceutical industry than chemical genomics.
Outline

Cheminformatics Basics

Structure Formats
Similarity Searching
Physicochemical Properties
Clustering

Hands-on Section

Compound Import/Export
Object Classes
Compound Structure Depictions
Compound Properties
Compound Similarity Searching
Compound Clustering
Clustering Methods

- **Principal component analysis (PCA)**
  - Reduction technique of multivariate data to principal components to identify hidden variances

- **Multidimensional scaling**
  - Displays distance matrix of objects in spacial plot

- **Hierarchical Clustering**
  - Iterative joining of items by decreasing similarity

- **Jarvis-Patrick Clustering**
  - Joins items based on intersects among nearest neighbor vectors

- **Binning Clustering**
  - Uses similarity cutoff for grouping of items

- Many additional clustering algorithms are being used in this field.
Example: PCA of Small Molecule Properties

- **Tannic Acid** (MW 1600)
- **Colistimethate** (MW 1400)
- **Cobalamine** (MW 1100)

![Butterfly](image)

Comb1 Comb2 Comb3 Comb4 Bioact

- **56421** (MW 270)
- **19737** (MW 390)
Cheminformatics in R

Why cheminformatics in R?

- Open source
- Efficient data structures and graphics utilities
- Access to many clustering and machine learning algorithms
- Integration with bioscience packages

R packages for cheminformatics

- Bioconductor
  - ChemmineR (Cao et al., 2008)
  - eiR
  - fmcsR
  - ChemmineOB: R interface to subcomponents of OpenBabel (Link)
  - ChemMine Tools: web interface to Chemmine utilities (Backman et al., 2011)

- CRAN
  - rcdk
  - rpubchem
Outline

Cheminformatics Basics
  Structure Formats
  Similarity Searching
  Physicochemical Properties
  Clustering

Hands-on Section
  Compound Import/Export
  Object Classes
  Compound Structure Depictions
  Compound Properties
  Compound Similarity Searching
  Compound Clustering
Import of SD Files

Download R code for exercises and open in RStudio session

Loading the ChemmineR package and its documentation
> library("ChemmineR") # Loads the package

Accessing ChemmineR PDF manual and help documents
> library(help="ChemmineR") # Lists all functions and classes
> vignette("ChemmineR") # Opens PDF manual from R
> ?MW # Opens help for MW function

Import SD file into SDFset object
> sdfset <- read.SDFset("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/Samples/sdfsample.sdf")
> sdfset # Returns summary of SDFset
> valid <- validSDF(sdfset) # Identifies invalid SDFs in SDFset objects
> sdfset <- sdfset[valid] # Removes invalid SDFs, if there are any

Load sample SD file provided by package
> data(sdfs各样le)
> sdfset <- sdfs各样le
> sdfset # Returns summary of SDFset

An instance of "SDFset" with 100 molecules
Write first 4 molecules to SD file

```r
> write.SDF(sdfset[1:4], file="sub.sdf", sig=TRUE)
> list.files(pattern="sub.sdf")
```

```
[1] "sub.sdf"
```

Write all molecules to several files each containing 50 entries

```r
> write.SDFsplit(x=sdfset, filetag="myfile", nmol=50)
```

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>myfile001_050.sdf</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>myfile051_100.sdf</td>
</tr>
</tbody>
</table>

Reimports newly created SD file

```r
> sdfsetsub <- read.SDFset("sub.sdf")
> sdfsetsub
```

An instance of "SDFset" with 4 molecules
Exercise I: Import/Export

Task 1  Open the PubChem site and search for 'p450 inhibitor'. Download the resulting 15 query hits to an SD file named 'p450.sdf' using the 'Structure Download' option on the right.

Task 2  Import the 'p450.sdf' into your R session. Here is a backup of this file in case there are difficulties with the PubChem site.

Task 3  Check in R the number of compounds stored in 'p450.sdf'.

Task 4  Write the structures in inversed order back to an SD file.

Task 5  Write the structures to several SD files each containing 5 molecules.
Outline

Cheminformatics Basics
  Structure Formats
  Similarity Searching
  Physicochemical Properties
  Clustering

Hands-on Section
  Compound Import/Export
  Object Classes
  Compound Structure Depictions
  Compound Properties
  Compound Similarity Searching
  Compound Clustering
Most Important S4 Objects in ChemmineR

Molecular structure containers

- **SDFstr**: intermediate string class to facilitate SD file import; not important for end user
- **SDF**: container for single molecule imported from an SD file
- **SDFset**: container for many SDF objects; most important container for end user

Structure descriptor containers

- **AP**: container for atom pair (AP) descriptors of a single molecule
- **APset**: container for many AP objects
- **FP**: container for fingerprint of a single molecule
- **FPset**: container for fingerprints of many molecules

Important methods operating on SDFset-type containers

- Object slots: cid, header, atomblock, bondblock, datablock
- Structure depiction: plot

Coerce one class to another

- Standard syntax as(..., "...") works in most cases. For details see R help with ?"SDFset-class". 
Several methods are available to return the different data components of SDF/SDFset containers in batches. The following examples list the most important ones.

`view(sdfset[1:4])` # Summary view of several molecules
`length(sdfset)` # Returns number of molecules
`sdfset[[1]]` # Returns single molecule from SDFset as SDF object
`sdfset[[1]][[2]]` # Returns atom block from first compound as matrix
`sdfset[[1]][[2]][1:4,]`
`c(sdfset[1:4], sdfset[5:8])` # Concatenation of several SDFsets

The grepSDFset function allows string matching/searching on the different data components of an SDFset. By default the function returns a SDF summary of the matching entries. Alternatively, an index of the matches can be returned with the setting `mode="index"`.

`grepSDFset("650001", sdfset, field="datablock", mode="subset")`
`# To return index, set mode="index"`
Accessing *SDF*/SDFset Components

Methods for retrieving header, atom, bond and data blocks

```r
> atomblock(sdf); sdf[[2]]; sdf[["atomblock"]]
> # All three methods return the same component
> header(sdfset[1:4])
> atomblock(sdfset[1:4])
> bondblock(sdfset[1:4])
> datablock(sdfset[1:4])
```

Utilities to manage compound IDs and to keep them unique

```r
> sdfid(sdfset[1:4])
[1] "650001" "650002" "650003" "650004"
> # Retrieves CMP IDs from Molecule Name field in header block.
> cid(sdfset[1:4])
[1] "CMP1" "CMP2" "CMP3" "CMP4"
> # Retrieves CMP IDs from ID slot in SDFset.
> unique_ids <- makeUnique(sdfid(sdfset))
[1] "No duplicates detected!"
> # Creates unique IDs by appending a counter to duplicates.
> cid(sdfset) <- unique_ids # Assigns uniquified IDs to ID slot
```
Exercise II: SDFset Containers

**Task 1** Assign custom names to ID slot in SDFset and export object to SD file so that the custom IDs are used as IDs in the header block.

**Task 2** Extract the bondblock of all structures in an SDFset, rbind them with do.call and write the resulting matrix to a tabular file.

**Task 3** Replace atomblock of first molecule in SDFset with atomblock of second molecule. Check the result with "==".
Outline

Cheminformatics Basics
  Structure Formats
  Similarity Searching
  Physicochemical Properties
  Clustering

Hands-on Section
  Compound Import/Export
  Object Classes
  **Compound Structure Depictions**
  Compound Properties
  Compound Similarity Searching
  Compound Clustering
Rendering Chemical Structure Images

Plot compound Structures with R's graphics device

```r
> data(sdfsample); sdfset <- sdfsample
> plot(sdfset[1:4], print=FALSE) # print=TRUE returns SDF summaries
```

CMP1

CMP2

CMP3

CMP4
Customize Structure Rendering

Show atom block position numbers next to the atom symbols. For more details, consult help documentation with \texttt{?plotStruc} or \texttt{?plot}.

\begin{verbatim}
> plot(sdfset["CMP1"], atomnum = TRUE, noHbonds=F , no_print_atoms = ",
+ atomcex=0.8, sub=paste("MW: ", MW(sdfsamp["CMP1"])), print=FALSE)
\end{verbatim}

MW: 456.49162
Substructure coloring

Substructure highlighting by atom numbers

```r
> plot(sdfset[1], print=FALSE, colbonds=c(22, 26, 25, 3, 28, 27, 2, 23, 21, 18, 8, 19, 20, 24))
```

CMP1
Online Structure Viewing with ChemMine Tools

Plot structures using web service ChemMine Tools:

```python
> sdf.visualize(sdfset[1:4])
```

**Figure:** Visualization page created by calling `sdf.visualize`. 
Exercise III: Rendering Compound Structures

Task 1  Plot the structures of compound IDs "42631481", "42631375", "42631371" and "42631260" of the p450 SDFset that you created in Exercise I.

Task 2  Render the same structures online with Chemmine Tools.
Outline

Cheminformatics Basics
- Structure Formats
- Similarity Searching
- Physicochemical Properties
- Clustering

Hands-on Section
- Compound Import/Export
- Object Classes
- Compound Structure Depictions

Compound Properties
- Compound Similarity Searching
- Compound Clustering
Several methods and functions are available to compute basic compound descriptors, such as molecular formula (MF), molecular weight (MW), and frequencies of atoms and functional groups. In many of these functions, it is important to set `addH=TRUE` in order to include/add hydrogens that are often not specified in an SD file.

```r
> propma <- atomcountMA(sdfset, addH=FALSE)
> propma[1:4,]
   C  H  N  O  S  F  Cl
CMP1 23 28  4  6  0  0  0
CMP2 18 23  5  3  0  0  0
CMP3 18 18  4  3  1  0  0
CMP4 21 27  5  5  1  0  0
```

Data frame provided by library containing atom names, atom symbols, standard atomic weights, group and period numbers.

```r
> data(atomprop)
> atomprop[1:4,]
Number Name Symbol Atomic_weight Group Period
1 1 hydrogen H 1.007940 1 1
2 2 helium He 4.002602 18 1
3 3 lithium Li 6.941000 1 2
4 4 beryllium Be 9.012182 2 2
```
Molecular Weight, Formula and Functional Groups

Compute MW and formula

```r
> MW(sdfset[1:4], addH=FALSE)

CMP1   CMP2   CMP3   CMP4
456.4916 357.4069 370.4255 461.5346
```

```r
> MF(sdfset[1:4], addH=FALSE)

CMP1   CMP2   CMP3   CMP4
"C23H28N4O6"  "C18H23N5O3"  "C18H18N4O3S"  "C21H27N5O5S"
```

Enumerate functional groups

```r
> groups(sdfset[1:4], groups="fctgroup", type="countMA")

R NH2  R2NH  R3N  ROP3  ROH  RCHO  RCO R COOH  R COOR  R OR  R CCH  R CN
 CMP1   0    2    1    0    0    0    0    0    0    2    0    0
 CMP2   0    2    2    0    1    0    0    0    0    0    0    0
 CMP3   0    1    1    0    1    0    1    0    0    0    0    0
 CMP4   0    1    3    0    0    0    0    0    0    2    0    0
```
Aggregate Many Molecular Properties

Combine MW, MF, charges, atom counts, functional group counts and ring counts in one data frame

```r
> propma <- data.frame(MF=MF(sdfset, addH=FALSE), MW=MW(sdfset, addH=FALSE),
+   Ncharges=sapply(bonds(sdfset, type="charge"), length),
+   atomcountMA(sdfset, addH=FALSE), groups(sdfset,
+   type="countMA"), rings(sdfset, upper=6, type="count",
+   arom=TRUE))
> propma[1:4,]

<table>
<thead>
<tr>
<th></th>
<th>MF</th>
<th>MW</th>
<th>Ncharges</th>
<th>C</th>
<th>H</th>
<th>N</th>
<th>O</th>
<th>S</th>
<th>F</th>
<th>Cl</th>
<th>RNH2</th>
<th>R2NH</th>
<th>R3N</th>
<th>ROPO3</th>
<th>ROH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP1</td>
<td>C23H28N4O6</td>
<td>456.4916</td>
<td>0</td>
<td>23</td>
<td>28</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMP2</td>
<td>C18H23N5O3</td>
<td>357.4069</td>
<td>0</td>
<td>18</td>
<td>23</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CMP3</td>
<td>C18H18N4O3S</td>
<td>370.4255</td>
<td>0</td>
<td>18</td>
<td>18</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CMP4</td>
<td>C21H27N5O5S</td>
<td>461.5346</td>
<td>0</td>
<td>21</td>
<td>27</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>RCHO</th>
<th>RCOR</th>
<th>RCOOH</th>
<th>RCOOR</th>
<th>ROR</th>
<th>RCCH</th>
<th>RCN</th>
<th>RINGS</th>
<th>AROMATIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>CMP2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>CMP3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>CMP4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Assign Molecular Properties to SDF Data Block

The following shows an example for assigning the values stored in a matrix (e.g. property descriptors) to the data block components in an *SDFset*. Each matrix row will be assigned to the corresponding slot position in the *SDFset*.

```r
> datablock(sdfset) <- propma # Works with all SDF components
> datablock(sdfset)[1]
```

$CMP1

<table>
<thead>
<tr>
<th>MF</th>
<th>MW</th>
<th>Ncharges</th>
<th>C</th>
<th>H</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;C23H28N4O6&quot;</td>
<td>&quot;456.4916&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;23&quot;</td>
<td>&quot;28&quot;</td>
<td>&quot;4&quot;</td>
</tr>
<tr>
<td>0</td>
<td>S</td>
<td>F</td>
<td>Cl</td>
<td>RNH2</td>
<td>R2NH</td>
</tr>
<tr>
<td>&quot;6&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>R3N</td>
<td>ROPO3</td>
<td>ROH</td>
<td>RCHO</td>
<td>RCOR</td>
<td>RC0OH</td>
</tr>
<tr>
<td>&quot;1&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>RC00R</td>
<td>RR</td>
<td>RCCH</td>
<td>RCN</td>
<td>RINGS</td>
<td>AROMATIC</td>
</tr>
<tr>
<td>&quot;0&quot;</td>
<td>&quot;2&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;4&quot;</td>
<td>&quot;2&quot;</td>
</tr>
</tbody>
</table>

Convert data block of *SDFset* to matrix.

```r
> blockmatrix <- datablock2ma(datablocklist=datablock(sdfset))
> blockmatrix[1:2,1:12]
```

<table>
<thead>
<tr>
<th>MF</th>
<th>MW</th>
<th>Ncharges</th>
<th>C</th>
<th>H</th>
<th>N</th>
<th>O</th>
<th>S</th>
<th>F</th>
<th>Cl</th>
<th>RNH2</th>
<th>R2NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP1</td>
<td>&quot;C23H28N4O6&quot;</td>
<td>&quot;456.4916&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;23&quot;</td>
<td>&quot;28&quot;</td>
<td>&quot;4&quot;</td>
<td>&quot;6&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>CMP2</td>
<td>&quot;C18H23N5O3&quot;</td>
<td>&quot;357.4069&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;18&quot;</td>
<td>&quot;23&quot;</td>
<td>&quot;5&quot;</td>
<td>&quot;3&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot;</td>
</tr>
</tbody>
</table>
Charges and Missing Hydrogens

The function `bonds` returns information about the number of bonds, charges and missing hydrogens in `SDF` and `SDFset` objects. It is used by many other functions (e.g. MW, MF, atomcount, atomcuntMA and plot) to correct for missing hydrogens that are often not specified in SD files.

```r
> bonds(sdfset[[1]], type="bonds")[1:4,]

          atom Nbondcount Nbondrule charge
 1          O       2          2       2      0
 2          O       2          2       2      0
 3          O       2          2       2      0
 4          O       2          2       2      0

> bonds(sdfset[1:2], type="charge")

$CMP1
NULL

$CMP2
NULL

> bonds(sdfset[1:2], type="addNH")

CMP1  CMP2
 0      0
```
Ring Perception and Aromaticity Assignment

The function `rings` identifies all possible rings in one or many molecules using the exhaustive ring perception algorithm from Hanser et al. (1996). In addition, the function can return all smallest possible rings as well as aromaticity information.

```r
> (ringatoms <- rings(sdfset[1], upper=Inf, type="all", arom=TRUE, inner=FALSE))

$RINGS
$RINGS$ring1
[1] "N_10" "O_6" "C_32" "C_31" "C_30"

$RINGS$ring2
[1] "C_12" "C_14" "C_15" "C_13" "C_11"

$RINGS$ring3
[1] "C_23" "O_2" "C_27" "C_28" "O_3" "C_25"

$RINGS$ring4

$RINGS$ring5
[1] "O_3" "C_28" "C_27" "O_2" "C_23" "C_21" "C_18" "C_22" "C_26" "C_25"

$AROMATIC
ring1 ring2 ring3 ring4 ring5
TRUE FALSE FALSE TRUE FALSE
Highlight Rings in Structure Image

For visual inspection, the corresponding compound structure can be plotted with the ring bonds highlighted in color.

```r
> atomindex <- as.numeric(gsub(".*_", ",", unique(unlist(ringatoms))))
> plot(sdfset[1], print=FALSE, colbonds=atomindex)
```

CMP1

![Chemical Structure](image)
The `sdfStream` function allows to stream through SD Files with millions of molecules without consuming much memory. During this process any set of descriptors, supported by ChemmineR, can be computed. In addition to descriptor values, the function returns a line index that gives the start and end positions of each molecule in the source SD File. This line index can be used by the downstream `read.SDFindex` function to retrieve specific molecules of interest from the source SD File without reading the entire file into R.

```r
> write.SDF(sdfset, "test.sdf")
> desc <- function(sdfset) {
+   cbind(SDFID=sdfid(sdfset),
+         MW=MW(sdfset),
+         groups(sdfset),
+         rings(sdfset, type="count", upper=6, arom=TRUE)
+   )
+ }
> sdfStream(input="test.sdf", output="matrix.xls", fct=desc, Nlines=1000, silent=TRUE)
> read.delim("matrix.xls", row.names=1)[1:3,1:10]
```

<table>
<thead>
<tr>
<th>SDFLineStart</th>
<th>SDFLineEnd</th>
<th>SDFID</th>
<th>MW</th>
<th>RNH2</th>
<th>R2NH</th>
<th>R3N</th>
<th>ROPO3</th>
<th>ROH</th>
<th>RCHO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP1</td>
<td>1</td>
<td>203</td>
<td>650001</td>
<td>456.4916</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMP2</td>
<td>204</td>
<td>381</td>
<td>650002</td>
<td>357.4069</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CMP3</td>
<td>382</td>
<td>550</td>
<td>650003</td>
<td>370.4255</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise IV: Compound Properties

Task 1  Compute for p450 SDFset from Exercise I all possible compound properties.
Task 2  Assign the property matrix to the data block in the corresponding SDFset.
Task 3  Export the modified SDFset object to an SD file and inspect the result.
Outline

Cheminformatics Basics
- Structure Formats
- Similarity Searching
- Physicochemical Properties
- Clustering

Hands-on Section
- Compound Import/Export
- Object Classes
- Compound Structure Depictions
- Compound Properties
- Compound Similarity Searching
- Compound Clustering
Structure Descriptor Containers: APset/FPset

The function `sdf2ap` computes atom pair descriptors for one or many compounds (Chen and Reynolds, 2002; Cao et al., 2008). It returns a searchable atom pair database stored in a container of class `APset`, which can be used for structural similarity searching and clustering.

```r
> apset <- sdf2ap(sdfset)
> apset

An instance of "APset" with 100 molecules

Most methods working on `SDFset` objects work the same way on descriptor objects.

```r
> showClass("APset")
> cid(apset)
> view(apset)
> as(apset, "list")
```

The `FPset` class stores fingerprints of small molecules in a matrix-like representation where every molecule is encoded as a fingerprint of the same type and length.

```r
> (fpset <- desc2fp(apset))

An instance of a 1024 bit "FPset" with 100 molecules

```r
> view(fpset[1])

$CMP1
An instance of "FP"
<<fingerprint>>
0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 ... length: 1024
```
The `cmp.search` function searches an atom pair database for compounds that are similar to a query compound.

```r
> cmp.search(apset, apset["CMP1"], type=3, cutoff = 0.3, quiet=TRUE)

  index  cid   scores
   1   1  CMP1  1.0000000
   2  96  CMP96 0.3516643
   3  67  CMP67 0.3117569
   4  88  CMP88 0.3094629
   5  15  CMP15 0.3010753
```

**Compound similarity searching with FPset**

```r
> fpset1024 <- names(rev(sort(table(unlist(as(apset, "list")))))[1:1024])
> fpset <- desc2fp(apset, descnames=fpset1024, type="FPset")
> fpSim(fpset["CMP1"], fpset, method="Tanimoto", cutoff=0.2, top=6)

   CMP1    CMP96    CMP67    CMP31    CMP88    CMP15
0.0000000 0.4300000 0.3859060 0.3855856 0.3804714 0.3738602
```
The \texttt{fpSim} function can be used for any type of binary fingerprint, here PubChem fingerprints extracted from the data block of the \texttt{SDFset} object.

\begin{verbatim}
> fpset <- fp2bit(sdfsamp, type=3)
> fpSim(fpset[1], fpset, method="Tanimoto", cutoff=0.2, top=6)

   CMP1   CMP33   CMP98   CMP86   CMP4   CMP70
0.0000000 0.6950000 0.6763485 0.6666667 0.6448980 0.6422018

Pairwise comparisons are supported as well
> fpSim(fpset[1], fpset[2])

   CMP2
0.5364807
\end{verbatim}
Maximum Common Substructure (MCS) Searching

The \textit{fmcsR} package provides support for identifying strict MCSs and mismatch tolerant flexible FMCSs among compounds.

\begin{verbatim}
> library(fmcsR); data(fmcstest) # Loads library and test sdfset object
> test <- fmcs(fmcstest[1], fmcstest[2], au=2, bu=1) # Searches for MCS with mismatches
> plotMCS(test) # Plots both query compounds with MCS in color
\end{verbatim}

FMCS-based structure similarity searching

\begin{verbatim}
> fmcsBatch(sdfset[[1]], sdfset)[1:2,]
\end{verbatim}

\begin{tabular}{cccccc}
  Query_Size & Target_Size & MCS_Size & Tanimoto_Coefficient & Overlap_Coefficient \\
  CMP1 & 33 & 33 & 33 & 1.0000000 & 1.0000000 \\
  CMP2 & 33 & 26 & 11 & 0.2291667 & 0.4230769 \\
\end{tabular}
Searching PubChem from ChemmineR

*ChemmineR* supports searching of the PubChem database by compound IDs or via a structure similarity search using PubChem fingerprints. The following searches PubChem by structure similarity and stores the results in an SDFset object.

```r
> compounds <- searchSim(sdfset[1])
> compounds

An instance of "SDFset" with 10 molecules
```
**Exercise V: Compound Similarity Searching**

**Task 1** Convert the p450 SDFset from Exercise I into 3 searchable descriptor databases containing: (1) atom pairs, (2) atom pair fingerprints and (3) PubChem fingerprints.

**Task 2** Perform a structure similarity search against all three databases with the first compound in p450 SDFset as query. Compare the ranking of the three different search results.
Outline

Cheminformatics Basics
- Structure Formats
- Similarity Searching
- Physicochemical Properties
- Clustering

Hands-on Section
- Compound Import/Export
- Object Classes
- Compound Structure Depictions
- Compound Properties
- Compound Similarity Searching

Compound Clustering
Binning Clustering

Compound libraries can be clustered into discrete similarity groups with the binning clustering function `cmp.cluster`.

```
c1 <- cmp.cluster(fpset, cutoff=c(0.3, 0.6, 0.9), method="Tanimoto", +quiet=TRUE)[1:8,]
```

```
sorting result...

> c1[1:8,]

ids  CLSZ_0.3  CLID_0.3  CLSZ_0.6  CLID_0.6  CLSZ_0.9  CLID_0.9
1 CMP1    100        1      91        1        1        1
2 CMP2    100        1      91        1        1        2
3 CMP3    100        1      91        1        1        3
4 CMP4    100        1      91        1        1        4
6 CMP6    100        1      91        1        1        6
7 CMP7    100        1      91        1        1        7
8 CMP8    100        1      91        1        1        8
9 CMP9    100        1      91        1        1        9
```

```
> cluster.sizestat(c1, cluster.result=2)

  cluster size  count
1          91 0.08791209
```
The Jarvis-Patrick clustering algorithm is widely used in cheminformatics (Jarvis and Patrick, 1973) because it scales to very large numbers of compounds. The following performs standard Jarvis-Patrick clustering and computes the nearest neighbor table on the fly.

```r
> jarvisPatrick(nearestNeighbors(fpset, numNbrs=6), k=5, mode="a1a2b")
```

```
CMP1  CMP2  CMP3  CMP4  CMP5  CMP6  CMP7  CMP8  CMP9  CMP10  CMP11  CMP12  CMP13
  1    2    3    4    5    6    7    8    9    10     11     12     13
CMP14  CMP15  CMP16  CMP17  CMP18  CMP19  CMP20
  14   15    16    17    18    19    20
```

Output nearest neighbor table (matrix)

```r
> nnm <- nearestNeighbors(fpset, numNbrs=6)
> nnm$similarities[1:4,]
```

```
CMP1  CMP33  CMP98  CMP86  CMP4  CMP70
sim  1  0.6950000  0.6763485  0.6666667  0.6448980  0.6422018
sim  1  0.7823834  0.7475248  0.7348837  0.7281553  0.6666667
sim  1  0.6871795  0.6446701  0.6283186  0.6276596  0.6263158
sim  1  0.8177570  0.7348837  0.7287449  0.7136564  0.7105263
```
Multi-Dimensional Scaling (MDS)

Multidimensional scaling (MDS) algorithms start with a matrix of item-item distances and then assign coordinates for each item in a low-dimensional space to represent the distances graphically.

```r
> simMA <- sapply(cid(fpset), function(x) fpSim(fpset[x], fpset, sorted=FALSE))
> plot(cmdscale(as.dist(1-simMA)))
```

![Plot of cmdscale](attachment:image.png)
Hierarchical Clustering

The following performs hierarchical clustering of compound structure similarities (distances). The resulting dendrogram is then plotted next to a heatmap of the corresponding similarity matrix.

> library(gplots)
> hc <- hclust(as.dist(1-simMA), method="single")
> heatmap.2(1-simMA, Rowv=as.dendrogram(hc), Colv=as.dendrogram(hc),
+ col=colorpanel(40, "darkblue", "yellow", "white"),
+ density.info="none", trace="none")
Exercise VI: Compound Clustering

Task 1  Cluster the structures in p450 SDFset with the binning clustering algorithm.
Task 2  Cluster the structures in p450 SDFset with the Jarvis-Patrick clustering algorithm.
Task 3  Cluster the structures in p450 SDFset with the MDS algorithm.
Task 4  Cluster the structures in p450 SDFset with the hierarchical clustering algorithm.
> sessionInfo()

R version 3.0.0 (2013-04-03)
Platform: x86_64-pc-linux-gnu (64-bit)

locale:
  [1] LC_CTYPE=en_US.UTF-8   LC_NUMERIC=C
  [3] LC_TIME=en_US.UTF-8    LC_COLLATE=en_US.UTF-8
  [5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
  [7] LC_PAPER=C             LC_NAME=C
  [9] LC_ADDRESS=C           LC_TELEPHONE=C

attached base packages:
[1] grid       stats       graphics   grDevices  utils     datasets   methods
[8] base

other attached packages:
[1] gplots_2.11.0.1   MASS_7.3-26   KernSmooth_2.23-10 caTools_1.14
[5] gdata_2.12.0.2    gtools_2.7.1   fmcsR_1.2.0    ChemmineR_2.12.1

loaded via a namespace (and not attached):
[1] bitops_1.0-5    DBI_0.2-5    digest_0.6.3    RCurl_1.95-4.1 tools_3.0.0
Bibliography I

URL http://www.hubmed.org/display.cgi?uids=21576229

URL http://www.hubmed.org/display.cgi?uids=18596077

URL http://www.hubmed.org/display.cgi?uids=12444738

URL http://pubs.acs.org/doi/abs/10.1021/ci960322f

URL http://www.hubmed.org/display.cgi?uids=12767139