Task 1: Phylogenetic Analysis of FAP Sequences

Simon Safron [Mn: 11923407], Alexander Veith [Mn: 12122739], Miriam Überbacher [Mn:01627576]

2025-12-11

Contents

1 Retrieving FAP Sequences via BLAST Search

1.1 Overview

To initiate the sequence alignment and subsequent phylogenetic analysis, 25 additional FAP sequences were retrieved from the NCBI database and saved as `FAP_BLAST.fas`. An R script was developed that retrieves the first 25 BLAST-aligned sequences by providing a query accession number, with results automatically saved to `data_raw/FAP_BLAST.fas`.

1.2 R Implementation

The following R script implements the BLAST query and sequence retrieval workflow:

```r
# Load required packages
if (!requireNamespace("rentrez", quietly = TRUE)) {
  install.packages("rentrez")
}
if (!requireNamespace("xml2", quietly = TRUE)) {
  install.packages("xml2")
}
library(rentrez)
library(xml2)

# Set email for NCBI
Sys.setenv(ENTREZ_EMAIL = "mobofo3315@feralrex.com")

# Function to submit BLAST query and get RID and estimated wait time
submit_blast <- function(query_seq) {
  blast_url <- paste0(
    "https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Put&PROGRAM=blastp&DATABASE=nr&QUERY=",
    URLencode(query_seq)
  )
  res <- httr::GET(blast_url)
  content <- httr::content(res, as = "text", encoding = "UTF-8")

  rid <- sub(".*RID = ([A-Z0-9]+).*", "\\1", content)
  rtoe <- as.numeric(sub(".*RTOE = ([0-9]+).*", "\\1", content))

  if (nchar(rid) == 0) stop("Failed to retrieve RID from BLAST submission")

  return(list(rid = rid, wait = rtoe))
}

# Function to poll BLAST results until ready and parse XML
wait_for_blast_results <- function(rid, max_attempts = 20, wait_sec = 15) {
  for (i in seq_len(max_attempts)) {
    url <- paste0(
      "https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Get&RID=",
      rid, "&FORMAT_TYPE=XML"
    )
    res <- httr::GET(url)
    content <- httr::content(res, as = "text", encoding = "UTF-8")

    if (grepl("<BlastOutput>", content)) {
      cat(sprintf("BLAST results ready on attempt %d.\n", i))
      return(read_xml(content))
    } else {
      cat(sprintf(
        "Attempt %d: Results not ready, retrying in %d seconds...\n",
        i, wait_sec
```

```
      ))
      Sys.sleep(wait_sec)
    }
  }
  stop("Failed to retrieve valid BLAST XML after retries.")
}

# Function to extract unique accession IDs from BLAST XML
extract_accessions <- function(xml_doc, max_hits = 25) {
  hits <- xml_find_all(xml_doc, "//Hit")
  accessions <- unique(xml_text(xml_find_all(hits, "Hit_accession")))
  accessions <- sub("\\..*$", "", accessions)  # remove version suffix
  head(accessions, max_hits)
}

# Query protein accession
query_protein <- "XP_001703004"

# Submit BLAST and get RID + wait time
blast_info <- submit_blast(query_protein)
cat("RID:", blast_info$rid, "\nEstimated wait time (seconds):", blast_info$wait, "\n")

Sys.sleep(blast_info$wait + 5)  # initial wait

# Poll until BLAST results are ready and parse XML
blast_xml <- wait_for_blast_results(blast_info$rid)

# Extract accession IDs from hits
accessions <- extract_accessions(blast_xml)
cat(sprintf("Found %d accessions.\n", length(accessions)))

# Fetch sequences in FASTA format using Entrez
fasta_seqs <- rentrez::entrez_fetch(
  db = "protein",
  id = paste(accessions, collapse = ","),
  rettype = "fasta",
  retmode = "text"
)

# Save sequences to a FASTA file in data_raw folder
dir.create("data_raw", showWarnings = FALSE)
output_file <- "data_raw/FAP_BLAST.fas"
writeLines(fasta_seqs, con = output_file)
cat(sprintf("Saved %d sequences to %s\n", length(accessions), output_file))
```

1.3 Alternative Python Implementation

If retrieval is unsuccessful with the R script, you can increase the `max_attempts` and `wait_sec` parameters and retry. Alternatively, the following Python implementation is generally more robust:

```python
# Requirements for the script
# pip install biopython

from Bio.Blast import NCBIWWW, NCBIXML
from Bio import Entrez, SeqIO
import os

Entrez.email = "mobofo3315@feralrex.com"

# Create output directory if needed
os.makedirs("data_raw", exist_ok=True)

# 1. Run BLASTp for XP_001703004
result_handle = NCBIWWW.qblast("blastp", "nr", "XP_001703004", hitlist_size=25)
blast_records = NCBIXML.read(result_handle)

# 2. Collect accession IDs from hits
accessions = []
for alignment in blast_records.alignments:
    # Extract accession (remove version)
    acc = alignment.accession.split('.')[0]
    if acc not in accessions:
        accessions.append(acc)
    if len(accessions) >= 25:
        break

print(f"Found {len(accessions)} accessions.")

# 3. Fetch sequences in FASTA format
handle = Entrez.efetch(
    db="protein",
    id=",".join(accessions),
    rettype="fasta",
    retmode="text"
)
records = list(SeqIO.parse(handle, "fasta"))

# 4. Save to file
output_file = r"C:\Users\...\data_raw\FAP_BLAST.fas"
SeqIO.write(records, output_file, "fasta")
print(f"Saved {len(records)} sequences to {output_file}.")
```

2 Data Preparation

2.1 Sequence Consolidation and Duplicate Removal

After acquiring the FAP sequences, they were combined with the existing `FAP_genbank.fas` dataset. All duplicate sequences were removed, and the refined dataset was saved as `FAP_all.fas`. The following code was executed to determine the number of duplicate sequences:

```
duplicates <- duplicated(FAP_all)
sum(duplicates)

# Output: [1] 4
```

Result: Four duplicate sequences were identified and removed from the combined dataset.

3 Phylogenetic Analysis

3.1 Model Selection and Testing

Following model testing, the results were evaluated to select the best-fitting substitution model:

```
head(mt, 5)

#           Model df    logLik      AIC
# 1           WAG 63 -18678.95 37483.90
# 2         WAG+I 64 -18581.27 37290.54
# 3       WAG+G(4) 64 -18333.38 36794.76
# 4     WAG+G(4)+I 65 -18320.26 36770.52
# 5           JTT 63 -18899.60 37925.19
#              AICw     AICc          AICcw
# 1 1.231348e-155 37499.23 2.074524e-155
# 2 1.197702e-113 37306.39 1.558333e-113
# 3  5.452270e-06 36810.61  7.093960e-06
# 4  9.999945e-01 36786.89  9.999929e-01
# 5 1.845146e-251 37940.52 3.108624e-251
#        BIC
# 1 37759.85
# 2 37570.87
# 3 37075.09
# 4 37055.23
# 5 38201.14
```

3.1.1 Best Fit Model

The model selection analysis revealed **WAG+G(4)+I** as the optimal substitution model with the highest AIC weight (0.9999945). The fitted model parameters are presented below:

```
fit

# model: WAG+G(4)+I
# loglikelihood: -18302.52
# unconstrained loglikelihood: -3743.818
# Proportion of invariant sites: 0.02643574
# Model of rate heterogeneity: Discrete gamma model
# Number of rate categories: 4
# Shape parameter: 2.081987
#        Rate Proportion
# 1 0.0000000 0.02643574
# 2 0.3110929 0.24339107
# 3 0.6814733 0.24339107
# 4 1.1010182 0.24339107
# 5 2.0150298 0.24339107
# Rate matrix: WAG
```

3.2 Maximum Likelihood Tree Construction

A maximum likelihood phylogenetic tree was constructed and midpoint-rooted for better visualization. Bootstrap values greater than 70% were displayed to indicate branch support:

```
tree_rooted <- midpoint(fit$tree)

plotBS(tree_rooted,
       trees = fit$bs,
       p = 70,
       type = "phylogram",
       cex = 0.6,
       main = "Maximum Likelihood Tree of FAP Sequences")

add.scale.bar()
```

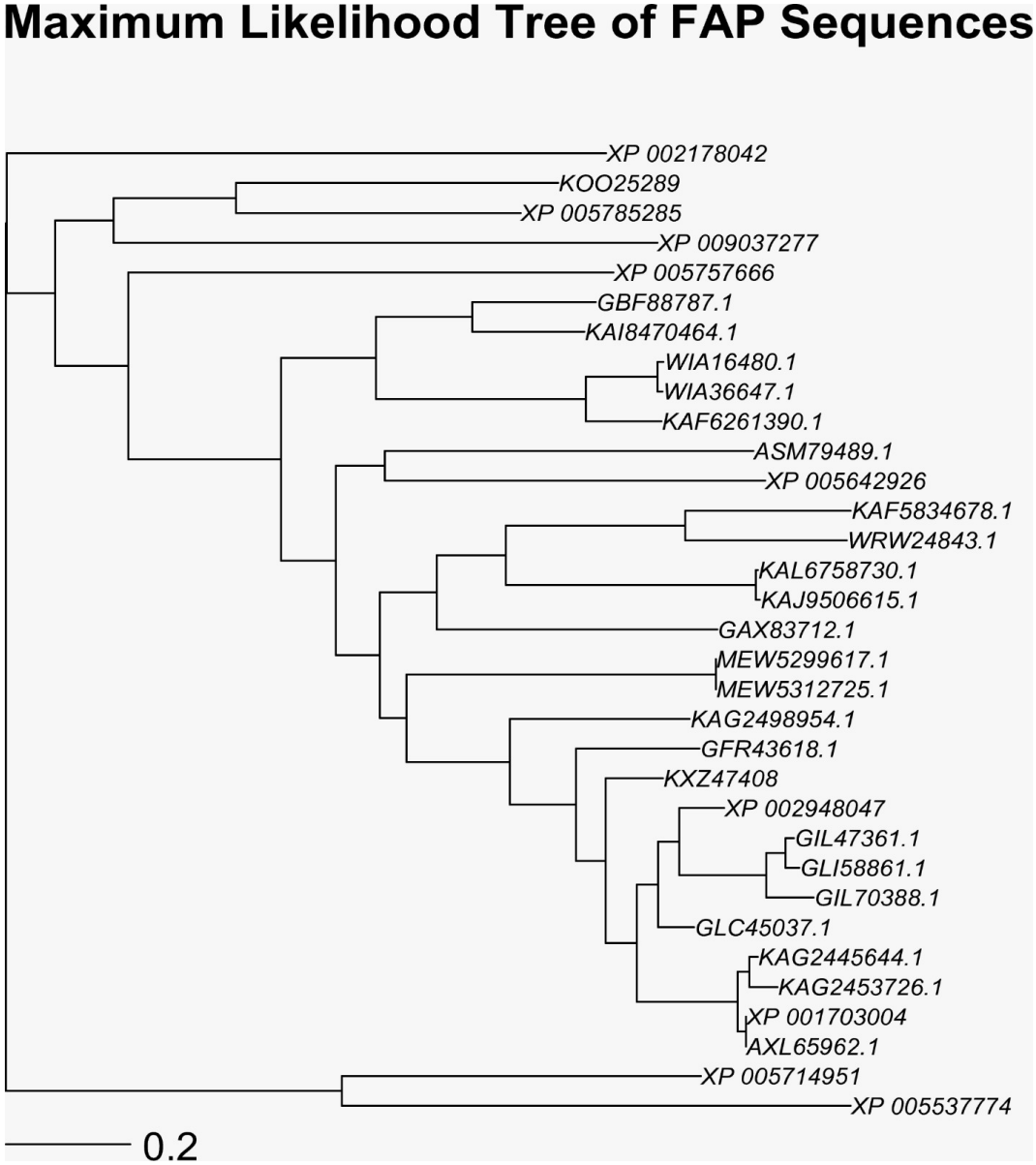

Figure 1: Maximum Likelihood phylogenetic tree of FAP sequences estimated under the WAG+G(4)+I model. Bootstrap values (>70%) are displayed at internal nodes.

References

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2025. *Rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Paradis, Emmanuel, Simon Blomberg, Ben Bolker, Joseph Brown, Santiago Claramunt, Julien Claude, Hoa Sien Cuong, et al. 2024. *Ape: Analyses of Phylogenetics and Evolution*. https://github.com/emmanuelparadis/ape.

Paradis, Emmanuel, and Klaus Schliep. 2019. "Ape 5.0: An Environment for Modern Phylogenetics and Evolutionary Analyses in R." *Bioinformatics* 35: 526–28. https://doi.org/10.1093/bioinformatics/bty633.

Schliep, Klaus. 2011. "Phangorn: Phylogenetic Analysis in r." *Bioinformatics* 27 (4): 592–93. https://doi.org/10.1093/bioinformatics/btq706.

Schliep, Klaus, Emmanuel Paradis, Leonardo de Oliveira Martins, Alastair Potts, and Iris Bardel-Kahr. 2024. *Phangorn: Phylogenetic Reconstruction and Analysis*. https://github.com/KlausVigo/phangorn.

Schliep, Klaus, Alastair J. Potts, David A. Morrison, and Guido W. Grimm. 2017. "Intertwining Phylogenetic Trees and Networks." *Methods in Ecology and Evolution* 8 (10): 1212–20.

Wickham, Hadley, Jim Hester, and Jeroen Ooms. 2025. *Xml2: Parse XML*. https://xml2.r-lib.org.

Winter, David. 2025. *Rentrez: Entrez in r*. https://github.com/ropensci/rentrez/.

Winter, David J. 2017. "rentrez: An r Package for the NCBI eUtils API." *The R Journal* 9: 520–26.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC.

———. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2025. *Knitr: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.