Best Practices for Constructing Reproducible QSAR Models

Chanin Nantasenamat

Abstract Quantitative structure-activity/property relationship (QSAR/QSPR) has been instrumental in unraveling the origins of the mechanism of action for biological activity of interest by means of mathematical formulation as a function of the physicochemical description of chemical structures. Of the growing number of QSAR models being published in the literature, it is estimated that the majority of these models are not reproducible given the heterogeneity of the components of the QSAR model setup (e.g., descriptor, learning algorithm, learning parameters, open source and commercial software, different software versions, etc.) and the limited availability of the underlying raw data and analysis source codes used to construct these models. This inherently poses a challenge for newcomers and practitioners in the field to reproduce or make use of the published QSAR models. However, this is expected to change in light of the growing momentum for open data and data sharing that are being encouraged by funders, publishers and journals as well as driven by the next-generation of researchers who embrace open science for pushing science forward. This chapter examines these issues and provides general guidelines and best practices for constructing reproducible QSAR models.

Cite this article as:

Nantasenamat C. (2020) Best Practices for Constructing Reproducible QSAR Models. In: Roy K. (eds) Ecotoxicological QSARs. Methods in Pharmacology and Toxicology. Humana, New York, NY.

Chanin Nantasenamat
Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok 10700, Thailand

1 Quantitative structure-activity relationship

Quantitative structure-activity relationship (QSAR) is an exciting field that harnesses past biological activity data to drive further experimentations by enabling the prediction/design of biological activity of new compounds, deducing the important molecular features giving rise to good or poor biological activity, prioritizing compounds from a large chemical library, etc. (1, 2). QSAR has successfully been demonstrated to be useful for modeling a wide range of biological and chemical endpoints as summarized in Table 1. In almost 60 years since the coining of the QSAR term by Corwin Hansch (3) the field has evolved from classical QSAR models (i.e., consisting of a few compounds and described by simple descriptors) to complex machine learning-based QSAR models (i.e., encompassing several hundred to thousands of descriptors as modeled by non-linear learning algorithms) (4). The field of QSAR has witnessed its ups and downs (5) and has become pillars for drug discovery (6) and regulatory purposes (7).

2 Laboratory notebooks: Past and Present

Historically, the documentation of experimental results had traditionally been kept within the confinement of paper-based notebooks whereby the scientific benefits of which is to allow subsequent reproduction of the documented experiment while its legal use is to serve as a proof of inventorship (8).

The electronic laboratory notebooks have been introduced as a digital alternative to the paper-based version but with augmented capabilities such as search capability,

Table 1 Summary of target biological and chemical endpoints investigated by QSAR models. Aside from the drug discovery class, other endpoints are categorized according to the convention described by Piir *et al.* (9).

Endpoints	Examples
Physical and chemical properties	Boiling point, melting point, octanol-water partition coefficient, water solubility, etc.
Environmental fate	Biodegradation, bioconcentration, adsorption/desorption in soil, etc.
Ecotoxicity	Acute toxicity to fish, short-term toxicity to Daphnia, toxicity to plants, etc.
Human health	Acute inhalation toxicity, skin irritation, mutagenicity, etc.
Toxicokinetics	Blood-brain barrier penetration, skin penetration, metabolism, etc.
Drug discovery	Enzyme inhibition, enzyme activation, pharmacokinetics, etc.

integration with instrumentation (10) as well as collaborative writing and archiving of results figures and tables. Scientists are increasingly adopting the use of electronic laboratory notebooks in their research laboratories owing to the inherent need to organize the growing volume of biological data (11) with the benefit of being able to access these documents via the internet at any place and time.

With the rising awareness on research reproducibility, scientists are increasingly sharing these notebooks publicly so as to support the open science initiative and in doing so fosters the sharing of associated raw data and analysis code that would have otherwise remained within the confinements of laboratory computers or individual researcher's personal computer (i.e., known as *dark data*).

3 Data sharing

Publishers of research journals are encouraging or requiring that researchers share the scripts and codes used to analyze the data as a condition of publication. Failure to do so (i.e., owing to privacy or safety) may require a written statement justifying the reason. A notable example is the appointment of *reproducibility editors* for overseeing the code and data sets submitted by authors to the Applications and Case Studies (ACS) section of the Journal of the American Statistical Association (JASA). In an editorial article by the Editor-in-Chief of the Journal of Chemical Information and Modeling, William L. Jorgensen formulated a set of guidelines for submitting QSAR work to the journal. A key issue pertaining to data sharing is highlighted in one of the recommendation as follows: *All data and molecular structures used to carryout a QSAR/QSPR study are to be reported in the paper and/or in its Supporting Information, or be readily available, without infringements or restrictions*. Furthermore, publishers (Springer Nature (12)) and journals (PeerJ, PLoS One (13), etc.) have established similar requirements. Of particular note, Vasilevsky et al. (14) performed an analysis of the pervasiveness and quality of data sharing policies in the biomedical literature and found that 11.9% of journals explicitly stated that data sharing was required as a condition of publication.

The advantage of imposing data sharing as a condition for publishing is that potential unintended errors (e.g., missing data, mislabeling, corrupted files, etc.) may be identified prior to publication, which would consequently solve any future problems that may hamper the reproducibility of subsequent works (15). On the other side of the coin, possible reasons that authors may be reluctant to share the proprietary data is that it would reveal the confidentiality of compounds. In addressing this issue, Gedeck *et al.* (16) described an approach for facilitating data sharing and the development of collaborative QSAR models while not revealing the structural information. Polanski *et al.* (17) reviewed the contributing factors for robust QSAR models and of particular note is their proposition that QSAR is highly data dependent and that the underlying data may inherently produce noise that may arise from many factors such as the molecular conformation, computed descriptors, algorithms used, etc.

Table 2 Summary of different dimensions of molecular descriptors. Adapted from Grisoni *et al.* (20).

Dimensions	Description
0-dimensional (0D)	Molecules are directly described by the chemical formula pertaining to atom counts, molecular weight, sum/average of molecular property, etc.
1-dimensional (1D)	Molecules are characterized by substructural features that considers the presence/absence of molecular fragments or functional groups,
2-dimensional (2D)	Molecules are described by the presence and type of chemical bonds that are used to connect atoms together.
3-dimensional (3D)	Molecules are perceived as a geometrical object in space that are characterized by the nature and connectivity of atoms together with the their spatial representation.
4-dimensional (4D)	Representation of the molecule-receptor interaction by means of molecular interaction fields that is generated from grid-based mapping of probes in relation to thousands of evenly spaced grid points.
Higher dimensions	These high dimensional models may be characterized by different induced-fit and solvation models.

4 Data, chemical structure, conformation and descriptors

As we have seen, the data availability is an important prerequisite for model reproducibility. Aside from this, is a series of additional hurdles and challenges that may affect the reproducbility of the QSAR model. Inherently, QSAR models are reliant on the underlying chemical structures that may produce a myriad of possible descriptors that may range anywhere from simple descriptors to various other dimensions ranging from 0-dimensional to 6-dimensional descriptors (18, 19, 20) as summarized in Table 2.

The concept of structure-activity cliffs (21, 22) demonstrated that even a minor change in the chemical structure (i.e., addition or deletion of a methyl group or even the stereoisomeric placement of functional groups may be a deciding factor whether the compound can or cannot bind to the intended target protein) can give rise to significant changes to the observed activity. Such induced-fit of ligands to their target proteins may be affected by the structure-activity cliff concept, but a question arises as to the importance of conformation on other sets of compounds. The bioactive conformation of compounds are known to be principal drivers of their resulting biological activity and thus several QSAR studies have addressed this area.

A notable example is the work of Guimarães *et al.* (23) in which they performed an investigation comparing 2D and 3D QSAR models for a set of halogenated anes-

thetics. Surprisingly, their results indicated that the 2D model provided comparable performance to that of the 3D model thereby suggesting that the 2D descriptors were also robust in their particular investigation. Thus, one can conclude that the influence of the molecular conformation on the resulting QSAR model is system dependent and must therefore be subjected to careful investigation on a case-by-case basis.

Another interesting work by Pissurlenkar *et al.* (24) tackled the traditional paradigm of QSAR that is the concept of *one chemical, one structure, one parameter value* as proposed by the authors. Their development of the so-called ensemble QSAR (*e*QSAR) model takes into account descriptors generated from a set of low-energy conformers instead of the traditional approach of using only one low-energy conformer. The study for the first time establishes the possibility of incorporating conformation flexibility into QSAR models and thus opens up a new area for further exploration of this important paradigm. Recently, Wicker and Cooper (25) proposed a new molecular descriptor $nConf_{20}$ based on chemical connectivity for capturing the conformational space of a molecule. To facilitate usage by the scientific community, the authors also provided the Python code and the accompanying data set (i.e., containing both the calculate molecular descriptors and the class label that can be used for QSAR model building) in the Supporting Information of their article.

5 QSAR model building process

The general procedures for constructing QSAR models is summarized in chronological order in Table 5 and Figure 1. A more in-depth treatment on recommendations and best practices for QSAR model development is described by excellent review articles by Dearden *et al.* (26) and Tropsha *et al.* (27, 28). The concepts presented in Table 5 and the aforementioned articles on best practices of QSAR model development helps to ensure that robust and accurate models are built. In addition to this, there are emerging efforts in the QSAR literature that is targeted at the following issues:

i Determine the confidence level for predictions obtained from QSAR models through the use of conformal predictions.

ii Assess the modelability of data sets so as to elucidate the feasibility of obtaining robust models (29).

iii Constructing interpretable QSAR models that can be of practical use for biologists and medicinal chemists (18).

iv Ensuring the reproducibility of QSAR models such that other research groups can make use of or extend published models.

In efforts to encourage the development of high-quality QSAR models, the Organization for Economic Cooperation and Development (OECD) had formulated a simple set of rules as outlined in Table 3. Criteria 2 of the OECD principles stressed

that robust QSAR models should have *unambiguous algorithm*. At first glance, one would assume that details on the components used in the formulation of the QSAR model that are described in the Materials and Methods section of research articles would be enough to allow reproducibility of the model. As such information are descriptive in nature and as detailed as it may be, one can assume that there may potentially be some elements of ambiguity that may consequently lead to slightly different outcomes (if not different results) from that of the original model. Roy *et al.* (30) had pointed out in their investigation that QSAR are highly dynamic models that can easily be perturbed upon changes in the underlying algorithm for descriptor calculation, software version or software availability using the Dragon software. Moreover, a summary of factors influencing the reproducibility of QSAR models based on our lab's own experience is provided in Table 4.

Piir et al. (9) performed a systematic review of the QSAR literature consisting of 1,533 articles pertaining to 79 biological and chemical endpoints. Their results indicated that 42.5% of articles may be potentially reproducible (i.e., and thus complies with the five OECD principles) given that interested readers invest the necessary effort in retracing the protocol step-by-step using the same software and version. Furthermore, it was suggested that of the machine learning algorithm used in the QSAR literature, multiple linear regression seemed to be afford the most reproducibility owing to its simplicity (i.e., inclusion of MLR equations in the research article). In spite of this, it was found that only 51% were technically complete while the other majority were lacking significant details for reproducibility. Moreover, the authors also provided recommendations and best practices for QSAR reporting.

Early efforts by Spjuth *et al.* (34) had laid important foundations for interoperable QSAR data sets via the use of a QSAR markup language (QSAR-ML) in which the authors established the markup language to house meta data information that defines

Table 3 Summary of OECD principles for QSAR model building.

No.	OECD principles	Description
1	Defined endpoint	To ensure clarity in the endpoint being predicted as they may be derived from different experimental methods or conditions.
2	Unambiguous algorithm	To ensure that underlying details of the model is transparent so as to facilitate model reproducibility.
3	Defined applicability domain	To define the biological/chemical landscape in which the model can reliably make predictions.
4	Measures of model performance	To evaluate the internal and external predictive ability of the model.
5	Mechanistic interpretation	To ensure that the model can be interpreted such that the underlying mechanism of action of compounds are revealed.

pertinent information about the QSAR data set consisting of: chemical structures, descriptors, endpoint and meta data (e.g., authors, license, source reference, etc.). QSAR-ML is implemented via a set of plug-ins in the Bioclipse software via simple to use graphical tools (35). Although useful, but QSAR-ML considers only the pre-modeling phases, which encompasses procedures *1-4*, while the modeling phases spanning procedures *5-9* were not covered.

Further efforts in driving the reproducibility of QSAR models forward was set forth by the works of Ruusmann *et al.* (36, 37) in which they introduced the QSAR DataBank repository (QsarDB). The QsarDB data format is conceptually similar to that of QSAR-ML but extends it to also include model information. Particularly, Predictive Model Markup Language (PMML) is an open standards for encoding information pertaining to the machine learning model thereby allowing model sharing. The flexibility of PMML permits it to act as an intermediary in encoding the essence of the model from amongst the different machine learning softwares and tools that are available (i.e., which is comparable to an interpreter who can speak many languages). In their work, the authors propose the use of the R language for carrying out the model building procedures in the R programming environment fol-

Table 4 Key factors influencing the reproducibility of QSAR models.

No.	Factors	Description
1	Data set	To achieve reproducibility of a QSAR model, the original data set should be available. At a minimum, this entails the provision of the chemical representation and bioactivity values. Other useful information may include references to the original data source.
2	Chemical representation	Availability of chemical representation such as IUPAC name, SMILES notation or other forms of identifier number.
3	Descriptors	The provision of computed descriptors would help to solve any potential issues pertaining to accessibility to commercial software or software updates that may alter descriptor calculation results.
4	Model's parameters/ details	Name and version of software used for multivariate analysis; learning parameters used in the formulation of the model; classical QSAR models readily provide this from the MLR equations.
5	Predicted endpoint values	Availability of the experimental and calculated endpoint values enable readers to compare their own reproduction of the model with that of the original model's results.
6	Data splits	Availability of precise details as to which compound belongs to which data splits (e.g., internal, external, calibration or validation sets) would facilitate comparison with the user's reproduction of models. Details on data split ratios (80/20 split or 70/30 split) or whether undersampling or oversampling were used.

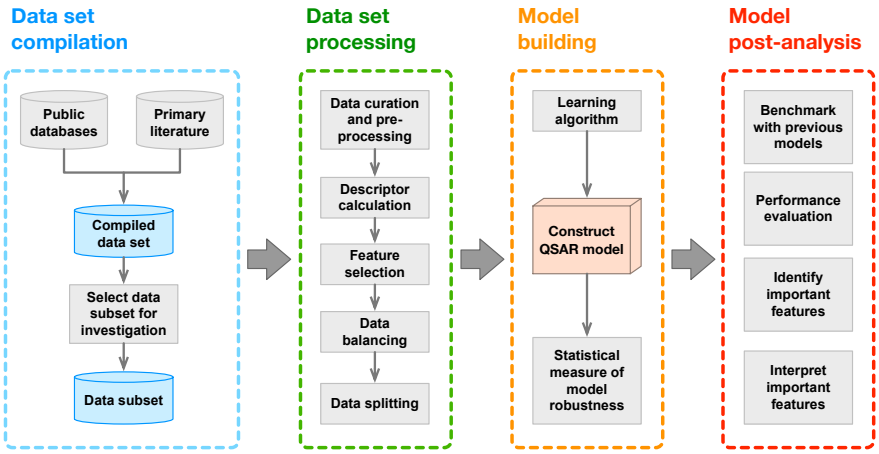

Fig. 1 Schematic representation of the QSAR modeling workflow.

lowed by using the author's own R package *rQsarDB* (38) for data conversion from CSV format to the QsarDB format as well as modifying the contents of existing QsarDB archive directories from within the R environment.

In regards to the pre-modeling phase, data compilation and curation can be considered to be the most time consuming procedures in the QSAR model building process. Aside from the issue of time consumption, the quality of the resulting model is dependent on the quality of the curated data. Thus, the important concept in computer science of *garbage in, garbage out* has become ever more important in the context of QSAR model building as attested by the important articles from Fourches *et al.* (39, 40, 41).

In later sections of this chapter, we describe the use of the Jupyter notebook for performing all of the aforementioned procedures encompassing the pre-processing, construction, validation and evaluation of the robustness of the QSAR model. Such coverage naturally facilitates reproducible construction of QSAR models as the precise protocol, learning function, learning parameters and performance metrics are housed within the Jupyter notebook file. It is increasingly becoming common practice for researchers to share their Jupyter notebook along with accompanying data sets on public repositories such as GitHub or Bitbucket as well as the QsarDB.

Table 5 Summary of procedures for QSAR model building.

No.	Procedures	Description
1	Data compilation	The very essence of QSAR models lies in the compilation of the data set. Potential data sources include the primary literature and curated/semi-curated bioactivity databases
2	Select data subset for investigation	An extension of the previous step is selecting a subset of data from the original data set for further investigation
3	Data curation and pre-processing	This is probably the most time consuming phase of the entire model building process as it entails cleaning the data (e.g., dealing with missing data), normalizing variables (e.g., logarithmic transformation), removal of salts/metals/ duplicates, normalizing chemical structures (e.g., selecting) appropriate tautomers), etc.
4	Descriptor calculation	An important component of the model building process is deciding how to represent the molecular features and physicochemical properties of the compounds of interest. A wide range of open source and commercial software are available. Decisions will have to be made as to use a few interpretable features or to use a large set of features that may or may not be interpretable.
5	Feature selection	Once descriptors are generated, the initial set of descriptors are normally subjected to removal of low variance variables followed by removal of collinear (redundant) variables. Again, there exists a large collection of algorithms for reducing the features (e.g., stepwise linear regression, genetic algorithm, particle swarms optimization, etc.).
6	Data balancing	A common problem for the development of classification models is that the classes of active and inactive compounds are often imbalanced where either classes may be significantly smaller or larger than the other. Such imbalanced data set is not suitable for model building and the classes will have to be balanced either via undersampling or oversampling as well as via more sophisticated approaches such as the SMOTE algorithm.
7	Data splitting	Partitioning the data set into various subsets (e.g., training, calibration, external validation and cross-validation sets) is a common practice for validating the model robustness whether it is capable of reliable prediction on unseen data samples, or for optimizing and tuning the model parameters.
8	Learning algorithm	The highlight of the QSAR model building process is making use of the aforementioned curated data for multivariate analysis so as to correlate computed descriptors with the endpoint values of interest. Learning algorithm can be either supervised or unsupervised (i.e., making use of or not making use of the endpoint variable in the learning process) and the resulting model can be interpretable or not interpretable (black-box models) (18).
9	Statistical measures of model robustness	Model robustness and its reliability are traditionally assessed via various metrics such as R^2, Q^2, RMSE and Y-scrambling. In recent years, conformal predictions (31, 32, 33) and other metrics have also been introduced.

6 Interactive notebooks

Electronic notebooks merely refers to the archiving of explanatory text of what was done and how while the associated data and analysis code may or may not be provided with the notebooks. There are now *interactive notebooks* that makes it possible for the code used to perform the data analysis to be shown alongside the explanatory text and visualizations (e.g., images, plots, etc.). As a result, this afford easy comprehension of the experimental results and the underlying code while also facilitating reproducible research.

A widely adopted interactive notebook that is used in the scientific community is known as the Jupyter notebook (i.e., previously known as iPython notebook). The original iPython notebook was created in 2001 by Fernando Perez and had since evolved to the more general and powerful Jupyter notebook (http://www.jupyter.org/) with support for more than 40 programming languages (e.g., Python, R, Javascript, Latex, etc.).

For the sake of data sharing, it is common practice to store the Jupyter notebooks (i.e., used hereafter to also refer to the iPython notebook) on GitHub (i.e., or other web repository such as BitBucket). Such notebook files can then be rendered as static HTML via the nbviewer (http://nbviewer.jupyter.org/). Moreover, GitHub also makes it possible for Jupyter notebook files to render directly on its repositories. Owing to the static nature of the rendered notebook the resulting HTML is consequently not interactive and therefore not amenable to modifications. A first step towards solving this limitation is made by the Freeman laboratory at Janelia Research Campus in their development of *binder* (http://mybinder.org/), a web service that converts Jupyter notebook files hosted on GitHub to executable and interactive notebooks. Recently, there is a web service known as the *Code Ocean* that not only allow the sharing of the raw data and associated analysis codes but also enable users the capability of running the analysis codes (i.e., supports several open source languages such as R and Python as well as commercial languages such as MATLAB and Stata).

7 Tutorials on using Jupyter notebook for QSAR modeling

7.1 Tutorial 1: Installing Miniconda

Before we begin, let's familiarize ourselves with Conda, which is a package manager that we will be using to manage the installation of packages in supported languages such as R and Python. The reason for using this package manager is that it will simplify the installation of packages by automatically taking care of installing the prerequisites (dependencies) that are needed to run the package of interest. Some packages that may be a challenge to install if performed manually (i.e., requiring

the compilation of C++ code via the use of additional libraries namely Boost) such as the *rdkit* can be easily installed via a one-line command (shown below).

Conda comes in two versions: (1) Anaconda and (2) Miniconda. In this tutorial, we will be using the Miniconda version owing to its requirement of less computer resources. To get started, we will need to install Miniconda by following the steps below:

i In a web browser, go to *https://docs.conda.io/en/latest/miniconda.html*

ii Download the appropriate installer that matches your operating system (Windows, Mac OS X or Linux) and bit version (32-bit or 64-bit).

iii Once Miniconda is installed, try running the *conda* command in a terminal window (i.e., also known as the command prompt window). To a blank prompt that looks like the following:

```
$
```

Run the conda command as follows (press the **Enter** button after typing the command):

```
$ conda
```

If installation went successfully, the following output should be displayed:

```
$ conda
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:

positional arguments:
  command
    clean        Remove unused packages and caches.
    config       Modify configuration values in .condarc. This is modeled
                 after the git config command. Writes to the user .condarc
                 file (/Users/chanin/.condarc) by default.
    create       Create a new conda environment from a list of specified
                 packages.
    help         Displays a list of available conda commands and their help
                 strings.
    info         Display information about current conda install.
    install      Installs a list of packages into a specified conda
                 environment.
    list         List linked packages in a conda environment.
    package      Low-level conda package utility. (EXPERIMENTAL)
    remove       Remove a list of packages from a specified conda environment.
    uninstall    Alias for conda remove. See conda remove --help.
    search       Search for packages and display associated information. The
                 input is a MatchSpec, a query language for conda packages.
                 See examples below.
    update       Updates conda packages to the latest compatible version. This
                 command accepts a list of package names and updates them to
                 the latest versions that are compatible with all other
                 packages in the environment. Conda attempts to install the
                 newest versions of the requested packages. To accomplish
                 this, it may update some packages that are already installed,
                 or install additional packages. To prevent existing packages
                 from updating, use the --no-update-deps option. This may
                 force conda to install older versions of the requested
                 packages, and it does not prevent additional dependency
                 packages from being installed. If you wish to skip dependency
                 checking altogether, use the '--force' option. This may
                 result in an environment with incompatible packages, so this
                 option must be used with great caution.
    upgrade      Alias for conda update. See conda update --help.

optional arguments:
  -h, --help     Show this help message and exit.
  -V, --version  Show the conda version number and exit.
```

```
conda commands available from other packages:
  env
```

7.2 Tutorial 2: Installing packages in conda

As stated previously, conda supports the management of packages of languages such as R and Python. Thus, in this tutorial we will present examples for installing packages in both languages.

7.2.1 Installing Python packages

Installing Python packages is very simple, which can be performed by typing the following command:

```
$ conda install package-name
```

where **package-name** refers to the Python package name. For instance, if we would like to install the *jupyter* package then enter the following:

```
$ conda install jupyter
```

Then, it will ask to confirm that we would like to proceed with the installation, which we will enter **y** as the answer and press the **Enter** button.

```
Proceed ([y]/n)?
```

Instead of manual confirmation for ever package that we would like to install, the confirmation process can be automated by invoking the **-y** function as follows:

```
$ conda install jupyter -y
```

Now this time, the installation process will proceed without manual confirmation, which becomes very convenient if installing more than one package.

To check if the package has been successfully installed, use the **list** function as follows:

```
$ conda list
```

```
$ conda list
# packages in environment at /Users/chanin/miniconda2:
#
# Name                    Version                   Build  Channel
appnope                   0.1.0                    py27_0
asn1crypto                0.24.0                   py27_0
backports                 1.0                      py27_0
backports.functools_lru_cache 1.5                      py27_1
backports_abc             0.5                      py27_0
beautifulsoup4            4.7.1                    py27_1
blas                      1.0                         mkl
bleach                    1.5.0                    py27_0
boost                     1.56.0                   py27_3    rdkit
bzip2                     1.0.6                         3
```

```
ca-certificates           2019.1.23                     0
cairo                     1.14.8                        0
certifi                   2019.3.9                 py27_0
cffi                      1.9.1                    py27_0
chardet                   3.0.4                    py27_1
chembl-webresource-client 0.8.51                   pypi_0    pypi
chembl_webresource_client 0.9.31                   py27_0    chembl
click                     6.7                      pypi_0    pypi
conda                     4.6.14                   py27_0
conda-env                 2.6.0                         1
configparser              3.5.0                    py27_0
cryptography              2.6.1           py27ha12b0ac_0
cycler                    0.10.0                   py27_0
dash                      0.17.5                   pypi_0    pypi
dash-core-components      0.5.0                    pypi_0    pypi
dash-html-components      0.6.1                    pypi_0    pypi
dash-ly                   0.17.3                   pypi_0    pypi
dash-renderer             0.7.3                    pypi_0    pypi
decorator                 4.0.11                   py27_0
easydict                  1.6                      pypi_0    pypi
entrypoints               0.2.2                    py27_1
enum34                    1.1.6                    py27_0
flask                     0.12.2                   pypi_0    pypi
flask-compress            1.4.0                    pypi_0    pypi
flask-seasurf             0.2.2                    pypi_0    pypi
fontconfig                2.12.1                        3
freetype                  2.5.5                         2
functools32               3.2.3.2                  py27_0
futures                   3.2.0                    py27_0
get_terminal_size         1.0.0                    py27_0
gevent                    1.1.2                    pypi_0    pypi
gevent-openssl            1.2                      py27_0    chembl
glew                      1.13.0                        0    mw
greenlet                  0.4.12                   pypi_0    pypi
grequests                 0.2.0                    pypi_0    pypi
html5lib                  0.999                    py27_0
icu                       54.1                          0
idna                      2.8                      py27_0
ipaddress                 1.0.18                   py27_0
ipykernel                 4.5.2                    py27_0
ipython                   5.3.0                    py27_0
ipython_genutils          0.2.0                    py27_0
ipywidgets                6.0.0                    py27_0
itsdangerous              0.24                     pypi_0    pypi
jinja2                    2.9.5                    py27_0
jsonschema                2.5.1                    py27_0
jupyter                   1.0.0                    py27_7
jupyter_client            5.0.0                    py27_0
jupyter_console           5.1.0                    py27_0
jupyter_core              4.3.0                    py27_0
libiconv                  1.14                          0
libpng                    1.6.27                        0
libxml2                   2.9.4                         0
linecache2                1.0.0                    py27_0
lxml                      3.8.0                    pypi_0    pypi
markupsafe                0.23                     py27_2
matplotlib                2.0.0               np111py27_0
mistune                   0.7.4                    py27_0
mkl                       2017.0.1                      0
nbconvert                 5.1.1                    py27_0
nbformat                  4.3.0                    py27_0
nose                      1.3.7                    py27_1
notebook                  4.4.1                    py27_0
numpy                     1.11.3                   py27_0
openbabel                 2.4.1                    py27_3    openbabel
openssl                   1.1.1b               h1de35cc_1
pandas                    0.19.2              np111py27_1
```

```
pandocfilters             1.4.1                    py27_0
path.py                   10.1                     py27_0
pathlib2                  2.2.0                    py27_0
pexpect                   4.2.1                    py27_0
pickleshare               0.7.4                    py27_0
pip                       18.1                     pypi_0    pypi
pixman                    0.34.0                        0
pkgconfig                 1.2.2                    pypi_0    pypi
plip                      1.3.4                    pypi_0    pypi
plotly                    2.0.10                   pypi_0    pypi
pmw                       2.0.1                    py27_0    mw
prompt_toolkit            1.0.13                   py27_0
ptyprocess                0.5.1                    py27_0
pyasn1                    0.1.9                    py27_0
pycosat                   0.6.3            py27h1de35cc_0
pycparser                 2.17                     py27_0
pygments                  2.2.0                    py27_0
pymol                     1.8.0.0.r4144            py27_0    mw
pyopenssl                 16.2.0                   py27_0
pyparsing                 2.1.4                    py27_0
pyqt                      5.6.0                    py27_2
pysocks                   1.6.8                    py27_0
python                    2.7.6                         0
python-dateutil           2.6.0                    py27_0
pytz                      2017.2                   py27_0
pyzmq                     16.0.2                   py27_0
qt                        5.6.2                         0
qtconsole                 4.3.0                    py27_0
rdkit                     2016.09.4           np111py27_1    rdkit
readline                  6.2                           2
requests                  2.5.3                    pypi_0    pypi
requests-cache            0.4.13                   pypi_0    pypi
ruamel_yaml               0.11.14                  py27_1
scandir                   1.5                      py27_0
scikit-learn              0.18.1              np111py27_1
scipy                     0.19.0              np111py27_0
seaborn                   0.8.1                    pypi_0    pypi
setuptools                41.0.1                   py27_0
simplegeneric             0.8.1                    py27_1
singledispatch            3.4.0.3                  py27_0
sip                       4.18                     py27_0
six                       1.10.0                   py27_0
soupsieve                 1.8                      py27_0
sqlite                    3.13.0                        0
ssl_match_hostname        3.4.0.2                  py27_1
subprocess32              3.2.7                    py27_0
terminado                 0.6                      py27_0
testpath                  0.3                      py27_0
tk                        8.5.18                        0
tornado                   4.4.2                    py27_0
traceback2                1.4.0                    py27_0
traitlets                 4.3.2                    py27_0
unittest2                 1.1.0                    py27_0
unittest2six              0.0.0                    py27_0    chembl
urllib3                   1.20                     pypi_0    pypi
wcwidth                   0.1.7                    py27_0
werkzeug                  0.12.2                   pypi_0    pypi
wheel                     0.29.0                   py27_0
widgetsnbextension        2.0.0                    py27_0
yaml                      0.1.6                         0
zlib                      1.2.8                         3
```

A quick inspection of the first column indicated that the *jupyter* package was indeed installed.

7.2.2 Installing multiple Python packages at once

Furthermore, we can also install multiple packages at once as follows:

```
$ conda install jupyter scipy numpy matplotlib scikit-learn
```

7.2.3 Installing Python packages from channels

If the package that we want to install is not available in the default conda repository, then we may need to install these packages from channels (i.e., third-party package repositories other than that provided by Anaconda.org containing the package of interest).

For example, let's say that we would like to install the *rdkit* package, then we will call the following commands:

```
$ conda install -c rdkit rdkit
```

where **-c rdkit** represents the rdkit channel for which we subsequently call upon the **rdkit** package for installation.

Now that the packages are in place, we are ready to proceed to the next step in running the Jupyter notebooks.

7.3 Tutorial 3: Running the Jupyter notebook

To launch the Jupyter notebook, open up a terminal window and enter the following commands:

```
$ jupyter notebook
```

If everything went well, the terminal should display the following message and a new internet browser window will automatically appear (as shown in Figure 2) where the internet browser will be directed to the URL http://localhost:8888/tree/.

```
[I 11:16:10.203 NotebookApp] Serving notebooks from local directory: /Users/chanin
[I 11:16:10.203 NotebookApp] 0 active kernels
[I 11:16:10.203 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 11:16:10.203 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip
      confirmation).
```

A newly created Jupyter notebook file will display an empty cell box (Figure 3) that can house either the code (i.e., both the input code and their corresponding output) or their descriptive text (in Markdown language). The combined use of code and descriptive text in a Jupyter notebook is the hallmark of this platform as it facilitates easy sharing and comprehension of the code's input and output results in an intuitive and rapid manner. An example of a Jupyter notebook showing the step-by-step procedures of how to read and write molecules using the rdkit package in Python is shown in Figure 4.

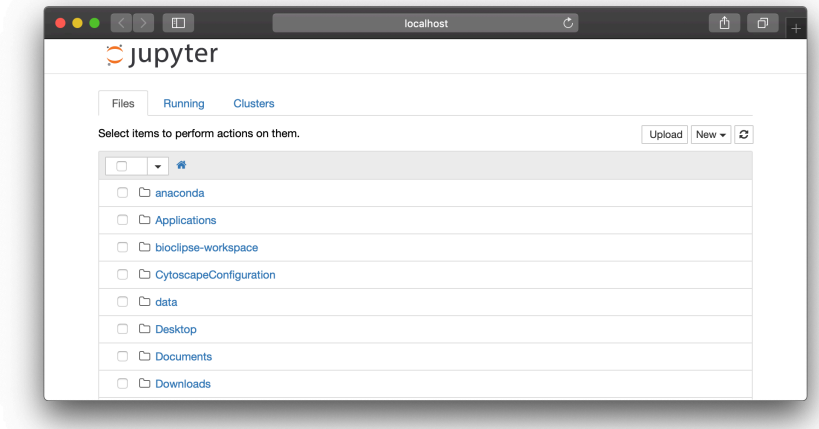

Fig. 2 Screenshot of the default page of Jupyter showing the hard disk content.

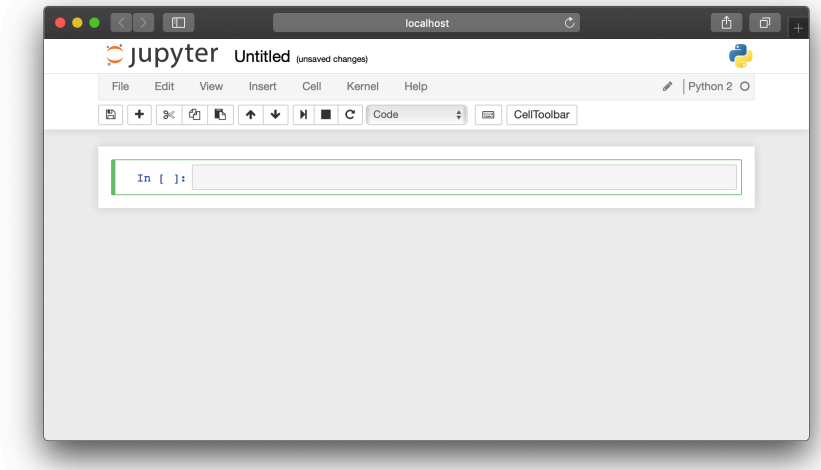

Fig. 3 Screenshot of a newly created Jupyter notebook.

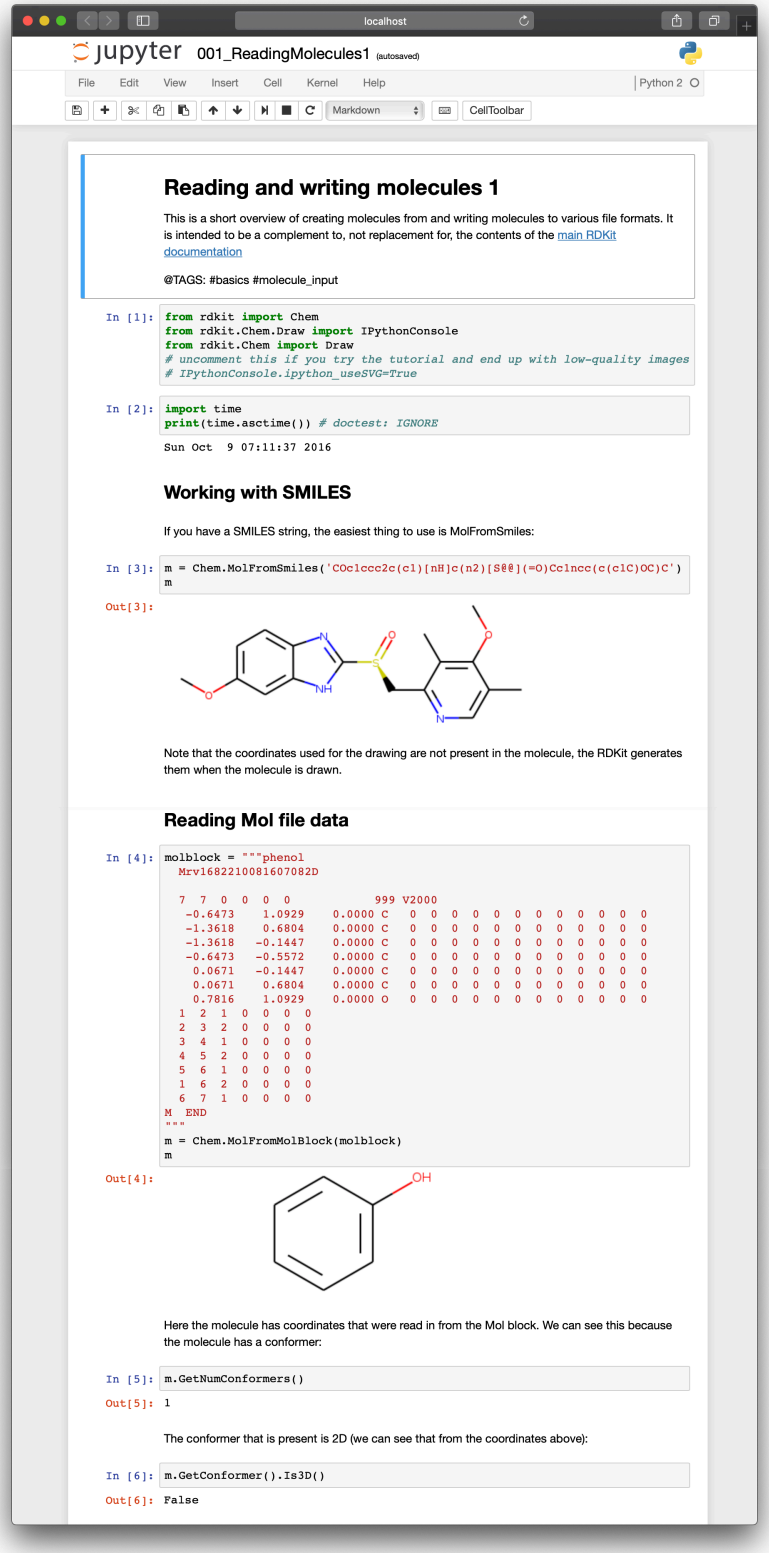

Fig. 4 Screenshot of a Jupyter notebook from an rdkit tutorial by Greg Landrum (42) demonstrating how to read and write molecules.

8 Conclusion

The field of QSAR has grown rapidly and has become a pillar of drug discovery and for regulatory purposes owing to its robustness in effectively predicting endpoints of interests as well as providing pertinent insights for model interpretation. In spite of its usefulness, the literature is still predominated by QSAR models that may not be reproducible. As such, this limiting factor hinders future usage of QSAR models especially in situations where the molecular descriptors may not be computed due to updates or changes to the software or simply due to their unavailability. Similar situations may apply if in the future, significant updates to operating systems may render incompatibility issues with the descriptor or multivariate software. In light of these challenges, interactive notebooks together with exported environment file (i.e., containing information on the modules and specific versions used at the time of code runtime) makes it possible to share the exact replica of the computing environment from the researcher's own computer to their reader's computer. Furthermore, the emergence of container technologies such as Docker and Singularity (not discussed in this chapter) paves further road in creating a suitable environment for facilitating research reproducibility. It is anticipated that the next generation of data-driven biologists would embrace such technologies as a gold standard or best practice for performing computer-based research.

Acknowledgements This work is supported by the Research Career Development Grant (No. RSA6280075) from the Thailand Research Fund.

References

1. Nantasenamat C, Isarankura-Na-Ayudhya C, Naenna T, Prachayasittikul V. A practical overview of quantitative structure-activity relationship. EXCLI J. 2009;8(7):74–88.
2. Nantasenamat C, Isarankura-Na-Ayudhya C, Prachayasittikul V. Advances in computational methods to predict the biological activity of compounds. Exp Opin Drug Discov. 2010;5(7):633–654.
3. Hansch C, Maloney PP, Fujita T, Muir RM. Correlation of Biological Activity of Phenoxyacetic Acids with Hammett Substituent Constants and Partition Coefficients. Nature. 1962;194(4824):178–180.
4. Fujita T, Winkler DA. Understanding the Roles of the "Two QSARs". J Chem Inf Model. 2016;56(2):269–274.
5. Cherkasov A, Muratov EN, Fourches D, Varnek A, Baskin II, Cronin M, et al. QSAR modeling: where have you been? Where are you going to? J Med Chem. 2014;57(12):4977–5010.
6. Sprous DG, Palmer RK, Swanson JT, Lawless M. QSAR in the pharmaceutical research setting: QSAR models for broad, large problems. Curr Top Med Chem. 2010;10(6):619–637.

7. Fjodorova N, Novich M, Vrachko M, Smirnov V, Kharchevnikova N, Zholdakova Z, et al. Directions in QSAR modeling for regulatory uses in OECD member countries, EU and in Russia. J Environ Sci Health C Environ Carcinog Ecotoxicol Rev. 2008;26(2):201–236.
8. Garabedian TE. Laboratory record keeping. Nat Biotechnol. 1997;15(8):799–800.
9. Piir G, Kahn I, Garcia-Sosa AT, Sild S, Ahte P, Maran U. Best Practices for QSAR Model Reporting: Physical and Chemical Properties, Ecotoxicity, Environmental Fate, Human Health, and Toxicokinetics Endpoints. Environ Health Perspect. 2018;126(12):126001.
10. Rubacha M, Rattan AK, Hosselet SC. A review of electronic laboratory notebooks available in the market today. J Lab Autom. 2011;16(1):90–98.
11. Mascarelli A. Research tools: Jump off the page. Nature. 2014;507(7493):523–525.
12. Macmillan Publishers Limited. Announcement: Where are the data? Nature. 2016;537(7619):138.
13. Celi LA, Citi L, Ghassemi M, Pollard TJ. The PLOS ONE collection on machine learning in health and biomedicine: Towards open code and open data. PLoS ONE. 2019;14(1):e0210232.
14. Vasilevsky NA, Minnier J, Haendel MA, Champieux RE. Reproducible and reusable research: are journal data sharing policies meeting the mark? PeerJ. 2017;5:e3208.
15. Greenwald NF, Bandopadhayay P, Beroukhim R. Open data: Spot data glitches before publication. Nature. 2017;550(7676):333.
16. Gedeck P, Skolnik S, Rodde S. Developing Collaborative QSAR Models Without Sharing Structures. J Chem Inf Model. 2017;57(8):1847–1858.
17. Polanski J, Bak A, Gieleciak R, Magdziarz T. Modeling robust QSAR. J Chem Inf Model. 2006;46(6):2310–2318.
18. Shoombuatong W, Prathipati P, Owasirikul W, Worachartcheewan A, Simeon S, Anuwongcharoen N, et al. Towards the Revival of Interpretable QSAR Models. In: Roy K, editor. Advances in QSAR Modeling: Applications in Pharmaceutical, Chemical, Food, Agricultural and Environmental Sciences. Cham: Springer International Publishing; 2017. p. 3–55. Available from: https://doi.org/10.1007/978-3-319-56850-8˙1.
19. Guha R, Willighagen E. A survey of quantitative descriptions of molecular structure. Curr Top Med Chem. 2012;12(18):1946–1956.
20. Grisoni F, Consonni V, Todeschini R. Impact of Molecular Descriptors on Computational Models. In: Brown JB, editor. Computational Chemogenomics. New York: Humana Press; 2018. p. 171–209.
21. Guha R, Van Drie JH. Structure–activity landscape index: identifying and quantifying activity cliffs. J Chem Inf Model. 2008;48(3):646–658.
22. Sisay MT, Peltason L, Bajorath J. Structural interpretation of activity cliffs revealed by systematic analysis of structure-activity relationships in analog series. J Chem Inf Model. 2009;49(10):2179–2189.

23. Guimaraes MC, Duarte MH, Silla JM, Freitas MP. Is conformation a fundamental descriptor in QSAR? A case for halogenated anesthetics. Beilstein J Org Chem. 2016;12:760–768.
24. Pissurlenkar RR, Khedkar VM, Iyer RP, Coutinho EC. Ensemble QSAR: a QSAR method based on conformational ensembles and metric descriptors. J Comput Chem. 2011;32(10):2204–2218.
25. Wicker JG, Cooper RI. Beyond Rotatable Bond Counts: Capturing 3D Conformational Flexibility in a Single Descriptor. J Chem Inf Model. 2016;56(12):2347–2352.
26. Dearden J, Cronin M, Kaiser K. How not to develop a quantitative structure–activity or structure–property relationship (QSAR/QSPR). SAR QSAR Environ Res. 2009;20(3-4):241–266.
27. Tropsha A. Best practices for QSAR model development, validation, and exploitation. Mol Inform. 2010;29(6-7):476–488.
28. Tropsha A, Gramatica P, Gombar VK. The importance of being earnest: validation is the absolute essential for successful application and interpretation of QSPR models. QSAR Comb Sci. 2003;22(1):69–77.
29. Golbraikh A, Muratov E, Fourches D, Tropsha A. Data set modelability by QSAR. J Chem Inf Model. 2014;54(1):1–4.
30. Roy PP, Kovarich S, Gramatica P. QSAR model reproducibility and applicability: a case study of rate constants of hydroxyl radical reaction models applied to polybrominated diphenyl ethers and (benzo-)triazoles. J Comput Chem. 2011;32(11):2386–2396.
31. Svensson F, Aniceto N, Norinder U, Cortes-Ciriano I, Spjuth O, Carlsson L, et al. Conformal Regression for Quantitative Structure-Activity Relationship Modeling-Quantifying Prediction Uncertainty. J Chem Inf Model. 2018;58(5):1132–1140.
32. Bosc N, Atkinson F, Felix E, Gaulton A, Hersey A, Leach AR. Large scale comparison of QSAR and conformal prediction methods and their applications in drug discovery. J Cheminform. 2019;11(1):4.
33. Lampa S, Alvarsson J, Arvidsson Mc Shane S, Berg A, Ahlberg E, Spjuth O. Predicting Off-Target Binding Profiles With Confidence Using Conformal Prediction. Front Pharmacol. 2018;9:1256.
34. Spjuth O, Willighagen EL, Guha R, Eklund M, Wikberg JE. Towards interoperable and reproducible QSAR analyses: Exchange of datasets. J Cheminform. 2010;2:5.
35. Spjuth O, Helmus T, Willighagen EL, Kuhn S, Eklund M, Wagener J, et al. Bioclipse: an open source workbench for chemo- and bioinformatics. BMC Bioinformatics. 2007;8:59.
36. Ruusmann V, Sild S, Maran U. QSAR DataBank - an approach for the digital organization and archiving of QSAR model information. J Cheminform. 2014;6:25.
37. Ruusmann V, Sild S, Maran U. QSAR DataBank repository: open and linked qualitative and quantitative structure-activity relationship models. J Cheminform. 2015;7:32.

38. Ruusmann V, Sild S, Maran U. r-qsardb R package; 2012. https://code.google.com/archive/p/r-qsardb/.
39. Fourches D, Muratov E, Tropsha A. Trust, but verify: on the importance of chemical structure curation in cheminformatics and QSAR modeling research. J Chem Inf Model. 2010;50(7):1189–1204.
40. Fourches D, Muratov E, Tropsha A. Trust, but Verify II: A Practical Guide to Chemogenomics Data Curation. J Chem Inf Model. 2016;56(7):1243–1252.
41. Fourches D, Muratov E, Tropsha A. Curation of chemogenomics data. Nat Chem Biol. 2015;11(8):535.
42. Landrum G. Reading and writing molecules 1; 2016. https://raw.githubusercontent.com/greglandrum/rdkit-tutorials/master/notebooks/001_ReadingMolecules1.ipynb.