Lab 3

`mkdir ~/lab03-$MYGIT/GLB1`

- Creates a new directory named `GLB1` inside the path `~/lab03-$MYGIT` to organize files related to the GLB1 study.

`ncbi-acc-download -F fasta -m protein "NP_000395.3"`

- Downloads the protein sequence corresponding to the NCBI accession number `NP_000395.3` in FASTA format using the `ncbi-acc-download` tool.

`blastp -db ../allprotein.fas -query NP_000395.3.fa -outfmt 0 -max_hsps 1 -out GLB1.blastp.typical.out`

- Performs a protein BLAST (`blastp`) search using the input sequence `NP_000395.3.fa` against the protein database `../allprotein.fas`. The results are saved in a typical BLAST output format (`outfmt 0`) with only one high-scoring pair (`-max_hsps 1`) per match in the file `GLB1.blastp.typical.out`.

`less GLB1.blastp.typical.out`

- Opens the BLAST result file `GLB1.blastp.typical.out` for viewing one screen at a time.

`blastp -db ../allprotein.fas -query NP_000395.3.fa -outfmt "6 sseqid pident length mismatch gapopen evalue bitscore pident stitle" -max_hsps 1 -out GLB1.blastp.detail.out`

- Performs a `blastp` search with a custom tabular output format (`outfmt 6`) that includes details such as subject sequence ID, percentage identity, alignment length, mismatches, gaps, e-value, bit score, and sequence title. Results are saved in `GLB1.blastp.detail.out`.

```
less -s GLB1.blastp.detail.out
```

- Opens the BLAST detailed result file `GLB1.blastp.detail.out` for viewing, suppressing repeated blank lines (`-s`).

```
awk '{if (6<1e-30)print6<1e-30)print1 }' GLB1.blastp.detail.out > GLB1.blastp.detail.filtered.out
```

- Filters the detailed BLAST output file `GLB1.blastp.detail.out` using `awk` to retain lines where the e-value (column 6) is less than `1e-30`. Saves the filtered results in `GLB1.blastp.detail.filtered.out`.

```
wc -l GLB1.blastp.detail.filtered.out
```

- Counts the number of lines in the filtered file `GLB1.blastp.detail.filtered.out`. This provides the number of hits meeting the specified e-value threshold.

```
grep -o -E "[1].[a-z]+" GLB1.blastp.detail.filtered.out | sort | uniq -c
```

- Extracts specific patterns (`[1].[a-z]+`) from `GLB1.blastp.detail.filtered.out`, sorts them, and counts unique occurrences. This summarizes specific data fields in the filtered BLAST results.

Lab 4

```
muscle -align ~/lab03-$MYGIT/GLB1/NP_000395.3.fa -output ~/lab04-$MYGIT/GLB1/GLB1.homologs.al.fas
```

- Aligns the protein sequence from `NP_000395.3.fa` using MUSCLE and saves the multiple sequence alignment output in FASTA format to `~/lab04-$MYGIT/GLB1/GLB1.homologs.al.fas`.

```
Rscript --vanilla ~/lab04-$MYGIT/plotMSA.R
~/lab04-$MYGIT/GLB1/GLB1.homologs.al.fas
```

- Executes an R script (`plotMSA.R`) without environment variables (`--vanilla`) to visualize the multiple sequence alignment saved in `~/lab04-$MYGIT/GLB1/GLB1.homologs.al.fas`. The script likely generates and saves the alignment visualization as a PDF.

```
alignbuddy -al ~/lab04-$MYGIT/GLB1/GLB1.homologs.al.fas
```

- Uses the AlignBuddy tool to load the multiple sequence alignment file `GLB1.homologs.al.fas` located in `~/lab04-$MYGIT/GLB1/`. The `-al` flag indicates an operation related to alignment manipulation, analysis, or format conversion, depending on the context.

```
awk '{for(i=1;i<=length($0);i++){if(substr($0,i,1)=="-") g[i]++}}
END{for(i=1;i<=length(g);i++){if(g[i]>0) count++} print count}'
~/lab04-s1-tfiorillo/GLB1/GLB1.homologs.al.fas
```

- Iterate through the alignment file (`GLB1.homologs.al.fas`).
- For each position in the sequence, count how many times a gap (`"-"`) appears in the column of sequences.
- The `g[i]++` counts the gaps per position, and then `count++` counts how many columns have at least one gap.
- The output is the number of columns (positions) with at least one gap.

```
awk 'NR==1{n=length($0); next} {for(i=1;i<=n;i++){if(substr($0,i,1) !=
"-"){cols[i]=substr($0,i,1)}}} END{for(i=1;i<=n;i++){if(cols[i] != ""
&& !seen[cols[i]]++){count++}} print count}'
~/lab04-s1-tfiorillo/GLB1/GLB1.homologs.al.fas
```

- This `awk` command processes the alignment file (`GLB1.homologs.al.fas`) by:
    - Extracting the length of the first sequence (`NR==1{n=length($0); next}`).
    - Iterating through each column of the alignment for every sequence, and storing the non-gap characters (`substr($0,i,1) != "-"`).

    - In the `END` block, it counts how many unique characters (excluding gaps) are present in each column of the alignment across all sequences, incrementing `count` each time a new, unique character is found.
    - The output is the number of columns that contain at least one unique character, excluding gaps.

```
t_coffee -other_pg seq_reformat -in
~/lab04-s1-tfiorillo/GLB1/GLB1.homologs.al.fas -output sim
```

- This `t_coffee` command reformats the input sequence alignment file (`GLB1.homologs.al.fas`) using the `seq_reformat` function. The `-output sim` option specifies that the output should be in a similarity matrix format, which provides pairwise similarity scores between the sequences in the alignment.

```
alignbuddy -pi ~/lab04-s1-tfiorillo/GLB1/GLB1.homologs.al.fas | awk '
(NR>2) { for (i=2;i<=NF;i++){ sum+=$i; num++ } } END{
print(100*sum/num) }'
```

- **`alignbuddy -pi ~/lab04-s1-tfiorillo/GLB1/GLB1.homologs.al.fas`: Runs `alignbuddy` with the `-pi` flag, which likely produces a pairwise identity matrix or a related output from the sequence alignment (`GLB1.homologs.al.fas`).**
- **`awk ' (NR>2) { for (i=2;i<=NF;i++){ sum+=$i; num++ } } END{ print(100*sum/num) }'`: This `awk` command processes the output of `alignbuddy`, skipping the first two lines (`NR>2`), then iterating through each column (from column 2 onwards) to sum up the values and count how many values are processed. Finally, it calculates the average of these values and multiplies by 100 to express the result as a percentage.**
- **Purpose: The final output is the average identity percentage (or another measure of similarity) based on the pairwise comparison data produced by `alignbuddy`.**

Lab 5

```
mkdir ~/lab05-$MYGIT/GLB1
```

- Creates a directory named `GLB1` in the `~/lab05-$MYGIT/` path for organizing files related to the current analysis.

```
cd ~/lab05-$MYGIT/GLB1
```

- Changes the working directory to `~/lab05-$MYGIT/GLB1`, where the subsequent analysis will take place.

```
sed 's/ /_/g' ~/lab04-$MYGIT/GLB1/GLB1.homologs.al.fas | seqkit grep -v -r -p "dupelabel" > ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.fas
```

- Uses `sed` to replace spaces with underscores in the sequence file `GLB1.homologs.al.fas` and pipes it to `seqkit` to remove sequences containing the string `"dupelabel"`. The processed file is saved as `GLB1.homologsf.al.fas` in the working directory.

```
iqtree -s ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.fas -bb 1000 -nt 2
```

- Runs IQ-TREE to construct a phylogenetic tree using the alignment file `GLB1.homologsf.al.fas`, with 1000 bootstrap replicates (`-bb 1000`) and 2 CPU threads (`-nt 2`).

```
gotree reroot midpoint -i ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.fas.treefile -o ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile
```

- Uses `gotree` to reroot the phylogenetic tree (`GLB1.homologsf.al.fas.treefile`) at the midpoint and outputs the rerooted tree to `GLB1.homologsf.al.mid.treefile`.

```
nw_order -c n ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile | nw_display -w 1000 -b 'opacity:0' -s > ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile.svg -
```

- Uses `nw_order` to reorder the tree (`GLB1.homologsf.al.mid.treefile`) by node names and pipes the output to `nw_display` to visualize the tree. The visualization is saved as `GLB1.homologsf.al.mid.treefile.svg`.

```
convert ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile.svg
~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile.pdf
```

- Converts the tree visualization from SVG format (`GLB1.homologsf.al.mid.treefile.svg`) to PDF format (`GLB1.homologsf.al.mid.treefile.pdf`) using `convert`.

```
nw_order -c n ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile |
nw_display -
```

- Reorders the tree again by node names and visualizes it with `nw_display`. This command seems to have a similar effect as the earlier tree visualization step.

Lab 6

```
java -jar ~/tools/Notung-3.0_24-beta/Notung-3.0_24-beta.jar -s
~/lab05-$MYGIT/species.tre -g
~/lab06-$MYGIT/GLB1/GLB1.homologs.al.mid.treefile --reconcile
--speciestag prefix --savepng --events --outputdir
~/lab06-$MYGIT/GLB1/
```

- Executes the `Notung` tool to reconcile a gene tree (`GLB1.homologs.al.mid.treefile`) with a species tree (`species.tre`).
  - `--reconcile`: Performs gene-tree-species-tree reconciliation.
  - `--speciestag prefix`: Uses a prefix to tag species names.
  - `--savepng`: Saves the output as a PNG image.
  - `--events`: Includes event annotations (e.g., gene duplication and loss events).
  - `--outputdir`: Specifies the directory for output files (`~/lab06-$MYGIT/GLB1/`).

```
python2.7 ~/tools/recPhyloXML/python/NOTUNGtoRecPhyloXML.py -g
~/lab06-$MYGIT/GLB1/GLB1.homologs.al.mid.treefile.rec.ntg
--include.species
```

- Runs a Python script (`NOTUNGtoRecPhyloXML.py`) to convert the reconciliation output from `Notung` (`GLB1.homologs.al.mid.treefile.rec.ntg`) into a `RecPhyloXML` format, including species data (`--include.species`).

```
thirdkind -Iie -D 40 -f
~/lab06-$MYGIT/GLB1/GLB1.homologs.al.mid.treefile.rec.ntg.xml -o
~/lab06-$MYGIT/GLB1/GLB1.homologs.al.mid.treefile.rec.svg
```

- Uses `thirdkind` to generate a visual representation of the reconciliation data in SVG format.
    - `-Iie`: Specifies input and output file options (likely for XML to SVG conversion).
    - `-D 40`: Sets a parameter (possibly related to image dimensions or scaling).
    - `-f`: Specifies the input XML file (`GLB1.homologs.al.mid.treefile.rec.ntg.xml`).
    - `-o`: Specifies the output file (`GLB1.homologs.al.mid.treefile.rec.svg`).

```
convert -density 150
~/lab06-$MYGIT/GLB1/GLB1.homologs.al.mid.treefile.rec.svg
~/lab06-$MYGIT/GLB1/GLB1.homologs.al.mid.treefile.rec.pdf
```

- Uses `convert` to convert the SVG tree image (`GLB1.homologs.al.mid.treefile.rec.svg`) into a PDF (`GLB1.homologs.al.mid.treefile.rec.pdf`), setting the image density to 150 DPI for higher quality.

Lab 8

```
mkdir ~/lab08-$MYGIT/GLB1 && cd ~/lab08-$MYGIT/GLB1
```

- Creates a new directory `GLB1` under `~/lab08-$MYGIT/` and changes the working directory to it, preparing for the next set of commands.

```
sed 's/*//' ~/lab04-$MYGIT/GLB1/GLB1.homologs.fas >
~/lab08-$MYGIT/GLB1/GLB1.homologs.fas
```

- Uses `sed` to remove any asterisks (`*`) from the file `GLB1.homologs.fas` (likely related to alignment). The cleaned file is saved in the working directory.

```
cp ~/lab05-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile
~/lab08-$MYGIT/GLB1
```

- Copies the tree file (`GLB1.homologsf.al.mid.treefile`) from the previous lab into the new directory `~/lab08-$MYGIT/GLB1`.

```
rpsblast -query ~/lab08-$MYGIT/GLB1/GLB1.homologs.fas -db
~/data/Pfam/Pfam -out ~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out -outfmt
"6 qseqid qlen qstart qend evalue stitle" -evalue .0000000001
```

- Runs `rpsblast`, a tool for comparing protein sequences against the Pfam database. It:
  - Takes the query sequences from `GLB1.homologs.fas`.
  - Uses the Pfam database (`-db ~/data/Pfam/Pfam`).
  - Outputs results to `GLB1.rps-blast.out` in tabular format, with selected columns (`qseqid`, `qlen`, `qstart`, `qend`, `evalue`, and `stitle`).
  - Sets a stringent e-value threshold (`-evalue .0000000001`).

```
Rscript --vanilla ~/lab08-$MYGIT/plotTreeAndDomains.r
~/lab08-$MYGIT/GLB1/GLB1.homologsf.al.mid.treefile
~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out
~/lab08-$MYGIT/GLB1/GLB1.tree.rps.pdf
```

- Executes an R script (`plotTreeAndDomains.r`) to visualize the phylogenetic tree (`GLB1.homologsf.al.mid.treefile`) and the domain hits (`GLB1.rps-blast.out`) as a combined plot. The output is saved as a PDF (`GLB1.tree.rps.pdf`).

```
mlr --inidx --ifs "\t" --opprint cat
~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out | tail -n +2 | less -S
```

- Uses `mlr` (Miller) to read and print the content of `GLB1.rps-blast.out`, skipping the first line (`tail -n +2`) and displaying it in a readable format (`less -S`), with tab-separated fields (`--ifs "\t"`).

```
cut -f 1 ~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out | sort | uniq -c
```

- Extracts the first field (`-f 1`) from the `rps-blast` output, sorts the entries, and counts the occurrences of each unique sequence ID (`qseqid`). This helps determine how many sequences hit each query.

```
cut -f 6 ~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out | sort | uniq -c
```

- Extracts the sixth field (`-f 6`, which is likely the Pfam domain or family name) from the `rps-blast` output, sorts the results, and counts the number of times each unique domain is found.

```
awk '{a=$4-$3;print $1,'\t',a;}'
~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out | sort -k2nr
```

- Uses `awk` to calculate the length of the aligned region (`$4-$3`) for each match in `GLB1.rps-blast.out` (using the `qstart` and `qend` fields). It then sorts the results by the calculated length (`-k2nr`).

```
cut -f 1,5 -d $'\t' ~/lab08-$MYGIT/GLB1/GLB1.rps-blast.out
```

- Extracts the first and fifth fields (`qseqid` and `evalue`) from the `rps-blast` output, which provides the sequence IDs and their corresponding e-values for further analysis or filtering.