Computer-Aided Molecular Binding Affinity Prediction

Liu Zhanhao

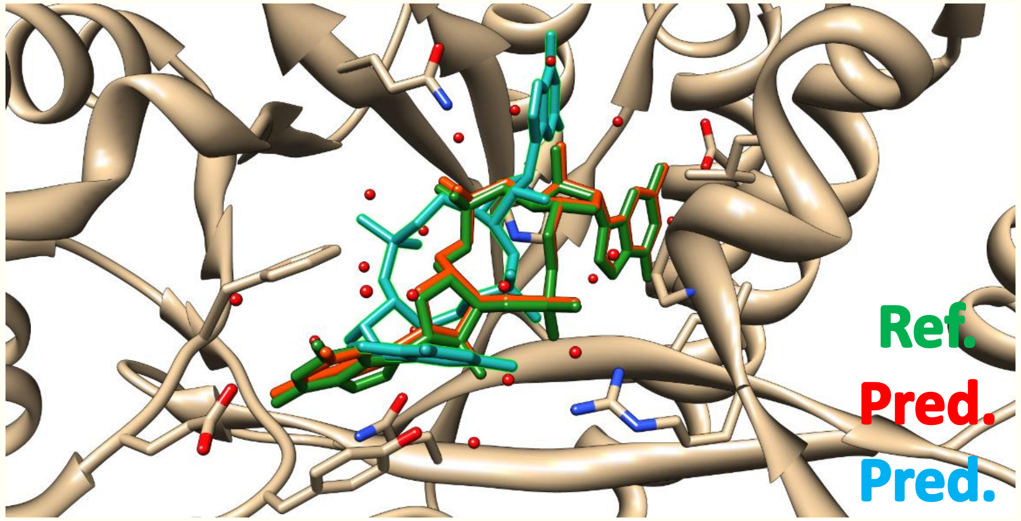
Department of Computer Science & Engineering
The Chinese University of Hong Kong
https://github.com/jasperlau95
Email: jasperlau95@gmail.com

Introduction

Molecular Docking

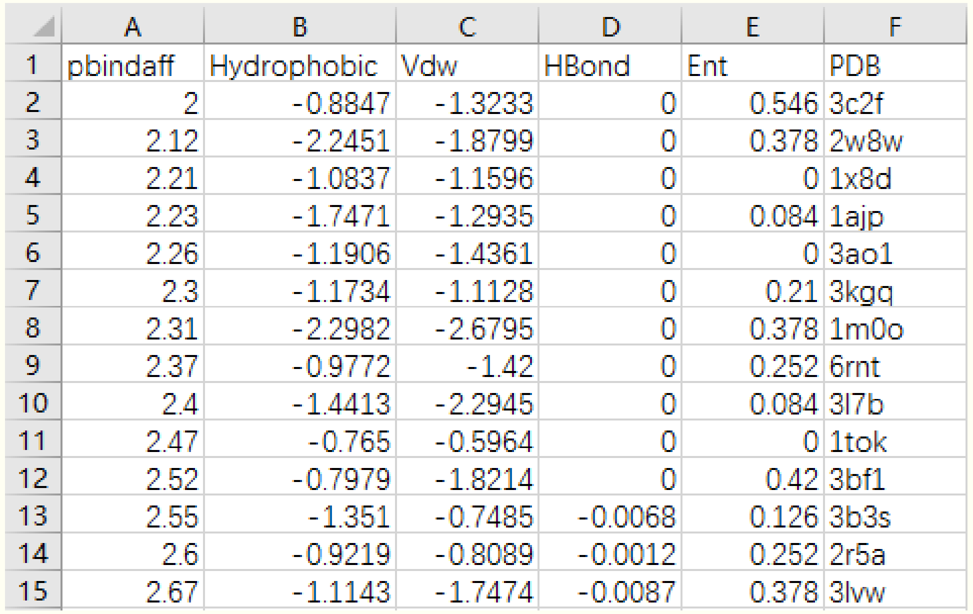
A computational technique that aims to predict whether and how a particular small molecule will stably bind to a target protein.

It is an important component of many drug discovery projects when the structure of the protein is available.

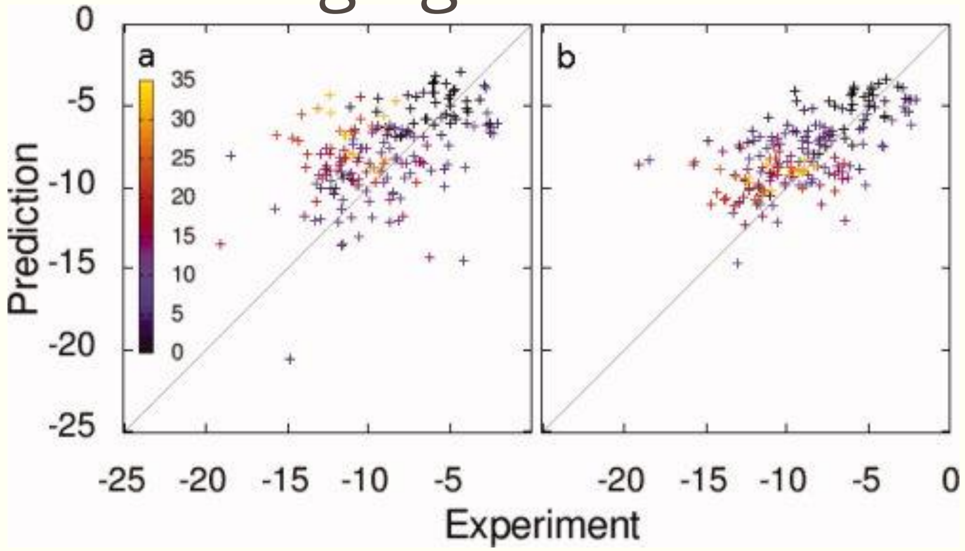


Molecular Docking

Accurately predicting the binding affinities of large sets of diverse protein-ligand complexes is an extremely challenging task.



	A	B	C	D	E	F
1	pbindaff	Hydrophobic	Vdw	HBond	Ent	PDB
2	2	-0.8847	-1.3233	0	0.546	3c2f
3	2.12	-2.2451	-1.8799	0	0.378	2w8w
4	2.21	-1.0837	-1.1596	0	0	1x8d
5	2.23	-1.7471	-1.2935	0	0.084	1ajp
6	2.26	-1.1906	-1.4361	0	0	3ao1
7	2.3	-1.1734	-1.1128	0	0.21	3kgq
8	2.31	-2.2982	-2.6795	0	0.378	1m0o
9	2.37	-0.9772	-1.42	0	0.252	6rnt
10	2.4	-1.4413	-2.2945	0	0.084	3l7b
11	2.47	-0.765	-0.5964	0	0	1tok
12	2.52	-0.7979	-1.8214	0	0.42	3bf1
13	2.55	-1.351	-0.7485	-0.0068	0.126	3b3s
14	2.6	-0.9219	-0.8089	-0.0012	0.252	2r5a
15	2.67	-1.1143	-1.7474	-0.0087	0.378	3lvw



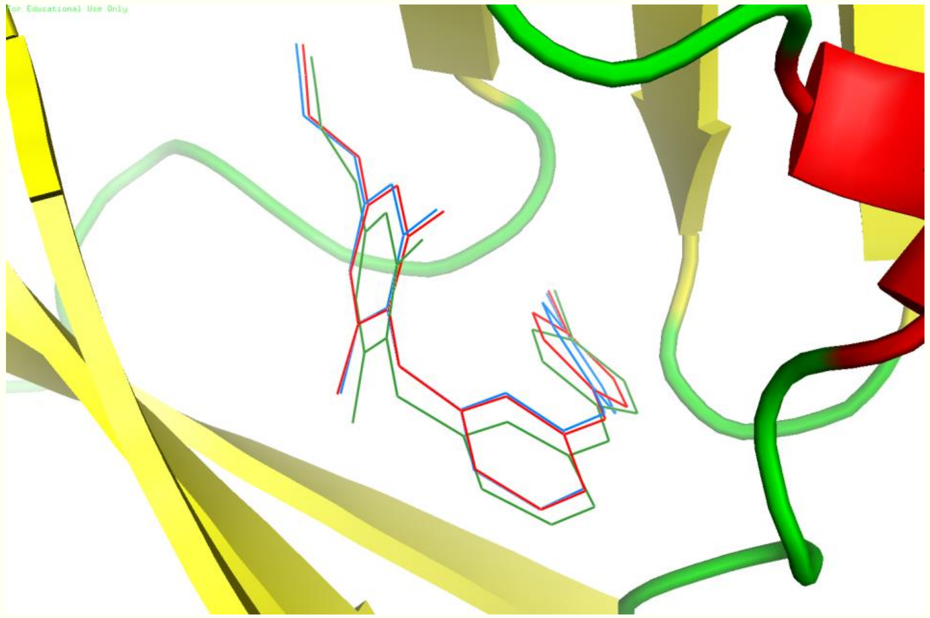
Large error in scoring
SD = 2.75 kcal/mol
K_d = 104x

Oleg Trott and Arthur J. Olson. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010.

Motivation

Acceptable success rate in pose prediction

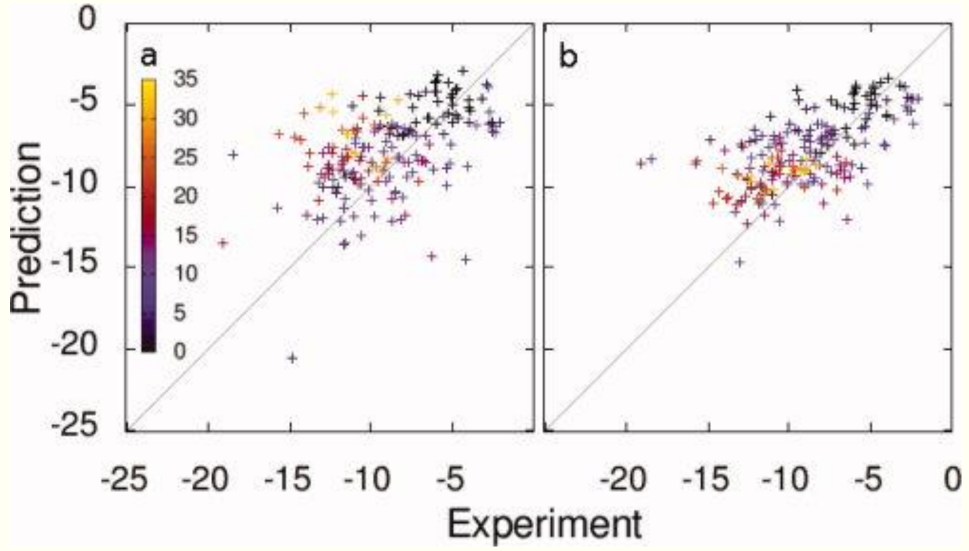
65%



Large error in scoring

SD = 2.75 kcal/mol

K_d = 104x

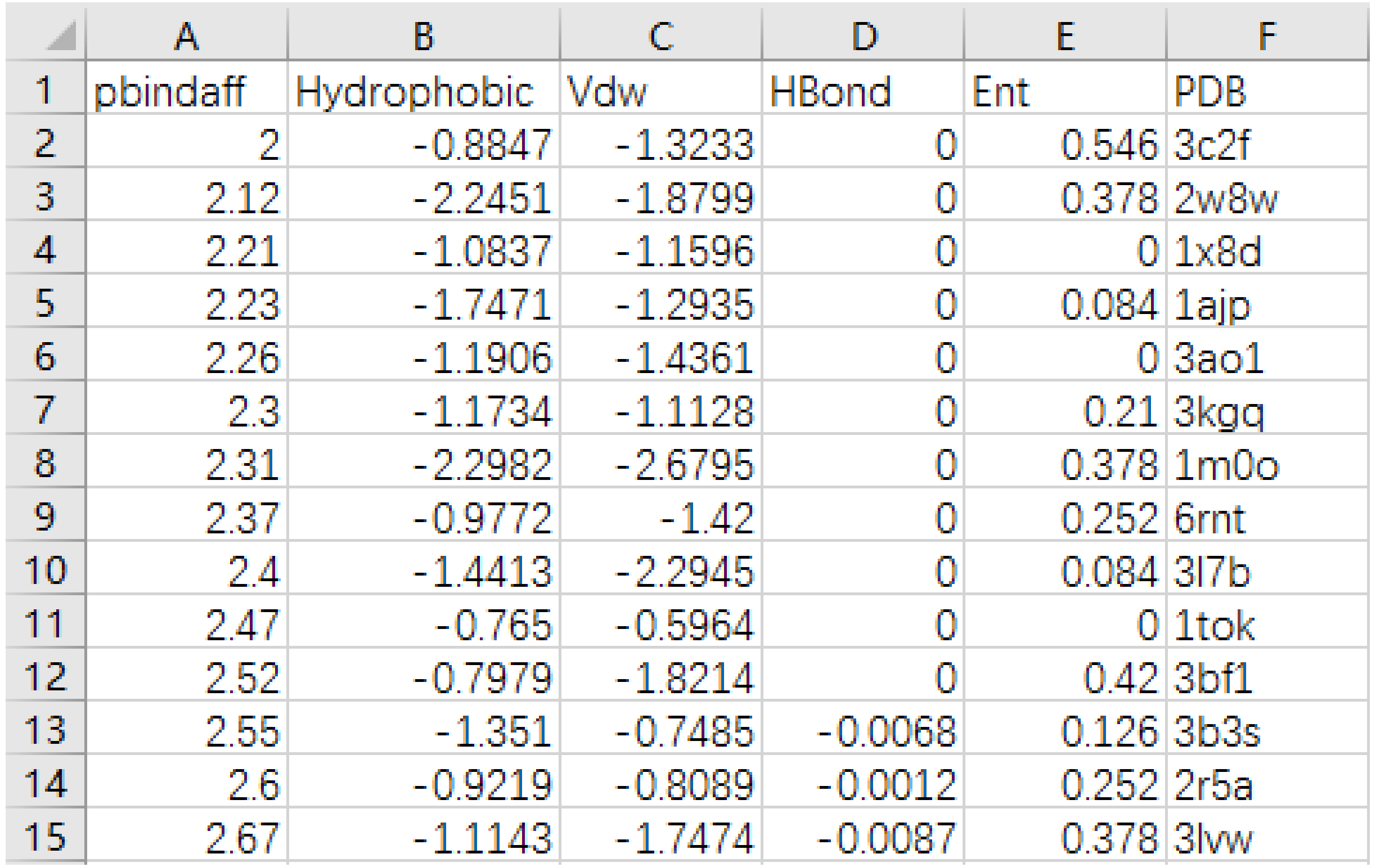


Oleg Trott and Arthur J. Olson. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010.

Scoring

What we have

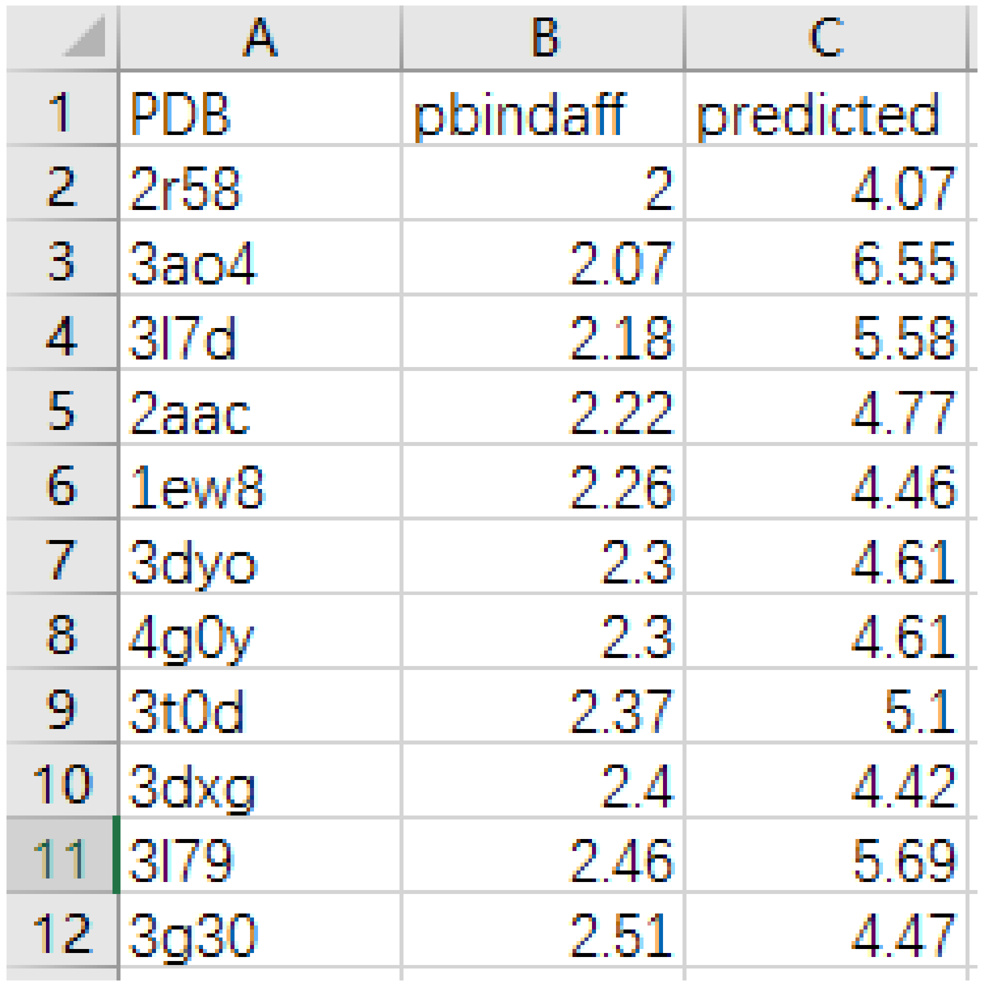
Features of the proteins



	A	B	C	D	E	F
1	pbindaff	Hydrophobic	Vdw	HBond	Ent	PDB
2	2	-0.8847	-1.3233	0	0.546	3c2f
3	2.12	-2.2451	-1.8799	0	0.378	2w8w
4	2.21	-1.0837	-1.1596	0	0	1x8d
5	2.23	-1.7471	-1.2935	0	0.084	1ajp
6	2.26	-1.1906	-1.4361	0	0	3ao1
7	2.3	-1.1734	-1.1128	0	0.21	3kgq
8	2.31	-2.2982	-2.6795	0	0.378	1m0o
9	2.37	-0.9772	-1.42	0	0.252	6rnt
10	2.4	-1.4413	-2.2945	0	0.084	3l7b
11	2.47	-0.765	-0.5964	0	0	1tok
12	2.52	-0.7979	-1.8214	0	0.42	3bf1
13	2.55	-1.351	-0.7485	-0.0068	0.126	3b3s
14	2.6	-0.9219	-0.8089	-0.0012	0.252	2r5a
15	2.67	-1.1143	-1.7474	-0.0087	0.378	3lvw

What we want

Predicted value of the binding affinity



	A	B	C
1	PDB	pbindaff	predicted
2	2r58	2	4.07
3	3ao4	2.07	6.55
4	3l7d	2.18	5.58
5	2aac	2.22	4.77
6	1ew8	2.26	4.46
7	3dyo	2.3	4.61
8	4g0y	2.3	4.61
9	3t0d	2.37	5.1
10	3dxg	2.4	4.42
11	3l79	2.46	5.69
12	3g30	2.51	4.47

Classical Scoring Functions

- Predetermined theory-inspired functional form
- Cyscore

- **Multiple Linear Regression**
  - $\Delta G_{bind} = kh \bullet \Delta G_{hydrophobic}$
  - $+ kv \bullet \Delta G_{vdw}$
  - $+ kb \bullet \Delta G_{hbond}$
  - $+ ke \bullet \Delta G_{entropy}$
  - $+ C$

Machine-Learning Scoring Functions

- **No modelling assumptions**
- **Implicitly capture the binding effects**

- **Random Forest**

  RF-Score, CScore, B2Bscore, SFCscore^{RF}

- **Super Vector Regression**

  SVR-KB, SVR-EP, SVR-Score, ID-Score , MD-SVR

- **Neural Networks**

  RF-Score, CScore, B2Bscore, SFCscoreRF

Random Forest Binding Affinity Prediction

Three models:

- **RF::Cyscore**

  $$\Delta Gbind = RF(\Delta Ghydrophobic, \Delta Gvdw, \Delta Ghbond, \Delta Gentropy)$$

- **RF::CyscoreVina**

  4 Cyscore features + 6 Vina features

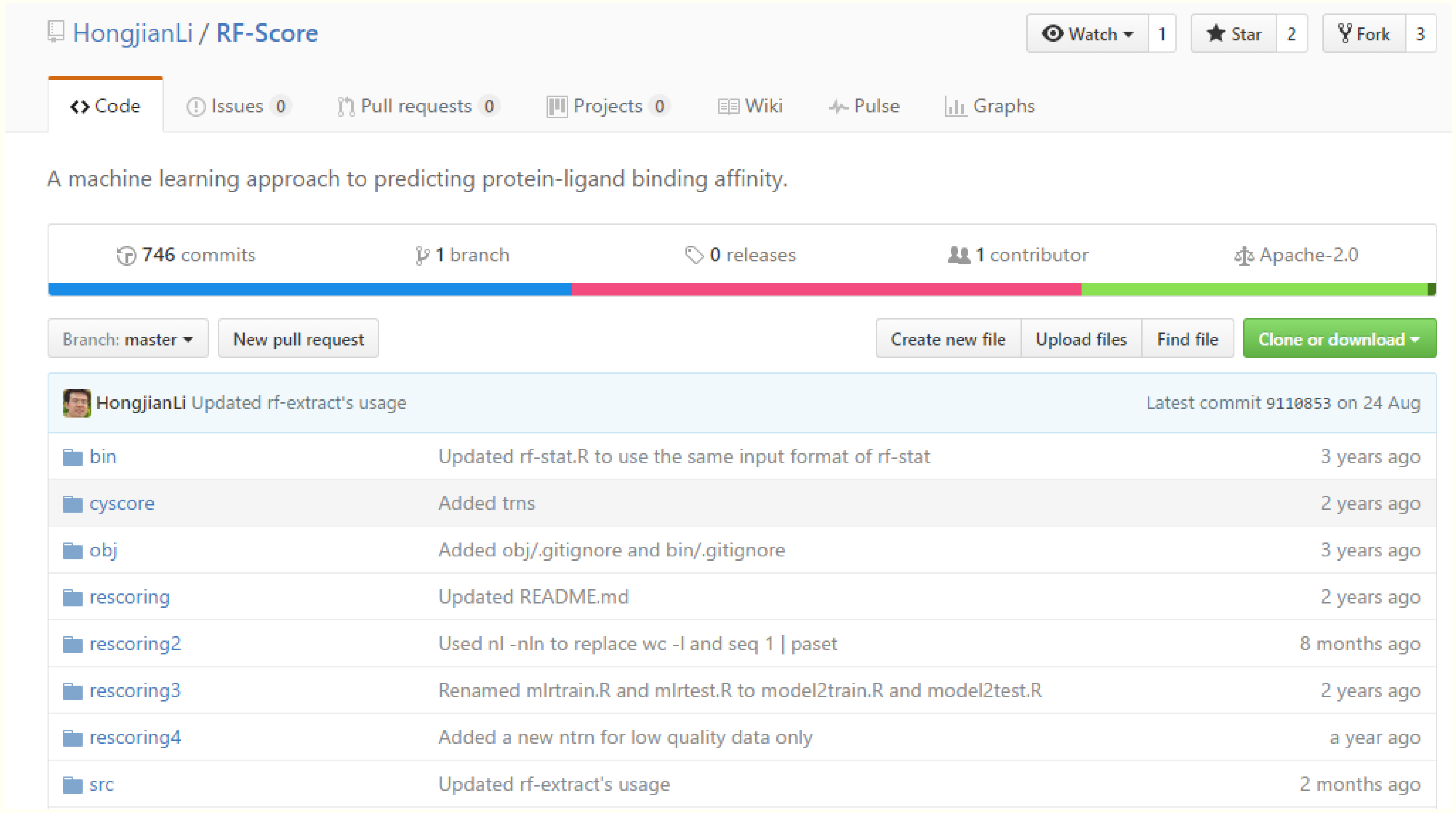
- **RF::CyscoreVinaElem**

  4 Cyscore features + 6 Vina features + 36 RF-Score features

Week 1 Progress

Prepare

Tool:


HongjianLi / RF-Score

Watch 1 | Star 2 | Fork 3

<> Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

A machine learning approach to predicting protein-ligand binding affinity.

746 commits | 1 branch | 0 releases | 1 contributor | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

HongjianLi Updated rf-extract's usage — Latest commit 9110853 on 24 Aug

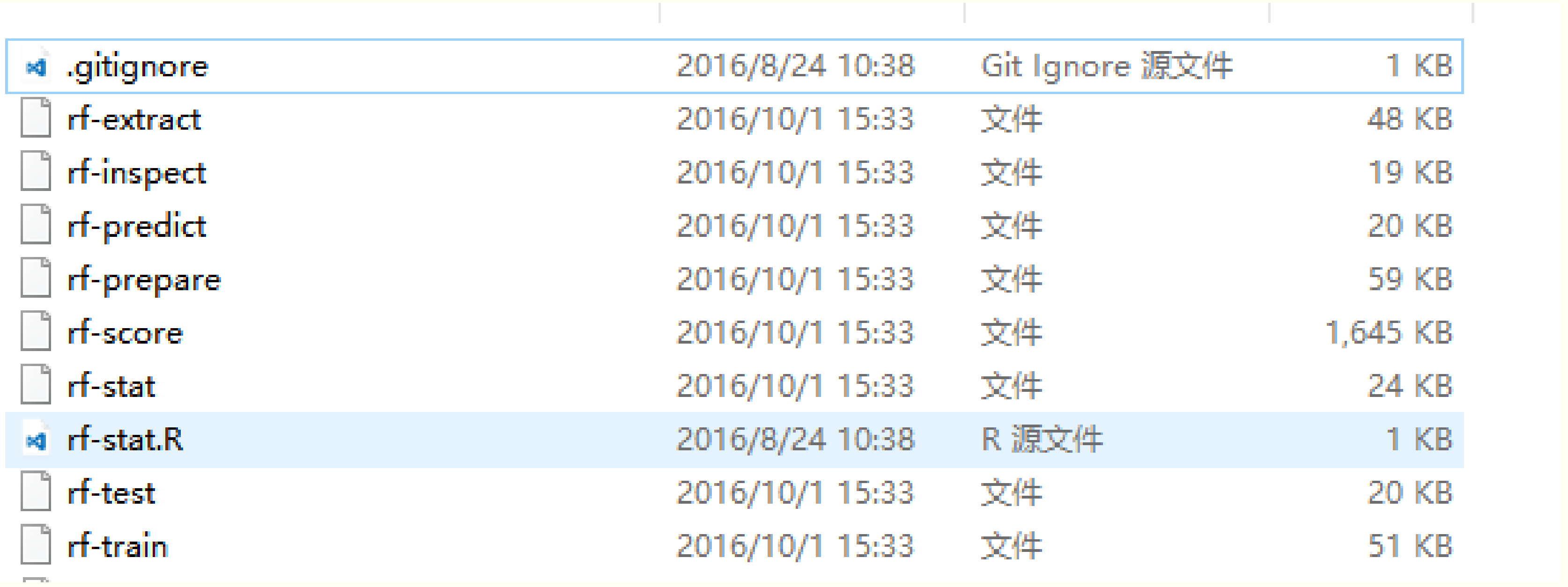
bin	Updated rf-stat.R to use the same input format of rf-stat	3 years ago
cyscore	Added trns	2 years ago
obj	Added obj/.gitignore and bin/.gitignore	3 years ago
rescoring	Updated README.md	2 years ago
rescoring2	Used nl -nln to replace wc -l and seq 1 | paset	8 months ago
rescoring3	Renamed mlrtrain.R and mlrtest.R to model2train.R and model2test.R	2 years ago
rescoring4	Added a new ntrn for low quality data only	a year ago
src	Updated rf-extract's usage	2 months ago

https://github.com/HongjianLi/RF-Score

Prepare

Compilation:

In Ubuntu, after using ‘make’ instruction to compile the RF-Score, we have all these 8 executables.

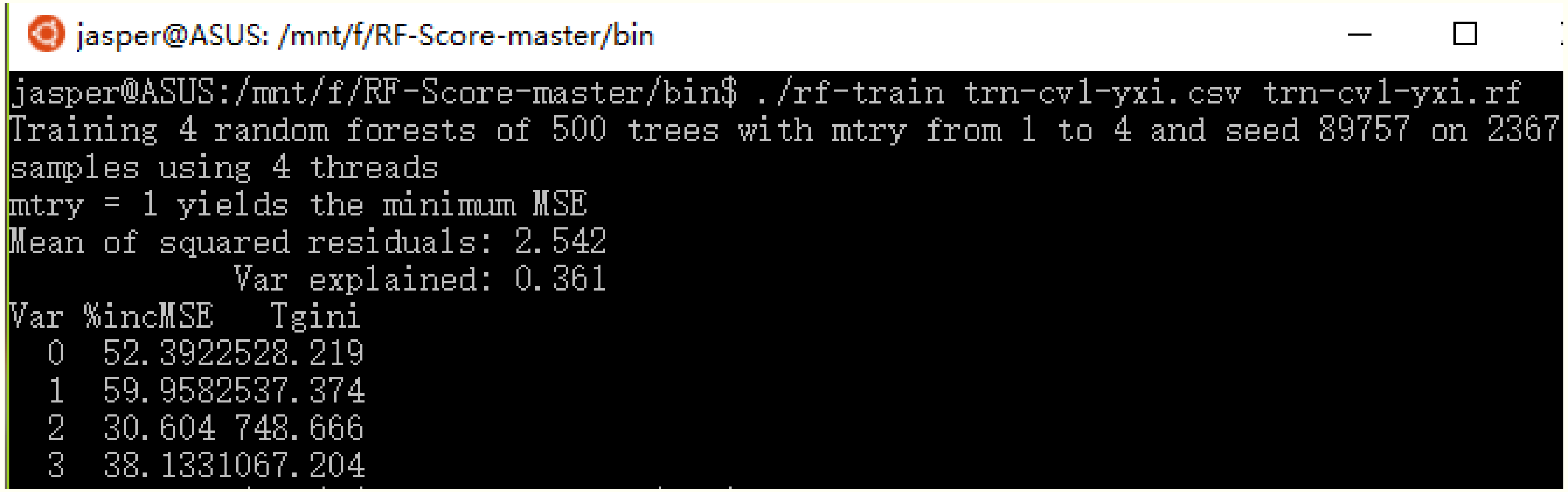


.gitignore	2016/8/24 10:38	Git Ignore 源文件	1 KB
rf-extract	2016/10/1 15:33	文件	48 KB
rf-inspect	2016/10/1 15:33	文件	19 KB
rf-predict	2016/10/1 15:33	文件	20 KB
rf-prepare	2016/10/1 15:33	文件	59 KB
rf-score	2016/10/1 15:33	文件	1,645 KB
rf-stat	2016/10/1 15:33	文件	24 KB
rf-stat.R	2016/8/24 10:38	R 源文件	1 KB
rf-test	2016/10/1 15:33	文件	20 KB
rf-train	2016/10/1 15:33	文件	51 KB

Building a regression Model

Train:

./rf-train trn-cv1-yxi.csv trn-cv1-yxi.rf



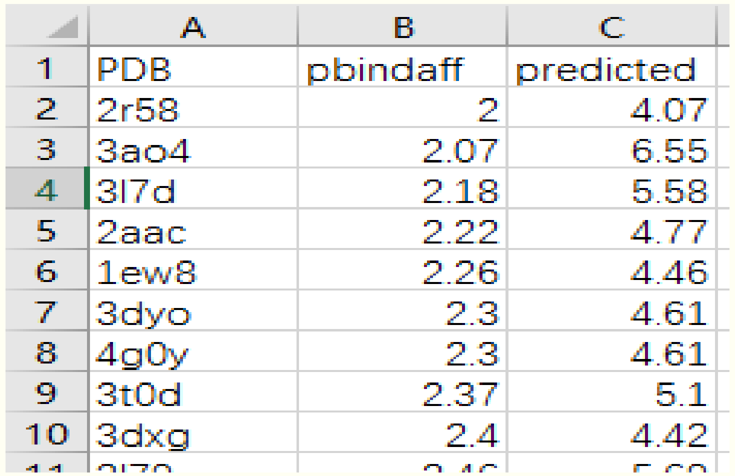
```
jasper@ASUS:/mnt/f/RF-Score-master/bin$ ./rf-train trn-cv1-yxi.csv trn-cv1-yxi.rf
Training 4 random forests of 500 trees with mtry from 1 to 4 and seed 89757 on 2367
samples using 4 threads
mtry = 1 yields the minimum MSE
Mean of squared residuals: 2.542
            Var explained: 0.361
Var %incMSE   Tgini
  0  52.3922528.219
  1  59.9582537.374
  2  30.604 748.666
  3  38.1331067.204
```

Then we can get a regression model “trn-cv1-yxi.rf”

Generate prediction

Predict:

```
./rf-test  trn-cv1-yxi.rf  tst-cv1-yxi.csv  >  tst-cv1-iyp.csv
```



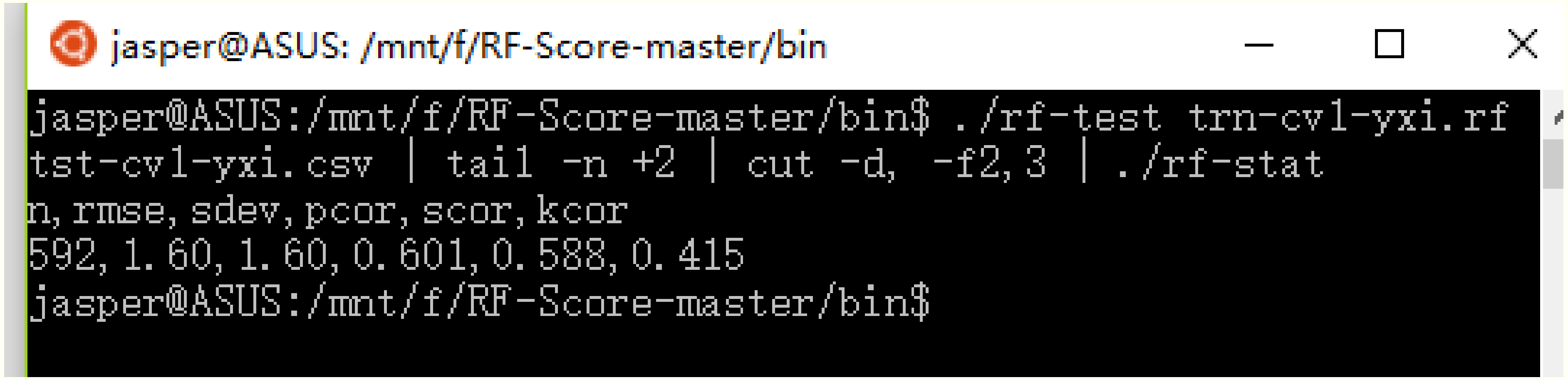
	A	B	C
1	PDB	pbindaff	predicted
2	2r58	2	4.07
3	3ao4	2.07	6.55
4	3l7d	2.18	5.58
5	2aac	2.22	4.77
6	1ew8	2.26	4.46
7	3dyo	2.3	4.61
8	4g0y	2.3	4.61
9	3t0d	2.37	5.1
10	3dxg	2.4	4.42

Then we can find a file tst-cv1-iyp.csv, which is the prediction output for the test sample.

Calculate performance metrics

Evaluate:

./rf-test trn-cv1-yxi.rf tst-cv1-yxi.csv | tail –n +2 | cut –d, -f2,3 | ./rf-stat



```
jasper@ASUS:/mnt/f/RF-Score-master/bin$ ./rf-test trn-cv1-yxi.rf
tst-cv1-yxi.csv | tail -n +2 | cut -d, -f2,3 | ./rf-stat
n,rmse,sdev,pcor,scor,kcor
592,1.60,1.60,0.601,0.588,0.415
jasper@ASUS:/mnt/f/RF-Score-master/bin$
```

Performance Metrics

- Root Mean Square Error RMSE (the smaller, the better)
- Standard Deviation in linear correlation SD (the smaller, the better)
- Pearson correlation coefficient Rp (the bigger, the better)
- Spearman correlation coefficient Rs (the biggerer, the better)

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y^{(n)} - p^{(n)})^2} \qquad SD = \sqrt{\frac{1}{N-2}\sum_{n=1}^{N}(y^{(n)} - \overline{p}^{(n)})^2}$$

$$R_p = \frac{N\sum_{n=1}^{N} p^{(n)} y^{(n)} - \sum_{n=1}^{N} p^{(n)} \sum_{n=1}^{N} y^{(n)}}{\sqrt{(N\sum_{n=1}^{N}(p^{(n)})^2 - (\sum_{n=1}^{N} p^{(n)})^2)(N\sum_{n=1}^{N}(y^{(n)})^2 - (\sum_{n=1}^{N} y^{(n)})^2)}}$$

$$R_s = \frac{N\sum_{n=1}^{N} p_r{}^{(n)} y_r{}^{(n)} - \sum_{n=1}^{N} p_r{}^{(n)} \sum_{n=1}^{N} y_r{}^{(n)}}{\sqrt{(N\sum_{n=1}^{N}(p_r{}^{(n)})^2 - (\sum_{n=1}^{N} p_r{}^{(n)})^2)(N\sum_{n=1}^{N}(y_r{}^{(n)})^2 - (\sum_{n=1}^{N} y_r{}^{(n)})^2)}}$$

Calculate performance metrics

Compare:

My performance metrics:

Metrics	Value
RMSE	1.60
SDEV	1.60
PCOR	0.601
SCOR	0.588
KCOR	0.415

Reference Value:

Metrics	Value
RMSE	1.60
SDEV	1.60
PCOR	0.601
SCOR	0.588
KCOR	0.415

It can be seen that my result is the same as the Reference(by Jacky), I guess the reason is that we both use the same software(RF-Score) and platform(Ubuntu) to compile the program.

Week 2 Progress

Data partition

- PDBbind v2013 Benchmark
- Refined set (N=2959)
- 5-fold cross validation, Round-robin scheduling

Partition	N	Lowest pKd	Highest pKd
1	592	2.00	11.74
2	592	2.00	11.80
3	592	2.00	11.85
4	592	2.00	11.92
5	591	2.05	11.72

- 1 fold for testing and 4 folds for training

5-Fold Cross Validiction (RF::Cyscore)

RF::Cyscore(my result)

#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
avg		1.59	1.59	0.603	0.587

RF::Cyscore(Reference)

#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
avg		1.59	1.59	0.603	0.587

5-Fold Cross Validiction (RF::CyscoreVina)

RF::CyscoreVina(my result)

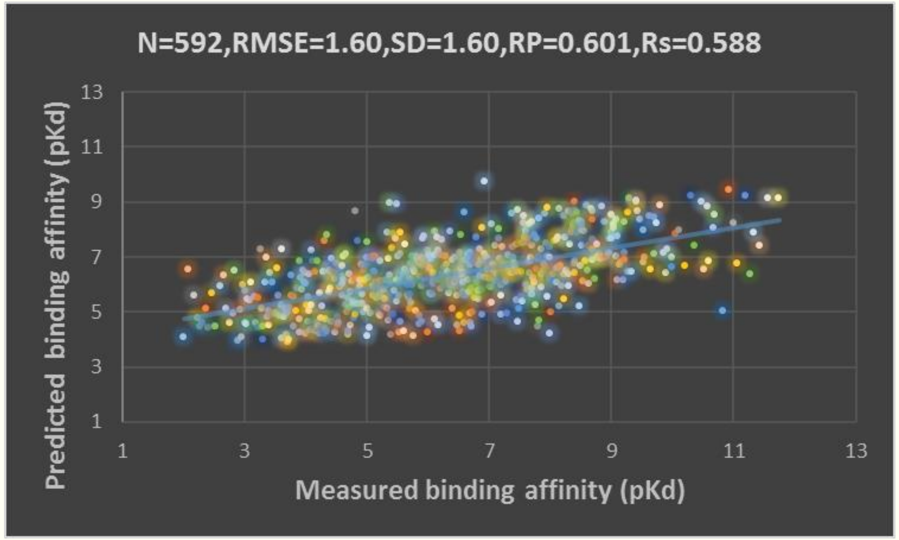
#	N	RMSE	SD	Rp	Rs
1	592	1.41	1.41	0.708	0.709
2	592	1.38	1.37	0.730	0.725
3	592	1.49	1.49	0.668	0.665
4	592	1.51	1.51	0.657	0.661
5	591	1.42	1.42	0.701	0.692
avg		1.44	1.44	0.693	0.690

RF::CyscoreVina(Reference)

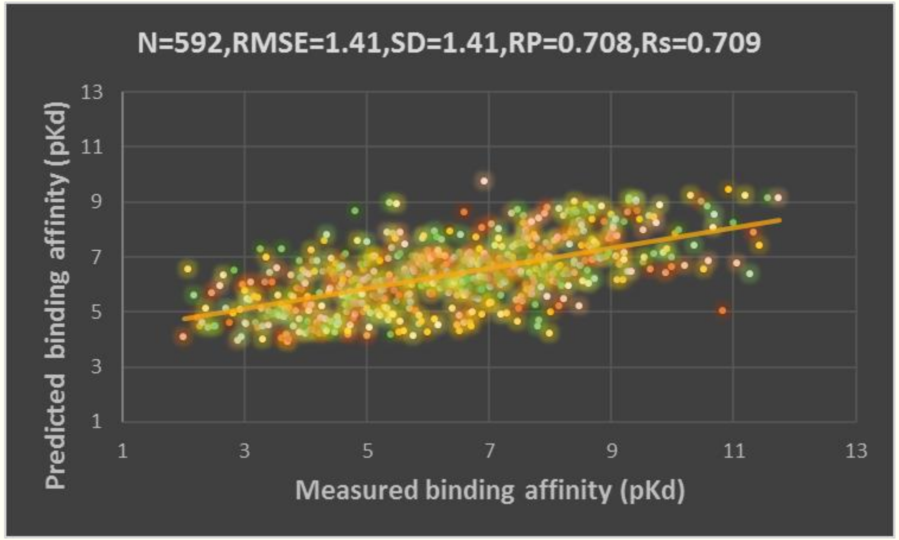
#	N	RMSE	SD	Rp	Rs
1	592	1.41	1.41	0.708	0.709
2	592	1.38	1.37	0.730	0.725
3	592	1.49	1.49	0.668	0.665
4	592	1.51	1.51	0.657	0.661
5	591	1.42	1.42	0.701	0.692
avg		1.44	1.44	0.693	0.690

Comparison

Model	#	RMSE	SD	Rp	Rs
MLR::Cyscore	592	1.66	1.66	0.556	0.559
RF::Cyscore	592	1.59	1.59	0.603	0.587
RF::CyscoreVina	592	1.44	1.44	0.693	0.690

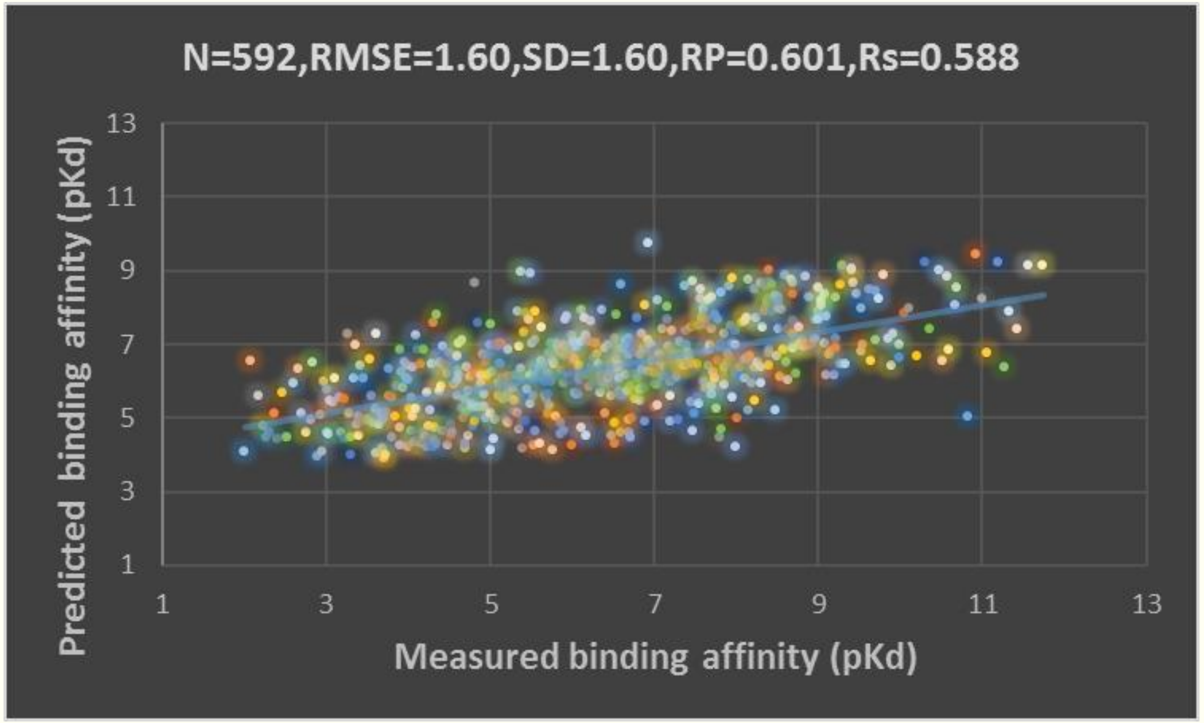


RF::Cyscore(4 features)

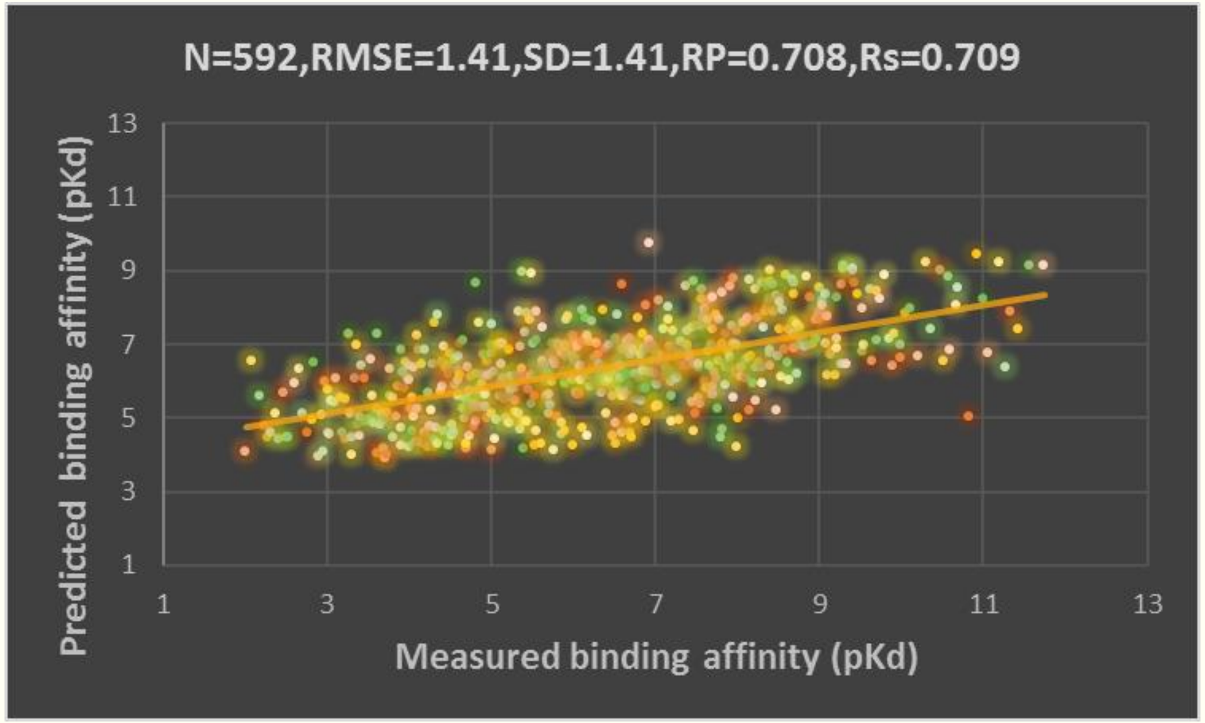


RF::CyscoreVina(10 features)

Comparision



RF::Cyscore(4 features)



RF::CyscoreVina(10 features)

Conclusion

MLR::Cyscore < RF::Cyscore < RF::CyscoreVina

In conclusion, we have shown that changing the regression model from multiple linear regression to random forest improved performance. Adding more features further improved performance.

Explore

Change the data partition method:

Sort the full set(N=2959) by pbindaff, use the 592 samples with the lowest or highest pbindaff values for testing, and use the remaining 2959-592 samples for training.

Partition	#	Lowest pKd	Highest pKd
Lowest	592	2.00	4.55
Highest	592	8.15	11.92

5-Fold Cross Validiction (RF::Cyscore)

RF::Cyscore(my result)

#	N	RMSE	SD	Rp	Rs
Lowest	592	2.88	0.66	0.185	0.198
Highest	592	2.99	0.88	0.154	0.100

RF::Cyscore(Reference)

#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
	avg	1.59	1.59	0.603	0.587

Comparison

Model	#	RMSE	SD	Rp	Rs
RF::Cyscore (systematic sampling)	592	1.59	1.59	0.603	0.587
RF::Cyscore (Lowest test set)	592	2.88	0.66	0.185	0.198
RF::CyscoreVina (Highest test set)	592	2.99	0.88	0.154	0.100

Conclusion

- **The reason why Round-robin scheduling is better**
  - Random Forest’s prediction result have upper bound and lower bound, which are the maximum value and minimum value of the training set respectively.(add a pic.)、

Conclusion

- **Systematic sampling test and training set has better performance**
  - have smaller RMSE, bigger Rp and Rs

- **SD(Standard Deviation) performance metrics cannot reflect the accuracy of the prediction very well.**

- **The reason why systematic sampling is better**
  - Random Forest's prediction result have upper bound and lower bound, which are the maximum value and minimum value of the training set respectively.

Week 3 Progress

Another Tool

- **R:**
- R is a free software environment for statistical computing and graphics.
- We want to use the packages(such as RandomForest, Deep Learning Neural Network…) in R to generate the regression model and repeat the experiments.

5-Fold Cross Validiction (MLR:Cyscore)

MLR::Cyscore(Using R)

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.62	1.62	0.589	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.542	0.557
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.556	0.559

MLR::Cyscore(Reference)

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.62	1.62	0.589	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.542	0.557
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.556	0.559

Fixed equation, same training set and test set $\Rightarrow$ same result

5-Fold Cross Validiction (RF::Cyscore)

RF::Cyscore(Using R)

#	N	RMSE	SD	Rp	Rs
1	592	1.59	1.59	0.604	0.592
2	592	1.52	1.51	0.655	0.638
3	592	1.66	1.66	0.559	0.542
4	592	1.62	1.63	0.583	0.581
5	591	1.58	1.58	0.609	0.579
avg		1.59	1.59	0.602	0.586

RF::Cyscore(Reference)

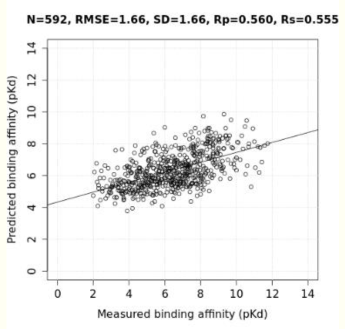
#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
avg		1.59	1.59	0.603	0.587

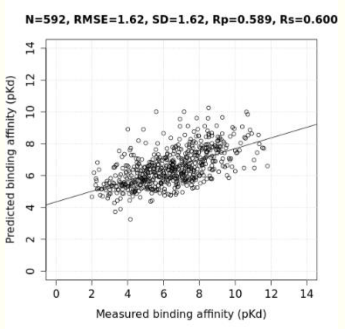
Same method, different tools → close result

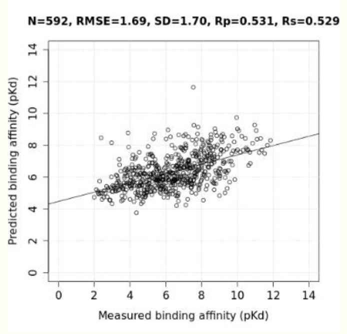
Comparision

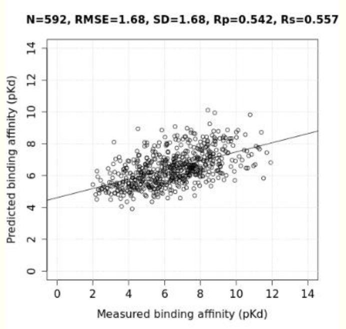
#	N	RMSE	SD	Rp	Rs
MLR::Cyscore	592	1.66	1.66	0.556	0.559
RF::Cyscore	592	1.59	1.59	0.602	0.586

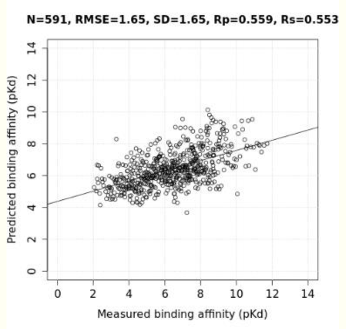
MLR::Cyscore



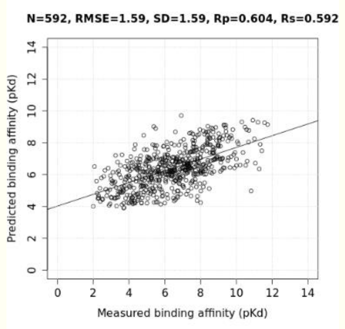


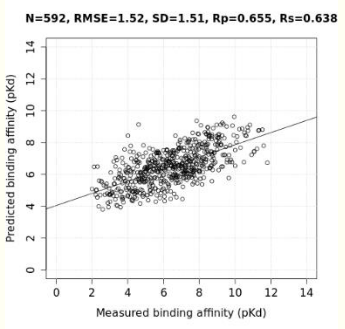


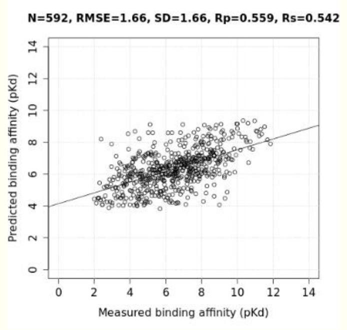


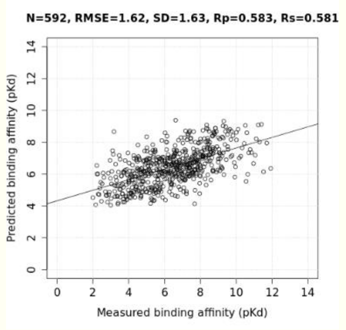


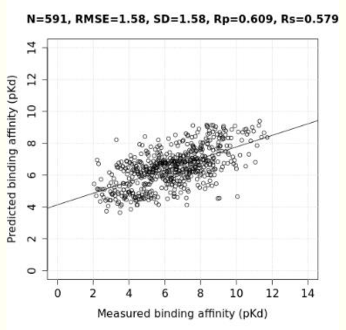
RF::Cyscore







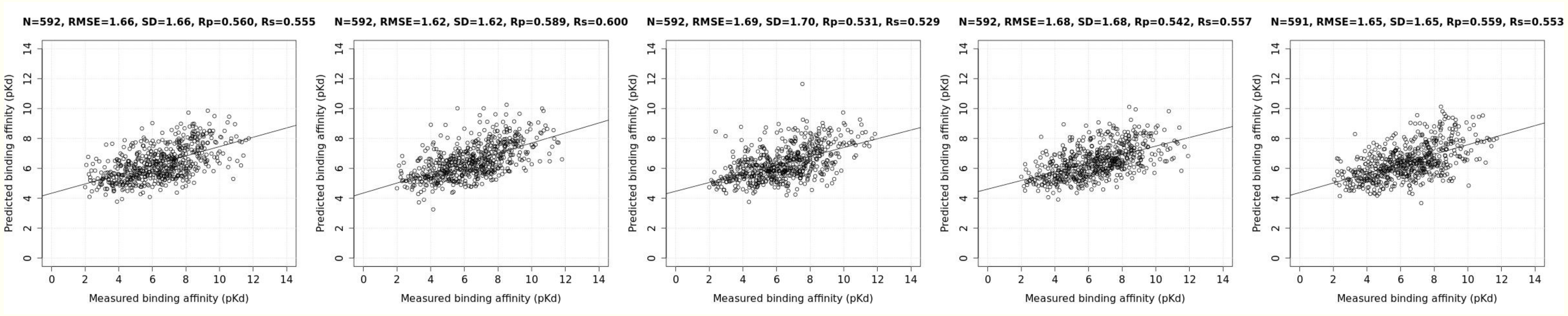




Correlation Plots

set 1 set 2 set 3 set 4 set 5

MLR::Cyscore



N=592, RMSE=1.66, SD=1.66, Rp=0.560, Rs=0.555

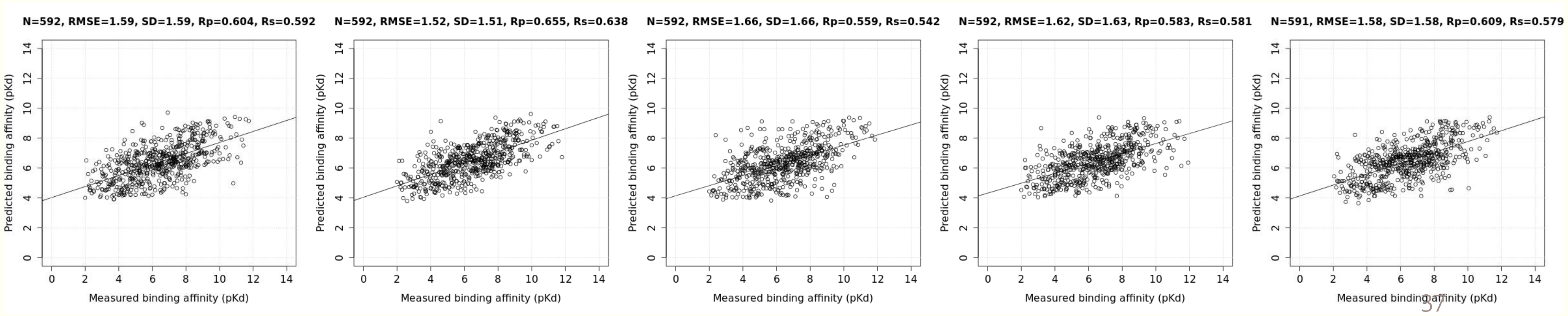
N=592, RMSE=1.62, SD=1.62, Rp=0.589, Rs=0.600

N=592, RMSE=1.69, SD=1.70, Rp=0.531, Rs=0.529

N=592, RMSE=1.68, SD=1.68, Rp=0.542, Rs=0.557

N=591, RMSE=1.65, SD=1.65, Rp=0.559, Rs=0.553

RF::Cyscore



N=592, RMSE=1.59, SD=1.59, Rp=0.604, Rs=0.592

N=592, RMSE=1.52, SD=1.51, Rp=0.655, Rs=0.638

N=592, RMSE=1.66, SD=1.66, Rp=0.559, Rs=0.542

N=592, RMSE=1.62, SD=1.63, Rp=0.583, Rs=0.581

N=591, RMSE=1.58, SD=1.58, Rp=0.609, Rs=0.579

Conclusion

- The predicted results for the same training set and test set are consistent using different tools but same methods.

- RF::Cyscore outperforms MLR::Cyscore

Week 4 Progress

Random (decision) Forest

1995: firstly introduced by Tin Kam Ho
2001: extension, Leo Breiman (What we use)

Random Forest = Bagging + decision trees

Bagging(Bootstrap aggregating)

Implement:

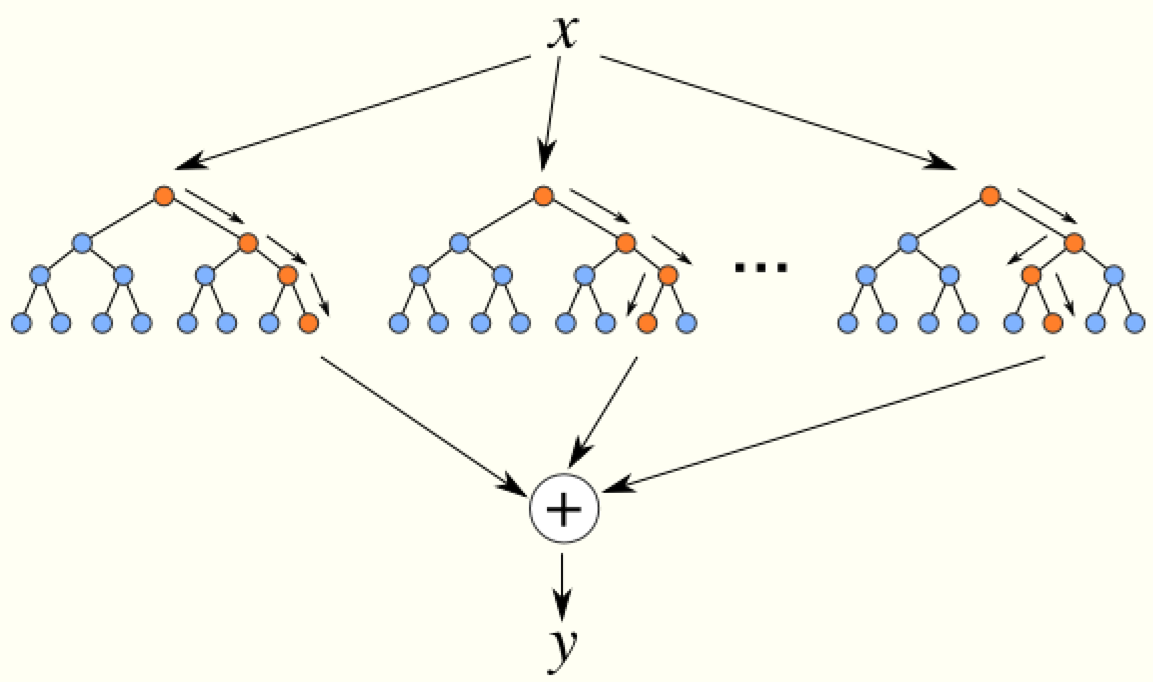
given a standard training set D of size n, bagging generates k new training sets Di, each of size n′, by sampling from D uniformly and with replacement. (put the data back after sampling)

Aim:

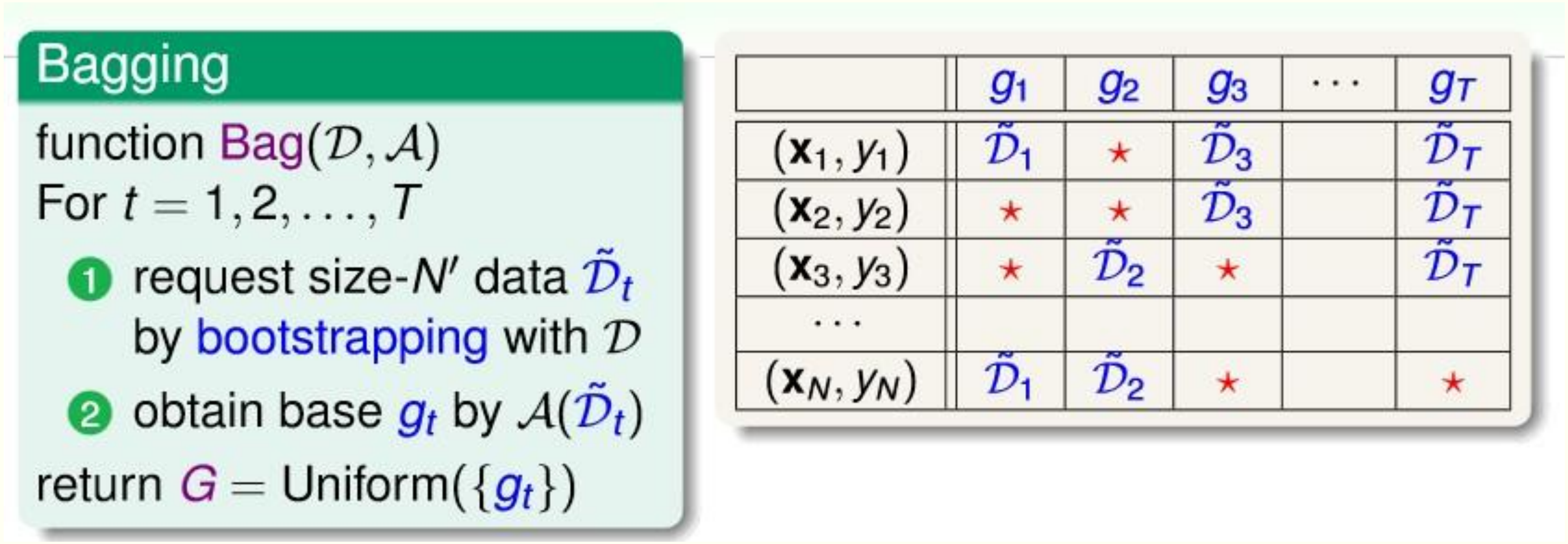
average a given procedure over many samples, to reduce its variance.

Decision tree

Binary decision trees:



Random (decision) Forest



```
Bagging
function Bag($\mathcal{D}, \mathcal{A}$)
For $t = 1, 2, \ldots, T$
    1 request size-$N'$ data $\tilde{\mathcal{D}}_t$
      by bootstrapping with $\mathcal{D}$
    2 obtain base $g_t$ by $\mathcal{A}(\tilde{\mathcal{D}}_t)$
return $G = \text{Uniform}(\{g_t\})$
```

	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	★	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	★	★	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	★	$\tilde{\mathcal{D}}_2$	★		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\tilde{\mathcal{D}}_2$	★		★

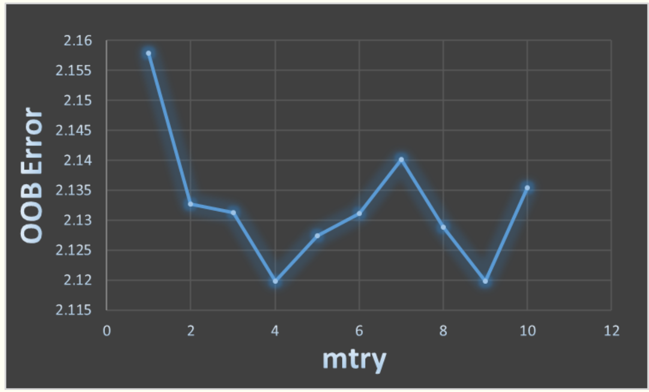
(X[1],Y[1])....(X[n],Y[n]) is the training data, we need to train T(ntree) decision trees g[1]....g[t]...g[T]. Each time we sample N'(mtry) data from D randomly and with replacement to train the tree g[t]. On the right table, the * in each column is the data that is not chosen to train the tree g[t], we call it Out Of Bag(OOB) sample of g[t], and these out of bag samples can be used as test set for each tree.

tuneRF( )

tuneRF( ): tune randomForest for the optimal *mtry* parameter

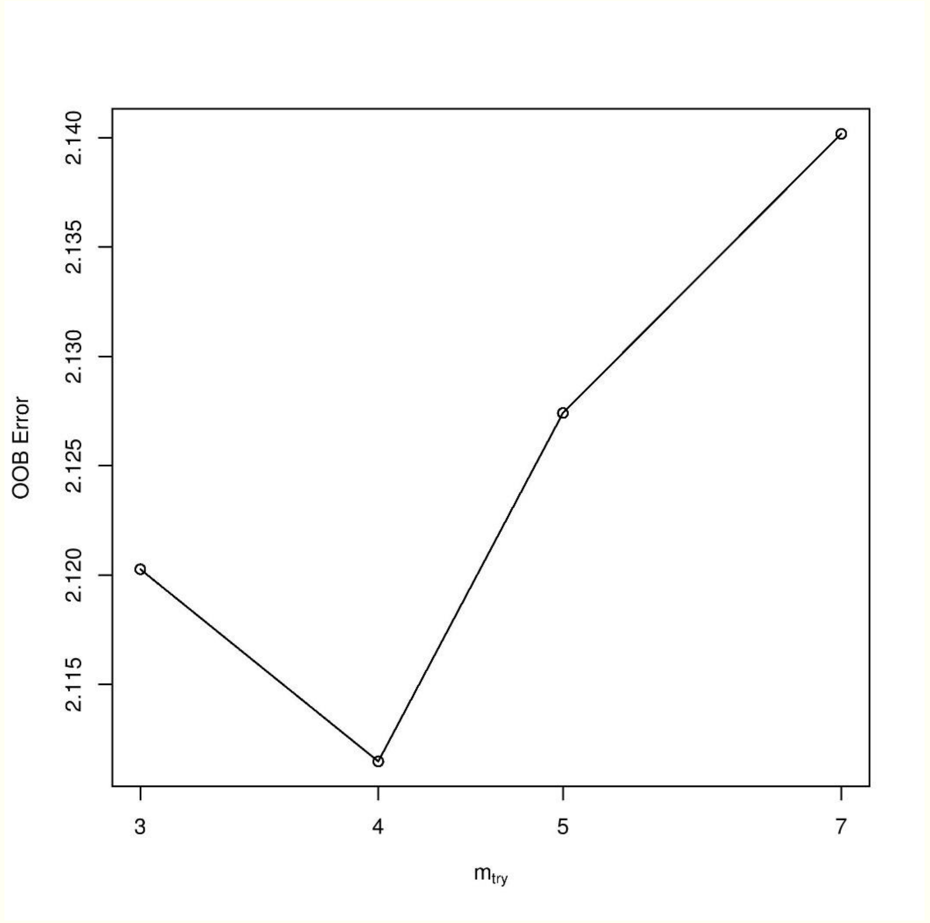
mtry	OOB Error	mtry	OOB Error
1	2.157958	6	2.131094
2	2.132698	7	2.140175
3	2.131236	8	2.128880
4	2.119785	9	2.119842
5	2.127408	10	2.135400

Best → 4

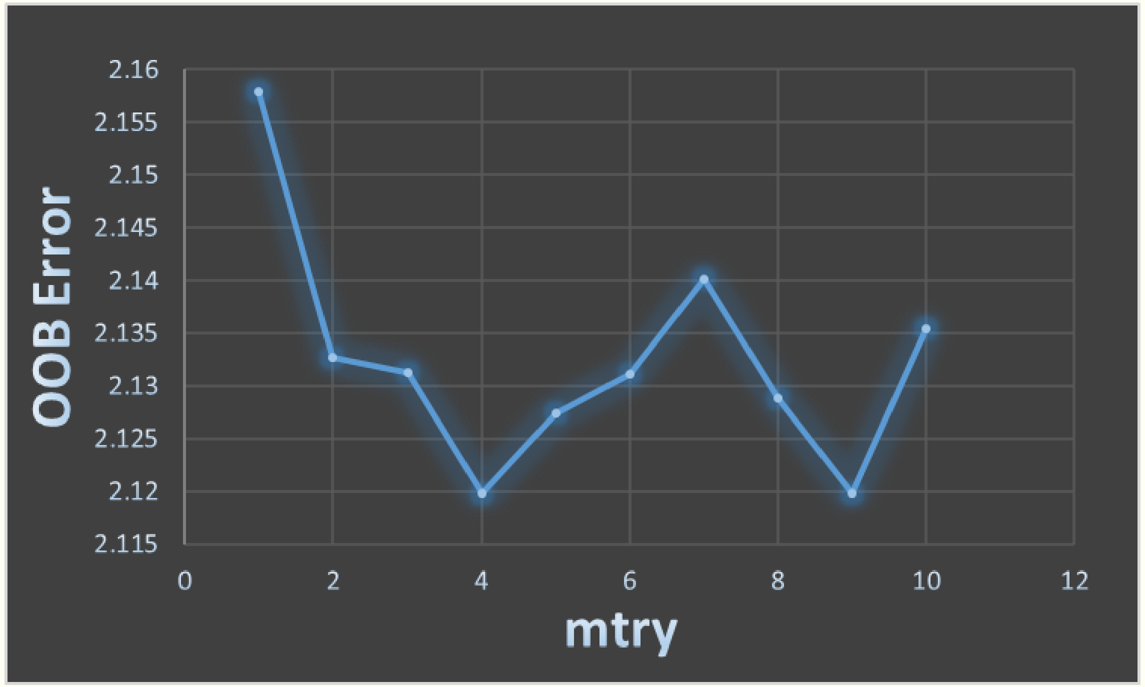


(CyscoreVina, improve=1e-5, ntree=500)

tuneRF( )



OOB Error
2.140
2.135
2.130
2.125
2.120
2.115
3
4
5
7
m_{try}



OOB Error
2.16
2.155
2.15
2.145
2.14
2.135
2.13
2.125
2.12
2.115
0
2
4
6
8
10
12
mtry

Random Forest with different mtry

mtry	#	RMSE	SD	Rp	Rs	OOB Error
2	592	1.41	1.41	0.708	0.707	2.132698
4	592	1.41	1.41	0.708	0.708	2.119785
6	592	1.41	1.41	0.708	0.708	2.131094
8	592	1.41	1.41	0.708	0.706	2.128880
10	592	1.41	1.41	0.707	0.705	2.135400
avg		1.41	1.41	0.708	0.707	2.129571

ntree = 500

Random Forest with different ntree

Ntree	#	RMSE	SD	Rp	Rs	OOB Error
30	592	1.45	1.45	0.690	0.691	2.337980
60	592	1.42	1.42	0.702	0.702	2.227404
125	592	1.42	1.42	0.703	0.703	2.205851
250	592	1.41	1.41	0.708	0.709	2.135712
500	592	1.41	1.41	0.708	0.708	2.123670
Avg		1.42	1.42	0.702	0.703	2.206123

mtry = 4

When ntree is big enough, increasing it did not increase the performance

Support Vector Machine

SVM::Cyscore(my result)

#	N	RMSE	SD	Rp	Rs
1	592	1.65	1.64	0.570	0.559
2	592	1.56	1.56	0.626	0.612
3	592	1.69	1.68	0.544	0.529
4	592	1.64	1.64	0.572	0.567
5	591	1.61	1.61	0.590	0.577
Avg		1.63	1.63	0.580	0.569

RF::Cyscore(Reference)

#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
	avg	1.59	1.59	0.603	0.587

Random Forest outperforms Support Vector Machine in this problem

Week 5 Progress

Another Tool

- **Tensorflow:**

- TensorFlow is an open source software library for machine learning in various kinds of perceptual and language understanding tasks.

MLR

- We want to try to implement MLR first to see if we can achieve a consistent result as in R
- Aim: to find all the coefficients W[i] of the MLR equation:
- p = W[0] * x[0] + W[1] * x[1] + W[2] * x[2] + W[3] * x[3] + b

- Tensorflow results:

Inconsistent

Intercept	Hydrophobic	Vdw	HBond	Ent
3.803937	-0.649290	-0.770088	-8.054253	-1.416916

- R results(Reference):

Intercept	Hydrophobic	Vdw	HBond	Ent
3.800991	-0.650034	-0.769652	-9.111314	-1.462372

MLR

- Tensorflow results:

Inconsistent

Intercept	Hydrophobic	Vdw	HBond	Ent
3.803937	-0.649290	-0.770088	-8.054253	-1.416916

- R results(Reference):

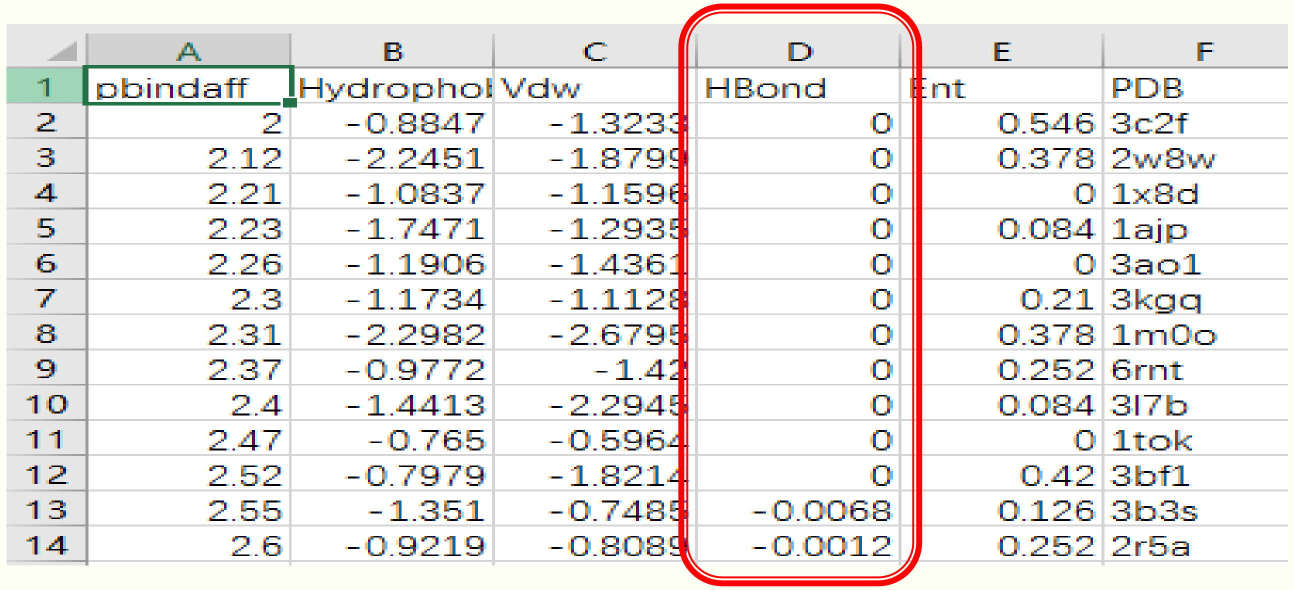
Intercept	Hydrophobic	Vdw	HBond	Ent
3.800991	-0.650034	-0.769652	-9.111314	-1.462372

Try to find out the reason why the coeffient of HBond is wrong:

1) Set HBond as random variable in tensorflow Still Inconsistent
2) Try other optimizer in tensorflow Still Inconsistent
3) Look into the HBond

Analysis

- Look into HBond:



	A	B	C	D	E	F
1	pbindaff	Hydrophol	Vdw	HBond	Ent	PDB
2	2	-0.8847	-1.3233	0	0.546	3c2f
3	2.12	-2.2451	-1.8799	0	0.378	2w8w
4	2.21	-1.0837	-1.1596	0	0	1x8d
5	2.23	-1.7471	-1.2935	0	0.084	1ajp
6	2.26	-1.1906	-1.4361	0	0	3ao1
7	2.3	-1.1734	-1.1128	0	0.21	3kgq
8	2.31	-2.2982	-2.6795	0	0.378	1m0o
9	2.37	-0.9772	-1.42	0	0.252	6rnt
10	2.4	-1.4413	-2.2945	0	0.084	3l7b
11	2.47	-0.765	-0.5964	0	0	1tok
12	2.52	-0.7979	-1.8214	0	0.42	3bf1
13	2.55	-1.351	-0.7485	-0.0068	0.126	3b3s
14	2.6	-0.9219	-0.8089	-0.0012	0.252	2r5a

- all of HBond’s are 0 or nearly 0, this feature is negligible in the MLR model
- So we can just ignore this feature and continue…

Multiple Linear Regressor

MLR::Cyscore(Tensorflow)

train step = 2000

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.61	1.61	0.590	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.543	0.558
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.557	0.559

MLR::Cyscore(R)

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.62	1.62	0.589	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.542	0.557
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.556	0.559

DNN Regressor

DNN::Cyscore

train step = 2000, hidden_units=[10,20,10,20]

#	N	RMSE	SD	Rp	Rs
1	592	1.64	1.64	0.570	0.563
2	592	1.59	1.59	0.608	0.606
3	592	1.68	1.68	0.546	0.544
4	592	1.65	1.65	0.564	0.566
5	591	1.64	1.64	0.569	0.559
avg		1.64	1.64	0.571	0.568

MLR::Cyscore(Reference)

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.62	1.62	0.589	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.542	0.557
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.556	0.559

DNN::Cyscore

train step = 2000,   hidden_units=[10,20,10,20]

#	N	RMSE	SD	Rp	Rs
1	592	1.64	1.64	0.570	0.563
2	592	1.59	1.59	0.608	0.606
3	592	1.68	1.68	0.546	0.544
4	592	1.65	1.65	0.564	0.566
5	591	1.64	1.64	0.569	0.559
avg		1.64	1.64	0.571	0.568

RF::Cyscore

#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
avg		1.59	1.59	0.603	0.587

SVM::Cyscore

#	N	RMSE	SD	Rp	Rs
1	592	1.65	1.64	0.570	0.559
2	592	1.56	1.56	0.626	0.612
3	592	1.69	1.68	0.544	0.529
4	592	1.64	1.64	0.572	0.567
5	591	1.61	1.61	0.590	0.577
avg		1.63	1.63	0.580	0.569

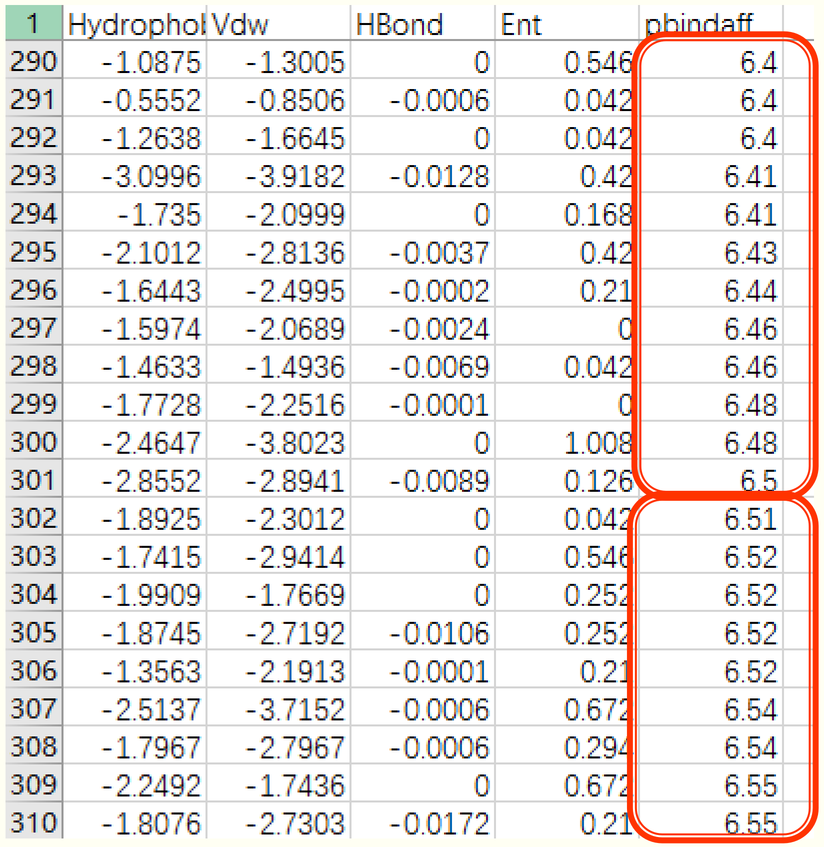
MLR::Cyscore

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.62	1.62	0.589	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.542	0.557
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.556	0.559

#	RMSE	SD	Rp	Rs
MLR::Cyscore	1.66	1.66	0.556	0.559
RF::Cyscore	1.59	1.59	0.602	0.586
SVM::Cyscore	1.63	1.63	0.58	0.569
DNN::Cyscore ([10 20 10])	1.64	1.64	0.57	0.566
DNN::Cyscore ([20 40 20])	1.64	1.64	0.57	0.566
DNN::Cyscore ([10 20 10 20])	1.64	1.64	0.571	0.568
DNN::Cyscore ([10 20 10 20 10])	1.64	1.64	0.571	0.567

DNN Classification()

- **Change the Regression problem into a Classification problem:**

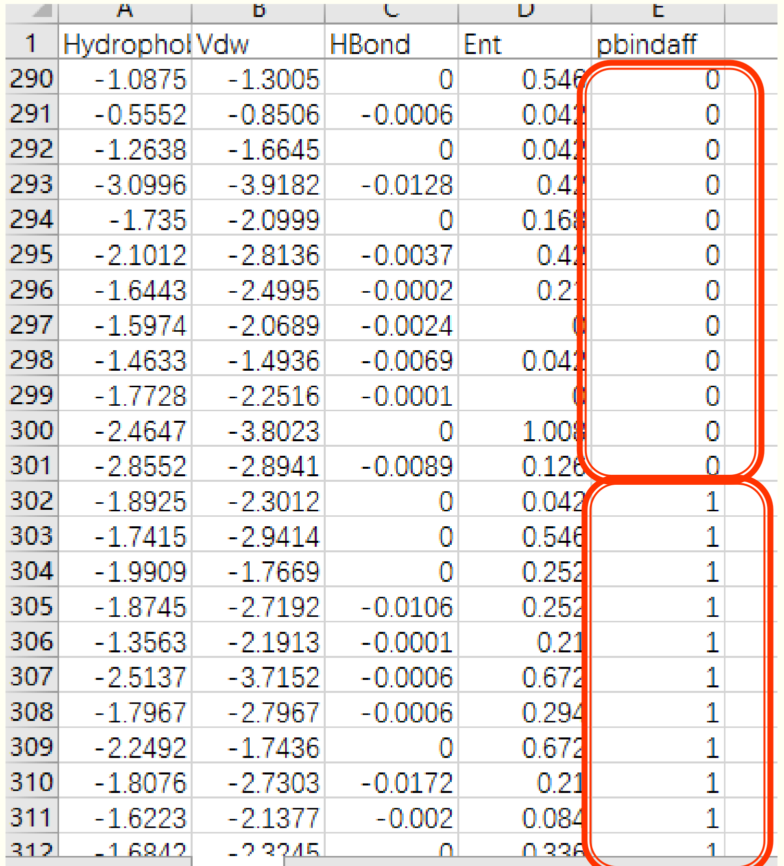


1	Hydrophol	Vdw	HBond	Ent	pbindaff
290	-1.0875	-1.3005	0	0.546	6.4
291	-0.5552	-0.8506	-0.0006	0.042	6.4
292	-1.2638	-1.6645	0	0.042	6.4
293	-3.0996	-3.9182	-0.0128	0.42	6.41
294	-1.735	-2.0999	0	0.168	6.41
295	-2.1012	-2.8136	-0.0037	0.42	6.43
296	-1.6443	-2.4995	-0.0002	0.21	6.44
297	-1.5974	-2.0689	-0.0024	0	6.46
298	-1.4633	-1.4936	-0.0069	0.042	6.46
299	-1.7728	-2.2516	-0.0001	0	6.48
300	-2.4647	-3.8023	0	1.008	6.48
301	-2.8552	-2.8941	-0.0089	0.126	6.5
302	-1.8925	-2.3012	0	0.042	6.51
303	-1.7415	-2.9414	0	0.546	6.52
304	-1.9909	-1.7669	0	0.252	6.52
305	-1.8745	-2.7192	-0.0106	0.252	6.52
306	-1.3563	-2.1913	-0.0001	0.21	6.52
307	-2.5137	-3.7152	-0.0006	0.672	6.54
308	-1.7967	-2.7967	-0.0006	0.294	6.54
309	-2.2492	-1.7436	0	0.672	6.55
310	-1.8076	-2.7303	-0.0172	0.21	6.55

Max pbindaff = 11.72
Min pbindaff = 2
(11.72 + 2) / 2 ≈ 6.5

Pbindaff <=6.5(mean) as
type 0(week binding)

Pbindaff >6.5 as
type 1(strong binding)



	A	B	C	D	E
1	Hydrophol	Vdw	HBond	Ent	pbindaff
290	-1.0875	-1.3005	0	0.546	0
291	-0.5552	-0.8506	-0.0006	0.042	0
292	-1.2638	-1.6645	0	0.042	0
293	-3.0996	-3.9182	-0.0128	0.42	0
294	-1.735	-2.0999	0	0.168	0
295	-2.1012	-2.8136	-0.0037	0.42	0
296	-1.6443	-2.4995	-0.0002	0.21	0
297	-1.5974	-2.0689	-0.0024	0	0
298	-1.4633	-1.4936	-0.0069	0.042	0
299	-1.7728	-2.2516	-0.0001	0	0
300	-2.4647	-3.8023	0	1.008	0
301	-2.8552	-2.8941	-0.0089	0.126	0
302	-1.8925	-2.3012	0	0.042	1
303	-1.7415	-2.9414	0	0.546	1
304	-1.9909	-1.7669	0	0.252	1
305	-1.8745	-2.7192	-0.0106	0.252	1
306	-1.3563	-2.1913	-0.0001	0.21	1
307	-2.5137	-3.7152	-0.0006	0.672	1
308	-1.7967	-2.7967	-0.0006	0.294	1
309	-2.2492	-1.7436	0	0.672	1
310	-1.8076	-2.7303	-0.0172	0.21	1
311	-1.6223	-2.1377	-0.002	0.084	1
312	-1.6842	-2.3245	0	0.336	1

Accuracy:

- 4 layer DNN with 10, 20, 10,20 units respectively, training step=2000
- Classify into 2 types: 0 for pbindaff <=6.5, 1 for pbindaff >6.5

Accuracy = 0.682432

- Classify into 3 types: 0 for pbindaff <=5.2, 1 for pbindaff in (5.2,8.5] , 2 for pbindaff>8.5

Accuracy = 0.616554

- Classify into 4 types: 0 for pbindaff <=4.4, 1 for pbindaff in (4.4,6.9] , 2 for pbindaff in (6.9,9.3], 3 for pbindaff > 9.3

Accuracy = 0.508446

Conclusion

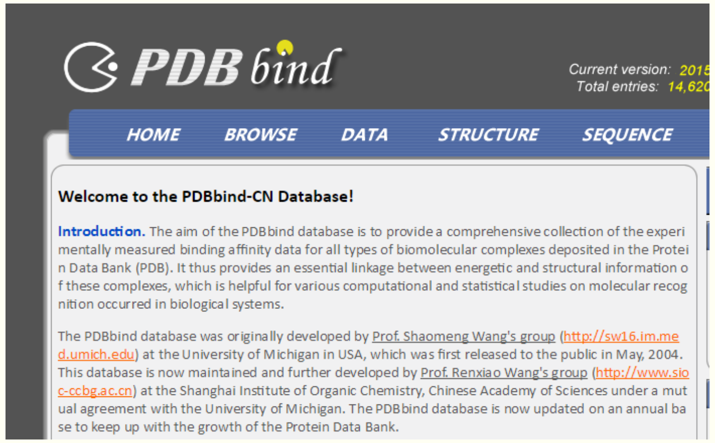
- **MLR::Cyscore < DNN::Cyscore < SVM::Cyscore < RF::Cyscore**
- **Classification model is unfit for this problem**

Extract features

- Raw data for Protein-Ligand Binding affinity

PDBbind database is to provide a comprehensive collection of the experimentally measured binding affinity data for all types of biomolecular complexes deposited in the Protein Data Bank (PDB).

- http://www.pdbbind.org.cn/

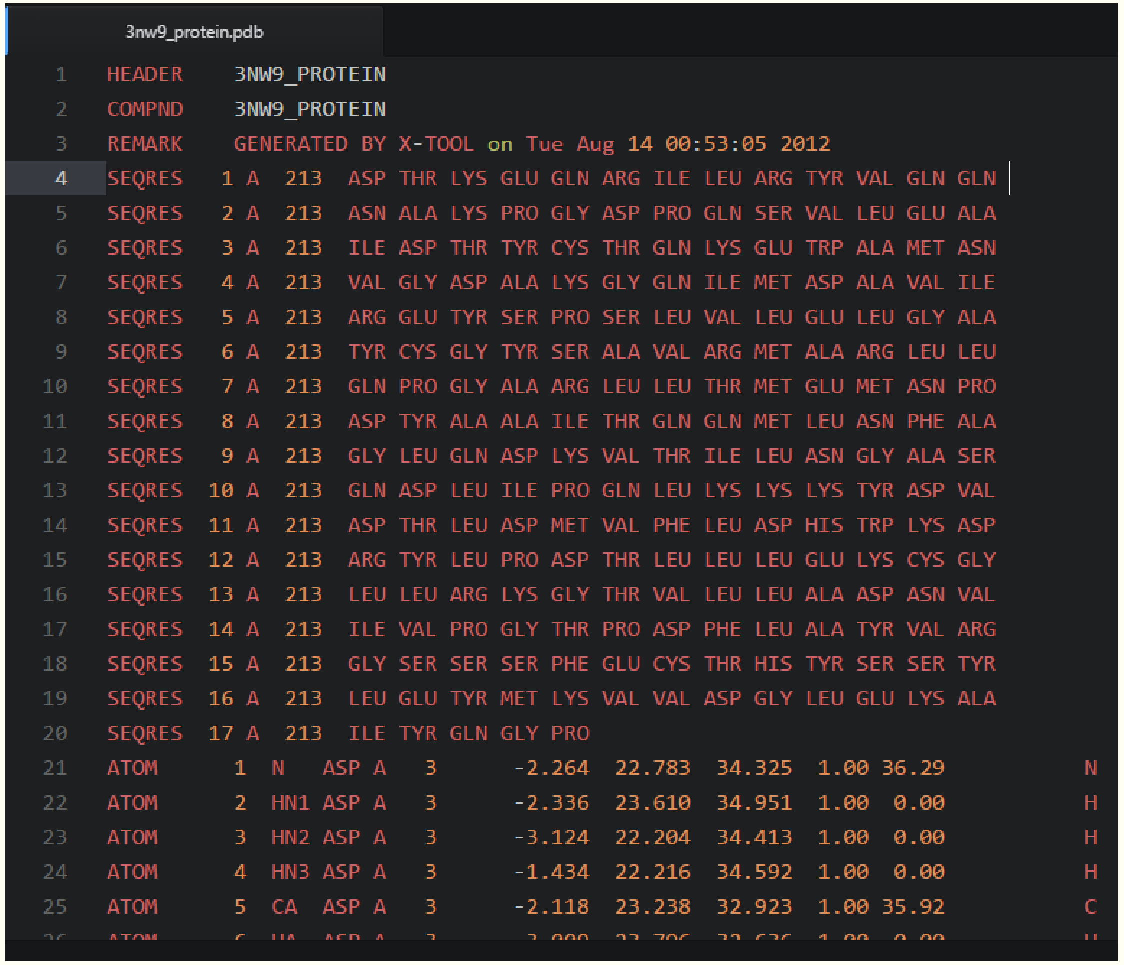


Welcome to the PDBbind-CN Database!

Introduction. The aim of the PDBbind database is to provide a comprehensive collection of the experimentally measured binding affinity data for all types of biomolecular complexes deposited in the Protein Data Bank (PDB). It thus provides an essential linkage between energetic and structural information of these complexes, which is helpful for various computational and statistical studies on molecular recognition occurred in biological systems.

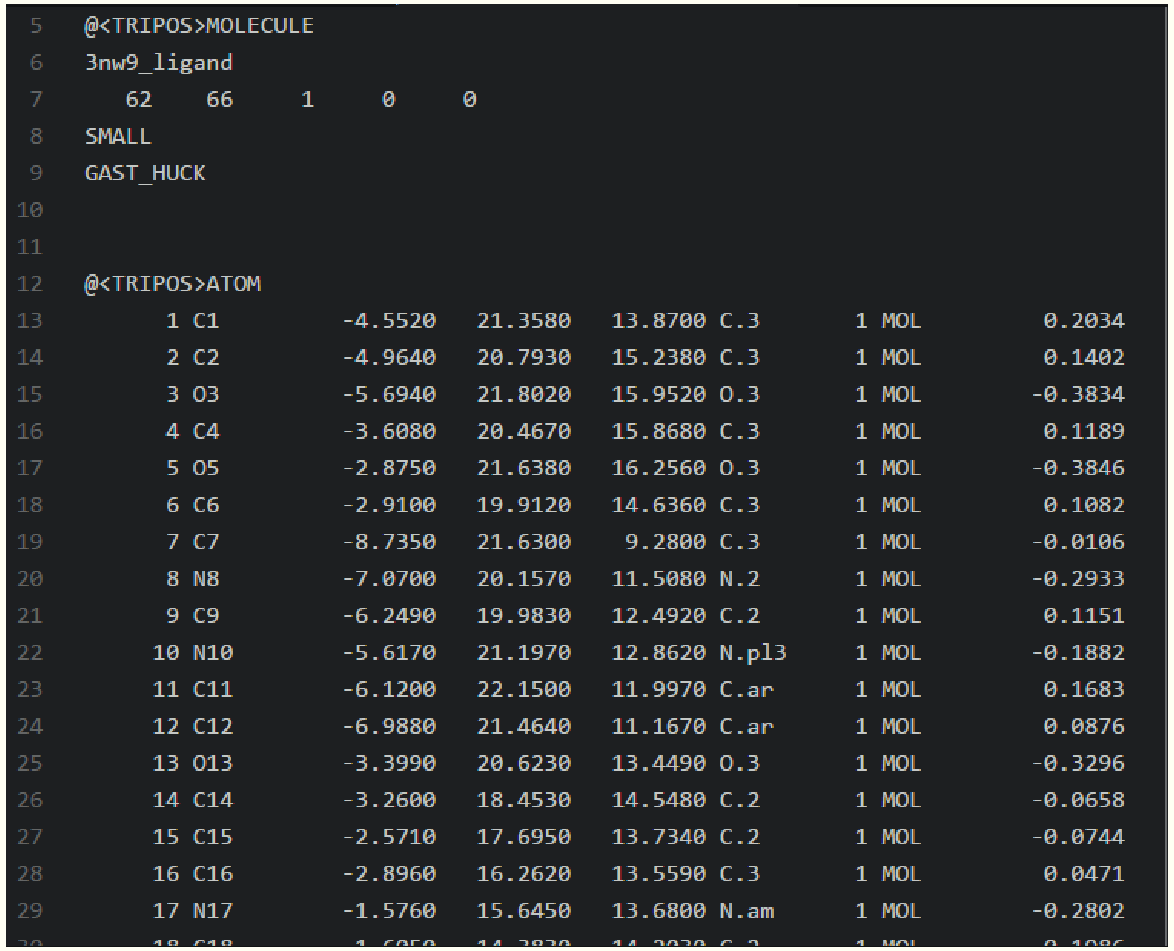
The PDBbind database was originally developed by Prof. Shaomeng Wang's group (http://sw16.im.med.umich.edu) at the University of Michigan in USA, which was first released to the public in May, 2004. This database is now maintained and further developed by Prof. Renxiao Wang's group (http://www.sioc-ccbg.ac.cn) at the Shanghai Institute of Organic Chemistry, Chinese Academy of Sciences under a mutual agreement with the University of Michigan. The PDBbind database is now updated on an annual base to keep up with the growth of the Protein Data Bank.

Protein



```
HEADER    3NW9_PROTEIN
COMPND    3NW9_PROTEIN
REMARK    GENERATED BY X-TOOL on Tue Aug 14 00:53:05 2012
SEQRES   1 A  213  ASP THR LYS GLU GLN ARG ILE LEU ARG TYR VAL GLN GLN
SEQRES   2 A  213  ASN ALA LYS PRO GLY ASP PRO GLN SER VAL LEU GLU ALA
SEQRES   3 A  213  ILE ASP THR TYR CYS THR GLN LYS GLU TRP ALA MET ASN
SEQRES   4 A  213  VAL GLY ASP ALA LYS GLY GLN ILE MET ASP ALA VAL ILE
SEQRES   5 A  213  ARG GLU TYR SER PRO SER LEU VAL LEU GLU LEU GLY ALA
SEQRES   6 A  213  TYR CYS GLY TYR SER ALA VAL ARG MET ALA ARG LEU LEU
SEQRES   7 A  213  GLN PRO GLY ALA ARG LEU LEU THR MET GLU MET ASN PRO
SEQRES   8 A  213  ASP TYR ALA ALA ILE THR GLN GLN MET LEU ASN PHE ALA
SEQRES   9 A  213  GLY LEU GLN ASP LYS VAL THR ILE LEU ASN GLY ALA SER
SEQRES  10 A  213  GLN ASP LEU ILE PRO GLN LEU LYS LYS LYS TYR ASP VAL
SEQRES  11 A  213  ASP THR LEU ASP MET VAL PHE LEU ASP HIS TRP LYS ASP
SEQRES  12 A  213  ARG TYR LEU PRO ASP THR LEU LEU LEU GLU LYS CYS GLY
SEQRES  13 A  213  LEU LEU ARG LYS GLY THR VAL LEU LEU ALA ASP ASN VAL
SEQRES  14 A  213  ILE VAL PRO GLY THR PRO ASP PHE LEU ALA TYR VAL ARG
SEQRES  15 A  213  GLY SER SER SER PHE GLU CYS THR HIS TYR SER SER TYR
SEQRES  16 A  213  LEU GLU TYR MET LYS VAL VAL ASP GLY LEU GLU LYS ALA
SEQRES  17 A  213  ILE TYR GLN GLY PRO
ATOM      1  N   ASP A   3      -2.264  22.783  34.325  1.00 36.29           N
ATOM      2  HN1 ASP A   3      -2.336  23.610  34.951  1.00  0.00           H
ATOM      3  HN2 ASP A   3      -3.124  22.204  34.413  1.00  0.00           H
ATOM      4  HN3 ASP A   3      -1.434  22.216  34.592  1.00  0.00           H
ATOM      5  CA  ASP A   3      -2.118  23.238  32.923  1.00 35.92           C
```

Ligand

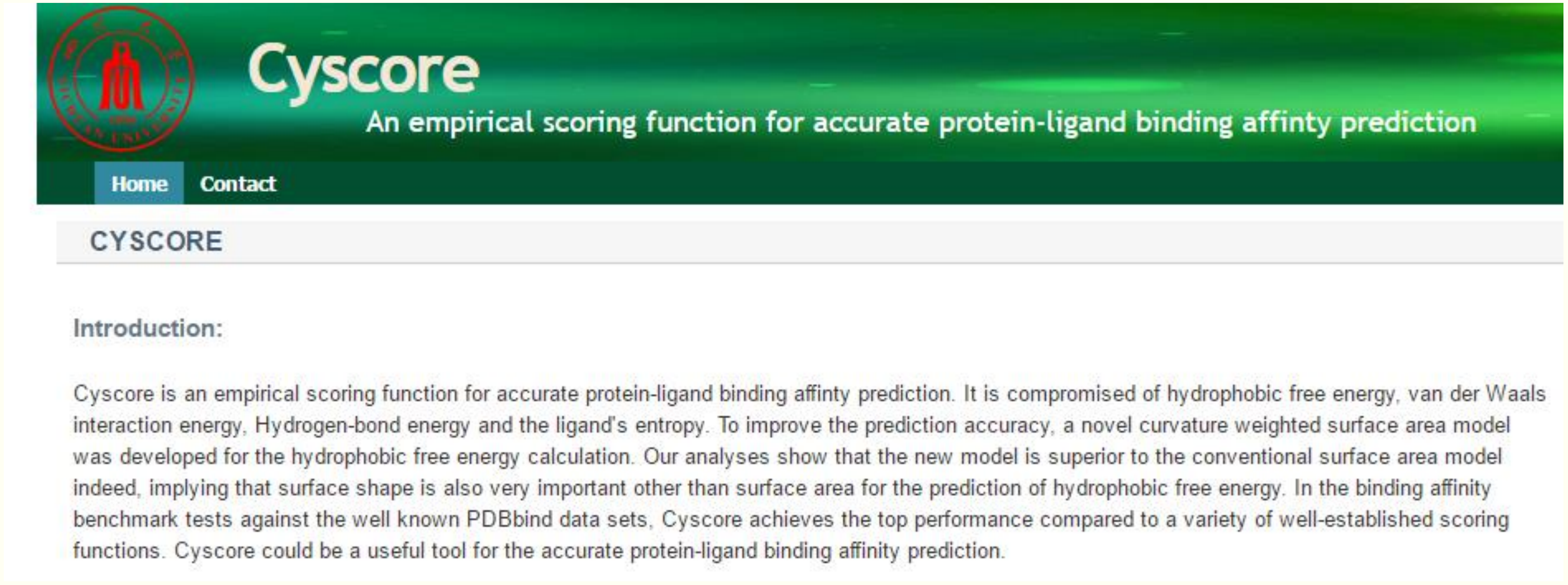


```
@<TRIPOS>MOLECULE
3nw9_ligand
   62    66     1     0     0
SMALL
GAST_HUCK


@<TRIPOS>ATOM
      1 C1         -4.5520   21.3580   13.8700 C.3       1 MOL        0.2034
      2 C2         -4.9640   20.7930   15.2380 C.3       1 MOL        0.1402
      3 O3         -5.6940   21.8020   15.9520 O.3       1 MOL       -0.3834
      4 C4         -3.6080   20.4670   15.8680 C.3       1 MOL        0.1189
      5 O5         -2.8750   21.6380   16.2560 O.3       1 MOL       -0.3846
      6 C6         -2.9100   19.9120   14.6360 C.3       1 MOL        0.1082
      7 C7         -8.7350   21.6300    9.2800 C.3       1 MOL       -0.0106
      8 N8         -7.0700   20.1570   11.5080 N.2       1 MOL       -0.2933
      9 C9         -6.2490   19.9830   12.4920 C.2       1 MOL        0.1151
     10 N10        -5.6170   21.1970   12.8620 N.pl3     1 MOL       -0.1882
     11 C11        -6.1200   22.1500   11.9970 C.ar      1 MOL        0.1683
     12 C12        -6.9880   21.4640   11.1670 C.ar      1 MOL        0.0876
     13 O13        -3.3990   20.6230   13.4490 O.3       1 MOL       -0.3296
     14 C14        -3.2600   18.4530   14.5480 C.2       1 MOL       -0.0658
     15 C15        -2.5710   17.6950   13.7340 C.2       1 MOL       -0.0744
     16 C16        -2.8960   16.2620   13.5590 C.3       1 MOL        0.0471
     17 N17        -1.5760   15.6450   13.6800 N.am      1 MOL       -0.2802
```

Extract features

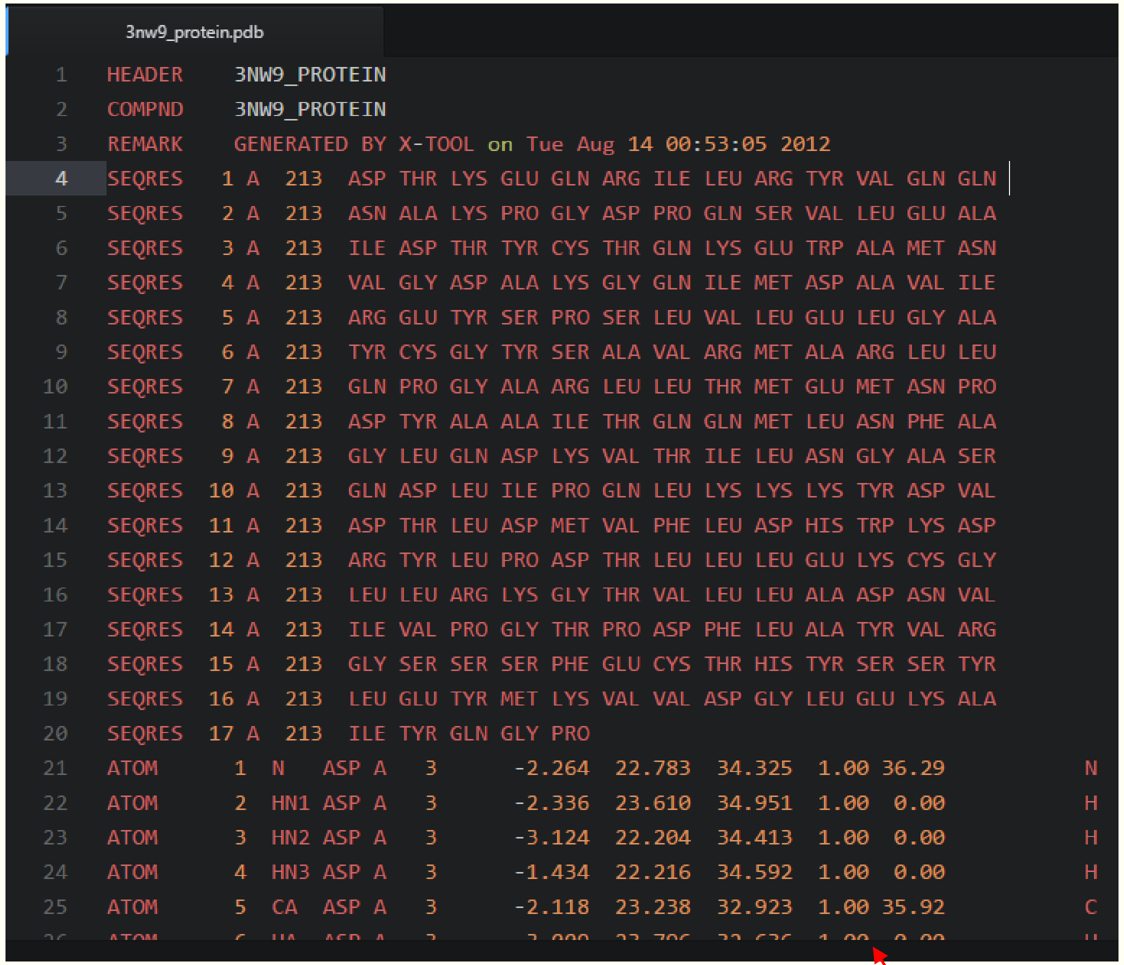
- Cyscore


Cyscore
An empirical scoring function for accurate protein-ligand binding affinty prediction
Home
Contact
CYSCORE

Introduction:

Cyscore is an empirical scoring function for accurate protein-ligand binding affinty prediction. It is compromised of hydrophobic free energy, van der Waals interaction energy, Hydrogen-bond energy and the ligand's entropy. To improve the prediction accuracy, a novel curvature weighted surface area model was developed for the hydrophobic free energy calculation. Our analyses show that the new model is superior to the conventional surface area model indeed, implying that surface shape is also very important other than surface area for the prediction of hydrophobic free energy. In the binding affinity benchmark tests against the well known PDBbind data sets, Cyscore achieves the top performance compared to a variety of well-established scoring functions. Cyscore could be a useful tool for the accurate protein-ligand binding affinity prediction.

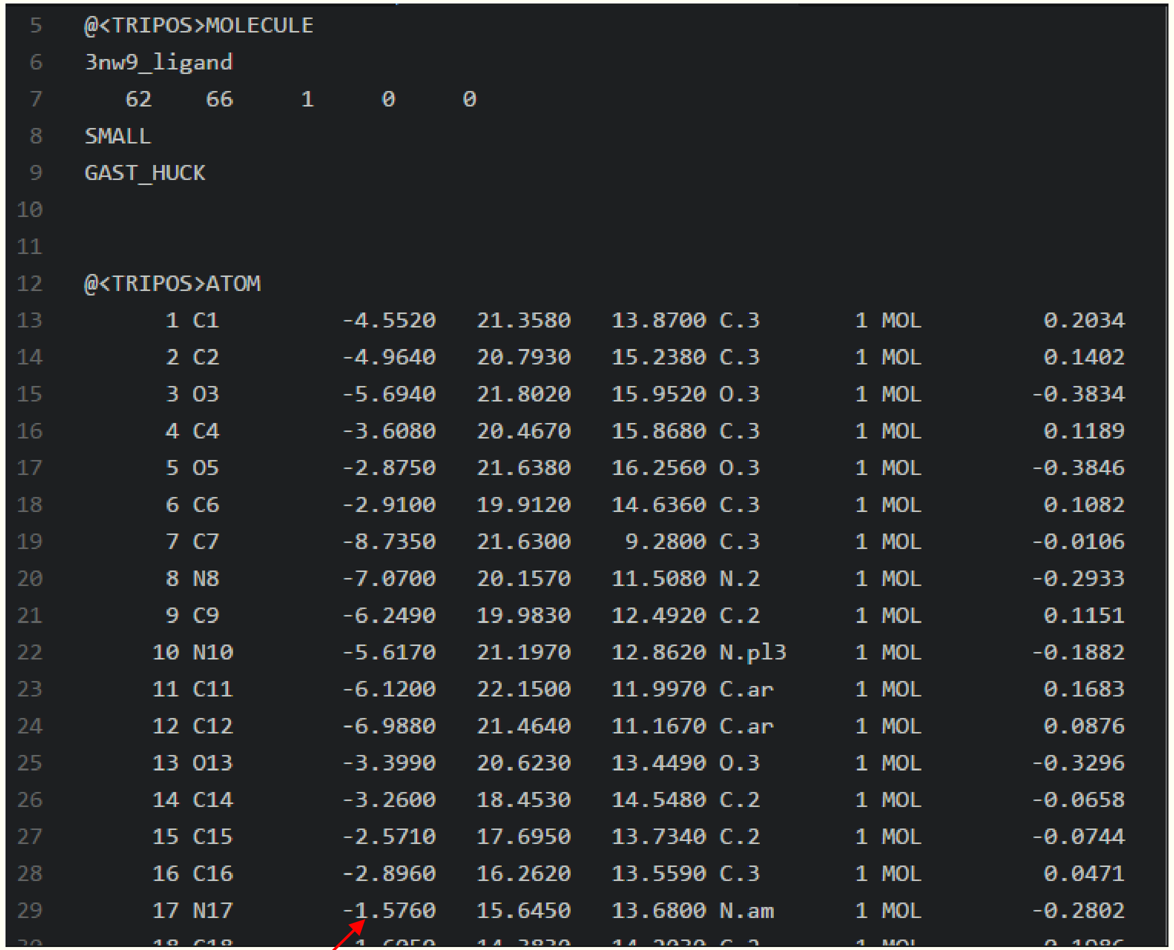
Protein



3nw9_protein.pdb

```
HEADER    3NW9_PROTEIN
COMPND    3NW9_PROTEIN
REMARK    GENERATED BY X-TOOL on Tue Aug 14 00:53:05 2012
SEQRES   1 A  213  ASP THR LYS GLU GLN ARG ILE LEU ARG TYR VAL GLN GLN
SEQRES   2 A  213  ASN ALA LYS PRO GLY ASP PRO GLN SER VAL LEU GLU ALA
SEQRES   3 A  213  ILE ASP THR TYR CYS THR GLN LYS GLU TRP ALA MET ASN
SEQRES   4 A  213  VAL GLY ASP ALA LYS GLY GLN ILE MET ASP ALA VAL ILE
SEQRES   5 A  213  ARG GLU TYR SER PRO SER LEU VAL LEU GLU LEU GLY ALA
SEQRES   6 A  213  TYR CYS GLY TYR SER ALA VAL ARG MET ALA ARG LEU LEU
SEQRES   7 A  213  GLN PRO GLY ALA ARG LEU LEU THR MET GLU MET ASN PRO
SEQRES   8 A  213  ASP TYR ALA ALA ILE THR GLN GLN MET LEU ASN PHE ALA
SEQRES   9 A  213  GLY LEU GLN ASP LYS VAL THR ILE LEU ASN GLY ALA SER
SEQRES  10 A  213  GLN ASP LEU ILE PRO GLN LEU LYS LYS LYS TYR ASP VAL
SEQRES  11 A  213  ASP THR LEU ASP MET VAL PHE LEU ASP HIS TRP LYS ASP
SEQRES  12 A  213  ARG TYR LEU PRO ASP THR LEU LEU LEU GLU LYS CYS GLY
SEQRES  13 A  213  LEU LEU ARG LYS GLY THR VAL LEU LEU ALA ASP ASN VAL
SEQRES  14 A  213  ILE VAL PRO GLY THR PRO ASP PHE LEU ALA TYR VAL ARG
SEQRES  15 A  213  GLY SER SER SER PHE GLU CYS THR HIS TYR SER SER TYR
SEQRES  16 A  213  LEU GLU TYR MET LYS VAL VAL ASP GLY LEU GLU LYS ALA
SEQRES  17 A  213  ILE TYR GLN GLY PRO
ATOM      1  N   ASP A   3      -2.264  22.783  34.325  1.00 36.29           N
ATOM      2  HN1 ASP A   3      -2.336  23.610  34.951  1.00  0.00           H
ATOM      3  HN2 ASP A   3      -3.124  22.204  34.413  1.00  0.00           H
ATOM      4  HN3 ASP A   3      -1.434  22.216  34.592  1.00  0.00           H
ATOM      5  CA  ASP A   3      -2.118  23.238  32.923  1.00 35.92           C
```

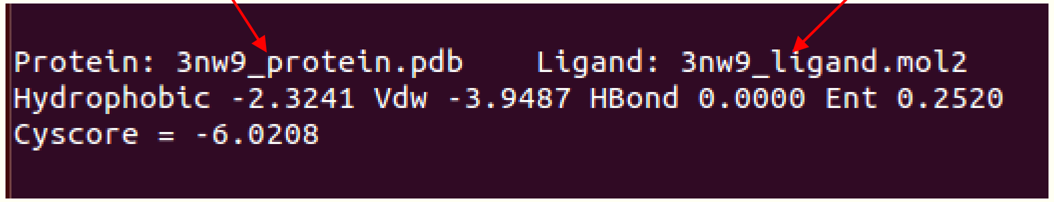
Ligand



```
@<TRIPOS>MOLECULE
3nw9_ligand
   62    66     1     0     0
SMALL
GAST_HUCK


@<TRIPOS>ATOM
      1 C1         -4.5520   21.3580   13.8700 C.3       1 MOL       0.2034
      2 C2         -4.9640   20.7930   15.2380 C.3       1 MOL       0.1402
      3 O3         -5.6940   21.8020   15.9520 O.3       1 MOL      -0.3834
      4 C4         -3.6080   20.4670   15.8680 C.3       1 MOL       0.1189
      5 O5         -2.8750   21.6380   16.2560 O.3       1 MOL      -0.3846
      6 C6         -2.9100   19.9120   14.6360 C.3       1 MOL       0.1082
      7 C7         -8.7350   21.6300    9.2800 C.3       1 MOL      -0.0106
      8 N8         -7.0700   20.1570   11.5080 N.2       1 MOL      -0.2933
      9 C9         -6.2490   19.9830   12.4920 C.2       1 MOL       0.1151
     10 N10        -5.6170   21.1970   12.8620 N.pl3     1 MOL      -0.1882
     11 C11        -6.1200   22.1500   11.9970 C.ar      1 MOL       0.1683
     12 C12        -6.9880   21.4640   11.1670 C.ar      1 MOL       0.0876
     13 O13        -3.3990   20.6230   13.4490 O.3       1 MOL      -0.3296
     14 C14        -3.2600   18.4530   14.5480 C.2       1 MOL      -0.0658
     15 C15        -2.5710   17.6950   13.7340 C.2       1 MOL      -0.0744
     16 C16        -2.8960   16.2620   13.5590 C.3       1 MOL       0.0471
     17 N17        -1.5760   15.6450   13.6800 N.am      1 MOL      -0.2802
```

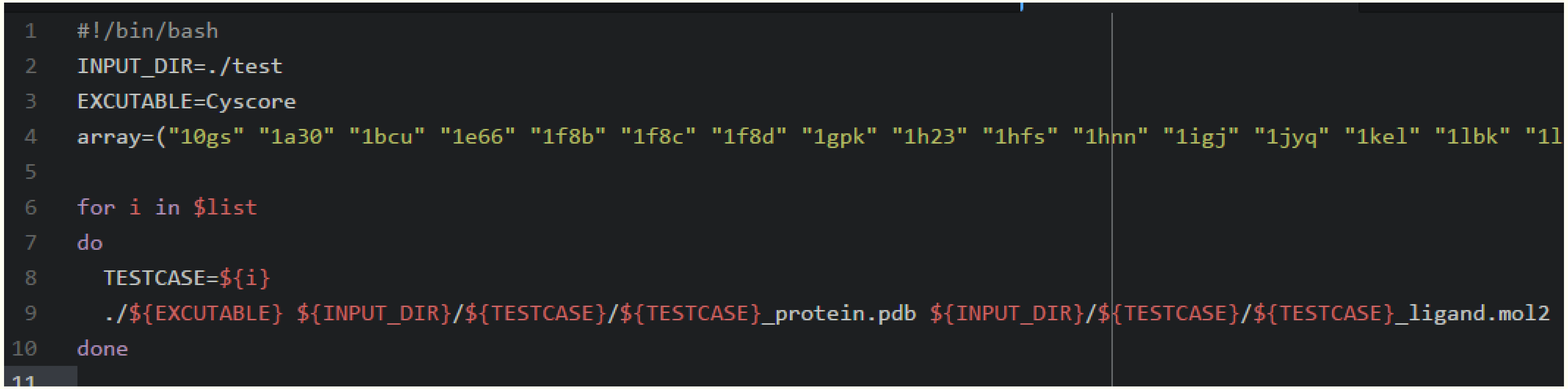
Features



```
Protein: 3nw9_protein.pdb    Ligand: 3nw9_ligand.mol2
Hydrophobic -2.3241 Vdw -3.9487 HBond 0.0000 Ent 0.2520
Cyscore = -6.0208
```

Extract features

Because Cyscore is a command line application under Linux-x86, so I write a program to grep all the names of the corresponding file in pdbbind_v2014_core_set and write a shell script to run Cyscore to get all their features:



```
#!/bin/bash
INPUT_DIR=./test
EXCUTABLE=Cyscore
array=("10gs" "1a30" "1bcu" "1e66" "1f8b" "1f8c" "1f8d" "1gpk" "1h23" "1hfs" "1hnn" "1igj" "1jyq" "1kel" "1lbk" "1l

for i in $list
do
  TESTCASE=${i}
  ./${EXCUTABLE} ${INPUT_DIR}/${TESTCASE}/${TESTCASE}_protein.pdb ${INPUT_DIR}/${TESTCASE}/${TESTCASE}_ligand.mol2
done
```

Conclusion

- **Breakthrough point (maybe)**
  - The data Jacky used in his pass research is derived by the Cyscore v1.1.4, and now the author of Cyscore publishes the Cyscore v2.0.0 and it improves the hydrophobic free energy calculation.
  - Try to derive a set of new features of pdbbind_v2015_refined_set, and see if it can improve the prediction accuracy.

☺ thanks