AlphaFold による複数の準安定状態をもつタンパク質の構造予測

分子科学研究所・計算科学研究センター

大貫　隼

AlphaFold-based structure prediction of proteins with multiple metastable states

Research Center for Computational Science, Institute for Molecular Science
Jun Ohnuki

（投稿日 2024/11/30、再投稿日 2025/1/10、受理日 2025/2/18）

キーワード：AlphaFold、構造予測、AI

概要

2024 年のノーベル化学賞に象徴されるように、Google DeepMind 社が開発した AlphaFold2（AF2）はタンパク質構造予測に革命を起こした。本稿を執筆している 2024 年 11 月には最新版である AlphaFold3（AF3）もオープンソース化され、リガンドや核酸との複合体構造も非商業目的であれば自由に予測を行えるようになった。AF2 や AF3 の解説や基本的な利用方法は多くの Web ページや書籍で既にまとめられているので、本稿では AF2 を用いた複数の準安定状態をもつタンパク質の構造予測に焦点を当て、その手順をまとめる。AF2（おそらく AF3 も）は予測にバイアスが存在することが知られており、複数状態を探索するには工夫が必要となる。ここでは AF2 での構造予測に用いられる多重配列アラインメントのサブサンプリング、また共進化情報をもとにした変異導入によって構造予測のバイアスを克服する方法を示す。

イントロダクション

AF2 の登場以降、タンパク質の構造予測そしてデザインにおいて AI は中心的な役割を担うようになった。それを象徴するように、2024 年のノーベル化学賞はタンパク質デザインで先駆的研究を行う David Baker 氏（ワシントン大学）とともに、AF2 を開発した Google DeepMind 社の Demis Hassabis 氏、John Jumper 氏に授与されることとなった。AF2 はアミノ酸配列から多重配列アラインメント（MSA）を構築し、この MSA に含まれる情報からニューラルネットワークを用いてタンパク質構造を高精度で予測する[1]。本稿を執筆している 2024 年 11 月には最新版である AF3 もオープンソース化され、リガンドや核酸との複合体構造も非商業目的であれば自由に予測を行えるようになった[2]。AF2 や AF3 によってタンパク質の構造的理解は今後ますます進んでいくだろう。

AF2 は高精度のタンパク質構造予測を可能にし、構造生物学、構造バイオインフォマティクスに革命を起こしたが、弱点も指摘されていた。それが構造予測のバイアスである。タンパク質は多くの場合、複数の準安定状態を有し、それらの状態間を切り替えることで機能する。しかし AF2 の予測は特定の構造状態に偏ってしまい、構造状態探索が困難とされてきた。このバイアスは学習データやクエリ配列から構築される MSA にある偏りに起因すると考えられ、テンプレート構造の利用や MSA を改変する等によって克服する方法が提案されている[3]。

本稿ではトランスポータータンパク質を対象に著者らが取り組んできた内容をもとに、複数構造状態を効率的に探索するプロトコールをまとめる[4,5]。AF2 や AF3 の基本的な使い方は多くの Web ページや書籍にまとめられているので（例えば[6]）、複数構造状態をどのように AF2 で探索するかに重点を置き紹介する。なお、本稿では AF2 のみを取り扱うが、構造予測のバイアスは AF3 にも依然として潜んでいることが報告されている[2]。そのためここで紹介する方法は AF3 にも応用できる可能性はあるが、現時点では未検証である。現在もしくは今後 AF3 を利用される本稿の読者には、是非ここで学んだ方法やアイデアが AF3 にも適用できるかどうか研究を進めていってほしい。

装置・ソフトウェア

AF2 は GitHub からダウンロードしインストールすれば使用できるが、3 テラバイト程度のディスク容量に加え GPU も必要となってくるので、自前の計算機で気軽に始めるにはハードルが高い。これを回避するには、例えばスパコンの利用が 1 つの手である。例えば、分子科学研究所の計算科学研究センターは AF2、AF3 ともに導入済みのスパコンを保有しており、利用申請と審査を経ることで利用可能である。より気軽に始めるには Google Colab 版の AF2 や、AF2 の MSA 構築を高速化した ColabFold（CF）[7]を利用すれば良い。これらはクラウド環境で構造予測を行うので、ローカル環境にインストールする必要がなくすぐに予測を開始できる。CF では後述する MSA サブサンプリングのためのパラメータを調整できる。また、CF をローカル環境で実行する LocalColabFold は、本家の AF2 と異なり大規模サイズのデータベースを用意する必要がなく、また Google Colab 版のような利用時間等の制限がないため、大量の構造予測を行いたい場合などに便利である。

本稿で紹介するプロトコールでは LocalColabFold を利用した（以後、特に断りがない限り LocalColabFold を指して略称 CF を用いる）。導入手順は CF の GitHub ページを参照されたい。CF は NVIDIA 製の GPU を搭載したマシンであれば、Linux、Windows（WSL を利用）で高速に実施できる。また、複数構造状態を探索するための変異導入では、変異を導入するアミノ酸残基を決定するため、MSA の Direct Coupling Analysis（DCA）を行う。そのために必要な python ライブラリとして、pydca[8]を導入しておく。

計算手順

１）アミノ酸配列の用意
２）ColabFold による構造予測
３）MSA サブサンプリング
４）Direct Coupling Analysis と変異導入

計算の詳細

１）アミノ酸配列の用意

構造を予測したいタンパク質のアミノ酸配列が AF2/CF の入力となる。アミノ酸配列は UniProt データベース等から入手できる。ダウンロードされたファイルは fasta 形式となっており、これをそのまま次の CF の入力に利用できる。なお多くのタンパク質では、UniProt の Structure 欄に X 線結晶構造解析等で得られた既知構造とともに AF2 によって予測された構造も記載されている。こうした AF2 予測構造は AlphaFold Structure Database[9]にまとめられている。構造予測を始める前にまずこの情報も見てみると良いだろう。

２）ColabFold による構造予測

CF は以下のようにして実行できる。

```
$ colabfold_batch --num-seeds 20 --max-msa 512:1024 input.fasta ./output_dir/
```

input.fasta が上記 1 で入手した fasta 形式のアミノ酸配列であり、予測結果は output_dir 下に保存される。output_dir に保存される出力は構造ファイル（PDB 形式）と予測スコアである（予測スコアについては次の 3 で言及する）。ここで用いたオプションの意味は以下の通りである（ここでは用いていないオプションについては CF の GitHub ページを参照のこと）。

--num-seed：乱数の種の数を指定する。上の例では 20 を指定。乱数の種ごとに 5 つの予測構造モデルが出力されるので、ここでは合計 100 構造が得られる。
--max-msa：MSA の中から実際に構造予測に使用されるアミノ酸配列の数（これを “MSA の深さ” と表現することがある）を指定する。2 つの整数を指定し、前者は MSA をクラスタリングした後に実際に予測に用いられるクラスターの数、後者はそれとは別に追加で選ばれ構造予測時の情報として用いられる配列の数である。小さな値を設定すると “浅い” MSA を用い構造予測が行われる。

例として、硝酸と亜硝酸を輸送するトランスポーターNarK のアミノ酸配列（UniProt ID: P10903、アミノ酸残基数 : 463）に対し構造予測を行う場合、著者の計算環境（GPU: NVIDIA A4000, CPU: Intel Core i7-12700）では 2 時間弱で 100 構造が得られる。

３）MSA サブサンプリング

AF2/CF ではクエリ配列から MSA を構築し、MSA に含まれる類縁タンパク質のアミノ酸配列集団をランダムにサブサンプリングして構造予測を行う。MSA サブサンプリングによって集める配列数（MSA の深さ）は、CF のオプション--max-msa によって指定される。AF2/CF は MSA から得られるアミノ酸残基間の共進化情報を構造予測に利用しているので、MSA の深さを浅くすることで共進化情報の統計的不確かさが増大し、多様な構造状態が予測できるようになる[10]。

先に例として挙げた NarK について、オプション--max-msa の値を 512:1024、256:512、128:256、64:128、32:64、16:32、8:16 と減らしていったときの予測構造の分布を図 1 に示す[5]。トランスポータータンパク質は外開き、閉、内開きの三構造状態を交互に切り替えることで基質を輸送するが、深い MSA（例えば 512:1024）では内開き構造のみが予測されて

しまう。一方で、MSA を浅くしていくと閉構造や外開き構造も含んだ幅広い構造状態が予測されていることがわかる。なお、この例の NarK では外開き構造は実験的に得られておらず、CF によって未解明構造も予測できることを示している。予測構造の信頼性は、CF が出力するいくつかのスコアによって評価できる。例えば pLDDT スコアはアミノ酸残基周囲の局所環境の信頼性を 0 から 100 までの数値で表し、値が大きいほど信頼性が高いことを示す。NarK について予測された外開き構造を図 2 で示し、pLDDT スコアに従って各残基を色付けしている。N 末端や C 末端など柔軟性が高く特定の構造をとらないような領域を除いて、高い信頼度で予測構造が得られていることがわかる。

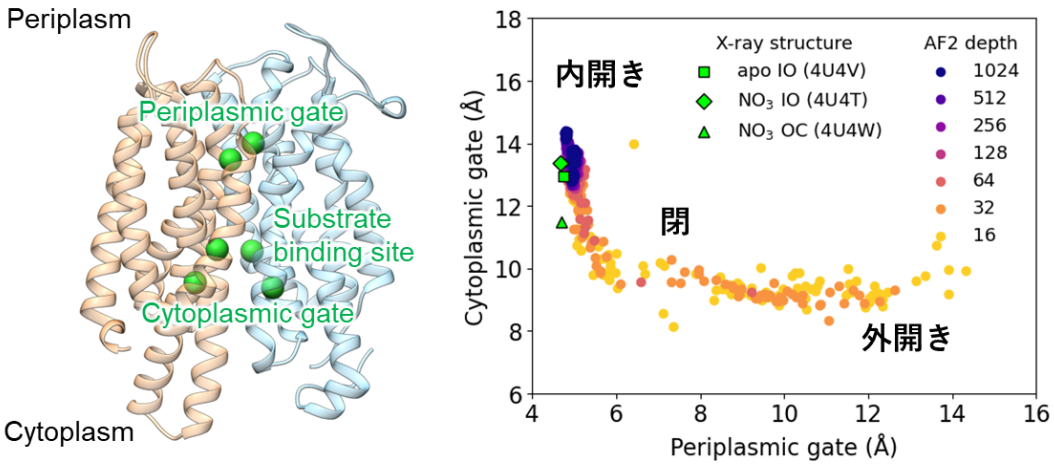


図 1：CF による硝酸・亜硝酸トランスポーター—NarK の構造予測
NarK は periplasm 側、cytoplasm 側それぞれに基質透過経路のゲートが存在し、periplasm ゲートが開いた外開き構造、cytoplasm ゲートが開いた内開き構造、両ゲートが閉じた閉構造がある。MSA を浅くする（構造予測に用いる類縁タンパク質の配列数を減らす）ことで内開き構造のみならず、閉構造、外開き構造を含む多様な構造状態が予測できるようになる。

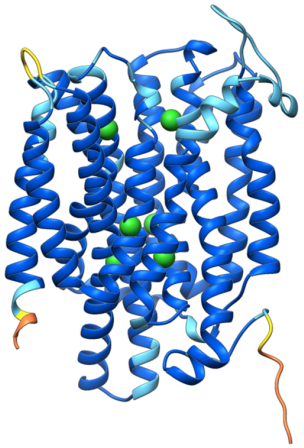


図 2：NarK の外開き予測構造
pLDDT スコアに従ってアミノ酸残基を色付けしている（青: pLDDT ≥ 90; シアン: 70 ≤ pLDDT < 90; 黄: 50 ≤ pLDDT < 70; 橙: pLDDT < 50）。

CF の構造予測にバイアスがある際、どの程度 MSA を浅くすれば良いかは決まった指針があるわけでなく、タンパク質ごとに異なる。そのため手探りで幾通りかの深さを試す必要がある。このとき MSA を毎回構築し直す必要はなく、始めに作った MSA を次回以降は利用できる。CF では a3m 形式の MSA ファイルを出力するので、MSA の深さを変えて次に予測を行う際は、先述の 2 で示した colabfold_batch コマンドにおいて input.fasta をこの a3m 形式の MSA に置き換えれば良い。

一般にはより浅い MSA を使うほど予測構造の多様性が増加する。しかし浅い MSA では統計的不確かが増大するため、信頼性の低い構造が予測される可能性があることに注意が必要である。NarK の構造予測を例にすると、試みた中で最も浅い MSA（--max-msa 8:16）では膜貫通ヘリックスが部分的にほどけた構造（図 3 上段）や、誤ったトポロジーをもつ構造（図 3 下段）が出てきてしまう。そのためスコアと構造の確認を十分に行う必要がある。

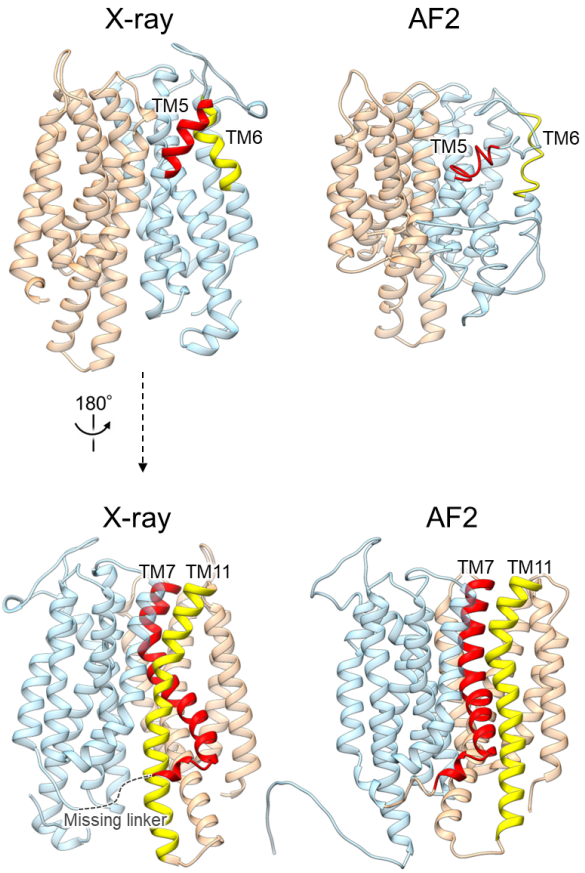


図 3：最も浅い MSA から予測された NarK 予測構造
結晶構造と比べると、膜貫通ヘリックスが部分的にほどけている構造（上段）やトポロジーが正しくない構造（下段）が予測されてしまうことがある。

4）Direct Coupling Analysis と変異導入

上記 3 では、MSA のサブサンプリングに含まれる統計的不確かによって共進化情報を乱し、複数構造状態を予測できるようにした。しかし、もし MSA 内の共進化情報を事前に知ることができれば、高い共進化傾向によってバイアスをもたらしているアミノ酸残基ペアに変異を導入することで異なる構造状態を予測きるはずである。この方法を最後に紹介したい。

MSA 内の共進化情報を抽出する方法はいくつかあるが、ここでは Direct Coupling Analysis（DCA）を用いる。DCA は、python であれば pydca を導入することで簡単に実施できる [8]。pydca は python 3.5 以降の環境であれば用いることができ、コマンドライン上で

$ pip install pydca

を実行することで導入できる（著者は python 3.5.4、pydca 1.23 を利用）。

CF の MSA は a3m 形式で出力されるが、pydca では FASTA 形式を入力とする必要がある。クエリ配列には存在しない領域（挿入領域）について、a3m 形式では類縁タンパク質のアミノ酸配列を小文字で表し、FASTA 形式ではクエリ配列側に “-” を入れる（FASTA

形式の場合、大文字と小文字の区別はない)。今回興味があるのはクエリ配列に存在する領域の中での共進化情報なので、挿入領域を考慮する必要がない。そのため例えば Linux であれば

$ tr -d a-z msa.a3m | grep -v“#” > msa.fasta

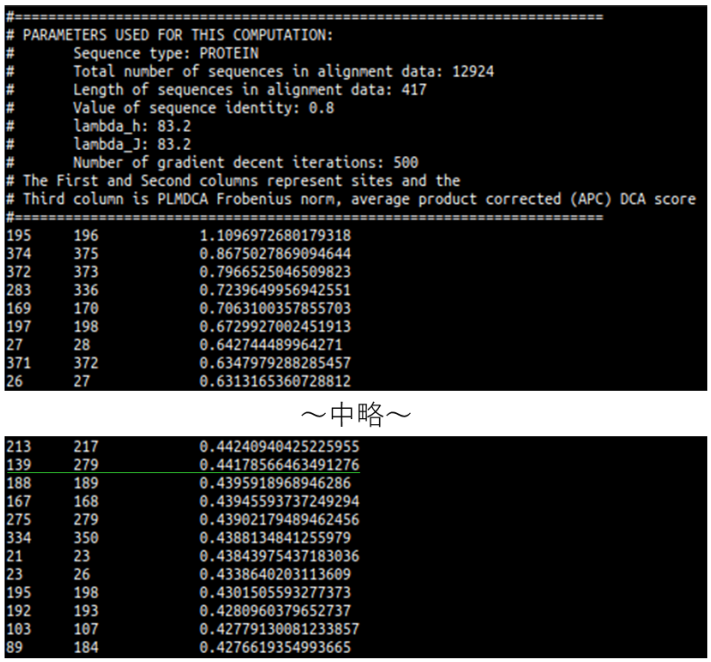
などとすれば良い（a～z の小文字を削除し、#で始まるコメント行を削除して msa.fasta に保存している)。

pydca は python スクリプト内およびコマンドライン上で使用できるが、ここではコマンドラインで実行する方法を示す。

$ plmdca compute_fn PROTEIN msa.fasta --max_iterations 500 --num_threads 6 --apc

plmdca は疑尤度最大化 DCA を実施することを意味する（平均場 DCA を用いる mfdca もあるが、一般に疑尤度最大化の方が信頼性が高いとされる)。compute_fn は、残基間の共進化スコア計算に Frobenius ノルムを用いることを指定し、PROTEIN は入力となる MSA（msa.fasta) がタンパク質のアミノ酸配列の集合であることを指定している (pydca は RNA の塩基配列も対象にできる)。オプション--max_iterations は疑尤度最大化のイテレーション回数の最大値、--num_threads はスレッド数、--apc は共進化スコアを average-product correlation によって補正することを意味する。それぞれの詳細については、文献[8,11]を参照のこと。

pydca の出力ファイルには 1、2 列目にペアとなる 2 つの残基番号、3 列目に共進化スコアが記載され（図 4)、正に大きなスコアをもつ残基ペアほど共進化傾向が強い。あとはこの結果をもとに共進化傾向の強い残基ペアに変異を導入すれば良いのだが、ここで注意すべきことがある。それは、図 4 に示すように高い共進化スコアをもつ残基ペアの多くは同じ二次構造内や同じドメイン内に存在することである。しかし例えばトランスポータータンパク質の場合には、外開き、閉、内開きの 3 構造状態でドメイン間の残基–残基コンタク



```
#=================================================================
# PARAMETERS USED FOR THIS COMPUTATION:
#       Sequence type: PROTEIN
#       Total number of sequences in alignment data: 12924
#       Length of sequences in alignment data: 417
#       Value of sequence identity: 0.8
#       lambda_h: 83.2
#       lambda_J: 83.2
#       Number of gradient decent iterations: 500
# The First and Second columns represent sites and the
# Third column is PLMDCA Frobenius norm, average product corrected (APC) DCA score
#=================================================================
195     196             1.1096972680179318
374     375             0.8675027869094644
372     373             0.7966525046509823
283     336             0.7239649956942551
169     170             0.7063100357855703
197     198             0.6729927002451913
27      28              0.642744489964271
371     372             0.6347979288285457
26      27              0.6313165360728812
～中略～
213     217             0.44240940425225955
139     279             0.44178566463491276
188     189             0.4395918968946286
167     168             0.43945593737249294
275     279             0.43902179489462456
334     350             0.4388134841255979
21      23              0.43843975437183036
23      26              0.4338640203113609
195     198             0.4301505593277373
192     193             0.4280960379652737
103     107             0.42779130081233857
89      184             0.4276619354993665
```

図 4：pydca の出力ファイル
1、2 列目はペアとなる 2 つの残基番号、3 列目が共進化スコアである。正に大きなスコアをもつアミノ酸残基ペアほど共進化傾向が強く、高スコアのものから順に記載されている。ここでは 0x1T の MSA を入力として得た結果の一部を示しており、翻訳開始メチオニンを除去した MSA を用いたため残基番号は 1 つずれている。高スコアをもつ残基ペアの大半は配列上で近く、タンパク質ドメイン内部の共進化傾向を示している。しかし一部には、緑下線で示すペア（G140–D280）のように配列上離れたドメイン間残基ペアも高スコアをもつ。

トに顕著な変化が生じる。よって、共進化スコアが高く、かつ野生型配列で予測された構造状態に特有のコンタクト残基ペアに変異を入れなければ意味がない。構造状態特有のコンタクトを特定することは容易ではないが、実験や分子シミュレーション等の情報があれば、それと共進化情報を組み合わせ変異導入先を決めると良いであろう。

例えば図 5 に示すように、シュウ酸トランスポーターOxlT では AF2 の構造予測が外開き構造に偏る。そこで外開き構造（および閉構造）に特有のコンタクト残基を MD 計算によって特定し、その中から共進化スコアが高い残基ペア G140-D280（図 4 緑下線）に変異を導入した。G140-D280 は外開き構造のとき水素結合を形成し安定化しているため、D280 を疎水性残基に変更（ここではロイシンを選択）し水素結合が形成できない変異型とした。すると、D280L 変異型では期待通り内開き構造が得られることがわかった（アラニン等、他の疎水性残基への変異でも同様の結果が得られる）。この変異では外開き構造を不安定化することで内開き構造の予測を可能にしたが、それとは異なり内開き構造を安定化するような変異を導入することもできる。図 5 に示すように、内開き構造に特有のコンタクト残基ペアをシステインに変異することで、CF はジスルフィド結合を形成させようと内開き構造を予測するようになる。

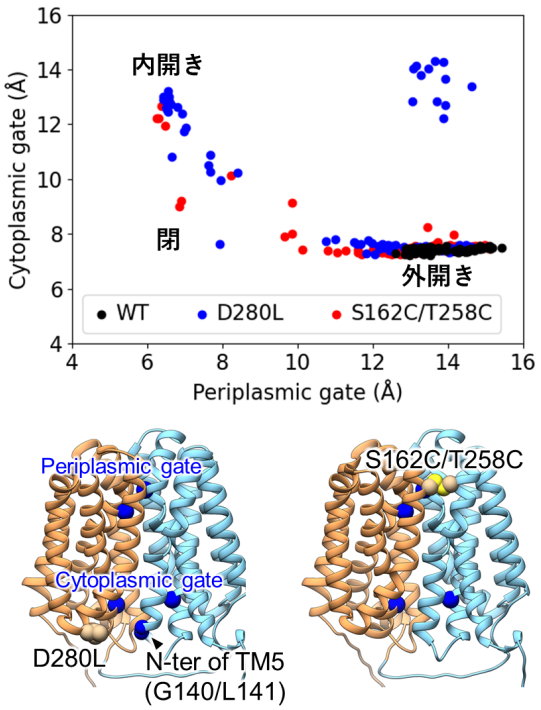


図 5：CF によるシュウ酸トランスポーターOxlT の構造予測
外開き構造でドメイン間コンタクトを形成する D280（G140/L141 とコンタクトを形成）は共進化スコアが高く、変異を導入することで内開き構造を予測するようになる。また、内開き構造でドメイン間コンタクトを形成する S162 と T258 をシステインに変異すれば、AF2 はジスルフィド結合を予測し内開き構造が得られる。

複数構造状態の探索の簡便性でいえば 3 で述べた浅い MSA をまず試みた方が良いであろう。しかし変異による方法は、単なる構造探索のみならず構造状態間の安定性を変化させる変異候補の情報を提供できるという利点がある。AF3 も登場したことにより、タンパク質構造予測は今後複合体等のより大規模で複雑な対象へと広がっていくだろう。ここで示したプロトコールがそうした研究の一助になれば幸いである。

文献

1) Jumper, J., et al., *Nature*, **596**, 583–589 (2021)
2) Abramson, J., et al., *Nature*, **630**, 493–500 (2024)
3) Sala, D., et al., *Curr. Opin. Struct. Biol.*, **81**, 10264 (2023)

4) Ohnuki, J., et al., *J. Phys. Chem. Lett.*, **15**, 725–732 (2024)
5) Ohnuki, J. & Okazaki, K.-I., *J. Phys. Chem. B*, **128,** 7530–7537 (2024)
6) 富井健太郎 編, *AlphaFold 時代の構造バイオインフォマティクス実践ガイド（実験医学別冊　最強のステップUP シリーズ）*, 羊土社 (2024)
7) Mirdita, M., et al., *Nat. Methods*, **19**, 679–682 (2022)
8) Zerihun, M. B., et al., *Bioinformatics*, **36**, 2264–2265 (2020)
9) Varadi, M., et al., *Nucleic Acids Res.*, **50**, D439–D444 (2022)
10) del Alamo, D., et al., *eLife*, **11**, e7575 (2022)
11) Ekeberg, M., et al., *Phys. Rev. E*, **87**, 012707 (2013)