Phase 3.8 Command Line Tutorial: Developing Pharmacophore Models

This tutorial demonstrates how to develop pharmacophore models and 3D QSAR models by running Phase utilities and backend modules through the command line.

Phase projects created through the command line are very different from projects created using Maestro, so you will not be able to open a command line project from Maestro. However, the pharmacophore hypotheses created through the command line will be recognized by the Phase graphical interface, so they can be imported into the applications Advanced Pharmacophore Screening, Simplified Pharmacophore Modeling and Screening, and Manage Pharmacophore Hypotheses.

It is strongly recommended that users first examine the Phase User Manual or Quick Start Guide so that they are familiar with certain terminology that appears in this tutorial.

1.1 Conventions

Shell commands that the user should type will be preceded by the "`Shell>`" prompt and they will appear in fixed font, e.g.,

```
Shell> cd pharm_tutorial
```

Parameters for which the user must supply a value will appear in italics and in brackets, e.g.,

```
-mae <maeFile>
```

Important information about changes from previous versions of the software will be indented and italicized:

> *If you have developed pharmacophore models using the Phase 3.5 command tools, you should take note of...*

1.2 Getting Started

In order to run the programs described in this tutorial, Phase 3.8 must be installed on the machine you are currently using, as well as on any remote hosts on which jobs will be run.[1] You must also define the `SCHRODINGER` environment variable so that it points to the directory in which the Schrödinger software is installed.

[1] Schrödinger software is frequently installed on a single filesystem that is accessible to multiple hosts, so strictly speaking, not every host must have its own local installation.

To develop pharmacophore models, you must have a set of molecules stored in one or more Maestro files. Each molecule should be represented by multiple low-energy structures that provide good coverage of that molecule's conformational space. The user is referred to the MacroModel documentation for more details about creating conformational models. If QSAR models are to be developed, then each molecule that will be used to train models should contain an activity property, expressed either in concentration units or as -log[concentration].

Within each Maestro file, conformers for a single molecule must be grouped in consecutive CT blocks. A molecule may be stored as a full CT block (`f_m_ct`) followed by partial CT blocks (`p_m_ct`), or it may be stored as a set of consecutive full CT blocks. If two consecutive full CT blocks differ only in their stereochemistry, they will be treated as conformers of a single molecule unless the titles for those two blocks are different, or the `-stereo` flag is supplied to `pharm_project`. Note that `-stereo` works only when the Maestro file is annotated with stereochemical properties (`s_st_Chirality*`, `s_st_EZ*`).

1.3 Pharmacophore Model Utilities

The following utility programs should be present in the directory `$SCHRODINGER/utilities`:

`pharm_help`	–	Prints a help message summarizing the command line pharmacophore model workflow, including all the utilities that follow.
`pharm_project`	–	Creates a new command line pharmacophore model project and adds molecules to an existing project.
`pharm_data`	–	Performs various operations on the project data.
`pharm_create_sites`	–	Does setup/cleanup for the job that creates pharmacophore sites.
`pharm_find_common`	–	Does setup/cleanup for the job that identifies common pharmacophores.
`pharm_score_actives`	–	Does setup/cleanup for the job that scores hypotheses with respect to actives.
`pharm_score_inactives`	–	Does setup/cleanup for the job that scores hypotheses with respect to inactives.
`pharm_cluster_hypotheses`	–	Does setup/cleanup for the job that clusters hypotheses by geometric similarity.
`pharm_cluster_modes`	–	Clusters ligands by possible binding mode.
`pharm_build_qsar`	–	Does setup/cleanup for the job that builds QSAR models.
`pharm_archive`	–	Preserves project data in a tar archive.
`align_hypoPair`	–	Aligns/merges a pair of hypotheses.
`create_xvolShell`	–	Creates a shell of excluded volume spheres around one or more ligands. Provides a means of defining shape-based constraints for database searching.
`create_xvolClash`	–	Creates excluded volumes using actives and inactives that have been aligned to a hypothesis. Excluded volumes are

	placed in locations that would cause steric clashes only for the inactives.
`create_xvolReceptor`	– Creates excluded volumes using a receptor structure or a portion thereof.
`phase_qsar_stats`	– Extracts statistics from a hit file that contains QSAR predictions.
`qsarVis`	– Standalone graphical interface for visualizing QSAR models. Must have xterm emulation.

Except where noted in this tutorial, all changes to project files should be done only through the use of the `pharm_*` utilities above.

In addition to the above utilities, you should find the following *top-level* programs in the `$SCHRODINGER` directory:

`phase_feature`	– Creates pharmacophore sites.
`phase_partition`	– Identifies common pharmacophores containing a given number of sites.
`phase_multiPartition`	– Runs `phase_partition` one or more times to identify common pharmacophores containing the largest possible number of sites.
`phase_scoring`	– Scores hypotheses with respect to actives.
`phase_inactive`	– Scores hypotheses with respect to inactives.
`phase_hypoCluster`	– Clusters hypotheses by geometric similarity.
`phase_multiQsar`	– Builds 3D QSAR models for a collection of hypotheses, and generates a statistical summary for each model.
`phase_qsar`	– Builds a single QSAR model, and generates detailed output.
`phase_find_matches`	– Aligns structures to a hypothesis.

1.4 Creating a New Command Line Project

Place a copy of the file `pharm_tutorial.tar.gz` in a directory in which you have write privileges and extract the file as follows:

```
Shell> gunzip -c pharm_tutorial.tar.gz | tar xf -
```

Move to the `pharm_tutorial/` directory and create a subdirectory to hold the project:

```
Shell> cd pharm_tutorial
Shell> mkdir myProj
```

`myProj`/ will be the root directory for the project, and you need to be in that directory when running any of the pharmacophore model utilities described in Section 1.3. Accordingly, move to the project directory:

```
Shell> cd myProj
```

Before running any command line utility, it's a good idea to simply type the name of that utility with no arguments to see the usage message. We will be creating the project using `pharm_project`, so type the following command:

```
Shell> $SCHRODINGER/utilities/pharm_project
```

Please examine the usage message produced so that you are aware of the available options and parameters.

We will now create a new project and populate it with a set of 36 endothelin ligands that were taken from the literature (see Krystek, S.R.; Hunt, J.T. Jr.; Stein, P.D.; Stouch, T.R Three-Dimensional Quantitative Structure-Activity Relationships of Sulfonamide Endothelin Inhibitors. *J. Med. Chem.* **1995**, *38*, 659-668). Conformations for these molecules have already been created and are stored in the file `pharm_tutorial/userFiles/endo.mae.gz`. While in the directory `pharm_tutorial/myProj`, issue the following command to create a new project and import the 36 endothelin ligands:

```
Shell> $SCHRODINGER/utilities/pharm_project -new \
       -mae ../userFiles/endo.mae.gz \
       -act r_user_Activity \
       -conf r_mmod_Relative_Potential_Energy-MMFF94s
```

We are using the `-act` option because the Maestro file contains an activity property and we intend to use the associated data to build QSAR models. If you know the activity values but they are not stored in the Maestro file, then you can manually add them to the file `MasterData.tab` after you import the structures (see Section 1.5). You may omit the `-act` option whenever you create non-QSAR projects.

We are using the `-conf` option because the Maestro file contains relative conformational energies, and we intend to use those values when we score hypotheses. If you omit the `-conf` option here, you can still add the conformational energies to the project later using the `pharm_data` utility.

Output will be written to the screen as each block of conformers is split out from `endo.mae`. When the utility finishes, the following files will be created:

`ligands/`	–	Subdirectory that holds all structural data for the project ligands.
`ligands/*.mae`	–	Individual ligand files split out from the file `endo.mae`.
`MasterData.tab`	–	A specially formatted text file that holds project data required in various steps of the workflow. Certain modifications are permitted by hand or through the use of `pharm_data`.
`MasterData.backup`	–	A backup copy of `MasterData.tab`. Used to revert changes you make to `MasterData.tab`. Do not modify.
`ProjectLigands.inp`	–	Ligand records file. Provides a compact summary of project data, and serves as a template for creating subsets of ligands to align to a hypothesis. While the file can be modified without affecting the

		integrity of project, it is recommended that you leave it as is, and make a copy of the file if you need to define a subset.
`FeatureFreq.tab`	–	Feature frequency file. Used to set minimum and maximum allowed feature frequencies for common pharmacophore perception.
`FeatureTol.tab`	–	Feature matching tolerances that can be applied when hypotheses are scored with respect to actives.
`pharma_feature.ini`	–	Default pharmacophore feature definitions. You may replace this file with customized definitions, which you would normally create through the Maestro graphical interface (Create Pharmacophore Hypothesis Manually → Edit Features; or Develop Common Pharmacophore Hypotheses → Create Sites → Edit Features).

1.5 The Master Data File

Of particular interest at this point is the file `MasterData.tab`, which contains various pieces of ligand data that are used throughout the workflow. You may examine this file using a text editor, but be careful not to make any changes right now. At the top of the file, you will find a description of the data it contains, and a number of rules regarding how the data may be modified:

```
################################################################################
#                                                                              #
# Phase Master Data File                                                       #
#                                                                              #
# You may change PHARM_SET, LIGAND_GROUP, MUST_MATCH, QSAR_SET, ACTIVITY and   #
# 1D_VALUE.  To propagate changes to the project, use 'pharm_data -commit'.    #
# To revert to the most recently committed version of MasterData.tab, use      #
# 'pharm_data -restore'.                                                       #
#                                                                              #
# PHARM_SET:    Allowed values are "active", "inactive" and "none".            #
#               active   - Used to identify common pharmacophores and to score #
#                          hypotheses.  There must be at least two ligands and #
#                          two ligand groups with PHARM_SET = active.          #
#               inactive - Used to measure the degree to which hypotheses      #
#                          discriminate actives from inactives.                #
#               none     - Not used in pharmacophore model development.        #
#                                                                              #
# LIGAND_GROUP: Allowed values are 0, 1, 2, etc.  Ligands assigned to the same #
#               group will be treated as interchangeable for purposes of       #
#               finding common pharmacophores.                                 #
#                                                                              #
# MUST_MATCH:   Allowed values are "true" and "false".  If "true", all common  #
#               pharmacophores will be required to match the ligand, or at     #
#               least one ligand from the group to which it is assigned.       #
#                                                                              #
# QSAR_SET:     Allowed values are "train", "test" and "none".                 #
#               train - Used to develop QSAR models.  Recommend at least five  #
#                       training set ligands for each PLS factor.              #
#               test  - Used to test QSAR models.                              #
#               none  - QSAR models not applied to these ligands.              #
#                                                                             #
# ACTIVITY:      Ligand activity.  Values should increase with potency, for   #
#                example, -logKi or -logIC50.  If activity is unknown, the    #
#                value should be "missing".                                   #
#                                                                             #
# 1D_VALUE:      A conformationally-independent numerical property that may be #
#                used during hypothesis scoring to influence or control the   #
#                selection of reference ligands.                              #
#                                                                             #
###############################################################################
```

The `LIGAND_GROUP` property allows you to define subsets of compounds that you wish to treat equivalently when finding common pharmacophores, and/or that you wish to keep together when assigning QSAR training and test sets. For example, if a particular active is represented by two tautomeric forms, `pharm_project` automatically treats the two forms as separate molecules, because they have different connection tables. However, if you assign the same `LIGAND_GROUP` value to both forms, they will be treated as interchangeable when perceiving common pharmacophores. In other words, if a pharmacophore is matched by one form, it's the same as if it were matched by both. You can include any number of compounds in a single group, and you can create as many groups as you like, although a given compound can appear in only one group, and all compounds in a particular group must have the same `PHARM_SET` value. Note that when you create groupings among actives, the language of common pharmacophore perception is changed from "match a minimum required number of actives" to "match a minimum required number of active groups," where an active group contains one or more actives (see Section 1.9). For simplicity, we shall normally use the term "active" rather than "active group," but the user should be aware that the latter term is operative when ligand groups have been defined.

`MUST_MATCH` is relevant when the number of actives that must match the pharmacophore is smaller than the total number of actives. In some instances you may not have a preference for which subset of actives matches the pharmacophore, but in other instances you may wish to require specific actives to match, for example, the most active ligand. Note that if you've defined ligand groups, the `MUST_MATCH` value must be the same for every ligand in a given group. Furthermore, if `MUST_MATCH` is "true" for a group, matching any ligand from that group satisfies the `MUST_MATCH` requirement.

Note that the property corresponding to `1D_VALUE` is created for you, i.e., it does not come from the original Maestro file you imported. `1D_VALUE` allows you to control the set of ligands that can act as a reference when you score hypotheses with respect to actives. By default, any of the actives may be the source of a hypothesis, but if you assign a non-zero `1D_VALUE` for certain actives, and a zero value for the remaining actives, then you can force hypotheses to come from only those actives with a non-zero `1D_VALUE`. This is similar to using `ACTIVITY` to bias the selection of reference ligands to favor those with higher activities, but `1D_VALUE` allows you to control the selection more precisely.

The Maestro files in the `ligands/` subdirectory contain properties that are linked to those in `MasterData.tab` by way of the property name block:

```
###############################################################################
```

```
LIGAND_NAME_PROPERTY = s_phase_Ligand_Name
PHARM_SET_PROPERTY = s_phase_Pharm_Set
LIGAND_GROUP_PROPERTY = i_phase_Ligand_Group
MUST_MATCH_PROPERTY = b_phase_Must_Match
QSAR_SET_PROPERTY = s_phase_QSAR_Set
ACT_PROPERTY = r_phase_Ligand_Activity
1D_PROPERTY = r_phase_Ligand_1D_Property
CONF_PROPERTY = r_mmod_Relative_Potential_Energy-MMFF94s
################################################################################
```

It is critical that you *not* make any changes to this portion of `MasterData.tab`. Note that the `CONF_PROPERTY` values do not actually appear in the master data file, because they are conformationally dependent, and because there is really no reason you would need or want to change those values.

Moving down the file, you will see a block of data for each ligand stored in the project, e.g.,

```
################################################################################
LIGAND_NAME = mol_1
TITLE = "endo-1"
PHARM_SET = active
LIGAND_GROUP = 1
MUST_MATCH = false
QSAR_SET = train
ACTIVITY = 5.509
1D_VALUE = 0.0
################################################################################
```

As indicated at the top of the file, you are allowed to change the values assigned to `PHARM_SET`, `LIGAND_GROUP`, `MUST_MATCH`, `QSAR_SET`, `ACTIVITY`, and `1D_VALUE`. Changes can be made by hand, or with the aid of the utility `pharm_data`, which allows you to perform a number of numerical and logical operations on the data.

To view the allowed `pharm_data` operations, simply type the name of that utility with no arguments:

```
Shell> $SCHRODINGER/utilities/pharm_data
```

This produces the following output:

```
 pharm_data - Performs various operations on MasterData.tab and propagates
 any changes in this file to the rest of the project.  This includes changes
 that may have been made by hand.

 Usage: pharm_data [-log | -exp]
                   [-multiply <scale>]
                   [-active <aboveVal>]
                   [-inactive <belowVal>]
                   [-group [<titles>] | -ungroup]
                   [-train <numTrain> [-rand <seed> [-pharm_set] [-sort]]]
                   [-conf <confProp>]
                   [-commit]
```

```
                  [-restore]

-log                   - Perform -log10(ACTIVITY) conversion.
-exp                   - Perform 10^(-ACTIVITY) conversion.
-multiply <scale>      - Perform <scale>*ACTIVITY operation.  Done prior to
                         -log10(ACTIVITY) and after 10^(-ACTIVITY).
-active <aboveVal>     - Use an activity threshold to split compounds into
                         "active" and "none" PHARM_SET categories.  All
                         compounds are affected.
-inactive <belowVal> - Use an activity threshold to assign compounds to the
                         "inactive" PHARM_SET category.  Only compounds with
                         activities below the threshold are affected.  May be
                         used in combination with the -active option.
-group [<titles>]      - Assign the same LIGAND_GROUP value to compounds with
                         the same title and PHARM_SET.  If a comma-separated
                         list of titles is provided, groupings are restricted
                         to compounds that share one of supplied titles.  Any
                         commas within a title must be replaced by \, and the
                         entire title must be enclosed in double quotes.  For
                         example, 4-pregnene-3,11,20-trione must be specified
                         as "4-pregnene-3\,11\,20-trione".  Compounds already
                         assigned to non-singleton groups are not affected.
-ungroup                 Remove existing groupings.  Not valid in combination
                         with -group.
-train <numTrain>      - Split compounds into "train" and "test" QSAR_SET
                         categories.  Compounds for which QSAR_SET = none are
                         skipped, as are compounds with ACTIVITY = missing.
                         By default, the first <numTrain> qualifying compounds
                         will be assigned QSAR_SET = train.  If groupings are
                         defined, <numTrain> is interpreted as the number of
                         training set groups, and qualifying compounds in the
                         same group are always assigned the same QSAR_SET
                         value.
-rand <seed>           - Assign "train" and "test" QSAR_SET categories
                         randomly using the supplied random seed integer.
                         Must lie between 1 and 2147483646.  Valid only when
                         -train <numTrain> is used.
-pharm_set             - Consider PHARM_SET when assigning random QSAR_SET
                         categories.  If this flag is used, compounds with
                         PHARM_SET = "active" or "inactive" will always be
                         assigned to the "train" QSAR_SET if they qualify.
                         Valid only when -rand <seed> is used.
-sort                  - Sort by activity and sample from equal-sized bins
                         so that training and test sets cover the activity
                         coordinate as uniformly as possible.  If groupings
                         are defined, the average activity of each group is
                         is used.  Valid only with -rand <seed>.
-conf <confProp>       - The name of the relative conformational energy
                         property exactly as it appears in the Maestro file(s)
                         originally supplied when pharm_project was run.  Use
                         this option if you wish to score hypotheses with
                         respect to relative conformational energy, but you
                         neglected to define the property when pharm_project
                         was run.
-commit                - Commit changes in MasterData.tab to project.  Any
                         forward project data affected by these changes will
                         be removed upon confirmation by the user.
```

```
 -restore               - Restore previous MasterData.tab file.  This would
                          normally be done when the user decides that he does
                          not want to remove forward data (e.g., after a
                          -commit operation is aborted).  May not be used in
                          combination with any other option.
```

Note that if you make any changes to `MasterData.tab`, whether by hand or through operations supported by `pharm_data`, you must use the `-commit` option to update the Maestro files in the `ligands/` subdirectory. If you do not update the Maestro files, various Phase backend modules will not be using the modified values because they read the property data directly from the Maestro files. If you plan to run `pharm_data` multiple times to make a series of changes, then you need only supply the `-commit` flag the final time you run `pharm_data`. You can even run `pharm_data` and supply nothing but the `-commit` flag. This is how you would commit changes made by hand.

If you have completed any forward steps in the project workflow, the results generated in those steps may be invalidated by changes you make to `MasterData.tab`. When you attempt to commit the changes, you will be supplied with a list of forward files that will be invalidated, and you will be given a chance to abort the commit operation. If you choose to abort, you can rerun `pharm_data` with the `-restore` flag to revert to the previous version of `MasterData.tab` (i.e., the data stored in `MasterData.backup`).

If your activities are expressed in concentration units (e.g. K_i or IC_{50} values) and you intend to create QSAR models, then you *must* use the `-log` and `-commit` options. You must also perform the `-log` conversion on concentrations if you plan to assign `PHARM_SET` categories using the `-active` or `-inactive` options, because these assignments are based on the assumption that `ACTIVITY` increases as potency increases. In the current example, the activity data are -log[IC_{50}] values, so no conversion is necessary.

1.6 Defining Active and Inactive Sets

The activity values in `MasterData.tab` range from 4.276 to 8.398. These are molar pIC_{50} values, so the corresponding concentrations range from about 53 μM to 4 nM. Clearly not all of the compounds in this data set are highly active, so we would not expect all of them to satisfy the pharmacophore model we ultimately seek. Accordingly, we will define an active set and an inactive set, where members of the former are all expected to satisfy the pharmacophore model, whereas one or more members of the latter may not. We will set a threshold of 7.3 for the active set (IC_{50} =50 nM) and a threshold of 5.0 for the inactive set (IC_{50} = 10 μM).

```
Shell> $SCHRODINGER/utilities/pharm_data -active 7.3 -inactive 5.0
```

Note that `pharm_data` reminds you to commit your changes, i.e.,

```
pharm_data successfully completed

Use 'pharm_data -commit' to propagate your changes to the rest of the project
```

If you examine the PHARM_SET values in MasterData.tab, you will see that a total of five compounds (mol_5, mol_8, mol_9, mol_16, mol_17) have been assigned to the active set, while a total of six compounds (mol_20, mol_23, mol_29, mol_31, mol_32, mol_34) have been assigned to the inactive set. The remaining compounds fall into a "gray" area of activity and will not be considered when developing pharmacophore models.

Note that one need not define an inactive set, but there is no harm in doing so. The active set contributes the pool of pharmacophores from which hypotheses are developed, and these same compounds are used to assign an initial set of scores to the hypotheses. The inactive set may be used subsequently to assign adjusted scores that reflect the degree to which the hypotheses distinguish actives from inactives. This is particularly useful if every compound in the active set is based on a common scaffold, which can give rise to a number of spurious hypotheses that contain features which have nothing do with ligand binding.

1.7 Defining QSAR Training and Test Sets

The project contains a total of 36 compounds, and by default, all of them have been assigned to the QSAR training set. However, one should always hold out a test set to validate QSAR models after they are developed, because there is no reason to be confident of a model if it has not been tested on compounds that played no role in developing that model.[2]

We will be selecting a random training set of 27 compounds, with the remaining nine compounds going into the test set. We will make the further restriction that compounds for which PHARM_SET is either "active" or "inactive" cannot be included in the QSAR test set. In doing so, we guarantee that the test set has no role in the development or ranking of pharmacophore models, which could ultimately affect the set of QSAR models generated and/or the characteristics of those models. Note that this restriction will lead to a test set for which all activities are in the "gray" area as discussed in Section 1.6. This means that test set statistics which depend on the spread in the experimental activities (Q^2, Pearson r) will tend to be lower than they might otherwise be if compounds outside the gray area were in the test set. If we wished to include test set compounds with activities toward the extremes, we could simply edit MasterData.tab and change a few of the "active" and "inactive" PHARM_SET values to "none" before choosing the QSAR training and test sets. For this tutorial, however, we will simply keep all the extreme values in the training set. Accordingly, issue the following command to randomly assign a 27/9 QSAR_SET split, subject to the PHARM_SET restriction:

```
Shell> $SCHRODINGER/utilities/pharm_data -train 27 -rand 123456789 \
       -pharm_set -sort
```

[2] Note that the use of a true test set is superior to internal cross-validation techniques such as leave-*n*-out, where small subsets of the training set are temporarily held out and predicted using models built from the remainder of the training set. Although cross-validated R^2 is a ubiquitous quantity in the QSAR literature, it is of limited value in assessing how the associated model will perform on new compounds, in part because it tends to mirror the overly optimistic statistics obtained for the training set (see Appendix D). For these and other reasons, Phase users are strongly encouraged to validate all models using an external test set.

We are using the `-sort` flag to help ensure that the training and test sets are sampled uniformly along the experimental activity coordinate.[3]

When `pharm_data` finishes, you should examine `MasterData.tab` to verify that there are 9 compounds with a `QSAR_SET` value of "`test`", and that the `PHARM_SET` value for each of those compounds is "`none`".

Note that if you had defined ligand groupings, *`<numTrain>`* would have been interpreted as the number of training set groups, and all the compounds in a given group would have been assigned to either the training set or the test set

Now commit the changes in `MasterData.tab` to the remainder of the project:

```
Shell> $SCHRODINGER/utilities/pharm_data –commit
```

Output will be written to the screen as each Maestro file in the `ligands/` directory is updated. If you now examine the file `ProjectLigands.inp`, you will see a tabular summary of all the committed data, mirroring the property values in `MasterData.tab`.

1.8 Creating Pharmacophore Sites

Before we can develop pharmacophore models, we must map a set of pharmacophore feature definitions to the conformations of each ligand. This process will identify *pharmacophore sites*, which are the geometric locations of hydrogen bond acceptors (A), hydrogen bond donors (D), hydrophobic groups (H), negative ionizable centers (N), positive ionizable centers (P) and aromatic rings (R). Pharmacophore models will be developed by enumerating and comparing groups of pharmacophore sites among the ligands in the active `PHARM_SET`.

We will setup the site creation job using `pharm_create_sites`, so type the name of that utility with no arguments to see the available options:

```
Shell> $SCHRODINGER/utilities/pharm_create_sites
```

[3] Uniform sampling is achieved by sorting all *eligible* compounds/groups (i.e. those not already assigned to the QSAR training set on account of `PHARM_SET`) in order of increasing activity, defining a series of bins containing equal (or nearly equal) numbers of sorted compounds/groups, and randomly selecting one compound/group from each bin to go into either the test or training set, whichever represents a minority of the eligible compounds/groups. In the current example there are 25 eligible compounds (36 compounds – 5 actives – 6 inactives), and 9 of these are to be assigned to the test set, which is the minority set. Therefore, bins of average width 25/9 are defined, and one test set compound is selected randomly from each. To achieve the correct average bin width, the left inclusive boundaries of the bins are a series of integers separated by near multiples of 25/9: *i*, *i*+2, *i*+5, *i*+8, *i*+11, *i*+13, etc., where the starting position *i* is a randomly chosen integer between 1 and 25, and a *modulo* 25 operation is applied to integers greater than 25. If ligand groups are defined, the activity used for sorting purposes is the mean activity of the group.

The only setup option is '`-fd <fdFile>`', which allows you to specify a customized feature definition file. As noted in Section 1.4, it is recommended that such a file be created through Maestro. If you have a customized feature definition file from a previous version of Phase, then it is best to simply invoke Maestro to modify the default Phase 3.8 definitions according to your custom patterns.

We will be using the default feature definitions, so setup the site creation job as follows:

```
Shell> $SCHRODINGER/utilities/pharm_create_sites –setup
```

This will create the following files:

`create_sites_feature.ini` –	A copy of the default feature definition file `pharma_feature.ini`.
`create_sites_phase.inp` –	Main input file for `phase_feature`.

The `create_sites` prefix that appears in each file is linked to the job name that will be specified when the `phase_feature` job is launched. This sort of file naming convention is a recurring theme throughout the workflow, and because subsequent steps depend on the existence of files created in previous steps, it's important that you not change the names of any input or output job files.

You may examine the `phase_feature` input file, but do not modify it. When you are ready, launch the job as follows:

```
Shell> $SCHRODINGER/phase_feature create_sites
```

You may monitor the progress of the job by examining the output written to the file `create_sites_phase.log`, e.g.,

```
Shell> tail –100 create_sites_phase.log
```

When the job has completed, you should see the following output at the end of the log file:

```
=====================================================
            LIST OF FEATURES
            LIGAND NAME : endo-36
=====================================================
feature  0  feature type  0  id  A  ptype  -3  atoms  12  11  13
feature  1  feature type  0  id  A  ptype  -7  atoms  9  7  6
feature  2  feature type  0  id  A  ptype  -7  atoms  10  7  6
feature  3  feature type  0  id  A  ptype  -3  atoms  13  12  14
feature  4  feature type  0  id  D  ptype  -1  atoms  19  8
feature  5  feature type  0  id  D  ptype  -1  atoms  24  23
feature  6  feature type  0  id  D  ptype  -1  atoms  28  23
feature  7  feature type  1  id  H  ptype  0  atoms  27
feature  8  feature type  2  id  H  ptype  0  atoms  16
feature  9  feature type  1  id  R  ptype  -8  atoms  11  12  13  14  15
feature  10  feature type  1  id  R  ptype  -8  atoms  1  2  3  4  5  6
=====================================================
write ligand  35  to file  ligands/mol_36_sites.phs
conformations  3
total ligands = 36
```

At this point you may safely run the cleanup step to extract results from the output archive `create_sites_ligands_output.tar`:

```
Shell> $SCHRODINGER/utilities/pharm_create_sites –cleanup
```

You should now find two additional types of files to the `ligands/` directory:

`ligands/*_sites.phs`	– Pharmacophore site coordinates for each conformation.
`ligands/*_xyz.phc`	– Atomic coordinates extracted from the ligand Maestro files. These files have a “stripped-down” format that allows rapid access to conformer structural data in subsequent steps of the workflow.

You may also examine the file `CreateSitesData.tab` to see the number of occurrences of each type of pharmacophore feature in each project ligand. You should not make any changes to this file because it is required for the next step in the workflow.

With pharmacophore sites now stored in the project, we are ready to move on to the perception of common pharmacophores.

1.9 Perceiving Common Pharmacophores

The basic assumption made at this point is that the pharmacophore model will be found within the five ligands for which `PHARM_SET = active`. Henceforth, we shall simply refer to these ligands as the “actives.” We can require that the model be common to all five actives or common to some smaller subset (at least two actives). The recommended approach is to first require that all actives match the pharmacophore, and reduce the number only if no common pharmacophores are found.

We must also decide how many sites will be contained in the common pharmacophores. Currently, this can range from three to seven, but the most meaningful and useful pharmacophore models typically contain between four and six sites. Accordingly, we shall use these limits in the current example, and we will refer to them as `minSites` and `maxSites`, respectively.

Perception of common pharmacophores is carried out by way of `phase_multiPartition`, which runs `phase_partition` one or more times, starting at `maxSites` and reducing to `maxSites`-1,…,`minSites`, if necessary, until common pharmacophores are found. Thus the only results retained are those that correspond to the largest number of sites in the range [`minSites`, `maxSites`] for which common pharmacophores are found (i.e., no mixing of common pharmacophores that contain different numbers of sites).

Each time `phase_partition` is run, a series of binary decision trees are created to partition all possible *n*-point pharmacophores, according to their *intersite distances*, into a set of high-dimensional boxes, with each box containing a cluster of pharmacophores that are similar enough to be considered equivalent. An *n*-point pharmacophore contains $(n\cdot(n-1))/2$ unique intersite distances, so each box has $(n\cdot(n-1))/2$ dimensions. The size of the boxes is user-controllable and determines the allowed range in each intersite distance, hence the similarity of the pharmacophores in each cluster.

In order for a box to give rise to a common pharmacophore, it must contain at least one pharmacophore from each of a minimum required number of actives (or active groups – see Section 1.5). As discussed above, the minimum number may be all actives, or a smaller subset, depending upon what the user has chosen. When a box is found to contain a common pharmacophore, we say that it *survives* the partitioning process. All other boxes are eliminated. For more details, the reader is referred to the Phase User Manual or Quick Start Guide, both of which describe the partitioning process in greater depth.

We will be setting up the job using `pharm_find_common`, so type the name of that utility with no arguments to see the available options:

```
Shell> $SCHRODINGER/utilities/pharm_find_common
```

Note that in addition to the maximum number of sites in the pharmacophores and the minimum number of matching actives, you can also control the pharmacophore feature frequencies by way of the file `FeatureFreq.tab`. Use a text edit to examine this file. The contents should be as follows:

```
################################################################################
#                                                                              #
# Feature frequency file. Used to set mimimum and maximum allowed feature     #
# frequencies for common pharmacophore perception.  You may change these      #
# limits, but do not make any other modifications to this file.               #
#                                                                              #
################################################################################
A 0 4
D 0 4
H 0 4
N 0 4
P 0 4
Q 0 0
R 0 4
X 0 4
Y 0 4
Z 0 4
END_OF_FEATURE_DATA
```

For example, the text ‘`A 0 4`’ indicates that each common pharmacophore shall be restricted to contain between zero and four acceptors. If you had some prior knowledge of the problem at hand, you could adjust these frequencies to narrow the focus in accordance with that knowledge. For example, suppose it has been established that all actives bind to a specific site on the receptor through a hydrogen bond, where the ligand acts as an acceptor. In that case, you have justification to require that each common pharmacophore contain at least one acceptor.

The default frequencies should cover every reasonable possibility and even some unlikely ones. While it is perfectly fine to retain the defaults, we will make the assumption that the pharmacophore model we seek contains no more than three occurrences of any particular type of feature. Accordingly, change all of the ranges, except for that of feature type `Q`, from '`0 4`' to '`0 3`':

```
A 0 3
D 0 3
H 0 3
N 0 3
P 0 3
Q 0 0
R 0 3
X 0 3
Y 0 3
Z 0 3
```

The ranges for `Q` should not be altered because this feature type has a special meaning that the user need not be concerned with. `X`, `Y`, and `Z` are custom feature types, which can be created when you customize your pharmacophore feature definitions. They will have no effect on the current project.

We will setup the job to search for pharmacophores that contain a maximum of six sites, require that all five of our actives match the pharmacophore, and limit the search according to the feature frequencies in `FeatureFreq.tab`:

```
Shell> $SCHRODINGER/utilities/pharm_find_common -setup -sites 6 -match 5 –freq
```

Note that if you omit the `–match` option, the utility will assume that you want to match all active set ligands. If you omit the `–freq` option, the default ranges '`0 4`' will be used for everything (except `Q`).

This command creates the file `find_common_phase.inp`, which is the main input file for `phase_multiPartition`. The contents of this file should be as follows:

```
JOB_NAME=find_common
MIN_INTERSITE_DIST=2
NUM_SITES=6
FINAL_BOX_SIZE=1
MAXIMUM_DEPTH=5
MIN_NUM_LIGANDS_PER_BOX=5
MIN_MAX_SITES=0,0,3
MIN_MAX_SITES=1,0,3
MIN_MAX_SITES=2,0,3
MIN_MAX_SITES=3,0,3
MIN_MAX_SITES=4,0,3
MIN_MAX_SITES=5,0,0
```

```
MIN_MAX_SITES=6,0,3
MIN_MAX_SITES=7,0,3
MIN_MAX_SITES=8,0,3
MIN_MAX_SITES=9,0,3
VARIANT_NAMES=000122,000126,000166,000226,000266,000666,001226,001266,001666,002266
VARIANT_NAMES=002666,012266,012666,022666,122666
USE_VOLUME=true
USE_SELECTIVITY=false
USE_PROPERTY=true
ALIGN_CUTOFF=1.2
ALIGN_WEIGHT=1
VECTOR_CUTOFF=0.5
VECTOR_WEIGHT=1
VOLUME_WEIGHT=1
SELECTIVITY_WEIGHT=1
PROPERTY_WEIGHT=0
BOXES_TO_KEEP=10
PENALTY_CONST=1.1
MIN_BOXES=10
MAX_BOXES=50
LIGAND_DIR=ligands
BOXES_DIR=boxes
RESULT_DIR=results
PROPERTY_NAME=none
FEATURE_ALIGN_CUTOFF_FILE=none
LIGAND_NAME=mol_5
LIGAND_NAME=mol_8
LIGAND_NAME=mol_9
LIGAND_NAME=mol_16
LIGAND_NAME=mol_17
```

The parameters controlling common pharmacophore perception are:

`MIN_INTERSITE_DIST`	– The minimum allowed distance between any two sites in a pharmacophore. This parameter may be used to summarily reject pharmacophores that contain, e.g., an acceptor site and a donor site from the same oxygen. The default is 2.0Å, and it is recommended that this value not be decreased.
`NUM_SITES`	– The maximum number of sites in each pharmacophore. Change only by way of the `-sites` option supplied to `pharm_find_common`. The minimum number of sites is specified when `phase_multiPartition` is run.
`FINAL_BOX_SIZE`	– The size of the boxes into which pharmacophores will ultimately be partitioned. The default and recommended value is 1.0Å. As this value is decreased, it becomes increasingly difficult to identify common pharmacophores. As the value is increased, the number of pharmacophores in surviving boxes usually increases, which may lead to increased memory/disk usage, and longer processing times when hypotheses have to be scored.
`MAXIMUM_DEPTH`	– This is the depth of the binary tree used in the partitioning process. It reflects the number of splits made in reaching the

		`FINAL_BOX_SIZE`. The initial box size is given by `FINAL_BOX_SIZE`·($2^{\text{MAXIMUM\_DEPTH}}$), which is 32 with the current settings. This means that intersite distances as large as 32 Å will be considered when perceiving pharmacophores. Ligands larger than this are exceedingly rare, so it's unlikely that you'd ever need to increase this parameter.
`MIN_NUM_LIGANDS_PER_BOX`	–	The minimum number of actives (or active groups) that must match a pharmacophore in order for it to be considered common. Change only by way of the `-match` option supplied to `pharm_find_common`.
`MIN_MAX_SITES`	–	These parameters reflect the feature frequencies we supplied through the file `FeatureFreq.tab`. Note the following numeric→letter mappings: 0→A, 1→D, 2→H, 3→N, 4→P, 5→Q, 6→R, 7→X, 8→Y, 10→Z.
`VARIANT_NAMES`	–	The pharmacophore families that will be considered. For example, `000122` corresponds to the family AAADHH, i.e., all six-point pharmacophores containing three acceptors, one donor, and two hydrophobes. The list you see was created by considering only the counts of features in the various actives, and thus represent the variants that *could* yield a common pharmacophore. There is no guarantee, however, that a common pharmacophore will be found for each of these variants. You can manually delete any variants you are not interested in, making sure to leave a single comma separator between the remaining variants.
`LIGAND_NAME`	–	The actives that will be used to identify common pharmacophores. Do not change by hand.

Leave the settings in `find_common_phase.inp` as they are, and launch the `phase_multiPartition` job as follows:

```
Shell> $SCHRODINGER/phase_multiPartition -minSites 4 find_common
```

This will result in a search for common pharmacophores containing 6 sites, followed by, if necessary, searches for common pharmacophores containing 5 and 4 sites. Smaller numbers of sites are investigated only if common pharmacophores are not found for a larger number of sites.

This job requires less than one minute on an Intel dual core machine @1.86 GHz. You may monitor the progress of the job by examining the contents of the file `find_common_partition.log`. When the job is finished, you should see the following output at the end of the log file:

```
PROCESSING VARIANT :  000122
NUMBER OF HYPOTHESES FOR THIS VARIANT :  33
VARIANT DONE !!!
```

```
THE FOLLOWING VARIANTS WERE REJECTED :


FIND COMMON PHARMACOPHORE STEP COMPLETED SUCCESSFULLY !!!
Job completed.
```

In this case, common pharmacophores containing 6 sites were found, so there was no attempt to identify common pharmacophores containing 5 or 4 sites. Now run the cleanup step:

```
Shell> $SCHRODINGER/utilities/pharm_find_common -cleanup
```

You will be directed to examine the file `FindCommonPharmData.tab`, the contents of which should be as follows:

```
################################################################################
#                                                                              #
# PHASE Common Pharmacophore Data. NUM_BOXES is the number of surviving boxes #
# for a given variant, and it represents the maximum number of hypotheses that #
# can be generated in the Score Actives step.                                 #
#                                                                              #
################################################################################

VARIANT    NUM_BOXES
--------------------
ADHHRR        108
ADHRRR         64
AADHHR        176
AAHHRR        188
AAADRR         80
AAADHH         33
AAAHRR        138
DHHRRR         17
AADHRR        337
AHHRRR         43
AADRRR         24
AAADHR        203
AAHRRR         51
AAAHHR        106
--------------------
Total:       1568
```

Each of the 1568 surviving boxes contains a number of very similar pharmacophores, any of which could be considered to be a common pharmacophore. However, using a variety of user-adjustable criteria, we shall select a single pharmacophore from each box, and these will be deemed common pharmacophore hypotheses. The process by which the selections are made is covered in the next section.

1.10 Scoring Hypotheses with Respect to Actives

A brief overview of the active scoring procedure is provided here; the reader is referred to the Phase User Manual or Quick Start guide for more details about the underlying algorithms and the user-adjustable parameters.

In the scoring step, a single pharmacophore is selected from each surviving box, and this pharmacophore serves as one possible hypothesis to explain ligand binding. When a pharmacophore is being considered as a candidate to represent an entire box, it is temporarily treated as a *reference* and the other pharmacophores in the box are aligned to it using a standard least-squares procedure. Each pharmacophore can be traced back to a single conformation of some active, and when pharmacophores are aligned to the reference, their associated ligands are subjected to the same transformations. The result is a multi-ligand alignment of pharmacophores and conformations.

For each reference pharmacophore, a score is computed to measure the overall quality of the multi-ligand alignment:

$$\begin{aligned} Score_{site+vector+volume} &= alignWeight \cdot Score_{site} \\ &+ vectorWeight \cdot Score_{vector} \\ &+ volumeWeight \cdot Score_{volume} \end{aligned} \qquad (1)$$

Here, *alignWeight*, *vectorWeight*, and *volumeWeight* are user-adjustable parameters, and the various *Score* terms are obtained by averaging over all pairs of alignments with respect to the reference:

$Score_{site}$	=	Average value of 1.0 – $RMSD_{site}/alignCutoff$
$Score_{vector}$	=	Average cosine between vector features (acceptors, donors, aromatic rings)
$Score_{volume}$	=	Average $Volume_{common}/Volume_{total}$

$Score_{volume}$ is a measure of the overall superposition of each ligand with respect to the reference ligand, using van der Waals models of all heavy atoms.

Note that if a given ligand contributes more than one pharmacophore to the box, each of these pharmacophores is aligned to the reference, and the one yielding the highest pairwise score is selected for the multi-ligand alignment and the average score computation.

There is an additional user-adjustable parameter, *vectorCutoff*, which is used in conjunction with *alignCutoff* to reject certain reference pharmacophores. If the best multi-ligand alignment for a given reference pharmacophore contains any pairwise alignment with $RMSD_{site}$ greater than *alignCutoff*, that pharmacophore is automatically eliminated as a possible hypothesis. A similar condition holds for *vectorCutoff*, except that elimination occurs if the cosine falls *below* the threshold. Often times if a violation is observed for one pharmacophore in a box, it will be observed for others, because it may be due to one or more ligands that align poorly with the others. These ligands tend to exert their effect regardless of which pharmacophore is acting as the reference. Consequently, it is not uncommon for the scoring process to eliminate everything in some boxes, so that the number of hypotheses identified is smaller than the number of surviving boxes. Occasionally, you may observe that no hypotheses are identified. In such cases, you should increase *alignCutoff* and/or decrease *vectorCutoff*.

If activity data are known, then $Score_{site+vector+volume}$ can be augmented with a term that will favor the selection of reference ligands with higher activities:

$$Score_{site+vector+volume+activity} = Score_{site+vector+volume} + activityWeight \cdot Activity_{ref} \qquad (2)$$

In general, any conformationally-independent property can be incorporated into scoring to bias the selection of reference ligands:

$$Score_{site+vector+volume+property} = Score_{site+vector+volume} + propertyWeight \cdot propertyValue_{ref} \qquad (3)$$

As discussed in Section 1.5, the `1D_VALUE` property in `MasterData.tab` may be used in this manner to exercise precise control over which ligands may act as a reference. One need only assign `1D_VALUE` equal 0 or 1, and choose a property weight that is large enough to provide sufficient elevation of the scores for the desired reference ligands.

The scoring formula from eq. 1, 2, or 3 is used to rank the pharmacophores in a given box, and the highest scoring pharmacophore is selected as the hypothesis for that box.

Once hypotheses have been selected, additional terms may be incorporated to arrive at an overall survival score for each hypothesis:

$$\begin{aligned} Score_{survival} &= Score_{site+vector+volume[+property]} + confWeight \cdot ConfEnergy_{ref} \\ &+ selectivityWeight \cdot Score_{selectivity} + matchWeight^{(m-1)} \end{aligned} \qquad (4)$$

$ConfEnergy_{ref}$ is the reference ligand relative conformational energy, $Score_{selectivity}$ is an empirical estimate of the rarity of the hypothesis, *confWeight*, *selectivityWeight* and *matchWeight* are user-adjustable parameters, and *m* is the number of ligands in the alignment, i.e., the number of ligands that match the pharmacophore.

Armed with this basic knowledge of the scoring procedure, we are ready to examine the options for setting up the job that scores hypotheses with respect to actives. Accordingly, issue the following command:

```
Shell> $SCHRODINGER/utilities/pharm_score_actives
```

Observe that in addition to the various scoring function weight parameters, there is a '`-tol`' option to apply feature matching tolerances when multi-ligand alignments are generated. These tolerances are stored in the file `FeatureTol.tab`, whose contents should be as follows:

```
################################################################################
#                                                                              #
# Feature matching tolerances applied when hypotheses are scored with respect  #
# to actives. You may change the tolerances, but do not make any other        #
# modifications to this file. To completely disable the use of tolerances,    #
# remove the FEATURE_ALIGN_CUTOFF_FILE option from score_actives_phase.inp.   #
#                                                                             #
###############################################################################
A 1
D 1
H 1.5
N 0.75
P 0.75
R 1.5
X 1
Y 1
Z 1
END_OF_FEATURE_DATA
```

If you choose to employ feature matching tolerances, they will be applied to the positions of the pharmacophore sites in each pairwise alignment. So, for example, a given pairwise alignment would be rejected if any acceptor site failed to align to within 1.0 Å. This file contains reasonable defaults, but as you gain more experience, you may decide that other values are more appropriate. If you choose not to apply feature matching tolerances, then the *alignCutoff* parameter will be the only threshold applied to the site positions in multi-ligand alignments.

In this example we will forgo the use of feature matching tolerances, and setup the scoring job as follows:

```
Shell> $SCHRODINGER/utilities/pharm_score_actives –setup -act 1.0 -conf 0.01
```

Here, we are using ligand `ACTIVITY` (as opposed to `1D_VALUE`), with a weight of 1.0, and relative conformational energy with a weight of 0.01. Whenever you incorporate any property into the scoring function, you should examine the range of values to help you arrive at an appropriate weight. The `ACTIVITY` property of the actives spans a range of about 0.5 units, so the chosen weight will lead to a maximum differential of 0.5 in the activity-based scoring term. By contrast, the relative conformational energies for the same ligands are almost two orders of magnitude larger, so we are using a proportionately smaller weight to prevent the associated term from completely dominating the scoring function.

The above command creates the main input file `score_actives_phase.inp`, whose contents should be as follows:

```
JOB_NAME=score_actives
MIN_INTERSITE_DIST=2
NUM_SITES=5
FINAL_BOX_SIZE=1
MAXIMUM_DEPTH=5
MIN_MAX_SITES=0,0,0
MIN_MAX_SITES=1,0,0
MIN_MAX_SITES=2,0,0
MIN_MAX_SITES=3,0,0
MIN_MAX_SITES=4,0,0
MIN_MAX_SITES=5,0,0
MIN_MAX_SITES=6,0,0
MIN_MAX_SITES=7,0,0
```

```
MIN_MAX_SITES=8,0,0
MIN_MAX_SITES=9,0,0
VARIANT_NAMES=012266,012666,001226,002266,000166,000122,000266,122666,001266,022666
VARIANT_NAMES=001666,000126,002666,000226
USE_VOLUME=true
USE_SELECTIVITY=true
USE_PROPERTY=true
ALIGN_CUTOFF=1.2
ALIGN_WEIGHT=1
VECTOR_CUTOFF=0.5
VECTOR_WEIGHT=1
VOLUME_WEIGHT=1
SELECTIVITY_WEIGHT=1
PROPERTY_WEIGHT=1
BOXES_TO_KEEP=10
PENALTY_CONST=1
MIN_BOXES=10
MAX_BOXES=50
LIGAND_DIR=ligands
BOXES_DIR=boxes
RESULT_DIR=ScoreActivesResults
PROPERTY_NAME=r_phase_Ligand_Activity
CONFORMATION_PROPERTY=r_mmod_Relative_Potential_Energy-MMFF94s,-0.01
FEATURE_ALIGN_CUTOFF_FILE=none
LIGAND_NAME=mol_5
LIGAND_NAME=mol_8
LIGAND_NAME=mol_9
LIGAND_NAME=mol_16
LIGAND_NAME=mol_17
```

Most of the parameters are easily recognized based on previous discussion. Note that `r_phase_Ligand_Activity` is being treated as a generic, conformationally-independent property. If you had setup the job using the `-prop` flag rather than the `-act` flag, you would instead see the property `r_phase_Ligand_1D_Property`. The parameter `PROPERTY_WEIGHT` corresponds to the weight that we assigned to reference ligand activity.

The `CONFORMATION_PROPERTY` record contains both the name of the relative conformational energy property and the weight we assigned to it. Observe that the weight has been negated for us automatically because we want the score to increase as the relative conformational energy decreases. It's easy to forget this, so `pharm_score_actives` always makes sure the correct algebraic sign is used.

The presence of the options `USE_VOLUME`, `USE_SELECTIVITY` and `USE_PROPERTY` indicate that these components of scoring may be skipped. To disable, simply change the corresponding parameter value from "`true`" to "`false`".

The less obvious parameters are defined as follows:

`PENALTY_CONST` – Corresponds to *matchWeight*. If you assign a value larger than 1.0, survival scores will be higher for hypotheses that match larger numbers actives. This

		parameter will be relevant only if you did not require all actives to match when common pharmacophores were identified.
`BOXES_TO_KEEP`	–	The percentage of top-scoring hypotheses to retain on a per variant basis. The default of 10 means that for each variant scored, only hypotheses with scores in the top 10% will be kept.
`MIN_BOXES`	–	The minimum number of hypotheses to retain for each variant. Enforced when `BOXES_TO_KEEP` results in fewer than `MIN_BOXES`.
`MAX_BOXES`	–	The maximum number of hypotheses to retain for each variant. Enforced when `BOXES_TO_KEEP` results in more than `MAX_BOXES`.

Leave all options at their current setting and submit the active scoring job as follows:

```
Shell> $SCHRODINGER/phase_scoring score_actives
```

This job requires about 5 minutes on an Intel dual core @1.86 GHz. You may monitor the progress of the job by examining the contents of the file `score_actives_scoring.log`. When the job is finished, you should see the following output at the end of the log file:

```
Scoring Variant AAAHHR  ( 14 of 14 )

Starting Score by Vector and Site Alignment ...
Alignment Scoring :    0%  done
Alignment Scoring :   10%  done
Alignment Scoring :   20%  done
Alignment Scoring :   30%  done
Alignment Scoring :   40%  done
Alignment Scoring :   50%  done
Alignment Scoring :   60%  done
Alignment Scoring :   70%  done
Alignment Scoring :   80%  done
Alignment Scoring :   90%  done
Alignment Scoring :  100%  done
Starting Score by Volume ...
Volume Scoring :   10%  done
Volume Scoring :   20%  done
Volume Scoring :   30%  done
Volume Scoring :   40%  done
Volume Scoring :   50%  done
Volume Scoring :   60%  done
Volume Scoring :   70%  done
Volume Scoring :   80%  done
Volume Scoring :   90%  done
Volume Scoring :  100%  done
Variant Scoring Completed

************************************************

Writing output files. Please, wait ...
Scoring Job Completed Successfully !!!
```

Now run the cleanup step:

```
Shell> $SCHRODINGER/utilities/pharm_score_actives -cleanup
```

The utility will unpack the archived scoring results and write summaries to the files `ScoreActivesData.tab` and `ScoreActivesData.csv`. Both files contain the same information, but `ScoreActivesData.tab` is formatted for easy viewing, while `ScoreActivesData.csv` can be loaded into a spreadsheet.

Examine the file `ScoreActivesData.tab` with a text editor. It contains results for 188 hypotheses, so we merely summarize the contents:

```
##########################################################################################
#                                                                                        #
#                    PHASE Hypothesis Scoring Data.  DO NOT MODIFY!!!                    #
#                                                                                        #
##########################################################################################

HypoID        Survival    Site    Vector   Volume    Prop     Conf   Select  Matches  RefLig
------------------------------------------------------------------------------------------
DHHRRR_49      15.0552   0.9684   0.9996   0.9127   8.3980   -0.000  2.7765     5     mol_8
DHHRRR_64      14.9417   0.9645   0.9995   0.9175   8.3980   11.521  2.7774     5     mol_8
DHHRRR_67      14.9204   0.9680   0.9996   0.9003   8.3980   12.212  2.7767     5     mol_8
.
.
.
AADHRR_466     14.2913   0.9651   0.9997   0.9393   8.3980   11.521  2.1044     5     mol_8
AAAHHR_482     14.2893   0.9692   0.9998   0.9131   8.3980   12.212  2.1313     5     mol_8
AAAHHR_485     14.2893   0.9692   0.9998   0.9131   8.3980   12.212  2.1313     5     mol_8
.
.
.
AAADRR_38      13.8075   0.9732   0.9997   0.9401   8.3980   11.521  1.6117     5     mol_8
AAADHH_130     13.7466   0.9445   0.9986   0.8872   8.3980   26.793  1.7861     5     mol_8
AAADHH_46      13.7466   0.9445   0.9986   0.8872   8.3980   26.793  1.7861     5     mol_8
```

For convenience, hypotheses are sorted by decreasing survival score as computed in eq. 4, and there is a column for each quantity that contributes to the survival score. A unique ID is assigned to each hypothesis, according to its variant and the index of its surviving box. The `Matches` column indicates that all hypotheses matched a total of 5 actives, which is precisely what we required when common pharmacophores were identified. The `Site`, `Vector` and `Volume` scores are all restricted to the interval [0,1], while `Selectivity` is defined to lie between 0 and 3. A logarithmic scale is used in the estimation of selectivity, so a value of x means that the hypothesis is expected to match only 1 out of every 10^x drug-like molecules by chance. The `Prop` column is the reference ligand activity and `Conf` is the reference ligand relative conformational energy.

Observe that `mol_8` is the reference ligand for all hypotheses. This ligand has the highest activity, so incorporation of activity into the scoring function does appear to have yielded the intended result.

For each hypothesis in `ScoreActivesData.tab`, you will find a series of files in the `hypotheses/` subdirectory:

<hypoID>`.def` –	Feature definitions.
<hypoID>`.mae` –	Aligned actives.

`<hypoID>.tab` – Primary hypothesis data.
`<hypoID>.xyz` – Hypothesis site coordinates.

These files have the same format as those created by the Phase GUI when hypotheses are exported, so you can import the primary hypothesis file `<hypoID>.xyz` directly into the Phase graphical interface. You may also use these hypotheses to search databases through the command line using `$SCHRODINGER/phasedb_find_matches`.

A number of hypotheses from this example are assigned very similar scores, and this is not uncommon. Two hypotheses may actually originate from the same reference conformation and belong to the same variant, but differ only in the order in which sites of the same pharmacophoric type are mapped (e.g., $A_1H_2H_3R_4R_5R_6$ vs. $A_1H_3H_2R_4R_5R_6$). So the alignments produced may be very nearly the same, or even identical. In Section 1.12, we demonstrate how a subsequent clustering technique can be applied to identify hypotheses that are essentially equivalent and those that are not.

Even hypotheses that come from different variants may yield very similar scores and alignments. This is fairly common when ligands are built on a common scaffold, because any number of hypotheses may arise from that scaffold. But with information only from actives, it is not always straightforward to determine which of these hypotheses are spurious. Scoring with respect to inactives can provide additional discrimination in such cases, and this is covered in the next section.

1.11 Scoring Hypotheses with Respect to Inactives

A molecule that resembles known actives may lack activity itself for any number of reasons, including poor solubility, steric clashes with the receptor, or excessive entropy loss upon binding. These factors are unrelated to the pharmacophore model, but of course there are analogs of actives whose failure to bind is due primarily to a lack of required pharmacophoric elements. So if a particular hypothesis represents the correct pharmacophore model, then some fraction of inactive compounds would be expected to provide a less than satisfactory match to that hypothesis. This is the premise behind scoring with respect to inactives.

The inactive score is directly analogous to the multi-ligand alignment score defined in eq. 1:

$$Score_{inactive} = alignWeight \cdot Score_{site} + vectorWeight \cdot Score_{vector} + volumeWeight \cdot Score_{volume} \qquad (5)$$

If you have searched 3D databases using Phase, you may recognize this formula as an average *fitness* score for the inactives. With default weights of 1.0, $Score_{inactive}$ will have a maximum value of 3.0.

Because inactives are not necessarily expected to match all sites in the hypothesis, the $RMSD_{site}$ component of $Score_{site}$ must be modified to account for this. Accordingly, if a given inactive matches only *k* out of *n* sites in a hypothesis, the effective *n*-point RMSD is computed from the *k*-point RMSD as follows:

$$RMSD_n = \{(1/n)\cdot[k\cdot RMSD_k^2 + (n-k)\cdot alignCutoff^2]\}^{1/2} \qquad (6)$$

Note that if an inactive fails to match at least 3 sites in the hypothesis, an unambiguous alignment cannot be obtained, and its contribution to $Score_{site}$ will be 0.

After attempting to obtain alignments for all inactives, an adjusted score is computed:

$$Score_{adjusted} = Score_{survival} - inactiveWeight\cdot Score_{inactive} \qquad (7)$$

where *inactiveWeight* is a user-adjustable parameter. Observe that hypotheses which are readily matched by inactives will suffer the greatest penalty and hence will receive the lowest adjusted scores.

To setup the inactive scoring job, we need only supply the *inactiveWeight* parameter:

```
Shell> $SCHRODINGER/utilities/pharm_score_inactives –setup -w 1.0
```

You may examine the contents of the main input file `score_inactives_inactive.inp`. The following is a summary of the contents:

```
inactiveWeight=1
phaseOptionsFile=score_inactives_phase.inp
ligandArchive=score_inactives_ligandFiles.tar
ligandDir=.ligands.tmp
hypoArchive=score_inactives_hypoFiles.tar
hypoDir=.hypotheses.tmp
survivalScore(DHHRRR_49)=15.0552
survivalScore(DHHRRR_64)=14.9417
survivalScore(DHHRRR_67)=14.9204
.
.
.
survivalScore(AAADRR_38)=13.8075
survivalScore(AAADHH_130)=13.7466
survivalScore(AAADHH_46)=13.7466
featureFile=score_inactives_feature.ini
tableFile=ScoreInactivesData.tab
csvFile=ScoreInactivesData.csv
```

Aside from `inactiveWeight`, you should not modify any parameter values. You may, however, delete `survivalScore` records for hypotheses that you do not wish to score. For this example, leave the file as is, and submit the inactive scoring job as follows:

```
Shell> $SCHRODINGER/phase_inactive score_inactives
```

This job requires less than one minute on an Intel dual core @1.86 GHz. You may monitor the progress of the job by examining the contents of the file `score_inactives_inactive.log`. When the job is finished, you should see the following output at the end of the log file:

```
mol_34: Finding matches to hypothesis AAADHH_130 . . .
mol_34: Number of matches = 36
mol_34: Top fitness value = 2.00419  number of sites matched = 5
mol_34: Finding matches to hypothesis AAADHH_46 . . .
mol_34: Number of matches = 36
mol_34: Top fitness value = 2.00419  number of sites matched = 5
phase_inactive successfully completed
```

Now run the cleanup step:

```
Shell> $SCHRODINGER/utilities/pharm_score_inactives -cleanup
```

As in the previous section, two summary tables are produced: `ScoreInactivesData.tab` and `ScoreInactivesData.csv`. Use a text editor to examine `ScoreInactivesData.tab`. A summary of the contents is as follows:

```
HypoID        Survival  Inactive  Adjusted
------------------------------------------
DHHRRR_49      15.0552    1.3727   13.6825
DHHRRR_64      14.9417    1.7809   13.1608
DHHRRR_67      14.9204    1.7250   13.1954
AHHRRR_120     14.9110    1.2890   13.6220
DHHRRR_66      14.8624    1.7406   13.1218
AHHRRR_187     14.8228    1.7440   13.0788
.
.
.
AAADRR_38      13.8075    1.8170   11.9905
AAADHH_130     13.7466    2.6149   11.1317
AAADHH_46      13.7466    2.6149   11.1317
```

As a result of the comparatively low inactive scores assigned to `DHHRRR_49` and `AHHRRR_120`, a fairly pronounced gap is now seen between the adjusted scores for these two hypotheses and all other hypotheses. Based on these results, we would now have somewhat greater confidence in `DHHRRR_49` and `AHHRRR_120`. This doesn't necessarily mean that these hypotheses are the most likely to be correct, but they are at least reasonably consistent with the behavior of the known actives and inactives in the data set.

1.12 Clustering Hypotheses by Geometric Similarity

As noted in Section 1.10, it is not uncommon to observe two or more hypotheses with very similar or even identical scores and physical characteristics. This is a consequence of the way in which common pharmacophores are perceived (see Section 1.9). Since the partitioning algorithm operates on an ordered set of intersite distances, it is necessary to consider all permutations among sites of the same type when enumerating pharmacophores. So, for example, the permutations $A_1H_2H_3R_4R_5R_6$ and $A_1H_3H_2R_4R_5R_6$ represent the same basic pharmacophore, but their 15-dimensional intersite distance vectors would generally not be identical, and they may in fact be dissimilar enough to end up in different boxes. As a result, a given box may have a *mirror box* that contains many (though not necessarily all) of the same pharmacophores, giving rise to a *mirror hypothesis* that is indistinguishable (or nearly so) from the original. However, these sorts of redundancies are readily identified by applying a technique that clusters hypotheses based on geometric similarity.

The top-level program `phase_hypoCluster` performs hierarchical agglomerative clustering of hypotheses based on the following similarity measure applied to a pair of hypotheses *i* and *j*, after performing a least-squares alignment of their matching site points:

$$Sim(i,j) = \frac{<i|j>}{\sqrt{<i|i><j|j>}} \tag{8}$$

where

$$<i|j> = alignWeight \cdot Score_{site}(i,j) + vectorWeight \cdot Score_{vector}(i,j) \tag{9}$$

The computation of $<i|j>$ is directly analogous to that of $Score_{site+vector+volume}$ (see eq. 1), except that no volume term is included in $<i|j>$, because doing so would incorporate effects that are unrelated to the geometric characteristics of the hypotheses themselves. Thus whereas a volume term may be helpful in determining the best multi-ligand alignment when scoring hypotheses, it is less helpful when computing the geometric similarity between two hypotheses.

Note that in cases where more than one mapping of the site points is possible, the alignment yielding the highest similarity is used. Note also that when hypotheses *i* and *j* are associated with different variants, the similarity is automatically set to zero. This is done so that the clustering procedure remains true to its original purpose, i.e., distinguishing which hypotheses are geometrically equivalent and which are not. Allowing different variants to cluster together would tend to cloud the situation, when our ultimate goal is to clarify and simplify the hypothesis scoring results. Users who are interested in comparing hypotheses from different variants may wish to run `align_hypoPair` or `phase_hypoSimCalc`, both of which are located in `$SCHRODINGER/utilities`.

We will be setting up the `phase_hypoCluster` job using `pharm_cluster_hypotheses`, so type the name of that utility with no arguments to see the available options:

```
Shell> $SCHRODINGER/utilities/pharm_cluster_hypotheses
```

The only option we need to decide upon is the linkage method, which determines how inter-cluster similarities are measured. This has an impact on the shapes of the clusters that are created, because at each step in the hierarchical, agglomerative clustering process, the most similar pair of clusters is merged. The available linkage methods are:

single	– The similarity between clusters is the highest similarity between any two objects from the two clusters. Produces diffuse, elongated clusters.
average	– The similarity between clusters is the average similarity between all pairs of objects from the two clusters.

complete – The similarity between clusters is the lowest similarity between any two objects from the two clusters. Produces compact, spherical clusters.

In most cases you will want complete linkage because it tends to yield clusters that are more distinct. Accordingly, setup the job as follows:

```
Shell> $SCHRODINGER/utilities/pharm_cluster_hypotheses -setup -link complete
```

This will create the input file `cluster_hypotheses_hypoCluster.inp`, whose contents are summarized as follows:

```
phaseOptionsFile=cluster_hypotheses_phase.inp
hypoArchive=cluster_hypotheses_hypoFiles.tar
hypoDir=.hypotheses.tmp
featureDefFile=cluster_hypotheses_feature.ini
linkageMethod=complete
clusterFile=cluster_hypotheses_hypoCluster.clu
survivalScore(DHHRRR_49)=15.0552
survivalScore(DHHRRR_64)=14.9417
survivalScore(DHHRRR_67)=14.9204
.
.
.
survivalScore(AAADRR_38)=13.8075
survivalScore(AAADHH_130)=13.7466
survivalScore(AAADHH_46)=13.7466
```

Aside from `linkageMethod`, you should not modify any parameter values. You may, however, delete `survivalScore` records for hypotheses that you do not wish to cluster. For this example, leave the file as is, and submit the hypothesis clustering job as follows:

```
Shell> $SCHRODINGER/phase_hypoCluster cluster_hypotheses
```

This job requires less than a minute on an Intel dual core @1.86 GHz. You may monitor the job's progress by examining the contents of the file `cluster_hypotheses_hypoCluster.log`. When the job is finished, you should see the following output at the end of the log file:

```
ADHRRR_26       14.6824     +
                            |
ADHRRR_211      14.6694     +
                         merge
AADHRR_552      14.3323     +

phase_hypoCluster successfully completed
```

Unlike other jobs, you may actually need to examine the contents of the log file, because it can help you to arrive at an appropriate clustering level, which is specified when the cleanup step is run. Accordingly, use a text editor to open `cluster_hypotheses_hypoCluster.log`, and move down the file until you see the following:

```
                              Cluster Counts and Merging Similarities
```

```
                           187      186      185      184      183      182
HypoID         Survival  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000
AHHRRR_167      14.7935     +        +        +        +        +        +
                          merge      |        |        |        |        |
AHHRRR_163      14.7935     +        +        +        +        +        +

AHHRRR_166      14.7642     +        +        +        +        +        +
                                            merge      |        |        |
AHHRRR_170      14.7642     +        +        +        +        +        +

AHHRRR_169      14.6857     +        +        +        +        +        +
```

This table summarizes the entire clustering process, which starts with 188 *singleton* clusters, merges the most similar pair to reduce the number of clusters to 187, and then repeats until everything is merged into a single cluster. The vertical order in which the hypotheses appear reflects the natural groupings established during the clustering process, so when a merge occurs, it always involves two adjacent vertical blocks, i.e., any given cluster is always comprised of a contiguous range of rows.

In the first step, hypotheses `AHHRRR_167` and `AHHRRR_163` are merged into a single cluster. The similarity is 1.0, so these hypotheses are evidently geometrically identical. In the second step, hypotheses `AHHRRR_166` and `AHHRRR_170` are merged, also with a similarity of 1.0. The merging of identical geometries continues on like this, until the total number of clusters is reduced to 108. This will be apparent if you move down the file until you see the following:

```
                               Cluster Counts and Merging Similarities
                           109      108      107      106      105      104
HypoID         Survival  1.00000  0.99439  0.99389  0.99372  0.99358  0.99335
AHHRRR_167      14.7935     +        +        +        +        +        +
                            |        |        |        |        |        |
AHHRRR_163      14.7935     +        +        +        +        +        +

AHHRRR_166      14.7642     +        +        +        +        +        +
                            |        |        |        |        |        |
AHHRRR_170      14.7642     +        +        +        +        +        +
                                                              merge      |
AHHRRR_169      14.6857     +        +        +        +        +        +
                            |        |        |        |        |        |
AHHRRR_165      14.6857     +        +        +        +        +        +
```

When the cluster count is reduced to 108, the clusters merged have a similarity of 0.99439, so they are not identical. To see where the merge actually occurred, simply move down the file and look for the “merge” marker in the corresponding column. You should eventually see the following:

```
AAADHH_46       13.7466     +        +        +        +        +        +

DHHRRR_67       14.9204     +        +        +        +        +        +
                                   merge      |        |        |        |
DHHRRR_66       14.8624     +        +        +        +        +        +

DHHRRR_64       14.9417     +        +        +        +        +        +
```

Once again, two individual hypotheses (i.e., singletons) are being merged to form a new cluster.

Moving back up the file, we see that in the formation of 105 clusters, a pair of non-singleton clusters is being joined. The merging similarity is 0.99358, which corresponds to the most dissimilar pair of hypotheses between the two clusters, because we are using complete linkage. If we had been using, say, average linkage, the similarity would have been an average among all pairs. It's important to keep in mind that the merging similarity is not necessarily the similarity between the two vertically adjacent hypotheses (in this case `AHHRRR_170` and `AHHRRR_169`), but rather the similarity between the two vertically adjacent clusters, and that similarity depends on the linkage method.

At this stage, the clusters being merged are still highly similar, so the hypotheses contained therein are probably almost indistinguishable. What we are interested in is identifying the point at which a fairly pronounced drop occurs in the merging similarity, which would indicate that we have begun joining clusters that contain hypotheses with more noticeable differences. If you move further down the file you should see:

```
                              Cluster Counts and Merging Similarities
                         79       78       77       76       75       74
HypoID         Survival  0.86978  0.72970  0.71944  0.71710  0.71658  0.70925
AHHRRR_167      14.7935     +        +        +        +        +        +
                            |        |        |        |        |        |
AHHRRR_163      14.7935     +        +        +        +        +        +
                            |        |        |        |        |        |
AHHRRR_166      14.7642     +        +        +        +        +        +
```

In forming 79 clusters, the merging similarity drops from 0.97529 to 0.86978, which is the most dramatic drop seen so far. However, there is an even more pronounced drop in forming 78 clusters.

So we can be fairly confident that the original 188 hypotheses are reasonably well represented by 80 or so clusters. Armed with this knowledge, we can run the cleanup step and request an easily interpreted report for the formation of 80 clusters:

```
Shell> $SCHRODINGER/utilities/pharm_cluster_hypotheses -cleanup -report 80
```

Note that you can run the cleanup step as many times as you wish in order to examine reports for different clustering levels. And if you prefer not to go through the log file to determine the most pronounced drop in merging similarity, then you can simply specify a merging similarity that you feel reflects the point at which hypotheses are essentially equivalent. For example, you would specify '`-report 0.9`' if you want the merging of clusters to stop as soon as the merging similarity is less than or equal to 0.9. For the current example, this would be equivalent to '`-report 79`' because in merging 80 clusters to form 79, the merging similarity drops from 0.97529 to 0.86978. If you were interested in clusters that contain only geometrically identical hypotheses, then you would specify '`-report 1.0`'. This would be equivalent to '`-report 109`' because the merging similarity is 1.0 all the way down to the formation of 109 clusters. A subtle point to remember is that when you specify a merging similarity, the inequality condition is *less* *than or equal to*, so keep this in mind if the behavior seen for '`-report 0.9`' seems inconsistent with the behavior for '`-report 1.0`'.

When the cleanup step has finished, you are directed to examine the file `ClusterHypothesesData.tab`, whose contents are summarized as follows:

```
HypoID          Survival   Cluster   Size     AvgSim
----------------------------------------------------
AHHRRR_167       14.7935      1         6     0.98398
AHHRRR_163       14.7935      1         6     0.98398
AHHRRR_166       14.7642      1         6     0.98812
AHHRRR_170       14.7642      1         6     0.98812
AHHRRR_169       14.6857      1         6     0.99073
AHHRRR_165       14.6857      1         6     0.99073
----------------------------------------------------
AHHRRR_190       14.7943      2         3     0.98584
AHHRRR_189       14.7160      2         3     0.98895
AHHRRR_187       14.8228      2         3     0.98091
----------------------------------------------------
.
.
.
----------------------------------------------------
AADHRR_411       14.2506     76         2     0.97585
AADHRR_412       14.2217     76         2     0.97585
----------------------------------------------------
ADHHRR_168       14.8126     77         1     0.00000
----------------------------------------------------
ADHRRR_26        14.6824     78         1     0.00000
----------------------------------------------------
ADHRRR_211       14.6694     79         1     0.00000
----------------------------------------------------
AADHRR_552       14.3323     80         1     0.00000
```

From this clustering report you can readily identify instances where a large number of hypotheses are satisfactorily represented by a single hypothesis. In selecting a representative of a given cluster, you may wish to choose the hypothesis with the highest survival score, or the one that exhibits the highest average similarity to other members of its cluster, i.e., the hypothesis that best represents the cluster in an average sense.

To observe in more concrete terms exactly how similar the clustered hypotheses are, consider `AAAHRR_422` and `AAAHRR_424`, the two most dissimilar hypotheses in any single cluster (similarity = 0.97529). You can align `AAAHRR_422` onto `AAAHRR_424` by running the `align_hypoPair` utility as follows:

```
Shell> $SCHRODINGER/utilities/align_hypoPair -fixed hypotheses/AAAHRR_424 \
       -free hypotheses/AAAHRR_422 -new NEW
```

The utility finds four 6-point mappings, and by default creates a new hypothesis (with file prefix "`NEW`"), for the best one. You should see the following summary for the top-ranked match:

```
Match 1: RMSD = 0.0586, Number of sites matched = 6
```

```
 Fixed      Free                         New                     Deviation
------------------------------------------------------------------------
  0 A  <-->  0 A        0 A    5.4563   -1.4832    3.4750       0.0405
  1 A  <-->  1 A        1 A    3.5906   -1.9744   -0.1870       0.0496
  2 A  <-->  2 A        2 A    4.3793   -0.6104   -2.1598       0.0352
  7 H  <-->  7 H        7 H    7.6583   -1.4302    4.5317       0.0539
 10 R  <--> 10 R       10 R    2.5896    1.7310    0.3089       0.0472
 11 R  <--> 11 R       11 R    0.4479    0.7635   -0.4481       0.1008
```

Thus when `AAAHRR_422` is aligned onto `AAAHRR_424`, the deviations in the site point positions are quite tiny compared to the tolerances (typically 1-2 Å) normally used when searching a database for matches to a hypothesis. Consequently, essentially every molecule that matches `AAAHRR_422` would also match `AAAHRR_424`.

1.13 Building 3D QSAR Models

Please note: a basic overview of the Phase QSAR method is provided in this section. For a complete, technical description, the reader is referred to Appendix A. For details about the statistics of QSAR models, the user is referred to Appendix B. The use of *t*-value filters is discussed in Appendix C, and cross-validation techniques are covered in Appendix D

While scoring with respect to inactives provides an indication of the ability of hypotheses to discriminate between compounds with high and low activities, QSAR models tackle the problem across a continuous range of activities. In this section, we develop 3D QSAR models for all hypotheses at once, and we demonstrate how the most interesting models can be examined in detail.

Before creating a 3D QSAR model, the training set structures must first be aligned to the hypothesis of interest. An alignment is generated for each conformer that matches at least three sites in the hypothesis, using fitness (site, vector, volume) to determine the best alignment among all conformers. The aligned structures are then placed in a regular rectangular grid, which divides the space they occupy into uniformly-sized cubes. At this point, two varieties of models may be created: atom-based and pharmacophore-based.

In atom-based models, the full 3D chemical structure is considered, with atoms treated as van der Waals spheres. Each atom is placed into one of six categories:

D – Hydrogen bond donor
H – Hydrophobic/non-polar
N – Negative ionic
P – Positive ionic
W – Electron-withdrawing (includes hydrogen bond acceptors)
X – Miscellaneous (all other types of atoms)

Atoms are assigned to these categories using very simple rules, e.g., category D is assigned to hydrogens bonded to polar atoms (N, O, P, S); category H is assigned to carbons and halogens, and to hydrogens that are attached to carbons; category N is assigned to atoms carrying a negative ionic charge, etc. The rules are generally consistent with the default pharmacophore feature definitions, but there are some important differences. For example, the pharmacophore feature definitions use complex rules to identify hydrophobic regions, whereas atom-based QSAR does not. Pharmacophore feature definitions can treat a given atom as part of two different pharmacophore sites, e.g., the nitrogen in a pyridine can be both an acceptor and part of an aromatic ring. Atom-based QSAR requires that each atom be assigned to only one category.

In sharp contrast, pharmacophore-based QSAR models are concerned only with the sites on each structure that can be matched to the hypothesis. Each site is treated as a sphere, whose radius is user-adjustable. These are the same sites that are created when pharmacophore models are developed, so the site categories are A, D, H, N, P, R (and X, Y, Z for custom features). Pharmacophore-based models offer the advantage of not being as sensitive to the quality of the overall structural superpositions. Hence the models generated have greater applicability to compounds from diverse chemical families. However, pharmacophore-based models cannot account for factors beyond the pharmacophore model itself, such as possible steric clashes with the receptor. This requires consideration of the entire molecular structure, i.e., an atom-based model.

Whether the model is atom-based or pharmacophore-based, each structure is represented as a bit string, where each bit is associated with both a cube in the grid and a category (atom or site). A particular bit is set by a structure if it contains an atom/site that occupies the associated cube, and that atom/site is a member of the category associated with that bit. The bit string is simply a collection of zeros and ones, and these are treated as independent variables in the QSAR model. Because the number of bits is typically much larger than the number of training set compounds, the system is said to be highly *underdetermined*. For this reason, Phase QSAR models are created by applying partial least-squares (PLS) regression to this large collection of binary-valued variables.

The choice of which type of model to create usually comes down to whether or not the training set compounds are sufficiently rigid and congeneric. If the structures contain a relatively small number of rotatable bonds and some common structural framework, then an atom-based model may work quite well. However, if the compounds are highly flexible or if they exhibit significant chemical diversity, a pharmacophore-based model may be more appropriate. If the choice is still not clear, then it is easy enough to create both types of models and examine the test set statistics to see which approach produces models with the most predictive power. In this tutorial, we are working with fairly rigid, congeneric structures, so we shall be creating atom-based models.

QSAR jobs are setup using `pharm_build_qsar`, so type the name of that utility with no arguments to see the available options:

```
Shell> $SCHRODINGER/utilities/pharm_build_qsar
```

Observe that in addition to the model type (`atom` vs. `pharm`), we can specify the size of the grid on which models are built, the maximum number of PLS factors in each model, parameters to control the elimination of independent variables based on *t*-value (see Appendix C), and whether or not to consider MacroModel atom types when computing volume scores. By considering atom types, the alignments selected will favor superposition of chemically similar atoms.

We will be using defaults for the model type (`atom`) and the grid spacing (1.0 Å), a maximum of 3 PLS factors, and we will retain only the independent variables with an absolute *t*-value of at least 2.0. Accordingly, setup the job as follows:

```
Shell> $SCHRODINGER/utilities/pharm_build_qsar -setup -factors 3 \
       -tvalue 2.0 -exclude 3 -rand 123456789
```

The option '`-exclude 3`' indicates that we want to estimate *t*-values by holding out random sets of 3 compounds from the training set, and '`-rand 123456789`' is the random seed for this process.

This command creates the main input file `build_qsar_multiQsar.inp`, whose contents are summarized as follows:

```
modelType=atom
numTrain=27
gridSpacing=1
maxFactors=3
tvalueFilter=2
tvalueExclude=3
tvalueSeed=123456789
actProperty=r_phase_Ligand_Activity
useVolumeGroups=false
gzipAlignments=true
phaseOptionsFile=build_qsar_phase.inp
ligandArchive=build_qsar_ligandFiles.tar
ligandDir=.ligands.tmp
hypoArchive=build_qsar_hypoFiles.tar
hypoDir=.hypotheses.tmp
hypoID=DHHRRR_49
hypoID=DHHRRR_64
hypoID=DHHRRR_67
.
.
.
hypoID=AAADRR_38
hypoID=AAADHH_130
hypoID=AAADHH_46
featureFile=build_qsar_feature.ini
tableFile=BuildQsarData.tab
csvFile=BuildQsarData.csv
resultDir=BuildQsarResults
resultArchive=build_qsar_results.tar
```

Aside from `modelType`, `gridSpacing`, `maxFactors`, and `useVolumeGroups` (which is `false` because we did not use the `-atomVol` flag), you should not modify any parameter values. You may, however, delete `hypoID` records to eliminate hypotheses for which you do not wish to build QSAR models. For now, though, leave the file as is.

Note that if we had chosen to create pharmacophore-based models, a file named `build_qsar_feature.rad` would also have been created, with an adjustable radius for each possible pharmacophore feature type:

```
A 1.0
D 1.0
H 1.0
N 1.0
P 1.0
Q 1.0
R 1.0
X 1.0
Y 1.0
Z 1.0
```

However, since we are creating atom-based models, there should be no such file. We leave it as an exercise for the user to create an alternate set of pharmacophore-based models.

Submit the multiple QSAR job as follows:

```
Shell> $SCHRODINGER/phase_multiQsar build_qsar
```

This job requires about 4 minutes on an Intel dual core @1.86 GHz. You may monitor the job's progress by examining the contents of the file `build_qsar_multiQsar.log`. When the job is finished, you should see the following sort of output at the end of the log file:

```
Building QSAR model for AAADRR_38 . . .
Building QSAR model for AAADHH_130 . . .
Building QSAR model for AAADHH_46 . . .

Creating resultArchive build_qsar_results.tar . . .

CPU time = 243.94 sec

phase_multiQsar successfully completed

Driver script for parent phase_multiQsar job build_qsar finished
Current time: Mon Jan 27 14:08:02 2014
Elapsed time = 00:04:08

phase_multiQsar results for build_qsar are complete
```

Now run the cleanup step:

```
Shell> $SCHRODINGER/utilities/pharm_build_qsar -cleanup
```

Use a text editor to examine the file `BuildQsarData.tab`. You will see a number of statistical quantities (please see Appendix B for further details):

SD	– Standard deviation of regression. An effective RMS error in the training set fit, corrected for the degrees of freedom in the model.

R^2	–	Coefficient of determination. Measures the fraction of the variance in the training set activity data that's accounted for by the model.
F	–	Variance ratio statistic for the model. Larger values indicate greater statistical significance.
P	–	Significance level of F. Smaller values indicate greater confidence.
R^2-CV	–	Cross-validated R^2. See Appendix D for more details.
Stability	–	Stability of model predictions to changes in the training set. See Appendix D for more details.
RMSE	–	RMS error in the test set predictions.
Q^2	–	Analogous to R^2, but computed with respect to the test set predictions. Will be negative if the average squared error is larger than the variance in the test set activity data.
R-Pearson	–	Pearson correlation coefficient computed between the predicted and observed test set activities. Ranges from –1 to 1.

Ideally, a model should yield a large F value and comparable prediction errors for both the training and test sets, i.e., SD ≈ RMSE. Comparing R^2 and Q^2 is not as meaningful since the test set may not span as large a range of activities as the training set, leading to small (and sometimes negative) Q^2 values, even if the predictions are relatively accurate. Moreover, the test set activities may be accurately predicted in a relative sense, but a slight systematic error can drastically reduce Q^2. R-Pearson provides a better measure of relative accuracy.

Overall, it's important to recognize that there is no single statistic that unequivocally tells us which model is best. The battery of statistics should be considered in totality, and some measure of common sense must be applied. For example, IC_{50} values may only be accurate to a multiplicative factor of 2, so the corresponding –log[IC50] values would only be accurate to ±log(2). Therefore, if the SD statistic is smaller than this experimental uncertainty, the data are clearly being over-fit, and the model is bound to yield spurious predictions on certain compounds outside the training set, even if the test set predictions appear satisfactory. Furthermore, it is difficult to justify fitting the Y values to an accuracy that exceeds fluctuations in predictions caused by small changes in the training set. Accordingly, as PLS factors are added to the model, one should become suspicious if the R^2 value overtakes and exceeds the Stability by a non-trivial amount.

To illustrate how one might decide which models are superior, consider the results for `ADHHRR_168`. When three PLS factors are used, there is a very tight fit of the training set data, and the F value of 108.6 indicates a high level of statistical significance compared to most other models. However, the test set RMSE is about 50% larger than the training set SD, which results in an unimpressive Q^2 value of 0.3727. The Pearson-R value of 0.6729 suggests modest, but not strong, relative accuracy. There is really only one obvious clue from the training set statistics that the 3-factor model might not perform well, and that is the fact that R^2 of 0.9341 is noticeably higher than the Stability of 0.8746.

By contrast, consider the 3-factor model for `DHHRRR_1`. Here, the training set is well fit, the test set RMSE is essentially the same as the training set SD, and Q^2 and Pearson-R indicate satisfactory accuracy in both absolute and relative senses. Accordingly, we would view this model as superior to the previous one. Consistent with this observation is the fact that the 3-factor R^2 of 0.9172 is essentially equal to the 3-factor Stability of 0.9179.

The summary provided in `BuildQsarData.tab` is very helpful for identifying the most promising models. But no doubt you will want more information about certain models. In the subdirectory `BuildQsarResults/` you will find a number of files for each hypothesis:

`<`*hypoID*`>_align.maegz`	– Aligned structures for training and test set compounds. Training set compounds appear first.
`<`*hypoID*`>.def`	– Feature definitions.
`<`*hypoID*`>.mae`	– Reference ligand structure.
`<`*hypoID*`>_order.dat`	– File that defines the overall order of compounds in `<`*hypoID*`>_align.mae`.
`<`*hypoID*`>.qsar`	– QSAR model file.
`<`*hypoID*`>_qsar.inp`	– Main input file for a `phase_qsar` job.
`<`*hypoID*`>.tab`	– Primary hypothesis data, with QSAR model flag activated.
`<`*hypoID*`>.xyz`	– Hypothesis site coordinates.

The Phase GUI will recognize the primary hypothesis file `<`*hypoID*`>.xyz` and database searches conducted with any of these hypotheses will automatically apply the associated QSAR model to the hits.

More importantly, this directory contains everything you will need in order to run a `phase_qsar` job for any particular hypothesis. `phase_qsar` will output complete details about the model, along with predictions for the training and test set compounds. Accordingly, move to the `BuildQsarResults/` directory:

```
Shell> cd BuildQsarResults
```

Use a text editor to examine the input file `DHHRRR_1_qsar.inp`. You will see a number of options, which you need not change, followed by information about the indexing of ligands. The training set compounds are placed before the test set compounds, and you may wish to refer to these indices when you examine out output from `phase_qsar`.

While still in the `BuildQsarResults/` directory, submit the `phase_qsar` job as follows:

```
Shell> $SCHRODINGER/phase_qsar DHHRRR_1
```

This job should take only 2-3 seconds. You may verify that it has finished by examining the file `DHHRRR_1_qsar.log`, which should contain the values of various environment variables followed by:

```
phase_qsar successfully completed
```

Now use a text editor to examine the file `DHHRRR_1_qsar.out`. You will find complete model statistics, the Cartesian coordinates and regression coefficient for each “bit” in the model, training and test set predictions, and the actual bit values for each compound. The predictions and bits are also written in comma-separated format to the files `DHHRRR_1_pred.csv` and `DHHRRR_1_bits.csv`, respectively. Thus you may readily load the predictions into a spreadsheet program to create a scatter plot, such as in Figure 1, and you may create models outside the framework of Phase that utilize the Phase QSAR volume bits. If you have an installation of Canvas, you can in fact use `$SCHRODINGER/utilities/canvasPLS` to build a QSAR model that's identical to the one produced by Phase. Or, you can combine the volume bits with other independent variables, such as LogP, to incorporate additional effects into the QSAR model.

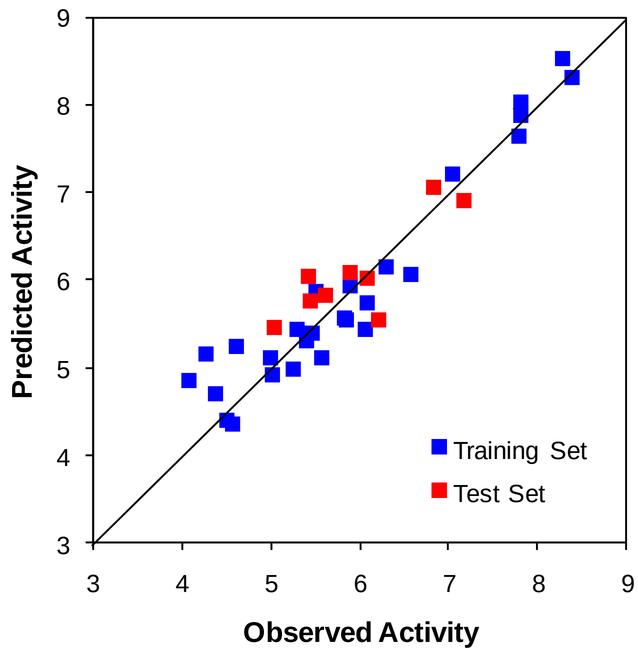

Figure 1. Scatter plot of predicted vs. observed activities generated by the atom-based QSAR model created from hypothesis DHHRRR_1.

Before proceeding to the next section, remember to return to the root project directory:

```
Shell> cd ..
```

1.14 Visualizing QSAR Models

If you are running the Schrödinger software on a Linux machine with xterm emulation, then you should be able to run the utility `qsarVis` to visualize QSAR models that you create in command line projects. You may invoke the `qsarVis` utility with `-help` to see the available options:

```
Shell> $SCHRODINGER/utilities/qsarVis -help
```

Observe that you must supply the name of the hypothesis used to create the QSAR model, and the name of a single compound in the project. By default, you will be viewing the bits set by that structure, for which the associated regression coefficients exceed applicable positive and

negative thresholds. This allows you to identify favorable (blue) and unfavorable (red) regions of the structure, and to control whether you see weak, moderate or strong effects (larger
We first visualize the 3-factor QSAR model for hypothesis `DHHRRR_1`, displaying the bits set by its reference ligand, `mol_8`, which happens to be the most active compound in the data set. We will specify regression coefficient tolerances of –0.03 and +0.03 because these values are intermediate between the extreme negative and positive regression coefficients for the 3-factor model when effects from all atom classes are considered. You may examine the data under "`PLS Regression Coefficient Limits`" in the file `BuildQsarResults/DHHRRR_1_qsar.out` to see the various limits for the current hypothesis.

While in the project directory, issue the following command from an xterm window:

```
Shell> $SCHRODINGER/utilities/qsarVis -hyp BuildQsarResults/DHHRRR_1 \
       -mol mol_8 -nc -0.03 -pc 0.03 -npls 3
```

This should launch a new graphics window entitled "Visualization Tookit – OpenGL," with an interactive 3D image of the ligand, hypothesis, and QSAR model. To rotate the image, use left-click-drag; to translate, use middle-click-drag; to zoom, use right-click-drag. As noted previously, the blue regions of the structure imply favorable interactions with the receptor, whereas the red regions imply unfavorable interactions. As expected, favorable interactions predominate for this highly active compound. Figure 2 provides a sample view of the image.

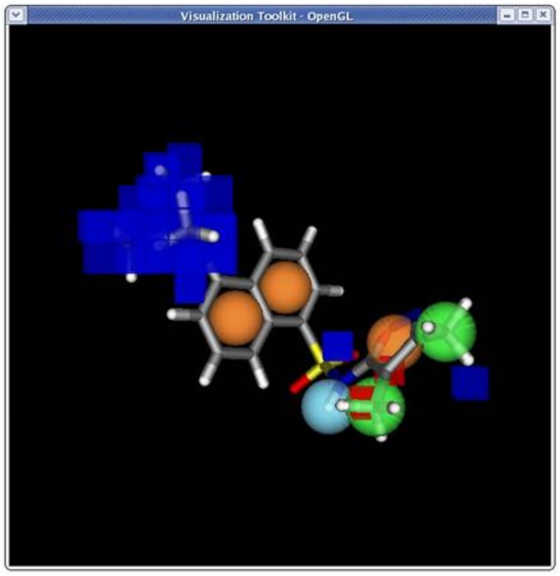

Figure 2. Sample view of the DHHRRR_1 QSAR model, with bits set by the strongly active ligand mol_8. Effects from all atoms classes are represented.

In contrast, we can visualize the model in the context of the least active compound as follows:

```
Shell> $SCHRODINGER/utilities/qsarVis -hyp BuildQsarResults/DHHRRR_1 \
       -mol mol_20 -nc -0.025 -pc 0.025 -npls 3
```

Observe that the favorable interactions are now far less prevalent than they were for `mol_8`, Figure 3.

So far, we have been visualizing the combined effects of all atom types, but we can restrict the view to show only the bits set by a particular class of atom. If, for example, we were interested in seeing only the effects attributed to electron-withdrawing atoms, then we could launch the visualizer as follows:

```
Shell> $SCHRODINGER/utilities/qsarVis -hyp BuildQsarResults/DHHRRR_1 \
       -mol mol_8 -class W -nc -0.01 -pc 0.01 -npls 3
```

Observe that the occupied volumes are now restricted to regions around oxygens and nitrogens, Figure 4.

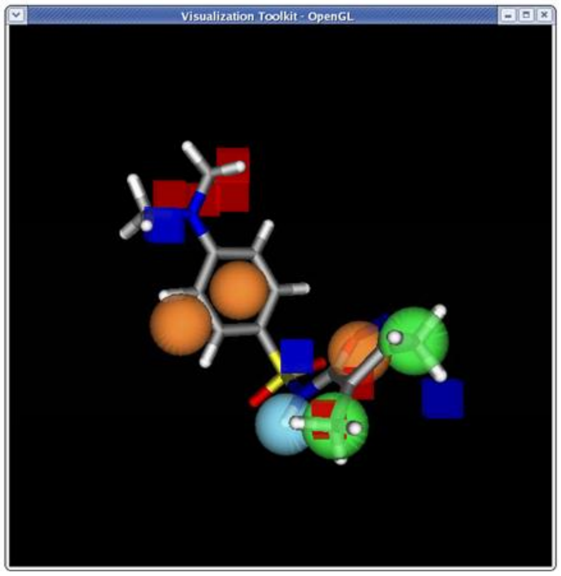

Figure 3. Sample view of the DHHRRR_1 QSAR model, with bits set by the inactive mol_20. Effects from all atoms classes are represented.

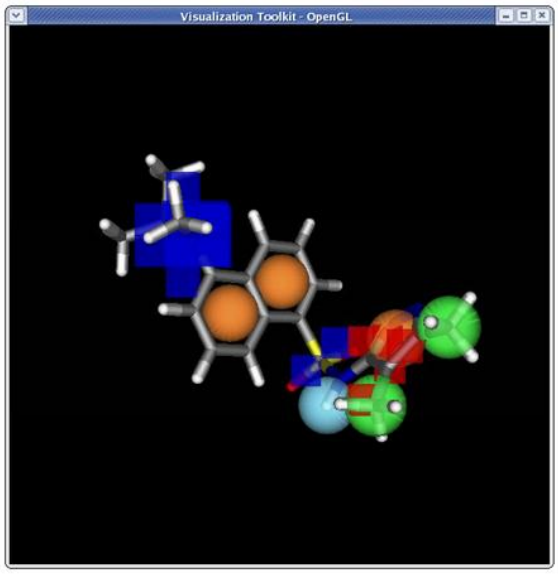

Figure 4. Sample view of the DHHRRR_1 QSAR model, with bits set by the strongly active ligand mol_8. Effects from electron-withdrawing atoms are represented.

1.15 Applying Models to New Compounds

While hypotheses and QSAR models can be used to align/predict compounds that are stored in a Phase 3D database, you may not want to go to the trouble of creating a database if you simply want to align/predict relatively small number of compounds that you didn't include in the original project. In this section, we show how you can perform a *file-based* search on a set of new compounds, which may or may not be represented by multiple conformers.

File-based searches are performed by running `phase_find_matches`, so type the name of that program with no arguments to see the available options:

```
Shell> $SCHRODINGER/phase_find_matches
```

Observe that you can search structures in one or more Maestro/SD files, a Phase database, or a Phase command line project. In this section we work with a Maestro file; alignment of project ligands is covered in Section 1.16, in conjunction with the creation of excluded volumes.

We first demonstrate the case where each of the compounds is already represented by multiple low-energy conformers. While in the root project directory, issue following command to launch the file-based searching job:

```
Shell> $SCHRODINGER/phase_find_matches ../userFiles/endo.mae.gz \
       BuildQsarResults/DHHRRR_1 align_confs -match 3 –nosort -NOCHECKPOINT
```

Note that we are simply searching the Maestro file we used to create the project initially, but it could just as well be any multi-conformer Maestro file. We are requesting matches to the hypothesis involving at least three of the six sites and since we did not request multiple hits per molecule (`-report <n>`), each molecule searched will produce no more than one hit. The use of `-nosort` ensures that hits will be returned in the order the corresponding structures are encountered in the Maestro file being searched.

This job requires only a few seconds on an Intel dual core @1.86 GHz. You may monitor the job's progress by examining the contents of the file `align_confs.log`. When the job is finished, the file `align_confs.okay` will appear in the current directory, and you will see the following sort of output at the end of `align_confs.log`:

```
.
.
.
Concatenating hits from completed subjobs

Concatenating hits from align_confs_sub_0-hits.maegz to align_confs-
hits.maegz . . .
Number of hits concatenated so far = 36

Hits from completed subjobs successfully combined

All subjobs successfully completed
CPU time = 0.08 sec

phase_find_matches successfully completed
```

```
Driver script for phase_find_matches job align_confs finished
Current time: Mon Jan 27 14:52:59 2014
Elapsed time = 00:00:03
phase_find_matches results for align_confs are complete
```

To view the aligned hits, import the file `align_confs-hits.maegz` into Maestro. Upon doing so, the Project Table will contain a number of properties created during the search, including predictions from the QSAR model.

You may also run the utility `phase_qsar_stats` to get a summary of the QSAR predictions in the hit file just created, and to create a comma-separated file that can be imported into a spreadsheet program to create a scatter plot:

```
Shell> $SCHRODINGER/utilities/phase_qsar_stats -hypo BuildQsarResults/DHHRRR_1 \
       -hits align_confs-hits.maegz -act r_user_Activity –plot align_confs.csv
```

The following output should be written to the screen:

```
Model Summary
-------------

Model Type = "atom"
Number of PLS Factors = 3
Grid Spacing = 1.00

Grid Limits:

        Min       Max
 x   -11.000    16.000
 y   -11.000    14.000
 z    -9.000    12.000

PLS Regression Statistics:

#Factors     S.D.      R^2     R^2-CV  Stability    F        P
    1       0.5856    0.7948    0.7020    0.9724    96.8   4.429e-10
    2       0.4467    0.8854    0.7395    0.9387    92.7   5.130e-12
    3       0.3879    0.9172    0.7484    0.9179    84.9   1.374e-12

PLS Regression Coefficients and T-Values:

                      T(#Factors in Regression)
Factor    Coeff       T(1)      T(2)      T(3)
   1      0.3272     9.841    12.902    14.857
   2      0.1862               4.356     5.016
   3      0.0991                         2.971


PLS Regression Coefficient Limits

Class   #Factors      Min         Max
----------------------------------------
  D         1     -2.633e-02   2.431e-02
```

```
           2     -1.756e-02   3.647e-02
           3     -2.085e-02   6.291e-02
----------------------------------------
  H        1     -3.103e-02   3.845e-02
           2     -5.093e-02   5.328e-02
           3     -7.617e-02   5.593e-02
----------------------------------------
  W        1     -3.437e-02   3.848e-02
           2     -2.436e-02   5.407e-02
           3     -3.221e-02   5.741e-02
----------------------------------------
  X        1     -2.431e-02   2.431e-02
           2     -1.402e-02   2.014e-02
           3     -1.050e-02   2.948e-02
----------------------------------------
 All       1     -3.437e-02   5.850e-02
           2     -4.125e-02   7.960e-02
           3     -4.724e-02   9.864e-02
----------------------------------------


Activity Predictions
--------------------

Number of molecules with experimental activities = 36
Number of molecules predicted = 36

#Factors  Variance(Exp)  Variance(Pred)     RMSE       Q^2     R-Pearson
---------------------------------------------------------------------------
    1        1.2673          1.0496        0.5734    0.7405      0.8630
    2        1.2673          1.1194        0.4149    0.8641      0.9304
    3        1.2673          1.1405        0.3640    0.8954      0.9466

Activity_Exp  Activity_Pred(1)  Activity_Pred(2)  Activity_Pred(3)
-----------------------------------------------------------------
   5.5090          6.3933            6.0474            5.8623
   5.4560          6.2736            5.8760            5.7591
   5.4690          6.2586            5.6856            5.3915
.
.
.
   4.2760          5.3698            4.9678            5.1445
   6.5850          5.6935            5.6012            6.0497
   6.2920          5.6917            5.6515            6.1465
```

The QSAR model statistics and activity predictions are identical to those created by `phase_qsar` in the previous section, although note that `phase_qsar_stats` treats all hits as if they were test set molecules. You should find the same activity predictions in the file `align_confs.csv`:

```
Title,Activity_Exp,Activity_Pred(1),Activity_Pred(2),Activity_Pred(3)
"endo-1",5.5090,6.3933,6.0474,5.8623
"endo-2",5.4560,6.2736,5.8760,5.7591
"endo-3",5.4690,6.2586,5.6856,5.3915
.
.
.
```

```
"endo-34",4.2760,5.3698,4.9678,5.1445
"endo-35",6.5850,5.6935,5.6012,6.0497
"endo-36",6.2920,5.6917,5.6515,6.1465
```

If you wish to search a file that contains only one conformer per molecule, you can request that *flexible* searching be done, where conformers are generated on-the-fly, using a rapid torsional sampling technique combined with a soft non-bonded potential. Although these conformers are not fully minimized, they are satisfactory for identifying whether or not a molecule can match a hypothesis. If reliable activity predictions are sought, however, it is recommended that fully minimized conformers be created using MacroModel, in a manner consistent with the original conformers from which the pharmacophore and QSAR models were developed.

We demonstrate flexible file-based searching using the same 36 endothelin ligands, starting with a single low-energy structure for each molecule, and generating a maximum of 100 conformers per molecule. Further, we will require that each molecule match all six sites in the hypothesis:

```
Shell> $SCHRODINGER/phase_find_matches ../userFiles/endo_1conf.mae.gz \
       BuildQsarResults/DHHRRR_1 align_flex -flex -max 100 -match 6 \
       -nosort -NOCHECKPOINT
```

This job requires only a few seconds on Intel dual core @1.86 GHz. You may monitor the job's progress by examining the contents of the file `align_flex.log`. When the job is finished, the file `align_flex.okay` will appear in the current directory, and you should see the following sort of output at the end of the log file:

```
.
.
.
Concatenating hits from completed subjobs

Concatenating hits from align_flex_sub_0-hits.maegz to align_flex-hits.maegz
. . .
Number of hits concatenated so far = 24

Hits from completed subjobs successfully combined

All subjobs successfully completed
CPU time = 0.06 sec

phase_find_matches successfully completed

Driver script for phase_find_matches job align_flex finished
Current time: Mon Jan 27 14:59:12 2014
Elapsed time = 00:00:08
phase_find_matches results for align_flex are complete
```

1.16 Creating Excluded Volumes

A molecule may satisfy a pharmacophore model, yet fail to bind to the associated receptor because doing so would result in a destabilizing close contact between the molecule and the receptor. Excluded volumes provide a means of restricting the space that can be occupied by any molecule that matches a pharmacophore model, thereby incorporating steric constraints imposed by the receptor. There are three utilities that allow you to create excluded volumes in an automated fashion, using varying amounts of ligand and/or receptor information:

`create_xvolShell`	– Creates a shell of excluded volume spheres around one or more ligands. Provides a means of defining shape-based queries for database searching.[4]
`create_xvolClash`	– Creates excluded volumes using actives and inactives that have been aligned to a hypothesis. Excluded volumes are placed in locations that would cause steric clashes only for the inactives.
`create_xvolReceptor`	– Creates excluded volumes using a receptor structure or a portion thereof.

We start off with `create_xvolShell`, so type the name of that utility with no arguments to see the available options:

```
Shell> $SCHRODINGER/utilities/create_xvolShell
```

You would normally run this utility if you believe that one or more ligands effectively sweep out the space of the binding pocket. In other words, if a molecule that's aligned to the hypothesis cannot fit into the shape defined by the excluded volume shell, then it is presumed that the molecule would experience a steric clash with the receptor.

Because the shell typically obscures the view of the ligand(s) it surrounds, it's a good idea to start off by using the `-cut` flag to create only half the shell. This allows you to verify that the shell is in fact being created the way you expected, after which you can create the full shell. Accordingly, while in the project directory, issue the following command to create half a shell around the reference ligand from the `DHHRRR_1` hypothesis:

```
Shell> $SCHRODINGER/utilities/create_xvolShell \
       -hypo BuildQsarResults/DHHRRR_1 -buff 2.0 -cut
```

You should see a message indicating that the excluded volumes file `BuildQsarResults/DHHRRR_1.xvol` has been created. To view the excluded volumes, simply import this hypothesis into the Manage Pharmacophore Hypotheses workflow of the Phase graphical interface. Figure 5 contains a sample view.

If you choose the CPK molecular representation for the ligand with CPK percentage equal to 100, you will see that the surfaces of all heavy atoms in the ligand are at least 2 Å from the surface of any excluded volume sphere.

Now go back and create the full excluded volume shell:

[4] Users may also wish to explore the use of `$SCHRODINGER/phase_shape`, which performs bona fide shape-based alignment and searching (i.e., a pharmacophore hypothesis is not used).

```
Shell> $SCHRODINGER/utilities/create_xvolShell \
       -hypo BuildQsarResults/DHHRRR_1 -buff 2.0
```

This hypothesis could now be used to search a database or a structure file for matches to the hypothesis, with the restriction that the aligned structures fit within the shell. If you wish to exercise additional control over the shape of each match, you can search using the `volumeCutoff` option (Section 1.15) to eliminate matches that fit within the shell, but which you would consider to be too small relative to the reference ligand. For example, setting `volumeCutoff` equal to 0.9 would force all matches to overlap 90% with the reference ligand.

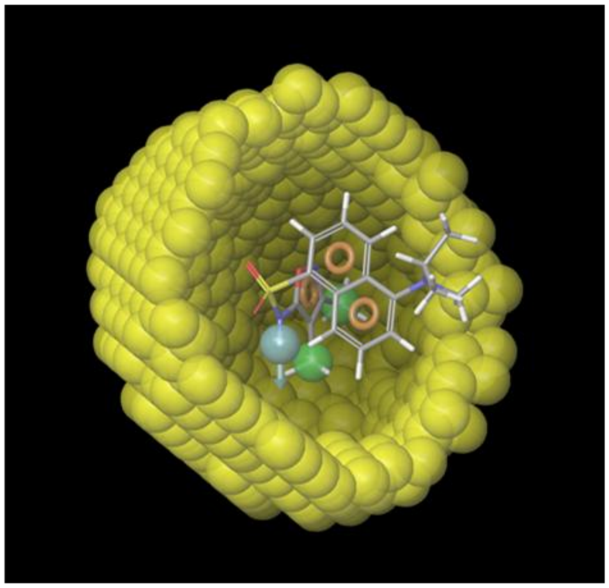

Figure 5. Sample view of the excluded volume shell created for the DHHRRR_1 hypothesis using create_xvolShell with the -cut flag.

Next we create excluded volumes using information from both actives and inactives. The utility `create_xvolClash` automatically places excluded volume spheres in positions that would create clashes only for the inactives. To see the available options, type the name of the utility with no arguments:

```
Shell> $SCHRODINGER/utilities/create_xvolClash
```

Notice that the '`-freq` *`<minClash>`*' option allows you to incorporate a certain level of confidence in the excluded volumes that are created. By increasing the *`<minClash>`* parameter beyond the default of one, an excluded volume sphere will be created in a given location only if it would cause a clash for more than one inactive.

Before we can run the `create_xvolClash` utility, we need to align our actives and inactives to a hypothesis, using and `phase_find_matches`. Accordingly, we must create files that contain `LIGAND_NAME` records for just those molecules:

```
Shell> cp ProjectLigands.inp actives.inp
Shell> cp ProjectLigands.inp inactives.inp
```

Use a text editor to open the file `actives.inp` and delete all `LIGAND_NAME` records for which `PHARM_SET` is not “`active`”. Upon doing so, the contents of the file should be:

```
###############################################################################################
#                                                                                             #
#           Ligand Records File. All text following a pound sign "#" is ignored.            #
#                                                                                             #
###############################################################################################
LIGAND_DIR = ligands    #  PHARM_SET  LIGAND_GROUP  MUST_MATCH  QSAR_SET  ACTIVITY  1D_VALUE  #
###############################################################################################
LIGAND_NAME = mol_5     #  active     5             false       train     7.824     0.0       #
LIGAND_NAME = mol_8     #  active     8             false       train     8.398     0.0       #
LIGAND_NAME = mol_9     #  active     9             false       train     8.301     0.0       #
LIGAND_NAME = mol_16    #  active     16            false       train     7.824     0.0       #
LIGAND_NAME = mol_17    #  active     17            false       train     7.796     0.0       #
```

Similarly, edit the file `inactives.inp` and delete all `LIGAND_NAME` records for which `PHARM_SET` is not “`inactive`”. The contents of the file should be:

```
###############################################################################################
#                                                                                             #
#           Ligand Records File. All text following a pound sign "#" is ignored.            #
#                                                                                             #
###############################################################################################
LIGAND_DIR = ligands    #  PHARM_SET  LIGAND_GROUP  MUST_MATCH  QSAR_SET  ACTIVITY  1D_VALUE  #
###############################################################################################
LIGAND_NAME = mol_20    #  inactive   20            false       train     4.076     0.0       #
LIGAND_NAME = mol_23    #  inactive   23            false       train     4.62      0.0       #
LIGAND_NAME = mol_29    #  inactive   29            false       train     4.377     0.0       #
LIGAND_NAME = mol_31    #  inactive   31            false       train     4.502     0.0       #
LIGAND_NAME = mol_32    #  inactive   32            false       train     4.561     0.0       #
LIGAND_NAME = mol_34    #  inactive   34            false       train     4.276     0.0       #
```

Before proceeding you should consider whether or not there are any obvious inconsistencies in assuming that all of these compounds are inactive because of steric clashes. We will be creating excluded volumes for hypothesis `DHHRRR_1`, and according to the file-based searching results from Section 1.15, all of the inactives except `mol_20 and mol_34` match 6 out of 6 sites in this hypothesis. So, assuming that `DHHRRR_1` adequately embodies the characteristics required for ligand binding, there is some question as to whether these two compounds are inactive because of steric clashes or for other reasons, such as poor solubility or lack of a critical interaction with the receptor. But note that `mol_20` matches the same 5 out of 6 sites as `mol_18`, with almost identical fitness, yet the activity of `mol_18` is 6.097, two full orders of magnitude higher. Hence it would appear that additional factors are contributing to the inactivity of `mol_20`, possibly a steric clash involving the N,N-dimethyl substituent on the phenyl ring. So we shall retain it for purposes of creating excluded volumes. A similar analysis for `mol_34` does not provide as strong of an argument for retaining it. This molecule matches only 4 out of 6 sites, which may partially explain its low activity. It’s also one of the least bulky molecules, occupying very little space beyond that of larger, more active molecules such as `mol_8`. Nevertheless, we shall retain `mol_34` for this exercise, keeping in mind that it might not provide any useful information about excluded volumes.

Recall that we have already created excluded volumes for the `DHHRRR_1` hypothesis, so we should remove the associated file before we run `phase_find_matches`:

```
Shell> rm BuildQsarResults/DHHRRR_1.xvol
```

This will prevent those excluded volumes from being applied when we create alignments for the actives and inactives.

We will be searching the command line project directly, so we need to specify the absolute path to the files `actives.inp` and `inactives.inp` when running `phase_find_matches`. Accordingly, while in the project directory, define the environmental variable `TPATH` as follows:

csh/tcsh:
```
Shell> setenv TPATH `pwd`
```

sh/bash/ksh:
```
Shell> export TPATH=`pwd`
```

Then launch the job to align the actives as follows:

```
Shell> $SCHRODINGER/phase_find_matches $TPATH/actives.inp \
       BuildQsarResults/DHHRRR_1 align_actives -match 6 \
       -nosort -NOCHECKPOINT
```

Now launch the job to align inactives as follows:

```
Shell> $SCHRODINGER/phase_find_matches $TPATH/inactives.inp \
       BuildQsarResults/DHHRRR_1 align_inactives -match 3 \
       -nosort -NOCHECKPOINT
```

In this case, we are requiring that only 3 of 6 sites be matched, although we know from previous work that "`–match 4`" would achieve the same results.

After both jobs complete, we will have two hit files, `align_actives-hits.maegz` and `align_inactives-hits.maegz`, which we may now use as input to `create_xvolClash`. Accordingly, issue the following command to create excluded volumes that will clash only with the inactives:

```
Shell> $SCHRODINGER/utilities/create_xvolClash \
       -hypo BuildQsarResults/DHHRRR_1 \
       -pos align_actives-hits.maegz \
       -neg align_inactives-hits.maegz -buff 2.0
```

The following output should be written to the screen:

```
Number of actives = 5
Number of inactives = 6
Total number of excluded volume spheres created = 25

Creating excluded volumes file BuildQsarResults/DHHRRR_1.xvol

create_xvolClash successfully completed
```

You may import the `DHHRRR_1` hypothesis into the Manage Pharmacophore Hypotheses workflow of the Phase graphical interface to view the excluded volumes. Figure 6 contains a sample view. You may also import the active and inactive hit files as well, to verify that excluded volume violations would occur only for the inactives.

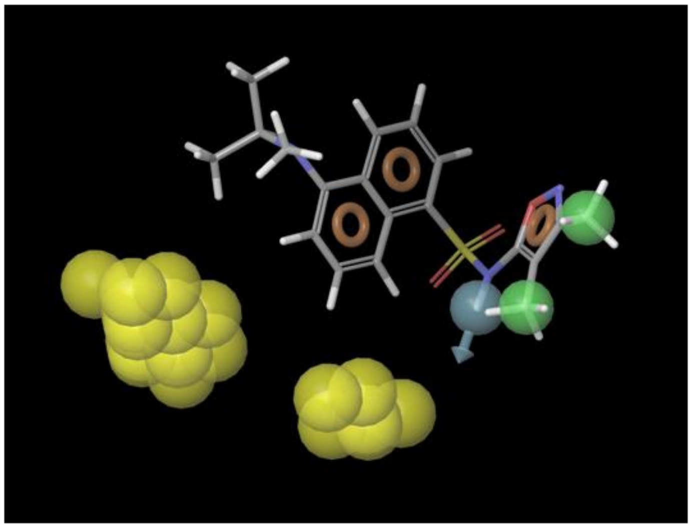

Figure 6. Sample view of the excluded volumes created for the DHHRRR_1 hypothesis using create_xvolClash.

As a final example in this section, we demonstrate how to create excluded volumes from the atoms in a receptor structure. This would normally be done in conjunction with a hypothesis that's been created from a ligand bound to the receptor, using the Manage Pharmacophore Hypotheses workflow of the Phase graphical interface.

Receptor-based excluded volumes are created by way of `create_xvolReceptor`, so type the name of that utility with no arguments to see the available options:

```
Shell> $SCHRODINGER/utilities/create_xvolReceptor
```

Observe that you must supply a receptor structure, or a portion thereof, in a Maestro file. Be aware that the receptor structure you provide *should not* contain a bound ligand, because the utility will consider all atoms in the file when creating excluded volumes.

In the directory `../userFiles`, you should find the following files derived from a co-crystallized complex of p38 MAP kinase:

`1M7Q_hypo.def`	–	Hypothesis feature definitions.
`1M7Q_hypo.mae`	–	Reference ligand structure (i.e., the crystal structure for the p38 inhibitor).
`1M7Q_hypo.tab`	–	Primary hypothesis data.
`1M7Q_hypo.xyz`	–	Hypothesis site coordinates.
`1M7Q_protein.mae`	–	p38 MAP kinase crystal structure (does not contain bound inhibitor).

While in the project directory, issue the following command to create the receptor-based excluded volumes:

```
Shell> $SCHRODINGER/utilities/create_xvolReceptor -hypo ../userFiles/1M7Q_hypo \
       -receptor ../userFiles/1M7Q_protein.mae -buff 2 -limit 5
```

In addition to a 2 Å buffer, we are limiting the thickness of the excluded volume shell by considering p38 atoms that are no more than 5 Å from the reference ligand.

Upon completion, the file `../userFiles/1M7Q_hypo.xvol` is created with a total of 46 excluded volume spheres. You may now import `1M7Q_hypo` into the Manage Pharmacophore Hypotheses workflow of the Phase graphical interface to view the excluded volumes. Figure 7 contains a sample view.

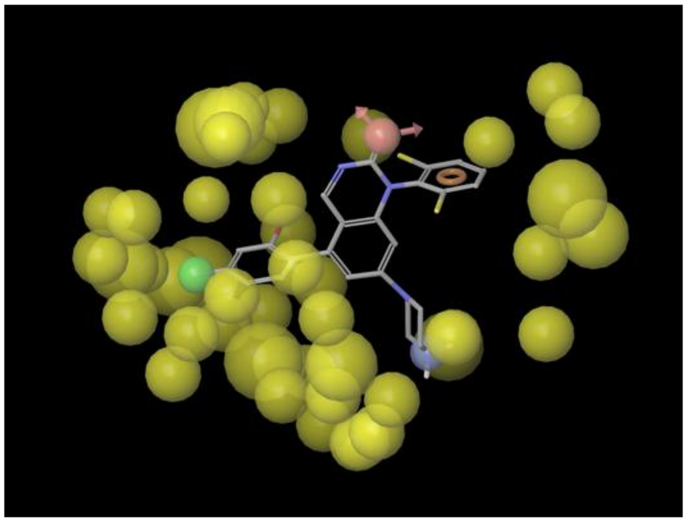

Figure 7. Sample view of the excluded volumes created for the p38 hypothesis 1M7Q_hypo using create_xvolReceptor.

Feel free to import the receptor structure as well to verify that the excluded volume spheres coincide with the positions of the receptor atoms.

1.17 Running Jobs on Multiple Processors

The top-level programs `phase_partition`, `phase_multiPartition`, `phase_scoring`, `phase_inactive` and `phase_multiQsar` may be run as multiple-CPU jobs on an appropriately configured cluster, a multiprocessor Linux workstation, or across a series of networked hosts. It is beyond the scope of this tutorial to provide step-by-step instructions for setting up an environment to allow multiple-CPU jobs, so it is assumed that the user is already able to run other Schrödinger software on multiple CPUs. For further details, please consult the Product Installation Guide.

In the case of `phase_partition`, `phase_multiPartition` and `phase_scoring`, each processor operates on a particular subset of variants. By contrast, each processor associated with a `phase_inactive` or `phase_multiQsar` job works on a particular subset of hypotheses. To launch a multiple-CPU job, one need only use the `-HOST` option to indicate how to the job should be distributed. All four programs use the same syntax, so we provide examples for `phase_inactive` only. You are encouraged to repeat the steps in Section 1.11, submitting `phase_inactive` as a multi-CPU job.

Suppose the file `$SCHRODINGER/schrodinger.hosts` contains a host entry named `hostX` with16 processors, and suppose we'd like to split a `phase_inactive` job across 4 of those processors. Using the example from Section 1.11, we would issue the following command to launch the job:

```
Shell> $SCHRODINGER/phase_inactive –HOST hostX:4 score_inactives
```

This command will result in the creation of 4 subjobs, each of which will be assigned ¼ of the hypotheses in the file `score_inactives_inactive.inp`. The first subjob will process hypotheses 1, 5, 9, etc.; the second subjob will process hypotheses 2, 6, 10, etc.; and so on. This scheme provides near optimal load balancing, as it is very unlikely that any one processor would be saddled with a disproportionate number of "expensive" hypotheses.

If you do not currently have access to a machine with multiple processors, you can still observe how jobs are distributed by mimicking a request for 4 processors on the *localhost*. For example:

```
Shell> $SCHRODINGER/phase_inactive –HOST localhost:4 score_inactives
```

This command will result in the launching of 4 subjobs that will run as concurrent processes on the machine you are logged into. The behavior will be much the same as if you were actually running on 4 independent CPUs, except that it won't be any faster than a regular single-processor job.

If you wish to split a job across more than one host on the network, then you must enclose the list of hosts in double quotes and separate them with spaces. For example:

```
Shell> $SCHRODINGER/phase_inactive -HOST "hostX:4 hostY:8" score_inactives
```

In this example, a total of 12 subjobs would be launched.

Appendix A: Phase QSAR Method

Phase QSAR models are developed from a series of molecules, of varying activity, which have all been aligned to a common pharmacophore hypothesis that's associated with a single reference ligand.

A QSAR model may be atom-based or pharmacophore-based, the difference being whether all atoms are taken into account, or merely the pharmacophore sites that can be matched to the hypothesis.

A rectangular grid is defined to encompass the space occupied by a training set of aligned molecules. This grid divides space into uniformly-sized cubes, typically one angstrom on each side. In atom-based models, the grid is populated by van der Waals spheres, with radii that depend on the atom type. In pharmacophore-based models, the grid is populated by the pharmacophore sites that match the hypothesis, with each site represented by a sphere with a user-definable radius.

A given atom or pharmacophore site will occupy the space of one or more cubes in the grid. Occupation of a cube is deemed to occur if the center of that cube falls within the radius of the atom or site. A given cube may be occupied by more than one atom or site, and that occupation may come from the same molecule or from different molecules.

Each occupied cube gives rise to one or more *volume bits*. A volume bit is allocated for each different class of atom/site that occupies a cube.

In pharmacophore-based models, sites are assigned to classes that are determined by the feature definitions used to create the hypothesis (e.g., A, D, H, N, P, R). In atom-based models, there are 6 distinct atom classes that have some correspondence/similarity with pharmacophore feature types, but atom classes are assigned using fixed internal rules, not the hypothesis feature definitions:

D – Hydrogen bond donor (hydrogens bonded to N, O, P, S)
H – Hydrophobic/non-polar (C, H-C, Cl, Br, F, I)
N – Negative ionic (formal negative ionic charge)
P – Positive ionic (formal positive ionic charge)
W – Electron-withdrawing (N, O)
X – Miscellaneous (all other types of atoms)

So, if a particular cube is occupied by a "D" from molecule 1, an "H" from molecule 5, and a "P" from molecule 9, that cube would be allocated three volume bits. If a cube is never occupied by any molecule in the training set, no volume bits would be allocated. Hence, each volume bit must be *set* by at least one molecule in the training set.

The pool of volume bits provides a means of characterizing the molecules. In atom-based models, the pattern of volume bits that are set by a molecule encodes the size, shape, and chemical characteristics of that molecule. In pharmacophore-based models, the pattern of set bits determines which subset of critical pharmacophore features that molecule contains, and the positions of those features in relation to other molecules.

If a binary scheme (0/1) is used to denote which bits are set by each molecule, a table of bit values may be assembled:

$molecule_1$ 0 1 1 0 0 1 1 0 0 1 0 1 0 . . .
$molecule_2$ 0 1 1 0 0 0 1 0 0 1 0 1 0 . . .
$molecule_3$ 0 0 0 1 0 1 0 0 1 0 1 0 1 . . .
.
.
.

For atom-based models, there are usually several hundred or more volume bits for each series of aligned molecules. For pharmacophore-based models, that number is much smaller, usually only a few dozen. The number of bits increases as the grid spacing becomes finer, and, in the case of atom-based models, as the molecules become larger.

To generate a QSAR model, the 0/1 bit values are treated as independent variables in partial least-squares (PLS) regression analysis. This involves finding a linear least-squares relationship between the activity data and a special set of orthogonal factors which are linear combinations of the bit value variables.

More precisely, if there are *n* molecules in the training set and *v* volume bits, let the $n \times v$ matrix **X** represent the above table of volume bits, and let the $n \times 1$ vector **y** represent the activity values for the training set molecules. Creation of the PLS regression model proceeds as follows:

```
Center each column $j$ of $\mathbf{X}$:
for $j = 1,\ldots,v$
    $\mu_j^x = \frac{1}{n}\sum_{k=1}^{n} \mathbf{X}(k,j)$
    for $i=1,\ldots,n$
        $\mathbf{X}(i,j) \rightarrow \mathbf{X}(i,j) - \mu_j^x$
    next $i$
next $j$

Center $\mathbf{y}$:
$\mu^y = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}(i)$
for $i=1,\ldots,n$
    $\mathbf{y}(i) \rightarrow \mathbf{y}(i) - \mu^y$
next $i$

Determine PLS factors and regression coefficients for up to $m$ PLS factors ($m \le v$):
$\mathbf{X}_1 = \mathbf{X}$
```

```
for $k = 1,\ldots,m$
    Compute the vector of weights that define PLS factor $k$:
    $\mathbf{w}_k = \mathbf{X}_k^T\mathbf{y} / |\mathbf{X}_k^T\mathbf{y}|$        $(\mathbf{w}_k \in \mathbf{R}^{v\times 1})$
    Project the rows of $\mathbf{X}_k$ onto factor $k$:
    $\mathbf{t}_k = \mathbf{X}_k\mathbf{w}_k$        $(\mathbf{t}_k \in \mathbf{R}^{n\times 1})$
    Project $\mathbf{t}_k$ onto each column of $\mathbf{X}_k$:
    $\mathbf{p}_k = \mathbf{X}_k^T\mathbf{t}_k / |\mathbf{t}_k^T\mathbf{t}_k|$        $(\mathbf{p}_k \in \mathbf{R}^{v\times 1})$
    Compute the kth PLS regression coefficient by projecting $\mathbf{t}_k$ onto $\mathbf{y}$:
    $\mathbf{b}(k) = \mathbf{t}_k^T\mathbf{y} / |\mathbf{t}_k^T\mathbf{t}_k|$        $(\mathbf{b} \in \mathbf{R}^{m\times 1})$
    Orthogonalize $\mathbf{X}_k$ w.r.t. PLS factor $k$:
    $\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{t}_k\mathbf{p}_k^T$
next $k$
```

For a regression with m PLS factors, the fitted activities $\hat{\mathbf{y}}$ are then given by:

$$\hat{\mathbf{y}}(i) = \mu^y + \sum_{k=1}^{m} \mathbf{b}(k)\mathbf{t}_k(i) \qquad i=1,\ldots,n$$

To apply the m-factor PLS model to a new set of n_T ligands with bit value matrix $\tilde{\mathbf{X}}$, the regression coefficients $\mathbf{b}$ must first be translated back to the space of the original $\mathbf{X}$ variables:

Define

$$\mathbf{W} \equiv [\mathbf{w}_1 \ldots \mathbf{w}_m] \qquad (\mathbf{W} \in \mathbf{R}^{v\times m})$$
$$\mathbf{P} \equiv [\mathbf{p}_1 \ldots \mathbf{p}_m] \qquad (\mathbf{P} \in \mathbf{R}^{v\times m})$$
$$\mathbf{b}^x \equiv \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1}\mathbf{b} \qquad (\mathbf{b}^x \in \mathbf{R}^{v\times 1})$$

The coefficients $\mathbf{b}^x$ may then be used to predict activities for the new ligands:

$$\hat{\mathbf{y}}(i) = \mu^y + \sum_{j=1}^{v} \left[\tilde{\mathbf{X}}(i,j) - \mu_j^x\right]\mathbf{b}^x(j) \qquad i=1,\ldots,n_T$$

Appendix B: Phase QSAR Statistics

Training Set & Model

$m =$ number of PLS factors in the model
$n =$ number of molecules in the training set
$df_1 = m$ (degrees of freedom in model)
$df_2 = n - m - 1$ (degrees of freedom in data)
$y_i =$ observed activity for training set molecule *i*
$\hat{y}_i =$ predicted activity for training set molecule *i*

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ (mean observed activity)

$\sigma_y^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2$ (variance in observed activities)

$sse = \sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2$ (sum of squared errors)

$\sigma_{err}^2 = \frac{sse}{n}$ (variance in errors)

$ssy = \sum_{i=1}^{n}\left(\hat{y}_i - \bar{y}\right)^2$

$SD = sqrt\left(\frac{sse}{df_2}\right)$ (standard deviation of regression)

$R^2 = 1.0 - \frac{\sigma_{\mathrm{err}}^2}{\sigma_y^2}$ (R-squared; coefficient of determination)

Note that R^2 can never be negative. That’s because the regression coefficients are optimized to minimize *sse*. The worst-case scenario is when the independent variables have absolutely no statistical relationship with activity. Under those circumstances, the regression coefficients will all be zero, and the model will contain only an intercept parameter, the value of which will be $\bar{y}$. Thus every predicted activity will be $\bar{y}$, and σ_{err}^2 will be equal to σ_y^2, yielding $R^2 = 0$.

$F = \frac{ssy/df_1}{sse/df_2}$ (F statistic; overall significance of model)

$P = B\left\{df_2/2,\ df_1/2,\ df_2/(df_2 + F\cdot df_1)\right\}$ (probability that correlation could occur by chance)

where

$$B(a,b,x) \equiv \frac{\int_0^x \xi^{a-1}(1-\xi)^{b-1} d\xi}{\int_0^1 \xi^{a-1}(1-\xi)^{b-1} d\xi}$$

Test Set Predictions

$T =$ The test set of molecules
$n_T =$ number of molecules in *T*
$y_j =$ observed activity for molecule $j \in T$
$\hat{y}_j =$ predicted activity for molecule $j \in T$

$$RMSE = \left[\frac{1}{n_T}\sum_{j \in T}(\hat{y}_j - y_j)^2\right]^{\frac{1}{2}}$$ (root-mean-squared error)

$$Q^2 = R^2(T)$$

The formulas for R^2 and Q^2 are equivalent, with the only difference being that Q^2 is computed using the observed and predicted activities for the test set. However, Q^2 can take on negative values. This happens whenever the variance in the test set errors is larger than the variance in the observed test set activities. Often times, the test set does not have as large a range of activity values as the training set (so the variance in *y* is smaller), and the errors for the test set tend to be larger than those for the training set (so the variance in the errors is larger), so it's not uncommon to see negative Q^2 values from time to time.

$$Pearson\ r = \frac{\sum_{j \in T}(y_j - \bar{y}(T))(\hat{y}_j - \bar{\hat{y}}(T))}{\left[\sum_{j \in T}(y_j - \bar{y}(T))^2 \sum_{j \in T}(\hat{y}_j - \bar{\hat{y}}(T))^2\right]^{\frac{1}{2}}}$$

Because the observed and predicted activities are shifted by their sample means and the expression is normalized for variance, the Pearson correlation coefficient, unlike Q^2, is insensitive to systematic errors in the predictions. So if the rank order of the activity predictions is basically correct, but there's a significant constant shift in the values or a compression of the predicted activity coordinate, the Pearson correlation coefficient may still be quite high, even if Q^2 is small or negative.

Appendix C: Phase QSAR T-Value Filters

Another important statistic to consider when building QSAR models is the *t*-value associated with each regression coefficient in volume bit space. This is a measure of the statistical significance of a particular bit *j* in the overall regression model containing a given number of PLS factors:

$$t\text{-value}\left(\mathbf{b}^X(j)\right) = \frac{\mathbf{b}^X(j)}{\sigma_{\mathbf{b}^X(j)}}$$

The quantity $\sigma_{\mathbf{b}^X(j)}$ is the standard deviation in the coefficient $\mathbf{b}^X(j)$, which is a measure of how much the coefficient is expected to vary across different training sets drawn from a single population. Computation of the vector $\mathbf{b}^X$ is covered in Appendix A.

For significance testing, we are interested in the probability that the *t*-value would be larger than a particular value if the null hypothesis were true, i.e., if $\mathbf{b}^X(j)$ were zero. This probability is governed by the Student's *t*-distribution and is given by:

$$P = B\left(df/2,\ 0.5,\ df/(df+t^2)\right)$$

where

$$B(a,b,x) = \frac{\int_0^x \xi^{a-1}(1-\xi)^{b-1}d\xi}{\int_0^1 \xi^{a-1}(1-\xi)^{b-1}d\xi}$$

The above formulation corresponds to the two-sided test, so *P* accounts for both negative and positive values of *t*. A tabulation of *t*-values for various degrees of freedom *df* and probabilities *P* is provided below. It is standard practice to adopt the 95% significance level, so *P* = 0.05 may be used to guide the selection of an appropriate *t*-value cutoff for training sets of different sizes. For all but the smallest training sets, a *t*-value of about 2 is sufficient for 95% significance.

df	*P* = 0.1	*P* = 0.05	*P* = 0.025
5	2.01	2.57	3.16
10	1.81	2.45	2.63
15	1.75	2.13	2.49
20	1.72	2.09	2.42
30	1.70	2.04	2.36
40	1.68	2.02	2.33
50	1.68	2.01	2.31

Whereas the regression coefficient $\mathbf{b}^X(j)$ is computed using exact formulas, the standard deviation $\sigma_{\mathbf{b}^X(j)}$ must be estimated from a series of training sets drawn from a population. Phase QSAR employs a leave-*n*-out procedure (10% by default) to build a series of models from sub-populations of the user-defined training set, and then calculates a mean and standard deviation for each coefficient from the values observed across the series of models. The *t*-value for a given coefficient is then calculated as the ratio of the mean to the standard deviation.

Given a particular threshold, each volume bit whose *t*-value falls below that threshold is eliminated, and a model is built from the remaining volume bits. A complicating detail of this process is that each regression coefficient $\mathbf{b}^X(j)$ and its *t*-value are dependent upon the number of PLS factors included in the QSAR model. As a result, the volume bits that survive the *t*-value filter in the one-factor model are generally not the same volume bits that survive the *t*-value filter in the two-factor model, and so forth. However, the PLS build-up procedure described in Appendix A requires a single set of independent variables for all models. To associate different sets of independent variables with different numbers of factors would require that the entire algorithm be repeated from scratch with each set of variables, increasing the maximum number of factors by one each time. Unfortunately, this would require a complete restructuring of how Phase QSAR models are developed, stored, applied, and visualized.

To circumvent this complication, Phase retains only the volume bits that pass the *t*-value filter in every model (i.e., the intersection of surviving bits from the one-factor model, the two-factor model, etc.). It's important to remember, however, that as the requested maximum number of factors (maxFactors) is increased, the set of bits retained gets smaller and smaller. Thus if one chooses, e.g., maxFactors = 8, the pool of independent variables will be considerably smaller than the pool that would result from, say, maxFactors = 3. In effect, the final model can be *starved* of information if maxFactors is too large, resulting in significant degradation of the training set fit. Accordingly, when applying a *t*-value filter, it's critical to be conservative when choosing maxFactors. Fortunately, Phase QSAR usually achieves a strong fit to the training set data by the time two or three factors are included, so it is rarely necessary, or advisable, to choose maxFactors larger than three. Following this guideline, *t*-value filtered models will include only statistically significant volume bits, with little or no degradation of the training set fit.

Appendix D: Phase QSAR Stability

PLS-based software packages frequently report cross-validated R^2 values, and the accompanying user manuals often assert that this parameter allows one to determine the "optimal" number of PLS factors and/or to assess the predictive value of a given model. Justification for such claims is rarely provided, yet there is still widespread belief that cross-validated R^2 is a reasonable substitute for external validation against a true test set. This of course is not true, since there is no statistic derivable from the training set data alone that can reliably predict how well a model will perform on new compounds. Thus one cannot really determine from cross-validated R^2, whether one model is better than another, or whether 3 PLS factors is too many.

Another purported attribute of cross-validated R^2 is that it measures the sensitivity of a model to changes in the training set. This is a reasonable quantity to want to measure, since models that contain significant chance correlations tend to be most sensitive to changes in the training set. Unfortunately, cross-validated R^2 is not the most effective way to measure this phenomenon, either, because it tends to agree largely with ordinary R^2, as will be shown.

A more informative cross-validated statistic, which we refer to as *Stability*, is computed as follows:

1. Build PLS models containing 1 factor, 2 factors, etc. using the full training set.
2. Randomly remove *n* compounds from the training set.
3. Build PLS models containing 1 factor, 2 factors, etc., using the compounds that remain in the training set.
4. Predict the activities of the *n* compounds that were excluded.
5. Return the *n* compounds to the training set.
6. Repeat steps 2-5 until all compounds have been excluded at least once. If a compound is excluded more than once, take an average of its predicted activities.
7. For models containing 1 factor, 2 factors, etc., compute the R^2 value between the leave-*n*-out predictions and the predictions obtained in step 1. This R^2 value is the *Stability* of the model.

The term Stability has been adopted because the reference points for the R^2 calculation are the predictions from the original model. Any deviation from the original model will therefore lead to a decrease in the Stability statistic. Cross-validated R^2 measures something quite different, because the reference points are the experimental activities, so a deviation from the original model does not always cause a decrease in cross-validated R^2.

To illustrate how Stability and cross-validated R^2 compare, a number of scatter plots have been assembled in Figures D.1-D.3. Each plot illustrates the relationship between a pair of statistics computed from the 188 QSAR models developed in Section 1.13.

Figure D.1 contains plots of cross-validated R^2 against ordinary R^2 for models containing 1, 2 and 3 PLS factors. Observe that the two statistics correlate strongly among different hypotheses, with Pearson r values of 0.953, 0.773 and 0.821. This indicates that cross-validated R^2 is telling us nearly the same thing as ordinary R^2. And since ordinary R^2 always increases as more PLS factors are added, cross-validated R^2 will tend to do the same. Yet we know that predictions on test set compounds often degrade as more PLS factors are retained, so it cannot be wise to conclude that more factors is almost always better.

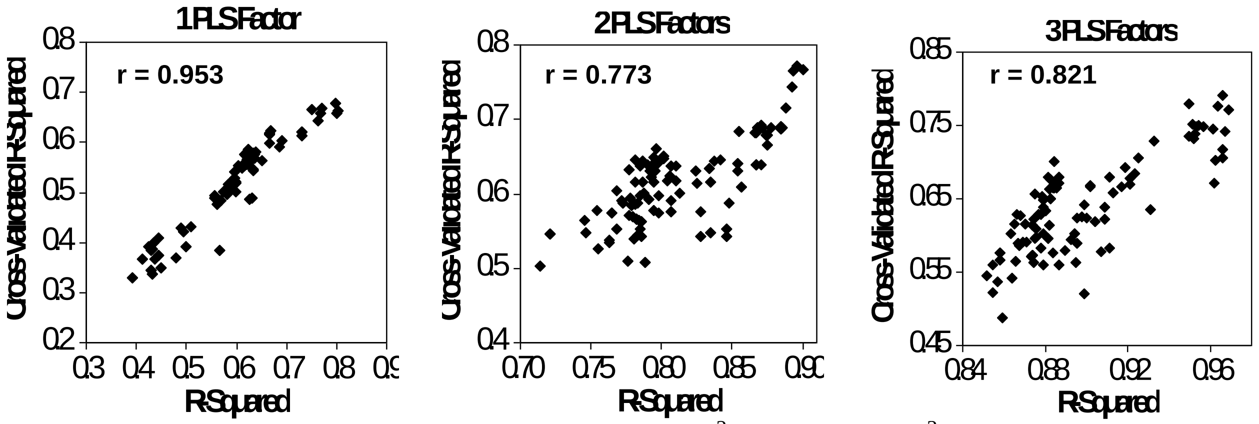

Figure D.1. The relationship between cross-validated R^2 and ordinary R^2 for the 188 QSAR models developed in Section 1.13.

Figure D.2 illustrates this assertion quite dramatically. Here, cross-validated R^2 is plotted against test set Q^2 for each of the 188 hypotheses. Although there's a reasonably strong correlation of 0.695 for 1-factor models, it drops to 0.170 for 2-factor models, and the relationship becomes decidedly negative by the time 3 factors are included, with a correlation of -0.465. So whereas cross-validated R^2 is a reasonably good indicator of which 1-factor models will perform best on the test set, it is of absolutely no value for 3-factor models. In fact, one could argue that 3-factor models with lower cross-validated R^2 values should be preferred. Given the strong 3-factor correlation in Figure D.1, this should come as no surprise. Often times, models with weaker R^2 (and cross-validated R^2) are the ones that are NOT over-fit, so the predictions on new compounds tend to be better.

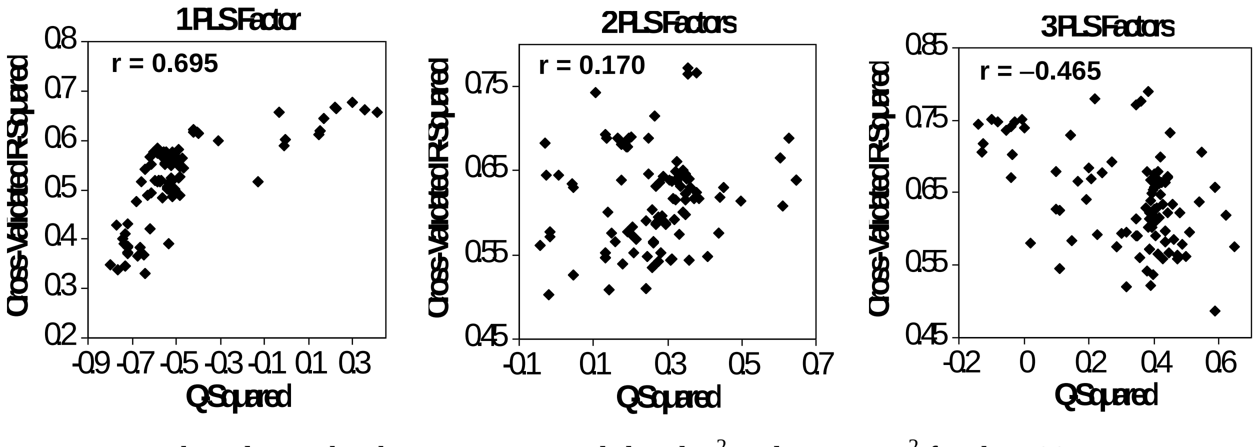

Figure D.2. The relationship between cross-validated R^2 and test set Q^2 for the 188 QSAR models developed in Section 1.13.

In stark contrast to cross-validated R^2, the relationship between Stability and Q^2 grows progressively stronger as more factors are included, Figure D.3. While it's fair to say that

Stability is of no value for 1-factor models, it provides some indication of which 2-factor and 3-factor models should be avoided. Though it must be conceded that some of the models with high Q^2 values have relatively low Stability, one should not necessarily consider this to be a failure of the Stability statistic. Depending on the test set being predicted, it is still possible to obtain accurate predictions with an unstable model. However, as more factors are added, models with very high Stability are more likely, on average, to yield better predictions than models with very low Stability.

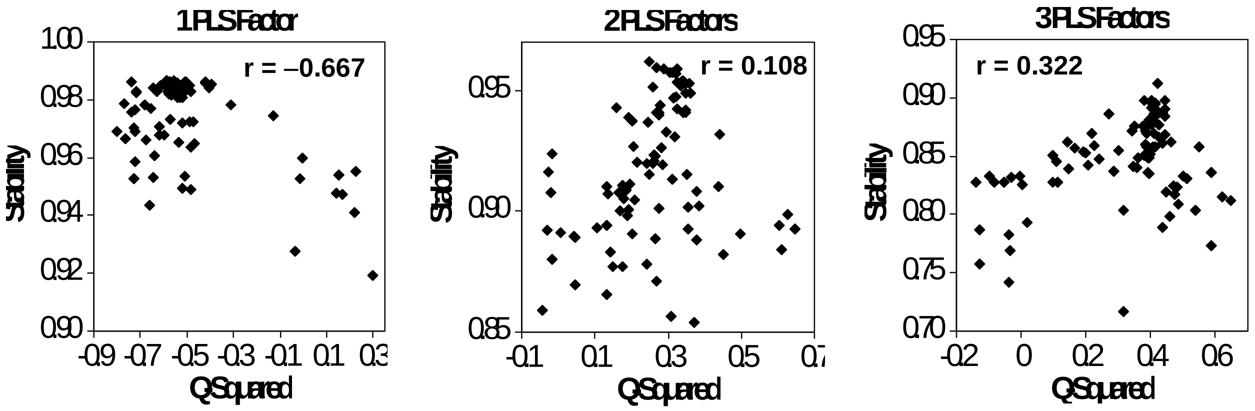

Figure D.3. The relationship between Stability and test set Q^2 for the 188 QSAR models developed in Section 1.13.