**PowerPoint notes**

**Introduction:**

Variant classification

Workflow

Ποιο είναι το πρόβλημα;

Θεωρία για τα κριτήρια

**Methods:**

Βιβλιοθήκες που χρησιμοποιήθηκαν

Κώδικας – εργαλεία που χρησιμοποιήθηκαν

Databases που χρησιμοποιήθηκαν

**Results**

**Conclusions – προεκτάσεις (π.χ. επιπλέον πίνακες)**

Screenshots από παραδείγματα, τι δίνω ως ερώτημα, τι παίρνω ως απάντηση

Live demonstration

Εστιάζουμε στα κριτήρια

Γραφικό εργαλείο για τη βάση(π.χ. pgadmin) – οπτικοποίηση της βάσης

Παράδειγμα από entry βάσης

Pgadmin – postres dbs

**Introduction**

The ACMG 2015 criteria refer to the standards and guidelines established by the American College of Medical Genetics and Genomics (ACMG) and the Association for Molecular Pathology (AMP) in 2015 for the interpretation of sequence variants. These guidelines provide a structured framework to classify genetic variants based on evidence into 5 tiers: pathogenic, likely pathogenic, uncertain significance, likely benign and benign.

There is a total of 28 criteria: 16 pathogenic and 12 benign

We will focus on only 4 of them. More specifically, PS1(strong evidence of pathogenicity), PM5(moderate evidence of pathogenicity), PP5(supporting evidence of pathogenicity) and BP6(supporting evidence of the variant being benign).

In summary, the ACMG 2015 criteria provide a comprehensive, evidence-based framework to guide clinical laboratories and genetics professionals in interpreting the clinical significance of genetic variants, facilitating standardized decision-making to support patient care and research.

Το κύριο πρόβλημα που θέλαμε να λύσουμε ήταν να αξιολογήσουμε διάφορες μεταλλαγές με βάση τα ACMG 2015 κριτήρια. Για τον σκοπό αυτό χρησιμοποιήθηκε μόνο ένας μικρός αριθμός κριτηρίων(PS1, PM5, PP5 και BP6),καθώς ήταν μικρής κλίμακας το project. Με αυτό τον τρόπο, μπορεί να γίνει μία πρώτη αξιολόγηση για το κατά πόσο μια μεταλλαγή είναι παθογόνος ή όχι. Το πλεονέκτημα που φέρει αυτού του είδους η αξιολόγηση είναι ότι δίνει την πιθανότητα για μία νέα μεταλλαγή, για την οποία δεν υπάρχει προηγούμενη καταγραφή στην ClinVar, να είναι παθογόνος.

**Methods**

Για να φτάσουμε όμως στο σημείο της αξιολόγησης και της κατηγοριοποίησης μιας μεταλλαγής με βάση τα στοιχεία που υπάρχουν διαθέσιμα στην ClinVar, έπρεπε να χτιστεί και το αντίστοιχο υπόβαθρο. Γι’ αυτό, δημιουργήθηκε αρχικά μια βάση δεδομένων, η οποία παίρνει δεδομένα από την ClinVar, και δημιουργεί έναν πίνακα που περιέχει τα απαραίτητα για την αξιολόγηση στοιχεία και επιπλέον δεδομένα που θα ήταν χρήσιμα για τον χρήστη. Μια καταγραφή της βάσης δεδομένων μοιάζει κάπως έτσι(screenshot entry db). Ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η Python και για τη δημιουργία της βάσης χρησιμοποιήθηκε η PostgreSQL. Βιβλιοθήκες της Python που χρησιμοποιήθηκαν είναι¨

* psycopg2
* pandas
* urllib
* re
* traceback
* os
* typing
* subprocess
* collections

Επικεντρωθήκαμε κυρίως στις μεταλλαγές ενός μόνο γονιδίου, του TP53, λόγω του τεράστιου όγκου δεδομένων που υπάρχουν στην ClinVar. TP53 is an extremely important gene because it encodes the p53 protein, a critical tumor suppressor often called the “guardian of the genome”. p53 regulates key cellular processes such as cell cycle arrest, DNA repair, apoptosis (programmed cell death), senescence and metabolism to maintain genomic stability and prevent cancer development. Mutations in TP53 are found in about 50% of human cancers and lead to loss of this protective function, allowing uncontrolled cell proliferation and tumor progression. Additionally, many TP53 mutations give the protein oncogenic gain-of-function properties that further promote malignancy, metastasis, and resistance to chemotherapy and radiation. Because of its central role in guarding against cancer, TP53 mutations are one of the most common and consequential genetic alterations in cancer biology, making it a major focus for research and potential targeted therapies.

Li-Fraumeni Syndrome – germline mutations of TP53 (κληρονομικός καρκίνος)

**Workflow**

1. ClinVar (raw) data file (variant\_summary.txt)
2. Python Script (db\_no\_sub.py)
   * Unzips/filters variant\_summary.txt from ClinVar
   * Preprocess the data
   * Adds ACMG logic
3. PostgreSQL Database(clinvar\_db)
   * Table: gene\_variants
4. API – FastAPI(api.py)
   * Reads gene\_variants
   * Serves JSON output
5. User/Client(e.g. browser, app)

Πώς λειτουργεί η βάση;

Η βάση δεδομένων χρησιμοποιεί και στηρίζεται στο variant\_summary.txt της ClinVar. Αφού κατέβει το αρχείο, επιλέγονται κάποιες από τις στήλες του αρχείου, οι οποίες θα δημιουργήσουν τον βασικό πίνακα της βάσης, gene\_variants. Ακολουθεί επεξεργασία των δεδομένων με σκοπό να προστεθούν στήλες που δεν υπάρχουν στο variant\_summary.txt, αλλά είναι χρήσιμες για την αξιολόγηση των variants. Στις νέες στήλες περιλαμβάνονται οι molecular consequence, hgvs\_c, hgvs\_p, protein\_pos κλπ. Στη συνέχεια, γίνεται δημιουργία κατάλληλων λεξικών για τις παθογόνες μεταλλαγές που υπάρχουν καταγεγραμμένες στην ClinVar, καθώς και για τις αμινοξικές θέσεις, οι οποίες φέρουν παθογόνα μεταλλάγματα.

Το API δέχεται ερωτήματα του χρήστη, τα οποία κυρίως αφορούν μία συγκεκριμένη μεταλλαγή με βάση το hgvs\_c και το hgvs\_p. Από εκεί το API συνδέεται με τη βάση, όπου υπάρχει ο αλγόριθμος για την αξιολόγηση μεταλλαγμάτων με βάση τα ACMG 2015 κριτήρια και απαντάει σε μορφή JSON.

**Python libraries**

*psycopg2:* connect Python applications with PostgreSQL databases, allowing us to execute SQL queries, manage database transactions, and manipulate data efficiently within Python code. It acts as a database adapter that complies with the Python DB-API 2.0 specification, enabling robust, secure and high-performance interaction with PostgreSQL.

*pandas:* efficient data manipulation and analysis. Pandas provides powerful, flexible data structures – most notably DataFrames and Series – that allow you to easily load, clean, transform, analyze and visualize structured data such as tables, time series and statistical datasets.

*urllib:* work with URLs and web-related tasks. It is a built-in standard library that allows you to open and read URLs, make HTTP requests (such as GET and POST), and interact with web resources without needing extra packages. It handles fetching data from the internet, parsing URLs, encoding query parameters, managing HTTP headers and dealing with errors or redirects.

*re:* work with regular expressions, which are powerful tools for searching, matching and manipulating text based on specific patterns. The re module lets you define patterns to find, extract, or replace substrings within larger strings efficiently.

*traceback:* extract, format and print stack traces of a program, especially when exceptions occur. It provides a standardized way to access detailed information about the call stack at the point where an error was raised, allowing developers to see the sequence of function calls that led to the error, the exact file and line number, and the error type with a message. This makes debugging and error diagnosis much easier by giving a clear, structured report of what went wrong and where in the code. The traceback module is more flexible than the default interpreter output for tracebacks. It offers tools to customize how tracebacks are displayed or logged, capture them for later use and work with them programmatically, which can be useful in advanced debugging, error logging, or building custom error handlers. In summary, the traceback library helps you understand and handle exceptions better by providing access to detailed, formatted stack traces in your Python programs.

*os:* interact with the operating system in a portable and convenient way. It provides functions to perform a wide range of operating system related tasks such as:

1. File and directory management
2. Process management
3. Environment variables
4. File permissions and metadata

The os module abstracts differences between operating systems, making Python code that manipulates the filesystem or environment compatible across platforms like Windows, macOS and Linux.

*typing:* provide type hints or type annotations in the code, which specify the expected data types of variables, function parameters and return values. This helps improve code clarity, maintainability and tooling support by making the developers intent explicit about what types are expected. The typing library provides a standardized way to add type information to Python code, enhancing clarity, error detection and development tools support.

*subprocess:* create and manage new child processes, allowing your Python code to run external programs or system commands. It enables you to start other applications, capture their output (standard output and error), send input to them and get their exit status codes for success or failure. This makes it possible to interact programmatically with the operating system or other software outside the Python process. Subprocess is essential when you want to launch and control external processes and system commands directly from Python scripts, enabling automation, workflow integration and interaction with other software or shell utilities.

*collections:* it provides a set of specialized container datatypes that extend and improve the capabilities of Python’s built-in containers like list, dict, set and tuple. These specialized data structures help write cleaner, more efficient and more expressive code by offering alternatives tailored for common programming needs. Overall, the collections module helps you solve common data handling problems more effectively and write Pythonic, maintainable programs with better performance and clarity compared to using basic built-in types alone. In brief, collections is essential when you want specialized, high-performance alternatives to the standard containers to optimize and clarify your code.