Genotypes Loader Documentation

Carolyn T Caron et al., University of Saskatchewan, Pulse Bioinfo

Apr 21, 2023

Contents:

This module provides a drush command to load genotypic data from a variety of file formats. Data is saved in GMOD Chado in accordance with ND Genotypes including use of the `genotype_call` table.

Note: For further support of genotypic data such as variant/marker pages and a variant by germplasm matrix, see ND Genotypes.

CHAPTER 1

Installation

Install this module as you would any other Drupal module after ensuring you have the following dependencies.

1.1 Dependencies

- Tripal 3.x
- PostgreSQL 9.3 and up (at least 9.4 is recommended)
- Drush

CHAPTER 2

Configuration

2.1 Controlled Vocabulary Terms

This module currently expects the following controlled vocabulary terms already exist in chado:

Purpose	Term Name	Controlled Vocabulary
Sample type	genomic_DNA	sequence
Sample => germplasm relationship	is_extracted_from	stock_relationship
Marker => variant relationship	is_marker_of	stock_relationship
Marker type	Indicated by user	sequence
Variant type	Indicated by user	sequence
Property type for free-text marker description	marker_type	feature_property

You can configure these terms through a settings form under module configuration. In the future, you will also be able to configure the controlled vocabularies for these terms as well.

2.2 Database Write Options

There is configuration to specify whether you wan the loader to insert only, update only or update/insert as needed.

CHAPTER 3

Usage

3.1 Drush Command

- Command: load-genotypes
- Alias: load-geno
- **Arguments:**
  - input-file: The filename of the matrix file for upload
  - sample-file: The filename of a tab-delimited file specifying for each sample name in the genotypes file: the name of the stock in the database, the stock accession ID, the name of the germplasm, the germplasm Accession ID, type fo germplasm, and organism (optional). See "samples.list" in the sample_files folder for an example.
- **Options:**
  - variant-type: The Sequence Ontology (SO) term name that describes the type of variants in the file (eg. SNP, MNP, indel).
  - marker-type: A free-text title that describes the marker technology used to generate the genotypes in the file (e.g. "Exome Capture", "GBS", "KASPar", etc.).
  - ndgeolocation: A meaningful location associated with this natural diversity experiment. For example, this could be the location the assay was completed in, the location the germplasm collection was from, or the location the markers were developed at. This should be the description field of your ndgeolocation.
  - organism: The organism of the reference genome which was used for aligning reads to call the variants. If there is an empty value in the "Organism" column of the sample file, the loader will default to this parameter.
  - project-name: All genotypes will be grouped via a project to allow users to specify a particular dataset.

This loader supports 3 different file formats (described under file formats below) and will auto-detect which format you have provided.

3.2 Example Usage

- Load a genotype matrix file (mymatrix.tsv) using the sample/germplasm information provided in samples.list. With this example, you will be prompted to enter each of the options listed above.

```
drush load-genotypes mymatrix.tsv samples.list
```

- Load a VCF file (mygenotypes.vcf) using the sample/germplasm information provided in samples.list but provide the command with all the options upfront to avoid prompting.

```
drush load-genotypes sample.vcf samples.list --organism="Citrus sinensis" \
  --variant-type="SNP" --marker-type="genetic_marker" \
  --project-name="Citrus Demonstration Genotypic Data"
```

CHAPTER 4

File Formats

4.1 Genotypes File

This module supports loading of three types of genotype files:

4.1.1 1. VCF

```
##fileformat=VCFv4.0
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=1000GenomesPilot-NCBI36
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  Ross Prado   Ash
1A      14370   .       G       A       29      PASS    NS=3;DP=14;AF=0.5;DB;H2 GT:GQ:DP:HQ     0|0:48:1:51,51  1|0:48:8:51,51  1/1:43:5:.,.
1A      17330   .       T       A       3       q10     NS=3;DP=11;AF=0.017 GT:GQ:DP:HQ     0|0:49:3:58,50  0|1:3:5:65,3    0/0:41:3
1A      1110696 rs6040355       A       G,T     67      PASS    NS=2;DP=10;AF=0.333,0.667;AA=T;DB       GT:GQ:DP:HQ     1|2:21:6:23,27  2|1:2:0:18,2    2/2:35:4
```

(continues on next page)

(continued from previous page)

```
1A    1230237 .       T       .       47      PASS    NS=3;DP=13;AA=T GT:GQ:DP:HQ
↪0|0:54:7:56,60  0|0:48:4:51,51  0/0:61:2
1A    11111   1subfield       C       A       50      PASS    A=1;B=2;C=3     GT
↪0/1     ./.     1/1
```

4.1.2 2. Genotype Matrix

A tab-delimited data file where each line corresponds to a SNP and columns correspond to germplasm assayed. Expected columns: (1) Marker Name, (2) Chromosome Name, (3) Position on Chromosome, (4+) Sample Genotype Calls.

```
Marker name   Chromosome      Position        1048-8R 964a-46 Giftgi
FcChr1Ap11111 1A      11111   CC      AC      AA
FcChr1Ap22222 1A      22222   GG      GC      GG
FcChr1Ap33333 1A      33333   TA      AA      GA
```

4.1.3 3. Genotype Flat-file

A tab delimited data file where each line is a genotypic call. Expected columns: (1) Marker name, (2) Chromosome Name, (3) Position on Chromosome, (4) Sample Name, (5) Genotype call.

```
Marker name   Chromosome      Position        Sample name     Genotype call
FcChr1Ap11111 1A      11111   Ross    CC
FcChr1Ap11111 1A      11111   Prado   CC
FcChr1Ap11111 1A      11111   Ash     CC
FcChr1Ap11111 1A      11111   Piero   CT
FcChr1Ap11111 1A      11111   Tai     CC
FcChr1Ap11111 1A      11111   Beverly TC
FcChr1Ap11111 1A      11111   Argent  CC
FcChr1Ap11111 1A      11111   Trenus  TT
FcChr1Ap11111 1A      11111   Zapelli CC
FcChr1Ap11111 1A      11111   Amato   CG
```

4.2 Samples File

All formats require a separate samples file describing the germplasm assayed. This file is expected to be a tab-delimited file with the following columns: (1) Sample name in the genotypes file, (2) Sample name, (3) Sample accession, (4) Germplasm name, (5) Germplasm accession.

The next two columns are optional: (6) Germplasm type (otherwise it is currently assumed to be of type 'Individual' from the stock_type cv) and (7) Organism (this allows multiple organisms in your genotypes file, assuming they have all been aligned to the same genome. Otherwise, the default value is the organism you specified as an option).

```
Sample name   Sample_name     Sample_Accession        Germplasm_name  Germplasm_
↪Accession     Germplasm_Type  Organism
Ross  Ross_110201     Catsam1 Ross    Catgerm1        Individual      Felis catus
Prado Prado_110201    Catsam2 Prado   Catgerm2        Individual      Felis catus
Ash   Ash_110201      Catsam3 Ash     Catgerm3        Individual      Felis catus
Piero Piero_110201    Catsam4 Piero   Catgerm4        Individual      Felis catus
Tai   Tai_110201      Catsam5 Tai     Catgerm5        Individual      Felis catus
```

(continues on next page)

(continued from previous page)

```
Beverly       Beverly_110201  Catsam6 Beverly Catgerm6        Individual      Felis␣
↪catus
Argent        Argent_110201   Catsam7 Argent  Catgerm7        Individual      Felis␣
↪catus
Trenus        Trenus_110201   Catsam8 Trenus  Catgerm8        Individual      Felis␣
↪catus
Zapelli       Zapelli_110201  Catsam9 Zapelli Catgerm9        Individual      Felis␣
↪catus
Amato Amato_110201 Catsam10 Amato Catgerm10 Individual Felis catus
```

CHAPTER 5

Demonstration

This demonstration will walk you through loading the sample files that come with the module. This is meant to show you the full process of loading a VCF file using this module and also as a means to evaluate the module.

Warning: These instructions should only be followed on a DEVELOPMENT site. Data will be inserted into your database.

5.1 Step 1: Preparation

5.1.1 Organism

Before loading a VCF file you need to ensure your have a chado organism record for the species the data is from. This can be done through the administrative interface by navigating to Content > Tripal Content > Add Tripal Content and clicking on Organism. For our sample file we need to add *Citrus sinensis* as they did in the Tripal Tutorial: Organism.

Note: The system does handle using a different species for the genomic backbone and variants then for the germplasm the data is associated with. For example, when genotyping wild species you will often align your data against a cultivated reference. In this case, you would supply the cultivated species for the `--organism` parameter and then the wild species would be indicated with each individual.

5.1.2 Genome

Next, we need to import the genome our genotypic data was aligned to. This can be done through the built-in Tripal GFF3 and/or FASTA importers as shown in the Tripal Tutorial: Genomes & Genes. Our genotypic data is aligned to the Citrus sinensis scaffold00001 imported in the linked Tripal tutorial.

5.1.3 Project

All genotypic data points from a single file are grouped using a chado project. To create a chado project go to Content > Tripal Content > Add Tripal Content and then select "Project" under General.

- **Name**: Citrus Demonstration Genotypic Data
- **Description**: This project contains demonstration data imported via the University of Saskatchewan, Pulse Bioinformatcis Genotypes Loader. This data is not real and should not be used in analysis.

5.2 Step 2: Prepare the Files

This has already been done for you with the VCF file at `sample_files/sample.vcf` and the associated germplasm samples file at `sample_files/samples.list`. For importing your own data, any VCF file following the VCF 4+ specification. The samples file then describes each of the germplasm samples in the VCF file with one row per sample. Full information on these file formats is available under "File Formats".

5.3 Step 3: Import genotypic data

Now we bring all that preparation into a single command to start the import process.

```
cd $DRUPAL_ROOT/sites/all/modules/genotypes_loader/sample_files
drush load-genotypes sample.vcf samples.list --organism="Citrus sinensis" \
  --variant-type="SNP" --marker-type="genetic_marker" \
  --project-name="Citrus Demonstration Genotypic Data"
```

Note: You will be prompted for your database user password during this process.

CHAPTER 6

Data Storage

Genotypic data is stored through use of a custom table (genotype_call) created by this module. This table provides a centralized, relational table which pulls all the information for a given genotypic call (marker assay result on a given germplasm for a specific project) together in a single record. It also supports flexible storage for all metadata associated with a genotype assay result through a PostgreSQL JSONB metadata column. We went with this backwards compatible approach to make supporting large genotypic datasets more efficient then chado alone. For more information on our schema and the reasons we went with this approach see *our schema documentation*.

Note: This loader stores data as expected by the ND Genotypes.

6.1 Chado Schema and Extensions

There are currently two ways to store your genotypic data in Chado v1.3 with this module providing a third, more efficient way. You can see a comparison of the various methods below which should make it clear why we've gone with the storage method we have.

6.1.1 Comparison of Methods

Method	Name	Custom Tables	Supports Meta-data	# Tables	Comments
1	ND Experiment	No	Yes	14	Not suitable beyond 3 million genotype calls.
2	Stock Genotype	No	No	10	A good alternative if you don't want to use custom tables but have a lot of data. Similar efficiency to Method #2 but less support for meta-data.
3	Genotype Call	Yes	Yes	10	Most efficient; although it touches the same number of tables as Method #3 there are less records per genotype call

All three methods store Markers & Variants in the same way. For the purposes of this module, a variant is a location on the genome where variation has been detected and has a type of SNP, MNP, Indel, etc. A marker then indicates which method the genotype calls associated with it were determined by. For example, you may have a variant on Chromosome 1 at position 45678 that you detected variation through two different methods. Each method would be indicated as a marker and all the genotype calls detected by that method would be attached to the appropriate marker and not directly to the variant. This has been determined necessary since the level of trust and how you interpret any quality meta-data will depend on the method.

6.1.2 Our Method: Custom Genotype Call Table

Now, lets consider the same example as in Method 1 (one VCF line with three alleles and six samples):

# Records	Tables	Example	Explanation
2	feature	"LcChr1p555" and "LcChr1p555 GBS Marker"	One each for variant and marker where the variant may already exist.
2	featureloc	Chr1:554-555 for each.	Locate each of the variant and marker on the chromsome.
1	feature_relationship	"LcChr1p555 GBS Marker" is_marker_of "LcChr1p555"	Link the marker and variant.
6	genotype_call	All Foreign Keys with the exception of any quality information you want to store in the meta-data column	This links the marker, variant, allele call, stock and project all in one and stores any addition quality information in the meta-data column.

Total: 11 records per line in a VCF with only 6 stocks and 3 alleles per variant.

Notice how efficient this method is. This is because (1) most of the foreign key connections are taking place in a single table (genotype_call) and (2) there now only needs to be a single record in the genotype table for "AA" rather than one record per marker using the previous method. For further comparison, the same 100,000 line VCF file would now only take 1,100,000 records to store not including the records for your chromosomes, which already exist, those for your stocks, only 6 per file, and those for you alleles (genotype table), which likely already exist. Furthermore, storing meta-data doesn't increase the number of records like it would in the first method.

Note: For more information about the two methods supported by core Chado, see the ND Genotypes documentation on schema.
