Introduction to chemoinformatics for AI drug discovery

@fmkz___, @iwatobipen

Version 0.40002(Draft) 2019/03/20

目次

Chapter 1: Introduction

Chemoinformatics is a methodology that is used to analyze mainly chemical-related data using a computer and solve various problems. The term chemoinformatics was defined in the late 1990s and early 2000s, and in the pharmaceutical industry and pharmaceutical academia, the relationship between drug effects and compound characteristics is analyzed, large amounts of compound information are visualized, and compound similarity It is used in a wide variety of processes, including gender-based clustering.

In recent years, drug discovery applications for deep learning have been explored, but not only in conventional chemoinformatics such as **new design proposals** and **synthetic route proposals**, as well as QSAR (Quantitative Structure-Activity Relationship) for predicting activity and physical properties. Applied research to areas that were not being conducted is also actively conducted.

Compound design is innovative

> What kind of compound should we make in the first place? And how to synthesize it? The process of thinking about the background is an area where background knowledge and imagination are required, and conventionally it has been recognized that it is a difficult area for people other than to bear, but the advancement of what is also called AI to such areas is here It progressed rapidly in several years (2017-2019).

Cheminformatics has already been used in various situations, but there was not much relevant information. There are several possible reasons for this, but there is no doubt that the two main reasons are that there were no open source toolkits and no public databases. However, with the advent of RDKit, an open source chemoinformatics toolkit called RDKit and a public database called ChEMBL, this has been resolved.

In recent years, as with bioinformatics in chemoinformatics, a lot of information can be obtained immediately by searching on the web, and it is possible to learn by yourself, but as a set of information to take a first step, We decided to prepare "the content that could learn the basics of chemoinformatics and apply them". Considering the recent AI drug discovery boom, the latter chapter contains chapters on compound activity prediction and compound proposal using deep learning used in the context of “AI drug discovery”, so one-stop learning So you should be able to keep up with the recent trends.

What is RDKit

warning

> Here is a subsection of @ iwatobipen's talk about RDKit. At the draft stage, the words such as "I will say" or "based on" are used as they are, and the self-proclaimed is a "comprehensive" @ iwatobipen-style style of "gozuru" tone.

My name is @iwatobipen, who writes a part of this book. I'm going to talk hot about RDKit here.

What is the RD of RDKit? Actually, it is an abbreviation of **Rational Discovery** , and a framework that is the predecessor of the current open source was developed in 2000. It's so old and old. Then, in 2006, the code became open source and was released from sourceforge. Readers who think that Python's chemoinformatics toolkit includes OpenBabel besides RDKit will also be welcome. OpenBabel was first released in 2005. All come with a toolkit that has more than 10 years of history. I remember that OpenBabel was the major in around 2012, when the deaf people began to be interested in this area. At that time, there were almost no articles in Japanese, and the person who wrote this book was a trial and error writing the code of RDKit referring to the chemo info cookbook of @fmkz___ who is a co- author of this book and a pioneer in the industry Oh. If you want to keep track of chemoinfo related history, you should read this article.

Developer Greg Landorum says

> RDKit is the Swiss Army Knife in chemoinformatics and is a collection of various functional pieces
>
> — Greg Landorum

This is exactly the expression which got the target. As you can see if you look at the link:official document , it already has various features. Starting with reading and writing of compound information, drawing of structure, 3D structure conformation generation, R group decomposition, descriptor, fingerprint calculation, pharmacophore calculation etc. Oh. It can cover a wide range from analysis to visualization. Furthermore, the tools developed by Contributor and others using RDKit are packed in the Contrib folder along with their hot feelings . How do you want to use it? Now I want to write code with RDKit as soon as possible, I cant't wait ;)

NOTE	@iwatobipen is, of course, one of the contributors, and provides code to quickly cluster a large number of compound libraries called Fastcluster . (by @fmkz___)

RDKit is also active in the development and user community, with more features being added. The style in which talented researchers from all over the world build up and develop as a whole is the strength and attraction of open source. If you have a chance, consider joining the annual RDKit User Group Meeting. It is hard to replace anything with Face2Face that users can discuss each other. In addition, I said that there was almost no information on Japanese at the time when the deaf began to use it, but in recent years there have been a lot of very good Japanese articles. Here are a few examples: There are many articles posted on Qiita.

In addition, RDKit-users-jp by volunteers has also been launched. If your question in English seems to be a bit ..., I would like to ask a question here. Also, Japanese documents are merged into the

latest version of RDKit's repository. This will also be helpful. This document only uses some of RDKit's features. You should still feel that you can do a lot of things. Once you have taken the first step of interest, you should go ahead with your own interest and motivation. If you do not understand something, ask the above community and post it to the repository of this book as an issue. **Well then let's get started!**

Main Japanese Commentary Site

- rdkit-users.jp
- RDKitドキュメンテーション非公式日本語版サイト:Unofficail site of rdkit documentation
- 化学の新しいカタチ:The shape of new chemistry

Target audience

The following people are assumed as readers.

- Postdoctoral student who wants to do data analysis of graduate students in pharmacy and medicine and pharmacy
- Pharmacist at a pharmaceutical company who wants to analyze his own data
- Those who feel the need for chemoinformatics in drug discovery chemists and those who are assigned suddenly due to the power of mystery
- Bioinformaticians who are thinking of learning chemoinformatics
- People who are interested in AI drug discovery but do not know what to start with

About the code of this book

All of the programming code used in this book is located in the notebooks directory of the py4cheminformatics repository of Mishima.syk. The first one of each of the  chapter please see properly because it stretched a link to the chapter of Jupyter notebook to.

The installation of Chapter 2 will enable you to use git commands, so you can download all the data in this manual including pdf with the following command

```
$ git clone https://github.com/Mishima-syk/py4chemoinformatics.git
```

bonus

Chemoinformatics or Cheminformatics?

Chemoinformatics or Cheminformatics? Originally I remember that Bio and the combination of the word “Chemo” appeared, but it was widely separated from Chem for a while by the launch of the Journal of Cheminformatics.

According to the recent Google trend, it seems either way, but personally I think that it is better to put emphasis on Rhyme, so I will use Chemo in this book.

Acknowledgment

We would like to thank the following people for their bug fixes and suggestions for improvement when writing this document:

@antiplastics, @bonohu, @ReLuTropy, @ski_nanko, @torusengoku, @yamasaKit_ @4Elemento, @4Elemento, thanks a lot for tranlation task!!!! (from @iwatobipen)

From here onwards I wrote while listening to Nujabes-reflection eternal by @fmkz___ 20/03/20

First of all, I would like to thank the @bonohu which triggered me to write this book. @Bonohu's Dr. Bono's analysis of life science data. At athe meeting of Mishima.syk we talked that "The Bono book Chemoinformatics version" would be nice. There is no doubt that what triggered me to write this book is, "Well, if yes, why not write?" Also, link: @souyakuchan Drug Advent Calendar 2018, written in Japanese has also become a good stimulus for writing. In other words, I think that I did not start to move specifically if I did not make a chapter here.

Also, it is the existence of y-sama that should not be forgotten. Mishima.syk y-sama has been away at the beginning and has fallen forever on 2019/01/06. He wrote wonderful post such as Python environment construction of the person who aims at the data scientist 2016 and Small talk about drug likeness: written in Japanese. If he was alive, we would probably write by three people and the content would have been more complete. This event also gave us a strong motivation to write.

Finally, I would like to thank the participants who participated in Mishima.syk for drinking good wine and beer and having a hot discussion every time. Some content is based on the presentation at Mishima.syk, and has been revised based on your feedback.

If you have read this book, and if you feel that chemoinformatics is interesting or you want to do drug discovery, please join Mishima.syk. I think it will be fun. In future drug discovery research, it will be important to push each other across affiliations and improve their skills. In fact, I think it is already such a society. I hope this book will help you have a pleasant research life.

> I do what I want to do I live myself, I have no regrets in my life. Life enjoys winning. I think it would be fun to enjoy your life by chasing your joy to the fullest by saying that you hate something you hate. I wish you all the best in your life.
>
> — y__sama

License

Chapter 2: Prepare the Environment for Chemoinformatics

We will build the environment required for this document.

About Anaconda

Anaconda is a package for easy environment creation and management for doing machine learning. You can also easily install packages, like RDKit, which will be explained later.

Q&A

Why use Anaconda?

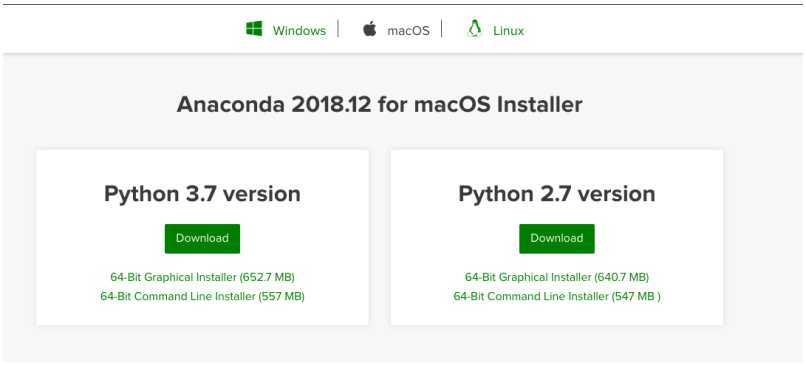
> The programming language Python has a relatively large number of standard libraries, but you need to install the libraries for chemoinformatics yourself. This is not a big deal if you get used to it, but it will be troublesome for beginners. Anaconda comes into play in order to reduce this effort.

There are two major versions of Python: 2.x and 3.x.

> Support for 2.x will end in 2020, so new learners do not need to use 2.x.

How to install Anaconda

Now let's install Anaconda. Visit the official site and download the Python 3 installer for your environment. If the OS is Linux / Mac, you can select the installer of GUI / CUI, so download Python 3.7 64-bit command line installer.




```
$ bash ~/Downloads/Anaconda3-4.1.0-Linux-x86_64.sh # Please change the installer name accordingly
```

Press Enter

```
Welcome to Anaconda3 2018.12

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Continue to press Enter and enter yes with yes, no

```
Do you accept the license terms? [yes|no]
[no] >>>
```

I am asked where to install, but the default location is usually fine. Press Return.

```
Anaconda3 will now be installed into this location:
/Users/kzfm/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below
```

You will be asked if you want to install VSCode after installation as well, so press No.

```
Thank you for installing Anaconda3!

===========================================================================

Anaconda is partnered with Microsoft! Microsoft VSCode is a streamlined
code editor with support for development operations like debugging, task
running and version control.

To install Visual Studio Code, you will need:
  - Internet connectivity

Visual Studio Code License: https://code.visualstudio.com/license

Do you wish to proceed with the installation of Microsoft VSCode? [yes|no]
>>> Please answer 'yes' or 'no':
>>>
```

Once the Anaconda installation is complete, you will be able to use the 'conda' command from a command prompt or terminal.

Build a Virtual Environment and Install a Package

Python installed with Anaconda is 3.7, but the latest RDKit distributed at the time of this writing requires Python 3.6. So build a virtual environment with conda and install the required version of Python. After the -n of the command is "py4chemoinformatics", but you can use any name you like. After creating the virtual environment, install the packages used in this chapter and later.

```
$ conda create -n py4chemoinformatics python3.6
$ source activate py4chemoinformatics # Mac/Linux
$ activate py4chemoinformatics # Windows

# install packages
$ conda install -c conda-forge rdkit
$ conda install -c conda-forge seaborn
$ conda install -c conda-forge ggplot
$ conda install -c conda-forge git
```

Description of installed package

RDKit

RDKit is one of the most commonly used toolkits in the field of chemoinformatics. One of the so-called open source software (OSS), which can be used free of charge. For more information Please refer to Introduction.

seaborn

It is one of the packages for visualizing statistical data.

ggplot

One of the graph drawing packages is that it can draw rationally with a consistent grammar . Originally developed for the statistical analysis language R, it was ported to Python by the company yhat .

Git

It is a version control system. I will not explain Git in this book, but if you do not know Git at all , take a look at Git Primer, which can be understood by monkeys.

As explained in "Introduction", all data including pdf will be downloaded by the following command, so please download it as necessary.

```
$ git clone https://github.com/Mishima-syk/py4chemoinformatics.git
```

Learn more about Conda

Why create a virtual environment

Some systems use Python internally to provide various features, so changing the Python version for a particular package can cause problems. Virtual environments solve these problems. Even if the package requires different library versions, you can set up a virtual Python environment for trial and error. If it becomes unnecessary, the virtual environment can be easily deleted without causing any problems in the original environment. So, by being able to create separate development environments in one system, you will not be bothered by library dependencies problems and Python version differences that often occur during development.

In this document, only one virtual environment is prepared for this document, but in practice many virtual environments are often created and developed. Therefore, I will list the conda subcommands that I use frequently.

```
$ conda install <package name>  # install package
$ conda create -n <Name-of-virtual-environment> python = <version>  # Create virtual
environment.
$ conda info -e  # Display virtual environment list created
$ conda remove -n <environment-name> # Virtual environment deletion
$ source activate <environment-name> # Using virtual environment ( Mac/Linux)
$ activate <environment-name> # Using virtual environment (Windows)
$ source deactivate # leaving virtual environment
$ conda list # Display a list of libraries installed in the virtual environment you
are using now
```

Chapter 3: Basics of Python programming

Python basics

This chapter introduces web sites and books **for effective learning** for python beginners. If you have something that is not understood in the following chapter, these information will help you.

Would like to learn Python from books

Pythonスタートブック増補改訂版:Python start book

We recommend the book if you are beginner of programming.

みんなのPython 第4版:Python for everyone

If you have any experience in programming such as Javascript and/or Java, and would like to learn python now, this book is recommended.

Would like to learn Python from any sources

Python Boot Camp(tutorial for python biginners)

This is a python tutorial event for beginner held by　PyconJP. The events held on all the places of Japan. How about join the event when it take place neighbour?

Local communities

It seems good to increase your motivation to join study workshops for beginners or for professionals. You can find many workshops in connpass site.

udemy/python

It is effective way to learn programming with online learning service but we have never tried. You should ask a reputation around. And also there are many resources in YouTube.

If you have something that is not understood in this book

py4chemoinformaticsのissues

We are happy to answer your question if you put questions in the issue of py4chemoinformatics. If there are something that is difficult to understand we will correct them. The cycle will make the document better and everybody will be happy ;)

stackoverflow

Stack Overflow is good community. You should search in SOF first and then ask the community.

Mishima.syk

Mishima.syk is the community where people who write the book gather. Topics are not limited to python but there are many presentations about python now. Discussion level is high but the community is also beginner friendly. We have planned hands-on sessions and they have an established reputation. The community members should be able to answer your questions.

Let's use it conveniently with Jupyter notebook

By using Jupyter notebook, it is easy to write code and check the results.

The Jupyter Notebook is an open-source web application that allows you to embed code, rich text, math equation and etc. And it is easy to make high quality visualizations of the results. It is a nice platform for chemoinformatics because Jupyter Notebook can run code and draw chemical structures and many kinds of plots. Also, it has many features which improve programming productivity such as syntax highlight and auto indent. We recommend to use Jupyter especially for programming beginners.

How to use?

from terminal (in Windows, anaconda prompt)

```
$ jupyter notebook
```

After type the command above, Jupyter Notebook will be launched. In this book, all code is run on Jupyter Notebooks.

For machine learning with Python

Machine learning is a must for learning informatics not only chemoinformatics. Some background knowledge of machine learning is required in the following sessions. Scikit-learn is used for machine learning with python. Scikit-learn is de facto standard for machine learning library for python. We use the package without any descriptions but we would like to share some links for beginners.

Introduction to Machine Learning with Python

You can learn basics of machine learning with python. It is easy to read because there is less mathematical representations.

sklearn-tutorial

Sklearn tutorial hands-on by @y-sama. Written in jupyter notebook.

Chapter 4: Public databases for chemoinformatics

The section describes common databases which are used for chemoinformatics.

ChEMBL

ChEMBL is a manually curated database of ADMET, physchem and bioactive molecules with drug like properties. The data is mostly curated from medicinal chemistry journals and updated every 3-4 months.

The database is useful for drug discovery research because user can access a QSAR information and background knowledge of original reference journal from the database.

NOTE	Originaly, ChEMBL was commercial database named StARlite. Details are described in this silde deck about ChEMBL.

PubChem

PubChem is an open chemistry database of biological activities and molecules which is maintained by NCBI. It has more than 50 million compounds data and more than 1 million of biological assay dataset. Its large dataset is one of the main features of pubchem. Another feature is that the database grows up by data registration from academia, being this the biggest difference point to ChEMBL. You can check more details of the data source from current URL.

Especially pubchem has large amount of an eary stage screening data, so it will be useful when user would like to analyze or mining it.

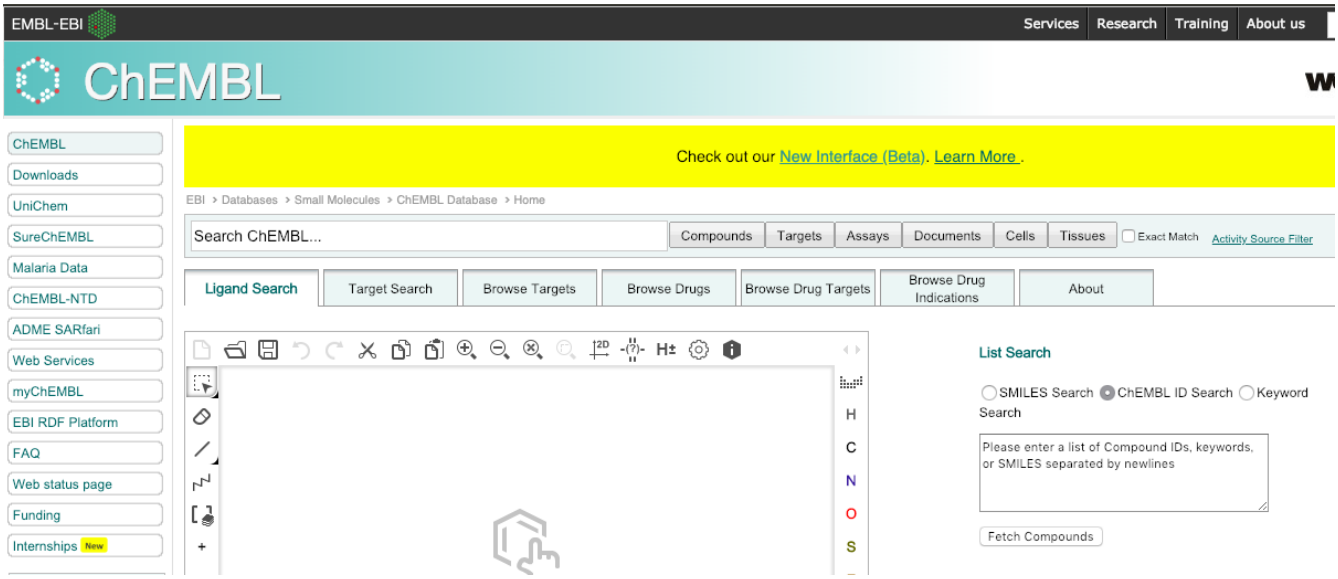
Which database should I use ChEMBL or PubChem?

> We think ChEMBL is preferred for QSAR analysis because ChEMBL provides many data such as IC50 and user can access to original journal for QSAR model interpretation.

Search Data which you want in ChEMBL

NOTE	User interface of ChEMBL is refleshing and testing beta version now. In this section describes how to search data from new UI because the UI will be main near the future.

At first, go to ChEMBL and click the link 'Check out our New Interface (Beta)' on the top of the screen. Then you can move to new search page.



Mainly ChEMBL has 4 data categories and each data has an unique id and has relations to other categories. Brief introductions are below.

Targets

The category has assay and reported journal informations of target molecules.

Compounds

The category has basic physicochemical properties of molecules such as Molecular Weight, whether the molecule passes Lipinksy's Rule of 5 or not. And other information about the molecule such as clinical, related assays which are stored in ChEMBL and summary of journals.

Assays

The category has relationship between assay information and original journal and link for the compounds which was assayed.

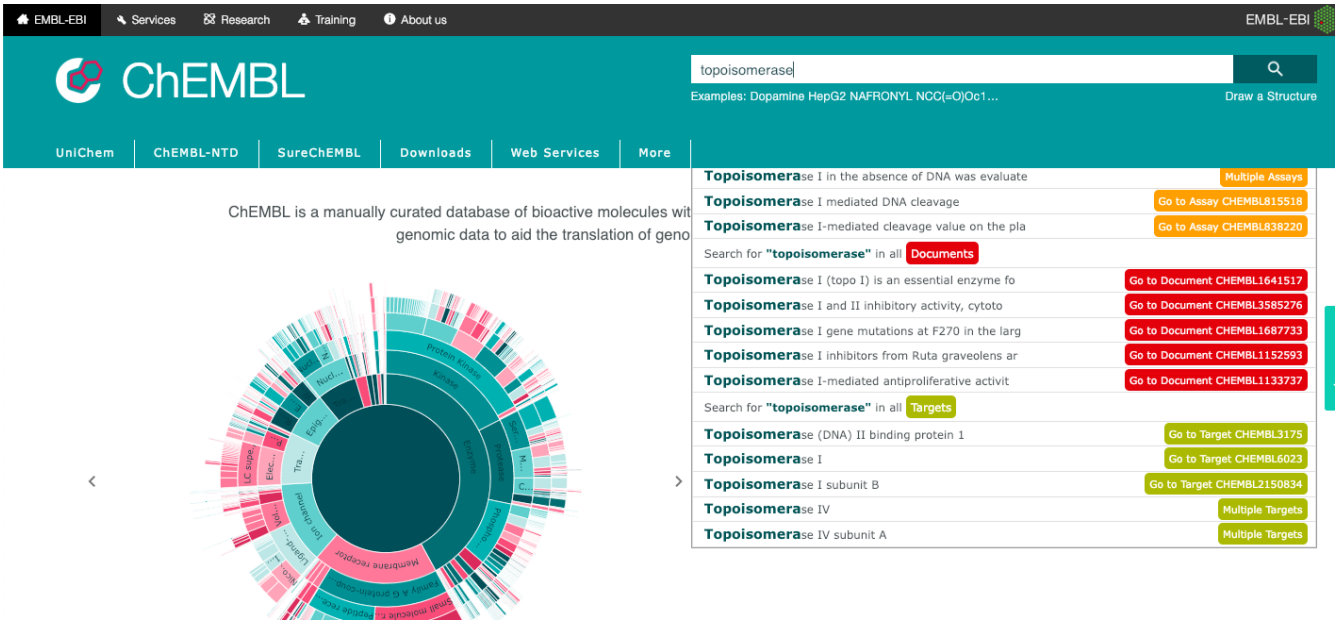
Documents

The category has journal name, title, abstruct and link to related journals and link to data of the comounds which are used in the journal.

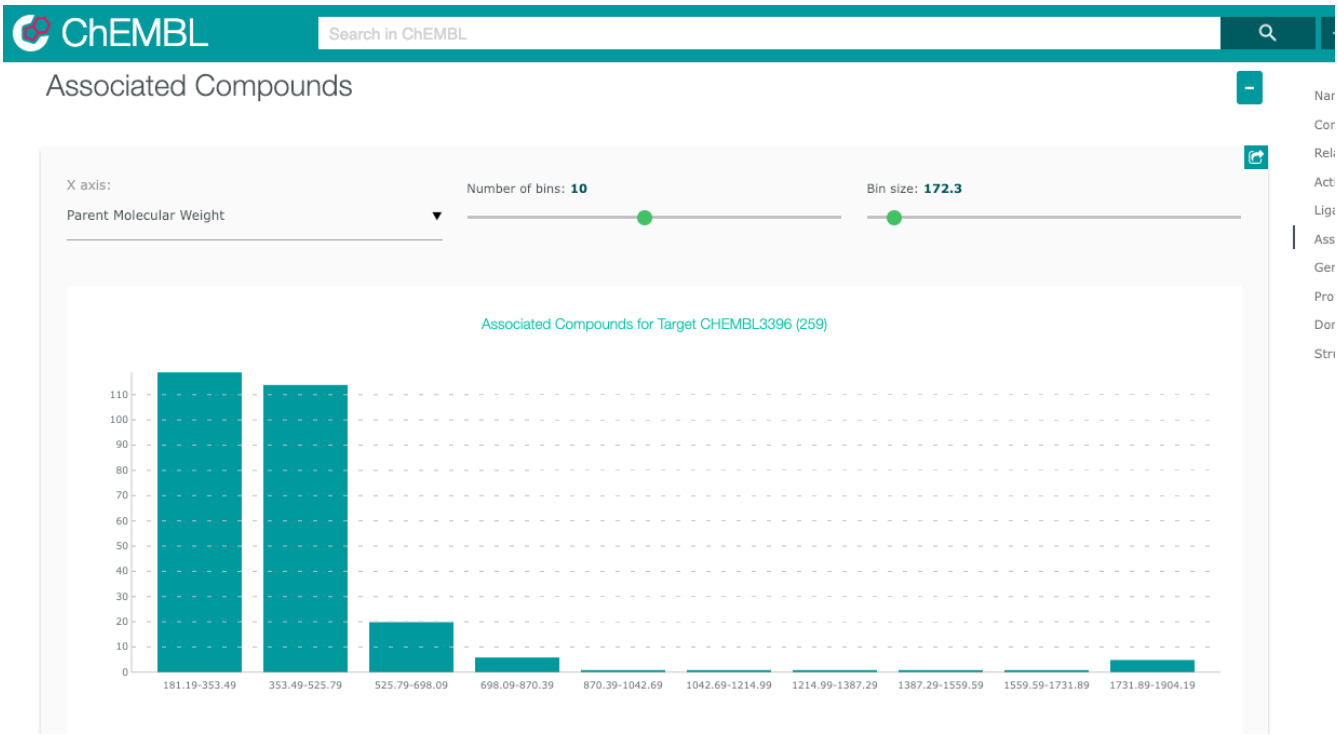
If you want to find compounds which are related to a specific target

It is very common that we want to know how long a target has been studied, how many compounds are synthesized and how kinds of scaffolds are there.

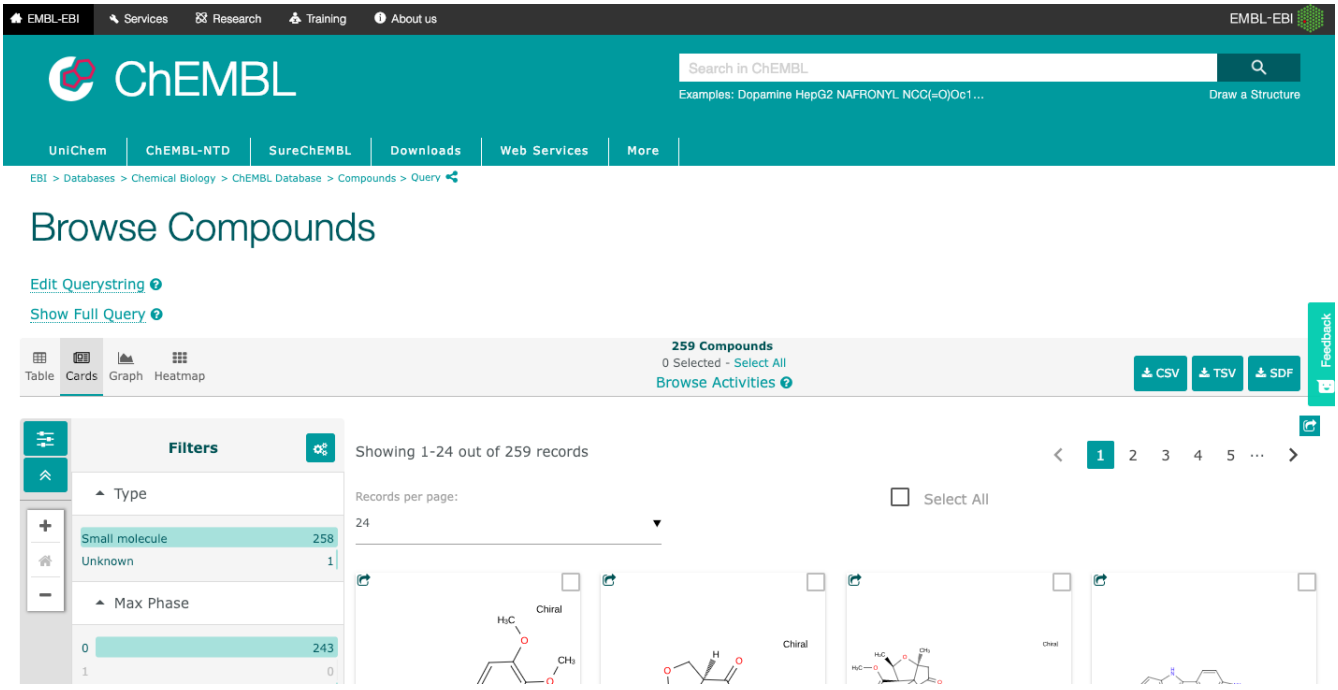
In this section, let's search Topoisomerase2 which is known popular target of cancer chemocerapy treatments. When you input the word **topoisomerase** in to the form which is located on top of the screen and search you can see the result as below.



The system provides candidates list with suggest feature. So you should select TOP2B. You can find section of 'Associated Comounds' when you scroll the screen, you shoud click the title of graph named **Associated Compounds for Target CHEMBL3396** then related compounds list display will appear.



There are 259 compounds in the result. All data can see by scrolling the screen. And data can be downloaded as CSV, TSV and SDF format when you click the icon which is located on top right side of the screen. TIPS:: TSV means tab separated value, CSV means camma separated value



If you want to retrieve comound structures and assays data from ChEMBL

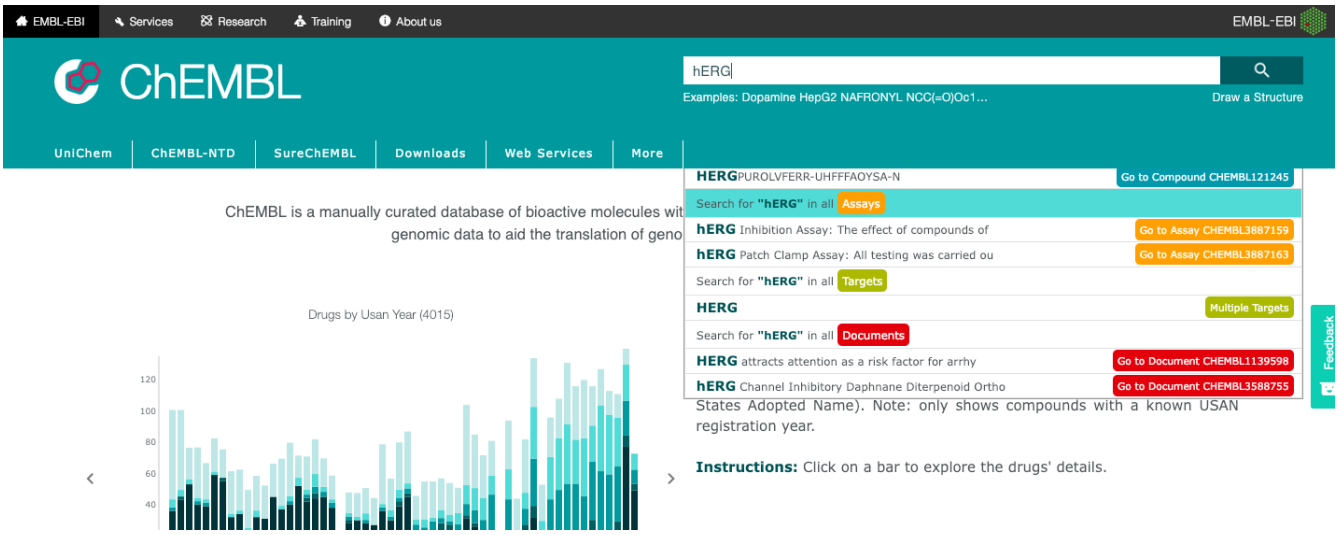
It is needed the structures and activity details for compounds when you would like to build QSAR model. You can download the data for QSAR from **Assay** page in ChEMBL.

You can follow the steps outlined below.

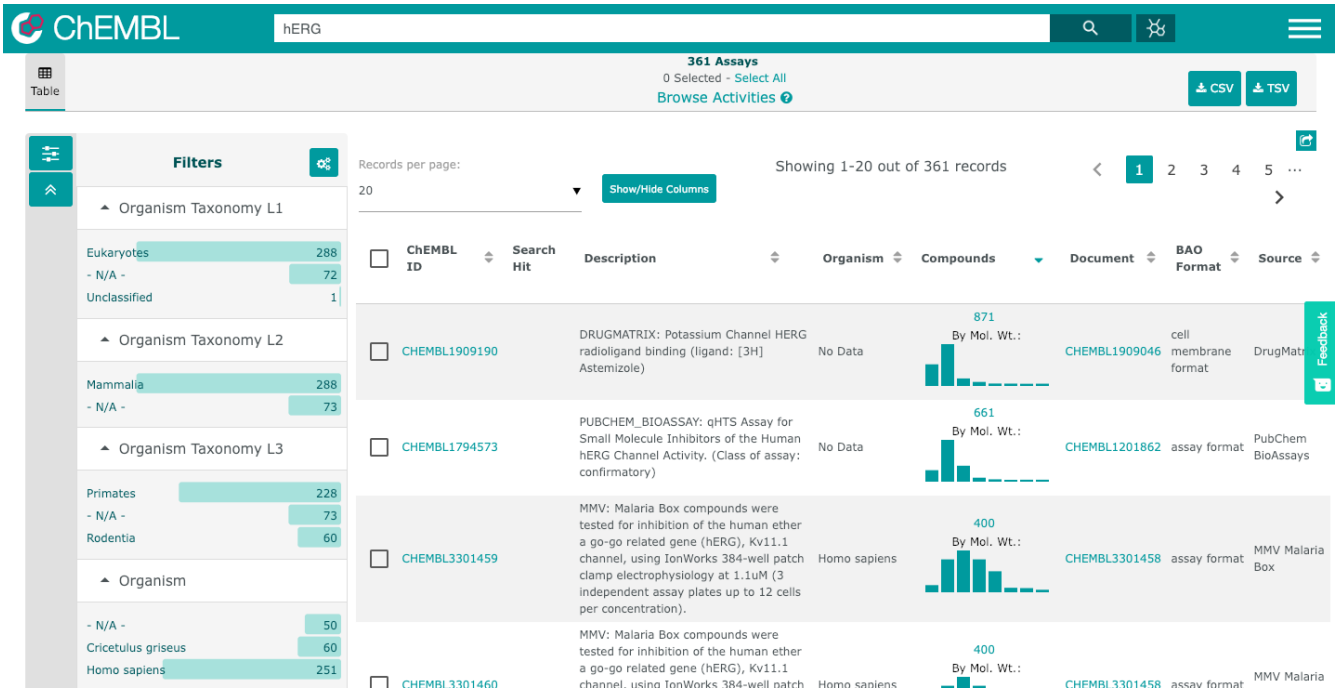
- Search journal data and the retrieve assay data which is related to the journal.
- Search the target which you want to use and retrieve assay data which is related to the target.

In the section, let's try the second approach, retrieve data from the target. We supporse that we would like to build QSAR model for hERG inhibition, hERG, Kv11.1 channel is best known for its contribution to the electrical activity of the heart. The hERG blocker will have risk of cardiotoxicity.

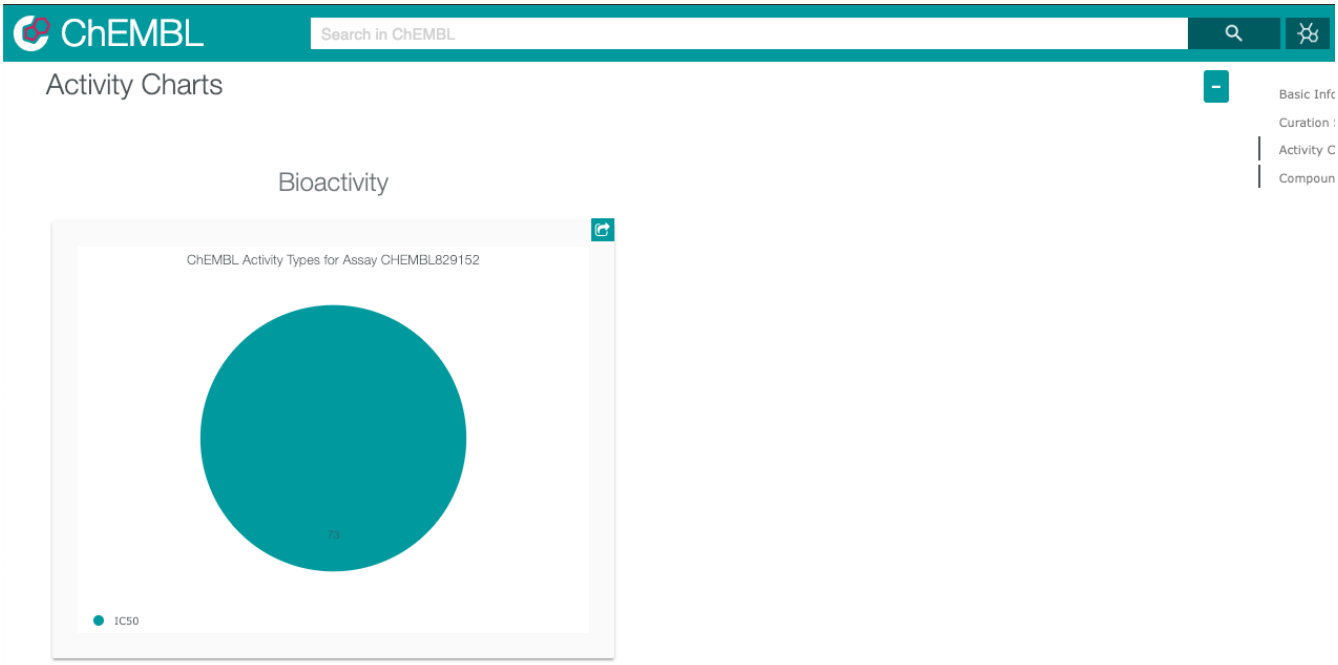
Input **hERG** to search form and push **Search hERG for all assays**. You will can get 361 or more hits.



Sort in descending order of number of data for modeling. Click **Compounds** on the header to do it.



Click CHEMBL829152 which has largest data in the results the assay page will open. Click pi chart of acitivity then details of the data will be shown then select all and download the data as TSV format.



NOTE

The data might be garbled when you open the data on text editer like ^@C^@h^@E^@M^@B^@L^@. This reason is that the data encoded as utf-16-le. (Because the encoding is preferred for Excel)

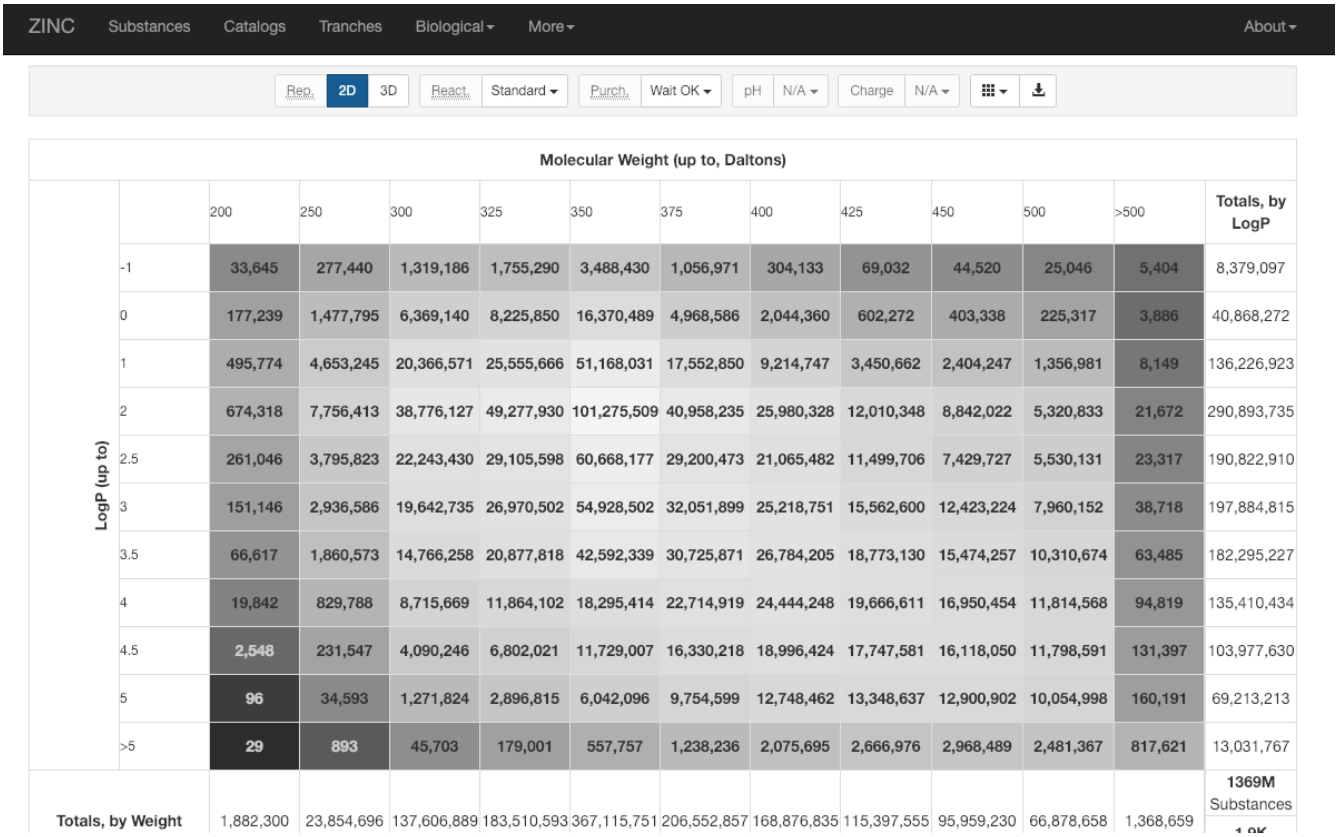
If you are using vi, you can fix the issue by just typing ':e ++enc=utf16le'.

Other useful datbases

ZINC

ZINC is a database which collected commercial available reagents. Current version is 15 and about 750 million comounds are recorded. User can download 3D molecular structure data because originally the data base is developed for assuming docking simulation. I think that conduct virtual screening with data from ZINC, purchase hit comounds and assay these compounds is the main usage.

How to download data? Click Tranches tab, then you can see on the next screen, the table which is devided the vertical axis shows LogP the horizontal axis shows molecular weight display a table of how many compounds are listed.



Molecular Weight (up to, Daltons)

LogP (up to)	200	250	300	325	350	375	400	425	450	500	>500	Totals, by LogP
-1	33,645	277,440	1,319,186	1,755,290	3,488,430	1,056,971	304,133	69,032	44,520	25,046	5,404	8,379,097
0	177,239	1,477,795	6,369,140	8,225,850	16,370,489	4,968,586	2,044,360	602,272	403,338	225,317	3,886	40,868,272
1	495,774	4,653,245	20,366,571	25,555,666	51,168,031	17,552,850	9,214,747	3,450,662	2,404,247	1,356,981	8,149	136,226,923
2	674,318	7,756,413	38,776,127	49,277,930	101,275,509	40,958,235	25,980,328	12,010,348	8,842,022	5,320,833	21,672	290,893,735
2.5	261,046	3,795,823	22,243,430	29,105,598	60,668,177	29,200,473	21,065,482	11,499,706	7,429,727	5,530,131	23,317	190,822,910
3	151,146	2,936,586	19,642,735	26,970,502	54,928,502	32,051,899	25,218,751	15,562,600	12,423,224	7,960,152	38,718	197,884,815
3.5	66,617	1,860,573	14,766,258	20,877,818	42,592,339	30,725,871	26,784,205	18,773,130	15,474,257	10,310,674	63,485	182,295,227
4	19,842	829,788	8,715,669	11,864,102	18,295,414	22,714,919	24,444,248	19,666,611	16,950,454	11,814,568	94,819	135,410,434
4.5	2,548	231,547	4,090,246	6,802,021	11,729,007	16,330,218	18,996,424	17,747,581	16,118,050	11,798,591	131,397	103,977,630
5	96	34,593	1,271,824	2,896,815	6,042,096	9,754,599	12,748,462	13,348,637	12,900,902	10,054,998	160,191	69,213,213
>5	29	893	45,703	179,001	557,757	1,238,236	2,075,695	2,666,976	2,968,489	2,481,367	817,621	13,031,767
Totals, by Weight	1,882,300	23,854,696	137,606,889	183,510,593	367,115,751	206,552,857	168,876,835	115,397,555	95,959,230	66,878,658	1,368,659	1369M Substances

Select dataset which you want and click down load button, you can get text file which listed URL of the dataset. The data can get with accessing the URL.

統合TV:Togo TV

Togo TV is a video site which describes useful database and tools and is managed and maintaind by Database Center for Life Science(DBCLS). As its name suggests that there are many videos about bioinformatics, but there are some chemoinformatics videos are provided. Please reffer the site. journal・dictionary・programminc might be useful. **Language of TogoTV is Japanese**

- PubChemを利用して化学物質やアッセイの結果を調べる 2017/Search compound and assay data by using PubChem 2017
- ChEMBLを使って医薬品候補となる化合物について調べる/Search drug candidate comounds with

ChEMBL

NOTE

If reader know other useful databases for chemoinformatics please inform us. Issue or Pull requests are also appreciated.

Chapter 5: Handling Structural Information with RDKit

In this chapter we will learn the basics of reading molecules with RDKit.

What is SMILES?

Simplified molecular input line entry system(SMILES) is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings. More detials are described in SMILES Tutorial. For example **c1ccccc1** means that there are six aromatic carbon atoms and has a loop structure which is connected with start and end, you know it means benzene.

Let's draw chemical strcutre with SMILES :)

We could understand SMILES can represent molecules, so let's read SMILES and draw molecule. At first import Chem class from RDKit to do that. And the function in the second line named 'IPythonConsole' is read for drawing molecules on Notebook. **The majority of the basic molecular functionality is found in module rdkit.Chem**

```
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
```

RDKit has MolFromSmiles method which reads SMILES. RDKit mol object can be constructed from SMILES with the function like below.

```
mol = Chem.MolFromSmiles("c1ccccc1")
```

Next we draws molecular structure. It is very simple, just evaluate mol object.

```
mol
```

Molecular structure will be drawn like following figure.

```
In [31]: from rdkit import Chem
         from rdkit.Chem.Draw import IPythonConsole

In [32]: mol = Chem.MolFromSmiles("c1ccccc1")

In [35]: mol
Out[35]:
```

Both methods connect atoms with bonds(2D Structure) and SMILES can represent same molecule. 2D structure is easy to understand for us and SMILES is not. But SMILES can define molecule as ASCII strings so SMILES can store molecule in low data volume.

How to handle multiple molecules at once ?

There are several ways to store multiple molecules in a file but SDF format file is common.

What's sdf format?

There is MOL format which was developed by MDL. SDF format is an extension of this MOL format. In particular multiple compounds are delimited by lines consisting of four dollar signs (). A feature of the SDF format is its ability to include associated data.

Huge differnce between MOL format and SMILES format is that MOL format can store 3d geometry information of molecule so MOL format can describe not only 2D but also stereo chemistry.

Download sdf file from ChEMBL

Refer to chapter 4, down load Topoisomerase II inhibitor data(CHEMBL669726) from ChEMBL as sdf file format.

NOTE

Specially, open the link page and input 'CHEMBL66926' to search form then search results will be appeared. Then click compounds tab, select all and down load as SDF. File download will start and get file as compressed gzip format. Extract the file with guzip command or using an appropriate soft then rename the file to ch05_compounds.sdf.

Handling sdf with RDKit

SDMolSupplier method is used as sdf file reader of RDKit. Please note that we use mols variable instead of mol because we handle multiple molecules. There isn't a rule for variables naming but you should use variables name which is easy to understand in order to reduce the unnecessary mistakes.

```
mols = Chem.SDMolSupplier("ch05_compounds.sdf")
```

Check how many coumpounds are read. len method is used to count number.

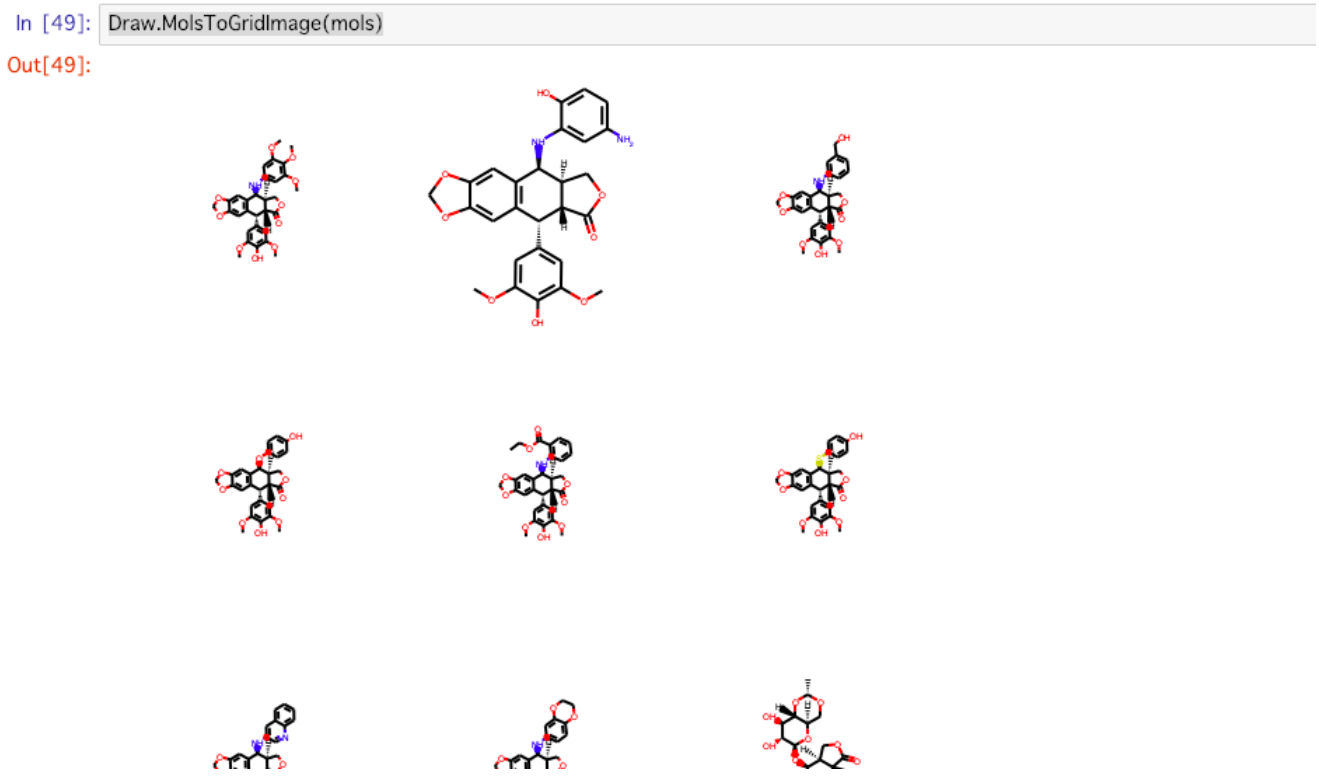
```
len(mols)
```

Total 34 molecules are read.

Draw moleculear structures

You can draw molecule one by one with for loop but it is redundant. RDKit has method which can draw multiple molecules at once, so try to use the function named MolsToGridImage method. For your information the function has molsPerRow option which can change number of molecules per row.

```
Draw.MolsToGridImage(mols)
```



(bonus)

Following code shows draw molecule one by one with loop for your information.

```
from IPython.core.display import display
for mol in mols:
    display(mol)
```

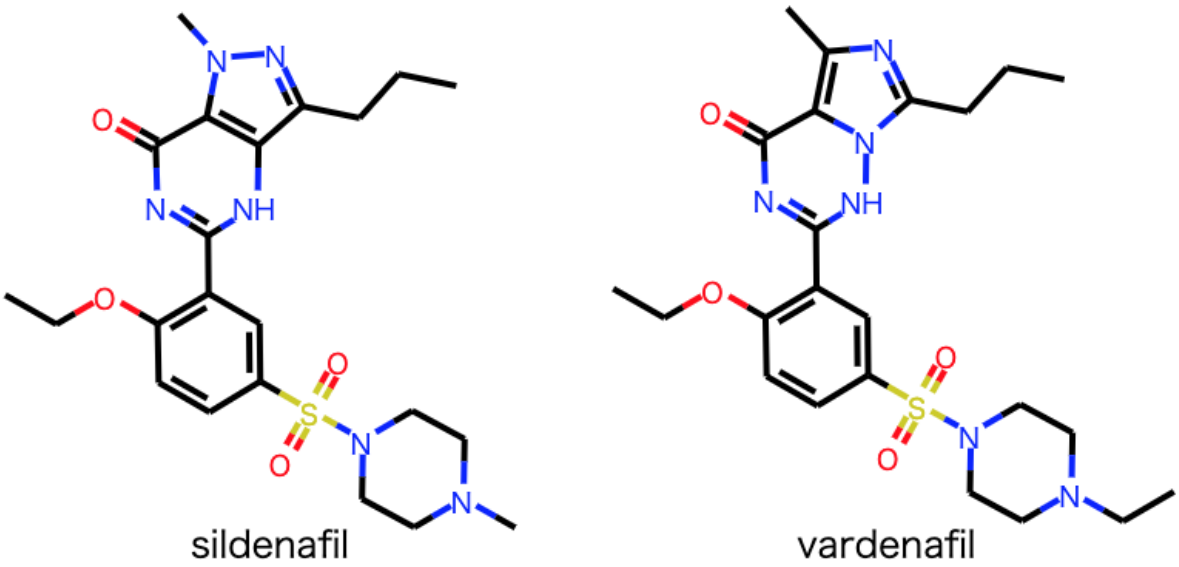
Let's try to do hetero shuffling

At the leard optimization satage of drug discovery, it often happens that researchers would like to improve molecular properties without changing molecular shape. In this case medicinal chemists of chage atoms such as carbon, nitrogen, sulphur and oxygen which in aromatic rings and it generats good profile molecules sometime. The approach which exchange aromatic atoms (except hydrogen) is called heteroshuffling.

The heteroshuffling strategy is expected to improve physchem properties keeping potency, improve potency and claim avoidance.

Pfizer's Sildenafil and GSK's Vardinafil are well-known examples where small structural differences can affect selectivity and pharmacokinetics.

The two structures are very similar except that the arrangement of the nitrogen atoms in the central ring structure is different. Both two molecules inhibt same target protein but their biological activities and pharmacokenetics are different.



Following code shows how to generate image described above. Please note that the code is not just using Draw.MolsToGridImage but align to core structure and add legends option to draw

molecular's name.

```
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from rdkit.Chem import rdDepictor
from rdkit.Chem import rdFMCS
from rdkit.Chem import TemplateAlign
IPythonConsole.ipython_useSVG = True
rdDepictor.SetPreferCoordGen(True)

sildenafil = Chem.MolFromSmiles(
'CCCC1=NN(C)C2=C1NC(=NC2=O)C1=C(OCC)C=CC(=C1)S(=O)(=O)N1CCN(C)CC1')
vardenafil = Chem.MolFromSmiles(
'CCCC1=NC(C)=C2N1NC(=NC2=O)C1=C(OCC)C=CC(=C1)S(=O)(=O)N1CCN(CC)CC1')
rdDepictor.Compute2DCoords(sildenafil)
rdDepictor.Compute2DCoords(vardenafil)
res = rdFMCS.FindMCS([sildenafil, vardenafil], completeRingsOnly=True, atomCompare
=rdFMCS.AtomCompare.CompareAny)
MCS = Chem.MolFromSmarts(res.smartsString)
rdDepictor.Compute2DCoords(MCS)

TemplateAlign.AlignMolToTemplate2D(sildenafil, MCS)
TemplateAlign.AlignMolToTemplate2D(vardenafil, MCS)
Draw.MolsToGridImage([sildenafil, vardenafil], legends=['sildenafil', 'vardenafil'])
```

HeteroShuffle class is defined to generate hetero shuffled molecules. To generate the objects, it is needed to input the molecule which would like to do hetero shuffle and core structure to shuffle. The target atoms are aromatic atoms in the core and atoms which has no substituent. The function named make_connector generates reaction objects to construct molecules from shuffled core and substituents. The function named re_construct_mol reconstruct molecules with the reaction objects.

To generate possible combinations of atoms, the code pass candidates of atomic numbers (C, S, N, O) and number of atoms which constructs target ring. Invalid molecule will be removed after possible combinations is generated.

```
class HeteroShuffle():

    def __init__(self, mol, query):
        self.mol = mol
        self.query = query
        self.subs = Chem.ReplaceCore(self.mol, self.query)
        self.core = Chem.ReplaceSidechains(self.mol, self.query)
        self.target_atomic_nums = [6, 7, 8, 16]


    def make_connectors(self):
        n = len(Chem.MolToSmiles(self.subs).split('.'))
```

```
        map_no = n+1
        self.rxn_dict = {}
        for i in range(n):
            self.rxn_dict[i+1] = AllChem.ReactionFromSmarts(
'[{0}*][*:{1}].[{0}*][*:{2}]>>[*:{1}][*:{2}]'.format(i+1, map_no, map_no+1))
        return self.rxn_dict

    def re_construct_mol(self, core):
        '''
        re construct mols from given substructures and core
        '''
        keys = self.rxn_dict.keys()
        ps = [[core]]
        for key in keys:
            ps = self.rxn_dict[key].RunReactants([ps[0][0], self.subs])
        mol = ps[0][0]
        try:
            smi = Chem.MolToSmiles(mol)
            mol = Chem.MolFromSmiles(smi)
            Chem.SanitizeMol(mol)
            return mol
        except:
            return None

    def get_target_atoms(self):
        '''
        get target atoms for replace
        target atoms means atoms which don't have anyatom(*) in neighbors
        '''
        atoms = []
        for atom in self.core.GetAromaticAtoms():
            neighbors = [a.GetSymbol() for a in atom.GetNeighbors()]
            if '*' not in neighbors and atom.GetSymbol() !='*':
                atoms.append(atom)
        print(len(atoms))
        return atoms

    def generate_mols(self):
        atoms = self.get_target_atoms()
        idxs = [atom.GetIdx() for atom in atoms]
        combinations = itertools.product(self.target_atomic_nums, repeat=len(idxs))
        smiles_set = set()
        self.make_connectors()
        for combination in combinations:
            target = copy.deepcopy(self.core)
            #print(Chem.MolToSmiles(target))
            for i, idx in enumerate(idxs):
                target.GetAtomWithIdx(idx).SetAtomicNum(combination[i])
            smi = Chem.MolToSmiles(target)
            #smi = smi.replace('sH','s').replace('oH','o').replace('cH3','c')
            #print('rep '+smi)
```

```
            target = Chem.MolFromSmiles(smi)
            if target != None:
                n_attachment = len([atom for atom in target.GetAtoms() if atom
.GetAtomicNum() == 0])
                n_aromatic_atoms = len(list(target.GetAromaticAtoms()))
                if target.GetNumAtoms() - n_attachment == n_aromatic_atoms:
                    try:
                        mol = self.re_construct_mol(target)
                        if checkmol(mol):
                            smiles_set.add(Chem.MolToSmiles(mol))
                    except:
                        pass
        mols = [Chem.MolFromSmiles(smi) for smi in smiles_set]
        return mols
```

The checkmol function which is used to avoid molecule such as c1coooo1 is defied as aromatic. I defined molecule which is allowd contain O, S is only five menbered hetero aromatic rings.

```
def checkmol(mol):
    arom_atoms = mol.GetAromaticAtoms()
    symbols = [atom.GetSymbol() for atom in arom_atoms if not atom.IsInRingSize(5)]
    if symbols == []:
        return True
    elif 'O' in symbols or 'S' in symbols:
        return False
    else:
        return True
```

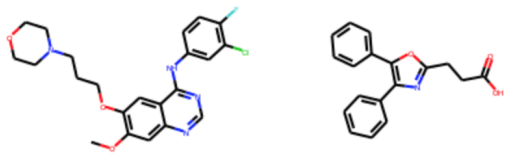
Use the function.

```
# Gefitinib
mol1 = Chem.MolFromSmiles('COC1=C(C=C2C(=C1)N=CN=C2NC3=CC(=C(C=C3)F)Cl)OCCCN4CCOCC4')
core1 = Chem.MolFromSmiles('c1ccc2c(c1)ncn2')
#  Oxaprozin
mol2 = Chem.MolFromSmiles('OC(=O)CCC1=NC(=C(O1)C1=CC=CC=C1)C1=CC=CC=C1')
core2 =  Chem.MolFromSmiles('c1cnco1')
```

Original molecule.

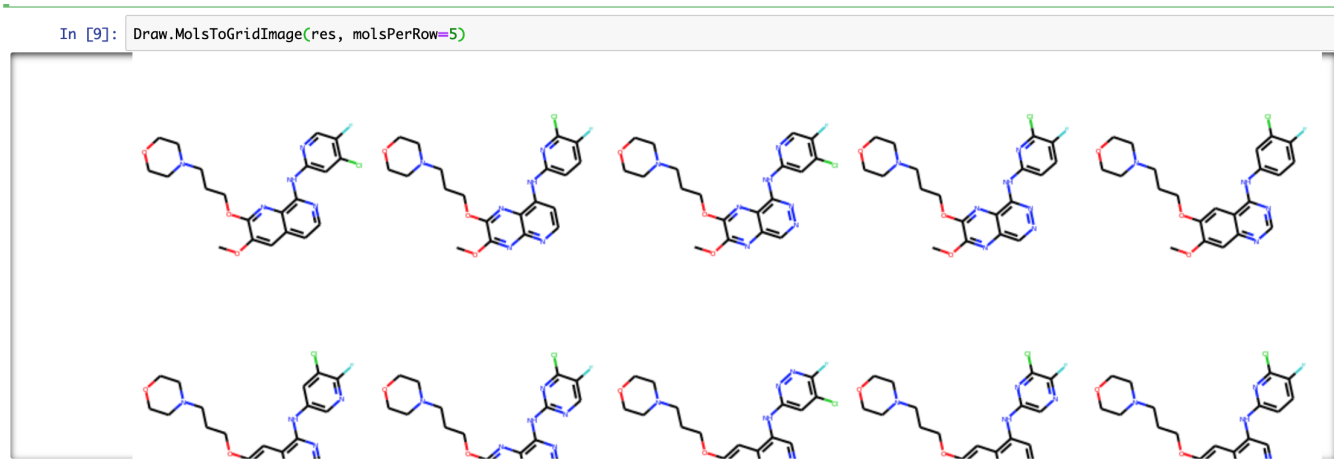
```
: Draw.MolsToGridImage([mol1, mol2])
```

:



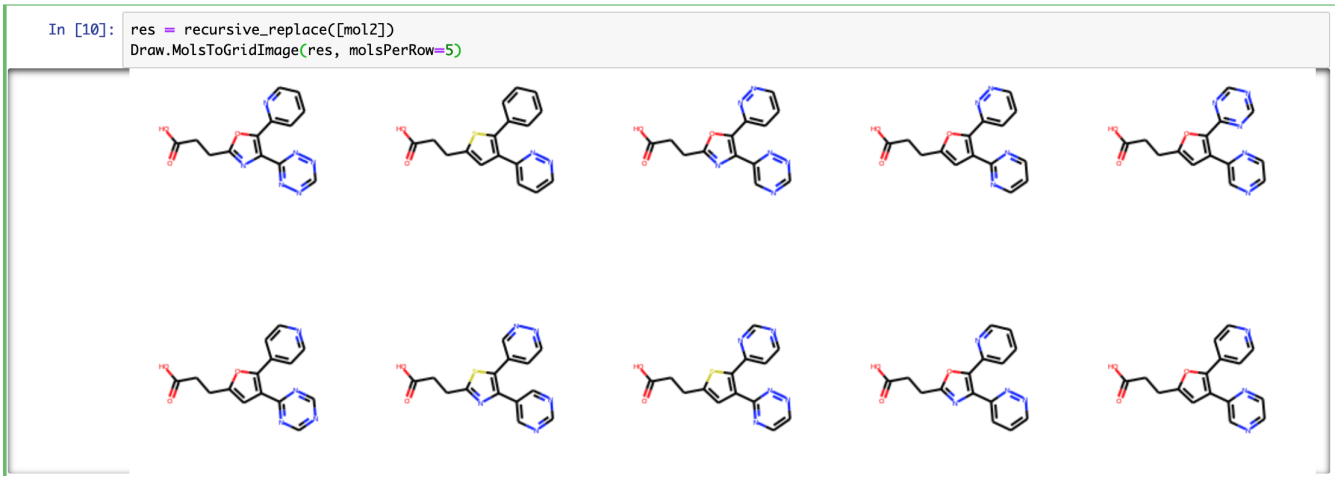
```
ht=HeteroSuffle(mol1, core1)
res=ht.generate_mols()
print(len(res))
Draw.MolsToGridImage(res, molsPerRow=5)
```

The image is part of the results Gefitinib as input. The molecules which is different from original molecule are generated. And quinazoline part is changed because I set quinazoline as core.




```
ht=HeteroSuffle(mol2, core2)
res=ht.generate_mols()
print(len(res))
Draw.MolsToGridImage(res, molsPerRow=5)
```

This is the result of Oxaprozin is used as input. This molecul has ink:https://en.wikipedia.org/wiki/Oxazole[oxazole] which is five menbered ring as core. There are several hetero aromatic rings that conatin oxygen, sulphur such as thiophen furan.



What's on your mind? Two examples were shown. The first one is a case of aromatic rings are quinazoline and benzene. Qunazoline is the ring which is fused ring of benzene and pyrimidine. The candidates atoms for six membered aromatic rings will be carbon and nitrogen atoms. (Of cource if we consider for pyririum ion, oxygen will be candidate of atoms but these charged substructure is not common for drug discovery. So we ommited the atom.) Oxaprozin has an oxazole rings. The candidates of atoms for five membered aromatic rings will be carbon, nitrogen,

sulphur and oxygen. The second one is introduced as an example of five membered hetero aromatic rings. HeteroShuffled molecules are generated in the both case.

Describes about hetero shuffling more

In the article J. Med. Chem. 2012, 55, 11, 5151-5164 analyzed the effect of nitrogen shuffling for PIM-1 kinase inhibitor project with Fragment Molecular Orbital method which is a method of quantum chemistry. And another article J. Chem. Inf. Model. 2019, 59, 1, 149-158 described mechanism of the stackibng between Asp-Arg salt bridge and hetero rings with quantum chemistry calclation. The approach seems to be good indicator for substituents design.

Also, an example of hetero shuffling for improving the bio availability is ink:https://dx.doi.org/10.1021/jm101027s[J. Med. Chem. 2011, 54, 8, 3076-3080]

Chapter 6: Try to evaluate the similarity of compounds

What does it mean that compounds are similar ?

Expressions that are somewhat shape is similar are not scientific. In the chemoinformatics, similarity or unsimilarity (distance) is used as quantitative metrics.

In this section, we will introduce two major metrics.

Descriptor

A parameter that represents the overall characteristics of a molecule numerically is called a descriptor. Many descriptors are proposed so far, such as molecular weight, polar surface area (PSA) and partition coefficient (logP). It is possible to evaluate a similarity between two molecules with these descriptors. Please note that descriptor represents whole molecular feature as a numeric value and it is not a local feature.

NOTE	There are cases where commercial software is needed to calculate some descriptors.

Fingerprint

A fingerprint is another feature, and is a binary representation of a partial structure of a molecule as a binary 0, 1, and it corresponds to the presence or absence of a partial structure and on (1) or off (0) of a bit, and represents a set of partial structures Represents the characteristics of the molecule. There are two types of fingerprints, fixed-length FP and variable-length FP. Formerly, MACSKey fixed-length FP (FP whose partial structure and index have been determined in advance) was used, but now ECFP 4 (It is common to use a variable-length FP called Morgan2).

As for the RDKit fingerprint, please read Developper of RDKit, Greg's Slide for details.

Let's do similarity evaluation using this ECFP 4 (Morgan 2) this time.

Difference between SMILES and fingerprint

SMILES is an ASCII string representation of the structure, and a fingerprint is a binary representation of the presence or absence of a substructure. The difference is that the former is one of the **structural expressions** , while the latter is one of the **feature expressions** . Since only the presence or absence of partial structures is expressed, information such as the relationship between partial structures (how connected by positional relationship) is lost, and the original structure is not restored.

Some people call it Bag-of-Fragments because it corresponds to Bag-of-Words often used in text-mining.

Let's calculate similarity

Let's evaluate the similarity of toluene and chlorobenzene as simple molecules.

```
from rdkit import Chem, DataStructs
from rdkit.Chem import AllChem, Draw
from rdkit.Chem.Draw import IPythonConsole
```

Read molecule from SMILES.

```
mol1 = Chem.MolFromSmiles("Cc1ccccc1")
mol2 = Chem.MolFromSmiles("Clc1ccccc1")
```

Confirm it by visual observation.

```
Draw.MolsToGridImage([mol1, mol2])
```

Generate radius 2 morgan fingerprint which corresponds to ECFP4.

```
fp1 = AllChem.GetMorganFingerprint(mol1, 2)
fp2 = AllChem.GetMorganFingerprint(mol2, 2)
```

Tanimoto coefficient is used for similarity evaluation.

```
DataStructs.TanimotoSimilarity(fp1, fp2)
# 0.5384615384615384
```

Virtual screening

So far we have described how to evaluate the similarity of compounds. Using this index of

similarity to select a specific group of compounds from a large number of compounds is called virtual screening.

For example, if a compound that is likely to be a drug is published in a patent or a paper, or a compound that is likely to be promising is found in our assay system, similar compounds in the compound library database of our company or the database of commercially available compounds are more promising I want to find out if there is something like that. Here, it is possible to purchase an analog of influenza drug which is known as a neuraminidase inhibitor Inavir link:Find out using ZINC.

The molecular weight of Inavir was about 350, and LogP was about -3. So we selected the fraction of the molecular weight 350-375 and LogP -1 from ZINC. This section is divided into 16 files, but download and use only the first set.

NOTE We described how to download the data in chapter 4.

We can perform shell command on jupyter notebook by starting from !. The following is an example of downloading ZINC data set with wget command on jupyter notebook

```
!wget http://files.docking.org/2D/EA/EAED.smi
```

Read SMILES from file and make it a mol object, but skip the first line because it is a header. Also, the last character of each line is a newline character, so it is excluded as l [:-1]. Finally, find out how many compounds there are.

```
mols = []
with open("EAED.smi") as f:
    f.readline()
    for l in f:
        mol = Chem.MolFromSmiles(l[:-1])
        mols.append(mol)
print(len(mols))
# 195493
```

Next, prepare a function to check the degree of similarity with Inavir (LANIMAMIBIR).

```
laninamivir = Chem.MolFromSmiles(
"CO[C@H]([C@H](O)CO)[C@@H]1OC(=C[C@H](NC(=N)N)[C@H]1NC(=O)C)C(=O)O")
laninamivir_fp = AllChem.GetMorganFingerprint(laninamivir, 2)

def calc_laninamivir_similarity(mol):
    fp = AllChem.GetMorganFingerprint(mol, 2)
    sim = DataStructs.TanimotoSimilarity(laninamivir_fp, fp)
    return sim
```

Check it.

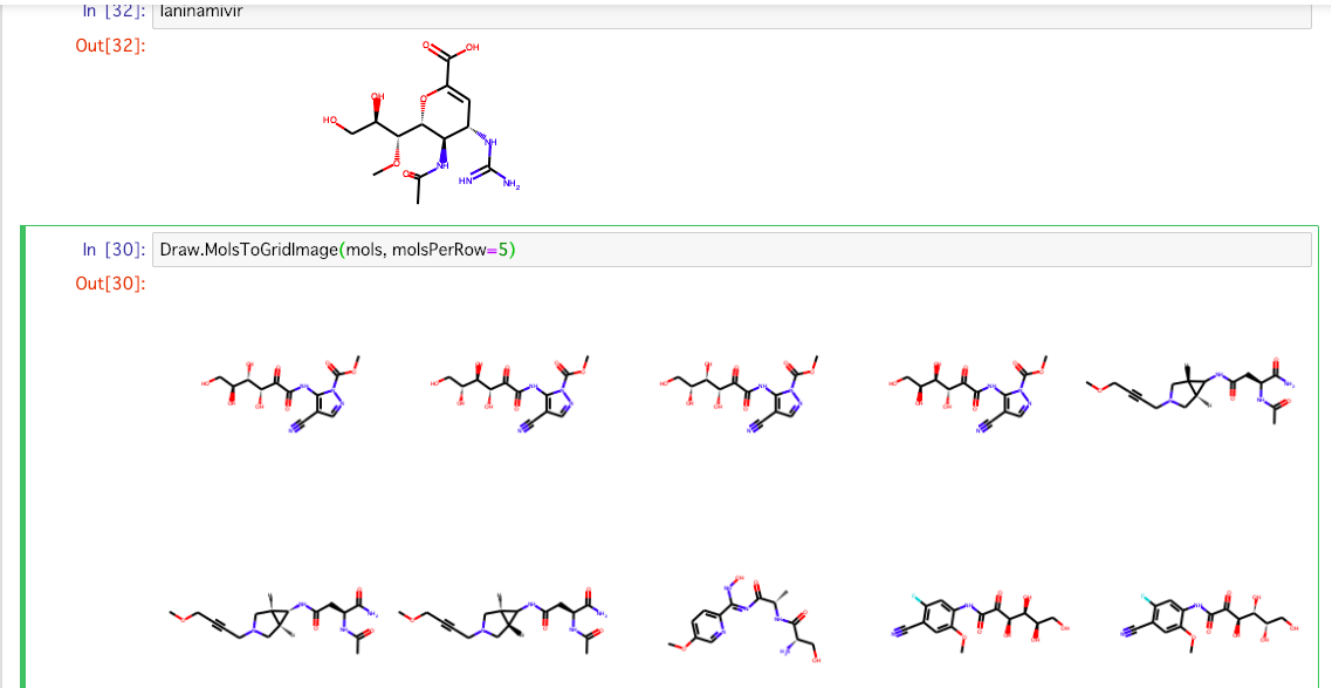
```
similar_mols =[]
for mol in mols:
    sim = calc_laninamivir_similarity(mol)
    if sim > 0.2:
        similar_mols.append((mol, sim))
```

Sort the results in descending order of similarity and retrieve only the first ten.

```
similar_mols.sort(key=lambda x: x[1], reverse=True)
mols = [l[0] for l in similar_mols[:10]]
```

Let's draw them.

```
Draw.MolsToGridImage(mols, molsPerRow=5)
```



As you can see if the similarity is confirmed, about 200,000 compounds examined this time can only find a compound with a maximum similarity is 23%. However, ZINC contains 750 million entries, so there should be many more similar compounds in it.

Clustering

For example, when purchasing a commercial compound and creating a library, we want to have as much diversity as possible, so we organize similar compounds and select a representative of them so that only similar compounds are not biased. In this way, if you want to organize compounds by structural similarity, use a method called clustering.

Clustering of 5614 hits from Novrtis's antimalarial assay

Import library for clustering and reading data.

```
from rdkit.ML.Cluster import Butina
mols = Chem.SDMolSupplier("ch06_nov_hts.sdf")
```

If RDKit can not read the molecule for some reason, it will generate None instead of a mol object. Since passing this None to the GetMorganFingerprintAsBitVect method results in an error, so we generate a fingerprint while excluding None.

```
fps = []
valid_mols = []

for mol in mols:
    if mol is not None:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2)
        fps.append(fp)
        valid_mols.append(mol)
```

Generate a distance matrix (a lower triangular distance matrix) from the fingerprints.

```
distance_matrix = []
for i, fp in enumerate(fps):
    similarities = DataStructs.BulkTanimotoSimilarity(fps[i], fps[:i+1])
    distance_matrix.extend([1-sim for sim in similarities])
```

Cluster compounds using a distance matrix. The third argument is the distance threshold. In this example, clustering is performed on compounds with a distance of 0.2 or 80% or more.

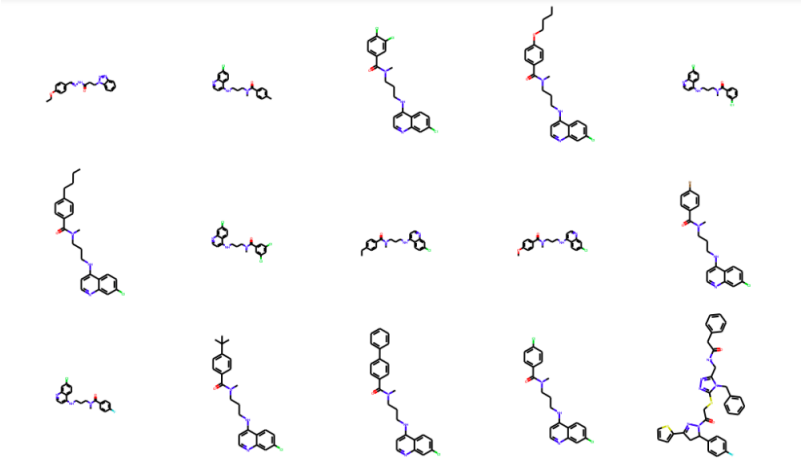
```
clusters = Butina.ClusterData(distance_matrix, len(fps), 0.2, isDistData=True)
```

Check number of cluster.

```
len(clusters)
#2492
```

Visualize structures of first cluster.

```
mols_ =[valid_mols[i] for i in clusters[0]]
Draw.MolsToGridImage(mols_, molsPerRow=5)
```



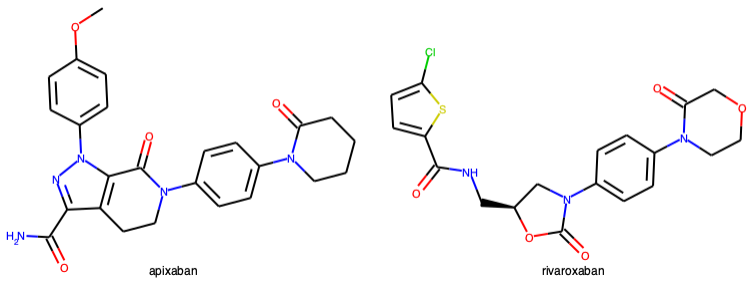
In this case, clustering was performed using the library provided in RDKit, but some methods can be used with Scikit-learn. And in practice this method is often used.

Structure Based Drug Design(SBDD)

Here we evaluate the similarity of apixaban and rivaroxaban, which are marketed as anticoagulants.

```
apx = Chem.MolFromSmiles("COc1ccc(cc1)n2nc(C(=O)N)c3CCN(C(=O)c23)c4ccc(cc4)N5CCCCC5=O
")
rvx = Chem.MolFromSmiles("Clc1ccc(s1)C(=O)NC[C@H]2CN(C(=O)O2)c3ccc(cc3)N4CCOCC4=O")
```

```
Draw.MolsToGridImage([apx, rvx], legends=["apixaban", "rivaroxaban"])
```



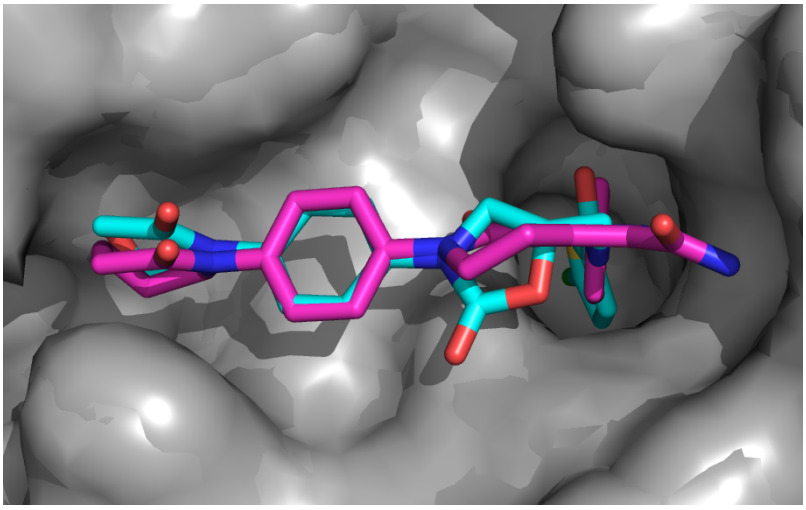
The structures are quite similar as you can see, but both of these two compounds are known to bind similarly to the same pocket of the serine protease FXa and to inhibit the function of the protein.

```
apx_fp = AllChem.GetMorganFingerprint(apx, 2)
rvx_fp = AllChem.GetMorganFingerprint(rvx, 2)

DataStructs.TanimotoSimilarity(apx_fp, rvx_fp)
# 0.40625
```

It's about 40% similar. In fact, both apixaban and rivaroxaban have their complex crystal structures solved and were drawn using PyMOL.

NOTE

It does not explain how to use PyMOL because it exceeds the contents of this document, but if you are interested, Please refer to here.



As you can see from the figure, apixaban and rivaroxaban are beautifully overlapping in three dimensions. In particular, methoxyphenyl and chlorothiol are located in a site called S1 pocket and are said to have some kind of strong interaction. As the ligand binding sites (pockets) of proteins become clearer, it becomes easier for the medicinal chemist to develop a strategy for the next modification, and the success rate and progress rate of the project will increase.

An approach that optimizes the structure based on the shape of the protein determined by X-ray or cryo-electric testing is called Structure Based Drug Design (SBDD). Also, if you know the pocket, you can screen for compounds that physically bind to the pocket, which is called structure-based virtual screening (SBVS), and ligand-based virtual screening as you did in the previous chapter. It may be distinguished from ligand-based virtual screenig(LBVS).

History of Xa inhibitors and the importance of quantum chemistry calculation

Although the contents of the chemoinformatics in this book are far apart, it is very useful in molecular design to trace the history of FXa inhibitors and to understand what improvements have been made through generations. In addition, since the interpretation of the S1 pocket interaction is very difficult visually and in classical mechanics, it can be interpreted only by quantum chemical calculation such as Fragment Molecular Orbital Method (FMO), so it is a mistake that quantum chemical calculation becomes essential in future molecular design I think.

Chapter 7: Assessing similarity using graph structures

A graph is data consisting of nodes (vertices) and edges (branches) that indicate the connection between nodes. The chemical structure can be represented by this graph. In other words, we can represent atoms in a graph with nodes and bonds as edges.

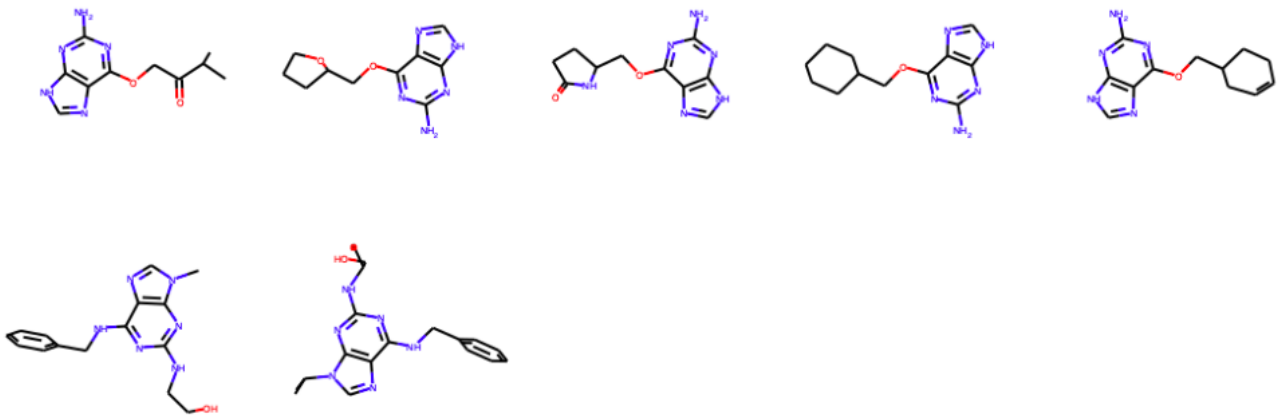
In general, fingerprints like those introduced in Chapter 6 are often used to evaluate the similarity between molecules, but there is also a method to evaluate similarity using a graph structure. The MCS (Maximum Common Substructure) introduced next refers to the common substructure of the target molecule set. The more common substructures, the more similar their molecules are.

Classification by major skeleton (MCS)

Maximum Common Substructure (MCS) is the largest common substructure in a given group of chemical structures. RDKit provides a module called rdFMCS for MCS search.

This time, we will use the file cdk2.sdf provided in rdkit as sample data for MCS search. RDConfig.RDDocsDir is a variable that represents the directory of sample data, and there is a file called cdk2.sdf under the Books/data/ directory, so set the file path with the os.path.join method. Note that os.path.join is a python built-in module for absorbing differences in os paths.

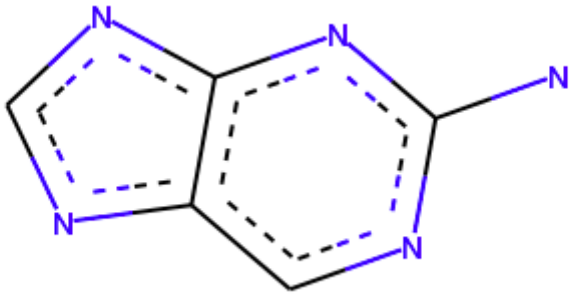
```
import os
from rdkit import Chem
from rdkit.Chem import RDConfig
from rdkit.Chem import rdFMCS
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
filepath = os.path.join(RDConfig.RDDocsDir, 'Book', 'data', 'cdk2.sdf')
mols = [mol for mol in Chem.SDMolSupplier(filepath)]
# 構造を確認します
Draw.MolsToGridImage(mols[:7], molsPerRow=5)
```



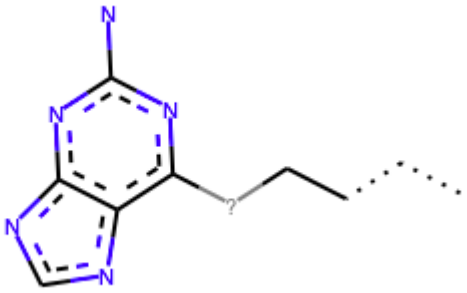
Acquires MCS using the loaded molecule. With RDKit, you can specify multiple options for how to get MCS. The following shows an example of each option.

1. Default
2. Any atom can be used (as long as there is an order of structure and bond)
3. The bond order may be any (for example, benzene and cyclohexane have the same MCS)

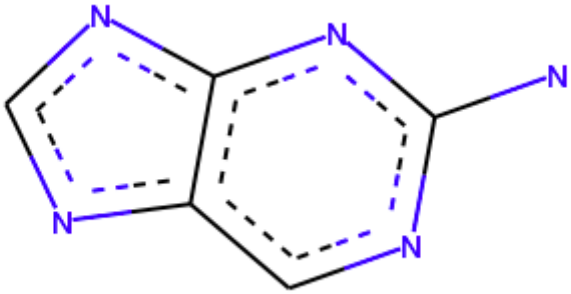
```
result1 = rdFMCS.FindMCS(mols[:7])
mcs1 = Chem.MolFromSmarts  (result1.smartsString)
mcs1
print(result1.smartsString)
#[#6]1:[#7]:[#6](:[#7]:[#6]2:[#6]:1:[#7]:[#6]:[#7]:2)-[#7]
```



```
result2 = rdFMCS.FindMCS(mols[:7], atomCompare=rdFMCS.AtomCompare.CompareAny)
mcs2 = Chem.MolFromSmarts(result2.smartsString)
mcs2
print(result2.smartsString)
#[#6]-,:[#6]-,:[#6]-[#6]-[#8,#7]-[#6]1:[#7]:[#6](:[#7]:[#6]2:[#6]:1:[#7]:[#6]:[#7]:2)-
[#7]
```

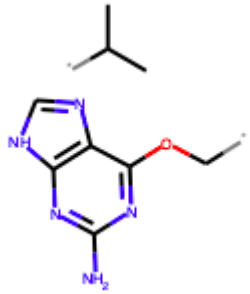


```
result3 = rdFMCS.FindMCS(mols[:7], bondCompare=rdFMCS.BondCompare.CompareAny)
mcs3 = Chem.MolFromSmarts(result3.smartsString)
mcs3
print(result3.smartsString)
#[#6]1:[#7]:[#6](:[#7]:[#6]2:[#6]:1:[#7]:[#6]:[#7]:2)-[#7]
```



In RDKit, Fraggle Similarity is implemented as an algorithm to quantify similarity based on MCS. By using this, clustering and analysis based on similarity can be performed.

```
from rdkit.Chem.Fraggle import FraggleSim
sim, match = FraggleSim.GetFraggleSimilarity(mols[0], mols[1])
print(sim, match)
#0.925764192139738 *C(C)C.*COc1nc(N)nc2[nH]cnc12
match_st = Chem.MolFromSmiles(match)
match_st
```



Thus, FraggleSimilarity returns similarities and matched substructures. It is often closer to the feeling of a chemist than the similarity using ECFP. Please refer to the reference link for details.

Reference link

- Efficient Heuristics for Maximum Common Substructure Search

- Fraggle – A new similarity searching algorithm

Matched Molecular Pair and Matched Molecular Series

At the structural optimization stage of drug discovery research, how to convert the starting compound (lead compound) is very important, but as the stage progresses, which structural conversion affects the activity and physical properties It is also very important to carry out a retrospective analysis of what it has exerted.

TIP If you are interested, you may read https://sar.pharm.or.jp/wp-content/uploads/2018/09/SARNews_19.pdf.

Matched Molecular Pair (MMP) is a pair of molecules that differ only in the partial structure of some of the two molecules but are otherwise identical. As an example, chlorobenzene and fluorobenzene are MMPs because they differ only in Cl and F groups. By analyzing a large number of changes in the characteristics of such pairs, you can grasp the trend of substituent conversion. This is called Matched Molecular Pair Analyisis (MMPA). By performing MMPA on large-scale data, it is possible to extract universal rules for property changes caused by substituent changes. If you understand these rules, you will be able to proceed efficiently with structural optimization.

Here, we analyze MMP using RDKit/Contrib/MMPA[mmpa] provided in Contrib of RDKit.

Move to Contrib/mmpa under the RDKit installation location and execute the python script sequentially.

```
python rfrag.py <name of the File you want to implement the MMPA> #save file name of
the data that was fragmented
# For example
# python rfrag.py <data/sample.smi >data/sample_fragmented.txt

python indexing.py <can be in the previous command Fragment file > MMP_ output file
.CSV
# eg
# python index.py <data/sample_fragmented.txt >data/mmp.csv
```

Executing the above command will generate a csv file of molecule A, molecule B, ID of molecule A, ID of molecule B, SMIRKS of converted structure, and common part structure (context). MMPA can be performed by linking activity and physical properties based on this data.

NOTE SMIRKS is a method to express conversion of molecules by string notation like SMILES.

A method called Matched Molecular Series (MMS) has also been proposed as an extension of MMP. Although MMP is a pair of molecules, MMS is a list of this pair as a group of three or more with

common structure.

I will actually make MMS. The following example uses data from Factor Xa in ChEMBL. For the implementation of MMS, we use the code of the presentation by Noel O'Byle's RDKit UGM.

Let's actually make an MMS. In the following example, Factor Xa data was downloaded from ChEMBL and used as an example. For the implementation of MMS, we use the code of the presentation by Noel O'Byle's RDKit UGM .

First, loading the library to be used, loading the data, and desalting using SaltRemover.

```
import sys
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import rdMMPA
from rdkit.Chem import RDConfig
from rdkit.Chem import rdBase
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from rdkit.Chem import SaltRemover
mmpapath = os.path.join(RDConfig.RDContribDir, 'mmpa')
sys.path.append(mmpapath)
df = pd.read_csv('Chembl_FXa.txt', sep='\t')
remover = SaltRemover.SaltRemover()
mols = []
for i, smi in enumerate(df.CANONICAL_SMILES):
    try:
        mol = Chem.MolFromSmiles(smi)
        mol.SetProp('CMPD_CHEMBLID', df.CMPD_CHEMBLID[i])
        mol = remover.StripMol(mol)
        mols.append(mol)
    except:
        print(smi)
```

Then, import the mmpa rfrag registered in RDKit contrib, and divide the molecule into fragments.

```
import rfrag
rfragdata = []
for i, smi in enumerate(df.CANONICAL_SMILES):
    try:
        out = rfrag.fragment_mol(smi, df.CMPD_CHEMBLID[i])
        rfragdata.append(out)
    except:
        print(smi, df.CMPD_CHEMBLID[i])
```

Define a function to create an MMS. The code is almost the same as that described in the UGM document, but I changed the reading destination from a file to a list in order to do all processing on Jupyter.

Here is an overview of the MMS creation process.

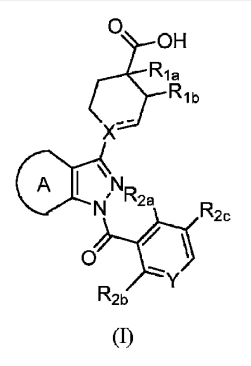
1. Cut each molecule according to a certain rule (cut by rotatable bond etc.)
2. Cut fragments create a dictionary of keys, store the fragments of molecules with the same key in the dictionary value

By repeating the above process, molecules with common scaffold can be organized. Molecules that are grouped in a common scaffold will be molecules that have different non-scaffold substituents.

What is a scaffold?

In drug discovery, there is a stage of structural optimization at the stage before preclinical studies, in which the major non-skeleton part of the compound is converted briefly into a balanced property suitable for drugs.

This main skeleton is called a scaffold. For example, in this patent, the part except R is fixed and this main skeleton is called a scaffold.

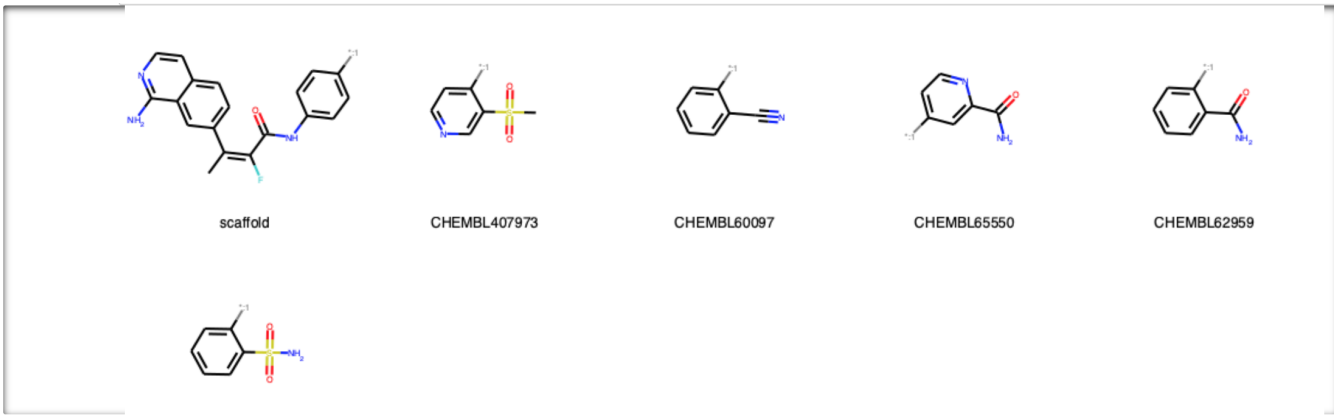


(I)

```
from collections import namedtuple

Frag = namedtuple( 'Frag', ['id', 'scaffold', 'rgroup'] )

class Series():
    def __init__( self ):
        self.rgroups = []
        self.scaffold = ""

def getFrags(rfrags):
    frags = []
    for lines in rfrags:
        for line in lines:
            broken = line.rstrip().split(",")
            if broken[2]: # single cut
                continue
            smiles = broken[-1].split(".")
            mols = [Chem.MolFromSmiles( smi ) for smi in smiles]
            numAtoms = [mol.GetNumAtoms() for mol in mols]
            if len(numAtoms) < 2:
                continue
            if numAtoms[0] > 5 and numAtoms[1] < 12:
                frags.append(Frag(broken[1], smiles[0], smiles[1]))
            if numAtoms[1] > 5 and numAtoms[0] < 12:
                frags.append(Frag(broken[1], smiles[1], smiles[0]))
    frags.sort(key=lambda x:(x.scaffold, x.rgroup))
    return frags

def getSeries(frags):
    oldfrag = Frag(None, None, None)
    series = Series()
    for frag in frags:
        if frag.scaffold != oldfrag.scaffold:
            if len(series.rgroups) >= 2:
                series.scaffold = oldfrag.scaffold
                yield series
            series = Series()
        series.rgroups.append((frag.rgroup, frag.id))
        oldfrag = frag
    if len(series.rgroups) >= 2:
        series.scaffold = oldfrag.scaffold
        yield series
```

We are ready to make an MMS. Visualize only data that has four or more substituent conversions for the same scaffold.

```
frags = getFrags(rfragdata)
series = getSeries(frags)
series =[i for i in series]
from IPython.display import display
for s in series[:50]:
    mols = [Chem.MolFromSmiles(s.scaffold)]
    ids = ['scaffold']
    for r in s.rgroups:
        rg = Chem.MolFromSmiles(r[0])
        mols.append(rg)
        ids.append(r[1])
    if len(mols) > 5:
        display(Draw.MolsToGridImage(mols, molsPerRow=5, legends=ids))
        print("#########")
```



Five scaffolds for MMS were displayed for the scaffold.

NOTE Activity prediction can also be performed using this MMS.

Visualize MMP networks using Cytoscape

WARNING This content is beyond the content of the introductory, so please skip if you are not interested.

MMP can be thought of as a graph structure that uses pre-conversion and post-conversion information as nodes and conversion rules as edges. This graph structure can be intuitively understood by using network visualization tools such as Cytoscape.

In addition to the MMPA introduced earlier, RDKit has another project called mmpdb. It is provided as a command line tool group and database system, so it has the feature of being easy to manage in the long run. In this section, we introduce the visualization of MMP using mmpdb and Cytoscape.

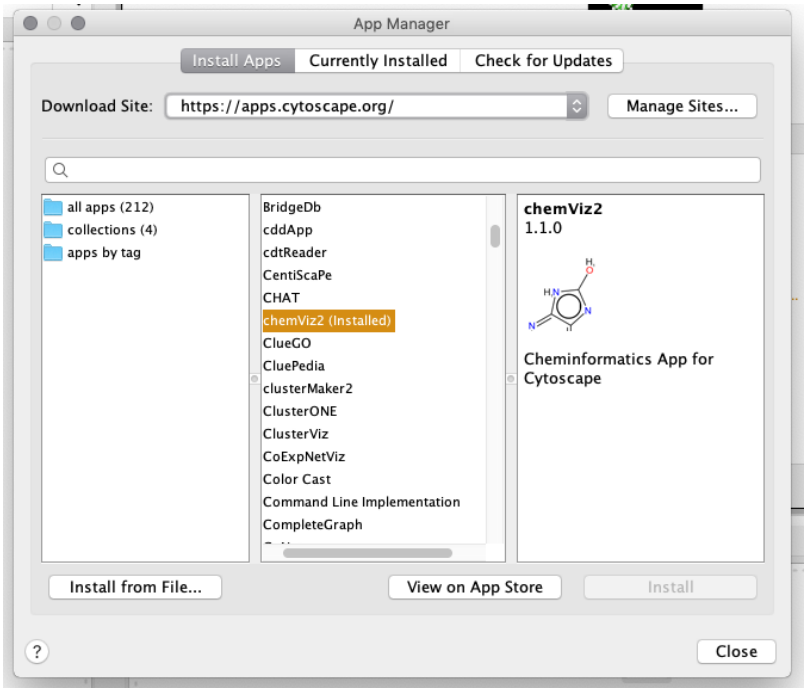
NOTE mmpdb: An Open Source Matched Molecular Pair Platform for Large Multi-Property Datasets

Cytoscape installation

Cytoscape is an open source network visualization software widely used in various scenes. You can display the structure network by using the compound structure display plug-in.

Installation is as easy as downloading the corresponding OS installer from the download site and installing according to the instructions.

When installation is complete, launch Cytoscape and install the Chemviz2 plug-in for drawing compound structures. The procedure is easy, select chemviz2 from Apps → App Manager and install it.



create a gml file from mmpdb

The data to be used this time are 151 compounds of <Inhibition of recombinant GSK3-beta> J. Med. Chem. (2008) 51: 2062-2077 . In principle, MMPA does not use HTS-like search data but scaffolds such as structure optimization.

I will put the flow of the command. SMILES text and activity and property data need to be registered separately in the database.

```
$ mmpdb fragment smiles.txt -o CHEMBL930273.fragments     # fragmentation
$ mmpdb index CHEMBL930273.fragments -o CHEMBL930273.db   # make db
$ mmpdb loadprops -p act.txt CHEMBL930273.db              # load properties
```

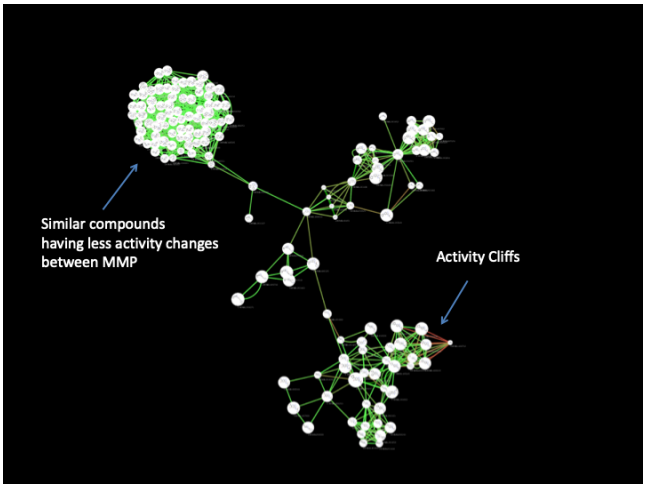
After that we will create a gml file for reading by Cytoscape, but this is beyond the scope of this document and will be omitted. If you are interested, you may want to read the code directly, but the flow is as follows.

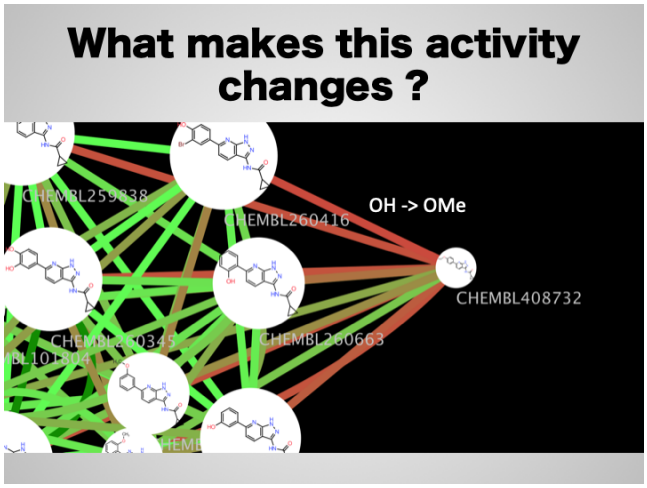
1. Make a gml file using mmpdb and python-igraph
2. Read gml file by Cytoscape

3. Assign attributes to each parameter in Cytoscape to make it easier to understand visually
    a. Corresponds to the physical value of the node size
    b. Corresponds to the active color of the edge color
    c. Draw a structure with chemviz2 plugin and paste it to a node

Interpretation

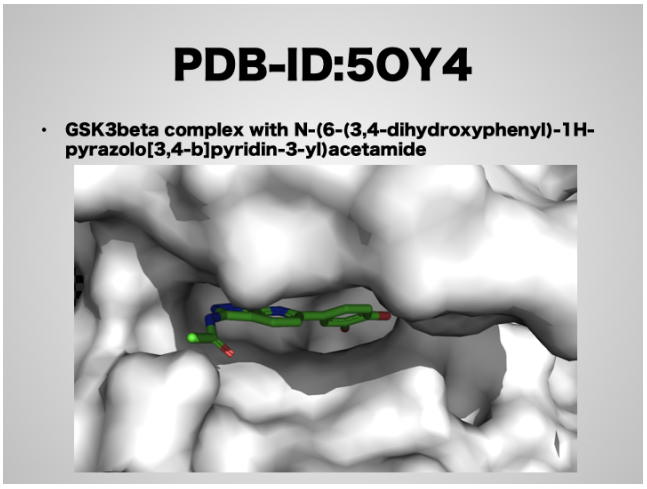
Let's look at the MMP network. MMP with little difference in activity is solidified in the upper left. In the lower right, red edges (a large difference in activity) are observed. MMPs are also called Activity Cliffs, even if such small substituent changes produce large activity differences. It is important not to overlook such changes in activity, as Activity Cliff is generally a breakthrough in drug discovery projects.

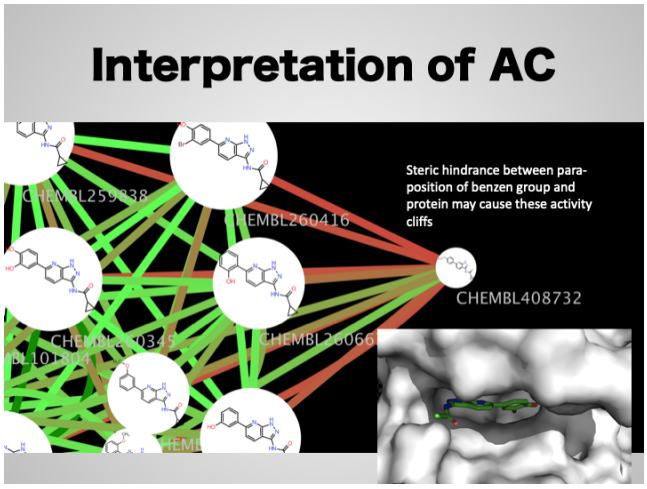




It has been found that the substitution of the OH group with the MeO group causes the loss of activity when we actually confirm what substitution has been made.

Since MMP alone can simply know the facts like this, I searched for a complex crystal structure of the analogue in order to consider it a little deeper. Then , a complex of GDB3β and a similar compound was found as PDBID:5OY4.





If you replace the OH group with the MeO group, it will likely hit the wall of the pocket. In other words, this Activity Cliff is considered to be caused by steric hindrance of ligand and protein.

Chapter 8: Want to have lots of compounds at once

In order to see how much data is distributed, it is common to map in an appropriate space. Especially in chemoinformatics the word chemical space is used.

What is Chemical Space

Chemical space refers to the arrangement of compounds in an n-dimensional space at some scale. In general, two or three dimensions are often used (for human understanding). Although various methods have been proposed for the scale, ie, similarity, it is often decided that a distance that well characterizes a compound is defined.

This time, we will visualize which pharmaceutical company is developing what kind of compound for the antagonist of Orexin Receptor, which is known as a target for sleep medicine. See Chapter 4 for how to download data. This time we used the data of 10 papers in the table.

There are two main things I want to know this time:

- Were there companies that developed similar compounds?
- Has Merck optimized only similar frameworks, or did it optimize multiple frameworks?

Table 1. Orexin Receptor Antagonist

Doc ID	Journal	Pharma
CHEMBL3098111	Bioorg. Med. Chem. Lett. (2013) 23:6620-6624	Merck
CHEMBL3867477	Bioorg Med Chem Lett (2016) 26:5809-5814	Merck
CHEMBL2380240	Bioorg. Med. Chem. Lett. (2013) 23:2653-2658	Rottapharm
CHEMBL3352684	Bioorg. Med. Chem. Lett. (2014) 24:4884-4890	Merck
CHEMBL3769367	J. Med. Chem. (2016) 59:504-530	Merck
CHEMBL3526050	Drug Metab. Dispos. (2013) 41:1046-1059	Actelion
CHEMBL3112474	Bioorg. Med. Chem. Lett. (2014) 24:1201-1208	Actelion
CHEMBL3739366	MedChemComm (2015) 6:947-955	Heptares
CHEMBL3739395	MedChemComm (2015) 6:1054-1064	Actelion

CHEMBL3351489	Bioorg. Med. Chem. (2014) 22:6071-6088	Eisai

Mapping using Euclidean distance

Use ggplot for the drawing library. Principal component analysis (PCA) is used to distribute and visualize similar compounds close together. At first we import necessary library

```
from rdkit import Chem, DataStructs
from rdkit.Chem import AllChem, Draw
import numpy as np
import pandas as pd
from ggplot import *
from sklearn.decomposition import PCA
import os
```

Load the downloaded sdf, and create fingerprints for each compound, enabling correspondence between drug companies and document IDs. If you have any questions please check Chapter 6.

```
oxrs = [("CHEMBL3098111", "Merck" ),("CHEMBL3867477", "Merck" ),
          ("CHEMBL2380240", "Rottapharm" ),("CHEMBL3352684", "Merck" ),
          ("CHEMBL3769367", "Merck" ),("CHEMBL3526050", "Actelion" ),
          ("CHEMBL3112474", "Actelion" ),("CHEMBL3739366", "Heptares" ),
          ("CHEMBL3739395", "Actelion" ), ("CHEMBL3351489", "Eisai" )]

fps = []
docs = []
companies = []

for cid, company in oxrs:
    sdf_file = os.path.join("ch08", cid + ".sdf")
    mols = Chem.SDMolSupplier(sdf_file)
    for mol in mols:
        if mol is not None:
            fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2)
            arr = np.zeros((1,))
            DataStructs.ConvertToNumpyArray(fp, arr)
            docs.append(cid)
            companies.append(company)
            fps.append(arr)
fps = np.array(fps)
companies = np.array(companies)
docs = np.array(docs)
```

If you check the information of the fingerprint, you can see that data of 293 compounds are obtained from 10 articles.

```
fps.shape
# (293, 2048)
```

You are now ready for principal component analysis. The number of principal components can be specified by n_components, but this time, I want to scatter two dimensions, so I set it to 2.

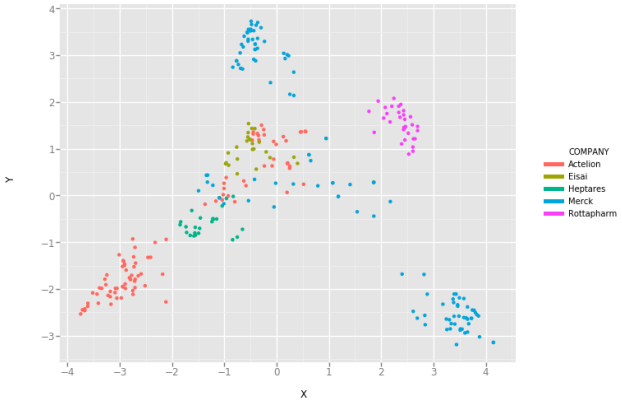
```
pca = PCA(n_components=2)
x = pca.fit_transform(fps)
```

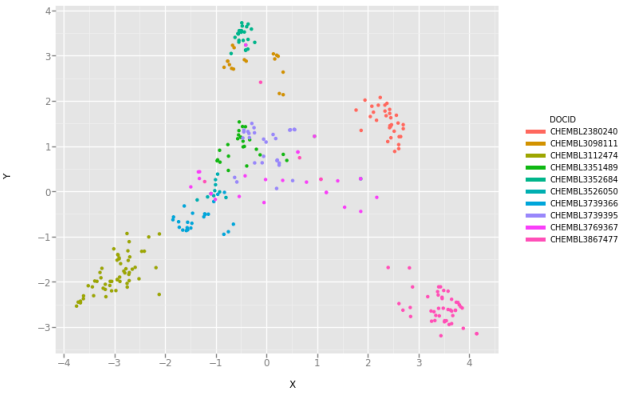
Draw. I changed the color option according to each label, so I chose two attributes, COMPANY and DOCID.

```
d = pd.DataFrame(x)
d.columns = ["PCA1", "PCA2"]
d["DOCID"] = docs
d["COMPANY"] = companies
g = ggplot(aes(x="PCA1", y="PCA2", color="COMPANY"), data=d) + geom_point() + xlab("X
") + ylab("Y")
g
```

You can now see what compounds each pharmaceutical company has optimized. Merck, Acterion, Eisai and Heptaress seem to have optimized similar compounds, as there is an overlapping area in the center of the chemical space. It is interesting to see whether the Acterion has been successfully deployed in a unique direction (lower left) or has not been deployed and has advanced into the red ocean center.

Also, Merck seems to have optimized various frameworks. I don't know if I'm optimizing at the same time or running ahead for backup, but it's no doubt that there were a lot of skeletal optimizations running, so it's probably an attractive target. In fact, SUVOREXANT was launched.





patinformatics

In this chapter, we use dissertation data, but we do not use dissertation data when performing such analysis in a real field. Because when a company disseminates, it means that the project is over (whether it went to clinical or failed and closed). In the actual situation, analysis is performed using patent data.

Based on the analysis and experience of Medicinal Chemist and the insights of these companies, the project will proceed with a belief in their own successes while inferring the situation of other companies.

Mapping using tSNE

It is said that tSNE has better resolution than PCA and is closer to the sense of medicinal chemist. Sklearn just changes PCA to TSNE.

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=0)
tx = tsne.fit_transform(fps)
```

As you can see when drawing, it is separated better than PCA.

```
d = pd.DataFrame(tx)
d.columns = ["PCA1", "PCA2"]
d["DOCID"] = docs
d["COMPANY"] = companies
g = ggplot(aes(x="PCA1", y="PCA2", color="COMPANY"), data=d) + geom_point() + xlab("X
") + ylab("Y")
g
```

There are many other drawing methods besides PCA and tSNE introduced this time, so it is good to check.

Chapter 9: Basics of Quantitative Structure-Activity Relationship (QSAR)

The correlation between chemical structure and biological activity is called Structure Activity Relationship (SAR) or Quantitative SAR (QSAR). In general, **similar compounds** are known to exhibit **similar biological activities**, and it is very important in drug discovery research to understand this correlation and apply it to drug design.

In addition, there are two types of problems such as classification problems to estimate which class a compound belongs to, such as cell death or toxicity, or toxicity, and regression problems to estimate continuous values such as % inhibition.

Consider the cause of no effect (classification problem)

Label the ones with an IC50 less than 1 uM with hERG inhibition and the others with no hERG inhibition using 73 data from ChEMBL hERG inhibition assay.

First, import the necessary libraries.

```
from rdkit import Chem, DataStructs
from rdkit.Chem import AllChem, Draw
from rdkit.Chem.Draw import IPythonConsole
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.ensemble import RandomForestClassifier
```

Processing of tab-delimited text downloaded with ChEMBL is almost the same as in Chapter 8, but this time I want liveness data, so I search for the column **STANDARD_VALUE** and retrieve the numerical value. If this value is less than 1000 nM, label it as POS, otherwise label it as NEG. At the end, I will make the label numpy array.

```
mols = []
labels = []
with open("ch09_compounds.txt") as f:
    header = f.readline()
    smiles_index = -1
    for i, title in enumerate(header.split("\t")):
        if title == "CANONICAL_SMILES":
            smiles_index = i
        elif title == "STANDARD_VALUE":
            value_index = i
    for l in f:
        ls = l.split("\t")
        mol = Chem.MolFromSmiles(ls[smiles_index])
        mols.append(mol)
        val = float(ls[value_index])
        if val < 1000:
            labels.append("POS")
        else:
            labels.append("NEG")

labels = np.array(labels)
```

Then convert the mol object into a fingerprint. From this fingerprint, create a model to predict the presence or absence of hERG inhibition.

```
fps = []
for mol in mols:
    fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2)
    arr = np.zeros((1,))
    DataStructs.ConvertToNumpyArray(fp, arr)
    fps.append(arr)
fps = np.array(fps)
```

Divide the data set into two of the training set test set. The test set will be used later to evaluate the accuracy of the created prediction model.

```
x_train, x_test, y_train, y_test = train_test_split(fps, labels)
```

To create a predictive model, just create an instance and train it with the fit method.

```
rf = RandomForestClassifier()
rf.fit(x_train, y_train)
```

Predict the test set you split up earlier.

```
y_pred = rf.predict(x_test)
```

Create a Confusion matrix.

What is confusion matrix?

Confusion matrix is a table that summarizes the results of class classification. It is possible to visualize clearly whether the class is classified correctly, and as TP and TN are many and FP and FN are small, it is possible to classify better.

		Actual class	
		Positive	Negative
Predicted class	Positive	True Positive(TP)	False Positive(FP)
	Negative	False Negative(FN)	True Negative(TN)

```
confusion_matrix(y_test, y_pred)
#array([[11,  1],[ 5,  2]])
```

11	**1**
5	2

Let's look at the F1 score.

```
f1_score(y_test, y_pred, pos_label="POS")
#0.4
```

It is not very good.

NOTE Because the train_test_split function randomly splits the training set and test set, the value of the confidence matrix, F1 score changes with each execution.

With F1 score

- The ratio of what is truly correct among what is predicted to be correct is called accuracy rate precision = TP / (TP + FP)
- The rate at which the correct thing is predicted to be correct is called the recall rate recall = TP / (TP + FN)

The F1 score is the harmonic mean of the precision rate and the recall rate

It is calculated by F1 = 2 * (precision * recall) / (precision + recall)

Predict the efficacy of drugs (regression problem)

Regression models, as discussed earlier, are models that predict continuous values. This time, create a regression model of RandomForest, and evaluate its accuracy with R2. Let's use the data from hERG's assay data used in classification problems. Import the required libraries first.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from math import log10
```

We labeled it for classification problems, but now we want to predict continuous values, so we convert it to pIC50. (We will supplement later on why it is convenient to use pIC50)

```
pIC50s = []
with open("ch09_compounds.txt") as f:
    header = f.readline()
    for i, title in enumerate(header.split("\t")):
        if title == "STANDARD_VALUE":
            value_index = i
    for l in f:
        ls = l.split("\t")
        val = float(ls[value_index])
        pIC50 = 9 - log10(val)
        pIC50s.append(pIC50)

pIC50s = np.array(pIC50s)
```

Divide the data set into two: training set and test set. The fingerprint uses what was created at the time of classification model.

```
x_train, x_test, y_train, y_test = train_test_split(fps, pIC50s)
```

I will train. In the case of Scikit-learn, this procedure is fit and predict with almost the same method in any method.

```
rf = RandomForestRegressor()
rf.fit(x_train, y_train)
```

Let's predict.

```
y_pred = rf.predict(x_test)
```

Let's put out the prediction accuracy with R2.

```
r2_score(y_test, y_pred)
#0.52
```

Is there anything like that?

With R2 score

It is often used as one of the evaluation indicators for the goodness of fit of regression, also called the determination coefficient.

Model applicability (applicability domain)

The method introduced here is a model generated based on the hypothesis that **similar compounds exhibit similar biological activities**. What is the prediction accuracy if there is no compound that is similar to the training set?

Of course, the predicted value is not reliable in that case. In other words, is the prediction likely to be that prediction? The degree of reliability always goes around. The extent to which such models can be trusted or applied is called the applicability domain. In this regard, the scope of application and model application Mr. Kaneko, Meiji University are detailed.

(Extra column) How reliable can the applicability domain be?

Long time ago, do the similar compounds of Dr. Hugo Kubinyi show similar activities? I remember the question that I was impressed by the fact that converting the estradiol OH group into a Methoxy group gave examples of the loss of activity.

The applicability domain is a method to measure the accuracy of the prediction from the similarity of the training set. Here comes the question of who the similarity is for. It is our hand that we think this compound and this compound are similar, but it is ultimately determined by the protein whether it is similar or not. Therefore, the activity can not always be predicted from the similarity, and the activity often disappears even if the similarity is extremely high. In particular, Activity Cliff, described in the context of MMP, gives such an event its name.

Chapter 10: Introduction to Deep Learning

For this chapter, we will use deep learning to create QSAR models and generation models.

About deep learning

Neurons exist in the brain of an organism, and they form a network to transmit information, store and learn. The Artificial Neural Network (ANN) is a mathematical model of this network structure.

In general ANN, an input layer for inputting information for learning, an intermediate layer (or hidden layer) for learning a response (corresponding to the firing of a nerve synapse) based on a pattern of input information, and a third-layer or final output layer. However, deep learning enables highly accurate predictions by layering multiple hidden layers.

Although I will not explain this in this book in particular, if you want to write and understand the code from scratch yourself, Deep Learning from scratch can be helpful. Also, if you want to learn about theory properly, we recommend deep learning.

About TensorFlow and Keras

Tensorflow is a framework for machine learning developed by Google and released as OSS. It is often used mainly as a deep learning framework.

NOTE	Tensorflow has recently made a major update from 1.x to 2.x, but since the 2.x version has just appeared and there is little reference information, it uses the 1.x system. Also, since the API differs depending on the version of the same 1.x, if there is code you want to run, please be careful about which version is written.

Keras is a high-level API backed by a low-level framework such as Tensoflow, so you can write code more easily. Keras has been developed independently of Tensorflow, but recently Tensorflow comes with Keras. So you can use Keras without installing separately. The Tensorflow bundled version of Keras may not be the latest version of the home.

It’s annoying to decide which Keras to use, but for the sake of convenience, we will use Tensorflow-integrated Keras.

Relationship between Keras and Tensorflow

I will organize Keras and Tensoflow a little while referring to the official blog. Originally Keras was developed as a separate project from Tensoflow (and, of course, still), to use Keras, Tensorflow had to be installed. However, around the timing of the major version upgrade of Keras 2.x 2017, the Tensoflow project has integrated Keras. The English below is an excerpt of the above link. It is now possible to call Keras from Tensorflow.

TensorFlow integration Although Keras has supported TensorFlow as a runtime backend since December 2015, the Keras API had so far been kept separate from the TensorFlow codebase. This is changing: the Keras API will now become available directly as part of TensorFlow, starting with TensorFlow 1.2. This is a big step towards making TensorFlow accessible to its next million users.

Let's install

Let's install Tensorflow and Keras. When installing with anaconda, the package to be installed differs slightly depending on whether you use a GPU compatible version or a CPU version.

```
# CPU only
$ conda install -c conda-forge tensorflow
# GPU enabled
$ conda install -c anaconda tensorflow-gpu
```

NOTE	You can also use the pip command to install TensorFlow. In that case , please refer to the official document. But basically, if you make an environment with Conda, it is desirable to put a package with Conda.

Reference link

- https://keras.io/#installation
- https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html

About Google colab

Google colab

Google colaboratory is a Jupyter notebook environment that can be run on the cloud. The framework for deep learning such as Theano, Thensorflow, Keras, Pytorch is already installed and time is limited, but because GPU can be used, it is very attractive that deep learning can be used without a GPU machine at hand is.

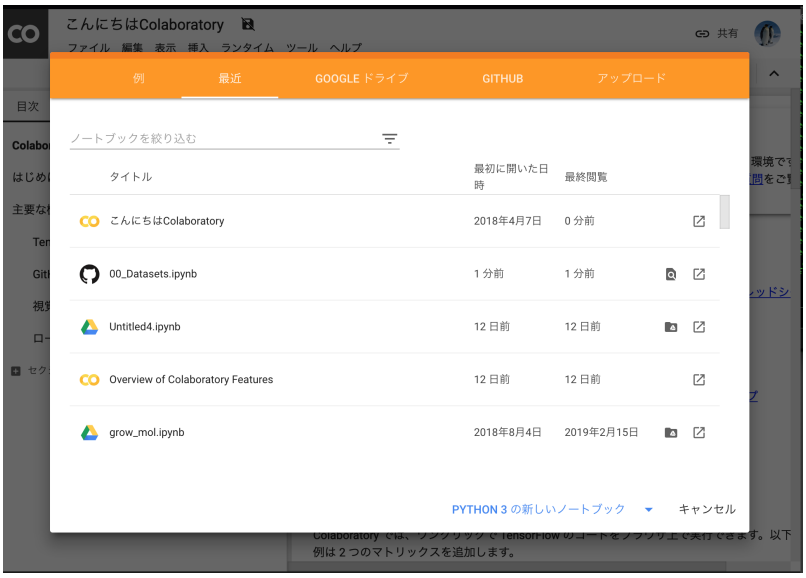
You need a Google account to use it, so if you do not have a Google account, it is a good idea to get an account at this opportunity. If you have a Google account, you can also run GitHub-style

notebooks directly on Colab. Let's open the Scikit-learn hands-on notebook previously used in Mishima.syk.

NOTE

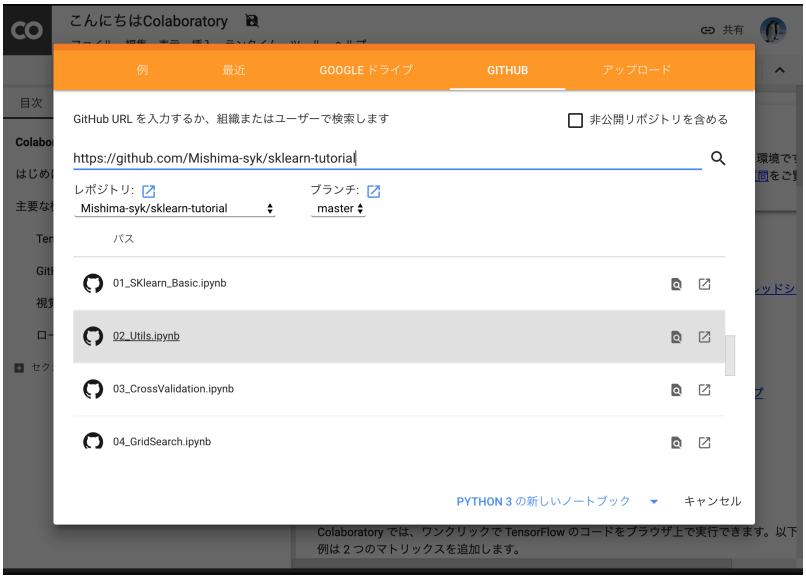
It is a notebook created by @y__sama, but it is possible to learn from Auto data preparation to AutoSklearn.

First of all, go to Google colaboratory. If you do not get the screen below, please execute "Open Notebook" from "File" on the top left

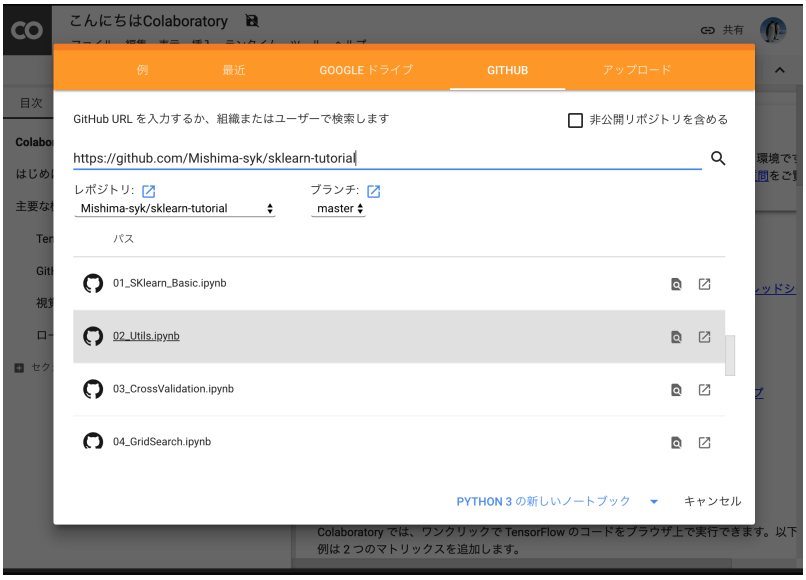


Next, click the tab named GitHub, copy and paste the following URL, and you can move the code from Jupyter Notebook.

https://github.com/Mishima-syk/sklearn-tutorial



When you open the notebook, you will see a screen similar to the Jupyter Notebook. You can execute the code of the cell with Shift + Return key.



To see the libraries available by default in Google Colab, type '! Pip freeze' in the cell and it will be listed.

- absl-py==0.7.0
- alabaster==0.7.12
- snip ;)
- yellowbrick==0.9.1
- zict==0.1.3
- zmq==0.0.0

Python deep learning framework

There are many Python deep learning frameworks. Mainly Theano, Tensorflow, Keras, MXNet, Chainer, PyTorch, etc.

Various deep learning documents often use one of the above frameworks for implementation. You may want to try it and choose a framework that is easy to use.

Chapter 11: Structure-activity relationship using deep learning

In this chapter, structure activity correlation analysis is performed using DNN.

Predictive model construction using DNN

First, let's build a simple prediction model using DNN. Here we use the same data as in Chapter 9. First, create a classification model and label the Positive label as [0, 1] and the Negative label as a [1, 0] two-dimensional OneHot vector. If you create a model using Keras Model object, you can get the expected value of each of the above two dimensions. You can use Numpy's Argmax function to know which class it is likely to belong to.

NOTE	OneHot vector is a vector in which one value is 1 and the other is 0. When considering a classification problem of 10 classes, such as [1, 0, 0, 0, 0, 0, 0, 0, 0, 0], a vector such that somewhere is 1 and the remaining 9 are 0 I can express a class. In the above example, there are two classes of Positive / Negative, so the OneHot vector is two-dimensional.

Import the required libraries.

```
from rdkit import Chem, DataStructs
from rdkit.Chem import AllChem, Draw
from rdkit.Chem.Draw import IPythonConsole
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, f1_score
from thensorflow.python.keras.layers import Iput
from thensorflow.python.keras.layers import Dense
from thensorflow.python.keras.layers import Dropout
from thensorflow.python.keras.layers import Activation
from thensorflow.python.keras.Model import Model
```

Next, read the data. In Chapter 9 we put "POS" / "NEG" in the list of labels, so it was a one-dimensional representation, but this time it is two-dimensional.

```
mols = []
labels = []
with open("ch09_compounds.txt") as f:
    header = f.readline()
    smiles_index = -1
    for i, title in enumerate(header.split("\t")):
        if title == "CANONICAL_SMILES":
            smiles_index = i
        elif title == "STANDARD_VALUE":
            value_index = i
    for l in f:
        ls = l.split("\t")
        mol = Chem.MolFromSmiles(ls[smiles_index])
        mols.append(mol)
        val = float(ls[value_index])
        if val < 1000:
            labels.append([0,1]) # Positive
        else:
            labels.append([1,0]) # Negative
labels = np.array(labels)
```

Next, create classification models and regression models sequentially.

The first is a regression model, and the input uses the same ECFP as in Chapter 9. In order to construct DNN, it is necessary to specify the dimension of input data explicitly, so we define the variable nBits.

TIP Specifying an appropriate integer in random_state for train_test_split is useful for verification because the same data is obtained each time.

```
nBits = 2048
fps = []
for mol in mols:
    fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=nBits)
    arr = np.zeros((1,))
    DataStructs.ConvertToNumpyArray(fp, arr)
    fps.append(arr)
fps = np.array(fps)

x_train1, x_test1, y_train1, y_test1 = train_test_split(fps, labels, random_state=794)
```

Create a neural network whose inputs are 2048 dimensions, the total connection layer of 300 neurons is three layers, and the final output layer is two. We used ReLU for the activation function and Softmax for two-dimensional multiclass classification for the output layer.

The Dropout layer plays a role to prevent overlearning by randomly deleting neurons.

Keras constructs a model by defining the model and then calling the compile function. Although

optimizer and loss need to be changed according to the purpose, in this case 'categorical_crossentropy' was used, but there are many other than adam optimizer , so it will actually require trial and error which is appropriate.

TIP ReLU is often used because it can overcome the problem of gradient disappearance of Sigmoid function.

```
# Define DNN classifier model
epochs = 10
inputlayer1 = Input(shape=(nBits, ))
x1 = Dense(300, activation='relu')(inputlayer1)
x1 = Dropout(0.2)(x1)
x1 = Dense(300, activation='relu')(x1)
x1 = Dropout(0.2)(x1)
x1 = Dense(300, activation='relu')(x1)
output1 = Dense(2, activation='softmax')(x1)
model1 = Model(inputs=[inputlayer1], outputs=[output1])

model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'
])
```

NOTE: Keras provides a Sequential model, which can be used to describe the network more simply than the example above (Functional API). The reason we defined the model in the Functional API is that it is easy to handle multiple inputs and more complex models if you get used to it. If you are interested in writing Sequential please check out the official site and Qiita

NOTE DNN optimizes the model while iterating the Backpropagation procedure, which compares the actual value with the predicted value predicted based on the initial randomly generated weight, and updates the weight so as to minimize the difference (LOSS). You It is Epochs that specifies the number of repetitions. You may seem to get smarter as you increase Epochs, but there is a risk of computational cost and over-learning, so it is not good if it is long. Observe Loss, Accuracy, etc. and find the appropriate number of Epochs.

Why is there a risk of overlearning when increasing Epochs?

Using training data, we will adjust the weight to reduce the error between the correct value and the predicted value for each Epoch. If it is learned using a sufficient amount of training data and it is repeated too much, the generalization performance of the model will be reduced since the same training data will be learned over and over again.

To judge overtraining, if you evaluate and plot the accuracy of Training set / Validation set for each Epoch, you can check whether the accuracy of Validation set does not change or deteriorate while accuracy of Training set improves. Keras has a function called Eary stopping, which allows you to stop learning if the performance of the model does not change even if you have learned a certain number of times.

See the introduction and references of Early stopping for more information.

After building the model, you can do fit / predict in the same way as Scikit-learn.

```
hist1 = model1.fit(x_train1, y_train1, epochs=epochs)
```

Finally, let's visualize the result.

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(range(epochs), hist1.history['acc'], label='acc')
plt.legend()
plt.plot(range(epochs), hist1.history['loss'], label='loss')
plt.legend()
```

In this example, the model has good accuracy around 6Epoch.

Next, verify with test data.

```
y_pred1 = model1.predict(x_test1)
y_pred_cls1 = np.argmax(y_pred1, axis=1)
y_test_cls1 =np.argmax(y_test1, axis=1)
confusion_matrix(y_test_cls1, y_pred_cls1)
```

A little subtle ,,,,

The regression model is basically the same as the classification problem above. This time it is a regression, so the last output layer is the value itself, ie one dimensional The activation function is 0-1 in Sigmoid etc., so it is Linear. The training data uses the code of Chapter 9.

```
from math import log10
from sklearn.metrics import r2_score
pIC50s = []
with open("ch09_compounds.txt") as f:
    header = f.readline()
    for i, title in enumerate(header.split("\t")):
        if title == "STANDARD_VALUE":
            value_index = i
    for l in f:
        ls = l.split("\t")
        val = float(ls[value_index])
        pIC50 = 9 - log10(val)
        pIC50s.append(pIC50)

pIC50s = np.array(pIC50s)
x_train2, x_test2, y_train2, y_test2 = train_test_split(fps, pIC50s, random_state=794)
```

Next, define the model. Note that the Loss part is MSE, unlike the classification model above.

```
epochs = 50
inputlayer2 = Input(shape=(nBits, ))
x2 = Dense(300, activation='relu')(inputlayer2)
x2 = Dropout(0.2)(x2)
x2 = Dense(300, activation='relu')(x2)
x2 = Dropout(0.2)(x2)
x2 = Dense(300, activation='relu')(x2)
output2 = Dense(1, activation='linear')(x2)
model2 = Model(inputs=[inputlayer2], outputs=[output2])
model2.compile(optimizer='adam', loss='mean_squared_error')
```

If you can do this, the rest is the same.

```
hist = model2.fit(x_train2, y_train2, epochs=epochs)
y_pred2 = model2.predict(x_test2)
r2_score(y_test2, y_pred2)
plt.scatter(y_test2, y_pred2)
plt.xlabel('exp')
plt.ylabel('pred')
plt.plot(np.arange(np.min(y_test2)-0.5, np.max(y_test2)+0.5), np.arange(np.min(
y_test2)-0.5, np.max(y_test2)+0.5))
```

What do you think. The prediction model looks a bit like UnderEstimate. The DNN needs to tune a number of parameters, such as the number of layers to overlap, the percentage of dropouts, the number of neurons in the hidden layer, and the type of activation function. This example was hard-coded, but it is also interesting to compare the performance of the models by changing various parameters.

I will devise a descriptor (neural fingerprint)

So far, we have created models of RandomForest and DNN using molecular fingerprints as input. One of the reasons why DNN has received a great deal of attention is that models can recognize feature quantities even if people do not extract feature quantities.

For example, in image classification, a human defined the feature quantity called SIFT, and a model was created using this as an input, but the current DNN basically uses the pixel information of the image itself.

In terms of chemoinformatics, SIFT is equivalent to a molecular fingerprint. So isn't it possible to improve DNN's performance by changing this (input) to a more primitive expression? It is extremely natural to think that. In 2015, Alan Aspuru-Guzik et al's group at Harvard University proposed the Neural Finger print/NFP as a challenge.

The differences between ECFP and NFP used so far are shown by citing figures in their papers.

Algorithm 1 Circular fingerprints

Input: molecule, radius R, fingerprint length S
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow [\mathbf{r}_a, \mathbf{r}_1, \dots, \mathbf{r}_N]$ ▷ concatenate
    $\mathbf{r}_a \leftarrow \text{hash}(\mathbf{v})$ ▷ hash function
    $i \leftarrow \text{mod}(r_a, S)$ ▷ convert to index
    $\mathbf{f}_i \leftarrow 1$ ▷ Write 1 at index
Return: binary vector $\mathbf{f}$

Algorithm 2 Neural graph fingerprints

Input: molecule, radius R, hidden weights $H_1^1 \dots H_R^5$, output weights $W_1 \dots W_R$
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow \mathbf{r}_a + \sum_{i=1}^{N} \mathbf{r}_i$ ▷ sum
    $\mathbf{r}_a \leftarrow \sigma(\mathbf{v} H_L^N)$ ▷ smooth function
    $\mathbf{i} \leftarrow \text{softmax}(\mathbf{r}_a W_L)$ ▷ sparsify
    $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{i}$ ▷ add to fingerprint
Return: real-valued vector $\mathbf{f}$

Figure 2: Pseudocode of circular fingerprints (*left*) and neural graph fingerprints (*right*). Differences are highlighted in blue. Every non-differentiable operation is replaced with a differentiable analog.

ECFP (Circular Fingerprints) converts information from each atom of input molecules to atoms in the vicinity of N (N is arbitrary) into Hash values (Mod in this example) to arbitrary values, and converts them into vectors of fixed length was. Roughly speaking, it is an image such as using the one where the presence or absence of the partial structure is corrected to the bit information of 0/1. On the other hand, NFP introduced this time is similar in concept to ECFP, but the part of Hash function is Sigmoid, and the part to be discretized with Mod is Softmax. Therefore, it is expected that input datasets will generate molecular fingerprints more flexibly than ECFP.

A number of implementations have been published to GitHub since this paper was published, but each implementation does not work with Keras 1.x or Keras / Tensorflow, even if the Backend is Theano or Keras / Tensorflow There are a lot of environment-dependent things that are surprisingly difficult to handle. Unfortunately there is no one that works in the environment we built this time, so I created one that works with Keras 2.x / Python 3.6 based on this code .

Was it effective to use the classical method with pixel as it is in image classification?

SIFT was proposed in 1999. According to the original paper, the difficulty in dealing with pixels themselves in object (image) recognition seems to be in dealing with objects that differ in position, rotation, size (scale), light intensity, etc. It seems that various methods have been studied to convert these fluctuating values into universal features. There is no way to use the pixels themselves, but the machine learning that I started with python, which I purchased when studying machine learning , has an example of learning and classifying human face image data. Here, with the pixel data as input, the feature of the face is extracted and classified by principal component analysis. I have not been able to find a document that was clearly valid on this question, but I think it was valid depending on the task. Please comment if you have any details.

```
git clone https://github.com/iwatobipen/keras-neural-graph-fingerprint.git
```

If you look at the code in the example.py file, you will find the atmosphere somehow. In the previous examples, molecule representations were generated using RDKit for this example, but this time the fingerprint itself is learned by DNN.

So, representing molecules as a graph is the input. As Atom_matrix, (max_atoms, num_atom_features) is used as Edge_matrix, (max_atoms, max_degree) as bond_tensor, and three matrices (max_atoms, max_degree, num_bond_features) are used. Since each molecule has a different number of atoms, max_atoms defines the maximum number of atoms. By doing this, it becomes input of the same matrix size for each numerator and batch learning becomes possible.

If you want to execute Example, please enter the following command.

```
python example.py
```

Reference link: - NGF-paper - DeepChem-paper - keiserlab - HIPS NFP - Theano base - for keras1.x - ericmjl/graph_fp - DeepChem - About Eary stopping - SIFT original Paper

Chapter 12: Let the computer think about the chemical structure

A generation model is one of the things that Deep Learning has had a great impact on the medicinal chemistry. In particular, the evolution of generation models in the last few years is amazing. Here, let's propose a new synthesis proposal using REINVENT developed by Marcus Olivecrona.

What is a Generation Model?

The prediction model built in Chapter 11 is generally called a discrimination model. On the other hand, by modeling the distribution of inputs, it is possible to generate sampling or input data from the model. This is called a generative model.

For more details , we recommend reading PRML 1.5.4

Preparation

Install a deep learning library called PyTorch with conda. It does not work with the new version, so specify the version and install it.

What is pytorch?

Like keras, it is a library to use TensorFlow more conveniently.

```
$ conda install pytorch=0.3.1 -c pytorch
```

Then clone REINVENT itself from GitHub.

```
$ cd <path to your working directory>
$ git clone https://github.com/MarcusOlivecrona/REINVENT.git
```

Next, download a pre-trained model with about 1.1 million data sets of ChEMBL and replace it with the original data. This data takes five or six hours using the GTX 1080Ti GPU machine, but if you want to train yourself, the GPU machine is a must.

```
$ wget https://github.com/Mishima-syk/13/raw/master/generator_handson/data.zip
$ unzip data.zip
$ mv data ./REINVENT/
```

Now you are ready.

Illustration

Here we create a model that produces an analogue of the antidiabetic drug sitagliptin, known commercially as Januvia.

First, train the model to generate a highly similar structure using the tanimoto coefficients as scores. This time I will train 3000 steps, but it will take about 7 or 8 hours with Macbook Air, which is a little earlier. If you can not wait, please use the data here.

```
./main.py --scoring-function tanimoto --scoring-function-kwargs query_structure
'N[C@@H](CC(=O)N1CCn2c(C1)nnc2C(F)(F)F)Cc3cc(F)c(F)cc3F' --num-steps 3000 --sigma 80
```

From here, I will launch jupyter notebook.

Load the necessary libraries. Specify the REINVENT directory for sys.path.append.

```
%matplotlib inline
import sys
sys.path.append("[Your REINVENT DIR]")
from rdkit import Chem
from rdkit.Chem import AllChem, DataStructs, Draw
import torch
from model import RNN
from data_structs import Vocabulary
from utils import seq_to_smiles
```

Next, sample 50 compounds from the trained model.

```
voc = Vocabulary(init_from_file="/Users/kzfm/mishima_syk/REINVENT/data/Voc")
Agent = RNN(voc)
Agent.rnn.load_state_dict(torch.load("sitagliptin_agent_3000/Agent.ckpt"))
seqs, agent_likelihood, entropy = Agent.sample(50)
smiles = seq_to_smiles(seqs, voc)
```

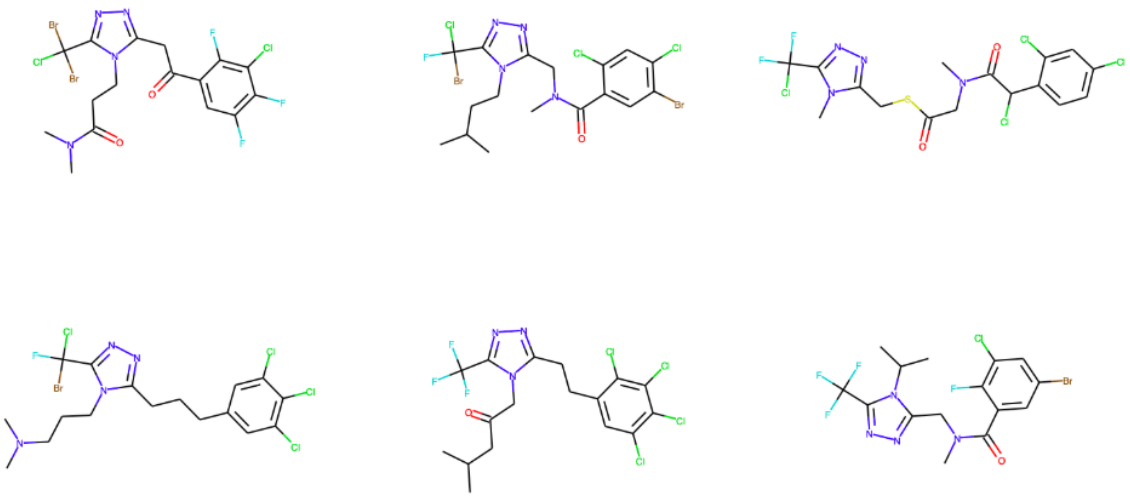
Let's see what kind of structure was actually generated.

```
mols = []
for smi in smiles:
    mol = Chem.MolFromSmiles(smi)
    if mol is not None:
        mols.append(mol)

Draw.MolsToGridImage(mols, molsPerRow=3, subImgSize=(500,400))
```

Is there anything like that?

In [3]:
```
mols = []
for smi in smiles:
    mol = Chem.MolFromSmiles(smi)
    if mol is not None:
        mols.append(mol)

Draw.MolsToGridImage(mols, molsPerRow=3, subImgSize=(500,400))
```

Out[3]:



About REINVENT

By all means, please read Molecular De Novo Design through Deep Reinforcement Learning

Chapter 13: Conclusion

To learn more

NOTE

If you are interested in what you are interested in, you can send requests to the issue that you want to know more, or reply with twitter. Also recommended suggestions are saved.

Those who want to learn more machine learning

You should aim to be able to read through Pattern Recognition and Machine Learning(PRML). The PDF can be downloaded for free.

If you find that PRML is tough, you may find it easier to search for "before reading PRML" etc. so you should choose the one that suits you.

Want to learn more about Chemoinformatics from IT

Since this book focuses on AI drug discovery, it explains the basics of machine learning and analysis methods, but chemoinformatics is, like bioinformatics, an efficient way of expressing molecules and data. It also includes storage methods and fast search technology. If you are interested in chemoinformatics as such informatics (IT aspect), it is recommended to read more from Chemoinformatics: Basic Concepts and Methods and dig deeper on topics of interest.

For a deeper understanding of medicinal chemistry and chemoinformatics

If you belong to the pharmacokinetics, toxicity, or pharmacology of a pharmaceutical company or academia and want to know the point of this book by all means , we recommend that you read Drug-Like Properties: Concepts, Structure Design and Methods from ADME to Toxicity Optimization. You This is a text that is generally read by new employees who are assigned to the synthesis department of a pharmaceutical company, so it would be fun for anyone who has read this book. If there is a part that I can not catch up with, I can go over the related books, and I think it is good to learn further from this book as a clue.

In addition, people who are involved in pharmacokinetics should be able to use it as a strength in PBPK modeling if they can understand QSAR / QSPR in this document . Since optimization of kinetic profiles is very important for drug differentiation strategies, it may be very useful to have strong QSPR + PBPK.

If you want to be a drug designer

Although this book has introduced informatics methods based on low molecular weight compounds, understanding of the target protein is essential when interpreting the results. In other words, drug design can not be done without understanding the three-dimensional structure of proteins. Therefore, it is good to read and learn books related to SBDD.

NOTE	Unfortunately I have not studied SBDD in books, so please tell someone good books

Furthermore, since SBDD deals with proteins, it is not necessary to concatenate it with chemoinformatics and bioinformatics. If you understand both in the framework of drug discovery, you will be able to think in more depth, so let's be able to do both. That is absolutely fun. DRY analysis books and information technology that supports life science data analysis will surely help your career.

As mentioned in Chapter 6, quantum chemical calculation is important to understand protein-ligand interactions. In particular, the ability to interpret interactions based on quantum chemistry in future SBDDs can be stated to be essential. Without prejudice think about the chemical in orbit concept - the basic quantum chemistry please read the like. If you're using Gamess, you'll be able to help the new version of Quantum Chemical Beginners' Manual. At least save energy decomposition analisys, which will increase your ability to interpret calculations and contribute to your project. Furthermore, FMO is needless to say, but it is an indispensable tool, so understanding each component will help drug design more than that

Beyond the "end"

You can add more advanced content than this manual as a chapter. Please do PR. Add them to the contributor and specify the author at the beginning of the chapter.

>>>