Omics Database Generator Manual

Joseph Guhlin

Contents

1 Introduction

This software takes genomic data, including sequence data, gene definitions, annotations, expression, and other data to compile a queryable graph-theory based database for intra- and interspecies comparisons. Primarily analysis is completed by using established tools and common assumptions to process the output of those programs. Custom data can also be entered into the database for advanced queries.

Queries can be completed by command-line, a provided web interface, or programmatically either through an API or directly to the database.

Any genomes that have data available in standard file formats are compatible with this software, not just plant and bacteria genomes.

Need for this Pipeline

With a rapid expansion of genomic resources, researchers need the ability to rapidly compare genomes and putative coding regions to existing, annotated, curated genomes and datasets to derive maximum information.

Genomic resources are increasing at an accelerated rate. Just as well, existing genomic resources are being updated with new assemblies and new annotations. Assemblies and annotations can differ widely between releases due to new software releases, additional sequencing, and new information regarding genetic data. Existing material may refer to an older annotation or assembly and becomes difficult to use with newer releases unless additional data mining is performed. This pipeline automates many of these steps and returns the data in a comprehensive database.

Genomic comparisons are also incredibly useful in this time of accelerated data expansion. This pipeline automates protein blast searches for homology in the same species and between species, with a focus on comparisons between other species in the search for statistically significant orthologs.

Genomic context is important for helping to validate GWAS candidates before performing additional labwork. Genomic context is also important in comparisons between species as well as deriving an additional understanding of a genome. This pipeline automates the importation of Gene Ontology (GO) terms and Plant Ontology (PO) terms. The importation of Interproscan results is also handled and extrapolated to link the gene models to the GO terms as appropriate.

Terms for Plant Ontology are imported if a TAIR style file is provided. Expression data from Cufflinks output is also imported and interpreted using Pearson correlation looking for both positive and negative correlation and tissue or experimental condition specificity.

Additionally, flanking sequence tagged data is mapped onto the assemblies, and Affymetrix probe sets are mapped to the mRNA models. This allows rapid remapping when a new assembly is released.

This allows searches via gene model s or chromosome and base pair location numbers (in the form of Chr:Loc such as Chr2:340812 or 2:543923), and returns the nearest genes, and if there is enrichment of GO terms or PO terms, and if there is enrichment of GO/PO terms in orthologous regions of other species. Enrichment for tissue specificity or experimental condition specificity is also provided if the data is available. Mutants and Affymetrix probeset data are also returned if in the nearby genomic neighborhood.

What this Pipeline is Not

This pipeline does not, at this time, do any gene prediction. The pipeline takes in the GFF files provided from either annotation sources or gene prediction software. ODG provides an additional layer of annotation on top of the gene predictions from software, and could be used to provide some easy, comparative-based annotations.

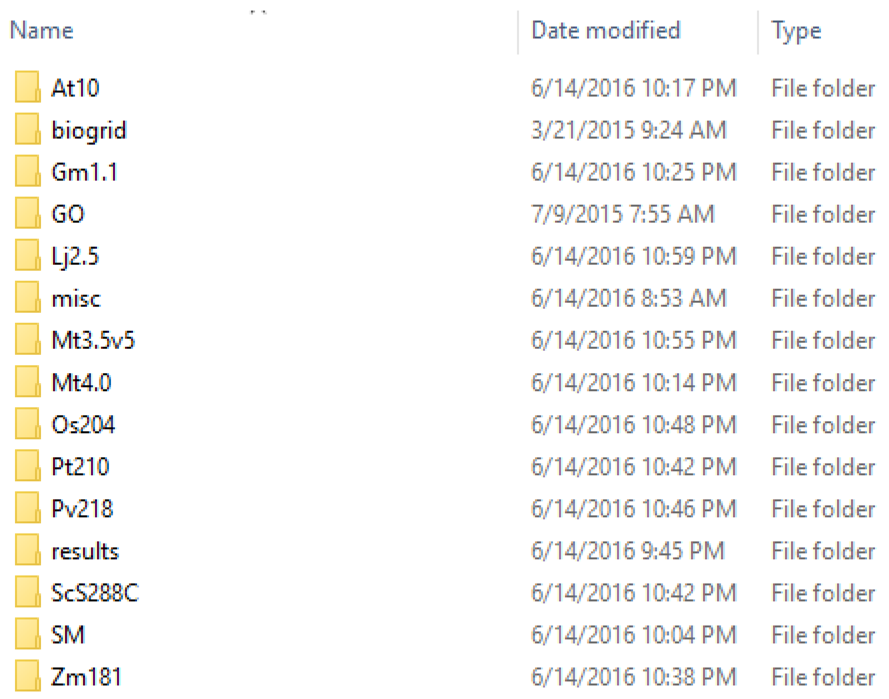
Brief Overview of Data from this Pipeline

This pipeline combines several different types of data, as well as additional analysis of this data, into a unified database. This allows some unique queries that would be difficult or impossible with only the raw files. Some examples include:

- Translate Gene IDs from one annotation to another annotation
- Orthologs between species, including Blast Score Ratios (BSR)
- Genomic neighborhoods(a location on a chromosome and all nearby genes), including data such as
    - Expression enrichment for genomic neighborhoods
    - Gene Ontology / Plant Ontology term enrichment for genomic neighborhoods
    - Families and Domains via Interproscan enrichment for genomic neighborhoods
    - Similar neighborhoods in species and out species
- Neighborhoods can also be created by co-expression and negatively correlated genes, as well as orthologs
- Possible locations of Transposon-mediated mutant insertions
- Possible miRNA locations
- Ontology queries for genes and regions
- Syntenic regions based on gene models and blast searches
    - Can be used as additional evidence for Gene ID translations between annotation versions
    - Can be used with protein blast results between species as well

2 Getting Started

After extracting the compressed archive (.zip or .tgz) of ODG, you can begin placing your data files in the *data* directory. For each assembly and annotation data, you will create a directory. It is recommended to name each directory with something descriptive, such as "Arabidopsis_thaliana_10" or "At10" to help you in the configuration step. Each assembly should be placed into individual folders, with annotation and other related data in the same folder. ODG provides the concept of a species and a "version," which could mean the assembly release version, or could mean strain or accession depending on your use-case. See *Fig 2.1* for an example of what a data directory's contents can be.

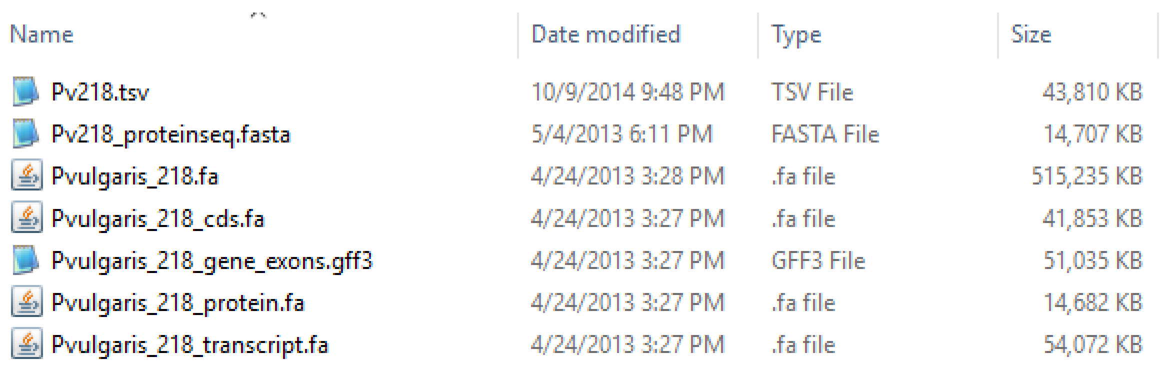


Name	Date modified	Type
At10	6/14/2016 10:17 PM	File folder
biogrid	3/21/2015 9:24 AM	File folder
Gm1.1	6/14/2016 10:25 PM	File folder
GO	7/9/2015 7:55 AM	File folder
Lj2.5	6/14/2016 10:59 PM	File folder
misc	6/14/2016 8:53 AM	File folder
Mt3.5v5	6/14/2016 10:55 PM	File folder
Mt4.0	6/14/2016 10:14 PM	File folder
Os204	6/14/2016 10:48 PM	File folder
Pt210	6/14/2016 10:42 PM	File folder
Pv218	6/14/2016 10:46 PM	File folder
results	6/14/2016 9:45 PM	File folder
ScS288C	6/14/2016 10:42 PM	File folder
SM	6/14/2016 10:04 PM	File folder
Zm181	6/14/2016 10:38 PM	File folder

Figure 2.1. Example contents of *data* directory; "misc," "biogrid," and "GO" contain additional annotation metadata while "results" is auto-generated during later processing steps. The rest of the directories represent species.

Place all associated files for each assembly and version in their associated directories as exampled in Figure 2.2. Accepted file types and formats are:

- FASTA files for Assembly, Proteins, Transcripts, miRNA definitions, and gene sequences
- GFF3 files for genes and other features definitions also appended FASTA assembly
- TSV – InterProScan Results, BLAST+ Results
- .hmm.tbl – HMM Results
- Pathways – Must be the same format as PlantCyc
- .assoc – GAF 2.0 – Ontological Associations File
- .gtf / .fpkm_tracking – Cufflinks Expression



Name	Date modified	Type	Size
Pv218.tsv	10/9/2014 9:48 PM	TSV File	43,810 KB
Pv218_proteinseq.fasta	5/4/2013 6:11 PM	FASTA File	14,707 KB
Pvulgaris_218.fa	4/24/2013 3:28 PM	.fa file	515,235 KB
Pvulgaris_218_cds.fa	4/24/2013 3:27 PM	.fa file	41,853 KB
Pvulgaris_218_gene_exons.gff3	4/24/2013 3:27 PM	GFF3 File	51,035 KB
Pvulgaris_218_protein.fa	4/24/2013 3:27 PM	.fa file	14,682 KB
Pvulgaris_218_transcript.fa	4/24/2013 3:27 PM	.fa file	54,072 KB

Figure 2.2. Contents of the Pv218 folder, showing the assembly FASTA file (Pvulgaris_218.fa), the CDS FASTA file (Pvulgaris_218_cds.fa), gene definition file (Pvulgaris_218_gene_exons.gff3), protein FASTA file (Pvulgaris_218_protein.fa), and the InterProScan results file (Pv218.tsv).

Global Metadata Files

Gene Ontology

GO/go.obo – Download the latest version of go.obo at http://geneontology.org/page/download-ontology

ENZYME

misc/enzyme.dat – Download the latest version of enzyme.dat from ftp://ftp.expasy.org/databases/enzyme

UNIPATHWAY

misc/unipathway.obo – Download the latest version of unipathway.obo from UniPathway's website

Molecular Interactions

misc/mi.obo – Download the latest from http://ontologies.berkeleybop.org/mi.obo

Pre-Configuration Processing

Running InterProScan

https://github.com/ebi-pf-team/interproscan/wiki/HowToRun

We suggest the following options, as they allow ODG to make the maximum number of connections in the database.

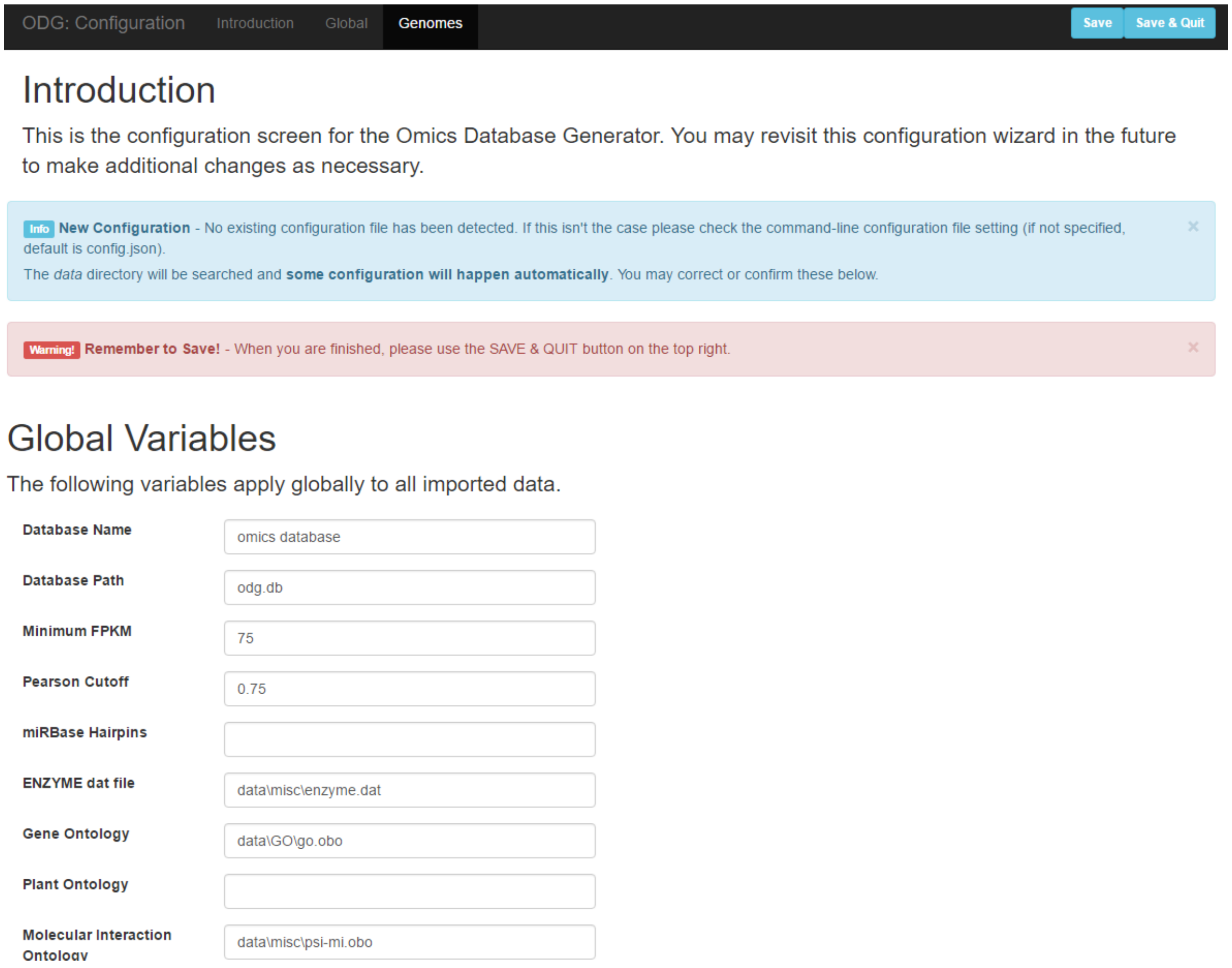
- --goterms
- --iprlookup
- --pathways

If you alter any of the default output formats, you must be certain TSV is a selected output format, as this is the format ODG will read in.

After InterProScan: Copy the .tsv files to your data/<Genome Name>/ directories and re-start the configuration screen. Make sure to select the .tsv files for InterProScan for each genome you processed.

Configuring the Database

To configure the database click on start-config.bat or run bash ./config.sh and point your web browser to http://localhost:33333/ This will give you the initial configuration screen (unless you are editing a previous configuration). It may take a moment to load depending on the size of your dataset. Figures 2.3 and 2.4 show initial configuration parameters and an example of a specific genome configuration. Remember, each genome must be in a separate folder in the data directory, this translates into separate entries for the configuration program. Once everything is configured, you must save the configuration file.



Introduction

This is the configuration screen for the Omics Database Generator. You may revisit this configuration wizard in the future to make additional changes as necessary.

Info **New Configuration** - No existing configuration file has been detected. If this isn't the case please check the command-line configuration file setting (if not specified, default is config.json).
The *data* directory will be searched and **some configuration will happen automatically**. You may correct or confirm these below.

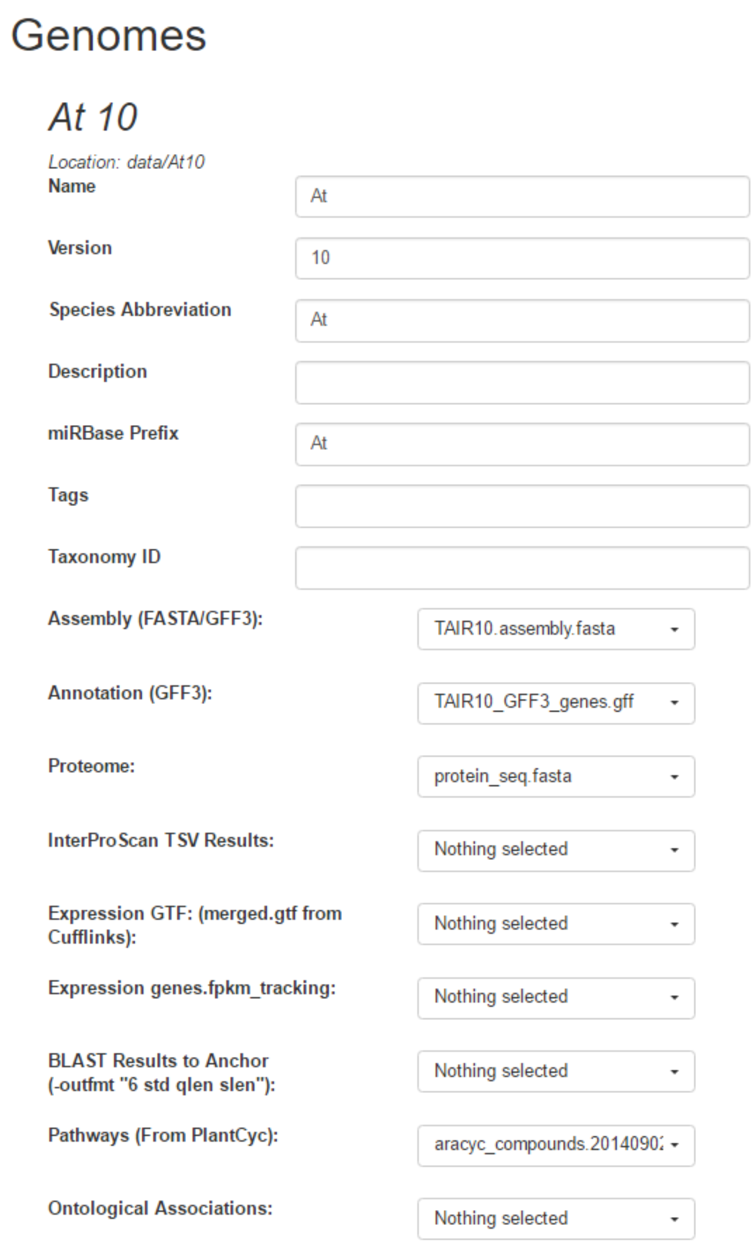
Warning! **Remember to Save!** - When you are finished, please use the SAVE & QUIT button on the top right.

Global Variables

The following variables apply globally to all imported data.

Database Name	omics database
Database Path	odg.db
Minimum FPKM	75
Pearson Cutoff	0.75
miRBase Hairpins	
ENZYME dat file	data\misc\enzyme.dat
Gene Ontology	data\GO\go.obo
Plant Ontology	
Molecular Interaction Ontology	data\misc\psi-mi.obo

Figure 2.3. The initial configuration screen. Here you can set the database name, the path of the database, which is important if you are building multiple ODG instances, and set several other variables. If you are not using files, you may leave them blank. When possible, ODG's configuration screen tries to identify and fill in certain files when they are present.


Genomes

At 10

Location: data/At10

Name	At
Version	10
Species Abbreviation	At
Description	
miRBase Prefix	At
Tags	
Taxonomy ID	
Assembly (FASTA/GFF3):	TAIR10.assembly.fasta
Annotation (GFF3):	TAIR10_GFF3_genes.gff
Proteome:	protein_seq.fasta
InterProScan TSV Results:	Nothing selected
Expression GTF: (merged.gtf from Cufflinks):	Nothing selected
Expression genes.fpkm_tracking:	Nothing selected
BLAST Results to Anchor (-outfmt "6 std qlen slen"):	Nothing selected
Pathways (From PlantCyc):	aracyc_compounds.2014090:
Ontological Associations:	Nothing selected

Figure 2.4. The initial screen for a genome configuration entry. When you are first editing a genome ODG will attempt to fill in as much data as possible and attempt to select the appropriate files.

Post-Configuration Steps

Generating BLAST+ Scripts

If you wish to run BLAST+ on a separate machine, such as a remote server, please copy everything in your ODG directory, including all subdirectories, to that machine. You can then execute "**create-blast-scripts.sh**" or "**create-blast-scripts.bat**" (when using a Windows machine). To then run the BLAST+ commands, you will execute "**run-blast-scripts.sh**" (or "**run-blast-scripts.bat**"). This process will take some additional time. Advanced users can examine other options by running ODG without any arguments.

Generating InterProScan Results

While the detailed instructions for running InterProScan locally is beyond the scope of this manual, it is relatively straight-forward. Please be aware InterProScan must be run on a UNIX machine, neither Windows nor Mac OS X will suffice.

Please begin here: https://github.com/ebi-pf-team/interproscan/wiki/HowToDownload

Please Note: Once you have downloaded InterProScan, you must also download the Panther Models. This is a very large file that you will want to keep between subsequent runs of InterProScan if at all possible.

Generating the Database

Run "create-db.sh" or "create-db.bat" as necessary for your machine. This process can take a long time depending on the number of genomes and other data being connected in the database.

Querying the Database

Web-based Query Mode

To begin the web-based query server, simply run the command "query-server.sh" or "query-server.bat" as necessary for your machine. Then you may set your browser to http://localhost:6789

The query server can be terminated by switching to the window and hitting Ctrl-C.

Command-line Query Mode

The following queries are built-in and may be used from the command line. An example would be './odg.sh query-tassel –s "Species Name" –v "Version" tassel-output.txt'

Species name and version should always be provided and based on their entry in the original configuration. Input files should be tab-delimited unless otherwise specified. The “-o” option can be passed to alter the output file names.

query-tassel

Input file is the –stats.txt file from TASSEL output. Command-line should be run such as:

‘./odg.sh query-tassel –s “Species Name” –v “version” chr1-stats.txt’

It is advisable to filter the TASSEL file to only the top hits, to avoid analyzing each individual SNP in the genome regardless of statistical significance.

query-gene-list

Computed aggregate stats for a gene list, given in the format of gene_id followed by allele frequency.

‘./odg.sh query-gene-list –s “Species Name” –v “version” gene_list.tsv’

annotate-gene-list

Annotate a gene list with further information from the database, the input file should be a file with the first column being gene-ids.

‘./odg.sh annotate-gene-list –s “Species Name” –v “version” gene_list.tsv’

coexpression-network

Generates a basic co-expresison network based on Pearson-correlation coefficients. Input file should be a list of gene names. For example this could be done given a list of genes that are statistically significant in a GWAS analysis. Output files include a TSV and a GDF file, the latter which can be viewed using Gephi software or other GDF-supporting graph interfaces.

‘./odg.sh coexpression-network –s “Species Name” –v “version” gene_list.tsv’

get-biological-processes

Given a gene list, identifies GO terms labelled as biological process and outputs a list.

‘./odg.sh biological-processes –s “Species Name” –v “version” gene_list.tsv’

get-biological-processes-all-genes

Identifies all genes with a biological process GO-term for a species. Useful for generating a null distribution for biological process tests for a given gene list.

‘./odg.sh biological-processes –s “Species Name” –v “version”’

Internal Data Structure

Nodes

In a graph database a node is a piece of data that serves the same function as a 'row' in a spreadsheet or SQL table. A node is connected by relationships, which may be incoming, outgoing, or non-directional. Nodes will typically have properties, which are analagous to a column entry of a row in a spreadsheet. All nodes also have one or more “type” which tells you what type of data the node contains, and which types of relationships are possible. For example a “gene” node may be a “parent” of an mRNA node, which is itself a parent of an “exon” node .

Each node has a unique, internally generated “id” number. These ID numbers are useful to refer to nodes in scripts and multiple-query analyses, however they are guaranteed to change if the database is ever regenerated, even if no input files change.

Relationships

Nodes are connected to each other by relationships, which have no direct analogy in spreadsheets. Relationships themselves may have properties and a type, and all relationships have a start and an end, which are both nodes. A node may have a relationship to itself, but this is not common in the output database of this project.

Examining the Internal Data Structure

The best way to examine the internal database, not through the ODG query lens, is to download the database host software from https://neo4j.com/

By downloading this, it is possible to open the database directly in Neo4J’s web-service query, from which you can see all node labels and relationship types, and explore them. This web-based interface allows a graphical interpretation of the networks provided. Further queries can be written directly in Neo4J’s query-language, CYPHER, with more information available here: https://neo4j.com/developer/cypher-query-language/

This interface is useful for building queries, and testing concepts and ideas. However, for any complicated analyses, the best approach is to use a programming or scripting language and hook into the database. This will often require the host software and is beyond the scope of this guide.

Language Hooks

Language hooks allow you to directly connect from a programming language such as R or python to the output database of Neo4J and write advanced scripts and analyses.

R-language

https://neo4j.com/developer/r/

Python

https://neo4j.com/developer/python/

Java

https://neo4j.com/developer/java/

Perl

https://neo4j.com/developer/perl/

Additional Language Hooks

Additional languages can often be found on the internet, and some are available here:

https://neo4j.com/developer/language-guides/