Software Manual

Institute of Bioinformatics, Johannes Kepler University Linz

Rchemcpp – Similarity measures for chemical compounds

Michael Mahr and Günter Klambauer

Institute of Bioinformatics, Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
klambauer@bioinf.jku.at

Version 2.16.0, October 30, 2017

Contents

1 Introduction

The Rchemcpp package (Klambauer *et al.*, 2015) implements functions that compute similarities between small drug-like molecules. Similarity measures for compounds are highly relevant since they offer the possibility to find chemical clusters or to build models that predict the biological activities and responses of a new drug. Further a database can be screened for structural analogs to given lead compound. This is also crucial in drug-design. This package extends the functionality of the ChemmineR Bioconductor package using the functions and data structures that are introduced there, such as the reading of molecule structures from SDF files and handling them as S4 objects in R.

The following similarity functions/kernels are implemented:

- the marginalized graph kernel between labeled graphs (Kashima *et al.*, 2004).
- extensions of the marginalized kernel (Mahé *et al.*, 2004).
- Tanimoto kernels (Ralaivola *et al.*, 2005).
- graph kernels based on tree patterns (Mahé and Vert, 2009).
- kernels based on pharmacophores for 3D structure of molecules (Mahé *et al.*, 2006).

See http://www.bioinf.jku.at/software/Rchemcpp/ for additional information. A webservice for finding structural analogs using Rchemcpp is available at http://www.bioinf.jku.at/services/analoging/.

2 Getting started and quick start

To load the package, enter the following in your R session:

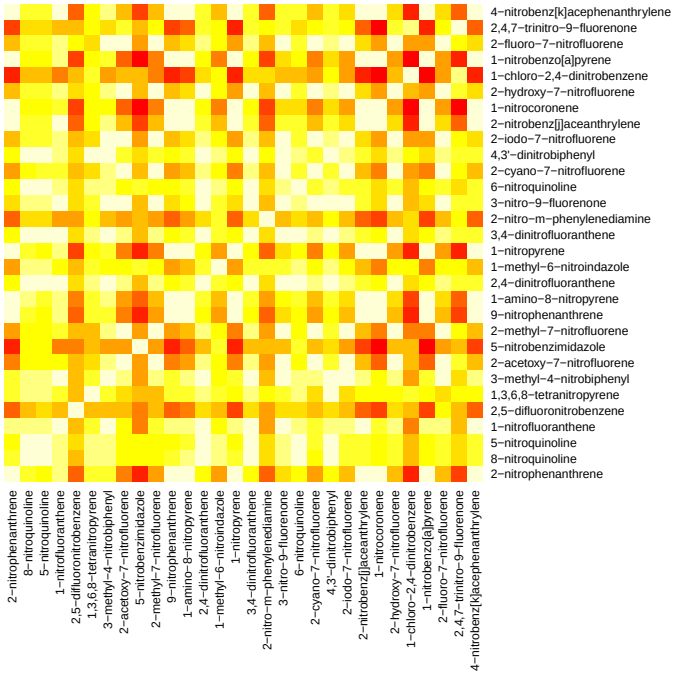
```
> library(Rchemcpp)
```

We enter the filename of and SDF file to the function sd2gram. This function computes the similarity of the molecules with the marginalized kernel (Kashima *et al.*, 2004) approach.

```
> sdfolder <- system.file("extdata",package="Rchemcpp")
> sdf <- list.files(sdfolder,full.names=TRUE,pattern="small")
> K <- sd2gram(sdf)
```

The similarity values are now stored in K. We visualize this matrix as a heatmap.

```
> heatmap(K,Rowv=NA,Colv=NA,scale="none")
```



3 Molecular similarity for clustering

Based on the similarity measure we can run clustering algorithms on the data in order to find groups among the molecules. We use Affinity Propagation Clustering (Frey and Dueck, 2007) as implemented by Bodenhofer *et al.* (2011) for this task, because the cluster centers are real molecules.

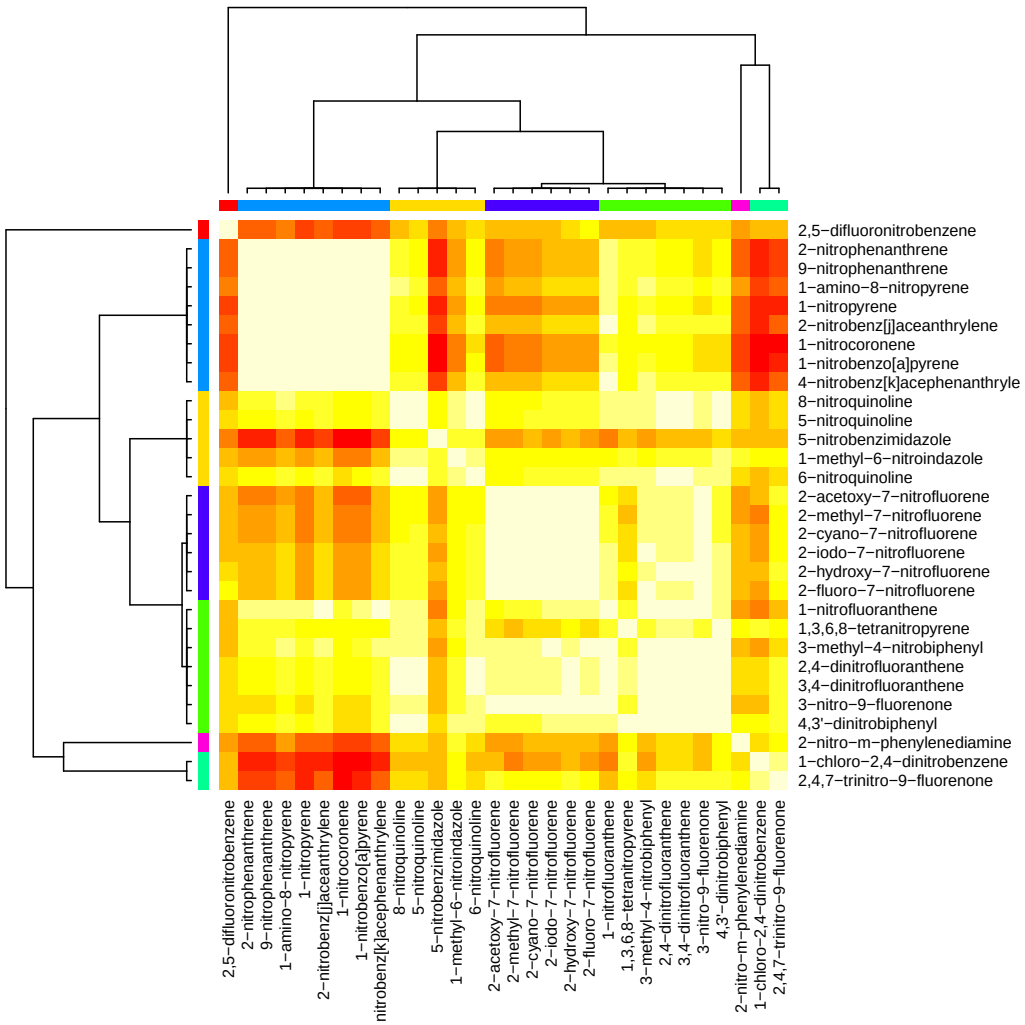
```
> library(apcluster)
> r <- apcluster(K)

> heatmap(r,K)
```

4 Molecular similarity for prediction of chromosome aberration

The similarity measures can be used for building models to predict features of the molecules, for example with the help of Support Vector Machines (SVMs). Mohr *et al.* (2010) used this approach to predict the Ames test (Mortelmans and Zeiger, 2000), that tests whether a compound leads to chromosome aberration. The authors put together a data set of 940 compounds with known results of the Ames test. This data set is included in this package.

```
> library(kernlab)
> chromosomeAberrationDataSet <- list.files(sdfolder,
+                   full.names=TRUE,pattern="Mohr.sdf")
```


2,5−difluoronitrobenzene
2−nitrophenanthrene
9−nitrophenanthrene
1−amino−8−nitropyrene
1−nitropyrene
2−nitrobenz[j]aceanthrylene
1−nitrocoronene
1−nitrobenzo[a]pyrene
4−nitrobenz[k]acephenanthryle
8−nitroquinoline
5−nitroquinoline
5−nitrobenzimidazole
1−methyl−6−nitroindazole
6−nitroquinoline
2−acetoxy−7−nitrofluorene
2−methyl−7−nitrofluorene
2−cyano−7−nitrofluorene
2−iodo−7−nitrofluorene
2−hydroxy−7−nitrofluorene
2−fluoro−7−nitrofluorene
1−nitrofluoranthene
1,3,6,8−tetranitropyrene
3−methyl−4−nitrobiphenyl
2,4−dinitrofluoranthene
3,4−dinitrofluoranthene
3−nitro−9−fluorenone
4,3'−dinitrobiphenyl
2−nitro−m−phenylenediamine
1−chloro−2,4−dinitrobenzene
2,4,7−trinitro−9−fluorenone

```
> KCA <- sd2gramSpectrum(chromosomeAberrationDataSet,
+                  detectArom=FALSE,depthMax=4,silentMode=TRUE)
> response <- getMoleculePropertyFromSDF(chromosomeAberrationDataSet,
+                  "chromosome_damage")
> # C was set to 0.1 for computational speed - should be set to a higher value
> model <- ksvm(KCA,y=as.factor(response),kernel="matrix",cross=10,type="C-svc",C=0.1)
```

An estimated accuracy of the prediction is

```
> print(1-model@cross)
```

```
[1] 0.6287234
```

With this model we can predict whether a certain molecule will be positive in the Ames test

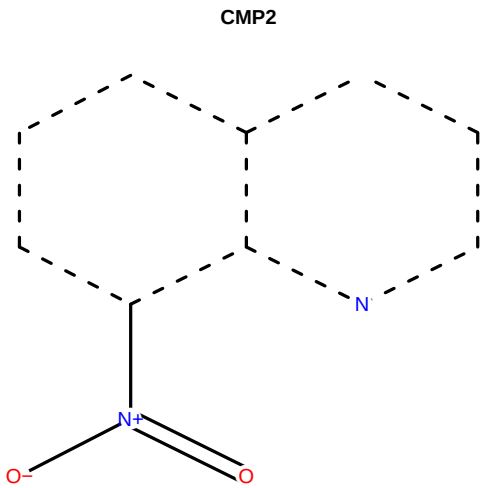
```
> predict(model,as.kernelMatrix(KCA[3,SVindex(model),drop=FALSE]))
```

```
[1] false
Levels: false true
```

5 Working with SDFset objects from the package ChemmineR

In the ChemmineR package SDFset objects are defined, that make the handling of molecules in R easy and provide a lot of utility functions, like plotting.

```
> library(ChemmineR)
> sdfFileName <- list.files(sdfolder,full.names=TRUE,pattern="small")
> sdfSet <- read.SDFset(sdfFileName)
> plot(sdfSet[2],print=FALSE)
```



It is possible to run the molecule kernels directly on `SDFset` objects.

```
> K1 <- sd2gramSubtree(sdfSet,silentMode=TRUE)
```

6 Compound Screening: Comparing one compound against a data set

Comparing the objects in one set (lead compound) to the objects in another set (database of compounds) is possible.

```
> leadCompound <- sdfSet[1]
> compoundDataBase <- sdfSet[1:20]
> K2 <- sd2gramSubtree(leadCompound,compoundDataBase,silentMode=TRUE)
```

7 Graph kernels

These kernels are based on sets of molecular fragments. Molecular fragments can be either *walks*, i.e. a sequence of atoms connected by bonds, or *subtrees*, i.e. directed tree-patterns. A molecule is represented as graph. If we consider two molecules as graphs X and Y, then the kernel K is:

$$K(X,Y) = \sum_{p \in \mathcal{P}} N(p,X) \cdot N(p,Y), \tag{1}$$

where $\mathcal{P}$ is the set of molecular fragments (all possible walks or subtrees, and the function $N(p,X)$ usually (e.g. for the spectrum kernel 7.2) counts, how often the pattern p occurs in molecule graph X. The function $N(p,X)$ will be defined for the different kernel types in the following.

Kernel similarity measures can be normalized to values between 0 and 1. Usually the following kernel normalization is used

$$K_{\text{norm}}(X,Y) = \frac{K(X,Y)}{\sqrt{K(X,X) \cdot K(Y,Y)}}. \tag{2}$$

7.1 Implementation

Graph kernels are efficiently implemented in `Chemcpp`(Mahé *et al.*, 2007) or its `R` interface `Rchemcpp` (Klambauer *et al.*, 2015).

- **sd2gram** Implements the marginalized kernel 7.5.
- **sd2gramSpectrum** Implements the spectrum kernel 7.2, the Tanimoto kernel 7.3, the MinMax kernel 7.4, the lambda-k kernel 7.7 and the approximation of the marginalized kernel 7.6.

- **sd2gramSubtree** Implements the subtree kernel 7.9.

For all these kernels the set of molecular fragments are non-tottering walks of or up to a certain length n given by the `depthMax` parameter. With the option `onlyDepthMax=FALSE` all molecular fragments up to a length of n atoms/vertices are in the set $\mathcal{P}$, if `onlyDepthMax=TRUE` only molecular fragments with n atoms/vertices are in the set $\mathcal{P}$. For all kernels there is a parameter `returnNormalized` that scales the similarity measures to values between 0 and 1.

7.2 The Spectrum Kernel

Here the function $N(p, X)$ counts how often the walk p occurs in the graph X. The kernel is

$$K(X, Y) = \sum_{p \in \mathcal{P}} N(p, X) \cdot N(p, Y), \tag{3}$$

with

$$N(p, X) = \#\{p \in X\}. \tag{4}$$

For simplicity we denote a function $\#$ that the number of occurences of walk p in graph X. The normalized version of this kernel is:

$$K_{\text{norm}}(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X) \cdot K(Y, Y)}} \tag{5}$$

7.3 The Tanimoto Kernel

Here the function $N(p, X)$ indicates whether walk p occurs in the graph X. The kernel is

$$K(X, Y) = \sum_{p \in \mathcal{P}} N(p, X) \cdot N(p, Y), \tag{6}$$

with

$$N(p, X) = \mathbf{1}\{p \in X\}. \tag{7}$$

For simplicity we denote an indicator function $\mathbf{1}$ that is one if the walk p occurs in the graph X and is zero otherwise. What is usually considered as the Tanimoto kernel (Ralaivola *et al.*, 2005) is the normalized version:

$$K_{\text{Tanimoto}}(X, Y) = \frac{K(X, Y)}{K(X, X) + K(Y, Y) - K(X, Y)}. \tag{8}$$

7.4 The MinMax Kernel

This kernel (Ralaivola *et al.*, 2005) is a variation of the kernel presented above. Here the function $N(p, X)$ counts how often the walk p occurs in the graph X.

$$K_{\max}(X, Y) = \sum_{p \in \mathcal{P}} \max(N(p, X), N(p, Y)), \tag{9}$$

$$K_{\min}(X, Y) = \sum_{p \in \mathcal{P}} \min(N(p, X), N(p, Y)) \tag{10}$$

$$\tag{11}$$

with

$$N(p, X) = \#\{p \in X\} \tag{12}$$

the number of occurences of walk p in the graph X. The MinMax kernel is the already a normalized version:

$$K_{\text{minmax}}(X, Y) = \frac{K_{\min}(X, Y)}{K_{\max}(X, Y)}. \tag{13}$$

7.5 The Marginalized Kernel

The marginalized kernel suggested by Kashima *et al.* (2003, 2004) . Here the function $N(p, X)$ counts how often the walk p occurs in the graph X and weights it by the probability that it occurs. The kernel is

$$K(X, Y) = \sum_{p \in \mathcal{P}} N(p, X) \cdot N(p, Y), \tag{14}$$

with

$$N(p, X) = \sum_{h \in \mathcal{H}(\mathcal{X})} w(h, X) \cdot \mathbf{1}(h = p), \tag{15}$$

where $\mathcal{H}(\mathcal{X})$ is the set of walks of graph X, and $w(h, X)$ is the probability that the walk h occurs in X. This probability is influenced by the stopping probability (parameter `stopP`). The indicator function $\mathbf{1}(h = p)$ is one if the atoms and bonds of the walk h match the given walk p. The sum in Eq. 14 is a sum over an infinite number of walks. However, the probability of a walk decreases exponentially with its length, therefore the kernel converges.

7.6 The Marginalized Kernel Approximation

This is an approximation of the marginalized kernel (Kashima *et al.*, 2003, 2004) presented above, but the set of walks here is finite, since the length of the walks is bounded by n. Here the function $N(p, X)$ counts how often the walk p occurs in the graph X and weights it by the probability that it occurs. The kernel is

$$K(X, Y) = \sum_{p \in \mathcal{P}} N(p, X) \cdot N(p, Y), \tag{16}$$

with

$$N(p, X) = \sum_{h \in \mathcal{H}(\mathcal{X})} w(h, X) \cdot \mathbf{1}(h = p), \tag{17}$$

where $\mathcal{H}(\mathcal{X})$ is the set of walks of graph X, and $w(h, X)$ is the probability that the walk h occurs in X. This probability is influenced by the stopping probability (parameter `stopP`). The indicator function $\mathbf{1}(h = p)$ is one if the atoms and bonds of the walk h match the given walk p.

Because the length of the walks is limited by the parameter n, this is only an approximation of the marginalized kernel. Note that because of the random walk process, the probabilities of the walkds exponentially decrease with their lengths. Long walks are barely taken into account in the marginalized kernel formulation. The normalized version of this kernel is:

$$K_{\text{norm}}(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X) \cdot K(Y, Y)}} \tag{18}$$

7.7 The Lambda-k Kernel

Here the function $N(p, X)$ counts how often the walk p occurs in the graph X and weights it by a function of the length of the walk. The kernel is

$$K(X, Y) = \sum_{p \in \mathcal{P}} N(p, X) \cdot N(p, Y), \tag{19}$$

with

$$N(p, X) = \lambda^{|p|} \cdot \#\{p \in X\}. \tag{20}$$

For simplicity we denote a function $\#$ that the number of occurences of walk p in graph X. For $\lambda = 1$ this kernel corresponds exactly to the spectrum kernel. For $\lambda > 1$ the influence of longer walks is higher, and for $\lambda < 1$ the similarity measure is more influenced by shorter walks. The normalized version of this kernel is:

$$K_{\text{norm}}(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X) \cdot K(Y, Y)}} \tag{21}$$

7.8 Efficiency of walk kernels

A walk of a graph $X = (V, E)$ with vertices V and edges E is a sequence $v_1, \ldots, v_n \in V$ such that $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, n-1$. This implementation mostly uses non-tottering walks, that is walks with $v_i \neq v_{i+2}$ or provides an option to remove tottering walks. We denote $\mathcal{H}_n(X)$ the set of walks of graph X with length n. The computation of these kernels is very efficient, since there is a bijection between the pairs of walks $p \in \mathcal{H}_n(X)$ and $q \in \mathcal{H}_n(Y)$ with the same label sequences and the walks on the product graph $r \in \mathcal{H}_n(X \times Y)$. Therefore we only have to count the walks with length n on the product graph of X and Y.

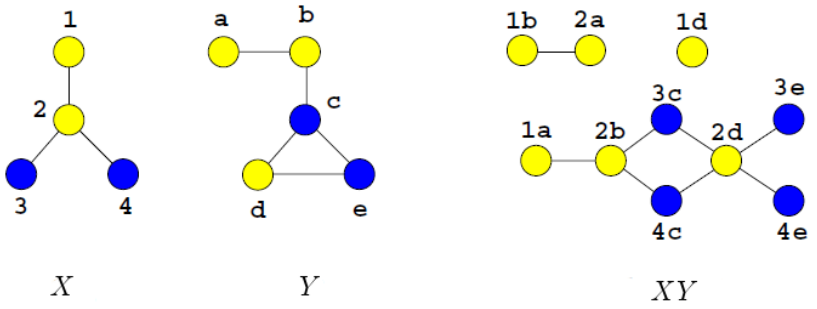


Figure 1: An example for a product graph. Figure taken from Vert (2007).

The product graph between two labeled graphs $X = (V_X, E_X)$ and $Y = (V_Y, E_Y)$ is defined as the graph $Z = (V_{X\times Y}, E_{X\times Y})$ with

$$V_{X\times Y} = \{(v_1, v_2) \in V_X \times X_Y : l(v_x) = l(v_y)\} \tag{22}$$
$$E_{X\times Y} = \{\big((v_1, v_2), (v_1', v_2')\big) \in V_X \times X_Y : (v_1, v_1') \in X \text{ and } (v_2, v_2') \in Y\}, \tag{23}$$

where $l(v)$ denotes the label of the vertex v. For a graphical explanation of the definition of the producte graph, see Fig. 7.8.

7.9 The Subtree Kernel

This graph kernel is based on the detection of common subtrees: the so-called tree-pattern graph kernels, originally introduced in (Ramon and Gärtner, 2003), and revisited in (Mahé *et al.*, 2006). For more details on the kernel definitions please refer to (Mahé *et al.*, 2006). The size of the trees can either be bounded by the size of the tree (number of atoms) or the number of branches. Both trees of a given length and up to a given length can be used for the calculation of the kernel.

8 The Pharmacophore Kernel

The pharmacophore kernel is based on the 3D structure of molecules (Mahé *et al.*, 2006). The kernels are based on the comparison of the three-point pharmacophores present in the 3D structures of molecules, a set of molecular features known to be particularly relevant for virtual screening applications. There is a computationally demanding exact implementation of these kernels, as well as fast approximations related to the classical fingerprint-based approaches.

- **sd2gram3Dpharma** Implements the exact pharmacophore kernel.
- **sd2gram3Dspectrum** Implements some approximations of the pharmacophore kernel.

9 How to cite this package

If you use this package for research that is published later, you are kindly asked to cite it as follows: (Klambauer *et al.*, 2015).

To obtain BibTEX entries of the reference, you can enter the following into your R session:

```
> toBibtex(citation("Rchemcpp"))
```

```
author = {Klambauer, Guenter and Wischenbart, Martin and Mahr, Michael and Unterthiner, Thomas and Mayr, Andreas and Hochreiter, Sepp},
title = {Rchemcpp: a web service for structural analoging in ChEMBL, Drugbank and the Connectivity Map},
year = {2015},
doi = {10.1093/bioinformatics/btv373},
URL = {http://bioinformatics.oxfordjournals.org/content/early/2015/06/17/bioinformatics.btv373.abstract},
eprint = {http://bioinformatics.oxfordjournals.org/content/early/2015/06/17/bioinformatics.btv373.full.pdf+html},
journal = {Bioinformatics}
}
```

References

Bodenhofer, U., Kothmeier, A., and Hochreiter, S. (2011). APCluster: an R package for affinity propagation clustering. *Bioinformatics*, **27**, 2463–2464.

Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, **315**, 972–977.

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press.

Kashima, H., Tsuda, K., and Inokuchi, A. (2004). Kernels for graphs. In *Kernel Methods in Computational Biology*, pages 155–170. MIT Press.

Klambauer, G., Wischenbart, M., Mahr, M., Unterthiner, T., Mayr, A., and Hochreiter, S. (2015). Rchemcpp: a web service for structural analoging in chembl, drugbank and the connectivity map. *Bioinformatics*.

Mahé, P. and Vert, J.-P. (2009). Graph kernels based on tree patterns for molecules. *Mach. Learn.*, **75**(1), 3–35.

Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., and Vert, J.-P. (2004). Extensions of marginalized graph kernels. In R. Greiner and D. Schuurmans, editors, *Proc of the 21st ICML*, pages 552–559. ACM Press.

Mahé, P., Ralaivola, L., Stoven, V., and Vert, J.-P. (2006). The pharmacophore kernel for virtual screening with support vector machines. *J Chem Inf Model*, **46**(5), 2003–2014.

Mahé, P., Perret, J.-L., Akutsu, T., and Vert, J.-P. (2007). Chemcpp: A C++ toolbox for chemoinformatics focusing on the computation of kernel functions between chemical compounds. *http://chemcpp.sourceforge.net/html/index.html*.

Mohr, J., Jain, B., Sutter, A., Laak, A. T., Steger-Hartmann, T., Heinrich, N., and Obermayer, K. (2010). A maximum common subgraph kernel method for predicting the chromosome aberration test. *J Chem Inf Model*, **50**(10), 1821–1838.

Mortelmans, K. and Zeiger, E. (2000). The ames salmonella/microsome mutagenicity assay. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, **455**(1), 29 – 60.

Ralaivola, L., Swamidass, S. J., Saigo, H., and Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Netw*, **18**(8), 1093–1110.

Ramon, J. and Gärtner, T. (2003). Expressivity versus efficiency of graph kernels. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74.

Vert, J. P. (2007). Graph kernels and applications in chemoinformatics. In *International Workshop on Graph-based Representations in Pattern Recognition (Gbr 2007), Alicante, Spain, June 13, 2007*.