ProtocolCaptureJufo9D

- Overview
- Background
- Protocol
    - Environment
    - Dataset creation - MP Chains
    - Dataset creation - soluble chains
    - Preparation of input files
    - Create datasets for cross-validation
    - Generate ANN input files
    - Train ANNs
    - Analyze output
    - Create single chain PDBs to compare to other methods
    - Run OCTOPUS and TMbetaNet as comparison
    - Comparison between methods
    - Create topology prediction
    - Compare eukaryotic and prokaryotic prediction accuracies
    - Compute kink prediction accuracies for BCL::Jufo9D and other methods

Overview

This is the protocol capture page for Jufo9D as described in *Koehler-Leman et al., Proteins, Structure, Function, and Bioinformatics, 2013.*

http://onlinelibrary.wiley.com/doi/10.1002/prot.24258/abstract;jsessionid=230D18CEAE744F065ADC2F8101A28922.d04t03

Background

Link to be included: Koehler-Leman et al., Proteins, Structure, Function, and Bioinformatics, 2013 once available.

Protocol

Environment

Use these commands to setup your environment to replicate this protocol capture.

If you are replicating this protocol capture, first copy the folder elsewhere and update paths accordingly

```
/bin/tcsh
cd /blue/meilerlab/protocols/Jufo9D/2012-06-01-Nominal Title/
source protocol_capture_cshrc
```

Dataset creation - MP Chains

A list of all membrane protein chains, for which a structure has been determined, was downloaded from the Pdbtm [41-42] website (Nov. 2011). Similar sequences were excluded by culling this list with the PISCES server [43-44]. The parameters included a percent sequence identity 30%, resolution 0 – 3 Å, R-factor 0.25, sequence length 40 – 10,000 residues, non X-ray entries as well as CA-only chains were included. BCL::PDBConvert (Woetzel, N. submitted) was used to convert non-natural amino acids into their natural counterparts and to transform the protein into the membrane coordinate frame using the membrane definitions provided by the Pdbtm website. The membrane normal aligns with the z-coordinate in the PDB file with the membrane center being at z = 0. We assume a constant thickness of 20 Å for the membrane core and 10 Å for the transition region on either side of the membrane (Figure 1A). Residues in the 2.5 Å gap regions between membrane core and transition region or transition region and solution were disregarded to obtain more distinct regions for the ANN to identify. DSSP [45] (version of 2011) was used for all PDB structures to obtain a consistent SS identification. Helices with less than five residues and strands with less than three residues were disregarded to focus the prediction on long SS elements. This procedure resulted in a list of 226 chains in 177 membrane proteins.

Step	Commands	Comments	Outputs

download list of TM chains		download list of all TM chains from PDBTM website ([http://pdbtm.enzim.hu/?m=download)]	sorted output should be pdbtm_all_chains_2011-12-07_sorted.ls
cull chains for sequence identity	Your thresholds for culling selected PDB list: Sequence percentage identity: <= 30% Resolution: 0.0 ~ 3.0 R-factor: 0.25 Sequence length: 40 ~ 10000 Non X-ray entries: Included CA-only entries: Included Cull PDB by chain	cull chains using PISCES using these parameters	cullpdb_pc30_res3.0_R0.25_inclNOTXRAY_ inclCA_d120229_chains304.6610
remove unwanted chains		remove EM structures and N/A or proteins with unknown resolution	current_MP_chains.txt

Dataset creation - soluble chains

A pre-compiled list of PDB chains was downloaded from the PISCES protein sequence culling server (date 12/02/2011) [43-44]. The list contained sequences with a percentage sequence identity 30%, resolution 0 – 2 Å, R-factor 0.25, sequence length 40 – 10,000 residues, non X-ray entries as well as CA-only chains were excluded. Membrane proteins were excluded from this list. BCL::PDBConvert (Woetzel, N. submitted) was used to convert non-natural amino acids into their natural counterparts and DSSP [45] was used to identify SS elements. Helices shorter than five residues and strands shorter than three residues were disregarded. The result was a list of 6,223 chains in 6,048 soluble proteins.

Step	Commands	Comments	Output
download list of chains		download file for all proteins from PISCES website (http://dunbrack.fccc.edu/Guoli/pisces_download.php)	cullpdb_pc30_res2.0_R0.25_d111202_chains6540
remove unwanted chains		remove all MP chains - cull against pdbtm_all_chains_2011-12-07_sorted.ls remove EM structures and N/A or proteins with unknown resolution remove chains shorter than 40 residues	current_sol_chains_longer39.ls

Preparation of input files

Step	Commands	Comments	Output
run PDBConvert	001_PDBConvert_convertAAs.txt	runs it for all MP chains to get fasta and to convert to natural AA type (DSSP requires that) correct U, Z, B in fasta into X	
run DSSP	002_run_dssp_over_db.pl		output goes into database_MPs/
download xml files	003_download_xml.txt	download xml files for MPs	output goes into database_MPs/
create biomolecule	004_PDBConvert_biomolecule.txt check_pdb_for_membrane_position_database.pl or check_pdb_for_membrane_position.pl	run PDBConvert to create biomolecule check for correct membrane orientation, either for individual pdb or for database of pdbs	
run SSPred	007_runss.txt	runs SSPrediction for PsiPred and ProfPhD	creates .ascii, .ascii6, .rdb, and .psipred_ss2 files in the database

All steps above should be carried out for both the MP database and the soluble protein database.

Step	Commands	Comments	Output
pairwise sequence alignment	005_sequence_alignment_calc_seqid.pl	make pairwise sequence alignments (because PISCES does not remove ALL similar sequences) output is analyzed in alignments_MPvsMP.txt<$thread>, all chains with sequence similarity higher than 90% are clustered to get only a single representative of these	alignments/
analyze clusters	010_analyze_clusters.pl	analyze clusters using cutoff of 90% clusters can be visualized using 011_visualize_clusters.pl removed chains after clustering: removed chains are in clustering/remove_chains_after_clustering.ls	clustering/clusters_MP_seqids_90.out and current_MP_chains_after_clustering.txt
create oligomeric state dictionary	012_create_oligomeric_state_dictionary.pl		oligomeric_state_dictionary.txt

Create datasets for cross-validation

The databases were split into five subsets for cross-validation. For the membrane proteins -helical bundles as well as -barrels were distributed as equally as possible. The soluble proteins were distributed randomly.

To train a single ANN, three of the five subsets were used for training (see Figure 1B) and one subset was used for monitoring the training process to avoid overtraining. The fifth subset was used as an independent test set for computing the prediction accuracies. 20 networks were trained such that the independent as well as the monitoring permuted through the five datasets (Figure 1B).

Step	Commands	Comments	Output
assign alpha/beta to chains	013_add_folds_into_chains_file.pl	create file that contains alpha/beta distinction in the membrane for the proteins	datasets_all.ls
distribute into 5 subsets		distribute into 5 subsets: distribute A-folds and B-folds from datasets_all.ls into 5 subsets using create_dataset.pl distribute chains from current_sol_in_datasets_chains.ls randomly into subsets these files are the input files for creating the ANN input files!!!	dataset1.ls ... dataset5.ls

Generate ANN input files

Figure 2 shows the input parameters used (see Supplementary Table S1): (a) five amino acid properties including steric parameter, volume, polarizability, iso-electric point, solvent-accessible surface area [10]; (b) six free energies for SS type (helix, strand, coil), residue environment (membrane bilayer, interface, solution) [25] and the nine combinations of both; (c) the position-specific scoring matrices (PSSM) from PsiBlast [8] after six iterations (see [46]). For each residue all of these parameters were collected over a sequence window of 31 residues. The optimal size of the input window was determined by testing all odd window sizes between 15 and 39 residues.

In addition, "global" parameters were considered for each protein: (a) the number of residues in the protein chain; (b) the oligomeric state (monomer vs. oligomer); (c) the average of all amino acid specific parameters over the entire protein chain including their properties, free energies, and the PSSM values. This resulted in (31 residues x (20 numbers from PSSM + 20 amino acid properties)) + (2 parameters: oligomeric state, length) + (40 averages) = 1282 input parameters that represent the residue at the center of the window.

Step	Commands	Comments	Output
generate descriptors	014_generate_descriptors1.txt	generate descriptors as input to ANNs	descriptors_PDB_9D_10000_w31.dat
convert to binary file	bclWill.exe GenerateDataset -source 'File(filename=descriptors_PDB_9D_10000_w31.dat,number chunks=1,chunks=[0])' -output descriptors_PDB_9D_10000_w31.bin	convert it into a binary file for easier reading	descriptors_PDB_9D_10000_w31.bin

Train ANNs

The datasets (the term "dataset" corresponds to the input and output parameters for each residue in a protein sequence) were randomized and balanced for each protein subset independently. For balancing, an over-sampling procedure was used to represent each of the nine states equally often and avoid a bias in the predictions towards the more abundant states. This approach also increases the entropy in the input data and maximizes the information gain the ANN can achieve.

The ANNs were three-layer feed-forward networks with a sigmoidal activation function and trained through back-propagation of errors. The hidden layer contained 32 neurons – a number that was optimized by testing 4, 8, 16, 32, 64, and 128 neurons. The three subsets used for training contained a total of 270,000 instances, 90,000 instances were in the monitoring dataset, and 90,000 instances in the independent dataset. The training protocol consisted of three consecutive steps using a simple propagation algorithm: (1) 50 steps with weight update after each step with momentum = 0.0 and the learning rate = 10^{-3}; (2) 10 steps with batch update with momentum = 0.5 and the learning rate = $5 \cdot 10^{-6}$; (3) 100 steps with weight update after each step with momentum = 1.0 and the learning rate = $5 \cdot 10^{-6}$. As a post-processing step the outputs of the four ANNs were averaged that used the same independent subset.

Step	Commands	Comments	Output
create directories		train 20 ANNs: 5 subsets: first number 0..4 for independent set, second number (unequal first number) 0..4 for monitoring set created different directories with these numbers: 01, 02, ..., 43	train_10_combi/
training run 1	bcl-apps-static_03192012.exe TrainModel 'NeuralNetwork(transfer function = Sigmoid, weight update = Simple(eta=0.001,alpha=0), objective function = RMSD, steps per update=1, hidden architecture(32))' -training 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[0,5)~~[0]~~[1]")' -monitoring 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[1]")' -independent 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[0]")' -print_training_predictions descriptors_PDB_9D_10000_w31_2012-03-16.train0 -print_monitoring_predictions descriptors_PDB_9D_10000_w31_2012-03-16.mon0 -print_independent_predictions descriptors_PDB_9D_10000_w31_2012-03-16.ind0 -feature_labels features_code_1282.object -result_labels results_code_9D.object -scheduler Serial -final_objective_function RMSD -storage_model 'File(directory=.)' -max_iterations 50	for each ANN trained for 3 consecutive runs (here example for 01, see 01/run0.log ... 01/run2.log) 1st training run	000000.model
training run 2	bcl-apps-static_03192012.exe TrainModel 'NeuralNetwork(initial network file = 000000.model, transfer function = Sigmoid, weight update = Simple(eta=0.000005,alpha=0.5), objective function = RMSD, steps per update=0, hidden architecture(32))' -training 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[0,5)~~[0]~~[1]")' -monitoring 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[1]")' -independent 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[0]")' -print_training_predictions descriptors_PDB_9D_10000_w31_2012-03-16.train1 -print_monitoring_predictions descriptors_PDB_9D_10000_ w31_2012-03-16.mon1 -print_independent_predictions descriptors_PDB_9D_10000_w31_2012-03-16.ind1 -feature_labels features_code_1282.object -result_labels results_code_9D.object -scheduler Serial -final_objective_function RMSD -storage_model 'File(directory=.)' -max_iterations 10	2nd training run	000001.model

training run 3	bcl-apps-static_03192012.exe TrainModel 'NeuralNetwork(initial network file = 000001.model, transfer function = Sigmoid, weight update = Simple(eta=0.000005,alpha=1), objective function = RMSD, steps per update=0, hidden architecture(32))' -training 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[0,5)~~[0]~~ [1]")' -monitoring 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[1]")' -independent 'Subset(filename=descriptors_PDB_9D_10000_w31_2012-03-16.bin,number chunks=5,chunks="[0]")' -print_training_predictions descriptors_PDB_9D_10000_w31_2012-03-16.train2 -print_monitoring_predictions descriptors_PDB_9D_10000_w31_2012-03-16.mon2 -print_independent_predictions descriptors_PDB_9D_10000_w31_2012-03-16.ind2 -feature_labels features_code_1282.object -result_labels results_code_9D.object -scheduler Serial -final_objective_function RMSD -storage_model 'File(directory=.)' -max_iterations 100	3rd training run	000002.model

Analyze output

To report the prediction accuracies as well as the confidence measure, an average of four network outputs was computed only considering ANNs belonging to a single set that share the same independent dataset (Figure 1B). This setup (a) ensures that the reported accuracies originate from ANNs that were not trained on the test set, and (b) prediction accuracies can be reported for each protein in the dataset as always four ANNs exist that were trained with this particular protein in the independent dataset. The final output from the web-server is the average over all 20 ANNs and computes a confidence measure that constitutes the difference between the highest and second highest output probability for each residue.

To calculate the per-residue prediction accuracies, the outputs of the four ANNs in a single set were averaged. The outputs per set were compared to the actual state on a per-residue basis: if the predicted state was a TM helix and the actual state was a TR helix, the counts in this particular 9x9 matrix element (see Figure 3) was increased by one. After obtaining all counts for the 9x9 matrix over a single set, the counts were divided by the number of residues in this region (sum over each row) to arrive at the percentage of predicted residues in each matrix element. The percentages of predicted residues were then averaged over the five sets of ANNs. This cross-validation and averaging procedure circumvents that a “bad choice” of proteins in an independent dataset biases the prediction accuracies.

The counts for the three-state SS prediction, three-state TM span prediction, or two-state TM helix/TM strand prediction were calculated as described in Supplementary Figure S1. The counts were divided by the total number of counts per row to arrive at the percentages of predicted residues and these percentages were later averaged over the five sets of ANNs.

Step	Commands	Comments	Output
create test input files		create and test input files with/without gaps and minSSEsize • commandlines are in readme_create_dataset_gaps2.5_minSSEsize0.txt and readme_create_dataset_gaps2.5_minSSEsize3.5.txt • they create testModel* files in ANN directories	
analyze using ANN model files	019_analyze_overprediction_9D_newformat.pl or 027_analyze_overprediction_9D_all-in-line.pl	analyze predicted output that was generated using the ANN model files • if actual output and predicted output is in two different lines: use 019... • if actual output and predicted output is in one line: use 027... • this is for a single output file from a single ANN	
combine outputs from 4 ANNs	020_avg_ANN_outputs_to_1_prediction.pl or 029_avg_ANN_outputs_to_1_prediction_all-in-line.pl	combine and analyze outputs from 4 different ANNs which all have the same independent dataset • if actual output and predicted output is in two different lines: use 020... • if actual output and predicted output is in one line: use 029... • this is written into train_10k_combi32/analysis*txt files which are analyzed in Excel, these tables are in the manuscript	

Create single chain PDBs to compare to other methods

Only single chains can be used as input to other methods, therefore creation of single chain PDBs is required.

Step	Commands	Comments	Output
create single chains	023_PDBConvert_create_chains.txt	create single chains for PDB files to compare to other methods • individual chains can be tested and analyzed using 026_test_chains_with_jufo_sumpred_analyze.pl where the output of 4 ANNs is taken as the final output	

Run OCTOPUS and TMbetaNet as comparison

Submits a list of fasta files to the servers, downloads, and reformats the output.

Step	Commands	Comments	Output
run Octopus	016_octopus_run_from_fasta.pl	run OCTOPUS for list of input files runs via 015_octopus_submit.pl for Octopus	written to database
run TMBetaNet	018_tmbetanet_run_from_fasta.pl	run TMBETANET for list of input files runs via 017_tmbetanet_submit.pl for TMbetanet	written to database
reformat TMBetaNet	reformat_tmbetanet.pl	reformat the TMbetanet output to remove single empty lines at beginning of the file	

Comparison between methods

Step	Commands	Comments	Output
run statistics for BCL::Jufo9D	run_statistics2.txt	compare Jufo to other methods obtains statistics from individual .jufo9d output files	statistics_other_methods/
calculate per-residue accuracies	024_calculate_per-res-acc_avg_stdev_for_methods.pl	analyzes output for 2-state output	
calculate avg and stdev	028_calculate_avg_stdev_for_methods.pl	analyzes output for 3- and 9-state outputs	
get 2-state accuracies for BCL::Jufo9D	034_avg_ANN_outputs_to_2-state_all-in-line.pl	get 2-state accuracies for Jufo prediction use 034... which requires that the ANN actual output and predicted output is in one line	
calculate information gain	038_calculate_infogain.pl	calculate information gain this takes outputs*_avgANNs.txt as input files which is from complete protein sequences, unbalanced, no gaps, no minSSEsizes	results_infogain.txt

Create topology prediction

Topology prediction is not the actual topology which indicates the inside/outside orientation of a MP. It rather indicates TM-helix and TM-strand spans along the protein sequence.

To directly compare the nine-state output of BCL::Jufo9D to the two-state output of, for instance Octopus, we summed the non-TM helix probabilities to arrive at a two-state prediction (Supplementary Figure S1). To remove the resulting bias towards non-TM states from adding background probabilities of 11.1%, the result needs to be corrected by adding or subtracting ½•(8•11.1%-11.1%)=38.9% from the two states, respectively. This procedure ensures that the total of all prediction probabilities remains 100%. The identical correction was applied to the TM strand prediction.

Furthermore, a post-processing step has been applied for noise reduction. Lengths of SS elements were calculated where kinks of one or two residues were regarded as TM helix residues but were retained in the final prediction. Since the topology prediction output considers only long SS elements that can span the membrane, helices shorter than 11 residues (including kinks) and strands shorter than 5 residues were removed. Including this post-processing step resulted in an increase in prediction accuracy of ~3% over all residues in the dataset.

Step	Commands	Comments	Input/Output
topology prediction	039_topology_from_averaged_ANN_files.pl	create and benchmark topology prediction topology is predicted using the 7*0.111 baseline correction for TMstrand and TMhelix prediction	input files are train_10k_combi32/outputs*_avgANNs.txt and train_10k_combi32//*testModel*_unbal_nonrand_nogaps.2.dat output files are topology/topology_testModel_ds??.dat which are analyzed using 027_... with results in topology/topology_accuracies_ds?.dat
topology prediction from jufo9D files +FOR SERVER+		039_topology_from_jufo_output.pl is for server and calculates topology from jufo9D output	

Compare eukaryotic and prokaryotic prediction accuracies

One reviewer raised the question how BCL::Jufo9D would perform on eukaryotic and prokaryotic proteins. The 5 datasets (dataset?.ls) were split into eukaryotic/prokaryotic proteins and the predictions were analyzed.

Step	Commands	Comments	Output
get species from NCBI		split datasets in eukaryotic and prokaryotic use NCBI to get a species list containing taxonomy ID and class (bacterial, eukaryotic, ...)	species-list_ord
split datasets into pro/eukaryote	042_identify_prokaryote-eukaryote_for_datasets.pl	use datasets to create descriptors?_test_eukaryote_unbal_nonrand_nogaps.dat and descriptors1_test_eukaryote_unbal_nonrand_nogaps.bin as above	dataset1_eukary
analyze accuracies	027_analyze_overprediction_9D_all-in-line.pl	test accuracies	outputs are: train_10k_comb and train_10k_comb

Compute kink prediction accuracies for BCL::Jufo9D and other methods

We defined a kink as one or two coil residues in TM helices longer than 11 residues. We considered the kink as accurately identified if it was predicted within five residues in either direction (N- or C-terminal of the actual kink).

Step	Commands	Comments
get real and jufo-predicted kinks	043_kinks.pl	get real kinks and BCL::Jufo9D predicted ones: uses input files descriptors/descriptors?_unbal_nonrand_nogaps.dat and /home/koehlej/projects/jufo/039_training_9D/train_10k_combi32/outputs?_avgANNs.txt
get kinks from PsiPred	044_concatenate_PsiPred_for_kinks.pl 045_compute_kinks_for_PsiPred.pl	get PsiPred predicted kinks: concatenate all Psipred files into one that can be directly compared since residues numbering is identical; use 044... use 045... to compute kinks for PsiPred
get kinks from Octopus	046_concatenate_PsiPred_for_kinks.pl 047_compute_kinks_for_Octopus.pl	get Octopus predicted kinks: concatenate all Octopus files into one that can be directly compared since residues numbering is identical; use 046... use 047... to compute kinks for Octopus:

get kinks from TMkink	058_TMkink_run_from_fasta.pl 058_TMkink_run_from_fasta_part2.pl 059_reformat_TMkink.pl 060_compute_kinks_for_TMkink.pl	get predicted kinks from TMkink (Bowies method: http://tmkinkpredictor.mbi.ucla.edu/) use kinks/TMkink_chains.ls to run 058.. which runs 057_TMkink_submit.pl • this submits the fasta files to the TMkink server • you get emails that contain the codes where the result pages can be found use TMkink-method/TMkink_result_pages.txt and 058... to download files into TMkink-method/ these files are reformatted using 059... and analyzed using 060... with the use of get_kink_locations_proteins.pl and kinks/TMkink_locations1.txt results are compiled into results_kinks.txt