

File formats for polyploid genotypes

Lindsay Clark, University of Illinois, Urbana-Champaign

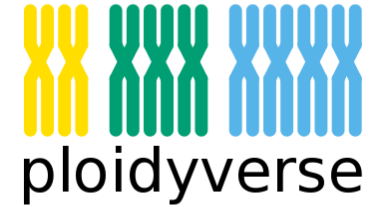
Excellence in Breeding meeting, CIP, 8 May 2019

See <https://lvclark.github.io> for copies of my presentation materials

Discussion points

- ▶ What information needs to be in the file?
 - ▶ Probabilities and likelihoods?
 - ▶ SNPs vs. haplotypes?
 - ▶ Sample metadata?
- ▶ How much of a concern is file size?
- ▶ Practicality from a software perspective

Ploidyverse



- ▶ Loose organization of scientists working on polyploid marker analysis
- ▶ Initiated by Paul Blischak at Arizona State
- ▶ Want to set some standards to make sure our software is interoperable
- ▶ Common datasets for testing and benchmarking software
- ▶ GitHub organization with R packages in development:
<https://github.com/ploidyverse/>
- ▶ Website: <https://ploidyverse.github.io/index.html>
- ▶ Discussion group:
<https://groups.google.com/forum/#!forum/ploidyverse>

Welcome to the Ploidyverse!



The Ploidyverse is an integrated collection of R packages for the analysis of mixed-ploidy genetic data. The project is still in the developing stages, but we will be updating this site and the associated GitHub repos as we finish each part of the project, so please stay tuned!

The `ploidyverse` package is the main R package responsible for installing and managing all other packages in the Ploidyverse. It can be installed from GitHub using `devtools`, as demonstrated below:

```
# Install devtools if you don't already have it
install.packages("devtools")

# Then install and load the ploidyverse package
devtools::install_github("ploidyverse/ploidyverse")
library(ploidyverse)
```



ploidyverse

Integrated software for the analysis of mixed-ploidy genomic data

<https://ploidyverse.github.io>

Repositories 3

People 7

Teams 0

Projects 0

Settings

Find a repository...

Type: All

Language: All

Customize pins

New

ploidyverseVcf

Functions and methods for working with VCFs in the ploidyverse

C++ ★ 3 GPL-3.0 Updated 11 days ago



ploidyverse.github.io

Ploidyverse website

HTML ★ 1 CC-BY-SA-4.0 Updated 11 days ago



ploidyverse

Install and manage packages in the ploidyverse

R ★ 1 GPL-3.0 Updated on Oct 3, 2018



Top languages

C++ R HTML

People










7 >



Invite someone

This group does not have a welcome message.

[Add welcome message](#)

-  Haplotypes in our shared VCF format haplotype ploidyverseClasses vcf
By me - 9 posts - 7 views Mar 22
-  moving forward with the ploidyverse
By Paul Blischak - 1 post - 5 views Jan 21
-  Meeting at PAG
By Paul Blischak - 15 posts - 18 views Jan 10
-  New VCF-style file format
By Paul Blischak - 11 posts - 35 views 11/2/18
-  **Link to PAG abstract draft (1)**
By me - 3 posts - 16 views 10/25/18
-  Anyone else attend PAG? (1)
By me - 1 post - 12 views 10/6/18
-  Website
By Paul Blischak - 2 posts - 10 views 10/6/18
-  GitHub Organization
By Paul Blischak - 9 posts - 19 views 9/28/18
-  Benchmarking Datasets
By Paul Blischak - 6 posts - 15 views 9/12/18

Variant Call Format (VCF)

- ▶ Standard for representing SNP genotypes in bioinformatics workflows
- ▶ Tab-delimited text, with binary version and tools for rapid access
- ▶ Self-documenting; header lines can store any metadata
- ▶ For every marker, stores name, chromosome, position, alternative allele, quality, any other data
- ▶ Stores most probable genotype, but can also include read depth, likelihoods and probabilities of all possible genotypes, anything else

Example from VCF specification

<https://samtools.github.io/hts-specs/VCFv4.3.pdf>

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference-file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##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=A,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 NA00001 NA00002 NA00003
20 14370 rs6054257 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:
20 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
20 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
20 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
20 1234567 microsat1 GTC G,GTCT 50 PASS NS=3;DP=9;AA=G GT:GQ:DP 0/1:35:4 0/2:17:2 1/1:40:3
```

```

##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##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=A,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	NA00001	NA00002	NA00003
20	14370	rs6054257	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:...
20	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
20	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
20	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
20	1234567	microsat1	GTC	G,GTCT	50	PASS	NS=3;DP=9;AA=G	GT:GQ:DP	0/1:35:4	0/2:17:2	1/1:40:3

Markers

Samples

Representing polyploids in VCF: The GT (genotype) field

- ▶ 0 represents the reference allele
- ▶ 1 represents the first alternative allele, 2 the second alt. allele, etc.
- ▶ 0/0 is a diploid homozygote for the reference, 0/1 a heterozygote
- ▶ ./. is a diploid with missing data
- ▶ A tetraploid simplex genotype could be 0/0/0/1, missing is ./././.
- ▶ If alleles are phased across markers, it can be indicated by replacing / with |
- ▶ Polyploids already supported by VCF, utilized by GATK software

```
FORMAT      NA00001      NA00002      NA00003
GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:..
GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3 0/0:41:3
B GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2 2/2:35:4
GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
GT:GQ:DP      0/1:35:4      0/2:17:2      1/1:40:3
```

Representing polyploids in VCF: The GL (genotype likelihood) and GP (genotype posterior probability) fields

- ▶ Useful since genotypes are uncertain in polyploids without very high read depth.
- ▶ For every marker*sample, have a comma-separated list of floating point numeric values
- ▶ One value for every possible genotype
- ▶ VCF specification indicates the ordering of genotypes for any ploidy and number of alleles.

Description of
genotype
ordering from
VCF specification

Example values for a tetraploid:

GT:GP 0/0/0/1:0.033,0.684,0.233,0.045,0.005

Alternatively, the same can be achieved recursively with the following pseudocode:

```
Ordering(P, N, suffix=""):  
  for a in 0...N  
    if (P==1) println str(a) + suffix  
    if (P>1) Ordering(P-1, a, str(a) + suffix)
```

Conversely, the index of the value corresponding to the genotype $k_1 \leq k_2 \leq \dots \leq k_P$ is

$$\text{Index}(k_1/k_2/\dots/k_P) = \sum_{m=1}^P \binom{k_m+m-1}{m}$$

Examples:

- for $P=2$ and $N=1$, the ordering is 00,01,11
- for $P=2$ and $N=2$, the ordering is 00,01,11,02,12,22
- for $P=3$ and $N=2$, the ordering is 000, 001, 011, 111, 002, 012, 112, 022, 122, 222

Representing polyploids in VCF: The AD (allele depth) field

- ▶ Shows the sequence read depth for the reference allele and every alternative allele.
- ▶ These are the data that are used to make the genotype calls.
- ▶ Useful to keep in final output so that anyone can independently assess genotype quality, or call the genotypes with a different method.

Example values for a tetraploid with one alt. allele:

GT:AD:GP 0/0/0/1:13,6:0.033,0.684,0.233,0.045,0.005

SNPs vs. haplotypes?

- ▶ Although VCF is typically used for SNP data, alleles can be any length, allowing representation of alleles as haplotypes rather than single nucleotides
- ▶ Using reduced representation DNA sequencing (GBS, RAD) SNPs can be grouped into the tags they came from
- ▶ Less ambiguity
- ▶ Haplotypes can facilitate the separation of homeologous loci
- ▶ Downside: Downstream software might not be designed for haplotypes

POS	REF	ALT
3668	A	G
3670	G	A
3673	C	T

Vs.

POS	REF	ALT
3668	AAGCAC	GAGCAC, AAACAT, AAACAC

Options discussed in ploidyverse group:

(1) SNPs with phasing

- ▶ VCF has one line per SNP
- ▶ Pipe symbol and phase set (PS) field indicate haplotypes
- ▶ Advantage: people are used to working with SNPs
- ▶ Disadvantage: we only see phasing for most probable genotype

POS	REF	ALT	FORMAT	Sample1
3668	A	G	GT:PS	1 0:3668
3670	G	A	GT:PS	0 1:3668
3673	C	T	GT:PS	0 1:3668



(Seems preferred by most ploidyverse members)

Options discussed in ploidyverse group: (2) Haplotypes and multiallelic genotypes

- ▶ VCF has one line per tag location
- ▶ Multiple alternative alleles, one per non-reference haplotype
- ▶ Advantage: information represented fully
- ▶ Disadvantages:
 - ▶ Not a lot of existing software to work with multiallelic genotypes.
 - ▶ Number of possible genotypes becomes enormous, and GP/GL fields will need one value for each possible genotype in each sample

Genotype order for diploid with four alleles:

- 0: 0/0 5: 2/2
- 1: 0/1 6: 0/3
- 2: 1/1 7: 1/3
- 3: 0/2 8: 2/3
- 4: 1/2 9: 3/3

POS	REF	ALT	FORMAT	Sample1
3668	AAGCAC	GAGCAC,AAACAT,AAACAC	GT:GP	1/2:0,0,0.04,0,0.94,0.02,0,0,0,0

Options discussed in ploidyverse group:

(2a) Only display relevant GP/GL values

- ▶ VCF has one line per tag location
- ▶ Multiple alternative alleles, one per non-reference haplotype
- ▶ Only show GP/GL values above certain threshold, number genotype
- ▶ (Gabriel Margarido's idea)
- ▶ Advantage: Information represented fully without making file huge
- ▶ Disadvantage: Outside of current VCF specifications

Genotype order
for diploid with
four alleles:

0: 0/0	5: 2/2
1: 0/1	6: 0/3
2: 1/1	7: 1/3
3: 0/2	8: 2/3
4: 1/2	9: 3/3

POS	REF	ALT	FORMAT	Sample1
3668	AAGCAC	GAGCAC, AAACAT, AAACAC	GT:GP	1/2:2=0.04, 4=0.94, 5=0.02

Options discussed in ploidyverse group:

(3) Pseudo-biallelic haplotypes

- ▶ VCF has one line per haplotype
- ▶ This is most similar to how polyRAD handles genotypes
- ▶ Advantage: Genotypes expressed in terms of allele copy number, easily converted to numeric (posterior mean) format
- ▶ Disadvantage: Can't solve for multiallelic genotype probabilities

POS	REF	ALT	FORMAT	Sample1
3668	NANCAN	AAGCAC	GT:GP	0/0:1,0,0
3668	NANCAN	GAGCAC	GT:GP	0/1:0.02,0.94,0.04
3668	NANCAN	AAACAT	GT:GP	0/1:0.04,0.94,0.02
3668	NANCAN	AAACAC	GT:GP	0/0:1,0,0

Working with VCFs in R: VariantAnnotation

- ▶ Bioconductor package
- ▶ Powerful and flexible interface where you can decide which fields and genomic regions to import
- ▶ Can read and write VCF files in chunks to conserve memory
- ▶ Useful for any R package that would want to import or export in VCF format

ploidyverseVcf R package: Generating and utilizing ploidyverse VCFs

- ▶ In development at <https://github.com/ploidyverse/ploidyverseVcf>
- ▶ Builds on Bioconductor's VariantAnnotation R package for working with VCFs
- ▶ For use by other ploidyverse package developers
- ▶ Includes a specification document describing how a VCF should be formatted for use in the ploidyverse
 - ▶ AD and GP fields for quantifying genotype uncertainty
 - ▶ Suggestion to include metadata about samples (species, ploidy), reference genome, pipelines used for SNP discovery, genotype calling, imputation, etc.
 - ▶ Instructions for use in reference-free pipelines (include full RAD tag)

Variant Call Format specification for the ploidyverse

The Variant Call Format, or VCF, is the standard in bioinformatics for storing information about DNA sequence variation. One VCF file can contain genotypes across many loci and individuals. Because VCF is highly flexible, widely used, and self-documenting, we adopt it here as the primary format for transferring data among ploidyverse packages, and between the ploidyverse and other software.

VCF is stored as tab-delimited text, with a corresponding binary format called BCF. Here we describe guidelines for VCF files imported into or exported from ploidyverse packages, and generally set standards for storing genotype data from polyploid organisms. These guidelines are meant to make it easier to transfer data among software and researchers, rather than impose restrictions. If your needs deviate in some way from what is described here and you would like to suggest some changes, you may file an issue or pull request on GitHub, or start a discussion on the ploidyverse Google Group.

Additionally, the `VariantAnnotation` package, from the Bioconductor collection of R packages, implements flexible S4 classes for storing any or all data from a VCF file within an R object, and has functions for reading and writing VCF files. The `ploidyverseVcf` R package extends `VariantAnnotation` and is designed to assist package developers and users with meeting the ploidyverse VCF guidelines.

Guiding principles

VCF data within the ploidyverse should:

- Be fully compatible with the [current VCF specification](#).
- Be self-documenting. Someone 50 years from now should be able to open up the file and understand what is going on and how to use the data.
- Be adapted for use with non-model organisms. There may be no reference genome, or a reference genome for a different species from the one being studied.
- Provide allelic read depths and posterior probability estimates for all genotype calls. Allele dosage will be uncertain without very high depth sequencing, and that uncertainty must be quantified in the file.

https://github.com/ploidyverse/ploidyverseVcf/blob/master/inst/doc/ploidyverse_VCF_specification.md

(To be updated once an ideal format is decided)

Using ploydyverseVcf in your R package

Lindsay V. Clark, University of Illinois, Urbana-Champaign 07 April 2019

Purpose of this document

This document is intended for R package developers who wish to utilize S4 classes from the ploydyverse in order to make their package interoperable with other ploydyverse packages. It assumes basic familiarity with the process of creating an R package.

How to indicate dependency on `ploydyverseVcf`

The first question you should ask yourself is, will the ability to read and write VCFs be mandatory for using your package? Or, will users frequently want to import or export data in a different format, and not care about incorporating other ploydyverse software into their workflow?

`ploydyverseVcf` depends on the Bioconductor package `VariantAnnotation`. Like many Bioconductor packages, `VariantAnnotation` can be unavoidably cumbersome to install. We on the ploydyverse core team believe that it is worthwhile for its flexibility in importing, storing, and exporting genomic variants and metadata, but we want to minimize how much we force it onto people who don't want it.

Option A: mandatory installation of `ploydyverseVcf`

In this case, you will indicate dependency on `ploydyverseVcf` in the standard way that you would for any R package. In your `DESCRIPTION` file, list `ploydyverseVcf` on the `Imports` line.

```
Imports: ploydyverseVcf
```

https://github.com/ploydyverse/ploydyverseVcf/blob/master/inst/doc/package_developer_guide.md

ploidyverseVcf R package: VCF validation



- ▶ Functions to check a VCF object in R and make sure it meets ploidyverse specifications before output to file
- ▶ Adds tags to the file header:
 - ▶ "File valid for calling genotypes with ploidyverse software."
 - ▶ "File contains genotype calls from ploidyverse software."
 - ▶ "File meets ploidyverse standards for data archiving."

```
myvcf <- readvcf("mygenotypes.vcf")  
myvcf <- markvalidity(myvcf)  
writevcf(myvcf, "mygenotypes.vcf")
```

ploidyverseVcf R package: Accessors

- ▶ Function to take a data frame (i.e. spreadsheet) of sample metadata and add it to the VCF header
- ▶ Function to add information about software used for generating the VCF to the VCF header

```
myVcf <- readVcf("mygenotypes.vcf")
sampleinfo(myVcf) <- data.frame(row.names = c("Sample1", "Sample2"),
                                Species = rep("Solanum tuberosum", 2),
                                Ploidy = rep("4x", 2))
software(myVcf) <- rbind(software(myVcf),
                        data.frame(row.names = "GenotypeCalls",
                                    Software = "polyRAD",
                                    Version = "1.1",
                                    Model = "PopStructure",
                                    Description = "Genotype calling with polyRAD"))
writeVcf(myVcf, "mygenotypes.vcf")
```



ploidyverseVcf R package: Utility functions for multiallelic genotypes

- ▶ Implemented with Rcpp, so that they can be used from R or from compiled code
- ▶ If we are working with multiallelic genotypes, where loci vary in how many alleles they have, compiled code will be necessary since loops are very slow in R
- ▶ Functions include
 - ▶ Multinomial and Dirichlet-multinomial probability
 - ▶ Enumeration and indexing of genotypes in VCF order
 - ▶ Enumerate gametes and generate self-fertilization matrix
 - ▶ Convert genotype probabilities to allele copy number probabilities
- ▶ To-do: Conversion functions for whatever GP format we decide to go with

```
> enumerateGenotypes(4, 3)
      [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    1
[3,]    0    0    1    1
[4,]    0    1    1    1
[5,]    1    1    1    1
[6,]    0    0    0    2
[7,]    0    0    1    2
[8,]    0    1    1    2
[9,]    1    1    1    2
[10,]   0    0    2    2
[11,]   0    1    2    2
[12,]   1    1    2    2
[13,]   0    2    2    2
[14,]   1    2    2    2
[15,]   2    2    2    2
```

Discussion

- ▶ What are your preferences and concerns in terms of file format?
- ▶ What information is downstream software going to need from a VCF?
- ▶ What information is important for archiving?
- ▶ Is there additional computational infrastructure (e.g. software like ploidiverseVcf) that would make this easier to work with?