

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, genome maps, 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
Visualizing genomic loci with genome maps	5
Working with phylogenetic trees	6
Visualizing trees	6
Customizing visualizations	7
Font selection	7
Bibliography	9

Working with sequence data

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

Loading data

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

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

(
  seq_1: "AAGGGACACTGATTTCTCCCACAGCTGGCCGTGGACCGTAGTGTTCAGAACGCCACACAC",
  seq_2: "GCAATGGAGACAACATAGCCAACCTACCTACTAGATGCCCTAGATCTGCCGCA",
  seq_3: "GGAACCTGGCGTTACAGACAGTTGTGAGCCACCACATGGGCCTGGGATTAAATTATAAAGCTCCTC",
)
```

FASTA rendering

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

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

>seq_1
AAGGGACACTGATTTCTCCCACAGCTGGCCGTGGACCGTAGTGTTC
AGAACGCCACACAC
>seq_2
GCAATGGAGACAACATAGCCAACCTACCTACTAGATGCCCTAGATCTGCCG
CA
>seq_3
GGAACCTGGCGTTACAGACAGTTGTGAGCCACCACATGGGCCTGGGATT
AAATTATAAAGCTCCTC
```

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 logos¹ 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)
```

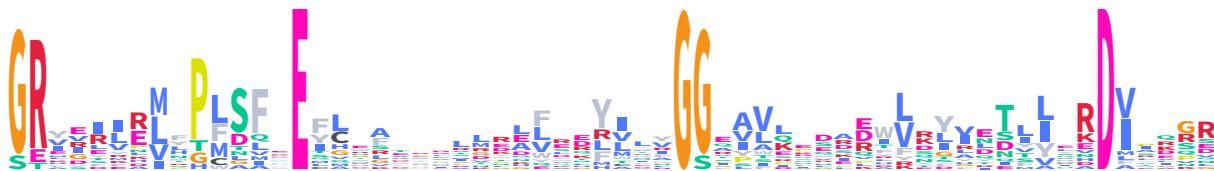


Figure 2. Sequence logo for positions 100–160, showing conservation and residue frequency.

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				Negatively charged			
Alanine	Ala	A	#4d78ff	Aspartic acid	Asp	D	#ff07b8
Histidine	His	H	#4d78ff	Glutamic acid	Glu	E	#ff07b8
Isoleucine	Ile	I	#4d78ff	Positively charged			
Leucine	Leu	L	#4d78ff	Lysine	Lys	K	#e51f3e
Methionine	Met	M	#4d78ff	Arginine	Arg	R	#e51f3e
Valine	Val	V	#4d78ff	Cysteine			
Polar				Cysteine	Cys	C	#494e5b
Serine	Ser	S	#00c990	Glycine			
Threonine	Thr	T	#00c990	Glycine	Gly	G	#f59116
Glutamine	Gln	Q	#00c990	Proline			
Asparagine	Asn	N	#00c990	Proline	Pro	P	#dce100
Aromatic							
Phenylalanine	Phe	F	#bac1d2				
Tryptophan	Trp	W	#bac1d2				
Tyrosine	Tyr	Y	#bac1d2				

Nucleic acid palettes

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

DNA palette

Adenine	A	#00c990
Cytosine	C	#4d78ff
Guanine	G	#ff07b8
Thymine	T	#f59116

RNA palette

Adenine	A	#00c990
Cytosine	C	#4d78ff
Guanine	G	#ff07b8
Uracil	U	#f59116

Visualizing genomic loci with genome maps

Genome maps enable visualization of the genes and other genomic elements within a locus, highlighting their order, orientation, and length. `genotypst` provides a `render-genome-map` function that produces a genome map from an array of dictionaries, each representing a genomic feature that will be plotted:

- `start` (required): Start coordinate.
- `end` (required): End coordinate.
- `strand`: Feature orientation (1 or "+" for the positive strand, -1 or "-" for the negative strand). none draws an undirected block.
- `label`: Feature label
- `color`: Fill color.

```
#let f_plasmid_locus = (
  (start: 65556, end: 66065, strand: -1, label: [_ygfA_]),
  (start: 66118, end: 66407, label: [_oriT_], color: rgb("#696975")),
  (start: 66479, end: 66862, strand: 1, label: [_traM_], color: rgb("#62B9F2")),
  (start: 66977, end: 67055, strand: -1, label: [_finP_]),
  (start: 67049, end: 67738, strand: 1, label: [_traJ_], color: rgb("#F7ED6C")),
  (start: 67837, end: 68232, strand: 1, label: [_traY_]),
  (start: 68265, end: 68630, strand: 1, label: [_traA_]),
  (start: 68645, end: 68956, strand: 1, label: [_traL_]),
  (start: 68978, end: 69544, strand: 1, label: [_traE_]),
)

#render-genome-map(
  f_plasmid_locus,
  coordinate-axis: true,
  unit: "bp",
)
```

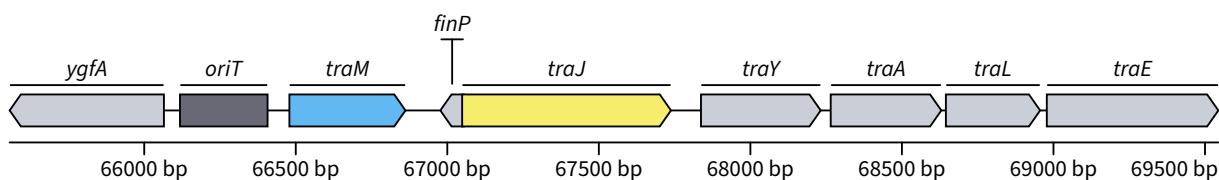


Figure 3. Genome map showing the genes within the 65,556–69,544 bp region of the F plasmid of *Escherichia coli* K-12 (GenBank: AP001918.1).

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),
  ),
)
```

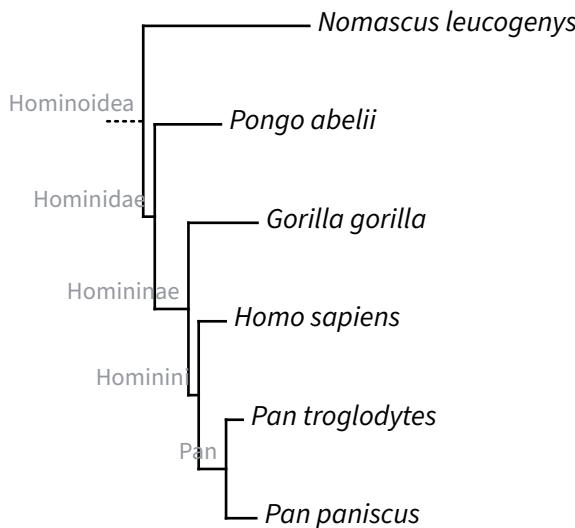
Visualizing trees

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 tree².

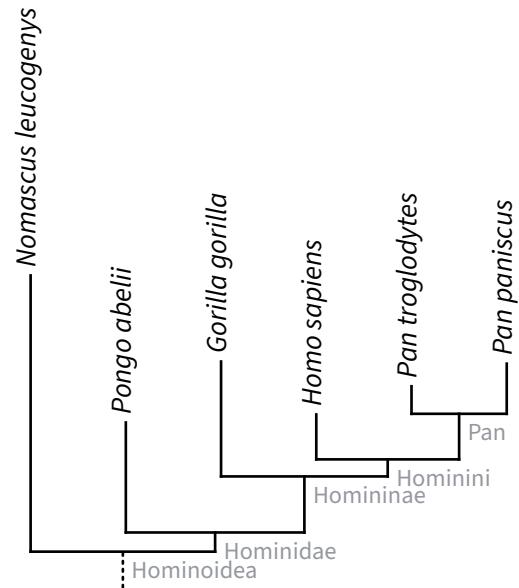
```
#let hominoidea_tree = parse-newick(read("/docs/data/hominoidea.nwk"))
```

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": "AGTCTCAAGATAACTTCGAAACAACTTC",
  "seq2": "AGTTCCAAGTGGATTGGAATTGAACCTT",
  "seq3": "ACTCT-CGGATGGATTGGATACAAACTTT",
  "seq4": "AGTCT---GATTGATGTGGATACAAACTTC",
  "seq5": "AGTCT--GGGTGGATTGG-AACAAATTT",
  "seq6": "CAGTGCTCCCTGGTGGTGG-ACCATCTTAC",
  "seq7": "AGTCTCAAGACGGATACTG--ATGCCCTAT",
)

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

```

seq1	AGTCTCAAGATAACTTCGAAACAACTTC	seq1	AGTCTCAAGATAACTTCGAAACAACTTC
seq2	AGTTTCCAAGTGGATTGGAATTGAACTTT	seq2	AGTTTCCAAGTGGATTGGAATTGAACTTT
seq3	ACTCT-CGGATGGATTCGGATACAAACTTT	seq3	ACTCT-CGGATGGATTCGGATACAAACTTT
seq4	AGTCT---GATTGATGTGGATACAAACTTC	seq4	AGTCT---GATTGATGTGGATACAAACTTC
seq5	AGTCT--GGGTGGATTGG-AACAAATTTC	seq5	AGTCT--GGGTGGATTGG-AACAAATTTC
seq6	CAGTGCTCCCTGGTGGTGG-ACCATCTTAC	seq6	CAGTGCTCCCTGGTGGTGG-ACCATCTTAC
seq7	AGTCTCAAGACGGATACTG--ATGCCCTAT	seq7	AGTCTCAAGACGGATACTG--ATGCCCTAT

Default document font for raw text

Custom font (Maple Mono)

Sequence logos, genome maps, 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: "Libertinus Serif")
  render-sequence-logo(dna_msa)
}
```

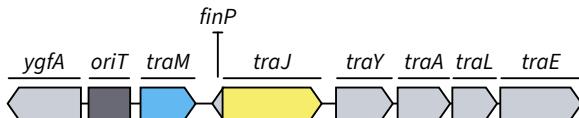


Default document font

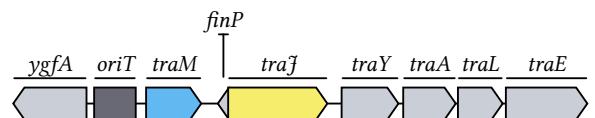


Custom font (Libertinus Serif)

```
#context {
  show text: set text(font: "Libertinus Serif")
  render-genome-map(f_plasmid_locus)
}
```

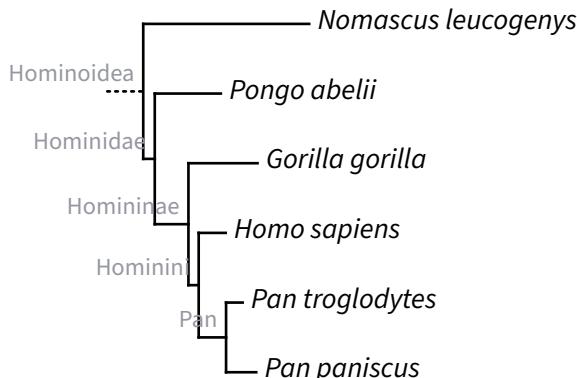


Default document font

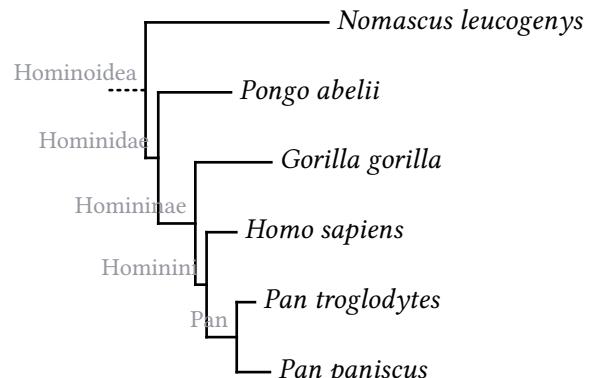


Custom font (Libertinus Serif)

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



Default document font



Custom font (Libertinus Serif)

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).