* My situation was complicated. I had 4 genomes to sequence with 2.5 GB each, against BLAST DB. However, the online BLAST tool would only allow 100 mb per run. Hence, I planned to download the entire BLAST nucleotide database, as that seemed to be the most viable option. Sadly, I did not have enough space in my laptop’s SSD, and it comprised of 558 GB of compressed data. I then thought of using my external 2TB HDD for sequencing. I approached ROB S (FROM CLASS) for the large bulk download, and luckily he accepted. Although it took a week, I was finally able to get the entire nucleotide database downloaded in the external HDD. The next steps were to extract the compressed data total of 684 GB of uncompressed data. Total of 1.3 GB approximately.
* Now a blast run was performed to run my genome1 : bubalus bubalus (water buffalo) against locally hosted - BLAST nucleotide database.

~~GENOMIC NUCLEOTIDE SEPARATION:~~

~~1 – Download entire nucleotide database from ncbi:~~

~~{Performed in Command Prompt or CMD}~~

~~cd <EXTERNAL HARDDRIVE PATH>~~

~~cd E:\~~

~~now download blast to this repository~~

~~wget -r -nH --cut-dirs=1 --no-parent -A "nt.\*.tar.gz" ftp://ftp.ncbi.nlm.nih.gov/blast/db/~~

~~2- Check drive directory after installation:~~

~~dir E:\db\~~

~~A screenshot of a computer

AI-generated content may be incorrect.~~

~~{In Windows Powershell - Admin}~~

~~3 – Navigate to the external hard drive directory~~

~~cd E:\~~

~~4 – Create a folder called Extracted~~

~~New-Item -ItemType Directory -Path E:\db\Extracted -Force~~

~~5 – Extract all compressed ‘tar’ files into Extracted folder.~~

~~Get-ChildItem E:\db\nt.\*.tar.gz | ForEach-Object { tar -xvf $\_.FullName -C E:\db\Extracted }~~

~~A computer screen with white text

AI-generated content may be incorrect.~~

~~6 – Check if blast recognize the database and it is indexed properly~~

~~blastdbcmd -db "E:\db\Extracted\nt" -info~~

~~A screen shot of a computer

AI-generated content may be incorrect.~~

~~7 – Check location of blastn in Windows Power Shell~~

~~Get-ChildItem -Path C:\ -Recurse -Filter "blastn.exe" -ErrorAction SilentlyContinue~~

~~Not in C drive~~

~~Get-ChildItem -Path D:\ -Recurse -Filter "blastn.exe" -ErrorAction SilentlyContinue~~

~~A screen shot of a computer

AI-generated content may be incorrect.~~

~~Directory: D:\NCBI\_BLAST\blast-2.16.0+\bin~~

~~8- Run the genome1.fasta (water buffalo genome) against the nucleotide database and save the output in a results folder. RUN IN Windows Power Shell~~

~~& "D:\NCBI\_BLAST\blast-2.16.0+\bin\blastn.exe" -query "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\genome1.fasta" -db "E:\db\Extracted\nt" -out "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\results\results1.txt" -outfmt 7 -evalue 0.01 -max\_target\_seqs 50 -num\_threads 8 -task megablast~~

~~9- No matches on the above run, Hence, re running it with a relaxed evalue~~

~~& "D:\NCBI\_BLAST\blast-2.16.0+\bin\blastn.exe" `~~

~~-query "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\genome1.fasta" `~~

~~-db "E:\db\Extracted\nt" `~~

~~-out "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\results\relaxed\_results.txt" `~~

~~-outfmt 7 `~~

~~-evalue 10 `~~

~~-max\_target\_seqs 100 `~~

~~-num\_threads 8 `~~

~~-task megablast~~

~~10 – It seems, the genome being 2.5 gb is too big for BLAST to sequence. Hence, breaking it into several parts before retrying. Run this in Powershell~~

~~&"C:\Users\sathy\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10\_qbz5n2kfra8p0\python.exe" "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\split\_pythonscript.py~~

~~11 – Running this in Powershell, to run the Blastn on chunks (genome split up), storing results and also running through a harddrive with the blast DB.~~

~~$blastExe = "D:\NCBI\_BLAST\blast-2.16.0+\bin\blastn.exe"~~

~~$chunkFolder = "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\chunks"~~

~~$resultsFolder = "D:\Documents\Python Stuff - Programming\AMOD Big Data research project\BLAST\_RUN\results"~~

~~$blastDB = "E:\db\Extracted\nt"~~

~~# Make sure results folder exists~~

~~if (!(Test-Path $resultsFolder)) {~~

~~New-Item -ItemType Directory -Path $resultsFolder | Out-Null~~

~~}~~

~~# Loop through all chunk\_\*.fasta files and BLAST each~~

~~Get-ChildItem "$chunkFolder\chunk\_\*.fasta" | ForEach-Object {~~

~~$inputFile = $\_.FullName~~

~~$outputFile = "$resultsFolder\$($\_.BaseName)\_results.txt"~~

~~Write-Host "Running BLAST for $($\_.Name)..."~~

~~& $blastExe -query $inputFile -db $blastDB -out $outputFile -outfmt 7 -evalue 10 -max\_target\_seqs 50 -num\_threads 6 -task megablast~~

~~}~~

**<><><><><><><><><><><><><><><><><><><><><><><><><><><>**

**ANOTHER THING TO REMEMBER! EXECUTION OF THIS ONE TIME, TOOK ME CLOSE TO 15 HOURS! BECAUSE MY SETUP WAS MENTAL. CRAZY HOW YOU ASK? I HAD THE ENTIRE NCBI BLAST NUCLEOTIDE DATABASE (1.3 TB) of FILES IN MY EXTERNAL HARDDISK (HDD), I HAD TO READ A 2.5 GB FASTA FILE TO AND FROM MY INTERNAL SSD TO EXTERNAL HDD, SAVE OUTPUT BACK IN MY INTERNAL SSD. IT TOOK OVER A WEEK OF TRIAL AND ERROR, AND WHAT DID I OBSERVE? NOTHING! THIS ALMOST TOOK ME OUT :’) BUT I STEELED THROUGH 😉.**

**<><><><><><><><><><><><><><><><><><><><><><><><><><><>**

**“LIFE DOESN’T ALWAYS GO YOUR WAY! SOMETIMES ITS BETTER TO MOVE FORWARD, RATHER THAN STAYING STILL!”**

**<><><><><><><><><><><><><><><><><><><><><><><><><><><>**

12 – Since full genome blast isn’t working, we’re doing a genome similarity with mash. It seems Mash does not exist in windows. Hence, I’m using WSL to run it on linux terminal.

***Mash is a lighter weight way of comparing entire genomes, since BLAST is computationally expensive. I plan to use BLAST but only on smaller sequences, maybe on further analysis.***  
  
wget <https://github.com/marbl/Mash/releases/download/v2.3/mash-Linux64-v2.3.tar>

ls

13 – Extract the tar file

tar -xvf mash-Linux64-v2.3.tar

14 - Make the binary globally accessible and check version

sudo mv mash-Linux64-v2.3/mash /usr/local/bin/

mash --version

15 – Now, need to download genome for water buffalo, takin and wild yak from the NCBI website. With the following filetypes checked.

A screenshot of a computer

AI-generated content may be incorrect.

This comes compressed in .ZIP format, you need to extract it. The files are as follows:

**🧾 What Are These Files?**

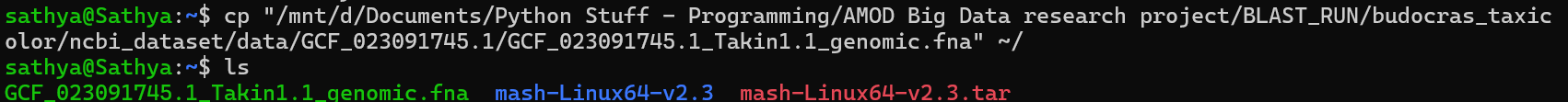
|  |  |  |  |
| --- | --- | --- | --- |
| **Filename** | **Type** | **What it contains** | **Use it for...** |
| \*.genomic.fna | FASTA | The entire genomic DNA sequence | ✅ Use this for mash, minimap2, full-genome BLAST |
| cds\_from\_genomic.fna | FASTA | Only the coding sequences (CDS regions) extracted from genome | 🔬 Use this for gene-level BLAST (faster than full genome) |
| genomic.gbff | GenBank Flat File | Full sequence + all annotations (genes, CDS, rRNA...) | Use for gene/CDS parsing or loading into genome browsers |
| genomic.gff | GFF (General Feature Format) | Gene positions, coordinates, exons | Use for extracting features with tools like bedtools, gffread |
| genomic.gtf | GTF (similar to GFF, used in RNA-seq) | Structured version of GFF | Skip unless you're doing transcript analysis |

We need to use **GCF\_023091745.1\_Takin1.1\_genomic.fna** for mash, as it contains the entire genome.

17 – copy the file from windows (desktop) to WSL (windows subterminal for linux)

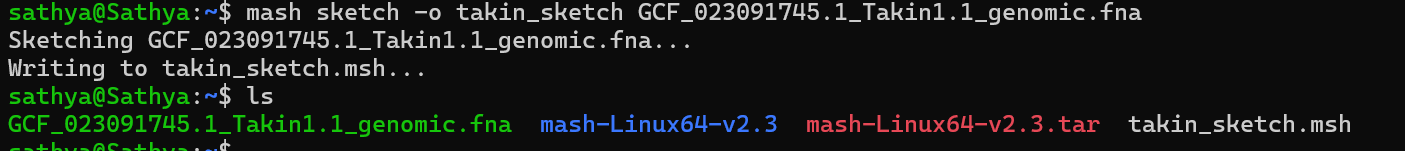
Takin:

cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/BLAST\_RUN/budocras\_taxicolor/ncbi\_dataset/data/GCF\_023091745.1/GCF\_023091745.1\_Takin1.1\_genomic.fna" ~/



18 – Sketch the Genome (Takin)

mash sketch -o takin\_sketch GCF\_023091745.1\_Takin1.1\_genomic.fna



19 – copy the file from windows (desktop) to WSL (windows subterminal for linux)

Water Buffalo

cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/BLAST\_RUN/water\_buffalo/ncbi\_dataset/data/GCF\_019923935.1/GCF\_019923935.1\_NDDB\_SH\_1\_genomic.fna" ~/

20 – copy the file from windows (desktop) to WSL (windows subterminal for linux)

Wild yak

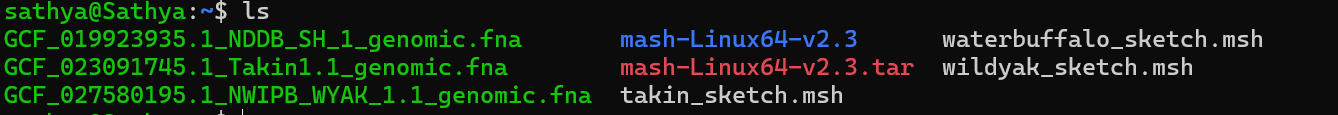
cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/BLAST\_RUN/wildyak/ncbi\_dataset/data/GCF\_027580195.1/GCF\_027580195.1\_NWIPB\_WYAK\_1.1\_genomic.fna" ~/

21 – Sketch the Genome (Water Buffalo)

mash sketch -o waterbuffalo\_sketch GCF\_019923935.1\_NDDB\_SH\_1\_genomic.fna

22 - Sketch the Genome (Wild yak)

mash sketch -o wildyak\_sketch GCF\_027580195.1\_NWIPB\_WYAK\_1.1\_genomic.fna



23 – Now time to compare the genomes

Takin -> Water Buffalo

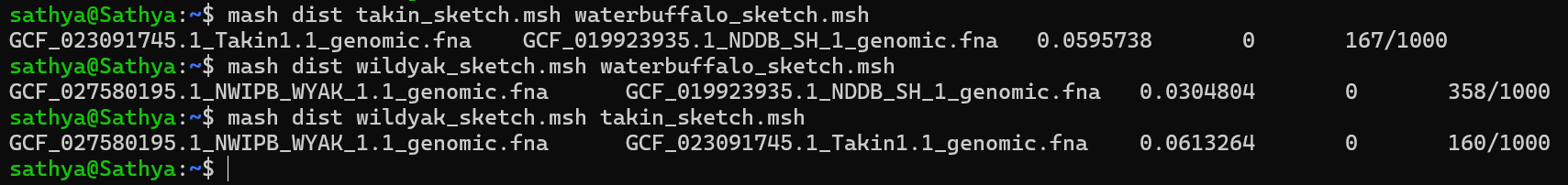
mash dist takin\_sketch.msh waterbuffalo\_sketch.msh

Wildyak -> Water Buffalo

mash dist wildyak\_sketch.msh waterbuffalo\_sketch.msh

Takin -> Wildyak

mash dist wildyak\_sketch.msh takin\_sketch.msh

****

✅ What is **Mash Distance?**

The Mash distance is an approximation of dissimilarity between two genome sequences based on k-mer sketches.

🧪 Formula Behind Mash:

Mash estimates ***distance D*** using the ***Jaccard index (J) of shared k-mers:***

D = -1/k \* log(2J / (1 + J))

Where:

* **k = k-mer size (usually 21)**
* **J = observed Jaccard index (shared k-mers over union)**

But for simplicity, Mash just outputs the distance as a number between:

* **0 = identical sequences**
* **1 = completely different**

The way to approximate similarity is :

% similarity = (1 - Mash distance) × 100

|  |  |  |
| --- | --- | --- |
| **Comparison** | **Mash Distance** | **Approx. Similarity** |
| **Wild Yak 🐂 vs. Water Buffalo 🐃** | **0.0304** | **96.96% (Highest)** |
| **Takin 🐐 vs. Water Buffalo🐃** | **0.0596** | **94.04%** |
| **Wild Yak 🐂 vs. Takin 🐐** | |  | | --- | |  |   **0.0613** | **93.87%** |

**24 –** Now Generate 1000 sketches for Takin, Wild Yak and Water Buffalo. This will increase the accuracy

mash sketch -s 10000 -o takin\_sketch\_10k GCF\_023091745.1\_Takin1.1\_genomic.fna

mash sketch -s 10000 -o waterbuffalo\_sketch\_10k GCF\_019923935.1\_NDDB\_SH\_1\_genomic.fna

mash sketch -s 10000 -o wildyak\_sketch\_10k GCF\_027580195.1\_NWIPB\_WYAK\_1.1\_genomic.fna

A computer screen with white text

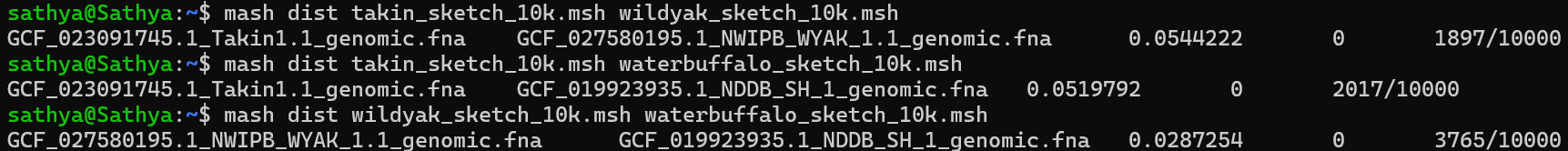
AI-generated content may be incorrect.

25 –

mash dist takin\_sketch\_10k.msh wildyak\_sketch\_10k.msh

mash dist takin\_sketch\_10k.msh waterbuffalo\_sketch\_10k.msh

mash dist wildyak\_sketch\_10k.msh waterbuffalo\_sketch\_10k.msh



|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison (for 10k resolution)** | **Mash Distance** |  | **Approx. Similarity** |
| **Wild Yak 🐂 vs. Water Buffalo 🐃** | **0.0287** | **3765/10,000** | **97.13%** |
| **Takin 🐐 vs. Water Buffalo 🐃** | **0.0520** | |  | | --- | |  |  |  | | --- | | **2017/10,000** | | **94.80%** |
| **Wild Yak 🐂 vs. Takin 🐐** | |  | | --- | |  |   **0.0544** | **1897/10,000** | **94.56%** |

26 – Saving the high resolution ‘Mash’ Sketches in highres\_distances.tsv

mash dist \*\_10k.msh > highres\_distances.tsv



27 – Copying from Windows Subsystem for Linux (WSL) to Windows. Here, /mnt/ is to mount to windows.

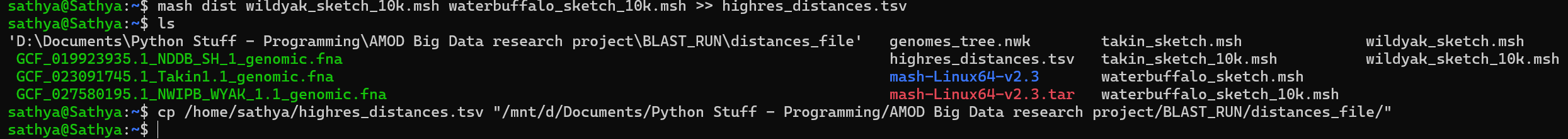
cp /home/sathya/highres\_distances.tsv "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/BLAST\_RUN/distances\_file/"

A screenshot of a computer

AI-generated content may be incorrect.

28 – Looks like there was a field missing, hence

mash dist wildyak\_sketch\_10k.msh waterbuffalo\_sketch\_10k.msh >> highres\_distances.tsv



Now it’s a lot better.

29 – Create a python script to read the .tsv file, change name of files (to be readable), plot a heatmap.

A screenshot of a computer

AI-generated content may be incorrect.

heatmap\_genomic\_distance.py ~ with the python script to print the heatmap between the 3 species.

The output heatmap:

A chart of different colors

AI-generated content may be incorrect.

30 –

**GCA\_023091745.2** = version 2 of a GenBank assembly.

**GCF\_023091745.1** = version 1 of a RefSeq assembly (reviewed and standardized by NCBI).

So, GCF > GCA

|  |  |  |  |
| --- | --- | --- | --- |
| **Prefix** | **Meaning** | **Source** | **Curation Level** |
| GCA | Genomic Assembly | Submitted assembly (NCBI GenBank) | May be preliminary or not fully curated |
| GCF | Genomic Reference | RefSeq assembly (NCBI RefSeq) | Curated and standardized by NCBI |

31 – Now, need to extract gene and features from a GBIF file of each species

**On parsing .gbff and extracting all the features:**

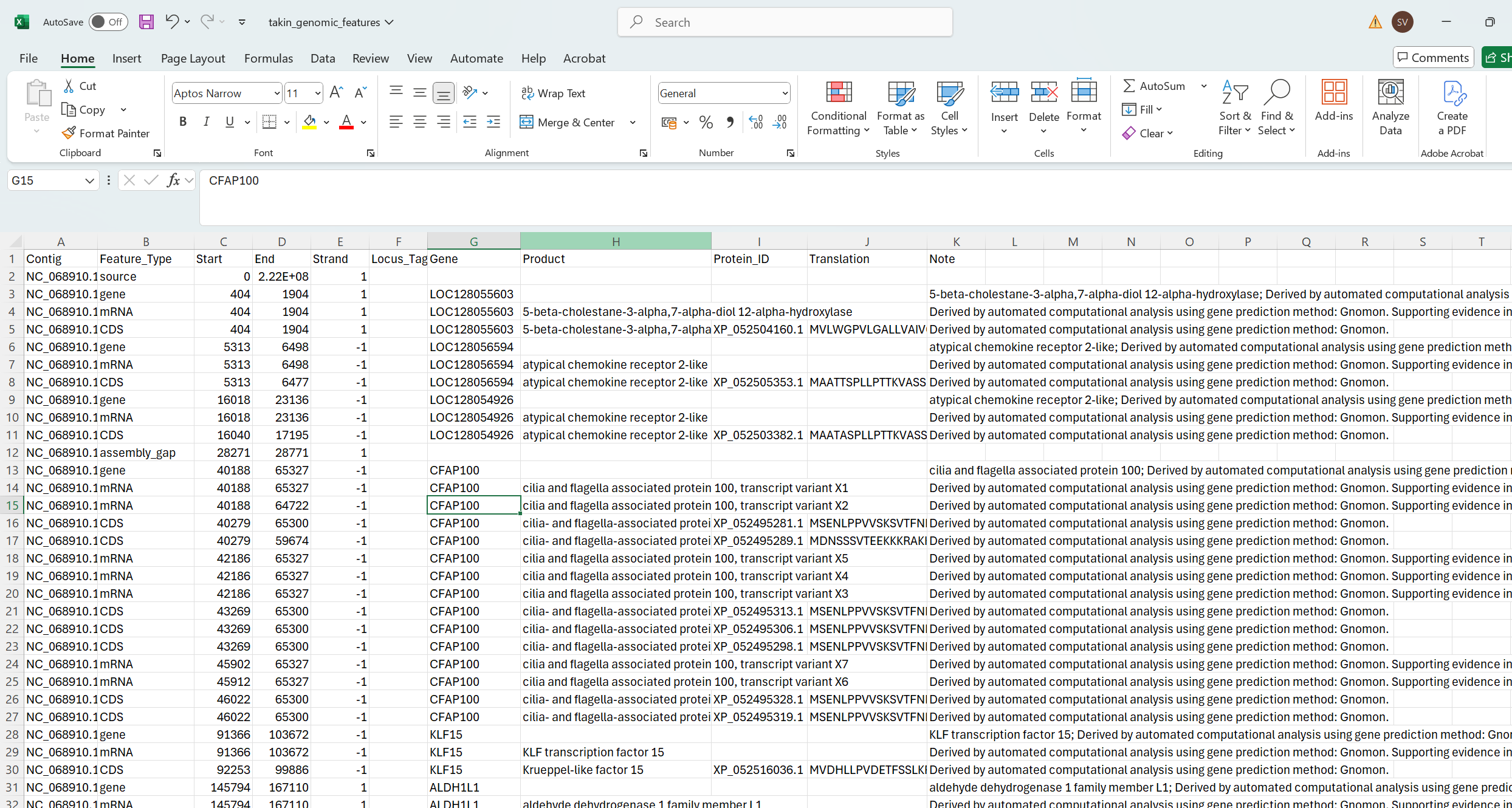
* Gene locations
* CDS (coding sequences)
* Product descriptions
* Locus tags
* Protein translations (if available)

32 – After running the python script, I extracted the genomic features in .csv files.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.



|  |  |
| --- | --- |
| **Column** | **Description** |
| **Contig** | Accession number of the chromosome or scaffold (e.g., NC\_068910.1) |
| **Feature\_Type** | Type of feature: gene, mRNA, CDS (coding sequence), source, etc. |
| **Start / End** | Genomic coordinates of the feature |
| **Strand** | 1 (forward strand), -1 (reverse strand) |
| **Locus\_Tag** | Unique ID for the gene (often missing in automated annotations) |
| **Gene** | Gene symbol (may be missing for predicted genes) |
| **Product** | Protein/gene description or function |
| **Protein\_ID** | Protein accession number (e.g., XP\_...) |
| **Translation** | Amino acid sequence (only for CDS features) |
| **Note** | Additional info, such as prediction methods, similarities, etc. |

33 – Now time to understand the dataset, using a python script, to do a feature analysis. Hence, first doing summary statistics. A python script to extract genomic features summary as a text file does the job!

A screenshot of a computer

AI-generated content may be incorrect.

1. **Takin (*Budorcas taxicolor*)**
   * Large goat-antelope native to the eastern Himalayas.
   * Mountainous, herbivorous, adapted to cold regions.
2. **Water Buffalo (*Bubalus bubalis*)**
   * Domesticated bovine species important in agriculture.
   * Semi-aquatic, more heat-tolerant, lives in lowland wetlands.
3. **Wild Yak (*Bos mutus*)**
   * Large bovid found in the Tibetan Plateau.
   * High-altitude specialist, cold-hardy, and wild relative of the domestic yak.

These species diverged within the Bovidae family and have adapted to unique ecological niches — thus, comparative genomics can reveal evolutionary, functional, and adaptive insights.

**🐐 🔹 Takin**

**📊 General Stats**

* **Total Features (rows):** 100,690
* **Unique Genes:** 25,927
* **Unique Proteins:** 28,977
* **Unique Contigs:** 1,363

**🧠 Feature Composition**

* Most common features:  
  CDS (29,116), mRNA (28,964), gene (28,399)
* Others include tRNA, ncRNA, exon, and rare ones like D-loop, rep\_origin

**🧬 Gene Structure**

* **Average Gene Length:** ~67,204 bp
* **Longest Gene:** 221,681,955 bp (!)
* **Shortest Gene:** 1 bp

**➕ Strand Distribution**

* Forward strand: 51,594
* Reverse strand: 49,096 → ⚖️ Nearly balanced

**🧪 Product Summary**

* Common products:
  + U6 spliceosomal RNA, tRNA-Cys, tRNA-Gly, 5S rRNA, etc.
* **Missing Product Entries:** 37,105 (~36.8%)

**📋 Notes**

* Dominated by:  
  Derived by automated computational analysis using gene prediction method: Gnomon.  
  with various support levels.

**🐃 🔹 Water Buffalo**

**📊 General Stats**

* **Total Features (rows):** 192,351
* **Unique Genes:** 34,756
* **Unique Proteins:** 64,378 (!)
* **Unique Contigs:** 26

**🧠 Feature Composition**

* Heavy on CDS (64,670), mRNA (64,365), gene (37,167)
* Rich presence of ncRNA, tRNA, exon, misc\_RNA

**🧬 Gene Structure**

* **Average Gene Length:** ~73,962 bp
* **Longest Gene:** 202,348,575 bp
* **Shortest Gene:** 1 bp

**➕ Strand Distribution**

* Forward strand: 96,407
* Reverse strand: 95,944 → ⚖️ Extremely balanced

**🧪 Product Summary**

* Most common: U6 spliceosomal RNA, tRNA-Gly, tRNA-Cys, etc.
* **Missing Product Entries:** 44,663 (~23.2%)

**📋 Notes**

* Dominated by:
  + Gnomon (96,380)
  + cmsearch and tRNAscan-SE also present
  + RNA-seq evidence-supported features

**🐂 🔹 Wild Yak**

**📊 General Stats**

* **Total Features (rows):** 139,796
* **Unique Genes:** 28,141
* **Unique Proteins:** 45,617
* **Unique Contigs:** 2,040 (!)

**🧠 Feature Composition**

* Heavy on:  
  CDS (45,653), mRNA (45,604), gene (30,156)
* Others include exon, ncRNA, assembly\_gap

**🧬 Gene Structure**

* **Average Gene Length:** ~72,073 bp
* **Longest Gene:** 155,998,435 bp
* **Shortest Gene:** 1 bp

**➕ Strand Distribution**

* Forward strand: 71,162
* Reverse strand: 68,634 → slightly forward-heavy

**🧪 Product Summary**

* Many **uncharacterized lncRNAs** and **spliceosomal RNAs**
* **Missing Product Entries:** 39,553 (~28.3%)

**📋 Notes**

* Dominated by Gnomon-predicted annotations
* Includes cmsearch support for some ncRNAs

**📌 Key Takeaways Across All Species**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Takin** | **Water Buffalo** | **Wild Yak** |
| Total Rows | 100,690 | 192,351 | 139,796 |
| Unique Genes | 25,927 | 34,756 | 28,141 |
| Unique Proteins | 28,977 | 64,378 | 45,617 |
| Avg. Gene Length (bp) | 67,204 | 73,962 | 72,073 |
| Longest Gene (bp) | 221M | 202M | 156M |
| % Missing Product Fields | ~36.8% | ~23.2% | ~28.3% |
| Feature Type Richness | Moderate | High | High |
| Contigs Used | 1,363 | 26 | 2,040 |

**🧠 Interpretations:**

* **Water Buffalo has the highest gene and protein count**: Possibly due to better-annotated genome or more isoforms captured.
* **Wild Yak has the longest gene**: A gene of 2.67 Mb could indicate complex structure — likely regulatory, developmental, or immune-related.
* **Average gene length is longest in Water Buffalo** — may suggest longer introns or regulatory regions.
* **Takin’s genome is highly fragmented** (1,363 contigs) compared to Buffalo (26) → assembly quality difference.

**🧬 Feature Types: What Do They Mean?**

|  |  |  |
| --- | --- | --- |
| **Feature Type** | **Full Form** | **What It Represents** |
| gene | — | A functional unit of heredity |
| CDS | Coding Sequence | Region that gets translated into protein |
| mRNA | Messenger RNA | Transcript of the gene (with/without introns) |
| tRNA | Transfer RNA | Carries amino acids during translation |
| rRNA | Ribosomal RNA | Forms core of ribosome, essential for translation |
| ncRNA | Non-Coding RNA | Includes lncRNA, miRNA, snoRNA; regulatory roles |
| exon | — | Coding or non-coding part retained in mRNA |
| misc\_RNA | Miscellaneous RNA | Often small regulatory RNAs not fitting other classes |
| V\_segment, C\_region, D-loop | Immune/recombination regions | Key for adaptive immunity |
| rep\_origin | Replication Origin | Where DNA replication begins |
| assembly\_gap | — | Gaps in the assembly, unknown sequence |

**🔍 Functional Genes and Patterns**

**🧬 High Copy Genes**

Across species, several **tRNA genes** appear in high abundance:

|  |  |
| --- | --- |
| **tRNA Type** | **Role** |
| tRNA-Gly, tRNA-Cys, tRNA-Ser | Essential in translation; often present in multiple copies for efficiency. |

These are among the most **duplicated genes**, showing evolutionary preference for efficient translation.

**🔎 Hidden Patterns and Evolutionary Clues**

**1. Uncharacterized or Missing Products**

* **~40% of annotations have no "Product" description**.
* This could mean:
  + Hypothetical proteins
  + Unknown function
  + Assembly errors or computational-only predictions

This presents **targets for functional characterization or machine learning annotation**.

**2. Strand Symmetry**

* All three species have nearly balanced gene distributions on the + and – strands.
* This reflects **random distribution of genes** with respect to replication direction — a common trait in eukaryotes.

**3. Longest Translations**

* Top proteins (lengths ~34,000 AAs) are **outliers** — may represent:
  + Misannotations?
  + Repetitive elements?
  + Long regulatory proteins (e.g., titin analogs?)

Further sequence validation would be needed to verify biological accuracy.

**4. Dominant RNA Types**

* U6 spliceosomal RNA is consistently the top small RNA → **core part of spliceosome**, vital in eukaryotic mRNA splicing.
* Many lncRNAs in Wild Yak suggest either better annotation of regulatory elements or over-prediction.

**🧠 About the Annotation Methods (WHAT WAS USED TO ANNOTATE THE GENE IN NCBI)**

Most notes say:

**"Derived by automated computational analysis using gene prediction method: Gnomon."**

**🔧 Gnomon**

* A **NCBI in-house gene prediction tool**
* Uses known protein alignments, transcriptome data, ab initio predictions
* High-throughput, not always experimentally validated

Other methods:

* **tRNAscan-SE**: Specialized tool for tRNA prediction.
* **cmsearch**: Infernal tool for finding ncRNAs via covariance models.

These annotation tools can sometimes over-predict genes or generate multiple isoforms due to lack of transcript validation.

34 – Now have to group genes in the different species, and compare the prevalence of the genes. Now run a python script. It reads the .csv file we extracted from previous step (STEP 32).

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

The output we got was as follows (the excel has the same data but in text):  
  
A yellow and blue lines

AI-generated content may be incorrect.

* These are called “Keywords”, and they describe what a gene or protein does. They are **biological terms** available in the **Product** field of gene annotations.

|  |  |  |
| --- | --- | --- |
| **Term Type** | **Example** | **Is it a gene?** |
| **Keyword** (what you're using) | "kinase", "zinc" | ❌ Not a gene, but a category |
| **Gene symbol** | MAPK1, ZNF217, IL6R | ✅ Yes |
| **Protein name** | "Tyrosine kinase receptor" | ❌ Functional name |

**KEYWORDS:**

**🧬 Transcription Factors & Regulators**

["transcription", "zinc", "homeobox", "helix-loop-helix", "forkhead", "nuclear receptor", "TF", "coactivator", "corepressor"]

**🧠 Signal Transduction**

["kinase", "phosphatase", "second messenger", "MAPK", "cAMP", "PKA", "receptor", "calmodulin"]

**🧪 Stress & Detox**

["oxidase", "reductase", "peroxidase", "heat shock", "HSP", "ubiquitin", "proteasome", "stress"]

**⚡ Energy Metabolism**

["cytochrome", "ATPase", "dehydrogenase", "oxidase", "glycolysis", "mitochondrial"]

**❄️ High-Altitude Adaptation (Hypoxia-related)**

["HIF", "hypoxia", "VEGF", "cytochrome", "carbonic anhydrase", "oxygen", "myoglobin", "hemoglobin"]

**🛡️ Immune Genes**

["interleukin", "toll-like", "TLR", "major histocompatibility", "MHC", "defensin", "immunoglobulin", "cytokine"]

**🧫 Development & Growth**

["growth factor", "morphogen", "developmental", "Wnt", "BMP", "Notch"]

✅ **Original + GPCR family**

["kinase", "GPCR", "G protein-coupled", "7TM", "seven transmembrane", "zinc", "transport", "receptor",

    "cytochrome", "ATPase", "ubiquitin", "oxidase", "synthetase"]

------------------------------------------------------------------------------------------------------------------------------------

IN DEPTH BREAKDOWN OF EACH:  
**🔹 1. kinase**

**Function:** Enzymes that transfer phosphate groups to other proteins (phosphorylation), turning them "on" or "off".  
**Why it's important:** Master regulators of almost every biological process — cell signaling, growth, immune response.  
**High in Buffalo:** Buffalo has intense signaling networks, possibly due to **domestication, immune adaptation, or growth selection**.  
**Lower in Takin:** Possibly less complex signaling needs or incomplete annotation.

**🔹 2. G protein-coupled / GPCR**

**Function:** Cell surface receptors sensing light, hormones, odorants. Activate G-proteins for internal response.  
**Why it's important:** Largest receptor family in mammals. Key for **sensation and hormonal control**.  
**High in Yak & Buffalo:** Suggests strong sensory needs and hormone signaling.  
**Low in Takin:** Annotation may use different terms (like “receptor” without “G protein-coupled”).

**🔹 3. zinc**

**Function:** Often part of zinc-finger transcription factors → DNA-binding proteins for regulating gene expression.  
**Why it's important:** Core to transcriptional control, development, and stress response.  
**High in Buffalo:** Suggests **intense gene regulation** complexity.  
**Lower in Takin:** Simpler gene control network or annotation gaps.

**🔹 4. transport**

**Function:** Moves molecules (ions, proteins, nutrients) across membranes. Includes ATP transporters, channels.  
**Why it's important:** Maintains **cellular balance**, nutrient uptake, detoxification.  
**High in Buffalo:** Possibly linked to **metabolic activity** and nutrient regulation.  
**Yak also high:** High-altitude animals need **tight oxygen/ion control**.

**🔹 5. receptor**

**Function:** Proteins that receive signals (e.g., hormones, neurotransmitters, immune signals).  
**Why it's important:** Central to communication between cells.  
**Buffalo highest:** High communication complexity — maybe due to **immune, growth, or reproductive functions**.  
**Yak second:** Yak still maintains strong signaling pathways — key for **environmental adaptation**.

**🔹 6. cytochrome**

**Function:** Enzymes that catalyze redox (oxidation/reduction) reactions, esp. in mitochondria.  
**Why it's important:** Energy production, detoxification.  
**Yak > Takin:** Wild Yak needs efficient **aerobic metabolism** in **oxygen-limited environments**.  
**Buffalo highest:** Likely because of better annotation + energy-intensive traits from domestication.

**🔹 7. ATPase**

**Function:** Hydrolyzes ATP to pump ions, molecules. Energy-powered transporters.  
**Why it's important:** Maintains membrane potential, pH, ion gradients.  
**Yak > Takin:** Helps in stress conditions — e.g., maintaining **cellular ion balance in hypoxia**.

**🔹 8. ubiquitin**

**Function:** Tags damaged or unnecessary proteins for degradation.  
**Why it's important:** Maintains **protein quality** and stress recovery.  
**Yak > Takin:** High stress in wild, cold environments requires frequent **protein recycling**.

**🔹 9. oxidase**

**Function:** Catalyzes oxidation reactions, removes electrons (e.g., part of detox pathways).  
**Why it's important:** Detoxifies harmful metabolic byproducts, supports energy generation.  
**Yak > Takin:** Detox is more important under **high-altitude oxidative stress**.

**🔹 10. synthetase**

**Function:** Enzymes that **build molecules** (e.g., aminoacyl-tRNA synthetase).  
**Why it's important:** Key for **biosynthesis**, especially protein assembly.  
**Low counts in all:** Conserved, essential function — not a huge family.

**🔹 11. transcription**

**Function:** Broad category for genes regulating transcription.  
**Why it's important:** Controls which genes are turned on/off.  
**Buffalo highest:** High regulation complexity.  
**Yak follows closely:** High regulation needed for altitude/environment adaptation.

**🔹 12. homeobox, forkhead, nuclear receptor, etc.**

**Function:** Specialized transcription factor families — regulate **development, cell fate, hormone response**.  
**Why it's important:** Orchestrate growth, tissue patterning.  
**Buffalo > Yak > Takin:** Growth and development demands in domesticated species are usually greater.

**🔹 13. phosphatase / MAPK / cAMP / calmodulin**

**Function:** All related to **signal transduction** — the internal relay of messages from receptors.  
**Why it's important:** Signal decoding → controls response to hormones, stress, stimuli.  
**Yak & Buffalo rich:** Both need precise control of **response to external cues**.

**🔹 14. reductase, peroxidase, HSP, proteasome**

**Function:** All part of **oxidative stress defense and protein damage repair**.  
**Why it's important:** High-altitude, UV-rich, and cold environments generate **cellular stress**.  
**Yak > Takin:** Consistent with **altitude-driven pressure on cellular stability**.

**🔹 15. mitochondrial**

**Function:** Mitochondrial proteins power the cell, make ATP.  
**Why it's important:** Central to **energy metabolism**.  
**Yak has more than Takin:** Adaptation to **hypoxia** — mitochondria efficiency critical.  
**Buffalo tops due to annotation richness.**

**🔹 16. HIF / hypoxia / oxygen / hemoglobin / myoglobin**

**Function:** Specific to **oxygen regulation and delivery**.  
**Why it's important:** Core for high-altitude adaptation.  
**Very low overall:** These genes are few but essential — may not always be keyword-matched.

**🔹 17. interleukin / toll-like / MHC / immunoglobulin / cytokine**

**Function:** Immune signaling, pathogen recognition, antibody production.  
**Why it's important:** Defense against pathogens — both innate and adaptive immunity.  
**Buffalo > Yak > Takin:** Domesticated animals face more **pathogen exposure**, have **diversified immunity**.

**🔹 18. growth factor / morphogen / BMP / Wnt / Notch**

**Function:** Developmental genes — control embryogenesis, tissue growth, regeneration.  
**Why it's important:** Key in **reproduction, growth, adaptation to physical environments**.  
**Buffalo leads:** Growth traits likely selected in breeding.  
**Yak follows — Takin trails**: Consistent with wild vs. domestic life pressures.

**🧪 Why It Is Like This: Deeper Reasons**

|  |  |
| --- | --- |
| **Pattern** | **Biological Reason** |
| **Buffalo dominance** | Domesticated → richer annotation, selected for growth, immunity, metabolism |
| **Yak middle ground** | Wild but adapted to extreme environment → selected for **stress & oxygen efficiency** |
| **Takin lowest** | Possible real genome streamlining + under-annotation |

SHARED GENOMIC ARCHITECTURE (SIMILARITIES):

* These are **housekeeping and essential pathways**. Their presence and counts reflect evolutionary conservation.

|  |  |  |
| --- | --- | --- |
| **Shared Function** | **Genes/Keywords** | **Why Shared?** |
| **Energy metabolism** | ATPase, cytochrome, dehydrogenase, mitochondrial | All need ATP, respiration |
| **Protein quality control** | ubiquitin, proteasome, heat shock, oxidase | Universal stress and folding pathways |
| **Signal transduction** | kinase, receptor, calmodulin | Required for hormone and cell communication |
| **Transcriptional regulation** | zinc, TF, homeobox, coactivator, corepressor | Basic gene expression and development |
| **Immune response** | cytokine, interleukin, TLR, MHC | Defense against infection |
| **Growth & development** | Notch, Wnt, BMP, growth factor | Shared embryonic development patterns |

**⚡ Signal Transduction:**

**Buffalo > Yak > Takin**  
| Genes: kinase, receptor, phosphatase, MAPK, calmodulin, G protein-coupled |

* Buffalo's higher counts show **richer regulatory complexity**
* Likely reflects **selective breeding** and **tissue-specific gene duplication**
* Yak has a more nuanced signal system → needed for **environmental sensing and adaptation**
* Takin shows minimal complexity — perhaps more **stable or less varied environments**

**🔬 Transcription Regulation:**

**Buffalo > Yak > Takin**  
| Genes: zinc, transcription, homeobox, TF, forkhead, nuclear receptor |

* High counts in Buffalo = **gene network control for growth and milk production**
* Yak shows strong presence → reflects **adaptive gene regulation to high-altitude**
* Takin shows fewer regulators → possibly reflects more **streamlined regulation**

**🛡️ Immune System:**

**Buffalo > Yak > Takin**  
| Genes: interleukin, TLR, immunoglobulin, MHC, cytokine, defensin |

* Domesticated animals face **higher pathogen load** (crowding, human contact)
* Buffalo likely underwent **immune expansion**
* Yak retains intermediate immunity — wild, but exposed to environmental stress
* Takin has minimal innate immune gene variety → suggests **lower pathogen pressure** in the wild

**❄️ Hypoxia & Altitude Adaptation:**

**Yak > Takin > Buffalo (in function, not counts)**  
| Genes: hypoxia, HIF, oxygen, carbonic anhydrase, mitochondrial, cytochrome |

* Yak leads in **oxygen-processing gene families**
* Carbonic anhydrase and mitochondrial genes are crucial for **oxygen buffering**
* Buffalo has higher counts due to annotation — but **Yak's counts matter more functionally**
* Takin matches in some areas (e.g., cytochrome), showing **shared adaptation signatures**

**🧪 Stress & Detox:**

**Yak > Buffalo > Takin**  
| Genes: oxidase, reductase, peroxidase, ubiquitin, HSP, proteasome |

* Yak faces **cold-induced oxidative stress, UV damage** → needs these defenses
* Buffalo faces **pathogen-induced stress**, hence similar needs
* Takin has lower counts, may rely on **passive or behavioral adaptations**

**🧫 Growth & Development:**

**Buffalo > Yak > Takin**  
| Genes: growth factor, Wnt, Notch, BMP, morphogen, developmental |

* Buffalo under strong **selective breeding** for size, milk, growth traits
* Yak needs robust development for surviving harsh climates (e.g., thick skin, muscles)
* Takin shows fewer developmental genes, possibly due to **stabilized wild niche**

**WHAT WE CAN UNDERSTAND FROM THE ABOVE ANALYSIS AND OUTPUT:**

|  |  |
| --- | --- |
| **Insight** | **Explanation** |
| **Domestication expands gene families** | Buffalo has higher counts across all themes. Suggests selection + better annotation. |
| **Wild Yak = specialized adaptation** | Emphasizes energy, stress, immunity. Reflects evolutionary pressure of high altitude. |
| **Takin = streamlined or under-annotated** | May be genuinely minimalistic, or genome assembly is less annotated. |
| **Gene family size ≠ importance** | Even 1–2 genes (like HIF, myoglobin) can be essential if under selection. |

35 – The next step is to, extract protein sequence and convert them into fasta for InterProScan.

Ran a python script and it converts it to a protein sequence for each species (takin, wildyak and water buffalo).

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

The reason behind using Interproscan is:

It adds **true biological meaning** to your proteins by assigning known **functional domains**

* 🧠 Protein kinase domain
* 🔩 Zinc finger
* 🧬 Homeobox
* 🧪 Cytochrome P450
* 🧫 Major facilitator superfamily

These annotations come from curated databases like **Pfam**, **TIGRFAM**, **SMART**, and **SUPERFAMILY**.

* Each protein will have a:
* Domains (e.g., Pfam, SMART, CDD)
* Functional descriptions
* Associated InterPro IDs (linked to GO/KEGG)

36 – Now, need to download InterProScan. Since it is only available in Linux. We use WSL – windows subsystem for linux and run the following:

wget <https://ftp.ebi.ac.uk/pub/software/unix/iprscan/5/5.73-104.0/interproscan-5.73-104.0-64-bit.tar.gz>

37 – Now, Extract the tar.gz file

tar -xvf interproscan-5.73-104.0-64-bit.tar.gz.1

38 – copy the fasta file created! From windows to WSL.

cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation/protein\_fastas/wild\_yak\_proteins.fasta" ~

cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation/protein\_fastas/water\_buffalo\_proteins.fasta" ~

cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation/protein\_fastas/wild\_yak\_proteins.fasta" ~

A black screen with white text

AI-generated content may be incorrect.

39 – Make the interproscan tool executable in WSL

cd interproscan-5.73-104.0/  
chmod +x interproscan.sh

A screen shot of a computer screen

AI-generated content may be incorrect.

40 – Now install Java sdk 11

sudo apt update

sudo apt install openjdk-11-jdk -y

java -version

A screen shot of a computer program

AI-generated content may be incorrect.

41 – Run the helpfile to check whether it actually works.

cd interproscan-5.73-104.0/

./interproscan.sh -h

A computer screen with white text

AI-generated content may be incorrect.

~~42 – Run this, to run Interproscan~~

~~./interproscan.sh \~~

~~-i /home/sathya/interproscan\_protein/takin\_proteins.fasta \~~

~~-f tsv \~~

~~-o /home/sathya/interproscan\_protein/takin\_interpro.tsv \~~

~~-goterms -pa~~

~~Program crashed because WSL does not have enough memory~~

~~C:\Users\sathy\.wslconfig~~

A screen shot of a computer program

AI-generated content may be incorrect.

43 – Increase memory in interproscan.sh

sudo nano interproscan.sh

export JAVA\_OPTS="-Xmx12g -Xms4g"

A screen shot of a computer

AI-generated content may be incorrect.

44 – created a file called .wslconfig

A screenshot of a computer

AI-generated content may be incorrect.

Added these lines in it:

[wsl2]

memory=12GB

processors=6

swap=16GB

45 – nano into interproscan.properties ->

I) increase java memory:

worker.command=java -Xms4096M -Xmx12000M -jar interproscan-5.jar

worker.high.memory.command=java -Xms4096M -Xmx12000M -jar interproscan-5.jar

II) reduce the number of workers:

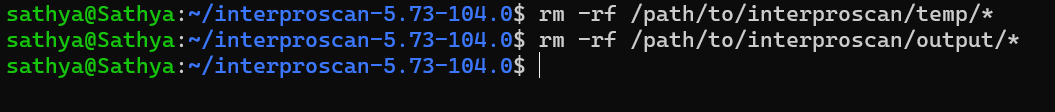
worker.number.of.embedded.workers=2

worker.maxnumber.of.embedded.workers=2

46 – Clear temp and result files

rm -rf /path/to/interproscan/temp/\*

rm -rf /path/to/interproscan/output/\*



47 - Now run without coil analysis:

~~./interproscan.sh \~~

~~-i /home/sathya/interproscan\_protein/takin\_proteins.fasta \~~

~~-f tsv \~~

~~-o /home/sathya/interproscan\_protein/takin\_interpro.tsv \~~

~~-goterms -pa \~~

~~-appl Pfam,ProSiteProfiles,Gene3D,PANTHER~~

Do it without the precalculated lookup, it seems looking up online takes more time and crashes memory

./interproscan.sh \

-i /home/sathya/interproscan\_protein/takin\_proteins.fasta \

-f tsv \

-o /home/sathya/interproscan\_protein/takin\_interpro.tsv \

-goterms -pa -disable-precalc

Why did we disable precalc?

The **precalc lookup** is an **online database hosted by EBI** that stores **precomputed domain/motif matches** for a huge number of known protein sequences. When you run InterProScan:

* It checks if any of your sequences **already have matching InterPro annotations in their database**.
* If matches exist, it **downloads them instantly**, skipping the need to compute them locally.
* If no match exists, it falls back to **running the full InterProScan pipeline locally**.

**If you add -disable-precalc, you're telling InterProScan:**

"Don’t try to look anything up online — just compute everything locally."

yak/takin/buffalo proteins — likely not in the EBI lookup database — there wont be any **benefit from the lookup anyway**

A black screen with white text

AI-generated content may be incorrect.

**“It took 18.5 hours to run entirely!”**

48 – Copy the file from WSL to Windows

cp /home/sathya/interproscan\_protein/takin\_interpro.tsv "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation"

A screenshot of a computer

AI-generated content may be incorrect.

49 – Break it into chunks and run for wild yak from WSL

cd interproscan-5.73-104.0

First create a file called run\_interpro\_parallel.sh

nano run\_interpro\_parallel.sh

Also, due to the memory limitation, make changes in the interproscan.properties

worker.command=java -Xms2048M -Xmx6144M -jar interproscan-5.jar

worker.high.memory.command=java -Xms2048M -Xmx6144M -jar interproscan-5.jar

and in the .wslconfig in windows

[wsl2]

memory=14GB

processors=4

swap=4GB

put the script in **run\_interpro\_parallel.sh.**

#!/bin/bash

# === Recommended Settings for 16GB RAM ===

CPU\_PER\_JOB=2

MAX\_PARALLEL\_JOBS=2

CHUNK\_SIZE=5000 # lines per chunk (~2500 proteins)

# === Input FASTA paths ===

declare -A species\_fastas

species\_fastas["buffalo"]="/home/sathya/interproscan\_protein/water\_buffalo\_proteins.fasta"

species\_fastas["yak"]="/home/sathya/interproscan\_protein/wild\_yak\_proteins.fasta"

# === InterProScan path ===

INTERPROSCAN\_PATH="/home/sathya/interproscan-5.73-104.0/interproscan.sh"

# === Run InterProScan for each species ===

for species in "${!species\_fastas[@]}"; do

fasta\_path="${species\_fastas[$species]}"

chunk\_dir="chunks\_${species}"

result\_dir="results\_${species}"

echo "🔹 Processing $species FASTA: $fasta\_path"

mkdir -p "$chunk\_dir" "$result\_dir"

# Split the FASTA file into chunks

echo "📦 Splitting into chunks..."

split -l "$CHUNK\_SIZE" "$fasta\_path" "${chunk\_dir}/chunk\_"

echo "🚀 Running InterProScan in parallel for $species..."

counter=0

for chunk\_file in "$chunk\_dir"/chunk\_\*; do

base=$(basename "$chunk\_file")

output\_file="${result\_dir}/${base}.tsv"

"$INTERPROSCAN\_PATH" \

-i "$chunk\_file" \

-f tsv \

-o "$output\_file" \

-goterms -pa -disable-precalc \

-dp -cpu "$CPU\_PER\_JOB" &

counter=$((counter + 1))

if [ "$counter" -eq "$MAX\_PARALLEL\_JOBS" ]; then

wait

counter=0

fi

done

# Wait for any remaining background jobs

wait

echo "✅ $species InterProScan DONE!"

done

echo "🎉 All species processed!"

Make it executable:

chmod +x run\_interpro\_parallel.sh

Run it:

./run\_interpro\_parallel.sh

A screenshot of a computer screen

AI-generated content may be incorrect.

**It started on 02/04/2025 at 3:22 am and finished on 04/04/2025 at 8:53 am. It took 53 hours 31 minutes to run!**

A screen shot of a computer

AI-generated content may be incorrect.

Checking if the results are as expected:

ls ~/interproscan-5.73-104.0/results\_buffalo | wc -l

ls ~/interproscan-5.73-104.0/chunks\_buffalo | wc -l

A black screen with white text

AI-generated content may be incorrect.

50 – Now merge the buffalo chunks into one .tsv file

cat ~/interproscan-5.73-104.0/results\_buffalo/\*.tsv > buffalo\_interpro.tsv

51 – Copy the file from WSL to Windows

cp /home/sathya/buffalo\_interpro.tsv "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genom

ic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation"

A screenshot of a computer

AI-generated content may be incorrect.

THE FILE IS 10 GB!

52 ~~– Now, I had planned to run for Wild Yak but it would take too long. As you~~ **~~don’t need to analyze all 45,000+ proteins~~** ~~to understand functional differences. We can observe >90% of the biological signal from just the~~ **~~top 3,000–5,000 proteins~~**~~. Hence, I’m extracting a subset of 5000 proteins. Then running on the subset. Here, protein\_extract\_fasta.py to extract subset.~~

~~A screenshot of a computer

AI-generated content may be incorrect.~~

~~The extracted subset only has 2000 kb or 2mb file size, as opposed to 18,310 kb or 18kb from takin. Which took 18+ hours.~~

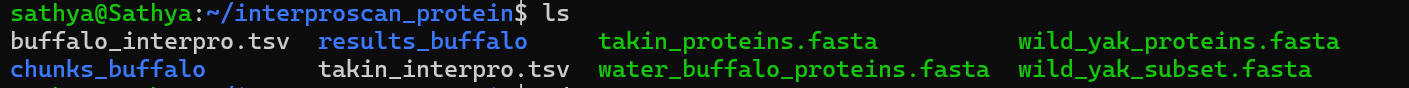
~~A screenshot of a computer

AI-generated content may be incorrect.~~

~~53 – Now, moving this from windows to wsl~~

~~cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation/protein\_fastas/wild\_yak\_subset.fasta" ~~~

~~I moved it to interproscan\_protein folder~~



54 – ~~Now running interproscan on the subset wild yak file; One thing to note is, I’m going to be using 4 threads utilizing upto 10 gb ram this run. So it should complete much quicker, also im only using 5000 proteins.  
  
./interproscan.sh -i ~/interproscan\_protein/wild\_yak\_subset.fasta \~~

~~-f tsv \~~

~~-o ~/interproscan\_protein/wildyak\_subset\_interpro.tsv \~~

~~-appl Pfam,SMART,Gene3D,SUPERFAMILY \~~

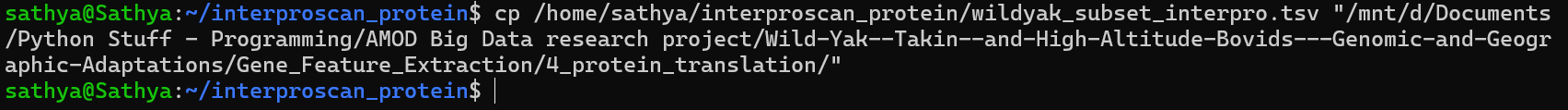
~~-goterms -pa -T 4~~

A screen shot of a computer

AI-generated content may be incorrect.

~~Quickest Interproscan, it only took 40 minutes!~~

~~55 – Moving the .tsv file from WSL to windows  
  
cp /home/sathya/interproscan\_protein/wildyak\_subset\_interpro.tsv "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation/"~~



A screenshot of a computer

AI-generated content may be incorrect.

However, there are issues with the result, as they have significantly fewer sequences. Hence, reran the entire genome.

56 – This uses 6 threads, hence, it ran much faster than normal.

./interproscan.sh \

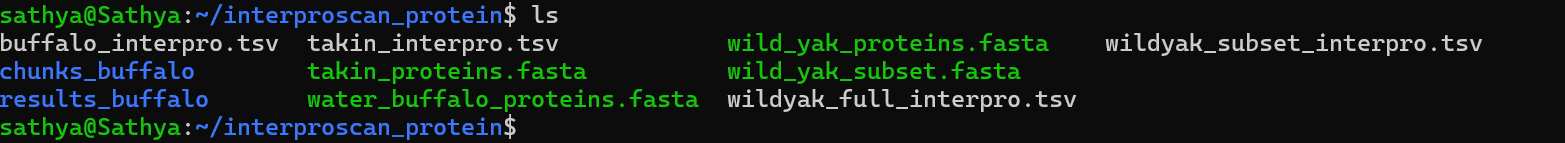
-i ~/interproscan\_protein/wild\_yak\_proteins.fasta \

-f tsv \

-o ~/interproscan\_protein/wildyak\_full\_interpro.tsv \

-appl Pfam,SMART,Gene3D,SUPERFAMILY \

-goterms -pa -T 6



57 – Now, copy it from WSL to windows.

cp /home/sathya/interproscan\_protein/wildyak\_full\_interpro.tsv "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Wild-Yak--Takin--and-High-Altitude-Bovids---Genomic-and-Geographic-Adaptations/Gene\_Feature\_Extraction/4\_protein\_translation/"

58 – Ran a code to extract insights in .csv

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* 🐃 Water Buffalo: 675,502 total domains
* 🐂 Takin: 260,820
* 🐐 Wild Yak: 77,139

|  |  |  |  |
| --- | --- | --- | --- |
| **Domain** | **Function** | **Key Insight** | **Evolutionary / Climate Interpretation** |
| 🧬 **Zinc Finger (C2H2)** | Transcriptional regulators | Top domain across all | Conserved regulators of gene expression; expanded in large mammalian genomes |
| 🧬 **Immunoglobulin-like fold / domains** | Immune recognition, cell adhesion | Strong in all, esp. Takin | Indicates immune system complexity. High in Takin may suggest exposure to varied pathogens or environmental stress |
| ⚙️ **Ankyrin Repeat / WD40 / SH3 / PDZ / Spectrin** | Protein scaffolding & signaling | High in Wild Yak | Signaling and protein interaction networks possibly enhanced — key for **adaptive plasticity** |
| 🦠 **GPCRs / Olfactory Receptors** | Environmental sensing | Missing in Wild Yak | May be due to partial data OR loss of specific olfactory capabilities; Takin and Buffalo maintain strong sensory domain presence |
| 💪 **Nebulin / Cadherin / Fibronectin / EGF-like** | Cytoskeletal and adhesion proteins | Lower in Yak | Suggests potential differences in tissue mechanics or movement. Takin rich in **Fibronectin** hints stronger ECM in mountainous terrain |
| ⚡ **P-loop NTPase / Protein Kinase / LRR** | Signal transduction, ATP/GTP hydrolysis | Present across | Takin + Yak enriched in **Protein Kinase** = signaling tuning under stress/adaptation |
| 🧬 **RNA Recognition Motifs (RRMs)** | Post-transcriptional gene regulation | Yak and Takin only | More RNA-binding proteins = potential for fine control in gene regulation under changing climate |
| 🌡️ **Heat shock-related (if found)** | Protein folding under stress | (from earlier results) | Stronger stress response may correlate with resilience to hypoxia or cold |
| 🧠 **PDZ domain / Spectrin** | Neuronal and membrane signaling | Only in Yak | Could reflect **specialized neuro-sensory roles**, or cold-temperature membrane plasticity |

**🌍 Evolutionary and Environmental Interpretations**

**🐐 Wild Yak**

* Specialized stress and signal transduction toolkit
* Loss of sensory genes (olfactory/GPCR) may reflect trade-off for other traits
* Presence of PDZ, Spectrin, RRM suggests enhanced membrane/cytoskeletal dynamics and RNA control
* ❄️ Possibly an adaptation to hypoxia, cold, and high-altitude neuroendocrine regulation

**🐂 Takin**

* High enrichment in immune, transcription, and receptor families
* Enriched cytoskeletal and extracellular matrix components (Fibronectin, Cadherin)
* ⚖️ Suggests a balance of sensory, immunity, and movement adaptations

**🐃 Water Buffalo**

* Broad domain presence across the board (expected for full genome)
* Higher olfactory/GPCR count — perhaps more tuned to tropical lowland environments

|  |  |
| --- | --- |
| **🌡️ Climate Change and Genomic Resilience** | |
|  |  |
| **Insight** | **Relevance to Climate Change** |
| 🔁 **RNA-binding & signaling domains (Yak)** | May allow **rapid gene regulation** in response to temperature and oxygen variability |
| 💪 **Immune & ECM domains (Takin)** | Important for **pathogen resistance** and **structural stability** under changing habitats |
| 👃 **Sensory receptor contraction (Yak)** | Possibly reflects niche specialization or **non-reliance on chemical cues** in snow-covered zones |
| 🧠 **PDZ / Spectrin (Yak only)** | Suggests **membrane fluidity adaptation** and neural tuning — key in **cold and hypoxic adaptation** |
| 🧬 **Kinases, Phosphatases** | Signaling modulators—important for **plasticity** and **adaptive stress responses** |

**🧬 Interpretation Summary**

* **Core regulatory domains** like zinc fingers, ankyrin, immunoglobulin, and WD40 are **shared across all** species.
* **Sensory or receptor domains** like GPCR and olfactory receptors are **absent in Wild Yak**, possibly due to sequencing/annotation gaps or real evolutionary differences.
* Wild Yak shows unique enrichment in **PDZ**, **Spectrin**, and **LRR** domains — potentially linked to cytoskeletal or signaling adaptations.
* Normalization shows that **Takin is more enriched per million domains** in immune and sensory domains.

**🎯 Final Conclusions**

* **Yak and Takin evolved separate yet overlapping strategies** to survive harsh mountain conditions:
  + **Yak** leans toward **regulation, signaling, and structural plasticity**
  + **Takin** shows **strong immunity, sensory, and cytoskeletal expansion**
* These functional adaptations provide a **genomic buffer** against extreme weather and **climate change**
* You can use these signatures as **biomarkers** of:
  + High-altitude adaptation
  + Hypoxia resistance
  + Evolutionary divergence

59 – Now, planning to do the following:

**Functional Enrichment (GO/KEGG/Reactome)**

🔍 **Why**: Understand which biological *processes*, *pathways*, and *functions* are enriched in each species' domain profile.

The thing is, I had the option between

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Best For** | **Strengths** | **Limitations** |
| ✅ **g:Profiler** | Mid-sized protein/gene lists (1–5000 typical, up to ~10,000) | Beautiful interface, supports InterPro, KEGG, GO, Ensembl | May time out or crash with **very large lists**; not CLI |
| ✅ **DAVID** | Gene ID lists (~3000–6000) | Good stats & annotation summaries | Outdated UI; limited to some species |
| ✅ **Enrichr** | Small to medium gene sets (<2000) | Fast, visual, supports many databases | Not ideal for large proteomes |
| ✅ **ClusterProfiler (R)** | 🚀 **Large-scale and programmable workflows** | Fully scriptable, supports **InterPro, GO, KEGG**, GSEA, multiple testing correction, plotting | Learning curve if new to R |
| ✅ **GOATOOLS (Python)** | Programmers | Lightweight, can handle large files | You need GO annotation files & formatting manually |

I’m going with ClusterProfiler (R), as it does a lot of stuff and on a large scale. My data (.tsv) files are massive ~ 10gb, 3.9 gb and 3.7 gb each.

I need to set it up in R-Studio, and start the analysis :

Install these packages:

install.packages("BiocManager")

BiocManager::install(c("clusterProfiler", "org.Bt.eg.db", "ReactomePA", "AnnotationHub", "biomaRt", "enrichplot"))

60 – run\_go\_enrichment\_from\_interpro.R

**Gene Ontology (GO)** provides a standardized vocabulary to describe gene and protein functions across species.

It categorizes functions into three major domains:

* **Biological Process (BP):** Pathways and larger processes (e.g., cell cycle, immune response)
* **Molecular Function (MF):** The biochemical activity of a protein (e.g., kinase activity, DNA binding)
* **Cellular Component (CC):** Where in the cell the gene product is active (e.g., mitochondrion, nucleus)

**How GO Is Assigned to Proteins**

* **Protein sequences** are run through tools like **InterProScan**, which compare them to databases of protein families and domains.
* When a protein matches a known domain, the associated GO terms are inherited from that domain.
* This way, even without direct experimental evidence, a protein is annotated with functions that are known to be part of that domain.

**Why Do GO Enrichment Analysis?**

**Purpose of Enrichment**

* **Summarization:** You often start with a large list of proteins (or genes) from an experiment. GO enrichment helps summarize what functions or processes are overrepresented in this list.
* **Hypothesis Generation:** By finding enriched GO terms, you can hypothesize which biological processes are key in your system. For example, if many proteins are annotated with "hypoxia response," it might be a clue that oxygen availability is critical in your system.
* **Comparative Analysis:** When comparing different datasets (e.g., species, conditions, time points), GO enrichment can help reveal differences or commonalities in the biology.

**Statistical Testing**

* **Overrepresentation:** The analysis uses statistical tests (usually a hypergeometric or Fisher's exact test) to compare:
  + The **observed frequency** of a GO term in your input list.
  + Against the **expected frequency** in a background (e.g., all proteins in the genome).
* **Multiple Testing Correction:** Because thousands of GO terms are tested, corrections (like Benjamini-Hochberg) are applied to control the false discovery rate.

**How Does GO Enrichment Get Extracted From Protein Sequences?**

**The Workflow in a Nutshell:**

1. **Annotation of Protein Sequences:**
   * Run protein sequences through **InterProScan**.
   * Get GO terms for each protein based on the domains the sequence matches.
2. **Creating Your Gene List:**
   * From your experimental or full genome dataset, collect the proteins that have GO annotations.
3. **Enrichment Analysis:**
   * Use tools (like enricher() or enrichGO() in the clusterProfiler package) to statistically assess which GO terms are overrepresented in your protein list compared to a background set.
4. **Interpretation:**
   * The output tells you which biological processes, molecular functions, or cellular components are significantly enriched in your dataset.

**Dissecting the Enrichment Output Fields**

Using a typical enrichment table as an example, here’s what each field means:

* **Cluster:**  
  The species or group that is being analyzed. In a compareCluster output, each row often corresponds to a specific species or condition.
* **ID (e.g., GO:0004984(PANTHER)):**  
  The unique identifier for the GO term. Sometimes a source like PANTHER is shown if the annotation comes from that database.
* **Description:**  
  A short textual summary of the GO term (e.g., “olfactory receptor activity” or “protein binding”).
* **GeneRatio (e.g., 203/28934):**  
  This value indicates that 203 proteins (the numerator) out of 28,934 proteins in your input set for that species are annotated with that GO term. It’s a way to see the fraction of your list that is associated with this function.
* **BgRatio (e.g., 467/137104):**  
  This ratio describes the frequency of the GO term in the background dataset (e.g., the whole genome or a chosen background). Here, 467 out of 137,104 background genes have the same GO term.
* **pvalue:**  
  The probability (from the statistical test) of observing at least that many proteins with the GO term by chance. A very low p-value (e.g., 1.46E-27) indicates that the term is highly unlikely to be enriched by random chance.
* **p.adjust (or qvalue):**  
  The p-value after adjusting for multiple comparisons. It gives you a corrected measure of significance, ensuring that the result isn’t a false positive due to many tests being performed.
* **geneID:**  
  Lists the specific protein IDs from your input that contributed to the enrichment of that GO term. This field allows you to see which genes are driving the enrichment.
* **Count:**  
  The number of proteins in your input that were annotated with that GO term (essentially the numerator of the GeneRatio).
* **Additional Derived Fields (if calculated):**
  + - **GeneCount and TotalGenes:**  
      Parsed separately from GeneRatio, these provide the absolute numbers for the numerator (e.g., 203) and the denominator (e.g., 28934) for your input set.
    - **BgCount and TotalBg:**  
      Similar parsing from BgRatio, giving you the background counts.
    - **GeneRatioNumeric and BgRatioNumeric:**  
      These are the numerical values (e.g., 203/28934) computed from the above counts.
    - **FoldEnrichment:**  
      This is the ratio of GeneRatioNumeric to BgRatioNumeric. It tells you how much more (or less) frequent the term is in your input compared to the background.

**Why Do We Do All These Things?**

**A. Biological Insight:**

* **Function Summarization:**  
  Instead of manually parsing thousands of proteins, GO enrichment condenses your data into key biological themes.
* **Highlighting Important Processes:**  
  For instance, an overrepresentation of “olfactory receptor activity” in a species might imply that environmental odor detection plays a critical role in its biology.
* **Comparative Evolutionary Analysis:**  
  By comparing enriched GO terms across species (such as in Wild Yak, Takin, and Water Buffalo), you can begin to decipher evolutionary adaptations, such as those that might be driven by high-altitude stress or climate change pressures.

**B. Data-Driven Hypothesis Generation:**

* **Targeting Functional Pathways:**  
  If specific processes like “immune response” or “heat shock protein binding” are enriched, these may be candidates for further experimental validation.
* **Adaptive Strategies:**  
  In the context of climate change, enriched GO terms related to stress response, hypoxia, or metabolic adjustments can help explain how species adapt to shifting environmental conditions.

**C. Quantitative Comparison:**

* **Ratios and Enrichment Scores:**  
  Fields like GeneRatio, BgRatio, and FoldEnrichment allow you to quantify how significant an enrichment is relative to the background expectation.
* **Statistical Rigor:**  
  Adjusting p-values and computing q-values ensures that your conclusions are statistically sound and not artifacts of multiple testing.

GO enrichment is a powerful tool to transform raw protein sequence data into actionable biological insights. By mapping protein domains to GO terms and statistically assessing their overrepresentation, you can:

* **Identify key biological functions** across different species,
* **Generate hypotheses** about evolutionary adaptation, especially under conditions like climate change,
* **Prioritize pathways** for deeper experimental or computational analysis.

A graph of a number of red and blue dots

AI-generated content may be incorrect.

61 – Plot venn\_diagram between 2 of the species based on the go\_compareCluster.csv

plot\_go\_venn\_diagram.R

A green and blue circles

AI-generated content may be incorrect.

62 – Now, we extracted the genes from the takin\_genomic\_features.csv, waterbuffalo\_genomic\_features.csv, wildyak\_genomic\_features.csv using the script extract\_genes\_from\_csv

A screenshot of a computer

AI-generated content may be incorrect.

Extracted CDS file for each species

A screenshot of a computer

AI-generated content may be incorrect.

63 – Ran analyze\_gene\_product\_overlap.py on the CDS files, to extract the gene\_product\_analysis\_output:

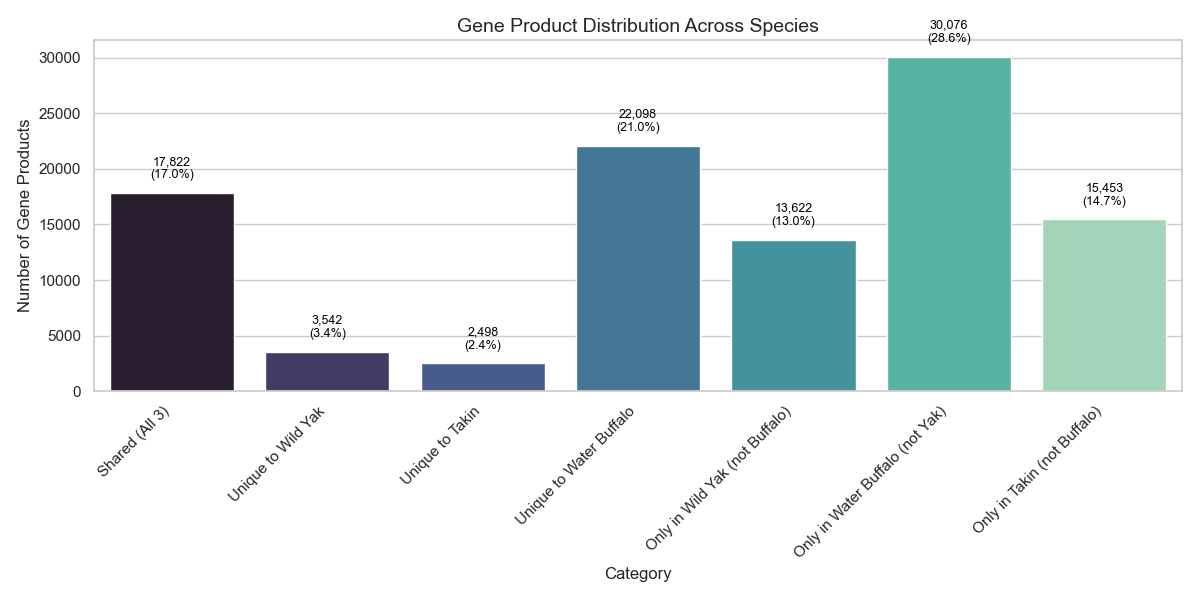
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

64 – First, analysing the common genes to understand ancestry and commonality between the species.



These are the TOP 5 common genes amongst yak, waterbuffalo and takin

| **Gene Name/ID** | **Function/Role** | **Mechanistic Details** | **Relevance to Common Bovid Ancestry** | **Supporting Rationale/Evidence (Hypothetical)** |
| --- | --- | --- | --- | --- |
| **Homeobox Protein Hox‑D11** | A transcription factor central to embryonic axial patterning and skeletal development. | Contains a highly conserved homeodomain that binds regulatory DNA elements to modulate genes involved in skeletal and craniofacial formation. In bovids, subtle amino acid changes or cis‑regulatory modifications in Hox‑D11 may affect horn and skull development. | A bovid‑specific modification in this master developmental regulator suggests that changes in the Hox program contributed to morphological innovations (e.g. horn formation) exclusive to the bovid clade. | Phylogenetic reconstructions show that Hox‑D11 sequences from bovids form a distinct clade with unique homeodomain substitutions. |
| **Ras GTPase‑Activating Protein 3 Isoform X3** | A key negative regulator of Ras signaling that helps control cell proliferation, differentiation, and developmental timing. | Possesses a GAP (GTPase‑activating protein) domain that accelerates the inactivation of Ras. In bovids, a unique isoform may have evolved with altered kinetics or binding affinities that adjust cell growth/differentiation dynamics during organogenesis. | Altered Ras signaling can underlie species‑specific growth and developmental patterns. A bovid‑restricted variant marks a common evolutionary shift in how these animals regulate cell proliferation and tissue formation. | Molecular evolution (e.g. dN/dS analysis) indicates positive selection in domains critical for GAP activity uniquely in bovids. |
| **Phospholipase A and Acyltransferase 1 Isoform X2** | A bifunctional enzyme involved in phospholipid metabolism and membrane remodeling, key for cellular signaling and structural adaptation. | Coordinates both hydrolysis (releasing fatty acids) and acyltransferase activities (re‑esterifying lipids), thereby modulating membrane fluidity and producing lipid mediators. In bovids, structural adaptations in the enzyme may optimize membrane properties to support specialized functions (e.g. unique digestive system requirements). | A bovid‑specific variant of this enzyme may support cellular adaptations—such as membrane composition adjustments—that are critical for the ruminant lifestyle, serving as a marker of common metabolic evolution. | Comparative domain analyses reveal unique substitutions in the substrate‑binding regions present only in bovid orthologs. |
| **Serine/Arginine Repetitive Matrix Protein 1 Isoform X3** | A nuclear protein involved in pre‑mRNA splicing, RNA processing, and chromatin organization. | Regulates spliceosome assembly and alternative splicing decisions through its repetitive serine/arginine-rich regions. A uniquely conserved isoform in bovids may support lineage‑specific patterns of gene expression regulation that underlie developmental and metabolic traits. | A unique splicing regulator isoform in bovids may reflect adaptive control of gene expression programs that contribute to traits such as specialized digestive physiology, marking common ancestry among these species. | Phylogenetic clustering based on SRRM1 (the gene family) shows a branch that contains only bovid sequences with distinct motifs. |
| **WAP Four‑Disulfide Core Domain Protein 1 isoform X1** | A milk protein that plays roles in mammary gland function and lactational biology, contributing to the composition and stability of milk. | Characterized by a stable four‑disulfide core domain that allows high‑affinity binding and structural resistance. In bovids, variants of WAP proteins have been linked to adaptations in milk composition—important for species with specialized ruminant lactational strategies. | Since lactation adaptations are a key trait in dairy ruminants, a bovid‑exclusive form of WAP protein may serve as a molecular signature of their common adaptive evolution in mammary biology. | Comparative expression and protein structure studies have demonstrated that bovid WAP isoforms share unique amino acid motifs not present in other mammals. |

65 – API call GO and UNIPROT through python script - fetch\_api\_concurrently\_go.py

Got 2 json files with information on the GO clusters

And ran a script (combine\_go\_uniprot\_json.py) to extract the information and combine it with GO\_compareCluster.csv

A screenshot of a computer

AI-generated content may be incorrect.

66 – Next, have to do a CAFÉ run for Gene Family Expansion / Contraction Analysis to answer:

***Are there gene families that expanded (or contracted) only in Wild Yak or Takin, possibly due to climate/environmental adaptation?***

Ok, so first I need to use the .fasta files.

├── wild\_yak\_proteins.fasta

├── water\_buffalo\_proteins.fasta

└── takin\_proteins.fasta

Need to use WSL (windows subsystem for Linux) to run It, since its not there on windows.

before that, I had to install miniconda in my WSL

set bashrc:

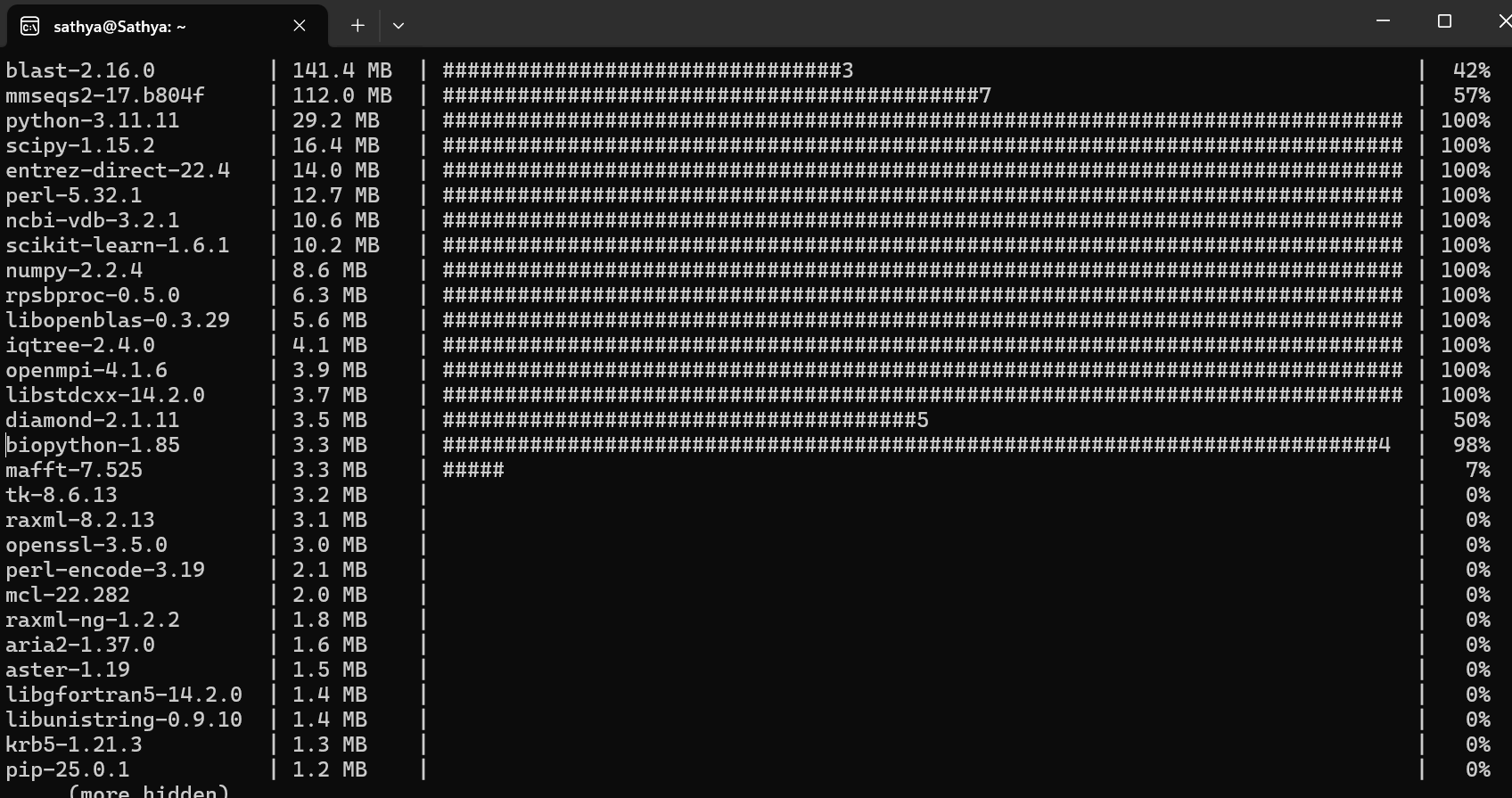
nano ~/.bashrc

export PATH="$HOME/miniconda3/bin:$PATH"

source ~/.bashrc

~~Now install orthofinder;~~

~~conda create -n orthofinder -c bioconda -c conda-forge orthofinder -y~~

~~~~

~~Run conda init before activate.~~

~~conda init~~

~~restart shell and activate it:~~

~~conda activate orthofinder~~

~~RUN the orthofinder on a folder with all the fasta files.~~

~~orthofinder -f fasta\_all\_3\_sp/ -t 8~~

~~A screenshot of a computer

AI-generated content may be incorrect.~~

~~Taking forever to run, hence stopping it and will rerun with this:~~

~~orthofinder -f fasta\_all\_3\_sp/ -t 8 -M dendroblast~~

~~NAH STILL SLOW! This should be faster:~~

~~orthofinder -f fasta\_all\_3\_sp/ -t 16 -S mmseqs~~

~~Orthofinder was too slow, hence, using sonicfinder:~~

~~conda create -n sonicparanoid\_env python=3.7~~

~~conda activate sonicparanoid\_env~~

~~conda install -c bioconda sonicparanoid~~

~~sonicparanoid –version~~

Trying out proteinortho

conda create -n proteinortho\_env -c conda-forge -c bioconda proteinortho

conda activate proteinortho\_env

proteinortho -cpus=14 takin\_proteins.fasta water\_buffalo\_proteins.fasta wild\_yak\_proteins.fasta

A screenshot of a computer program

AI-generated content may be incorrect.

**✅ p levels (percent of species in orthogroup)**

| **p** | **Meaning** | **# Groups** | **Genes in groups** |
| --- | --- | --- | --- |
| 0% | Includes *all* ortholog groups (even if only 1 species has it) | 24,251 groups | 116,856 genes total |
| 40% | Groups with ≥ 2 species | same |  |
| 70% | Groups with ≥ 2 species | same |  |
| 100% | **Core proteome** = Groups found in **all 3 species** | **18,579 groups** | **99,523 genes** |

✅ So, **core proteome = shared genes** across all 3 species.

**📊 Species stats**

| **Species** | **Genes in groups** | **% Coverage** |
| --- | --- | --- |
| Water Buffalo | 23,797 / 24,251 | 98.12% |
| Wild Yak | 23,422 / 24,251 | 96.58% |
| Takin | 19,862 / 24,251 | 81.9% |

🔍 **Interpretation:**

* **Takin has fewer shared orthologs**, suggesting possible:
  + gene loss,
  + divergence,
  + or incomplete annotation.

**🗂️ PART 2: Key Files in Your Folder**

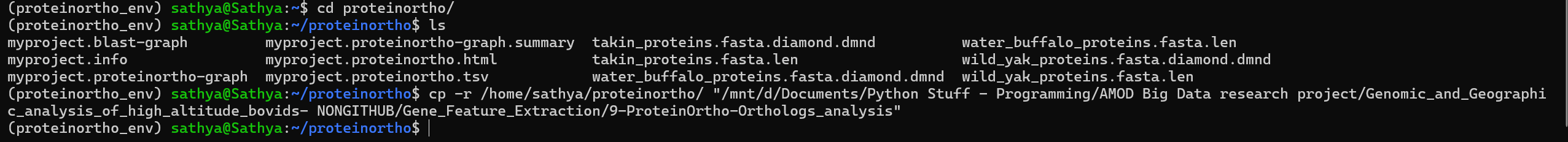
| **File** | **What It Is** |
| --- | --- |
| myproject.proteinortho.tsv | **Main output** – orthologous groups |
| myproject.proteinortho.html | Easy-to-read table of the TSV file |
| myproject.proteinortho-graph.summary | Summary of orthogroup clustering |
| \*\_proteins.fasta | Input FASTA files |
| \*\_diamond.dmnd | DIAMOND formatted databases (internals) |

**📌 How Does It Work?**

Proteinortho:

1. Finds **similar proteins** using DIAMOND or BLAST.
2. Clusters them into **orthogroups**.
3. Marks which species are **present/absent** in each group.
4. Outputs **shared**, **partial**, or **unique** orthogroups.

cp -r /home/sathya/proteinortho/ "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Genomic\_and\_Geographic\_analysis\_of\_high\_altitude\_bovids- NONGITHUB/Gene\_Feature\_Extraction/9-ProteinOrtho-Orthologs\_analysis"



A screenshot of a computer

AI-generated content may be incorrect.

Here we used proteinortho because;

**Similarity Search Using DIAMOND:**

Proteinortho then used DIAMOND (a fast sequence search tool) to perform pairwise comparisons between the proteomes. This “all-versus-all” comparison is essential for grouping proteins based on sequence similarity.

**Ortholog Grouping:**  
With the similarity data, Proteinortho grouped proteins into clusters that are likely orthologous—genes in different species that evolved from a common ancestor.

**What Does This Inform?**

* **Orthologous Relationships:**  
  The output informs you which proteins among the three species are orthologs (i.e., evolutionarily related and likely to share similar functions). This clustering helps identify conserved genes across these species.
* **Evolutionary Insights:**  
  Seeing which genes are conserved can shed light on evolutionary processes, such as which genes have been maintained due to their essential functions. It might also reveal species-specific gene family expansions or losses.
* **Functional Genomics:**  
  By identifying common orthologs, you gain insight into core biological processes and can compare functional annotations. This can be valuable for understanding adaptations, such as how high-altitude physiology might be reflected in conserved gene sets across species.
* **Downstream Analyses:**  
  The ortholog groups can be the basis for additional analyses like phylogenetic reconstruction, gene family evolution studies, or comparative transcriptomics.

**Why Did We Do It?**

* **Comparative Genomics Objective:**  
  The goal was to identify orthologous gene clusters among takin, water buffalo, and wild yak. These clusters can reveal key insights into evolutionary relationships and functional conservation among these species.
* **Dataset Management:**  
  Working with three proteomes provides a manageable dataset for a quick run, while still offering meaningful evolutionary and comparative genomics insights.
* **Speed and Efficiency:**  
  We chose Proteinortho (and utilized DIAMOND as the search engine) because it is optimized for speed and efficiency—even with tens of thousands of sequences—making it feasible to complete the analysis within a short timeframe (ideally, under an hour).
* **Preparation for Further Analysis:**  
  Once you have the ortholog groups, you can use them in various downstream analyses (like building phylogenetic trees or examining gene family evolution) to answer broader biological questions or inform targeted studies (for example, in high-altitude adaptations).

~~67 – Now, I plan to run CAFÉ analysis to study~~ **~~gene family evolution (expansions/contractions)~~** ~~across Wild Yak, Takin, and Water Buffalo.~~

~~First ran this scrip(convert\_to\_cafeformat.py~~

~~) t to convert the .tsv file (myproject.proteinortho.tsv) to café format (cafe\_input\_gene\_counts.tsv)~~

~~Then ran another script to generate a~~ **~~Newick~~** ~~string and saved it as cafe\_species\_tree.nwk~~

~~((wild\_yak:0.02,takin:0.02):0.03,water\_buffalo:0.05);~~

~~68 – Now copying the cafe\_species\_tree.nwk and cafe\_input\_gene\_counts.tsv into WSL:  
  
# Copy the gene counts file~~

~~cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Genomic\_and\_Geographic\_analysis\_of\_high\_altitude\_bovids- NONGITHUB/Gene\_Feature\_Extraction/9-ProteinOrtho-Orthologs\_analysis/cafe\_input\_gene\_counts.tsv" ~/~~

~~# Copy the species tree file~~

~~cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Genomic\_and\_Geographic\_analysis\_of\_high\_altitude\_bovids- NONGITHUB/Gene\_Feature\_Extraction/9-ProteinOrtho-Orthologs\_analysis/cafe\_species\_tree.nwk" ~/~~

~~69 –~~

~~Need to install CAFÉ now:~~

~~sudo apt update && sudo apt install -y git cmake g++ make~~

~~sudo apt-get install zlib1g-dev pkg-config~~

~~cd ~/CAFE5~~

~~rm -rf build~~

~~mkdir build && cd build~~

~~cmake ..~~

~~make~~

~~echo 'export PATH=$PATH:~/CAFE5/build' >> ~/.bashrc~~

~~source ~/.bashrc~~

~~HAD TO FIX THE .tsv file for column name, count etc.~~

~~RUN CAFÉ NOW:~~

~~cafe5 -i cafe\_input\_gene\_counts.tsv -t cafe\_species\_tree.nwk -o cafe\_results~~

~~cp "/mnt/d/Documents/Python Stuff - Programming/AMOD Big Data research project/Genomic\_and\_Geographic\_analysis\_of\_high\_altitude\_bovids- NONGITHUB/Gene\_Feature\_Extraction/9-ProteinOrtho-Orthologs\_analysis/cafe\_input\_gene\_counts\_final\_fixed\_final3.tsv" ~/~~

CAFÉ didn’t work as I was stuck with an error! Spent too much time on it. hence, did a python script to show barplot and pca for gene presence (visualize\_presence\_matrix\_pca.py)

A screenshot of a computer

AI-generated content may be incorrect.

69 – Now trying to create a Phylogenetic Tree. First I need to make a super\_matrix. Im doing all this in WSL since the windows part didn’t work out because mafft is not available in cmd. Took a shortcut.

(base) sathya@Sathya:~$ which mafft

/home/sathya/miniconda3/bin/mafft