



# KATH: A No-Coding Data Processing Aid for Genetic Researchers

Kajus Černiauskas, Dainius Kirsnauskas, Paulius Preikša, Gabrielius Salyga, Nojus Sajauskas, Jūnius Vaitkus, Gerda Žemaitaitė, Kazimieras Bagdonas

Kaunas University of Technology  
2024

# Motivation

The following issues have been identified in discussion with researchers from Harvard University:

1. Complexity of the software tools;
2. Lack of programming skills among scientist;
3. Programmers lack domain knowledge in genetics;
4. There are few bioinformaticians;
5. Even high-level specialists manually refactor and analyse data;

# DNA analysis tools

- Existing examples of workflow automation systems:
  - NextFlow framework for creating custom workflows
  - SnakeMake for processing sequencing data with pipelines

**Table 1**  
DNA tools comparison

Tool	Purpose	Can be run locally
SpliceAI	SpliceAI is a deep-learning-based tool used to score variants.	Yes
REVEL	REVEL is an ensemble method for predicting the pathogenicity of missense variants based on a combination of scores from 13 individual tools.	Yes
CADD	CADD is a tool for scoring the deleteriousness of single nucleotide variants, multi-nucleotide substitutions as well as insertion/deletions variants in the human genome.	No
Pangolin	Pangolin is a deep-learning based method for predicting splice site strengths.	Yes
Eve	EVE is a model for the prediction of clinical significance of human variants based on sequences of diverse organisms across evolution.	No
Metadome	MetaDome analyses the mutation tolerance at each position in a human protein.	No
AlphaMissens	AI model that predicts whether genetic mutations in proteins are likely to be harmless or disease-causing.	Yes

# Variation of gene analysis example

Enter a variant below to see its [SpliceAI](#) and [Pangolin](#) scores

[\[more details\]](#)

Examples (on hg38):

chr8-140300616-T-G

chr8 140300616 T G

NM\_001089.3(ABCA3):c.875A>T (p.Glu292Val)

[\[show more examples\]](#)

NM\_001089.3(ABCA3):c.875A>T (p.Glu292Val)

Genome version: ☐ hg19 ☒ hg38

Max distance:  [?](#) ☐ masked scores [?](#) ☐ REF & ALT scores [?](#)

Submit

related web tools:

[liftover](#): for variants/positions/intervals (hg19 <=> hg38 <=> T2T)

[CMA search](#): search OMIM by interval, gene or phenotype

[TGG Viewer](#): igv.js-based web viewer for public reference tracks and, optionally, private data in Google Storage buckets

March 7, 2024

- added warning for insertion variants with delta scores  $\geq 0.2$  and position = 0bp saying that they may be difficult to interpret due to [issue #67](#). Thanks to @SophieCandille for the issue report and example variant: [2:47790924 C>CAGTTG](#)

- moved server to Google Cloud Run to better support higher usage

[\[show older updates\]](#)

SpliceAI scores: [?](#)

Variant	Gene	<input type="checkbox"/> = MANE Select transcript	<input type="checkbox"/> = non-coding transcript	$\Delta$ type	$\Delta$ score <a href="#">?</a>	position <a href="#">?</a>
NM_001089.3(ABCA3):c.875A>T (p.Glu292Val) ⇒ 16:2317763 T>A <a href="#">UCSC</a> , <a href="#">gnomAD</a>	ABCA3 (ENSG00000167972.14/ENST00000301732.10/NM_001089.3)  protein coding <a href="#">MANE Select transcript</a> (minus strand) <a href="#">OMIM</a> , <a href="#">GTEx</a> , <a href="#">gnomAD</a> , <a href="#">ClinGen</a> , <a href="#">Ensembl</a> , <a href="#">Decipher</a> , <a href="#">GeneCards</a>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Acceptor Loss	0.21	106 bp
				Donor Loss	0.00	497 bp
				Acceptor Gain	0.01	1 bp
				Donor Gain	0.04	128 bp

MANE Select Transcript

or

All Transcripts

# Proliferation of LLM models

**Table 2**

LLM models for user assistance

Model	Release Date	Params (B)	Context Length
OLMo	2024/02	1,7	2048
Gemma	2024/02	2-7	8192
SOLAR	2023/12	10.7	4096
phi-2	2023/12	2.7	2048
Zephyr	2023/11	7	8192

**Table 3**

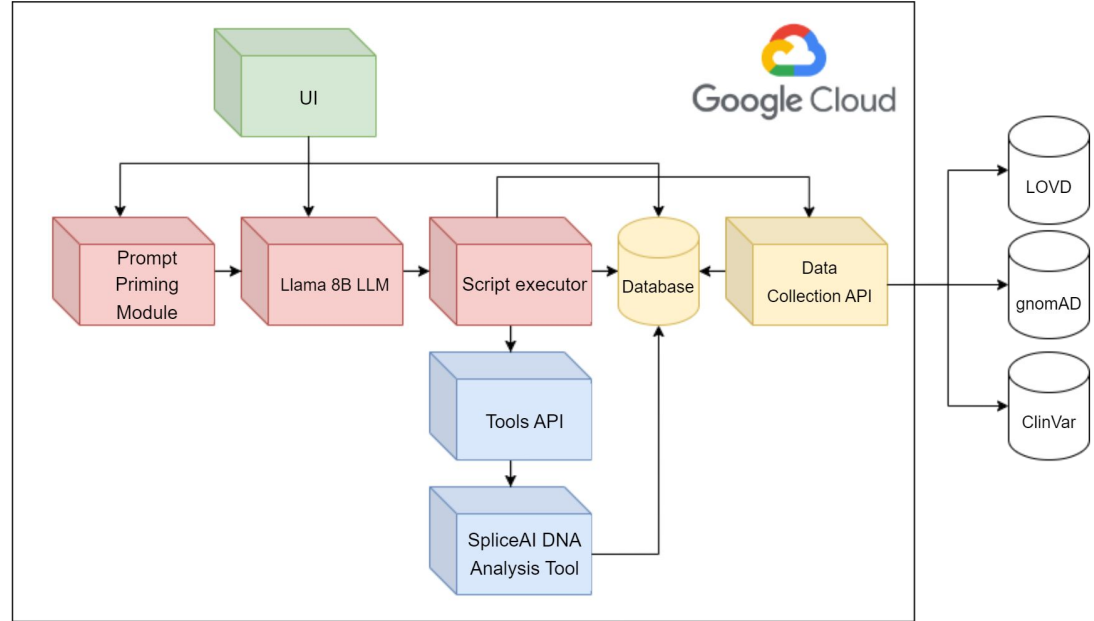
LLM models for script generation

LLM models for script generation

Language Model	Release date	Parameters (B)	Context Length
Dolphin	Nov-23	7	32768
DeciCoder-1B	Aug-23	1.1	2048
CodeGen2.5	Jul-23	7	2048
XGen-7B	Jun-23	7	8192
StarCoder	May-23	1.1-15	8192
MPT-7B-Instruct	May-23	59k (Samples)	NA
databricks-dolly-15k	Apr-23	15k (Samples)	NA
OIG	Mar-23	44,000k (Samples)	NA
StarChat Alpha	May-23	16	8192

# KATH (TRL - 4) prototype

- UI has been developed as a web service
- Llama3 8B LLM model has been employed in the KATH system
- Data collection and refactoring modules are implemented
- System is deployed on Google Cloud



# KATH system user interface

The screenshot displays the KATH system user interface. On the left, a sidebar contains a 'Workspace' tree with a hierarchical structure: 'Root' (expanded) containing 'Parent1', 'Parent2', 'Parent3', 'Child1' (selected), 'Child2' (expanded), and 'Child3'. 'Child2' contains files 'File1.format' through 'File9.format'. A context menu is open over 'Child1', showing options: 'New Folder...', 'New File...', 'View .csv file', 'Rename', and 'Delete'. The main area has two tabs: 'Functionality' (active) and 'History'. The 'Functionality' tab shows a list of users: 'You' and 'Kath'. Under 'You', there is a text entry: 'Calculate frequency of a1 allele of the APOE gene using advanced calculation system for better accuracy.' Under 'Kath', there is a text entry: 'Focusing on the ε4 allele, which includes the a1 allele, we proceed to calculate its frequency. In this case, the frequency of the ε4 allele directly represents the frequency of the a1 allele. Therefore, the frequency of the ε4 allele (q) is 0.17.' Below this, there are three paragraphs of text explaining the calculation process and its implications. At the bottom, there is a search bar with the text 'How can I help you?' and a play button icon.

Functionality History <

Workspace

- √ Root
  - > Parent1
  - > Parent2
  - √ Parent3
    - > Child1
      - New Folder...
      - New File...
      - View .csv file
      - Rename
      - Delete
    - √ Child2
      - File1.format
      - File2.format
      - File3.format
      - File4.format
      - File5.format
      - File6.format
      - File7.format
      - File8.format
      - File9.format
    - > Child3
    - > Parent4

**You**

Calculate frequency of a1 allele of the APOE gene using advanced calculation system for better accuracy.

**Kath**

Focusing on the ε4 allele, which includes the a1 allele, we proceed to calculate its frequency. In this case, the frequency of the ε4 allele directly represents the frequency of the a1 allele. Therefore, the frequency of the ε4 allele (q) is 0.17.

Accounting for linkage disequilibrium is unnecessary in this scenario since we're only considering the APOE gene. Statistical analysis beyond direct calculation isn't required either, given that we have the allele frequency directly from the population data.

Validation of the calculated allele frequencies can be crucial, but for the sake of this demonstration, we'll proceed without it. Interpretation of the results reveals that the frequency of the a1 allele of the APOE gene in our hypothetical population is approximately 0.17.

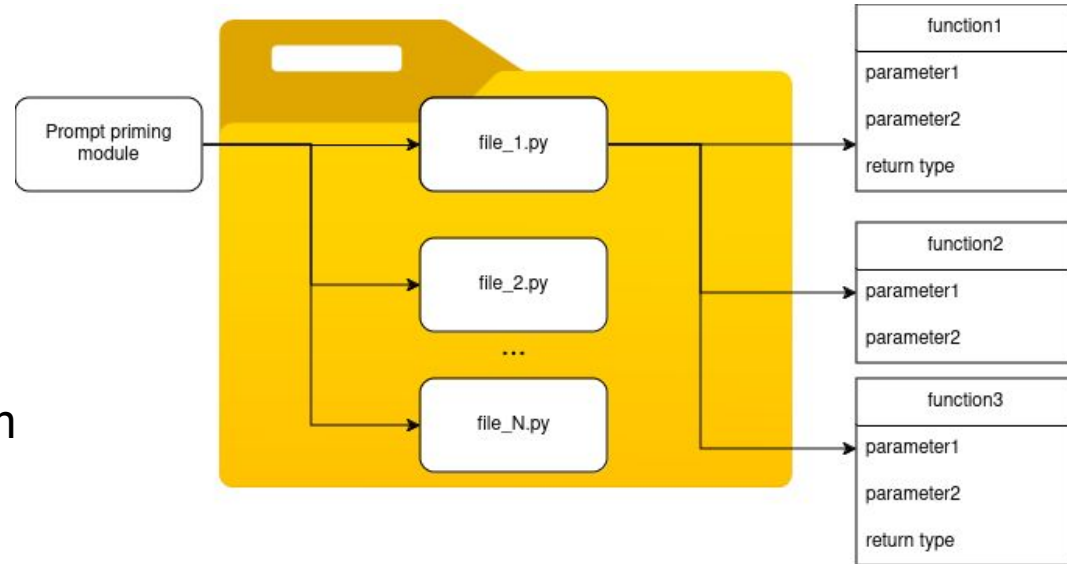
This indicates that around 17% of individuals in this population carry the a1 allele. Such information is invaluable for understanding genetic diversity within populations and can have significant implications for disease risk assessment and personalized medicine approaches.

In summary, by following these steps and utilizing available population data, we can accurately estimate the frequency of the a1 allele of the APOE gene, providing insights into the genetic makeup of populations and its implications for health and medicine.

How can I help you?

# Prompt Priming Module

- Prompt priming generation is automated
- System uses industry standardized documentation files from the integrated DNA analysis tools
- Custom functionality description files can be included in the system





# Results 1: User query

## Prime prompt:

You are Artificial Intelligence tool. Your purpose is to convert scientists prompts into Python code. You have function for downloading data from databases. When person asks you to download data use "store\_database" Python function.

store\_database function download data from GnomAd, Clinvar and LOVD. To download data pass name of database in lowercase. Examples: store\_database('clinvar'), store\_database('gnomad'), store\_database('lovd'). When person asks to download data from database, print just generated function with passed arguments and nothing more.

You have "merge\_lovd\_with\_clinvar" and "merge\_lovd\_with\_gnomad" functions. If persons requests merging of data use this function without parameters.

You also have "convert\_to\_vcf" function that will convert data to vcf format. If persons requests converting to vcf use this function without parameters.

You also have "process\_with\_splice\_ai" function to process data with SpliceAI tool. It returns processed data. When you use this function provide variable to save data. Example, "processed\_data = process\_with\_splice\_ai". Returned object has "display" method. Use it to display data when person asks to display processed data.

## Input:

Download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI. Display results.

## Results 2.1: Unprompted LLM output (Llama3)

```
def main():  
    # Ask user for sample name, output folder, and organism  
    sample_name = input("Enter sample name: ")  
    output_folder = input("Enter output folder: ")  
    organism = input("Enter organism (default: human): ") or "human"  
  
    # Get Clinvar and LOVD data  
    clinvar_data = get_clinvar_data()  
    lovd_data = get_lovd_data()  
  
    # Merge data  
    merged_data = mergeClinvarLOVDdata(clinvar_data, lovd_data)  
  
    # Convert to vcf format  
    vcf_file = convert_to_vcf(merged_data)  
  
    # Pass to SpliceAI  
    splice_ai(vcf_file, sample_name, output_folder, organism)
```

## Results 2.2: LLM output with prompt priming (Llama3)

```
def main():
    database = input("Enter database name: ")
    database = database.lower()

    if database in ['clinvar', 'lovd', 'gnomad']:
        print(store_database(database))
    elif database == 'merge lovd with clinvar':
        print(merge_lovd_with_clinvar())
    elif database == 'merge_lovd_with_gnomad':
        print(merge_lovd_with_gnomad())
    else:
        print("Invalid operation")
        data_from_clinvar = store_database('clinvar')
        data_from_lovd = store_database('lovd')

    merged_data = merge_lovd_with_clinvar()
    converted_data = convert_to_vcf(merged_data)
    result = process_with_splice_ai(converted_data)
    print(result)
```

## Results 3.1: Unprompted LLM output (ChatGPT)

```
import pandas as pd
from pybedtools import BedTool
import subprocess

# Step 2: Merge Data
# Assuming you have merged ClinVar and LOVD data into a DataFrame called
merged_data
merged_data = pd.concat([clinvar_data, lovd_data], ignore_index=True)

# Step 3: Convert to VCF Format
# Assuming you have a function to convert DataFrame to BED format
bed_data = merged_data_to_bed(merged_data)

# Convert BED to VCF using bedtools
bedtool = BedTool.from_dataframe(bed_data)
vcf_output = bedtool.intersect(b='/path/to/reference_genome.bed',
                               wa=True, v=True).to_dataframe()
vcf_output.to_csv('merged_data.vcf', sep='\t', index=False,
                  header=False)

# Step 4: Pass to SpliceAI
# Assuming SpliceAI is installed and available in the environment
subprocess.run(['spliceai', '-I', 'merged_data.vcf', '-O',
                'spliceai_predictions.vcf'])
```

## Results 3.2: LLM output with prompt priming (ChatGPT)

```
store_database('clinvar')  
store_database('lovd')  
merge_lovd_with_clinvar()  
convert_to_vcf()  
processed_data = process_with_splice_ai()  
processed_data.display()
```

# Results 4: SpliceAI output

```
##fileformat=VCFv4.2
```

```
##FILTER=<ID=PASS,Description="All filters passed">
```

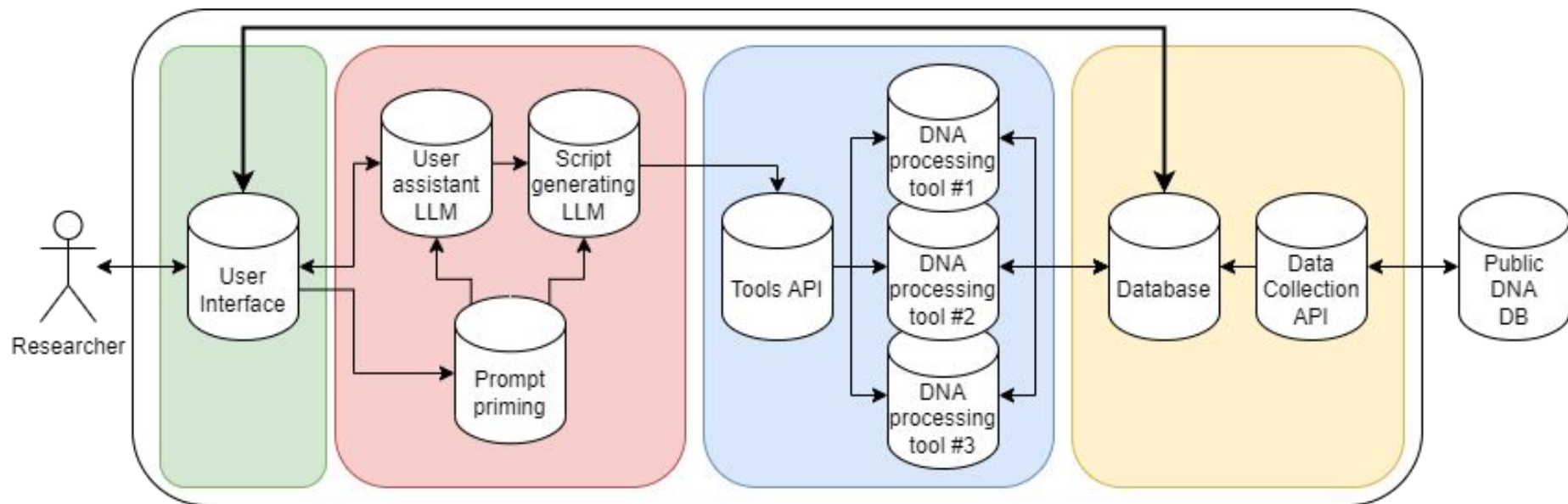
```
##contig=<ID=6,length=63719980>
```

```
##INFO=<ID=SpliceAI,Number=.,Type=String,Description="SpliceAIv1.3.1 variant annotation. These include delta scores (DS) and delta positions (DP) for acceptor gain (AG), acceptor loss (AL), donor gain (DG), and donor loss (DL). Format: ALLELE|SYMBOL|DS_AG|DS_AL|DS_DG|DS_DL|DP_AG|DP_AL|DP_DG|DP_DL">
```

```
#CHROM POS ID REF ALT QUAL FILTER INFO
```

```
6 63789078 . A G . . SpliceAI=G|EYS|0.00|0.00|0.00|0.00|3|9|-20|9
```

# KATH architecture



# Conclusions

1. A KATH system architecture was created for the no-coding application of DNA analysis tools and a TRL-4 prototype was developed;
2. Preliminary results prove the validity of the approach;
3. Further prompt priming research, analysis of specialized LLM models, and incorporation of additional DNA analysis tools will be performed;
4. Minimal Viable Product (TRL-7) shall be delivered to Harvard University researchers by the beginning of Q3 2024 and validation and user experience tests will be performed to fine-tune the system.



Thank you for your time!