TRANSCRIPTOMICS

Annotation

Day 03

https://tttorres.github.io/transcriptomics/

Transcriptome Annotation

First things first!

Downloading databases for today's activities

1. Create a new folder and navigate to it

```
mkdir ~/rnaseq/00-Databases/
cd ~/rnaseq/00-Databases/
```

2. Download databases (copy commands from the course page)

```
wget https://ftp.uniprot.org/pub/databases/uniprot/curren
wget hhttps://ftp.ebi.ac.uk/pub/databases/Pfam/current_re
```

Transcriptome Annotation

Meaning from seguences

Transcriptome Annotation

Meaning from seguences

> 📝 Assigning biological meaning to assembled sequences by identifying coding regions, functional domains, and associated biological processes.

Why is it Important?

- To understand gene functions.
- To identify potential coding sequences and functional domains.
- To classify transcripts based on biological processes, molecular functions, and cellular components.

Transcriptome Annotation

Predicting function from sequence

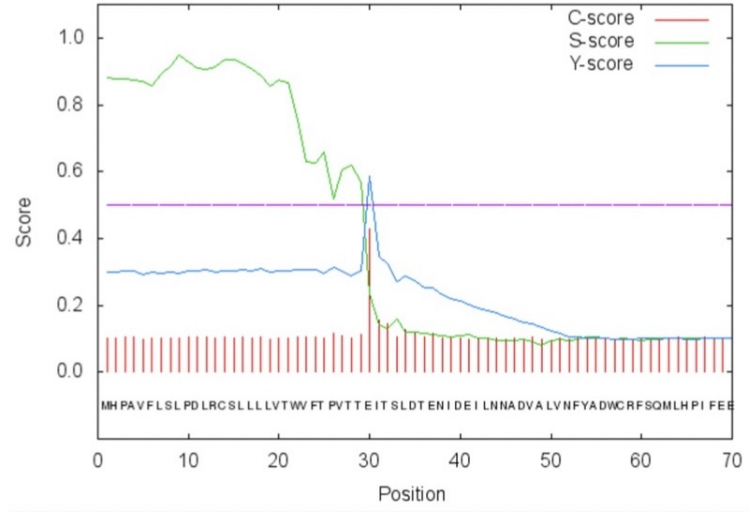
Sequence similarity

```
Query           THVHRPYNEHKSLSGTARYMSINTHLGREQSRRDDLESMGHVFMYFLRGSLPW--QGLKA
                T    P +  K   GT  Y S + HLG    RR DLE +G      L   LPW  Q L A
Database Match  TGDFKP-DPKKMHNGTIEYTSRDAHLG-VPTRRADLEILGYNLIEWLGAELPWVTQKLLA
```

Sequence composition

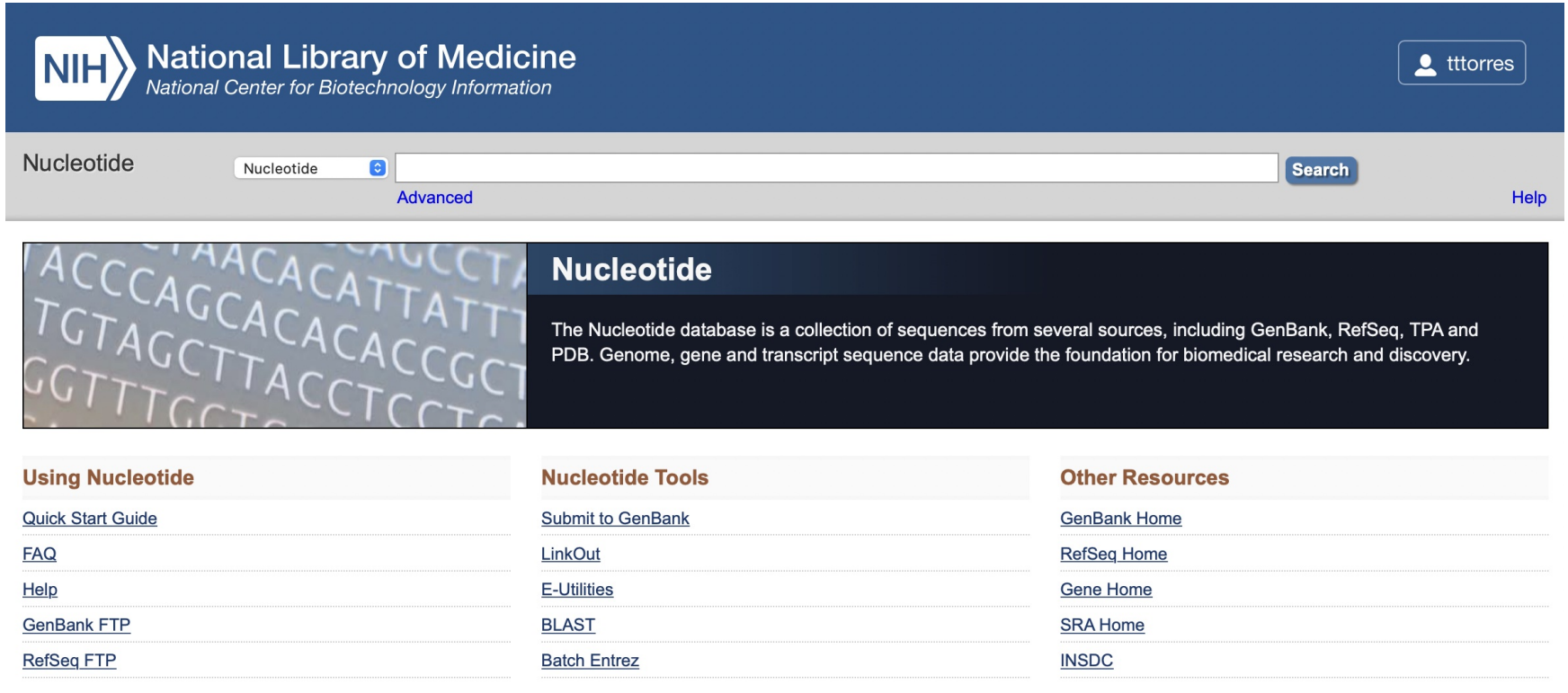
Predict functions of sequence using machine learning methods for pattern recognition.

- Neural Networks
- Hidden Markov Models



Predicting function from sequence

Sequence similarity



Predicting function from sequence

Sequence similarity

1. Go to the `03-Assembly` folder

```
cd ~/rnaseq/03-Assembly/
```

2. Check the first 20 lines of the assembly

```
# head -20 ${SPECIES}.Trinity.fasta
head -20 Chom.Trinity.fasta
```

Predicting function from sequence

Sequence similarity

1. Go to the `03-Assembly` folder

```
cd ~/rnaseq/03-Assembly/
```

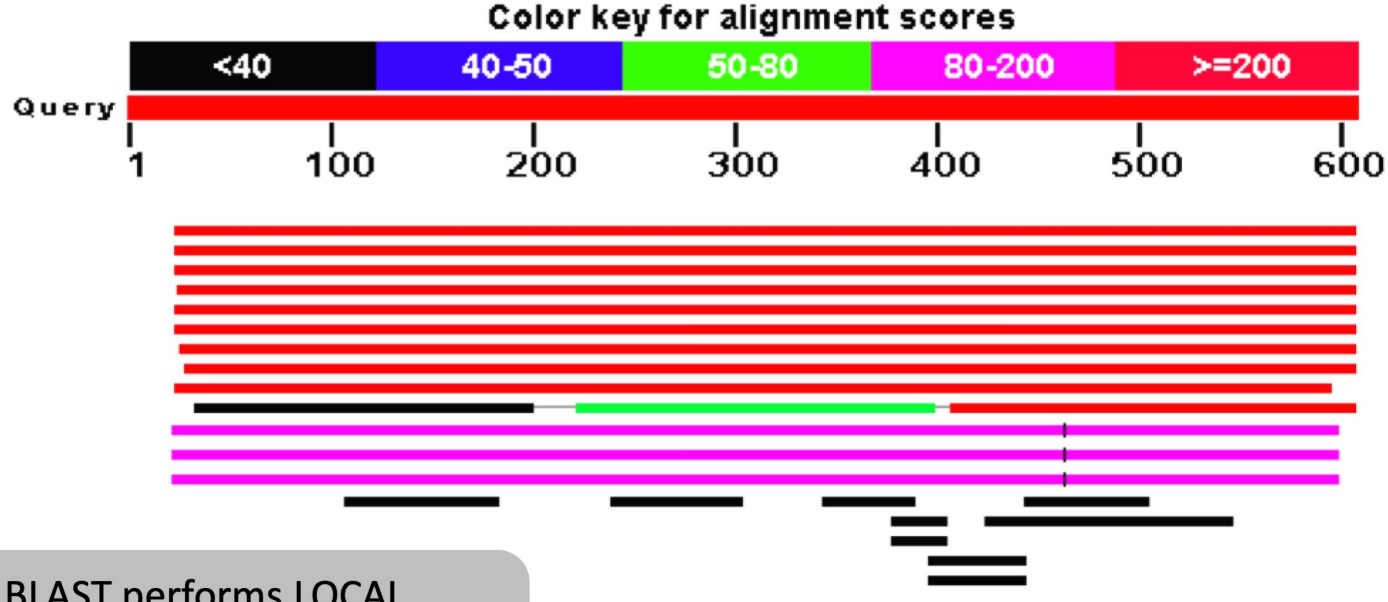
2. Check the first 20 lines of the assembly

```
head -20 Chom.Trinity.fasta
>TRINITY_DN1870_c0_g1_i1 len=208 path=[0:0-207]
ACTGAATCTATCATGAAACGTACCGCTTTGGTAGCCAACACCTCTAACATGCCTGTCGCT
GCTCGTGAAGCTTCAATTTACACTGGTATTACCTTATCTGAATACTTCCGTGATATGGGC
TACAACGTATCTATGATGGCTGATTCTACCTCTCGTTGGGCTGAAGCTCTTCGTGAAATT
TCTGGTCGTTTGGCTGAAATGCCTGCCG
...
```

3. Copy the first sequence and blast it against the GenBank's database

Sequence similarity

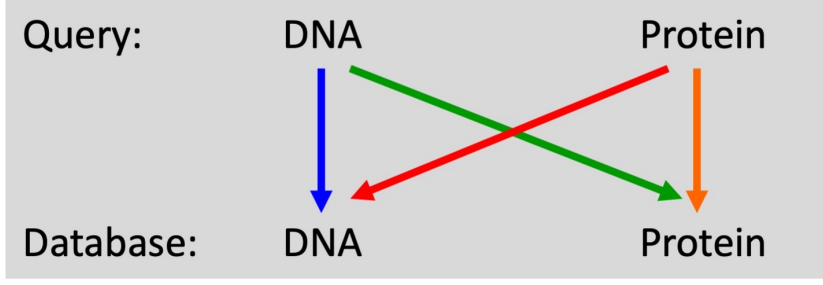
BLAST - Basic Local Alignment and Search Tool



BLAST performs LOCAL alignments e finds regions with high similarity

Sequence similarity

BLAST - Basic Local Alignment and Search Tool



blastn – nucleotides vs. nucleotides

blastp – protein vs. protein

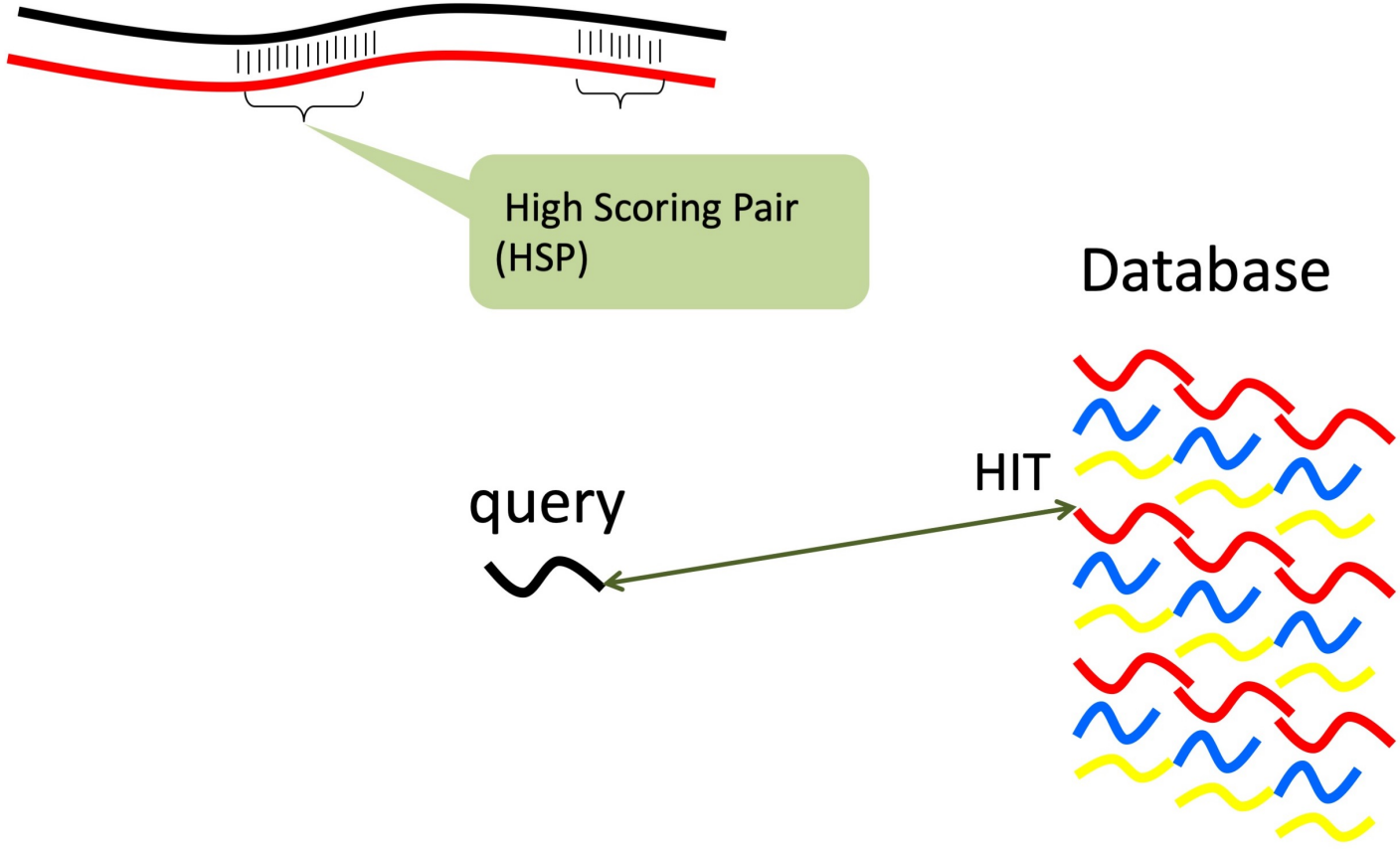
blastx – translated query vs. protein database

tblastn – proten vs. translated nucleotide database

tblastx – translated query vs. translated nucleotide database

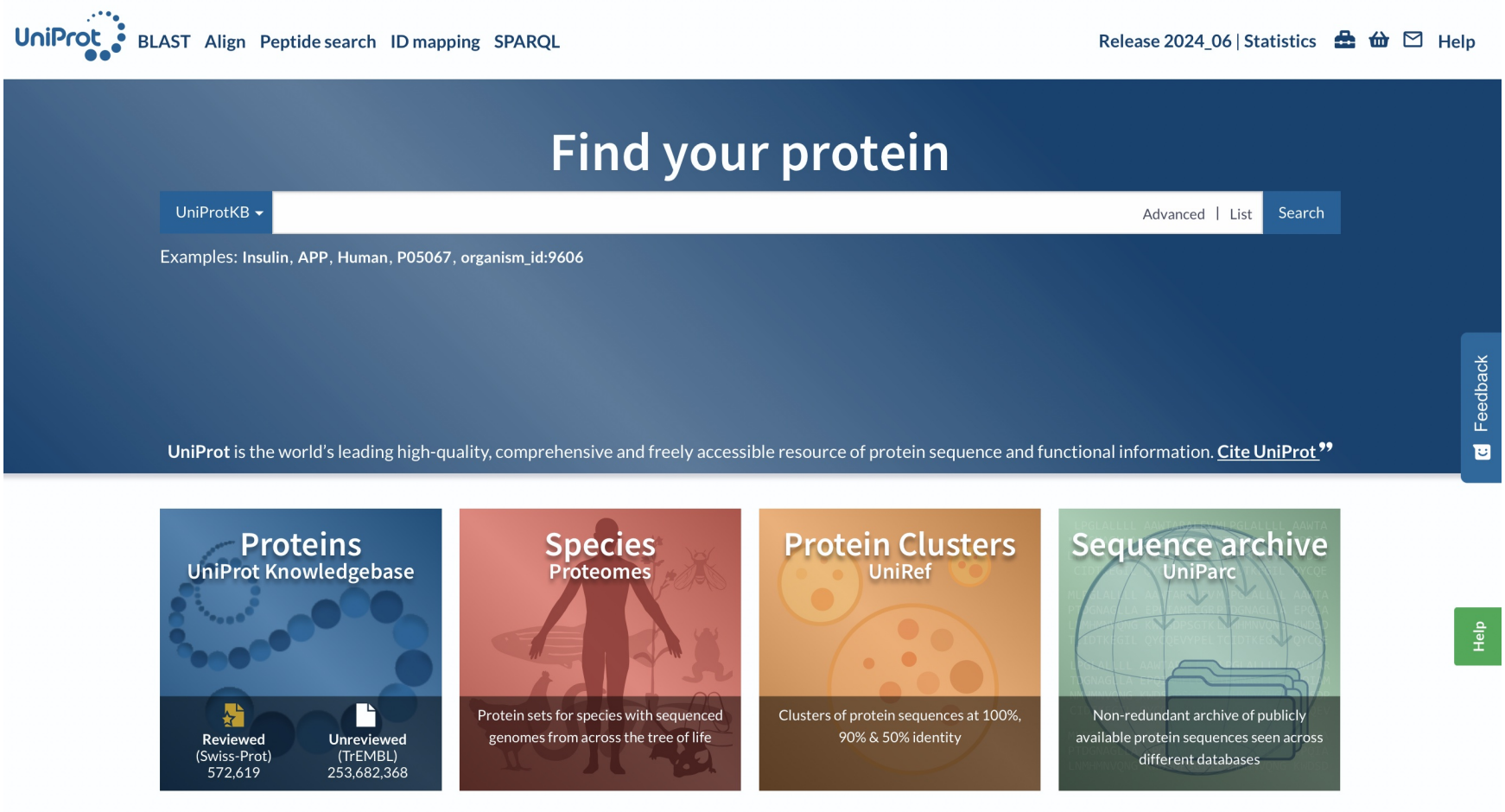
Sequence similarity

BLAST - Basic Local Alignment and Search Tool



Predicting function from sequence

Sequence similarity



Predicting function from sequence

Sequence similarity

1. Go to the `03-Assembly` folder

```
cd ~/rnaseq/03-Assembly/
```

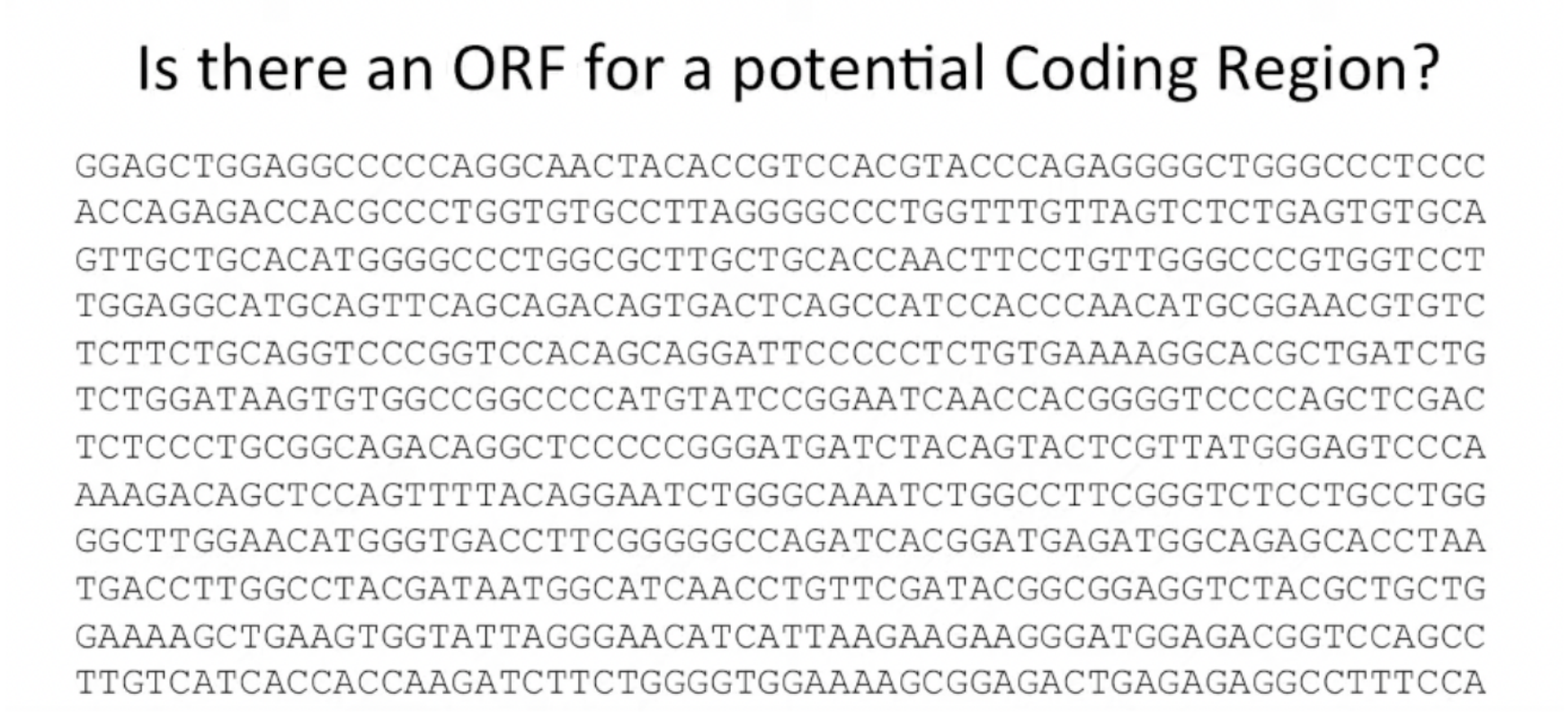
2. Check the first 20 lines of the assembly

```
head -20 Chom.Trinity.fasta
>TRINITY_DN1870_c0_g1_i1 len=208 path=[0:0-207]
ACTGAATCTATCATGAAACGTACCGCTTTGGTAGCCAACACCTCTAACATGCCTGTCGCT
GCTCGTGAAGCTTCAATTTACACTGGTATTACCTTATCTGAATACTTCCGTGATATGGGC
TACAACGTATCTATGATGGCTGATTCTACCTCTCGTTGGGCTGAAGCTCTTCGTGAAATT
TCTGGTCGTTTGGCTGAAATGCCTGCCG
...
```

3. Copy the first sequence and blast it against the UniProt's database

Predicting function from sequence

No sequence similarity!



Predicting function from sequence

No sequence similarity!

Is there an ORF for a potential Coding Region?

GGAGCTGGAGGCCCCCAGGCAACTACACCGTCCACGTACCCAGAGGGGCTGGGCCCTCCC
ACCAGAGACCACGCCCTGGTGTGCCTTAGGGGCCCTGGTTTGTTAGTCTCTGAGTGTGCA
GTTGCTGCAC**ATGGGGCCCTGGCGCTTGCTGCACCAACTTCCTGTTGGGCCCGTGGTCCT**
TGGAGGCATGCAGTTCAGCAGACAGTGACTCAGCCATCCACCCAACATGCGGAACGTGTC
TCTTCTGCAGGTCCCGGTCCACAGCAGGATTCCCCCTCTGTGAAAAGGCACGCTGATCTG
TCTGGATAAGTGTGGCCGGCCCCATGTATCCGGAATCAACCACGGGGTCCCCAGCTCGAC
TCTCCCTGCGGCAGACAGGCTCCCCCGGGATGATCTACAGTACTCGTTATGGGAGTCCCA
AAAGACAGCTCCAGTTTTACAGGAATCTGGGCAAATCTGGCCTTCGGGTCTCCTGCCTGG
GGCTTGGAACATGGGTGACCTTCGGGGGCCAGATCACGGATGAGATGGCAGAGCACCTAA
TGACCTTGGCCTACGATAATGGCATCAACCTGTTCGATACGGCGGAGGTCTACGCTGCTG
GAAAAGCTGAAGTGGTATTAGGGAACATCATTAAGAAGAAGGGATGGAGACGGTCCAGCC

Predicting function from sequence

ORF Finder

Finds all open reading frames and provides translations

NIH National Library of Medicine
National Center for Biotechnology Information

tttorres

Open Reading Frame Finder

ORF finder searches for open reading frames (ORFs) in the DNA sequence you enter. The program returns the range of each ORF, along with its protein translation. Use ORF finder to search newly sequenced DNA for potential protein encoding segments, verify predicted protein using newly developed SMART BLAST or regular BLASTP.

This web version of the ORF finder is limited to the subrange of the query sequence up to 50 kb long. Stand-alone version, which doesn't have query sequence length limitation, is available for Linux x64.

Examples (click to set values, then click Submit button) :

- NC_011604 Salmonella enterica plasmid pWES-1; genetic code: 11; 'ATG' and alternative initiation codons; minimal ORF length: 300 nt
- NM_000059; genetic code: 1; start codon: 'ATG only'; minimal ORF length: 150 nt

Enter Query Sequence

Enter accession number, gi, or nucleotide sequence in FASTA format:

From: **To:**

Predicting function from sequence

ORF Finder

1. Go to the `03-Assembly` folder

```
cd ~/rnaseq/03-Assembly/
```

2. Check the first 20 lines of the assembly

```
head -20 Chom.Trinity.fasta
>TRINITY_DN1870_c0_g1_i1 len=208 path=[0:0-207]
ACTGAATCTATCATGAAACGTACCGCTTTGGTAGCCAACACCTCTAACATGCCTGTCGCT
GCTCGTGAAGCTTCAATTTACACTGGTATTACCTTATCTGAATACTTCCGTGATATGGGC
TACAACGTATCTATGATGGCTGATTCTACCTCTCGTTGGGCTGAAGCTCTTCGTGAAATT
TCTGGTCGTTTGGCTGAAATGCCTGCCG
...
```

3. Copy the first sequence and paste it in ORF finder

Predicting function from sequence

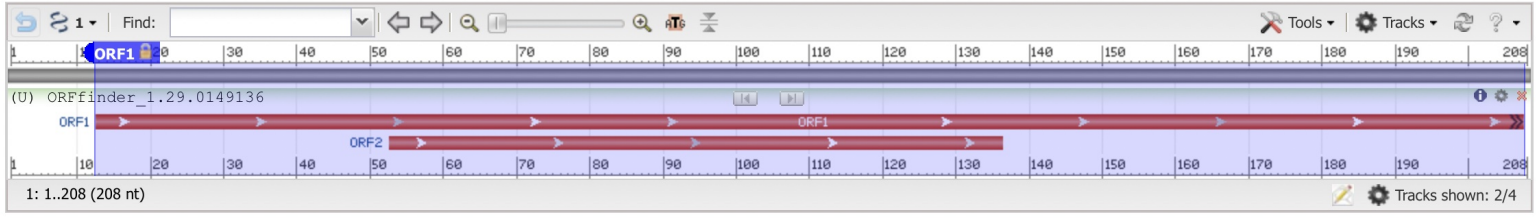
ORF Finder

< ORFfinder submitting page > Help

Open Reading Frame Viewer

Sequence

ORFs found: 2 Genetic code: 1 Start codon: 'ATG' and alternative codons



Six-frame translation...

ORF1 (64 aa) Display ORF as... Mark

```
>lcl|ORF1
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWA
EALREISGRLAEMPA
```

Mark subset... Marked: 0 Download marked set as Protein FASTA

Label	Strand	Frame	Start	Stop	Length (nt | aa)
ORF1	+	1	13	>207	195 | 64
ORF2	+	2	53	136	84 | 27

ORF1

SmartBLAST

BLAST

Marked set (0)

SmartBLAST best hit titles...

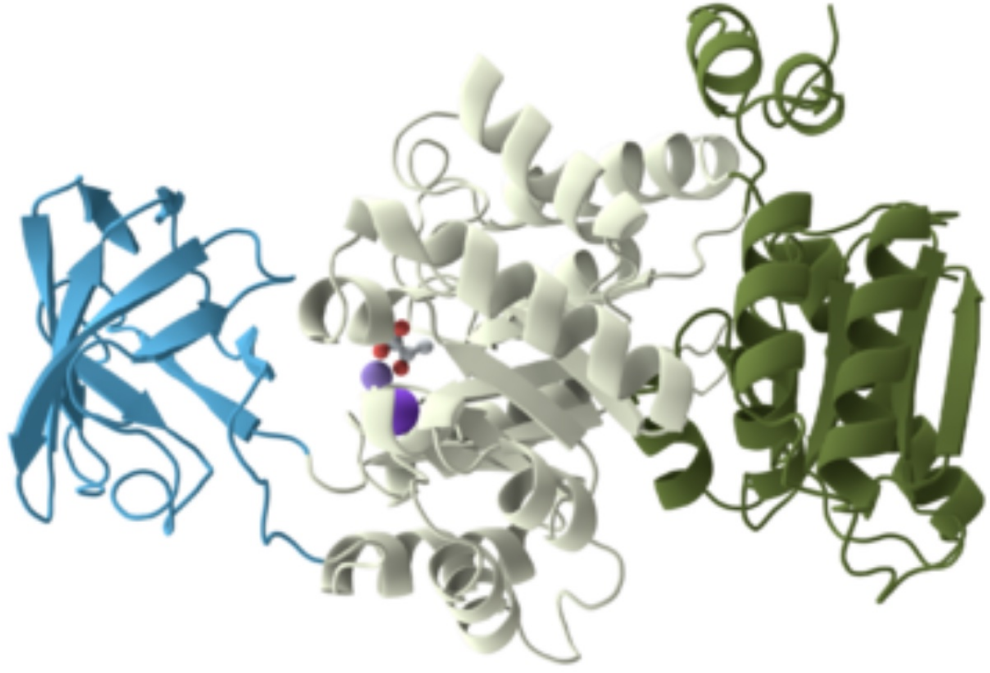
BLAST

BLAST Database:

UniProtKB/Swiss-Prot (swissprot)

Predicting function from sequence

Functional domains in putative CDS



Hints at substrate binding or catalytic activity

DNA, RNA, calcium, phoshate, etc.

Glycoslase, methylase, kinase, nuclease, lipase, protease, etc.

Predicting function from sequence

Pfam database

Large collection of protein families, each represented by multiple sequence alignments and hidden Markov models

Predicting function from sequence

Pfam database

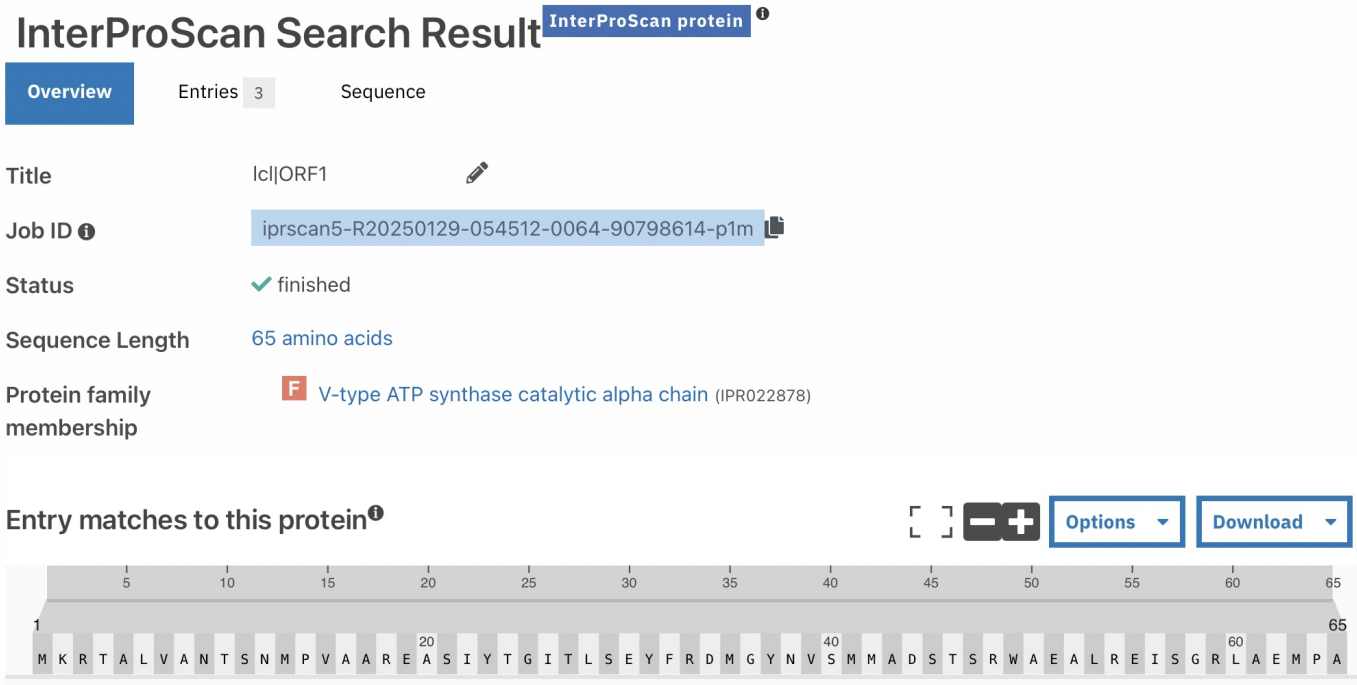
1. Copy the longest ORF generated by ORF Finder

```
>lcl|ORF1
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWA
EALREISGRLAEMPA
```

2. Paste it in Pfam's `SEQUENCE SEARCH`

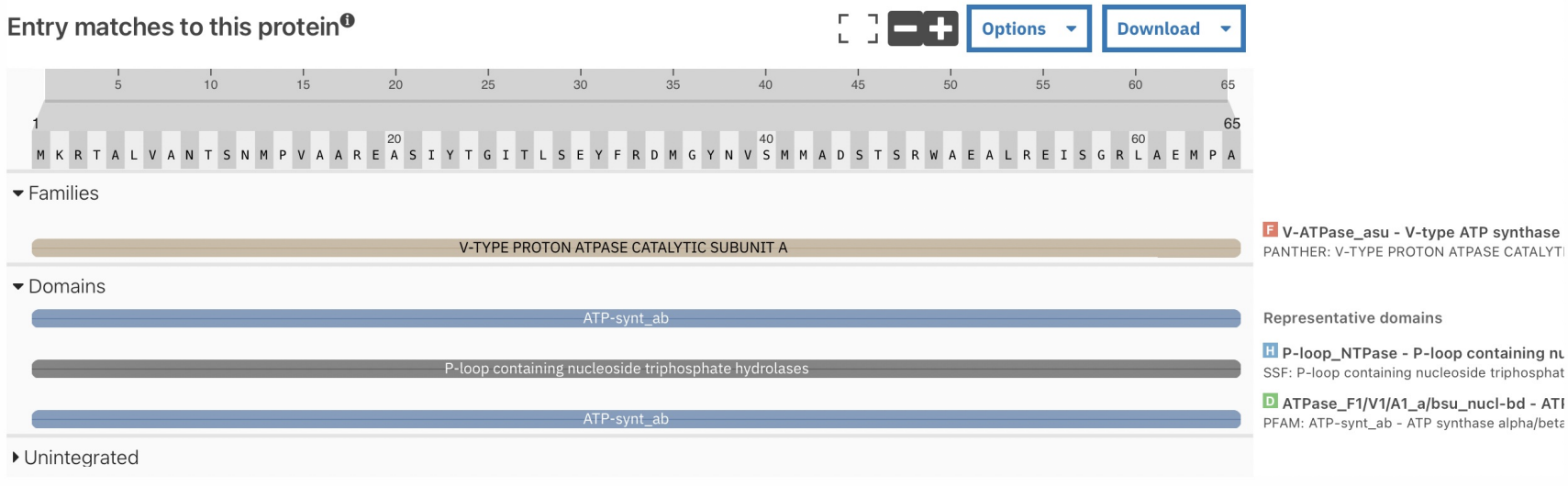
Predicting function from sequence

Pfam database



Predicting function from sequence

Pfam database



InterPro GO terms

Biological Process

- ATP metabolic process (GO:0046034)

Molecular Function

- ATP binding (GO:0005524)
- proton-transporting ATPase activity, rotational mechanism (GO:0046961)

Cellular Component

None

PANTHER GO terms

Biological Process

- proton transmembrane transport (GO:1902600)

Molecular Function

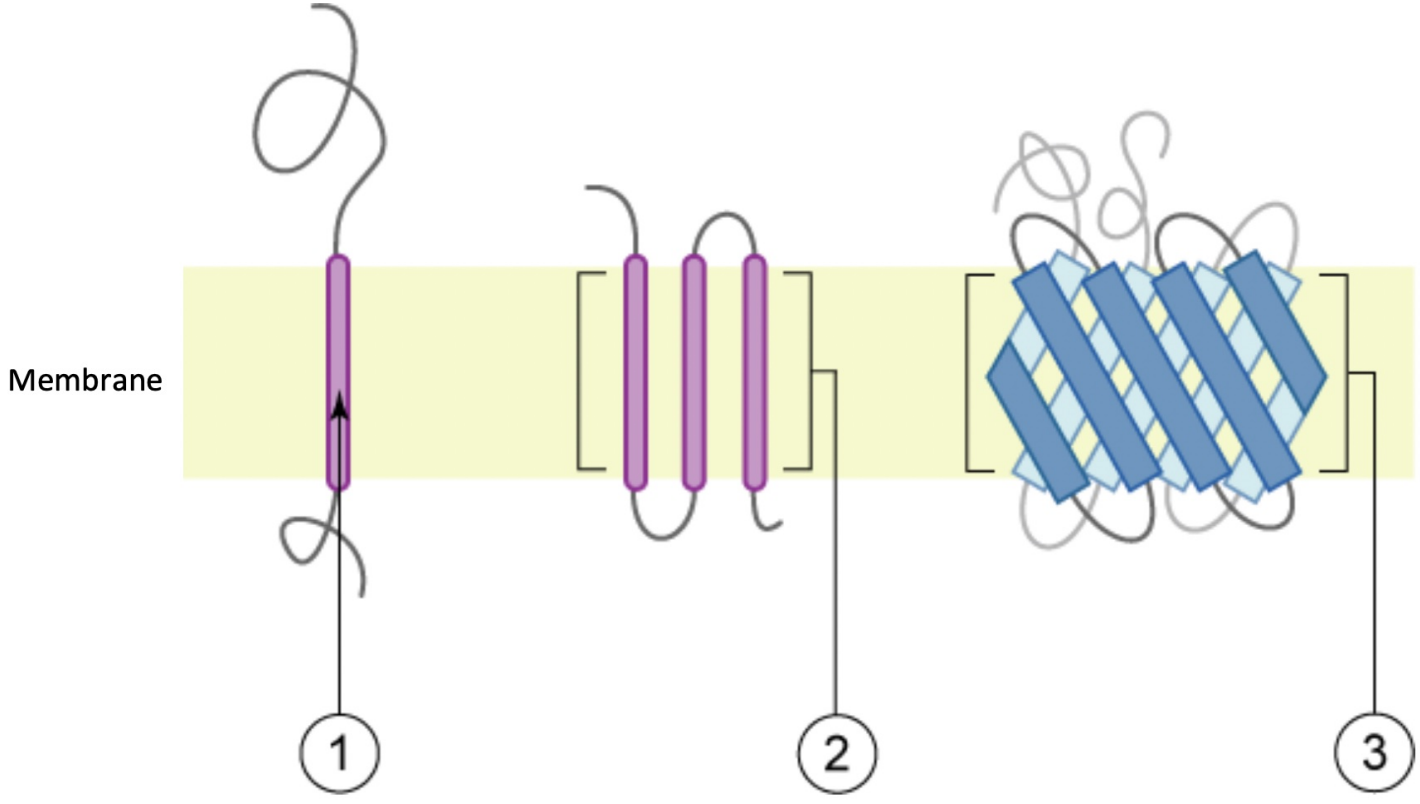
- proton-transporting ATPase activity, rotational mechanism (GO:0046961)

Cellular Component

- lysosomal membrane (GO:0005765)

Predicting function from sequence

Transmembrane Proteins



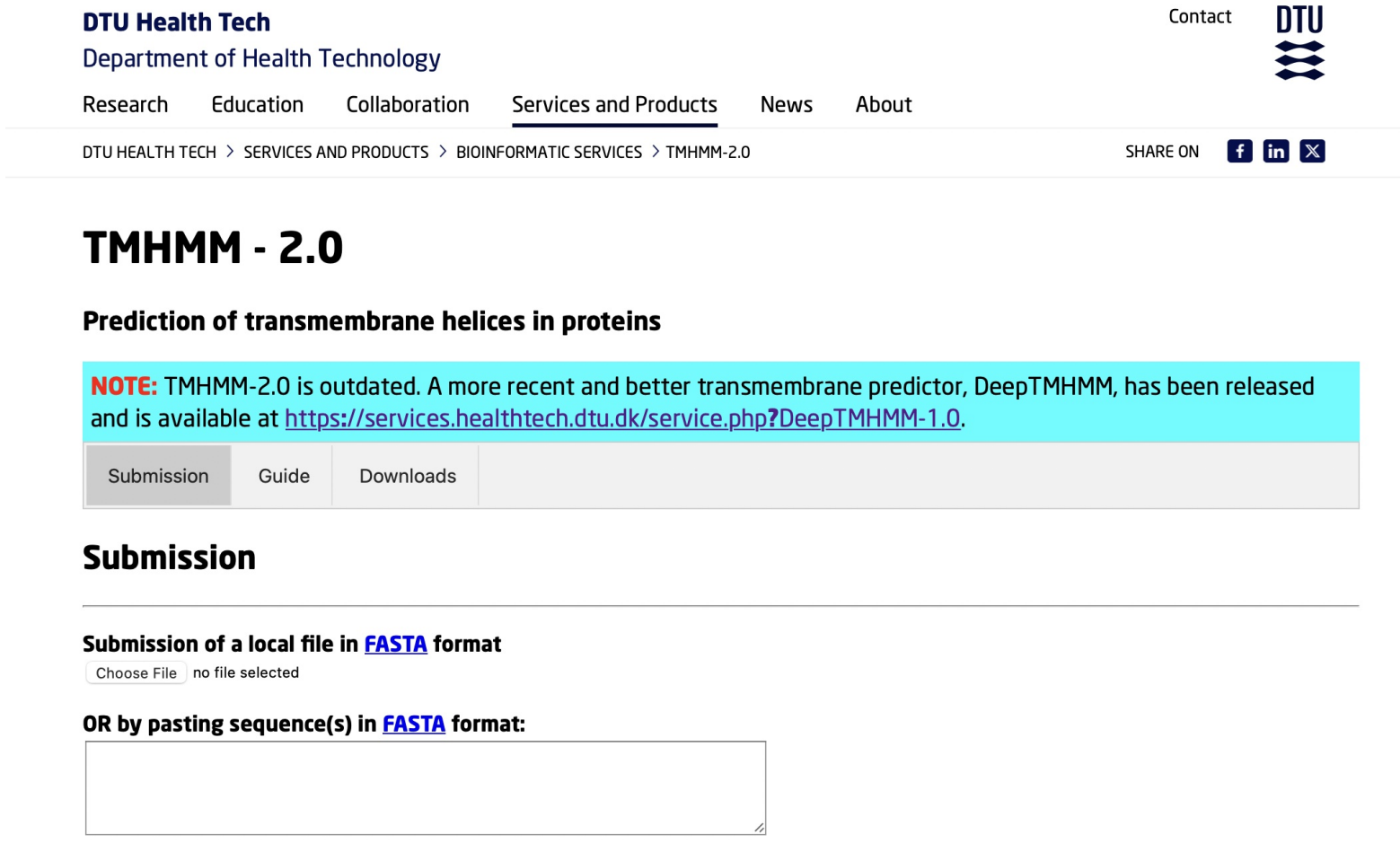
Single transmembrane α-helix (bitopic membrane protein)

Polytopic transmembrane α-helical protein

Polytopic transmembrane β-sheet protein

Predicting function from sequence

Transmembrane Proteins: DeepTMHMM



Predicting function from sequence

Transmembrane Proteins: DeepTMHMM

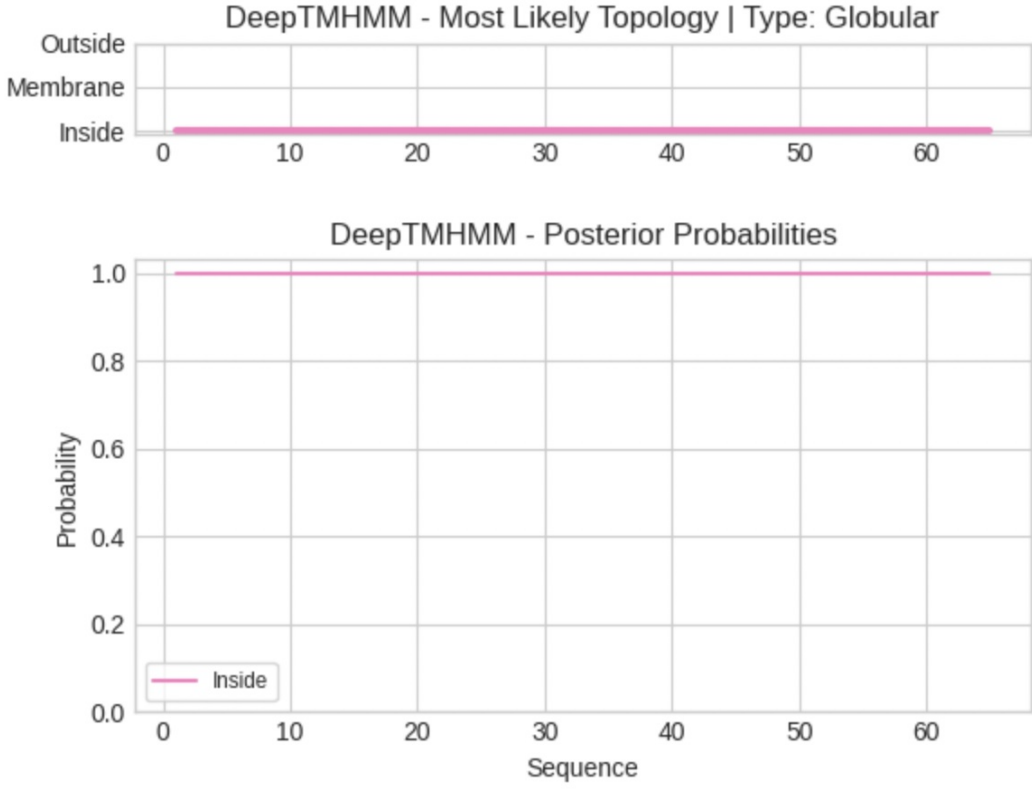
1. Copy the longest ORF generated by ORF Finder

```
>lcl|ORF1
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWA
EALREISGRLAEMPA
```

2. Paste it in DeepTMHMM's `Sequence Submission` box

RNA-Seq: Annotation

Methods to predict gene function



You can download the probabilities used to generate this plot here

Predicted Topologies

```
>lcl_ORF1 | GLOB
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWAEALREISGRLAEMPA
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

Predicting function from sequence

Transmembrane Proteins: DeepTMHMM

1. Copy the longest ORF generated by ORF Finder

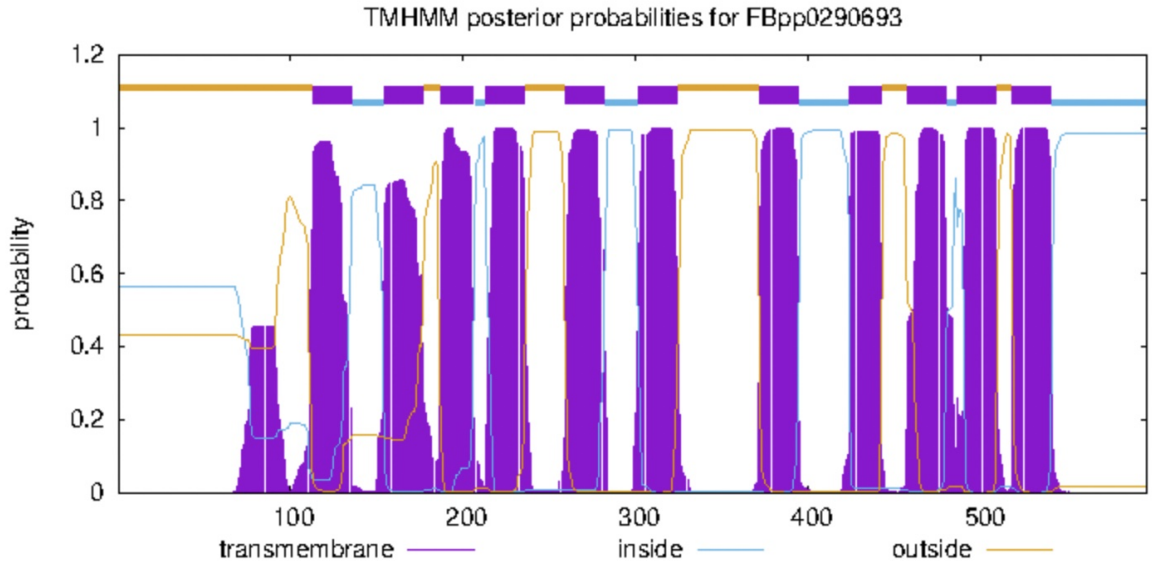
```
>lcl|ORF1
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWA
EALREISGRLAEMPA
```

2. Paste it in DeepTMHMM's `Sequence Submission` box

3. Repeat with the sequence in the course page

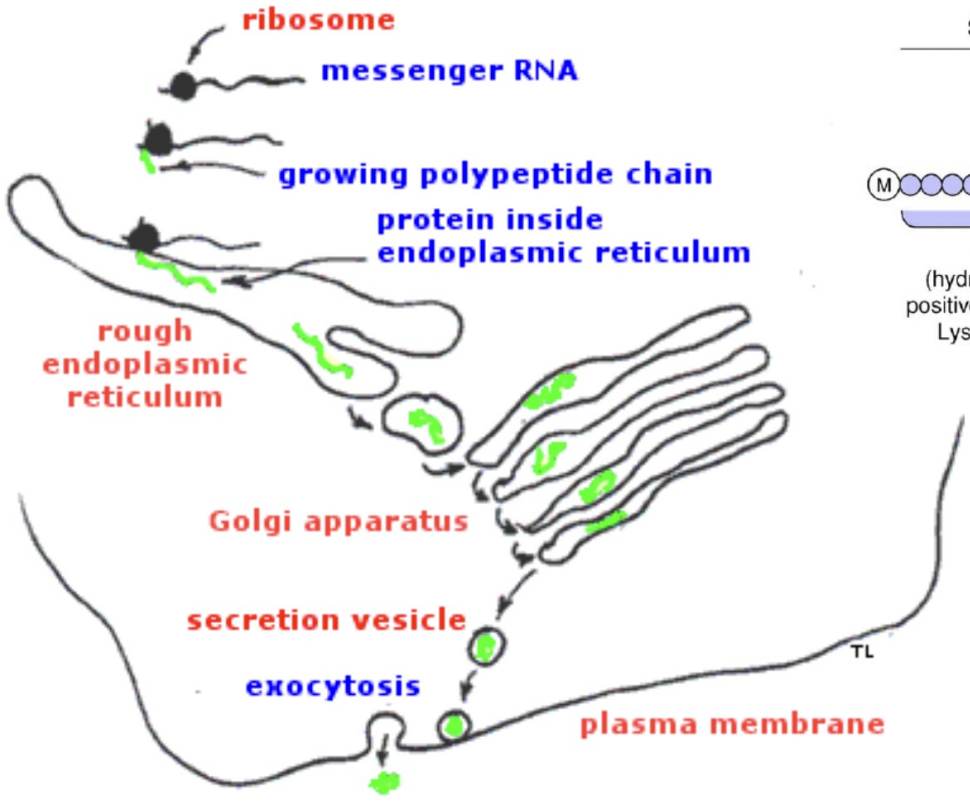
Predicting function from sequence

Transmembrane Proteins: DeepTMHMM

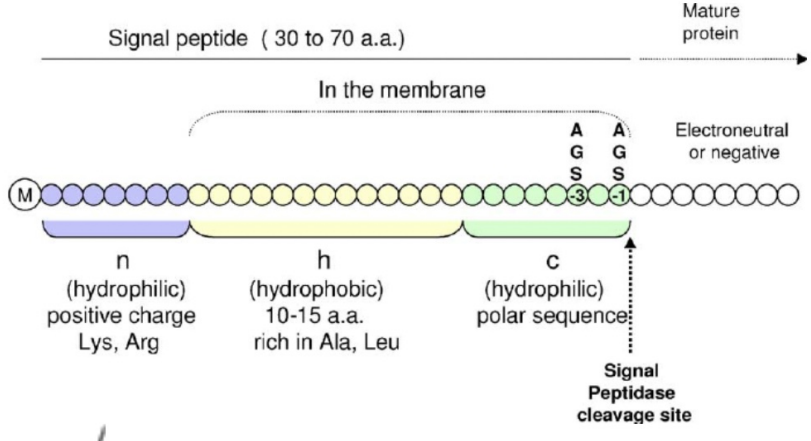


Predicting function from sequence

Predicting Secreted Proteins



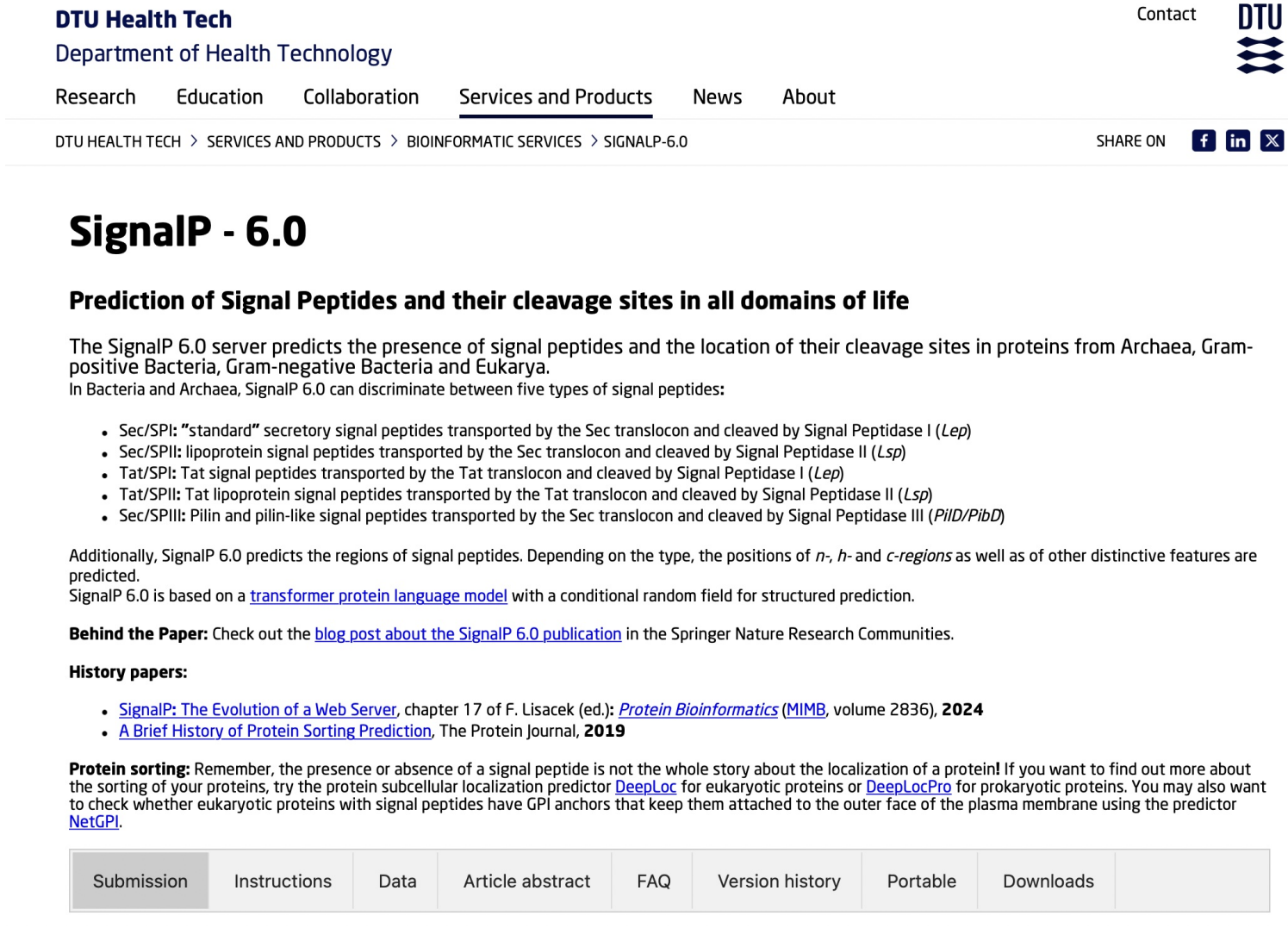
(from: https://courses.washington.edu/conj/cell/secretion.htm)



(from: Vaccine 23(15):1770-8)

Predicting function from sequence

Predicting Secreted Proteins: SignalP



DTU Health Tech
Department of Health Technology

Contact DTU

Research Education Collaboration Services and Products News About

DTU HEALTH TECH > SERVICES AND PRODUCTS > BIOINFORMATIC SERVICES > SIGNALP-6.0

SHARE ON

SignalP - 6.0

Prediction of Signal Peptides and their cleavage sites in all domains of life

The SignalP 6.0 server predicts the presence of signal peptides and the location of their cleavage sites in proteins from Archaea, Gram-positive Bacteria, Gram-negative Bacteria and Eukarya.
In Bacteria and Archaea, SignalP 6.0 can discriminate between five types of signal peptides:

- Sec/SPI: "standard" secretory signal peptides transported by the Sec translocon and cleaved by Signal Peptidase I (*Lep*)
- Sec/SPII: lipoprotein signal peptides transported by the Sec translocon and cleaved by Signal Peptidase II (*Lsp*)
- Tat/SPI: Tat signal peptides transported by the Tat translocon and cleaved by Signal Peptidase I (*Lep*)
- Tat/SPII: Tat lipoprotein signal peptides transported by the Tat translocon and cleaved by Signal Peptidase II (*Lsp*)
- Sec/SPIII: Pilin and pilin-like signal peptides transported by the Sec translocon and cleaved by Signal Peptidase III (*PilD/PibD*)

Additionally, SignalP 6.0 predicts the regions of signal peptides. Depending on the type, the positions of *n-*, *h-* and *c-regions* as well as of other distinctive features are predicted.
SignalP 6.0 is based on a transformer protein language model with a conditional random field for structured prediction.

Behind the Paper: Check out the blog post about the SignalP 6.0 publication in the Springer Nature Research Communities.

History papers:

- SignalP: The Evolution of a Web Server, chapter 17 of F. Lisacek (ed.): *Protein Bioinformatics* (MIMB, volume 2836), **2024**
- A Brief History of Protein Sorting Prediction, The Protein Journal, **2019**

Protein sorting: Remember, the presence or absence of a signal peptide is not the whole story about the localization of a protein! If you want to find out more about the sorting of your proteins, try the protein subcellular localization predictor DeepLoc for eukaryotic proteins or DeepLocPro for prokaryotic proteins. You may also want to check whether eukaryotic proteins with signal peptides have GPI anchors that keep them attached to the outer face of the plasma membrane using the predictor NetGPI.

Submission | Instructions | Data | Article abstract | FAQ | Version history | Portable | Downloads

Predicting function from sequence

Predicting Secreted Proteins: SignalP

1. Copy the longest ORF generated by ORF Finder

```
>lcl|ORF1
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWA
EALREISGRLAEMPA
```

2. Paste it in SignalP's `Submit Data` box

Predicting function from sequence

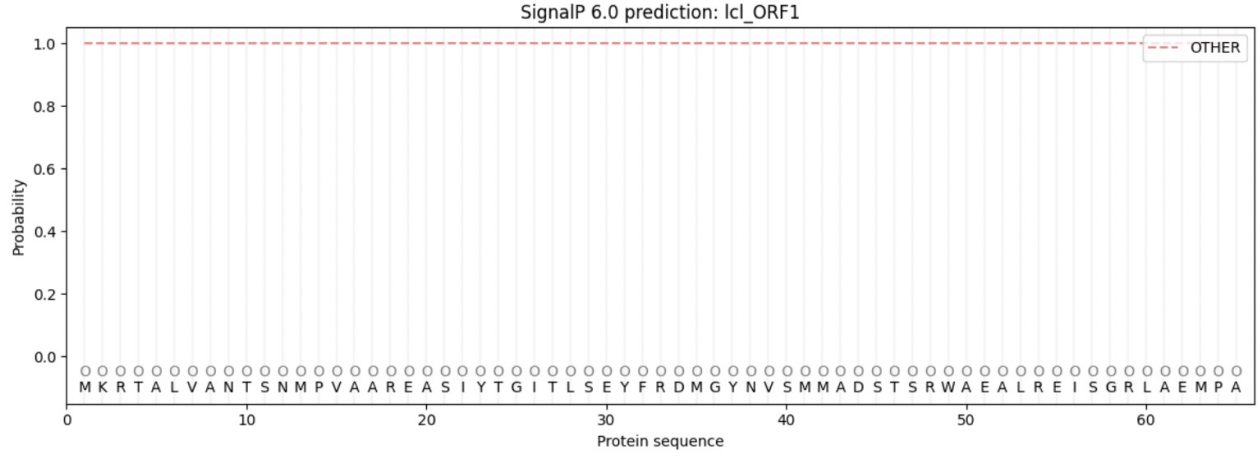
Predicting Secreted Proteins: SignalP

Predicted proteins

lcl_ORF1
Prediction: Other

Protein type	Other	Signal Peptide (Sec/SPI)
Likelihood	0.9999	0.0001

Download: PNG / EPS / Tabular



Predicting function from sequence

Predicting Secreted Proteins: SignalP

1. Copy the longest ORF generated by ORF Finder

```
>lcl|ORF1
MKRTALVANTSNMPVAAREASIYTGITLSEYFRDMGYNVSMMADSTSRWA
EALREISGRLAEMPA
```

2. Paste it in SignalP's `Submit Data` box

3. Repeat with the sequence in the course page

Predicting function from sequence

Predicting Secreted Proteins: SignalP

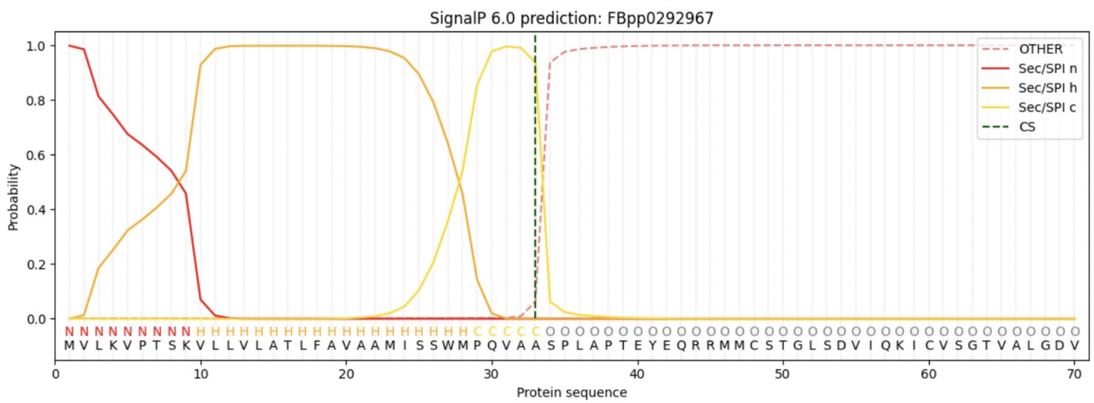
Predicted proteins

FBpp0292967
Prediction: Signal Peptide (Sec/SPI)

Cleavage site between pos. 33 and 34.
Probability 0.937172

Protein type	Other	Signal Peptide (Sec/SPI)
Likelihood	0.0025	0.9975

Download: PNG / EPS / Tabular



Predicting function from sequence

Gene Ontology terms

Biological process

- Behavior
- Cell communication
- Cell growth and maintenance
- Death
- Developmental processes
- Perception of external stimulus
- Physiological processes
- Viral life cycle

Molecular function

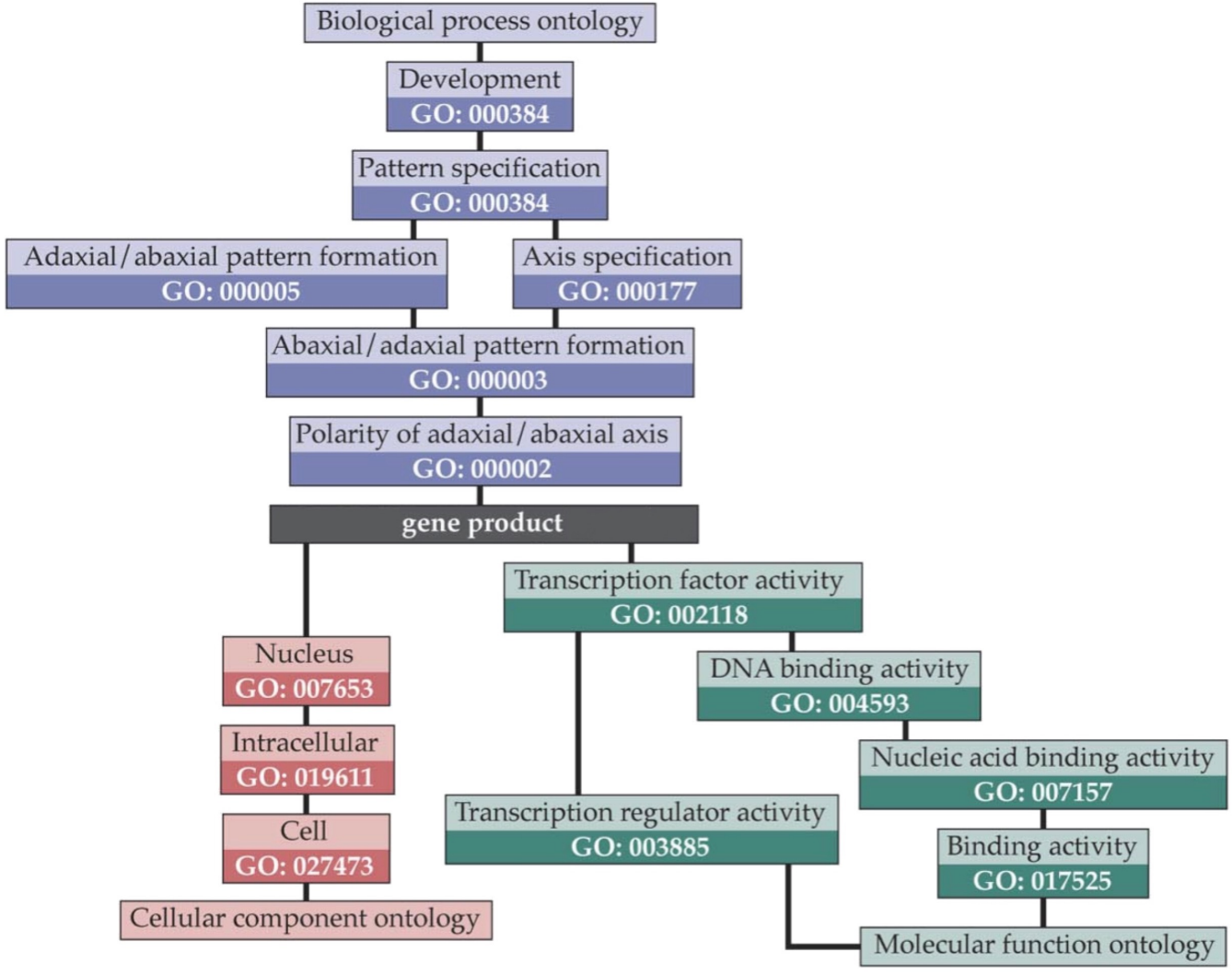
- Antitoxin
- Anticoagulant
- Antioxidant
- Apoptosis regulator
- Cell cycle regulator
- Cytoskeletal regulator
- Defense/immunity protein
- Vitamin transporter
- and many more

Cellular component

- Cell fraction
- Cell wall
- Extracellular
- Intracellular
- Membrane
- Unlocalized

Predicting function from sequence

Gene Ontology terms



RNA-Seq: Annotation

Transcriptome Annotation Workflow

1. **Coding Region Identification:** Detecting open reading frames (ORFs).
2. **Functional Annotation:** Link sequences to known proteins.
3. **Prediction of sequence features:** Identifying conserved functional domains and signal peptides
4. **GO Term Association:** Assigning Gene Ontology terms for functional classification.
5. **Integration:** Compiling results into a single database

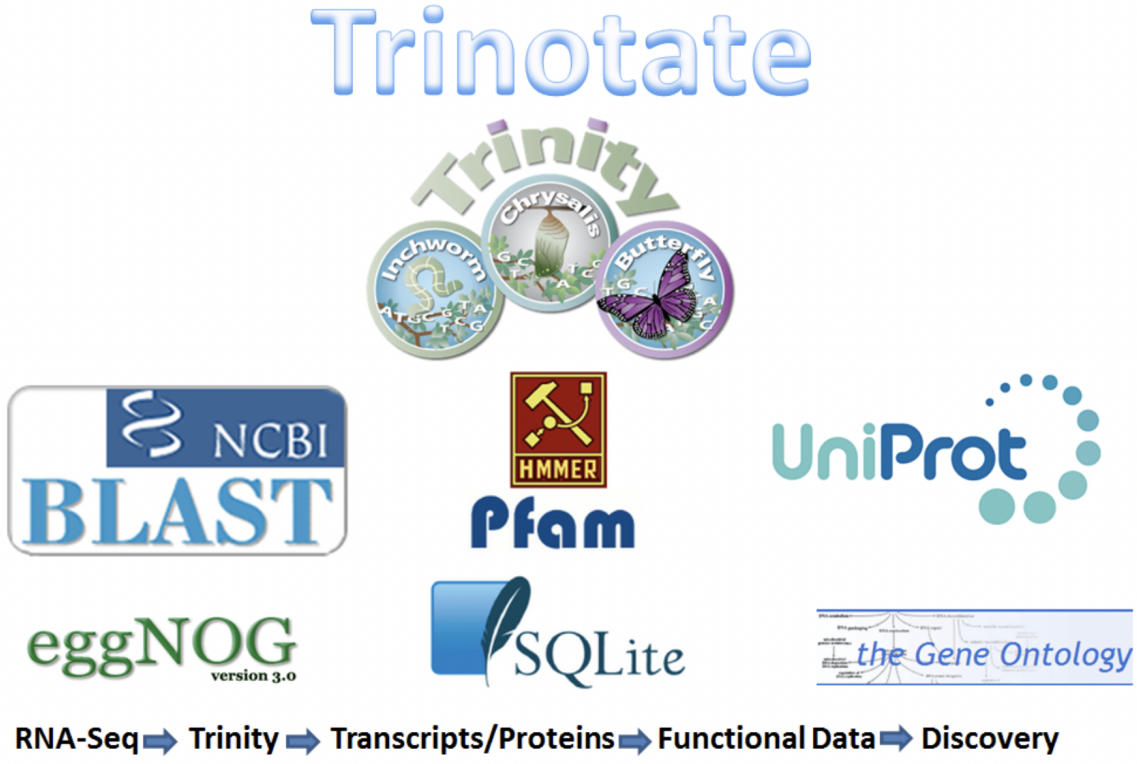
RNA-Seq: Annotation

Transcriptome Annotation Workflow

1. **Coding Region Identification:** Transdecoder
2. **Functional Annotation:** Blast and HMMER.
3. **Domain Search:** signalP and tmhmm.
4. **GO Term Association:** eggNOG-mapper
5. **Integration:** Trinnotate

Predicting function from sequence

Tool for integration



Predicting function from sequence

Identifying Coding Regions with TransDecoder

> TransDecoder predicts open reading frames (ORFs) in transcript sequences. These ORFs represent potential protein-coding regions.

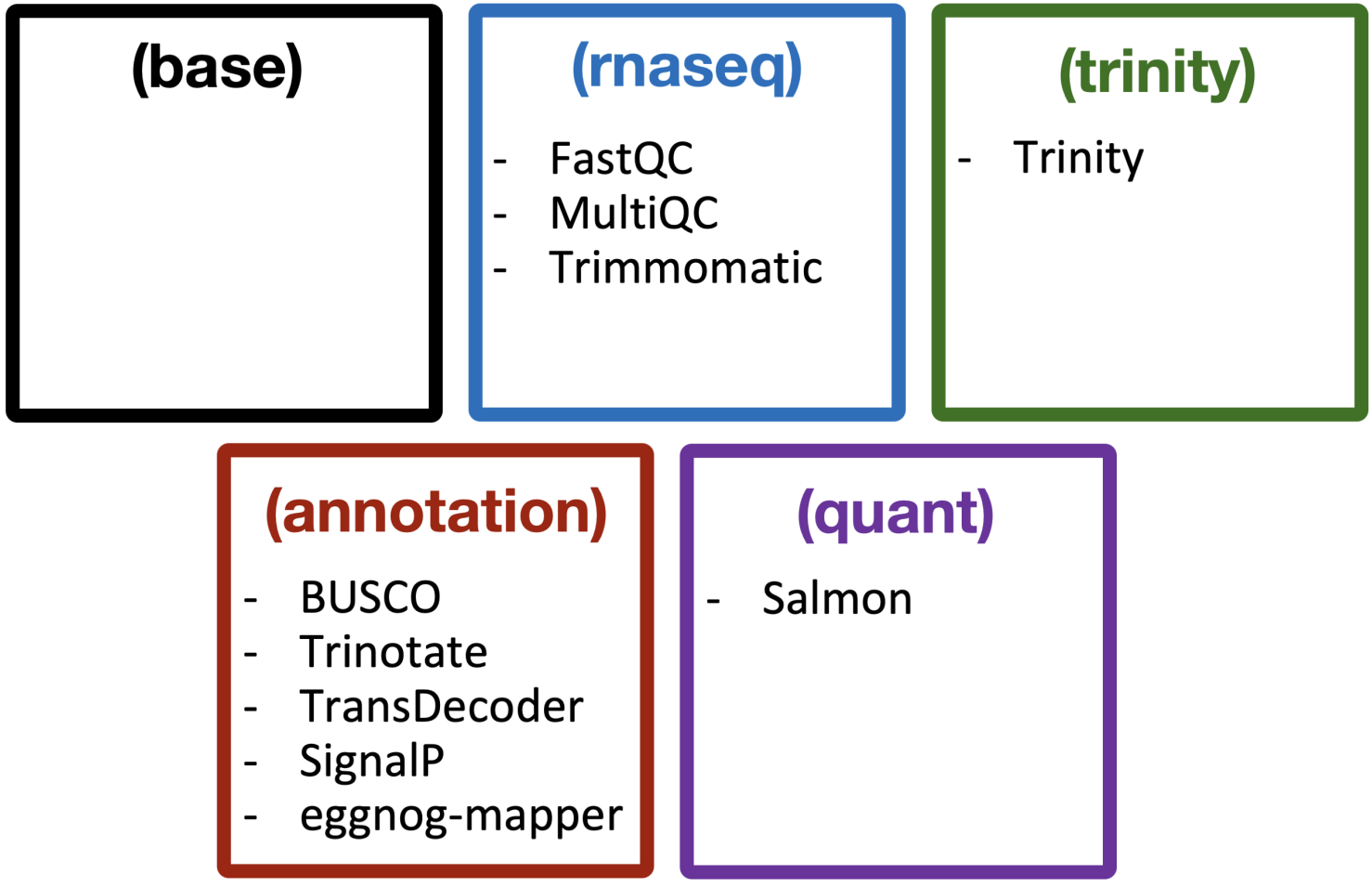
Key Steps:

1. **LongOrfs:** Identifies likely coding regions based on sequence properties.
2. **Predict:** Refines ORF predictions by incorporating similaty and coding potential.

https://github.com/TransDecoder/TransDecoder/wiki

Predicting function from sequence

Conda environments



Predicting function from sequence

Identifying Coding Regions with TransDecoder

Our Trinity files:

```
Trinity --seqType fq \
     --left ~/rnaseq/02-FilteredReads/${SPECIES}_R1_paire
     --right ~/rnaseq/02-FilteredReads/${SPECIES}_R2_pair
     --CPU 6 \
     --max_memory 6G \
     --output ${SPECIES}.trinity.fasta \
     --full_cleanup
```

Fixing names:

```
cd ~/rnaseq/03-Assembly/
mv ${SPECIES}.trinity.fasta.Trinity.fasta ${SPECIES}.Trinity.fasta
mv ${SPECIES}.trinity.fasta.Trinity.fasta.gene_trans_map \
   ${SPECIES}.Trinity.fasta.gene_trans_map
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

1. Create and navigate to a new folder

```
mkdir ~/rnaseq/04-Annotation/
cd ~/rnaseq/04-Annotation/
```

2. Activate environment:

```
conda activate annotation
```

3. Run TransDecoder.LongOrfs:

```
TransDecoder.LongOrfs \
        -t ~/rnaseq/03-Assembly/${SPECIES}.Trinity.fasta
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `base_freqs.dat`

```
cd ${SPECIES}.Trinity.fasta.transdecoder_dir
head base_freqs.dat
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `base_freqs.dat`

```
cd ${SPECIES}.Trinity.fasta.transdecoder_dir
head base_freqs.dat
A       552053  0.306
C       348809  0.194
G       348809  0.194
T       552053  0.306
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `longest_orfs.cds`

```
head longest_orfs.cds
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `longest_orfs.cds`

```
>TRINITY_DN1811_c0_g1_i1.p1 type:5prime_partial \
TRINITY_DN1811_c0_g1_i1:1-639(+)
CAGGATGTCTATAAAATTGGTGGTATTGGTACAGTACCCGTGGGTCGTGTTGAAACT
>TRINITY_DN1811_c0_g1_i1.p2 type:3prime_partial \
TRINITY_DN1811_c0_g1_i1:351-1(-)
ATGAGCAGTATGACAATCCAAAACTGGAGTATAACCATTGGCAATTTGTCCAGGATG
>TRINITY_DN1803_c0_g1_i1.p1 type:internal \
TRINITY_DN1803_c0_g1_i1:1-444(+)
ATCGTACTCCACTTGACAAGCAACATGAAATTTTTAATTGTTTTCGCTTTGGCTTTG
>TRINITY_DN1854_c0_g1_i1.p1 type:internal \
TRINITY_DN1854_c0_g1_i1:311-3(-)
GAAAAACGTAAACTGAAATTGGTATGGCGTAATATAATTGCATTCGGTTATTTGCAT
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `longest_orfs.gff3`

```
head longest_orfs.gff3
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `longest_orfs.gff3`

```
TRINITY_DN1811_c0_g1_i1 transdecoder    gene    1       7
TRINITY_DN1811_c0_g1_i1 transdecoder    mRNA    1       7
TRINITY_DN1811_c0_g1_i1 transdecoder    exon    1       7
TRINITY_DN1811_c0_g1_i1 transdecoder    CDS     1       6
TRINITY_DN1811_c0_g1_i1 transdecoder    three_prime_UTR 6
TRINITY_DN1811_c0_g1_i1 transdecoder    gene    1       7
TRINITY_DN1811_c0_g1_i1 transdecoder    mRNA    1       7
TRINITY_DN1811_c0_g1_i1 transdecoder    five_prime_UTR  3
TRINITY_DN1811_c0_g1_i1 transdecoder    exon    1       7
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `longest_orfs.pep`

```
head longest_orfs.pep
```

Predicting function from sequence

Identifying Coding Regions with TransDecoder

4. Check files: `longest_orfs.pep`

```
>TRINITY_DN1811_c0_g1_i1.p1 type:5prime_partial gc:univers
QDVYKIGGIGTVPVGRVETGILKPGMVVNFAPVNLVTEVKSVEMHHEALSEAMPGDN
>TRINITY_DN1811_c0_g1_i1.p2 type:3prime_partial gc:univers
MSSMTIQNWSITIGNLSRMIKNNYLSCKISSSLRRTIFGITGNITATQFLNRNVFNI
>TRINITY_DN1803_c0_g1_i1.p1 type:internal gc:universal TRI
IVLHLTSNMKFLIVFALALATSASAELVSRSVVVPVLENEGRITNGQTASVGQFPYQ
>TRINITY_DN1854_c0_g1_i1.p1 type:internal gc:universal TRI
EKRKLKLVWRNIIAFGYLHLAALYGAYLLFTSAKWQTIAFAFGLYVVSGLGITAGAH
>TRINITY_DN1821_c0_g1_i1.p1 type:internal gc:universal TRI
SEVKVKVRINHHGIVLISSASLVDKKELEEPQTPQPPEQQINTEQPASGEGQPANAG
```

Predicting function from sequence

Identify ORFs with similarity to known proteins

BLAST

> Search a protein database such as Swissprot (fast) or Uniref90 (slow but more comprehensive) using BLAST+

Predicting function from sequence

Identify ORFs with similarity to known proteins

5. Prepare databases

```
cd ~/rnaseq/00-Databases/
gunzip uniprot_sprot.fasta.gz
gunzip Pfam-A.hmm.gz
makeblastdb -in uniprot_sprot.fasta -dbtype prot
hmmpress Pfam-A.hmm
```

Predicting function from sequence

Identify ORFs with similarity to known proteins

5. Search a protein database such as Swissprot (fast) or Uniref90 (slow but more comprehensive) using BLAST+

```
cd ~/rnaseq/04-Annotation/
blastp -query ${SPECIES}.Trinity.fasta.transdecoder_dir/longest_orfs.
       -db uniprot_sprot.fasta  -max_target_seqs 1 \
       -outfmt 6 -evalue 1e-5 -num_threads 8 > blastp.out
```

- -query: Input file name
- -db: BLAST database name
- -max_target_seqs: Maximum number of aligned seqs to keep
- -outfmt: alignment view options (6 = Tabular)
- -evalue: Expectation value (E) threshold for saving hits
- -num_threads
- >

Identify ORFs with similarity to known proteins

BLAST results

```
head blastp.out
TRINITY_DN1811_c0_g1_i1.p1      sp|P05303|EF1A2_DROME   9
TRINITY_DN1803_c0_g1_i1.p1      sp|P08897|COGS_HYPLI    4
TRINITY_DN1854_c0_g1_i1.p1      sp|Q9BH41|FAD9_ACHDO    7
TRINITY_DN1805_c0_g1_i1.p1      sp|C0HK92|Y5076_DROME   7
TRINITY_DN1864_c0_g1_i1.p1      sp|Q9VHG4|RENR_DROME    5
TRINITY_DN1834_c0_g1_i1.p1      sp|Q00449|MDR49_DROME   6
TRINITY_DN1855_c0_g1_i1.p1      sp|Q27331|VATA2_DROME   9
TRINITY_DN1826_c0_g1_i1.p1      sp|Q7KN62|TERA_DROME    9
TRINITY_DN1826_c0_g1_i1.p1      sp|Q7KN62|TERA_DROME    4
TRINITY_DN1832_c0_g1_i1.p1      sp|P55828|RS20_DROME    9
```

BLAST results

Columns in BLAST `-outfmt 6` (default format)

1. **qseqid** – Query sequence ID.
2. **sseqid** – Subject sequence ID (from the database).
3. **pident** – Percentage of identical matches in the alignment.
4. **length** – Alignment length (number of aligned bases/amino acids).
5. **mismatch** – Number of mismatched positions in the alignment.
6. **gapopen** – Number of gap openings in the alignment.
7. **qstart** – Start position of the alignment in the query sequence.
8. **qend** – End position of the alignment in the query sequence.
9. **sstart** – Start position of the alignment in the subject sequence.
10. **send** – End position of the alignment in the subject sequence.
11. **evalue** – E-value (statistical measure of alignment significance).
12. **bitscore** – Bit score of the alignment (higher values indicate better alignments).

Predicting function from sequence

Identify Functional Domains

HMMER

> Functional domains are conserved regions within proteins responsible for specific biochemical activities. HMMER uses Hidden Markov Models (HMMs) to identify these domains.

Predicting function from sequence

Identify Functional Domains

HMMER

6. Search for functional domains using HMMER

```
hmmscan --cpu 8 --domtblout ${SPECIES}.hmmscanPFAM.out Pf
    ${SPECIES}.Trinity.fasta.transdecoder_dir/longest_orf
```

➡️ -cpu: number of threads

➡️ -domtblout: save parseable table of per-domain hits to file

Predicting function from sequence

Identify Functional Domains

HMMER

```
head hmmscanPFAM.out
#
# target name         accession   tlen query name
#-------------------- ---------- ----- ------------
GTP-eEF1A_C           PF22594.2    101 TRINITY_DN1811_c0_g
GTP_EFTU_D3           PF03143.23   106 TRINITY_DN1811_c0_g
GTP_EFTU_D3           PF03143.23   106 TRINITY_DN1811_c0_g
GTP_EFTU_D2           PF03144.31    73 TRINITY_DN1811_c0_g
GTP_EFTU_D2           PF03144.31    73 TRINITY_DN1811_c0_g
GTP_EFTU_D4           PF14578.12    86 TRINITY_DN1811_c0_g
GTP_EFTU_D4           PF14578.12    86 TRINITY_DN1811_c0_g
```

Predicting function from sequence

Identify Functional Domains

> Identifying signal peptides during genome or transcriptome annotation is crucial for understanding the functionality and localization of proteins. Signal peptides are short amino acid sequences at the N-terminal of a protein that direct the protein to specific cellular compartments, often for secretion or localization to organelles like the endoplasmic reticulum (ER), Golgi apparatus, or lysosomes.

Predicting function from sequence

Identify signal peptides

SignalP, this step will not be done during the course

1. Run signalp

```
signalp6 --fastafile longest_orfs.pep \
         --output_dir ${SPECIES}_signalp \
         --organism eukarya
         -f short \
         -n signalp.out
```

Predicting function from sequence

Annotating Gene Ontology (GO) Terms

Gene Ontology (GO) terms classify gene functions into three main categories:

1. **Biological Processes:** Pathways and processes.
2. **Molecular Functions:** Activities performed by gene products.
3. **Cellular Components:** Locations within the cell.

RNA-Seq: Annotation

Annotating Gene Ontology (GO) Terms

EggNOG-Mapper, this step will not be done during the course

> EggNOG-mapper is a tool for fast functional annotation of novel sequences. It uses precomputed Orthologous Groups (OGs) and phylogenies from the EggNOG database (http://eggnog5.embl.de) to transfer functional information from fine-grained orthologs only.

RNA-Seq: Annotation

Annotating Gene Ontology (GO) Terms

Eggnog-Mapper

1. Run

```
emapper.py -i longest_orfs.pep \
           -o eggnog \
           --output_dir ~/rnaseq/04-Annotation/${SPECIES}/
           --cpu 8 \
           --tax_scope 6656
```

- -i: input file for annotation
- -o: output file
- --output_dir: output directory
- --cpu 8: number of threads
- --tax_scope: taxon (6656 for Arthropoda)

RNA-Seq: Annotation

Integrating Results with Trinotate

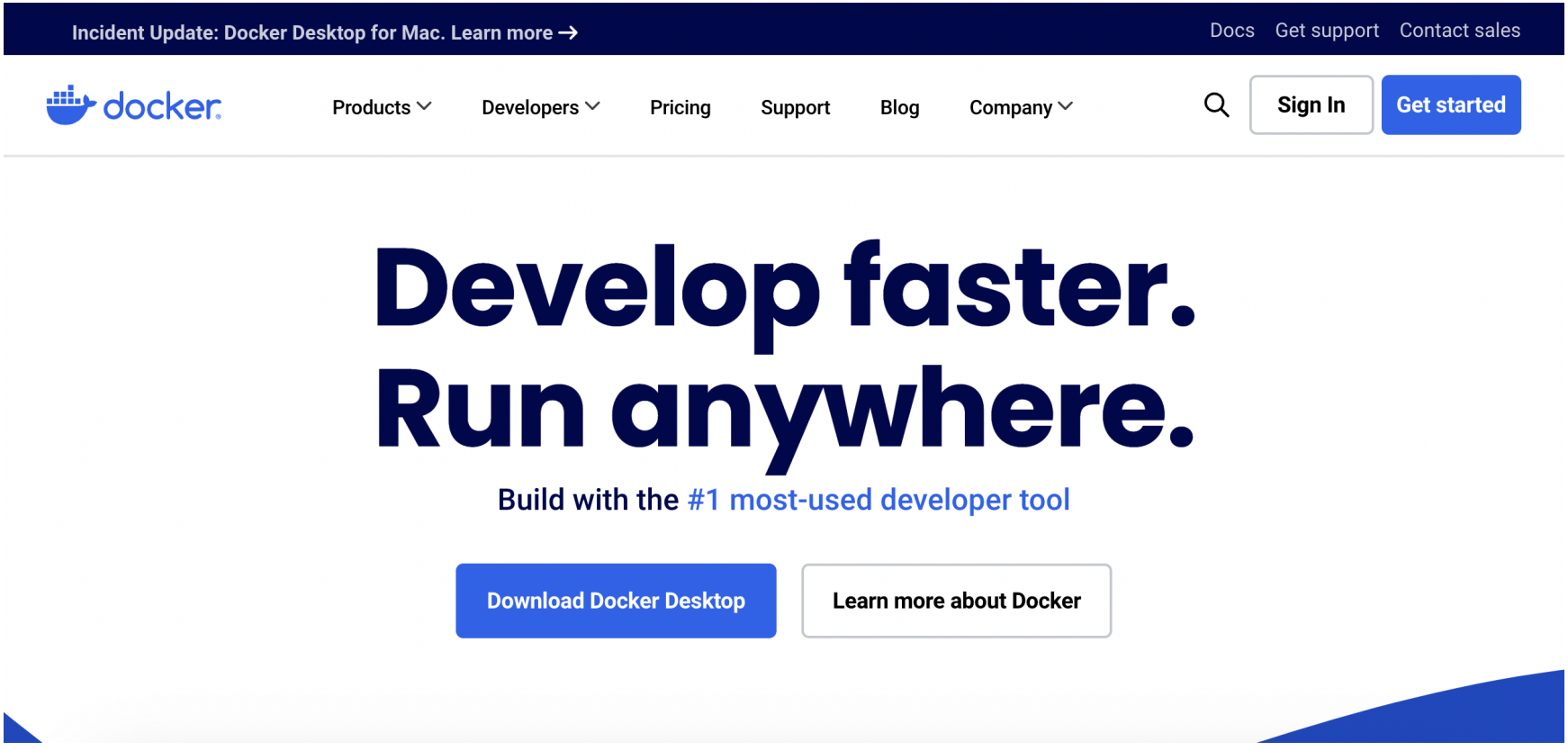
What is Trinotate?

> Trinotate is a comprehensive annotation platform that integrates multiple data sources into a single SQLite database, simplifying downstream analyses.

Installing software using Docker

Issues with any installation

> Trinotate in Conda does not work for me



Installing software using Docker

Issues with Trinotate installation

1. Download docker.
2. Follow the instructions for installation
3. Open Docker Desktop and create a login
4. On Terminal:

```
docker --version
```

Installing software using Docker

Issues with Trinotate installation

5. Download Trinotate docker image

```
docker pull trinityrnaseq/trinotate
```

6. Run Trinotate interactively

```
docker run --rm -it \
                  -v `pwd`:/data \
                  -v /tmp:/tmp \
                  -e TRINOTATE_HOME=/usr/local/src/Trinota
                  trinityrnaseq/trinotate bash
```

7. Test Trinotate

```
$TRINOTATE_HOME/Trinotate
```

RNA-Seq: Annotation

Integrating Results with Trinotate

1a. Activante conda environment

```
conda activate annotation
```

1b. Activate docker container

```
docker run --rm -it \
           -v `pwd`:/data \
           -v /tmp:/tmp \
           -e TRINOTATE_HOME=/usr/local/src/Trinotate \
           trinityrnaseq/trinotate bash
```

RNA-Seq: Annotation

Integrating Results with Trinotate

2a. Prepare the Trinotate SQLite database in conda

```
Trinotate --create \
          --db ${SPECIES}_Trinotate.sqlite \
          --trinotate_data_dir ~/rnaseq/04-Annotation/\
          ${SPECIES}/Trinotate
```

2b. Prepare the Trinotate SQLite database in docker

```
$TRINOTATE_HOME/Trinotate --create \
          --db ${SPECIES}_Trinotate.sqlite \
          --trinotate_data_dir /data/rnaseq/04-Annotation/
          ${SPECIES}/Trinotate
```

RNA-Seq: Annotation

Integrating Results with Trinotate

Trinotate SQLite database

```
Trinotate --create \
          --db ${SPECIES}_Trinotate.sqlite \
          --trinotate_data_dir ~/rnaseq/04-Annotation/\
          ${SPECIES}/Trinotate
```

- --create: create database
- --db: database name
- --trinotate_data_dir: output directory

RNA-Seq: Annotation

Integrating Results with Trinotate

3a. Integrating results (conda)

```
Trinotate --db ${SPECIES}_Trinotate.sqlite --init \
--gene_trans_map /data/rnaseq/03-Assembly/${SPECIES}/${SP
--transcript_fasta /data/rnaseq/03-Assembly/${SPECIES}/${
--transdecoder_pep /data/rnaseq/04-Annotation/${SPECIES}/
```

3b. Integrating results (docker)

```
$TRINOTATE_HOME/Trinotate --db ${SPECIES}_Trinotate.sqlit
--gene_trans_map /data/rnaseq/03-Assembly/${SPECIES}/${SP
--transcript_fasta /data/rnaseq/03-Assembly/${SPECIES}/${
--transdecoder_pep /data/rnaseq/04-Annotation/${SPECIES}/
```

RNA-Seq: Annotation

Integrating Results with Trinotate

Integrating results

```
$TRINOTATE_HOME/Trinotate --db ${SPECIES}_Trinotate.sqlit
--gene_trans_map /data/rnaseq/03-Assembly/${SPECIES}/${SP
--transcript_fasta /data/rnaseq/03-Assembly/${SPECIES}/${
--transdecoder_pep /data/rnaseq/04-Annotation/${SPECIES}/
```

- --db: database name
- --gene_trans_map: Trinity file map isoforms/genes
- --transcript_fasta: Trinity assembled transcripts
- -transdecoder_pep: Transdecoder ORFs

RNA-Seq: Annotation

Integrating Results with Trinotate

Integrating results

With the database set, we can load several data:

```
Trinotate --db <sqlite.db> --LOAD_swissprot_blastp <file.outfmt6>
Trinotate --db <sqlite.db> --LOAD_pfam <file>
Trinotate --db <sqlite.db> --LOAD_signalp <file>
Trinotate --db <sqlite.db> --LOAD_EggnogMapper <file>
Trinotate --db <sqlite.db> --LOAD_tmhmmv2 <file>
Trinotate --db <sqlite.db> --LOAD_deeptmhmm <file.gff3
```

Expression data can also be loaded in Trinotate database

RNA-Seq: Annotation

Integrating Results with Trinotate

Loading BLAST results against SwissProt database

```
$TRINOTATE_HOME/Trinotate \
             --db ${SPECIES}_Trinotate.sqlite \
             --LOAD_swissprot_blastp blastp.out
```

RNA-Seq: Annotation

Integrating Results with Trinotate

Loading HMMScan results

```
$TRINOTATE_HOME/Trinotate \
             --db ${SPECIES}_Trinotate.sqlite \
             --LOAD_pfam hmmscanPFAM.out
```

RNA-Seq: Annotation

Integrating Results with Trinotate

Loading signalp results

```
$TRINOTATE_HOME/Trinotate \
             --db ${SPECIES}_Trinotate.sqlite \
             --LOAD_signalp signalp/prediction_results.tx
```

RNA-Seq: Annotation

Integrating Results with Trinotate

Loading EggnogMapper results

```
$TRINOTATE_HOME/Trinotate \
              --db ${SPECIES}_Trinotate.sqlite \
              --LOAD_EggnogMapper eggnog.emapper.annotatio
```

RNA-Seq: Annotation

Generate the Trinotate Annotation Report

Check if the database was populated

```
$TRINOTATE_HOME/Trinotate \
             --db ${SPECIES}_Trinotate.sqlite \
             --report >${SPECIES}_TrinotateReport.xls
```

RNA-Seq: Annotation

The Trinotate Annotation Report

The report is a tab-delimited output with the following columns:

```
0       #gene_id
1       transcript_id
2       sprot_Top_BLASTX_hit
3       infernal
4       prot_id
5       prot_coords
6       sprot_Top_BLASTP_hit
7       Pfam
8       SignalP
9       TmHMM
10      eggnog
11      Kegg
12      gene_ontology_BLASTX
13      gene_ontology_BLASTP
14      gene_ontology_Pfam
15      transcript # optional, use --incl_trans
16      peptide # optional, use --incl_pep
```