Exploration of Shared SNPs in Thaps
Chr1-qfiltered

July 26, 2017

Rambling exploration of SNP positions shared between two or more of the isolates. Code is included to document it thoroughly, (even if largely uninteresting to anyone else), and I will summarize it as I go.

Contents

1 History

This was added to SVN 1/26/2014; not sure when it was started, but earliest related emails I see are from 1/21/14.

```
r413 | ruzzo | 2014-01-26 08:22:37 -0800 (Sun, 26 Jan 2014) | 2 lines

adding shared-snp analysis.
```

2 Preliminaries

NOTE: Some comments in code and some parts of the text, especially specific numbers and general conclusions, are based on Unqfiltered, Chr1, Medium stringency (i.e., "[[2]]" below) analysis. The broad picture does not appear to change with other choices, but details do, and the text is neither fully parameterized nor fully updated, so proceed with caution.

Load utility R code; do setup:

```
source('../../../R/wlr.R') # load util code; path relative this folder or sibling in scripts/larrys

## Running as: ruzzo @ bicycle.cs.washington.edu; SVN Id, I miss you.  $Id: wlr.R  2017-07-21 or later $

setup.my.wd('shared-snps') # set working dir; UPDATE if this file moves, or if COPY/PASTE to new file
setup.my.knitr('figs-knitr/')
generic.setup('figs-mine/')
```

3 Major Analysis/Performance Parameters.

Choices here control how this file is processed, what data is analyzed, speed, etc. Set them carefully before running "make." Major choices are:

1. WHICH SNP TABLES ARE LOADED??? The logical vector `load.tb` selects the desired combination of SNP tables to load, in the order `full.unfiltered, chr1.unfiltered, full.qfiltered, chr1.qfiltered`. E.g., `load.tb=(T, F, T, F)` loads *full* tables for *both* q- and un-qfiltered data. Primary analysis is only performed on one of them, but the others are retained for comparison/debugging.

2. WHICH MAIN ANALYSIS??? If multiple tables are loaded, which is used for the main analysis? Parameter `pri` is a permutation of 1:4, corresponding to `load.tb`; the first loaded table in that order becomes the analysis focus. The default `pri=c(1,2,3,4)` looks at un-q-filtered data in preference to q-filtered, and full tables in preference to Chr1 within each group.

   (Choice of data for the "Table 1" coverage summary in section 5 is independent of this; full genome data is prefered over Chr 1 for both q- and unq-filtered reads; change `tset.picker` calls near the end of that section to modify this.)

3. CLEAR CACHE??? `clear.cache=T` forces Knitr cache removal at the start of the run; especially important if the previous parameters have changed since the last run.

4. HOW MANY BOOTSTRAP REPLICATES??? The variable `nboot` is a major performance factor; 1000 reps takes several hours. Set to 5 for debug and quick look; 100 or more for final run.

5. TRUNCATE TABLES TO Chrs ONLY??? I.e., remove mitochondrial-, plastid-, and BD- contigs.

The following code chunk sets the first four parameters based on where it's run. To prototype/debug on a laptop, faster is better—run on Chr1 with small `nboot`; when run on the linux servers, I typically do full genomes, more replicates. Just override them if these defaults don't work for you.

```
# for Makefile, params can be command line args, else base on system; see wlr.r for details.
# load.tb order: full.un, chr1.un, full.qfil,  chr1.qfil

params <- pick.params(
  mac   = list(load.tb=c(F,T,F,F), pri=1:4, clear.cache=F, nboot=  1, trunc.tables=T), # quick on lap
 #linux = list(load.tb=c(F,F,F,T), pri=1:4, clear.cache=F, nboot=  5, trunc.tables=T), # quick qfil on server
  linux = list(load.tb=c(T,F,T,F), pri=1:4, clear.cache=T, nboot=101, trunc.tables=T)  # full on server
)

# Alternatively, edit/uncomment the following to override the above as needed
#params<-pick.params(default=list(load.tb=c(T,T,T,T),pri=1:4,clear.cache=T,nboot=1000,trunc.tables=T))
print(params)

# $load.tb
# full.unf chr1.unf  full.qf  chr1.qf
#    FALSE     TRUE    FALSE     TRUE
#
# $pri
# [1] 3 4 1 2
#
# $clear.cache
# [1] TRUE
#
# $nboot
# [1] 5
#
# $trunc.tables
# [1] FALSE
```

CLEAR CACHE??!! Some code chunks use the knitr cache, but extent of cache consistency checks unknown. If in doubt, delete "cache/" (knitr's) directory to force rebuild. T/F set in params above will/won't force removal (actually, rename):

```
decache(params$clear.cache)

# Rename of 'cache' to 'cache96437' returned TRUE .
```

If still in doubt, also manually remove "00common/mycache/" (mine).

Load the main SNP data file(s) based on the parameters set in section 3.

```
# short names to keep the following chunk compact
tb <- params$load.tb
tset <- list(NULL, NULL, NULL, NULL) # tset = 'table set'
```

```
# see wlr.R for load paths
if(tb[1]){tset[[1]] <- load.snp.tables(use.chr1.tables = FALSE, data.name='full.tables.01.26.14')}
if(tb[2]){tset[[2]] <- load.snp.tables(use.chr1.tables = TRUE , data.name='full.tables.01.26.14')}

# Loading ../00common/mycache/snp.tables.chr1.unqfiltered.rda ...Loaded.

if(tb[3]){tset[[3]] <- load.snp.tables(use.chr1.tables = FALSE, data.name='full.tables.02.25.15')}
if(tb[4]){tset[[4]] <- load.snp.tables(use.chr1.tables = TRUE , data.name='full.tables.02.25.15')}

# Loading ../00common/mycache/snp.tables.chr1.qfiltered.rda ...Loaded.
# Bandaiding qfiltered tables...
```

Grrr! I should have excluded non-Chr contigs from full genome runs. Rather than change tons of code below to add mask params, I'm just going to truncate the tables, as follows. (See notes in wlr.r::make.mask for assumptions.)

```
if(params$trunc.tables){
  for(i in 1:4){
    if(!is.null(tset[[i]])){
      first.mito <- match("mitochondria.fasta", tset[[i]][[7]]$Chr)
      if(!is.na(first.mito)){ # will be NA for Chr1 tables
```

```
        for(j in 1:7){
          # hmmm... slow; wonder whether head(tset[[i]][[j]],first.mito-1) is faster;
          # ok, simple tests suggest not: system.time(head(data.frame(1:1e7,1:1e7),5e6))
          tset[[i]][[j]] <- tset[[i]][[j]][1:(first.mito-1),]
        }
      }
    }
  }
} else {
    cat('***\n*** DID YOU *REALLY* WANT UNTRUNCATED TABLES???\n***\n')
}

# ***
# *** DID YOU *REALLY* WANT UNTRUNCATED TABLES???
# ***
```

The tersely-named `tset` list is sometimes convenient, but give them more descriptive names, too.

```
snp.tables.full.unfiltered <- tset[[1]]; names(tset)[1] <- 'snp.tables.full.unfiltered'
snp.tables.chr1.unfiltered <- tset[[2]]; names(tset)[2] <- 'snp.tables.chr1.unfiltered'
snp.tables.full.qfiltered  <- tset[[3]]; names(tset)[3] <- 'snp.tables.full.qfiltered'
snp.tables.chr1.qfiltered  <- tset[[4]]; names(tset)[4] <- 'snp.tables.chr1.qfiltered'
```

Main analysis may just use one of the potentially 4 table sets. Pick it according to the priority specified in section 3, using the shorter name 'snp.tables' for this default choice.

```
snp.tables <- tset.picker(priority=params$pri, table.set=tset)
```

```
# Sanity check: unlike unqfiltered tables, bug in early code gave qfiltered ones different numbers
# of rows per strain, which breaks much code.  Verify this is no longer happening.
check.eq.nrows <- function(tables){
  if(!is.null(tables)){
    nrow.snp.tables <- unlist(lapply(tables,nrow))
    print(nrow.snp.tables)
    if(all(nrow.snp.tables == nrow.snp.tables[1])){
      cat('OK, all strains have same number of rows.\n')
    } else {
      cat('***\n*** Warning: Different strains have different numbers of rows! ***\n***\n')
    }
  }
}

dummy<-lapply(tset, check.eq.nrows)

#    1007    1012    1013    1014    1015    3367    1335
# 3042585 3042585 3042585 3042585 3042585 3042585 3042585
# OK, all strains have same number of rows.
#    1007    1012    1013    1014    1015    3367    1335
# 3042585 3042585 3042585 3042585 3042585 3042585 3042585
# OK, all strains have same number of rows.
```

Which tables have we got?:

```
# 'which.snp.tables' return summary of which tables, either as a char string (default), e.g.
# "Chr1-qfiltered", or as vector of 2 strings, e.g. c("full","unfiltered").
cat('This analysis uses: (', paste(unlist(lapply(tset,which.snp.tables)),collapse=', '), ') SNP tables.\n')

# This analysis uses: ( NULL, Chr1-unfiltered, NULL, Chr1-qfiltered ) SNP tables.

cat('Main shared SNP analysis focuses on', which.snp.tables(snp.tables), '\n')

# Main shared SNP analysis focuses on Chr1-qfiltered
```

A LaTeX hack: I want which.snp.tables info in doc title/page headers, but it is unknown until now, so the following writes a command definition `\whichsnptables` into the .aux file, which is read during the *next* LaTeX run, when `\begin{document}` is processed:

```
\makeatletter
\immediate\write\@auxout{\noexpand\gdef\noexpand\whichsnptables{Chr1-qfiltered}}
\makeatother
```

Subsequent analysis was initially all directed at Chr1. In general, I have *not* updated the discussion to reflect genome-wide analysis.

```
if(exists('snp.tables.chr1.qfiltered') && exists('snp.tables.chr1.unqfiltered')){
  # If have both, where is new unequal to old?
  uneq <- snp.tables.chr1.qfiltered[[1]]$Ref[1:chr1.len] != snp.tables.chr1.unqfiltered[[1]]$Ref[1:chr1.len]
  cat('Sum uneq:', sum(uneq,na.rm=T), '\n')
  cat('Sum NA:  ', sum(is.na(uneq)),  '\n')
  print(which(is.na(uneq))[1:10])
  seecounts(which(is.na(uneq))[1:4],snp.tables=snp.tables.qfiltered,debug=F)
}
```

In brief, “`snp.tables`” will be a list of 7 data frames, one per strain, giving read counts for each nucleotide at each position, SNP calls, etc.:

```
names(snp.tables)

# [1] "1007" "1012" "1013" "1014" "1015" "3367" "1335"

str(snp.tables[[1]])

# 'data.frame': 3042585 obs. of  16 variables:
#  $ snp   : int  0 0 0 0 0 0 0 0 0 0 ...
#  $ Chr   : chr  "Chr1" "Chr1" "Chr1" "Chr1" ...
#  $ Pos   : int  1 2 3 4 5 6 7 8 9 10 ...
#  $ Ref   : chr  "T" "C" "C" "A" ...
#  $ Cov   : num  0 2 3 4 4 4 7 8 9 10 ...
#  $ a     : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ g     : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ c     : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ t     : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ n     : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ .match: num  0 2 3 4 4 4 7 8 9 10 ...
#  $ exon  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
#  $ indel : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
#  $ chr   : Factor w/ 1 level "Chr1": 1 1 1 1 1 1 1 1 1 1 ...
#  $ pos   : int  1 2 3 4 5 6 7 8 9 10 ...
#  $ rawCov: num  1 3 4 5 7 7 10 12 13 15 ...
```

Just for background, also load the desert tables:

```
# from svn+ssh://ceg1.ocean.washington.edu/var/svn/7_strains/trunk/code/snpNB/data
#load('../../../data/ungit-data/des.rda')
load('../../../data/des.rda')
```

What’s the total length of all deserts in each strain? Big deserts (defined as “big.threshold” or longer)?

```
some.desert.stats <- function(big.threshold=0){
  desert.len <- unlist(lapply(des,function(x){sum(unlist(lapply(x,function(y){sum(y[,'Length'])})))}))
  bigdes.len <- unlist(lapply(des,function(x){sum(unlist(lapply(x,function(y){
                                                sum(y[y[,'Length']>=big.threshold,'Length'])})))}))
  rbind(desert.len, desert.pct=round( desert.len / genome.length.constants()$genome.length.trunc * 100),
        bigdes.len, bigdes.pct=round( bigdes.len / genome.length.constants()$genome.length.trunc * 100))
}
some.desert.stats(big.threshold=50000)

#               tp1007   tp1012  tp1013  tp1014   tp1015 thapsIT   tp1335
# desert.len 11146526 11332566 5801763 9464213 11251426 6780300 10883723
# desert.pct       36       36      19      30       36      22       35
# bigdes.len  3495805  3936973   55365 3627235  3727061   57119  4046934
# bigdes.pct       11       13       0      12       12       0       13
```

I.e., looking at all deserts, about 1/3 of L-clade, 1/5 of H-clade are in deserts, whereas, looking at the largest deserts ($> 50k$), only about 12% in L-clade (and none in H-clade). Note that the rough stats above include artifactual "deserts" created by gaps in the reference sequence, large genomic deletions, etc. A more careful analysis of this is found in nc-snps.rnw.

4 Refined SNP Calls

4.1 Method

It is appropriate that SNP calls should be conservative, to avoid many false positives, but, when a position is called a SNP in one isolate, we often see a significant number of reads for the same non-reference nucleotide at that position in other isolates, even if they are not called as SNPs. On the other hand, we sometimes see a position called a SNP in two or more isolates, but with *different* pairs of nucleotides, potentially suggesting technical errors. Analysis in this section attempts to refine the SNP calls by looking for issues such as these by looking at all 7 isolates jointly, at each position called a SNP in any of them.

For a given strain, the following function returns a vector of 0:4 to indicate which nonreference nucleotide has the maximum read count at the corresponding position. The values 1..4 indicate that the max count occurred at A, G, C, T, resp. (Ties are resolved arbitrarily ($a < g < c < t$), which possibly deserves further attention.) The value 0 means all nonreference counts are below threshold, based *either* on absolute count *or* as a fraction of coverage. Default only excludes 0 counts.

```
nref.nuc.new <- function(strain=1, mask=T, thresh.count=0, thresh.rate=0.0){
        # get read count for max nonref nuc
        nref <- apply(snp.tables[[strain]][mask, c('a', 'g', 'c', 't')], 1, max)
        # where does nref count match a (g,c,t, resp) count
        as <- ifelse(nref == snp.tables[[strain]][mask,'a'],1,0)
        gs <- ifelse(nref == snp.tables[[strain]][mask,'g'],2,0)
        cs <- ifelse(nref == snp.tables[[strain]][mask,'c'],3,0)
        ts <- ifelse(nref == snp.tables[[strain]][mask,'t'],4,0)
        # most positions will show 3 zeros and one of 1:4, so max identifies max nonref count;
        # ties broken arbitrarily  (a<g<c<t)
        merge <- pmax(as,gs,cs,ts)
        # but if max nonref count is zero or below threshold, return 0
        merge[nref == 0 | nref < thresh.count] <- 0
        merge[nref/snp.tables[[strain]][mask,'Cov'] < thresh.rate] <- 0
        return(merge)
}
```

Get union and intersection of the sets of called SNPs. ("$snp" is 0/1.) Also, 5-way (L-clade) and 4-way (L-excluding Gyre).

```
# 4-way union/intersection
u4.snps <- snp.tables[[1]]$snp
i4.snps <- snp.tables[[1]]$snp
for(i in c(2,5,7)) {
        u4.snps <- pmax(u4.snps, snp.tables[[i]]$snp)
        i4.snps <- pmin(i4.snps, snp.tables[[i]]$snp)
}
# 5-way: add gyre
u5.snps <- pmax(u4.snps, snp.tables[[4]]$snp)
i5.snps <- pmin(i4.snps, snp.tables[[4]]$snp)
# 7-way
union.snps     <- pmax(u5.snps, snp.tables[[3]]$snp, snp.tables[[6]]$snp)
intersect.snps <- pmin(i5.snps, snp.tables[[3]]$snp, snp.tables[[6]]$snp)
nu4snps <- sum(u4.snps)
nu5snps <- sum(u5.snps)
ni4snps <- sum(i4.snps)
ni5snps <- sum(i5.snps)
nusnps  <- sum(union.snps)
nisnps  <- sum(intersect.snps)
c(n4u=nu4snps, n5u=nu5snps, n7u=nusnps, n4i=ni4snps, n5i=ni5snps, n7i=nisnps)
```

```
#   n4u   n5u   n7u   n4i   n5i   n7i
# 18564 18696 47499 14365  7628  1641
```

There are nusnps=47499 positions called as SNPs in one or more strains (but only nisnps=1641 that are shared among all 7). Note that the 4-way union is only modestly larger (1.2923077 times larger) than the 4-way intersection, emphasizing the inherent similarities among these SNP sets. The corresponding 5-way numbers show that Gyre adds relatively little to the 5-way union vs the 4-way union, whereas it removes a fair bit from the 5-way intersection. However, much of that loss is simply because Gyre has fewer called SNPs: only 8331 vs 14365 in the 4-way intersection, and they are highly concordant:

```
sum(snp.tables[[4]]$snp*i4.snps)/sum(snp.tables[[4]]$snp)

# [1] 0.9156164
```

So, a likely source of the Gyre's difference in called SNPs is technical (lower read coverage, higher read error rate) rather than biological.

Inclusion of the 2 H-clade members, however, causes more dramatic changes in both union and intersection numbers. I examine all these relationships in more detail below, but first I examine what I believe to be a significant source of technical error in these comparisons—erroneous SNP calls, especially false negative calls.

It is appropriate that SNP calls should be conservative, to avoid many false positives, but, if a position is called a SNP in one strain, we often see a significant number of reads for the same non-reference nucleotide at that position in other strains, even if they are not called as SNPs. For my purposes below, these will be considered "shared SNPs," based on three different levels of permissiveness. Note that, e.g., $\geq$ 84% of all positions have zero reads for any non-reference nucleotide, and only a small fraction have 2 or more non-reference reads:

```
nonmatch <- rbind(
  unlist(lapply(snp.tables,function(x){sum(x$Cov-x$.match == 0)})),
  unlist(lapply(snp.tables,function(x){sum(x$Cov-x$.match == 1)})),
  unlist(lapply(snp.tables,function(x){sum(x$Cov-x$.match == 2)})),
  unlist(lapply(snp.tables,function(x){sum(x$Cov-x$.match == 3)})),
  unlist(lapply(snp.tables,function(x){sum(x$Cov-x$.match >= 4)})),
  unlist(lapply(snp.tables,function(x){sum((x$Cov-x$.match)[union.snps==0] >= 4)}))
)/nrow(snp.tables[[1]])*100
rownames(nonmatch) <- c('% ==0','% ==1','% ==2','% ==3','% >=4', '% >=4, nonSNP')
nonmatch

#                      1007        1012        1013        1014        1015      3367        1335
# % ==0         97.79158183 97.46508314 95.61527451 97.40214324 97.30475895 96.0503322 96.61320883
# % ==1          1.45497989  1.70969751  2.91324647  2.03304098  1.83551158  2.5290008  2.48311880
# % ==2          0.09754863  0.10606113  0.19447279  0.17521285  0.11904351  0.1896085  0.18402115
# % ==3          0.05166659  0.04680231  0.08019497  0.09935630  0.05324420  0.0796691  0.05416447
# % >=4          0.60422305  0.67235591  1.19681126  0.29024662  0.68744176  1.1513894  0.66548675
# % >=4, nonSNP  0.04486317  0.07924183  0.19759514  0.02635916  0.08581519  0.1951959  0.08811586
```

Build a table of max non-reference nucleotides at each position in the union.snps set. The three criteria are

- [[1]]: any non-zero count at any coverage is considered significant
- [[2]]: (count $\geq$ 2 and count/coverage $\geq$ 0.05) is considered significant
- [[3]]: (count $\geq$ 4 and count/coverage $\geq$ 0.10) is considered significant

In all three cases, the nonref nucleotide must also be consistent across all strains passing that threshold; see below.

```
non.refs <- vector('list',4)
for(i in 1:4){
  non.refs[[i]] <- matrix(0, nrow=nusnps, ncol=7)
  colnames(non.refs[[i]]) <- names(snp.tables)
  rownames(non.refs[[i]]) <-
            paste(snp.tables[[1]]$chr[union.snps==1], ':', snp.tables[[1]]$pos[union.snps==1], sep='')
}
for(j in 1:7){
  non.refs[[1]][,j] <- nref.nuc.new(j, mask=union.snps==1, thresh.count=0, thresh.rate=0.00)
  non.refs[[2]][,j] <- nref.nuc.new(j, mask=union.snps==1, thresh.count=2, thresh.rate=0.05)
  non.refs[[3]][,j] <- nref.nuc.new(j, mask=union.snps==1, thresh.count=4, thresh.rate=0.10)
}
```

For comparison, I want to look at unfiltered SAMTools SNP calls. In complete opposition to the measures of consistency imposed above, I'm going to simply force this into the "non.refs" structure constructed above by imagining that any position called a SNP in any strain has its max nonref count on "A", so any given position called a SNP in any strain will automatically be declared "consistent." This will allow the tree-code, etc. given below to work in a uniform way (even though interpretation of the results is different.) Results will be jammed into a 4th component of the "non.refs" list; i.e., we have a 4th criterion:

- [[4]]: all called SNPs at a given position are considered "consistent."

As this case was a late addition to the analysis, the commentary throughout this document has not necessarily been updated to reflect that this case is distinct from the first three.

```
for(j in 1:7){
  non.refs[[4]][,j] <- snp.tables[[j]]$snp[union.snps==1]
}
```

```
str(non.refs[[4]])

#  num [1:47499, 1:7] 0 0 0 0 0 0 0 0 1 0 ...
#  - attr(*, "dimnames")=List of 2
#   ..$ : chr [1:47499] "Chr1:333" "Chr1:417" "Chr1:435" "Chr1:438" ...
#   ..$ : chr [1:7] "1007" "1012" "1013" "1014" ...
```

"non.refs" indicates, among those positions in the union of all called SNPS having any non-reference read count above the thresholds listed above, the non-ref nucleotide having the highest read count in each strain. If, for a given position, the max of this code is the same as the min (among non-zero values), then every strain having over-threshold nonref reads in that position, in fact has most non-reference reads on the *same* nucleotide. These are defined as the "consistent" SNPs.

```
find.consistent <- function(nr){
  nr.max <- apply(nr,1,max)
  nr.min <- apply(nr,1,function(x){ifelse(max(x)==0,0,min(x[x>0]))})
  return(nr.min == nr.max)
}
consistent  <- lapply(non.refs, find.consistent)
```

4.2 Save them

```
# wrap this in a data structure to be cached:
Description <- [2753 chars quoted with ''']

refined.snps <-
  list(Description=Description,

       Data=list(
         based.on.which.snp.tables=which.snp.tables(),
         number.union.snps=nusnps,
         number.intersection.snps=nisnps,
         non.ref.nucleotide=non.refs,
         consistent.snps=consistent),

       Code=list(
         get.snps = function(strain, stringency=2){
           # return nusnps x 1 Bool vector of consistent SNPs @ specified stringency & strain
           return(refined.snps$Data$consistent.snps[[stringency]] &
                  refined.snps$Data$non.ref.nucleotide[[stringency]][,strain] > 0)
         }
         ,
         get.snp.locs.char = function(strain, stringency=2){
           # return char vector of locations of consistent SNPs @ specified stringency & strain
           snps <- refined.snps$Code$get.snps(strain, stringency)
           return(names(snps)[snps])
         }
         ,
         get.snp.locs.df = function(strain, stringency=2){
           # return data frame (Chr/Pos) of locations of consistent SNPs @ specified stringency & strain
           snplist <- strsplit(refined.snps$Code$get.snp.locs.char(strain, stringency), ':', fixed=TRUE)
           # strsplit returns long list of 2-vectors, 1st=chr, 2nd=char position
           df <- data.frame(Chr=           unlist(lapply(snplist,function(x){return(x[1])})),
                            Pos=as.integer(unlist(lapply(snplist,function(x){return(x[2])}))),
                            stringsAsFactors = FALSE)
           return(df)
         }
       )
  )

# dont't clobber existing .rda, but save if absent.  (delete to re-save)
# result for trunc, unfiltered tables saved to "data" else "mycache"
if(which.snp.tables() == 'trunc-unfiltered'){
  rda.refined <- '../../../data/refined.snps-trunc-unfiltered.rda'
} else {
  rda.refined <- paste('../00common/mycache/refined.snps', which.snp.tables(), 'rda', sep='.')
}
if(file.exists(rda.refined)){
  cat('Pre-existing file', rda.refined, 'unchanged.\n')
} else {
  cat('Saving', rda.refined, '...')
  save(refined.snps, file=rda.refined, compress=TRUE)
  cat('Saved.\n')
}

# Saving ../00common/mycache/refined.snps.Chr1-qfiltered.rda ...Saved.
```

Knitr seems to be failing to format the long char string above, which says:

```
cat(refined.snps$Description)

# Contents of this .rda file:
#
#   * Description: this text
#
#   * Data -- 5 items defining refined SNPs, at 4 different stringency levels, as defined
#     in shared-snps.rnw:
#
#     * based.on.which.snp.tables: {"Chr1","full","trunc"}-{"unfiltered","qfiltered"},
#       depending on which snp tables were used to build this data. ("trunc" = all Chrs.)
#
#     * number.union.snps: the total number of SNPs (SAMtools calls) in the union of SNPs
#       across all 7 strains.
#
#     * number.intersection.snps: similar, for the 7-way intersection.
#
#       nusnps/nisnps are easily recalculated from the data below, but their inclusion
#       may be convenient, e.g., to quickly see if the .rda represents the full genome
#       (nusnps=488848), or the chr 1 subset (nusnps=47499); (redundant with "based.on...";
#       numbers above are for unfiltered, perhaps slightly different if qfiltered)
#
#     * non.ref.nucleotide: 4 arrays, each nusnps x 7, of values 0..4 (0..1 in the 4th
#       array).  In the 1st 3 arrays, 0 means the given position in the given strain did
#       not have nonreference read counts above the corresponding filtering threshold,
#       i.e., is NOT a refined SNP in that strain, whereas 1..4 mean that it did pass
#       threshold, for A,C,G,T resp.  In the 4th array, this value is just 1/0,
#       indicating is/is not a called SNP in that strain.
```

```
#
#     * consistent.snps: 4 Bool vectors of length nusnps flagging positions whose nonref
#       nucs (wrt to the 4 filtering criteria) are deemed *consistent* across
#       all 7 strains.  For the 1st 3, this means all nonzero entries of non.ref.nuc
#       are equal, i.e., nonref read counts passing threshold are on the SAME nonref
#       nucleotide in all strains having over-threshold counts.  Just for comparison
#       and uniformity of data structures, the 4th is all TRUE, i.e., union of SNPs
#       across all strains, without any regard for thresholds or consistency.
#
#       In short, the refined SNPs according to our medium filtering criteria are
#       strains/positions where consistent.snps[[2]]==TRUE and non.ref.nucleotide[[2]]>0.
#
#       Rownames in both non.ref.nucs and consistent define location, e.g. "Chr1:333".
#
#   * Code -- simple routines to extract refined SNPs in (potentially) convenient formats:
#
#     * get.snps(strain, stringency=2)
#       returns nusnps x 1 Bool vector of consistent SNPs @ specified stringency in
#       given strain
#
#     * get.snp.locs.char(strain, stringency=2)
#       returns n x 1 char vector of locations of consistent SNPs @ specified stringency
#       in given strain, e.g. "Chr1:1234", where n == sum(get.snps(...))
#
#     * get.snp.locs.df(strain, stringency=2){
#       As above, but returns data frame (char vector Chr, int vector Pos) with the same info.
```

```
str(consistent[[1]])

#  Named logi [1:47499] TRUE FALSE TRUE TRUE TRUE TRUE ...
#  - attr(*, "names")= chr [1:47499] "Chr1:333" "Chr1:417" "Chr1:435" "Chr1:438" ...
```

```
consistent.count <- unlist(lapply(consistent, sum)) ; consistent.count

# [1] 44905 47108 47204 47499

inconsistent.count <- consistent.count[4] - consistent.count; inconsistent.count

# [1] 2594  391  295    0

inconsistent.percent <- inconsistent.count/consistent.count[4]*100; inconsistent.percent

# [1] 5.4611676 0.8231752 0.6210657 0.0000000
```

I.e., of the 47499 positions in which a SNP is called, 44905 are consistent by my loose definition, and 47204 are consistent by my tightest definition. The increase in concordance supports the view that the loose definition is too loose. Perhaps misleadingly, these counts include positions that are "consistent SNPs" in only one strain; more below. (*TODO* I suspect, but have not yet systematically checked, that most of the rest are positions with low coverage and/or very low read counts on the mixture of non-reference nucleotides.)

4.3 Examples: Consistent

Here are a few (nonrandomly selected) prototypical consistent SNPs:

```
esnps <- names(consistent[[2]][consistent[[2]]])
esnps2 <- as.integer(unlist(lapply(strsplit(esnps[c(7,11:13,92)],':',fixed=TRUE),function(x){x[2]})))
seecounts(esnps2,snp.tables=snp.tables)

#     chr  pos Ref Strain   A   G  C   T SNP  exon indel nrf rat
# 1  Chr1  567   T
# 2                  1007   0   0  1 25   0  TRUE FALSE
# 3                  1012   0   0 14 39   1  TRUE FALSE
# 4                  1013   0   0 13 87   0  TRUE FALSE
# 5                  1014   0   1  0 23   0  TRUE FALSE
# 6                  1015  0   0  8 40   1  TRUE FALSE
# 7                  3367  0   0 16 38   1  TRUE FALSE
# 8                  1335  0   0  2 99   0  TRUE FALSE
# 9  Chr1 1053   A
# 10                 1007 25   0  0  4   0  TRUE FALSE
# 11                 1012 35   0  0 12   0  TRUE FALSE
# 12                 1013  2   1  0 32   0  TRUE FALSE
# 13                 1014  5   0  0  5   0  TRUE FALSE
# 14                 1015 29   0  0 15   1  TRUE FALSE
# 15                 3367  2   0  0  7   0  TRUE FALSE
# 16                 1335 56   0  0 39   1  TRUE FALSE
# 17 Chr1 1055   G
# 18                 1007  0  23  0  1   0  TRUE FALSE
# 19                 1012  0  37  0  6   0  TRUE FALSE
# 20                 1013  1  39  0  6   0  TRUE FALSE
# 21                 1014  0   6  0  2   1  TRUE FALSE
# 22                 1015  0  26  0 14   0  TRUE FALSE
# 23                 3367  0  12  0  0   0  TRUE FALSE
# 24                 1335  0  54  0 32   1  TRUE FALSE
# 25 Chr1 1176   G
# 26                 1007  1  53  0  0   0 FALSE FALSE
# 27                 1012  0  54  0  0   0 FALSE FALSE
# 28                 1013 19  56  0  0   0 FALSE FALSE
# 29                 1014  0  28  0  0   0 FALSE FALSE
# 30                 1015  3  85  0  0   0 FALSE FALSE
# 31                 3367  9   2  0  0   1 FALSE FALSE
# 32                 1335  0 156  0  0   0 FALSE FALSE
# 33 Chr1 8685   G
# 34                 1007  6  15  0  0   0  TRUE FALSE
# 35                 1012 10  23  0  0   0  TRUE FALSE
# 36                 1013 18  21  0  0   1  TRUE FALSE
# 37                 1014  4   8  0  0   0  TRUE FALSE
# 38                 1015 10  24  0  0   1  TRUE FALSE
# 39                 3367  0   4  0  0   0  TRUE FALSE
# 40                 1335  5  32  0  0   0  TRUE FALSE
```

4.4 Examples: Inconsistent

Here is a brief look at some *in*-consistent positions. E.g., Chr1:2013 shows nontrivial counts on 3 alleles in Wales, as do 2319, 3286, 5002, 5433, whereas 7878 shows a different alternate allele in Italy than in Wales.

```
unc <- names(consistent[[2]][!consistent[[2]]])
unc2 <- as.integer(unlist(lapply(strsplit(unc[1:10],':',fixed=TRUE),function(x){x[2]})))
seecounts(unc2,snp.tables=snp.tables)

#     chr   pos Ref Strain  A   G  C   T SNP  exon indel nrf rat
# 1  Chr1  2013   T
# 2                   1007  4   0  0 15   0  TRUE FALSE
# 3                   1012  6   0  0 21   0  TRUE FALSE
# 4                   1013  7  10  0  6   1  TRUE FALSE
# 5                   1014  1   0  0  6   0  TRUE FALSE
# 6                   1015 13   0  0 13   1  TRUE FALSE
# 7                   3367  7   0  0 25   0  TRUE FALSE
# 8                   1335 16   0  0 42   1  TRUE FALSE
# 9  Chr1  2319   C
# 10                  1007  0  28 10  0   1  TRUE FALSE
# 11                  1012  0  43 17  0   1  TRUE FALSE
# 12                  1013 13  15  9  0   1  TRUE FALSE
# 13                  1014  0  18  6  0   1  TRUE FALSE
# 14                  1015  0  53 20  0   1  TRUE FALSE
# 15                  3367  4   0 24  0   0  TRUE FALSE
# 16                  1335  0 118 28  0   1  TRUE FALSE
# 17 Chr1  3286   T
# 18                  1007  4   0  1 10   0  TRUE FALSE
# 19                  1012  7   0  3 32   0  TRUE FALSE
# 20                  1013 34   0 30  1   1  TRUE FALSE
# 21                  1014  4   0  4 10   0  TRUE FALSE
# 22                  1015 11   0  6 31   0  TRUE FALSE
# 23                  3367  5   0 29  0   0  TRUE FALSE
# 24                  1335 14   0  3 55   0  TRUE FALSE
# 25 Chr1  5002   T
# 26                  1007  0  14  0  7   0  TRUE FALSE
# 27                  1012  0  20  0 19   1  TRUE FALSE
# 28                  1013 18  10  0 22   0  TRUE FALSE
# 29                  1014  0   5  0  2   0  TRUE FALSE
# 30                  1015  0  18  0 12   1  TRUE FALSE
# 31                  3367  0   0  0 31   0  TRUE FALSE
# 32                  1335  0  46  0 44   0  TRUE FALSE
# 33 Chr1  5178   G
# 34                  1007  0  20  0  0   0  TRUE FALSE
# 35                  1012  0  32  0  0   0  TRUE FALSE
# 36                  1013 47   9  0  0   1  TRUE FALSE
# 37                  1014  0  13  0  0   0  TRUE FALSE
# 38                  1015  0  30  0  0   0  TRUE FALSE
# 39                  3367 32  19  0  0   1  TRUE FALSE
# 40                  1335  0  38  0  2   0  TRUE FALSE
# 41 Chr1  5433   G
# 42                  1007  0  40  0  3   0  TRUE FALSE
# 43                  1012  0  53  0  5   0  TRUE FALSE
# 44                  1013 16  29  0  7   1  TRUE FALSE
# 45                  1014  9   8  0  0   1  TRUE FALSE
# 46                  1015  6  53  0  2   0  TRUE FALSE
# 47                  3367  8  37  0  0   0  TRUE FALSE
# 48                  1335  6  72  0  2   0  TRUE FALSE
# 49 Chr1  7858   C
# 50                  1007  0   0 42  0   0  TRUE FALSE
# 51                  1012  0   0 35  0   0  TRUE FALSE
# 52                  1013  0   0 81  8   0  TRUE FALSE
# 53                  1014  0   0 12  0   0  TRUE FALSE
# 54                  1015  0   0 71  0   0  TRUE FALSE
# 55                  3367 20   0  2  0   1  TRUE FALSE
# 56                  1335  0   0 83  0   0  TRUE FALSE
# 57 Chr1  8974   C
# 58                  1007  0   1  5  0   0  TRUE FALSE
# 59                  1012  0   2 13  0   0  TRUE FALSE
# 60                  1013  9  15  2  0   1  TRUE FALSE
# 61                  1014  0   1  1  0   0  TRUE FALSE
# 62                  1015  0   1  9  0   0  TRUE FALSE
# 63                  3367  2   0  1  0   0  TRUE FALSE
# 64                  1335  0  11 30  0   0  TRUE FALSE
# 65 Chr1 10099   T
# 66                  1007 16   0  0 24   0  TRUE FALSE
# 67                  1012 45   0  0 26   0  TRUE FALSE
# 68                  1013  0   2  6 55   0  TRUE FALSE
# 69                  1014 32   0  0 11   0  TRUE FALSE
# 70                  1015 38   0  0 37   0  TRUE FALSE
# 71                  3367  0   1  0  7   0  TRUE FALSE
# 72                  1335 52   0  0 61   1  TRUE FALSE
# 73 Chr1 15154   A
# 74                  1007 13   0  0  0   0 FALSE FALSE
# 75                  1012 37   0  0  1   0 FALSE FALSE
# 76                  1013  2   0 35  7   1 FALSE FALSE
# 77                  1014 10   0  0  0   0 FALSE FALSE
# 78                  1015 24   0  0  0   0 FALSE FALSE
# 79                  3367  3   0  0 12   1 FALSE FALSE
# 80                  1335 47   0  0  3   0 FALSE FALSE
```

4.5 Examples: Homozygous nonref

And at some *homozygous nonreference* positions (defined to be those with nonref fraction > 0.75):

```
hnr <- lapply(snp.tables, function(x){x$.match/x$Cov < 0.25})        # find them
hnr <- lapply(hnr, function(x){ifelse(is.na(x),FALSE,x)})            # remove NA
unlist(lapply(hnr,sum))                                              # count per strain

#  1007  1012  1013  1014  1015  3367  1335
#   316   247 12082   938   236 13863   167
```

Hmm, in L-clade, excluding the ref isolate (1335) this tracks time-in culture to some degree; Maybe many of these are in hemizygous regions. Next two chunks lifted from nc-snps to get tables for hemi-deletion.

```
cnv.chronly <- load.cnv.tables('../../../data/cnv.txt', chrs.only=TRUE)

str(cnv.chronly)

# 'data.frame': 1956 obs. of  11 variables:
#  $ strain   : Factor w/ 7 levels "IT","tp1007",..: 3 3 3 3 3 3 3 3 3 3 ...
#  $ chr      : Factor w/ 65 levels "BD1_7","BD10_65",..: 38 38 38 38 38 38 38 38 38 38 ...
#  $ start    : int  10601 112001 215001 358901 536501 554801 673401 781801 806901 853201 ...
#  $ end      : int  13500 116500 221100 370300 538600 559300 685000 787400 811100 855600 ...
#  $ length   : int  2900 4500 6100 11400 2100 4500 11600 5600 4200 2400 ...
#  $ filtered : logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
#  $ type     : Factor w/ 1 level "CNVnator": 1 1 1 1 1 1 1 1 1 1 ...
#  $ cov_ratio: num  0.63738 1.54893 1.65381 0.00204 0.68486 ...
#  $ dup_frac : num  0.41188 0.00908 0.01178 0.97997 0.0211 ...
#  $ iStart   : num  10601 112001 215001 358901 536501 ...
#  $ iEnd     : num  13500 116500 221100 370300 538600 ...

cnv.chronly[c(1:4,nrow(cnv.chronly)+c(-1,0)),]                     ## first/last few rows

#      strain   chr  start    end length filtered     type  cov_ratio   dup_frac   iStart     iEnd
# 1    tp1012  Chr1  10601  13500   2900    FALSE CNVnator 0.63738000 0.41187900    10601    13500
# 2    tp1012  Chr1 112001 116500   4500    FALSE CNVnator 1.54893000 0.00907677   112001   116500
# 3    tp1012  Chr1 215001 221100   6100    FALSE CNVnator 1.65381000 0.01178470   215001   221100
# 4    tp1012  Chr1 358901 370300  11400     TRUE CNVnator 0.00204431 0.97997300   358901   370300
# 1955 tp1335 Chr24 259901 278000  18100    FALSE CNVnator 1.41458000 0.38091100 31264334 31282433
# 1956 tp1335 Chr24 286901 289800   2900    FALSE CNVnator 1.74941000 0.74228100 31291334 31294233
```

```
get.cnv.dels <- function(cov.thresh.lo = 0.0,
                         cov.thresh.hi = 0.8,
                         cnv,
                         snp.tables = NULL,
                         DEBUG = FALSE
){
  # build list of 7 Bool vectors of genome length, with i-th == T iff
  #   * i-th pos is 'NA' in genome seq (if snp.tables are provided), or
  #   * in CNVnator call for coverage in half-open [cov.thresh.lo, hi), and
  #   * not marked 'filtered' by CNVnator
  cnv.deletions <- vector(mode='list',7)                 # make list of bool vectors
  if(is.null(snp.tables)){
    # if no tables, assume full
    t.len <- genome.length.constants()$genome.length.trunc
  } else {
    t.len <- nrow(snp.tables[[1]])
  }
  for(st in 1:7){
    if(is.null(snp.tables)){
      cnv.deletions[[st]] <- logical(t.len)                       # all F
    } else  {
      cnv.deletions[[st]] <- is.na(snp.tables[[st]]$Pos[1:t.len])   # NA positions in genome
    }
  }
  strain.names <- c(paste('tp10',c('07',12:15),sep=''),'IT','tp1335')
  names(cnv.deletions) <- strain.names
  for(i in 1:nrow(cnv)){
    if(!cnv$filtered[i] &&
        cnv$cov_ratio[i] >= cov.thresh.lo &&
       cnv$cov_ratio[i] <  cov.thresh.hi)
    {
      if(DEBUG){
        print(cnv[i,])
        print(as.character(cnv$strain[i]))
      }
      # following ASSUMES no CNVnator call crosses a chromosome bdry, & that
      # t.len ends at chr end (typically chr1 or chr24)
      if(cnv$iEnd[i] <= t.len){
        cnv.deletions[[as.character(cnv$strain[i])]][cnv$iStart[i]:cnv$iEnd[i]] <- TRUE
      }
    }
  }
  return(cnv.deletions)
}

# sanity check:
cnv.dels.38 <- get.cnv.dels(0.3, 0.8, cnv.chronly, snp.tables = NULL)
unlist(lapply(cnv.dels.38,sum)) # does it match low.length.38 in tic ?

#  tp1007  tp1012  tp1013  tp1014  tp1015      IT  tp1335
# 1672500 1781500 1383600 1313700  988400  320900 1453000

# 1672500 1781500 1399400 1313700 988400 336500 1453000 <== low.length.38 from tic (circa page 8)
# 1672500 1781500 1399400 1313700 988400 336500 1453000 <== low.length.38 from tic (pg9, 6/28/17)
rm(cnv.dels.38)
```

Slight discrepancy in H-clade that I should hunt down, but basically OK. (hmm; maybe untrunc tbls.)

```
# the ones we want for the current analysis:
hemi.masks <- get.cnv.dels(0.3, 0.8, cnv.chronly, snp.tables=snp.tables)

rbind(
  homnr        = unlist(lapply(hnr,sum)),
  hemi         = unlist(lapply(hemi.masks, sum)),
  homnr.unhemi = unlist(lapply(list(1,2,3,4,5,6,7), function(i){sum(hnr[[i]] & !hemi.masks[[i]])}))
)

#                1007  1012  1013  1014  1015  3367  1335
# homnr           316   247 12082   938   236 13863   167
# hemi          11761 16653 24337 11603 19824 49254 15367
# homnr.unhemi    316   246 11890   938   219 13474   167
```

```
# based on the thought that hnr in 1335 may reflect errors in the ref seq,
# are they shared with others?
unlist(lapply(hnr, function(x){sum(x & hnr[[7]])}))                 # hnr shared with 1335

# 1007 1012 1013 1014 1015 3367 1335
#   43   55   66   30   56   68  167

# answer: around 300 in each strain, of 558 in NY, genomewide,
# so that seems like a plausibly important factor.

hnr.lclade <- hnr[[1]] | hnr[[2]] | hnr[[4]] | hnr[[5]] | hnr[[7]]  # union over L-clade
sum(hnr.lclade)                                                     # count all in L-clade

# [1] 1496

sum(hnr[[3]] | hnr[[6]])                                            # present in H-clade

# [1] 18363

sum(hnr[[3]] & hnr[[6]])                                            # shared in H-clade

# [1] 7582
# look at a few in L-clade
w.hnr.l <- which(hnr.lclade)
seecounts(w.hnr.l[1:10],snp.tables=snp.tables)

#     chr  pos Ref Strain  A   G   C  T SNP  exon indel nrf rat
# 1  Chr1 1559   A
# 2                  1007  7   0   0 24   0  TRUE FALSE
# 3                  1012 11   0   0 37   0  TRUE FALSE
# 4                  1013  9   0   0  5   0  TRUE FALSE
# 5                  1014  4   0   0 16   0  TRUE FALSE
# 6                  1015 47   0   0 35   0  TRUE FALSE
# 7                  3367  0   0   0  0   0  TRUE FALSE
# 8                  1335 60   0   0 50   0  TRUE FALSE
# 9  Chr1 1575   G
# 10                 1007 24   7   0  0   0  TRUE FALSE
# 11                 1012 42  13   0  0   0  TRUE FALSE
# 12                 1013 17  16   0  0   0  TRUE FALSE
# 13                 1014 15   4   0  0   0  TRUE FALSE
# 14                 1015 43  31   0  0   1  TRUE FALSE
# 15                 3367  0   2   0  0   0  TRUE FALSE
# 16                 1335 34  74   0  0   0  TRUE FALSE
# 17 Chr1 1893   C
# 18                 1007  0   0  14 32   0  TRUE FALSE
# 19                 1012  0   0  38 52   0  TRUE FALSE
# 20                 1013  0   0  95 14   0  TRUE FALSE
# 21                 1014  0   0   5 31   0  TRUE FALSE
# 22                 1015  0   0  47 44   0  TRUE FALSE
# 23                 3367  0   0  29  0   0  TRUE FALSE
# 24                 1335  0   0  68 85   0  TRUE FALSE
# 25 Chr1 2223   A
# 26                 1007 25  13   0  0   0  TRUE FALSE
# 27                 1012 13  12   1  0   0  TRUE FALSE
# 28                 1013  5  24   0  0   0  TRUE FALSE
# 29                 1014  0   4   0  0   0  TRUE FALSE
# 30                 1015 19  22   0  0   1  TRUE FALSE
# 31                 3367 15   3   0  0   0  TRUE FALSE
# 32                 1335 33  22   0  0   0  TRUE FALSE
# 33 Chr1 2319   C
# 34                 1007  0  28  10  0   1  TRUE FALSE
# 35                 1012  0  43  17  0   1  TRUE FALSE
# 36                 1013 13  15   9  0   1  TRUE FALSE
# 37                 1014  0  18   6  0   1  TRUE FALSE
# 38                 1015  0  53  20  0   1  TRUE FALSE
# 39                 3367  4   0  24  0   0  TRUE FALSE
# 40                 1335  0 118  28  0   1  TRUE FALSE
# 41 Chr1 2502   A
# 42                 1007 14   2   0  0   0 FALSE FALSE
# 43                 1012 17   6   0  0   0 FALSE FALSE
# 44                 1013  6  13   0  0   0 FALSE FALSE
# 45                 1014  1   6   0  0   0 FALSE FALSE
# 46                 1015 20   7   0  0   0 FALSE FALSE
# 47                 3367  3   3   0  0   0 FALSE FALSE
# 48                 1335 29  17   0  0   0 FALSE FALSE
# 49 Chr1 2573   C
# 50                 1007  0   0  11 28   1  TRUE FALSE
# 51                 1012  0   0  30 50   1  TRUE FALSE
# 52                 1013  0   0 231 12   0  TRUE FALSE
# 53                 1014  0   0   4 18   1  TRUE FALSE
# 54                 1015  0   0  50 38   1  TRUE FALSE
# 55                 3367  0   0  71  0   0  TRUE FALSE
# 56                 1335  0   0  62 75   1  TRUE FALSE
# 57 Chr1 3938   G
# 58                 1007 12  20   0  0   0  TRUE FALSE
# 59                 1012  9  22   0  0   0  TRUE FALSE
# 60                 1013 35  19   0  0   0  TRUE FALSE
# 61                 1014  8   2   0  0   0  TRUE FALSE
# 62                 1015 25  53   0  0   0  TRUE FALSE
# 63                 3367 14  13   0  0   0  TRUE FALSE
# 64                 1335 59  42   0  0   0  TRUE FALSE
# 65 Chr1 4876   G
# 66                 1007  0   1   0  0   0 FALSE FALSE
# 67                 1012  1   4   0  0   0 FALSE FALSE
# 68                 1013  0   0   0  0   0 FALSE FALSE
# 69                 1014  1   0   0  0   0 FALSE FALSE
# 70                 1015  0   3   0  0   0 FALSE FALSE
# 71                 3367  4   4   0  0   0 FALSE FALSE
# 72                 1335  2   2   0  0   0 FALSE FALSE
# 73 Chr1 4938   T
# 74                 1007  0  43   0 23   1 FALSE FALSE
# 75                 1012  0  63   0 48   1 FALSE FALSE
# 76                 1013  0  83   0  2   0 FALSE FALSE
# 77                 1014  0  27   0  4   1 FALSE FALSE
# 78                 1015  0  75   0 47   1 FALSE FALSE
# 79                 3367  0  19   0 12   1 FALSE FALSE
# 80                 1335  0  57   0 59   1 FALSE FALSE

# one of those is a little weird:
xx<-snp.tables[[1]][149457,]
for (i in 2:7){xx <- rbind(xx,snp.tables[[i]][149457,])}
row.names(xx)<-names(snp.tables)
# My guess is that Chr/Pos/Ref are left as NA if coverage is zero.
xx

#      snp  Chr    Pos  Ref Cov a g c t n .match  exon indel  chr    pos rawCov
# 1007   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 1012   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 1013   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 1014   0 Chr1 149457    G   0 0 0 0 0 0      0 FALSE FALSE Chr1 149457      1
# 1015   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 3367   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 1335   0 Chr1 149457    G   0 0 0 0 0 0      0 FALSE FALSE Chr1 149457      1
```

5 Table 1 stats

Here is a brief summary of per-strain SNP counts, pairwise overlaps, and other conveniently available stats, such as those shown in Table 1 of the paper.

```
snp.counts    <- matrix(NA,7,4)
snp.pctofny   <- matrix(NA,7,4)
snp.pctofself <- matrix(NA,7,4)
snp.inter  <- matrix(NA,7,7)
snp.union  <- matrix(NA,7,7)
rownames(snp.counts)    <- names(snp.tables)
rownames(snp.pctofny)   <- names(snp.tables)
rownames(snp.pctofself) <- names(snp.tables)
rownames(snp.inter)  <- names(snp.tables)
colnames(snp.inter)  <- names(snp.tables)
rownames(snp.union)  <- names(snp.tables)
colnames(snp.union)  <- names(snp.tables)
for(stringency in 1:4){
  cat('\nStringency', stringency, ifelse(stringency==4,'(i.e. raw SAMTools SNP calls)',''),
      ':\n--------------\n')
  for(i in 1:7){
    f.snps.i <- refined.snps$Code$get.snps(i, stringency)
    snp.counts[i,stringency] <- sum(f.snps.i)
    for(j in i:7){
      f.snps.j <- refined.snps$Code$get.snps(j, stringency)
      snp.inter[i,j] <- sum(f.snps.i & f.snps.j)
      snp.union[i,j] <- sum(f.snps.i | f.snps.j)
    }
  }
  snp.pctofny  [,stringency] <- snp.inter[,7]/snp.counts[7,stringency]
  snp.pctofself[,stringency] <- snp.inter[,7]/snp.counts[ ,stringency]
  cat('Union Counts:\n');                  print(snp.union)
  cat('Intersect Counts:\n');              print(snp.inter)
  cat('Intersect as percent of union:\n'); print(snp.inter/snp.union*100,digits=3)
}

#
# Stringency 1  :
# --------------
# Union Counts:
#       1007  1012  1013  1014  1015  3367  1335
# 1007 18262 18723 36431 18614 18906 35671 18816
# 1012    NA 18475 36501 18769 19016 35729 18925
# 1013    NA    NA 29970 35615 36655 39685 36480
# 1014    NA    NA    NA 15827 18929 34748 18774
# 1015    NA    NA    NA    NA 18651 35878 19063
# 3367    NA    NA    NA    NA    NA 28699 35711
# 1335    NA    NA    NA    NA    NA    NA 18403
# Intersect Counts:
#       1007  1012  1013  1014  1015  3367  1335
# 1007 18262 18014 11801 15475 18007 11290 17849
# 1012    NA 18475 11944 15533 18110 11445 17953
# 1013    NA    NA 29970 10182 11966 18984 11893
# 1014    NA    NA    NA 15827 15549  9778 15456
# 1015    NA    NA    NA    NA 18651 11472 17991
# 3367    NA    NA    NA    NA    NA 28699 11391
# 1335    NA    NA    NA    NA    NA    NA 18403
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  96.2  32.4  83.1  95.2  31.7  94.9
# 1012   NA 100.0  32.7  82.8  95.2  32.0  94.9
# 1013   NA    NA 100.0  28.6  32.6  47.8  32.6
# 1014   NA    NA    NA 100.0  82.1  28.1  82.3
# 1015   NA    NA    NA    NA 100.0  32.0  94.4
# 3367   NA    NA    NA    NA    NA 100.0  31.9
# 1335   NA    NA    NA    NA    NA    NA 100.0
#
# Stringency 2  :
# --------------
# Union Counts:
#       1007  1012  1013  1014  1015  3367  1335
# 1007 17996 18521 37541 18222 18729 36692 18479
# 1012    NA 18326 37654 18474 18794 36815 18579
# 1013    NA    NA 30826 35649 37844 41411 37498
# 1014    NA    NA    NA 12861 18694 34625 18147
# 1015    NA    NA    NA    NA 18563 37002 18768
# 3367    NA    NA    NA    NA    NA 29507 36612
# 1335    NA    NA    NA    NA    NA    NA 17867
# Intersect Counts:
#       1007  1012  1013  1014  1015  3367  1335
# 1007 17996 17801 11281 12635 17830 10811 17384
# 1012    NA 18326 11498 12713 18095 11018 17614
# 1013    NA    NA 30826  8038 11545 18922 11195
# 1014    NA    NA    NA 12861 12730  7743 12581
# 1015    NA    NA    NA    NA 18563 11068 17662
# 3367    NA    NA    NA    NA    NA 29507 10762
# 1335    NA    NA    NA    NA    NA    NA 17867
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  96.1  30.0  69.3  95.2  29.5  94.1
# 1012   NA 100.0  30.5  68.8  96.3  29.9  94.8
# 1013   NA    NA 100.0  22.5  30.5  45.7  29.9
# 1014   NA    NA    NA 100.0  68.1  22.4  69.3
# 1015   NA    NA    NA    NA 100.0  29.9  94.1
# 3367   NA    NA    NA    NA    NA 100.0  29.4
# 1335   NA    NA    NA    NA    NA    NA 100.0
#
# Stringency 3  :
```

```
# --------------
# Union Counts:
#       1007  1012  1013  1014  1015  3367  1335
# 1007 16801 18054 36539 17040 18269 35673 17872
# 1012    NA 17738 36954 17864 18437 36089 18190
# 1013    NA    NA 30064 33057 37184 40928 36622
# 1014    NA    NA    NA  7895 18141 31952 17244
# 1015    NA    NA    NA    NA 18035 36335 18388
# 3367    NA    NA    NA    NA    NA 28785 35724
# 1335    NA    NA    NA    NA    NA    NA 17020
# Intersect Counts:
#       1007  1012  1013 1014  1015  3367  1335
# 1007 16801 16485 10326 7656 16567  9913 15949
# 1012    NA 17738 10848 7769 17336 10434 16568
# 1013    NA    NA 30064 4902 10915 17921 10462
# 1014    NA    NA    NA 7895  7789  4728  7671
# 1015    NA    NA    NA   NA 18035 10485 16667
# 3367    NA    NA    NA   NA    NA 28785 10081
# 1335    NA    NA    NA   NA    NA    NA 17020
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  91.3  28.3  44.9  90.7  27.8  89.2
# 1012   NA 100.0  29.4  43.5  94.0  28.9  91.1
# 1013   NA    NA 100.0  14.8  29.4  43.8  28.6
# 1014   NA    NA    NA 100.0  42.9  14.8  44.5
# 1015   NA    NA    NA    NA 100.0  28.9  90.6
# 3367   NA    NA    NA    NA    NA 100.0  28.2
# 1335   NA    NA    NA    NA    NA    NA 100.0
#
# Stringency 4 (i.e. raw SAMTools SNP calls) :
# --------------
# Union Counts:
#       1007  1012  1013  1014  1015  3367  1335
# 1007 16530 17707 35005 16864 17989 34289 17382
# 1012    NA 17019 35294 17276 18074 34563 17577
# 1013    NA    NA 25412 30445 35599 39448 34479
# 1014    NA    NA    NA  8331 17634 29704 16078
# 1015    NA    NA    NA    NA 17397 34876 17881
# 3367    NA    NA    NA    NA    NA 24613 33699
# 1335    NA    NA    NA    NA    NA    NA 15582
# Intersect Counts:
#       1007  1012  1013 1014  1015  3367  1335
# 1007 16530 15842  6937 7997 15938  6854 14730
# 1012    NA 17019  7137 8074 16342  7069 15024
# 1013    NA    NA 25412 3298  7210 10577  6515
# 1014    NA    NA    NA 8331  8094  3240  7835
# 1015    NA    NA    NA   NA 17397  7134 15098
# 3367    NA    NA    NA   NA    NA 24613  6496
# 1335    NA    NA    NA   NA    NA    NA 15582
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  89.5  19.8  47.4  88.6  20.0  84.7
# 1012   NA 100.0  20.2  46.7  90.4  20.5  85.5
# 1013   NA    NA 100.0  10.8  20.3  26.8  18.9
# 1014   NA    NA    NA 100.0  45.9  10.9  48.7
# 1015   NA    NA    NA    NA 100.0  20.5  84.4
# 3367   NA    NA    NA    NA    NA 100.0  19.3
# 1335   NA    NA    NA    NA    NA    NA 100.0

vs.stringency <- cbind(snp.counts, matrix(NA,7,1), round(snp.counts[,1:3]/snp.counts[,4]*100,1))
colnames(vs.stringency) <- c('[[1]]', '[[2]]', '[[3]]', '[[4]]', '----', '[[1]]%', '[[2]]%', '[[3]]%')

# SNPs vs filtering stringency (raw counts and as % of [[4]]).  Medium filter
# adds 10-20% in most cases.  Big exception is Gyre, where low coverage,
# high err rate and SAMTools conservatism seemed to seriously undercall:
print(vs.stringency)

#      [[1]] [[2]] [[3]] [[4]] ---- [[1]]% [[2]]% [[3]]%
# 1007 18262 17996 16801 16530   NA  110.5  108.9  101.6
# 1012 18475 18326 17738 17019   NA  108.6  107.7  104.2
# 1013 29970 30826 30064 25412   NA  117.9  121.3  118.3
# 1014 15827 12861  7895  8331   NA  190.0  154.4   94.8
# 1015 18651 18563 18035 17397   NA  107.2  106.7  103.7
# 3367 28699 29507 28785 24613   NA  116.6  119.9  117.0
# 1335 18403 17867 17020 15582   NA  118.1  114.7  109.2

# Intersect NY as % of self (vs stringency):
print(snp.pctofself*100, digits=3)

#       [,1]  [,2]  [,3]  [,4]
# 1007  97.7  96.6  94.9  89.1
# 1012  97.2  96.1  93.4  88.3
# 1013  39.7  36.3  34.8  25.6
# 1014  97.7  97.8  97.2  94.0
# 1015  96.5  95.1  92.4  86.8
# 3367  39.7  36.5  35.0  26.4
# 1335 100.0 100.0 100.0 100.0

# Intersect NY as % of NY (vs stringency):
print(snp.pctofny*100, digits=3)

#       [,1]  [,2]  [,3]  [,4]
# 1007  97.0  97.3  93.7  94.5
# 1012  97.6  98.6  97.3  96.4
# 1013  64.6  62.7  61.5  41.8
# 1014  84.0  70.4  45.1  50.3
# 1015  97.8  98.9  97.9  96.9
# 3367  61.9  60.2  59.2  41.7
# 1335 100.0 100.0 100.0 100.0
```

Quick look at coverage. Are there any NA?:

```
nacount <- NULL
for(i in 1:4){
  if(!is.null(tset[[i]])){
    nacount <- rbind(nacount,
                     unlist(lapply(tset[[i]], function(x){sum(is.na(x$Cov))}))
                     )
    rownames(nacount)[nrow(nacount)] <- names(tset)[i]
  }
}
nacount

#                             1007 1012 1013 1014 1015 3367 1335
# snp.tables.chr1.unfiltered    0    0    0    0    0    0    0
# snp.tables.chr1.qfiltered     0    0    0    0    0    0    0
```

Seemingly no. What's average in unq- vs q-filtered:

```
snp.tables.unqfil <- tset.picker(c(1,2), table.set = tset)
snp.tables.qfil   <- tset.picker(c(3,4), table.set = tset)
cov.unqfil <- unlist(lapply(snp.tables.unqfil, function(x){mean(x$Cov)}))
cov.qfil   <- unlist(lapply(snp.tables.qfil,   function(x){mean(x$Cov,na.rm=T)}))
cov.both <- rbind(cov.unqfil,cov.qfil,cov.qfil/cov.unqfil)
i <- 1
if(!is.null(snp.tables.unqfil)){
  rownames(cov.both)[i] <- which.snp.tables(snp.tables.unqfil)
  i <- i+1
}
if(!is.null(snp.tables.qfil)){
  rownames(cov.both)[i] <- which.snp.tables(snp.tables.qfil)
  i <- i+1
}
if(i==3){
  rownames(cov.both)[i] <- 'Ratio'
}
cat('Mean Coverage:\n'); cov.both

# Mean Coverage:
#                       1007       1012       1013       1014       1015       3367        1335
# Chr1-unfiltered 36.2816326 68.2005811 66.6908911 31.2663216 59.4704151 62.3834535 103.9124774
# Chr1-qfiltered  27.5849516 49.2557296 43.2293645 12.4319866 46.9874722 43.4403699  78.7789843
# Ratio            0.7603007  0.7222186  0.6482049  0.3976159  0.7900983  0.6963444   0.7581282
```

5.1 Table 1 Data

Throw together the conveniently-available Table 1 data, *in Table 1 row order*:

```
# if coverage unavailable, build NA vector
if(!is.null(cov.unqfil)){cov.unqfilv <- cov.unqfil} else {cov.unqfilv <- rep(NA,times=7)}
if(!is.null(cov.qfil  )){cov.qfilv   <- cov.qfil  } else {cov.qfilv   <- rep(NA,times=7)}
t1data.df <- data.frame(
  id        = st.locs(1:7, id=T, loc=F, date=F),
  loc       = st.locs(1:7, id=F, loc=T, date=F),
  date      = st.locs(1:7, id=F, loc=F, date=T),
  cov.unq   = cov.unqfilv,
  cov.q     = cov.qfilv,
  SNPs.4    = snp.counts[,4],
  SNPs.2    = snp.counts[,2],
  olap.ny.4 = snp.pctofny[,4]*100,
  olap.ny.2 = snp.pctofny[,2]*100
)
t1row.order <- c(7,1,2,5,3,6,4)
print(t1data.df[t1row.order,],digits=3)

#            id             loc date cov.unq cov.q SNPs.4 SNPs.2 olap.ny.4 olap.ny.2
# 1335 CCMP1335        New York 1958   103.9  78.8  15582  17867     100.0     100.0
# 1007 CCMP1007        Virginia 1964    36.3  27.6  16530  17996      94.5      97.3
# 1012 CCMP1012    W. Australia 1965    68.2  49.3  17019  18326      96.4      98.6
# 1015 CCMP1015     Puget Sound 1985    59.5  47.0  17397  18563      96.9      98.9
# 1013 CCMP1013           Wales 1973    66.7  43.2  25412  30826      41.8      62.7
# 3367 CCMP3367           Italy 2007    62.4  43.4  24613  29507      41.7      60.2
# 1014 CCMP1014 N. Pacific Gyre 1971    31.3  12.4   8331  12861      50.3      70.4
```

6 Shared-SNPs P-Value

Text of the main paper quotes a “p-value” for the observed degree of SNP sharing in L-clade (and/or L-clade excluding Gyre) under a null model that these isolates were sampled from a population globally in Hardy-Weinberg equilibrium. Details of this analysis are as follows.

6.1 SNP Concordance

Arbitrarily pick one isolate, say, A, as the “template”. Arbitrarily pick a heterozygous (aka “SNP”) position in A. Let p_1, and $q_1 = 1 - p_1$ be the frequencies in the overall population of the two nucleotides observed at that position in A. (Positions having 3 or 4 nucleotide variants segregating in the population are assumed to be negligibly rare.) Under the HWE null model, a second isolate B will also be heterozygous at the same position with probability $2p_1q_1 \leq 1/2$. Similarly, this position will be heterozygous in a third isolate C with the same probability, *independently*, and so on for isolates D and E. Overall, (assuming HWE) the probability that a heterozygous position in A is simultaneously heterozygous in the other 4 isolates is at most $1/2^4 = 1/16$. Continuing, suppose we pick a second heterozygous position in A, *on a different chromosome* with allele frequencies $p_2, q_2 = 1 - p_2$, say. Again assuming HWE, this position will be a SNP in all of B, C, D and E with probability $(2p_2q_2)^4 \leq 1/16$, and this is independent of the first position, since segregation on different chromosomes is unlinked. Repeat this at 24 heterozygous positions in A, one per chromosome. Then, the number of five-way concordant positions observed should be dominated by the number observed when sampling from a binomial distribution with parameters $n = 24$ and $p = 1/16$, i.e., we expect at most $1/16 = 6.25\%$ of positions to agree, or at most $24/16 = 1.5$ five-way concordant positions in total. In sharp contrast, choosing CCMP 1014 (North Pacific Gyre) as the template, we see many more five-way concordant positions than predicted under these assumptions:

```
gyre.count <- sum(snp.tables[[4]]$snp)
# NOTE: what we now calle "refined" SNPs were once called "filtered" SNPs and I have NOT tried
# to update variable names and annotation in the code below to reflect the terminology change...
# 'unfil.' => unfiltered for consistency; see below.
unfil.fiveway.count   <- sum( snp.tables[[4]]$snp * i4.snps)
unfil.fiveway.percent <- unfil.fiveway.count / gyre.count * 100
unfil.p.value <- pbinom(floor(unfil.fiveway.count/gyre.count*24)-1, 24, 1/16, lower.tail = FALSE)
consistency.comparison <-
  data.frame(
    fiveway.count   = unfil.fiveway.count,
    fiveway.percent = unfil.fiveway.percent,
    p.value         = unfil.p.value
  )
consistency.comparison

#   fiveway.count fiveway.percent      p.value
# 1          7628        91.56164 8.700771e-23
```

Namely, 8331 positions are called as SNPs in CCMP1014, of which 7628 or 91.5616373% are also called as SNPs in *all four* other L-clade isolates. 91.5616373% of 24 is 21.9747929, and the probability of seeing 21 or more “Heads” in 24 flips of a biased coin with $P(\text{Heads}) \leq 1/16$, i.e., our p-value under the HWE null hypothesis, is at most: 8.700771×10^{-23} based on this simple binomial model. This is obviously strong evidence against the null hypothesis.

This analysis is potentially overly-simplistic in four respects, addressed below.

1. “$2pq \leq 1/2$” is conservative. Neutral theory predicts that most variant nucleotides are rare in the population, so $2pq \ll 1/2$ is to be expected. This should make our quoted p-value very conservative.

2. Effect of Erroneous SNP calls. We base our analysis on *predicted* (by SAMTOOLS) heterozygous positions, not absolute-truth, which may affect our conclusions. However,

   - False negatives in A are irrelevant, since we never examine those positions. (This is the motivation for using CCMP1014 as the template; it has the lowest predicted SNP rate, likely due to a high false negative rate in that sequencing run. As noted elsewhere, it had the lowest coverage and lowest sequence quality of the 7 isolates, both of which impare SNP calling.)
   - False negatives in $BCDE$ make such positions appear *non*-concordant. For our purpose, this makes our statistic more conservative since it can only deflate a statistic that we argue is nevertheless unexpectedly large.
   - False positive calls in A are conservatively treated, as well: barring simultaneous false-positive calls in all of $BCDE$, such a position will appear non-concordant, again deflating the statistic. The *false* positive rates in B, C, D and E are unknown, but cannot exceed SAMTOOLS *total* positive rate, which is below 1% in all 7 isolates, suggesting a simultaneous $BCDE$ false positive rate $< 10^{-8}$, which will have a negligible effect.
   - A potentially more serious issue is a true positive in A aligned to false positives in BCD and/or E. (I.e., a position that is polymorphic in the population and heterozygous in A, under the HWE null model is likely to be homozygous for one of the two alleles in one or more of $BCDE$; false positive SNP calls in all of those isolates would make the site appear concordant, i.e., provide evidence against the null model.) However, (a) my impression is that SAMTOOLS is more prone to false negative calls than to false positive calls (see Section 4), and (b) we would need a high rate of false positives to turn a truely heterozygous but non-concordant A call into a false “concordant” call—I’d expect at most half (especially given point 1 above) of $BCDE$ to be heterozygous, but all would need to be falsely declared heterozygous. Such a high false positive rate on $BCDE$ seems unlikely (see previous bullet), and would likely be counterbalanced by a similarly increased rate of false positives on A, which, as noted, tend to deflate our statistic (previous bullet again).

- Systematic errors. If there were, say, a sequence-context-dependent bias in the DNA sequencing, mapping and/or SNP-calling that tended to suggest (or hide) a SNP at some position, we're going to systematically over- (or under-) estimate concordant SNPs across isolates. The discordance of called SNPs between the L- and H-clades and within the H-clade suggests that this is not a major problem, but it is worth noting as a possibility.

3. Discordant nucleotides at "concordant" SNP positions. A "shared" SNP at a given position might be, say, G/C in one isolate vs T/C in another, reflecting an unexpected tri-allelic position in the population or a technical sequencing error. It is inappropriate to count such a "shared" SNP position as evidence against the null hypothesis, since it isn't clear that it is truely shared. Instead, I will identify such inconsistent positions, based on the "stringency [[2]]" criteria established above, and treat each as non-concordant. I.e., a position will be considered to be a "5-way concordant SNP" if and only if it was called as a SNP by SAMTOOLS (independently) in all 5 L-clade isolates, *and* shows the same dominant non-reference nucleotide in all 5, according to criteria [[2]] above. As it turns out, this correction has a very minor effect on the resulting p-value:

```
# 'unfil.' => Ignoring "consistency";  'fil.' => Filtering for "consistency":
fil.fiveway.count   <- sum((snp.tables[[4]]$snp * i4.snps)[union.snps == 1] & consistent[[2]])
fil.fiveway.percent <- fil.fiveway.count / gyre.count * 100
fil.p.value <- pbinom(floor(fil.fiveway.count/gyre.count*24)-1, 24, 1/16, lower.tail = FALSE)
# append new stats to previous table for easy comparison
consistency.comparison <-
  rbind(consistency.comparison,
        data.frame(
          fiveway.count   = fil.fiveway.count,
          fiveway.percent = fil.fiveway.percent,
          p.value         = fil.p.value
        )
  )
rownames(consistency.comparison) <- c('unfiltered', 'consistency.filtered')
consistency.comparison

#                       fiveway.count fiveway.percent      p.value
# unfiltered                     7628        91.56164 8.700771e-23
# consistency.filtered           7537        90.46933 8.700771e-23
```

In particular, it removes 1.1% of five-way consistent positions (only 91 of 7628 positions), and still shows a highly significant p-value.

4. "$P(E[X]) \neq E[P(X)]$". I'm expressing this poorly, but finding the p-value based on the *expected* number of concordant positions is somewhat non-standard. A more typical set-up would use the *actual* value of some statistic, then calculate the probability of observing a value that extreme (or more extreme) under the null model. The fundamental problem is that we have thousands of SNPs, but I don't see an easy way to use more than 24 of them at a time, because potential genetic linkage seemingly destroys statistical independence, which is key to most simple analyses. A somewhat more formal, but still non-standard, approach is the following. Suppose we randomly sample one SNP per chromosome and count the number X of them that are 5-way concordant. What I outlined above calculated the p-value based on $E[X]$, the expected value of X, i.e., $P(E[X])$. Alternatively, we can calculate $E[P(X)]$, the expected p-value. (They are not the same.) In effect, this averages the p-values that would be seen over many different randomly-sampled sets of 24 SNPs. This is not difficult to calculate. First, the probability that we would observe $0 \leq i \leq 24$ concordant positions in a sample of 24, given that 90.47% of positions are concordant follows this binomial distribution:

```
x.equals.i.distribution <- dbinom(0:24, 24, fil.fiveway.percent/100)
print(x.equals.i.distribution, digits=3)

#  [1] 3.15e-25 7.19e-23 7.85e-21 5.46e-19 2.72e-17 1.03e-15 3.11e-14 7.58e-13 1.53e-11 2.58e-10
# [11] 3.68e-09 4.44e-08 4.57e-07 4.00e-06 2.98e-05 1.89e-04 1.01e-03 4.50e-03 1.66e-02 4.98e-02
# [21] 1.18e-01 2.14e-01 2.77e-01 2.28e-01 9.04e-02
```

Second, the p-value corresponding to $0 \leq i \leq 24$ observed concordant positions also follows a different binomial distribution:

```
p.val.of.x.equals.i <- c(1, pbinom(0:23, 24, 1/16, lower.tail = F))
print(p.val.of.x.equals.i, digits=3)

#  [1] 1.00e+00 7.88e-01 4.48e-01 1.87e-01 5.95e-02 1.49e-02 3.01e-03 4.99e-04 6.90e-05 8.02e-06
# [11] 7.89e-07 6.60e-08 4.72e-09 2.87e-10 1.49e-11 6.59e-13 2.46e-14 7.66e-16 1.98e-17 4.14e-19
# [21] 6.88e-21 8.70e-23 7.88e-25 4.56e-27 1.26e-29
```

Finally, the expected (or "average") p-value is just the weighted average of the latter values, weighted by the former:

```
e.of.p.of.x <- sum(x.equals.i.distribution * p.val.of.x.equals.i)
e.of.p.of.x

# [1] 1.33456e-14
```

This is still highly significant, but weaker than the $P(E[X])$ analysis, basically because $X < E[X]$ has a fair probability of occurring, and the corresponding p-value $P(X)$ rises rapidly as X declines.

Another way to look at the numbers:

```
pvdf <- data.frame(x.density=x.equals.i.distribution,
                   x.cdf=cumsum(x.equals.i.distribution),
                   pval.of.x=p.val.of.x.equals.i)
print(pvdf, digits=4)

#    x.density     x.cdf pval.of.x
# 1  3.155e-25 3.155e-25 1.000e+00
# 2  7.187e-23 7.219e-23 7.875e-01
# 3  7.846e-21 7.918e-21 4.476e-01
# 4  5.461e-19 5.541e-19 1.869e-01
# 5  2.722e-17 2.777e-17 5.950e-02
# 6  1.033e-15 1.061e-15 1.490e-02
# 7  3.106e-14 3.213e-14 3.010e-03
# 8  7.583e-13 7.904e-13 4.994e-04
# 9  1.530e-11 1.609e-11 6.899e-05
# 10 2.581e-10 2.742e-10 8.015e-06
# 11 3.675e-09 3.949e-09 7.887e-07
# 12 4.440e-08 4.835e-08 6.603e-08
# 13 4.566e-07 5.049e-07 4.716e-09
# 14 4.001e-06 4.506e-06 2.875e-10
# 15 2.984e-05 3.434e-05 1.493e-11
# 16 1.888e-04 2.232e-04 6.590e-13
# 17 1.008e-03 1.231e-03 2.456e-14
# 18 4.504e-03 5.735e-03 7.662e-16
# 19 1.663e-02 2.236e-02 1.977e-17
# 20 4.984e-02 7.220e-02 4.143e-19
# 21 1.183e-01 1.905e-01 6.877e-21
# 22 2.139e-01 4.043e-01 8.701e-23
# 23 2.768e-01 6.811e-01 7.884e-25
# 24 2.285e-01 9.096e-01 4.556e-27
# 25 9.037e-02 1.000e+00 1.262e-29
```

E.g., row 9 in that table says that the concordance rate (90%) is so high that a sample of 24 SNPs will almost always have 9 or more five-way concordant positions (probability of fewer is only 1.609e-11), while under the null model, seeing 9 or more is very unlikely (probability at most 6.899e-05). ***AM I OFF-BY-ONE INTERPRETING ROW 9 HERE??***

6.2 Notes

In earlier drafts, an analog of the above analysis was based on the concordance of *refined* SNPs. This now seems to me to be questionable, since the "refined" SNP calling makes SNPs called across L-clade non-independent. OTOH, the above analysis seems valid: SAMTOOLS was run on each isolate independently, and likewise “criterion [[2]]” is evaluated independently in each strain, and is being used here solely to remove SNP predictions, not to add them. “Systematic errors” as outlined above remain a potential problem, but again discordance with/within H-clade suggests that this is of limited concern.

For completeness, I did a similar analysis including a sample of H-clade comparisons: Gyre vs Italy, NY vs Italy, NY vs Italy+Wales, and of Italy vs Wales. As expected, none of these show a statistically significant p-value, although the $\approx 40\%$ concordance in the 2-way comparisons, while $< 1/2$ as predicted, is a bit higher than I expected based on “neutral theory implies many rare variants.” (I did not bother to include “criterion[[2]] filtering” in these calculations.)

```
# 'gi.twoway' => gyre vs italy 2-way concordance;
# 'ni.twoway' => new york vs italy 2-way concordance;
# not bothering with criterion[[2]] filtering
gi.twoway.count   <- sum(snp.tables[[4]]$snp * snp.tables[[6]]$snp)
gi.twoway.percent <- gi.twoway.count / gyre.count * 100
gi.p.value <- pbinom(floor(gi.twoway.count/gyre.count*24)-1, 24, 1/2, lower.tail = FALSE)
ny.count <- sum(snp.tables[[7]]$snp)
ni.twoway.count   <- sum(snp.tables[[7]]$snp * snp.tables[[6]]$snp)
ni.twoway.percent <- ni.twoway.count / ny.count * 100
ni.p.value <- pbinom(floor(ni.twoway.count/ny.count*24)-1, 24, 1/2, lower.tail = FALSE)
niw.threeway.count   <- sum(snp.tables[[7]]$snp * snp.tables[[6]]$snp * snp.tables[[3]]$snp)
niw.threeway.percent <- niw.threeway.count / ny.count * 100
niw.p.value <- pbinom(floor(niw.threeway.count/ny.count*24)-1, 24, 1/4, lower.tail = FALSE)
it.count <- sum(snp.tables[[6]]$snp)
iw.twoway.count   <- sum(snp.tables[[6]]$snp * snp.tables[[3]]$snp)
iw.twoway.percent <- iw.twoway.count / it.count * 100
iw.p.value <- pbinom(floor(iw.twoway.count/it.count*24)-1, 24, 1/2, lower.tail = FALSE)
consistency.comparison <-
  rbind(consistency.comparison,
        data.frame(
          fiveway.count   = c(gi.twoway.count,   ni.twoway.count,   niw.threeway.count,   iw.twoway.count),
          fiveway.percent = c(gi.twoway.percent, ni.twoway.percent, niw.threeway.percent, iw.twoway.percent),
          p.value         = c(gi.p.value,        ni.p.value,        niw.p.value,          iw.p.value)
        )
  )
colnames(consistency.comparison)[1:2] <- c('552232way.count', '552232way.percent') # old col names misleading
rownames(consistency.comparison)[3:6] <- c('gyre.vs.italy', 'new.york.vs.italy',    # new rows
                                           'ny.vs.it.plus.wales', 'it.vs.wales')
consistency.comparison

#                      552232way.count 552232way.percent      p.value
# unfiltered                      7628          91.56164 8.700771e-23
# consistency.filtered            7537          90.46933 8.700771e-23
# gyre.vs.italy                   3240          38.89089 9.242052e-01
# new.york.vs.italy               6496          41.68913 8.462719e-01
# ny.vs.it.plus.wales             3804          24.41278 7.533516e-01
# it.vs.wales                    10577          42.97323 8.462719e-01
```

6.3 P-Value: The Bottom Line

So, what to say in the body of the paper? $E[P(X)]$ is highly significant, and conservative, but complex to explain. $P(E[X])$ is simpler to explain, but may be criticized as misleading if we aren’t very careful in that explanation. I’m slightly leaning towards the last option, but want to sleep on it and draft the key sentence or two before settling.

7 Sharing

The following analysis looks at the sharing patterns among the consistent SNPs. I assume that shared SNPs reflect shared ancestry, and that SNPs accumulate slowly over time. Then, in outline, the story is consistent with what we have seen in other analyses—there seem to be 3 groups: 1013 (Wales) in one, 3367 (Italy) in another, and the other 5 in a third, with some hints as to the order of divergence. A caveat is that in a sexual population, non-shared SNPs do not immediately imply non-shared ancestry; they may merely reflect Hardy-Weinberg capturing a homozygous state in one isolate vs the other. (Or read errors, etc.) Thus, if we are right that the H-isolates retain sex, then the large number of “private” SNPs in H may be at least partially due to HWE.

Analysis is broken into cases based on how many strains share a particular SNP.

7.1 Code

To categorize SNPs by sharing patterns, first convert the 7-way consistent sharing pattern into a 7-bit binary number, and tabulate based on that:

```
# convert (n x 7) 0-1 matrix to n vector of 0-127
tobin <- function(x){
  bin <- integer(nrow(x)) # initialized to 0
  for(i in 1:7){
    bin <- bin*2 + as.integer(x[,i]>0)
  }
  return(bin)
}

# get full set of patterns
snp.pattern.all <- lapply(non.refs,tobin)
# prune to just the consistent ones
snp.pattern <- snp.pattern.all
for(i in 1:3){
  snp.pattern[[i]][!consistent[[i]]] <- NA
}

# analogous to built-in ``table'' but simpler.  Count entries in an integer
# vector sharing values in a (smallish) range.  Result is a 2-column matrix with
# the shared values in col 1 and count of occurrences of that value in col 2.
# Out-of-range values cause subscript error.
mytable <- function(vec, therange=range(vec,na.rm=T)){
  counts <- matrix(0,nrow=therange[2]-therange[1]+1,ncol=2,dimnames=list(NULL,c('val','count')))
  counts[1:nrow(counts),1] <- therange[1]:therange[2]
  for(i in 1:length(vec)){
    if(!is.na(vec[i])){
      counts[vec[i]-therange[1]+1,2] <- counts[vec[i]-therange[1]+1,2] + 1
    }
  }
  return(counts)
}

pattern.counts <- lapply(snp.pattern, function(x){mytable(x,c(0,127))})
```

To display the results, build a data frame whose i-th row, $0 \leq i \leq 127$ shows one of the 128 possible sharing patterns, with counts of the numbers of consistent, shared SNPs with that pattern according to criteria c1-c3.

```
tobitvec <- function(x){
  bitvec <- integer(7)
  for(i in 7:1){
    bitvec[i] <- x %% 2
    x <- x %/% 2
  }
  return(bitvec)
}

flg <- function(x){
  return(ifelse(x==1,'X',''))
}

pat.summary <- function(listOfTbls){
  mydf <- data.frame(pat=0:127,sharedBy=NA,
                     tp1007='',tp1012='',tp1013='',tp1014='',tp1015='',tp3367='',tp1335='',
                     count1=NA,count2=NA,count3=NA,count4=NA,stringsAsFactors=F)

  for(i in 1:128){
    bvec <- tobitvec(i-1)
    mydf[i,'sharedBy']=sum(bvec)
    mydf[i,'tp1007']=flg(bvec[1])
    mydf[i,'tp1012']=flg(bvec[2])
    mydf[i,'tp1013']=flg(bvec[3])
    mydf[i,'tp1014']=flg(bvec[4])
    mydf[i,'tp1015']=flg(bvec[5])
    mydf[i,'tp3367']=flg(bvec[6])
    mydf[i,'tp1335']=flg(bvec[7])
  }

  for(i in 1:length(listOfTbls)){
    tbl <- listOfTbls[[i]]
    if(!is.null(tbl)){
      mydf[,9+i] <- tbl[,2]  ## count1/2/3/4 are columns 10/11/12/13 in mydf
      #for(j in 1:length(tbl)){
      #  k <- as.integer(rownames(tbl)[j]);
      #  mydf[k+1,9+i] <- tbl[j]  ## count1/2/3 are columns 10/11/12
      #}
    }
  }

  mydf$pat <-as.octmode(mydf$pat)  # display bit pattern in octal
  return(mydf)
}

pat.summaries <- pat.summary(pattern.counts)
```

7.2 Sanity Checks

Some sanity checking: table sums equal to number of consistent positions?

```
all(consistent.count == apply(pat.summaries[,10:13],2,sum))

# [1] TRUE
```

More sanity checking: visually inspect a pattern with small counts, specifically pattern 12, i.e., consistent SNPs shared by only strains 1014 and 1015 (2nd and 3 rows from bottom, binary code $12 = 2^3 + 2^2$). There are only 10 such positions on Chr1. Chr1 2524239 has pattern 12 under criteria c1 and c2 but not c3; Chr1 1088766 has in c2 only. Both look good. Neither position is a *called* SNP except in 1015. However, all but 1 nonreference read agree with the called SNP (the exception being one read in Wales). Both 1014 and 1015 have at least 2 non-reference reads, comprising at least 5% of coverage, and in both strains, those reads are on the same non-reference base, satisfying criterion c2. The other strains have higher coverage and/or lower non-reference counts, so they do not satisfy c2. Position 2524239 also satisfies c1, but not c3, since 2 reads out of 35 is below the 10% threshold. (It is pattern 4 inder c3, i.e., a SNP private to 1015.) Position 1088766 is also pattern 4 under c3 (2 reads out of 56 in 1335 is below both thresholds), and it is not consistent under c1, since the single A read in 1013 is discordant with the other non-reference reads.

```
unlist(lapply(snp.pattern,function(x){sum(x==12,na.rm=T)}))

# [1]  4  1  6 12

sp1 <- snp.pattern[[1]]==12
sp2 <- snp.pattern[[2]]==12
sp3 <- snp.pattern[[3]]==12
sp4 <- snp.pattern[[4]]==12
c(sum(sp1,na.rm=T), sum(sp2,na.rm=T), sum(sp3,na.rm=T), sum(sp4,na.rm=T))

# [1]  4  1  6 12

r1 <- rownames(non.refs[[1]])[which(sp1)]
r2 <- rownames(non.refs[[2]])[which(sp2)]
r3 <- rownames(non.refs[[3]])[which(sp3)]
r4 <- rownames(non.refs[[4]])[which(sp4)]

r2
```

```
# [1] "Chr1:1799155"

c1 <- as.integer(unlist(lapply(strsplit(r1[1:min(20,length(r1))],':',fixed=TRUE),function(x){x[2]})))
c2 <- as.integer(unlist(lapply(strsplit(r2[1:min(20,length(r2))],':',fixed=TRUE),function(x){x[2]})))
c3 <- as.integer(unlist(lapply(strsplit(r3[1:min(20,length(r3))],':',fixed=TRUE),function(x){x[2]})))
c4 <- as.integer(unlist(lapply(strsplit(r4[1:min(20,length(r4))],':',fixed=TRUE),function(x){x[2]})))

c1

# [1]  614335  914018 1317406 2388286

c2

# [1] 1799155

c3

# [1]  371484  518347 1210354 2209068 2264683 2898352

c4

#  [1]  518347  691730  767408 1049906 1390437 2072951 2254059 2254789 2264683 2823796 2898352
# [12] 2998868

seecounts(c2,snp.tables=snp.tables)

#    chr     pos Ref Strain A G  C T SNP exon indel nrf rat
# 1 Chr1 1799155   C
# 2                    1007 0 0 10 1   0 TRUE FALSE
# 3                    1012 0 0 16 1   0 TRUE FALSE
# 4                    1013 0 0 10 0   0 TRUE FALSE
# 5                    1014 0 0  8 2   0 TRUE FALSE
# 6                    1015 0 0 12 3   1 TRUE FALSE
# 7                    3367 1 0  1 1   1 TRUE FALSE
# 8                    1335 0 0  7 1   0 TRUE FALSE
```

Position 1088766, however, in a good example of the situation that motivated this analysis—one strain has a G/C SNP and 5 of the other 6 strains have nonreference reads consistent with that SNP. Although, excluding 1015, the nonreference read counts are not high enough to justify a SNP call in any strain considered in isolation, the fact that they *consistently* agree with the 1015 SNP suggests that they are real. One alternative hypothesis is that there is some sequence-dependent bias at this locus that favors misreading a G as a C. On the other hand, one could equally well posit a shared SNP, and a locus-dependant bias that *supresses* C reads, explaining the unbalanced readout that we observe. However, it is hard to reconcile either view with the significant strain-specific patterns that we see in the shared SNPs (as seen below). I think a more likely explanation is that (a) there are some number of relatively rare SNPs present in each of the sampled populations, (b) some of these SNPs happened to be present in one or two cells of the roughly 5-10 cells that we believe constituted the founding population of the culture grown for sequencing, and (c) stochastic effects during culture growth and during sequencing may have further perturbed the apparent frequency of each variant, but the bottom line is that the above-threshold presence of consistent non-reference reads is evidence for shared SNPs at the population level (and the proportions of such reads represent estimates of the population-level frequencies of the variants, albeit a noisy estimate at any specific position).

An aside: I was curious to see whether there is any consistent pattern to positions that are called consistent SNPs in all but Italy, so I repeated the above, basically. My summary is that coverage in Italy tends to be below average in these positions, but otherwise they don't stand out. For the record:

```
abit <- snp.pattern[[2]]==125
abit[is.na(abit)]<-F
sum(abit)

# [1] 1352

rabit <- rownames(non.refs[[2]])[which(abit)]
rabits <- rabit[1:20]
cabit <- as.integer(unlist(lapply(strsplit(rabits,':',fixed=TRUE),function(x){x[2]})))
cabit
```

```
#  [1]  1244  1575  6485  7181  7220  7661  8144  8208  8518  8552  8567  8670  8685 14361 15254
# [16] 15280 16103 25546 30784 33852

seecounts(cabit,snp.tables=snp.tables)

#      chr   pos Ref Strain  A  G   C  T SNP  exon indel nrf rat
# 1   Chr1  1244   G
# 2                    1007  2 25   0  0   0  TRUE FALSE
# 3                    1012  3 32   0  0   0  TRUE FALSE
# 4                    1013 10 24   0  0   1  TRUE FALSE
# 5                    1014  3 17   0  0   0  TRUE FALSE
# 6                    1015 15 43   0  0   1  TRUE FALSE
# 7                    3367  0  1   0  0   0  TRUE FALSE
# 8                    1335 82 65   0  0   1  TRUE FALSE
# 9   Chr1  1575   G
# 10                   1007 24  7   0  0   0  TRUE FALSE
# 11                   1012 42 13   0  0   0  TRUE FALSE
# 12                   1013 17 16   0  0   0  TRUE FALSE
# 13                   1014 15  4   0  0   0  TRUE FALSE
# 14                   1015 43 31   0  0   1  TRUE FALSE
# 15                   3367  0  2   0  0   0  TRUE FALSE
# 16                   1335 34 74   0  0   0  TRUE FALSE
# 17  Chr1  6485   G
# 18                   1007 24 19   0  0   0  TRUE FALSE
# 19                   1012 29 29   0  0   0  TRUE FALSE
# 20                   1013 49 33   0  0   0  TRUE FALSE
# 21                   1014  6  5   0  0   0  TRUE FALSE
# 22                   1015 31 32   0  0   1  TRUE FALSE
# 23                   3367  0 37   0  0   0  TRUE FALSE
# 24                   1335 62 52   0  0   0  TRUE FALSE
# 25  Chr1  7181   G
# 26                   1007  0 30  29  0   0  TRUE FALSE
# 27                   1012  0 52  34  0   0  TRUE FALSE
# 28                   1013  0 19  72  0   0  TRUE FALSE
# 29                   1014  0 13   7  0   0  TRUE FALSE
# 30                   1015  0 40  33  0   1  TRUE FALSE
# 31                   3367  0 29   0  0   0  TRUE FALSE
# 32                   1335  0 78  73  0   0  TRUE FALSE
# 33  Chr1  7220   C
# 34                   1007 16  0  19  6   0  TRUE FALSE
# 35                   1012 38  0  22 11   0  TRUE FALSE
# 36                   1013 82  1  30  9   0  TRUE FALSE
# 37                   1014 12  0   6  2   0  TRUE FALSE
# 38                   1015 55  0  22  5   1  TRUE FALSE
# 39                   3367  0  0   8  0   0  TRUE FALSE
# 40                   1335 55  0  32 20   0  TRUE FALSE
# 41  Chr1  7661   T
# 42                   1007  0  0   9  9   0  TRUE FALSE
# 43                   1012  0  0   5 19   0  TRUE FALSE
# 44                   1013  0  0  24 14   1  TRUE FALSE
# 45                   1014  0  0   6  3   0  TRUE FALSE
# 46                   1015  0  0   5 34   0  TRUE FALSE
# 47                   3367  0  0   0  4   0  TRUE FALSE
# 48                   1335  0  0   4 24   0  TRUE FALSE
# 49  Chr1  8144   G
# 50                   1007  8  9   0  0   0  TRUE FALSE
# 51                   1012 12 10   0  0   1  TRUE FALSE
# 52                   1013 38 29   0  0   0  TRUE FALSE
# 53                   1014  5  4   0  0   0  TRUE FALSE
# 54                   1015 15 16   0  0   0  TRUE FALSE
# 55                   3367  0  0   0  0   0  TRUE FALSE
# 56                   1335 12 15   0  0   1  TRUE FALSE
# 57  Chr1  8208   G
# 58                   1007  0  6   0  7   1  TRUE FALSE
# 59                   1012  0 19   0 11   0  TRUE FALSE
# 60                   1013  0  1   0 48   0  TRUE FALSE
# 61                   1014  0  5   0  3   0  TRUE FALSE
# 62                   1015  0 19   0 11   1  TRUE FALSE
# 63                   3367  0  1   0  0   0  TRUE FALSE
# 64                   1335  0 28   0 16   1  TRUE FALSE
# 65  Chr1  8518   T
# 66                   1007  0  0  20 15   1 FALSE FALSE
# 67                   1012  0  0  40 20   1 FALSE FALSE
# 68                   1013  0  0  45 56   1 FALSE FALSE
# 69                   1014  0  0  10 16   0 FALSE FALSE
# 70                   1015  0  0  36 13   1 FALSE FALSE
# 71                   3367  0  0   0  2   0 FALSE FALSE
# 72                   1335  0  0 113 53   1 FALSE FALSE
# 73  Chr1  8552   G
# 74                   1007  3  9   0  0   0  TRUE FALSE
# 75                   1012 20 21   0  0   0  TRUE FALSE
# 76                   1013 28 16   0  0   1  TRUE FALSE
# 77                   1014  6  2   0  0   0  TRUE FALSE
# 78                   1015 14 13   0  0   0  TRUE FALSE
# 79                   3367  0 12   0  0   0  TRUE FALSE
# 80                   1335 24 47   0  0   0  TRUE FALSE
# 81  Chr1  8567   A
# 82                   1007 14 18   0  0   1  TRUE FALSE
# 83                   1012 26 30   0  0   1  TRUE FALSE
# 84                   1013 50 66   0  0   1  TRUE FALSE
# 85                   1014  1  3   0  0   0  TRUE FALSE
# 86                   1015 12 31   0  0   1  TRUE FALSE
# 87                   3367 22  0   0  0   0  TRUE FALSE
# 88                   1335 51 40   0  0   1  TRUE FALSE
# 89  Chr1  8670   A
# 90                   1007  7  0   0  5   0  TRUE FALSE
# 91                   1012 16  0   0 10   0  TRUE FALSE
# 92                   1013 16  0   0 11   0  TRUE FALSE
# 93                   1014  2  0   0  4   0  TRUE FALSE
# 94                   1015 14  0   0 10   1  TRUE FALSE
# 95                   3367  5  0   0  0   0  TRUE FALSE
# 96                   1335  7  0   0  6   0  TRUE FALSE
# 97  Chr1  8685   G
# 98                   1007  6 15   0  0   0  TRUE FALSE
# 99                   1012 10 23   0  0   0  TRUE FALSE
# 100                  1013 18 21   0  0   1  TRUE FALSE
# 101                  1014  4  8   0  0   0  TRUE FALSE
# 102                  1015 10 24   0  0   1  TRUE FALSE
# 103                  3367  0  4   0  0   0  TRUE FALSE
# 104                  1335  5 32   0  0   0  TRUE FALSE
# 105 Chr1 14361   A
# 106                  1007 20  7   0  0   0 FALSE FALSE
# 107                  1012 35  5   0  0   0 FALSE FALSE
# 108                  1013  1 11   0  0   1 FALSE FALSE
# 109                  1014  6  2   0  0   0 FALSE FALSE
# 110                  1015 35  7   0  0   0 FALSE FALSE
# 111                  3367  2  1   0  0   0 FALSE FALSE
# 112                  1335 50  8   0  0   0 FALSE FALSE
# 113 Chr1 15254   T
# 114                  1007 11  0   0 16   1 FALSE FALSE
# 115                  1012 26  0   0 38   1 FALSE FALSE
# 116                  1013 37  0   0 48   1 FALSE FALSE
# 117                  1014  3  0   0  8   1 FALSE FALSE
# 118                  1015 18  0   0 32   1 FALSE FALSE
# 119                  3367  0  0   0 73   0 FALSE FALSE
# 120                  1335 13  0   0 32   1 FALSE FALSE
# 121 Chr1 15280   T
# 122                  1007  0 13   0 20   1 FALSE FALSE
# 123                  1012  0 27   0 28   1 FALSE FALSE
# 124                  1013  0  5   0 64   0 FALSE FALSE
# 125                  1014  0  2   0  8   0 FALSE FALSE
# 126                  1015  0 19   0 29   1 FALSE FALSE
# 127                  3367  0  0   0 42   0 FALSE FALSE
# 128                  1335  0 21   0 70   1 FALSE FALSE
# 129 Chr1 16103   A
# 130                  1007 10  0  11  0   1 FALSE FALSE
# 131                  1012 44  0  19  0   1 FALSE FALSE
# 132                  1013 21  0  13  0   1 FALSE FALSE
# 133                  1014 14  0   2  0   0 FALSE FALSE
# 134                  1015 29  0  10  0   1 FALSE FALSE
# 135                  3367 33  0   0  0   0 FALSE FALSE
# 136                  1335 47  0  11  0   0 FALSE FALSE
# 137 Chr1 25546   A
# 138                  1007 23  0   0 14   1 FALSE FALSE
# 139                  1012 46  0   0 19   1 FALSE FALSE
# 140                  1013  6  0   0 42   1 FALSE FALSE
# 141                  1014  7  0   0 15   1 FALSE FALSE
# 142                  1015 52  0   0 17   1 FALSE FALSE
# 143                  3367 60  0   0  0   0 FALSE FALSE
# 144                  1335 67  0   0  5   0 FALSE FALSE
# 145 Chr1 30784   C
# 146                  1007 16  0  13  0   1  TRUE FALSE
# 147                  1012 33  0  32  0   1  TRUE FALSE
# 148                  1013 19  0  33  0   1  TRUE FALSE
# 149                  1014  4  0  11  0   1  TRUE FALSE
# 150                  1015 39  0  29  0   1  TRUE FALSE
# 151                  3367  0  0  55  0   0  TRUE FALSE
# 152                  1335 46  0  50  0   1  TRUE FALSE
# 153 Chr1 33852   C
# 154                  1007  0 24  25  0   1 FALSE FALSE
# 155                  1012  0 18  26  0   1 FALSE FALSE
# 156                  1013  0 28  33  0   1 FALSE FALSE
# 157                  1014  0  9   4  0   1 FALSE FALSE
# 158                  1015  0 19  28  0   1 FALSE FALSE
# 159                  3367  0  0  26  0   0 FALSE FALSE
# 160                  1335  0 30  53  0   1 FALSE FALSE
```

More sanity: there are 83 sites on Chr1 shared by zero strains in the tightest condition. (I.e., SAMTOOLS called it a SNP, but the read counts/proportions fall below our 3rd threshold). Are they due to low coverage? Seemingly yes:

```
zp3 <- snp.pattern[[3]] == 0
zr3 <- rownames(non.refs[[3]])[which(zp3)]
zc3 <- as.integer(unlist(lapply(strsplit(zr3[1:min(100,length(zr3))],':',fixed=TRUE),function(x){x[2]})))
zc3

#    [1]  16115  16615  19117  25748  43500  55857  56591  65787  66879  68328  80862  81001  90622
#   [14]  90721  91284 110754 116443 116453 120183 126702 127986 129056 147698 153874 159756 160912
#   [27] 161271 170686 180314 181477 182139 196862 196864 199166 206132 206143 221888 234931 242276
#   [40] 242914 244505 268954 274655 282391 282511 283646 289363 311952 312625 314132 326217 371008
#   [53] 376784 387078 387091 389263 395153 406158 410771 431788 438958 438976 443898 447253 448223
#   [66] 452774 488812 495476 498133 501830 501975 504462 506422 515441 515595 530113 530114 532320
#   [79] 534149 541667 543095 575081 585297 586276 612732 622585 651159 652889 655373 655380 657704
#   [92] 657955 658216 685697 687653 692115 692139 700484 700845 701061

seecounts(zc3[1:5], snp.tables=snp.tables)

#     chr   pos Ref Strain  A G  C  T SNP  exon indel nrf rat
# 1  Chr1 16115   T
# 2                     1007  0 0  0  5   0 FALSE FALSE
# 3                     1012  0 0  0  9   0 FALSE FALSE
# 4                     1013  0 0  0  6   0 FALSE FALSE
# 5                     1014  0 0  0  3   0 FALSE FALSE
# 6                     1015  0 0  0 10   0 FALSE FALSE
# 7                     3367  0 0  3  3   1 FALSE FALSE
# 8                     1335  0 0  0  6   0 FALSE FALSE
# 9  Chr1 16615   C
# 10                    1007  0 0 39  0   0 FALSE FALSE
# 11                    1012  0 0 54  0   0 FALSE FALSE
# 12                    1013  0 0  4  2   1 FALSE FALSE
# 13                    1014  0 0 19  0   0 FALSE FALSE
# 14                    1015  0 0 46  0   0 FALSE FALSE
# 15                    3367  0 0 13  0   0 FALSE FALSE
# 16                    1335  0 0 40  0   0 FALSE FALSE
# 17 Chr1 19117   A
# 18                    1007 16 0  0  0   0  TRUE FALSE
# 19                    1012 21 0  0  0   0  TRUE FALSE
# 20                    1013  1 0  0  1   0  TRUE FALSE
# 21                    1014  6 0  0  0   0  TRUE FALSE
# 22                    1015 21 0  0  0   0  TRUE FALSE
# 23                    3367  0 0  0  1   1  TRUE FALSE
# 24                    1335 24 0  0  0   0  TRUE FALSE
# 25 Chr1 25748   C
# 26                    1007  0 0 17  0   0 FALSE FALSE
# 27                    1012  0 0 36  0   0 FALSE FALSE
# 28                    1013  3 0  7  0   1 FALSE FALSE
# 29                    1014  1 0  4  0   0 FALSE FALSE
# 30                    1015  0 0 32  0   0 FALSE FALSE
# 31                    3367  0 0  1  0   0 FALSE FALSE
# 32                    1335  1 0 34  0   0 FALSE FALSE
# 33 Chr1 43500   A
# 34                    1007 10 0  0  3   1 FALSE FALSE
# 35                    1012 10 0  0  3   1 FALSE FALSE
# 36                    1013 10 0  1  1   0 FALSE FALSE
# 37                    1014  5 0  0  0   0 FALSE FALSE
# 38                    1015 11 0  0  2   0 FALSE FALSE
# 39                    3367  6 0  0  3   0 FALSE FALSE
# 40                    1335 13 0  0  1   0 FALSE FALSE
```

7.3 Main Analysis

Turning to the main analysis, there is a large increase in the number of consistent positions between the loose and medium stringency levels; medium and tight are similar in most respects. The likely interpretation is that the loose criterion is including many "SNPs" induced by read errors, and that either of the tighter criteria are successfully filtering them out. In the interest of simplicity, the narrative below will focus on the shared SNPs at the medium stringency level (the "count2" column in the data frame), although the numbers for all three (sometimes all 4) are displayed. Also note that the prose and some comments in the code were based on the Chr1 analysis, and so may occasionally be off-target for the whole-genome data.

```
# Show a subset of pat.summaries, optionally with totals of count_i in last row, and optionally
# aggregating low-count rows as ``Other''
#
#   sharedBy=c(2,4) selects SNPs shared by 2 or 4 strains,
#   subset=as.octmode('35') select those with sharing pattern a subset (optionally proper) of this
#   split=as.octmode('14') additionally restricts to patterns stradling split/subset minus split
#   c2.thresh=42 suppresses printout of rows with count2 < 42
#   restrict.to=c(0,42,127) restrict to these 3 rows
showgroup <- function(p.summ=pat.summaries, sharedBy=0:7, subset=127, split=NULL, proper.subset=F,
                      total=T, c2.thresh=0, fourteenth=F, restrict.to=NULL){
  # pick just those bit patterns that are subsets of 'subset'
  pick <- bitwAnd(0:127,bitwNot(subset))==0
  if(proper.subset){
    pick[subset+1] <- F
  }
  if(!is.null(split)){ # AND that stradle left/right subtrees
    cosplit <- bitwAnd(subset,bitwNot(split))
    pick <- pick & bitwAnd(0:127,split)!=0 & bitwAnd(0:127,cosplit)!=0
  }
  # and have desired shareBy counts
  pick <- pick & (p.summ$sharedBy %in% sharedBy)
  # and are among the set of interest
  if(!is.null(restrict.to)){
    pick <- pick & (0:127 %in% restrict.to)
  }
  # find rows with low counts
  pick.low <- pick & (p.summ$count2 < c2.thresh)
  # now show them
  show <- p.summ[pick & ! pick.low,]
  # rename columns just to narrow the printouts
  colnames(show) <- c('Pat','ShrBy','1007', '1012', '1013', '1014', '1015', '3367', '1335',
                      'count1', 'count2', 'count3','count4')
  show[,1] <- format(show[,1])  # convert octal col to char so can override in last row(2)
  nlow <- sum(pick.low)
  if(nlow > 0){
    n <- nrow(show)+1
    lows <- apply(p.summ[pick.low,10:13],2,sum)
    show[n,10:13] <- lows
    show[n,1:9] <- ''
    row.names(show)[n] <- 'Other'
    if(fourteenth){
      # do this: add 14th col just to hold this comment:
      show <- cbind(show,' '='', stringsAsFactors=F)
      show[n,14] <- paste('(', nlow, 'rows w/ c2 <', c2.thresh, ')')
    } else {
      ## or this (looks a bit funky, but fits across page without line-wrap):
      show[n,1:8] <-c('(', nlow, 'rows', 'w/', 'c2', '<', c2.thresh, ')')
    }
  }
  if(total){
    n <- nrow(show)+1
    tots <- apply(show[,10:13],2,sum)
    show[n,10:13] <- tots
    show[n,1:9] <- ''
    row.names(show)[n] <- 'Total'
    if(ncol(show)==14){show[n,14]<-''}
  }
  return(show)
}
```

First, are there any SNPs that are not "consistent SNPs?" Yes, a few in c3. As noted above, they seem to be mainly low-coverage positions.

```
showgroup(pat.summaries,0,total=F)   # chr1 totals: 0 0 83

#   Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 1    0     0                                         9     51    468      0
```

Next, look at completely shared SNPs, those found in all 7 strains.

```
showgroup(pat.summaries,7,total=F) # Chr1 count1 = 8593, count2 = 7054, count3 = 4790 c4=1641

#     Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 128 177     7    X    X    X    X    X    X    X 8132   6449     3873   1641
```

I.e., of the 47108 consistent positions, 6449 or 13.7% are shared by all 7 strains.

Next look at singletons, aka private SNPs—SNPs that are called in one strain and no other strain has a significant number of non-ref reads at that position. Presumably these are variants that arose in a given population after it separated from the others.

```
showgroup(pat.summaries,1)  # chr1 totals: 9669  18865  19670  23574

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 2     001     1                                  X     35     42     62    135
# 3     002     1                             X        7818   8912   9095  10949
# 5     004     1                        X              176    218    264    385
# 9     010     1                   X                    42     62     53    113
# 17    020     1              X                       8529   9752   9961  11677
# 33    040     1         X                              53     62    103    174
# 65    100     1    X                                   25     33     38    141
# Total                                               16678  19081  19576  23574
```

The import of shared/private SNPs changes between sexual and asexual populations. Presumably asexuals slowly gain and rarely lose private SNPs; shared ones predate separation of the lineages. In sexual lineages, however, SNPs may be rather freely "gained" or "lost," merely by recombination (converting between homo- and heterozygous in the sample we sequenced). Thus, the low private counts for the 5 L-isolates compared to the large count of het positions overall suggest that (a) they are asexual, and (b) none of them has been isolated from the others for very long (if at all). Conversely, the high counts for Italy and Wales suggest that (a) if asexual, they have been separated from each other and from the rest for a long time, but (b) if sexual, there is little surprise: we have ≈160K SNPs shared between the two (90K just in those two (below), plus 70K shared by all 7), and ≈90K additional positions that are het in one but not the other. These are close to, but not exactly equal to, the 1:2:1 ratios we would naively expect from two samples of a single HWE population. The most parsimonious explanation seems to be that the H-clade is sexual, but perhaps some het positions private to each population separates them.

Aside: counts of "consistent" SNPs minus these singletons yeilds count of shared SNPs:

```
singlets <- apply(pat.summaries[pat.summaries$sharedBy==1,10:13],2,sum)
rbind(consistent=consistent.count,singlets=singlets,shared=consistent.count-singlets)

#            count1 count2 count3 count4
# consistent  44905  47108  47204  47499
# singlets    16678  19081  19576  23574
# shared      28227  28027  27628  23925
```

The slightly higher count of shared positions in the medium case further supports this choice for subsequent analysis.

Next look at consistent SNPs shared between just a pair of isolates.

```
showgroup(pat.summaries,2) # chr 1 counts: 7641   9549   9472  6924

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 4     003     2                             X    X     87     15     37     31
# 6     005     2                        X         X     17     23     40     52
# 7     006     2                        X    X          65      9     20     41
# 10    011     2                   X              X      3      5      4     14
# 11    012     2                   X         X          41      4      2      5
# 13    014     2                   X    X                4      1      6     12
# 18    021     2              X                   X    102      7     24      9
# 19    022     2              X              X        8822   9349   8911   6177
# 21    024     2              X         X               62     15     21     46
# 25    030     2              X    X                    65      5      1      9
# 34    041     2         X                        X      2      4     23     36
# 35    042     2         X                   X          47      2     20     27
# 37    044     2         X              X                6     17     80    155
# 41    050     2         X         X                     5      1      3      7
# 49    060     2         X    X                         59      8     24     28
# 66    101     2    X                             X      3      5      7     20
# 67    102     2    X                        X          34      4     11     25
# 69    104     2    X                   X                2     10     40    107
# 73    110     2    X              X                     2      1      3      7
# 81    120     2    X         X                         49      5      7     31
# 97    140     2    X    X                              14     20     25     85
# Total                                                9491   9510   9309   6924
```

I.e., of the 9510 paired SNPs, 9349 or 98.3% are found between Italy and Wales, with comparatively few shared between any other pairs (only).

SNPs shared among exactly 3 isolates are relatively rare. (The 5 trios containing both Italy and Wales predominate in the loose set, probably because they share many pairs that become triples with the addition of a few read errors.)

```
showgroup(pat.summaries,3) # chr 1 counts: 1438    294    671  1034

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 8     007     3                        X    X    X      4      3     19     10
# 12    013     3                   X         X    X      3      2      2      2
# 14    015     3                   X    X         X      6      4      4      7
# 15    016     3                   X    X    X           1      0      1      2
# 20    023     3              X              X    X     89     23     35     17
# 22    025     3              X         X         X      8      4     23     21
# 23    026     3              X         X    X          84     31     37     32
# 26    031     3              X    X              X      3      1      0      0
# 27    032     3              X    X         X          66      9      6      5
# 29    034     3              X    X    X                2      5      1      1
# 36    043     3         X                   X    X      8     10     14      6
# 38    045     3         X              X         X     12     44    184    131
# 39    046     3         X              X    X           4     12     41     55
# 42    051     3         X         X              X      0      2      4      4
# 43    052     3         X         X         X           1      0      0      1
# 45    054     3         X         X    X                1      7     12     18
# 50    061     3         X    X                   X      1      8     11     12
# 51    062     3         X    X              X          86     21     29     36
# 53    064     3         X    X         X                2     17     52     60
# 57    070     3         X    X    X                     3      0      0      2
# 68    103     3    X                        X    X      2      3      6      8
# 70    105     3    X                   X         X      9     14     37     63
# 71    106     3    X                   X    X           4     10      9     27
# 74    111     3    X              X              X      0      1      1      1
# 75    112     3    X              X         X           0      1      0      0
# 77    114     3    X              X    X                2      4      4      8
# 82    121     3    X         X                   X      2      0      2      4
# 83    122     3    X         X              X          45      6     12     26
# 85    124     3    X         X         X                2      7     21     35
# 89    130     3    X         X    X                     1      1      1      1
# 98    141     3    X    X                        X      5      9     20     40
# 99    142     3    X    X                   X           3      3      9     15
# 101   144     3    X    X              X               18     74    159    355
# 105   150     3    X    X         X                     0      1      0      6
# 113   160     3    X    X    X                          6      3      7     23
# Total                                                 483    340    763   1034
```

Four-way sharing is more common, but dominated by the coastal (i.e., non-Gyre) L-clade isolates. This is likely a reflection of the strong 5-way sharing among the L-clade, from which the Gyre commonly drops out due to the lower coverage/higher error rate in that sequencing run.

```
showgroup(pat.summaries,4) # chr 1 counts: 564   1346   2552  3479

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 16    017     4                   X    X    X    X      1      2      1      2
# 24    027     4              X         X    X    X     15     16     37     24
# 28    033     4              X    X         X    X      4      2      4      6
# 30    035     4              X    X    X         X      2      4      0      0
# 31    036     4              X    X    X    X           5      0      3      3
# 40    047     4         X              X    X    X      9     26     68     60
# 44    053     4         X         X         X    X      0      1      1      2
# 46    055     4         X         X    X         X      8     15     24     36
# 47    056     4         X         X    X    X           2      2      2      5
# 52    063     4         X    X              X    X      9     12     34     21
# 54    065     4         X    X         X         X      8     21     68     48
# 55    066     4         X    X         X    X          15     43    102     76
# 58    071     4         X    X    X              X      0      2      0      0
# 59    072     4         X    X    X         X           4      2      1      2
# 61    074     4         X    X    X    X                1      2      1      7
# 72    107     4    X                   X    X    X      6     10      6     16
# 76    113     4    X              X         X    X      0      0      0      1
# 78    115     4    X              X    X         X      1      4      6      9
# 79    116     4    X              X    X    X           1      0      1      5
# 84    123     4    X         X              X    X      5      9     13      8
# 86    125     4    X         X         X         X      3      4     14     16
# 87    126     4    X         X         X    X          10     17     14     43
# 90    131     4    X         X    X              X      0      0      0      2
# 91    132     4    X         X    X         X           1      1      1      1
# 93    134     4    X         X    X    X                6      3      2      3
# 100   143     4    X    X                   X    X      1      3      4     20
# 102   145     4    X    X              X         X    598   1356   2429   2585
# 103   146     4    X    X              X    X           9     34     69    140
# 106   151     4    X    X         X              X      2      2      4     14
# 107   152     4    X    X         X         X           1      3      1      4
# 109   154     4    X    X         X    X               24     45     34    103
# 114   161     4    X    X    X                   X      3      8     10     18
# 115   162     4    X    X    X              X           8     11     20     33
# 117   164     4    X    X    X         X               19     51     71    163
# 121   170     4    X    X    X    X                     0      1      1      3
# Total                                                 781   1712   3046   3479
```

Five-way sharing is much more common, and is strongly dominated by the 5 L-clade isolates.

```
showgroup(pat.summaries,5) # chr 1 counts: 3969   5047   4624  6125

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 32    037     5              X    X    X    X    X     12     11      8      5
# 48    057     5         X         X    X    X    X      4      9      8     17
# 56    067     5         X    X         X    X    X     48    109    257    104
# 60    073     5         X    X    X         X    X      3      3      3      3
# 62    075     5         X    X    X    X         X      8      7     10     11
# 63    076     5         X    X    X    X    X           9     10     12      7
# 80    117     5    X              X    X    X    X      1      1      3      7
# 88    127     5    X         X         X    X    X     13     27     49     47
# 92    133     5    X         X    X         X    X      2      3      0      0
# 94    135     5    X         X    X    X         X      4      2      0      7
# 95    136     5    X         X    X    X    X           5      3      0      5
# 104   147     5    X    X              X    X    X    205    421    740   1160
# 108   153     5    X    X         X         X    X      4      0      0      7
# 110   155     5    X    X         X    X         X   4136   3560   2135   3228
# 111   156     5    X    X         X    X    X          11      7      9     43
# 116   163     5    X    X    X              X    X     15     14     21     33
# 118   165     5    X    X    X         X         X    318    591    957   1140
# 119   166     5    X    X    X         X    X          46    154    220    254
# 122   171     5    X    X    X    X              X      4      4      2      7
# 123   172     5    X    X    X    X         X           3      6      3      5
# 125   174     5    X    X    X    X    X                5     14     17     35
# Total                                                4856   4956   4454   6125
```

Six-way sharing is also common, with the sets *ex*cluding Gyre, Italy, or Wales having the most mutually-shared SNPs.

```
showgroup(pat.summaries,6) # chr 1 counts:  4166   4741   5312   4722

#        Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 64    077     6         X    X    X    X    X    X     43     46     62     26
# 96    137     6    X         X    X    X    X    X      8      6      8     14
# 112   157     6    X    X         X    X    X    X   1338   1076    665   1343
# 120   167     6    X    X    X         X    X    X   1305   2445   4098   1852
# 124   173     6    X    X    X    X         X    X     15      5      5      3
# 126   175     6    X    X    X    X    X         X   1709   1352    834   1416
# 127   176     6    X    X    X    X    X    X          57     79     43     68
# Total                                                4475   5009   5715   4722
```

8 Trees

So, overall, the picture looks like a long shared history (6449 7-way shared positions), followed by a split of the 5 L-isolates from the 2 H-isolates, then a long shared history in the 5 (3560 quintuples), in parallel with a long shared history in H- (9349 pairs), then separate histories in Italy and Wales (>8912 "private" SNPs in each, although again if they are sexual, many of these just reflect HWE), and very limited differentiation among the 5 L-isolates.

Branch lengths of course depend on filtering criteria used (and, of course, full vs Chr1 differ by about a factor of 10), but the tree *topology* appears to be fairly stable. Various versions are drawn below, exactly to explore how robust this story is. I think we should go with "medium stringency" SNP filtering (based on un-qfiltered reads).

NOTE: Much of this analysis make less sense for q-filtered read data, since (a) the point of the SNP filtering was to try to correct for noise in the raw reads, which may (or may not; haven't looked closely, yet) be largely fixed by qfiltering (e.g., "loose" or no SNP filtering may be more appropriate, post-q-filtering, esp. if we had re-run SAMTools to call SNPs based on the q-filtered reads), and (b) tree topology *does* appear to change, in that Gyre's coverage has been so sharply reduced by qfiltering that it clearly stands aside from the others (and that's confirmed by bootstrap), but this also seems to be clearly a technical rather than a biological artifact. SO, code below will run on q-filtered data, but *is not tuned to it*. Likewise, most comments in the prose below were made to describe the un-q-filtered data, and *are misleading and in some cases flatly wrong* for qfiltered data, but it doesn't seem worthwhile to bother with a rewrite...

Trees are coded in newick format, which doesn't seem to tolerate line-breaks; print with line-wrap:.

```
# wrap a long char string across multiple lines in printout
cat.hardwrap <- function(str,width=80){
  while(nchar(str)>width){
    cat(substr(str,1,width),'\n')
    str <- substr(str,width+1,nchar(str))
  }
  cat(str,'\n')
}
```

Trees are built as follows. Code for drawing, especially, is specific to the topology of the medium tree, and placement of some of the figure elements have been hand-optimized for this case; drawings for the other variants will not be as pretty.

```
# set up for tree figs

# the newick parser in ape seems to be confused by commas and parens in
# tip names, and blanks are not allowed, so replace by *, <, >, _, resp.
newick.name <- function(name){
  name <- gsub(' ', '_', name, fixed=TRUE)
  name <- gsub(',', '*', name, fixed=TRUE)
  name <- gsub('(', '<', name, fixed=TRUE)
  name <- gsub(')', '>', name, fixed=TRUE)
  return(name)
}
# undo above changes
newick.name.undo <- function(name){
 #name <- gsub('_', ' ', name, fixed=TRUE) # unnecessary; ape plot routine handles this one
  name <- gsub('*', ',', name, fixed=TRUE)
  name <- gsub('<', '(', name, fixed=TRUE)
  name <- gsub('>', ')', name, fixed=TRUE)
  return(name)
}

# make a newick string from tree;  see it below
# 'pre' is prefixed to ccmpid; 'nb' optionally included;
# 'alt' can be used instead of pre/ccmp/nb/where for less formal labeling
# 'newstyle'==T => new node label: [nb_]where[(pre-less-id)]
# 'newstyle'==F => old node label: [nb_][pre id]where
newickize <- function(tree,pre='CCMP',nb=TRUE,alt=F,newstyle=TRUE){
  if(is.null(tree$where)){
    # not a leaf; paste together newick from subtrees
    sub1 <- newickize(tree$sub1,pre=pre,nb=nb,alt=alt,newstyle=newstyle)
    sub2 <- newickize(tree$sub2,pre=pre,nb=nb,alt=alt,newstyle=newstyle)
    new <- paste( '(', sub1, ',', sub2, ')', sep='')
    if(!is.null(tree$length)){
      # internal node, add length
      return(paste(new, ':', tree$length, sep=''))
    } else {
      # top level; escape blanks and add trailing ';'
      return(paste(gsub(' ', '_', new), ';', sep=''))
    }
  } else {
    # a leaf; build label and branch length
    if(alt){
      # label is just alt; if alt omitted, default to where
      new <- newick.name(ifelse( is.null(tree$alt), tree$where, tree$alt ))
    } else {
      if(newstyle){
        # new node label = [nb_]where[(pre-less-id)]
        new <- ifelse( nb && !is.null(tree$nb), paste(tree$nb, '_', sep =''), '' )
        new <- newick.name(paste(new, tree$where, sep=''))
        new <- ifelse( is.null(tree$id), new, paste(new, '_(', tree$id, ')', sep='') )
        new <- newick.name(new)
      } else {
        # old style node label = [nb_][pre id]where
        new <- ifelse( nb && !is.null(tree$nb), paste(tree$nb, '_', sep =''), '' )
        new <- ifelse( is.null(tree$id), new, paste(new, pre, tree$id, '_', sep='') )
        new <- newick.name(paste(new, tree$where, sep=''))
      }
    }
    #add length to either
    new <- paste(new, ':', tree$length, sep='')
  }
  return(new)
}

# Make a tree as nested lists, **based on the chr1, count2 topology**, but using any of the counts.
#   Internal nodes have subtrees sub1/2 and length
#   Root has sub1/2, but no length
#   Leaves have where, length, optionally, id, alt, nb.  (Omit id for 'outgroup'. Use 'alt' for less formal
#     labeling in cartoon version; it defaults to 'where'. Use 'nb' to add abcde annotations for legend.)
# The single parameter v is any of the 4 count vectors contained in pat.summaries (most conveniently
# indexed in octal).  E.g., make.tree(pat.summaries[,'count2']) reproduces the count2 tree.
# (This was previously built by hand-pasting the edge lengths; tree.by.hand is retained in appendix
# for comparison, & its counts are in comments below).
#
make.tree <- function(v){
  pat.count <- function(pat, pat.counts=v){return(pat.counts[1+strtoi(pat,8)])}
  thetree <-
    list(
      sub1 = list(
        sub1 = list(
          sub1 = list(id=3367, length=pat.count('002'), where='Venice, Italy', alt='Venice'), #8813
          sub2 = list(id=1013, length=pat.count('020'), where='Wales, UK'),                    #9652
          length=pat.count('022')),                                                            #9365
        sub2 =  list(
          sub1 = list(
            sub1 = list(
              sub1 = list(id=1007, length=pat.count('100'), nb='e', where='Virginia, USA'),     #30
              sub2 = list(id=1012, length=pat.count('040'), nb='d', where='Perth, W. Australia', alt='Perth'), #61
              length=pat.count('140')),                                                         #19
            sub2 = list(
              sub1 = list(id=1015, length=pat.count('004'),nb='c', where='Washington, USA', alt='Puget Sound'), #207
              sub2 = list(id=1335, length=pat.count('001'), nb='b', where='New York, USA',   alt='NY'), #41
              length=pat.count('005')),                                                         #18
            length=pat.count('145')),                                                           #1005
          sub2 = list(id=1014, length=pat.count('010'), nb='a', where='N. Pacific Gyre'),       #61
          length=pat.count('155')),                                                             #3912
        length=pat.count('177')),                                                               #7054
      sub2 = list(length=0, where='outgroup')
    )
  return(thetree)
}
```

Code to plot a tree given newick description. Again, code is somewhat general, but has some specializations tied to the medium-stringency, full-genome, un-qfiltered data.

```
# run following 2 lines after an R upgrade
# update.packages()
# install.packages("ape")
library(ape)
show.tree <- function(newick.str=newick.medium,
                      col.edge  ='darkblue', lwd.edge =2,
                      col.elabel='darkblue',                cex.elabel=0.8, font.elabel=3,
                      col.arrow ='red',      lwd.arrow=1.5, cex.arrow =0.9, font.arrow =4,
                      col.clade ='black',    lwd.clade=1,   cex.clade =1.0, font.clade =3,
                      col.legbox='beige',                   cex.legend=0.8,
                      col.tip   ='darkblue',                                font.tip   =4,
                      plusx=FALSE, pltdebug=FALSE, total.snps=consistent.count[2],
                      straight.arrow=FALSE){

  ####
  #
  # ADJUST NEWICK & GET LENGTHS, COORDINATES
  #
  newick.str.noout <- sub('outgroup','_',newick.str) # Hide outgroup ('_' prints as blank)
  the.tree <- read.tree(text=newick.str.noout)

  ## nasty hack: ape's newick parser seems to be confused by commas, () in tip labels, so
  ## newickize replaced them by '*<>'; before plotting, I want to convert them back, and hope
  ## this doesn't break anything else...  And if a revised version of ape changes the internal
  ## representation of a tree, this may need to be redone.
  the.tree$tip.label <- newick.name.undo(the.tree$tip.label)

  # extract branch lengths as char string of comma-separated numbers via pattern matching hack:
  # lengths always preceded by colon
  lengths.ch <- strsplit(paste(newick.str,':'),'[^0-9][^:]*:')[[1]]

  # then convert to ints, dropping empty string at front
  lengths.int <- scan(what=integer(),quiet=T,sep=',',text=lengths.ch[-1])

  # then to data frame with named rows; a..g are terminal branches; others are internal.
  # a..e match legend in plot; f/g = wales/italy.  lengths appear in postfix order of
  # newick tree, and ape draws the 1st of them at the bottom of the plot.
  lmed <- data.frame(lengths=lengths.int,
                     row.names=c('g','f','fg','e','d','de','c','b','bc','bcde','a','abcde','all','out'))

  # extract counts needed for legend:
 #leg.counts <- c(      61, 41,207, 61, 30, 1005,   18, 19) #by hand, medium chr1
  leg.counts <- lmed[c('a','b','c','d','e','bcde','bc','de'),1]
  discord <- total.snps - sum(lmed$lengths)

  #tree.labels <- list( ## x,y,text; coords are all picked by eye
  #   3000, 3.62, paste(lmed['all'  ,1], 'shared by 7', sep='\n'), # 7054
  #   8900, 5.75, paste(lmed['abcde',1], 'by 5'       , sep='\n'), # 3912
  #  12000, 1.50, paste(lmed['fg'   ,1], 'shared by 2', sep='\n'), # 9365
  #  21000, 2.00, paste(lmed['f'    ,1], 'only\nin Wales'),        # 9652
  #  21000, 1.00, paste(lmed['g'    ,1], 'only\nin Italy'),        # 8813
  #  11500, 4.50, '*')
  # automating x-placement, below; retain above for comparison...
  tip <- integer(7)  # x coords of tree tips
  tip[1] <-sum(lmed[c('all','fg','g'),1])
```

```
tip[2] <-sum(lmed[c('all','fg','f'),1])
tip[3] <-sum(lmed[c('all','abcde','bcde','de','e'),1])
tip[4] <-sum(lmed[c('all','abcde','bcde','de','d'),1])
tip[5] <-sum(lmed[c('all','abcde','bcde','bc','c'),1])
tip[6] <-sum(lmed[c('all','abcde','bcde','bc','b'),1])
tip[7] <-sum(lmed[c('all','abcde','a'),1])

inode <- integer(5) # x coords of (some) internal nodes
inode[1] <- 0                                    # root
inode[2] <- lmed['all',1]                        # lca of all
inode[3] <- sum(lmed[c('all','fg'),1])           # lca H-clade
inode[4] <- sum(lmed[c('all','abcde'),1])        # lca L-clade
inode[5] <- sum(lmed[c('all','abcde','bcde'),1]) # lca L-clade, nonGyre
tree.labels <- list( ## x,y,text; y coords partially picked by eye
  sum(inode[c(1,2)])/2, 3.62, paste(lmed['all'  ,1], 'shared by 7', sep='\n'), # 7054
  sum(inode[c(2,4)])/2, 5.75, paste(lmed['abcde',1], 'by 5'       , sep='\n'), # 3912
  sum(inode[c(2,3)])/2, 1.50, paste(lmed['fg'   ,1], 'shared by 2', sep='\n'), # 9365
  (inode[3]+tip[2])/2,  2.00, paste(lmed['f'    ,1], 'only\nin 1013'),          # 9652
  (inode[3]+tip[1])/2,  1.00, paste(lmed['g'    ,1], 'only\nin 3367'),          # 8813
  sum(inode[c(4,5)])/2, 4.35, '* ')


tree.labels <- list( ## x,y,text; y coords partially picked by eye
  sum(inode[c(1,2)])/2, 3.62, paste(lmed['all'  ,1], 'in 7', sep='\n'), # 7054
  sum(inode[c(2,4)])/2, 5.75, paste(lmed['abcde',1], 'in 5', sep='\n'), # 3912
  sum(inode[c(2,3)])/2, 1.50, paste(lmed['fg'   ,1], 'in 2', sep='\n'), # 9365
  (inode[3]+tip[2])/2,  2.00, paste(lmed['f'    ,1], 'only\nin 1013'),   # 9652
  (inode[3]+tip[1])/2,  1.00, paste(lmed['g'    ,1], 'only\nin 3367'),   # 8813
  sum(inode[c(4,5)])/2, 4.35, '* ')

####
#
# BOGUS PLOT
#
# a messy bit: need string widths to set xlim; but strwidth needs x-scale so must plot first.
# M plot completely invisible, overlay 2nd plot via par(new=F...) .
#
# PROVISIONALLY set x.lim here at about 30% wider than tree; fine tune it for the real plot
# based on strwidth(tip labels) below.
#
provisional.tree.x.lim <- 1.3 * max(tip) # <== PROVISIONAL plot width
plot(0,0, type='n', bty='n', xaxt='n', yaxt='n', xlab='', ylab='', xlim=c(0,provisional.tree.x.lim), ylim=c(0,7))

tiplabel.x <- integer(7)
for(i in 1:7){
  # see warning above about internals of the.tree; labels have '_', printed as ' '.
  tiplabel.x[i] <- tip[i]+strwidth(gsub('_',' ',the.tree$tip.label[i],fixed=T), font=font.tip)
}


# visually show tip coords & max x to debug placement issues
plt.debug <- function(tree.x.lim, tip, tiplabel.x, spx=NULL, spy=NULL){
  if(pltdebug){ # F to hide/T to show debug
    cat('Tip labels:', paste(the.tree$tip.label,sep='',collapse='/'), '\n')
    axis(2) # useful only for placing labels
    for(i in 1:7){
      points(c(tip[i],tiplabel.x[i]),c(i,i)) # debug: do I have right tip coordinates?
    }
    lines(rep(tree.x.lim,2),c(0,7)) # where is right edge?
    if(!is.null(spx)){
      points(spx,spy) # show spline control points, for tweaking
    }
  }
}

plt.debug(provisional.tree.x.lim, tip, tiplabel.x)

label.end.H <- max(tiplabel.x[1:2])
label.end.L <- max(tiplabel.x[3:7])
clade.dx <- strwidth('x') # space between clade marker line and its label
xdel <- 3*clade.dx        # space between labeled clade tips and marker line

tree.x.lim <- 1.03*(max(tiplabel.x)+xdel)  # <== FINAL plot width
tree.y.lim <- 7
if(pltdebug){cat('Plot width hacking:', provisional.tree.x.lim, tree.x.lim, tree.x.lim/1.03/max(tip), clade.dx)}

par(new=T)  # I.e., NOT starting a new plot

####
#
```

```
# REAL PLOT
#
plot(the.tree,
     x.lim = c(0, tree.x.lim),
     y.lim = c(0, tree.y.lim),
     font=font.tip, label.offset=100,             # bold-italic, nudged slightly right
     tip.color=col.tip, edge.color=col.edge,
     edge.width=lwd.edge,
     edge.lty=c(1,1,1,1, 1 ,1,1,1,1,1,1,1,1,0)    # 5th is bottleneck edge; 14th is outgroup
    )
lines(00+c(0,0),c(3.5,6),col='white',lwd=6)       # Hide vertical line to outgroup
axis(1, pos=0.25, at=seq(0,25,by=5)*10^round(log10(max(tip)/25)))

if(pltdebug){text(tip[1]+100, 1.0, 'Venice, Italy (3367)', adj=0, font=font.tip)}

####
#
# BOTTLENECK ANNOTATION
#
# spline/elipse control points (spy/y) & tweaks thereto (dx/y)
dx <- 0.01 * tree.x.lim
dy <- .04
spx <- c(7400, 7400, 9900, 10500) # by eye, chr1, for comparison
spx <- c(inode[2]+dx,inode[2]+dx,inode[4]-3*dx,inode[4]-dx)
spy <- c( 3.8,  3.9,  5.6-dy,   5.6-dy)

plt.debug(tree.x.lim, tip, tiplabel.x, spx, spy)

if(T){
  #elipse version, defined by rect thru 2 middle pts of spx/y
  spf<-function(x){
    ifelse(x <= spx[2], spy[1],
           ifelse(x >= spx[3], spy[4],
                  spy[2]+(spy[3]-spy[2])*sqrt(pmax(0,1-((x-spx[3])/(spx[3]-spx[2]))^2))))
    }
} else {
  # spline version
  spf <- splinefun(spx,spy,method='hyman')
}
serx <- seq(spx[1],spx[length(spx)],length.out=50)
sery <- spf(serx)
tailx <- spx[1]
taily <- spy[1]
headx <- spx[4]
heady <- spy[4]
textx <- (headx+tailx)/2+(headx-tailx)*(-.01)
texty <- (heady+taily)/2+(heady-taily)*(-.10)
bottle.txt <- "inbreeding\nLoH / LoS"
if(!straight.arrow){
  arrows(headx,heady,headx+tree.x.lim*1e-3,heady, length=.1,col=col.arrow,lwd=lwd.arrow)
  lines(rev(serx), rev(sery), lty=c(5,1),col=col.arrow, lwd=lwd.arrow)
  textangle <- 66
  textadj <- c(0,0)
} else {
  # Tweak positioning slightly; visualize a rectangle from 7-node to base of L-clade;
  # center text, rotated, on diagonal towards L-clade; ditto the straight arrow.
  llx <- inode[2] # the aforementioned rectangle
  urx <- inode[4]
  lly <- 3.62
  ury <- 5.75
  # rect(llx,lly,urx,ury) # show rect for debug
  textx <- (llx+urx)/2    # center text
  texty <- (lly+ury)/2
  textangle <- atan(grconvertY(ury-lly,to='dev')/grconvertX(urx-llx,to='dev'))*360/(2*pi)
  textadj <- c(0.50, 0.43) #tweak position; ".5" = center in x , ".43" raises, THEN rotate.
  alpha <- .78 # fraction along diag at which arrow begins
  beta  <- .95 # ... and ends
  arrows((1-alpha)*llx + alpha*urx,
         (1-alpha)*lly + alpha*ury,
         (1-beta)*llx  + beta*urx,
         (1-beta)*lly  + beta*ury, length=.1,col=col.arrow,lwd=lwd.arrow,angle=25)
}
if(T){
  text(textx, texty, bottle.txt, srt=textangle, font=font.arrow, cex=cex.arrow,
       col=col.arrow, adj=textadj)
} else {
  # experiment at wrapping text along curved path; unpretty, but retain for now, maybe revisit
  bottlec <- strsplit(bottle,split=NULL)[[1]]
  for(i in 1:length(bottlec)){
    text(xser[i],yser[i],bottlec[i], srt=65, font=4, cex=.7, col=col.arrow)
    }
  }

  ####
  #
  # CLADE ANNOTATION
  #
  clade.L.x <- label.end.L + xdel
  clade.H.x <- label.end.H + xdel
  dy <-.33
  lines(rep(clade.L.x,2),c(3-dy,7+dy),lwd=lwd.clade,col=col.clade)
  lines(rep(clade.H.x,2),c(1-dy,2+dy),lwd=lwd.clade,col=col.clade)
  text(clade.L.x+clade.dx,5.0,'L-clade',srt=90,font=font.clade,cex=cex.clade,col=col.clade)
  text(clade.H.x+clade.dx,1.5,'H-clade',srt=90,font=font.clade,cex=cex.clade,col=col.clade)

  ####
  #
  # LEGEND
  #
  # parameter plusx controls whether we try to annotate b/c (+) and d/e (x) sharing in tree; I think
  # it looks cluttered, rather than adding clarity, so I vote no, but code is here, in case.  "Logic,"
  # if any, for my symbol choice is that + overlaid on x looks like the * at the next level; this
  # analogy is more visible if we use pch 3/4/8 rather than Courier or Helvetica chars, but probably
  # should use same in both tree & legend, which will take a modicum of additional work.
  legend.text <- c('a: only in 1014  ',
                   'b: only in 1335  ',
                   'c: only in 1015  ',
                   'd: only in 1012  ',
                   'e: only in 1007  ',
                   '*: shared by bcde',
                   paste(ifelse(plusx,'+:','  '),'shared by b/c '),
                   paste(ifelse(plusx,'x:','  '),'shared by d/e ')
  )

  legend.text <- c('a: only in 1014 ',
                   'b: only in 1335 ',
                   'c: only in 1015 ',
                   'd: only in 1012 ',
                   'e: only in 1007 ',
                   '*: in bcde      ',
                   paste(ifelse(plusx,'+:','  '),'in bc        '),
                   paste(ifelse(plusx,'x:','  '),'in de        '),
                   'Discordant SNPs '
  )
  legend.text <- paste(legend.text,format(c(leg.counts,discord),width=4),sep=' - ')
  legend.text <- paste(legend.text,' ') # add a little more right margin in box
  opar <- par(family='mono',cex=cex.legend)
  legend('topright', legend=legend.text, cex=cex.legend, inset=c(0.05,0), bg=col.legbox, box.col=col.legbox)
  par(opar)
  if(plusx){
    points(tree.labels[[16]],tree.labels[[17]]+.14,pch=8,col=col.elabel)
    points(tree.labels[[16]]+200,tree.labels[[17]]+1,pch=3,col=col.elabel)
    points(tree.labels[[16]]+200,tree.labels[[17]]-1,pch=4,col=col.elabel)
  }

  ####
  #
  # EDGE LENGTHS
  #
  for(i in seq(1,length(tree.labels)-ifelse(plusx,5,2),by=3)){
    if(F){ # T for \n in edge labels; F to remove (except "by 5")
      text(tree.labels[[i]], tree.labels[[i+1]], tree.labels[[i+2]])
    } else {
      # points(tree.labels[[i]], tree.labels[[i+1]], pch=3,col='green') # for debugging
      text(tree.labels[[i]], tree.labels[[i+1]], sub('\n([^z])',' \\1', tree.labels[[i+2]]),
           pos=3, offset=.4, font=font.elabel, col=col.elabel,cex=cex.elabel)
    }
  }
}
if(FALSE){#for debug convenience
  pdf(paperfig.path, width=8,height=5,onefile=TRUE,family='Helvetica',fonts='Courier',pointsize=10)
  show.tree(newick.medium, total.snps=consistent.count[2], pltdebug=F,straight.arrow=T)
  dev.off()
}
```

```
caption <- function(stringency,which.tables=which.snp.tables(string.val=F)){
  caption.where <- '(UNKNOWN genome subset).'
  if(which.tables[1]=='Chr1') {caption.where <- 'on Chr1.'}
  if(which.tables[1]=='full') {caption.where <- 'genome-wide.'}
  if(which.tables[1]=='trunc'){caption.where <- 'all Chrs.'}
  cap.stringency <- c(
    'loose SNP filters.',
    'medium SNP filters.',
    'strict SNP filters.',
    'unfiltered SNPs.')
  cap <- paste('Tree based on', which.tables[2], 'reads and', cap.stringency[stringency],
               ' ``Lengths\'\' are numbers of shared/private SNPs', caption.where)
  return(cap)
}
```

Trees based on all four SNP filtering criteria are shown below. Their topologies are exactly the same, although the branch lengths are different. In all four, the length of the branch labeled "*" is probably inflated by lower coverage and higher error rate in 1014, which may mask further legitimate sharing between it and the other L-isolates. The branch lengths among the other 4 are too short for their topology to be convincing without a more rigorous analysis (e.g., a bootstrap test), but detail there is irrelevant to the story.

My sense is that the "medium" version is the best for the paper, made here and shown in Fig 1. In theory, this should look exactly like Fig 3, but something is apparently different between Knitr and direct-to-pdf. (Increasing fig.width in Knitr's chunk headers from 8 (as in the pdf call below) to 9 helps somewhat, but probably still best to make the paper fig directly rather than via Knitr.)

```
###
#
# MAKE PROTOTYPE PDF FOR PAPER, *AND* SAVE DATA NEEDED TO BUILD IT
#
w.s.t. <- which.snp.tables()
if(w.s.t. == 'trunc-unfiltered'){
  rda.Description <- 'This .rda contains data to generate Fig 3; see shared.snps.rnw for details.'
  save(rda.Description, w.s.t., pat.summaries, consistent.count, file='Fig3-data.rda')
  paperfig.path <- paste('figs-mine/paperfig-medium-tree-', w.s.t., '--Fig3proto.pdf', sep='')
} else {
  paperfig.path <- paste('figs-mine/paperfig-medium-tree-', w.s.t., '.pdf', sep='')
}
pdf(paperfig.path, width=8,height=5,onefile=TRUE,family='Helvetica',fonts='Courier',pointsize=10)
newick.medium <- newickize(make.tree(pat.summaries[,'count2']))
show.tree(newick.medium, total.snps=consistent.count[2], pltdebug=F,straight.arrow=T)
dev.off()

# pdf
#   2
```

```
# fig.paths for knitr chunks below;  .h for "hand-made" trees; plain for automatic chr1/full versions
myfigpath   <- paste(getwd(), '/figs-knitr/newick-', which.snp.tables(), '-', sep='')
myfigpath.h <- paste(getwd(), '/figs-knitr/newick-', sep='')
```

Figure 2, i.e., criteria [[1]]:

```
newick.loose <- newickize(make.tree(pat.summaries[,'count1']))
show.tree(newick.loose, total.snps=consistent.count[1])
```

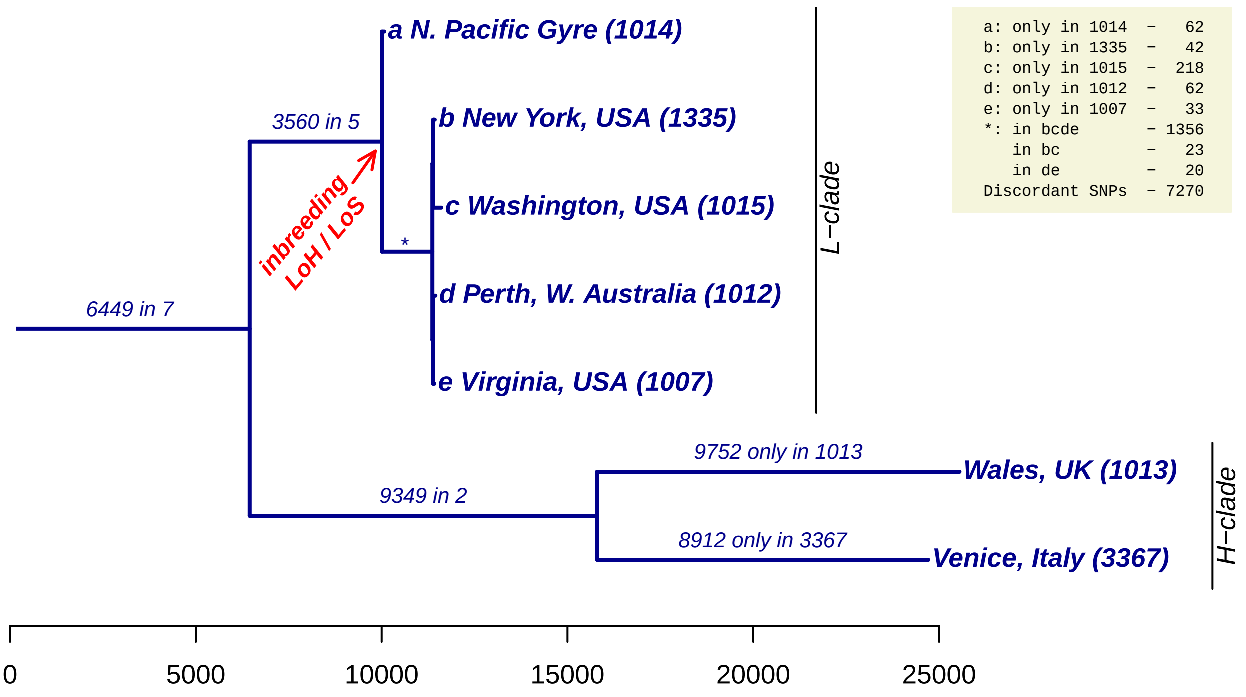
Figure 3, i.e. [[2]]:

```
# newick.medium <- newickize(tree.by.hand)
# simple.newick.medium <- newickize(tree.by.hand,alt=TRUE)
newick.medium <- newickize(make.tree(pat.summaries[,'count2']))
simple.newick.medium <- newickize(make.tree(pat.summaries[,'count2']),alt=TRUE)
show.tree(newick.medium, total.snps=consistent.count[2])
```

Figure 4, i.e. [[3]]:

```
newick.strict <- newickize(make.tree(pat.summaries[,'count3']))
show.tree(newick.strict, total.snps=consistent.count[3])
```

Figure 5, i.e. [[4]]:



```
a: only in 1014  –   62
b: only in 1335  –   42
c: only in 1015  –  218
d: only in 1012  –   62
e: only in 1007  –   33
*: in bcde       – 1356
   in bc         –   23
   in de         –   20
Discordant SNPs  – 7270
```

Figure 1: Proposed fig. for paper: Tree based on qfiltered reads and medium SNP filters. “Lengths” are numbers of shared/private SNPs on Chr1.

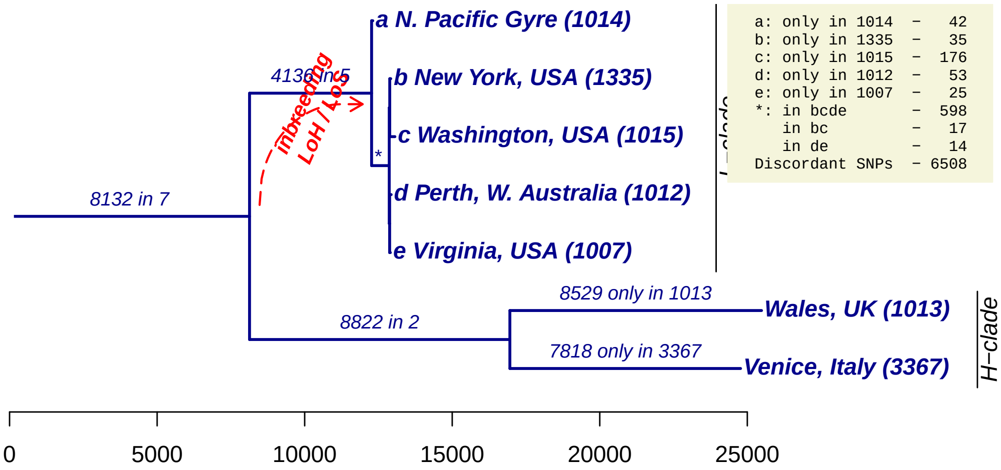


Figure 2: Tree based on qfiltered reads and loose SNP filters. “Lengths” are numbers of shared/private SNPs on Chr1.

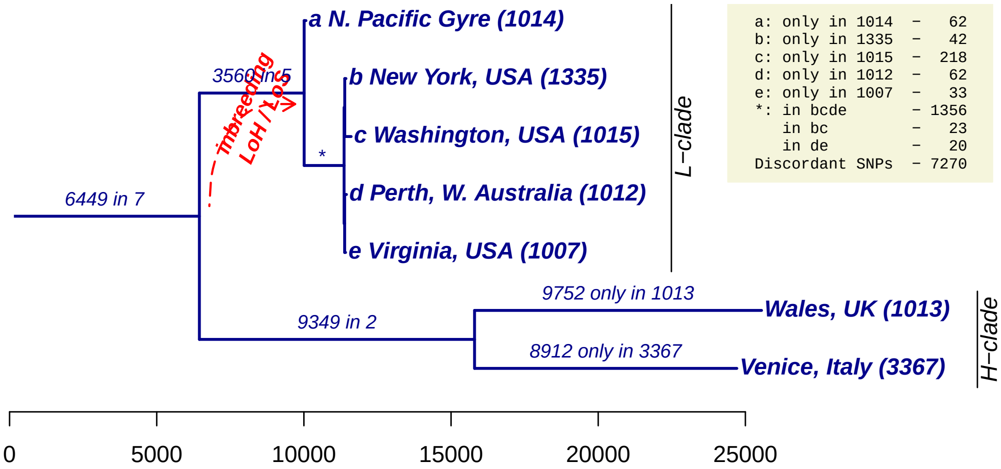


Figure 3: Tree based on qfiltered reads and medium SNP filters. “Lengths” are numbers of shared/private SNPs on Chr1.

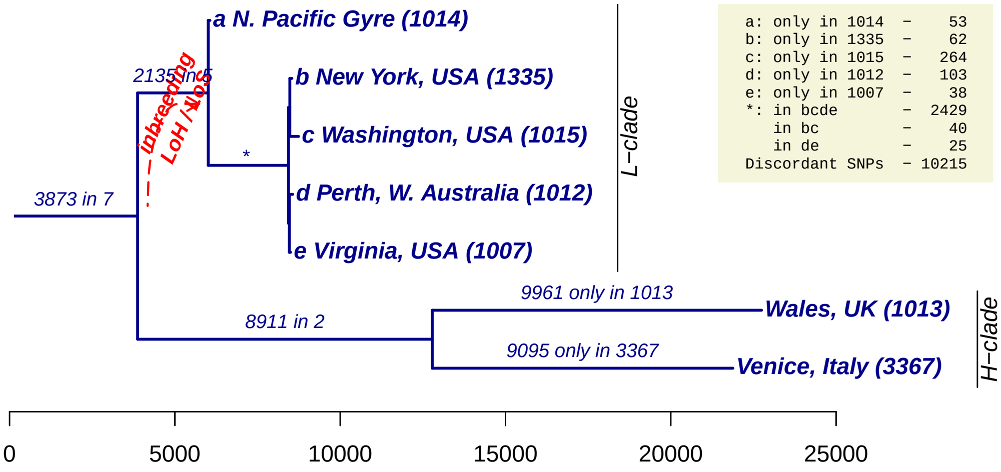


Figure 4: Tree based on qfiltered reads and strict SNP filters. “Lengths” are numbers of shared/private SNPs on Chr1.

```
newick.unfiltered <- newickize(make.tree(pat.summaries[,'count4']))
show.tree(newick.unfiltered, total.snps=consistent.count[4])
```

Some other versions of the trees are included in the appendix.

Counts for all tree edges in the medium tree:

```
#pat.summaries[c(128,110,102,6,97,19,9,2,5,33,65,17,3),]
tree.edges <- c(128,110,102,6,97,19,9,2,5,33,65,17,3)-1
non.edges <- setdiff(0:127, tree.edges)
sg.edges <- showgroup(restrict.to=tree.edges) ; sg.edges

#       Pat  ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 2     001     1                                   X     35     42     62    135
# 3     002     1                              X        7818   8912   9095  10949
# 5     004     1                         X              176    218    264    385
# 6     005     2                         X         X     17     23     40     52
# 9     010     1                    X                    42     62     53    113
# 17    020     1               X                       8529   9752   9961  11677
# 19    022     2               X              X        8822   9349   8911   6177
# 33    040     1          X                              53     62    103    174
# 65    100     1     X                                   25     33     38    141
# 97    140     2     X    X                              14     20     25     85
# 102   145     4     X    X              X         X    598   1356   2429   2585
# 110   155     5     X    X         X    X         X   4136   3560   2135   3228
# 128   177     7     X    X    X    X    X    X    X   8132   6449   3873   1641
# Total                                                38397  39838  36989  37342
```

Counts for the top 10 discordant patterns, i.e., SNPs whose sharing pattern does not match any of the bifurcations in the tree:

```
tenth <- sort(showgroup(restrict.to=non.edges)[-(length(non.edges)+1),'count2'],decreasing=T)[10]
sg.non.edges <- showgroup(restrict.to=non.edges, c2.thresh = tenth) ; sg.non.edges

#       Pat  ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 1     000     0                                          9     51    468      0
# 56    067     5          X    X         X    X    X     48    109    257    104
# 101   144     3     X    X              X               18     74    159    355
# 104   147     5     X    X              X    X    X    205    421    740   1160
```

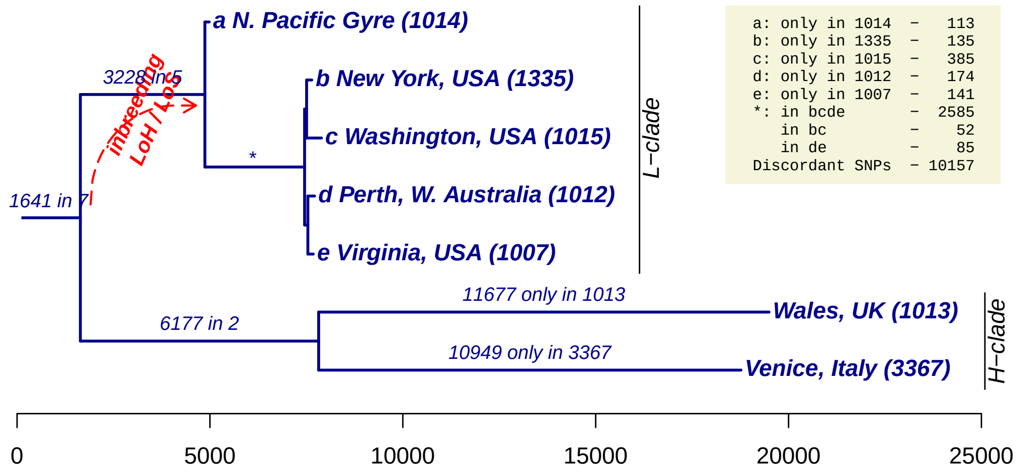


Figure 5: Tree based on qfiltered reads and unfiltered SNPs. “Lengths” are numbers of shared/private SNPs on Chr1.

```
# 112   157      6    X    X         X    X    X    X   1338   1076    665   1343
# 117   164      4    X    X    X         X                19     51     71    163
# 118   165      5    X    X    X         X         X     318    591    957   1140
# 119   166      5    X    X    X         X    X           46    154    220    254
# 120   167      6    X    X    X         X    X    X    1305   2445   4098   1852
# 126   175      6    X    X    X    X    X         X    1709   1352    834   1416
# 127   176      6    X    X    X    X    X    X           57     79     43     68
# Other    (    104  rows    w/    c2    <    51    )       1436    867   1703   2302
# Total                                                  6508   7270  10215  10157
```

And percent of discordant SNPs:

```
nsge <- nrow(sg.edges)
discordv <- consistent.count - sg.edges[nsge,c('count1','count2','count3','count4')] ; discordv

#        count1 count2 count3 count4
# Total   6508   7270  10215  10157

discordv.pct <- round(discordv/consistent.count*100,1) ; discordv.pct

#        count1 count2 count3 count4
# Total   14.5   15.4   21.6   21.4
```

In short, the sharing pattern observed at 7270 or 15.4% of the 47108 medium-stringency consistent SNPs positions observed across all 7 isolates are discordant with the medium tree. (The strict tree has slightly more.)

A majority of the discordant SNPs fall into one of three patterns: 6-way sharing excluding Gyre (likely a technical artifact since the low coverage in Gyre reduces our power to detect SNPs there), or 6-way sharing excluding one of the two H-isolates (likely a reflection of sexuality in the H-clade—SNP positions in a population in Hardy-Weinberg equilibrium are fairly likely to be homozygous for the reference allele in a given individual).

```
third.biggest <- sort(showgroup(pat.summaries,6)[-8,'count2'],decreasing=T)[3]
big.three <- showgroup(pat.summaries,6,c2.thresh = third.biggest); big.three

#       Pat  ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 112   157      6    X    X         X    X    X    X   1338   1076    665   1343
# 120   167      6    X    X    X         X    X    X   1305   2445   4098   1852
# 126   175     6    X    X    X    X    X         X   1709   1352    834   1416
# Other   (     4 rows   w/   c2    < 1076    )         123    136    118    111
# Total                                                4475   5009   5715   4722

big.three.frac <- sum(big.three[1:3,'count2'])/discordv$count2; big.three.frac

# [1] 0.6702889
```

I.e., 67% of discordant SNPs fall into one of these three categories.

Out of curiousity: what is the ratio of full genome to Chr 1 branch lengths. Except for the shortest few, generally ≈10x, as expected given the length of Chr 1:

```
# (vectors derived by editing Newick strings, and in that order)
print(
  c(Italy=86155, Wales=95697, IW=89598, Virg=330,     Aust=632,      VA=1296,
    Puget=2113,  NY=658,      PNY=480,  four=10059,   Gyre=568,      five=39517, all=69526) /
  c(Italy=8813,  Wales=9652,  IW=9365,  Virg=30,      Aust=61,       VA=19,
    Puget=207,   NY=41,       PNY=18,   four=1005,    Gyre=61,       five=3912,  all= 7054),
  digits=3)

# Italy Wales    IW  Virg  Aust    VA Puget    NY   PNY  four  Gyre  five   all
#  9.78  9.91  9.57 11.00 10.36 68.21 10.21 16.05 26.67 10.01  9.31 10.10  9.86

round(genome.length.constants()$genome.length.trunc / genome.length.constants()$chr1.length, digits=4)

# [1] 10.2879
```

9 Semi-Automated Tree-Building

Slightly formalizing the process above: Look for the bifurcation of the 7 strains that maximizes the number of shared SNPs *within* each side of the partition while minimizing the number and fraction of SNPs that are shared by subsets that include at least one strain on each side of the partition. The 2/5 split is the winner, with 6418 SNPs in confict with that partition (16% of the 39842 SNPs not shared by all 7; Chr1 data). The runner-up places the Gyre in a group by itself (7079 = 18% in conflict).

```
treepart <- function(p.summ=pat.summaries, root=127, verbose=T, stringency='count2'){
  root.shared <- p.summ[root+1,stringency]
  df<-NULL
  for(i in 1:floor(root/2)){
    if(bitwAnd(i,root)==i && i < root-i){
      l1 <- showgroup(p.summ,subset=i,split=NULL,proper.subset=F,total=T)
      l  <- l1[nrow(l1),stringency]
      r1 <- showgroup(p.summ,subset=root-i,split=NULL,proper.subset=F,total=T)
      r  <- r1[nrow(r1),stringency]
      c1 <- showgroup(p.summ,subset=root,split=i,proper.subset=T,total=T)
      c  <- c1[nrow(c1),stringency]
      df <- rbind(df, data.frame(pat=i,left=l,right=r,both=l+r,cross=c,all=l+r+c,ratio=c/(l+r+c),
                                 best='',stringsAsFactors=F))
    }
  }
  df$pat<-as.octmode(df$pat)
  maxl <- which.max(df$left)
  maxr <- which.max(df$right)
  maxb <- which.max(df$both)
  minc <- which.min(df$cross)
  minr <- which.min(df$ratio)
  df$best[c(maxl,maxr,maxb,minc,minr)] <- '<'
  df$best[maxl] <- paste(df$best[maxl], 'L') # max Left
  df$best[maxr] <- paste(df$best[maxr], 'R') # max Right
  df$best[maxb] <- paste(df$best[maxb], 'B') # max Both (L+R)
  df$best[minc] <- paste(df$best[minc], 'C') # min Cross
  df$best[minr] <- paste(df$best[minr], 'O') # min ratiO (Cross/(Left+Right+Cross)
  if(verbose){
    same <- all(maxl==c(maxr,maxb,minc,minr))
    cat('root:',       format(as.octmode(root),width=3),
        '; shared:',   root.shared,
        '.  max l',    format(as.octmode(df$pat[maxl]),width=3),
        ', max r',     format(as.octmode(df$pat[maxr]),width=3),
        ', max both',  format(as.octmode(df$pat[maxb]),width=3),
        ', min cross', format(as.octmode(df$pat[minc]),width=3),
        ', min ratio', format(as.octmode(df$pat[minr]),width=3),
        '. \nAll the same?:',same,
        '\n')
    cat('\n')
  }
  return(df)
}
```

```
treepart()

# root: 177 ; shared: 6449 .  max l 077 , max r 010 , max both 010 , min cross 010 , min ratio 010 .
# All the same?: FALSE
#    pat  left right  both cross   all     ratio      best
# 1   01    93 29241 29334 11376 40710 0.2794399
# 2   02  8963 17601 26564 14146 40710 0.3474822
# 3   03  9020 10496 19516 21194 40710 0.5206092
# 4   04   269 28545 28814 11896 40710 0.2922132
# 5   05   334 28340 28674 12036 40710 0.2956522
# 6   06  9190 10106 19296 21414 40710 0.5260133
# 7   07  9273 10006 19279 21431 40710 0.5264309
# 8   10   113 34247 34360  6350 40710 0.1559813 < R B C O
# 9   11   160 28961 29121 11589 40710 0.2846721
# 10  12  9029 12483 21512 19198 40710 0.4715795
# 11  13  9093 10343 19436 21274 40710 0.5225743
# 12  14   332 28414 28746 11964 40710 0.2938836
# 13  15   406 28242 28648 12062 40710 0.2962908
# 14  16  9257 10017 19274 21436 40710 0.5265537
# 15  17  9353  9934 19287 21423 40710 0.5262343
# 16  20  9803 16282 26085 14625 40710 0.3592483
# 17  21  9852  9610 19462 21248 40710 0.5219356
# 18  22 28064  5697 33761  6949 40710 0.1706952
# 19  23 28151   607 28758 11952 40710 0.2935888
# 20  24 10036  9264 19300 21410 40710 0.5259150
# 21  25 10112  9160 19272 21438 40710 0.5266028
# 22  26 28337   301 28638 12072 40710 0.2965365
# 23  27 28470   231 28701 12009 40710 0.2949889
# 24  30  9870 11459 21329 19381 40710 0.4760747
# 25  31  9925  9471 19396 21314 40710 0.5235569
# 26  32 28144  1982 30126 10584 40710 0.2599853
# 27  33 28241   485 28726 11984 40710 0.2943748
# 28  34 10109  9178 19287 21423 40710 0.5262343
# 29  35 10199  9087 19286 21424 40710 0.5262589
# 30  36 28423   226 28649 12061 40710 0.2962663
# 31  37 28587   166 28753 11957 40710 0.2937116
# 32  40   113 28782 28895 11815 40710 0.2902235
# 33  41   159 28529 28688 12022 40710 0.2953083
# 34  42  9027 10293 19320 21390 40710 0.5254237
# 35  43  9098 10173 19271 21439 40710 0.5266274
# 36  44   348 28314 28662 12048 40710 0.2959469
# 37  45   461 28196 28657 12053 40710 0.2960698
# 38  46  9283  9971 19254 21456 40710 0.5270450
# 39  47  9450  9910 19360 21350 40710 0.5244412
# 40  50   176 28634 28810 11900 40710 0.2923115
# 41  51   229 28429 28658 12052 40710 0.2960452
# 42  52  9094 10190 19284 21426 40710 0.5263080
# 43  53  9175 10091 19266 21444 40710 0.5267502
# 44  54   419 28216 28635 12075 40710 0.2966102
# 45  55   558 28112 28670 12040 40710 0.2957504
# 46  56  9360  9895 19255 21455 40710 0.5270204
# 47  57  9567  9841 19408 21302 40710 0.5232621
# 48  60  9873  9454 19327 21383 40710 0.5252518
# 49  61  9934  9320 19254 21456 40710 0.5270450
# 50  62 28157   478 28635 12075 40710 0.2966102
# 51  63 28278   380 28658 12052 40710 0.2960452
# 52  64 10140  9141 19281 21429 40710 0.5263817
# 53  65 10293  9068 19361 21349 40710 0.5244166
# 54  66 28519   200 28719 11991 40710 0.2945468
# 55  67 28886   147 29033 11677 40710 0.2868337
# 56  70  9941  9362 19303 21407 40710 0.5258413
# 57  71 10012  9247 19259 21451 40710 0.5269221
# 58  72 28240   396 28636 12074 40710 0.2965856
# 59  73 28379   312 28691 12019 40710 0.2952346
# 60  74 10223  9065 19288 21422 40710 0.5262098
# 61  75 10416  9000 19416 21294 40710 0.5230656
# 62  76 28629   131 28760 11950 40710 0.2935397
# 63  77 29112    84 29196 11514 40710 0.2828298       < L
```

Comparing the 5/2 split to the second-place NPG/rest split (below), the former has fewer pattern instances in conflict with the split (6418 vs 7079), as well as somewhat more random distribution of the conflicting patterns (92 vs 62 rows), whereas the 1/6 split has the majority of its conflicts (3912 of 7079, or 55%) concentrated in one pattern—the 5 NE strains. Collectively, these seem to favor the 5/2 split as the correct "history."

```
showgroup(pat.summaries,split=strtoi('022'), subset=127, proper.subset=T, c2.thresh=100)

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 56    067     5         X    X         X    X    X     48    109    257    104
# 104   147     5    X    X              X    X    X    205    421    740   1160
# 112   157     6    X    X         X    X    X    X   1338   1076    665   1343
# 118   165     5    X    X    X         X         X    318    591    957   1140
# 119   166     5    X    X    X         X    X          46    154    220    254
# 120   167     6    X    X    X         X    X    X   1305   2445   4098   1852
# 126   175     6    X    X    X    X    X         X   1709   1352    834   1416
# Other   (    85 rows   w/   c2    <  100    )        1415    801   1317   1735
# Total                                                6384   6949   9088   9004

showgroup(pat.summaries,split=strtoi('010'), subset=127, proper.subset=T, c2.thresh=100)

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 110   155     5    X    X         X    X         X   4136   3560   2135   3228
# 112   157     6    X    X         X    X    X    X   1338   1076    665   1343
# 126   175     6    X    X    X    X    X         X   1709   1352    834   1416
# Other   (    59 rows   w/   c2    <  100    )         470    362    335    590
# Total                                                7653   6350   3969   6577
```

Below is the full summary of shared SNPs that do *not* directly correspond to tree splits, e.g. deep coalescence, independent coincident mutations, false positives/false negatives in the shared SNP calls, loss of SNPs in hemizygous regions, etc. (Additionally, SAMTools' SNP calls exclude positions it judges to be homozygous, and I think it operates without regard to the reference sequence, so homozygous nonreference positions, while rare except in IT/Wales, often are not called SNPs by SAMTools, but are relevant for this analysis. Provided the position is called a SNP in some other isolate, the consistency filtering we've done above should recover it, but this is still worth keeping in mind when examining the data.)

First, here are SNPs that "coalesce" on the branch from the LCA of bcde, i.e., shared among some nonempty, proper subset of bcde other than bc or de. There are 8 such patterns: any of the 4 choose 3 trios plus any of the 4 pairs having exactly one of bc.

```
sg4 <- showgroup(pat.summaries, subset=strtoi('0145'), split=5, proper.subset = F)
sg4

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 34    041     2         X                        X      2      4     23     36
# 37    044     2         X              X                6     17     80    155
# 38    045     3         X              X         X     12     44    184    131
# 66    101     2    X                             X      3      5      7     20
# 69    104     2    X                   X                2     10     40    107
# 70    105     3    X                   X         X      9     14     37     63
# 98    141     3    X    X                        X      5      9     20     40
# 101   144     3    X    X              X               18     74    159    355
# 102   145     4    X    X              X         X    598   1356   2429   2585
# Total                                                 655   1533   2979   3492

sg4n <- nrow(sg4)
sg4pct <- round(sg4$count2[sg4n-1]/sg4$count2[sg4n]*100,1)
sg4pct

# [1] 88.5
```

So, of the 1533 SNPs found only in bcde, 88.5% have a sharing pattern consistent with the given tree structure.

Similarly, we analyze patterns relative to the root of the L-clade (14 patterns—any nonempty proper subset of bcde together with a):

```
sg5 <- showgroup(pat.summaries,subset=strtoi('0155'), split=8, proper.subset = F)
sg5

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 10    011     2                   X         X      3      5      4     14
# 13    014     2                   X    X           4      1      6     12
# 14    015     3                   X    X         X      6      4      4      7
# 41    050     2         X         X                     5      1      3      7
# 42    051     3         X         X              X      0      2      4      4
# 45    054     3         X         X    X                1      7     12     18
# 46    055     4         X         X    X         X      8     15     24     36
# 73    110     2    X              X                     2      1      3      7
# 74    111     3    X              X              X      0      1      1      1
# 77    114     3    X              X    X                2      4      4      8
# 78    115     4    X              X    X         X      1      4      6      9
# 105   150     3    X    X         X                     0      1      0      6
# 106   151     4    X    X         X              X      2      2      4     14
# 109   154     4    X    X         X    X               24     45     34    103
# 110   155     5    X    X         X    X         X   4136   3560   2135   3228
# Total                                                4194   3653   2244   3474

sg5n <- nrow(sg5)
sg5pct <- round(sg5$count2[sg5n-1]/sg5$count2[sg5n]*100,1)
```

I.e., of the 3653 SNPs found only in abcde, 97.5% have a sharing pattern consistent with the given tree structure.

Finally, how many SNPs have patterns inconsistent with the 5-2 split, i.e., include at least one strain on each side of the 5-2 split, but not shared by all 7?

```
sg7 <- showgroup(pat.summaries, subset=127, split=strtoi('022'), proper.subset=F)
sg7

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 4     003     2                             X    X     87     15     37     31
# 7     006     2                        X    X          65      9     20     41
# 8     007     3                        X    X    X      4      3     19     10
# 11    012     2                   X         X          41      4      2      5
# 12    013     3                   X         X    X      3      2      2      2
# 15    016     3                   X    X    X           1      0      1      2
# 16    017     4                   X    X    X    X      1      2      1      2
# 18    021     2              X                   X    102      7     24      9
# 20    023     3              X              X    X     89     23     35     17
# 21    024     2              X         X               62     15     21     46
# 22    025     3              X         X         X      8      4     23     21
# 23    026     3              X         X    X          84     31     37     32
# 24    027     4              X         X    X    X     15     16     37     24
# 25    030     2              X    X                    65      5      1      9
# 26    031     3              X    X              X      3      1      0      0
# 27    032     3              X    X         X          66      9      6      5
# 28    033     4              X    X         X    X      4      2      4      6
# 29    034     3              X    X    X                  2      5      1      1
# 30    035     4              X    X    X         X      2      4      0      0
# 31    036     4              X    X    X    X           5      0      3      3
# 32    037     5              X    X    X    X    X     12     11      8      5
# 35    042     2         X                   X          47      2     20     27
# 36    043     3         X                   X    X      8     10     14      6
# 39    046     3         X              X    X           4     12     41     55
# 40    047     4         X              X    X    X      9     26     68     60
# 43    052     3         X         X         X           1      0      0      1
# 44    053     4         X         X         X    X      0      1      1      2
# 47    056     4         X         X    X    X           2      2      2      5
# 48    057     5         X         X    X    X    X      4      9      8     17
# 49    060     2         X    X                         59      8     24     28
# 50    061     3         X    X                   X      1      8     11     12
# 51    062     3         X    X              X          86     21     29     36
# 52    063     4         X    X              X    X      9     12     34     21
# 53    064     3         X    X         X                2     17     52     60
# 54    065     4         X    X         X         X      8     21     68     48
# 55    066     4         X    X         X    X          15     43    102     76
# 56    067     5         X    X         X    X    X     48    109    257    104
# 57    070     3         X    X    X                     3      0      0      2
# 58    071     4         X    X    X              X      0      2      0      0
# 59    072     4         X    X    X         X           4      2      1      2
# 60    073     5         X    X    X         X    X      3      3      3      3
# 61    074     4         X    X    X    X                1      2      1      7
# 62    075     5         X    X    X    X         X      8      7     10     11
# 63    076     5         X    X    X    X    X           9     10     12      7
# 64    077     6         X    X    X    X    X    X     43     46     62     26
# 67    102     2    X                        X          34      4     11     25
# 68    103     3    X                        X    X      2      3      6      8
# 71    106     3    X                   X    X           4     10      9     27
# 72    107     4    X                   X    X    X      6     10      6     16
# 75    112     3    X              X         X           0      1      0      0
# 76    113     4    X              X         X    X      0      0      0      1
# 79    116     4    X              X    X    X           1      0      1      5
# 80    117     5    X              X    X    X    X      1      1      3      7
# 81    120     2    X         X                         49      5      7     31
# 82    121     3    X         X                   X      2      0      2      4
# 83    122     3    X         X              X          45      6     12     26
# 84    123     4    X         X              X    X      5      9     13      8
# 85    124     3    X         X         X                2      7     21     35
# 86    125     4    X         X         X         X      3      4     14     16
# 87    126     4    X         X         X    X          10     17     14     43
# 88    127     5    X         X         X    X    X     13     27     49     47
# 89    130     3    X         X    X                     1      1      1      1
# 90    131     4    X         X    X              X      0      0      0      2
# 91    132     4    X         X    X         X           1      1      1      1
# 92    133     5    X         X    X         X    X      2      3      0      0
# 93    134     4    X         X    X    X                6      3      2      3
# 94    135     5    X         X    X    X         X      4      2      0      7
# 95    136     5    X         X    X    X    X           5      3      0      5
# 96    137     6    X         X    X    X    X    X      8      6      8     14
# 99    142     3    X    X                   X           3      3      9     15
# 100   143     4    X    X                   X    X      1      3      4     20
# 103   146     4    X    X              X    X           9     34     69    140
# 104   147     5    X    X              X    X    X    205    421    740   1160
# 107   152     4    X    X         X         X           1      3      1      4
# 108   153     5    X    X         X         X    X      4      0      0      7
# 111   156     5    X    X         X    X    X          11      7      9     43
# 112   157     6    X    X         X    X    X    X   1338   1076    665   1343
# 113   160     3    X    X    X                          6      3      7     23
# 114   161     4    X    X    X                   X      3      8     10     18
# 115   162     4    X    X    X              X           8     11     20     33
# 116   163     5    X    X    X              X    X     15     14     21     33
# 117   164     4    X    X    X         X               19     51     71    163
# 118   165     5    X    X    X         X         X    318    591    957   1140
# 119   166      5    X    X    X         X    X              46    154    220    254
# 120   167      6    X    X    X         X    X    X   1305   2445   4098   1852
# 121   170      4    X    X    X    X                     0      1      1      3
# 122   171      5    X    X    X    X              X      4      4      2      7
# 123   172      5    X    X    X    X         X           3      6      3      5
# 124   173      6    X    X    X    X         X    X     15      5      5      3
# 125   174      5    X    X    X    X    X                5     14     17     35
# 126   175      6    X    X    X    X    X         X   1709   1352    834   1416
# 127   176      6    X    X    X    X    X    X          57     79     43     68
# 128   177      7    X    X    X    X    X    X    X   8132   6449   3873   1641
# Total                                               14516  13398  12961  10645

sg7n <- nrow(sg7)
sg7pct <- round(sg7$count2[sg7n-1]/sg7$count2[sg7n]*100,1)
sg7pct

# [1] 48.1
```

A more compact version of that table, showing only the larger counts:

```
thresh <- signif(.02 * sg7$count2[sg7n],1)
thresh

# [1] 300

showgroup(pat.summaries, subset=127, split=strtoi('022'), proper.subset=F, c2.thresh = thresh)

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 104   147      5    X    X              X    X    X    205    421    740   1160
# 112   157      6    X    X         X    X    X    X   1338   1076    665   1343
# 118   165      5    X    X    X         X         X    318    591    957   1140
# 120   167      6    X    X    X         X    X    X   1305   2445   4098   1852
# 126   175      6    X    X    X    X    X         X   1709   1352    834   1416
# 128   177      7    X    X    X    X    X    X    X   8132   6449   3873   1641
# Other   (    87 rows   w/   c2    <  300    )        1509   1064   1794   2093
# Total                                               14516  13398  12961  10645
```

So, of the 13398 SNPs found both in the L- and H-clades, 48.1% have a sharing pattern consistent with the given tree structure, i.e., are found in all 7 isolates. Among the others, three patterns dominate—(i) the 6-way pattern excluding the Gyre is the largest, plausibly explained by 7-way sharing from which the Gyre drops out due to low coverage/high error rate, (ii) the 6-way excluding Italy, and (iii) ditto for Wales. Origin of the later two cases is unclear, but may partly reflect Hardy-Weinberg—some positions that are *population-level* SNPs in those isolates will be homozygous-reference in the CCMP founder cell for IT or Wales. If I take the 7-way shared SNP count (69526) as a surrogate approximating the number of population-level SNPs in either IT or Wales that are shared with the L-clade, then I might expect, based on HWE, roughly half that number to to be lost (become homozygous) in IT, and a similar number in Wales. However, the observed counts of these positions are lower by ≈20K than I might have guessed from HWE, perhaps suggesting that IT and Wales are distinct populations, each with a pool of many thousand private polymorphisms.

In aggregate:

```
untreelike <-
  sg7$count2[sg7n]-sg7$count2[sg7n-1] +
  sg5$count2[sg5n]-sg5$count2[sg5n-1] +
  sg4$count2[sg4n]-sg4$count2[sg4n-1]
untreelike

# [1] 7219

consistent.count[2]

# [1] 47108

unpct <- round(untreelike/consistent.count[2]*100,1)
unpct

# [1] 15.3
```

I.e., 7219 or 15.3% of the 47108 consistent SNPs identified (by criterion 2) across all 7 isolates are discordant with the assumed tree.

Overall, based on this data, I take the following to be obvious: (a) separation of the the H-isolates from the L-isolates (and from each other??), and (b) near-identity of the L-isolates. Due to the small counts, the exact topology among the L-isolates (esp. bcde) is uncertain, but *any* topology there is consistent with the asexual/clonal/global-expansion hypothesis, so there is little point in examining this subtree more carefuly. Again, we believe the (apparent) slight separation of the Gyre from the other L-isolates is largely driven by technical artifacts (lower coverage/higher error rates) in the sequencing rather than by biological effects. However, the discord between Gyre SNPs and others is the major substantive ambiguity in the offered tree. Nevertheless, in the next section we show by a bootstrap analysis that the offered placement of Gyre with respect to the other 4 L-isolates is strongly supported by the data.

9.1 Bootstrap

How robust is the inferred tree? Italy/Wales seem clearly related to each other but separate from the other 5. Likewise, the 4 coastal L-isolates seem to be closely related, with little data to separate them (and perhaps little sense in trying). So, the key question here is whether the top level bifurcation is 2/5 or NPG/6. Here, we do a simple bootstrap test (on c2 numbers only) to see whether the 2/5 split is consistently the most parsimonious.

```
n2 <- sum(pattern.counts[[2]][,2]); n2

# [1] 47108
```

Conceptually, we sample, with replacement, n2=47108 SNP positions from among the 47108 positions declared consisent SNPs according to criterion c2, and recalculate the statistics examined above to see whether the 2/5 split again minimizes conflicting sharing patterns. This resampling/calculation is repeated nboot times (set near front of file). Since all that matters is the sharing pattern, this procedure is expedited by actually sampling 47108 independent integers in the range 0:127 with probabilities proportional to the patttern counts given in column 2 of pattern.counts[[2]]. The sample is then tabulated in a 128 row table analogous to pattern.summaries, for analysis by showgroups/treepart, as above.

```
boot.sample <- sample(0:127,n2,replace=T,prob=pattern.counts[[2]][,2])
str(boot.sample)

#  int [1:47108] 127 117 16 127 16 119 16 127 18 18 ...

boot.count <- mytable(boot.sample,c(0,127))
boot.count[c(1:4,125:128),] # show a few rows

#      val count
# [1,]   0    57
# [2,]   1    52
# [3,]   2  8971
# [4,]   3    20
# [5,] 124    10
# [6,] 125  1379
# [7,] 126    98
# [8,] 127  6408

boot.counts <- list(NULL,boot.count,NULL) # dummy list with just c2 summaries
cor(pattern.counts[[2]][,2],boot.counts[[2]][,2]) # just curious - how correlated are they?

# [1] 0.9998917

boot.summaries <- pat.summary(boot.counts)
showgroup(boot.summaries,c2.thresh=400) #show a few rows

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 3     002     1                             X          NA   8971     NA     NA
# 17    020     1              X                         NA   9531     NA     NA
# 19    022     2              X              X          NA   9413     NA     NA
# 102   145     4    X    X              X         X     NA   1329     NA     NA
# 104   147     5    X    X              X    X    X     NA    461     NA     NA
# 110   155     5    X    X         X    X         X     NA   3635     NA     NA
# 112   157     6    X    X         X    X    X    X     NA   1050     NA     NA
# 118   165     5    X    X    X         X         X     NA    574     NA     NA
# 120   167     6    X    X    X         X    X    X     NA   2468     NA     NA
# 126   175     6    X    X    X    X    X         X     NA   1379     NA     NA
# 128   177     7    X    X    X    X    X    X    X     NA   6408     NA     NA
# Other   (   117 rows   w/   c2    <  400    )          NA   1889     NA     NA
# Total                                                  NA  47108     NA     NA
```

Tree partition analysis (and how to pluck out only the best rows based on 3 smallest cross counts and “best” criteria):

```
tp <- treepart(boot.summaries,root=127) ; tp

# root: 177 ; shared: 6408 .  max l 077 , max r 010 , max both 010 , min cross 010 , min ratio 010 .
# All the same?: FALSE
#    pat  left right  both cross   all     ratio      best
# 1   01   109 29163 29272 11485 40757 0.2817921
# 2   02  9028 17453 26481 14276 40757 0.3502711
# 3   03  9100 10287 19387 21370 40757 0.5243271
# 4   04   260 28477 28737 12020 40757 0.2949187
# 5   05   331 28253 28584 12173 40757 0.2986726
# 6   06  9236  9908 19144 21613 40757 0.5302893
# 7   07  9331  9803 19134 21623 40757 0.5305346
# 8   10   133 34177 34310  6447 40757 0.1581814 < R B C O
# 9   11   192 28857 29049 11708 40757 0.2872635
# 10  12  9108 12216 21324 19433 40757 0.4768015
# 11  13  9190 10119 19309 21448 40757 0.5262409
# 12  14   336 28337 28673 12084 40757 0.2964889
# 13  15   420 28145 28565 12192 40757 0.2991388
# 14  16  9316  9808 19124 21633 40757 0.5307800
# 15  17  9428  9718 19146 21611 40757 0.5302402
# 16  20  9588 16402 25990 14767 40757 0.3623181
# 17  21  9645  9657 19302 21455 40757 0.5264126
# 18  22 27972  5747 33719  7038 40757 0.1726820
# 19  23 28079   605 28684 12073 40757 0.2962191
# 20  24  9808  9350 19158 21599 40757 0.5299458
# 21  25  9887  9231 19118 21639 40757 0.5309272
# 22  26 28230   322 28552 12205 40757 0.2994578
# 23  27 28383   245 28628 12129 40757 0.2975931
# 24  30  9671 11502 21173 19584 40757 0.4805064
# 25  31  9736  9505 19241 21516 40757 0.5279093
# 26  32 28062  1935 29997 10760 40757 0.2640037
# 27  33 28182   470 28652 12105 40757 0.2970042
# 28  34  9897  9250 19147 21610 40757 0.5302157
# 29  35  9991  9148 19139 21618 40757 0.5304120
# 30  36 28326   233 28559 12198 40757 0.2992860
# 31  37 28505   169 28674 12083 40757 0.2964644
# 32  40   120 28703 28823 11934 40757 0.2928086
# 33  41   175 28434 28609 12148 40757 0.2980592
# 34  42  9093 10087 19180 21577 40757 0.5294060
# 35  43  9177  9967 19144 21613 40757 0.5302893
# 36  44   340 28261 28601 12156 40757 0.2982555
# 37  45   457 28117 28574 12183 40757 0.2989180
# 38  46  9328  9784 19112 21645 40757 0.5310744
# 39  47  9503  9716 19219 21538 40757 0.5284491
# 40  50   196 28554 28750 12007 40757 0.2945997
# 41  51   261 28328 28589 12168 40757 0.2985499
# 42  52  9173  9966 19139 21618 40757 0.5304120
# 43  53  9271  9869 19140 21617 40757 0.5303874
# 44  54   425 28154 28579 12178 40757 0.2987953
# 45  55   575 28025 28600 12157 40757 0.2982801
# 46  56  9418  9692 19110 21647 40757 0.5311235
# 47  57  9647  9632 19279 21478 40757 0.5269770
# 48  60  9656  9511 19167 21590 40757 0.5297250
# 49  61  9726  9369 19095 21662 40757 0.5314915
# 50  62 28065   483 28548 12209 40757 0.2995559
# 51  63 28204   380 28584 12173 40757 0.2986726
# 52  64  9908  9237 19145 21612 40757 0.5302647
# 53  65 10067  9145 19212 21545 40757 0.5286209
# 54  66 28407   229 28636 12121 40757 0.2973968
# 55  67 28791   167 28958 11799 40757 0.2894963
# 56  70  9739  9405 19144 21613 40757 0.5302893
# 57  71  9820  9286 19106 21651 40757 0.5312216
# 58  72 28158   385 28543 12214 40757 0.2996786
# 59  73 28319   301 28620 12137 40757 0.2977893
# 60  74 10009  9147 19156 21601 40757 0.5299948
# 61  75 10211  9065 19276 21481 40757 0.5270506
# 62  76 28526   146 28672 12085 40757 0.2965135
# 63  77 29038    91 29129 11628 40757 0.2853007       < L
```

```
otp <- order(tp[,'cross'])[1:3]     # 3 smallest 'cross' counts
btp <- which(tp[,'best'] != '')     # 'best' by Left/Right/Both/Cross/ratiO
toptp <- unique(c(otp,btp,18,8))    # above, plus 5/2, 6/1 splits
print(tp[toptp,])                   # show the winners

#    pat  left right  both cross   all     ratio      best
# 8   10   133 34177 34310  6447 40757 0.1581814 < R B C O
# 18  22 27972  5747 33719  7038 40757 0.1726820
# 26  32 28062  1935 29997 10760 40757 0.2640037
# 63  77 29038    91 29129 11628 40757 0.2853007       < L
```

Now repeat the above nboot times, and summarize results:

```
nboot <- params$nboot #  default from params set in section 2
nboot <- ((nboot+2) %/% 4) * 4 + 1  # summary is cleaner if n mod 4 == 1, so int median/quartiles
cat('***\n*** Doing', nboot, 'bootstrap replicates.\n***\n')

# ***
# *** Doing 5 bootstrap replicates.
# ***

bcor <- numeric(nboot)
b52cross <- integer(nboot)
b61cross <- integer(nboot)
brev <- logical(nboot)
for(i in 1:nboot){
  boot.sample <- sample(0:127,n2,replace=T,prob=pattern.counts[[2]][,2])
  boot.count <- mytable(boot.sample,c(0,127))
  boot.counts <- list(NULL,boot.count,NULL) # dummy list with just c2 summaries
  boot.summaries <- pat.summary(boot.counts)
  tp <- treepart(boot.summaries,root=127, verbose=F)
  bcor[i] <- cor(pattern.counts[[2]][,2],boot.counts[[2]][,2]) # just curious - how correlated are they?
  b52cross[i] <- tp[18,'cross']
  b61cross[i] <- tp[ 8,'cross']
  brev[i] <- (b52cross[i] > b61cross[i])
  if(brev[i]){
    # show the unexpected ones; probably breaks w/ cache
    otp <- order(tp[,'cross'])[1:3]
    btp <- which(tp[,'best'] != '')
    toptp <- unique(c(otp,btp,18,8))
    print(tp[toptp,])
  }
}

#    pat  left right  both cross   all     ratio      best
# 8   10   112 34360 34472  6348 40820 0.1555120 < R B C O
# 18  22 28198  5555 33753  7067 40820 0.1731259
# 26  32 28273  1867 30140 10680 40820 0.2616365
# 63  77 29259    85 29344 11476 40820 0.2811367       < L
#    pat  left right  both cross   all     ratio      best
# 8   10   103 34256 34359  6326 40685 0.1554873 < R B C O
# 18  22 28070  5667 33737  6948 40685 0.1707755
# 26  32 28135  2002 30137 10548 40685 0.2592602
# 63  77 29045    93 29138 11547 40685 0.2838147       < L
#    pat  left right  both cross   all     ratio      best
# 8   10   129 34284 34413  6377 40790 0.1563373 < R B C O
# 18  22 28017  5778 33795  6995 40790 0.1714881
# 26  32 28110  2081 30191 10599 40790 0.2598431
# 63  77 29109    94 29203 11587 40790 0.2840647       < L
#    pat  left right  both cross   all     ratio      best
# 8   10   104 34312 34416  6376 40792 0.1563052 < R B C O
# 18  22 28127  5705 33832  6960 40792 0.1706217
# 26  32 28208  2002 30210 10582 40792 0.2594136
# 63  77 29232    88 29320 11472 40792 0.2812316       < L
#    pat  left right  both cross   all     ratio      best
# 8   10   117 34201 34318  6319 40637 0.1554987 < R B C O
# 18  22 28044  5689 33733  6904 40637 0.1698944
# 26  32 28130  1961 30091 10546 40637 0.2595172
# 63  77 29110    86 29196 11441 40637 0.2815415       < L

# summarize:
corsummary <- t(as.matrix(c(summary(bcor),sd=sd(bcor))))
row.names(corsummary) <- 'bcor'
bdelta <- b61cross-b52cross
brevp <- 100*brev   # make it percent reversed instead of logical
thesummary <- rbind(summary(b52cross),summary(b61cross),summary(c(bdelta)),summary(brevp))
row.names(thesummary) <- c('b52cross', 'b61cross', 'b61-b52', '% rev')
thesummary <- cbind(thesummary, sd=c(sd(b52cross),sd(b61cross),sd(bdelta),sd(brevp)))
```

SUMMARY: In 5 bootstrap replicates, we saw 5 samples with the 6/1 split having fewer conflicts than the 5/2 split, and the minimum separation between them was ≈ -13 sigma, hence highly statistically significant.

```
# 'opt' hacking is trying to force knitr to show more digits of bcor in summary, as Rstudio does, but
# it still fails...  Bottom line is the correlation seems to be  > .999 in all samples, rounds to 1.0,
# as seen in this run of 1001 samples cut/paste from Rstudio:
#          Min.        1st Qu.     Median      Mean        3rd Qu. Max.   sd
# bcor     "   0.9998" "   0.9999" "   0.9999" "   0.9999" "   1"  "   1" "   0.00003462"
# > max(bcor)
# [1] 0.9999915
o.opts <- options(digits=7,width=127)
format(rbind(corsummary,thesummary),scientific=F,digits=4,drop0trailing=T)

#          Min.            1st Qu.         Median          Mean            3rd Qu.         Max.            sd
# bcor     "   0.99990971" "   0.99996074" "   0.99996325" "   0.99995328" "   0.99996497" "   0.99996773" "   0.
# b52cross "6904"          "6948"          "6960"          "6974.8"        "6995"          "7067"          "  60.
# b61cross "6319"          "6326"          "6348"          "6349.2"        "6376"          "6377"          "  27.
# b61-b52  "-719"          "-622"          "-618"          "-625.6"        "-585"          "-584"          "  55.
# % rev    " 100"          " 100"          " 100"          " 100"          " 100"          " 100"          "   0"

options(o.opts)
```

Based on this, it is reasonable to claim that we are confident that the tree topology is as shown in the earlier figures, with the exception of the exact order of the splits with the 4 NE coastal isolates.

10 Notes

This section is a random brain dump of limitations of the current analysis, ideas for improvements, etc. In the main, these may not be worth doing, unless we see significant holes or get pushed by reviewers, etc, but I wanted to catalog before we forget them.
Noise: Various sources of “noise” in the data:

1. Read errors, low read depth — perhaps fixed by medium/strict thresholding

2. Deep coalescence

3. Skew because 1335 is the reference. (Julie notes we could partially fix this by remapping based on discovered SNPs, tho that wouldn’t fix gross misassembly in 1335, e.g. collapsed or misordered tandem duplicates, or segments missing in 1335 that are present in one or more other strains, etc.; much harder to fix those, let’s just hope they are rare...)

4. Varying error rates and sequencing depth among the 7. E.g., plausibly the 1000 SNPs shared by 4 but not by Gyre are a result of lower read depth (we missed a SNP that is actually present) and/or higher error rates (causing a position to appear inconsistent in gyre) in the gyre data. I can’t think of a way to correct for this effect. It might be possible, perhaps by simulation, to estimate the size of the effect and see whether it could explain ≈1000 SNPs.

5. Varying numbers of founder cells in the sequencing cultures. (Again, I made some attempts at modeling this, but nothing very satisfactory yet.)

6. Tri-allelic positions where stochastic fluctuation in sequence sampling promotes the rare allele to prominence. (Julie replies: "isn't this the same as more than one founder cell? If they are diploid there should only ever be two alleles, unless there were random and very rare, thus unlikely, trisomy events?" I agree, but it is a concrete example of an effect of multiple founders that might be important. Not sure this is the most important such effect...)

7. Gaps/indels - alignments are likely to be of lower quality in the vicinity of an indel, so, maybe lower coverage/more SNPs. We ignored them. Does this add any systematic bias? e.g. if one strain had more indels than another, would this confound other analyses? unclear. Julie suggested a paper titled "Barking up the wrong tree-length: yada yada yada gap penalties"; maybe relevant?

Other Items/Potential To Dos:

1. any spacial structure to various sub-classes?

2. after top level split, should I reanalyze halves of partition in isolation? said another way, I think the tree-building is sensible, but not sure it's optimal.

3. if we believe no sex, then I think gain of SNP should be more common than loss of SNP, since the later can only happen by (a) mutation reverting to reference, (b) second mutation matching nonreference, (c) homologous repair (look for blocks of LOH), or (d) false negative e.g. from low read depth. Does tree-building appropriately weight the gain vs loss cases? (Does it even care?)

4. should we weight coding and/or nonsynonomous SNPs more heavily? Julie says "you do not want to weight the coding or nonsynonomous/coding SNPs because for time you want the more clock-like neutral mutations." I.e., I got this backwards. Maybe should redo tree based on noncoding SNPs only.

5. We could also do an actual parsimony analysis based on 2-state model (homozygous-ref vs not), but I'm not quite sure how to handle this in a mixed sex/nosex case.

6. Might be interesting to look at sharing just within (shared?) deserts. Given tree model above and expectation that bottleneck followed split of H- from L-clades, I would expect little or no sharing of L-clade desert SNPs with H-clade; sharing between It/Wales might suggest "desert" is actually a region under strong purifying selection (e.g. a gene); sharing/non-sharing within L-clade deserts might suggest more about evo history of the 5.

11 Appendix: Old Trees, etc.

Tangents, old stuff of historical interest at best, etc..

11.1 HWE Sharing

Tangent: As a function of nonref allele freq, assuming HWE, what is probability that nonref allele will be seen in k strains, $0 \leq k \leq 4$ (Fig 6).

```
myfigpath.h <- paste(getwd(), '/figs-knitr/', sep='')
```

```
p <- (0:20)/20
q <- 1-p
r <- 2*p*q+p^2
plot(  p, 1*q^0*r^4, type='b',pch='4', ylab="share prob")
points(p, 4*q^2*r^3, type='b',pch='3')
points(p, 6*q^4*r^2, type='b',pch='2')
points(p, 4*q^6*r^1, type='b',pch='1')
points(p, 1*q^8*r^0, type='b',pch='0')
```

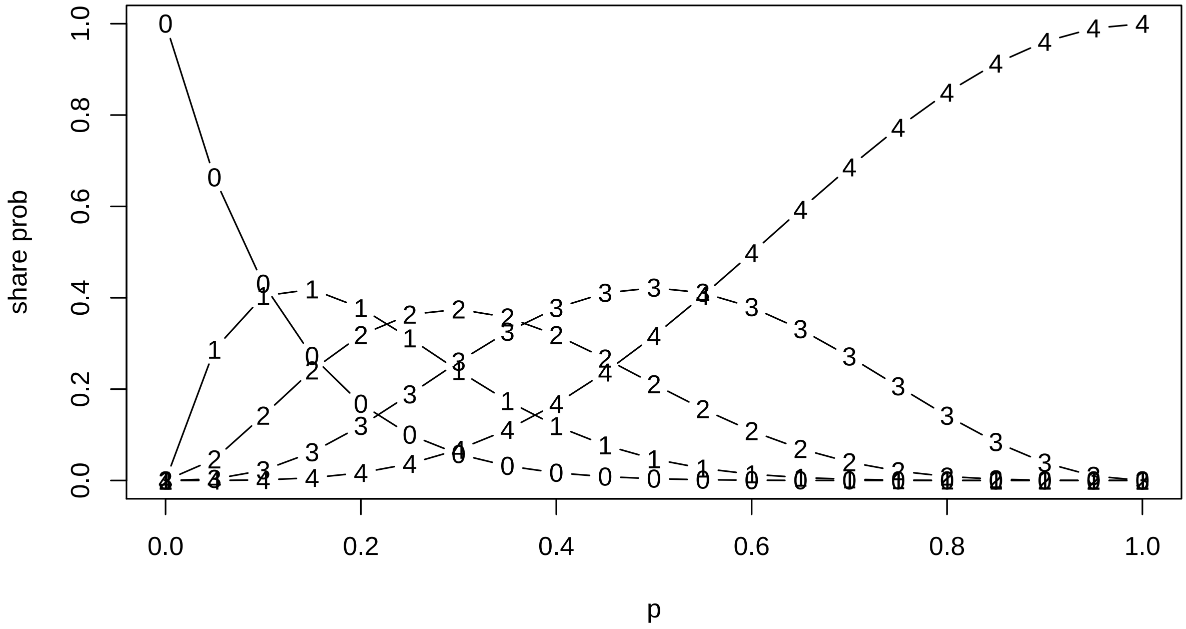


Figure 6: Sharing Probability

11.2 Old Tree Stuff

All based on un-q-filtered reads.

The first pass at the tree analysis was the Chr1 tree, *loose criteria* (c1); it is rendered via `http://iubio.bio.indiana.edu/treeapp/treeprint-form.html` as Fig 7, and in newick format is:

```
newick.chr1.loose <- '(((tp3367_Italy:4551,tp1013_Wales:4954):5920,(((tp1007_Virginia:10,tp1012_Australia:29):9,(
cat.hardwrap(newick.chr1.loose)

# (((tp3367_Italy:4551,tp1013_Wales:4954):5920,(((tp1007_Virginia:10,tp1012_Austra
# lia:29):9,(tp1015_Puget_Sound:90,tp1335_NY:13):11):320,tp1014_Gyre:22):3484):859
# 3,outgroup:0);
```

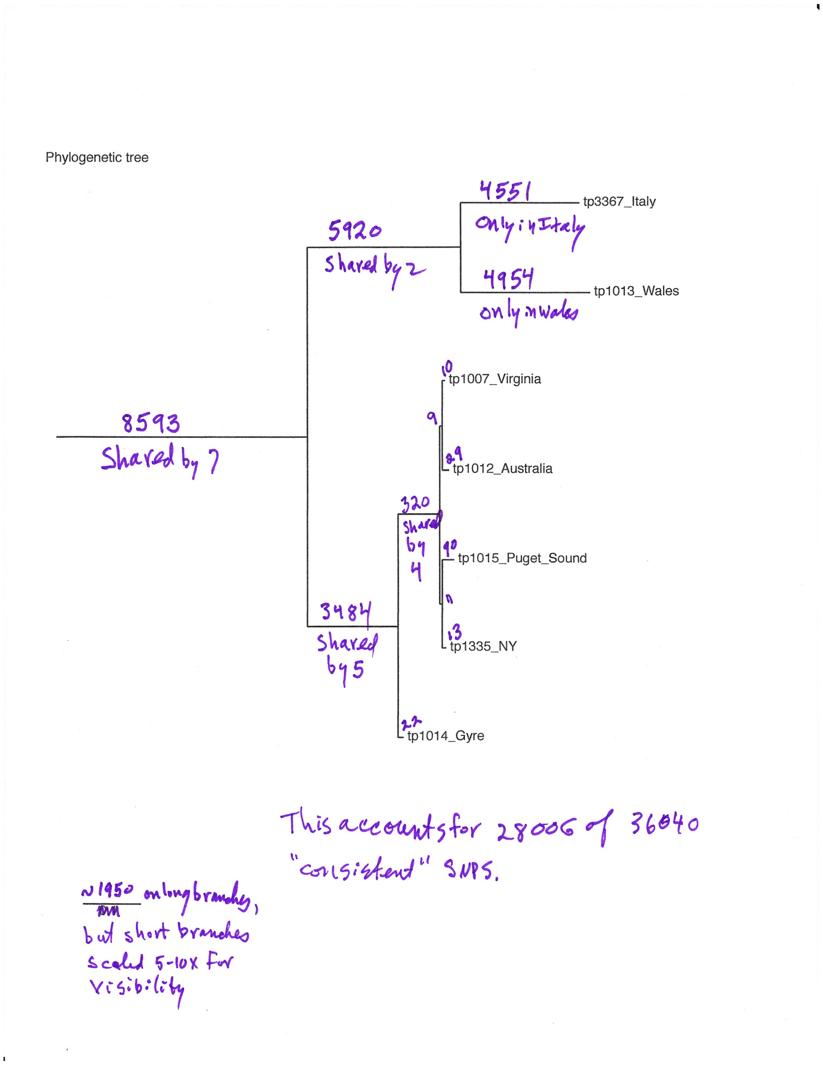
Chr 1 tree based on *medium criteria* (c2) has exactly the same topology is, although the branch lengths are different. As noted earlier, the length of the branch labeled “*” is probably inflated by lower coverage and higher error rate in 1014, which may mask further legitimate sharing between it and the other L-isloates. The branch lengths among the other 4 are too short for its topology to be convincing without a more rigorous analysis (e.g., a bootstrap test).

Chr1 tree, medium criteria, in newick format:

```
newick.chr1.med <- '(((tp3367_Italy:8813,tp1013_Wales:9652):9365,(((e_tp1007_Virginia:30,d_tp1012_Australia:61):1
cat.hardwrap(newick.chr1.med)

# (((tp3367_Italy:8813,tp1013_Wales:9652):9365,(((e_tp1007_Virginia:30,d_tp1012_Au
# stralia:61):19,(c_tp1015_Puget_Sound:207,b_tp1335_NY:41):18):1005,a_tp1014_Gyre:
# 61):3912):7054,outgroup:0);
```

NOTE: In early code, tree was not being recalculated; it was defined by constants in the following code chunk, hand-copied from the analysis above.



This accounts for 28006 of 36040 "consistent" SNPS.

$\frac{\sim 1950}{10M}$ on long branches, but short branches scaled 5-10x for visibility

Figure 7: Inferred Tree, based on Chr1, un-q-filtered reads, loose criteria. (Note: to visually resolve the edges among the 5, their lengths were scaled by 5x – 10x in this figure, but not in the newick description shown in the text.)

```
# tree parameters as nested lists
#   Internal nodes have subtrees sub1/2 and length
#   Root has sub1/2, but no length
#   Leaves have where, length, optionally, id, alt, nb.  (Omit id for 'outgroup'. Use 'alt' for less formal
#     labeling in cartoon version; it defaults to 'where'. Use 'nb' to add abcde annotations for legend.)
# This hand-made version is now subsumed by make.tree; retained for comparison
tree.by.hand <-
  list(
    sub1 = list(
      sub1 = list(
        sub1 = list(id=3367, length=8813, where='Venice, Italy', alt='Venice'),
        sub2 = list(id=1013, length=9652, where='Wales, UK'),
        length=9365),
      sub2 =  list(
        sub1 = list(
          sub1 = list(
            sub1 = list(id=1007, length=30, nb='e', where='Virginia, USA'),
            sub2 = list(id=1012, length=61, nb='d', where='Perth, W. Australia', alt='Perth'),
            length=19),
          sub2 = list(
            sub1 = list(id=1015, length=207,nb='c', where='Washington, USA', alt='Puget Sound'),
            sub2 = list(id=1335, length=41, nb='b', where='New York, USA',   alt='NY'),
            length=18),
          length=1005),
        sub2 = list(id=1014, length=61, nb='a', where='N. Pacific Gyre'),
        length=3912),
      length=7054),
    sub2 = list(length=0, where='outgroup')
  )

# historical, format example, and debug help:
oldwick.medium <- '(((CCMP3367_Italy:8813,CCMP1013_Wales:9652):9365,(((e_CCMP1007_Virginia:30,d_CCMP1012_Australia:61):19,(c_CCMP
# with simpler labeling for cartoon
simple.oldwick.medium <- '(((Italy:8813,Wales:9652):9365,(((Virginia:30,Australia:61):19,(Puget:207,NY:41):18):1005,Gyre:61):3912
cat.hardwrap(oldwick.medium)

# (((CCMP3367_Italy:8813,CCMP1013_Wales:9652):9365,(((e_CCMP1007_Virginia:30,d_CCM
# P1012_Australia:61):19,(c_CCMP1015_Puget_Sound:207,b_CCMP1335_NY:41):18):1005,a_
# CCMP1014_NPG:61):3912):7054,outgroup:0);

cat.hardwrap(simple.oldwick.medium)

# (((Italy:8813,Wales:9652):9365,(((Virginia:30,Australia:61):19,(Puget:207,NY:41)
# :18):1005,Gyre:61):3912):7054,outgroup:0);
```

Two other versions of the tree, for possible use in figs in the main paper.

Figure 8: [** as of 10/4/2015, this fig and next have stray stars on virginia, wales labels; probably due to hacking with commas in newick; not worth fixing unless we resurrect these trees for some purpose, but if so, see use of newick.name.undo in show.tree as probable fix. **]

```
tree.scale <- ifelse(which.snp.tables(string.val=F)[1]=='Chr1', 1, 10)
tree.x.lim <- 3e4 * tree.scale
the.simple.tree <- read.tree(text=simple.newick.medium)
plot(the.simple.tree, x.lim = tree.x.lim)
axis(1)
```

Figure 9:

```
plot(the.simple.tree, x.lim = tree.x.lim)
axis(1,(0:4)*7000*tree.scale,(0:4)*7000*tree.scale)
```

At some much earlier point, Tony ran the whole-genome version of the then-current code above, and manually entered tree branch lengths/legend for the resuting tree, shown in Fig 10. Code above can now automatically generate such a tree, but retain the following for comparison. The basic story seems clear—same topology and branch lengths scaled by about 10x, which is completely reasonable given that Chr1 is about 10% of the genome. Note that this tree is not being recalculated; it is defined by constants in the following code chunk.

```
fullgenome.newick.medium <- '(((3367_Italy:86155,1013_Wales:95697):89598,(((e_1007_VA:330,d_1012_Australia:632):1296,(c_1015_WA:2
cat.hardwrap(fullgenome.newick.medium)

# (((3367_Italy:86155,1013_Wales:95697):89598,(((e_1007_VA:330,d_1012_Australia:63
# 2):1296,(c_1015_WA:2113,b_1335_NY:658):480):10059,a_1014_NPG:568):39517):69526,o
# utgroup:0);
```

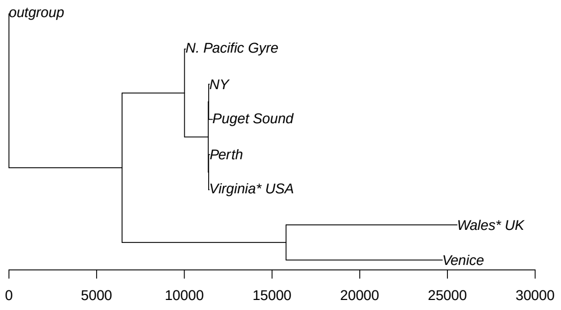


Figure 8: Tree based on qfiltered reads and medium SNP filters. “Lengths” are numbers of shared/private SNPs on Chr1. (no edge labels, nolegend)

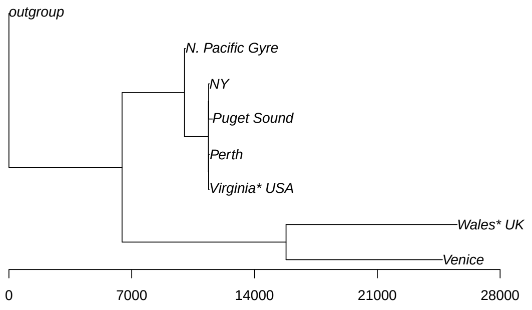


Figure 9: Tree based on qfiltered reads and medium SNP filters. “Lengths” are numbers of shared/private SNPs on Chr1. (no edge labels, no legend, short scale bar)

```r
legend.text <- c('a: only in 1014  ',
                 'b: only in 1335  ',
                 'c: only in 1015  ',
                 'd: only in 1012  ',
                 'e: only in 1007  ',
                 '*: shared by bcde',
                 '   shared by b/c ',
                 '   shared by d/e '
)
fullgenome.tree.x.lim <- 300000
fullgenome.counts <- c( 568, 658, 2113, 632, 330, 10059, 480, 1296 )
fullgenome.legend.text <- paste(legend.text,format(fullgenome.counts,width=5),sep=' - ')
fullgenome.tree.labels <- list( ## x,y,text
     41000,3.63,'69526\nshared by 7',
     90000,5.75,'39517\nby 5 (**)',
    115000,1.5, '89598\nshared by 2',
    210000,2.0, '95697 only\nin Wales',
    210000,1.0, '86155 only\nin Italy',
    113500,4.6, '*')
```

Figure 10:

```r
library(ape)
the.fullgenome.tree <- read.tree(text=fullgenome.newick.medium)
plot(the.fullgenome.tree, x.lim = fullgenome.tree.x.lim)
axis(1) # ; axis(2) useful only for placing labels
opar <- par(family='mono',cex=.8)
legend('topright', legend=fullgenome.legend.text)
par(opar)
for(i in seq(1,length(fullgenome.tree.labels)-2,by=3)){
  text(fullgenome.tree.labels[[i]], fullgenome.tree.labels[[i+1]], fullgenome.tree.labels[[i+2]])
}
```

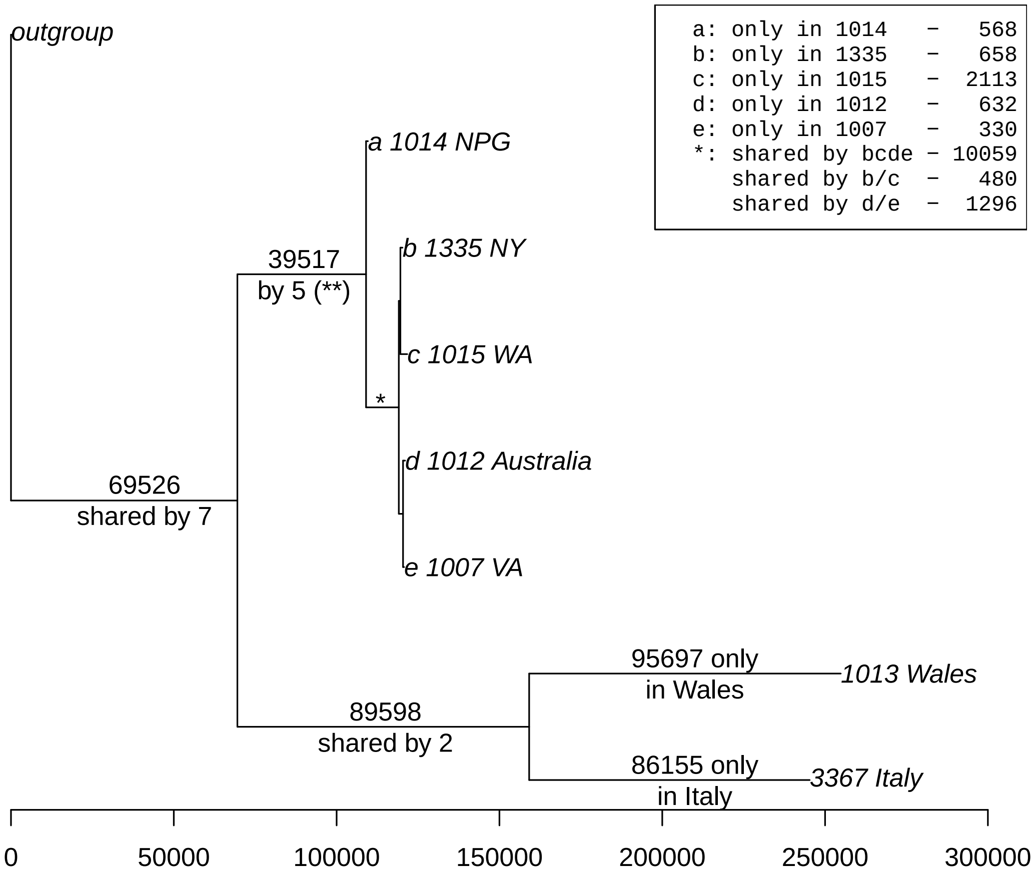


Figure 10: Tree based on unqfiltered reads and medium SNP filters. "Lengths" are numbers of shared/private SNPs genome-wide. (By-hand legacy version)