RDC-PANDA

User Manual

Version 1.0

Jianyang (Michael) Zeng and Bruce R. Donald

Contents

1. Introduction

RDC-PANDA (RDC-based SSE PAcking with NOEs for Structure Determination and NOE Assignment), is a suite of programs for nuclear Overhauser effect (NOE) assignment and high-resolution structure determination starting with a global fold calculated from exact solutions to the residual dipolar coupling (RDC) equations. RDC-PANDA is developed in the lab of Prof. Bruce Donald at Duke University. This user manual is for version 1.0 of the software.

RDC-PANDA is free software and can be redistributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (optionally) any later version. RDC-PANDA is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. Full licensing details, including citation requirements for the various different modules of the software, are found in the document **license.pdf** enclosed with this package distribution.

RDC-PANDA is specifically designed for automated NMR NOE assignment and protein structure determination package. It provides a novel approach for computing the initial structure template for NOE assignment by exactly solving backbones from RDCs and systematically choosing rotamers based on NOE pattern matching. RDC-PANDA mainly consists of following three algorithmic modules:

(1) **RDC-ANALYTIC**, which computes orientations and conformations of SSE backbones. Current version of RDC-ANALYTIC is designed to compute protein global fold backbone using NH and CH RDCs in one alignment medium. It determines the conformations and orientations of secondary structure elements (SSEs) by solving the RDC equations in closed form. It applies a systematic search approach to compute the global optimal solution of each SSE fragment that best fits the RDC data. The RDC-ANALYTIC module has been implemented in a standalone package. More details about configurations and input file formats of RDC-ANALYTIC can be found in its user manual (released with this package distribution).

(2) **PACKER**, which packs SSE backbones using spare NOE restraints. PACKER first extracts a set of sparse unambiguous NOE assignments using only chemical shift information. It then applies a 3D grid search to find all discrete translations that satisfy the sparse NOE distance restraints. When packing each pair of SSE backbones, all four-fold symmetries of SSE orientations are also taken into account.

(3) **HANA** (HAusdorff-based NOE Assignment), which uses the SSE backbones to place side-chains and assign NOEs. HANA first computes a set of initial NOE assignments by considering all pairs of protons that are possibly assigned to an NOE cross peak if the resonances of corresponding atoms fall within a tolerance window. Then a Hausdorff-based pattern matching technique is employed to deduce similarity between experimental and back-computed NOE spectra for each rotamer from a statistically diverse library, and drive the selection of optimal position-specific rotamers for filtering ambiguous NOE assignments.

In addition to above three modules, a local minimization approach is used to compute loops and refine side-chain conformations by fixing the core structure as a rigid body while allowing movement of loops and side-chains. Figure 1 shows the flow chart of RDC-PANDA. The input data to RDC-PANDA include: (1) the primary sequence of the protein; (2) the 3D NOE peak list from both 15N- and 13C-

edited spectra; (3) the resonance assignment list, including both backbone and side-chain resonance assignments; (4) the RDC data, including CH and NH RDCs, and the RDCs of other bond vectors (optional), such as CA-C' and N-C' bond vectors; (5) the TALOS table of dihedral angle ranges from the chemical shift analysis (optional); (6) the rotamer library. RDC-PANDA has been used to solve the new structures of the FF Domain 2 of human transcription elongation factor CA150 (RNA polymerase II C-terminal domain interacting protein) (FF2). The new NMR structures have been deposited into Protein Data Bank (PDB ID: 2KIQ).

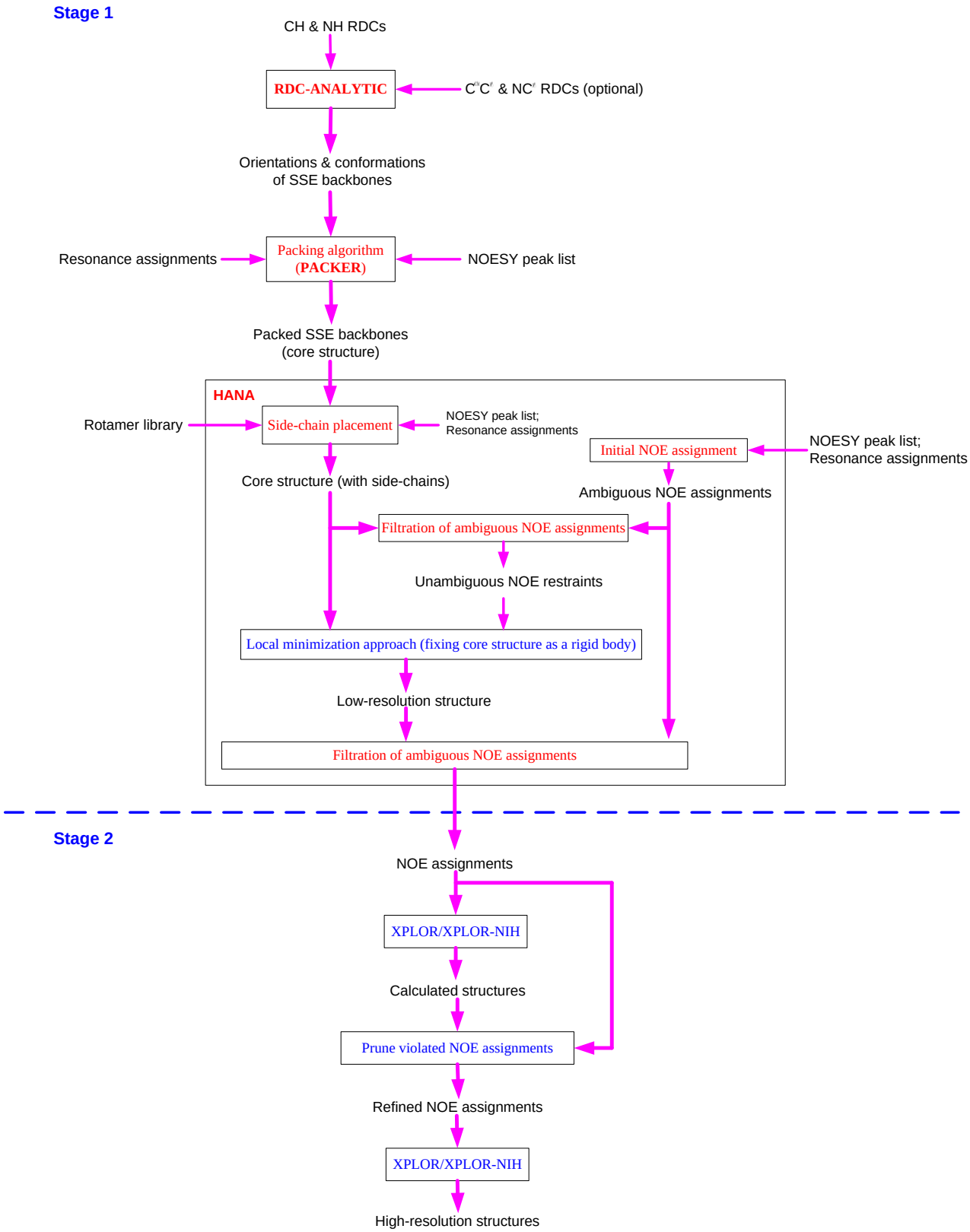

Figure 1. Flow chart of RDC-PANDA.

2. License Information

The source header below must be included in any modification or extension of the source code of RDC-PANDA.

Source Header

```
RDC-PANDA NOE Assignment and Protein Structure Determination Version 1.0
Copyright (C) 2001-2009 Bruce Donald Lab, Duke University


RDC-PANDA is free software: you can redistribute it and/or modify it under
the terms of the GNU Lesser General Public License as published by the Free
Software Foundation, either version 3 of the License, or (at your option)
any later version.

RDC-PANDA is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more
details.

You should have received a copy of the GNU Lesser General Public License
along with this library; if not, see:
<http://www.gnu.org/licenses/>.

There are additional restrictions imposed on the use and distribution of
this open-source code, including: (A) this header must be included in any
modification or extension of the code; (B) you are required to cite our
papers in any publications that use this code. The citation for the various
different modules of our software, together with a complete list of
requirements and restrictions are found in the document license.pdf enclosed
with this distribution.

Contact Info:
Bruce R. Donald
Duke University
Department of Computer Science
Levine Science Research Center (LSRC)
Durham, NC 27708-0129
USA
e-mail: www.cs.duke.edu/brd/


<signature of Bruce Donald>, 01 December, 2009
Bruce R. Donald, Professor of Computer Science and Biochemistry
```

3. Citation Requirements

You are required to cite our papers in any publications that use this code. The primary citation corresponding to this software is [1]. The papers that can be cited based-on or related-to this software are listed below.

[1] Jianyang Zeng, Jeffrey Boyles, Chittaranjan Tripathy, Lincong Wang, Anthony Yan, Pei Zhou, and Bruce Randall Donald. High-resolution protein structure determination starting with a global fold calculated from exact solutions to the RDC equations. Journal of Biomolecular NMR, 45(3):265-281, 2009.

[2] Bruce R. Donald and Jeffrey Martin. Automated NMR Assignment and Protein Structure Determination using Sparse Dipolar Coupling Constraints. Progress in Nuclear Magnetic Resonance Spectroscopy, 55(2):101-127, 2009.

[3] Lincong Wang, Ramgopal R. Mettu, and Bruce R. Donald. A Polynomial-Time Algorithm for De Novo Protein Backbone Structure Determination from NMR Data. Journal of Computational Biology, 13(7):1276-1288, 2006.

[4] Lincong Wang and Bruce Randall Donald. Analysis of a Systematic Search-Based Algorithm for Determining Protein Backbone Structure from a Minimal Number of Residual Dipolar Couplings. In Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB'04), Stanford CA, pages 319–330, 2004.

4. Installation

System Requirements

To use this software, Java Runtime Environment from Sun Microsystems (JRE) must be installed. The newest version of JRE can be downloaded from http://java.sun.com/javase/downloads/.

How to install the program?

(1) Install Java Runtime Environment from Sun Microsystems (JRE). The newest version of JRE can be downloaded from http://java.sun.com/javase/downloads/.

(2) Create a working directory. Use gunzip and tar commands to decompress Rdc-Analytic_Panda_V1.0_Jan_2010.tgz, and extract all files in the working directory.

(3) Install Xplor/Xplor-NIH. The newest version of Xplor/Xplor-NIH can be download from http://nmr.cit.nih.gov/xplor-nih/.

How to run the program?

To run the RDC-Panda software, go to the RDC-PANDA working directory (which contains the bin sub-directory), and use the following command:

```
java –cp ./bin/ RdcPanda
```

Or alternatively use the following commands:

```
chmod 755 RdcPanda
./RdcPanda
```

If the program runs successfully, the following txt interface should be given:

```
*********************************************************************************
**        ,rMr.,------.,rMr.           RDC-PANDA Version 1.0                     **
**        (GNP'       `?ND)            Contact Info:                              **
**          P  xMx.  ,yMb  ?              Bruce Donald                             **
**          (  ?X_O  O_XP  )              Duke University                          **
**          (     qp     )   RDCs         Department of Computer Science           **
**           1  `--'`--'  7                Levine Science Research Center (LSRC)     **
**         _____        ,__)              Durham                                   **
**        (--|__) _..._  _| _,             NC 27708-0129, USA                       **
**          _|   (_|| |(_|(_|                                                       **
**         (                                                                       **
**                                                                                 **
**         J. Zeng, J. Boyles, C. Tripathy, L. Wang, A. Yan, P. Zhou and B.R. Donald.  **
**         "High-Resolution Protein Structure Determination Starting with a Global     **
**         Fold Calculated from Exact Solutions to the RDC Equations." J. Biom. NMR,   **
**         45(3):265-281, 2009.                                                    **
*********************************************************************************
RDC-PANDA>
```

Once the program is started, the desired functions can be executed through the command line. The details of commands and corresponding parameter files will be described next. The templates of command lines can be found in ./doc/command-template.txt.

The structure determination through RDC-Panda is generally divided into steps:

(1). RDC-ANALYTIC is used to compute the SSE backbones from RDCs.

(2). PACKER is invoked to pack SSEs.

(3). HANA is used to compute the rotamers based on NOE pattern-matching technique. The loops are computed through the local minimization approach. The NOE assignments are computed by HANA.

(4). Xplor is called to calculate the structures based on the NOE assignment table computed by HANA.

How to compile the program? (optional)

(1) Install Java Development Kit (JDK). The newest version of JRE can be downloaded from http://java.sun.com/javase/downloads/.

(2) Modify the Makefile under the RDC-PANDA working directory as needed. In particular, specify your working directory and the paths to your Java compiler and resources.

(3) Under the RDC-PANDA working directory, type 'make'.

5. System Configurations

File Organization

The directory structure of RDC-PANDA is organized as follows:

./experiments/
This directory contains data and input parameter files used for the experiments with proteins FF2 and ubiquitin.

./bin/
This directory contains the java executable files.

./src/
This directory contains the java source codes.

./inputFiles/
This directory contains input data and parameter files.

./outputFiles/
This directory is used as the location where output files are written to.

./doc/
This directory contains the documentations of the program.

./system/
This directory contains the system configuration files. The directory path of input data and parameter files can be set in ./system/system-parameters.txt.

./xplor/
This directory is used for running Xplor scripts and storing corresponding results.

Required NMR Data and Formats

RDCs:
CH and NH RDCs are required. Other RDCs such as CaC' and NC' RDCs are optional. Format: residue index, RDC value (unscaled), experimental error, 0.0. In RDC-ANALYTIC, RDCs are in standard Xplor format.

Resonance assignments:

The resonance assignment list should contain both backbone and side-chain resonance assignments.
Format: XEASY format, CYANA format or BMRB deposition format.

Sequence:

The primary sequence of the protein.
Format: residue name, residue index.

NOESY cross peaks:

3D or 4D ^{15}N- or ^{13}C-edited NOESY peak list.
Format: XEASY format or NMRVIEW format.

TALOS dihedral angle restraints (Optional):

The TALOS table provides interval restraints of dihedral angles estimated from chemical chemical shift analysis.
Format: residue index, residue name, Phi or Psi, average value, error bar.

Atom Naming Schemes

In RDC-Panda, the following four atom naming schemes in the input files are supported:

PDB-NEW: eg HA1, HA2 (Gly);
PDB-OLD: eg 1HA, 2HA (Gly);
BMRB-NEW: eg HA2, HA3 (Gly);
BMRB-OLD: eg 2HA, 3HA (Gly).

In the program, all atom names are first converted to the PDB-NEW naming scheme.

Input Parameter File Format

This section describes the different commands and the corresponding required parameters. Each command line consists of command and the file name of input parameter file. The command and the parameter file name must be separated by space. In the input parameter file, lines starting with '//' are parsed as comments. Parameter names are single words; parameter values follow the corresponding parameter name on the same line and are separated by '='. Each parameter line ends with the symbol ';'. The following gives an example of an input parameter file:

```
      //reading protein sequence
      sequence = UBQ.seq;

      //format of resonance assignments.
      //"CYANA" for the CYANA format, "BMRB" for BMRB format, "XEASY" for XEASY
format.
      resFormat=CYANA;

      //atome naming scheme in the resonance list
      resNameScheme=BMRB-NEW;//eg HB1, HB2 for GLU, etc.

      //name of resonance assignment file:
      resonance =ubc_cyana_prot-convert.txt;
```

```
//reading NOESY peak lists
3D-N15-NOESY=ubc_n15_noe.peaks;
3D-C13-NOESY =ubc_c13_noe.peaks;
...
```

All input parameter files are stored in directory ‘inputFiles’.

System Files

All system files of RDC-PANDA are stored in sub-directory “./system/”. This sub-directory includes the rotamer library, system parameters (e.g., location of the input files) and BMRB statistical information.

6. RDC-PANDA Commands

`RdcExactHelix RdcExactHelix.input`

This command computes the helix backbone from RDCs using the RDC-EXACT algorithm. CH and NH RDCs are required as input files, while CaC’ and NC’ RDCs are optional. A typical input parameter file includes the following parameters:

`sequence`
: File name of protein primary sequence.

`chFile`
: File name of CH RDCs.

`nhFile`
: File name of NH RDCs.

`cocaFile`
: File name of CaC’ RDCs. This parameter is set to be ‘null’ if CaC’ RDCs are not available.

`conFile`
: File name of NC’ RDCs. This parameter is set to be ‘null’ if NC’ RDCs are not available.

`wtCoCA, wtCoN`
: Weighting factor between CH, NH RDCs and CaC’ and NC’ RDCs in the scoring function used in systematic search for computing the dihedral angle solutions.

`Syy, Szz`
: Diagonal elements of Saupe order matrix of alignment tensor.

`ResBounds`
: Residue boundaries of the helix. The start and end residues should be separated by ‘-’.

`PrePdbName`

The PDB file name of previous backbone fragment. This is used for computing a long helix when the helix is divided into two parts. This parameter is set to be ‘null’ for computing a single fragment.

`OutPdbName`
File name of output PDB file name of computed backbone.

`TALOS`
File name of the TALOS angle table.

This command is not longer under the maintenance. This updated version of this function can be found in the latest version of RDC-ANALYTIC.

RdcExactHelixWOAT RdcExactHelixWOAT.input

This command line computes the helix backbone without requiring the alignment tensor information. The input parameter file is similar to RdcExactHelix.input except the inputs of diagonal elements of Saupe matrix are not required.

This command is not longer under the maintenance. This updated version of this function can be found in the latest version of RDC-ANALYTIC.

RdcExactSheet RdcExactSheet.input

This command is used to compute the beta sheet backbone from RDCs using the RDC-EXACT algorithm. CH and NH RDCs are required as input files, while CaC’ and NC’ RDCs are optional. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`chFile`
File name of CH RDCs.

`nhFile`
File name of NH RDCs.

`cocaFile`
File name of CaC’ RDCs. This parameter is set to be ‘null’ if CaC’ RDCs are not available.

`conFile`
File name of NC’ RDCs. This parameter is set to be ‘null’ if NC’ RDCs are not available.

`wtCoCA, wtCoN`
Weighting factor between CH, NH RDCs and CaC’ and NC’ RDCs in the scoring function used in systematic search for computing the dihedral angle solutions.

`Syy, Szz`
Diagonal elements of Saupe order matrix of alignment tensor.

`SseBoundaries`
Residue boundaries for all beta strands. The start and end residues should be separated by '-'.

`PrePdbName`
The PDB file name of previous backbone fragment. This is used for computing a long helix when the helix is divided into two parts. This parameter is set to be 'null' for computing a single fragment.

`hBonds`
File name of hydrogen bonds used for packing bete strands.

`OutPdbName`
File name of output PDB file name of computed backbone.

`TALOS`
File name of the TALOS angle table.

This command is not longer under the maintenance. This updated version of this function can be found in the latest version of RDC-ANALYTIC.

SSEPacking packingSSEs.input

This command line packs SSEs using the PACKER algorithm. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`IsOne2One`
whether it is one to one packing, or one to all packing (namely packing one structure to all structure in an ensemble). It is set to be 1 for one-to-one packing, 0 for one to all packing.

`SSES`
PDB names of two SSE fragments, which are separated by '-'.

`outPackedStructures`
File name of the output packed structures (only the common name of all structures in the ensemble).

`NOE-Table`
File name of NOE restraint table used for packing SSEs.

`NOEFormat`
Format of NOE restraint table. It can be 'XPLOR' or 'CYANA' format.

`isSkipSymCheck`
Whether the symmetry checking step is inputted. It can be ‘1’ or ‘0’.

`SymID`
Assigned symmetry ID (i.e., 0, 1,2,3), if previous "isSkipSymCheck" is set to be 1.

`isSkipOutEnsemble`
Whether the step of outputting the whole ensemble of packed is skipped. If this step is skipped, only the mean structure of all packed structures is outputted.

`InputFileLocation`
Location of the input ensemble of SSE fragments in the all-to-one packing case.

addSideChains addSideChains.input

This commands places the side-chains onto backbone using the NOE pattern-matching technique. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`haErr, h1Err, c13Err, hnErr, nErr`
Error windows in each dimension (ppm).

`resFormat`
Format of resonance assignments. It can be ‘CYANA’ format, ‘BMRB’ format or ‘XEASY’ format.

`resNameScheme`
Atom naming scheme. It can be ‘BMRB-OLD’, ‘BMRB-NEW’, ‘PDB-NEW’ or ‘PDB-OLD’.

`resonance`
File name of resonance assignment list.

`NOESY-Format`
Format of NOESY cross peak list. It can be ‘XEASY’ format or ‘NMRVIEW’ format.

`3D-N15-NOESY`
File name of 3D 15N-labeled NOESY cross peak list.

`3D-C13-NOESY`
File name of 3D 13C-labeled NOESY cross peak list.

`LocationBBEnsemble`
Location of backbone ensemble.

`outStructureName`

Name of the output structure name (including both backbones and side-chains).

addRandomLoops addRandomLoops.input

This command simply merges the SSE PDB and loop PDB into a PDB that contains the whole structure. It prepares the input PDB files to Xplor in the local minimization approach. A typical input parameter file includes the following parameters:

`LocationSSEEnsemble`
Location of the input SSE ensemble.

`LoopPDBName`
File name of loop PDB.

`outCompleteStructures`
File name of output structures.

HANA_NOE_Asg HANA_NOE_Asg.input

This command executes the HANA NOE assignment algorithm given the backbone solved from RDC-EXACT. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`haErr, h1Err, c13Err, hnErr, nErr`
Error windows in each dimension (ppm).

`Backbone`
PDB file name of the backbone.

`PdbNameScheme`
Atom naming scheme of backbone PDB. It can be ‘BMRB-OLD’, ‘BMRB-NEW’, ‘PDB-NEW’ or ‘PDB-OLD’.

`isWholeStructure`
Whether the whole structure (i.e. including sidechains and backbone) is used for pruning ambiguous NOEs. It is set to be 1, if the original backbone PDB file is used for pruning ambiguous NOEs.

`resFormat`
Format of resonance assignments. It can be ‘CYANA’ format, ‘BMRB’ format or ‘XEASY’ format.

`resNameScheme`
Atom naming scheme. It can be ‘BMRB-OLD’, ‘BMRB-NEW’, ‘PDB-NEW’ or ‘PDB-OLD’.

`resonance`
File name of resonance assignment list.

`NOESY-Format`
Format of NOESY cross peak list. It can be ‘XEASY’ format or ‘NMRVIEW’ format.

`3D-N15-NOESY`
File name of 3D 15N-labeled NOESY cross peak list.

`3D-C13-NOESY`
File name of 3D 13C-labeled NOESY cross peak list.

`noeLimit`
Distance upper limit used for pruning ambiguous NOE assignment. The original distance bound calibrated if the value of noeLimit is negative.

`isOutORFormat`
Whether multiple NOE assignments (assigned to the same NOESY cross peak) are outputted in OR format.

`outNoeName`
File name of the output NOE table, which is in the standard Xplor format. The default atom name scheme used in the output NOE assignment table is PDB-NEW.

`IsOriginalUp`
In the output NOE table, whether the original NOE upper bounds (i.e. calibrated from NOESY peak intensity) are displayed. The calibrated distance is used, if the parameter is set to be 1. If the parameter is 0, the distances from the structure after combining side-chains and backbone are used as the NOE upper bounds.

`IsCheck`
Whether checking the NOE assignment results by comparing them with the reference structure:

`refPdb`
File name of the reference structure PDB, when ‘ischeck’ is 1.

`refNameScheme`
Atom naming scheme for the reference structure. It can be ‘BMRB-OLD’, ‘BMRB-NEW’, ‘PDB-NEW’ or ‘PDB-OLD’.

`CalAlignmentTensor calAlignmentTensor.input`

This command computes the alignment tensor given CH and NH RDCs, and PDB. Although CaC’ and NC’ RDCs can be inputted into the program, only CH and NH RDCs are used for computing alignment tensor. However, the RMSD values between experimental and back-computed RDCs for CaC’, NC’, CH and NH RDCs are all reported. A typical input parameter file includes the following parameters:

`chFile`
File name of CH RDCs.

`nhFile`
File name of NH RDCs.

`cocaFile`
File name of CaC' RDCs. This parameter is set to be 'null' if CaC' RDCs are not available.

`conFile`
File name of NC' RDCs. This parameter is set to be 'null' if NC' RDCs are not available

`InputPdbName`
File name of input PDB.

NOEAsgFromCS asg_noe_cs.input

This command assigns NOEs based on only chemical shift information. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`haErr, h1Err, c13Err, hnErr, nErr`
Error windows in each dimension (ppm).

`resFormat`
Format of resonance assignments. It can be 'CYANA' format, 'BMRB' format or 'XEASY' format.

`resNameScheme`
Atom naming scheme. It can be 'BMRB-OLD', 'BMRB-NEW', 'PDB-NEW' or 'PDB-OLD'.

`resonance`
File name of resonance assignment list.

`NOESY-Format`
Format of NOESY cross peak list. It can be 'XEASY' format or 'NMRVIEW' format.

`3D-N15-NOESY`
File name of 3D 15N-labeled NOESY cross peak list.

`3D-C13-NOESY`
File name of 3D 13C-labeled NOESY cross peak list.

`isUnique`
Whether only those unique NOE assignments are outputted, or all ambiguous NOE assignments are outputted.

`isOutORFormat`
Whether multiple NOE assignments (assigned to the same NOESY cross peak) are outputted in OR format.

`outNoeName`
File name of the output NOE table, which is in the standard Xplor format. The default atom name scheme used in the output NOE assignment table is PDB-NEW.

CompareTwoNOETables CompareTwoNOETables.input

This command compares two NOE table and outputs the summary. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`NOEFormat`
Format of source NOE table. It can be ‘XPLOR’ format or ‘CYANA’ format.

`NOE-Table`
File name of source NOE table for comparison.

`RefNOEFormat`
Format of reference NOE table. It can be ‘XPLOR’ format or ‘CYANA’ format.

`Ref-NOE-Table`
File name of reference NOE table for comparison.

ReadPDBEnergy ReadNoeEnergy.input

This command reads the energy values from Xplor PDB files and output the top structures with lowest energies. A typical input parameter file includes the following parameters:

`LocationSSEEnsemble`
Location of the input Xplor PDB ensemble.

`IsCutOff`
Whehter a cutoff is used for selecting top ensemble of structure.

`Cutoff`
The cutoff of energy if IsCutOff is '1'.

`OutputNumber`

If the cutoff is not used, the top number of structures with lowest energies.

`rowPos, colPos`
Position of energy terms in the REMARK section of Xplor PDB files, starting from 0.

`isOutput`
Whether the same PDB with the remark is outputted.

`OutName`
File name of output PDBs.

CheckNoeByEnsemble CheckNoeByEnsemble.input

This command uses the voting scheme to obtain the consensus NOE assignments, which are consistent with a majority of the structure in the ensemble. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`InputPdbLocation`
Location of the input ensemble of PDBs used for pruning NOE assignments.

`PdbNameScheme`
Atom naming scheme of backbone PDB. It can be 'BMRB-OLD', 'BMRB-NEW', 'PDB-NEW' or 'PDB-OLD'.

`NOEFormat`
Format of source NOE table. It can be 'XPLOR' format or 'CYANA' format.

`NoeAtomNamingScheme`
Atom naming scheme in the NOE table.

`InputNoeTable`
File name of the input NOE table:

`isMultiAssignment`
Whether the input NOE table is in the multiple assignment format

`noeLimit`
Upper limit for pruning ambiguous NOEs. When it is negative, the original upper bound calibrated from peak intensity is used.

`NoeCutOff`
Percentage threshold of the structures in the ensemble for checking the consistency of NOE assignments. Suppose the NoeCutOff is 0.6. If more than 60% of structures in the ensemble are consistent with an NOE assignment, this NOE assignment is outputted. Otherwise it is pruned.

`OutNOETable`
File name of new NOE table after deleting those violated NOEs.

`isOutVioNOEs`
Whether those violated NOEs are outputted as the remarks.

`isOutMultiAsgFormat`
Whether new NOE assignments are outputted in the multiple NOE Assignment format.

NoeStatistics NoeStatistics.input

This command gives the statistical summary of an NOE assignment table. A typical input parameter file includes the following parameters:

`sequence`
File name of protein primary sequence.

`NOEFormat`
Format of source NOE table. It can be ‘XPLOR’ format or ‘CYANA’ format.

`NoeAtomNamingScheme`
Atom naming scheme in the NOE table.

`InputNoeTable`
File name of the input NOE table.

`isMultiAssignment`
Whether the input NOE table is in the multiple assignment format

mergeNClusterAll mergeNClusterAll.input

This command performs the clustering step over all packed structures in a single directory. A typical input parameter file includes the following parameters:

`InputFileLocation`
Location of the input ensemble directory.

`outPackedStructures`
Name of the output packed structure (only the common name of all structures in the enemble). For exameple, the name “H2H1New” will generate an ensemble of structures with file names H2H1New1.pdb, H2H1New3.pdb, H2H1New3.pdb, etc.

`resolCluster`
Resolution used in the clustering step.

7. Examples

Two examples, including tests on ubiquitin and FF2 structures, come with the distribution of RDC-PANDA (version 1.0). The input data, parameter files and shell scripts for running these two examples can be found in sub-directory "./experiments/". The details on running RDC-ANALYTIC on these two proteins can be found in the RDC-ANALYTIC manual (Section 6, page 9).

To run these two examples, namely testing RDC-PANDA on proteins ubiquitin and FF2, first clear the following directory and make them empty:

./inputFiles/

./xplor/local_min/

./xplor/final_cal/

and then type

chmod 700 test_NAME
./test_NAME

in which NAME can be ff2, ubq and eta for each testing case.

For example, to test RdcPanda on ubq protein, type

chmod 700 test_ubq
nohup ./test_ubq

The locations of intermediate results can be found in remark sections of the shell scripts. It is recommended that different testing cases should run in different RDC-PANDA working directories. The following gives an example of the shell scripts for running the ubiquitin test:

Shell Script File "test_ubq":

```
#! /bin/sh

#################################################################
#1.initialization:
#################################################################

#(1.1)set the XPLOR-NIH path
xplor=/usr/project/dlab/Users/Software/xplor-NIH-64/xplor-nih-2.16.0/bin/xplor

#(1.2) Empty the inputfile directory;
#to be safe, it would be better to empty these directories manually.
rm -rf $PWD/inputFiles/*
rm -rf $PWD/xplor/*

#(1.3) Copy the input files from experiments directory;
```

```
cp -av $PWD/experiments/ubq/inputFiles/* $PWD/inputFiles/

#(1.4) Copy the Xplor scripts to the Xplor working directory.
cp -av $PWD/experiments/ubq/xplor/* $PWD/xplor/


#################################################################
#2. Run RDC-Analytic program to compute the SSE backbones from RDCs.
#################################################################
#go to RDC-ANALYTIC directory, and compute SSE backbones.
cd  ../RdcAnalytic/

java ./analytic/RDCAnalytic > RDC-ANALYTIC.out

#copy the PDB files from RDC-ANALYTIC to the RDC-PANDA input file directory.
cp –av ./helix1.pdb ../RdcPanda/inputFiles/
cp –av ./sheet1.pdb ../RdcPanda/inputFiles/

#go to RDC-PANDA directory.
cd ../RdcPanda/

#################################################################
#3. Use Packer to pack SSEs:
#################################################################
#After computing comformations and orientations of SSE bacbone fragments,
#save the pdb of each SSE backbone fragment in the directory './inputFiles'.
#Then run the following command for packing helix and beta-sheet:
java -cp ./bin/ RdcPanda SSEPacking packingSSEs.input > packingSSE.out

#The ensemble of all packed structures will be stored in directory
#'./inputFiles/packing/'.


#################################################################
#4. Use HANA to place side-chains.
#################################################################
#Run the following command line to place side-chains onto all packed SSE backbone
#(i.e. core structures):
java -cp ./bin/ RdcPanda addSideChains addSideChains.input >addSideChains.out

#The ensemble of outputted packed SSE backbones with computed side-chains will be
#stored in directory './inputFiles/bbwsc/'.

#################################################################
#5. Compute the loops.
#################################################################
#First, random loops are added into the core structures (i.e. packed SSE backbones
#with side-chains), using the following commands:
java -cp ./bin/ RdcPanda  addRandomLoops addRandomLoops.input > addRandomLoops.out

#The ensemble of complete structures (with random loops) will be ouputted in
#directory './xplor/local_min/structure',
#which will be used as initial templates for the local minimization approach in
#Xplor. Then go to the directory   './xplor/local_min/', and run the following three
#Xplor scripts for the local minimization:
cd $PWD/xplor/local_min/
$xplor < min.inp > min.out
```

```
$xplor < dgsa.inp > dgsa.out
$xplor < refine.inp > refine.out



#Ensemble of final complete structures (both SSEs and loops) after the local
#minimization approach will be stored in  './xplor/local_min/final/'.

# (5.1) Use following Xplor-NIH script to evaluate the well-packed structures
#(WPSs).
$xplor -py calEnergy.py > computeWPS.out

#compute average structures for ensemble of packed structures computed by PACKER
#the average structure will be store in ./xplor/local_min/average_all.pdb
$xplor < average_all.inp > average_all.out

#compute average structures for ensemble of well-packed structures
#the average structure will be store in ./xplor/local_min/average_wps.pdb
$xplor < average_wps.inp > average_wps.out



#################################################################
#6. Compute the long loop in residues 50-65.
#################################################################
# Note: For short loops, we can directly obtain NOE
# assignments from the mean structures of the ensemble.

cd ..
cd ..

cp -av ./xplor/local_min/average_all.pdb ./xplor/final_cal/

#extract the unique NOE assignments for long loops in residues 33-41,
#using only chemical shift information.
#The result will be stored in  /xplor/final_cal/ubiquitin_noe_loops1_0.tbl.
java -cp ./bin/ RdcPanda NOEAsgFromCS asg_noe_cs1.input > LongLoopsNOEAsg1.out


#extract the unique NOE assignments for long loops in residues 50-65,
#using only chemical shift information.
#The result will be stored in  /xplor/final_cal/ubiquitin_noe_loops2_0.tbl.
java -cp ./bin/ RdcPanda NOEAsgFromCS asg_noe_cs2.input > LongLoopsNOEAsg2.out

#extract SSE fragments (including short loops), that is, excluding two long loops.
java -cp ./bin/ RdcPanda ReadPdbFragments readPdbFragments.input >
readPdbFragments.out
java -cp ./bin/ RdcPanda ReadPdbFragments readPdbFragments2.input >
readPdbFragments2.out

#Use HANA to compute NOE assignments in SSE and short loop regions.
#The result will be stored in /xplor/final_cal/ubiquitin_noe_sses.tbl.
java -cp ./bin/ RdcPanda HANA_NOE_Asg HANA_NOE_Asg_SSE.input > HANA_NOE_ASG_SSE.out

###################
#iteration 1 for computing long loops in residues 33-41 and residues 50-65:
cd $PWD/xplor/final_cal/
```

```
cp -av ./ubiquitin_noe_loops1_0.tbl ./ubiquitin_noe_loops1.tbl
cp -av ./ubiquitin_noe_loops2_0.tbl ./ubiquitin_noe_loops2.tbl

#empty intermediate directories:
rm -rf ./pdb/*

rm -rf ./accept/*

rm -rf ./top/*

$xplor < ubq_loops_fix.inp > compute_long_loops_1.out

cp -av ./pdb/* ./top/

###################
#iteration 2 for computing long loops:
cd ..
cd ..

#refine those unique NOE assignments in long loop regions (other NOEs are unchanged)
java -cp ./bin/ RdcPanda CheckNoeByEnsemble CheckNoeByEnsemble_loops1.input >
checkLoopNoeByEnsemble_1.out

java -cp ./bin/ RdcPanda CheckNoeByEnsemble CheckNoeByEnsemble_loops2.input >
checkLoopNoeByEnsemble_2.out

cd $PWD/xplor/final_cal/

#empty intermediate directories:
rm -rf ./pdb/*

rm -rf ./accept/*

rm -rf ./top/*
$xplor < ubq_loops_fix.inp > compute_long_loops_2.out

#move all 10 structures to ./xplor/final_cal/top/.
cp -av ./pdb/* ./top/

#compute the inital structure template for final structure calculation:
$xplor < average_top_fix.inp > average_top_fix2.out

#Go to RdcPanda working directory:
cd ..
cd ..

###################
#iteration 3 for computing long loops:
java -cp ./bin/ RdcPanda HANA_NOE_Asg HANA_NOE_Asg_fix.input > HANA_NOE_ASG_fix.out

cd $PWD/xplor/final_cal/
cp -av ./ubiquitin_noe.tbl ./ubiquitin_noe_0.tbl

#empty intermediate directories:
rm -rf ./pdb/*
```

```
rm -rf ./accept/*

rm -rf ./top/*

$xplor < ubq_loops_fix2.inp > compute_long_loops_3.out

#move all 10 structures to ./xplor/final_cal/top/.
cp -av ./pdb/* ./top/

#compute the inital structure template for final structure calculation:
$xplor < average_top_fix.inp > average_top_fix3.out

#Go to RdcPanda workding directory:
cd ..
cd ..

######################################################################
#7. Final structure calculation
######################################################################

#########################
# Round 1:

# Use HANA to compute the NOE assignments.
java -cp ./bin/ RdcPanda HANA_NOE_Asg HANA_NOE_Asg_fix.input > HANA_NOE_ASG_1.out


cd $PWD/xplor/final_cal/

cp -av ./ubiquitin_noe.tbl ./ubiquitin_noe_0.tbl

#empty intermediate directories:
rm -rf ./pdb/*

rm -rf ./accept/*

rm -rf ./top/*
$xplor < ubq_1.inp > final_cal_1.out

#Go to RdcPanda workding directory:
cd ..
cd ..
#move all 10 structures to ./xplor/final_cal/top/.
java -cp ./bin/ RdcPanda ReadPDBEnergy ReadNoeEnergy.input > ReadEnergy_final_1.out


cd $PWD/xplor/final_cal/
$xplor < average_top_1.inp > average_top_1.out

#Go to RdcPanda workding directory:
cd ..
cd ..

#########################
#Round 2:
```

```
java -cp ./bin/ RdcPanda CheckNoeByEnsemble CheckNoeByEnsemble.input >
checkNoeByEnsemble_2.out

cd $PWD/xplor/final_cal/

#empty intermediate directories:
rm -rf ./pdb/*

rm -rf ./accept/*

rm -rf ./top/*
$xplor < ubq_1.inp > final_cal_2.out

#Go to RdcPanda workding directory:
cd ..
cd ..
#move all 10 structures to ./xplor/final_cal/top/.
java -cp ./bin/ RdcPanda ReadPDBEnergy ReadNoeEnergy.input > ReadEnergy_final_2.out

cd $PWD/xplor/final_cal/
$xplor < average_top_2.inp > average_top_2.out

#Go to RdcPanda workding directory:
cd ..
cd ..
```

---

8. RDC-PANDA Class Summary

This section provides a summary of the RDC-PANDA classes, which will be useful for users to modify and extend the RDC-PANDA source code. More details about the definitions of the RDC-PANDA classes and methods in the java source code can be found in sub-directory "./doc/javadoc/" (in html format).

- **Assign**: This class provides functions that involve NOE assignment, such as initial NOE assignment using only chemical shift information, and prune ambiguous NOE assignments using given the structural information, etc.

- **BackNoe**: This class provides functions that involve back-computed NOEs.

- **Cartesian:** This class defines Cartesian coordinates for individual atom in protein structure.

- **Const:** This class defines various global constants for the entire program.

- **Dipolar:** This class provides data structures processing experimental RDC data.

- **EigenvalueDecomposition:** This class is extended from the Jama package, which provides the methods for the eigenvalue decomposition of a matrix.

- **H1CS:** This class provides data structures that involve chemical shifts.

- **Hbond:** This class defines data structures related to h-bonds.

- **Hdist:** This class provides data structures related to back-computed NOE distance restraints.

- **IDof2aryStructure:** This class provides functions for processing files with secondary structure element information.

- **Maths:** This class is modified from the Jama numeric package, which provides some math operations that are used in the program.

- **Matrix:** This class is modified from the Jama numeric package, which provides some matrix operations that are used in the program.

- **Model:** This class provides functions that compute backbone from RDC-EXACT algorithm.

- **ModelRdc:** This class provides methods for computing all the backbone dihedral (phi,psi) angles of an alpha-helix or a beta-strand based on exact solution and a systematic search.

- **Noe**: This class provides data structures and functions for processing NOE assignment.

- **Noesy**: This class provides data structures and functions for processing NOESY spectra.

- **Pdb:** This class provides data structures and functions for processing and generating PDB files and extracting backbone angles, etc.

- **PdbRdc:** This class provides various SVD methods for best fitting RDCs and a structural fragment, and grid search method for computing the three Euler angles.

- **PhiPsi:** This class provides various methods for exact solution and systematic search including the computation of rotation to the first peptide plane, the DFS-search and solver of the quartic equation and the computation phi and psi angles from two vectors in consecutive planes, etc.

- **Rotamer:** This class provides data structures and functions related to rotamers.

- **RotaPattern**: This class implements data structures and functions that deal with the rotamer properties, such as NOE pattern for each rotamer.

- **SingularValueDecomposition:** This class provides functions for SVD operations (from Jama package).

- **SSEPacking:** This class provides functions for packing SSE fragments.

- **vdw:** This class provides methods for computing the vdw energy term.