

Y-chromosomal analysis tools: logic flow document

Abstract

The aim of this programme is to take literature and consumer Y-DNA data from genetic testing companies, and appropriate meta-data from individuals’ genealogical histories, and translate them into a geographically and temporally encoded phylogenic tree.

This document describes the coding logic behind the workflow. It is an evolving document designed to communicate and expedite the workflow, rather than a formal description of how the processes work.

Author

Iain McDonald, 26 Jan 2018

Key

Standard flowchart terminology is used for the most part. In addition:

RED items are things that need attention

PURPLE items are data obtained from other sources

YELLOW items are queries to the file system

GREEN items are database entries

BLUE items are storage lists kept in RAM by the programme

Overall programme structure

There are five main challenges to overcome, corresponding to the six work pages listed:

Work Package A: How can we take this diverse dataset and create a manageable and standardised series of calls from it?

Work Package B: How can we parse these calls to create the best haplogroup tree?

Work Package C: Can we chart the progression of Y-STR mutations and perform haplogroup predictions on testers with or without any non-NGS SNP testing?

Work Package D: Can we map the movement of people across Europe using the paper origins of individuals and ancient DNA results?

Work Package E: Can we assign ages to these haplogroups (and by extension the migrations they trace)? How well can we make these line up with known historical events? How can we interface these outputs with those of others?

Work Package F: What can we provide by way of informative and useful outputs?

Work Package A

The primary difficulties here involve managing the large variety of possible input files, the diversity of their respective information, and the differences in their quality standards. The large disc space of these datasets is also an issue, so clearly unimportant variant candidates need to be carefully rejected first. Typical datasets might include:

FASTQ - raw sequences of reads that need aligned to a reference genome (e.g. GRCh38), sources may include literature data or ancient DNA. These may be large fractions of a GB in size.

BAM - aligned data sequences that need called for variants. Sources may include raw BigY data and are typically ~500 MB.

BigY Build 38 VCF - variant call files from the original BAMs. These should have accompanying coverage files (BED) in which regions we are ok to say a variant is passed. The BED regions also define the count for the age analysis. Typical sizes are ~15 / 80 MB zipped/unzipped, and we anticipate thousands of these files.

BigY Build 37 - any legacy data from the old Build 37 data (mapped to GRCh37 / hg19). BAM files can be split back to FASTQ files and remapped to Build 38. VCFs can be processed using “liftOver”, which is a simpler co-ordinate conversion. However, this does not allow for better mapping of reads in Build 38, so is not ideal. Size is 1-2 MB zipped.

FGC YElite / WGS - BED files need generated for these using callableLoci in GATK. This is something FGC can do. Variants are stored in gtype files.

YSeq WGS - the Y chromosomal (chrY) data needs extracted from the results, then processed. Status of VCF/BED extraction is currently unclear.

BigY CSV, SNP pack tests, individual YSeq/FTDNA SNP tests, Chromo2 and literature data - these are essentially a list of variants for which a test is positive. Sometimes there are lists of negatives too. Rarely are the information on negative versus null. These data are unsuitable for some analyses (e.g. age analysis) and have limited input into the haplotree, where they would be expected to be added last.

Variants from these inputs will need to be ingested for later comparison, along with associated meta-data like surnames, ancestral origins and STR results. Other STR results, without SNP results, are desirable for later geographical analysis, where they will provide additional datapoints.

Work Package B

Once ingested and homogenised, this data needs to be parsed to form a haplotree. The difficulties here are minimising the amount of data that needs processed at one time, in order to reduce RAM requirements; and dealing with low quality and missing (uncalled) results.

Parsing will typically require sorting the data into groups of people and clades (haplogroups) of variants to form “blocks”. Missing calls will either need to be presumed positive or negative, based foremost on logic, but also on low-quality results in NGS data.

The resulting tree structure is used as input to WPs C, D and E, each of which needs to be capable of running from this input alone, or by including data from the others. Other outputs may compare the output haplotree to those from other sites (e.g. FTDNA’s haplotree, ISOGG) to identify where updates need to be made.

Work Package C

This package centres around making use of STR results: both those from the NGS tests and STR-specific tests at the various companies (predominantly FTDNA but also YSeq). Using the NGS test results, we can attempt generation of the ancestral STRs for that clade. This will let us see how STRs have varied across history, and identify the key mutations present in each haplogroup. A further stage of this would be haplogroup estimation for non-NGS-tested individuals.

Not all haplogroups will have a corresponding, stable, identifiable STR mutation. Hence this is only going to be possible for some levels in the tree, and must be an approximate science. Haplogroup estimation can therefore only be done in an approximate sense, but it should provide enough useful data to use in WPs D and E.

Work Package D

This package tries to connect haplogroups to migrations by looking at how the distribution of testers within that haplogroup deviates from the weighted average. This must be done on a clade-by-clade basis to avoid founder effects (e.g. the origin of R-M269 is in Russia, but most R-M269 is now in Europe, due to a founder effect at the R-L11 level).

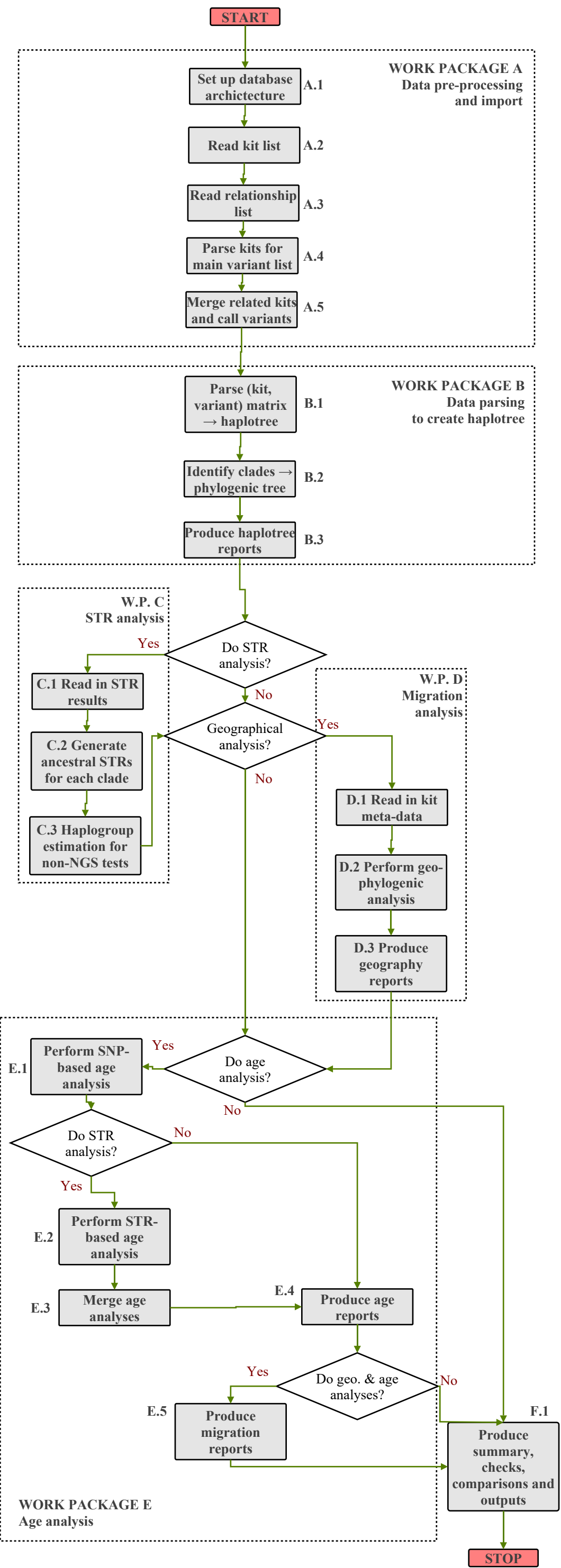
Difficulties here involve getting a sufficient number of diverse members for each clade (hence why the haplogroup estimation in WP C is important). The data are also heavily biased, since some European regions (e.g. the British Isles) have populations and diaspora that are much more widely tested than other countries (e.g. France, Eastern Europe). Weighting entries to debias them is important, as well as correctly weighting sub-clades and ancient DNA results.

Work Package E

This package performs the age analysis, to put each haplogroup in an historical context. The basic principles behind this are effectively worked out, but take a lot of CPU power. STR-based age analysis can be done fairly easily, but the results will need carefully calibrated and cross-checked for consistency. Treatment of uncertainties is also very important. Most of the problems here are likely to be mathematical in nature.

Work Package F

A range of outputs from this data can be considered, from a list of ages and co-ordinates, to bespoke queries on individual tests (haplogroup estimation, STR mutation timeline, migration pathway), to animations of migrations across Europe.



Item A.1: Database architecture

This step sets up the initial database structures for the programme (or imports them), along with any parametric setup files and data pointers.

Uses global parameters: **ResetDB** **Boolean**: Drop all tables and remake database
External inputs: User parameter file

Database tables

The following tables describe the database structures. Grey fields are currently unused. Red fields are not in the current (24 Jan 2018) schema.

PERSON – primary list of people				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.4	-
firstName	TEXT DEFAULT NULL	Name of tester	A.4	-
middleName	TEXT DEFAULT NULL	Name of tester	A.4	-
surName	TEXT DEFAULT NULL	Name of tester	A.4	-
maidenName	TEXT DEFAULT NULL	Name of tester	A.4	-
yHaplogroupId	INTEGER DEFAULT NULL	Most-recent Y-DNA hg	B	-
mtHaplogroupId	INTEGER DEFAULT NULL	Most-recent mt-DNA hg	-	-
sortidx	INTEGER	Haplotree matrix sort index	B	-

DATABASE – primary list of kits				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.2	-
seq	INTEGER DEFAULT 0	Order files arbitrarily	B	-
kitName	TEXT	A name for the dataset	A.2,A.4	-
DNAID	INTEGER references person(ID)	Whose DNA is this?	A.2,A.4	A.4
contact	TEXT	Holder's contact info	A.2	-
kitId	TEXT [INDEX fileidx]	Lab ID	A.2	-
surNameID	INTEGER references surName(ID)	MDKA surName	A.2,A.4	-
countryID	INTEGER references country(ID)	MDKA country	A.2,A.4	-
birthYr	INTEGER DEFAULT NULL	MDKA birth year	A.2,A.4	-
labID	INTEGER references lab(id)	Testing lab	A.2	-
testTypeID	INTEGER REFERENCES testtype(ID)	Type of DNA test	A.2	-
buildID	INTEGER references build(ID)	Reference build for the test	A.2	-
importDt	TEXT	Date originally imported	A.2	-
fileNm	TEXT	Normalized file/path name	A.2	-
origFileNm	TEXT	Original file name from user	A.2	-
otherInfo	TEXT	Freeform text entered by user	A.2,A.4	-
normalOrigID	INTEGER references origin(ID)	Normalized ancestral origin	A.2,A.4	-
lat	TEXT	Ancestral location latitude	A.2,A.4	-
lng	TEXT	Ancestral location longitude	A.2,A.4	-
policyVer	TEXT DEFAULT NULL	Data policy agreed to	A.2	-
accessToken	TEXT DEFAULT NULL	-	-	-
updated	TEXT	Date the record was updated	A.2	-
approxHg	TEXT	Haplogroup assigned by DW	A.2	-

SURNAME – list of surnames				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.2	-
surName	TEXT	Name	A.2	-

TESTTYPE – list of tests				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.2	-
testNm	TEXT	The name of the test	A.2	-
description	TEXT	Further description	A.2	-
isNGS	BOOLEAN	Is it a sequencing test?	A.2	-
tagNm	TEXT	Field in hg-R schema	-	-
priority	INTEGER	Field in hg-R schema	-	-

CONTIG – defines regions of interest				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.1	-
buildID	INTEGER references build(ID)	Reference build, e.g. hg38	A.1	-
description	TEXT	e.g. chrY	A.1	-
length	INTEGER	e.g. 57227415	A.1	-

COUNTRY – list of countries				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.2	-
country	TEXT	Name	A.2	-

ORIGIN – list of origins				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.2	-
origin	TEXT	Name	A.2	-

LAB – list of labs				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.2	-
labNm	TEXT	Name	A.2	-

BEDRANGES – minimum and maximum ranges in BED file entries				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.5	-
minaddr	INTEGER [INDEX rangeidx]	Position of minimum	A.5	-
maxaddr	INTEGER [INDEX rangeidx2]	Position of maximum	A.5	-

BED – entries present in each BED file				
Name	Type	Description	Filled in	Used in
pID	INTEGER REFERENCES dataset(ID)	Person	A.5	-
bID	INTEGER REFERENCES bedranges(ID)	BED entry	A.5	-

TREE – description of the haplogroup tree				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	B	-
Input is needed on this design, perhaps conforming to an established standard				

ALLVARIANTS – positions and counts for every entry present in VCF files				
Name	Type	Description	Filled in	Used in
pID	INTEGER REFERENCES dataset(ID)	Person	A.4	-
buildID	INTEGER REFERENES build(ID)	Build	A.4	-
pos	INTEGER	Position	A.4	-
refAlleleID	INTEGER REFERENCES alleles(ID)	Reference allele	A.4	-
derAlleleID	INTEGER REFERENCES alleles(ID)	Derived allele	A.4	-
npos	INTEGER	# positive calls	A.4	-
nneg	INTEGER	# negative calls	A.4	-
nmix	INTEGER	# mixed calls	A.4	-
fpos	INTEGER	# failed + calls	A.4	-
fneg	INTEGER	# failed - calls	A.4	-
fmix	INTEGER	# failed 0/1 calls	A.4	-
bneg	INTEGER DEFAULT NULL	# - calls from BED	A.5	-

VCFCALLS – entries present in each VCF file				
Name	Type	Description	Filled in	Used in
pID	INTEGER REFERENCES dataset(ID)	Person	A.5	-
vID	INTEGER REFERENCES variants(ID)	VCF entry	A.5	-
... [INDEX vefidx]				
alleleID	INTEGER REFERENCES alleles(ID)	Allele called	A.5	-
ruleID	INTEGER REFERENCES rules(ID)	Reason called	A.5	-

VCFSTATS – reported statistics on VCF files				
Name	Type	Description	Filled in	Used in
pID	INTEGER REFERENCES dataset(ID)	Person	B	-
ny	INTEGER	-	B	-
nv	INTEGER	-	B	-
ns	INTEGER	-	B	-
ni	INTEGER	-	B	-
nr	INTEGER	-	B	-

BEDSTATS – reported statistics on BED files				
Name	Type	Description	Filled in	Used in
pID	INTEGER REFERENCES dataset(ID)	Person	A.5	-
coverage1	INTEGER	Total coverage	A.5	-
coverage2	INTEGER	Cov. for age analysis	A.5	-
nranges	INTEGER	# entries	A.5	-

META – meta-data for the entire data set or run				
Name	Type	Description	Filled in	Used in
desc	TEXT	-	-	-
val	TEXT	-	-	-

STRS – table for STR definitions				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	C	-
strName	TEXT	Name	C	-
ordering	INTEGER	Put STRs in order	C	-

STRCALLS – tester's calls for STRs				
Name	Type	Description	Filled in	Used in
pID	INTEGER REFERENCES people(ID)	Person	C	-
sID	INTEGER REFERENCES strs(ID)	STR	C	-
val	INTEGER	Allele	C	-

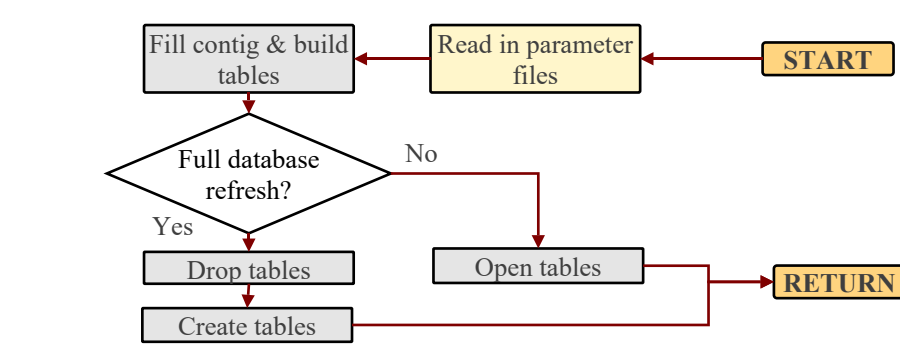
VARIANTS – list of known variants [INDEX varidx on buildID,pos]				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.5	-
buildID	INTEGER REFERENCES build(ID)	Build	A.5	-
pos	INTEGER	Position	A.5	-
anc	INTEGER REFERENCES alleles(ID)	Ancestral allele	A.5	-
der	INTEGER REFERENCES alleles(ID)	Derived allele	A.5	-
sortidx	INTEGER	Haplotree sort index	B	-

VRULECOUNTS – counts of numbers of calls made under each rule				
Name	Type	Description	Filled in	Used in
vID	INTEGER REFERENCES variants(ID)	-	A.5	-
rID	INTEGER REFERENCES rule(ID)	-	A.5	-
count	INTEGER	-	A.5	-

VARIANTQUALITY – quality flags for variants				
Name	Type	Description	Filled in	Used in
vID	INTEGER REFERENCES variants(ID)	-	A.5	-
qualsum	FLOAT	Call quality sum	A.5	-
qualpos	BOOLEAN	Tree pos questionable	B	-

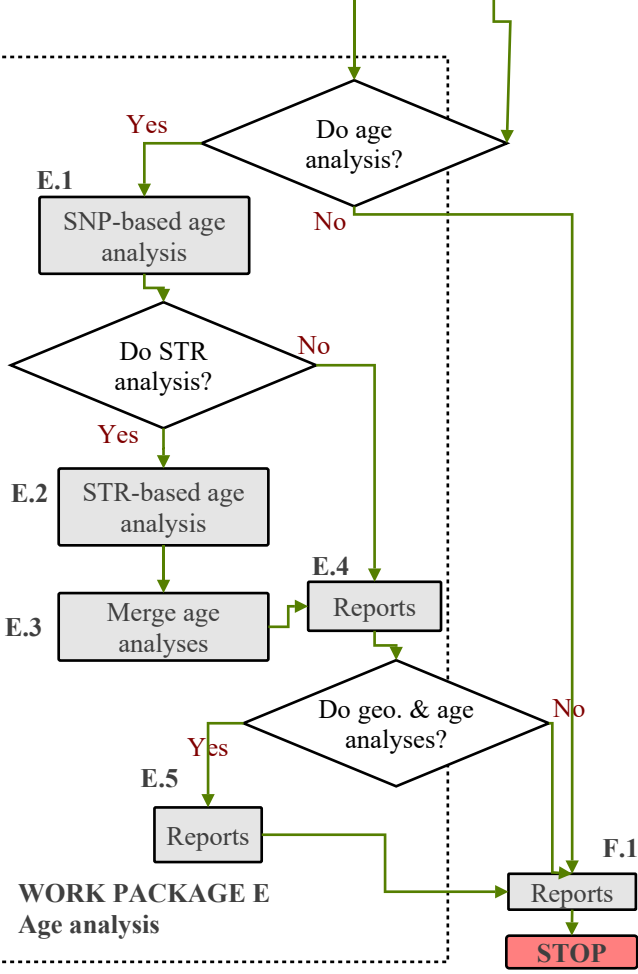
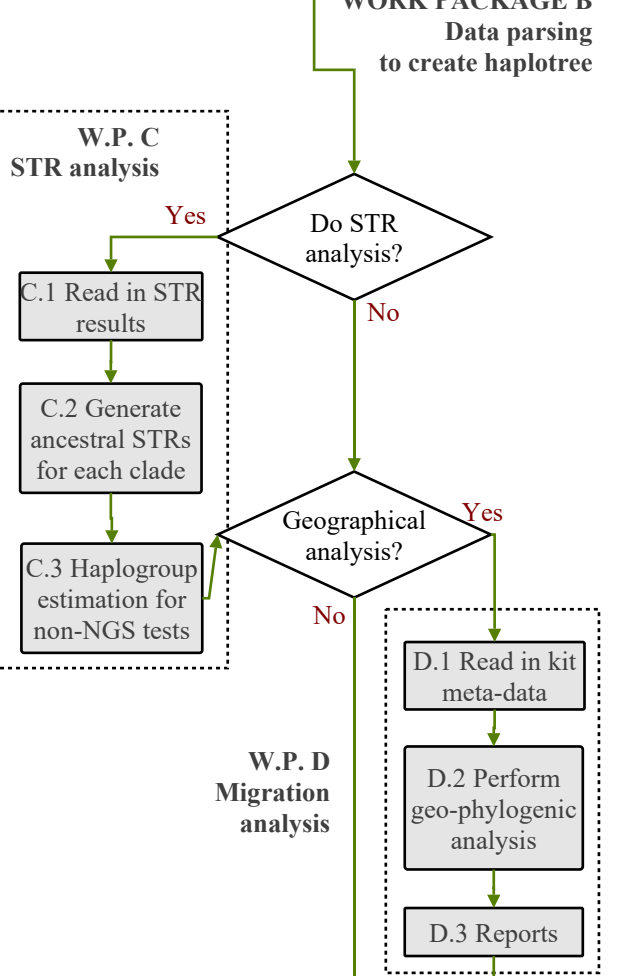
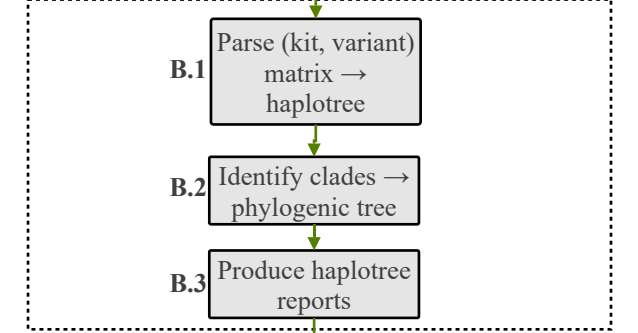
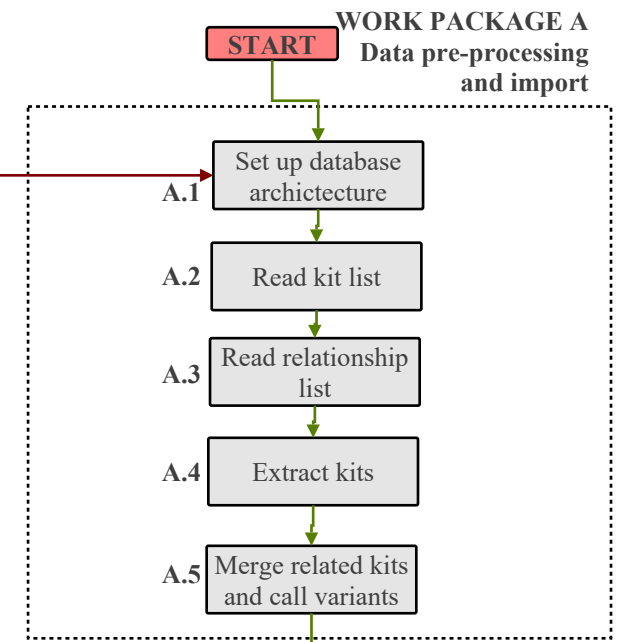
ALLELES – list of alleles in sequencing calls				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.5	-
allele	TEXT	e.g. A,C,G,T,AT,...	A.5	-

SNPNAMES – names and aliases associated with variants				
Name	Type	Description	Filled in	Used in
vID	INTEGER REFERENCES variants(ID)	-	A.5	-
snpname	TEXT	e.g. U106	A.5	-



BUILD – list of reference genome assemblies used				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.1	-
buildNm	TEXT	-	A.1	-

AGEBED – restricted list of ranges that are used to calculate the ages of clades				
Name	Type	Description	Filled in	Used in
ID	INTEGER PRIMARY KEY	-	A.5	-
bID	INTEGER REFERENCES bedranges(ID)	-	A.5	-



Item A.2: Read in list of kits

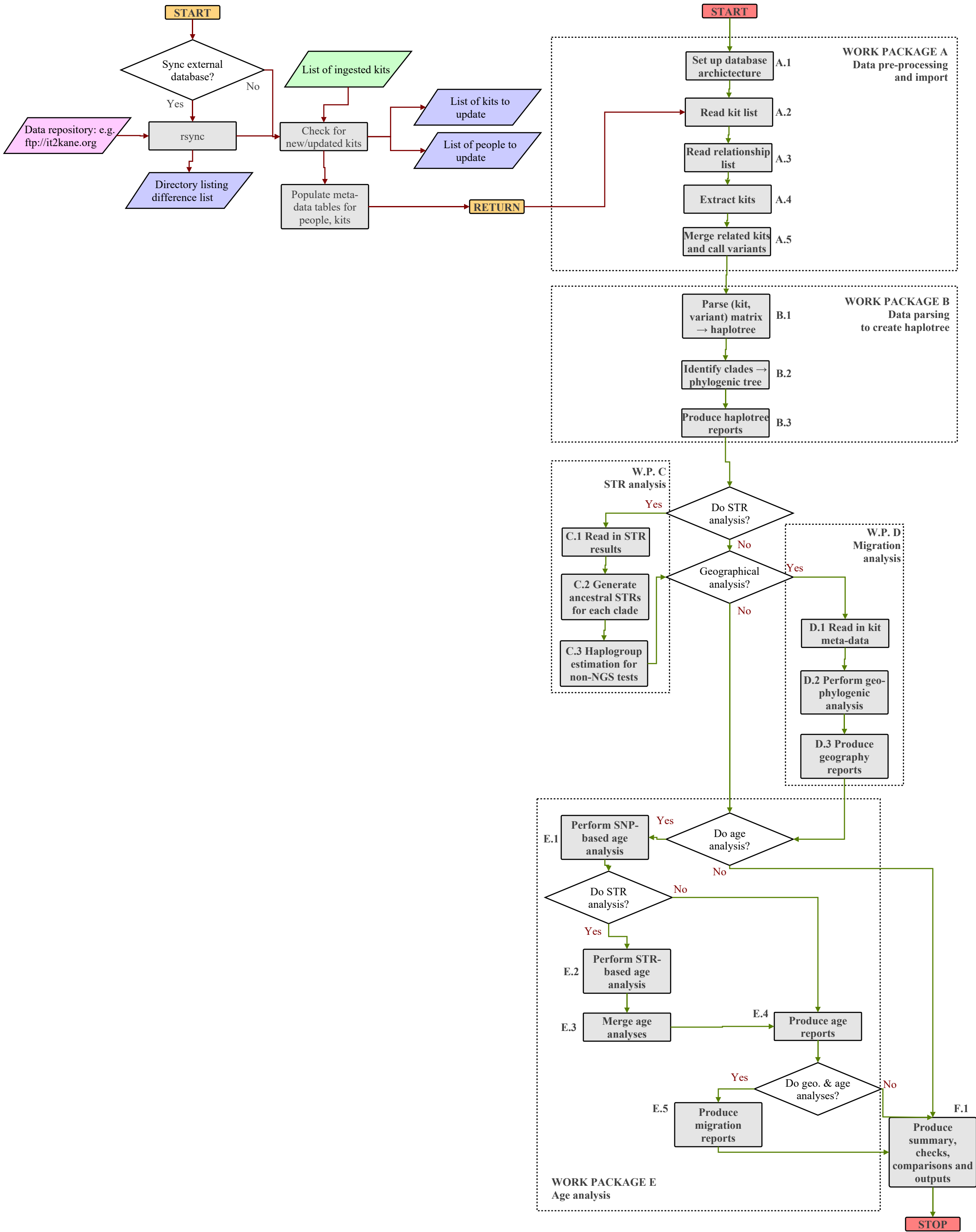
This step identifies whether any new kits are to be read in.

Uses global parameters:
SyncToDB Boolean: synchronise to external database?

External inputs:
Data repository

Internal inputs:
Internal database **kits**
Internal database **people**

Produces:
List of **kits** to update
List of **people** to update



Item A.3: Read in list of relationships

This list identifies and describes the relationships between different kits. Applications are twofold. Firstly, kits belonging to the same person need to be merged for best effect (A.4). Kits belonging to closely related individuals (e.g. father/son or cousin pairs) may be best merged as well. Secondly, known relationships become important when performing the age analysis, in defining limits to relationships. The same list can be used to define the ages of ancient DNA samples. The input list may be spread over one or more files, and contain three data types:

- MDKA: provides a date (with or without uncertainty) for the kit holder’s most-distant known ancestor’s birth. Relationships cannot be younger than the latter of two tests’ MDKAs.
- DoB: provides a date of birth for a tester, or a probability distribution function (PDF) describing this. This will mainly be useful for carbon-14 dates from ancient DNA results, which may come as arbitrarily complex PDFs.
- Link: a known common ancestor links two tests. This may be two tests from the same person. It may have some uncertainty, and it may be an arbitrarily complex function (e.g. for testers known/suspected to have a common surname origin or emigrant history). It can also be used to set arbitrary boundaries like “clades must be related after the Norman Conquest” or “must be related within the Corded Ware Culture period”, etc.

Data formats may be:

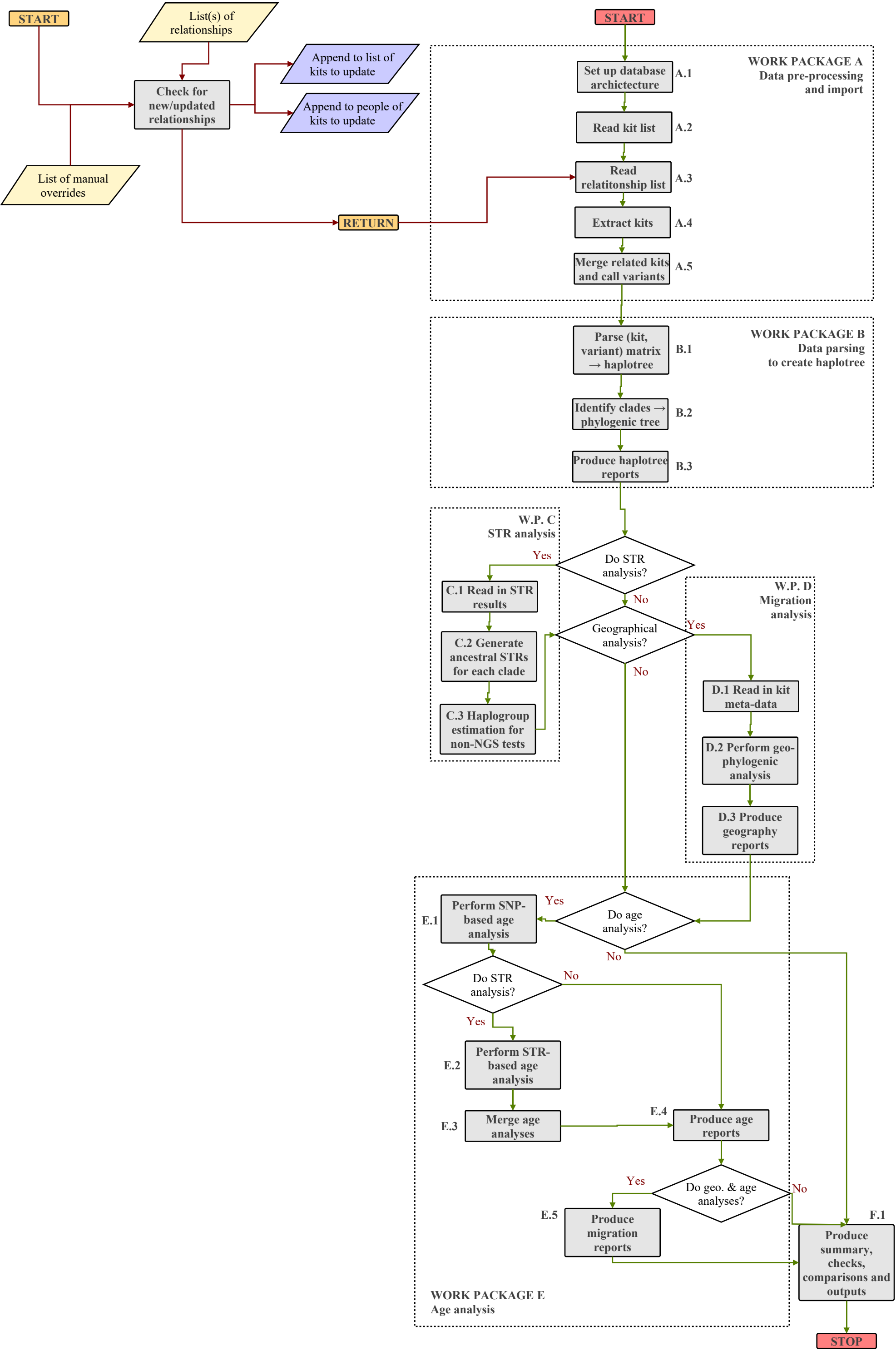
Type=MDKA, Kit, Date, Unc, LowLimit, UpLimit, Comment
Kit Must be unique identifier (e.g. sequence number or include company)
Date As common epoch years (CE) [default=zeroage].
Unc Gaussian uncertainty parameter [default=zeroage**unc**].
Low/UpLimit Youngest/oldest possible date [default=currentYear/-inf].

Type=DoB, Kit, Date, Unc, LowLimit, UpLimit, PDFfile, Comment
PDFfile [Default=null] Contains arbitrary PDF formatted as:
Year **Probability**

Type=Link, Kit1, Kit2, Date, Unc, LowLimit, UpLimit, PDFfile, Merge?, Comment
Kit1,2 The two kits under consideration
Merge Boolean TRUE/FALSE to merge kits into the same person [default=false]

Manual over-rides to table entries can be made at this point too. This allows us to set things such as people's ancestral locations and MDKA information with more accurate information that they have provided but is not available from the primary data sources. Data formats may be:
Table, KitID, PersonID, Field, Data, Comment
e.g.:
Kits, 123456, , mdkaDoB, 1700, IM: updated from E-mail

Uses global parameters:
RelationFile File(s) containing list of relationships
zeroage Default DoB of testers (c. 1950).
zeroageunc Associated uncertainty (c. 16 years).



Item A.4: Extract kits

This will be one of the harder parts of this work package to get right, due to the volume of data involved and the diversity of data we have to deal with. This diversity is likely to require specific analyses for each kind of data.

As well as BigY, FGC YElite and WGS, and YSeq WGS tests, we may have to deal with arbitrary lists of calls from CSV calls from BigY, National Geographic, Chromo2, YSeq, literature etc. These may be presented as hg36, hg37 or hg38 positions, or as named SNPs.

BAM and FASTQ files from literature sources may include modern or ancient DNA. Initial quality filters for ancient DNA will likely need to be lowered.

Extraneous data (e.g. negative calls, failed calls, quality data) cannot be completely thrown away at this point, because we don't know which variants we're interested in until we've looked at all the ZIP files. However, the unzipped VCF files *may* ultimately become too large to reasonably store on disc. We can either: (1) keep the unzipped files if they're not too large; or (2) remove the unzipped files each time, then re-unzip them again later. Let the user decide via **keepunzip**. The logic can be simplified if we don't actually edit the files, but can read them from their ZIPped state.

We are only interested in variants where there are a mixture of positive and negative calls. Every other variant is either shared among everyone (above the tree) or never positive (outside the tree). The first VCF pass selects any variant which has the following: \$7=PASS and GT=0/0 (reference call), 1/1 (derived call) or 0/1 (mixed call). These are parsed into an array of: *positions*, *allele(ref)*, *allele(der)*, and a series of seven counts representing the possible states: positive passed, positive failed, mixed passed, mixed failed, negative passed, negative failed and negative passed by BED file (this last one can't be parsed at this time).

At the end of this process, we can select those variants where $(count(positive\ passed)>0 \parallel count(mixed\ passed)>0) \ \&\& \ (count(negative\ passed)>0 \parallel sum(all\ counts)<total)$ as those we want to investigate: these are variants found in someone but not everyone (or those where we need to investigate the BED files to make this distinction), so are those needed to make the tree. These make the master **variants** list, and we assume for now that the reference allele is the ancestral allele (positives in the reference sequence are dealt with in the next item).

Uses global parameters:

- useBigY** Boolean: extract new BigY results?
- useFGC** Boolean: extract new FGC results?
etc.
- usehg37** Boolean: extract build 37 results?
- keepunzip** Boolean: keep or delete unzipped files?

External inputs:

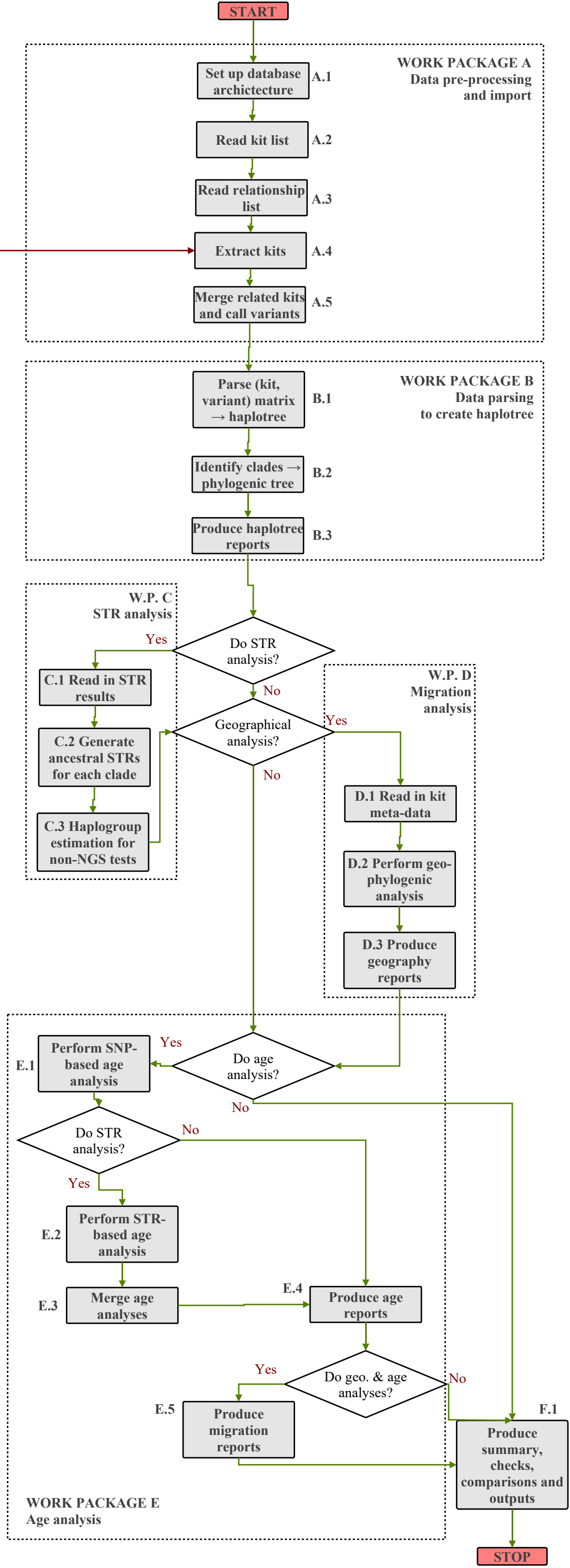
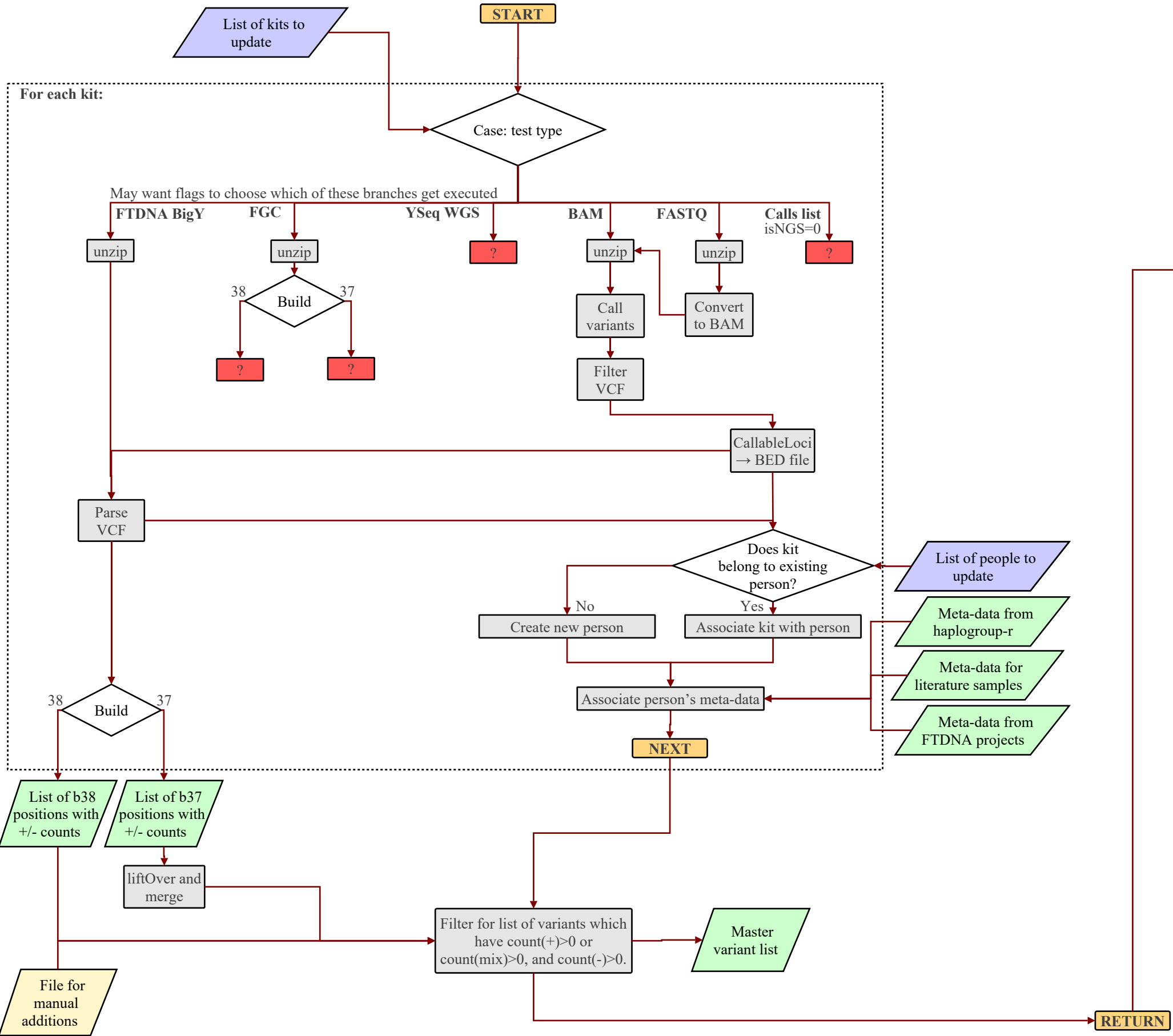
Data repository meta-data

Internal inputs:

Lists of **kits** and **people** to update

Produces:

- Master list of **allvariants**
- Master list of interesting **variants**



Item A.5: Merge related kits / call variants

Several kits may be associated with one person. This may include any form of combinations, e.g.: BigY+YElite, BigY+YSeq/WGS, Y-STRs + literature PGP test, Geno 2.0 + BigY, Y-STRs + YSeq SNPs, etc.

One method of dealing with this is to call a list of (people,variants), rather than (kits, variants) to form the haplotree with. Care must be taken to ensure that the most secure determination of a call is taken: e.g., how does a rejected BigY call with BQ=40, MQ=40 and 10/14 reads positive compare with a FGC ** call? This may require careful thought, and the option of guiding the tree structure in part B with a series of manually injected calls may be necessary.

Even if we take a pure (people,variants) list of calls, the merger of two tests must result in a new BED file. This can be a straight merger of data.

E.g.:
chrY 120000 125000
chrY 127000 128000
and
chrY 124000 126000
chrY 129000 130000
would merge to:
chrY 120000 126000
chrY 127000 128000
chrY 129000 130000

Where individual SNP calls have been made by packs or individual tests, these entries will need added into the BED file, e.g.:

chrY 12060400 12060401
for a Z301 call.

At this point, we also try to fix positives in the reference sequence. These occur because the reference sequence is primarily made from the Y-DNA of a R-U152 male, hence is positive for all SNPs from Y-DNA Adam down through the R-U152 sequence and below. YBrowse's SNP names list maintains the ancestral version, so we can use this to correct these reference-sequence positives when their ancestral values don't match our reference sequence values.

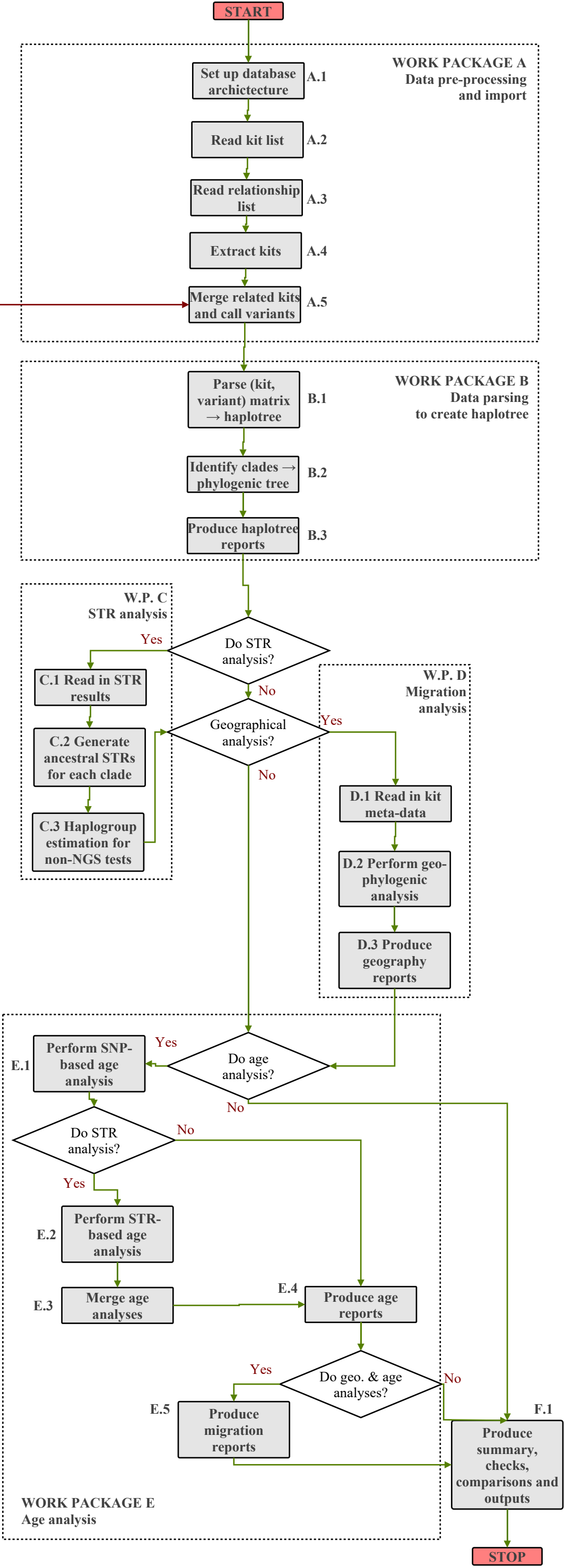
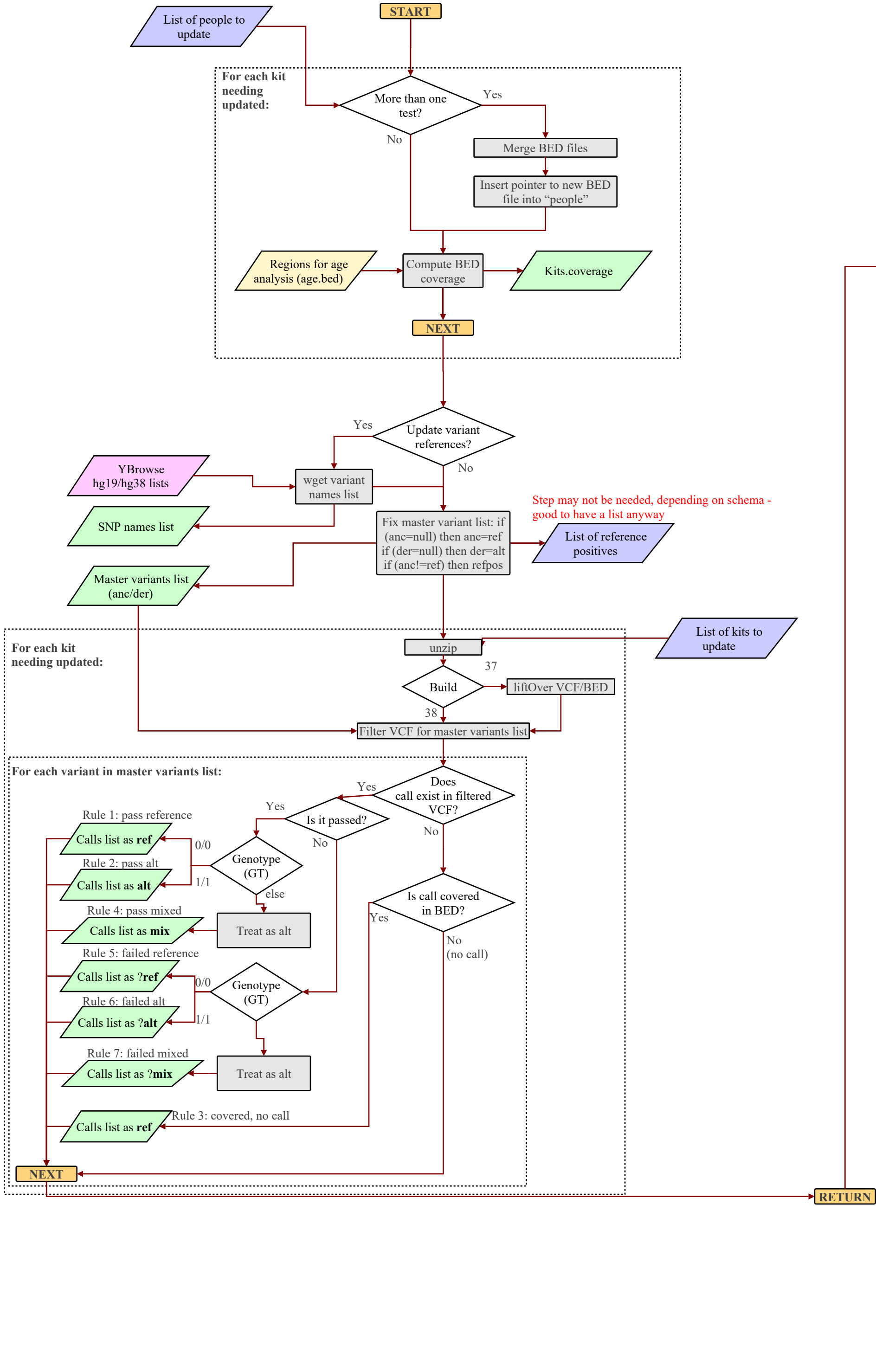
Once this is done, we can make our calls. For calls which pass the quality criteria (field[6] = "PASS" [zero-indexed]), the genotype (field[10,1]) will represent 0/0 for the reference allele, 1/1 for the derived allele, and may be something else (0/1, 1/2, etc.) for a mixture of calls. Note that the derived allele may be comma-separated if there is more than one possible alternative for that call in that test. In the case of mixed (ref/alt or ref/alt1/alt2) calls, we have to choose one to go forwards with.

The quality filter needs recorded: this is -10*log10(Probability call is wrong). So QUAL=10 implies P=0.1, or a 10% chance of a bad call. In a test of 10 million base pairs, QUAL < 70 may be spurious. Variants may also be rejected because the depth (DP) is too high: this could occur if multiple regions of DNA are mapped to the same co-ordinates because they are too similar. It may become more logical to grade the rules more accurately than one simple quality setting.

External inputs:
YBrowse's list of variants

Internal inputs:
Master list of **variants**
List of **people** to update

Produces:
Merged coverage files for people with more than one test
Coverage statistics for each **kit**
Reference positive corrections for the **variants** list
A list of **snpname**s
Set of **calls** for each person/variant
Call quality details on each variant



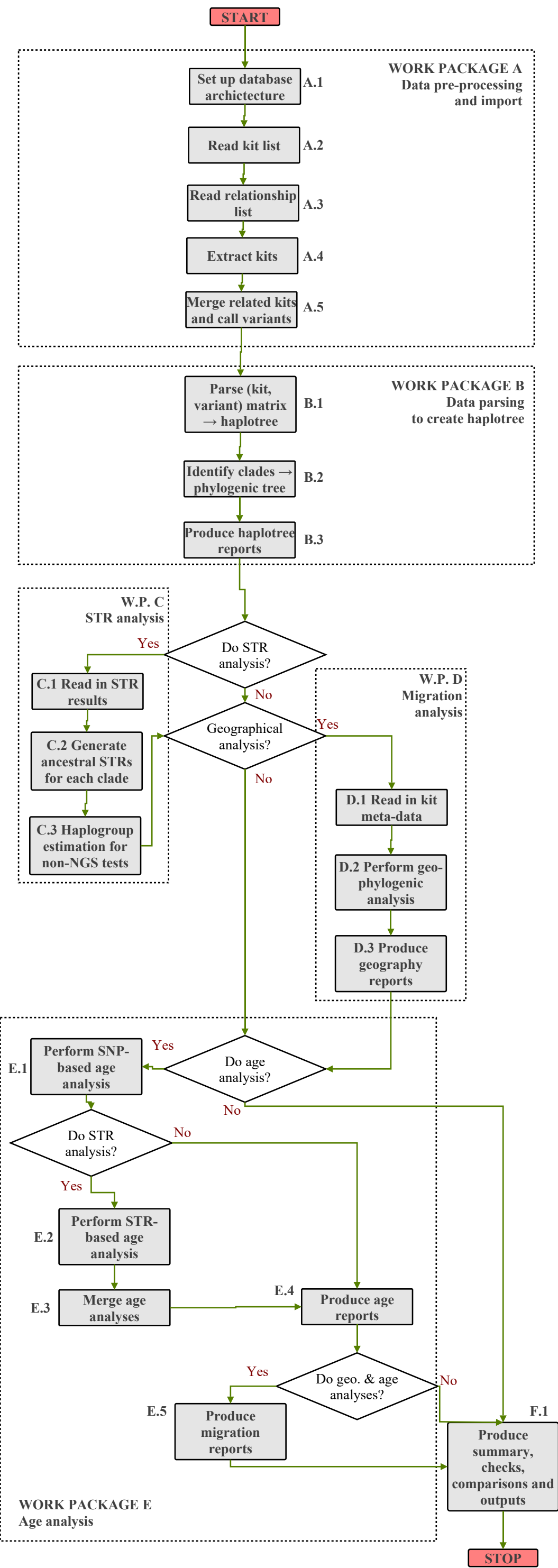
Item B.1: Parse variants to form a haplotree

This is one of the more computationally and logically challenging aspects of this process. There are several issues:

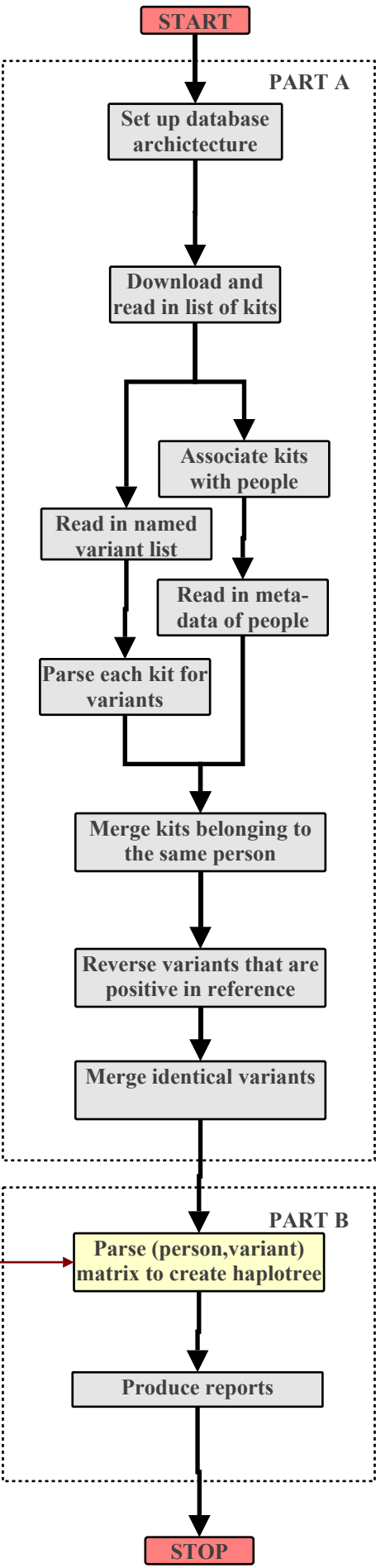
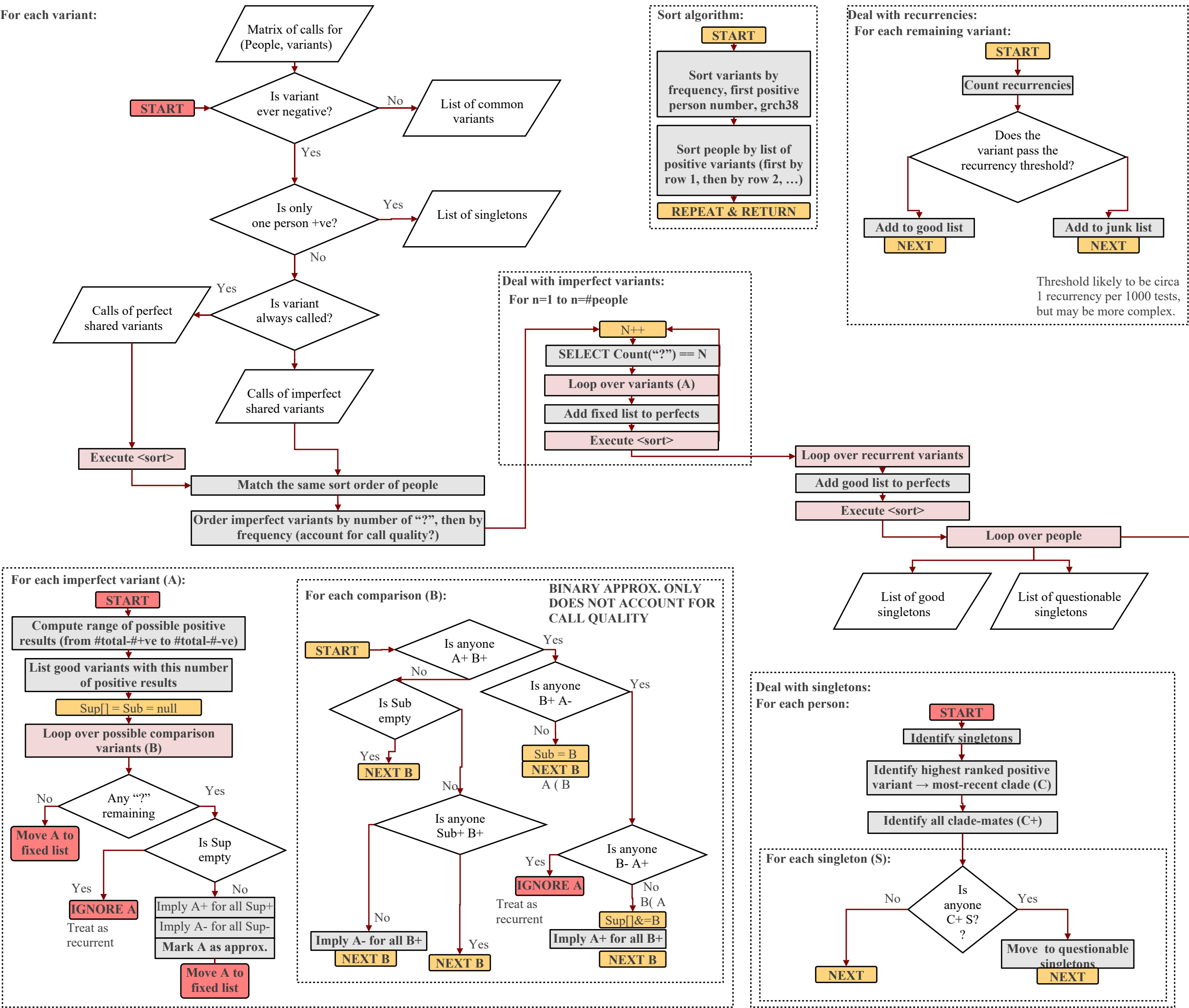
- Although rare, a handful of mutations may back-mutate in individual lines, and we need to correct for this.
- Mutations can be recurrent, meaning they cannot be parsed into a phylogenic tree, and need split up.
- Many mutations will be called sporadically among a few tests.
- Indel notation needs standardised among the tests:
https://genome.sph.umich.edu/wiki/Variant_Normalization

Many of these processes have known solutions. However, the common solutions, e.g., <http://www.it2kane.org/2016/07/combining-variant-compare-files/> have problems with scalability, reference positives, treating bad data and questionable variants. A more streamlined approach is desired that reduces both CPU time and RAM.

This process will involve several steps, including calculating a quality statistic for each variant, adding each variant into the tree in terms of its quality, and sorting the tree and its variants.



For each variant:



Select * where (npos + nneg == total) and (npos>1) and (nneg>0)

Basic list		Count	First	75848	274005	75848	289065	584862	B75780	284744	65049	120386	5962	216600	22654	N4826	224096
FGC13445	4484946	3	2	-	+	-	-	-	-	-	-	-	-	-	+	-	+
FGC13446	7678200	2	2	-	+	-	-	-	-	-	-	-	-	-	-	-	+
FGC13458	13908115	2	2	-	+	-	-	-	-	-	-	-	-	-	-	-	+
S18823	14695706	2	3	-	-	+	-	-	-	-	-	-	-	+	-	-	-
CTS7658	15482393	2	3	-	-	+	-	-	-	-	-	-	-	+	-	-	-
S1894	16098109	3	4	-	-	-	+	-	+	-	+	-	-	-	-	-	-
S1900	17358211	3	4	-	-	-	+	-	+	-	+	-	-	-	-	-	-
S1911	21003759	7	2	-	+	-	+	-	+	+	+	-	-	-	+	-	+

Sort by:
count,first,position

		Count	First	274005	75848	289065	584862	B75780	284744	65049	120386	5962	216600	22654	N4826	224096
S1911	21003759	7	2	-	+	-	+	-	+	+	+	-	-	+	-	+
FGC13445	4484946	3	2	-	+	-	-	-	-	-	-	-	-	+	-	+
S1894	16098109	3	4	-	-	-	+	-	+	-	+	-	-	-	-	-
S1900	17358211	3	4	-	-	-	+	-	+	-	+	-	-	-	-	-
FGC13446	7678200	2	2	-	+	-	-	-	-	-	-	-	-	-	-	+
FGC13458	13908115	2	2	-	+	-	-	-	-	-	-	-	-	-	-	+
S18823	14695706	2	3	-	-	+	-	-	-	-	-	+	-	-	-	-
CTS7658	15482393	2	3	-	-	+	-	-	-	-	-	+	-	-	-	-

Sort first row, + to -

Count	First	75848	584862	284744	65049	120386	22654	224096	274005	289065	B75780	5962	216600	N4826
S1911	21003759	7	1	+	+	+	+	+	+	-	-	-	-	-
FGC13445	4484946	3	2	+	-	-	-	+	+	-	-	-	-	-
S1894	16098109	3	4	-	+	+	-	+	-	-	-	-	-	-
S1900	17358211	3	4	-	+	+	-	+	-	-	-	-	-	-
FGC13446	7678200	2	2	+	-	-	-	-	+	-	-	-	-	-
FGC13458	13908115	2	2	+	-	-	-	+	-	-	-	-	-	-
S18823	14695706	2	3	-	-	-	-	-	-	+	-	-	+	-
CTS7658	15482393	2	3	-	-	-	-	-	-	+	-	-	+	-

Sort subsequent rows from first to last positive result. Check results consistent: last positive row is the same. Reject if not.

Current row = 2
Last positives all same

Count	First	75848	584862	284744	65049	120386	22654	224096	274005	289065	B75780	5962	216600	N4826
S1911	21003759	7	1	+	+	+	+	+	+	+	-	-	-	-
FGC13445	4484946	3	1	+	-	-	-	-	+	+	-	-	-	-
S1894	16098109	3	4	-	+	+	-	+	-	-	-	-	-	-
S1900	17358211	3	4	-	+	+	-	+	-	-	-	-	-	-
FGC13446	7678200	2	2	+	-	-	-	-	+	-	-	-	-	-
FGC13458	13908115	2	2	+	-	-	-	-	+	-	-	-	-	-
S18823	14695706	2	3	-	-	-	-	-	-	+	-	-	+	-
CTS7658	15482393	2	3	-	-	-	-	-	-	+	-	-	+	-

Sort subsequent rows from first to last positive result. Check results consistent: last positive row is the same. Reject if not.

Current row = 3
Last positives all same

Count	First	75848	22654	224096	584862	284744	65049	120386	274005	289065	B75780	5962	216600	N4826
S1911	21003759	7	1	+	+	+	+	+	+	-	-	-	-	-
FGC13445	4484946	3	1	+	+	+	+	+	-	-	-	-	-	-
S1894	16098109	3	4	-	-	-	+	+	-	-	-	-	-	-
S1900	17358211	3	4	-	-	-	+	+	+	-	-	-	-	-
FGC13446	7678200	2	2	+	-	+	-	-	-	-	-	-	-	-
FGC13458	13908115	2	2	+	-	+	-	-	-	-	-	-	-	-
S18823	14695706	2	3	-	-	-	-	-	-	+	-	-	+	-
CTS7658	15482393	2	3	-	-	-	-	-	-	+	-	-	+	-

Group everyone with zero positive variants together

Count	First	75848	224096	22654	584862	284744	120386	65049	274005	289065	216600	B75780	5962	N4826
S1911	21003759	7	1	+	+	+	+	+	+	-	-	-	-	-
FGC13445	4484946	3	1	+	+	+	-	-	-	-	-	-	-	-
S1894	16098109	3	4	-	-	-	+	+	+	-	-	-	-	-
S1900	17358211	3	4	-	-	-	+	+	+	-	-	-	-	-
FGC13446	7678200	2	1	+	+	-	-	-	-	-	-	-	-	-
FGC13458	13908115	2	1	+	+	-	-	-	-	-	-	-	-	-
S18823	14695706	2	9	-	-	-	-	-	-	+	+	-	-	-
CTS7658	15482393	2	9	-	-	-	-	-	-	+	+	-	-	-

Add some new variants
(All have a full set of calls but many FAIL)

Count	First	75848	224096	22654	584862	284744	120386	65049	289065	216600	274005	B75780	5962	N4826
S1911	21003759	7	1	+	+	+	+	+	+	+	-	-	-	-
FGC13445	4484946	3	1	+	+	+	-	-	-	-	-	-	-	-
S1894	16098109	3	4	-	-	-	+	+	+	-	-	-	-	-
S1900	17358211	3	4	-	-	-	+	+	+	-	-	-	-	-
FGC13446	7678200	2	1	+	+	-	-	-	-	-	-	-	-	-
FGC13458	13908115	2	1	+	+	-	-	-	-	-	-	-	-	-
S18823	14695706	2	9	-	-	-	-	-	-	+	+	-	-	-
CTS7658	15482393	2	9	-	-	-	-	-	-	+	+	-	-	-

ZS9483	19123304	2?	-	-	-	-	-	-	-	-	?	-	?	+	-	+	-
-	26662460	2?	-	-	-	-	-	-	-	-	+	-	?	-	?	+	?
CTS11041	20752203	7?	-	+	+	?	?	+	+	+	+	?	-	?	-	-	?
BY28746	26662435	5?	-	?	-	+	+	+	+	+	?	+	-	?	-	-	?
BY28749	26662459	5?	-	?	-	+	+	+	+	+	?	+	-	?	-	-	?
BY28747	26662448	4?	-	?	-	-	+	+	?	?	?	+	-	?	-	-	?