

A Graph-based Approach for Designing Extensible Pipelines (Supplementary Information)

Maíra R. Rodrigues¹, Wagner C. S. Magalhães¹, Moara Machado¹ Eduardo Tarazona-Santos^{1,*}

¹ Departamento de Biologia Geral, Universidade Federal de Minas Gerais,
Av. Antonio Carlos 6627, Pampulha, Caixa Postal 486, 31270-910,
Belo Horizonte, Brazil.

February 2, 2012

1 The Performance Measure

The performance measure indicates the time that a tool takes to process one or more input files. It is used to identify the fastest pipeline, in case there are alternative pipelines for the same task. It is calculated as follows:

$$performance_measure = (time / filesizesum) * 10$$

where *time* is the time the tool took to execute (in seconds), and *filesizesum* is the size of its input file (in KB) (or files, in case the tool requires extra input files). This evaluation must be performed by the pipeline system's administrator on the analysis server where the tools are executed to avoid a difference in performance due to distinct machine configurations. It is also recommended that the pipeline system's administrator keep sample input files for each supported data types and used this same set in all evaluations. For example, files *sdat1test* and *sdat2test* are used as input to evaluate all tools that accept the SDAT format as input format. This avoids a bias due to sample size. We provide a java program at the project's website (<http://code.google.com/p/dynamic-pipeline/>) to help calculating the performance measure.

2 Pipeline System Algorithm

The pipeline system algorithm is formalised in Algorithm 1. The function *createIODependencies()* in line 16 is formalised in Algorithm 2.

3 List of Tools in the Dynamic Pipeline Application

Currently, our format conversion pipeline handles data formats compatible with the following software: PolyPhred (for polymorphism identification from aligned raw sequences reads) ?, PHASE (to infer chromosome phase) ?, DnaSP (for general population genetics analysis) ?, Structure (for population structure inferences) ?, Sweep (for natural selection inferences) ?, Haploview (for linkage disequilibrium analysis) ? and R-based tools for population genetics and genetic epidemiology such as HierFstat (for inferences about population structure) ?. More information about these software programs can be found on their website listed in Table 1. The pipeline also handles general purpose file formats such as SDAT (Figure 1), NEXUS and PrettyBase.

Software	Website
FastPhase	http://stephenslab.uchicago.edu/software.html
Structure	http://pritch.bsd.uchicago.edu/software.html
Phase	http://stephenslab.uchicago.edu/software.html
PolyPhred	http://droog.gs.washington.edu/polyphred/
DnaSP	http://www.ub.edu/dnasp/
Sweep	http://www.broadinstitute.org/mpg/sweep/
Haploview	http://www.broadinstitute.org/scientific-community/science/programs/medical-and-population-genetics/haploview/haploview
R-HierFstat	http://www2.unil.ch/popgen/softwares/hierfstat.htm
R-Adegenet	http://adegenet.r-forge.r-project.org/

Table 1: List of websites for the software programs with data formats supported by our format conversion application.

Algorithm 1 Pipeline system algorithm.

```
1: input: start, end, toolRegistry, wt
2: while Linej in toolRegistry do
3:   Ej(i, o, t, l, c, XI, XO, f, b, r, e) = parseEntry(Linej)
4:   if !containVertex(G, Ej[i]) then
5:     addVertex(G, Ej[i])
6:   end if
7:   if !containVertex(G, Ej[o]) then
8:     addVertex(G, Ej[o])
9:   end if
10:  if wt = "performance" then
11:    wj = Ej[f]
12:  else if wt = "dependencies" then
13:    wj = length(Ej[XI])
14:  end if
15:  addEdge(G, Ej[i], Ej[o], Ej[t], wj)
16:  createIODependencies(G, Ej[i], Ej[o], Ej[t], Ej[XI], Ej[XO])
17: end while
18: P = findToolPath(start, end, G)
19: previousTargetn = null
20: for all tn ∈ P do
21:   index = 0
22:   tgtn = edgeTarget(tn)
23:   IEtgtn = incomingEdges(tgtn)
24:   SIEtgtn = {(t1, ..., tk) : IEtgtn | tn = tk}
25:   for all tm ∈ SIEtgtn do
26:     tm = edgeSource(tgtm)
27:     if srcm ≠ previousTargetn then
28:       LI[index] = srcm
29:       index = index ++
30:     end if
31:   end for
32:   previousTargetn = tgtn
33: end for
34: LO = findOutputs(P)
35: (.inputs, .exec, .outputs) = writeExecutables(LI, P, E, LO)
```

4 Complete Tool Registry for the Format Conversion Pipeline

The complete Tool Registry for the Format Conversion pipeline application is presented in Table 1. Each line contains information about one conversion tool that composes the pipeline system. The tool registry was instantiated by both the tools' providers and the system administrator on request from the providers. A different application will have its own tool registry, instantiated with the appropriate tools for that specific application.

Algorithm 2 Algorithm for the function *createIODependencies()*. It adds the scripts' input and output dependencies to the Graph G .

```

1: input:  $G, i, o, t, XI, XO$ 
2:  $ew = MINWEIGHT$ 
3: if  $|XI| \geq 1$  then
4:   for all  $d_k \in XI$  do
5:     if  $containVertex(G, d_k)$  then
6:        $d_k^n = d_k$ 
7:       while  $containVertex(G, d_k^n)$  do
8:          $d_k^{n+1} = d_k^n$ 
9:       end while
10:       $addVertex(G, d_k^n)$ 
11:       $addEdge(G, d_k^n, o, t, w)$ 
12:    else
13:       $addVertex(G, d_k)$ 
14:       $addEdge(G, d_k, o, t, w)$ 
15:    end if
16:  end for
17: end if
18: if  $|XO| \geq 1$  then
19:   for all  $p_k \in XO$  do
20:     if  $containVertex(G, p_k)$  then
21:        $p_k^n = p_k$ 
22:       while  $containVertex(G, p_k^n)$  do
23:          $p_k^{n+1} = p_k^n$ 
24:       end while
25:        $addVertex(G, p_k^n)$ 
26:        $addEdge(G, i, p_k^n, t, w)$ 
27:     else
28:        $addVertex(G, p_k)$ 
29:        $addEdge(G, i, p_k, t, w)$ 
30:     end if
31:   end for
32: end if

```

SDAT Format

	1369	1572	1709	1715	1756	1778	1867	1908	1961	1992
NCP001	TC	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP002	TT	GG	GG	GG	GG	AA	GG	TT	GG	AA
NCP003	TT	GG	GG	GG	GG	AA	GG	TT	GG	AA
NCP004	TT	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP005	TT	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP006	TT	GG	GG	GG	GG	GG	GG	TT	GG	GG
NCP007	TT	GG	GG	GG	GG	GG	GG	TT	GG	GG
NCP008	TT	GG	GG	GG	GG	AA	GG	TT	GG	AA
NCP009	TT	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP010	CC	GG	GG	GG	GG	GG	GG	??	??	??
NCP011	TT	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP012	TC	GG	GG	GG	GG	GG	GG	TT	GG	GG
NCP013	TC	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP014	TT	??	??	??	??	??	GG	TT	GG	AG
NCP015	TT	GG	GG	GG	GG	AG	GG	TT	GG	AG
NCP016	TT	GG	GG	AG	GG	AG	GG	TT	AG	AG

The first line must have the allele positions separated by blank spaces.

Other lines:

- 1 - Individuals name
- 2 - Alleles for each position declared on first line

Figure 1: Sample SDAT-formatted data file.

INPUT	OUTPUT	SCRIPT	LANGUAGE	CODE	DEPENDENCIES	SUBPRODUCTS	PERFORMANCE	OBSERVATIONS	PROVIDER	CONTACT
POLY.OUT	PrettyBase	PolyPhasedPrettyBase.pl	perl	1	-	-	0.0048	-	Wagner Magalhães	@
PrettyBase	SDAT	PrettyBase2SDAT.pl	perl	2	-	-	0.0117	-	Wagner Magalhães	@
SDAT	PrettyBase	SDAT2PrettyBase.pl	perl	3	-	-	0.0200	-	Wagner Magalhães	@
SDAT	PHASE.Format	SDAT2Phase.pl	perl	4	-	-	0.0267	-	Wagner Magalhães	@
SDAT	Structure.Format	SDAT2Structure.pl	perl	5	-	mainparam.extraparam	0.1471	-	Wagner Magalhães	@
SDAT	NEXUS	SDAT2NEXUS.pl	perl	6	-	-	0.1586	-	Giordano Bruno	@
SDAT	R.HierStat	SDAT2RhierStat.pl	perl	7	SDAT	-	0.0250	Each fastPHASE file corresponds to a population and only one population file is mandatory. When using two or more files, the number of loci must be the same in all files.	Wagner Magalhães and Giordano Bruno	@
SDAT	R.Genetics	SDAT2Rgenetics.pl	perl	8	SDAT	-	2.1050	Each SDAT file corresponds to a population and only one population file is mandatory. When using two or more files, the number of loci must be the same in all files.	Wagner Magalhães and Giordano Bruno	@
PHASE.OUT	Fasta	Phase2Fasta.pl	perl	9	Fragments.RefSeq	-	0.0163	-	Wagner Magalhães	@
PHASE.OUT	Haploview.Format	Phase2Haploview.pl	perl	10	-	-	0.0048	-	Wagner Magalhães	@
PHASE.OUT	Sweep.Format	Phase2Sweep.pl	perl	11	-	-	0.0048	-	Wagner Magalhães	@
FPHASE.OUT	R.HierStat	FastPhase2RhierStat.pl	perl	12	FPHASE.OUT	-	0.0269	Each fastPHASE file corresponds to a population and only one population file is mandatory. When using two or more files, the number of loci must be the same in all files.	Wagner Magalhães	@
FPHASE.OUT	R.Genetics	FastPhase2Rgenetics.pl	perl	13	FPHASE.OUT	-	0.0192	Each fastPHASE file corresponds to a population and only one population file is mandatory. When using two or more files, the number of loci must be the same in all files.	Wagner Magalhães	@
POLY.OUT	Fasta	PolyPhasedFasta.pl	perl	14	RefSeq	-	6.2951	-	Wagner Magalhães	@
NEXUS	R.HierStat	NEXUS2RhierStat.pl	perl	15	NEXUS	-	0.0259	Each NEXUS file corresponds to a population and only one population file is mandatory. When using two or more files, the number of loci must be the same in all files.	Wagner Magalhães	@

Table 2: Complete tool registry for the format conversion pipeline.