Student ID: 2598002w

In Silico Drug Discovery Program for Potential Virtual High-Throughput Screening

Summary

In Silico drug design has become a fundamental part of the drug discovery process, as it saves companies time, money and resources. Virtual Screening is a popular method used, for high-throughput filtering of molecules for desirable chemical properties.
Here a program was created that allows users to upload around 100 molecules, which can then be added into an SQL database, this database can then be queried effectively under different customisable parameters such as Lipinski's Rule of 5, Lead-Likeness and Bioavailability, these results can also be saved. This program is effective, however not scalable and in its current state has quite basic functions due to time constraints.

Introduction

In drug design there are many fundamental considerations, not only does the drug have to hit the drug target causing the desired outcome. The drug molecule has to be able to be absorbed by the body, digested, metabolised, and excreted by the body – this is referred to as ADME properties (Ekins and Rose, 2002). Which also means, the drug can't be toxic, or become toxic by being metabolised in the body, it can't be excreted too quickly, but also not remain in the body for too long, it has to be specific enough so not to cause unwanted effects in other areas of the body, amongst many other factors (Wishart, 2007).
The expense of developing drugs is immense, and most potential drugs fail, so having a way to identify molecules as potential ligands in silico has been a popular for many years as technology advances (Wadood et al., 2013).
There are many methods of in-silico drug design, a very common one, and the focus in this lab report is high-throughput virtual screening. This method allows to analyse and filter out millions of molecules without needing the time and financial cost of testing each molecule in the lab (Kitchen et al., 2004). Companies can initially create big databases of molecules from which can be filtered by chosen parameters, SQL servers are a great tool as querying is very efficient, so effective for high throughput screening. For example, this was used in pharmacogenetic study which created a robust database which allowed for screening for anti-cancer drugs (Smirnov et al., 2018).

The parameters from which to filter molecules in databases has come from current successful drugs and the chemical properties that make them effective (van de Waterbeemd and Gifford, 2003). This allows to screen 100,000s or 1,000,000s of molecules at once in a very cost-effective manner, and ensures in vivo tests have a better chance of succeeding.
The first to define a set of rules was Chris Lipinski, Lipinski's rules provide a basis of which

could give an indication that it has properties of a successful drug (Owens, 2003). Since then many more stringent rules, for example lead-likeness and bioavailability were developed, each can have varying filters and parameters (Ursu et al., 2011).

As well as ADME, the properties that are used for filtering include; solubility in fat – lipophilicity, and water – hydrophilicity, which can be measured using the partition coefficient (LogP), and distribution coefficient (LogD) (Oprea, 2002). Also, ring count, molecular weight, and polar surface area (PSA) can indicate hydrophobicity (Hill and Young, 2010). Permeability is important for a drug ligand, and can be assessed using molecular weight, and Hydrogen (H) Bond Donor/Acceptor counts (Lipinski et al., 1997). Finally, looking at rotatable bonds can indicate if the molecule will have many different conformations when docking (Veber et al., 2002).

The aim was to develop a program was to be able to take in molecules in an .sd/.sdf format and upload them with their chemical properties calculated to an SQL database, which can then be queried under given parameters (e.g., Lipinski's rule of 5).

Method

To start a database was created with 4 tables in MYSQL:

Compound:

Compound ID	# of Atoms	# of Bonds

Atom:

Atom #	Atom Type	X Coordinate	Y Coordinate	Z Coordinate	*Compound ID*

Bond:

Bond #	Atom 1 #	Atom 2 #	Bond Order	*Compound ID*

Parameters:

LogP	LogD	PSA	Mass	HBAC	HBDC	Ring Count	RBC	FARC	*Compound ID*

Table Set 1: Relational Schema for SQL database for molecule data. Primary keys have been underlined, and foreign keys are in italics. PSA = Polar Surface Area, HBAC = Hydrogen Bond Acceptor Count, HBDC = Hydrogen Bond Donor Count, RBC = Rotatable Bond Count, FARC = Fused Aromatic Ring Count.

The relation seen in table set 1, has been normalised to fourth normal form which creates a strong relation; preventing anomalies if data needed to inserted, removed or updated.

There were 2 programs created in java, the first called SDBMaker which takes molecule files, splits the information into the tables above and uploads the data to a given SQL database.

The second program queries the data in the database, which allows for filtered results under different parameters.

The code was based and developed off of a given code written by Dr. Graham Hamilton initially created by Dr. David P Leader. This code created a Graphical User Interface (GUI) that took single molecule files (.sd files), or a directory of .sd files using a LoadFile class and parsed the molecules into information for the compound, atom and bond tables, that could then be saved as a .sql file and uploaded to a database.
From there, code was added that could split files with multiple molecules (.sdf files), this was done by scanning the molecule and saving it to a string using the "$$$$" sign as an end point for each molecule, these strings would be sent in to be parsed in the FileParser class. In the FileParser, the molecule was split up into the first 3 tables from table set 1 – Compound, Atom, and Bond, for each table the information was taken and stored in the appropriate object – either Compound, Atom, or Bond. All the molecule objects were then stored in arrays of their object in the appropriate list class, either CompoundList, AtomList, or BondList.

After the file is parsed, the Calculator class is called which creates a Stats object. This Stats object calls all the calculations for each parameter and stores them as attributes. ChemAxon Marvin Beans plugins were used for all calculations (ChemAxon), where in each calculation method the molecule string was read into a MolImporter, and a Molecule object was created which the plugin used for all the calculations.

Other classes were called from the Calculator class which make up the parameters table, the calculations chosen were; LogP, LogD, PSA, H bond donor count, H bond acceptor count, ring count, rotatable bond count, fused aromatic ring count.

In the LogP class, LogP was calculated using the ChemAxon plugin called "LogPPlugin", default values for Chloride Ion Concentration (0.1), and Sodium Potassium Ion Concentration (0.1) were used. Also, the consensus LogP method was chosen, which was constructed with PHYSPROP database and the ChemAxon (Klopman et al., 1994) models. From this the true LogP value was used.
In the LogD class, LogD was calculated with the "LogDPlugin", and the pH was set to 7.4.
In the TPSA class, the PSA was calculated using the "TPSAPlugin" which is based on the (Ertl et al., 2000) method of calculation, and the pH was set to 7.4.
In the H bond donor and acceptor count was calculated using the "HBDAPlugin", values were kept at default which included the lower pH set to 2.0, the upper pH set to 12.0, the step was set to 2.0, and the pH was set to 7.4. The bond counts with multiplicity were used.
In the RingAndRot class, the ring count, rotatable bond count, fused aromatic ring count was calculated using the "TopologicalPlugin".
In the Mass class, the molecular weight was calculated by using the getMass() method from the Molecule object.

Once the objects are created each is stored in a Stats object, where all the molecule are stored in an array list of Stats sorted in the StatsList class.
Files are excluded if there is an issue parsing or with calculations, and if LogP or LogD had NaN returned then this was set to NULL in the database.

Once the calculations are finished the data can either be saved in .sql files which can be used to upload into SQL, or can be directly uploaded into a chosen SQL database assuming the correct tables have been added. For saving as .sql files a getFormatted() method was used in each table class to create a string of tabbed spaced variables for each table, and a loop was used to string each of the molecules together. Then uses the SaveFile method to save file to computer. Otherwise, if chosen to directly upload to the database, then the connection with the SQL server was obtained using the Connect class, which was called in the SQLConnectFrame class which allowed the user to enter the host username and password of the server. After this, the getSQLFormatted method was used for each class to create SQL insert statements which were batch added per table. Any double attribute was set to 5 significant figures. This program can now be closed.

Then SDBAccess Program can be used to query the database under the given parameters. This also was based off of a code by Dr. Graham Hamilton, and Dr. David P Leader, which could query a database by chosen bond order.
The main SDBAccess class created the GUI which comprised of 5 tabs which filter the database under different customisable parameters: Lipinskis Rules, Lead-Likeness, Bioavailability, Bond Search, and Browse All. On each tab on the east panel there were the parameter labels with the text areas where users can change the filter values. The text areas have the default numbers for the parameters, but users can easily change this. Then by clicking the submit button it calls the appropriate class (either Lipinskis, LeadLike, Bioavailability, BondSearch, or BrowseAll) which queries the database by first gaining a connection to the SQL server through the connect class, in each class the target database was soft coded in. Each of the class returns the results in a multi-dimensional object array, this is then passed back to the SDBAccess class which puts the results in a scrollable table. This is then added to the central panel for the user to view.
When the results show this makes the save button visible, which uses the SaveFile class to allow the user to save the results as tab separated file.

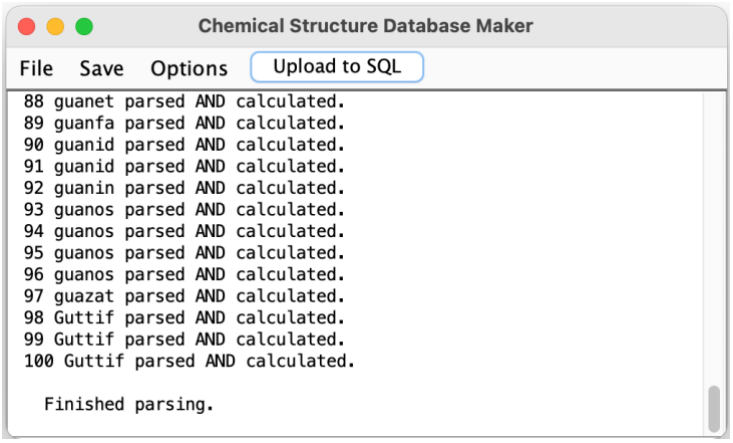


Figure 1: SDBMaker GUI after the file has been uploaded with information in feedback area

Results

The outcome program produces a simple GUI where first there is an options menu where there would be options to change the parameters by which the calculations are done, i.e. pH. This feature is not currently functional due to time constraints but would be an ideal feature. Then the file menu allows for different file types to be uploaded, which is parsed, and calculations made straight away. This takes some time that potentially would not be scalable, as 100 molecules to do this took 26 seconds, so for 1 million molecules would take 7 hours. More importantly however there would be an overflow issue with the arrays. When the file is being uploaded the feedback area ideally would update as each molecule is processed live, however the program ended up processing them pasting once all the files had been parsed (see Figure 1), which seemed to be an issues with compiling.

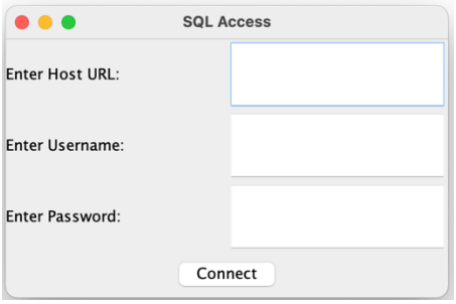


Figure 2: SQLConnectFrame for SQL log in GUI pop up

Once the file has been uploaded the save file options in the save menu become enabled so users can download the appropriate SQL files, each file is downloaded separately. Otherwise, there is an option to directly upload the to an SQL database, and once this button is clicked a log in window pops up allowing the user to enter the host, user and password of the desired SQL server (See Figure 2). Then the uploaded molecules will be entered in the SQL server, and the user will be informed once this this is complete in the feedback area.

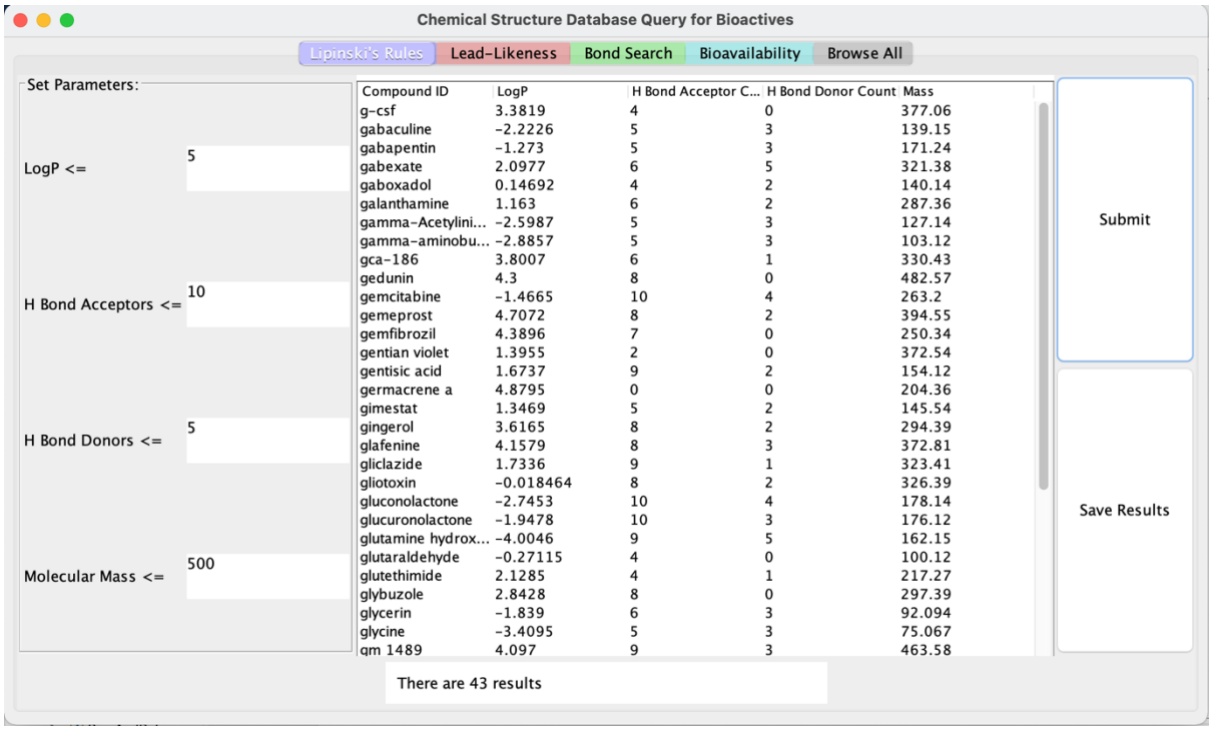
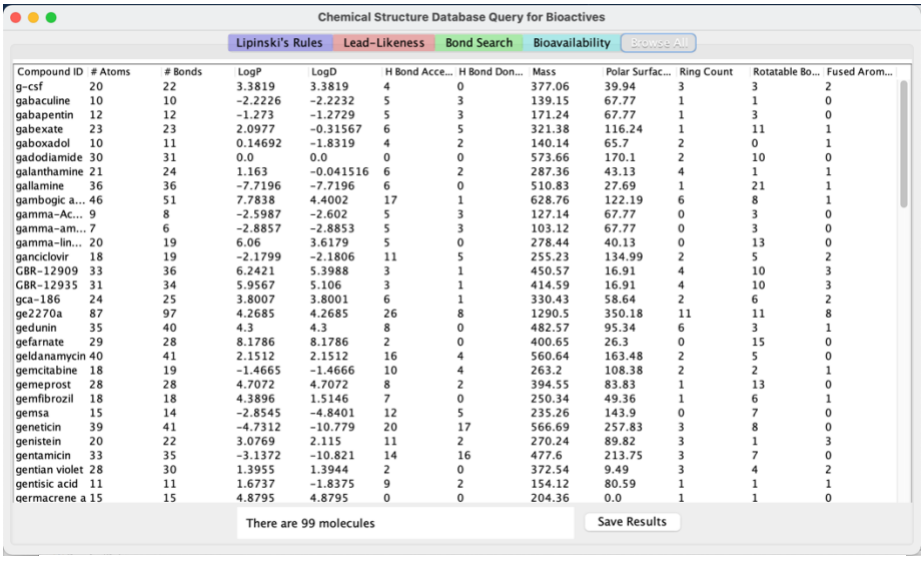


Figure 3: SDBAccess GUI for filtered results, with tabs at the top

The SDBAccess program creates a simple GUI with 5 tabs that allow for filtering with different parameters that have a default value given initially (See Figure 3), whoever the text boxes are easily customisable so users can set their own limits for the parameters. The first tab is Lipinski's Rules, which includes filters for LogP, H Bond Acceptors, H Bond Donors, molecular mass. The second tab was Lead-Likeness which had filters for LogP, upper LogD, lower LogD, H Bond Acceptors, H Bond Donors, rotatable bond count, ring count and molecular mass. The third tab is bioavailability and allows to filter for LogP, H Bond Acceptors, H Bond Donors, polar surface area, fused aromatic ring count, rotatable bond count, and molecular mass. The fourth tab allows to filter for bond order with a drop down for either choosing bond order 1 to 4. The final tab has 1 button which allows the user to view all the compounds in the database with all the values for the above filters, also including the number of atoms and number of bonds in the molecule (See Figure 4). All the tables are scrollable and have ability to sort the rows by values in a chosen column.



Compound ID	# Atoms	# Bonds	LogP	LogD	H Bond Acce...	H Bond Don...	Mass	Polar Surfac...	Ring Count	Rotatable Bo...	Fused Arom...
g-csf	20	22	3.3819	3.3819	4	0	377.06	39.94	3	3	2
gabaculine	10	10	-2.2226	-2.2232	5	3	139.15	67.77	1	1	0
gabapentin	12	12	-1.273	-1.2729	5	3	171.24	67.77	1	3	0
gabexate	23	23	2.0977	-0.31567	6	5	321.38	116.24	1	11	1
gaboxadol	10	11	0.14692	-1.8319	4	2	140.14	65.7	2	0	1
gadodiamide	30	31	0.0	0.0	0	0	573.66	170.1	2	10	0
galanthamine	21	24	1.163	-0.041516	6	2	287.36	43.13	4	1	1
gallamine	36	36	-7.7196	-7.7196	6	0	510.83	27.69	1	21	1
gambogic a...	46	51	7.7838	4.4002	17	1	628.76	122.19	6	8	1
gamma-Ac...	9	8	-2.5987	-2.602	5	3	127.14	67.77	0	3	0
gamma-am...	7	6	-2.8857	-2.8853	5	3	103.12	67.77	0	3	0
gamma-lin...	20	19	6.06	3.6179	5	0	278.44	40.13	0	13	0
ganciclovir	18	19	-2.1799	-2.1806	11	5	255.23	134.99	2	5	2
GBR-12909	33	36	6.2421	5.3988	3	1	450.57	16.91	4	10	3
GBR-12935	31	34	5.9567	5.106	3	1	414.59	16.91	4	10	3
gca-186	24	25	3.8007	3.8001	6	1	330.43	58.64	2	6	2
ge2270a	87	97	4.2685	4.2685	26	8	1290.5	350.18	11	11	8
gedunin	35	40	4.3	4.3	8	0	482.57	95.34	6	3	1
gefarnate	29	28	8.1786	8.1786	2	0	400.65	26.3	0	15	0
geldanamycin	40	41	2.1512	2.1512	16	4	560.64	163.48	2	5	0
gemcitabine	18	19	-1.4665	-1.4666	10	4	263.2	108.38	2	2	1
gemeprost	28	28	4.7072	4.7072	8	2	394.55	83.83	1	13	0
gemfibrozil	18	18	4.3896	1.5146	7	0	250.34	49.36	1	6	1
gemsa	15	14	-2.8545	-4.8401	12	5	235.26	143.9	0	7	0
geneticin	39	41	-4.7312	-10.779	20	17	566.69	257.83	3	8	0
genistein	20	22	3.0769	2.115	11	2	270.24	89.82	3	1	3
gentamicin	33	35	-3.1372	-10.821	14	16	477.6	213.75	3	7	0
gentian violet	28	30	1.3955	1.3944	2	0	372.54	9.49	3	4	2
gentisic acid	11	11	1.6737	-1.8375	9	2	154.12	80.59	1	1	1
germacrene a	15	15	4.8795	4.8795	0	0	204.36	0.0	1	1	0

Figure 4: SDBAccess GUI for all results, with tabs at the top

Discussion

The program is able to effectively create and query small databases, this was tested with 100 molecules which worked very effectively. However, in its current state wouldn't be appropriate on a large scale. This is because there are some fundamental flaws in the program that wouldn't allow it to deal with high-throughput data. For example, a batch method was used to add the SQL insert statements which can go out of memory at quite a low threshold. To counter act this, there could be a threshold of how many tuples are added at once, before executing a new batch to prevent the overload. This would be beneficial as it would keep the time benefits of adding in batches, over adding statements individually. Also, if arrays got too big it could cause an out of memory exception, so this could be dealt with in a similar way. Moreover, the time to run the program would not be scalable, however there are some ways that the program could be sped up. For example, the ChemAxon MolImporter method was called each time in the calculation classes, a more efficient way would include using the MolImorter once to create the Molecule object which could then be stored, and the Molecule could be passed into each table object creation rather than the String.

A benefit to the program was the customisability in the filtering parameters which can be very valuable. However, additional filters might be useful for example being able to take out a specific filter if unwanted, or also adding a filter that can set how many of the parameters must be met. This could be useful as it is well documented that a lot of successful drugs do not meet all the given rules (Hann et al.). Also, being able to change the parameters of the initial methods of calculations (e.g., pH) would allow for further customisations.
Another useful feature would allow the user to enter their specific SQL database for the queries in SDBAccess, (as in SDBMaker Figure 2).

Time constraints was a significant reason there is a lot of areas where the program could be further improved functionally. Ideally having the whole program run in 1 GUI would allow for a much more cohesive program and would easily allow for many more features. For example, it would make adding images to the compound results more straightforward, this would have been done using the MolExporter Plugin allow to download the molecule files of the filtered results.
Another useful feature would have been to have the ability to download a single molecule file of any molecule that doesn't upload, as the individual could more easily check for an issue in the file. Also, being able to download an .sdf file, or a directory of sd files from the filtered results, which would be useful for further analysis.

An additional aspect where the program could improve is in the feedback area, which would update in real-time as each molecule was being parsed. This would be more ideal for the user as they could have a better idea of how the program is running and would also be easier to see any molecules that did not parse. Also, in the results table for the original program 5 significant figures was chosen for double values, however this resulted in a table that is possibly harder to read. To resolve this, 3 significant figures or 2 decimal places is a better representation for the table, with more decimal places stored in the database.

A more advanced feature could include deep learning, which is now a tool being implemented in in-silico drug screening (Popova et al., 2018). It can be useful to helpful to screen drugs more accurately and efficiently.

Overall, the function met the main aim, with the user being able to upload around 100 molecules which can be uploaded to an SQL server. This can be queried with filtering for the given statistics and the parameters can easily be adjusted.

Acknowledgements

Thank you to Dr. Graham Hamilton and Dr. David P Leader for providing the base code. Including the basis for the SDBMarker and SDBAccess classes, as well as the code for the SaveFile, LoadFile, Compound, CompoundList, Atom, AtomList, Bond, BondList, Connect, DBQuery, ParamQuery and FileParser classes which only had small adjustments.

References:

ChemAxon ChemAxon; Marvin Suite - chemaxon.struc., chemaxon.formats., chemaxon.marvin.

Ekins, S., and Rose, J. (2002). In silico ADME/Tox: the state of the art. Journal of Molecular Graphics and Modelling *20*, 305–309.

Ertl, P., Rohde, B., and Selzer, P. (2000). Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties. J. Med. Chem. *43*, 3714–3717.

Hann, M.M., Leach, A.R., and Harper, G. Molecular Complexity and Its Impact on the Probability of Finding Leads for Drug Discovery. 9.

Hill, A.P., and Young, R.J. (2010). Getting physical in drug discovery: a contemporary perspective on solubility and hydrophobicity. Drug Discovery Today *15*, 648–655.

Kitchen, D.B., Decornez, H., Furr, J.R., and Bajorath, J. (2004). Docking and scoring in virtual screening for drug discovery: methods and applications. Nat Rev Drug Discov *3*, 935–949.

Klopman, G., Li, J.-Y., Wang, S., and Dimayuga, M. (1994). Computer Automated log P Calculations Based on an Extended Group Contribution Approach. J. Chem. Inf. Comput. Sci. *34*, 752–781.

Lipinski, C.A., Lombardo, F., Dominy, B.W., and Feeney, P.J. (1997). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Advanced Drug Delivery Reviews *23*, 3–25.

Oprea, T.I. (2002). Current trends in lead discovery: Are we looking for the appropriate properties? 10.

Owens, J. (2003). Chris Lipinski discusses life and chemistry after the Rule of Five. Drug Discovery Today *8*, 12–16.

Popova, M., Isayev, O., and Tropsha, A. (2018). Deep reinforcement learning for de novo drug design. Sci. Adv. *4*, eaap7885.

Smirnov, P., Kofia, V., Maru, A., Freeman, M., Ho, C., El-Hachem, N., Adam, G.-A., Ba-alawi, W., Safikhani, Z., and Haibe-Kains, B. (2018). PharmacoDB: an integrative database for mining in vitro anticancer drug screening studies. Nucleic Acids Research *46*, D994–D1002.

Ursu, O., Rayan, A., Goldblum, A., and Oprea, T.I. (2011). Understanding drug-likeness. WIREs Comput Mol Sci *1*, 760–781.

Veber, D.F., Johnson, S.R., Cheng, H.-Y., Smith, B.R., Ward, K.W., and Kopple, K.D. (2002). Molecular Properties That Influence the Oral Bioavailability of Drug Candidates. J. Med. Chem. *45*, 2615–2623.

Wadood, A., Ahmed, N., Shah, L., Ahmad, A., Hassan, H., and Shams, S. (2013). In-silico drug design: An approach which revolutionarised the drug discovery process. OA Drug Design and Delivery *1*.

van de Waterbeemd, H., and Gifford, E. (2003). ADMET in silico modelling: towards prediction paradise? Nat Rev Drug Discov *2*, 192–204.

Wishart, D.S. (2007). Improving Early Drug Discovery through ADME Modelling. Drugs in R & D *8*, 349–362.

Appendix:

Conformations are crucial when considering drug design, as different conformations could result in differences in docking with the target protein. Below is a flow diagram outlining how a program might generate and analyse these conformations.

Flow Diagram outline program to generate conformations

Read in the molecule file that contain coordinates (x, y and z), use the MolImporter Plugin to convert to ChemAxon Molecule object

Identify the rotatable bonds checking each atom in the molecule using the Topology Analyser Plugin - chemaxon.marvin.calculations.TopologyAnalyserPlugin, if Boolean true - it is a rotatable bond, if return false - not a rotatable bond

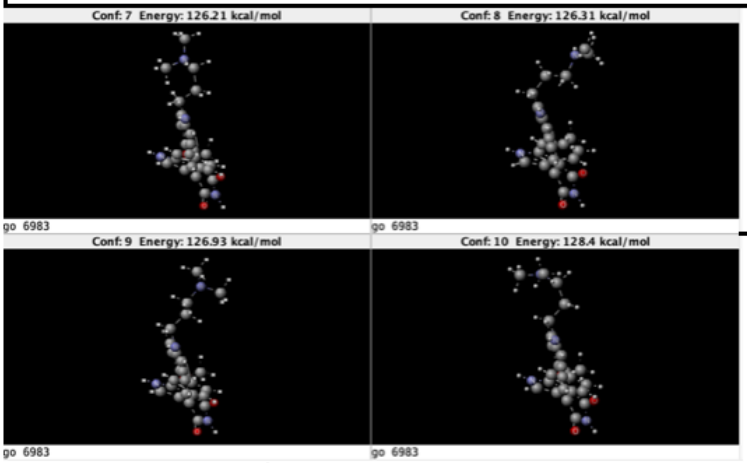
Rotate atoms in selected rotatable bonds using the Graphic Component class, chemaxon.marvin.space.GraphicComponent, saving the coordinates with each rotation

User has option to set length for parameter radii in Angstroms

Validate conformations further:
By using the Periodic Systems Plugin - ChemAxon.struc.PeriodicSystem, to find the Van Der Waals Bonds radii, and filter out those that don't meet the set length.

If there is a steric clash the coordinates will be excluded

Show visualisation of the conformations, with the option to remove any if further inspection identifies any anomalies. Using the Graphical Component class, see below for example:



Finally:
Read out coordinates of the conformations into new sd or mol files

If Boolean returns true

Exclude conformations that don't meet the parameter

If Boolean returns false

~Or~

Uses conformer plug - chemaxon.marvin.calculations.ConformerPlugin, in which would return the conformers in an array of Molecules, also can provide the number of conformers, also can provide Boolean of has valid 3D structure for conformers

Read in the molecule file that contain coordinates (x, y and z), use the MolImporter Plugin to convert to ChemAxon Molecule object