

# genotypst: A bioinformatics Typst package for sequence analysis and visualization

`genotypst` is a bioinformatics package for Typst that enables analysis and visualization of biological data. It provides functionality for parsing FASTA and Newick files and generating publication-ready visualizations, including multiple sequence alignments, sequence logos, and phylogenetic trees.

## Contents

Working with sequence data .....	2
Loading data .....	2
FASTA rendering .....	2
Multiple sequence alignments .....	2
Sequence logos .....	3
Color palettes .....	4
Amino acid palette .....	4
Nucleic acid palettes .....	4
Working with phylogenetic trees .....	5
Visualizing trees .....	5
Customizing visualizations .....	6
Font selection .....	6
Bibliography .....	8

## Working with sequence data

`genotypst` provides functions to parse sequence data and produce different visualizations.

### Loading data

The `parse-fasta-file` function reads FASTA files and returns a dictionary mapping sequence identifiers to their corresponding sequences.

```
#let sequences = parse-fasta(read("/docs/data/dna.fna"))

(
  seq_1: "AAGGGACACTGATTTTCTCCACAGCTGGGCCGTGGACCGTAGTGTTTCAAGAAGCCCACACAC",
  seq_2: "GCAATGGAGACAACATAGCCAACTACCTACTAGATGGCCTAGATCTGCCGCA",
  seq_3: "GGAAGTGGCGTTACAGACAGTTGTGAGCCACCACATGGGCCTGGGATTTAAATATTATAAAGCTCCTC",
)
```

### FASTA rendering

Use `render-fasta` to display sequences in the standard FASTA format.

```
#render-fasta(sequences, max-width: 50)

>seq_1
AAGGGACACTGATTTTCTCCACAGCTGGGCCGTGGACCGTAGTGTTTCA
AGAAGCCCACACAC
>seq_2
GCAATGGAGACAACATAGCCAACTACCTACTAGATGGCCTAGATCTGCCG
CA
>seq_3
GGAAGTGGCGTTACAGACAGTTGTGAGCCACCACATGGGCCTGGGATTTA
AATATTATAAAGCTCCTC
```

In this example, `max-width` controls how many characters appear per line (default is 60).

### Multiple sequence alignments

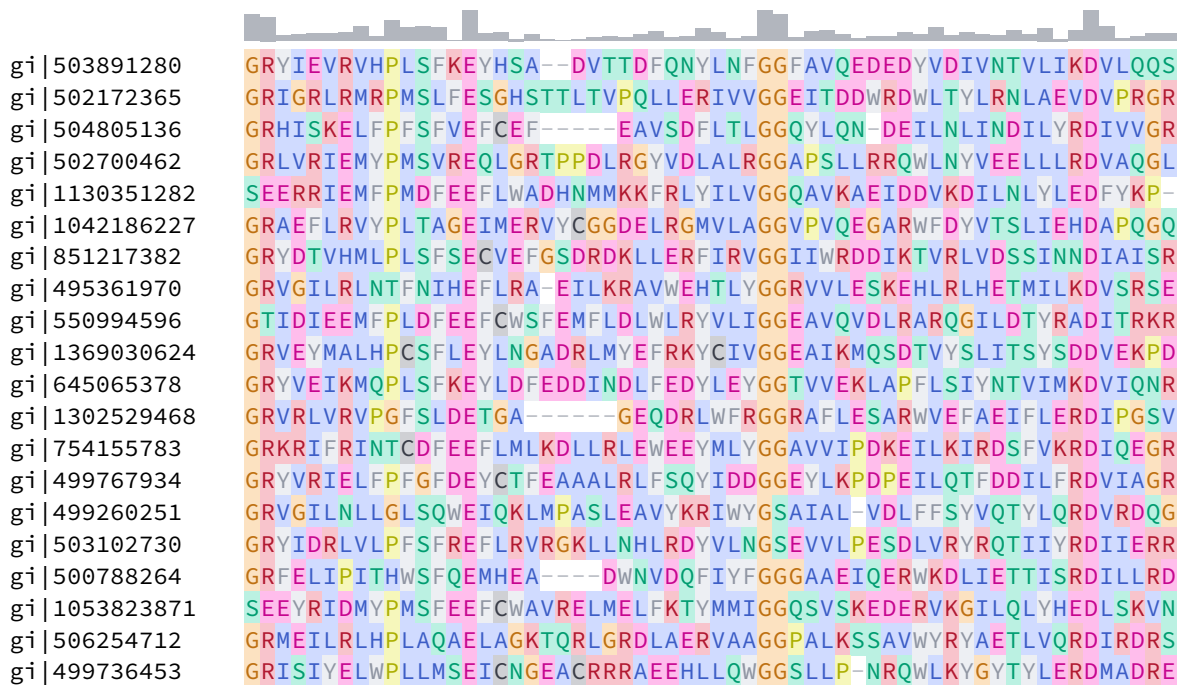
The `render-msa` function displays multiple sequence alignments with optional residue coloring and conservation bars.

In the example below:

- `colors: true` enables residue coloring based on biochemical properties.
- `conservation: true` adds conservation bars above the alignment.
- `start: 100` and `end: 160` limit the display to a specific region of interest (residues 100 to 160).

```
#let protein_msa = parse-fasta(read("/docs/data/msa.afa"))

#render-msa(
  protein_msa,
  start: 100,
  end: 160,
  colors: true,
  conservation: true,
)
```



**Figure 1.** MSA visualization for positions 100–160, with residue coloring and conservation bars enabled.

Residue coloring represents amino acid physicochemical properties. The sequence alphabet (amino acid, DNA, or RNA) is determined automatically and a suitable color palette is applied.

The bars above the alignment indicate the degree of conservation at each column.

## Sequence logos

Sequence logos<sup>1</sup> summarize conservation patterns within a sequence alignment and are commonly used to visualize binding sites, motifs, and functional domains. In a sequence logo, the total height of each stack represents the information content (in bits) at that position, while the height of individual letters reflects their relative frequencies.

In the example below, we visualize the same region as the MSA of the previous section (positions 100 to 160).

```
#render-sequence-logo(protein_msa, start: 100, end: 160)
```



**Figure 2.** Sequence logo for positions 100–160, showing conservation and residue frequency.

Like `render-msa`, `render-sequence-logo` automatically applies the appropriate color palette based on the sequence alphabet.

## Color palettes

`genotypst` uses predefined color palettes to assign colors to sequence residues.

### Amino acid palette

Amino acids are colored according to their physicochemical properties. Grouping residues by color helps reveal the chemical nature of conserved positions (e.g., whether a position is consistently hydrophobic or charged), which is often important for understanding protein structure, function, and evolution.

#### Hydrophobic

Alanine	Ala	A	#4d78ff
Histidine	His	H	#4d78ff
Isoleucine	Ile	I	#4d78ff
Leucine	Leu	L	#4d78ff
Methionine	Met	M	#4d78ff
Valine	Val	V	#4d78ff

#### Negatively charged

Aspartic acid	Asp	D	#ff07b8
Glutamic acid	Glu	E	#ff07b8

#### Positively charged

Lysine	Lys	K	#e51f3e
Arginine	Arg	R	#e51f3e

#### Polar

Serine	Ser	S	#00c990
Threonine	Thr	T	#00c990
Glutamine	Gln	Q	#00c990
Asparagine	Asn	N	#00c990

#### Cysteine

Cysteine	Cys	C	#494e5b
----------	-----	---	---------

#### Glycine

Glycine	Gly	G	#f59116
---------	-----	---	---------

#### Aromatic

Phenylalanine	Phe	F	#bac1d2
Tryptophan	Trp	W	#bac1d2
Tyrosine	Tyr	Y	#bac1d2





#### Proline

Proline	Pro	P	#dce100
---------	-----	---	---------





### Nucleic acid palettes

The DNA and RNA palettes assign a distinct color to each nucleotide.

### DNA palette

Adenine	A	
Cytosine	C	
Guanine	G	
Thymine	T	

### RNA palette

Adenine	A	
Cytosine	C	
Guanine	G	
Uracil	U	

## Working with phylogenetic trees

`genotypst` includes functions to parse and render phylogenetic trees. Trees can be created by parsing Newick-formatted strings with `parse-newick` or by manually constructing nested dictionary structures.

```
#let parsed_newick_tree = parse-newick(  
  "((('Leaf A':0.2,'Leaf B':0.1)'Internal node':0.3,'Leaf C':0.6)Root;"  
)  
  
#let manual_tree = (  
  rooted: true,  
  name: "Root",  
  length: none,  
  children: (  
    (  
      name: "Internal node",  
      length: 0.3,  
      children: (  
        (name: "Leaf A", length: 0.2, children: none),  
        (name: "Leaf B", length: 0.1, children: none),  
      ),  
    ),  
    (name: "Leaf C", length: 0.6, children: none),  
  ),  
)
```

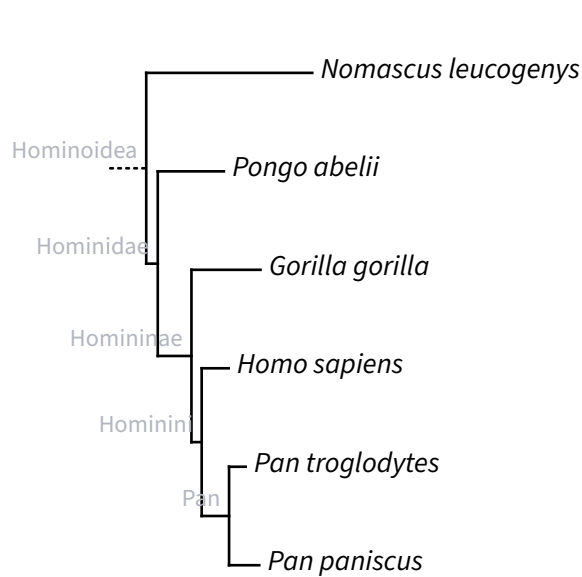
## Visualizing trees

`genotypst` can produce visualizations of phylogenetic trees. To illustrate this, we will read and render a Newick file containing a phylogeny of the *Hominoidea* superfamily, which was extracted from the Ensembl Compara species tree<sup>2</sup>.

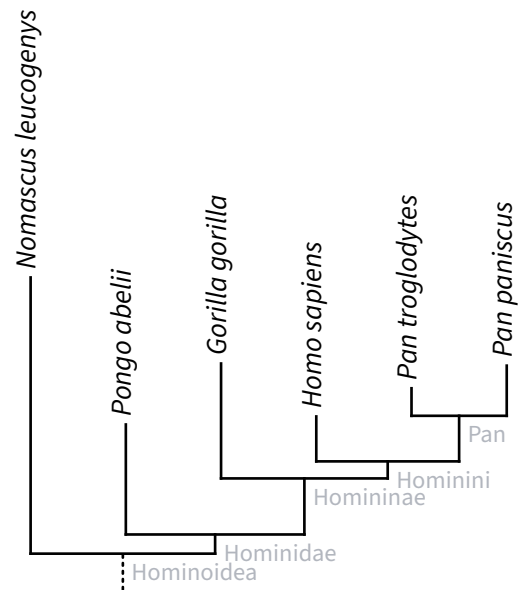
```
#let hominoidea_tree = parse-newick(read("/docs/data/hominoidea.nwk"))
```

To render the tree, use the `render-tree` function. By default, it produces a horizontal rectangular dendrogram, but a vertical layout can be specified using the `orientation: "vertical"` option.

```
#render-tree(hominoidea_tree, tip-label-italics: true, orientation:  
  "horizontal")  
#render-tree(hominoidea_tree, tip-label-italics: true, orientation: "vertical")
```



Tree with horizontal orientation



Tree with vertical orientation

## Customizing visualizations

### Font selection

By default, `render-fasta` and `render-msa` inherit the monospaced font used for raw text in your document. To use a different font, wrap the rendering function in a `context` block with a custom font for raw text.

```
#let dna_msa = (
  "seq1": "AGTCTCAAGATAACTTTGAAACAAACTTC",
  "seq2": "AGTTTCCAAGTGGATTTGGAATTGAACTTT",
  "seq3": "ACTCT-CGGATGGATTTCGGATACAAACTTT",
  "seq4": "AGTCT---GATTGATGTGGATACAAACTTC",
  "seq5": "AGTCT--GGGTGGATTTGG-AACAAATTTT",
  "seq6": "CAGTGCTCCCTGGTGGTGG-ACCATCTTAC",
  "seq7": "AGTCTCAAGACGGATACTG--ATGCCCTAT",
)

#context {
  show raw: set text(font: "Maple Mono")
  render-msa(dna_msa)
}
```

```
seq1 AGTCTCAAGATAACTTTTCGAAACAAACTTC
seq2 AGTTTCCAAGTGGATTTGGAATTGAACTTT
seq3 ACTCT-CGGATGGATTCGGATACAAACTTT
seq4 AGTCT---GATTGATGTGGATACAAACTTC
seq5 AGTCT--GGGTGGATTTGG-AACAAATTTT
seq6 CAGTGCTCCCTGGTGGTGG-ACCATCTTAC
seq7 AGTCTCAAGACGGATACTG--ATGCCCTAT
```

Default document font for raw text

```
seq1 AGTCTCAAGATAACTTTTCGAAACAAACTTC
seq2 AGTTTCCAAGTGGATTTGGAATTGAACTTT
seq3 ACTCT-CGGATGGATTCGGATACAAACTTT
seq4 AGTCT---GATTGATGTGGATACAAACTTC
seq5 AGTCT--GGGTGGATTTGG-AACAAATTTT
seq6 CAGTGCTCCCTGGTGGTGG-ACCATCTTAC
seq7 AGTCTCAAGACGGATACTG--ATGCCCTAT
```

Custom font (Maple Mono)

Sequence logos and trees are rendered using the default document font, rather than the monospaced font for raw text. To specify a custom font sequence logos and trees, use a `show text` rule instead.

```
#context {
  show text: set text(font: "New Computer Modern")
  render-sequence-logo(dna_msa)
}
```

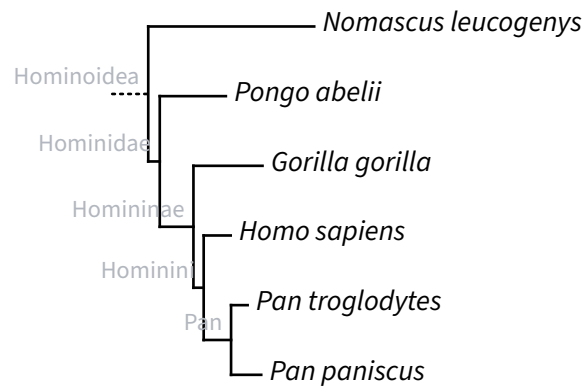


Default document font

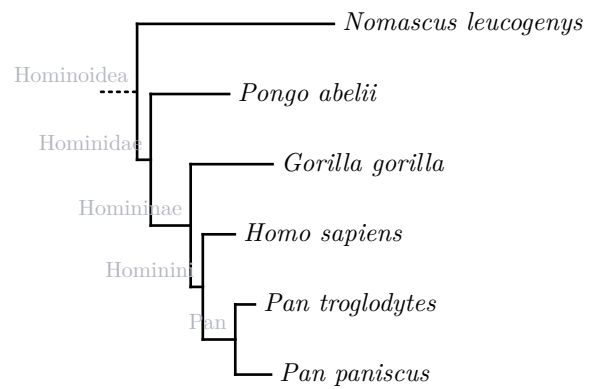


Custom font (New Computer Modern)

```
#let hominoidea_tree = parse-newick(read("/docs/data/hominoidea.nwk"))
#context {
  show text: set text(font: "New Computer Modern", size: 0.9em)
  #render-tree(hominoidea_tree, tip-label-italics: true)
}
```



Default document font



Custom font (New Computer Modern)

## Bibliography

1. Schneider, T. D. & Stephens, R. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research* **18**, 6097–6100 (1990).
2. Herrero, J. *et al.* Ensembl comparative genomics resources. *Database* **2016**, bav96 (2016).