**Identificarea, modificarea și compararea structurilor ADN**

**care determină proteinele**

Rareș-Gabriel Cosma, Alessia Arcaș, Codruța Burlea

*Rezumat*: Această lucrare detaliază un program bioinformatic capabil să identifice structuri de ADN generate la introducerea unui nume de proteină sau secvență de aminoacizi și poate simula mutații genetice care afectează structura proteinei rezultate. De asemenea, acesta poate compara mai multe secvențe polipeptidice. Aplicabilitatea acestui program este evidentă în laboratoare, unde cercetătorii pot experimenta, în mod eficient, modificarea genetică a proteinelor, generând proteine noi, cu structuri diferite, care pot îmbunătăți procese biologice sau pot contribui la vindecarea unor boli. Originalitatea acestei lucrări constă în selecționarea de codoni specifici fiecărui organism în procesul de translație și în funcția de comparație a proteinelor.

*Abstract*: This paper details a bioinformatics program capable of generating DNA structures by inputting a protein name or aminoacid sequence and simulating genetic mutations on them. This app is also capable of comparing multiple protein sequences. The applicability of this program can be evident in laboratory settings, where the scientists can efficiently experiment mutations resulting in new proteins, with different structures, which may improve biological processes or help cure certain diseases. The originality of this work lies in the selection of organism-specific codons during the translation process and in the protein comparison function.

1. **Introducere**

În ingineria genetică și în biologie, cercetătorii sunt adesea nevoiți să obțină ADN-ul aferent

unei proteine pentru a putea studia mutații genetice sau pentru a putea genera materiale sintetice din proteine speciale. Totuși, acest proces este unul anevoios din mai multe cauze:

1. **Există mai multe secvențe de ADN care pot codifica aceeași proteină**

Deoarece mai mulți codoni pot determina un singur aminoacid se pot obține un număr mare de combinații posibile ale ADN-ului.

1. **Organismele prezintă codoni specifici**

Anumiți codoni care detemină același aminoacid sunt mai frecvenți în anumite organisme decât în altele. Această proprietate a codului genetic ne deschide posibilitatea optimizării codonilor, îmbunătățind semnificativ performanța algoritmilor de generare a secvențelor ADN.

Alte soluții pentru optimizarea de secvențe, generarea arborilor filogenetici și prezicerea

efectelor mutațiilor genetice ar fi SIFT, AlphaMissense, PolyPhen-2, MEGA, Codon Optimization Tool. Acestea sunt folosite des în industrie și cercetare, dar sunt adesea greu de utilizat în scop didactic, sau de către începători. De asemenea, acestea pot necesita resurse de computaționale mari.

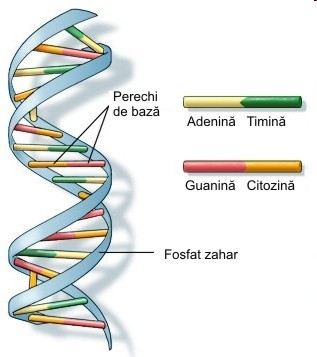
Scopul acestei lucrări este de a face mai simplă și mai eficientă analiza structurilor de ADN,

de a simula modificări în aceste structuri și de a eficientiza procesul de generare a ADN-ului prin alegerea codonilor specifici.

1. **Substrat biologic**

Pentru a putea înțelege funcționarea acestui program este necesar sa înțelegem mecanismul biologic al sintezei proteinelor pe baza informației genetice.

În nucleul fiecărei celule se află **ADN-ul** (acid dezoxiribonucleic), macromolecula informațională, polimer de nucleotide, care înmagazinează toate datele necesare pentru sinteza proteinelor. În ADN, informatia genetică este înscrisă într-o succesiune de baze azotate și este decodificată într-o secvență de **aminoacizi** în ribozomi. El este moștenit în mod egal de la mamă și de la tată și este organizat în cromozomi. Bazele azotate sunt de 4 tipuri: **Timină (T), Citozină (C), Guanină (G) și Adenină (A).** [1]

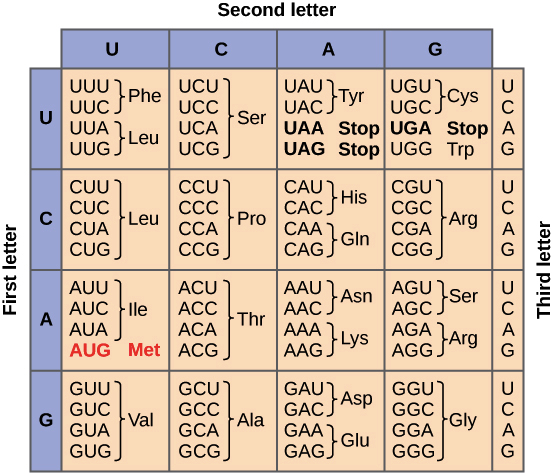


*Fig.1:Structura ADN-ului [2]*

În structura de dublu-helix a ADN-ului, bazele azotate se împerechează între ele după cum urmează: **adenină-timină** si **guanină-citozină.**

În prima etapă a sintezei proteinelor, ADN-ul este copiat pe o moleculă de **ARNm** (acid ribonucleic mesager). Acest proces se numește **transcripție**. Constă în preluarea unei jumătăți din bazele azotate prezente în ADN și înlocuirea timinei cu **uracil** (**U**).

Cea de-a doua etapă este **translația**. Informația genetică din ARNm este „citită”, decodificată, la nivelul ribozomilor și fiecarui grup de câte 3 nucleotide (**codon**) îi corespunde unui **aminoacid** adăugat lanțului proteic. [1]



*Fig.2:Structura codonilor corespunzători fiecărui aminoacid [3]*

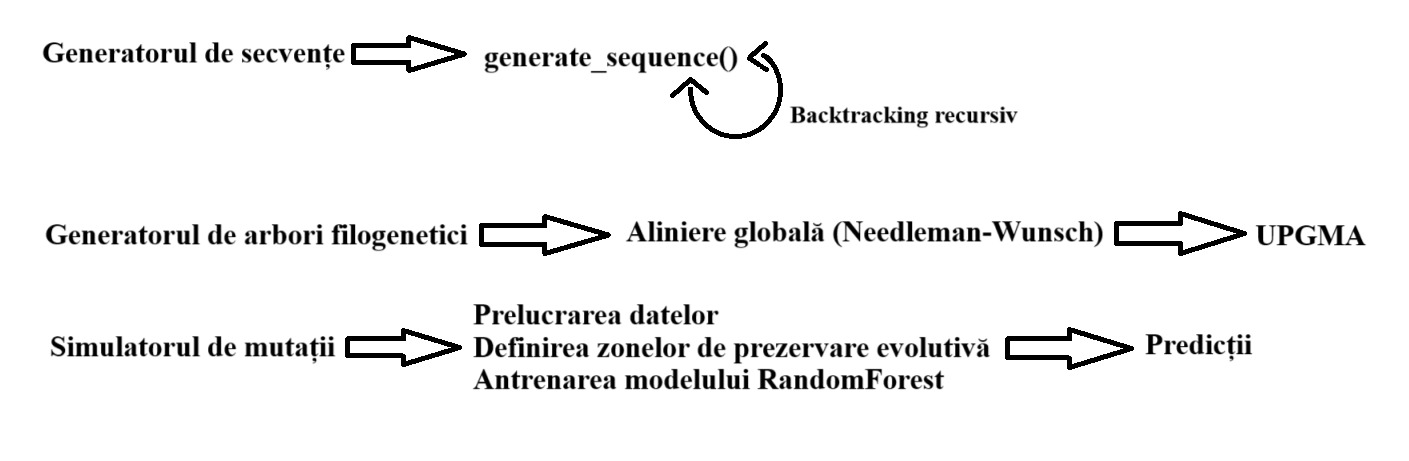
După cum se poate observa din figura de mai sus, mai mulți codoni corespund unui singur aminoacid. Această proprietate a codului genetic se numește **degenerarea codului genetic**.

1. **Generarea de secvențe**

În virtutea degenerării codului genetic, proprietatea că mai multi codoni sintetizează același aminoacid, posibilitățile combinatorice cresc exponențial. Programul nostru foloseste metoda **backtracking** pentru a genera cât mai multe combinații de ADN. Considerând că pentru o proteină cu un număr relativ mic de aminoacizi, insulina, sunt posibile aproximativ combinații, este mai inteligent să încercăm să generăm combinații mai relevante decât să generăm toate combinațiile.

Așadar, programul încearcă să optimizeze folosirea codonilor astfel încât primele combinații de ADN să fie cele mai ușor de sintetizat, scăzând performanța secvențelor pe măsură ce se generează tot mai multe. Această caracteristică a programului funcționează prin preluarea numelui proteinei și numele organismului care necesită acea proteină (secvențele de aminoacizi pot prezenta diferențe de la un organism la altul, chiar dacă sintetizează aceeasi proteină; ne este absolut necesar să preluăm proteina de la organismul corect atunci când testăm diferențele dintre proteine sau simulăm mutații), apoi preluarea codului genetic pentru alte proteine din cadrul aceluiași organism, calcularea frecvenței fiecărui codon în parte, iar la final, prioritizarea codonilor mai frecvenți în algoritmul de generare.

Am ales să folosim **limbajul C** pentru generarea eficientă a secvențelor, integrat în **limbajul Python** printr-o bibliotecă comună (.dll), pentru accesul ușor la baze de date și implementarea unei interfețe video native, funcțională pe orice sistem de operare modern. [4,6,7,9,11]



*Fig.3:Structura programului – imagine proprie*

1. **Simularea mutațiilor genetice**

O altă parte importantă a programului nostru este simulatorul de mutații genetice. Pentru a înțelege mai bine acest modul ne este necesar să înțelegem cum funcționează mutațiile genetice. Mutațiile genetice sunt schimbări în ADN, ARNm sau chiar și după ce o proteină a fost sintetizată. Din cauza complexității anumitor mutații, noi simulăm doar mutațiile care au loc la nivelul ADN-ului si ARNm-ului. Dintre acestea, programul poate simula mutațiile **punctiforme (substituțiile**) și **inserțiile/delețiile**.

Mutațiile punctiforme sunt mutații care schimbă doar o bază azotată în secvența ADN. Acestea pot fi **sinonime** (nu se schimbă aminoacidul, decât codonul; de obicei acestea nu au impact asupra funcției proteinei), **nonsinonime** (se schimbă codonul, iar odată cu el si aminoacidul; acestea pot avea impact asupra funcției proteinei) sau **nonsens** (adică prin substituție se formează un codon stop prematur; efect devastator la nivelul proteinei). Cele nonsinonime pot fi **conservative/neconservative** (după păstrarea caracteristicilor aminoacidului schimbat).

Inserțiile și delețiile sunt mutații în care se inserează sau se șterg un anumit număr de baze (acesta nu este neapărat să fie multiplu de 3: ACT-GAC-ACC prin insertie dupa primul codon a două baze azotate AT secvența devine ACT-ATG-ACA-CC), în cazul în care acest număr nu este multiplu de 3 mutația se numeste **frameshift**, adică mută cadranul cu totul. De obicei, acestea au efect devastator asupra proteinelor. [5,8,10]

Odată ce am simulat aceste mutații, programul afișează posibilitatea ca mutația simulată să fie una cu efect negativ, neutru sau pozitiv. Acest lucru se realizează diferit, în funcție de tipul de mutație.

În cazul mutațiilor punctiforme, putem utiliza mai mulți indicatori care ne arată detalii despre frecvența unei mutații în evoluție și despre impactul ei asupra funcției proteinei. Programul nostru folosește acești indicatori:

**4.1 BLOSUM (BLOck SUbstitution Matrix)**

BLOSUM [16] este o matrice de substituție care ne arată frecvența anumitor mutații în evoluție. Noi putem extrage o simplă aproximare a impactului mutației asupra proteinei în cauză considerând că mutațiile mai frecvente nu au un impact negativ asupra proteinelor, fiind tolerate, pe când cele mai rare duc la sintetizarea unei proteine nefuncționale, deci, fiind evitate. Matricea BLOSUM este calculată după această formulă:

(4.1)

unde este probablitatea ca aminoacizii i și j să fie schimbați într-o secvență,

și sunt frecvențele aminoacizilor în toate secvențele

iar este un coeficient de mărire (ajută la transformarea valorilor în valori ușor de procesat)

Bineînteles că acestă metodă nu este una cu o acuratețe impresionantă așa că am decis să folosim și alți indicatori.

**4.2 Distanța Grantham**

Distanța Grantham [12] este un indicator chimic, un indicator al impactului asupra **funcției** proteinei. Acesta are 3 variabile: compoziția, polaritatea și volumul molecular. Acest indicator este calculat astfel:

(4.2)

unde c=compoziție, p=polaritate și v=volum

sunt constantele pătratelor inverselor distanțelor medii pentru fiecare proprietate, respectiv egale cu 1.833, 0.1018 și 0.000399

**4.3 Indexul lui Sneath**

Indexul lui Sneath [15] este un indicator care ia în considerare 134 de factori structurali și funcționali pentru a compara aminoacizii într-un mod cât mai precis.

**4.4 Scorul mutației**

Pentru a putea evalua impactul unei mutații asupra proteinei am compus o ecuație care folosește toți indicatorii definiți mai sus:

(4.3)

unde S,B,D reprezintă valorile indicatorilor de mai sus pentru aminoacizii i și j

iar c este o constantă egală cu 3\*(!) (pentru a evita cazul în care = 0 => Scor = 0)

Pentru o ușoară manipulare a valorilor scorului, am decis sa îl normalizăm:

unde = -1086 și = 84.84

Pentru a putea calcula o limită care delimitează categoriile de efect asupra funcției proteinei am decis să ne folosim de date reale, experimentale, pentru a decide în ce interval al scorului mutația are un anumit efect.

Totuși, încercând să găsim date experimentale, am întâmpinat o problemă. Experimentele care generează datele sunt foarte costisitoare deci ele au fost efectuate doar pe secvențe care pot produce boli, iar și aceste secvențe în condiții de mediu foarte specifice. Experimentele de acest tip se numesc experimente **MAVE**, și se caracterizează prin faptul că printr-un singur experiment se poate afla efectul oricărei variante mutante, în urma unei mutații punctiforme.

Din moment ce ne dorim ca programul nostru să fie cât mai general și să poată fi folosit în cât mai multe cazuri, am decis sa folosim baza de date a modelului de IA **AlphaMissense**. Acest model de inteligență artificială este derivat din modelul **AlphaFold**, câștigător al premiului Nobel. Acesta este capabil să genereze structurile 3D ale proteinelor, antrenat pe baza de date **PDB** (Protein Data Base), cu o acuratețe de peste 90%. AlphaMissense este capabil sa prezică efectul mutațiilor punctiforme, prin analiza diferențelor dintre structurile 3D ale celor 2 secvențe, cu o acuratețe de peste 90% (testată pe baza de date ClinVar, pe mutații cunoscute asupra genomului uman). 11% din prezicerile acestui model sunt totuși nedeterminate din mai multe cauze. [14]

Noi preluăm datele din baza de date AlphaMissense, **determinăm regiunile conservate evolutiv**, calculăm scorul fiecărei variante existente în datele disponibile, și, luând în considerare poziția mutației corespunzătoare fiecărei variante calculate, scorul acesteia și predicția AlphaMissense asupra efectului mutației, putem antrena un model de inteligență artificială, mai exact un model de **RandomForest (pădure de arbori decizionali)**, pentru a clasifica cele două categorii de impact asupra funcției proteinei, obținând, astfel, o limită non-liniară între cele 2 categorii. În practică, programul nostru calculează un număr care desparte mutațiile cu efect negativ de cele cu efect pozitiv, bazat pe predicția AlphaMissense și pe poziția în secvență. Modelul nostru are o acuratețe de 94.98% și un scor F1 de 96%.

Definirea zonelor de conservare evolutivă se face în complexitate O(n), prin delimitarea zonelor în care media impactului mutațiilor este pozitivă, respectiv negativă. Zonele cu media impactului pozitivă pot fi considerate zone cu o conservare evolutivă scăzută deoarece majoritatea mutațiilor sunt pozitive, deci tolerate. Analog, zonele cu un impact mediu negativ pot fi considerate zone cu o conservare evolutivă ridicată deoarece majoritatea mutațiilor nu sunt acceptate. Programul realizează acest calcul prin iterarea tuturor pozițiilor din secvența dorită, calculează o medie a impactului folosid datele AlphaMissense pentru acea poziție, iar în final, definește o zonă de conservare evolutivă ca fiind o succesiune de poziții cu o medie similară.

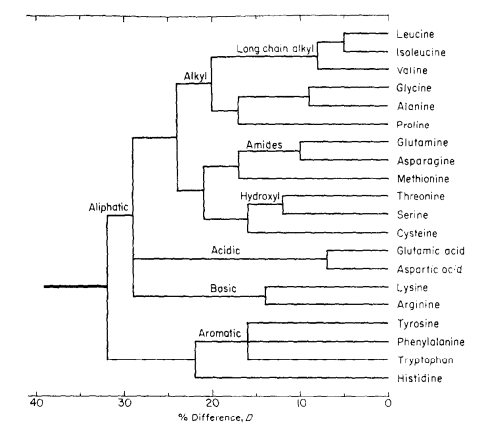
Considerăm ca abordarea noastră ar fi mai precisă decât AlphaMissense, la modul general, deoarece abordarea noastră ar putea decide categoria celor 11% dintre preziceri care rămân nerepartizate, prin adăugarea factorilor fizico-chimici în procesul de decizie. Un alt beneficiu al programului nostru ar fi eficiența. În cazul în care o mutație nu se află în baza de date AlphaMissense aceasta trebuie calculată utilizând modelul, fiind nevoie de o putere de calcul destul de mare, indisponibilă în laboratoare mici, sau în clinici. Prezicerea noastră rulează în complexitate redusă deoarece se bazează pe date precalculate și pe calcule matematice.

Pentru a testa totuși această funcție, noi ne-am limitat doar la mutațiile asupra insulinei. Pentru o generalizare cat mai bună a programului nostru, limitele ar trebui calculate pe grupe de proteine, realizate de un specialist.

În cazul inserțiilor/delețiilor, procesul de evaluare al impactului unei mutații se complică semnificativ, secvența modificată având aminoacizi noi sau lipsă. Pentru început, vom trunchia instanțele în care secvența modificată se încheie prematur cu un codon stop, sau suferă o mutație de tip frameshift, deoarece acestea sunt devastatoare pentru funcția proteinei, obținând scor minim fără a fi prelucrate. În cazurile în care mutația nu implică un codon stop prematur sau o mutare de cadran (frameshift), o să utilizăm un algoritm de aliniere a secvențelor, calculând probabilistic posibilitatea ca mutația să fie dăunătoare sau nu. Algoritmul folosit este **algoritmul de aliniere globală Needleman-Wunsch** [13]. Acesta constă în construirea unei matrici între secventa inițială și cea modificată, aplicarea unui scor fiecărei asemanări sau diferențe, fiind luate în considerare și spațiile goale din secvențe, iar apoi găsirea drumului cu cel mai mare scor. Această metodă nu este perfectă, ea în sine sintetizând cât de apropiate sunt 2 secvențe, nu cât de periculoasă este o mutație. Putem totuși estima acest lucru, luând în considerare că, de obicei, secvențele asemanătoare cu cea originală au sanse mai mari să nu prezinte schimbări majore în funcționalitate, pe când cele foarte diferite, au de obicei diferențe mari.

1. **Compararea proteinelor**

O altă funcționalitate importantă pentru oamenii de știință este posibilitatea de a vedea relațiile evoluționare dintre organisme sau variante ale unei proteine. Acest lucru este realizat în programul nostru prin generarea **arborilor filogenetici**. Arborii filogenetici sunt arbori care ordonează organisme sau variante ale unei proteine după relațiile evoluționare dintre ele.



*Fig.4:Arbore filogenetic pentru cei 20 de aminoacizi, bazat pe indexul lui Sneath [15]*

Când vom interpreta acești arbori trebuie să luăm în considerare că: nodurile frunză numite și „Operațional Taxonomic Units” sunt de fapt speciile/secvențele analizate, nodurile interne sunt specii ancestrale (de multe ori ipotetice), iar entitățile care au un strămoș comun vor fi în aceeași subarbore.

Acești arbori sunt în general împărțiți în 3 categorii: arbori **scalați**, arbori **aditivi** și arbori **nescalați**. În cazul arborilor nescalați, ramurile nu poartă informație, iar în cazul celor scalați și aditivi ramurile poartă informație referitoare la timpul necesar pentru o anumită mutație respectiv diferențele dintre 2 entități.

Din moment ce nu se pot construi totdeauna arbori aditivi pentru orice set de entități, vom folosi metoda **UPGMA** (Unweighted Pair Group Method with Arithmetic Mean) pentru a genera arbori scalați. Această metodă este similară cu algoritmii aglomerativi de clustering. Ea se caracterizează prin calcularea distanțelor dintre toate clusterele (average link) și asocierea unei înălțimi fiecărui nod acest lucru fiind foarte important după unirea a două clustere, deoarece dictează lungimea muchiilor dintre cele 2 noduri. Această metodă nu este perfectă deoarece se bazează pe ipoteza unui ceas molecular constant, adică se bazează pe faptul că rata de acumulare a mutațiilor este constantă.[13]

1. **Eficiența programului**

În cazul generatorului de secvențe ADN, putem analiza eficiența în materie de timp și spațiu.

Pentru 1.000.000 de secvențe de insulină generate programul utilizează 647MB în memorie. Totuși, din natura datelor, mai exact din faptul că secvențele nu diferă foarte mult ne putem da seama că **algoritmii de compresie** ar putea reduce semnificativ dimensiunea fișierelor produse de programul nostru. După testarea algoritmilor pe aceleași 1.000.000 de secvențe am ajuns la această histogramă:

*Fig.5:Histogramă reprezentativă pentru diferiții algoritmi de compresie – imagine proprie*

Când vine vorba de timp, nu putem analiza concret complexitatea sau timpul deoarece algoritmul este unul de backtracking, dar pe cele 1.000.000 de secvențe generate, timpul de execuție a fost de aproximativ 22s.

1. **Concluzie**

În viitor, acest program ar putea fi îmbunătățit prin adăugarea optimizării codonilor prin inteligența artificială, evaluarea mutațiilor dupa structura tridimensională a proteinelor și după secțiunile codificatoare care se prezervă în evolutie pentru grupuri de proteine, nu doar pentru insulină și o interfață web ușor accesibilă oricui. De altfel, pentru generalizarea programului, modelul RandomForest ar trebui antrenat pe un set de mii de proteine, grupate anterior pe clase de către un specialist.

În concluzie, acest program este o implementare software eficientă a proceselor întâlnite oriunde în natură, și poate fi de ajutor atât studenților și elevilor, ca material didactic util, dar și oamenilor de știință pentru evaluarea ușoară și eficientă a mutațiilor, compararea proteinelor sau generarea secvențelor ADN.

**8 Bibliografie**

1. ARINIȘ Ioana, MIHAIL Aurora, *BIOLOGIE – Manual pentru clasa a 9-a*, Colecția „LiceALL 2000”
2. Synevo, *Figura 1:* <https://www.synevo.ro/ziua-internationala-adn/>
3. Khan Academy, *Figura 2:* <https://www.khanacademy.org/science/ap-biology/gene-expression-and-regulation/translation/a/the-genetic-code-discovery-and-properties>
4. GeeksforGeeks: *Python* *Tkinter:* <https://www.geeksforgeeks.org/python-gui-tkinter/>
5. NCBI: *A new and updated resource for codon usage tables:* <https://pmc.ncbi.nlm.nih.gov/articles/PMC5581930/>
6. Integrare C/Python: <https://realpython.com/python-bindings-overview/>
7. Biblioteca ctypes: <https://stephenscotttucker.medium.com/interfacing-python-with-c-using-ctypes-classes-and-arrays-42534d562ce7>

[8] Zarnea, G., Popescu, O.V. – *Dicționar de microbiologie generală și biologie moleculară, Editura Academiei Române 2011*

[9] NCBI *–* [*National Center for Biotechnology Information*](https://www.ncbi.nlm.nih.gov/)

[10] *LibreTexts – Biology:*

[https://bio.libretexts.org/Bookshelves/Introductory\_and\_General\_Biology/Introductory\_Biol ogy\_(CK-12)/04%3A\_Molecular\_Biology/4.08%3A\_Mutation\_Types](https://bio.libretexts.org/Bookshelves/Introductory_and_General_Biology/Introductory_Biol%09ogy_(CK-12)/04%3A_Molecular_Biology/4.08%3A_Mutation_Types)

[11] Uniprot: [https://www.uniprot.org](https://www.uniprot.org/)

[12] Grantham, R.  "Amino acid difference formula to help explain protein evolution". *Science*.

[13] Curs de filogenetică, Daniela Zaharie: <https://staff.fmi.uvt.ro/~daniela.zaharie/bioinfo2016/curs/cu>

[rs9/bioinfo2016\_curs9.pdf](https://staff.fmi.uvt.ro/~daniela.zaharie/bioinfo2016/curs/cu)

[14] Cheng, J., Novati, G., Pan, J., Bycroft, C., Žemgulytė, A., Applebaum, T., Pritzel, A., Wong, L. H.,

Zielinski, M., Sargeant, T., Schneider, R. G., Senior, A. W., Jumper, J., Hassabis, D., Kohli, P., &

Avsec, Ž. „Accurate proteome-wide missense variant effect prediction with AlphaMissense”. Science.

[15] Sneath, P. H. A. (1966) "Relations between chemical structure and biological activity in peptides."

*Journal of Theoretical Biology*

[16] Biopython: <https://biopython.org/>

Rareș-Gabriel COSMA

Colegiul Național ,,Octavian Goga”

Matematică-Informatică intensiv Informatică

Sibiu, România

E-mail: cosma.rares.gabriel@gmail.com

Alessia ARCAȘ

Colegiul Național ,,Octavian Goga”

Matematică-Informatică intensiv Informatică

Sibiu, România

E-mail: arcasalessia@gmail.com

Prof. Codruța BURLEA

Colegiul Național ,,Octavian Goga”

Biologie

Sibiu, România

Email: codruta.burlea@cnogsibiu.onmicrosoft.com