Structural Biology: Assignment 3

University of Cambridge

Henrik Åhl

June 15, 2017

Preface

This is an assignment report in connection to the *Structural Biology* module in the Computational Biology course at the University of Cambridge, Lent term 2017. All related code is as of June 15, 2017 available through a Github repository by contacting hpa22@cam.ac.uk.

Exercises

1 The code produces in total eight different files, each representing various types of information produced throughout the analysis. The four figures produced can be seen in fig. 5 for DYR_ECOLI under the default parameters. Figure 5a corresponds to the distribution of weighted pairwise matches between sequences, given that their similarity fraction is greater than θ (70 %). Here, for every sequence, the number of *other* sequences with a > 70 % similarity to this adds an increment of one to the parameter d_i in the equation

$$W_i = \frac{1}{1 + d_i}.$$

in order to downscale weights for the sequences that are high in abundance (given our similarity threshold), since they add little additional information to what is already known. Figure 5b shows the predicted protein contact map between the residues in each sequence, i.e. how correlated certain positions are with each other based on the evolutionarily inferred contact (EIC) scores. Figure 5c in turn depicts the minimal distance between the amino acids in the crystal as a function of the corresponding DI rank, for the top 200 pairs. Lastly, fig. 5d shows the overlap between the experimentally observed structure and the predicted tertiary interactions, i.e. fig. 5b overlaid on the experimental data.

The output text and data files in turn contain information used in these plots to various extents. The MI_DI files contain information about the pairwise residues and their corresponding mutual information (MI) and direct information (DI) scores. Similarly, the aptly named DIScores files contain collected information about all the DI scores in the predicted and experimental case, as well as information about the physical distance and the conservation percentage for the pair in the alignments.

2 The accuracy of the method is directly dependent on the quality of the sequence alignment. As these often utilize multiple heuristic simplification in order to improve speed, longer sequences are bound to align improperly. While functionally important segments should be better conserved and align better, there is always the risk of error in this, which makes the method as a whole reliant both on the alignment method used, and the numbers of sequences aligned. Using more elaborate approaches such as direct dynamic programming is instead feasible when analysing smaller and/or fewer proteins. While cumbersome and time-consuming, one can also, by using prior information of how the sequences in question differ, tailor the alignment according to data at hand in order to improve upon this.

Another problem is the quality of the exper-

imental correlations, which depend both on the sample size and resolution of the protein in question. Depending on how different the homologs are from each other, they might suffer from severly different biological and technical variances.This problem might also obfuscate lowly correlated but functionally important residues, which affect the overall structure and stability by long-range interactions or simply aid the protein in folding into the native state.

Lastly, as a note besides the directly sequence oriented problems, the alignment data does not take into account environmental factors, such as how certain parts of the protein might interact with other molecules in a way such that protein functionality invisible in the primary sequence is conserved or maintained. If this is the case, and this functionality is not conserved spatially in relation to other residues, this might as well be obfuscated. In other words, while a high score indicates contact, a low score must not necessarily mean no contact between residues.

Other factors, such as the sequence length and variability between samples are also important, but are discussed throughout the report.

3 Figure 1 shows the change in percentage TP when amino acids of decreasing ranks are included in the measure. Clearly, when using the best ranking amino acids, there is a high certainty in the predicability of the model. However, quite quickly there is a steep decline, and the ability to predict couplings reach 50 % at ca rank 175. The initial dip is due to a low abundance in the pairs which are ranked the highest, which regains its ground as soon as more frequent pairs are taken into account.

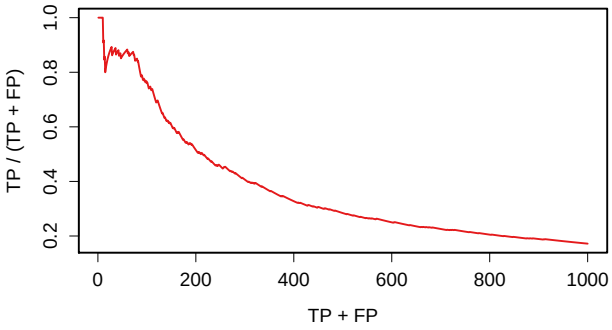


Figure 1

4 The parameter θ sets, as previously hinted, the similarity percentage cutoff between sequences. In our case, as it is set to $\theta = 0.3$, only sequences with more than $1.0 - 0.3 = 0.7 = 70$ % similarity will be processed when calculating the weights. This makes it so that, according to the aforementioned equation, sequences which are very similar will add less to the corresponding weights, as the information is duplicated in several parts of the input data, which here are the homologues. By doing so, we smoothen our distribution of weights, and thus makes it more variable due to smaller variation. We can also note how changing the θ parameter somewhat removes the initial bump, signifying that the initial dip indeed is because of low-abundance pairs distorting the prediction.

5 Pseudocounts are used for smoothing distributions in particular when some specific probabilistic outcomes are unlikely to occur in relation to the size of the data set. In the case of amino acids, this could be to add and increment to the number of times a residue has been observed. For each amino acid, a different increment would be attributed, based on a prior estimates of how likely they are to appear. This approach is done in order to avoid sharp peaks in the distribution of values in for example a position weight matrix, where small changes in the number of observations would have large effects the resulting value, in the case where pseudocounts are not used. In practice, pseudocounts can be set to calculate the posterior estima-

tor of the mean as

$$\theta_i^{PME} = \frac{n_i + \alpha_i}{|\mathbf{n}| + |\alpha|}$$

where n_i is the number of observations of component i, and α is the pseudocounts for that component [1].

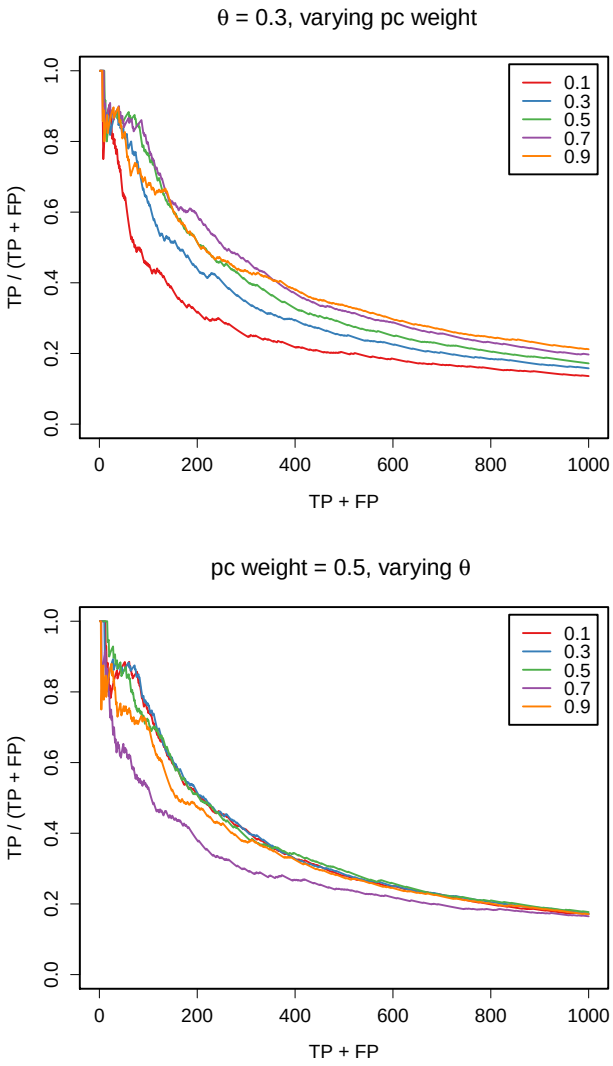


Figure 2

6 Figure 2 shows the changes in predictability when changing the two parameters. As we can note, changing the pseudocount weight appears to saturate at around 0.6. Having values under this makes the model underperform, where the lowest pseudocount weight leads to the worst performance. When instead keeping the pseudocount weights constant and instead changing θ, we see that there is a balancing act between extracting the vital information and managing not to throw it out. As we can see from the figure, only very little of the material is needed to be discarded in order to get a good estimate in this case.

In fig. 3 we can observe the sensitivity and precision for pseudocount weights and θ both in the range $[0, 1]$. While the minimum is clearly identifiable, there is no homogeneous slope towards it. Nevertheless, we

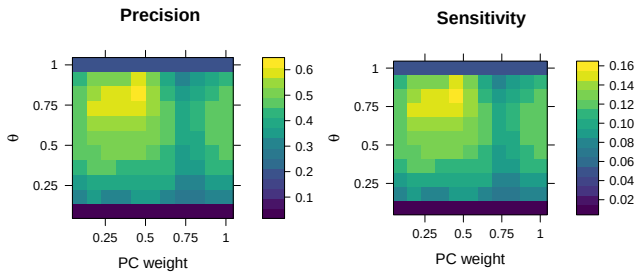


Figure 3

7 When comparing the three different proteins with each other using the default parameters we get the results shown in fig. 4. Notably, LACI_ECOLI performs the best, with the other bacterial species coming after. The human protein performs the worst in general. The origin of the Human protein is as it is simply because the highest ranked prediction is a false positive.

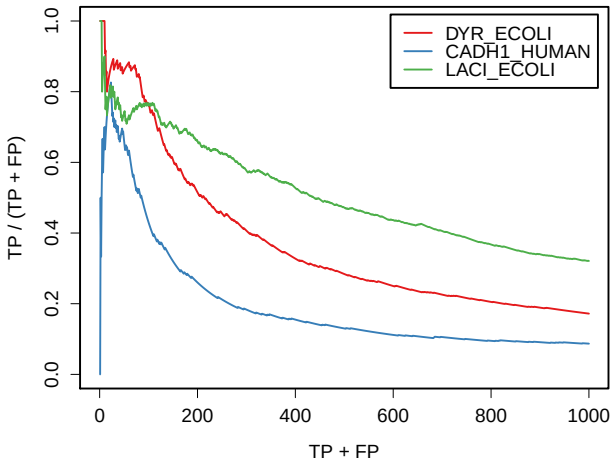


Figure 4

8 Given fig. 4, we can tie this information to the sequence lengths. From our data, we have that our data for LACI_ECOLI contains sequences of at most 360 residues, whereas CADH1_HUMAN has a more modest length of 151, and DYR_ECOLI of 159. Normally, we would expect sequences of longer lengths to match worse due to the increased risk of missmatches and tendency to include non-functional residues. When the sequences are longer, the low-abundance subsequences will also cause the distribution landscape to be rougher and hence not as robust in performance.

However, in our figure, we see the opposite effect, where longer sequences generally have higher predictive quality, aside from a discrepancy for high-scoring pairs. This could be because of different degrees of variance between homologs in our species, due to for example insertions and deletions. Indeed, a visual check on this affirms that the biological variance between the three cases is significant, with in particular DYR_ECOLI being generally very well conserved, whereas CADH1_HUMAN suffers from more indels than the ECOLI counterparts. It is also the case that LACI_ECOLI has the most sequences included in the data set, which might simply entail that the difference we see is due to a relative lack of data in the other cases.

9 Using the same parameters is clearly not a satisfying solution due to the differences in biological variance (and subsequently the error in alignment). There are naturally also differences in the frequency of the amino acids, which directly should affect the distribution of pseudocounts and their relative importance. For a new protein, it would be beneficial to take into account the factors previously mentioned, such as sequence lengths, variance in this, and conservation and variance in the corresponding subsequences.

10 The accuracy of the method depends inherently on the quality of the sequence alignment, and as a consequence then on the method used for aligning. Ideally, dynamic programming would render the mathematically optimal fit between the sequences, although the large number of sequences makes it computationally impossible for any substantial studies. This means that heuristic methods

must be applied in order to speed up the alignment, and the design and choice of these are thus bound to affect the quality of the alignment. Taking into account as much available data as possible, for example about the insertion/deletion frequencies and the abundance of the various k-mers can then be used to improve upon this [2].

Similarly to this, still relating to the sequences, is the accuracy of the data itself, e.g. in the case of the 3D structure. Many proteins have structures which are difficult to capture precisely, which means that the sequence data and the corresponding fold sometimes are not very reliable, and depends on the method used to capture the conformation [3, 4].

Aside of the sequences themselves, our approach does not take into account environmental factors, such as interacting ligands and ions which might aid or obstruct certain types of folds and change the overall folding trajectory. While highly correlated pairs might have importance in folding and making the protein reaching the native state, our method is unable to account for the fact that this does not happen in an isolated environment [5].

References

[1] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[2] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

[3] CF Pena, GL Mulvey, GA Snyder, L De Masi, EC von Rosenvinge, S Günther, GD Armstrong, MS Donnenberg, EJ Sundberg, M Tomich, et al. A consensus on protein structure accuracy in nmr? *Biol. Rev*, 77: 323–341.

[4] Ramachandran Rakesh and Narayanaswamy Srinivasan. Improving the accuracy of fitted atomic models in cryo-em density maps of protein assemblies using evolutionary information from aligned homologous proteins. *Data Mining Techniques for the Life Sciences*, pages 193–209, 2016.

[5] Debora S Marks, Thomas A Hopf, and Chris Sander. Protein structure prediction from sequence variation. *Nature biotechnology*, 30(11): 1072–1080, 2012.

A Code

```
#!/usr/bin/env Rscript
setwd("/local/data/public/hpa22/compbio/src/assignments/sba3/code/")
# Do the track analysis
library(plyr)
library(RColorBrewer)
palette(brewer.pal(n = 8, name = "Set1"))
lw.s = 2
par(lwd=lw.s, cex = 1.5, ps = 15)
library(prodlim)

scores = read.csv("../figures/DYR_ECOLI_e3_n2_m40_theta_0.3_pc_weight_0.5_
    DIScoresCompared.csv",    sep = ",", header = F)


TPs = function(nIncluded) {
  TP = sum(scores[1:nIncluded, 4] < 5.0)
  TP / nIncluded
}

fract = sapply(1:nrow(scores), TPs)
par(mfrow = c(1, 1))

plot(fract, type = "l", ylab = "TP", xlab = "TP + FP")
```

../code/exc_3.R

```
#!/usr/bin/env Rscript
setwd("/home/henrik/compbio/src/assignments/sba3/code/")
# Do the track analysis
library(plyr)
library(RColorBrewer)
palette(brewer.pal(n = 8, name = "Set1"))
lw.s = 2
par(lwd=lw.s, cex = 1.5, ps = 15)
library(prodlim)


real.files = list.files("../figures", pattern = "DIScoresC", full.names = T)
real.files = grep("DYR", real.files, value = T)
pred.files = list.files("../figures", pattern = "prediction_contacts.csv", full.names =
    T)
pred.files = grep("DYR", pred.files, value = T)

order.files  = as.numeric(sapply(sapply(real.files, function(x) strsplit(x,"_theta_")
    [[1]][2]), function(x) strsplit(x, "_")[[1]][1]))
order.files2 = as.numeric(sapply(sapply(real.files, function(x) strsplit(x,"pc_weight_")
    [[1]][2]), function(x) strsplit(x, "_")[[1]][1]))

real.files = real.files[order(order.files, order.files2)]
pred.files = pred.files[order(order.files, order.files2)]

a = expand.grid(unique(order.files), unique(order.files2))
a = a[order(a[,1], a[,2]), ]
name = paste(a[,1], a[,2], sep = "_")

```

```

theta.0.3.files = which(a[,1] == 0.3)[c(2,4,6,8,10)]
pc.0.5.files = which(a[,2] == 0.5)[c(2,4,6,8,10)]

TPs = function(nIncluded, scores) {
  TP = sum(scores[1:nIncluded, 4] < 5.0)
  TP / nIncluded
}

results = list()
for(ii in real.files[theta.0.3.files]){
  scores = read.csv(ii, sep = ",", header = F)
  fract = sapply(1:nrow(scores), function(x) TPs(x, scores))
  results = append(results, list(fract))
}

par(mfrow = c(2,1))
plot(NA, ylim= c(0,1), xlim = c(0,1000), xlab = "TP + FP", ylab = "TP / (TP + FP)", main = expression(" "*theta*" = 0.3, varying pc weight"))
for(ii in 2:length(results)) lines(results[[ii]], col = ii)
legend("topright", legend=a[theta.0.3.files, 2], col = 1:5, lty = 1, lwd = lw.s, inset = c(0.02,0.02))

################################################################################
results = list()
for(ii in real.files[pc.0.5.files]){
  scores = read.csv(ii, sep = ",", header = F)
  fract = sapply(1:nrow(scores), function(x) TPs(x, scores))
  results = append(results, list(fract))
}
plot(NA, ylim= c(0,1), xlim = c(0,1000), xlab = "TP + FP", ylab = "TP / (TP + FP)", main = expression("pc weight = 0.5, varying "*theta*" "))
for(ii in 1:length(results)) lines(results[[ii]], col = ii)
legend("topright", legend=a[pc.0.5.files, 1], col = 1:5, lty = 1, lwd = lw.s, inset = c(0.02,0.02))
```

../code/exc_6.R

```

#!/usr/bin/env Rscript
setwd("/home/henrik/compbio/src/assignments/sba3/code/")
# Do the track analysis
library(plyr)
library(RColorBrewer)
palette(brewer.pal(n = 8, name = "Set1"))
lw.s = 2
par(lwd=lw.s, cex = 1.5, ps = 15)
library(prodlim)
library(lattice)
library(gridExtra)
library(viridis)

real.files = list.files("../figures", pattern = "crystal_contacts.csv", full.names = T)
real.files = grep("DYR", real.files, value = T)
pred.files = list.files("../figures", pattern = "prediction_contacts.csv", full.names = T)
pred.files = grep("DYR", pred.files, value = T)

order.files  = as.numeric(sapply(sapply(real.files, function(x) strsplit(x,"_theta_")
```

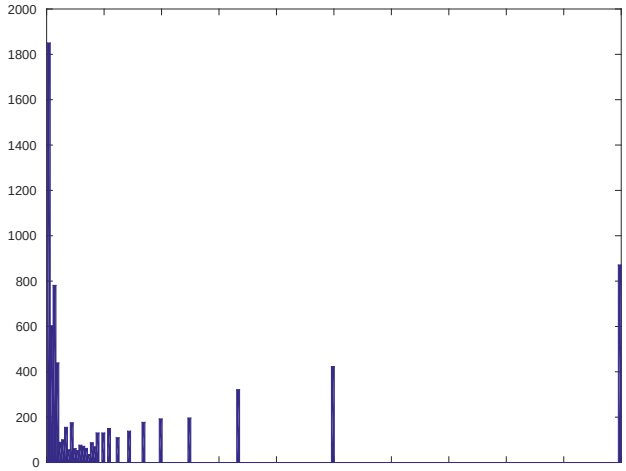
```
    [[1]][2]), function(x) strsplit(x, "_")[[1]][1]))
order.files2 = as.numeric(sapply(sapply(real.files, function(x) strsplit(x,"pc_weight_")
    [[1]][2]), function(x) strsplit(x, "_")[[1]][1]))

real.files = real.files[order(order.files, order.files2)]
pred.files = pred.files[order(order.files, order.files2)]

a = expand.grid(unique(order.files), unique(order.files2))
a = a[order(a[,1], a[,2]), ]
name = paste(a[,1], a[,2], sep = "_")

results = sapply(1:length(real.files), function(x){
  pred = read.csv(pred.files[x], sep = ",")
  real = read.csv(real.files[x], sep = ",")

  TP = sum(!is.na(row.match(pred, real))) # Is in both
  FP = sum(is.na(row.match(pred, real)))  # Is in pred but not in real
  FN = sum(is.na(row.match(real, pred)))  # Is in real but not in pred

  sens = TP / (TP + FN)
  prec = TP / (TP + FP)
  list(TP = TP, FP = FP, FN = FN, sens=sens, prec=prec)
})

colnames(results) = name

# x is PC-weight, y is
prec = results["prec",]
sens = results["sens",]
prec.m = matrix(unlist(prec), ncol=11, nrow=11, byrow=T)
sens.m = matrix(unlist(sens), ncol=11, nrow=11, byrow=T)
par(mfrow=c(2,1))
plot1 = levelplot(prec.m,  col.regions = viridis(21), xlab = "Pseudocount weight", ylab
    = expression(theta), main = "Precision", panel = panel.levelplot.raster)
plot2 = levelplot(sens.m,  col.regions = viridis(21), xlab = "Pseudocount weight", ylab
    = expression(theta), main = "Sensitivity")
grid.arrange(plot1, plot2, ncol = 1)
```

../code/tp_plot.R

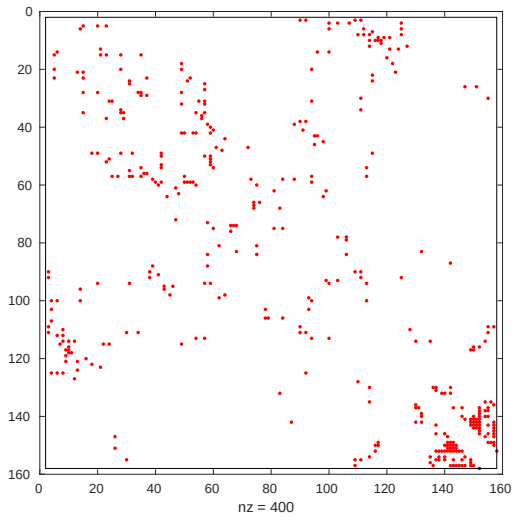
```
#!/usr/bin/env Rscript
setwd("/home/henrik/compbio/src/assignments/sba3/code/")
# Do the track analysis
library(plyr)
library(RColorBrewer)
palette(brewer.pal(n = 8, name = "Set1"))
lw.s = 2
par(lwd=lw.s, cex = 1.5, ps = 15)
library(prodlim)
real.files  = list.files("../figures", pattern = "DIScoresC", full.names = T)
real.files  = grep("DYR", real.files, value = T)
order.files  = as.numeric(sapply(sapply(real.files, function(x) strsplit(x,"_theta_")
    [[1]][2]), function(x) strsplit(x, "_")[[1]][1]))
order.files2 = as.numeric(sapply(sapply(real.files, function(x) strsplit(x,"pc_weight_")
    [[1]][2]), function(x) strsplit(x, "_")[[1]][1]))
real.files  = real.files[order(order.files, order.files2)]

a = expand.grid(unique(order.files), unique(order.files2))
```

```r
a = a[order(a[,1], a[,2]), ]
name = paste(a[,1], a[,2], sep = "_")


theta.0.3.files = which(a[,1] == 0.3)[c(2,4,6,8,10)]
pc.0.5.files    = which(a[,2] == 0.5)[c(2,4,6,8,10)]

TPs = function(nIncluded, scores) {
  TP = sum(scores[1:nIncluded, 4] < 5.0)
  TP / nIncluded
}

results = list()
for(ii in real.files[theta.0.3.files]){
  scores = read.csv(ii, sep = ",", header = F)
  fract = sapply(1:nrow(scores), function(x) TPs(x, scores))
  results = append(results, list(fract))
}

par(mfrow = c(2,1))
plot(NA, ylim= c(0,1), xlim = c(0,1000), xlab = "TP + FP", ylab = "TP / (TP + FP)", main
    = expression(" "*theta*" = 0.3, varying pc weight"))
for(ii in 1:length(results)) lines(results[[ii]], col = ii)
legend("topright", legend=a[theta.0.3.files, 2], col = 1:5, lty = 1, lwd = lw.s, inset =
    c(0.02,0.02), bg = "white")

##############################################################################
results = list()
for(ii in real.files[pc.0.5.files]){
  scores = read.csv(ii, sep = ",", header = F)
  fract = sapply(1:nrow(scores), function(x) TPs(x, scores))
  results = append(results, list(fract))
}
plot(NA, ylim= c(0,1), xlim = c(0,1000), xlab = "TP + FP", ylab = "TP / (TP + FP)", main
    = expression("pc weight = 0.5, varying "*theta*" "))
for(ii in 1:length(results)) lines(results[[ii]], col = ii)
legend("topright", legend=a[pc.0.5.files, 1], col = 1:5, lty = 1, lwd = lw.s, inset = c
    (0.02,0.02), bg = "white")
```

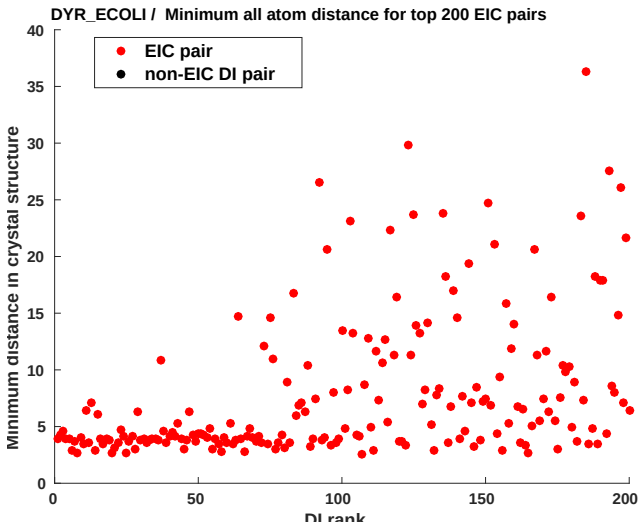
../code/varying.R



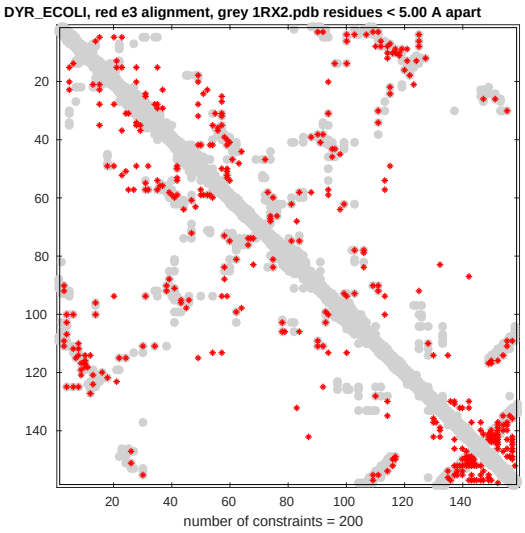
(a)



(b)



(c)



(d)

Figure 5