

Bioinformatics Toolbox

For Use with **MATLAB®**

- Computation
- Visualization
- Programming

User's Guide

Version 2



How to Contact The MathWorks



www.mathworks.com
comp.soft-sys.matlab
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com

Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Bioinformatics Toolbox User's Guide

© COPYRIGHT 2003–2006 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks, and SimBiology, SimEvents, and SimHydraulics are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

September 2003 Online only
June 2004 Online only
November 2004 Online only
March 2005 Online only
May 2005 Online only
September 2005 Online only
November 2005 Online only
March 2006 Online only
May 2006 Online only
September 2006 Online only

New for Version 1.0 (Release 13SP1+)
Updated for Version 1.1 (Release 14)
Updated for Version 2.0 (Release 14SP1+)
Updated for Version 2.0.1 (Release 14SP2)
Updated for Version 2.1 (Release 14SP2+)
Updated for Version 2.1.1 (Release 14SP3)
Updated for Version 2.2 (Release 14SP3+)
Updated for Version 2.2.1 (Release 2006a)
Updated for Version 2.3 (Release 2006a+)
Updated for Version 2.4 (Release 2006b)

Getting Started

1

What Is the Bioinformatics Toolbox?	1-2
Expected User	1-3
Installation	1-5
Required Software	1-5
Additional Software	1-5
Features and Functions	1-7
Data Formats and Databases	1-8
Sequence Alignments	1-9
Sequence Utilities and Statistics	1-10
Protein Property Analysis	1-11
Phylogenetic Analysis	1-12
Microarray Data Analysis	1-12
Mass Spectrometry Data Analysis	1-13
Graph Theory Functions	1-14
Graph Visualization Methods	1-15
Statistical Learning and Visualization	1-15
Prototype and Development Environment	1-16
Data Visualization	1-16
Algorithm Sharing and Application Deployment	1-17

Sequence Analysis

2

Example: Sequence Statistics	2-2
Determining Nucleotide Content	2-2
Getting Sequence Information into MATLAB	2-4
Determining Nucleotide Composition	2-5
Determining Codon Composition	2-9
Open Reading Frames	2-12

Amino Acid Conversion and Composition	2-15
Example: Sequence Alignment	2-18
Finding a Model Organism to Study	2-18
Getting Sequence Information from a Public Database ...	2-20
Searching a Public Database for Related Genes	2-23
Locating Protein Coding Sequences	2-25
Comparing Amino Acid Sequences	2-28
Sequence Tool	2-37
Importing a Sequence	2-37
Viewing Nucleotide Sequence Information	2-39
Searching for Words	2-41
Exploring Open Reading Frames	2-42
Viewing Amino Acid Sequence Statistics	2-45
Multiple Sequence Alignment Viewer	2-48
Loading Sequence Data and Viewing the Phylogenetic Tree	2-48
Selecting a Subset of Data from the Phylogenetic Tree ...	2-49
Aligning Multiple Sequences	2-50
Adjusting Multiple Alignments Manually	2-51

Microarray Analysis

3

Example: Visualizing Microarray Data	3-2
Overview of the Mouse Example	3-2
Exploring the Microarray Data Set	3-3
Spatial Images of Microarray Data	3-5
Statistics of the Microarrays	3-15
Scatter Plots of Microarray Data	3-16
Example: Analyzing Gene Expression Profiles	3-25
Overview of the Yeast Example	3-25
Exploring the Data Set	3-25
Filtering Genes	3-29
Clustering Genes	3-32
Principal Component Analysis	3-36

4

Example: Building a Phylogenetic Tree	4-2
Overview for the Primate Example	4-2
Searching NCBI for Phylogenetic Data	4-4
Creating a Phylogenetic Tree for Five Species	4-6
Creating a Phylogenetic Tree for Twelve Species	4-8
Exploring the Phylogenetic Tree	4-10
Phylogenetic Tree Tool Reference	4-14
Opening the Phylogenetic Tree Tool	4-14
File Menu	4-16
Tools Menu	4-25
Windows Menu	4-34
Help Menu	4-34

Examples

A

Sequence Analysis	A-2
Microarray Analysis	A-2
Phylogenetic Analysis	A-2

Index

Getting Started

This chapter is an overview of the functions and features in the Bioinformatics Toolbox. An introduction to these features will help you to develop a conceptual model for working with the toolbox and your biological data.

What Is the Bioinformatics Toolbox? (p. 1-2)	Description of this toolbox and the intended user
Installation (p. 1-5)	Required software and additional software for developing advanced algorithms
Features and Functions (p. 1-7)	Functions grouped into categories that support bioinformatic tasks

What Is the Bioinformatics Toolbox?

The Bioinformatics Toolbox extends MATLAB® to provide an integrated software environment for genome and proteome analysis. Scientists and engineers can answer questions, solve problems, prototype new algorithms, and build applications for drug discovery and design, genetic engineering, and biological research.

You can use the basic bioinformatic functions provided with this toolbox to create more complex algorithms and applications. These robust and well tested functions are the functions that you would otherwise have to create yourself.

- **Data formats and databases** — Connect to Web accessible databases with genomic and proteomic data. Read and convert between multiple data formats.
- **Sequence analysis** — Determine the statistical characteristics of a sequence, align two sequences, and multiply align several sequences. Model patterns in biological sequences using Hidden Markov Model (HMM) profiles.
- **Phylogenetic analysis** — Create and manipulate phylogenetic tree data.
- **Microarray data analysis** — Read, normalize, and visualize microarray data.
- **Mass spectrometry data analysis** — Analyze and enhance raw mass spectrometry data.
- **Statistical Learning** — Classify and identify features in data sets with statistical learning tools.
- **Programming interface** — Use other bioinformatic software (Bioperl and BioJava) within the MATLAB environment.

The field of bioinformatics is rapidly growing and will become increasingly important as biology becomes a more analytical science. The Bioinformatics Toolbox provides an open environment that you can customize for development and deployment of the analytical tools you will need.

Prototype and develop algorithms — Prototype new ideas in an open and extendable environment. Develop algorithms using efficient string processing

and statistical functions, view the source code for existing functions, and use the code as a template for customizing, improving, or creating your own functions. See “Prototype and Development Environment” on page 1-16.

Visualize data — Visualize sequences and alignments, gene expression data, phylogenetic trees, mass spectrometry data, protein structure, and relationships between data with interconnected graphs. See “Data Visualization” on page 1-16.

Share and deploy applications — Use an interactive GUI builder to develop a custom graphical front end for your data analysis programs. Create stand-alone applications that run separately from MATLAB. See “Algorithm Sharing and Application Deployment” on page 1-17.

Expected User

The Bioinformatics Toolbox is for computational biologists and research scientists who need to develop new algorithms or implement published ones, visualize results, and create stand-alone applications.

- **Industry/Professional** — Increasingly, drug discovery methods are being supported by engineering practice. This toolbox supports tool builders who want to create applications for the biotechnology and pharmaceutical industries.
- **Education/Professor/Student** — This toolbox is well suited for learning and teaching genome and proteome analysis techniques. Educators and students can concentrate on bioinformatic algorithms instead of programming basic functions such as reading and writing to files.

While the toolbox includes many bioinformatics functions, it is not intended to be a complete set of tools for scientists to analyze their biological data. However, MATLAB is the ideal environment for you to rapidly design and prototype the tools you need.

Installation

You don't need to do anything special when installing the Bioinformatics Toolbox. Install the toolbox from a CD or Web release using The MathWorks installer.

- “Required Software” on page 1-5 — List of MathWorks products you need to purchase with the Bioinformatics Toolbox
- “Additional Software” on page 1-5 — List of toolboxes from The MathWorks for advanced algorithm development

Required Software

The Bioinformatics Toolbox requires the following products from The MathWorks to be installed on your computer:

MATLAB

Provides a command-line interface and integrated software environment for the Bioinformatics Toolbox.

Version 2.4 of the Bioinformatics Toolbox requires MATLAB Version 7.3 on the Release 2006b CD.

Statistics Toolbox

Provides basic statistics and probability functions that the functions in the Bioinformatics Toolbox use.

Version 2.4 of the Bioinformatics Toolbox requires the Statistics Toolbox Version 5.3 on the Release 2006b CD.

Additional Software

MATLAB and the Bioinformatics Toolbox provide an open and extensible software environment. In this environment you can interactively explore ideas, prototype new algorithms, and develop complete solutions to problems in bioinformatics. The MATLAB language facilitates computation, visualization, prototyping, and deployment.

Using the Bioinformatics Toolbox in combination with other MATLAB toolboxes and products will allow you to solve multidisciplinary problems.

Distributed Computing Toolbox

Execute bioinformatic algorithms onto a cluster of computers. For an example of batch processing through distributed computing, see the `biodistcompdemo`.

Signal Processing Toolbox

Process signal data from bioanalytical instrumentation. Examples include acquisition of fluorescence data for DNA sequence analyzers, fluorescence data for microarray scanners, and mass spectrometric data from protein analyses.

Image Processing Toolbox

Create complex and custom image processing algorithms for data from microarray scanners.

Optimization Toolbox

Use nonlinear optimization for predicting the secondary structure of proteins and the structure of other biological macromolecules.

Neural Network Toolbox

Use neural networks to solve problems where algorithms are not available. For example, you can train neural networks for pattern recognition using large sets of sequence data.

Database Toolbox

Create your own in-house databases for sequence data with custom annotations.

MATLAB Compiler

Create stand-alone applications from MATLAB GUI applications, and create dynamic link libraries from MATLAB functions for use with any programming environment.

MATLAB® Builder for COM

Create COM objects to use with any COM-based programming environment.

MATLAB® Builder for Excel

Create Excel add-in functions from MATLAB functions to use with Excel spreadsheets.

Excel Link

Connect Microsoft Excel with the MATLAB workspace to exchange data and to use the computational and visualization functions in MATLAB.

Features and Functions

The Bioinformatics Toolbox includes many functions to help you with genome and proteome analysis. Most functions are implemented in M-code (the MATLAB programming language) with the source available for you to view. This open environment lets you explore and customize the existing toolbox algorithms or develop your own.

Data Formats and Databases (p. 1-8)	Access online databases, copy data into the MATLAB workspace, and read and write to files with standard bioinformatic formats.
Sequence Alignments (p. 1-9)	Compare nucleotide or amino acid sequences using pairwise and multiple sequence alignment functions.
Sequence Utilities and Statistics (p. 1-10)	Manipulate sequences and determine physical, chemical, and biological characteristics.
Protein Property Analysis (p. 1-11)	Determine protein characteristics and simulate enzyme cleavage reactions.
Phylogenetic Analysis (p. 1-12)	Explore phylogenetic data with functions and a GUI to draw phylograms (trees)
Microarray Data Analysis (p. 1-12)	Read, filter, normalize, and visualize microarray data.
Mass Spectrometry Data Analysis (p. 1-13)	Preprocess raw mass spectrometry data and use statistical learning functions to identify patterns.
Graph Theory Functions (p. 1-14)	Apply basic graph theory algorithms to sparse matrices.
Graph Visualization Methods (p. 1-15)	View relationships between data visually with interactive maps, hierarchy plots, and pathways.

Statistical Learning and
Visualization (p. 1-15)

Classify and identify features in data sets, set up cross-validation experiments, and compare different classification methods.

Prototype and Development
Environment (p. 1-16)

Create new algorithms, try new ideas, and analyze alternatives.

Data Visualization (p. 1-16)

Visually compare pairwise sequence alignments, multiply aligned sequences, gene expression data from microarrays, and plot nucleic acid and protein characteristics.

Algorithm Sharing and Application
Deployment (p. 1-17)

Create GUIs and stand-alone applications.

Data Formats and Databases

The Bioinformatics Toolbox supports access to many of the databases on the Web and other online data sources. It also reads many common genome file formats, so that you do not have to write and maintain your own file readers.

Web-based databases — You can directly access public databases on the Web and copy sequence and gene expression information into MATLAB.

The sequence databases currently supported are GenBank (`getgenbank`), GenPept (`getgenpept`), European Molecular Biology Laboratory EMBL (`getembl`), and Protein Data Bank PDB (`getpdb`). You can also access data from the NCBI Gene Expression Omnibus (GEO) web site by using a single function (`getgeodata`).

Get multiply aligned sequences (`gethmmalignment`), hidden Markov model profiles (`gethmmprof`), and phylogenetic tree data (`gethmmtree`) from the PFAM database.

Gene Ontology database — Load the database from the Web into a gene ontology object (`geneont`). Select sections of the ontology with methods for the `geneont` object (`getancestors`, `getdescendants`, `getmatrix`, `getrelatives`), and manipulate data with utility functions (`goannotread`, `num2goid`)

Read data from instruments — Read data generated from gene sequencing instruments (`scfread`, `joinseq`, `traceplot`), mass spectrometers (`jcampread`), and Agilent microarray scanners (`agferead`).

Reading data formats — The toolbox provides a number of functions for reading data from common bioinformatic file formats.

- Sequence data: GenBank (`genbankread`), GenPept (`genpeptread`), EMBL (`emblread`), PDB (`pdbread`), and FASTA (`fastaread`)
- Multiply aligned sequences: ClustalW and GCG formats (`multialignread`)
- Gene expression data from microarrays: Gene Expression Omnibus (GEO) data (`geosoftread`), GenePix data in GPR and GAL files (`gprread`, `galread`), SPOT data (`sptread`), Affymetrix data (`affyread`), and ImageGene results files (`imageneread`).

Note: The function `affyread` only works on PC supported platforms.

- Hidden Markov model profiles: PFAM-HMM file (`pfamhmmread`)

Writing data formats — The functions for getting data from the Web include the option to save the data to a file. However, there is a function to write data to a file using the FASTA format (`fastawrite`).

BLAST searches — Request Web-based BLAST searches (`blastncbi`), get the results from a search (`getblast`) and read results from a previously saved BLAST formatted report file (`blastread`).

MATLAB has built-in support for other industry-standard file formats including Microsoft Excel and comma-separated value (CSV) files. Additional functions perform ASCII and low-level binary I/O, allowing you to develop custom functions for working with any data format.

Sequence Alignments

You can select from a list of analysis methods to perform pairwise or multiple sequence alignment.

Pairwise sequence alignment — Efficient MATLAB implementations of standard algorithms such as the Needleman-Wunsch (`nwalign`) and Smith-Waterman (`swalign`) algorithms for pairwise sequence alignment.

The toolbox also includes standard scoring matrices such as the PAM and BLOSUM families of matrices (`blosum`, `dayhoff`, `gonnet`, `nuc44`, `pam`). Visualize sequence similarities with `seqdotplot` and sequence alignment results with `showalignment`.

Multiple sequence alignment — Functions for multiple sequence alignment (`multialign`, `proalign`) and functions that support multiple sequences (`multialignread`, `fastaread`, `showalignment`). There is also a graphical interface (`multialignviewer`) for viewing the results of a multiple sequence alignment and manually making adjustment.

Multiple sequence profiles — MATLAB implementations for multiple alignment and profile hidden Markov model algorithms (`gethmmprof`, `gethmmalignment`, `gethmmtree`, `pfamhmmread`, `hmmprofalign`, `hmmprofestimate`, `hmmprofgenerate`, `hmmprofmerge`, `hmmprofstruct`, `showhmmprof`).

Biological codes — Look up the letters or numeric equivalents for commonly used biological codes (`aminolookup`, `baselookup`, `geneticcode`, `revgeneticcode`).

Sequence Utilities and Statistics

You can manipulate and analyze your sequence to gain a deeper understanding of your data. Use a graphical user interface (GUI) with many of the sequence functions in the Bioinformatics Toolbox (`seqtool`).

Sequence conversion and manipulation — The toolbox provides routines for common operations, such as converting DNA or RNA sequences to amino acid sequences, that are basic to working with nucleic acid and protein sequences (`aa2int`, `aa2nt`, `dna2rna`, `rna2dna`, `int2aa`, `int2nt`, `nt2aa`, `nt2int`, `seqcomplement`, `seqrcomplement`, `seqreverse`).

You can manipulate your sequence by performing an in-silico digestion with restriction endonucleases (`restrict`) and proteases (`cleave`).

Sequence statistics — Determine various statistics about a sequence (`aacount`, `basecount`, `codoncount`, `dimercount`, `nmercount`, `ntdensity`, `codonbias`, `cpgisland`, `oligoprop`), search for specific patterns within a sequence (`seqshowwords`, `seqwordcount`), or search for open reading frames

(seqshoworfs). In addition, you can create random sequences for test cases (randseq).

Sequence utilities — Determine a consensus sequence from a set of multiply aligned amino acid, nucleotide sequences (seqconsensus, or a sequence profile (seqprofile). Format a sequence for display (seqdisp) or graphically show a sequence alignment with frequency data (seqlogo).

Additional functions in MATLAB efficiently handle string operations with regular expressions (regexp, seq2regexp) to look for specific patterns in a sequence and search through a library for string matches (seqmatch).

Look for possible cleavage sites in a DNA/RNA sequence by searching for palindromes (palindromes).

Protein Property Analysis

You can use a collection of protein analysis methods to extract information from your data. The toolbox provides functions to calculate various properties of a protein sequence, such as the atomic composition (atomiccomp), molecular weight (molweight), and isoelectric point (isoelectric). You can cleave a protein with an enzyme (cleave, rebasecuts) and create distance and Ramachandran plots for PDB data (pdbdistplot, ramachandran). The toolbox contains a graphical user interface for protein analysis (proteinplot) and plotting 3-D protein structures with information from the PDB database (pdbplot).

Amino acid sequence utilities — Calculate amino acid statistics for a sequence (aacount) and get information about character codes (aminolookup).

Phylogenetic Analysis

Functions for phylogenetic tree building and analysis.

Phylogenetic tree data — Read and write Newick formatted tree files (`phytreeread`, `phytreewrite`) into the MATLAB workspace as phylogenetic tree objects (`phytree`).

Create a phylogenetic tree — Calculate the pairwise distance between biological sequences (`seqpdist`), estimate the substitution rates (`dnds`, `dndsml`), build a phylogenetic tree from pairwise distances (`seqlinkage`, `seqneighjoin`, `reroot`), and view the tree in an interactive GUI that allows you to view, edit, and explore the data (`phytreetool` or `view`). This GUI also allows you to prune branches, reorder, rename, and explore distances.

Phylogenetic tree object methods — You can access the functionality of the `phytreetool` GUI using methods for a phylogenetic tree object (`phytree`). Get property values (`get`) and node names (`getbyname`). Calculate the patristic distances between pairs of leaf nodes (`pdist`, `weights`) and draw a phylogenetic tree object in a MATLAB figure window as a phylogram, cladogram, or radial treeplot (`plot`). Manipulate tree data by selecting branches and leaves using a specified criterion (`select`, `subtree`) and removing nodes (`prune`). Compare trees (`getcanonical`) and use Newick formatted strings (`getnewickstr`).

Microarray Data Analysis

MATLAB is widely used for microarray data analysis. However, the standard normalization and visualization tools that scientists use can be difficult to implement. The Bioinformatics Toolbox includes these standard functions.

Microarray data — Read Affymetrix GeneChip files (`affyread`) and plot data (`probesetplot`), ImaGene results files (`imageneread`), SPOT files (`sptread`) and Agilent microarray scanner files (`agferead`). Read GenePix GPR files (`gprread`) and GAL files (`galread`). Get Gene Expression Omnibus (GEO) data from the web (`getgeodata`) and read GEO data from files (`geosoftread`).

A utility function (`magetfield`) extracts data from one of the microarray reader functions (`gprread`, `agferead`, `sptread`, `imageneread`).

Microarray normalization and filtering — The toolbox provides a number of methods for normalizing microarray data, such as lowest normalization (`malowess`) and mean normalization (`manorm`), or across multiple arrays (`quantilenorm`). You can use filtering functions to clean raw data before analysis (`geneentropyfilter`, `genelowvalfilter`, `generangefilter`, `genevarfilter`), and calculate the range and variance of values (`exprprofrange`, `exprprofvar`).

Microarray visualization — The toolbox contains routines for visualizing microarray data. These routines include spatial plots of microarray data (`maimage`, `redgreenmap`), box plots (`maboxplot`), loglog plots (`maloglog`), and intensity-ratio plots (`mairplot`). You can also view clustered expression profiles (`clustergram`, `redgreenmap`). You can create 2-D scatter plots of principal components from the microarray data (`mapcaplot`).

Microarray utility functions — Use the following functions to work with Affymetrix and GeneChip data sets. Get library information for a probe (`probelibraryinfo`), gene information from a probe set (`probesetlookup`), and probe set values from CEL and CDF information (`probesetvalues`). Show probe set information from NetAffx (`probesetlink`) and plot probe set values (`probesetplot`).

The toolbox accesses statistical routines to perform cluster analysis and to visualize the results, and you can view your data through statistical visualizations such as dendrograms, classification, and regression trees.

Mass Spectrometry Data Analysis

The mass spectrometry functions are designed for preprocessing and classification of raw data from SELDI-TOF and MALDI-TOF spectrometers.

Reading raw data into MATLAB — Load raw mass/charge and ion intensity data from comma-separated-value (CSV) files, or read a JCAMP-DX formatted file with mass spectrometry data (`jcampread`) into MATLAB.

You can also have data in TXT files and use the function `importdata`.

Preprocessing raw data — Resample high-resolution data to a lower resolution (`msresample`) where the extra data points are not needed. Correct the baseline (`msbackadj`). Align a spectrum to a set of reference masses

(`msalign`) and visually verify the alignment (`msheatmap`). Normalize the area between spectra for comparing (`msnorm`), and filter out noise (`mslowess`, `mssgolay`).

Spectrum analysis — Load spectra into a GUI (`msviewer`) for selecting mass peaks and further analysis.

Graph Theory Functions

Graph theory functions in the Bioinformatics Toolbox apply basic graph theory algorithms to sparse matrices. A sparse matrix represents a graph, any nonzero entries in the matrix represent the edges of the graph, and the values of these entries represent the associated weight (cost, distance, length, or capacity) of the edge. Graph algorithms that use the weight information will cancel the edge if a NaN or an Inf is found. Graph algorithms that do not use the weight information will consider the edge if a NaN or an Inf is found, because these algorithms look only at the connectivity described by the sparse matrix and not at the values stored in the sparse matrix.

Sparse matrices can represent four types of graphs:

- **Directed Graph** — Sparse matrix, either double real or logical. Row (column) index indicates the source (target) of the edge. Self-loops (values in the diagonal) are allowed, although most of the algorithms ignore these values.
- **Undirected Graph** — Lower triangle of a sparse matrix, either double real or logical. An algorithm expecting an undirected graph ignores values stored in the upper triangle of the sparse matrix and values in the diagonal.
- **Direct Acyclic Graph (DAG)** — Sparse matrix, double real or logical, with zero values in the diagonal. While a zero-valued diagonal is a requirement of a DAG, it does not guarantee a DAG. An algorithm expecting a DAG will *not* test for cycles because this will add unwanted complexity.
- **Spanning Tree** — Undirected graph with no cycles and with one connected component.

There are no attributes attached to the graphs; sparse matrices representing all four types of graphs can be passed to any graph algorithm. All functions will return an error on nonsquare sparse matrices.

Graph algorithms do not pretest for graph properties because such tests can introduce a time penalty. For example, there is an efficient shortest path algorithm for DAG, however testing if a graph is acyclic is expensive compared to the algorithm. Therefore, it is important to select a graph theory function and properties appropriate for the type of the graph represented by your input matrix. If the algorithm receives a graph type that is different from what it expects, it will either:

- Return an error when it reaches an inconsistency, for example, if you pass a cyclic graph to the `graphshortestpath` function and specify `Acyclic` as the method property.
- Produce an invalid result. For example, if you pass a directed graph to a function with an algorithm that expects an undirected graph, it will ignore values in the upper triangle of the sparse matrix.

The graph theory functions include `graphallshortestpaths`, `graphconncomp`, `graphisdag`, `graphisomorphism`, `graphissspantree`, `graphmaxflow`, `graphminspantree`, `graphpred2path`, `graphshortestpath`, `graphtopoorder`, `graphtraverse`.

Graph Visualization Methods

Graph functions in the Bioinformatics Toolbox include viewing and manipulation tools that let you display interaction maps, hierarchy plots, or even pathways.

The graph visualization functions and methods begin with creating an object to hold graph data (`biograph`). Calculate the position of nodes (`dolayout`), and draw a graph with the results (`view`). Get handle information about the nodes (`getnodesbyid`), edges (`getedgesbynodeid`), and find relations between the nodes (`getancestors`, `getdescendants`, `getrelatives`).

You can also change programmatically the properties of your rendered graph.

Statistical Learning and Visualization

The Bioinformatics Toolbox provides functions that build on the classification and statistical learning tools in the Statistics Toolbox (`classify`, `kmeans`, `treefit`).

These functions include imputation tools (`knnimpute`), support for vector machine classifiers (`svmclassify`, `svmtrain`) and K-nearest neighbor classifiers (`knnclassify`).

Other functions for set up cross-validation experiments (`crossvalind`) and comparing the performance of different classification methods (`classperf`). In addition, there are tools for selecting diversity and discriminating features (`rankfeatures`, `randfeatures`).

Prototype and Development Environment

MATLAB is a prototyping and development environment where you can create algorithms and easily compare alternatives.

- **Integrated environment** — Explore biological data in an environment that integrates programming and visualization. Create reports and plots with the built-in functions for mathematics, graphics, and statistics.
- **Open environment** — Access the source code for the Bioinformatics Toolbox functions. The toolbox includes many of the basic bioinformatics functions you will need to use, and it includes prototypes for some of the more advanced functions. Modify these functions to create your own custom solutions.
- **Interactive programming language** — Test your ideas by typing functions that are interpreted interactively with a language whose basic data element is an array. The arrays do not require dimensioning and allow you to solve many technical computing problems,

Using matrices for sequences or groups of sequences allows you to work efficiently and not worry about writing loops or other programming controls.

- **Programming tools** — Use a visual debugger for algorithm development and refinement and an algorithm performance profiler to accelerate development.

Data Visualization

In addition, MATLAB 2-D and volume visualization features let you create custom graphical representations of multidimensional data sets. You can also create montages and overlays, and export finished graphics to a PostScript image file or copy directly into Microsoft PowerPoint.

Algorithm Sharing and Application Deployment

The open MATLAB environment lets you share your analysis solutions with other MATLAB users, and it includes tools to create custom software applications. With the addition of the MATLAB Compiler, you can create stand-alone applications independent of MATLAB, and with the addition of MATLAB Builder for COM, you can create GUIs and stand-alone applications within other programming environments.

- **Share algorithms with other MATLAB users** — You can share data analysis algorithms created in the MATLAB language across all MATLAB supported platforms by giving M-files to other MATLAB users. You can also create GUIs within MATLAB using the Graphical User Interface Development Environment (GUIDE).
- **Deploy MATLAB GUIs** — Create a GUI within MATLAB using GUIDE, and then use the MATLAB Compiler to create a stand-alone GUI application that runs separately from MATLAB.
- **Create dynamic link libraries (DLL)** — Use the MATLAB compiler to create dynamic link libraries (DLLs) for your functions, and then link these libraries to other programming environments such as C and C++.
- **Create COM objects** — Use MATLAB Builder for COM to create COM objects, and then use a COM compatible programming environment (Visual Basic) to create a stand-alone application.
- **Create Excel add-ins** — Use MATLAB Builder for Excel to create Excel add-in functions, and then use the add-in functions with Excel spreadsheets.

Sequence Analysis

Sequence analysis is the process you use to find information about a nucleotide or amino acid sequence using computational methods. Common tasks in sequence analysis are identifying genes, determining the similarity of two genes, determining the protein coded by a gene, and determining the function of a gene by finding a similar gene in another organism with a known function.

Example: Sequence Statistics (p. 2-2)

Starting with a DNA sequence, calculate statistics for the nucleotide content.

Example: Sequence Alignment (p. 2-18)

Starting with a DNA sequence for a human gene, locate and verify a corresponding gene in a model organism.

Sequence Tool (p. 2-37)

Use a graphical interface for the sequence functions.

Multiple Sequence Alignment Viewer (p. 2-48)

Use a graphical interface to visually inspect a multiple alignment and make manual adjustments.

Example: Sequence Statistics

After sequencing a piece of DNA, one of the first tasks is to investigate the nucleotide content in the sequence. Starting with a DNA sequence, this example uses sequence statistics functions to determine mono-, di-, and trinucleotide content, and to locate open reading frames.

Determining Nucleotide Content (p. 2-2)	Use the MATLAB Help browser to search the Web for information.
Getting Sequence Information into MATLAB (p. 2-4)	Find a nucleotide sequence in a public database and read the sequence information into MATLAB.
Determining Nucleotide Composition (p. 2-5)	Determine the monomers and dimers, and then visualize data in graphs and bar plots.
Determining Codon Composition (p. 2-9)	Look at codons for the six reading frames.
Open Reading Frames (p. 2-12)	Locate the open reading frames using a specific genetic code.
Amino Acid Conversion and Composition (p. 2-15)	Extract the protein-coding sequence from a gene sequence and convert it to the amino acid sequence for the protein.

Determining Nucleotide Content

In this example you are interested in studying the human mitochondrial genome. While many genes that code for mitochondrial proteins are found in the cell nucleus, the mitochondrial has genes that code for proteins used to produce energy.

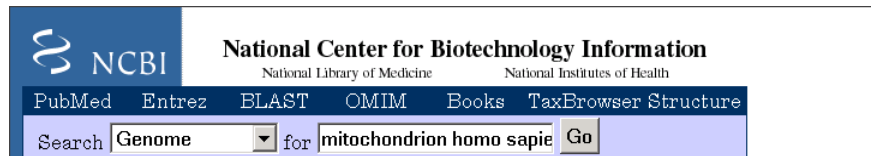
First research information about the human mitochondria and find the nucleotide sequence for the genome. Next, look at the nucleotide content for the entire sequence. And finally, determine open reading frames and extract specific gene sequences.

- 1 Use the MATLAB Help browser to explore the Web. In the MATLAB Command Window, type

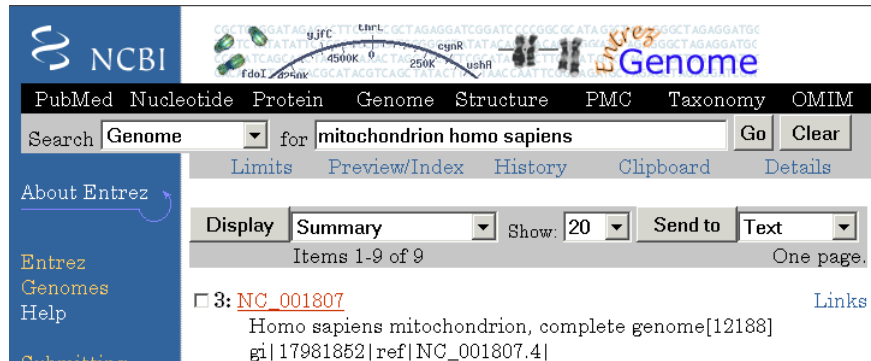
```
web('http://www.ncbi.nlm.nih.gov/')
```

A separate browser window opens with the home page for the NCBI Web site.

- 2 Search the NCBI Web site for information. For example, to search for the human mitochondrion genome, from the **Search** list, select Genome, and in the **for** box, enter mitochondrion homo sapiens.



The NCBI Web search returns a list of links to relevant pages.



- 3 Select a result page. For example, click the link labeled **NC_001807**.

The MATLAB Help browser displays the NCBI page for the human mitochondrial genome.

NCBI

PubMed Nucleotide Protein Genome Structure PopSet Taxonomy

Search **Genome** for

Homo sapiens mitochondrion, complete genome

Accession: [NC_001807](#)
 Total Bases Sequenced: 16571 bp
 Completed: Mar 11, 2001.

Views:
 Protein View
 Coding Regions
 RNA Genes

Organelles

Mitochondrion
 Organism: [Homo sapiens](#)
 Genetic Code: [2](#)
 Lineage: Eukaryota; Metazoa; Chordata; Craniata; Vertebrata;

Legend:

- CDS +strand
- CDS -strand
- RNA +strand
- RNA -strand

Getting Sequence Information into MATLAB

Many public databases for nucleotide sequences are accessible from the Web. The MATLAB Command Window provides an integrated environment for bringing sequence information into MATLAB.

The consensus sequence for the human mitochondrial genome has the GenBank accession number NC_001807. Since the whole GenBank entry is quite large and you might only be interested in the sequence, you can get just the sequence information.

- 1 Get sequence information from a Web database. For example, to get sequence information for the human mitochondrial genome, in the MATLAB Command Window, type

```
mitochondria = getgenbank('NC_001807','SequenceOnly',true);
```

MATLAB gets the nucleotide sequence from the GenBank database and creates a character array.

```
mitochondria =
gatcacaggctctatcacctattaaccactcacgggagctctccatgcat
ttggtatntttcgtctggggggtgtgcacgcgatagcattgagagcgctg
gagccggagcaccctatgtcgcagtatctgtctttgattcctgcctcatt
ctattatntatcgcacctacgttcaatattacaggcgaacatacctacta
aagt . . .
```

- 2 If you don't have a Web connection, you can load the data from a MAT-file included with the Bioinformatics Toolbox, using the command

```
load mitochondria
```

MATLAB loads the sequence mitochondria into the MATLAB workspace.

- 3 Get information about the sequence. Type

```
whos mitochondria
```

MATLAB displays information about the size of the sequence.

Name	Size	Bytes	Class
mitochondria	1x16571	33142	char array

Grand total is 16571 elements using 33142 bytes

Determining Nucleotide Composition

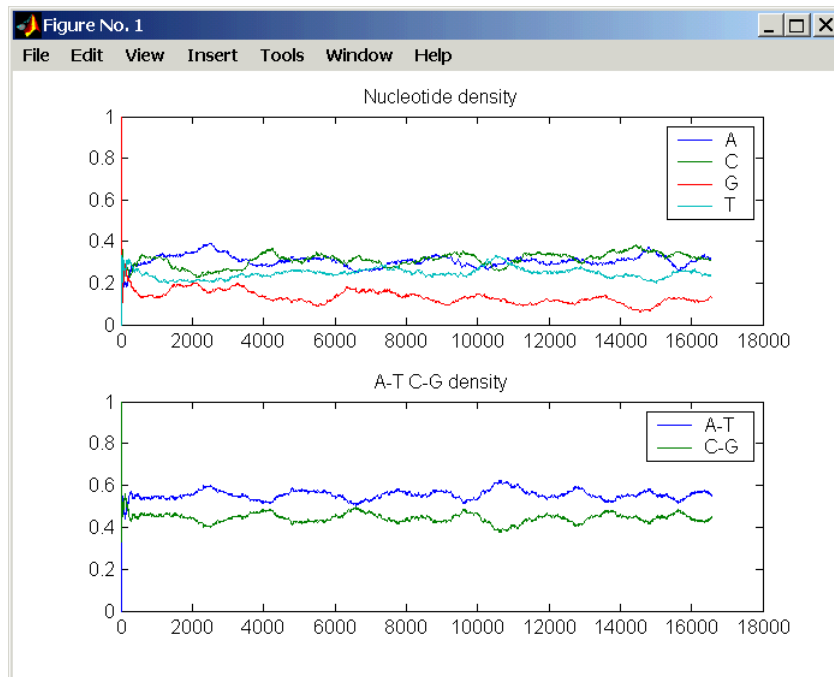
Sections of a DNA sequence with a high percent of A+T nucleotides usually indicate intergenic parts of the sequence, while low A+T and higher G+C nucleotide percentages indicate possible genes. Many times high CG dinucleotide content is located before a gene.

After you read a sequence into MATLAB, you can use the sequence statistics functions to determine if your sequence has the characteristics of a protein-coding region. This procedure uses the human mitochondrial genome as an example. See “Getting Sequence Information into MATLAB” on page 2-4.

- 1 Plot monomer densities and combined monomer densities in a graph. In the MATLAB Command Window, type

```
ntdensity(mitochondria)
```

This graph shows that the genome is A+T rich.



- 2 Count the nucleotides using the function `basecount(basecount(mitochondria))`

A list of nucleotide counts is shown for the 5'-3' strand.ans =

A: 5113

C: 5192

G: 2180
T: 4086

- 3** Count the nucleotides in the reverse complement of a sequence using the function `seqrcomplement`.

```
basecount(seqrcomplement(mitochondria))
```

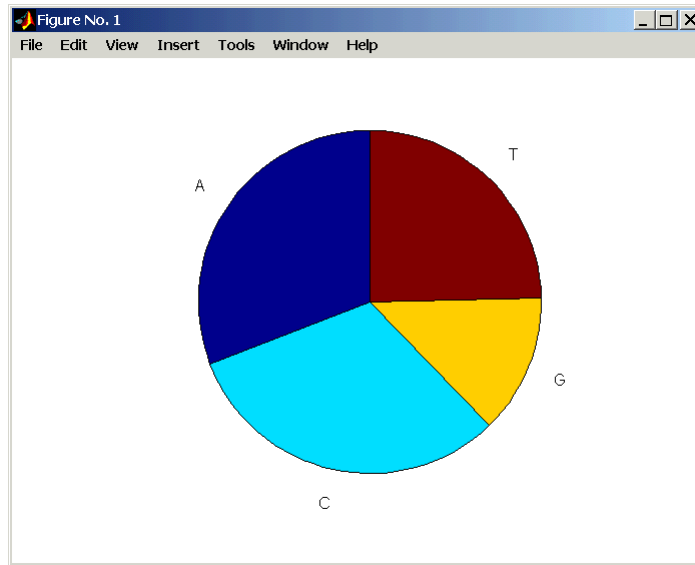
As expected, the nucleotide counts on the reverse complement strand are complementary to the 5'-3' strand.

```
ans =  
A: 4086  
C: 2180  
G: 5192  
T: 5113
```

- 4** Use the function `basecount` with the `chart` option to visualize the nucleotide distribution.

```
basecount(mitochondria,'chart','pie');
```

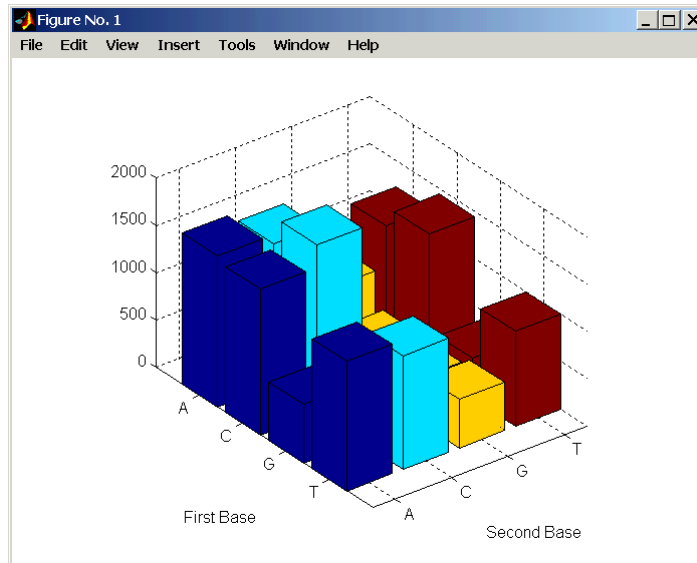
MATLAB draws a pie chart in a figure window.



5 Count the dimers in a sequence and display the information in a bar chart.

```
dimercount(mitochondria, 'chart', 'bar')
```

MATLAB lists the dimer counts and draws a bar chart.



Determining Codon Composition

Trinucleotides (codon) code for an amino acid, and there are 64 possible codons in a nucleotide sequence. Knowing the percent of codons in your sequence can be helpful when you are comparing with tables for expected codon usage.

After you read a sequence into MATLAB, you can analyze the sequence for codon composition. This procedure uses the human mitochondria genome as an example. See “Getting Sequence Information into MATLAB” on page 2-4.

- 1 Count codons in a nucleotide sequence. In the MATLAB Command Window, type

```
codoncount(mitochondria)
```

MATLAB displays the codon counts for the first reading frame.

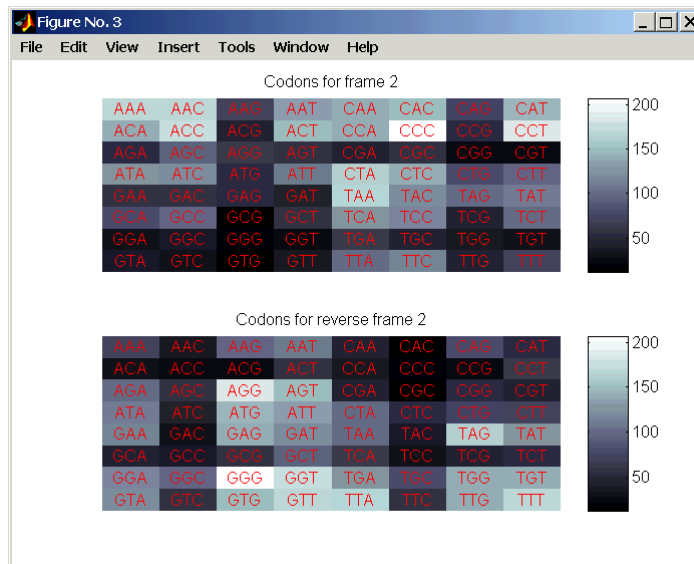
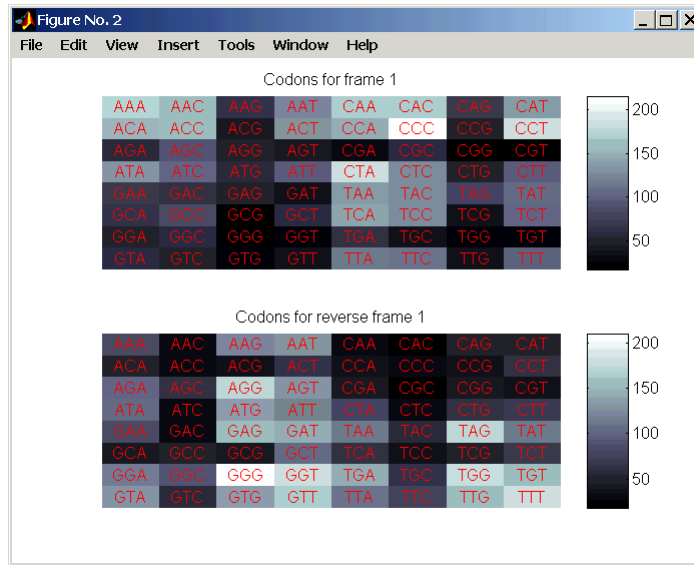
AAA - 172	AAC - 157	AAG - 67	AAT - 123
ACA - 153	ACC - 163	ACG - 42	ACT - 130
AGA - 58	AGC - 90	AGG - 50	AGT - 43
ATA - 132	ATC - 103	ATG - 57	ATT - 96
CAA - 166	CAC - 167	CAG - 68	CAT - 135

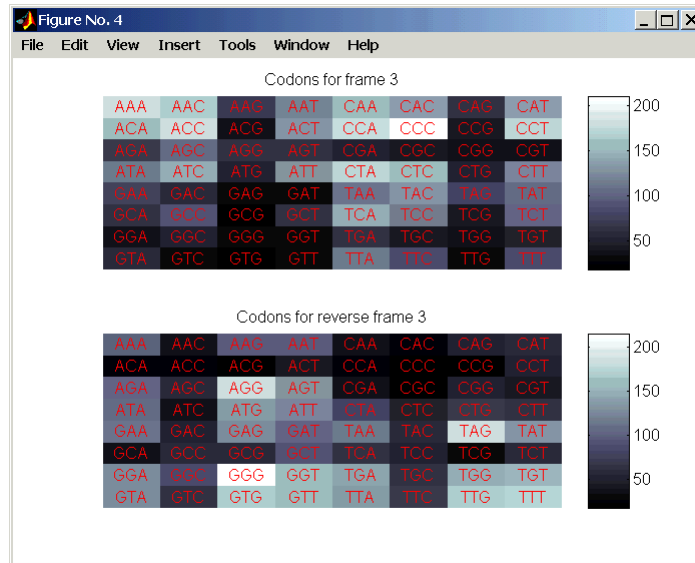
```
CCA-146  CCC-215  CCG-50  CCT-182
CGA-33   CGC-60   CGG-18  CGT-20
CTA-187  CTC-126  CTG-52  CTT-98
GAA-68   GAC-62   GAG-47  GAT-39
GCA-67   GCC-87   GCG-23  GCT-61
GGA-53   GGC-61   GGG-23  GGT-25
GTA-61   GTC-49   GTG-26  GTT-36
TAA-136  TAC-127  TAG-82  TAT-107
TCA-143  TCC-126  TCG-37  TCT-103
TGA-64   TGC-35   TGG-27  TGT-25
TTA-115  TTC-113  TTG-37  TTT-99
```

- 2** Count the codons in all six reading frames and plot the results in a heat map.

```
for frame = 1:3
    figure('color',[1 1 1])
    subplot(2,1,1);
    codoncount(mitochondria,'frame',frame,'figure',true);
    title(sprintf('Codons for frame %d',frame));
    subplot(2,1,2);
    codoncount(mitochondria,'reverse',true,...
                'frame',frame,...
                'figure',true);
    title(sprintf('Codons for reverse frame %d',frame));
end
```

MATLAB draws heat maps to visualize all 64 codons in the six reading frames.





Open Reading Frames

Determining the protein-coding sequence for a eukaryotic gene can be a difficult task because introns (noncoding sections) are mixed with exons. However, prokaryotic genes generally do not have introns and mRNA sequences have the introns removed. Identifying the start and stop codons for translation determines the protein-coding section, or open reading frame (ORF), in a sequence. Once you know the ORF for a gene or mRNA, you can translate a nucleotide sequence to its corresponding amino acid sequence.

After you read a sequence into MATLAB, you can analyze the sequence for open reading frames. This procedure uses the human mitochondria genome as an example. See “Getting Sequence Information into MATLAB” on page 2-4.

- 1 Display open reading frames (ORFs) in a nucleotide sequence. In the MATLAB Command Window, type

```
seqshoworfs(mitochondria);
```

If you compare this output to the genes shown on the NCBI page for NC_001807, there are fewer genes than expected. This is because vertebrate

mitochondria use a genetic code slightly different from the standard genetic code. For a table of genetic codes, see “Genetic Code”.

2 Display ORFs using the Vertebrate Mitochondrial code.

```
orfs= seqshoworfs(mitochondria,...
                  'GeneticCode','Vertebrate Mitochondrial',...
                  'alternativestart',true);
```

Notice that there are now two large ORFs on the first reading frame. One starts at position 4471 and the other starts at 5905. These correspond to the genes ND2 (NADH dehydrogenase subunit 2 [Homo sapiens]) and COX1 (cytochrome c oxidase subunit I) genes.

3 Find the corresponding stop codon. The start and stop positions for ORFs have the same indices as the start positions in the fields Start and Stop.

```
ND2Start = 4471;
StartIndex = find(orfs(1).Start == ND2Start)
ND2Stop = orfs(1).Stop(StartIndex)
```

MATLAB displays the stop position.

```
ND2Stop =
    5512
```

4 Using the sequence indices for the start and stop of the gene, extract the subsequence from the sequence.

```
ND2Seq = mitochondria(ND2Start:ND2Stop);
codoncount (ND2Seq)
```

The subsequence (protein-coding region) is stored in ND2Seq and displayed on the screen.

```
attaatcccctggcccaaccggtcatctactctaccatctttgcaggcac
actcatcacagcgctaagctcgactgattttttacctgagtaggcctag
aataaacatgctagcttttattccagttctaaccataaaataaacctt
cgttccacagaagctgccatcaagtatttcctcagcaagcaaccgcatc
cataatccttc . . .
```

5 Determine the codon distribution.

```
codoncount (ND2Seq)
```

The codon count shows a high amount of ACC, ATA, CTA, and ATC.

AAA - 10	AAC - 14	AAG - 2	AAT - 6
ACA - 11	ACC - 24	ACG - 3	ACT - 5
AGA - 0	AGC - 4	AGG - 0	AGT - 1
ATA - 22	ATC - 24	ATG - 2	ATT - 8
CAA - 8	CAC - 3	CAG - 2	CAT - 1
CCA - 4	CCC - 12	CCG - 2	CCT - 5
CGA - 0	CGC - 3	CGG - 0	CGT - 1
CTA - 26	CTC - 18	CTG - 4	CTT - 7
GAA - 5	GAC - 0	GAG - 1	GAT - 0
GCA - 8	GCC - 7	GCG - 1	GCT - 4
GGA - 5	GGC - 7	GGG - 0	GGT - 1
GTA - 3	GTC - 2	GTG - 0	GTT - 3
TAA - 0	TAC - 8	TAG - 0	TAT - 2
TCA - 7	TCC - 11	TCG - 1	TCT - 4
TGA - 10	TGC - 0	TGG - 1	TGT - 0
TTA - 8	TTC - 7	TTG - 1	TTT - 8

6 Look up the amino acids for codons ATA, CTA, ACC, and ATC.

```
aminolookup('code',nt2aa('ATA'))  
aminolookup('code',nt2aa('CTA'))  
aminolookup('code',nt2aa('ACC'))  
aminolookup('code',nt2aa('ATC'))
```

MATLAB displays the following

```
Ile isoleucine  
Leu leucine  
Thr threonine  
Ile isoleucine
```


Amino Acid Conversion and Composition

Determining the relative amino acid composition of a protein will give you a characteristic profile for the protein. Often, this profile is enough information to identify a protein. Using the amino acid composition, atomic composition, and molecular weight, you can also search public databases for similar proteins.

After you locate an open reading frame (ORF) in a gene, you can convert it to an amino sequence and determine its amino acid composition. This procedure uses the human mitochondria genome as an example. See “Open Reading Frames” on page 2-12.

- 1 Convert a nucleotide sequence to an amino acid sequence. In this example, only the protein-coding sequence between the start and stop codons is converted.

```
ND2AASeq = nt2aa(ND2Seq, 'geneticcode', ...
                'Vertebrate Mitochondrial');
```

The sequence is converted using the Vertebrate Mitochondrial genetic code. Because the property `AlternativeStartCodons` is set to `'true'` by default, the first codon `att` is converted to `M` instead of `I`.

```
MNPLAQPVIYSTIFAGTLITALSSHFFTWVGLMMLAFIPVLTCKMNP
RSTEA AIKYFLTQATASMILLMAILFNMLSGQWTMTNTTNQYSSLMIMM
AMAMKLGMAPFFHFVPEVTQGTPLTSGLLLLTWQKLAPISIMYQISPSLN
VLLLLTSLILSIMGSWGGLNQTQLRKILAYSSITHMGWMMAVLPYNPNM
TILNLTIIYIILTTTAFLLLNLSSTTTLLLSRTWNKLTWLTPLIPSTLLS
LGGLPPLTGFLPKWAIIEEFTKNNSLIPTIMATITLLNLYFYLRRIYST
SITLLPMSNNVMMKWQFEHTKPTPFLPTLIALTTLLLPISPFMLMIL
```

- 2 Compare your conversion with the published conversion in GenPept.

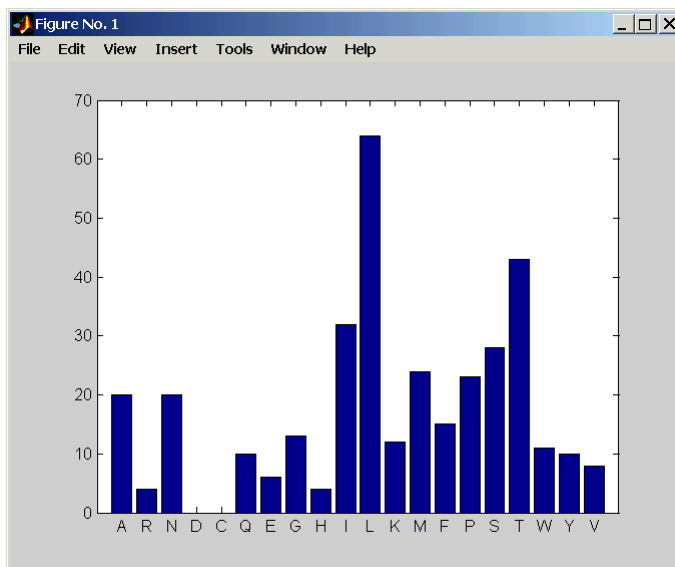
```
ND2protein = getgenpept('NP_536844', 'sequenceonly', true)
```

MATLAB gets the published conversion from the NCBI database and reads it into the MATLAB workspace.

- 3 Count the amino acids in the protein sequence.

```
account(ND2AASeq, 'chart', 'bar')
```

MATLAB draws a bar graph. Notice the high content for leucine, threonine and isoleucine, and also notice the lack of cysteine and aspartic acid.



4 Determine the atomic composition and molecular weight of the protein.

```
atomiccomp(ND2AASeq)  
molweight (ND2AASeq)
```

MATLAB displays the following.

```
ans =  
C: 1818  
H: 3574  
N: 420  
O: 817  
S: 25  
  
ans =  
3.8960e+004
```

If this sequence was unknown, you could use this information to identify the protein by comparing it with the atomic composition of other proteins in a database.

Example: Sequence Alignment

Determining the similarity between two sequences is a common task in computational biology. Starting with a nucleotide sequence for a human gene, this example uses alignment algorithms to locate a similar gene in another organism.

Finding a Model Organism to Study (p. 2-18)	Use the MATLAB Help browser to search the Web for information.
Getting Sequence Information from a Public Database (p. 2-20)	Find the nucleotide sequence for a human gene in a public database and read the sequence information into MATLAB.
Searching a Public Database for Related Genes (p. 2-23)	Find the nucleotide sequence for a mouse gene related to a human gene, and read the sequence information into MATLAB.
Locating Protein Coding Sequences (p. 2-25)	Convert a sequence from nucleotides to amino acids and identify the open reading frames
Comparing Amino Acid Sequences (p. 2-28)	Use global and local alignment functions to compare two amino acid sequences.

Finding a Model Organism to Study

In this example, you are interested in studying Tay-Sachs disease. Tay-Sachs is an autosomal recessive disease caused by the absence of the enzyme beta-hexosaminidase A (Hex A). This enzyme is responsible for the breakdown of gangliosides (GM2) in brain and nerve cells.

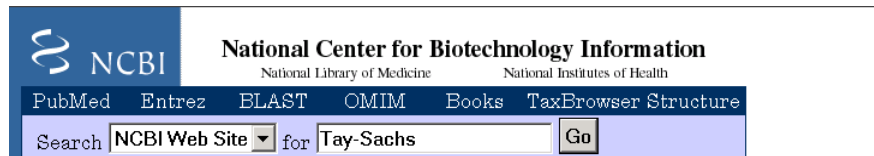
First, research information about Tay-Sachs and the enzyme that is associated with this disease, then find the nucleotide sequence for the human gene that codes for the enzyme, and finally find a corresponding gene in another organism to use as a model for study.

- 1 Use the MATLAB Help browser to explore the Web. In the MATLAB Command window, type

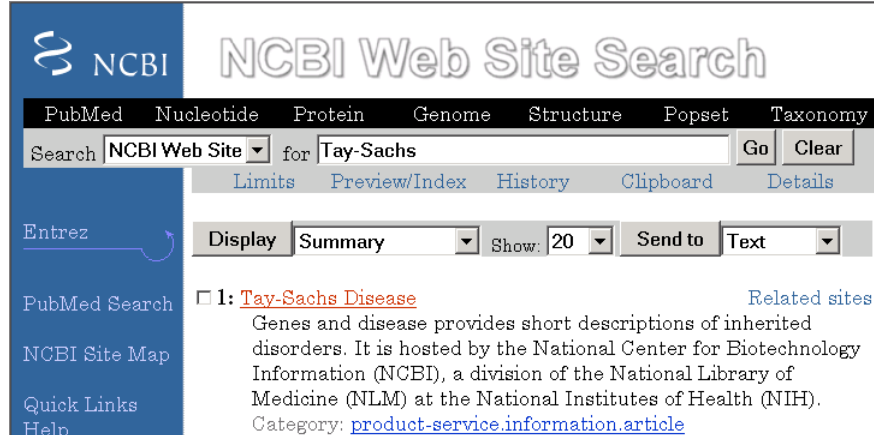
```
web('http://www.ncbi.nlm.nih.gov/')
```

The MATLAB Help browser opens with the home page for the NCBI web site.

- 2 Search the NCBI Web site for information. For example, to search for Tay-Sachs, from the **Search** list, select NCBI Web Site, and in the **for** box, enter Tay-Sachs.



The NCBI Web search returns a list of links to relevant pages.



- 3 Select a result page. For example, click the link labeled **Tay-Sachs Disease**

A page in the genes and diseases section of the NCBI Web site opens. This section provides a comprehensive introduction to medical genetics. In particular, this page contains an introduction and pictorial representation

of the enzyme Hex A and its role in the metabolism of the lipid GM2 ganglioside.



Navigation	<p><i>Genes and Disease</i> → Nutritional and Metabolic Diseases</p> <p>Tay-Sachs disease</p> <p>Tay-Sachs disease, a heritable metabolic disorder commonly associated with Ashkenazi Jews, has also been found in the French Canadians of Southeastern Quebec, the Cajuns of Southwest Louisiana, and other populations throughout the world. The severity of expression and the age at onset of Tay-Sachs varies from infantile and juvenile forms that exhibit paralysis, dementia, blindness and</p>
<p><u>About this book</u></p> <p><u>Nutritional and Metabolic Diseases</u></p> <p><u>Adrenoleukodystrophy</u></p> <p><u>Diabetes, type 1</u></p> <p><u>Gaucher disease</u></p> <p><u>Glucose galactose malabsorption</u></p> <p><u>Hereditary hemochromatosis</u></p> <p><u>Maple syrup urine disease</u></p> <p><u>Menkes syndrome</u></p>	

4 After completing your research, you have concluded the following:

The gene HEXA codes for the alpha subunit of the dimer enzyme hexosaminidase A (Hex A), while the gene HEXB codes for the beta subunit of the enzyme. A third gene, GM2A, codes for the activator protein GM2. However, it is a mutation in the gene HEXA that causes Tay-Sachs.

Getting Sequence Information from a Public Database

Many public databases for nucleotide sequences (for example, GenBank, EMBL-EBI) are accessible from the Web. The MATLAB Command Window with the MATLAB Help browser provide an integrated environment for searching the Web and bringing sequence information into MATLAB.

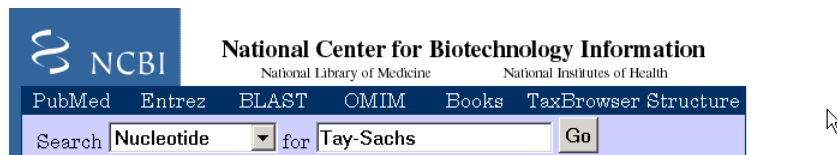
After you locate a sequence, you need to move the sequence data into the MATLAB workspace.

- 1 Open the MATLAB Help browser to the NCBI web site. In the **MATLAB Command Window**, type

```
web('http://www.ncbi.nlm.nih.gov/')
```

The MATLAB Help browser window opens with the NCBI home page.

- 2 Search for the gene you are interested in studying. For example, from the **Search** list, select Nucleotide, and in the **for** box enter Tay-Sachs.



The search returns entries for the genes that code the alpha and beta subunits of the enzyme hexosaminidase A (Hex A), and the gene that codes the activator enzyme. The NCBI reference for the human gene HEXA has accession number NM_000520.

The screenshot shows the NCBI Nucleotide search interface. The search term 'Tay-Sachs' is entered in the search box. The results are displayed in a table with three entries:

Item	Accession	Description	Links
1:	NM_000405	Homo sapiens GM2 ganglioside activator protein (GM2A), mRNA gi 16507969 ref NM_000405.2 [16507969]	Links
2:	NM_000521	Homo sapiens hexosaminidase B (beta polypeptide) (HEXB), mRNA gi 13128866 ref NM_000521.2 [13128866]	Links
3:	NM_000520	Homo sapiens hexosaminidase A (alpha polypeptide) (HEXA), mRNA gi 13128865 ref NM_000520.2 [13128865]	Links

3 Get sequence data into MATLAB. For example, to get sequence information for the human gene HEXA, type

```
humanHEXA = getgenbank('NM_000520')
```

Note Blank spaces in GenBank accession numbers use the underline character. Entering 'NM 00520' returns the wrong entry.

The human gene is loaded into the MATLAB workspace as a structure.

```
humanHEXA =
    LocusName: 'NM_000520'
    LocusSequenceLength: '2255'
    LocusNumberofStrands: ''
    LocusTopology: 'linear'
```



```

LocusMoleculeType: 'mRNA'
LocusGenBankDivision: 'PRI'
LocusModificationDate: '13-AUG-2006'
Definition: 'Homo sapiens hexosaminidase A (alpha polypeptide) (HEXA), mRNA.'
Accession: 'NM_000520'
Version: 'NM_000520.2'
GI: '13128865'
Project: []
Keywords: []
Segment: []
Source: 'Homo sapiens (human)'
SourceOrganism: [4x65 char]
Reference: {1x58 cell}
Comment: [15x67 char]
Features: [74x74 char]
CDS: [1x1 struct]
Sequence: [1x2255 char]
SearchURL: [1x108 char]
RetrieveURL: [1x97 char]

```

Searching a Public Database for Related Genes

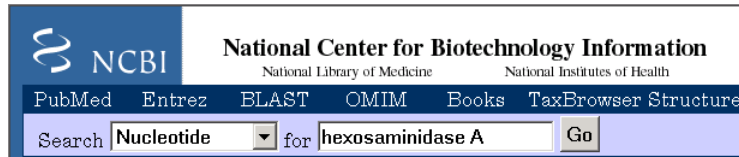
The sequence and function of many genes is conserved during the evolution of species through homologous genes. Homologous genes are genes that have a common ancestor and similar sequences. One goal of searching a public database is to find similar genes. If you are able to locate a sequence in a database that is similar to your unknown gene or protein, it is likely that the function and characteristics of the known and unknown genes are the same.

After finding the nucleotide sequence for a human gene, you can do a BLAST search or search in the genome of another organism for the corresponding gene. This procedure uses the mouse genome as an example.

- 1 Open the MATLAB Help browser to the NCBI Web site. In the **MATLAB Command** window, type

```
web('http://www.ncbi.nlm.nih.gov')
```

- 2 Search the nucleotide database for the gene or protein you are interested in studying. For example, from the **Search** list, select Nucleotide, and in the **for** box enter hexosaminidase A.



The search returns entries for the mouse and human genomes. The NCBI reference for the mouse gene HEXA has accession number AK080777.

Search **Nucleotide** for **hexosaminidase A** Go Clear

Display **Summary** Show: **20** Send to **Text**

Items 1-16 of 16 One page.

1: [AK080777](#) Links

Mus musculus 9.5 days embryo parthenogenote cDNA, RIKEN full-length enriched library, clone:B130019N09 product:hexosaminidase A, full insert sequence gi|26348756| dbj|AK080777.1|[26348756]

3 Get sequence information for the mouse gene into MATLAB. Type

```
mouseHEXA = getgenbank('AK080777')
```

The mouse gene sequence is loaded into the MATLAB workspace as a structure.

```

mouseHEXA =

        LocusName: 'AK080777'
      LocusSequenceLength: '1839'
    LocusNumberofStrands: ''
        LocusTopology: 'linear'
      LocusMoleculeType: 'mRNA'
    LocusGenBankDivision: 'HTC'
  LocusModificationDate: '02-SEP-2005'
        Definition: [1x150 char]
        Accession: 'AK080777'
          Version: 'AK080777.1'
            GI: '26348756'
        Project: []
        Keywords: 'HTC; CAP trapper.'
        Segment: []
          Source: 'Mus musculus (house mouse)'
    SourceOrganism: [4x65 char]
      Reference: {1x8 cell}
        Comment: [8x66 char]
        Features: [33x74 char]
          CDS: [1x1 struct]
        Sequence: [1x1839 char]
      SearchURL: [1x107 char]
    RetrieveURL: [1x97 char]

```

Locating Protein Coding Sequences

A nucleotide sequence includes regulatory sequences before and after the protein coding section. By analyzing this sequence, you can determine the nucleotides that code for the amino acids in the final protein.

After you have a list of genes you are interested in studying, you can determine the protein coding sequences. This procedure uses the human gene HEXA and mouse gene HEXA as an example.

- 1 If you did not retrieve gene data from the Web, you can load example data from a MAT-file included with the Bioinformatics Toolbox. In the **MATLAB Command** window, type

```
load hexosaminidase
```

MATLAB loads the structures humanHEXA and mouseHEXA into the MATLAB workspace.

- 2 Look for open reading frames in the human gene. For example, for the human gene HEXA, type

```
humanORFs=seqshoworfs(humanHEXA.Sequence)
```

seqshoworfs creates the output structure humanORFs. This structure gives the position of the start and stop codons for all open reading frames (ORFs) on each reading frame.

```
humanORFs =
```

```
1x3 struct array with fields:
```

```
Start
```

```
Stop
```

The Help browser opens with a listing for the three reading frames with the ORFs colored blue, red, and green. Notice that the longest ORF is on the third reading frame.

Frame 3

```

000001 cctccgagaggggagaccagcgggccaatgacaagctccaggetttggttttcgctgctgctggc
000065 ggcagcgttcgcaggacggggacggccctctggccctggcctcagaacttccaaacctccgac
000129 cagcgcctacgtcctttaccgcaacaactttcaatccagtagatgtcagctcggccgcgcagc
000193 ccggtgctcagtcctcgcagagggcctccagcctatcgtgacctgctttcgggtccgggtc
000257 ttggccccgctccttacctcacagggaaacggcatacactggagaagaatgtgtggttgcctc
000321 gtagtccacctggatgtaaccagctcctactttggagtcagtggaatataacctgacca
000385 taaatgatgaccaggtttactcctctctgagactgtctggggagctcctccaggtctggagac
000449 ttttagccagcttggttggaaatctgctgagggcacattctttatcaacaagactgagattgag
000513 gactttccccgctttcctcaccgggcttgcctgttggatacatctgccattacctgccactct
000577 ctagcatcctggacactctggatgtcatggcgtacaataaattgaacgtgttccactggcatct
000641 ggtagatgatacctccttcccatatgagagcttcaactttccagagctcatgagaaggggtcc
000705 tacaacctgtcaccacactctacacagcacaggatgtgaaggaggtcattgaaacgcacggc
000769 tccgggtatccgtgtgcttgcagagtttgacactcctggccacactttgctcctggggaccagg
000833 tatccctggattactgactccttgcctactctgggtctgagccctctggcacccttggaccagtg
000897 aatcccagctcacaataacctatgagttcatgagcaccattctcttagaagtcagctctgtct
000961 tcccagattttatcttcatcttggaggagatgaggttgatttcaactcctggaagtccaacct
001025 agagatccaggactttatgaggaagaaggcttcgggtgaggactcaagcagctggagtccttc
001089 tacatccagacgctgctggacatcgtctctcttattggcaagggtcatgtggtgtggcaggagg
001153 tgtttgatataaagttaagattcagccagacacaatcacaaggtgtggcagagggatctcc
001217 agtgaactatgaaggagctggaactggtcaccaggccggctccgggcccctctctctgcc
001281 ccttggtaacctgaacctatctctatggccctgactggaaggatttctacgtagtggacccc
001345 tggcatttgaaggtaccctgagcagaaggctctggtgatgtgtggagaggcttgtatgtgggg
001409 agaatatgtggacaacaacaacctggtcccaggctctggcccagagcaggggctgttgcgaa
001473 aggtgtggagcaacaagttgacatctgacctgacatttgcctatgaacgtttgtcacacttcc
001537 gctgtgagttgctgaggcaggtgtccaggcccaacctcaatgtaggctctctgtgagcagga
001601 gtttgaacagacctgagcccccaggcaccgaggagggtgctggctgtagggtgaatggtagtggag
001665 ccaggcttccactgcctcctggccaggggacggagccccttgcctctgctgcccttgcctgct
001729 gccctgtgcttggagagaaaggggcccgtgctggcctcgcattcaataaagagttaatgtggc
001793 attttctataataaacatggatacctgtgttataaaaaaaaaaggtggaatggcgttagggta
001857 agggcacagccaggtggagtcagtgctgcccctgaggtcttttaagttgagggctgggaatg
001921 aaacctatagcctttgtgctgtctgcccctgcccgtgagctatgtaacctcccctcccactcctg
001985 accatattccagacacctgccctaactcctcagcctgctcacttcaactctgcattatctcca
002049 aggcgctgggtataggaaaaagatgtaggggcttggaggtgtctggacagttggggagggctcc
002113 agacccaaacctggtcacaanaagagcctctccccatgcatactcaccctcccctccctaga
002177 gctattctccttgggtttcttgcctgctgcaattttatacaaccattatataaatattataaa
002241 cacatattgtctct

```

3 Locate open reading frames (ORFs) on the mouse gene. Type

```
mouseORFs = seqshoworfs(mouseHEXA.Sequence)
```

seqshoworfs creates the structure mouseORFS.

```
mouseORFs =
```

```
1x3 struct array with fields:
Start
```

Stop

The mouse gene shows the longest ORF on the first reading frame.

Frame 1

```

000001 gctgctggaaggggagctggccgggtgggccaatggccggctgcaggctctgggttctgctgctgc
000065 tggcggcggcgttggcttgcctggccacggcactgtggccgtggccccagtaacccaacctc
000129 ccaccggcgcctacacctgtacccaacaacttccagttccggtaaccatgtcagttcggccgcg
000193 caggcggcgtgcgtgctcctcgacgagcccttgcagctaccgtaacctgctcttgggtccg
000257 gctcttggccccgaccagcttctcaaataaacagcaaacgttggggaagaacattctgggtggt
000321 ctcgtgctcacagctgaatgtaatgaatttccctaattggagctcggtagaaaattacacctc
000385 accattaatgatgaccagtggttactcgcctctgagactgtctgggcccgtctccgaggtctgg
000449 agactttcagtcagcttgttggaaatcagctgagggcacgttctttatcaacaagacaagat
000513 taagactttcctcgattccctcacggggcgtactgctggatcacatctcgccattacctgcca
000577 ttgtctagcatcctggatacactggatgtcatggcatacaataaattcaacgtgttccactggc
000641 acttgggtggacgactcttcttcccataatgagagcttcccttcccagagctcaccagaaaggg
000705 gtcctcaacctgtcactcacatctacacagcacaggatgtgaaggaggtcattgaatcgc
000769 aggcttccgggtatccgtgtgctggcagaatttgacactcctggccacacttgtcctgggggc
000833 caggtgccctgggttataaacacctgctactctgggtctcactctctctggcacatttggacc
000897 ggtgaaccccagctcacaacgcacctatgacttcagagcacactctcctggagatcagctca
000961 gtcttcccggactttatctccacctgggaggggatgaagtcagcttccctcctggaagtcca
001025 acccaacatccaggccttcatgaagaaaagggcttactgactcaagcagctggagtcctt
001089 ctacatccagacgctgctggacatcgtctctgatgatgacaagggctatgtggtgtggcaggag
001153 gtattgataataaagtgaaggtcggccagatacaatcacaaggtgtggccgggaagaatgc
001217 cagtagagtacatgttggagatgcaagatcacccagggctggcttccggcccctgctgtctgc
001281 tcctggtacctgaaccgtgtaaagtatggccctgactggaaggacatgtacaagtggagccc
001345 ctggcgtttcatggtacgcctgaacagaaggtctggtcattggaggggagggcctgtatgtggg
001409 gagagtgtggacagcaccacctgggtcccagactctggcccagagcgggtgcccgtcgtga
001473 gagactgtggagcagtaacctgacaactaatatagacttggcctttaaacttgtcgcatttc
001537 cgttgtgagctggtgaggagaggaatccaggcccagcccactcagtgtaggctgctgtgagcagg
001601 agtttgagcagacttgagccaccagtgctgaacaccaggaggttgcctgtcctttgagtcagct
001665 gcgctgagcaccagggaggtgctggccttaagagagcaggtcccggggcagggctaatcttcc
001729 actgcctcccggccaggggagagcacccttggccctgtgccctgtgactacagagaaggagg
001793 ctggtgctggcactggtgttcaataaagatctatgtggcaatttctc

```

Comparing Amino Acid Sequences

You could use alignment functions to look for similarities between two nucleotide sequences, but alignment functions return more biologically meaningful results when you are using amino acid sequences.

After you have located the open reading frames on your nucleotide sequences, you can convert the protein coding sections of the nucleotide sequences to their corresponding amino acid sequences, and then you can compare them for similarities.

- Using the identified open reading frames, convert the DNA sequence to the amino acid sequences. Type

```
mouseProtein = nt2aa(mouseHEXA.Sequence)
```

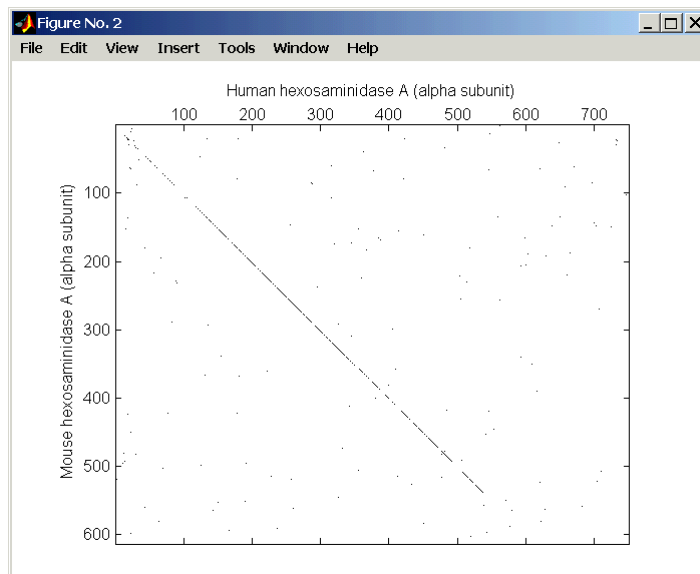
Remember that the human HEXA gene was on the third reading frame, so you need to indicate which frame to use.

```
humanProtein = nt2aa(humanHEXA.Sequence, 'frame', 3)
```

- Draw a dot plot comparing the human and mouse amino acid sequences. Type

```
seqdotplot(mouseProtein,humanProtein,4,3)
ylabel('Mouse hexosaminidase A (alpha subunit)')
xlabel('Human hexosaminidase A (alpha subunit)')
```

Dot plots are one of the easiest ways to look for similarity between sequences. The diagonal line shown below indicates that there may be a good alignment between the two sequences.



- 3** Globally align the two amino acid sequences, using the Needleman-Wunsch algorithm. Type

```
[GlobalScore, GlobalAlignment] = nwalignment(humanProtein,...
                                             mouseProtein)
showalignment(GlobalAlignment)
```

`showalignment` displays the global alignment of the two sequences in the Help browser. Notice that the calculated identity between the two sequences is 64.5 %.

The alignment is very good for the first 550 nucleotides, after which the two sequences appear to be unrelated. Notice that there is a stop (*) in the sequence at this point. If you shorten the sequence to include only the amino acids that are in the protein (after the first methionine and before the first stop) you might get a better alignment.

- 4** Trim the sequence from the first start amino acid (usually M) to the first stop (first *) and then try alignment again. Find the indices for the stops in the sequences.

```
humanStops = find(humanProtein == '*')
```

```
humanStops =  
    538    550    652    661    669
```

```
mouseStops = find(mouseProtein == '*')
```

```
mouseStops =  
    539    557    574    606
```

Looking at the amino acid sequence for humanProtein, the first M is at position 9, while the first M for the mouse protein is at 11.

- 5** Truncate the sequence to include only amino acids in the protein and the stop.

```
humanProteinORF = humanProtein(9:humanStops(1));
```

```
humanProteinORF =  
MTSSRLWFSLLLAAAFAGRATALWPWPQNFQTSQDRYVLYPNNFQFQYDV  
SSAAQPGCSVLDEAFQRYRDLLFGSGSWPRPYLTGKRHTLEKNVLVSVV  
TPGCNQLPTLESVENYTLTINDDQCLLLSETVWGALRGLETFSQLVWKSA  
EGTFFINKTEIEDFPRFPHRGLLLDTSRHYLPLSSILDTLDMAYNKLNV  
FHWHLVDDPSFPYESFTFPELMRKGSYNPVTHIYTAQDVKEVIEYARLRG  
IRVLAEFDTPGHTLSWGPPIGILLPCYSGSEPSGTFGPVNPVSLNNTYEF  
MSTFFLEVSSVFPDFYLHLGGDEVDFTCWKSNIQDFMRKKGFGEFQKQ  
LESFYIQTLLDIVSSYGKGYVVWQEVFDNKVKIQPDTIIQVWREDIPVNY  
MKELELVTKAGFRALLSAPWYLNRI SYGPDWKDFYVVEPLAFEGTPEQKA  
LVIGGEACMWGEYVDNTNLVPRLWPRAGAVAERLWSNKLTSDLTFAYERL
```

```
SHFRCELLRRGVQAQPLNVGFCEQEFEQT*
```

```
mouseProteinORF = mouseProtein(11:mouseStops(1))
```

```
mouseProteinORF =
MAGCRLWVSLLLAAALACLATALWPWPQYIQT YHRRYTLYPNNFQFRYHV
SSAAQAGCVVLDEAFRRYRNLLFGSGSWPRPSFSNKQQLGKNILVVSVV
TAECEFPNLESVENYTLTINDDQCLLASETVW GALRGLETFSQLVWKS A
EGTFFINKTKIKDFPRFPHRGVLLDTSRHYLPLSSILDTLDMAYNKFNV
FHWHLVDDSSFPYESFTFPELTRKGSFNPVTHIYTAQDVKEVIEYARLRG
IRVLAEFDTPGHTLSWGP GAPGLLTPCYSGSHLSGTFGPVNP SLNSTYDF
MSTLFLEISSVFPDFYLHLGGDEVDFTCWKS NPNIQAFM KKKGFTDFKQL
ESFYIQTLLDIVSDYDKGYV VVQEVFDNKVKVRPDTIIQVWREEMPVEYM
LEMQDITRAGFRALLSAPWYLN RVKYGPDWKDMYKVEPLAFHG TPEQKAL
VIGGEACMWGEYVDSTNLV PRLWPRAGAVAERLWSSNLTTNIDFAFKRLS
HFRCELVRRGIQAQPI SVGCCEQEFEQT*
```

6 Globally align the trimmed amino acid sequences. Type

```
[Score, Alignment] = nalign(humanProteinORF,...
                             mouseProteinORF);
showalignment(Alignment)
```

showalignment displays the results for the second global alignment. Notice that the percent identity for the untrimmed sequences is 54% and with trimmed sequences 83.3 percent.


```

humanORFs = seqshoworfs(humanHEXA.Sequence);
mouseORFs = seqshoworfs(humanHEXA.Sequence);

humanPORF = nt2aa(humanHEXA.Sequence(humanORFs(3).Start(1):...
                                     humanORFs(3).Stop(1)))
mousePORF = nt2aa(mouseHEXA.Sequence(mouseORFs(1).Start(1):...
                                     mouseORFs(1).Stop(1)))
[Scale, Alignment] = nwalgn(humanPORF, mousePORF)

```

Show the alignment in the Help browser.

```
showalignment(Alignment)
```

The result from first truncating a nucleotide sequence before converting to an amino acid sequence is the same as the result from truncating the amino acid sequence after conversion. See the result in step 6.

An alternative method to working with subsequences is to use a local alignment function with the nontruncated sequences.

- 8** Locally align the two amino acid sequences using a Smith-Waterman algorithm. Type

```
[LocalScore, LocalAlignment] = swalgn(humanProtein,...
                                       mouseProtein)
```

```
LocalScore =
    1057
```

```
LocalAlignment
RGDQR-AMTSSRLWFSLLLAAAFAGRATALWPWPQNFQTS DQRYV . . .
|| | ||:: || | |||||:| ||||| | :|| :||: . . .
RGAGRWAMAGCRLWVSLLLAAALACLATALWPWPQYIQTYHRRYT . . .
```

swalgn displays the local alignment of two sequences in the Help browser.

- 9** Show the alignment in color.

```
showalignment(LocalAlignment)
```


Sequence Tool

The Sequence Tool is a graphical interface (GUI) that integrates many of the sequence functions in the Bioinformatics Toolbox. Instead of entering commands in the MATLAB Command Window, you can select and enter options.

Importing a Sequence (p. 2-37)	Get sequence information from the NCBI Web database.
Viewing Nucleotide Sequence Information (p. 2-39)	View a graphic representation of sequence information for ORFs and CDSs.
Searching for Words (p. 2-41)	Search for characteristic words and sequence patterns.
Exploring Open Reading Frames (p. 2-42)	Identify the protein coding part of a nucleotide sequence and copy it into a new view.
Viewing Amino Acid Sequence Statistics (p. 2-45)	View an amino acid sequence for an ORF located in a nucleotide sequence.

Importing a Sequence

The first step when analyzing a nucleotide or amino acid sequence is to get sequence information into MATLAB. The `seqtool`, using functions in the Bioinformatics Toolbox, can connect to Web databases and read information into MATLAB.

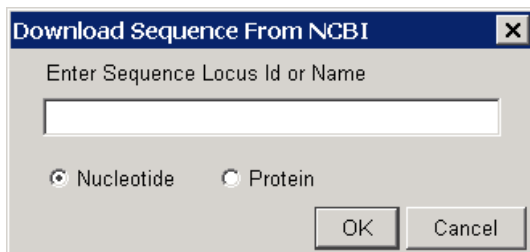
- 1 Open the sequence viewer. In the MATLAB Command Window, type

```
seqtool
```

The Sequence Viewer window opens without a sequence loaded. Notice that the panes to the right and bottom are blank.

- 2 To get a sequence from the NCBI database, select **File > Download Sequence from NCBI**.

The Download Sequence From NCBI dialog box opens.



- 3** In the **Enter Sequence** box, type an accession number for an NCBI database entry. For example, enter NM_000520 and select the **Nucleotide** option button. This is the human gene HEXA that is associated with Tay-Sachs disease.

MATLAB goes to the Web, loads information for the accession number you entered, and calculates some basic statistics.

The screenshot shows the 'Sequence Viewer - NM_000520' application window. The title bar includes standard window controls and the name of the sequence. The menu bar contains 'File', 'Edit', 'Sequence', 'Display', 'Window', and 'Help'. Below the menu is a toolbar with icons for search, zoom, and other functions. A 'Line length' dropdown is set to '60'. The main window is divided into several panes:

- Sequence View (Left):** A tree view showing the sequence and its various annotations: Sequence, ORF, Full Translation, Annotated CDS, CDS with Translation, Complement Sequence, Reverse Complement Sequ, Features, and Comments.
- Base Count (Left):** A table showing the frequency of nucleotides:

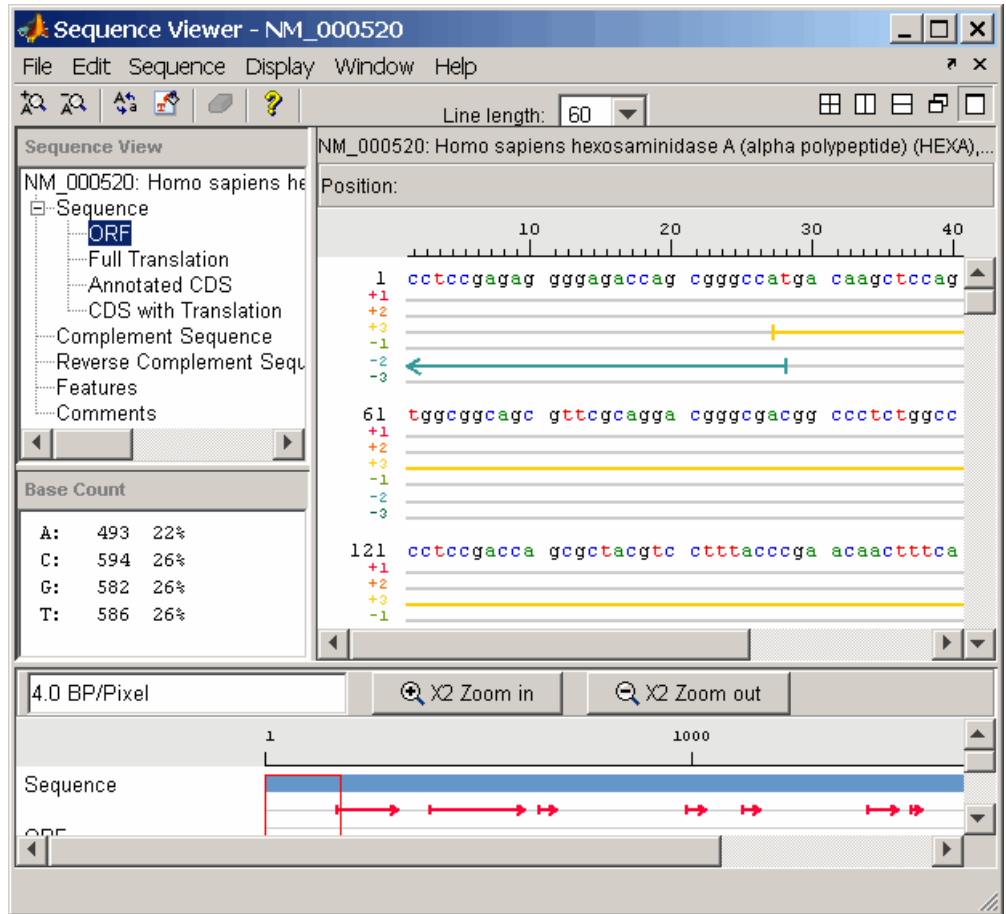
Nucleotide	Count	Percentage
A:	493	22%
C:	594	26%
G:	582	26%
T:	586	26%
- Main Sequence View (Right):** Displays the nucleotide sequence for 'NM_000520: Homo sapiens hexosaminidase A (alpha polypeptide) (HEXA),...'. The sequence is shown in a grid with line numbers (1, 61, 121, 181, 241, 301, 361, 421, 481, 541, 601, 661, 721) on the left. The sequence is color-coded by nucleotide: C (blue), G (green), A (red), and T (black). A scale at the top indicates positions 10, 20, 30, and 40.
- Zoom and Scale (Bottom):** A zoom control set to '4.0 BP/Pixel' with 'X2 Zoom in' and 'X2 Zoom out' buttons. Below this is a horizontal scale from 1 to 1000, with a red box highlighting a region of the sequence.

Viewing Nucleotide Sequence Information

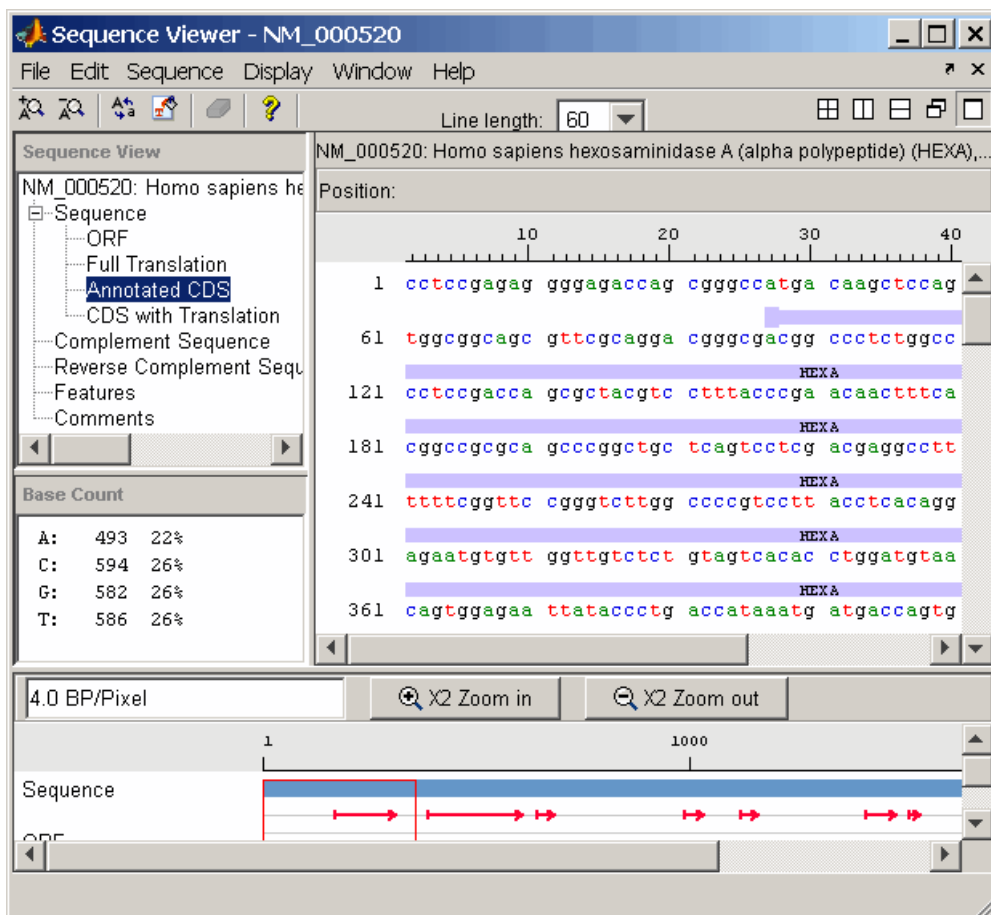
After you import a sequence into seqtool, you can read information stored with the sequence, or you can view graphic representations for ORFs and CDSs.

- 1 In the left pane tree, click **Comments**. The right pane displays general information about the sequence.
- 2 Now click **Features**. The right pane displays NCBI feature information, including index numbers for a gene and any CDS sequences.

3 Click ORF to show the search results for ORFs in the six reading frames.



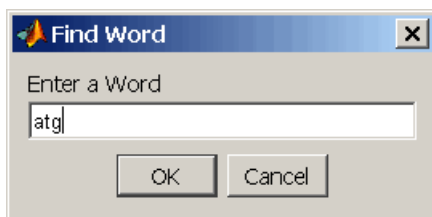
4 Click Annotated CDS to show the protein coding part of a nucleotide sequence.



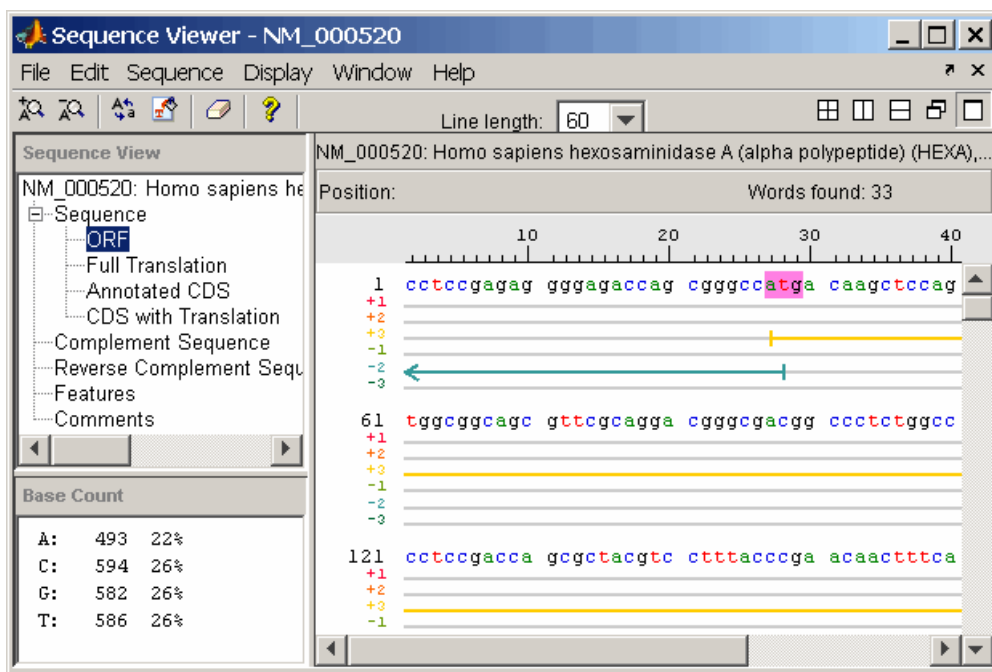
Searching for Words

Search for sequence patterns like the TATAA box and patterns for specific restriction enzymes.

- 1 From the **Sequence** menu, click Find Word.
- 2 In the **Enter a Word** box, type a sequence word or pattern. For example, enter atg.



seqtool searches and displays the location of the selected word.



3 To clear the display, on the toolbar, click the Clear Word Selection button



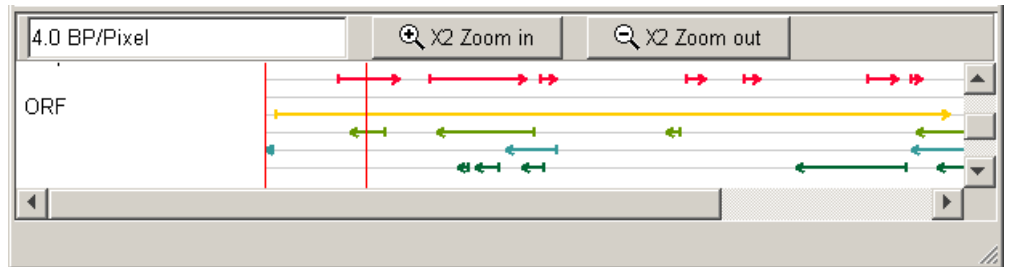
Exploring Open Reading Frames

Identifying coding sections of a nucleotide sequence is a common bioinformatics task. After locating the coding part of a sequence, you can

copy it to a new view, translate it to an amino acid sequence, and continue with your analysis.

- 1 In the left pane, click ORF.

seqtool displays the ORFs for the six reading frames in the right and lower window.

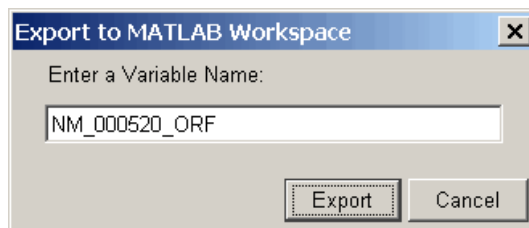


- 2 Click the longest ORF on reading frame 3.

The ORF is highlighted to indicate the part of the sequence that is selected.

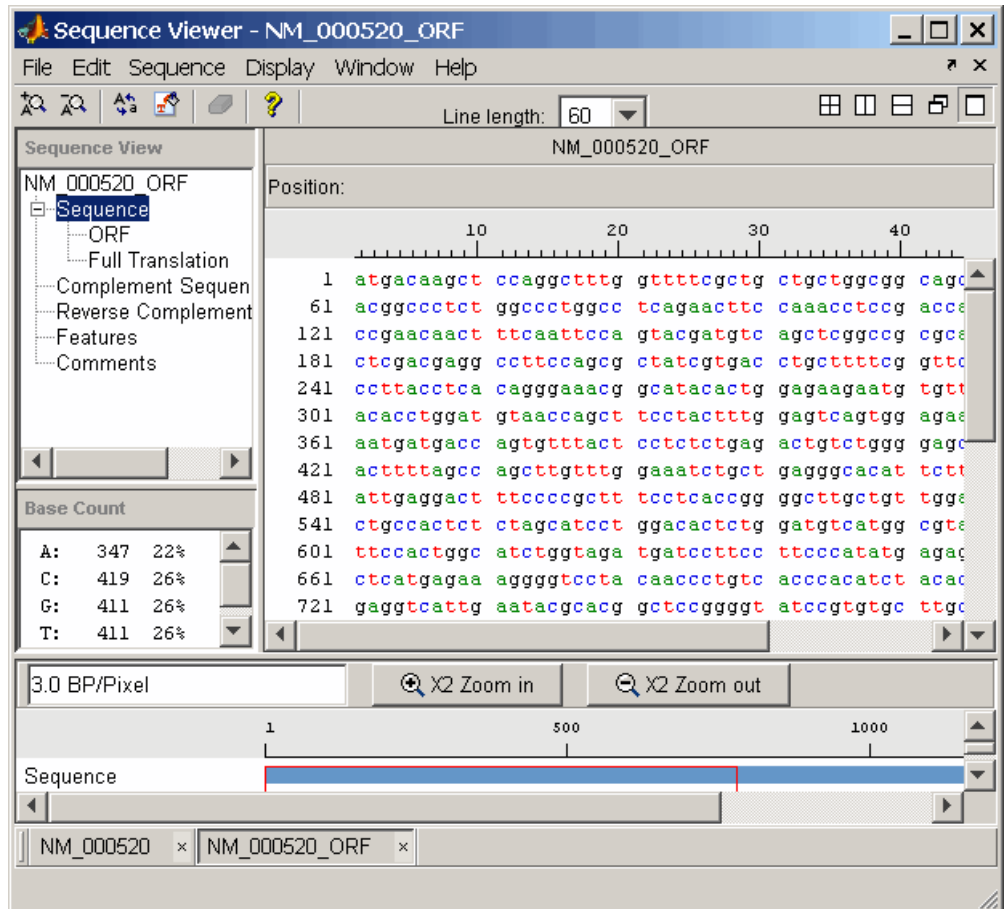


- 3 Right-click the selected ORF and then select **Export to Workspace**. Enter the name of a variable. For example, enter NM_000520_ORF.



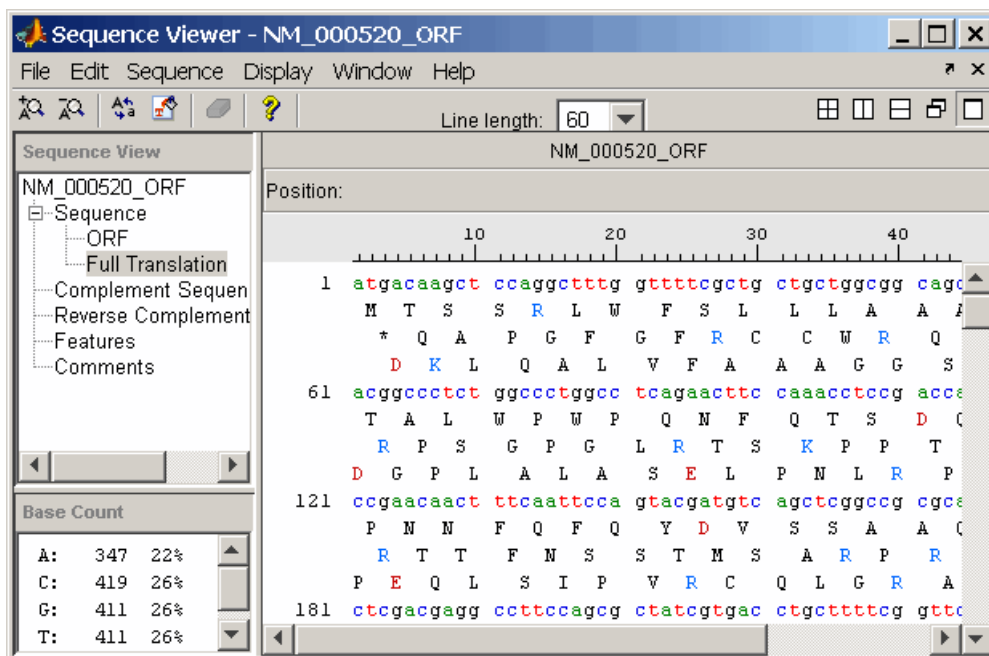
- 4 From the **File** menu, click **Import from Workspace**. Enter the name of a variable with an exported ORF. For example, enter NM_000520_ORF.

seqtool adds a tab at the bottom for the new sequence while leaving the original sequence open.



- 5 In the left pane, click Full Translation. From the **Display** menu, point to **Amino Acid Residue Display** and click **One Letter Code**.

seqtool displays the amino acid sequence below the nucleotide sequence.



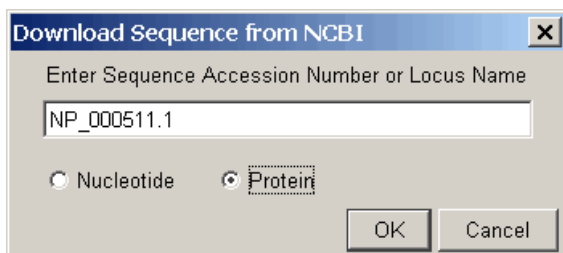
Viewing Amino Acid Sequence Statistics

You can import your own amino acid sequence, or you can get a protein sequence from the Genbank database. In this example, the Genbank accession number NP_000511.1 is the alpha subunit for a human enzyme associated with Tay-Sachs disease.

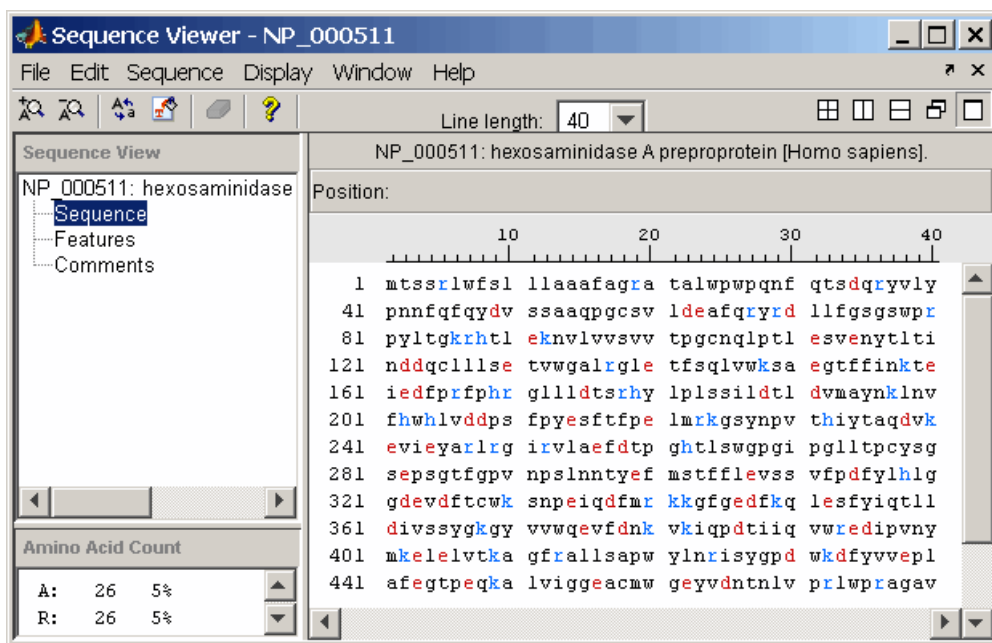
1 Select **File > Download Sequence from NCBI**.

The Download Sequence From NCBI dialog box opens.

2 In the **Enter Sequence** box, type an accession number for an NCBI database entry. For example, enter NP_000511.1 and select the **Protein** option button.



MATLAB goes to the Web and loads sequence information for the accession number you entered.



- 3 Select **Display > Amino Acid Color Scheme**, and then select either **Charge**, **Function**, **Hydrophobicity**, **Structure**, or **Taylor**. For example, select **Charge**.

The display colors change to highlight charge information about the amino acid residues.

Sequence Viewer - NP_000511

File Edit Sequence Display Window Help

Line length: 40

NP_000511: hexosaminidase A preproprotein [Homo sapiens].

Position:

10 20 30 40

```
1  mtssrlwfsl  llaaafagra  talwpwpqnf  qtsdqryvly
41  pnnfqfqydv  ssaqaqpcsv  ldeafqxyrd  llfgsgswpr
81  pyltgkrhrl  eknvlvsvv  tpgcnqlptl  esvenytlti
121 nddqc1llse  tvwgalrgle  tfsqlvwksa  egtffinkte
161 iedfprfphr  gl1ldtsrhy  lplssildtl  dvmaynklnv
201 fhwhlvddps  fpyesftfe  lmrkgsynpv  thiytaqdvk
241 evieyarlrg  irvlaefdtp  ghtlswgpgi  pglltpcysg
281 sepsgtfgpv  npslnntyef  mstfflevss  vfpdfylhlg
321 gdevdfcwk  snpeiqlfmr  kkgfgedfkq  lesfyiqtl1
361 divssykggy  vwqevfdnk  vkiqpdtiq  vwredipvny
401 mkelelvtk  a gfrallsapw  ylnrisygp  d wkdfyvvepl
441 afegtpeqka  lvi ggeacmw  ge yv dntnlv  prlwp ragav
```

Amino Acid Count

A:	26	5%
R:	26	5%

Multiple Sequence Alignment Viewer

The Multiple Sequence Alignment Viewer is a graphical user interface (GUI) that integrates many sequence and multiple alignment functions in the Bioinformatics Toolbox. Instead of entering commands in the MATLAB Command Window, you can select options and manually adjust multiple alignments.

Loading Sequence Data and Viewing the Phylogenetic Tree (p. 2-48)	Load unaligned sequence data into MATLAB and view it in a phylogenetic tree.
Selecting a Subset of Data from the Phylogenetic Tree (p. 2-49)	Select the human and chimp branches.
Aligning Multiple Sequences (p. 2-50)	Multiply align a set of sequences.
Adjusting Multiple Alignments Manually (p. 2-51)	Manually adjust multiple sequence alignment.

Loading Sequence Data and Viewing the Phylogenetic Tree

Load unaligned sequence data into MATLAB and view it in a phylogenetic tree.

1 Load sequence data.

```
load primatesdemodata
```

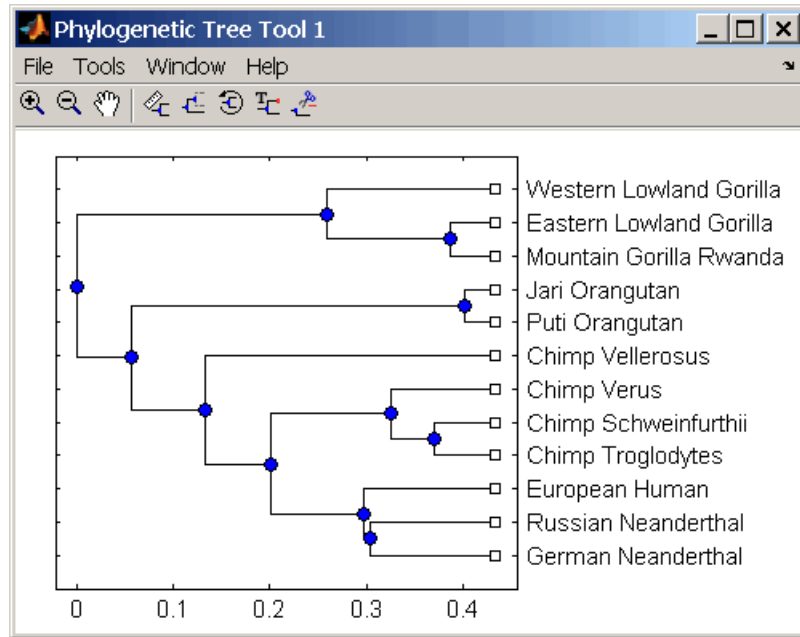
2 Create a phylogenetic tree.

```
tree = seqlinkage(seqpdist(primates), 'single', primates);
```

3 View the phylogenetic tree.

```
view(tree)
```

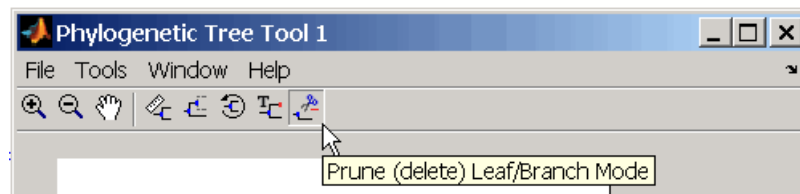
MATLAB creates a `phytree` object in the workspace and loads the sequence data into the Phylogenetic Tree Tool.



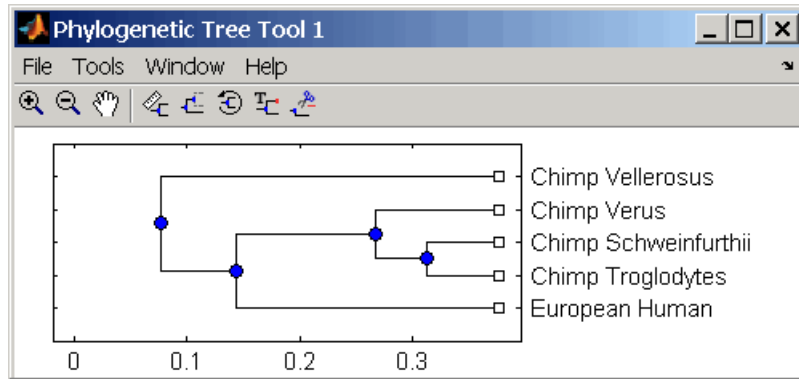
Selecting a Subset of Data from the Phylogenetic Tree

Select the human and chimp branches.

- 1 From the toolbar, click the **Prune** icon.

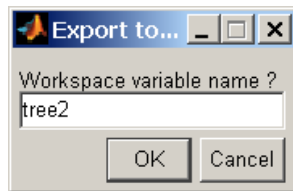


- 2 Click the branches to prune (remove) from the tree. For this example, click the branch nodes for gorillas, orangutans, and Neanderthals.



3 Export the selected branches to a second tree. Select **File > Export to Workspace**, and then click **Only Displayed**.

4 In the Export to dialog box, enter the name of a variable. For example, enter `tree2`, and then click **OK**.



5 Extract sequences from the tree object.

```
primates2 = primates(seqmatch(get(tree2, 'Leafnames'),{primates.Header}));
```

Aligning Multiple Sequences

After selecting a set of related sequences, you can multiply align them and view the results.

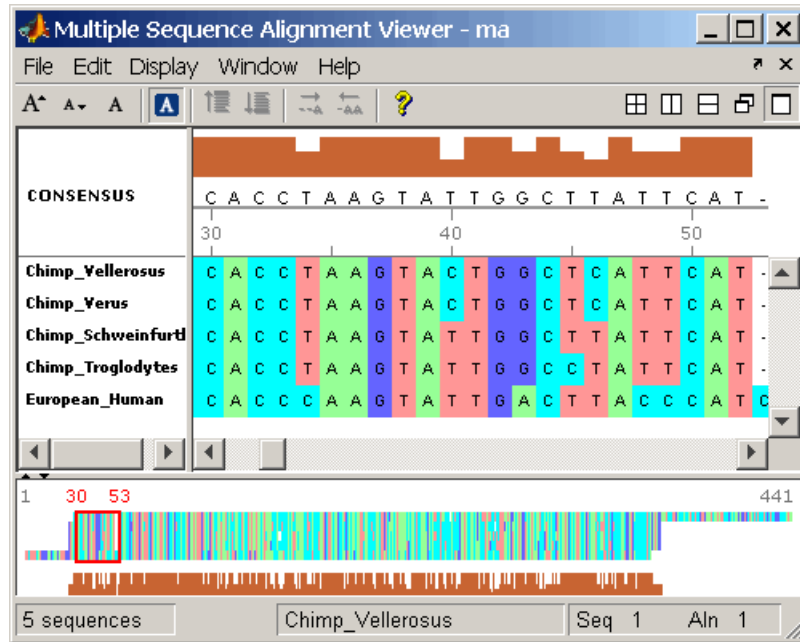
1 Align multiple sequences.

```
ma = multialign(primates2);
```

2 Load aligned sequences in the Multiple Alignment Viewer.

```
multialignviewer(ma);
```

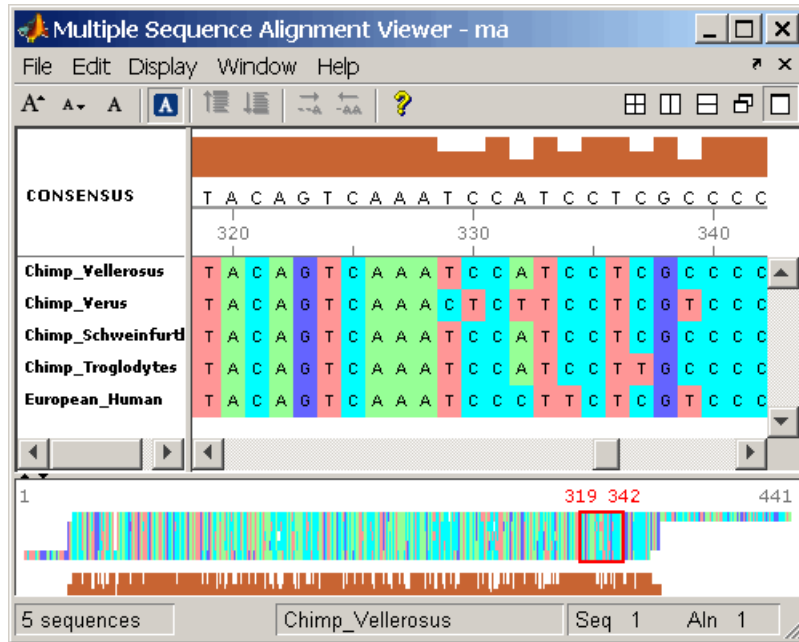
The aligned sequences appear in the Multiple Sequence Alignment Viewer.



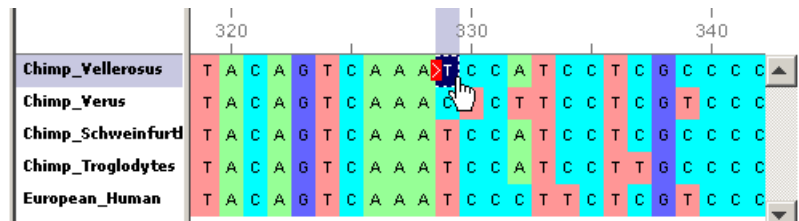
Adjusting Multiple Alignments Manually

Algorithms for aligning multiple sequences do not always produce an optimal result. By visually inspecting the alignment, you can identify areas that could use a manual adjustment to improve the alignment.

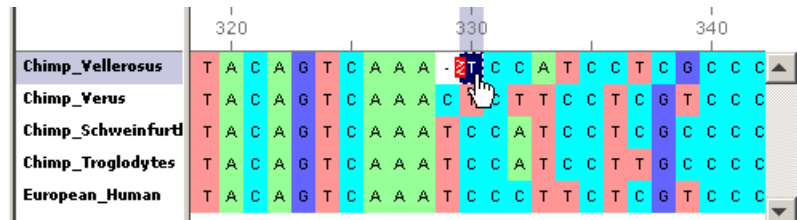
- 1 Identify an area where you could improve the alignment.



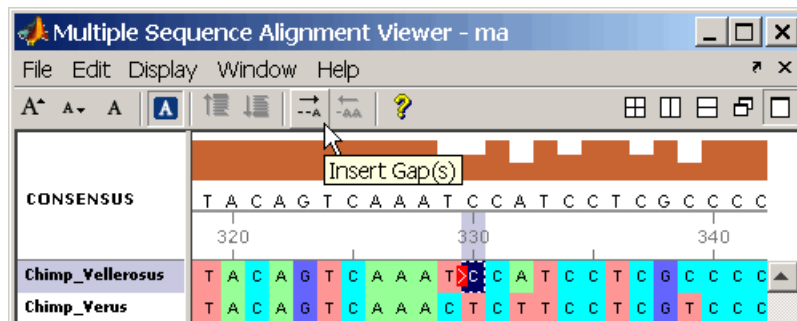
- 2 Click a letter to select it, and then move the cursor over the red direction bar. The cursor changes to a hand.



- 3 Click and drag the sequence to the right to insert a gap. If there is a gap to the left, you can also move the sequence to the left and eliminate the gap.

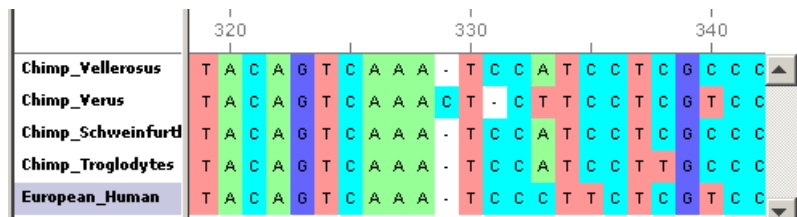


Alternately, to insert a gap, select a character, and then click the **Insert Gap** icon on the toolbar or press the spacebar.



Note You cannot delete or add letters to a sequence, but you can add or delete gaps. If all of the sequences at one alignment position have gaps, you can delete that column of gaps.

4 Continue adding gaps and moving sequences to improve the alignment.



Microarray Analysis

You can use gene expression profiles from microarray data to research the function of cells, compare the differences between healthy and diseased tissue, and observe changes with the application of drugs.

The examples in this chapter will help you to become more familiar with the functions in the Bioinformatics Toolbox for analyzing and visualizing gene expression patterns.

Example: Visualizing
Microarray Data (p. 3-2)

Create figures to visualize microarray data and get the data ready for analysis.

Example: Analyzing Gene
Expression Profiles (p. 3-25)

Analyze microarray data for patterns and plot the results.

Example: Visualizing Microarray Data

This example looks at the various ways to visualize microarray data. The microarray data for this example is from Brown, V.M., Ossadtchi, A., Khan, A.H., Yee, S., Lacan, G., Melega, W.P., Cherry, S.R., Leahy, R.M., and Smith, D.J.; "Multiplex three dimensional brain gene expression mapping in a mouse model of Parkinson's disease"; *Genome Research* 12(6): 868-884 (2002).

Overview of the Mouse Example (p. 3-2)	Pharmacological model of Parkinson's disease (PD) using a mouse brain.
Exploring the Microarray Data Set (p. 3-3)	Import data from the Web into MATLAB.
Spatial Images of Microarray Data (p. 3-5)	Visualize microarray data by plotting image maps.
Statistics of the Microarrays (p. 3-15)	Visualize distributions in microarray data.
Scatter Plots of Microarray Data (p. 3-16)	Visualize expression levels in microarray data.

Overview of the Mouse Example

The microarray data used in this example is available in a web supplement to the paper by Brown et al. and in the file `mouse_a1pd.gpr` included with the Bioinformatics Toolbox.

http://labs.pharmacology.ucla.edu/smithlab/genome_multiplex/

The microarray data is also available on the Gene Expression Omnibus Web site at

<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE30>

The GenePix GPR formatted file `mouse_a1pd.gpr` contains the data for one of the microarrays used in the study. This is data from voxel A1 of the brain of a mouse in which a pharmacological model of Parkinson's disease (PD) was induced using methamphetamine. The voxel sample was labeled with Cy3 (green) and the control, RNA from a total (not voxelated) normal mouse brain,

was labeled with Cy5 (red). GPR formatted files provide a large amount of information about the array, including the mean, median, and standard deviation of the foreground and background intensities of each spot at the 635 nm wavelength (the red, Cy5 channel) and the 532 nm wavelength (the green, Cy3 channel).

Exploring the Microarray Data Set

This procedure uses data from a study about gene expression in mouse brains as an example. See “Overview of the Mouse Example” on page 3-2.

- 1 Read data from a file into a MATLAB structure. For example, in the MATLAB Command Window, type

```
pd = gprread('mouse_a1pd.gpr')
```

MATLAB displays information about the structure:

```
pd =
    Header: [1x1 struct]
      Data: [9504x38 double]
    Blocks: [9504x1 double]
  Columns: [9504x1 double]
     Rows: [9504x1 double]
   Names: {9504x1 cell}
     IDs: {9504x1 cell}
ColumnNames: {38x1 cell}
   Indices: [132x72 double]
     Shape: [1x1 struct]
```

- 2 Access the fields of a structure using `StructureName.FieldName`. For example, you can access the field `ColumnNames` of the structure `pd` by typing

```
pd.ColumnNames
```

The column names are shown below.

```
ans =
    'X'
    'Y'
    'Dia.'
```

```
'F635 Median'  
'F635 Mean'  
'F635 SD'  
'B635 Median'  
'B635 Mean'  
'B635 SD'  
'% > B635+1SD'  
'% > B635+2SD'  
'F635 % Sat.'  
'F532 Median'  
'F532 Mean'  
'F532 SD'  
'B532 Median'  
'B532 Mean'  
'B532 SD'  
'% > B532+1SD'  
'% > B532+2SD'  
'F532 % Sat.'  
'Ratio of Medians'  
'Ratio of Means'  
'Median of Ratios'  
'Mean of Ratios'  
'Ratios SD'  
'Rgn Ratio'  
'Rgn Rt'  
'F Pixels'  
'B Pixels'  
'Sum of Medians'  
'Sum of Means'  
'Log Ratio'  
'F635 Median - B635'  
'F532 Median - B532'  
'F635 Mean - B635'  
'F532 Mean - B532'  
'Flags'
```

- 3** Access the names of the genes. For example, to list the first 20 gene names, type

```
pd.Names(1:20)
```

A list of the first 20 gene names is displayed:

```
ans =
    'AA467053 '
    'AA388323 '
    'AA387625 '
    'AA474342 '
    'Myo1b '
    'AA473123 '
    'AA387579 '
    'AA387314 '
    'AA467571 '
    ' '
    'Spop '
    'AA547022 '
    'AI508784 '
    'AA413555 '
    'AA414733 '
    ' '
    'Snta1 '
    'AI414419 '
    'W14393 '
    'W10596 '
```

Spatial Images of Microarray Data

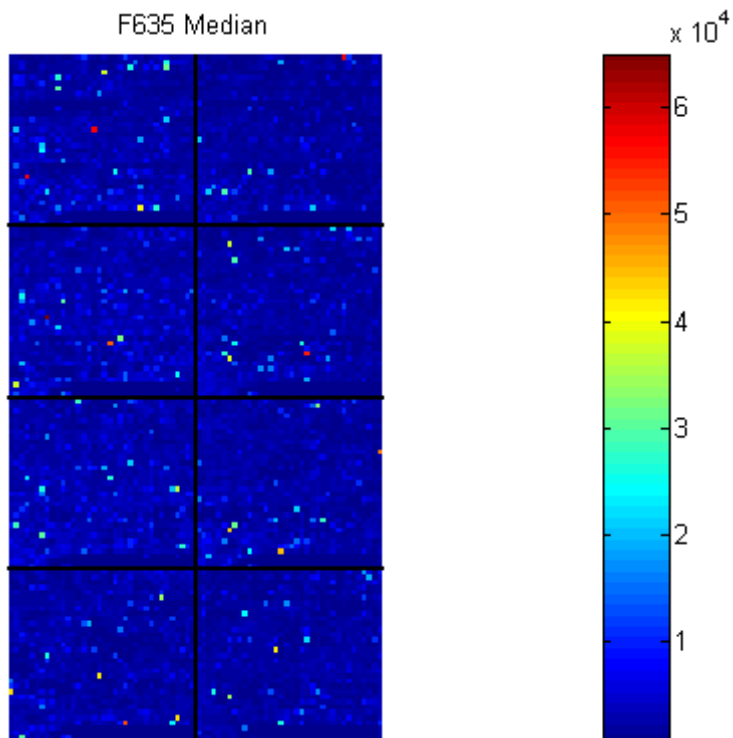
The function `mimage` can take a microarray data structure and create a pseudocolor image of the data arranged in the same order as the spots on the array. In other words, `mimage` plots a spatial plot of the microarray.

This procedure uses data from a study of gene expression in mouse brains. For a list of field names in the MATLAB structure `pd`, see “Exploring the Microarray Data Set” on page 3-3.

- 1 Plot the median values for the red channel. For example, to plot data from the field `F635 Median`, type

```
figure
mimage(pd, 'F635 Median')
```

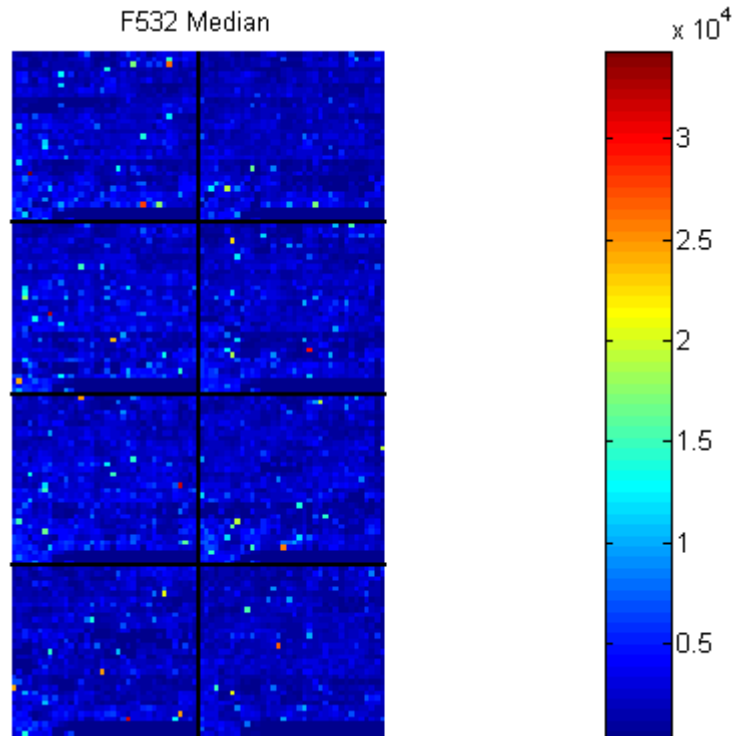
MATLAB plots an image showing the median pixel values for the foreground of the red (Cy5) channel.



- 2** Plot the median values for the green channel. For example, to plot data from the field F532 Median, type

```
figure  
mimage(pd, 'F532 Median')
```

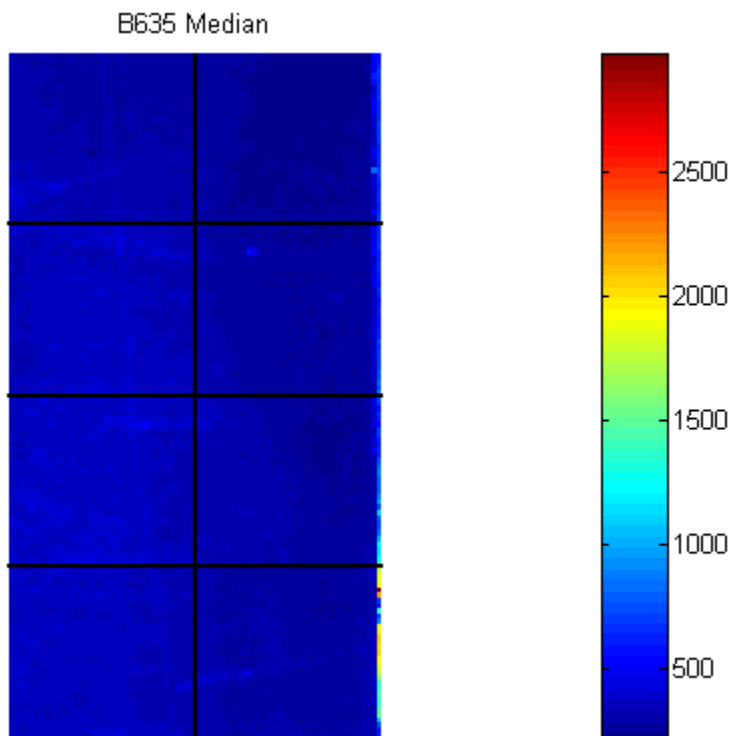
MATLAB plots an image showing the median pixel values of the foreground of the green (Cy3) channel.



- 3** Plot the median values for the red background. The field B635 Median shows the median values for the background of the red channel.

```
figure  
mimage(pd, 'B635 Median')
```

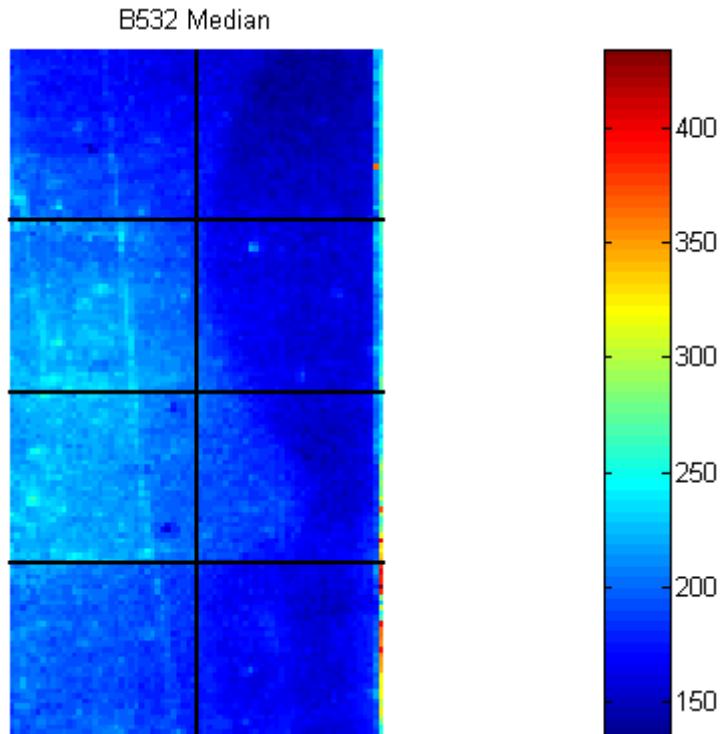
MATLAB plots an image for the background of the red channel. Notice the very high background levels down the right side of the array.



4 Plot the medial values for the green background. The field B532 Median shows the median values for the background of the green channel.

```
figure  
mimage(pd, 'B532 Median')
```


MATLAB plots an image for the background of the green channel.



- 5 The first array was for the Parkinson's disease model mouse. Now read in the data for the same brain voxel but for the untreated control mouse. In this case, the voxel sample was labeled with Cy3 and the control, total brain (not voxelated), was labeled with Cy5.

```
wt = gprread('mouse_a1wt.gpr')
```

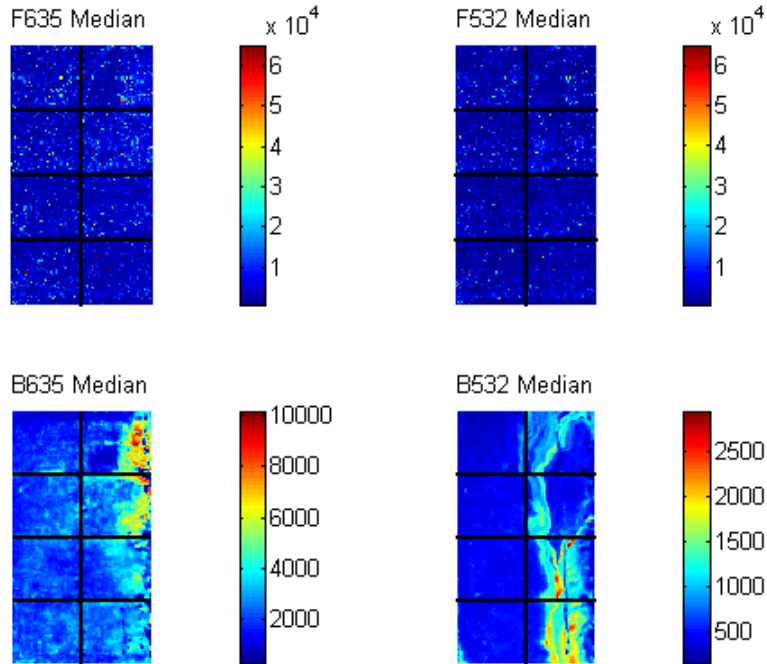
MATLAB creates a structure and displays information about the structure.

```
wt =  
    Header: [1x1 struct]  
    Data: [9504x38 double]  
    Blocks: [9504x1 double]  
    Columns: [9504x1 double]  
    Rows: [9504x1 double]  
    Names: {9504x1 cell}  
    IDs: {9504x1 cell}  
    ColumnNames: {38x1 cell}  
    Indices: [132x72 double]  
    Shape: [1x1 struct]
```

- 6** Use the function `mimage` to show pseudocolor images of the foreground and background. You can use the function `subplot` to put all the plots onto one figure.

```
figure  
subplot(2,2,1);  
mimage(wt, 'F635 Median')  
subplot(2,2,2);  
mimage(wt, 'F532 Median')  
subplot(2,2,3);  
mimage(wt, 'B635 Median')  
subplot(2,2,4);  
mimage(wt, 'B532 Median')
```

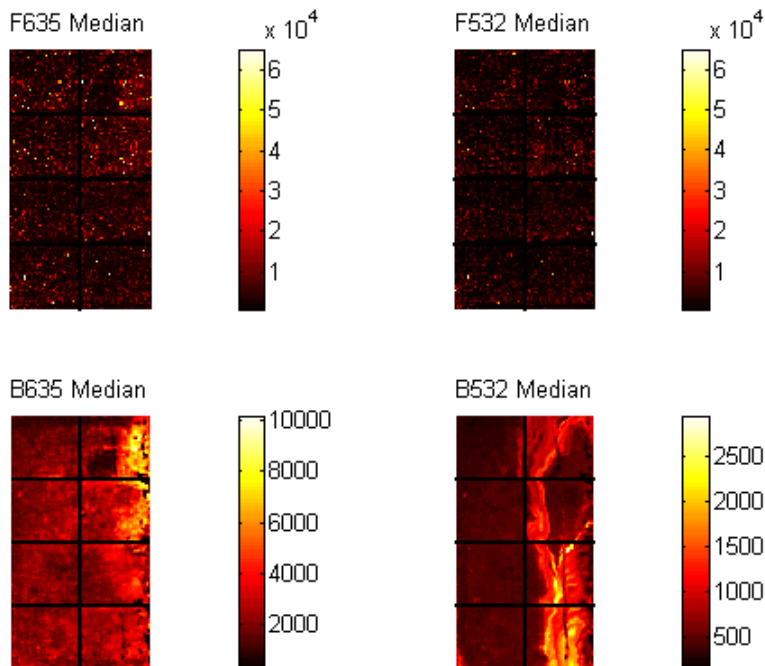
MATLAB plots the images.



- 7 If you look at the scale for the background images, you will notice that the background levels are much higher than those for the PD mouse and there appears to be something nonrandom affecting the background of the Cy3 channel of this slide. Changing the colormap can sometimes provide more insight into what is going on in pseudocolor plots. For more control over the color, try the `colormapeditor` function.

```
colormap hot
```

MATLAB plots the images.



- 8** The function `mimage` is a simple way to quickly create pseudocolor images of microarray data. However if you want more control over plotting, it is easy to create your own plots using the function `imagesc`.

First find the column number for the field of interest.

```
b532MedCol = find(strcmp(wt.ColumnNames, 'B532 Median'))
```

MATLAB displays

```
b532MedCol =  
16
```

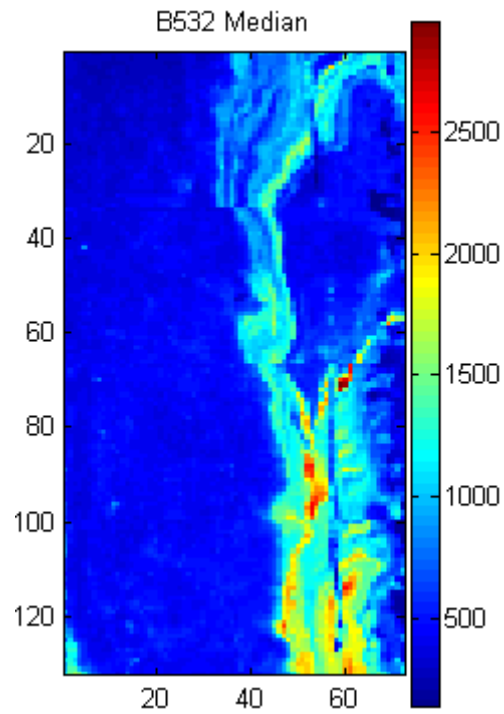
- 9** Extract that column from the field Data.

```
b532Data = wt.Data(:,b532MedCol);
```

10 Use the field `Indices` to index into the Data.

```
figure
subplot(1,2,1);
imagesc(b532Data(wt.Indices))
axis image
colorbar
title('B532 Median')
```

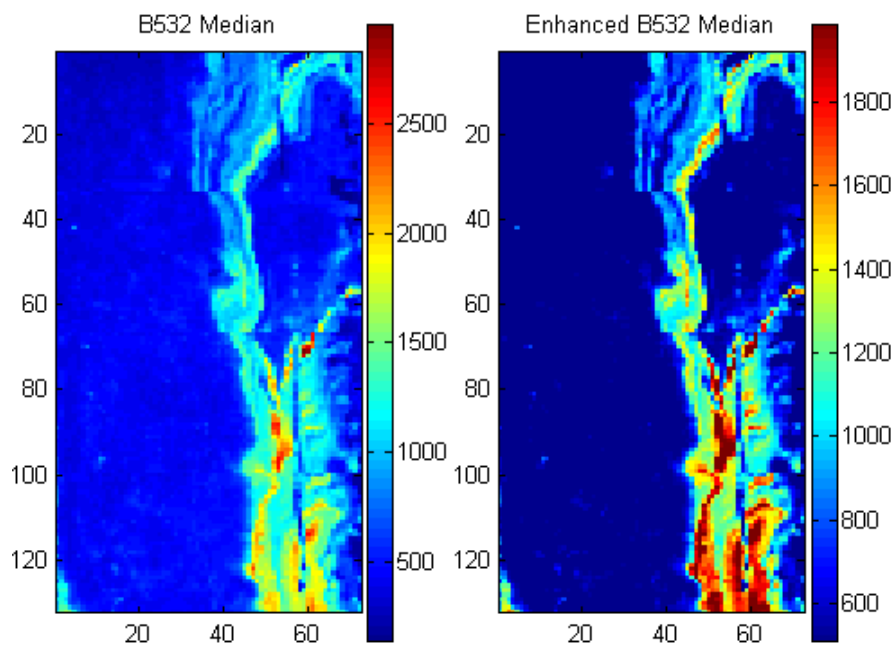
MATLAB plots the image.



- 11** Bound the intensities of the background plot to give more contrast in the image.

```
maskedData = b532Data;  
maskedData(b532Data<500) = 500;  
maskedData(b532Data>2000) = 2000;  
  
subplot(1,2,2);  
imagesc(maskedData(wt.Indices))  
axis image  
colorbar  
title('Enhanced B532 Median')
```

MATLAB plots the images.



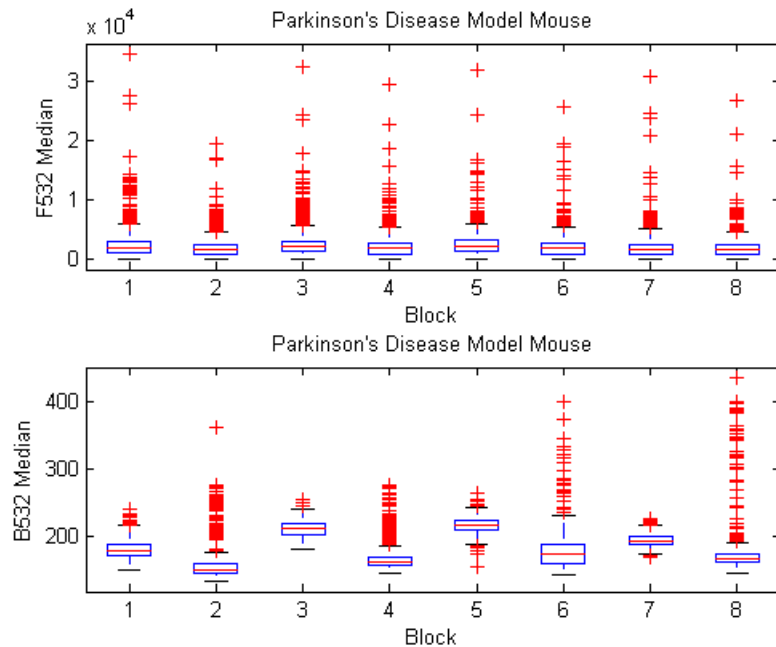
Statistics of the Microarrays

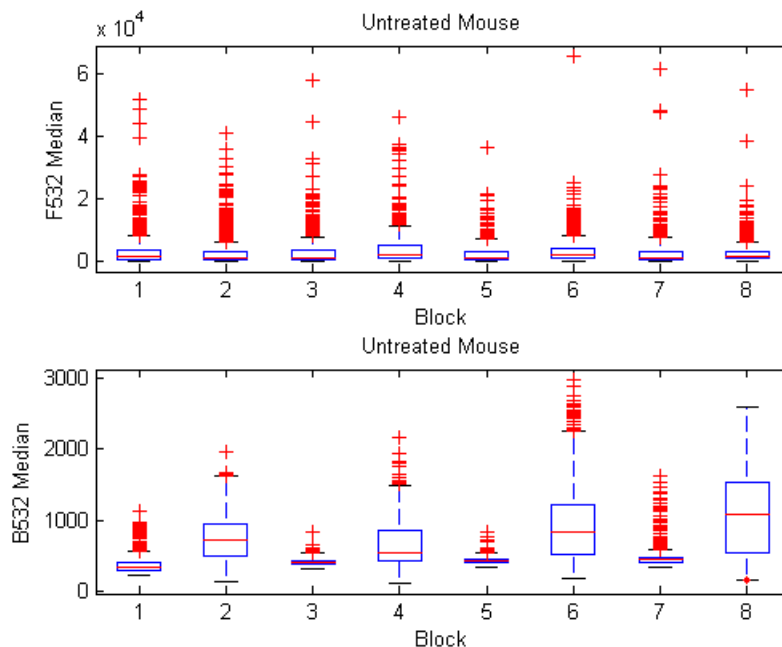
You can use the function `maboxplot` to look at the distribution of data in each of the blocks.

1 In the MATLAB Command Window, type

```
figure
subplot(2,1,1)
maboxplot(pd, 'F532 Median', 'title', 'Parkinson' 's Disease Model Mouse')
subplot(2,1,2)
maboxplot(pd, 'B532 Median', 'title', 'Parkinson' 's Disease Model Mouse')
figure
subplot(2,1,1)
maboxplot(wt, 'F532 Median', 'title', 'Untreated Mouse')
subplot(2,1,2)
maboxplot(wt, 'B532 Median', 'title', 'Untreated Mouse')
```

MATLAB plots the images.





2 Compare the plots.

From the box plots you can clearly see the spatial effects in the background intensities. Blocks numbers 1, 3, 5, and 7 are on the left side of the arrays, and numbers 2, 4, 6, and 8 are on the right side. The data must be normalized to remove this spatial bias.

Scatter Plots of Microarray Data

There are two columns in the microarray data structure labeled 'F635 Median - B635' and 'F532 Median - B532'. These columns are the differences between the median foreground and the median background for the 635 nm channel and 532 nm channel respectively. These give a measure of the actual expression levels, although since the data must first be normalized to remove spatial bias in the background, you should be careful about using these values without further normalization. However, in this example no normalization is performed.

- 1 Rather than working with data in a larger structure, it is often easier to extract the column numbers and data into separate variables.

```
cy5DataCol = find(strcmp(wt.ColumnNames,'F635 Median - B635'))
cy3DataCol = find(strcmp(wt.ColumnNames,'F532 Median - B532'))
cy5Data = pd.Data(:,cy5DataCol);
cy3Data = pd.Data(:,cy3DataCol);
```

MATLAB displays

```
cy5DataCol =
    34
```

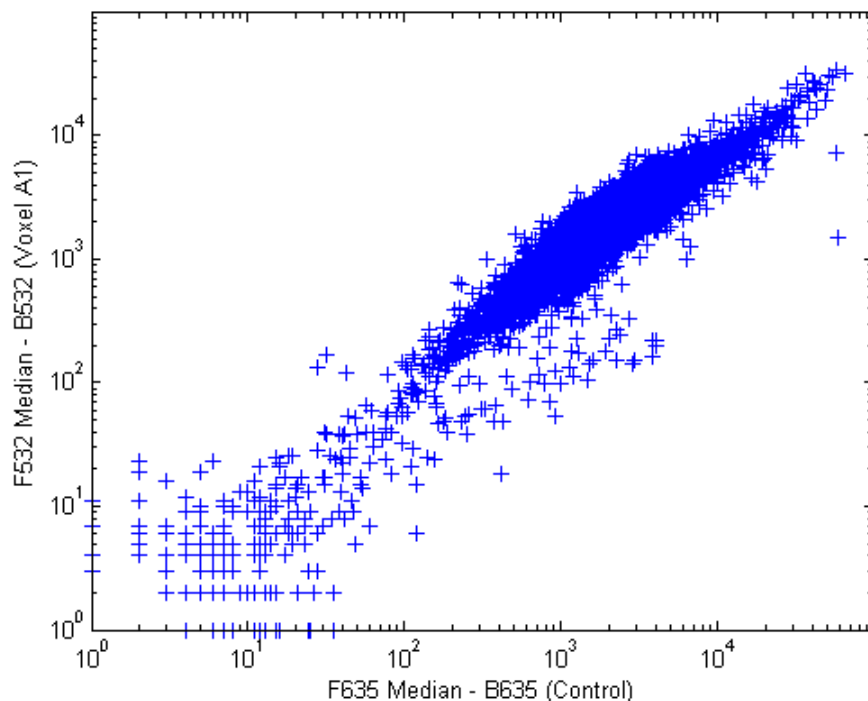
```
cy3DataCol =
    35
```

- 2 A simple way to compare the two channels is with a loglog plot. The function `maloglog` is used to do this. Points that are above the diagonal in this plot correspond to genes that have higher expression levels in the A1 voxel than in the brain as a whole.

```
figure
maloglog(cy5Data,cy3Data)
xlabel('F635 Median - B635 (Control)');
ylabel('F532 Median - B532 (Voxel A1)');
```

MATLAB displays the following messages and plots the images.

```
Warning: Zero values are ignored
(Type "warning off Bioinfo:MaloglogZeroValues" to suppress
this warning.)
Warning: Negative values are ignored.
(Type "warning off Bioinfo:MaloglogNegativeValues" to suppress
this warning.)
```



Notice that this function gives some warnings about negative and zero elements. This is because some of the values in the 'F635 Median - B635' and 'F532 Median - B532' columns are zero or even less than zero. Spots where this happened might be bad spots or spots that failed to hybridize. Points with positive, but very small, differences between foreground and background should also be considered to be bad spots.

- 3** Disable the display of warnings by using the warning command. Although warnings can be distracting, it is good practice to investigate why the warnings occurred rather than simply to ignore them. There might be some systematic reason why they are bad.

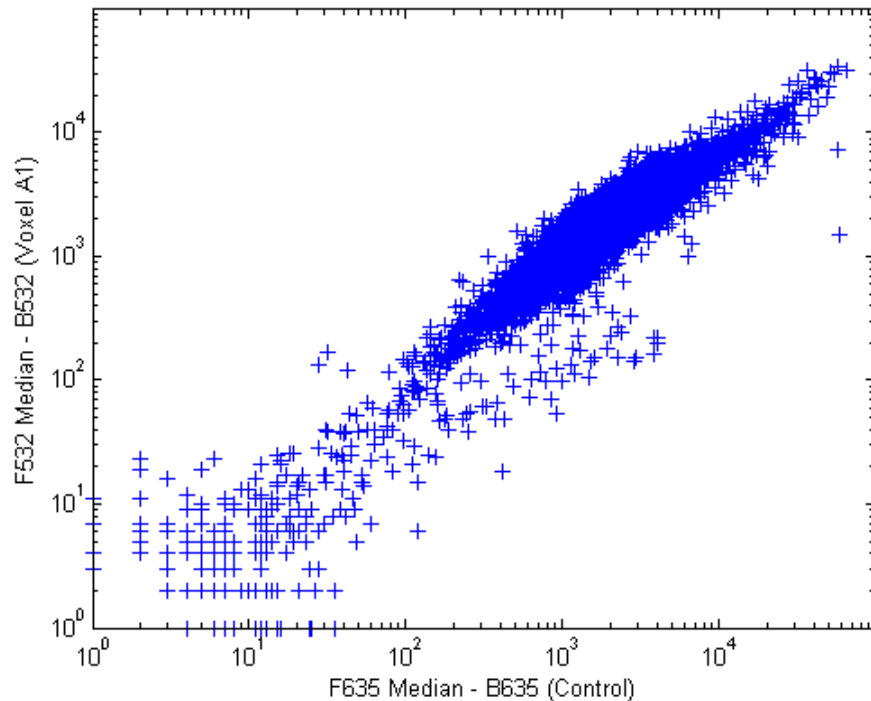
```
warnState = warning;           % First save the current warning
                               % state.
                               % Now turn off the two warnings.
warning('off','Bioinfo:MaloglogZeroValues');
warning('off','Bioinfo:MaloglogNegativeValues');
```

```

figure
maloglog(cy5Data,cy3Data)      % Create the loglog plot
warning(warnState);           % Reset the warning state.
xlabel('F635 Median - B635 (Control)');
ylabel('F532 Median - B532 (Voxel A1)');

```

MATLAB plots the image.



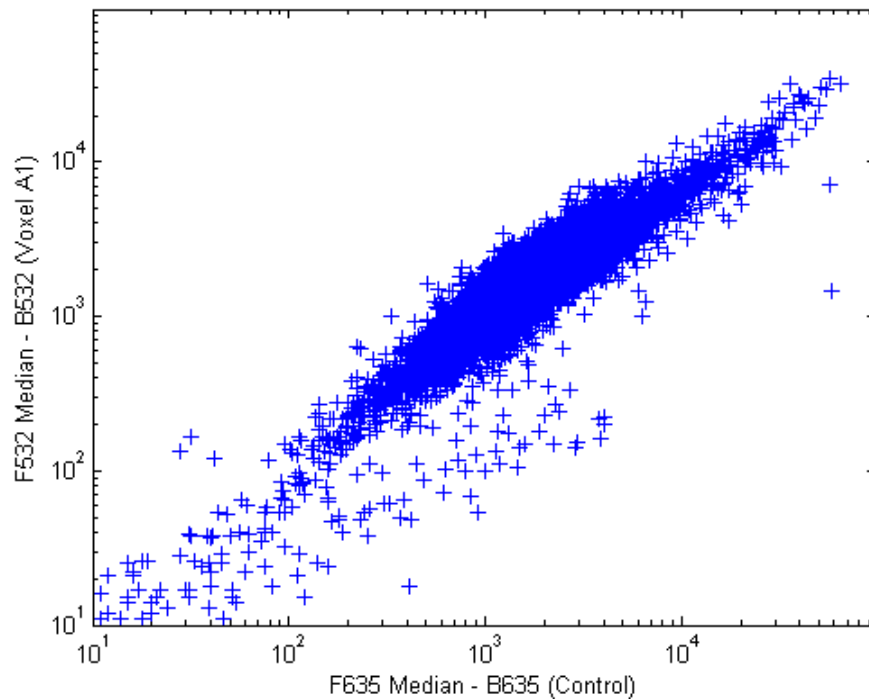
- 4** An alternative to simply ignoring or disabling the warnings is to remove the bad spots from the data set. You can do this by finding points where either the red or green channel has values less than or equal to a threshold value. For example, use a threshold value of 10.

```

threshold = 10;
badPoints = (cy5Data <= threshold) | (cy3Data <= threshold);

```

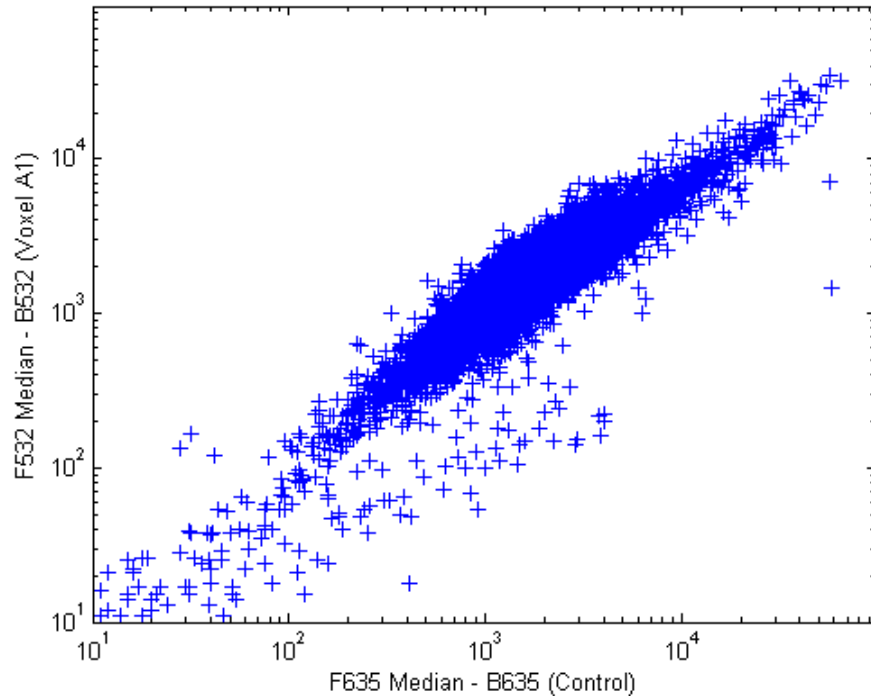
MATLAB plots the image.



5 You can then remove these points and redraw the loglog plot.

```
cy5Data(badPoints) = []; cy3Data(badPoints) = [];  
figure  
maloglog(cy5Data,cy3Data)  
xlabel('F635 Median - B635 (Control)');  
ylabel('F532 Median - B532 (Voxel A1)');
```

MATLAB plots the image.

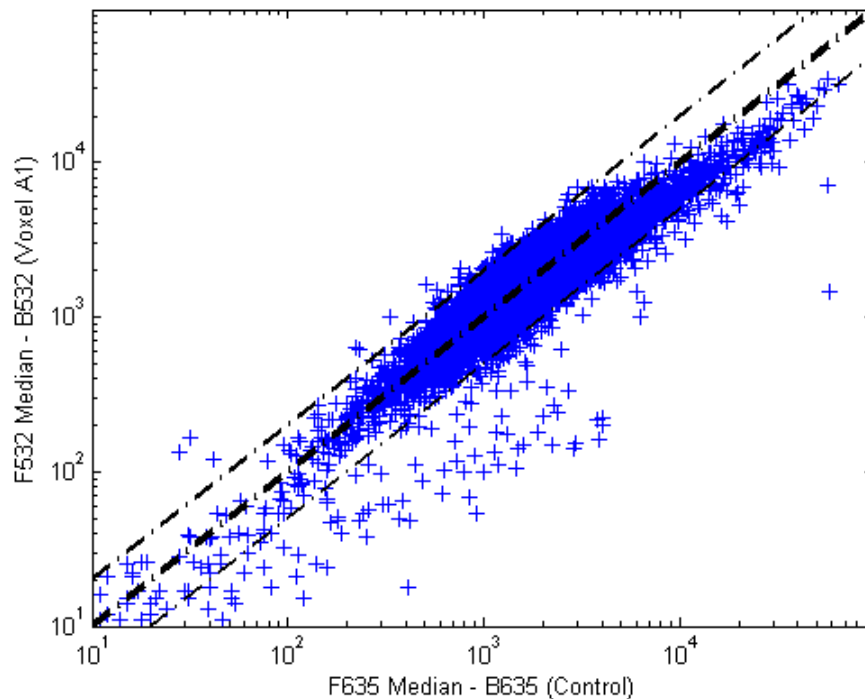


This plot shows the distribution of points but does not give any indication about which genes correspond to which points.

- 6 Add gene labels to the plot. Because some of the data points have been removed, the corresponding gene IDs must also be removed from the data set before you can use them. The simplest way to do that is `wt.IDs(~badPoints)`.

```
maloglog(cy5Data,cy3Data,'labels',wt.IDs(~badPoints),...
         'factorlines',2)
xlabel('F635 Median - B635 (Control)');
ylabel('F532 Median - B532 (Voxel A1)');
```

MATLAB plots the image.



7 Try using the mouse to click some of the outlier points.

You will see the gene ID associated with the point. Most of the outliers are below the $y = x$ line. In fact, most of the points are below this line. Ideally the points should be evenly distributed on either side of this line.

8 Normalize the points to evenly distribute them on either side of the line. Use the function `mameannorm` to perform global mean normalization.

```
normcy5 = mameannorm(cy5Data);
normcy3 = mameannorm(cy3Data);
```

If you plot the normalized data you will see that the points are more evenly distributed about the $y = x$ line.

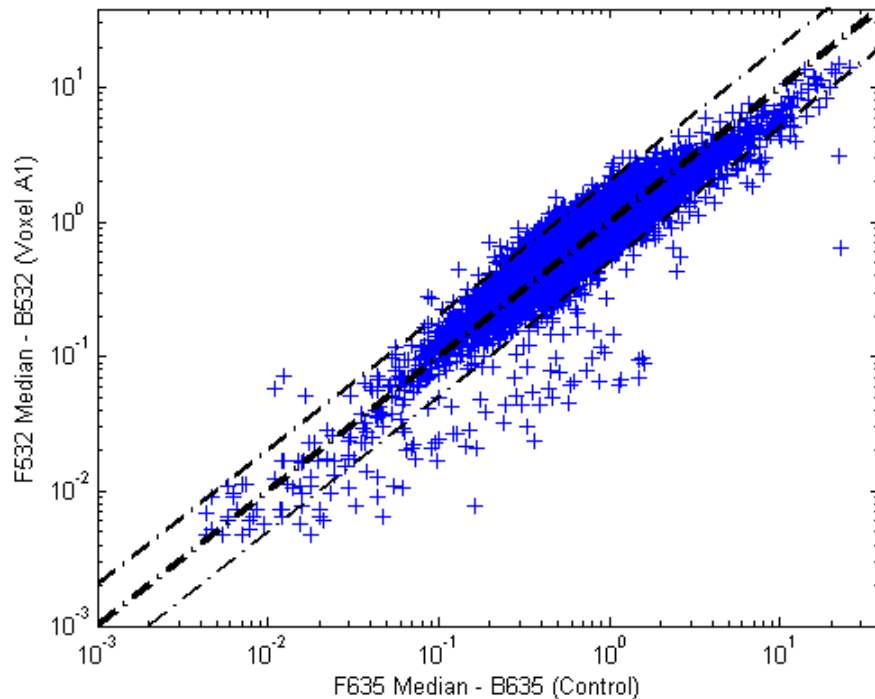
```
figure
```

```

maloglog(normcy5,normcy3,'labels',wt.IDs(~badPoints),...
         'factorlines',2)
xlabel('F635 Median - B635 (Control)');
ylabel('F532 Median - B532 (Voxel A1)');

```

MATLAB plots the image.



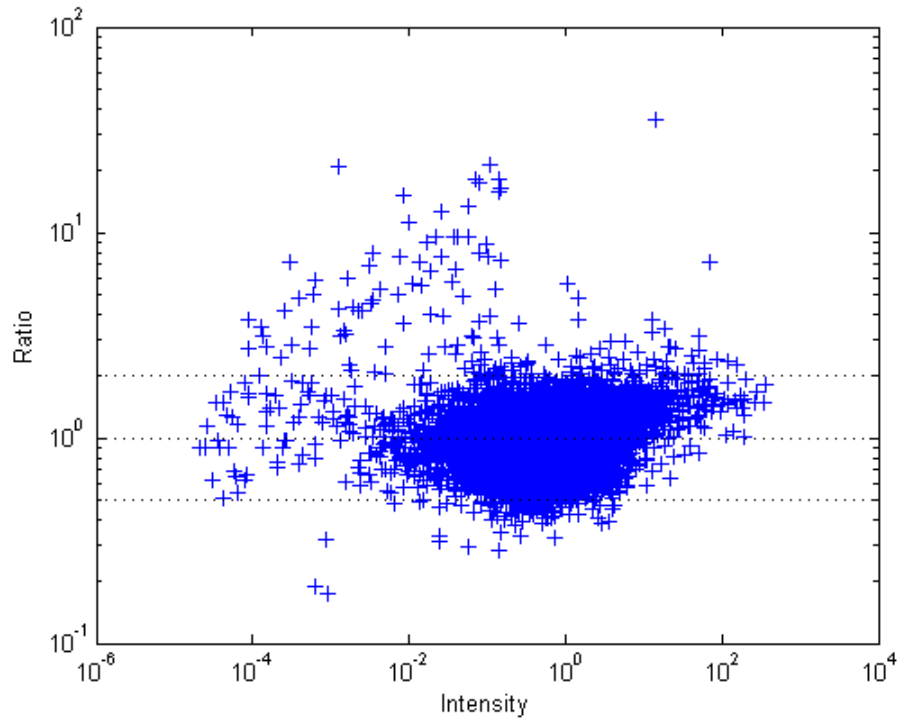
- 9 The function `mairplot` is used to create an Intensity vs. Ratio plot for the normalized data. This function works in the same way as the function `maloglog`.

```

figure
mairplot(normcy5,normcy3,'labels',wt.IDs(~badPoints),...
         'factorlines',2)

```

MATLAB plots the image.



10 You can click the points in this plot to see the name of the gene associated with the plot.

Example: Analyzing Gene Expression Profiles

This example demonstrates a number of ways to look for patterns in gene expression profiles.

Overview of the Yeast Example (p. 3-25)	Gene expression of yeast with shift from fermentation to respiration.
Exploring the Data Set (p. 3-25)	Import data from the Web into MATLAB.
Filtering Genes (p. 3-29)	Remove genes that are not expressed or do not change.
Clustering Genes (p. 3-32)	Identify relationships between genes using cluster techniques.
Principal Component Analysis (p. 3-36)	Reduce the dimensionality of large microarray data sets and look for signals in noisy data.

Overview of the Yeast Example

The microarray data for this example is from DeRisi, JL, Iyer, VR, and Brown, PO.; "Exploring the metabolic and genetic control of gene expression on a genomic scale"; Science, 1997, Oct 24;278(5338):680-6, PMID: 9381177.

The authors used DNA microarrays to study temporal gene expression of almost all genes in *Saccharomyces cerevisiae* during the metabolic shift from fermentation to respiration. Expression levels were measured at seven time points during the diauxic shift. The full data set can be downloaded from the Gene Expression Omnibus Web site at

<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE28>

Exploring the Data Set

The data for this procedure is available in the MAT-file `yeastdata.mat`. This file contains the `VALUE` data or `LOG_RAT2N_MEAN`, or \log_2 of ratio of `CH2DN_MEAN` and `CH1DN_MEAN` from the seven time steps in the experiment, the names of the genes, and an array of the times at which the expression levels were measured.

- 1** Load data into MATLAB.

```
load yeastdata.mat
```

- 2** Get the size of the data by typing

```
numel(genes)
```

MATLAB displays the number of genes in the data set. The MATLAB variable `genes` is a cell array of the gene names.

```
ans =  
    6400
```

- 3** Access the entries using MATLAB cell array indexing.

```
genes{15}
```

MATLAB displays the 15th row of the variable `yeastvalues`, which contains expression levels for the open reading frame (ORF) YAL054C.

```
ans =  
    YAL054C
```

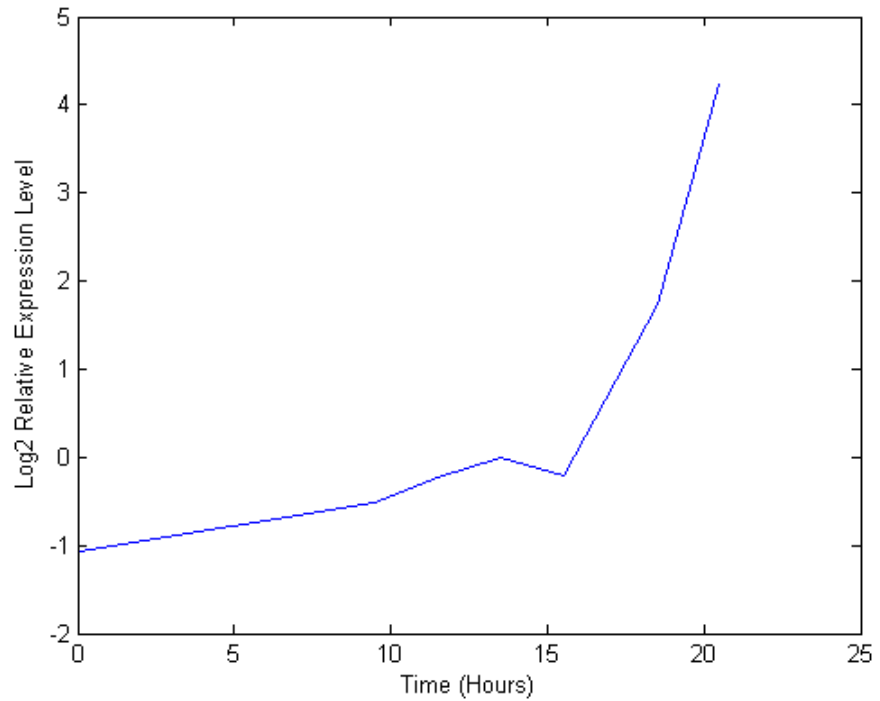
- 4** Use the function `web` to access information about this ORF in the Saccharomyces Genome Database (SGD).

```
url = sprintf(...  
    'http://genome-www4.stanford.edu/cgi-bin/SGD/...  
    locus.pl?locus=%s',...  
    genes{15});  
web(url);
```

- 5** A simple plot can be used to show the expression profile for this ORF.

```
plot(times, yeastvalues(15,:))  
xlabel('Time (Hours)');  
ylabel('Log2 Relative Expression Level');
```

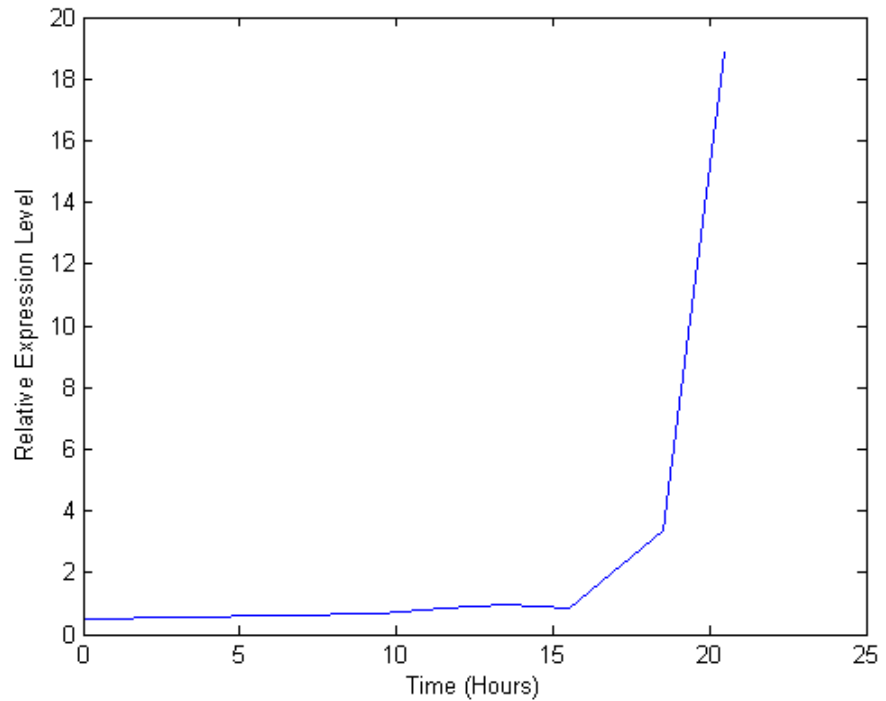
MATLAB plots the figure. The values are log2 ratios.



6 Plot the actual values.

```
plot(times, 2.^yeastvalues(15,:))  
xlabel('Time (Hours)');  
ylabel('Relative Expression Level');
```

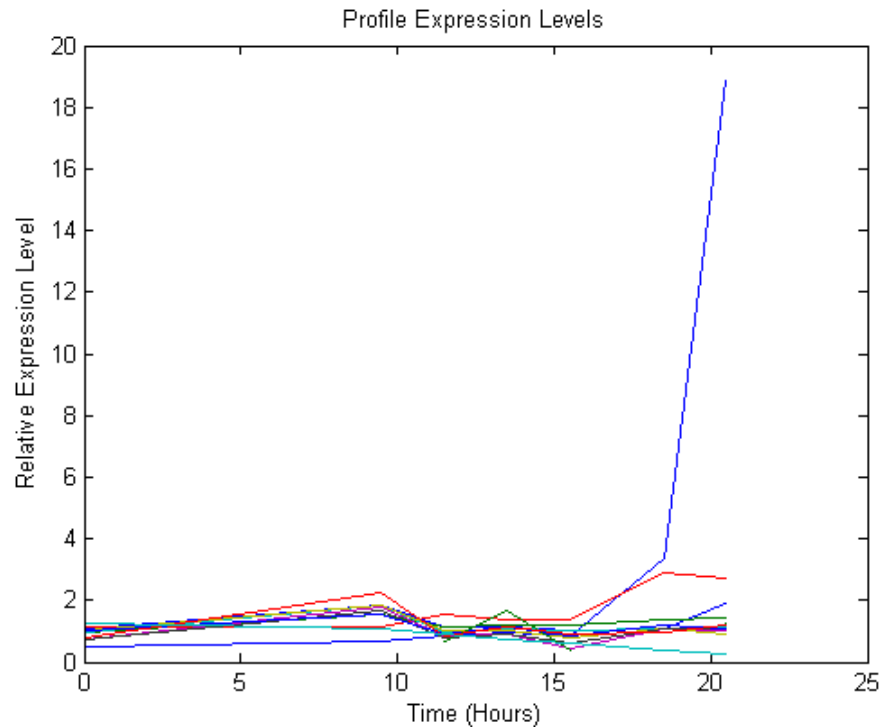
MATLAB plots the figure. The gene associated with this ORF, ACS1, appears to be strongly up-regulated during the diauxic shift.



7 Compare other genes by plotting multiple lines on the same figure.

```
hold on
plot(times, 2.^yeastvalues(16:26,:))
xlabel('Time (Hours)');
ylabel('Relative Expression Level');
title('Profile Expression Levels');
```

MATLAB plots the image.



Filtering Genes

The data set is quite large and a lot of the information corresponds to genes that do not show any interesting changes during the experiment. To make it easier to find the interesting genes, reduce the size of the data set by removing genes with expression profiles that do not show anything of interest. There are 6400 expression profiles. You can use a number of techniques to reduce the number of expression profiles to some subset that contains the most significant genes.

- 1 If you look through the gene list you will see several spots marked as 'EMPTY'. These are empty spots on the array, and while they might have data associated with them, for the purposes of this example, you can consider these points to be noise. These points can be found using the `strcmp` function and removed from the data set with indexing commands..

```
emptySpots = strcmp('EMPTY',genes);  
yeastvalues(emptySpots,:) = [];  
genes(emptySpots) = [];  
numel(genes)
```

MATLAB displays

```
ans =  
    6314
```

In the `yeastvalues` data you will also see several places where the expression level is marked as `NaN`. This indicates that no data was collected for this spot at the particular time step. One approach to dealing with these missing values would be to impute them using the mean or median of data for the particular gene over time. This example uses a less rigorous approach of simply throwing away the data for any genes where one or more expression levels were not measured.

- 2** Use function `isnan` to identify the genes with missing data and then use indexing commands to remove the genes.

```
nanIndices = any(isnan(yeastvalues),2);  
yeastvalues(nanIndices,:) = [];  
genes(nanIndices) = [];  
numel(genes)
```

MATLAB displays

```
ans =  
    6276
```

If you were to plot the expression profiles of all the remaining profiles, you would see that most profiles are flat and not significantly different from the others. This flat data is obviously of use as it indicates that the genes associated with these profiles are not significantly affected by the diauxic shift. However, in this example, you are interested in the genes with large changes in expression accompanying the diauxic shift. You can use filtering functions in the Bioinformatics Toolbox to remove genes with various types of profiles that do not provide useful information about genes affected by the metabolic change.

- 3** Use the function `genevarfilter` to filter out genes with small variance over time. The function returns a logical array of the same size as the variable genes with ones corresponding to rows of `yeastvalues` with variance greater than the 10th percentile and zeros corresponding to those below the threshold.

```
mask = genevarfilter(yeastvalues);
% Use the mask as an index into the values to remove the
% filtered genes.
yeastvalues = yeastvalues(mask,:);
genes = genes(mask);
numel(genes)
```

MATLAB displays

```
ans =
    5648
```

- 4** The function `genelowvalfilter` removes genes that have very low absolute expression values. Note that the gene filter functions can also automatically calculate the filtered data and names.

```
[mask, yeastvalues, genes] = genelowvalfilter(yeastvalues,genes,...
                                              'absval',log2(4));
numel(genes)
```

MATLAB displays

```
ans =
    423
```

- 5** Use the function `geneentropyfilter` to remove genes whose profiles have low entropy:

```
[mask, yeastvalues, genes] = geneentropyfilter(yeastvalues,genes,...
                                              'prctile',15);
numel(genes)
```

MATLAB displays

```
ans = 310
```

Clustering Genes

Now that you have a manageable list of genes, you can look for relationships between the profiles using some different clustering techniques from the Statistics Toolbox.

- 1** For hierarchical clustering, the function `pdist` calculates the pairwise distances between profiles, and the function `linkage` creates the hierarchical cluster tree.

```
corrDist = pdist(yeastvalues, 'corr');  
clusterTree = linkage(corrDist, 'average');
```

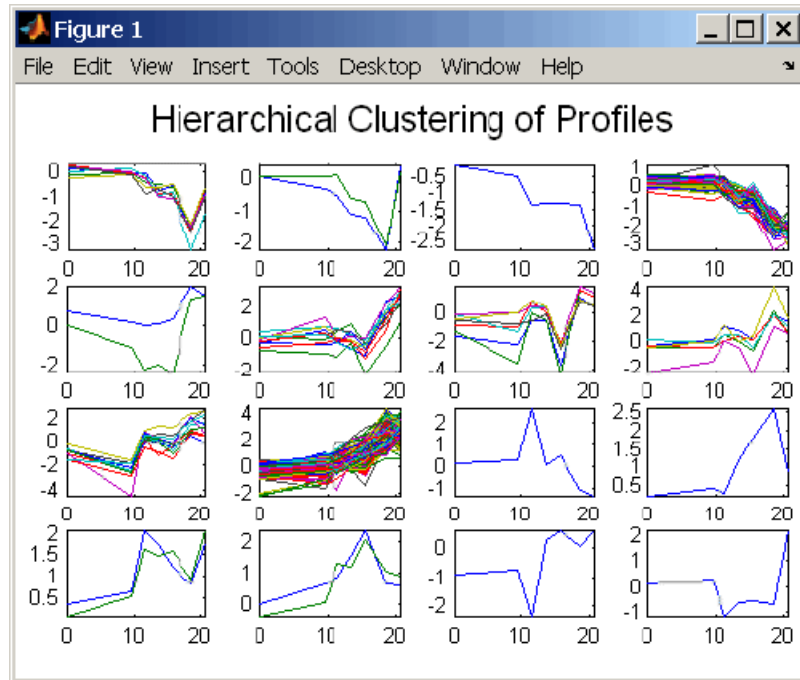
- 2** The function `cluster` calculates the clusters based on either a cutoff distance or a maximum number of clusters. In this case, the `'maxclust'` option is used to identify 16 distinct clusters.

```
clusters = cluster(clusterTree, 'maxclust', 16);
```

- 3** The profiles of the genes in these clusters can be plotted together using a simple loop and the function `subplot`.

```
figure  
for c = 1:16  
    subplot(4,4,c);  
    plot(times,yeastvalues((clusters == c),:));  
    axis tight  
end  
suptitle('Hierarchical Clustering of Profiles');
```

MATLAB plots the images.



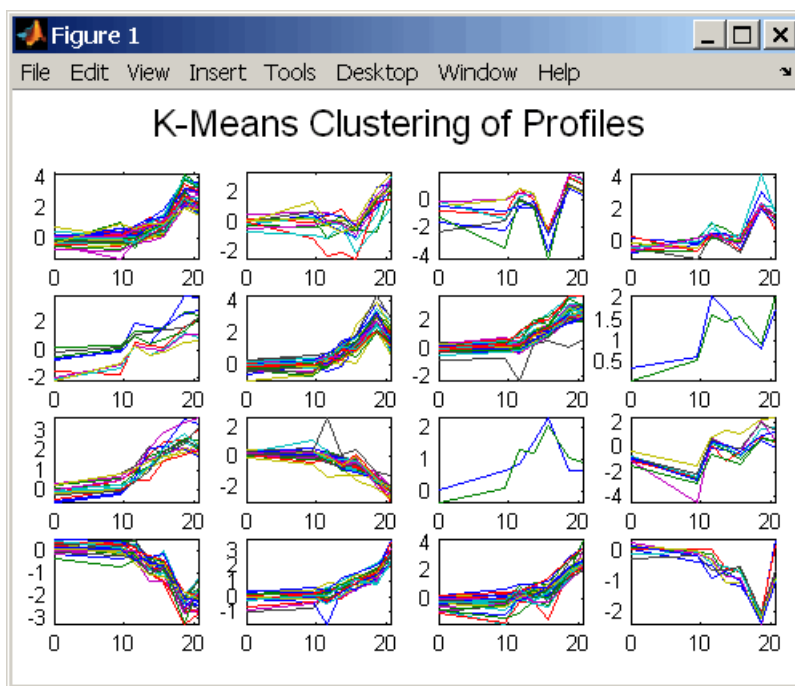
- 4 The Statistics Toolbox also has a K-means clustering function. Again, sixteen clusters are found, but because the algorithm is different these are not necessarily the same clusters as those found by hierarchical clustering.

```
[cidx, ctrs] = kmeans(yeastvalues, 16, ...
                    'dist','corr',...
                    'rep',5,...
                    'disp','final');

figure
for c = 1:16
    subplot(4,4,c);
    plot(times,yeastvalues((cidx == c),:));
    axis tight
end
suptitle('K-Means Clustering of Profiles');
```

MATLAB displays

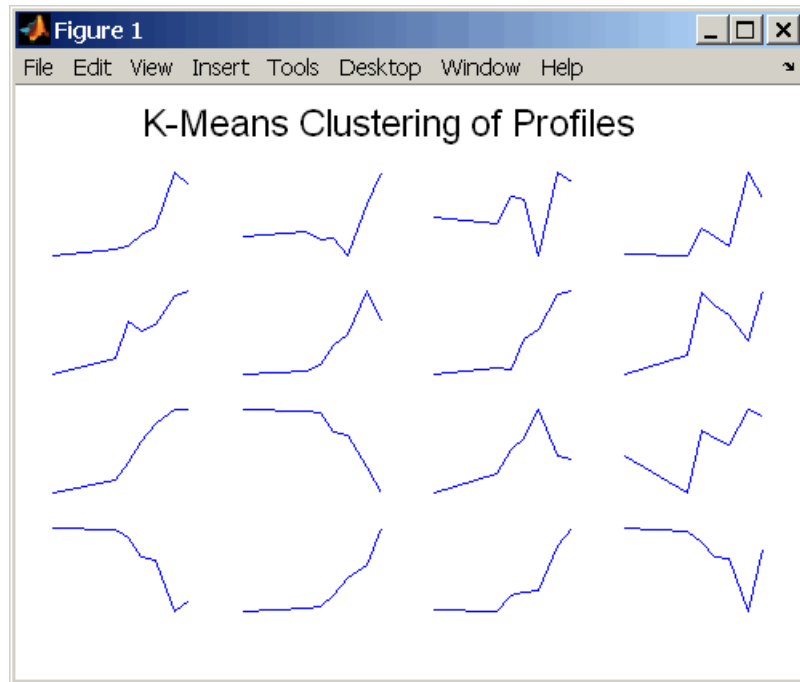
```
13 iterations, total sum of distances = 11.4042
14 iterations, total sum of distances = 8.62674
26 iterations, total sum of distances = 8.86066
22 iterations, total sum of distances = 9.77676
26 iterations, total sum of distances = 9.01035
```



5 Instead of plotting all of the profiles, you can plot just the centroids.

```
figure
for c = 1:16
    subplot(4,4,c);
    plot(times,ctrs(c,:));
    axis tight
    axis off    % turn off the axis
end
suptitle('K-Means Clustering of Profiles');
```

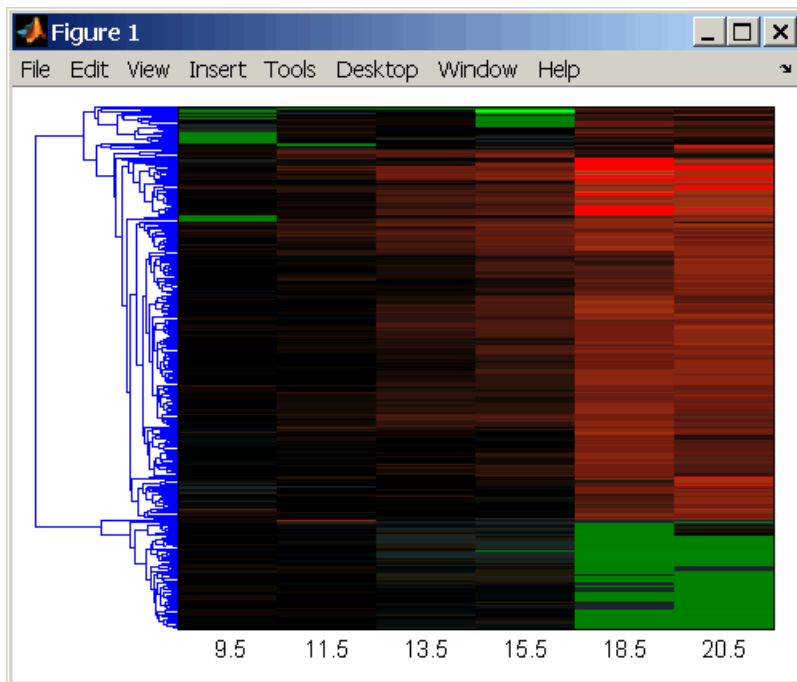
MATLAB plots the figure.



- 6 You can use the function `clustergram` to create a heat map and dendrogram from the output of the hierarchical clustering.

```
figure
clustergram(yeastvalues(:,2:end), 'RowLabels', genes, ...
            'ColumnLabels', times(2:end))
```

MATLAB plots the figure.



Principal Component Analysis

Principal-component analysis (PCA) is a useful technique you can use to reduce the dimensionality of large data sets, such as those from microarray analysis. You can also use PCA to find signals in noisy data.

- 1 Use the function `princomp` in the Statistics Toolbox to calculate the principal components of a data set.

```
[pc, zscores, pcvars] = princomp(yeastvalues)
```

MATLAB displays

```
pc =
```

```
Columns 1 through 4
```

```

-0.0245  -0.3033  -0.1710  -0.2831
 0.0186  -0.5309  -0.3843  -0.5419
 0.0713  -0.1970   0.2493   0.4042
 0.2254  -0.2941   0.1667   0.1705
 0.2950  -0.6422   0.1415   0.3358
 0.6596   0.1788   0.5155  -0.5032
 0.6490   0.2377  -0.6689   0.2601

```

Columns 5 through 7

```

-0.1155   0.4034   0.7887
-0.2384  -0.2903  -0.3679
-0.7452  -0.3657   0.2035
-0.2385   0.7520  -0.4283
 0.5592  -0.2110   0.1032
-0.0194  -0.0961   0.0667
-0.0673  -0.0039   0.0521

```

- 2** You can use the function `cumsum` to see the cumulative sum of the variances.

```
cumsum(pcvvars ./ sum(pcvvars) * 100)
```

MATLAB displays

```

ans =
 78.3719
 89.2140
 93.4357
 96.0831
 98.3283
 99.3203
100.0000

```

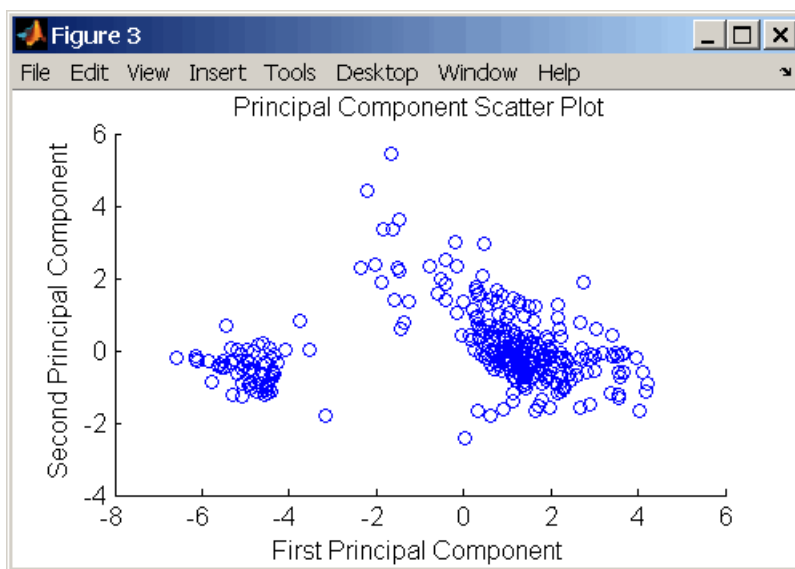
This shows that almost 90% of the variance is accounted for by the first two principal components.

- 3** A scatter plot of the scores of the first two principal components shows that there are two distinct regions. This is not unexpected, because the filtering

process removed many of the genes with low variance or low information. These genes would have appeared in the middle of the scatter plot.

```
figure
scatter(zscores(:,1),zscores(:,2));
xlabel('First Principal Component');
ylabel('Second Principal Component');
title('Principal Component Scatter Plot');
```

MATLAB plots the figure.



- 4 The function `gname` from the Statistics Toolbox can be used to identify genes on a scatter plot. You can select as many points as you like on the scatter plot.

```
gname(genes);
```

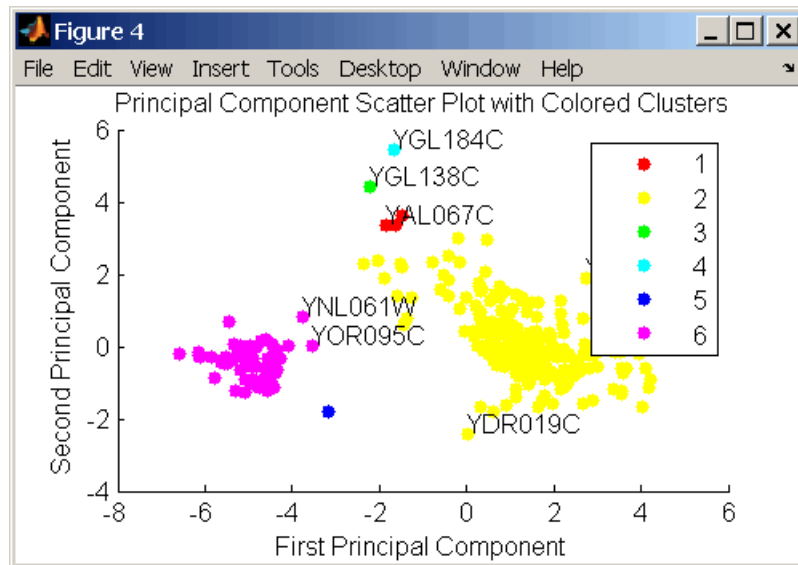
When you have finished selecting points, press **Enter**.

- 5 An alternative way to create a scatter plot is with the function `gscatter` from the Statistics Toolbox. `gscatter` creates a grouped scatter plot where

points from each group have a different color or marker. You can use `clusterdata`, or any other clustering function, to group the points.

```
figure
pcclusters = clusterdata(zscores(:,1:2),6);
gscatter(zscores(:,1),zscores(:,2),pcclusters)
xlabel('First Principal Component');
ylabel('Second Principal Component');
title('Principal Component Scatter Plot with Colored Clusters');
gname(genes) % Press enter when you finish selecting genes.
```

MATLAB plots the figure.



Phylogenetic Analysis

Phylogenetic analysis is the process you use to determine the evolutionary relationships between organisms. The results of an analysis can be drawn in a hierarchical diagram called a cladogram or phylogram (phylogenetic tree). The branches in a tree are based on the hypothesized evolutionary relationships (phylogeny) between organisms. Each member in a branch, also known as a monophyletic group, is assumed to be descended from a common ancestor. Originally, phylogenetic trees were created using morphology, but now, determining evolutionary relationships includes matching patterns in nucleic acid and protein sequences.

Example: Building a Phylogenetic Tree (p. 4-2)

Using data from mitochondrial D-loop sequences, create a phylogenetic tree for a family of primates.

Phylogenetic Tree Tool Reference (p. 4-14)

Description of menu commands and features for creating publishable tree figures.

Example: Building a Phylogenetic Tree

In this example, a phylogenetic tree is constructed from mitochondrial DNA (mtDNA) sequences for the family Hominidae. This family includes gorillas, chimpanzees, orangutans, and humans.

The following procedures demonstrate the phylogenetic analysis features in the Bioinformatics Toolbox. They are not intended to teach the process of phylogenetic analysis, but to show you how to use MathWorks products to create a phylogenetic tree from a set of nonaligned nucleotide sequences.

- “Overview for the Primate Example” on page 4-2 — Describes the biological background for this example.
- “Creating a Phylogenetic Tree for Five Species” on page 4-6 — Use the Jukes-Cantor method to calculate distances between sequences, and the Unweighted Pair Group Method Average (UPGMA) method for linking the tree nodes.
- “Creating a Phylogenetic Tree for Twelve Species” on page 4-8 — Add additional organisms to confirm the observed monophyletic groups.
- “Exploring the Phylogenetic Tree” on page 4-10 — Use the MATLAB command-line interface to programmatically determine characteristics in a phylogenetic tree.

For information on how to create a phylogenetic tree with multiply aligned sequences, see the function `phytree`.

Overview for the Primate Example

The origin of modern humans is a heavily debated issue that scientists have recently tackled by using mitochondrial DNA (mtDNA) sequences. One hypothesis explains the limited genetic variation of human mtDNA in terms of a recent common genetic ancestry, implying that all modern population mtDNA originated from a single woman who lived in Africa less than 200,000 years ago.

Why Use Mitochondrial DNA Sequences for Phylogenetic Study?

Mitochondrial DNA sequences, like the Y chromosome, do not recombine and are inherited from the maternal parent. This lack of recombination allows sequences to be traced through one genetic line and all polymorphisms assumed to be caused by mutations.

Mitochondrial DNA in mammals has a faster mutation rate than nuclear DNA sequences. This faster rate of mutation produces more variance between sequences and is an advantage when studying closely related species. The mitochondrial control region (Displacement or D-loop) is one of the fastest mutating sequence regions in animal DNA.

Neanderthal DNA

The ability to isolate mitochondrial DNA (mtDNA) from palaeontological samples has allowed genetic comparisons between extinct species and closely related nonextinct species. The reasons for isolating mtDNA instead of nuclear DNA in fossil samples have to do with the fact that

- mtDNA, because it is circular, is more stable and degrades slower than nuclear DNA.
- Each cell can contain a thousand copies of mtDNA and only a single copy of nuclear DNA.

While there is still controversy as to whether Neanderthals are direct ancestors of humans or evolved independently, the use of ancient genetic sequences in phylogenetic analysis adds an interesting dimension to the question of human ancestry.

References

Ovchinnikov I., et al. (2000). Molecular analysis of Neanderthal DNA from the northern Caucasus. *Nature* 404(6777), 490–493.

Sajantila A., et al. (1995). Genes and languages in Europe: an analysis of mitochondrial lineages. *Genome Research* 5 (1), 42–52.

Krings M., et al. (1997). Neanderthal DNA sequences and the origin of modern humans. *Cell* 90 (1), 19–30.

Jensen-Seaman, M., Kidd K. (2001). Mitochondrial DNA variation and biogeography of eastern gorillas. *Molecular Ecology* 10(9), 2241–2247.

Searching NCBI for Phylogenetic Data

The NCBI taxonomy Web site includes phylogenetic and taxonomic information from many sources. These sources include the published literature, Web databases, and taxonomy experts. And while the NCBI taxonomy database is not a phylogenetic or taxonomic authority, it can be useful as a gateway to the NCBI biological sequence databases.

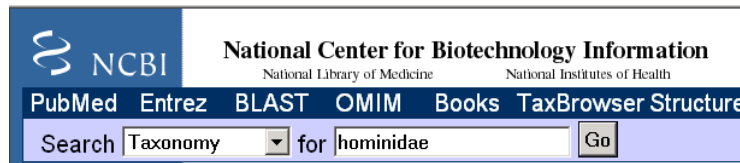
This procedure uses the family Hominidae (orangutans, chimpanzees, gorillas, and humans) as a taxonomy example for searching the NCBI Web site and locating mitochondrial D-loop sequences.

- 1 Use the MATLAB Help browser to search for data on the Web. In the MATLAB Command Window, type

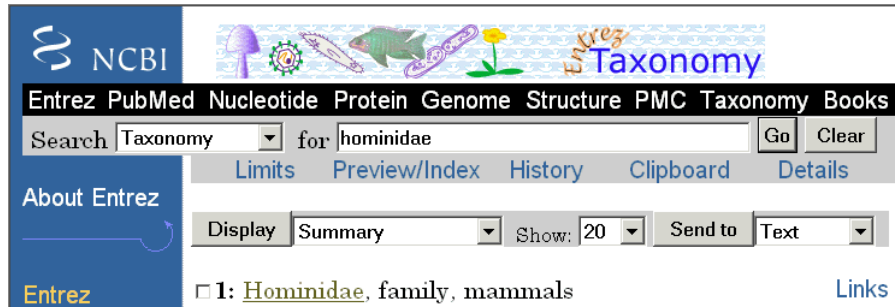
```
web('http://www.ncbi.nlm.nih.gov')
```

A separate browser window opens with the home page for the NCBI Web site.

- 2 Search the NCBI Web site for information. For example, to search for the human taxonomy, from the **Search** list, select Taxonomy, and in the **for** box, enter hominidae.



The NCBI Web search returns a list of links to relevant pages.



The screenshot shows the NCBI Entrez Taxonomy search interface. At the top, there is a navigation bar with links for Entrez, PubMed, Nucleotide, Protein, Genome, Structure, PMC, Taxonomy, and Books. Below this is a search bar with 'Taxonomy' selected and 'hominidae' entered. The search results are displayed in a table with columns for 'Display', 'Summary', 'Show', and 'Send to'. The first result is '1: [Hominidae](#), family, mammals'. There is also a 'Links' button on the right side of the result.

- 3 Select the taxonomy link for the family Hominidae. A page with the taxonomy for the family is shown.

o [Hominidae](#) *Click on organism name to get more information.*

o [Homo/Pan/Gorilla group](#)

o [Gorilla](#)

- [Gorilla gorilla](#) (gorilla)

o [Homo](#)

- [Homo sapiens](#) (human)

o [Pan](#) (chimpanzees)

- [Pan paniscus](#) (pygmy chimpanzee)
- [Pan troglodytes](#) (chimpanzee)

o [Pongo](#)

o [Pongo pygmaeus](#) (orangutan)

- [Pongo pygmaeus abelii](#) (Sumatran orangutan)
- [Pongo pygmaeus pygmaeus](#) (Bornean orangutan)

▪ [Pongo sp.](#)

Creating a Phylogenetic Tree for Five Species

Drawing a phylogenetic tree using sequence data is helpful when you are trying to visualize the evolutionary relationships between species. The sequences can be multiply aligned or a set of nonaligned sequences, you can select a method for calculating pairwise distances between sequences, and you can select a method for calculating the hierarchical clustering distances used to build a tree.

After locating the GenBank accession codes for the sequences you are interested in studying, you can create a phylogenetic tree with the data. For information on locating accession codes, see “Searching NCBI for Phylogenetic Data” on page 4-4.

- 1** Create a MATLAB structure with information about the sequences. This step uses the accession codes for the mitochondrial D-loop sequences isolated from different hominid species.

```
data = {'German_Neanderthal'      'AF011222';
        'Russian_Neanderthal'    'AF254446';
        'European_Human'        'X90314';
        'Mountain_Gorilla_Rwanda' 'AF089820';
        'Chimp_Troglodytes'     'AF176766';
        };
```

- 2** Get sequence data from the GenBank database and copy into MATLAB.

```
for ind = 1:5
    seqs(ind).Header = data{ind,1};
    seqs(ind).Sequence = getgenbank(data{ind,2},
                                    'sequenceonly', true);
end
```

- 3** Calculate pairwise distances and create a phytree object. For example, compute the pairwise distances using the Jukes-Cantor distance method and build a phylogenetic tree using the UPGMA linkage method. Since the sequences are not prealigned, `seqpdist` pairwise aligns them before computing the distances.

```
distances = seqpdist(seqs, 'Method', 'Jukes-Cantor', 'Alphabet', 'DNA');
tree = seqlinkage(distances, 'UPGMA', seqs)
```

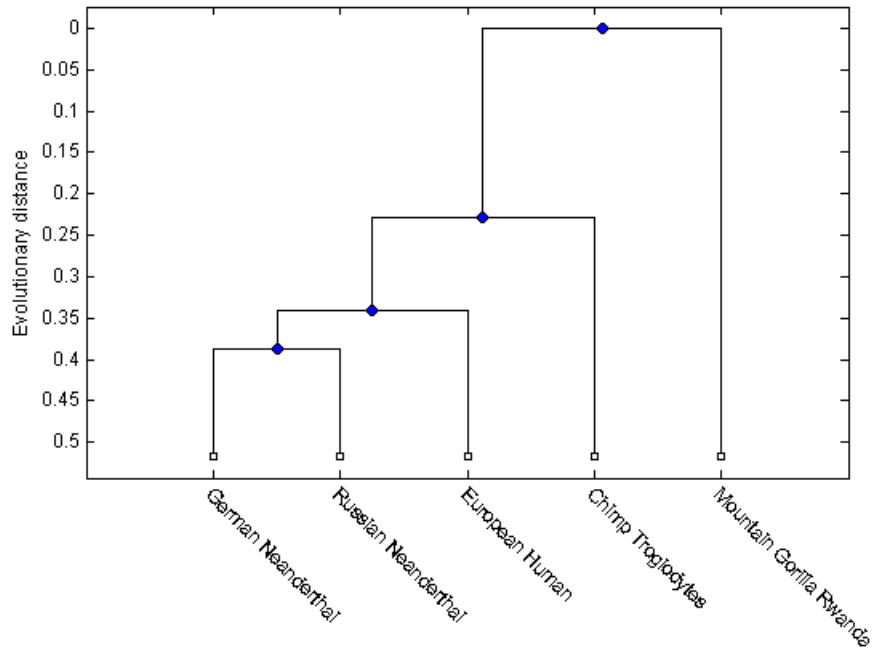
MATLAB displays information about the phytree object. The function `seqpdist` calculates the pairwise distances between pairs of sequences while the function `seqlinkage` uses the distances to build a hierarchical cluster tree. First, the most similar sequences are grouped together, and then sequences are added to the tree in descending order of similarity.

Phylogenetic tree object with 5 leaves (4 branches)

4 Draw a phylogenetic tree.

```
h = plot(tree,'orient','bottom');
ylabel('Evolutionary distance')
set(h.terminalNodeLabels,'Rotation',-45)
```

MATLAB draws a phylogenetic tree in a figure window. In the figure below, the hypothesized evolutionary relationships between the species is shown by the location of species on the branches. The horizontal distances do not have any biological significance.



Creating a Phylogenetic Tree for Twelve Species

Plotting a simple phylogenetic tree for five species seems to indicate a number of monophyletic groups (see “Creating a Phylogenetic Tree for Five Species” on page 4-6). After a preliminary analysis with five species, you can add more species to your phylogenetic tree. Adding more species to the data set will help you to confirm the groups are valid.

- 1 Add more sequences to a MATLAB structure. For example, add mtDNA D-loop sequences for other hominid species.

```
data2 = {'Puti_Orangutan'      'AF451972';
        'Jari_Orangutan'     'AF451964';
        'Western_Lowland_Gorilla' 'AY079510';
        'Eastern_Lowland_Gorilla' 'AF050738';
        'Chimp_Schweinfurthii'  'AF176722';
        'Chimp_Vellerosus'     'AF315498';
        'Chimp_Verus'         'AF176731';
        };
```

- 2 Get additional sequence data from the GenBank database, and copy the data into the next indices of a MATLAB structure.

```
for ind = 1:7
    seqs(ind+5).Header = data2{ind,1};
    seqs(ind+5).Sequence = getgenbank(data2{ind,2},
                                     'sequenceonly', true);
end
```

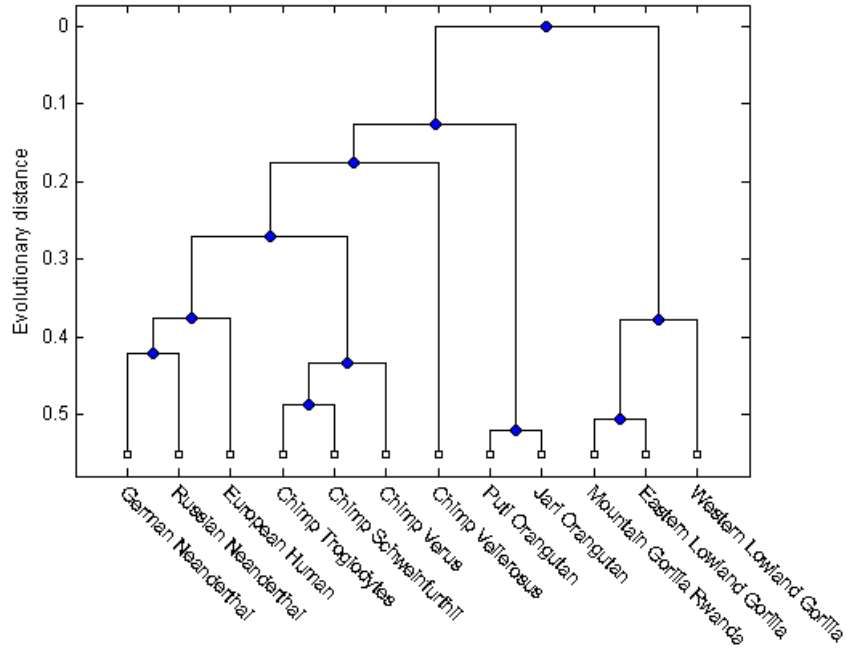
- 3 Calculate pairwise distances and the hierarchical linkage.

```
distances = seqpdist(seqs, 'Method', 'Jukes-Cantor', 'Alpha', 'DNA');
tree = seqlinkage(distances, 'UPGMA', seqs);
```

- 4 Draw a phylogenetic tree.

```
h = plot(tree, 'orient', 'bottom');
ylabel('Evolutionary distance')
set(h.terminalNodeLabels, 'Rotation', -45)
```

MATLAB draws a phylogenetic tree in a figure window. You can see four main clades for humans, gorillas, chimpanzee, and orangutans.



Exploring the Phylogenetic Tree

After you create a phylogenetic tree, you can explore the tree using the MATLAB command line or the phytreetool GUI. This procedure uses the tree created in “Creating a Phylogenetic Tree for Twelve Species” on page 4-8 as an example.

- 1 List the members of a tree.

```
names = get(tree,'LeafNames')
```

From the list, you can determine the indices for its members. For example, the European Human leaf is the third entry.

```
names =  
  
    'German_Neanderthal'  
    'Russian_Neanderthal'  
    'European_Human'
```

```
'Chimp_Troglodytes'
'Chimp_Schweinfurthii'
'Chimp_Verus'
'Chimp_Vellerosus'
'Puti_Orangutan'
'Jari_Orangutan'
'Mountain_Gorilla_Rwanda'
'Eastern_Lowland_Gorilla'
'Western_Lowland_Gorilla'
```

- 2** Find the closest species to a selected species in a tree. For example, find the species closest to the European human.

```
[h_all,h_leaves] = select(tree,'reference',3,
                          'criteria','distance',
                          'threshold',0.6);
```

`h_all` is a list of indices for the nodes within a patristic distance of 0.6 to the European human leaf, while `h_leaves` is a list of indices for only the leaf nodes within the same patristic distance.

A patristic distance is the path length between species calculated from the hierarchical clustering distances. The path distance is not necessarily the biological distance.

- 3** List the names of the closest species.

```
subtree_names = names(h_leaves)
```

MATLAB prints a list of species with a patristic distance to the European human less than the specified distance. In this case, the patristic distance threshold is less than 0.6.

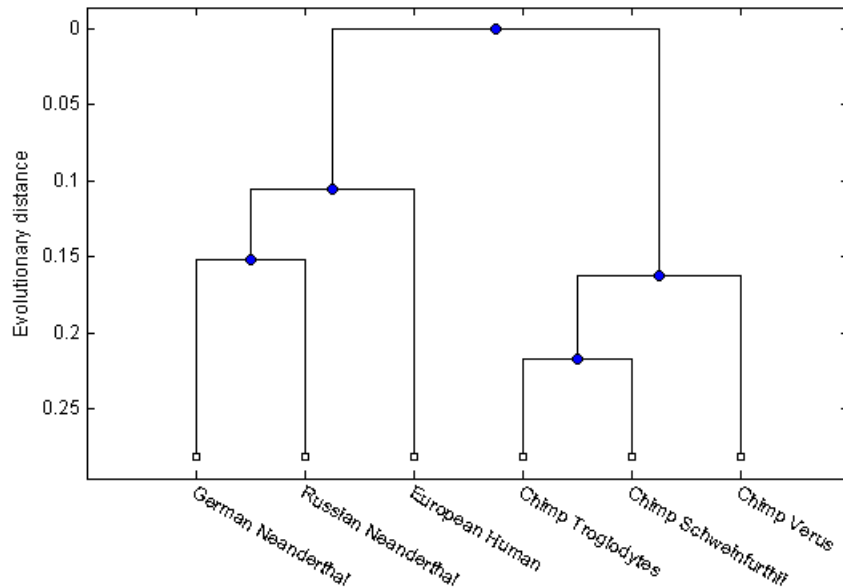
```
subtree_names =
'German_Neanderthal'
'Russian_Neanderthal'
'European_Human'
'Chimp_Schweinfurthii'
'Chimp_Verus'
'Chimp_Troglodytes'
```

- 4** Extract a subtree from the whole tree by removing unwanted leaves. For example, prune the tree to species within 0.6 of the European human species.

```
leaves_to_prune = ~h_leaves;  
pruned_tree = prune(tree,leaves_to_prune)  
h = plot(pruned_tree,'orient','bottom');  
ylabel('Evolutionary distance')  
set(h.terminalNodeLabels,'Rotation',-30)
```

MATLAB returns information about the new subtree and plots the pruned phylogenetic tree in a figure window.

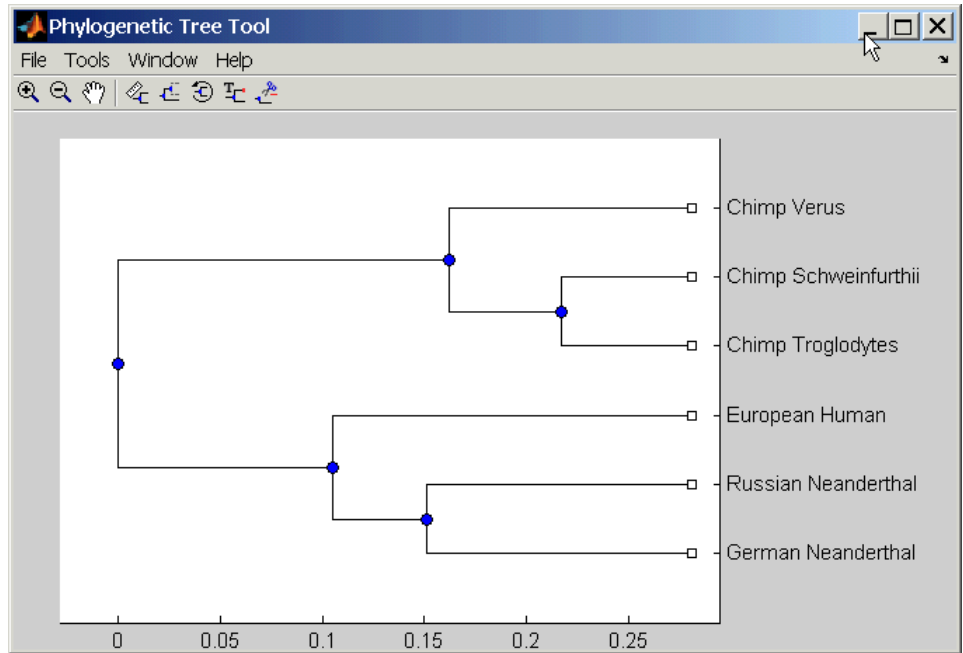
Phylogenetic tree object with 6 leaves (5 branches)



- 5** Explore, edit, and format a phylogenetic tree using an interactive GUI.

```
phytreetool(pruned_tree)
```

MATLAB opens the Phylogenetic Tree Tool window and draws the tree.



You can interactively change the appearance of the tree within the tool window. For information on using this GUI, see “Phylogenetic Tree Tool Reference” on page 4-14.

Phylogenetic Tree Tool Reference

The Phylogenetic Tree Tool is an interactive graphical user interface (GUI) that allows you to view, edit, format, and explore phylogenetic tree data. With this GUI you can prune, reorder, rename branches, and explore distances. You can also open or save Newick formatted files.

- “Opening the Phylogenetic Tree Tool” on page 4-14 — Draw a phylogenetic tree from data in a `phytree` object or a previously saved file.
- “File Menu” on page 4-16 — Open tree data from a Newick formatted file, copy data to a MATLAB figure window, another tool window, or the MATLAB workspace, and save tree data.
- “Tools Menu” on page 4-25 — Explore branch paths, rename and edit branch and leaf names, hide selected branches and leaves, and rotate branches.
- “Windows Menu” on page 4-34 — Switch to any open window.
- “Help Menu” on page 4-34 — Select quick links to the Bioinformatics Toolbox documentation for phylogenetic analysis functions, tutorials, and the `phytreetool` reference.

Opening the Phylogenetic Tree Tool

The Phylogenetic Tree Tool can read data from Newick and ClustalW tree formatted files.

This procedure uses the phylogenetic tree data stored in the file `pf00002.tree` as an example. The data was retrieved from the protein family (PFAM) Web database and saved to a file using the accession number PF00002 and the function `gethmmtree`.

- 1** Create a `phytree` object. For example, to create a `phytree` object from tree data in the file `pf00002.tree`, type

```
tr= phytreeread('pf00002.tree')
```

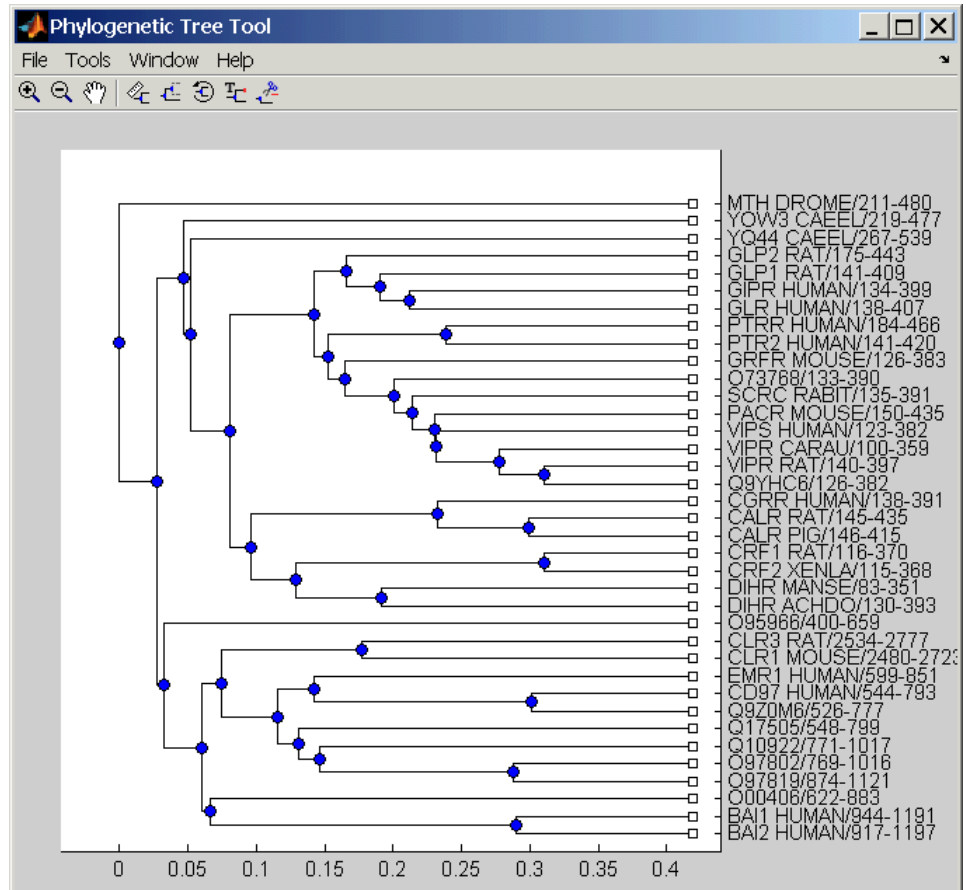
MATLAB creates a `phytree` object.

```
Phylogenetic tree object with 37 leaves (36 branches)
```

- 2** Open the Phylogenetic Tree Tool and draw a phylogenetic tree.

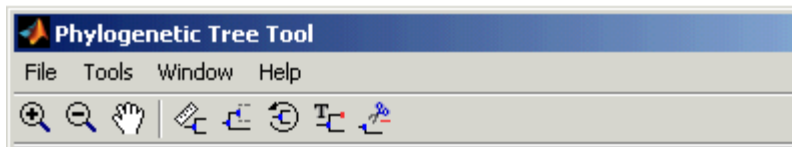
```
phytreetool(tr)
```

The Phylogenetic Tree Tool window opens.



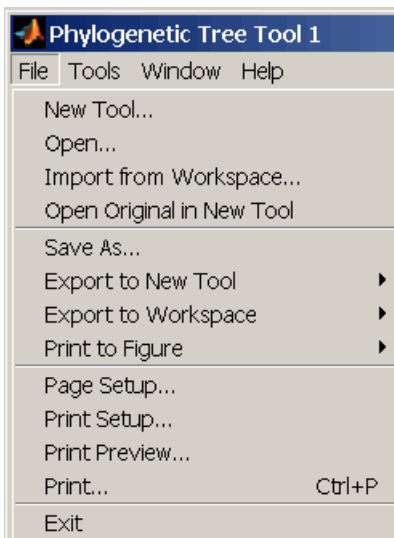
Alternately, if you do not give the `phytreetool` function an argument, the Select Phylogenetic Tree dialog box opens. Select a Newick formatted file and then click **Open**.

3 Select a command from the menu or toolbar.



File Menu

The **File** menu includes the standard commands for opening and closing a file, and it includes commands to use phytree object data from the MATLAB workspace. The File menu commands are shown below.

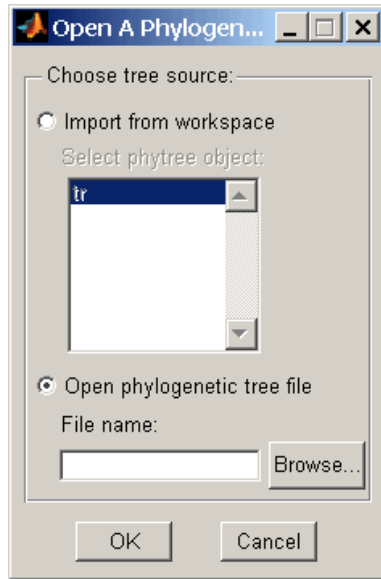


New Tool Command

Use the New Tool command to open tree data from a file into a second Phylogenetic Tree Tool window.

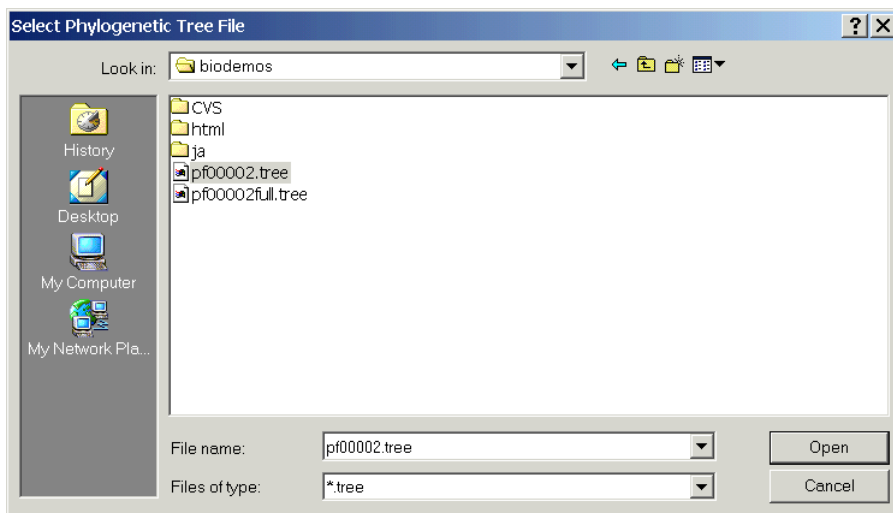
- 1 From the **File** menu, click **New Tool**.

The **Open A Phylogenetic Tree** dialog opens.



2 Choose the source for a tree.

- **MATLAB Workspace** — Select the **Import from workspace** options, and then select a phytree object from the list.
- **File** — Select the **Open phylogenetic tree file** option, click the **Browse** button, select a directory, select a file with the extension `.tree`, and then click **Open**. The Bioinformatics Toolbox uses the file extension `.tree` for Newick formatted files, but you can use any Newick formatted file with any extension.



MATLAB opens a second Phylogenetic Tree Tool window with tree data from the selected file.

Open Command

Use the **Open** command to read tree data from a Newick formatted file and display that data in a Phylogenetic Tree Tool.

- 1 From the **File** menu, click **Open**.

The **Select Phylogenetic Tree File** dialog box opens.

- 2 Select a directory, select a Newick formatted file, and then click **Open**. The Bioinformatics Toolbox uses the file extension `.tree` for Newick formatted files, but you can use any Newick formatted file with any extension.

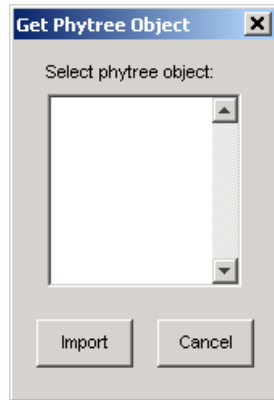
MATLAB replaces the current tree data with data from the selected file.

Import from Workspace Command

Use the **Import from Workspace** command to read tree data from a `phytree` object in the MATLAB workspace and display that data in a Phylogenetic Tree Tool.

- 1 From the **File** menu, click **Import from Workspace**.

The **Get Phytree Object** dialog box opens.



- 2 From the list, select a phytree object in the MATLAB workspace.
- 3 Click the **Import** button.

MATLAB replaces the current tree data in the Phylogenetic Tree Tool with data from the selected object.

Open Original in New Tool

There may be times when you make changes that you would like to undo. Phytreetool does not have an undo command, but you can get back to the original tree you started viewing with the **Open Original in New Tool** command.

From the **File** menu, click **Open Original in New Tool**.

A new Phylogenetic Tree Tool window opens with the original tree.

Save As Command

After you create a phytree object or prune a tree from existing data, you can save the resulting tree in a Newick formatted file. The sequence data used to create the phytree object is not saved with the tree.

- 1 From the **File** menu, click **Save As**.

The **Save Phylogenetic tree as** dialog box opens.

- 2 In the **Filename** box, enter the name of a file. The Bioinformatics Toolbox uses the file extension `.tree` for Newick formatted files, but you can use any file extension.

- 3 Click **Save**.

phytreetool saves tree data without the deleted branches, and it saves changes to branch and leaf names. Formatting changes such as branch rotations, collapsed branches, and zoom settings are not saved in the file.

Export to New Tool Command

Because some of the Phylogenetic Tree Tool commands cannot be undone (for example, the Prune command), you might want to make a copy of your tree before trying a command. At other times, you might want to compare two views of the same tree, and copying a tree to a new tool window allows you to make changes to both tree views independently .

- 1 From the **File** menu, point to the **Export to New Tool** submenu, and then click either **With Hidden Nodes** or **Only Displayed**.

A new **Phylogenetic Tree Tool** window opens with a copy of the tree.

- 2 Use the new figure to continue your analysis.

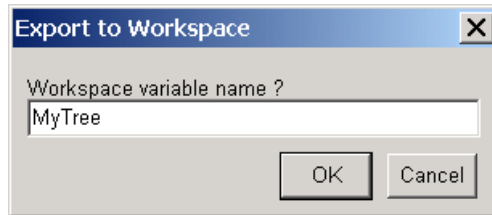
Export to Workspace Command

The Phylogenetic Tree Tool can open Newick formatted files with tree data. However, it does not create a phytree object in the MATLAB workspace. If you want to programmatically explore phylogenetic trees, you need to use the Export to Workspace command.

- 1 From the **File** menu, point to **Export to Workspace**, and then click either **With Hidden Nodes** or **Only Displayed**.

The **Export to Workspace** dialog box opens.

- 2 In the **MATLAB variable name** box, enter the name for your phylogenetic tree data. For example, enter MyTree.



- 3 Click **OK**.

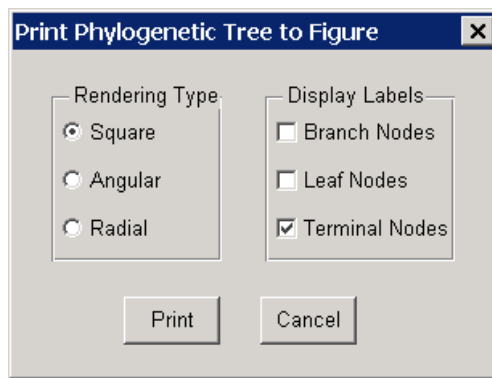
The `phytreetool` creates a `phytree` object in the MATLAB Workspace.

Print to Figure Command

After you have explored the relationships between branches and leaves in your tree, you can copy the tree to a MATLAB figure window. Using a figure window allows you to use all the MATLAB features for annotating, changing font characteristics, and getting your figure ready for publication. Also, from the figure window, you can save an image of the tree as it was displayed in the Phylogenetic Tree Tool window.

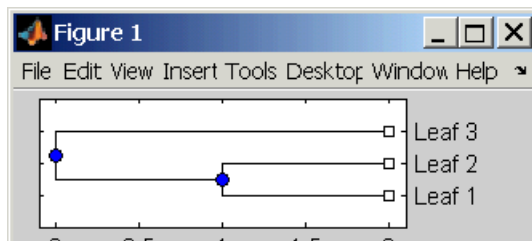
- 1 From the **File** menu, point to **Print to Figure**, and then click either **With Hidden Nodes** or **Only Displayed**.

The **Publish Phylogenetic Tree to Figure** dialog box opens.

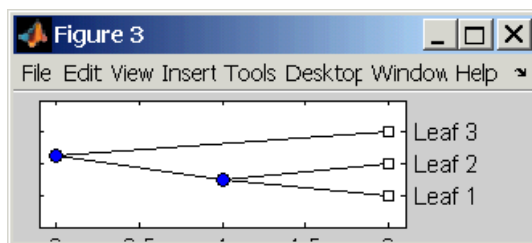


2 Select one of the Rendering Types, and then select the **Display Labels** you want on your figure.

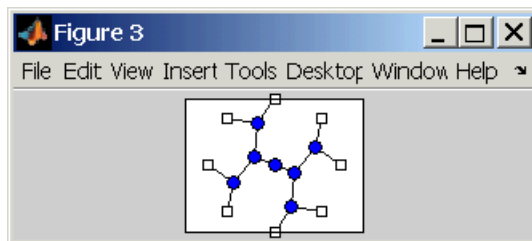
- **Square** (square branches)



- **Angular** (angular branches)



- **Radial**



3 Select the **Display Labels** you want on your figure. You can select from all to none of the options.

- **Branch Nodes** — Display branch node names on the figure.
- **Leaf Nodes** — Display leaf node names on the figure.
- **Terminal Nodes** — Display terminal node names on the right border.

4 Click the **Print** button.

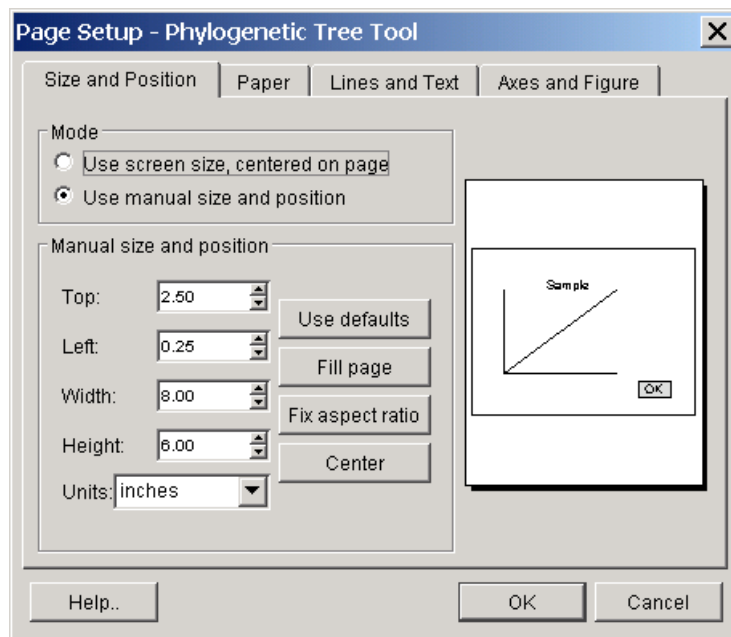
A new figure window opens with the characteristics you selected.

Page Setup Command

When you print from the Phylogenetic Tree Tool or a MATLAB figure window (with a tree published from the tool), you can specify setup options for printing a tree.

1 From the **File** menu, click **Page Setup**.

The **Page Setup - Phylogenetic Tree Tool** dialog box opens. This is the same dialog box MATLAB uses to select page formatting options.



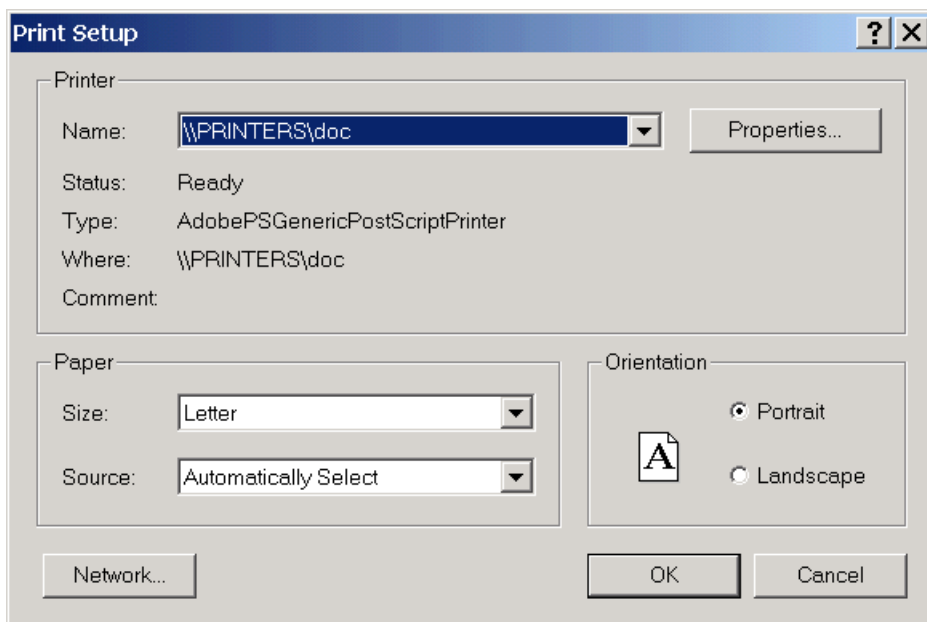
2 Select the page formatting options and values you want, and then click **OK**.

Print Setup Command

Use the Print Setup command with the Page Setup command to print a MATLAB figure window.

- 1 From the **File** menu, click **Print Setup**.

The Print Setup dialog box opens.



- 2 Select the printer and options you want, and then click **OK**.

Print Preview Command

Use the **Print Preview** command to check the formatting options you selected with the **Page Setup** command.

- 1 From the **File** menu, click **Print Preview**.

A window opens with a picture of your figure with the selected formatting options.

2 Click **Print** or **Close**.

Print

Use the **Print** command to make a copy of your phylogenetic tree after you use the **Page Setup** command to select formatting options.

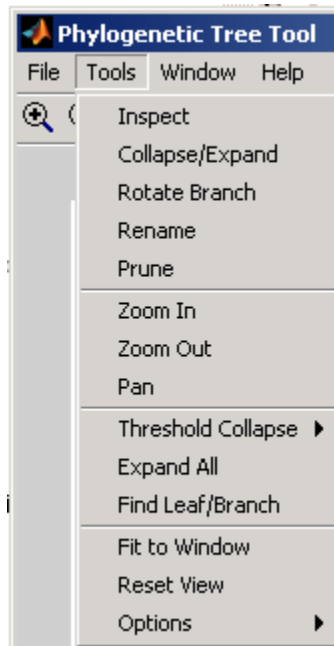
1 From the **File** menu, click **Print**.

The **Print** dialog box opens.

2 From the **Name** list, select a printer, and then click **OK**.


Tools Menu

The **Tools** menu and toolbar are where you will find most of the commands specific to trees and phylogenetic analysis. Use these commands and modes to interactively edit and format your tree. The Tools menu commands are shown below.



Inspect Mode Command

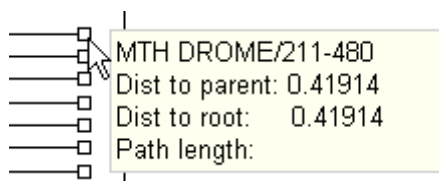
Use the inspect mode to compare path distances between sequences and to search for related sequences that might not be physically drawn close together.

- 1 From the **Tools** menu, click **Inspect**, or from the toolbar, click the Inspect Tool mode icon .

The **Phylogenetic Tree Tool** is set to inspect mode.

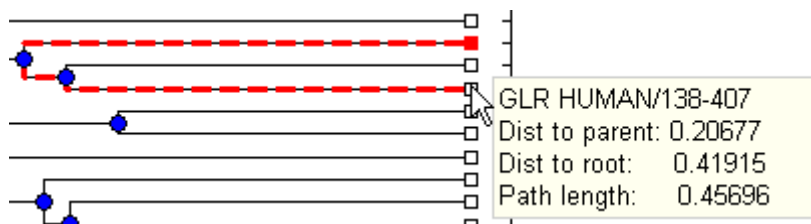
- 2 Point to a branch or leaf node.

A pop-up window opens with information about the patristic distances to parent and root nodes.




- 3 Click a branch or leaf node, and then move your mouse over another leaf node.

The tool highlights the path between nodes and displays the path length in the pop-up window. The path length is the patristic distances calculated by seqlinkage.



Collapse/Expand Branch Mode Command

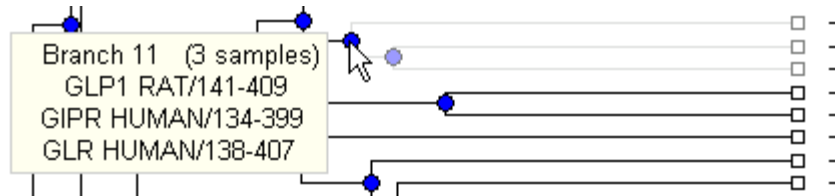
Some trees can have thousands of leaf and branch nodes. Displaying all the nodes can create a tree diagram that is unreadable. By collapsing some of the branches, you can better see the relationships between the remaining nodes.

- 1 From the **Tools** menu, click **Collapse/Expand**, or from the toolbar, click the Collapse/Expand node icon .

The **Phylogenetic Tree Tool** is set to collapse/expand mode.

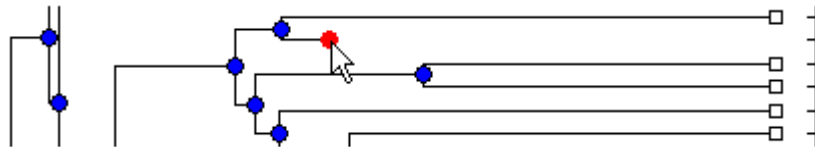
- 2 Point to a branch.

The selected paths to collapse (remove from view) are highlighted in gray.



- 3 Click the branch node.

The tool removes the display of branch and leaf nodes below the selected branch. The data is not removed.



- 4 To expand a branch, point to a collapsed branch and click.

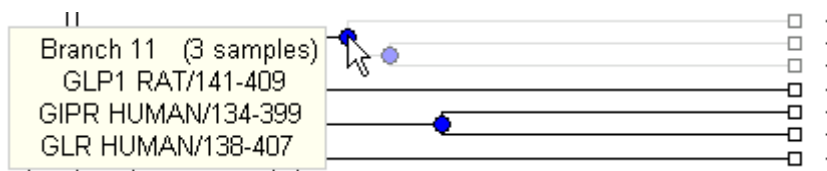
Rotate Branch Mode Command

A phylogenetic tree is initially created by pairing the two most similar sequences and then adding the remaining sequences in a decreasing order of similarity. You might want to rotate branches to emphasize the direction of evolution.

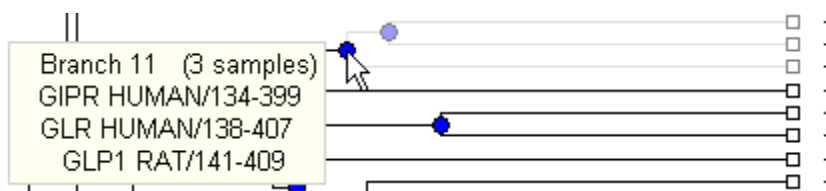
- 1 From the **Tools** menu, click **Rotate Branch**, or from the toolbar, click the Rotate Branch mode icon .

The **Phylogenetic Tree Tool** is set to rotate branch mode.

- 2 Point to a branch node.




- 3 Click the branch node.

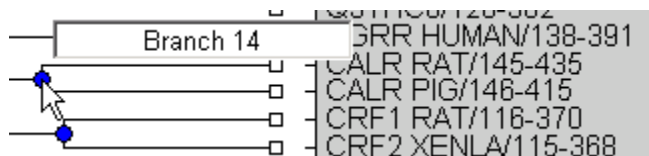


The branch and leaf nodes are rotated 180 degrees around the selected branch node.

Rename Leaf/Branch Mode Command

The Phylogenetic Tree Tool takes the node names from the phytree object and creates numbered branch names starting with Branch 1. You can edit and change or replace any of the leaf or branch names. Changes to branch and leaf names are saved when you use the **Save** command.

- 1 From the **Tools** menu, click **Rename**, or from the toolbar, click the Rename mode icon .
- 2 Click a branch or leaf node.



A text box opens with the current name of the node.


- 3 In the text box, edit or enter an new name.



4 To save your changes, click outside of text box.

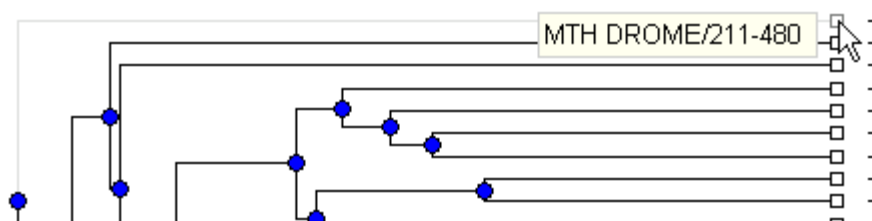
Prune (delete) Leaf/Branch Mode Command

Your tree might contain leaves that are far outside the phylogeny, or it might have duplicate leaves that you want to remove.

1 From the **Tools** menu, click **Prune**, or from the toolbar, click the prune icon .

The **Phylogenetic Tree Tool** is set to rename mode.

2 Point to a branch or leaf node.



For leaf node, the branch line connected to the leaf is highlighted in gray. For a branch nodes, the branch lines below the node are highlighted in light gray.

Note If you delete nodes (branches or leaves), you cannot undo the changes. The Phylogenetic Tree Tool does not have an Undo command.

3 Click the branch or leaf node.

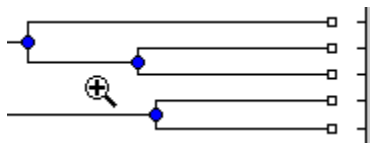
The branch is removed from the figure and the other nodes are rearranged to balance the tree structure. The phylogeny is not recalculated.

Zoom In, Zoom Out, and Pan Commands

The Zoom and Pan commands are the standard controls with MATLAB figures for resizing and moving the screen.

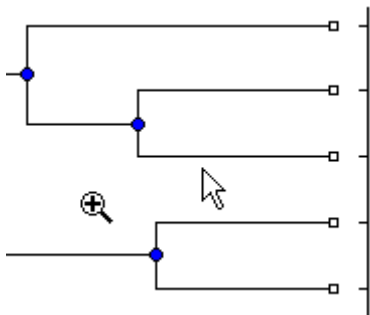
- 1 From the **Tools** menu, click **Zoom In**, or from the toolbar click the zoom in icon .


The tool activates zoom in mode and changes the cursor to a magnifying glass.






- 2 Place the cursor over the section of the tree diagram you want to enlarge and then click.

The tree diagram is enlarged to twice its size.



- 3 From the toolbar click the Pan icon .
- 4 Move the cursor over the tree diagram, left-click, and drag the diagram to the location you want to view.

Zoom In , **Zoom Out** , **Pan** 

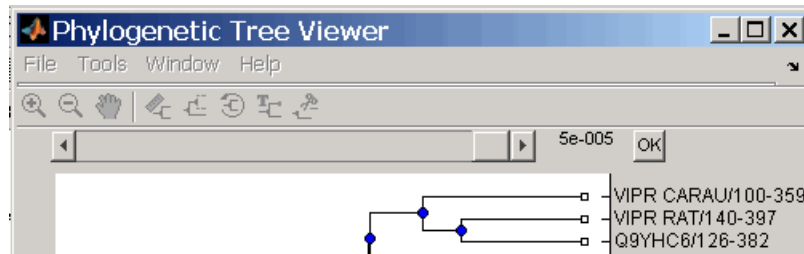
Threshold Collapse Command

Use the **Threshold Collapse** command to collapse the display of nodes using a distance criterion instead of interactively selecting nodes with the **Collapse/Expand** command. Branches with distances below the threshold are collapsed from the display.

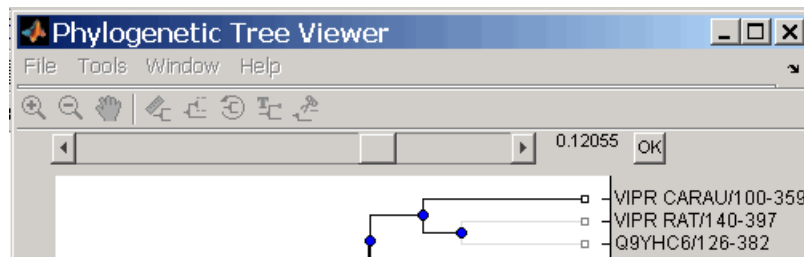
1 From the **Tools** menu, click **Threshold Collapse**, and select one of the following:

- **Distance to Leaves** — Sets the threshold starting from the right of the tree.
- **Distance to Root** — Sets the threshold starting from the root node at the left side of the tree.

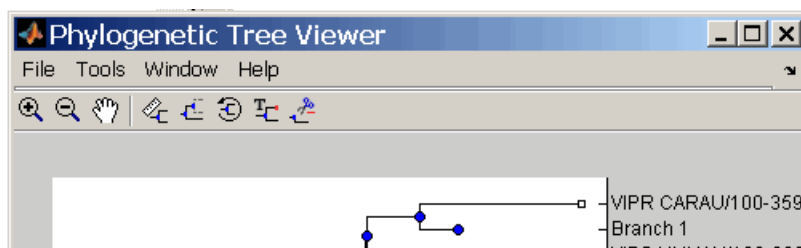
The collapse slider bar is displayed at the top of the diagram.



2 Click and drag the slider bar to the left to set the distance threshold.



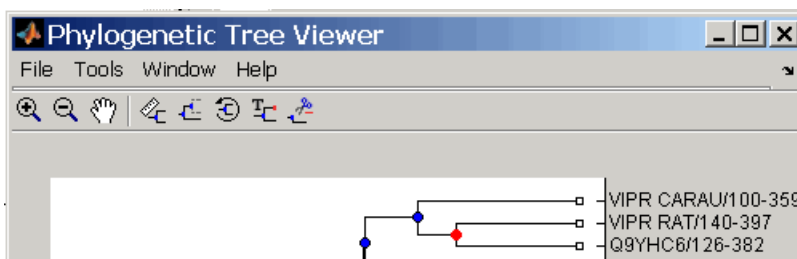
- 3 Click the **OK** button to the right of the slider. The nodes below the distance threshold are hidden.



Expand All Command

The data for branches and leaves you hide with the **Collapse/Expand** or **Threshold Collapse** commands are not removed from the tree. You can display the hidden data using these commands or display all hidden data with the **Expand All** command.

Select **Tools > Expand All**. The hidden branches and leaves are displayed.

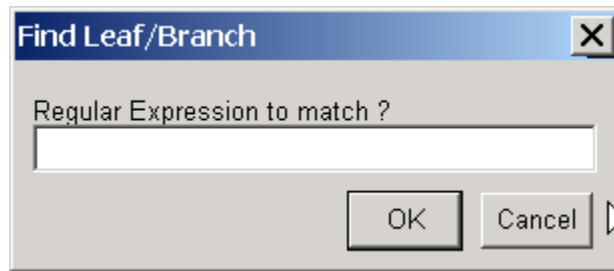


Find Leaf/Branch Command

Phylogenetic trees can have thousands of leaves and branches, and finding a specific node can be difficult. Use the Find command to locate a node using its name or part of its name.

- 1 Select **Tools > Find Leaf/Branch**.

The Find Leaf/Branch dialog box opens.



- 2 In the **Regular Expression to match** box, enter a name or partial name of a branch or leaf.
- 3 Click **OK**.

Fit to Window

After you hide nodes with the **Collapse/Expand** or **Threshold Collapse** commands, or delete nodes with the **Prune** command, there might be extra space in the tree diagram. Use the **Fit to Window** command to redraw the tree diagram to fill the entire figure window.

From the **Tools** menu, click **Fit to Window**.

Reset View Command

Use the Reset Window command to remove formatting changes such as rotations, collapsed branches, and zooms.

From the **Tools** menu, click **Reset Window**.

Options Submenu

Use the Options command to select the behavior for the zoom and pan modes.

- **Unconstrained Zoom** — Allow zooming in both horizontal and vertical directions.
- **Horizontal Zoom** — Restrict zoom to the horizontal direction.
- **Vertical Zoom** — Zoom only in the vertical direction (default).

- **Unconstrained Pan** — Allow panning in both horizontal and vertical directions.
- **Horizontal Pan** — Restrict panning to horizontal direction.
- **Vertical Pan** — Pan only in the vertical direction (default).

Windows Menu

The **Windows** menu is standard on MATLAB GUI and figure windows. Use this menu to select any opened window.

Help Menu

Use the **Help** menu to select quick links to the Bioinformatics Toolbox documentation for phylogenetic analysis functions, tutorials, and the `phytreetool` reference.

Examples

Use this list to find examples in the documentation.

Sequence Analysis

“Example: Sequence Statistics” on page 2-2

“Example: Sequence Alignment” on page 2-18

Microarray Analysis

“Example: Visualizing Microarray Data” on page 3-2

“Example: Analyzing Gene Expression Profiles” on page 3-25

Phylogenetic Analysis

“Example: Building a Phylogenetic Tree” on page 4-2

A

- amino acids
 - comparing sequences 2-28
 - composition 2-15
- applications
 - deploying 1-16
 - prototyping 1-16

B

- bioinformatics
 - application deployment 1-17
 - computation with MATLAB 1-2
 - data visualization 1-16
 - visualizing data 1-2
- Bioinformatics Toolbox
 - additional software 1-5
 - expected user 1-4
 - installation 1-5
 - required software 1-5

C

- clusters
 - gene expression data 3-32
- codons
 - nucleotide composition 2-9
- composition
 - amino acid 2-15
 - nucleotide 2-9
- conversions
 - nucleotide to amino acid 2-15

D

- data
 - filtering microarray data 3-29
 - getting into MATLAB 2-4
 - loading into MATLAB 3-25
 - microarray 3-3

- data formats
 - supporting functions 1-8
- data visualization
 - bioinformatics 1-16
- databases
 - getting information from 2-20
 - related genes 2-23
 - supporting functions 1-8

E

- examples
 - gene expression in mouse brain 3-2
 - gene expression in yeast metabolism 3-25
 - sequence alignment 2-18
 - sequence statistics 2-2

F

- features
 - prototyping 1-16
- functions
 - data formats 1-8
 - databases 1-8
 - graph theory 1-14
 - graph visualization 1-15
 - mass spectrometry analysis 1-13
 - microarray analysis 1-12
 - protein structure analysis 1-11
 - sequence alignment 1-9
 - sequence utilities 1-10
 - statistical learning 1-15

G

- gene expression profile
 - mouse brain 3-2
 - yeast metabolism 3-25
- genome data
 - with MATLAB structures 3-25
- graph theory

- supporting functions 1-14
- graph visualization
 - supporting functions 1-15

I

- installation
 - from CD or Web 1-5

M

- mass spectrometry analysis
 - supporting functions 1-13
- MATLAB structures
 - with genome data 2-4
- microarray
 - clustering genes 3-32
 - filtering data 3-29
 - mouse brain example 3-1
 - principal component analysis 3-36
 - scatter plots 3-16
 - spacial images 3-5
 - statistics 3-15
 - visualizing data 3-2
 - working with data 3-3
 - yeast example 3-1
- microarray analysis
 - supporting functions 1-12
- model organism
 - finding 2-18
- mouse brain
 - gene expression profile 3-2
 - microarray tutorial 3-2
- multiple sequence alignment
 - aligning sequences 2-50
 - manual adjustment 2-51
- Multiple Sequence Alignment Viewer
 - GUI 2-48

N

- NCBI
 - searching Web site 2-18
- nucleotides
 - composition in sequences 2-5
 - content in sequences 2-2
 - searching database 2-23

O

- open reading frames
 - searching for 2-12

P

- phylogenetic analysis
 - building tree 4-2
 - creating subtree 4-8
 - creating tree 4-6
 - exploring tree 4-10
 - GUI reference 4-14
 - reading data 2-48
 - searching NCBI 4-4
 - selecting subtree 2-49
- plots
 - scatter 3-16
- principal component analysis
 - filtering microarray data 3-36
- protein properties
 - analysis functions 1-11
- protein sequence
 - locating 2-25
- prototyping
 - supporting features 1-16

S

- sequence
 - amino acid conversion 2-15
 - codon composition 2-9

- comparing amino acids 2-28
- nucleotide content 2-2
- protein coding 2-25
- searching database 2-23
- statistics example 2-2
- sequence alignment
 - example 2-18
 - supporting functions 1-9
- sequence analysis
 - defined 2-1
 - using seqtool GUI 2-37
- sequence tool GUI
 - importing sequence 2-37
 - reading frames 2-42
 - searching words 2-41
 - statistics 2-45
 - viewing sequence 2-39
- sequence utilities
 - supporting functions 1-10

- sequences
 - nucleotide composition 2-5
- share algorithms
 - bioinformatics 1-17
- software
 - required 1-5
- spatial images
 - microarray 3-5
- statistical learning
 - supporting functions 1-15
- statistics
 - microarray 3-15
- structures
 - with genome data 3-25

V

- visualizing data
 - microarray 3-2