RUNNING IGGPIPE TO GENERATE IGG MARKERS

Ted Toal <twtoal@ucdavis.edu>

Table of Contents

1. **If not done already, download IGGPIPE files** 1
2. **If not done already, install IGGPIPE** 1
3. **Requirements for running IGGPIPE** 1
4. **Running IGGPIPE** 2
5. **Post-processing tools** 7
    5.1. **For problems and help:** 9
6. **Tables** 9

This describes how to run the IGGPIPE software to analyze two or more genome sequences and produce a file of candidate IGG marker primer pairs for amplifying length-polymorphic regions of those genomes.

1. If not done already, download IGGPIPE files

1. Browse to https://github.com/BradyLab/IGGPIPE

2. At bottom of right column on screen, click "Download ZIP" and choose a place to put it on your computer.

3. Unzip the zip file on your computer.

4. Rename the unzipped folder from "IGGPIPE-master" to just "IGGPIPE".

2. If not done already, install IGGPIPE

- Look inside the downloaded IGGPIPE folder on your computer for file INSTALL.pdf or INSTALL.html and open either one and follow the instructions.

3. Requirements for running IGGPIPE

1. **Obtain genome FASTA files**

   IGGPIPE uses as input two or more FASTA files of complete genome sequences (at least one of which is assembled into chromosomes, the other(s) can be in the scaffold state). You should know ahead of time **which** genomes you are comparing. If you haven’t already, download their genomic FASTA files, which are required.

2. **Work from the command line**

Most work here is done from the command line, by opening the Terminal application. Commands will be shown here and you may be able to get by with no knowledge of the command line, other than knowing how to start it (by starting the Terminal app on the Mac). However, if you to familiarize yourself with some of the basic command line commands, you may want to take a look at a short tutorial such as one of these:

http://www.davidbaumgold.com/tutorials/command-line

http://mac.appstorm.net/how-to/utilities-how-to/how-to-use-terminal-the-basics

Or a longer tutorial such as this one from UC Davis:

http://korflab.ucdavis.edu/Unix_and_Perl/current.html

3. **Work in the IGGPIPE main directory unless otherwise instructed**

   While working from the command line to install IGGPIPE, most of the time you will be in the IGGPIPE main directory, unless instructed otherwise. If you unzipped the IGGPIPE zip file in your Documents folder, you would change into the IGGPIPE directory with this command:

   ```
   cd ~/Documents/IGGPIPE
   ```

4. **Know how to use a plain text editor and have one available**

   You must have a plain text editor you know how to use. If nothing else, the Mac "TextEdit" program will work (use Plain Text format). The free open-source TextWrangler program is strongly recommended, available from the Apple App Store (Applications, App Store) or from:

   http://www.barebones.com/products/textwrangler

4. Running IGGPIPE

1. **Assign a single capital letter name to each genome**

   IGGPIPE makes frequent use of a single capital letter to refer to a genome. For example, filenames and tab-separated file column names use such letters. Choose a single capital letter you will use to represent each of your genomes. For example, I used H=Heinz and P=pennellii for some of my testing.

2. **Copy allParameters.mytemplate file to new file allParameters.XY**

   IGGPIPE has a number of parameters that must be set to the values you desire. These are contained in plain text files whose name starts with "allParameters". During IGGPIPE installation, a file should have been created named **allParameters.mytemplate**. This is a template containing some parameters already set correctly for your system. You need to copy and edit this file to change other parameters to the settings you want. You should make a new parameter file for each different set of settings you want to run with IGGPIPE. For example, each new set of genomes would have a different parameter file. Also, if you decided to make two different sets of markers from the same genomes, using two different parameter settings, you might make two different parameter files. Here, we assume to begin with that you are only running your genomes with a single set of parameters, and we will name the parameter file allParameters.XY where X and Y are the genome capital letters you chose. It is convenient to include those letters in the file name, to help you keep multiple parameter file straight. **Copy allParameters.mytemplate to allParameters.XY** (substituting your genome letters for X and Y), using either the Mac's Finder or via the command line. For example:

```
cd ~/Documents/IGGPIPE    (or whatever is appropriate for your system)
cp allParameters.mytemplate allParameters.HP    (here, X=H, Y=P)
```

3. **Open the allParameters.XY file in your plain text editor**

   Open the new **allParameters.XY** file created above in your plain text editor for editing. If you are in a hurry, you don't need to read anything in the file, but can simply **search for "# "**, which are comment lines marking items that may need to be changed. Each "#" comment says whether it must be changed, might need to be changed, or probably will never need to be changed, etc. Many of these items typically will never need to be changed, so the actual number of changes that need to be made is smaller than it might first appear. Some of the parameters have already been set correctly during installation of IGGPIPE. However, it is recommended that rather than hurrying, you take time to read through the file, as the comments explain the purpose of each parameter, and you will want to know this information, at least for key parameters, to select the right parameter values for your needs.

4. **Search for "# " and set parameters to desired values**

   Search for "# " in the allParameters.XY file and check each one to see if it needs to be changed. If so, set it to the value you desire. Parameters you will definitely want to review and consider are:

   a. K

   b. N_GENOMES

   c. GENOME_NUMBERS

   d. GENOME_1, GENOME_2, etc.

   e. LMIN

   f. DMAX

   g. AMIN and AMAX

   h. ADMIN and ADMAX

   i. NDAMIN

   j. OVERLAP_REMOVAL

   k. EPCR_MAX_DEV

   l. EPCR_MAX_MISMATCH and EPCR_MAX_GAPS

   After finishing changes, save the modified allParameters.XY file.

5. **Check Primer3 settings in primer3settings.txt**

The file **primer3settings.txt** contains parameter settings for Primer3, which is used to generate the actual primers. This file should have been edited during IGGPIPE installation to make any obvious changes you might need for your primers. However, it is possible that for a specific run of IGGPIPE, you might want to use different settings. If so, edit primer3settings.txt and make the desired changes. (You may want to save a backup copy of the original version).

6. **Run IGGPIPE with the command "make PARAMS=allParameters.XY ALL"**

The IGGPIPE software consists of multiple software applications that progressively analyze the genome sequence data and eventually produce candidate IGG marker primers. The task of running all this software has been automated using a "Makefile", which is a file with that name containing commands formatted correctly for reading the allParameters.XY parameter file and running the software applications. The Makefile is applied by using the application named "make", which was installed when IGGPIPE was installed, if it didn't already exist.

A big advantage of using "Makefile" and "make" is that if something goes wrong (and unfortunately, it probably will), the portion of the work successfully completed is not lost, and does not need to be repeated. This is important because it can take quite a long time to run genomes all the way through the IGGPIPE software. Depending on your computer speed and memory, it can take hours or even days.

You run "make" from the command line to run IGGPIPE. If an error occurs, "make" will stop, and an error message should be visible. If you are lucky, you will have no errors. I do not yet have enough experience running IGGPIPE on different genomes to anticipate how often errors will occur, or what will cause them. Please email me with information about errors, and their resolution if you were able to resolve them. I'll try to make improvements to IGGPIPE in error handling and in its input data format flexibility to try to prevent errors.

After the allParameters.XY file is edited and ready to go, **run the IGGPIPE pipeline from the IGGPIPE directory as follows**:

```
cd ~/Documents/IGGPIPE    (or whatever is appropriate for your system)
make PARAMS=allParameters.XY ALL    (replacing with your allParameters name)
```

If "make" stops with the message **ALL files are up to date**, it has completed the analysis successfully. Otherwise, look for an error message and try to diagnose it. You should first check to see if the common unique k-mers file named *isect.kmers* was created in the *Kmers* subfolder of the output subfolder. This is created early in the pipeline, and you should always check it to see how many common unique k-mers were found. This can be done with the *wc -l* command, which counts lines in a file, since the *isect.kmers* file contains one unique k-mer per line. The command line would look like this:

```
wc -l myOutFolder/Kmers/isect.kmers
```

If you have plenty of k-mers (say around 8 million or more) but not too many (say 40 million or more), that is good. If you feel you have too few k-mers, then you should increase K by 1 and try again, and if you have too many, decrease it by 1 and try again. You can run the pipeline only up until isect.kmers is created by using this command:

```
make PARAMS=myFilename kmerIsect
```

If there are plenty of common unique k-mers, you should next check the number of LCRs detected next, which you can again do with *wc -l*, since the LCRs_* file contains one LCR per line. The command line would look like this:

```
wc -l myOutFolder/LCR_*
```

If you had too few common k-mers you might also have too few LCRs. A million or more LCRs would be nice. The fewer you have, the fewer markers you are likely to get. If there are too few, check the log file output when findLCRs ran. It will show the number of common unique k-mers for each sequence ID, then the number remaining after it enforces Dmin, Lmin, and kMin on the reference genome. If these numbers fall dramatically towards 0, it indicates that either there are no good LCRs between the two genomes, or the parameters are too strict.

If there were plenty of LCRs, you should next check the number of non-overlapping IndelGroups found, which you can again do with *wc -l*, since the IndelGroupsNonoverlapping_* file contains one Indel Group per line. The command line would look like this:

```
wc -l myOutFolder/IndelGroupsNonoverlapping_*
```

Several thousand IndelGroups is nice to have.

If you run into problems, determine which step failed and check the input file(s) for that step to make sure everything looks ok.

If you fix something and want to retry running IGGPIPE, all you have to do is enter the same "make" command again. The "make" program automatically skips pipeline steps that don’t need to be repeated because the input files for those steps have not changed, and the output files were made with success previously. Therefore, it will normally resume by repeating the same step that failed and caused it to halt with an error. If the error still exists, it will halt again with the same error message. Otherwise, it will continue until it reaches the end successfully, or until another error happens. Therefore, each time you try to re-run the pipeline, you are just entering the command:

```
make PARAMS=allParameters.XY ALL    (replacing with your allParameters name)
```

Please email me with any bugs you find and perhaps fix. I am available to a limited extent via email, for a while, to try to assist in diagnosing problems.

If at any point you want to remove all files already generated and start anew, you can do that with this command:

```
make PARAMS=allParameters.XY CLEAN=1 ALL    (replacing with your allParameters name)
```

You can also run individual steps of the pipeline. To see how, use this command to get more complete usage information for running "make":

```
make usage
```

Again, your final goal is to have "make" stop with the message **ALL files are up to date**

7. **Open marker output files and inspect the results**

Unless you specifically changed the parameters otherwise, you will find the output files from the IGGPIPE run in a subdirectory of the IGGPIPE directory named something like outXY14, where

XY are the genome letters you chose, and 14 is the value of K for the k-mer size, which was one of the parameters in the parameter file.

Within that output subdirectory, you will find a number of files. Unless you changed the parameter settings otherwise, the file names are very long and cumbersome, because they include parameter values in them. You may want to copy files to a shorter name to work with them. The main ones of interest (using "*" in place of the long text), again assuming you didn't change their names in the parameter file, are:

a. MarkerCounts_*.plot.pdf is a pdf file showing plots of marker counts on chromosomes

b. MarkerDensity_*.plot.png is a png image file showing plots of marker density and position

c. MarkerOverlapping_*.tsv is a tab-separated file containing the candidate IGG markers, Table 1

d. MarkerNonoverlapping_*.tsv is a tab-separated file containing a non-overlapping version of the above, Table 1

Examine the .pdf and .png files. The .tsv files can be loaded into Excel to look at the markers, and they can also be post-processed (see below) to change them into other formats. The meaning of "overlapping" and "non-overlapping" should be clear from the explanation of the parameter OVERLAP_REMOVAL in the comments in allParameters.XY. The two .tsv files contain the IGG marker positions and primer sequences, among other things.

Several other ".tsv" tab-separated output files exist:

a. MarkerErrors_*.tsv contains candidate markers rejected because e-PCR failed, Tables 1, 2

b. NonvalidatedMarkers_*.tsv contains candidate markers not yet subjected to e-PCR, Table 1

c. IndelGroupsOverlapping_*.tsv contains overlapping regions of LCRs satisfying parameters for a possible IGG marker, Table 3

d. IndelGroupsNonoverlapping_*.tsv is like above but non-overlapping regions as per parameter OVERLAP_REMOVAL, Table 3

e. LCRs_*.tsv contains common unique k-mers assigned to locally conserved regions (LCRs), Table 4

f. BadKmers_*.tsv contains common unique k-mers rejected from assignment to any LCR, Table 4

Tables describing each column in each file type are at the end of this document.

Besides the sample parameter file "allParameters.template", there are several more sample "allParameters" parameter files in the subfolder "allParameters". These were used for testing IGGPIPE with various genomes. The subfolder "allParameters" was made to keep the files better organized and keep the root folder less messy. You may want to adopt the same strategy and put your "allParameters" files in the "allParameters" subfolder or in a subfolder you make yourself. When you do that, you run "make" by specifying the full path to the file, e.g.:

```
make PARAMS=myParamFolder/allParameters.XY ALL
```

5. Post-processing tools

1. **Finding InDels**

   An R program that is NOT run as part of the pipeline when the *make … ALL* target is built, but which can be run using *make … InDels*, is able to read a file of LCRs, non-overlapping InDelGroups, or non-overlapping Markers, extract the DNA sequences from the genomes in each LCR or Marker region and align them, then locate all InDels in the aligned sequences and write their positions to a file. The program is called alignAndGetIndels.R. You should already have set its input file name in your "allParameters" parameter file. Run it as follows:

   ```
   make PARAMS=allParameters.XY InDels    (replacing with your allParameters name)
   ```

   This produces a file in your output folder whose name ends with "indels.tsv", containing a table of all InDels found in the LCR or marker regions. Examine it to see the data it contains (see Table 5 of column descriptions).

2. **Plotting InDel information**

   Another R program that is NOT run as part of the pipeline when the *make … ALL* target is built, but which can be run using *make … plotInDels*, reads the InDels file produced by *make … InDels* and plots information from it in a pdf file. The program is called plotIndels.R. Run it as follows:

   ```
   make PARAMS=allParameters.XY plotInDels    (replacing with your allParameters name)
   ```

   This produces a file in your output folder whose name ends with "indels.pdf", containing plots of various InDel information. Examine it to see the plots it contains.

3. **Dot plots**

   The output file with the name "LCRs_*.tsv" (unless it was changed by you) contains locally conserved regions associated with common unique k-mers. It represents a whole genome alignment between the genomes used in IGGPIPE analysis. An R program, dotplot.R, is provided that can plot this data as a dot plot.

   This program is run by first copying the text file "dotplot.template" to a new name (e.g. dotplot.XY) and editing it to specify the parameters of the dot plot. Comments in the file describe each parameter. The program is then run from the command line with a command like this:

   ```
   cd ~/Documents/IGGPIPE    (or whatever is appropriate for your system)
   Rscript code/R/dotplot.R dotplot.XY    (or whatever name you gave the parameter file)
   ```

   When it finishes running, the dot plot output file can be found in the place and under the name specified in the parameter file. Use multiple parameter files with different settings to explore different regions of the genomes in greater resolution.

   The "dotplot.template" file is configured for generating a dot plot file using the LCRs generated via the allParameters.test.template configuration file.

   Besides the sample parameter file "dotplot.template" (which has settings for testing the IGGPIPE installation), there are several more sample "dotplot" parameter files in the subfolder "dotplot", that were used for plotting data from various genomes that IGGPIPE was tested with. You may want to put your own "dotplot" parameter files in subfolder "dotplot" or your own subfolder to keep them organized.

4. **Annotating marker files with other position data and producing GFF3 and GTF files**

You may want to make your marker data more conveniently available. For example, you might want to convert it to GFF3 file format so you can add a "marker" track to a genome browser. Or, you may have other genome position data that you would like to have associated with your marker data, such as a file giving positions of introgressions of one genome within another (you might want a column in the marker file showing which introgressions the marker was near). As another example, you might want to add a column in the marker file containing the names of the genes closest to the marker, and the distance to the genes. All of these situations and more can be handled by an R program, annotateMarkers.R, provided with IGGPIPE. The program can read and write files of type .tsv (tab-separated variable), .csv (comma-separated variable), .gff3 (general feature format), or .gtf (gene transfer format), all common formats used to hold genome browser track data or FASTA file annotation data. It can add, remove, edit, and rename columns. It can read two separate files and merge their data. It can convert from one of these file formats to another.

This program is run by first copying the text file "annotate.template" to a new name (e.g. annotateIntrogressions.XY or addGeneInfo.XY or makeGFF3.XY) and then editing it to specify the parameters for the annotation and/or file conversion. Comments in the file describe each parameter. The program is then run from the command line with a command like this:

```
cd ~/Documents/IGGPIPE    (or whatever is appropriate for your system)
Rscript code/R/annotate.R addGenes.XY    (or whatever name you gave the parameter file
```

When it finishes running, the output files can be found in the place(s) and under the name(s) specified in the parameter file.

Besides the sample parameter file "annotate.template" (which has settings for testing the IGGPIPE installation), there are several more sample "annotate" parameter files in the subfolder "annotate", with file names hinting at what they do, and comments at the start of each file describing what it does. It may be easier to copy one of these and modify it. You may want to put your own "annotate" parameter files in subfolder "annotate" or your own subfolder to keep them organized.

So, the idea is to use multiple parameter files with different settings to do different types of annotation and file conversion.

Some of the sample parameter files generate .gff3 files that can be added as a track to a genome browser, to display markers in the browser. Instructions for adding the track are given in comments at the start of the parameter file. Two marker files, one for Arabidopsis thaliana Col-0 vs. Ler-0 ecotypes, and the other for Solanum lycopersicum vs. Solanum pennellii genomes, were created to test IGGPIPE, and the marker files were converted to .gff3 files suitable for making a browser track. These files can be found in subdirectories of the "annotate" directory.

File formats can be finicky, especially .gff3 files. An incorrectly formatted file will cause problems with annotateFile.R. When you have problems, if you can submit an issue to the GitHub repository named "BradyLab/IGGPIPE", and attach or insert a copy of your parameter file, that would be helpful. A copy of the input data files would probably also be needed to debug problems, but GitHub does not allow files to be attached. You can email them to me, or find some other way to send them.

5.1. For problems and help:

- Post an issue on GitHub under BradyLab/IGGPIPE repository
- Contact me, Ted Toal, twtoal@ucdavis.edu [mailto:twtoal@ucdavis.edu]

6. Tables

Table 1. Columns in MarkersOverlapping_, MarkersNonoverlapping_, NonvalidatedMarkers_, MarkerErrors_ files; X,Y=chosen genome letters

Column	Description
NDA	Number of distinct amplicon sizes, in range NDAMIN..N_GENOMES
Xid	Genome X sequence ID
Xpct	Genome X percent of sequence ID length at which marker is located
XampLen	Genome X amplicon length
Yid	Genome Y sequence ID
Ypct	Genome Y percent of sequence ID length at which marker is located
YampLen	Genome Y amplicon length
YXdif	Difference in length between genomes X and Y amplicons, negative if genome X longer than genome Y
YXphase	Phase of amplicons between genomes X and Y, "+" if both amplicons run in same direction, "-" if opposite directions
prmSeqL	Left side or upstream primer sequence
prmSeqR	Right side or downstream primer sequence
prmTmL	Left side primer Tm
prmTmR	Right side primer Tm
prmLenL	Left side primer length
prmLenR	Right side primer length
XampPos1	Genome X amplicon starting (upstream) position
XampPos2	Genome X amplicon ending (downstream) position, XampPos2 always > XampPos1
YampPos1	Genome Y amplicon starting (upstream) position
YampPos2	Genome Y amplicon ending (downstream) position, YampPos2 > YampPos1 if YXphase is "+", < if "-"
kmer1	Common unique k-mer for left side primer region, canonical (exically smaller of k-mer and its reverse complement)
kmer1strand	N_GENOMES `"" and "-" characters for genomes 1..N_GENOMES. A ""` means k-mer 1 lies on the "+" strand in that genome, "-" means "-" strand.
kmer1offset	Offset in bp of outside (away from amplicon) edge of k-mer 1 from that end of the amplicon. A value of 0 means the amplicon and k-mer ends correspond, >0 means k-mer starts inside the amplicon, <0 means k-mers starts outside it.

Column	Description
kmer2	Common unique k-mer for right side primer region, canonical (exically smaller of k-mer and its reverse complement)
kmer2strands	Like kmer1strands, for k-mer 2.
kmer2offset	Like kmer1offset, for k-mer 2.
Xseq1	Genome X DNA sequence around left side primer region
Xseq2	Genome X DNA sequence around right side primer region
Yseq1	Genome Y DNA sequence around left side primer region
Yseq2	Genome Y DNA sequence around right side primer region

Table 2. Column reasonDiscarded in MarkerErrors_ files (see Table 1 for other columns)

reasonDiscarded	Description
found multiple	ePCR found multiple amplicons (expected reason)
not found	ePCR didn’t find amplicon (should never happen)
wrong seq id	ePCR sequence ID output is wrong (should never happen)
wrong pos	ePCR left and right position output is wrong (should never happen)
wrong posL	ePCR left position output is wrong (should never happen)
wrong posR	ePCR right position output is wrong (should never happen)

Table 3. Columns in IndelGroupsOverlapping_ and IndelGroupsNonoverlapping_ files; X,Y=chosen genome letters

Column	Description
kmer1	Common unique k-mer for left side primer region, canonical (lexically smaller of k-mer and its reverse complement)
kmer2	Common unique k-mer for right side primer region, canonical (exically smaller of k-mer and its reverse complement)
NDA	Number of distinct amplicon sizes, in range NDAMIN..N_GENOMES
Xid	Genome X sequence ID
Xpos1	Genome X position of upstream end of k-mer 1 on "+" strand
Xpos2	Genome X position of upstream end of k-mer 2 on "+" strand, Xpos1 < Xpos2 always
Xs1	Genome X k-mer 1 strand, "+" or "-"
Xs2	Genome X k-mer 2 strand, "+" or "-"
Xctg1	Genome X contig number within sequence Xid of contig containing k-mer 1
Xctg2	Likewise for k-mer 2, Xctg1 = Xctg2 always
XkkLen	Genome X distance from 5' end of k-mer 1 on "" strand to 5' end of k-mer 1 on "" strand
Xpct	Genome X percent of sequence ID length at which marker is located

Column	Description
Yid	Genome Y sequence ID
Ypos1	Genome Y position of upstream end of k-mer 1 on "+" strand
Ypos2	Genome Y position of upstream end of k-mer 2 on "+" strand, Ypos1 < Ypos2 if amplicon in X and Y genomes run in the same direction, > if opposite directions
Ys1	Genome Y k-mer 1 strand, "+" or "-"
Ys2	Genome Y k-mer 2 strand, "+" or "-"
Yctg1	Genome Y contig number within sequence Yid of contig containing k-mer 1
Yctg2	Likewise for k-mer 2, Yctg1 = Yctg2 always
YkkLen	Genome Y distance from 5' end of k-mer 1 on "" strand to 5' end of k-mer 1 on "" strand
Ypct	Genome Y percent of sequence ID length at which marker is located

Table 4. Columns in LCRs_ and BadKmers_ files; X,Y=chosen genome letters

Column	Description
(none, row name)	Common unique k-mer, canonical representation (the lexically smaller of k-mer and its reverse complement)
X.seqID	Genome X sequence ID
X.pos	Genome X position of upstream end of k-mer on "+" strand relative to start of X.seqID
X.strand	Genome X k-mer strand, "+" or "-"
X.contig	Genome X contig number within sequence X.seqID sequence of contig containing the k-mer
X.contigPos	Genome X position of upstream end of k-mer on "+" strand relative to start of X.contig
Y.seqID	Genome Y sequence ID
Y.pos	Genome Y position of upstream end of k-mer on "+" strand relative to start of Y.seqID
Y.strand	Genome Y k-mer strand, "+" or "-"
Y.contig	Genome Y contig number within sequence X.seqID sequence of contig containing the k-mer
Y.contigPos	Genome Y position of upstream end of k-mer on "+" strand relative to start of Y.contig
LCR	Integer LCR number to which this k-mer is assigned, each LCR has a unique LCR number assigned to it

Table 5. Columns in *.indels.tsv files; X,Y=chosen genome letters

Column	Description
ID	Unique ID tying row back to originating input file row. LCR input files: LCRnumber. InDelGroup and Markers files: refID_refPos1_refPos2.
phases	Phase of each genome incl. ref. genome, relative to ref. genome, string of +/- chars, + : same direction, - : opposite direction.",

Column	Description
idx	Starts at 1 and counts each InDel within an ID. For given ID (input row), number of InDels in that region is max idx value. If more than two genomes, entire region where alignment has a gap in one or more genomes is counted as one InDel even if multiple gap regions occur in different genomes.
Xdel,Ydel	Total number of deleted bps within the InDel in genomes X,Y. With 2 genomes, del = 0 in genome with insertion (no gaps), del > 0 in genome with deletion (gaps). With >2 genomes, del can be non-zero for all genomes. A genome has only insertions in the InDel if del is 0, and it has only deletions if end-start-1 = 0, and otherwise it has a mixture of at least one insertion and one deletion within the InDel interval.
Xid,Yid	Sequence ID of the InDel in genomes X,Y.
Xstart,Xend,Ystart,Yend	Overall InDel starting and ending position in genomes X,Y. start/end are positions of bps just BEFORE first and AFTER last InDel gap in any genome, so they refer to the same two bps in all genomes. Always start < end. If - phase, start is bp just AFTER, end is bp just BEFORE, opposite of +. Length of the InDel region in each genome is end-start-1.