Exploration of Shared SNPs in Thaps
trunc-qfiltered

June 29, 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  2017-06-26 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('f-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
#     TRUE    FALSE     TRUE    FALSE
#
# $pri
# [1] 3 4 1 2
#
# $clear.cache
# [1] TRUE
#
# $nboot
# [1] 101
#
# $trunc.tables
# [1] TRUE
```

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 'cache80604' 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')}

# Loading full tables from ../../../data/ungit-data/full.tables.01.26.14.rda ...Loaded.
# ../00common/mycache/snp.tables.chr1.unqfiltered.rda saved.

if(tb[2]){tset[[2]] <- load.snp.tables(use.chr1.tables = TRUE , data.name='full.tables.01.26.14')}
if(tb[3]){tset[[3]] <- load.snp.tables(use.chr1.tables = FALSE, data.name='full.tables.02.25.15')}

# Loading full tables from ../../../data/ungit-data/full.tables.02.25.15.rda ...Loaded.
# ../00common/mycache/snp.tables.chr1.qfiltered.rda saved.
# Bandaiding qfiltered tables...

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

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')
}
```

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
# 31301782 31301782 31301782 31301782 31301782 31301782 31301782
# OK, all strains have same number of rows.
#     1007     1012     1013     1014     1015     3367     1335
# 31301782 31301782 31301782 31301782 31301782 31301782 31301782
# 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: ( trunc-unfiltered, NULL, trunc-qfiltered, NULL ) SNP tables.

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

# Main shared SNP analysis focuses on trunc-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{trunc-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': 31301782 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/ 66 levels "BD1_7","BD10_65",..: 39 39 39 39 39 39 39 39 39 39 ...
#  $ 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
# 196296 197799 474613 128683  70687  15186
```

There are nusnps=474613 positions called as SNPs in one or more strains (but only nisnps=15186 that are shared among all 7). Note that the 4-way union is only modestly larger (1.5254229 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 89184 vs 128683 in the 4-way intersection, and they are highly concordant:

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

# [1] 0.7925973
```

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         97.71716831 97.35791400 95.45329400 97.29079003 97.18569697 95.89943474 96.48734376
# % ==1          1.48140448  1.75279158  3.01610304  2.08080805  1.88427930  2.58814338  2.54747477
# % ==2          0.11277633  0.12101867  0.22209918  0.18711714  0.13599226  0.21703876  0.19842640
# % ==3          0.05885927  0.05629072  0.09496264  0.10621121  0.06021063  0.09526295  0.06375675
# % >=4          0.62979162  0.71198502  1.21354113  0.33507358  0.73382084  1.20012017  0.70299831
# % >=4, nonSNP  0.08015518  0.12493857  0.25313575  0.04763946  0.13208513  0.28212771  0.13240460
```

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 [1:474613, 1:7] 0 0 0 0 0 0 0 0 1 0 ...
#  - attr(*, "dimnames")=List of 2
#   ..$ : chr [1:474613] "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 <- [2757 chars quoted with ''']

filtered.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(filtered.snps$Data$consistent.snps[[stringency]] &
                  filtered.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 <- filtered.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(filtered.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)
rda.filtered <- paste('../00common/mycache/filtered.snps', which.snp.tables(), 'rda', sep='.')
if(file.exists(rda.filtered)){
  cat('Pre-existing file', rda.filtered, 'unchanged.\n')
} else {
  cat('Saving', rda.filtered, '...')
  save(filtered.snps, file=rda.filtered, compress=TRUE)
  cat('Saved.\n')
}

# Pre-existing file ../00common/mycache/filtered.snps.trunc-qfiltered.rda unchanged.
```

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

```
cat(filtered.snps$Description)

# Contents of this .rda file:
#
#   * Description: this text
#
#   * Data -- 5 items defining filtered 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 filtered 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 filtered 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 filtered 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 [1:474613] TRUE FALSE TRUE TRUE TRUE TRUE ...
#  - attr(*, "names")= chr [1:474613] "Chr1:333" "Chr1:417" "Chr1:435" "Chr1:438" ...
```

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

# [1] 447177 469906 471171 474613

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

# [1] 27436  4707  3442     0

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

# [1] 5.7807098 0.9917554 0.7252224 0.0000000
```

I.e., of the 474613 positions in which a SNP is called, 447177 are consistent by my loose definition, and 471171 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
#  16069  14356 120037  11436   6862 142515   1854
```

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          16069   14356  120037   11436    6862 142515    1854
# hemi         1834990 1940024 1527725 1472095 1134652 480817 1596965
# homnr.unhemi    9650    7347  111674   10091    5113 140185    1829
```

```
# 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
#  517  592  748  362  617  793 1854

# 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] 31723

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

# [1] 188637
```

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

# [1] 73915

# 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 <- filtered.snps$Code$get.snps(i, stringency)
    snp.counts[i,stringency] <- sum(f.snps.i)
    for(j in i:7){
      f.snps.j <- filtered.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 184621 190979 363297 196256 197762 354191 199128
# 1012     NA 187793 364751 198002 198919 355526 200266
# 1013     NA     NA 296795 356666 366717 391621 364222
# 1014     NA     NA     NA 165741 196847 347035 195294
# 1015     NA     NA     NA     NA 191668 357845 198939
# 3367     NA     NA     NA     NA     NA 283086 355107
# 1335     NA     NA     NA     NA     NA     NA 187044
# Intersect Counts:
#        1007   1012   1013   1014   1015   3367   1335
# 1007 184621 181435 118119 154106 178527 113516 172537
# 1012     NA 187793 119837 155532 180542 115353 174571
# 1013     NA     NA 296795 105870 121746 188260 119617
# 1014     NA     NA     NA 165741 160562 101792 157491
# 1015     NA     NA     NA     NA 191668 116909 179773
# 3367     NA     NA     NA     NA     NA 283086 115023
# 1335     NA     NA     NA     NA     NA     NA 187044
# Intersect as percent of union:
#      1007 1012  1013  1014  1015  3367  1335
# 1007  100   95  32.5  78.5  90.3  32.0  86.6
# 1012   NA  100  32.9  78.6  90.8  32.4  87.2
# 1013   NA   NA 100.0  29.7  33.2  48.1  32.8
# 1014   NA   NA    NA 100.0  81.6  29.3  80.6
# 1015   NA   NA    NA    NA 100.0  32.7  90.4
# 3367   NA   NA    NA    NA    NA 100.0  32.4
# 1335   NA   NA    NA    NA    NA    NA 100.0
#
# Stringency 2  :
# --------------
# Union Counts:
#        1007   1012   1013   1014   1015   3367   1335
# 1007 181682 189049 374070 191223 196135 364611 195371
# 1012     NA 186199 376272 194060 197053 366680 196729
# 1013     NA     NA 304444 356961 378452 408556 373836
# 1014     NA     NA     NA 137771 193813 346571 187896
# 1015     NA     NA     NA     NA 190384 369360 195610
# 3367     NA     NA     NA     NA     NA 290844 364209
# 1335     NA     NA     NA     NA     NA     NA 180709
# Intersect Counts:
#        1007   1012   1013   1014   1015   3367   1335
# 1007 181682 178832 112056 128230 175931 107915 167020
# 1012     NA 186199 114371 129910 179530 110363 170179
# 1013     NA     NA 304444  85254 116376 186732 111317
# 1014     NA     NA     NA 137771 134342  82044 130584
# 1015     NA     NA     NA     NA 190384 111868 175483
# 3367     NA     NA     NA     NA     NA 290844 107344
# 1335     NA     NA     NA     NA     NA     NA 180709
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  94.6  30.0  67.1  89.7  29.6  85.5
# 1012   NA 100.0  30.4  66.9  91.1  30.1  86.5
# 1013   NA    NA 100.0  23.9  30.8  45.7  29.8
# 1014   NA    NA    NA 100.0  69.3  23.7  69.5
# 1015   NA    NA    NA    NA 100.0  30.3  89.7
# 3367   NA    NA    NA    NA    NA 100.0  29.5
# 1335   NA    NA    NA    NA    NA    NA 100.0
#
# Stringency 3  :
# --------------
# Union Counts:
#        1007   1012   1013   1014   1015   3367   1335
# 1007 169346 183708 363625 176590 190884 354428 188236
# 1012     NA 179828 368821 185220 192949 359446 191771
# 1013     NA     NA 296287 330697 371321 403242 364437
# 1014     NA     NA     NA  88328 187158 319824 176609
# 1015     NA     NA     NA     NA 184704 362666 191220
# 3367     NA     NA     NA     NA     NA 283373 355133
# 1335     NA     NA     NA     NA     NA     NA 171199
# Intersect Counts:
#        1007   1012   1013  1014   1015   3367   1335
# 1007 169346 165466 102008 81084 163166  98291 152309
# 1012     NA 179828 107294 82936 171583 103755 159256
# 1013     NA     NA 296287 53918 109670 176418 103049
# 1014     NA     NA     NA 88328  85874  51877  82918
# 1015     NA     NA     NA    NA 184704 105411 164683
# 3367     NA     NA     NA    NA     NA 283373  99439
# 1335     NA     NA     NA    NA     NA     NA 171199
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  90.1  28.1  45.9  85.5  27.7  80.9
# 1012   NA 100.0  29.1  44.8  88.9  28.9  83.0
# 1013   NA    NA 100.0  16.3  29.5  43.7  28.3
# 1014   NA    NA    NA 100.0  45.9  16.2  47.0
# 1015   NA    NA    NA    NA 100.0  29.1  86.1
# 3367   NA    NA    NA    NA    NA 100.0  28.0
# 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 161103 176738 343873 171675 185741 336599 180313
# 1012     NA 166089 346766 176177 186459 339458 182312
# 1013     NA     NA 247737 302322 352586 386037 339669
# 1014     NA     NA     NA  89184 179976 295574 162912
# 1015     NA     NA     NA     NA 174701 345396 184068
# 3367     NA     NA     NA     NA     NA 240413 331982
# 1335     NA     NA     NA     NA     NA     NA 153901
# Intersect Counts:
#        1007   1012   1013  1014   1015   3367   1335
# 1007 161103 150454  64967 78612 150063  64917 134691
# 1012     NA 166089  67060 79096 154331  67044 137678
# 1013     NA     NA 247737 34599  69852 102113  61969
# 1014     NA     NA     NA 89184  83909  34023  80173
# 1015     NA     NA     NA    NA 174701  69718 144534
# 3367     NA     NA     NA    NA     NA 240413  62332
# 1335     NA     NA     NA    NA     NA     NA 153901
# Intersect as percent of union:
#      1007  1012  1013  1014  1015  3367  1335
# 1007  100  85.1  18.9  45.8  80.8  19.3  74.7
# 1012   NA 100.0  19.3  44.9  82.8  19.8  75.5
# 1013   NA    NA 100.0  11.4  19.8  26.5  18.2
# 1014   NA    NA    NA 100.0  46.6  11.5  49.2
# 1015   NA    NA    NA    NA 100.0  20.2  78.5
# 3367   NA    NA    NA    NA    NA 100.0  18.8
# 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 184621 181682 169346 161103   NA  114.6  112.8  105.1
# 1012 187793 186199 179828 166089   NA  113.1  112.1  108.3
# 1013 296795 304444 296287 247737   NA  119.8  122.9  119.6
# 1014 165741 137771  88328  89184   NA  185.8  154.5   99.0
# 1015 191668 190384 184704 174701   NA  109.7  109.0  105.7
# 3367 283086 290844 283373 240413   NA  117.7  121.0  117.9
# 1335 187044 180709 171199 153901   NA  121.5  117.4  111.2

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

#       [,1]  [,2]  [,3]  [,4]
# 1007  93.5  91.9  89.9  83.6
# 1012  93.0  91.4  88.6  82.9
# 1013  40.3  36.6  34.8  25.0
# 1014  95.0  94.8  93.9  89.9
# 1015  93.8  92.2  89.2  82.7
# 3367  40.6  36.9  35.1  25.9
# 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  92.2  92.4  89.0  87.5
# 1012  93.3  94.2  93.0  89.5
# 1013  64.0  61.6  60.2  40.3
# 1014  84.2  72.3  48.4  52.1
# 1015  96.1  97.1  96.2  93.9
# 3367  61.5  59.4  58.1  40.5
# 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.full.unfiltered    0    0    0    0    0    0    0
# snp.tables.full.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
# trunc-unfiltered 37.0555484 70.8060724 69.6610432 33.1009373 61.5365159 64.0284488 107.7425968
# trunc-qfiltered  28.2750286 51.3249686 45.4036337 13.7261052 48.7880005 44.8042054  81.8823765
# Ratio             0.7630444  0.7248668  0.6517794  0.4146742  0.7928301  0.6997547   0.7599815
```

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   107.7  81.9 153901 180709     100.0     100.0
# 1007 CCMP1007        Virginia 1964    37.1  28.3 161103 181682      87.5      92.4
# 1012 CCMP1012    W. Australia 1965    70.8  51.3 166089 186199      89.5      94.2
# 1015 CCMP1015     Puget Sound 1985    61.5  48.8 174701 190384      93.9      97.1
# 1013 CCMP1013           Wales 1973    69.7  45.4 247737 304444      40.3      61.6
# 3367 CCMP3367           Italy 2007    64.0  44.8 240413 290844      40.5      59.4
# 1014 CCMP1014 N. Pacific Gyre 1971    33.1  13.7  89184 137771      52.1      72.3
```

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]]$snp)
# 'unfil.' => unfiltered for consistency; see below.
unfil.fiveway.count   <- sum( snp.tables[[4]]$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         70687        79.25973 4.142632e-19
```

Namely, 89184 positions are called as SNPs in CCMP1014, of which 70687 or 79.2597327% are also called as SNPs in *all four* other L-clade isolates. 79.2597327% of 24 is 19.0223358, and the probability of seeing 19 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: $4.1426317 \times 10^{-19}$ 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                    70687        79.25973 4.142632e-19
# consistency.filtered          69941        78.42326 1.976512e-17
```

In particular, it removes 0.8% of five-way consistent positions (only 746 of 70687 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 78.42% 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] 1.04e-16 9.04e-15 3.78e-13 1.01e-11 1.92e-10 2.80e-09 3.22e-08 3.01e-07 2.32e-06 1.50e-05
# [11] 8.18e-05 3.78e-04 1.49e-03 5.00e-03 1.43e-02 3.46e-02 7.07e-02 1.21e-01 1.71e-01 1.96e-01
# [21] 1.78e-01 1.23e-01 6.12e-02 1.93e-02 2.93e-03
```

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] 6.808209e-10
```

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  1.037e-16 1.037e-16 1.000e+00
# 2  9.043e-15 9.147e-15 7.875e-01
# 3  3.780e-13 3.871e-13 4.476e-01
# 4  1.008e-11 1.046e-11 1.869e-01
# 5  1.922e-10 2.027e-10 5.950e-02
# 6  2.795e-09 2.998e-09 1.490e-02
# 7  3.217e-08 3.517e-08 3.010e-03
# 8  3.007e-07 3.358e-07 4.994e-04
# 9  2.322e-06 2.658e-06 6.899e-05
# 10 1.500e-05 1.766e-05 8.015e-06
# 11 8.181e-05 9.947e-05 7.887e-07
# 12 3.784e-04 4.779e-04 6.603e-08
# 13 1.490e-03 1.968e-03 4.716e-09
# 14 4.999e-03 6.967e-03 2.875e-10
# 15 1.428e-02 2.124e-02 1.493e-11
# 16 3.459e-02 5.584e-02 6.590e-13
# 17 7.072e-02 1.266e-01 2.456e-14
# 18 1.210e-01 2.475e-01 7.662e-16
# 19 1.710e-01 4.185e-01 1.977e-17
# 20 1.963e-01 6.148e-01 4.143e-19
# 21 1.783e-01 7.931e-01 6.877e-21
# 22 1.235e-01 9.165e-01 8.701e-23
# 23 6.119e-02 9.777e-01 7.884e-25
# 24 1.934e-02 9.971e-01 4.556e-27
# 25 2.929e-03 1.000e+00 1.262e-29
```

E.g., row 9 in that table says that the concordance rate (78%) 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 2.658e-06), 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                     70687          79.25973 4.142632e-19
# consistency.filtered           69941          78.42326 1.976512e-17
# gyre.vs.italy                  34023          38.14922 9.242052e-01
# new.york.vs.italy              62332          40.50136 9.242052e-01
# ny.vs.it.plus.wales            35796          23.25911 7.533516e-01
# it.vs.wales                   102113          42.47399 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, $0 \le i \le 127$ 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] 133 139 132 417

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] 133 139 132 417

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"     "Chr2:713075"      "Chr2:1464209"     "Chr2:2406031"
#   [5] "Chr2:2480466"     "Chr2:2480532"     "Chr2:2480838"     "Chr2:2483322"
#   [9] "Chr2:2488863"     "Chr2:2489189"     "Chr2:2490933"     "Chr2:2492886"
#  [13] "Chr2:2492887"     "Chr2:2497794"     "Chr2:2500122"     "Chr2:2503000"
#  [17] "Chr2:2507585"     "Chr2:2507680"     "Chr2:2510117"     "Chr2:2513923"
#  [21] "Chr2:2515103"     "Chr2:2516669"     "Chr2:2516751"     "Chr2:2518558"
#  [25] "Chr2:2518653"     "Chr2:2518980"     "Chr2:2519285"     "Chr2:2519288"
#  [29] "Chr2:2519718"     "Chr2:2520984"     "Chr2:2521271"     "Chr2:2522648"
#  [33] "Chr2:2524223"     "Chr2:2524439"     "Chr2:2525160"     "Chr2:2525463"
#  [37] "Chr2:2527281"     "Chr2:2527916"     "Chr2:2528472"     "Chr2:2528769"
#  [41] "Chr2:2529076"     "Chr2:2529140"     "Chr2:2529684"     "Chr2:2530064"
#  [45] "Chr2:2530216"     "Chr2:2530239"     "Chr2:2530294"     "Chr2:2530768"
#  [49] "Chr2:2530896"     "Chr2:2531114"     "Chr2:2531498"     "Chr2:2531567"
#  [53] "Chr2:2532173"     "Chr2:2532365"     "Chr2:2533028"     "Chr2:2533171"
#  [57] "Chr2:2533440"     "Chr2:2534441"     "Chr2:2535121"     "Chr2:2535122"
#  [61] "Chr2:2535314"     "Chr2:2535493"     "Chr2:2535503"     "Chr2:2535509"
#  [65] "Chr2:2535862"     "Chr2:2536242"     "Chr2:2537201"     "Chr2:2537864"
#  [69] "Chr2:2537917"     "Chr2:2538072"     "Chr2:2538498"     "Chr2:2539318"
#  [73] "Chr2:2543595"     "Chr2:2545615"     "Chr2:2545798"     "Chr2:2546865"
#  [77] "Chr2:2546991"     "Chr2:2547055"     "Chr2:2547086"     "Chr2:2547120"
#  [81] "Chr2:2547155"     "Chr2:2547212"     "Chr2:2547248"     "Chr2:2547318"
#  [85] "Chr2:2547554"     "Chr2:2547938"     "Chr2:2547944"     "Chr2:2548131"
#  [89] "Chr2:2549281"     "Chr2:2551574"     "Chr2:2551930"     "Chr2:2554708"
#  [93] "Chr2:2554860"     "Chr2:2555005"     "Chr2:2555203"     "Chr2:2555820"
#  [97] "Chr3:192441"      "Chr3:496665"      "Chr4:1086589"     "Chr4:1393682"
# [101] "Chr4:1747983"     "Chr4:2314475"     "Chr5:7509"        "Chr5:141375"
# [105] "Chr5:1071721"     "Chr6:1330532"     "Chr7:399475"      "Chr7:1736991"
# [109] "Chr7:1813303"     "Chr8:556556"      "Chr10:54351"      "Chr10:95217"
# [113] "Chr10:947088"     "Chr11a:344258"    "Chr11b:75778"     "Chr12:214112"
# [117] "Chr12:458461"     "Chr12:507608"     "Chr13:96361"      "Chr13:375598"
# [121] "Chr14:284131"     "Chr15:417704"     "Chr16a:39914"     "Chr16a:177501"
# [125] "Chr16a:206719"    "Chr16a:394030"    "Chr17:461465"     "Chr19a_19:300076"
# [129] "Chr19a_19:303090" "Chr19c_29:64170"  "Chr19c_29:64811"  "Chr19c_29:65720"
# [133] "Chr20:230994"     "Chr20:486431"     "Chr20:519835"     "Chr22:380816"
# [137] "Chr23:190382"     "Chr23:274291"     "Chr24:114599"

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   62676  713075 2406031 2480466 2480838 2481998 2483322
# [12] 2488863 2489189 2490933 2492887 2497794 2500122 2503000 2507585 2507680

c2

#  [1] 1799155  713075 1464209 2406031 2480466 2480532 2480838 2483322 2488863 2489189 2490933
# [12] 2492886 2492887 2497794 2500122 2503000 2507585 2507680 2510117 2513923

c3

#  [1]  371484  518347 1210354 2209068 2264683 2898352 1276745 1464904 1464905 1766966 2347253
# [12] 2406031 2480532 2480838 2483322 2488863 2489189 2490933 2497794 2507585

c4

#  [1]  518347  691730  767408 1049906 1390437 2072951 2254059 2254789 2264683 2823796 2898352
# [12] 2998868   77394   77407  155680  761325  968120 1182096 1222176 1264023

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
# 9   Chr1  713075   T
# 10                      1007   0  0   0  37   0  TRUE FALSE
# 11                      1012   0  0   0  90   0  TRUE FALSE
# 12                      1013   0  0   0  65   0  TRUE FALSE
# 13                      1014   0  0   0  32   0  TRUE FALSE
# 14                      1015   0  0   0  84   0  TRUE FALSE
# 15                      3367   0  0   0  53   0  TRUE FALSE
# 16                      1335   0  0   0 109   0  TRUE FALSE
# 17  Chr1 1464209   T
# 18                      1007   0  0   0  22   0 FALSE FALSE
# 19                      1012   0  0   0  38   0 FALSE FALSE
# 20                      1013   0  0   0  22   0 FALSE FALSE
# 21                      1014   0  0   0  12   0 FALSE FALSE
# 22                      1015   0  0   0  30   0 FALSE FALSE
# 23                      3367   0  0   0  39   0 FALSE FALSE
# 24                      1335   0  0   0  81   0 FALSE FALSE
# 25  Chr1 2406031   C
# 26                      1007   0  0  18   0   0  TRUE FALSE
# 27                      1012   0  0  23   0   0  TRUE FALSE
# 28                      1013   0  0  46   0   0  TRUE FALSE
# 29                      1014   0  0  13   0   0  TRUE FALSE
# 30                      1015   0  0  34   0   0  TRUE FALSE
# 31                      3367   0  0  29   0   0  TRUE FALSE
# 32                      1335   0  0  68   0   0  TRUE FALSE
# 33  Chr1 2480466   A
# 34                      1007  26  0   0   0   0  TRUE FALSE
# 35                      1012  42  0   0   0   0  TRUE FALSE
# 36                      1013  39  0   0   0   0  TRUE FALSE
# 37                      1014   9  0   0   0   0  TRUE FALSE
# 38                      1015  49  0   0   0   0  TRUE FALSE
# 39                      3367  32  0   0   0   0  TRUE FALSE
# 40                      1335  77  0   0   0   0  TRUE FALSE
# 41  Chr1 2480532   G
# 42                      1007   0 25   0   0   0  TRUE FALSE
# 43                      1012   0 27   0   0   0  TRUE FALSE
# 44                      1013   0 43   0   0   0  TRUE FALSE
# 45                      1014   0  1   0   0   0  TRUE FALSE
# 46                      1015   0 23   0   0   0  TRUE FALSE
# 47                      3367   0 23   0   0   0  TRUE FALSE
# 48                      1335   0 71   0   0   0  TRUE FALSE
# 49  Chr1 2480838   T
# 50                      1007   0  0   0   8   0  TRUE FALSE
# 51                      1012   0  0   0  12   0  TRUE FALSE
# 52                      1013   0  0   0  24   0  TRUE FALSE
# 53                      1014   0  0   0   6   0  TRUE FALSE
# 54                      1015   0  0   0  15   0  TRUE FALSE
# 55                      3367   0  0   0   9   0  TRUE FALSE
# 56                      1335   0  0   0  81   0  TRUE FALSE
# 57  Chr1 2483322   A
# 58                      1007  22  0   0   0   0  TRUE FALSE
# 59                      1012  23  0   0   0   0  TRUE FALSE
# 60                      1013  52  0   0   0   0  TRUE FALSE
# 61                      1014  24  0   0   0   0  TRUE FALSE
# 62                      1015  55  0   0   0   0  TRUE FALSE
# 63                      3367  37  0   0   0   0  TRUE FALSE
# 64                      1335  82  0   0   0   0  TRUE FALSE
# 65  Chr1 2488863   C
# 66                      1007   0  0  26   0   0 FALSE FALSE
# 67                      1012   0  0  34   0   0 FALSE FALSE
# 68                      1013   0  0  27   0   0 FALSE FALSE
# 69                      1014   0  0  11   0   0 FALSE FALSE
# 70                      1015   0  0  34   0   0 FALSE FALSE
# 71                      3367   0  0  43   0   0 FALSE FALSE
# 72                     1335   0  0  71   0   0 FALSE FALSE
# 73  Chr1 2489189   C
# 74                     1007   0  0  32   0   0 FALSE FALSE
# 75                     1012   0  0  63   0   0 FALSE FALSE
# 76                     1013   0  0  44   0   0 FALSE FALSE
# 77                     1014   0  0  26   0   0 FALSE FALSE
# 78                     1015   0  0  59   0   0 FALSE FALSE
# 79                     3367   0  0  24   0   0 FALSE FALSE
# 80                     1335   0  0 110   0   0 FALSE FALSE
# 81  Chr1 2490933   G
# 82                     1007   0 25   0   0   0 FALSE FALSE
# 83                     1012   0 57   0   0   0 FALSE FALSE
# 84                     1013   0 40   0   0   0 FALSE FALSE
# 85                     1014   0  9   0   0   0 FALSE FALSE
# 86                     1015   0 36   0   0   0 FALSE FALSE
# 87                     3367   0 37   0   1   0 FALSE FALSE
# 88                     1335   0 57   0   0   0 FALSE FALSE
# 89  Chr1 2492886   T
# 90                     1007   0  0   0  27   0 FALSE FALSE
# 91                     1012   0  0   0  61   0 FALSE FALSE
# 92                     1013   0  0   0  41   0 FALSE FALSE
# 93                     1014   0  0   0  18   0 FALSE FALSE
# 94                     1015   0  0   0  53   0 FALSE FALSE
# 95                     3367   0  0   0  48   0 FALSE FALSE
# 96                     1335   0  0   0  80   0 FALSE FALSE
# 97  Chr1 2492887   G
# 98                     1007   0 22   0   0   0 FALSE FALSE
# 99                     1012   0 61   0   0   0 FALSE FALSE
# 100                    1013   0 35   0   0   0 FALSE FALSE
# 101                    1014   0 17   0   0   0 FALSE FALSE
# 102                    1015   0 55   0   0   0 FALSE FALSE
# 103                    3367   0 50   0   0   0 FALSE FALSE
# 104                    1335   0 85   0   0   0 FALSE FALSE
# 105 Chr1 2497794   T
# 106                    1007   0  0   0  35   0  TRUE FALSE
# 107                    1012   0  0   0  60   0  TRUE FALSE
# 108                    1013   0  0   0  58   0  TRUE FALSE
# 109                    1014   0  0   0  12   0  TRUE FALSE
# 110                    1015   0  0   0  64   0  TRUE FALSE
# 111                    3367   0  0   0  43   0  TRUE FALSE
# 112                    1335   0  0   0 107   0  TRUE FALSE
# 113 Chr1 2500122   A
# 114                    1007  18  0   0   0   0 FALSE FALSE
# 115                    1012  47  0   0   0   0 FALSE FALSE
# 116                    1013  34  0   0   0   0 FALSE FALSE
# 117                    1014   6  0   0   0   0 FALSE FALSE
# 118                    1015  35  0   0   0   0 FALSE FALSE
# 119                    3367  27  0   0   0   0 FALSE FALSE
# 120                    1335  51  0   0   0   0 FALSE FALSE
# 121 Chr1 2503000   T
# 122                    1007   0  0   0  29   0 FALSE FALSE
# 123                    1012   0  0   0  35   0 FALSE FALSE
# 124                    1013   0  0   0  57   0 FALSE FALSE
# 125                    1014   0  0   0  10   0 FALSE FALSE
# 126                    1015   0  0   0  34   0 FALSE FALSE
# 127                    3367   0  0   0  41   0 FALSE FALSE
# 128                    1335   0  0   0  28   0 FALSE FALSE
# 129 Chr1 2507585   A
# 130                    1007  34  0   0   0   0  TRUE FALSE
# 131                    1012  55  0   0   0   0  TRUE FALSE
# 132                    1013  32  0   0   0   0  TRUE FALSE
# 133                    1014  13  0   0   0   0  TRUE FALSE
# 134                    1015  41  0   0   0   0  TRUE FALSE
# 135                    3367  61  0   0   0   0  TRUE FALSE
# 136                    1335 104  0   0   0   0  TRUE FALSE
# 137 Chr1 2507680   A
# 138                    1007  26  0   0   0   0 FALSE FALSE
# 139                        1012  46  0   0   0   0 FALSE FALSE
# 140                        1013  32  0   0   0   0 FALSE FALSE
# 141                        1014  15  0   0   0   0 FALSE FALSE
# 142                        1015  54  0   0   0   0 FALSE FALSE
# 143                        3367  51  0   0   0   0 FALSE FALSE
# 144                        1335  78  0   0   0   0 FALSE FALSE
# 145 Chr1 2510117   C
# 146                        1007   0  0  19   0   0  TRUE FALSE
# 147                        1012   0  0  56   1   0  TRUE FALSE
# 148                        1013   0  0  42   0   0  TRUE FALSE
# 149                        1014   0  0  13   0   0  TRUE FALSE
# 150                        1015   0  0  39   0   0  TRUE FALSE
# 151                        3367   0  0  36   0   0  TRUE FALSE
# 152                        1335   0  0  92   0   0  TRUE FALSE
# 153 Chr1 2513923   A
# 154                        1007  39  0   0   0   0 FALSE FALSE
# 155                        1012  57  0   0   0   0 FALSE FALSE
# 156                        1013  23  0   0   0   0 FALSE FALSE
# 157                        1014   4  0   0   0   0 FALSE FALSE
# 158                        1015  39  0   0   0   0 FALSE FALSE
# 159                        3367  53  0   0   0   0 FALSE FALSE
# 160                        1335  53  0   0   0   0 FALSE 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] 13630

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                                       111    578   4755      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  77690  62182  38744  15186
```

I.e., of the 469906 consistent positions, 62182 or 13.2% 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    449    632   1129   2260
# 3     002     1                             X       73721  85117  87494 105614
# 5     004     1                        X             1720   2156   2729   4608
# 9     010     1                   X                   383    525    485   1231
# 17    020     1              X                      82364  94364  96464 113191
# 33    040     1         X                             502    655   1102   2450
# 65    100     1    X                                  231    339    496   2005
# Total                                              159370 183788 189899 231359
```

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 447177 469906 471171 474613
# singlets   159370 183788 189899 231359
# shared     287807 286118 281272 243254
```

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    994    298    532    587
# 5     005     2                        X         X    287    523   1088   1407
# 7     006     2                        X    X         624    138    282    590
# 10    011     2                   X              X    515    486    317    827
# 11    012     2                   X         X         565     49     32     93
# 13    014     2                   X    X              133    139    132    417
# 18    021     2              X                   X    998    167    337    402
# 19    022     2              X              X       82160  87499  83482  58009
# 21    024     2              X         X              686    195    410    625
# 25    030     2              X    X                   609     69     47     93
# 34    041     2         X                        X     42     92    313    368
# 35    042     2         X                   X         503    119    254    394
# 37    044     2         X              X               69    279   1001   1809
# 41    050     2         X         X                    13     24     53    105
# 49    060     2         X    X                       627    116    237    388
# 66    101     2    X                             X     29     47     73    314
# 67    102     2    X                        X         351     67     96    351
# 69    104     2    X                   X               39    122    329   1196
# 73    110     2    X              X                    12     11     29    150
# 81    120     2    X         X                       432     76     98    309
# 97    140     2    X    X                            955   1144   1235   2144
# Total                                               90643  91660  90377  70578
```

I.e., of the 91660 paired SNPs, 87499 or 95.5% 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    104    197    371    557
# 12    013     3                   X         X    X    257    226    146    338
# 14    015     3                   X    X         X   1041    984    660   1389
# 15    016     3                   X    X    X          46     60     49    152
# 20    023     3              X              X    X   1274    558   1020    533
# 22    025     3              X         X         X    135    216    466    522
# 23    026     3              X         X    X         793    431    763    789
# 26    031     3              X    X              X    268    233    151    361
# 27    032     3              X    X         X         698    131     74     86
# 29    034     3              X    X    X              103    119     91    219
# 36    043     3         X                   X    X     56     86    151    133
# 38    045     3         X              X         X    202    593   1970   1656
# 39    046     3         X              X    X          58    154    425    604
# 42    051     3         X         X              X     52     57     74    126
# 43    052     3         X         X         X           8     13     17     22
# 45    054     3         X         X    X               20     78    133    292
# 50    061     3         X    X                   X     52     80    131    115
# 51    062     3         X    X              X         703    269    454    469
# 53    064     3         X    X         X               53    184    458    601
# 57    070     3         X    X    X                    22      9     14     24
# 68    103     3    X                        X    X     24     34     46    143
# 70    105     3    X                   X         X     78    181    396    805
# 71    106     3    X                   X    X          32     66    109    377
# 74    111     3    X              X              X      6     11      8    139
# 75    112     3    X              X         X          10     11      8     26
# 77    114     3    X              X    X               12     36     56    365
# 82    121     3    X         X                   X     22     22     34     73
# 83    122     3    X         X              X         501    162    165    354
# 85    124     3    X         X         X               43     88    152    400
# 89    130     3    X         X    X                     9      9      9     27
# 98    141     3    X    X                        X     78    149    258    519
# 99    142     3    X    X                   X         386    409    463    755
# 101   144     3    X    X              X              383   1176   2395   4432
# 105   150     3    X    X         X                    28     51     55    238
# 113   160     3    X    X    X                       337    375    399    712
# Total                                                7894   7458  12171  18353
```

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    390    329    211    564
# 24    027     4              X         X    X    X    461    759   1423    771
# 28    033     4              X    X         X    X   1139    973    574    306
# 30    035     4              X    X    X         X    578    509    329    503
# 31    036     4              X    X    X    X         345    320    227    211
# 40    047     4         X              X    X    X    127    256    708    708
# 44    053     4         X         X         X    X     35     34     26     56
# 46    055     4         X         X    X         X    606    696    668    971
# 47    056     4         X         X    X    X          26     44     50     88
# 52    063     4         X    X              X    X    151    184    332    194
# 54    065     4         X    X         X         X    122    284    731    582
# 55    066     4         X    X         X    X         217    489   1025    851
# 58    071     4         X    X    X              X      9     20      7     28
# 59    072     4         X    X    X         X          41     36     21     31
# 61    074     4         X    X    X    X               20     46     51    116
# 72    107     4    X                   X    X    X     58     84    129    330
# 76    113     4    X              X         X    X      7      9      5     66
# 78    115     4    X              X    X         X    141    139    122    604
# 79    116     4    X              X    X    X           8      8     11    101
# 84    123     4    X         X              X    X     63     98     91    124
# 86    125     4    X         X         X         X     67    113    223    283
# 87    126     4    X         X         X    X          99    198    268    425
# 90    131     4    X         X    X              X      6      3      0     52
# 91    132     4    X         X    X         X          19     21     10     38
# 93    134     4    X         X    X    X               18     17     22    143
# 100   143     4    X    X                   X    X     37     58    103    190
# 102   145     4    X    X              X         X   5992  12644  22332  23189
# 103   146     4    X    X              X    X         196    510    969   1795
# 106   151     4    X    X         X              X     43     69     54    220
# 107   152     4    X    X         X         X          18     27     15     67
# 109   154     4    X    X         X    X             1227   1390   1065   1738
# 114   161     4    X    X    X                   X     74     96    113    207
# 115   162     4    X    X    X              X        1848   1932   1828   1014
# 117   164     4    X    X    X         X              237    627   1053   1752
# 121   170     4    X    X    X    X                    13     18     15     69
# Total                                                14438  23040  34811  38387
```

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   2247   1877   1193    620
# 48    057     5         X         X    X    X    X    221    219    189    324
# 56    067     5         X    X         X    X    X    556   1015   2544   1151
# 60    073     5         X    X    X         X    X     94     72     62     38
# 62    075     5         X    X    X    X         X    195    187    210    328
# 63    076     5         X    X    X    X    X         106    130    124    128
# 80    117     5    X              X    X    X    X     48     32     25    241
# 88    127     5    X         X         X    X    X    225    321    501    482
# 92    133     5    X         X    X         X    X     31     28     15     52
# 94    135     5    X         X    X    X         X    126    117     88    235
# 95    136     5    X         X    X    X    X          34     56     26    106
# 104   147     5    X    X              X    X    X   2073   4042   6741  10001
# 108   153     5    X    X         X         X    X     39     27     26     96
# 110   155     5    X    X         X    X         X  40157  35344  22417  30602
# 111   156     5    X    X         X    X    X         565    575    410    735
# 116   163     5    X    X    X              X    X    255    271    328    316
# 118   165     5    X    X    X         X         X   2726   5022   8440   9715
# 119   166     5    X    X    X         X    X         902   1976   3172   2688
# 122   171     5    X    X    X    X              X     41     19     13     70
# 123   172     5    X    X    X    X         X          58     71     45     86
# 125   174     5    X    X    X    X    X              659    682    468    782
# Total                                                51358  52083  47037  58796
```

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    850    847    827    485
# 96    137     6     X         X    X    X    X    X    405    324    240    333
# 112   157     6     X    X         X    X    X    X  13239  10814   6862  12202
# 120   167     6     X    X    X         X    X    X  11742  21003  35227  15091
# 124   173     6     X    X    X    X         X    X    131     87     47    114
# 126   175     6     X    X    X    X    X         X  16884  13630   8608  12697
# 127   176     6     X    X    X    X    X    X        2422   2412   1566   1032
# Total                                                45673  49117  53377  41954
```

8 Trees

So, overall, the picture looks like a long shared history (62182 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 (35344 quintuples), in parallel with a long shared history in H- (87499 pairs), then separate histories in Italy and Wales (>85117 "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)
```

```r
  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]){

  ####
  #
  # 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
  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 = tree.x.lim,
       y.lim = c(0,7),
       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]
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)
bottle.txt <- "inbreeding\nLoH / LoS"
if(T){
  text((headx+tailx)/2+(headx-tailx)*(-.01), (heady+taily)/2+(heady-taily)*(-.10),
       bottle.txt, srt=66, font=font.arrow, cex=cex.arrow, col=col.arrow)
} else {
  # experiment at wrapping text along curved path; not too pretty, 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)
    }
  }
}

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.

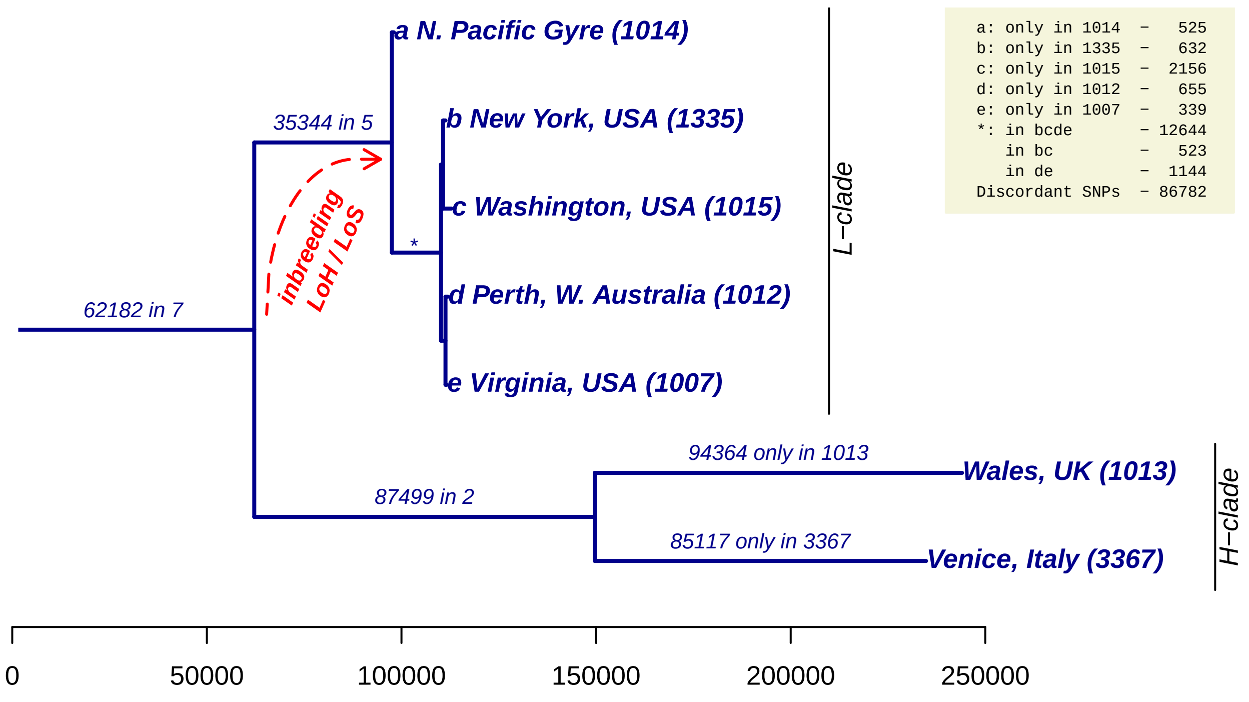
My sense is that the "medium" version is the best for the paper, made here and shown in Fig 1. In theory, this should look exactly like Fig 3, but something is apparently different between Knitr and direct-to-pdf. (Increasing fig.width in Knitr's chunk headers from 8 (as in the pdf call below) to 9 helps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

```
###
#
# MAKE PDF FOR PAPER
#
if(which.snp.tables() == 'trunc-unfiltered'){
  paperfig.path <- paste('figs-mine/Fig3-paperfig-medium-tree-', which.snp.tables(), '.pdf', sep='')
} else {
  paperfig.path <- paste('figs-mine/paperfig-medium-tree-', which.snp.tables(), '.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)
dev.off()

# pdf
#   2
```

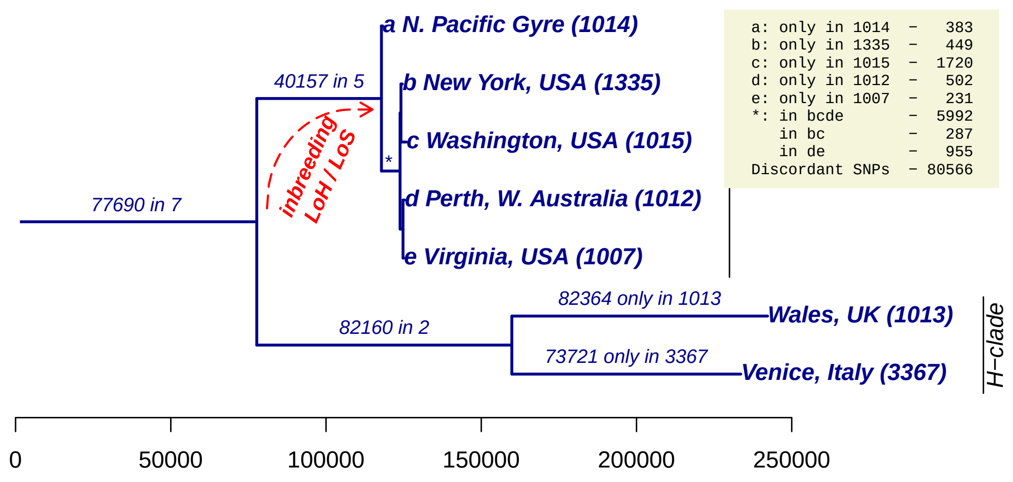
```
# fig.paths for knitr chunks below;  .h for "hand-made" trees; plain for automatic chr1/full versions
myfigpath   <- paste(getwd(), '/figs-knitr/newick-', which.snp.tables(), '-', sep='')
myfigpath.h <- paste(getwd(), '/figs-knitr/newick-', sep='')
```

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



```
a: only in 1014  -   525
b: only in 1335  -   632
c: only in 1015  -  2156
d: only in 1012  -   655
e: only in 1007  -   339
*: in bcde       - 12644
   in bc         -   523
   in de         -  1144
Discordant SNPs  - 86782
```

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



```
a: only in 1014  -   383
b: only in 1335  -   449
c: only in 1015  -  1720
d: only in 1012  -   502
e: only in 1007  -   231
*: in bcde       -  5992
   in bc         -   287
   in de         -   955
Discordant SNPs  - 80566
```

Figure 2: Tree based on qfiltered reads and loose SNP filters. "Lengths" are numbers of shared/private SNPs all Chrs.

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

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

```r
# 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]]:

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

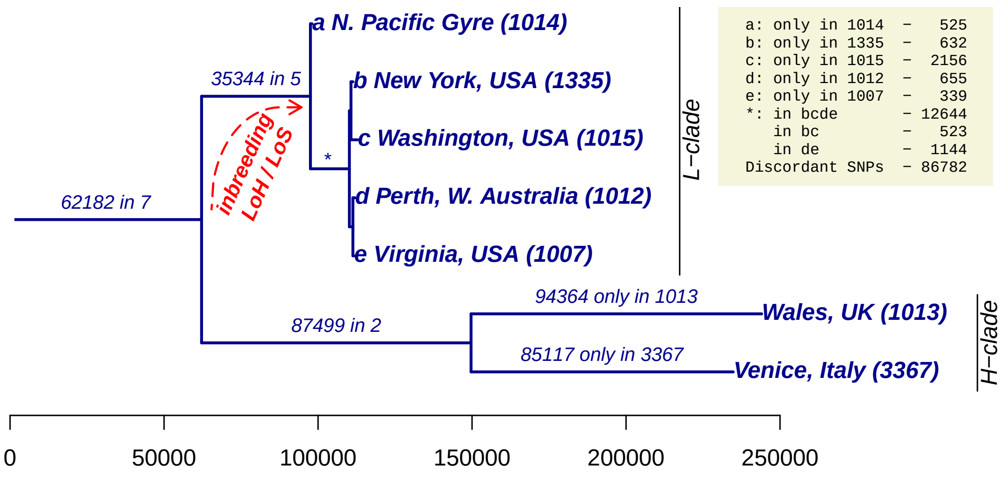
Figure 5, i.e. [[4]]:

```r
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:

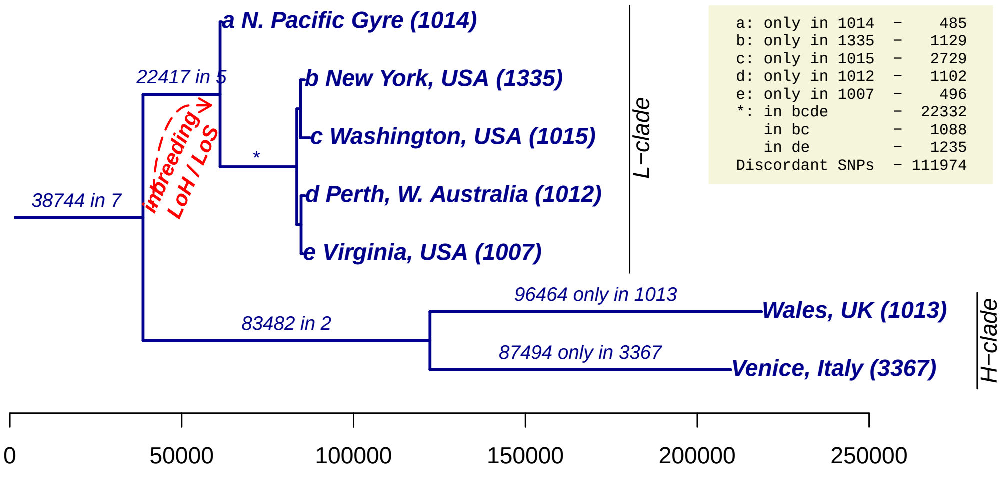
```r
#pat.summaries[c(128,110,102,6,97,19,9,2,5,33,65,17,3),]
tree.edges <- c(128,110,102,6,97,19,9,2,5,33,65,17,3)-1
non.edges <- setdiff(0:127, tree.edges)
sg.edges <- showgroup(restrict.to=tree.edges) ; sg.edges

#       Pat ShrBy 1007 1012 1013 1014 1015 3367 1335 count1 count2 count3 count4
# 2     001     1                                  X    449    632   1129   2260
# 3     002     1                             X       73721  85117  87494 105614
# 5     004     1                        X             1720   2156   2729   4608
# 6     005     2                        X         X    287    523   1088   1407
# 9     010     1                   X                   383    525    485   1231
# 17    020     1              X                      82364  94364  96464 113191
```



```
a: only in 1014  -    525
b: only in 1335  -    632
c: only in 1015  -   2156
d: only in 1012  -    655
e: only in 1007  -    339
*: in bcde       -  12644
   in bc         -    523
   in de         -   1144
Discordant SNPs  -  86782
```

Figure 3: Tree based on qfiltered reads and medium SNP filters. "Lengths" are numbers of shared/private SNPs all Chrs.



```
a: only in 1014  -    485
b: only in 1335  -   1129
c: only in 1015  -   2729
d: only in 1012  -   1102
e: only in 1007  -    496
*: in bcde       -  22332
   in bc         -   1088
   in de         -   1235
Discordant SNPs  - 111974
```

Figure 4: Tree based on qfiltered reads and strict SNP filters. "Lengths" are numbers of shared/private SNPs all Chrs.

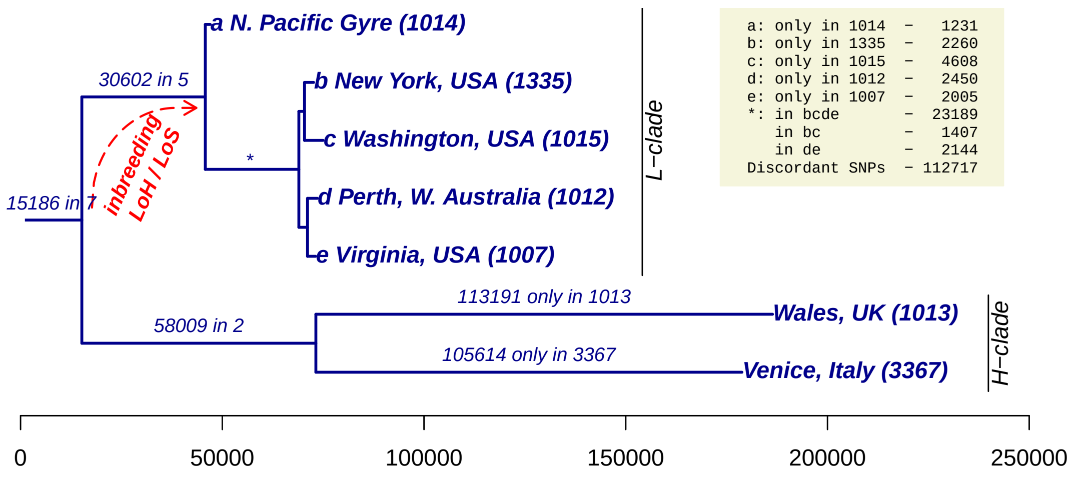


Figure 5: Tree based on qfiltered reads and unfiltered SNPs. “Lengths” are numbers of shared/private SNPs all Chrs.

```
# 19    022     2              X              X       82160  87499  83482  58009
# 33    040     1         X                             502    655   1102   2450
# 65    100     1    X                                  231    339    496   2005
# 97    140     2    X    X                             955   1144   1235   2144
# 102   145     4    X    X              X         X   5992  12644  22332  23189
# 110   155     5    X    X         X    X         X  40157  35344  22417  30602
# 128   177     7    X    X    X    X    X    X    X  77690  62182  38744  15186
# Total                                              366611 383124 359197 361896
```

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
# 32    037     5              X    X    X    X    X   2247   1877   1193    620
# 104   147     5    X    X              X    X    X   2073   4042   6741  10001
# 109   154     4    X    X         X    X             1227   1390   1065   1738
# 112   157     6    X    X         X    X    X    X  13239  10814   6862  12202
# 115   162     4    X    X    X              X        1848   1932   1828   1014
# 118   165     5    X    X    X         X         X   2726   5022   8440   9715
# 119   166     5    X    X    X         X    X         902   1976   3172   2688
# 120   167     6    X    X    X         X    X    X  11742  21003  35227  15091
# 126   175     6    X    X    X    X    X         X  16884  13630   8608  12697
# 127   176     6    X    X    X    X    X    X        2422   2412   1566   1032
# Other   (   105 rows   w/   c2    < 1390    )       25256  22684  37272  45919
# Total                                               80566  86782 111974 112717
```

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  80566  86782 111974 112717

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

#       count1 count2 count3 count4
# Total     18   18.5   23.8   23.7
```

In short, the sharing pattern observed at 86782 or 18.5% of the 469906 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  13239  10814   6862  12202
# 120   167     6    X    X    X          X    X    X  11742  21003  35227  15091
# 126   175     6    X    X    X    X     X         X  16884  13630   8608  12697
# Other   (     4 rows   w/   c2    < 10814    )        3808   3670   2680   1964
# Total                                                45673  49117  53377  41954

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

# [1] 0.5236915
```

I.e., 52.4% 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: 62182 .  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   1210 289197 290407 117895 408302 0.2887446
# 2   02  85695 179062 264757 143545 408302 0.3515658
# 3   03  86625 105697 192322 215980 408302 0.5289712
# 4   04   2734 279522 282256 126046 408302 0.3087078
# 5   05   3889 274296 278185 130117 408302 0.3186783
# 6   06  87989 100546 188535 219767 408302 0.5382462
# 7   07  89639  98363 188002 220300 408302 0.5395516
# 8   10   1103 332135 333238  75064 408302 0.1838443 < R B C O
# 9   11   2221 282010 284231 124071 408302 0.3038707
# 10  12  86269 123335 209604 198698 408302 0.4866447
# 11  13  87911 102474 190385 217917 408302 0.5337152
# 12  14   3398 276093 279491 128811 408302 0.3154797
# 13  15   6023 273221 279244 129058 408302 0.3160847
# 14  16  88762  98932 187694 220608 408302 0.5403060
# 15  17  92437  97647 190084 218218 408302 0.5344524
# 16  20  94942 165462 260404 147898 408302 0.3622270
# 17  21  95741  96070 191811 216491 408302 0.5302227
# 18  22 267558  61350 328908  79394 408302 0.1944492
# 19  23 269213   8703 277916 130386 408302 0.3193372
# 20  24  97293  91454 188747 219555 408302 0.5377270
# 21  25  98831  89139 187970 220332 408302 0.5396300
# 22  26 270478   4870 275348 132954 408302 0.3256266
# 23  27 273828   3327 277155 131147 408302 0.3212010
# 24  30  95536 112945 208481 199821 408302 0.4893951
# 25  31  97054  93029 190083 218219 408302 0.5344549
# 26  32 268332  21310 289642 118660 408302 0.2906182
# 27  33 271905   6449 278354 129948 408302 0.3182644
# 28  34  98145  89824 187969 220333 408302 0.5396324
# 29  35 101895  88428 190323 217979 408302 0.5338671
# 30  36 271890   3636 275526 132776 408302 0.3251907
# 31  37 280857   2716 283573 124729 408302 0.3054822
# 32  40   1233 283707 284940 123362 408302 0.3021342
# 33  41   1957 273177 275134 133168 408302 0.3261507
# 34  42  86469 103226 189695 218607 408302 0.5354052
# 35  43  87577  98843 186420 221882 408302 0.5434262
# 36  44   3668 272853 276521 131781 408302 0.3227537
# 37  45   5508 269028 274536 133766 408302 0.3276153
# 38  46  89196  97572 186768 221534 408302 0.5425739
# 39  47  91873  95971 187844 220458 408302 0.5399386
# 40  50   1782 275846 277628 130674 408302 0.3200425
# 41  51   3049 271596 274645 133657 408302 0.3273484
# 42  52  87080  99819 186899 221403 408302 0.5422530
# 43  53  88991  97918 186909 221393 408302 0.5422285
# 44  54   4434 270058 274492 133810 408302 0.3277231
# 45  55   8497 268202 276699 131603 408302 0.3223178
# 46  56  90128  96225 186353 221949 408302 0.5435903
# 47  57  95836  95357 191193 217109 408302 0.5317363
# 48  60  95713  93634 189347 218955 408302 0.5362575
# 49  61  96684  89422 186106 222196 408302 0.5441952
# 50  62 268717   6909 275626 132676 408302 0.3249458
# 51  63 270814   3906 274720 133582 408302 0.3271647
# 52  64  98527  88440 186967 221335 408302 0.5420865
# 53  65 101114  86697 187811 220491 408302 0.5400194
# 54  66 272743   2629 275372 132930 408302 0.3255678
# 55  67 278683   1453 280136 128166 408302 0.3139000
# 56  70  96340  90579 186919 221383 408302 0.5422041
# 57  71  98107  88583 186690 221612 408302 0.5427649
# 58  72 269573   4578 274151 134151 408302 0.3285583
# 59  73 273771   3195 276966 131336 408302 0.3216639
# 60  74  99536  87112 186648 221654 408302 0.5428678
# 61  75 105295  86101 191396 216906 408302 0.5312391
# 62  76 274535   1596 276131 132171 408302 0.3237089
# 63  77 288224    917 289141 119161 408302 0.2918453       < 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
# 4     003     2                             X    X    994    298    532    587
# 7     006     2                        X    X         624    138    282    590
# 8     007     3                        X    X    X    104    197    371    557
# 12    013     3                   X         X    X    257    226    146    338
# 16    017     4                   X    X    X    X    390    329    211    564
# 18    021     2              X                   X    998    167    337    402
# 20    023     3              X              X    X   1274    558   1020    533
# 21    024     2              X         X              686    195    410    625
# 22    025     3              X         X         X    135    216    466    522
# 23    026     3              X         X    X         793    431    763    789
# 24    027     4              X         X    X    X    461    759   1423    771
# 26    031     3              X    X              X    268    233    151    361
# 27    032     3              X    X         X         698    131     74     86
# 28    033     4              X    X         X    X   1139    973    574    306
# 29    034     3              X    X    X              103    119     91    219
# 30    035     4              X    X    X         X    578    509    329    503
# 31    036     4              X    X    X    X         345    320    227    211
# 32    037     5              X    X    X    X    X   2247   1877   1193    620
# 35    042     2         X                   X         503    119    254    394
# 39    046     3         X              X    X          58    154    425    604
# 40    047     4         X              X    X    X    127    256    708    708
# 48    057     5         X         X    X    X    X    221    219    189    324
# 49    060     2         X    X                        627    116    237    388
# 51    062     3         X    X              X         703    269    454    469
# 52    063     4         X    X              X    X    151    184    332    194
# 53    064     3         X    X         X               53    184    458    601
# 54    065     4         X    X         X         X    122    284    731    582
# 55    066     4         X    X         X    X         217    489   1025    851
# 56    067     5         X    X         X    X    X    556   1015   2544   1151
# 62    075     5         X    X    X    X         X    195    187    210    328
# 63    076     5         X    X    X    X    X         106    130    124    128
# 64    077     6         X    X    X    X    X    X    850    847    827    485
# 83    122     3    X         X              X         501    162    165    354
# 86    125     4    X         X         X         X     67    113    223    283
# 87    126     4    X         X         X    X          99    198    268    425
# 88    127     5    X         X         X    X    X    225    321    501    482
# 94    135     5    X         X    X    X         X    126    117     88    235
# 96    137     6    X         X    X    X    X    X    405    324    240    333
# 99    142     3    X    X                   X         386    409    463    755
# 103   146     4    X    X              X    X         196    510    969   1795
# 104   147     5    X    X              X    X    X   2073   4042   6741  10001
# 111   156     5    X    X         X    X    X         565    575    410    735
# 112   157     6    X    X         X    X    X    X  13239  10814   6862  12202
# 113   160     3    X    X    X                        337    375    399    712
# 115   162     4    X    X    X              X        1848   1932   1828   1014
# 116   163     5    X    X    X              X    X    255    271    328    316
# 117   164     4    X    X    X         X              237    627   1053   1752
# 118   165     5    X    X    X         X         X   2726   5022   8440   9715
# 119   166     5    X    X    X         X    X          902   1976   3172   2688
# 120   167     6    X    X    X         X    X    X  11742  21003  35227  15091
# 125   174     5    X    X    X    X    X              659    682    468    782
# 126   175     6    X    X    X    X    X         X  16884  13630   8608  12697
# 127   176     6    X    X    X    X    X    X         2422   2412   1566   1032
# Other   (    39 rows   w/   c2    <  100    )        3209   1750   1921   4847
# Total                                               75686  79394  97058  94037
```

```
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
# 10    011     2                   X              X    515    486    317    827
# 12    013     3                   X         X    X    257    226    146    338
# 13    014     2                   X    X              133    139    132    417
# 14    015     3                   X    X         X   1041    984    660   1389
# 16    017     4                   X    X    X    X    390    329    211    564
# 26    031     3              X    X              X    268    233    151    361
# 27    032     3              X    X         X         698    131     74     86
# 28    033     4              X    X         X    X   1139    973    574    306
# 29    034     3              X    X    X              103    119     91    219
# 30    035     4              X    X    X         X    578    509    329    503
# 31    036     4              X    X    X    X         345    320    227    211
# 32    037     5              X    X    X    X    X   2247   1877   1193    620
# 46    055     4         X         X    X         X    606    696    668    971
# 48    057     5         X         X    X    X    X    221    219    189    324
# 62    075     5         X    X    X    X         X    195    187    210    328
# 63    076     5         X    X    X    X    X         106    130    124    128
# 64    077     6         X    X    X    X    X    X    850    847    827    485
# 78    115     4    X              X    X         X    141    139    122    604
# 94    135     5    X         X    X    X         X    126    117     88    235
# 96    137     6    X         X    X    X    X    X    405    324    240    333
# 109   154     4    X    X         X    X             1227   1390   1065   1738
# 110   155     5    X    X         X    X         X  40157  35344  22417  30602
# 111   156     5    X    X         X    X    X         565    575    410    735
# 112   157     6    X    X         X    X    X    X  13239  10814   6862  12202
# 125   174     5    X    X    X    X    X              659    682    468    782
# 126   175     6    X    X    X    X    X         X  16884  13630   8608  12697
# 127   176     6    X    X    X    X    X    X         2422   2412   1566   1032
# Other   (    35 rows   w/   c2    <  100    )        2151   1232   1130   3730
# Total                                               87668  75064  49099  72767
```

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     42     92    313    368
# 37    044     2         X              X               69    279   1001   1809
# 38    045     3         X              X         X    202    593   1970   1656
# 66    101     2    X                             X     29     47     73    314
# 69    104     2    X                   X               39    122    329   1196
# 70    105     3    X                   X         X     78    181    396    805
# 98    141     3    X    X                        X     78    149    258    519
# 101   144     3    X    X              X              383   1176   2395   4432
# 102   145     4    X    X              X         X   5992  12644  22332  23189
# Total                                                6912  15283  29067  34288

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

# [1] 82.7
```

So, of the 15283 SNPs found only in bcde, 82.7% 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    515    486    317    827
# 13    014     2                   X    X              133    139    132    417
# 14    015     3                   X    X         X   1041    984    660   1389
# 41    050     2         X         X                    13     24     53    105
# 42    051     3         X         X              X     52     57     74    126
# 45    054     3         X         X    X               20     78    133    292
# 46    055     4         X         X    X         X    606    696    668    971
# 73    110     2    X              X                    12     11     29    150
# 74    111     3    X              X              X      6     11      8    139
# 77    114     3    X              X    X               12     36     56    365
# 78    115     4    X              X    X         X    141    139    122    604
# 105   150     3    X    X         X                    28     51     55    238
# 106   151     4    X    X         X              X     43     69     54    220
# 109   154     4    X    X         X    X             1227   1390   1065   1738
# 110   155     5    X    X         X    X         X  40157  35344  22417  30602
# Total                                               44006  39515  25843  38183

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

I.e., of the 39515 SNPs found only in abcde, 89.4% 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    994    298    532    587
# 7     006     2                        X    X         624    138    282    590
# 8     007     3                        X    X    X    104    197    371    557
# 11    012     2                   X         X         565     49     32     93
# 12    013     3                   X         X    X    257    226    146    338
# 15    016     3                   X    X    X          46     60     49    152
# 16    017     4                   X    X    X    X    390    329    211    564
# 18    021     2              X                   X    998    167    337    402
# 20    023     3              X              X    X   1274    558   1020    533
# 21    024     2              X         X              686    195    410    625
# 22    025     3              X         X         X    135    216    466    522
# 23    026     3              X         X    X         793    431    763    789
# 24    027     4              X         X    X    X    461    759   1423    771
# 25    030     2              X    X                   609     69     47     93
# 26    031     3              X    X              X    268    233    151    361
# 27    032     3              X    X         X         698    131     74     86
# 28    033     4              X    X         X    X   1139    973    574    306
# 29    034     3              X    X    X              103    119     91    219
# 30    035     4              X    X    X         X    578    509    329    503
# 31    036     4              X    X    X    X         345    320    227    211
# 32    037     5              X    X    X    X    X   2247   1877   1193    620
# 35    042     2         X                   X         503    119    254    394
# 36    043     3         X                   X    X     56     86    151    133
# 39    046     3         X              X    X          58    154    425    604
# 40    047     4         X              X    X    X    127    256    708    708
# 43    052     3         X         X         X           8     13     17     22
# 44    053     4         X         X         X    X     35     34     26     56
# 47    056     4         X         X    X    X          26     44     50     88
# 48    057     5         X         X    X    X    X    221    219    189    324
# 49    060     2         X    X                        627    116    237    388
# 50    061     3         X    X                   X     52     80    131    115
# 51    062     3         X    X              X         703    269    454    469
# 52    063     4         X    X              X    X    151    184    332    194
# 53    064     3         X    X         X               53    184    458    601
# 54    065     4         X    X         X         X    122    284    731    582
# 55    066     4         X    X         X    X         217    489   1025    851
# 56    067     5         X    X         X    X    X    556   1015   2544   1151
# 57    070     3         X    X    X                    22      9     14     24
# 58    071     4         X    X    X              X      9     20      7     28
# 59    072     4         X    X    X         X          41     36     21     31
# 60    073     5         X    X    X         X    X     94     72     62     38
# 61    074     4         X    X    X    X               20     46     51    116
# 62    075     5         X    X    X    X         X    195    187    210    328
# 63    076     5         X    X    X    X    X         106    130    124    128
# 64    077     6         X    X    X    X    X    X    850    847    827    485
# 67    102     2    X                        X         351     67     96    351
# 68    103     3    X                        X    X     24     34     46    143
# 71    106     3    X                   X    X          32     66    109    377
# 72    107     4    X                   X    X    X     58     84    129    330
# 75    112     3    X              X         X          10     11      8     26
# 76    113     4    X              X         X    X      7      9      5     66
# 79    116     4    X              X    X    X           8      8     11    101
# 80    117     5    X              X    X    X    X     48     32     25    241
# 81    120     2    X         X                        432     76     98    309
# 82    121     3    X         X                   X     22     22     34     73
# 83    122     3    X         X              X         501    162    165    354
# 84    123     4    X         X              X    X     63     98     91    124
# 85    124     3    X         X         X               43     88    152    400
# 86    125     4    X         X         X         X     67    113    223    283
# 87    126     4    X         X         X    X          99    198    268    425
# 88    127     5    X         X         X    X    X    225    321    501    482
# 89    130     3    X         X    X                     9      9      9     27
# 90    131     4    X         X    X              X      6      3      0     52
# 91    132     4    X         X    X         X          19     21     10     38
# 92    133     5    X         X    X         X    X     31     28     15     52
# 93    134     4    X         X    X    X               18     17     22    143
# 94    135     5    X         X    X    X         X    126    117     88    235
# 95    136     5    X         X    X    X    X          34     56     26    106
# 96    137     6    X         X    X    X    X    X    405    324    240    333
# 99    142     3    X    X                   X         386    409    463    755
# 100   143     4    X    X                   X    X     37     58    103    190
# 103   146     4    X    X              X    X           196    510    969   1795
# 104   147     5    X    X              X    X    X   2073   4042   6741  10001
# 107   152     4    X    X         X         X           18     27     15     67
# 108   153     5    X    X         X         X    X     39     27     26     96
# 111   156     5    X    X         X    X    X           565    575    410    735
# 112   157     6    X    X         X    X    X    X  13239  10814   6862  12202
# 113   160     3    X    X    X                          337    375    399    712
# 114   161     4    X    X    X                   X     74     96    113    207
# 115   162     4    X    X    X              X          1848   1932   1828   1014
# 116   163     5    X    X    X              X    X    255    271    328    316
# 117   164     4    X    X    X         X                237    627   1053   1752
# 118   165     5    X    X    X         X         X   2726   5022   8440   9715
# 119   166     5    X    X    X         X    X           902   1976   3172   2688
# 120   167     6    X    X    X         X    X    X  11742  21003  35227  15091
# 121   170     4    X    X    X    X                      13     18     15     69
# 122   171     5    X    X    X    X              X     41     19     13     70
# 123   172     5    X    X    X    X         X            58     71     45     86
# 124   173     6    X    X    X    X         X    X    131     87     47    114
# 125   174     5    X    X    X    X    X                659    682    468    782
# 126   175     6    X    X    X    X    X         X  16884  13630   8608  12697
# 127   176     6    X    X    X    X    X    X          2422   2412   1566   1032
# 128   177     7    X    X    X    X    X    X    X  77690  62182  38744  15186
# Total                                              153376 141576 135802 109223

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

# [1] 43.9
```

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

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

# [1] 3000

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   2073   4042   6741  10001
# 112   157     6    X    X         X    X    X    X  13239  10814   6862  12202
# 118   165     5    X    X    X         X         X   2726   5022   8440   9715
# 120   167     6    X    X    X         X    X    X  11742  21003  35227  15091
# 126   175     6    X    X    X    X    X         X  16884  13630   8608  12697
# 128   177     7    X    X    X    X    X    X    X  77690  62182  38744  15186
# Other   (    87 rows   w/   c2    < 3000    )       29022  24883  31180  34331
# Total                                              153376 141576 135802 109223
```

So, of the 141576 SNPs found both in the L- and H-clades, 43.9% 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 $\approx$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] 86204

consistent.count[2]

# [1] 469906

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

# [1] 18.3
```

I.e., 86204 or 18.3% of the 469906 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] 469906
```

Conceptually, we sample, with replacement, n2=469906 SNP positions from among the 469906 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 469906 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 [1:469906] 127 109 16 18 18 2 16 109 18 2 ...

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

#      val count
# [1,]   0   625
# [2,]   1   631
# [3,]   2 85325
# [4,]   3   292
# [5,] 124   722
# [6,] 125 13882
# [7,] 126  2292
# [8,] 127 62392

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] 0.9999865

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
# 1     000     0                                        NA    625     NA     NA
# 2     001     1                                  X     NA    631     NA     NA
# 3     002     1                             X          NA  85325     NA     NA
# 5     004     1                        X               NA   2121     NA     NA
# 6     005     2                        X         X     NA    550     NA     NA
# 9     010     1                   X                    NA    523     NA     NA
# 10    011     2                   X              X     NA    484     NA     NA
# 14    015     3                   X    X         X     NA   1022     NA     NA
# 17    020     1              X                         NA  93622     NA     NA
# 19    022     2              X              X          NA  87425     NA     NA
# 20    023     3              X              X    X     NA    549     NA     NA
# 23    026     3              X         X    X          NA    458     NA     NA
# 24    027     4              X         X    X    X     NA    736     NA     NA
# 28    033     4              X    X         X    X     NA   1042     NA     NA
# 30    035     4              X    X    X         X     NA    528     NA     NA
# 32    037     5              X    X    X    X    X     NA   1832     NA     NA
# 33    040     1         X                              NA    659     NA     NA
# 38    045     3         X              X         X     NA    612     NA     NA
# 46    055     4         X         X    X         X     NA    721     NA     NA
# 55    066     4         X    X         X    X          NA    478     NA     NA
# 56    067     5         X    X         X    X    X     NA    992     NA     NA
# 64    077     6         X    X    X    X    X    X     NA    751     NA     NA
# 97    140     2    X    X                              NA   1107     NA     NA
# 101   144     3    X    X              X               NA   1184     NA     NA
# 102   145     4    X    X              X         X     NA  12647     NA     NA
# 103   146     4    X    X              X    X          NA    515     NA     NA
# 104   147     5    X    X              X    X    X     NA   4052     NA     NA
# 109   154     4    X    X         X    X               NA   1408     NA     NA
# 110   155     5    X    X         X    X         X     NA  35461     NA     NA
# 111   156     5    X    X         X    X    X          NA    575     NA     NA
# 112   157     6    X    X         X    X    X    X     NA  10806     NA     NA
# 115   162     4    X    X    X              X          NA   1904     NA     NA
# 117   164     4    X    X    X         X               NA    653     NA     NA
# 118   165     5    X    X    X         X         X     NA   5063     NA     NA
# 119   166     5    X    X    X         X    X          NA   1925     NA     NA
# 120   167     6    X    X    X         X    X    X     NA  21159     NA     NA
# 125   174     5    X    X    X    X    X               NA    722     NA     NA
# 126   175     6    X    X    X    X    X         X     NA  13882     NA     NA
# 127   176     6    X    X    X    X    X    X          NA   2292     NA     NA
# 128   177     7    X    X    X    X    X    X    X     NA  62392     NA     NA
# Other   (    88 rows   w/   c2    <   400     )          NA  10473     NA     NA
# Total                                                  NA 469906     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: 62392 .  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   1256 288443 289699 118440 408139 0.2901953
# 2   02  85950 178885 264835 143304 408139 0.3511157
# 3   03  86873 104979 191852 216287 408139 0.5299347
# 4   04   2746 278944 281690 126449 408139 0.3098185
# 5   05   3927 273725 277652 130487 408139 0.3197122
# 6   06  88232  99824 188056 220083 408139 0.5392354
# 7   07  89887  97655 187542 220597 408139 0.5404948
# 8   10   1148 331649 332797  75342 408139 0.1845989 < R B C O
# 9   11   2263 281358 283621 124518 408139 0.3050872
# 10  12  86520 122657 209177 198962 408139 0.4874859
# 11  13  88143 101715 189858 218281 408139 0.5348202
# 12  14   3402 275412 278814 129325 408139 0.3168651
# 13  15   6089 272628 278717 129422 408139 0.3171028
# 14  16  88996  98179 187175 220964 408139 0.5413940
# 15  17  92728  96929 189657 218482 408139 0.5353127
# 16  20  94247 165944 260191 147948 408139 0.3624942
# 17  21  95027  96301 191328 216811 408139 0.5312185
# 18  22 266997  61606 328603  79536 408139 0.1948748
# 19  23 268618   8714 277332 130807 408139 0.3204962
# 20  24  96546  91668 188214 219925 408139 0.5388483
# 21  25  98115  89369 187484 220655 408139 0.5406369
# 22  26 269915   4918 274833 133306 408139 0.3266191
# 23  27 273243   3368 276611 131528 408139 0.3222628
# 24  30  94831 113202 208033 200106 408139 0.4902888
# 25  31  96325  93227 189552 218587 408139 0.5355700
# 26  32 267778  21354 289132 119007 408139 0.2915845
# 27  33 271371   6442 277813 130326 408139 0.3193177
# 28  34  97375  90017 187392 220747 408139 0.5408623
# 29  35 101208  88637 189845 218294 408139 0.5348521
# 30  36 271299   3653 274952 133187 408139 0.3263276
# 31  37 280336   2740 283076 125063 408139 0.3064226
# 32  40   1284 283167 284451 123688 408139 0.3030536
# 33  41   2006 272572 274578 133561 408139 0.3272439
# 34  42  86730 102533 189263 218876 408139 0.5362781
# 35  43  87827  98064 185891 222248 408139 0.5445400
# 36  44   3686 272305 275991 132148 408139 0.3237818
# 37  45   5570 268498 274068 134071 408139 0.3284935
# 38  46  89436  96842 186278 221861 408139 0.5435918
# 39  47  92135  95276 187411 220728 408139 0.5408158
# 40  50   1841 275289 277130 131009 408139 0.3209911
# 41  51   3109 271039 274148 133991 408139 0.3282975
# 42  52  87348  99102 186450 221689 408139 0.5431703
# 43  53  89239  97163 186402 221737 408139 0.5432879
# 44  54   4449 269457 273906 134233 408139 0.3288904
# 45  55   8622 267664 276286 131853 408139 0.3230591
# 46  56  90373  95507 185880 222259 408139 0.5445669
# 47  57  96176  94671 190847 217292 408139 0.5323971
# 48  60  95036  93947 188983 219156 408139 0.5369641
# 49  61  95981  89660 185641 222498 408139 0.5451525
# 50  62 268207   6970 275177 132962 408139 0.3257763
# 51  63 270256   3906 274162 133977 408139 0.3282632
# 52  64  97792  88671 186463 221676 408139 0.5431385
# 53  65 100405  86959 187364 220775 408139 0.5409309
# 54  66 272203   2669 274872 133267 408139 0.3265236
# 55  67 278088   1506 279594 128545 408139 0.3149540
# 56  70  95662  90856 186518 221621 408139 0.5430037
# 57  71  97409  88836 186245 221894 408139 0.5436726
# 58  72 269072   4619 273691 134448 408139 0.3294172
# 59  73 273297   3211 276508 131631 408139 0.3225151
# 60  74  98775  87363 186138 222001 408139 0.5439348
# 61  75 104642  86369 191011 217128 408139 0.5319952
# 62  76 273963   1642 275605 132534 408139 0.3247276
# 63  77 287626    974 288600 119539 408139 0.2928880       < 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   1148 331649 332797  75342 408139 0.1845989 < R B C O
# 18   22 266997  61606 328603  79536 408139 0.1948748
# 1    01   1256 288443 289699 118440 408139 0.2901953
# 63   77 287626    974 288600 119539 408139 0.2928880       < 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 101 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   1088 332249 333337  75187 408524 0.1840455 < R B C O
# 18  22 267474  61282 328756  79768 408524 0.1952590
# 1   01   1196 289143 290339 118185 408524 0.2892976
# 63  77 288371    896 289267 119257 408524 0.2919216       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1057 332177 333234  74775 408009 0.1832680 < R B C O
# 18  22 267508  61021 328529  79480 408009 0.1947996
# 1   01   1160 289146 290306 117703 408009 0.2884814
# 63  77 288256    873 289129 118880 408009 0.2913661       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1209 332226 333435  74716 408151 0.1830597 < R B C O
# 18  22 267615  61098 328713  79438 408151 0.1946289
# 1   01   1252 289364 290616 117535 408151 0.2879694
# 63  77 288335    981 289316 118835 408151 0.2911545       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1115 332625 333740  74945 408685 0.1833808 < R B C O
# 18  22 267324  61658 328982  79703 408685 0.1950231
# 1   01   1226 289045 290271 118414 408685 0.2897439
# 63  77 288062    967 289029 119656 408685 0.2927830       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1147 332068 333215  75372 408587 0.1844699 < R B C O
# 18  22 267629  61237 328866  79721 408587 0.1951139
# 1   01   1220 289521 290741 117846 408587 0.2884233
# 63  77 288475    921 289396 119191 408587 0.2917151       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1094 332239 333333  75144 408477 0.1839614 < R B C O
# 18  22 267590  61319 328909  79568 408477 0.1947919
# 1   01   1209 289230 290439 118038 408477 0.2889710
# 63  77 288291    936 289227 119250 408477 0.2919381       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1129 331922 333051  75091 408142 0.1839825 < R B C O
# 18  22 267142  61339 328481  79661 408142 0.1951796
# 1   01   1183 288932 290115 118027 408142 0.2891812
# 63  77 287829    914 288743 119399 408142 0.2925428       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1078 331642 332720  75177 407897 0.1843039 < R B C O
# 18  22 267228  61500 328728  79169 407897 0.1940907
# 1   01   1221 289013 290234 117663 407897 0.2884625
# 63  77 287876    933 288809 119088 407897 0.2919561       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1108 332043 333151  75154 408305 0.1840634 < R B C O
# 18  22 267364  61672 329036  79269 408305 0.1941416
# 1   01   1177 289100 290277 118028 408305 0.2890682
# 63  77 288049    914 288963 119342 408305 0.2922864       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1134 331690 332824  75413 408237 0.1847285 < R B C O
# 18  22 267249  61601 328850  79387 408237 0.1944630
# 1   01   1204 288856 290060 118177 408237 0.2894814
# 63  77 287661    938 288599 119638 408237 0.2930602       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1169 331836 333005  75157 408162 0.1841352 < R B C O
# 18  22 267443  61137 328580  79582 408162 0.1949765
# 1   01   1246 289175 290421 117741 408162 0.2884663
# 63  77 288198    960 289158 119004 408162 0.2915607       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1104 332220 333324  75231 408555 0.1841392 < R B C O
# 18  22 267058  61409 328467  80088 408555 0.1960275
# 1   01   1253 289103 290356 118199 408555 0.2893099
# 63  77 287999    960 288959 119596 408555 0.2927293       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1102 332340 333442  74746 408188 0.1831166 < R B C O
# 18  22 267721  61287 329008  79180 408188 0.1939792
# 1   01   1142 289604 290746 117442 408188 0.2877155
# 63  77 288502    885 289387 118801 408188 0.2910448       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1096 332522 333618  74810 408428 0.1831657 < R B C O
# 18  22 267564  61396 328960  79468 408428 0.1945704
# 1   01   1231 289156 290387 118041 408428 0.2890130
# 63  77 288348    926 289274 119154 408428 0.2917381       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1099 332217 333316  74923 408239 0.1835273 < R B C O
# 18  22 267385  61375 328760  79479 408239 0.1946874
# 1   01   1197 289117 290314 117925 408239 0.2888627
# 63  77 288070    946 289016 119223 408239 0.2920422       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1151 332282 333433  74768 408201 0.1831647 < R B C O
# 18  22 267648  61608 329256  78945 408201 0.1933974
# 1   01   1229 289252 290481 117720 408201 0.2883873
# 63  77 288379    948 289327 118874 408201 0.2912144       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1118 331932 333050  75132 408182 0.1840650 < R B C O
# 18  22 267582  61339 328921  79261 408182 0.1941805
# 1   01   1225 289232 290457 117725 408182 0.2884130
# 63  77 288235    956 289191 118991 408182 0.2915146       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1045 332077 333122  75027 408149 0.1838226 < R B C O
# 18  22 267395  61457 328852  79297 408149 0.1942844
# 1   01   1193 288978 290171 117978 408149 0.2890562
# 63  77 288010    891 288901 119248 408149 0.2921678       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1178 332105 333283  74818 408101 0.1833321 < R B C O
# 18  22 267564  61296 328860  79241 408101 0.1941701
# 1   01   1292 289112 290404 117697 408101 0.2884016
# 63  77 288323    930 289253 118848 408101 0.2912220       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1123 332123 333246  75316 408562 0.1843441 < R B C O
# 18  22 267365  61401 328766  79796 408562 0.1953094
# 1   01   1228 289097 290325 118237 408562 0.2893979
# 63  77 288070    965 289035 119527 408562 0.2925554       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1109 331845 332954  74932 407886 0.1837082 < R B C O
# 18  22 267420  61249 328669  79217 407886 0.1942136
# 1   01   1211 288896 290107 117779 407886 0.2887547
# 63  77 287997    930 288927 118959 407886 0.2916477       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1152 331872 333024  75339 408363 0.1844903 < R B C O
# 18  22 267153  61616 328769  79594 408363 0.1949099
# 1   01   1232 288949 290181 118182 408363 0.2894043
# 63  77 287763    957 288720 119643 408363 0.2929820       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1067 332072 333139  75272 408411 0.1843045 < R B C O
# 18  22 267734  61335 329069  79342 408411 0.1942700
# 1   01   1188 289283 290471 117940 408411 0.2887777
# 63  77 288273    931 289204 119207 408411 0.2918800       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1107 332585 333692  74967 408659 0.1834463 < R B C O
# 18  22 267848  61315 329163  79496 408659 0.1945289
# 1   01   1163 289634 290797 117862 408659 0.2884116
# 63  77 288339    917 289256 119403 408659 0.2921825       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1111 331964 333075  75043 408118 0.1838757 < R B C O
# 18  22 267208  61212 328420  79698 408118 0.1952818
# 1   01   1245 288957 290202 117916 408118 0.2889262
# 63  77 287952    902 288854 119264 408118 0.2922292       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1117 331600 332717  75224 407941 0.1843992 < R B C O
# 18  22 267370  61480 328850  79091 407941 0.1938785
# 1   01   1211 289091 290302 117639 407941 0.2883726
# 63  77 287962    927 288889 119052 407941 0.2918363       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1110 332218 333328  74679 408007 0.1830336 < R B C O
# 18  22 267669  61266 328935  79072 408007 0.1938006
# 1   01   1252 289330 290582 117425 408007 0.2878014
# 63  77 288511    920 289431 118576 408007 0.2906225       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1156 332267 333423  74685 408108 0.1830030 < R B C O
# 18  22 267426  61251 328677  79431 408108 0.1946323
# 1   01   1209 289277 290486 117622 408108 0.2882129
# 63  77 287916    924 288840 119268 408108 0.2922462       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1077 331827 332904  75204 408108 0.1842748 < R B C O
# 18  22 267213  60935 328148  79960 408108 0.1959285
# 1   01   1171 288869 290040 118068 408108 0.2893058
# 63  77 288131    886 289017 119091 408108 0.2918125       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1099 332443 333542  75194 408736 0.1839672 < R B C O
# 18  22 267643  61262 328905  79831 408736 0.1953119
# 1   01   1207 289344 290551 118185 408736 0.2891475
# 63  77 288192    920 289112 119624 408736 0.2926681       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1076 332146 333222  74863 408085 0.1834495 < R B C O
# 18  22 267763  61240 329003  79082 408085 0.1937881
# 1   01   1267 289142 290409 117676 408085 0.2883615
# 63  77 288460    909 289369 118716 408085 0.2909100       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1077 332176 333253  75044 408297 0.1837976 < R B C O
# 18  22 267789  61603 329392  78905 408297 0.1932539
# 1   01   1164 289199 290363 117934 408297 0.2888437
# 63  77 288326    907 289233 119064 408297 0.2916113       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1098 332186 333284  75189 408473 0.1840734 < R B C O
# 18  22 267816  61180 328996  79477 408473 0.1945710
# 1   01   1163 289497 290660 117813 408473 0.2884230
# 63  77 288547    855 289402 119071 408473 0.2915027       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1098 332046 333144  75150 408294 0.1840585 < R B C O
# 18  22 267518  61353 328871  79423 408294 0.1945240
# 1   01   1228 288934 290162 118132 408294 0.2893307
# 63  77 288237    902 289139 119155 408294 0.2918363       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1066 332545 333611  75075 408686 0.1836985 < R B C O
# 18  22 267809  61506 329315  79371 408686 0.1942102
# 1   01   1236 289204 290440 118246 408686 0.2893322
# 63  77 288457    886 289343 119343 408686 0.2920164       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1048 332753 333801  74630 408431 0.1827236 < R B C O
# 18  22 268561  61022 329583  78848 408431 0.1930510
# 1   01   1187 289914 291101 117330 408431 0.2872701
# 63  77 289021    844 289865 118566 408431 0.2902963       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1080 332679 333759  74430 408189 0.1823420 < R B C O
# 18  22 267689  61137 328826  79363 408189 0.1944271
# 1   01   1174 289272 290446 117743 408189 0.2884522
# 63  77 288140    971 289111 119078 408189 0.2917227       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1081 332018 333099  75086 408185 0.1839509 < R B C O
# 18  22 267605  61006 328611  79574 408185 0.1949459
# 1   01   1166 289258 290424 117761 408185 0.2884991
# 63  77 288192    904 289096 119089 408185 0.2917525       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1114 331620 332734  75019 407753 0.1839815 < R B C O
# 18  22 266972  61317 328289  79464 407753 0.1948827
# 1   01   1189 288513 289702 118051 407753 0.2895160
# 63  77 287622    912 288534 119219 407753 0.2923804       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1130 332138 333268  75042 408310 0.1837868 < R B C O
# 18  22 267820  61231 329051  79259 408310 0.1941148
# 1   01   1193 289360 290553 117757 408310 0.2884010
# 63  77 288265    926 289191 119119 408310 0.2917367       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1113 332374 333487  75242 408729 0.1840877 < R B C O
# 18  22 267641  61564 329205  79524 408729 0.1945641
# 1   01   1236 289184 290420 118309 408729 0.2894558
# 63  77 288468    898 289366 119363 408729 0.2920346       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1176 332239 333415  74775 408190 0.1831868 < R B C O
# 18  22 267737  61470 329207  78983 408190 0.1934957
# 1   01   1229 289220 290449 117741 408190 0.2884466
# 63  77 288216    934 289150 119040 408190 0.2916289       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1088 332082 333170  75305 408475 0.1843564 < R B C O
# 18  22 267763  61011 328774  79701 408475 0.1951184
# 1   01   1202 289255 290457 118018 408475 0.2889234
# 63  77 288539    876 289415 119060 408475 0.2914744       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1081 332029 333110  75249 408359 0.1842717 < R B C O
# 18  22 267378  61184 328562  79797 408359 0.1954089
# 1   01   1196 289257 290453 117906 408359 0.2887312
# 63  77 288070    937 289007 119352 408359 0.2922722       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1105 331970 333075  75010 408085 0.1838097 < R B C O
# 18  22 267456  61316 328772  79313 408085 0.1943541
# 1   01   1198 289263 290461 117624 408085 0.2882341
# 63  77 288142    933 289075 119010 408085 0.2916304       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1117 331732 332849  75407 408256 0.1847052 < R B C O
# 18  22 267113  61663 328776  79480 408256 0.1946818
# 1   01   1224 288880 290104 118152 408256 0.2894066
# 63  77 287705    959 288664 119592 408256 0.2929338       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1137 332017 333154  75210 408364 0.1841739 < R B C O
# 18  22 267440  61708 329148  79216 408364 0.1939838
# 1   01   1189 289133 290322 118042 408364 0.2890607
# 63  77 287958    957 288915 119449 408364 0.2925062       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1060 332127 333187  74769 407956 0.1832771 < R B C O
# 18  22 267320  61196 328516  79440 407956 0.1947269
# 1   01   1201 288758 289959 117997 407956 0.2892395
# 63  77 287988    894 288882 119074 407956 0.2918795       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1091 331836 332927  75518 408445 0.1848915 < R B C O
# 18  22 266893  61841 328734  79711 408445 0.1951572
# 1   01   1206 288799 290005 118440 408445 0.2899778
# 63  77 287810    926 288736 119709 408445 0.2930847       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1096 331558 332654  75098 407752 0.1841757 < R B C O
# 18  22 267263  61350 328613  79139 407752 0.1940861
# 1   01   1194 288884 290078 117674 407752 0.2885921
# 63  77 287955    913 288868 118884 407752 0.2915596       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1092 332374 333466  74958 408424 0.1835299 < R B C O
# 18  22 267481  61191 328672  79752 408424 0.1952677
# 1   01   1179 289344 290523 117901 408424 0.2886730
# 63  77 288481    896 289377 119047 408424 0.2914790       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1137 332326 333463  74819 408282 0.1832532 < R B C O
# 18  22 267629  61063 328692  79590 408282 0.1949388
# 1   01   1221 289539 290760 117522 408282 0.2878452
# 63  77 288445    935 289380 118902 408282 0.2912252       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1082 332312 333394  74641 408035 0.1829279 < R B C O
# 18  22 267054  61549 328603  79432 408035 0.1946696
# 1   01   1164 288868 290032 118003 408035 0.2891982
# 63  77 287738    896 288634 119401 408035 0.2926244       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1098 332518 333616  74739 408355 0.1830246 < R B C O
# 18  22 267725  61551 329276  79079 408355 0.1936526
# 1   01   1218 289516 290734 117621 408355 0.2880361
# 63  77 288406    910 289316 119039 408355 0.2915086       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1071 331552 332623  75435 408058 0.1848634 < R B C O
# 18  22 267017  61378 328395  79663 408058 0.1952247
# 1   01   1133 288711 289844 118214 408058 0.2896990
# 63  77 287646    882 288528 119530 408058 0.2929240       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1098 332105 333203  74825 408028 0.1833820 < R B C O
# 18  22 268130  61091 329221  78807 408028 0.1931412
# 1   01   1209 289421 290630 117398 408028 0.2877205
# 63  77 288769    901 289670 118358 408028 0.2900732       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1056 332349 333405  74768 408173 0.1831772 < R B C O
# 18  22 267617  61250 328867  79306 408173 0.1942951
# 1   01   1198 289261 290459 117714 408173 0.2883924
# 63  77 288234    911 289145 119028 408173 0.2916116       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1134 331631 332765  75151 407916 0.1842316 < R B C O
# 18  22 266934  61563 328497  79419 407916 0.1946945
# 1   01   1224 288614 289838 118078 407916 0.2894665
# 63  77 287736    967 288703 119213 407916 0.2922489       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1077 332415 333492  74725 408217 0.1830522 < R B C O
# 18  22 267611  61114 328725  79492 408217 0.1947298
# 1   01   1247 289414 290661 117556 408217 0.2879743
# 63  77 288339    947 289286 118931 408217 0.2913426       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1091 332027 333118  75453 408571 0.1846754 < R B C O
# 18  22 267451  61612 329063  79508 408571 0.1946002
# 1   01   1165 289098 290263 118308 408571 0.2895653
# 63  77 288092    895 288987 119584 408571 0.2926884       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1057 332387 333444  74868 408312 0.1833598 < R B C O
# 18  22 267732  61419 329151  79161 408312 0.1938738
# 1   01   1207 289431 290638 117674 408312 0.2881963
# 63  77 288514    905 289419 118893 408312 0.2911817       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1101 332116 333217  75086 408303 0.1838977 < R B C O
# 18  22 267164  61255 328419  79884 408303 0.1956488
# 1   01   1200 288871 290071 118232 408303 0.2895693
# 63  77 287998    953 288951 119352 408303 0.2923123       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1069 332419 333488  74856 408344 0.1833160 < R B C O
# 18  22 267621  61374 328995  79349 408344 0.1943190
# 1   01   1159 289124 290283 118061 408344 0.2891214
# 63  77 288327    889 289216 119128 408344 0.2917344       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1148 332556 333704  74841 408545 0.1831891 < R B C O
# 18  22 267952  61122 329074  79471 408545 0.1945220
# 1   01   1240 289552 290792 117753 408545 0.2882253
# 63  77 288516    955 289471 119074 408545 0.2914587       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1101 332137 333238  74929 408167 0.1835744 < R B C O
# 18  22 267922  61656 329578  78589 408167 0.1925413
# 1   01   1223 289571 290794 117373 408167 0.2875612
# 63  77 288715    895 289610 118557 408167 0.2904620       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1133 332067 333200  75143 408343 0.1840193 < R B C O
# 18  22 267756  61238 328994  79349 408343 0.1943195
# 1   01   1204 289116 290320 118023 408343 0.2890291
# 63  77 288490    931 289421 118922 408343 0.2912307       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1072 332726 333798  74937 408735 0.1833388 < R B C O
# 18  22 268007  61152 329159  79576 408735 0.1946885
# 1   01   1214 289526 290740 117995 408735 0.2886834
# 63  77 288542    897 289439 119296 408735 0.2918664       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1060 331873 332933  75398 408331 0.1846492 < R B C O
# 18  22 266840  61684 328524  79807 408331 0.1954468
# 1   01   1207 288474 289681 118650 408331 0.2905731
# 63  77 287491    844 288335 119996 408331 0.2938694       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1131 332016 333147  75348 408495 0.1844527 < R B C O
# 18  22 267402  61661 329063  79432 408495 0.1944504
# 1   01   1187 289224 290411 118084 408495 0.2890709
# 63  77 288210    877 289087 119408 408495 0.2923120       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1121 332634 333755  75084 408839 0.1836518 < R B C O
# 18  22 267858  61407 329265  79574 408839 0.1946341
# 1   01   1224 289747 290971 117868 408839 0.2882993
# 63  77 288530    902 289432 119407 408839 0.2920636       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1057 332011 333068  75162 408230 0.1841168 < R B C O
# 18  22 267047  61710 328757  79473 408230 0.1946770
# 1   01   1211 288611 289822 118408 408230 0.2900522
# 63  77 287751    897 288648 119582 408230 0.2929280       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1005 332163 333168  74892 408060 0.1835318 < R B C O
# 18  22 267603  61192 328795  79265 408060 0.1942484
# 1   01   1114 289175 290289 117771 408060 0.2886120
# 63  77 288165    866 289031 119029 408060 0.2916948       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1128 331715 332843  75146 407989 0.1841863 < R B C O
# 18  22 266872  61518 328390  79599 407989 0.1951008
# 1   01   1239 288474 289713 118276 407989 0.2899000
# 63  77 287614    902 288516 119473 407989 0.2928339       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1153 331873 333026  75528 408554 0.1848666 < R B C O
# 18  22 267239  61693 328932  79622 408554 0.1948873
# 1   01   1218 289133 290351 118203 408554 0.2893204
# 63  77 288098    926 289024 119530 408554 0.2925684       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1181 332057 333238  75164 408402 0.1840442 < R B C O
# 18  22 267262  61364 328626  79776 408402 0.1953369
# 1   01   1232 289008 290240 118162 408402 0.2893277
# 63  77 287991    913 288904 119498 408402 0.2925990       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1085 332114 333199  75384 408583 0.1845011 < R B C O
# 18  22 267795  61501 329296  79287 408583 0.1940536
# 1   01   1189 289403 290592 117991 408583 0.2887810
# 63  77 288337    904 289241 119342 408583 0.2920875       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1108 332117 333225  74967 408192 0.1836562 < R B C O
# 18  22 268019  60927 328946  79246 408192 0.1941390
# 1   01   1173 289434 290607 117585 408192 0.2880630
# 63  77 288403    929 289332 118860 408192 0.2911865       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1074 332681 333755  74654 408409 0.1827922 < R B C O
# 18  22 268204  60934 329138  79271 408409 0.1940971
# 1   01   1233 289838 291071 117338 408409 0.2873051
# 63  77 288804    935 289739 118670 408409 0.2905666       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1081 331931 333012  75016 408028 0.1838501 < R B C O
# 18  22 267248  61314 328562  79466 408028 0.1947562
# 1   01   1192 288850 290042 117986 408028 0.2891615
# 63  77 287800    898 288698 119330 408028 0.2924554       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1098 332250 333348  74978 408326 0.1836229 < R B C O
# 18  22 267508  61115 328623  79703 408326 0.1951945
# 1   01   1210 289169 290379 117947 408326 0.2888550
# 63  77 288035    885 288920 119406 408326 0.2924281       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1062 331758 332820  75472 408292 0.1848481 < R B C O
# 18  22 267169  61442 328611  79681 408292 0.1951569
# 1   01   1197 288874 290071 118221 408292 0.2895501
# 63  77 287845    897 288742 119550 408292 0.2928051       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1126 332248 333374  75028 408402 0.1837111 < R B C O
# 18  22 267470  61188 328658  79744 408402 0.1952586
# 1   01   1150 289188 290338 118064 408402 0.2890877
# 63  77 288350    892 289242 119160 408402 0.2917713       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1102 331714 332816  75144 407960 0.1841945 < R B C O
# 18  22 267284  61325 328609  79351 407960 0.1945068
# 1   01   1216 288884 290100 117860 407960 0.2889009
# 63  77 287923    948 288871 119089 407960 0.2919134       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1073 332654 333727  74606 408333 0.1827087 < R B C O
# 18  22 267923  61313 329236  79097 408333 0.1937071
# 1   01   1194 289413 290607 117726 408333 0.2883088
# 63  77 288537    905 289442 118891 408333 0.2911619       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1115 331827 332942  75207 408149 0.1842636 < R B C O
# 18  22 267373  61532 328905  79244 408149 0.1941546
# 1   01   1239 289020 290259 117890 408149 0.2888406
# 63  77 287967    943 288910 119239 408149 0.2921458       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1096 331728 332824  75524 408348 0.1849501 < R B C O
# 18  22 267499  61620 329119  79229 408348 0.1940232
# 1   01   1190 289150 290340 118008 408348 0.2889888
# 63  77 288115    904 289019 119329 408348 0.2922238       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1126 331817 332943  75278 408221 0.1844050 < R B C O
# 18  22 267225  61408 328633  79588 408221 0.1949630
# 1   01   1224 288880 290104 118117 408221 0.2893457
# 63  77 287983    901 288884 119337 408221 0.2923343       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1104 332480 333584  74763 408347 0.1830869 < R B C O
# 18  22 267561  61384 328945  79402 408347 0.1944474
# 1   01   1232 289180 290412 117935 408347 0.2888107
# 63  77 288409    862 289271 119076 408347 0.2916049       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1114 332099 333213  75182 408395 0.1840914 < R B C O
# 18  22 267363  61708 329071  79324 408395 0.1942335
# 1   01   1204 289092 290296 118099 408395 0.2891784
# 63  77 288290    869 289159 119236 408395 0.2919624       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1125 332695 333820  74685 408505 0.1828252 < R B C O
# 18  22 267604  61633 329237  79268 408505 0.1940441
# 1   01   1269 289222 290491 118014 408505 0.2888924
# 63  77 288106    947 289053 119452 408505 0.2924126       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1066 332944 334010  75159 409169 0.1836869 < R B C O
# 18  22 268341  61323 329664  79505 409169 0.1943085
# 1   01   1205 289938 291143 118026 409169 0.2884529
# 63  77 288912    951 289863 119306 409169 0.2915812       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1117 332286 333403  74972 408375 0.1835862 < R B C O
# 18  22 267625  61636 329261  79114 408375 0.1937288
# 1   01   1217 289401 290618 117757 408375 0.2883551
# 63  77 288278    916 289194 119181 408375 0.2918421       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1115 332162 333277  74871 408148 0.1834408 < R B C O
# 18  22 267735  61234 328969  79179 408148 0.1939958
# 1   01   1268 289409 290677 117471 408148 0.2878147
# 63  77 288359    921 289280 118868 408148 0.2912375       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1155 331780 332935  75515 408450 0.1848819 < R B C O
# 18  22 267313  61278 328591  79859 408450 0.1955172
# 1   01   1236 288846 290082 118368 408450 0.2897980
# 63  77 288241    937 289178 119272 408450 0.2920113       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1082 332301 333383  75049 408432 0.1837491 < R B C O
# 18  22 268080  61236 329316  79116 408432 0.1937067
# 1   01   1218 289711 290929 117503 408432 0.2876929
# 63  77 288710    950 289660 118772 408432 0.2907999       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1115 331740 332855  75429 408284 0.1847464 < R B C O
# 18  22 267114  61686 328800  79484 408284 0.1946782
# 1   01   1156 288920 290076 118208 408284 0.2895240
# 63  77 287972    945 288917 119367 408284 0.2923627       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1116 332395 333511  74763 408274 0.1831197 < R B C O
# 18  22 267226  61297 328523  79751 408274 0.1953370
# 1   01   1286 288827 290113 118161 408274 0.2894159
# 63  77 287756    942 288698 119576 408274 0.2928817       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1083 331731 332814  75098 407912 0.1841034 < R B C O
# 18  22 267237  61655 328892  79020 407912 0.1937183
# 1   01   1216 288726 289942 117970 407912 0.2892045
# 63  77 287693    917 288610 119302 407912 0.2924699       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1132 332497 333629  74856 408485 0.1832528 < R B C O
# 18  22 267630  61654 329284  79201 408485 0.1938896
# 1   01   1225 289078 290303 118182 408485 0.2893178
# 63  77 288228    933 289161 119324 408485 0.2921135       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1081 331712 332793  74993 407786 0.1839028 < R B C O
# 18  22 267312  61105 328417  79369 407786 0.1946340
# 1   01   1209 288948 290157 117629 407786 0.2884577
# 63  77 287747    864 288611 119175 407786 0.2922489       < L
#    pat   left  right   both  cross    all     ratio      best
# 8   10   1090 332121 333211  74760 407971 0.1832483 < R B C O
# 18  22 267570  61236 328806  79165 407971 0.1940457
# 1   01   1303 289219 290522 117449 407971 0.2878857
# 63  77 288124    940 289064 118907 407971 0.2914594       < 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 101 bootstrap replicates, we saw 101 samples with the 6/1 split having fewer conflicts than the 5/2 split, and the minimum separation between them was ≈ -18 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.
# bcor     "    0.999973432" "    0.999991997" "    0.99999417" "    0.999993603" "    0.999995979" "    0.999998
# b52cross "78589"            "79244"            "79432"           "79415.267326733" "79590"            "80088"
# b61cross "74430"            "74856"            "75049"           "75047.534653465" "75204"            "75528"
# b61-b52  "-4988"            "-4581"            "-4346"           "-4367.732673267" "-4206"            "-3660"
# % rev    "  100"            "  100"            "  100"           "  100"            "  100"            "  100"
#          sd
# bcor     "    0.000003499"
# b52cross "  268.69067312"
# b61cross "  246.249327486"
# b61-b52  "  276.221754795"
# % rev    "    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, $0 \le k \le 4$ (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')
```

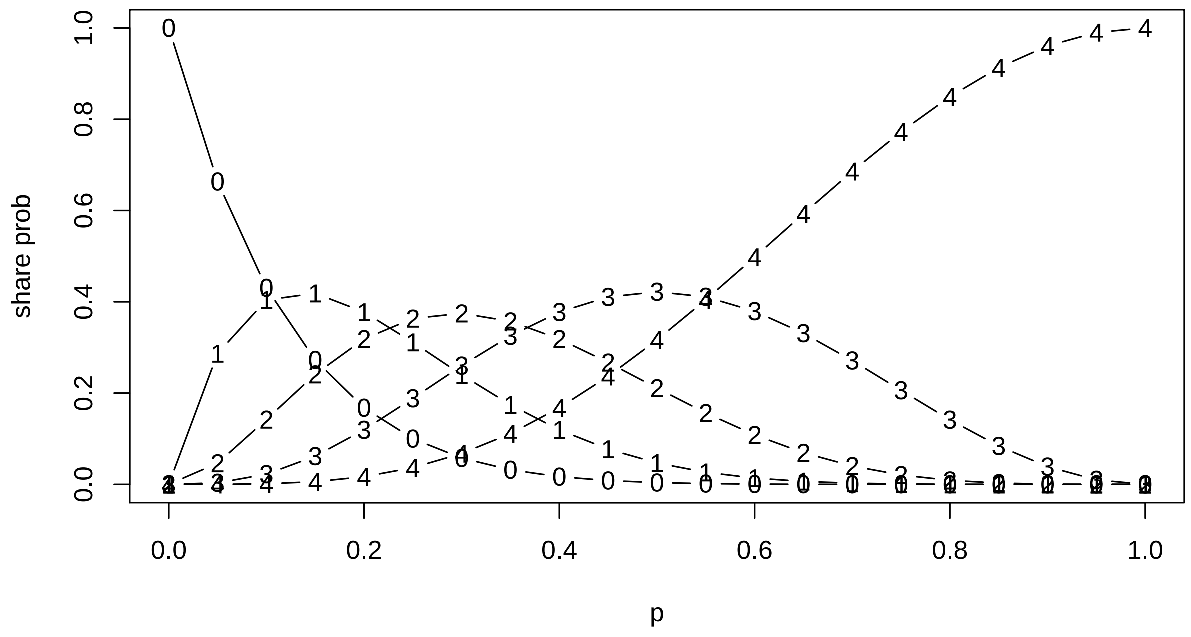


Figure 6: Sharing Probability

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.

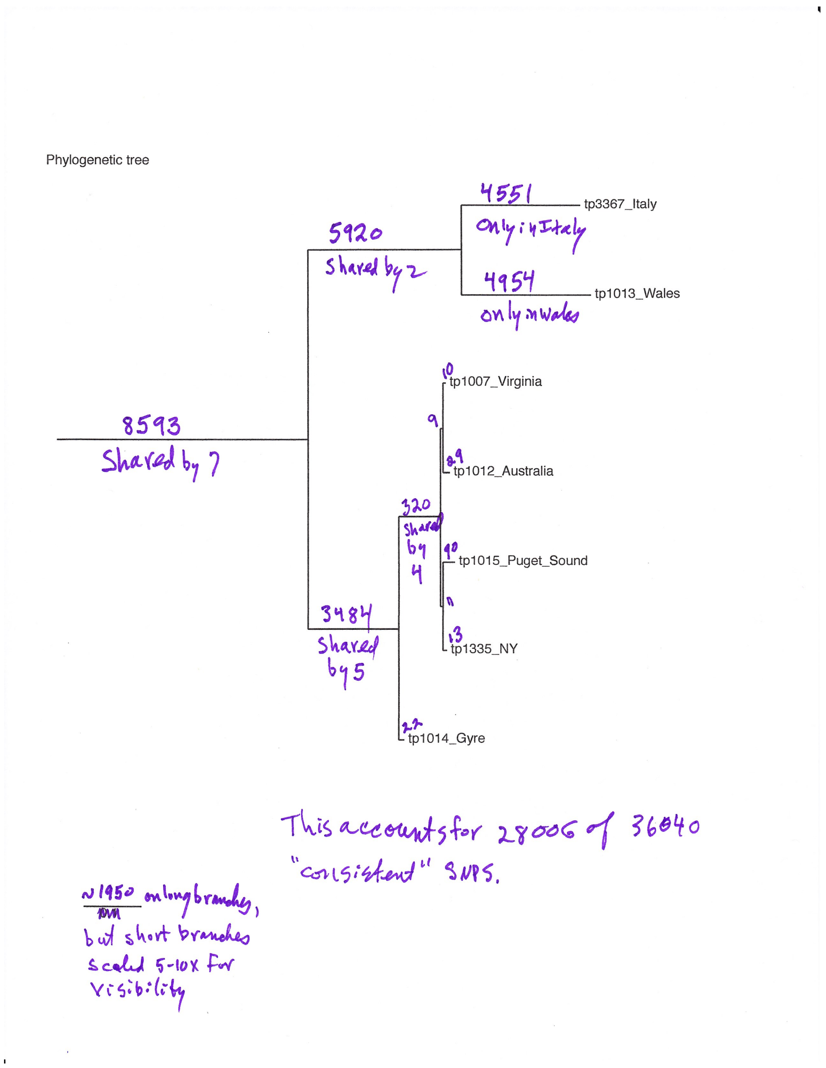


Figure 7: Inferred Tree, based on Chr1, un-q-filtered reads, loose criteria. (Note: to visually resolve the edges among the 5, their lengths were scaled by 5x – 10x in this figure, but not in the newick description shown in the text.)

```
# 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);
```

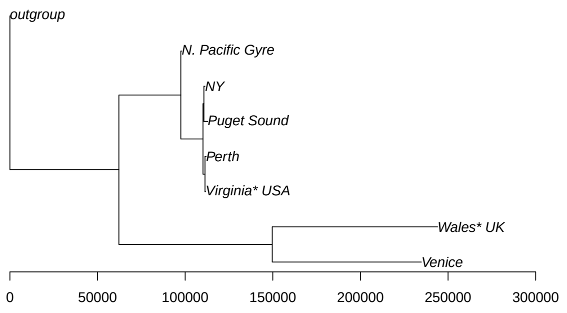


Figure 8: Tree based on qfiltered reads and medium SNP filters. "Lengths" are numbers of shared/private SNPs all Chrs. (no edge labels, nolegend)

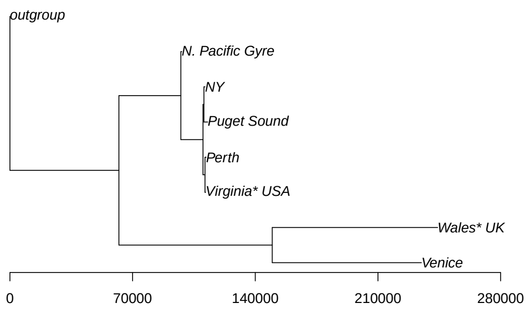
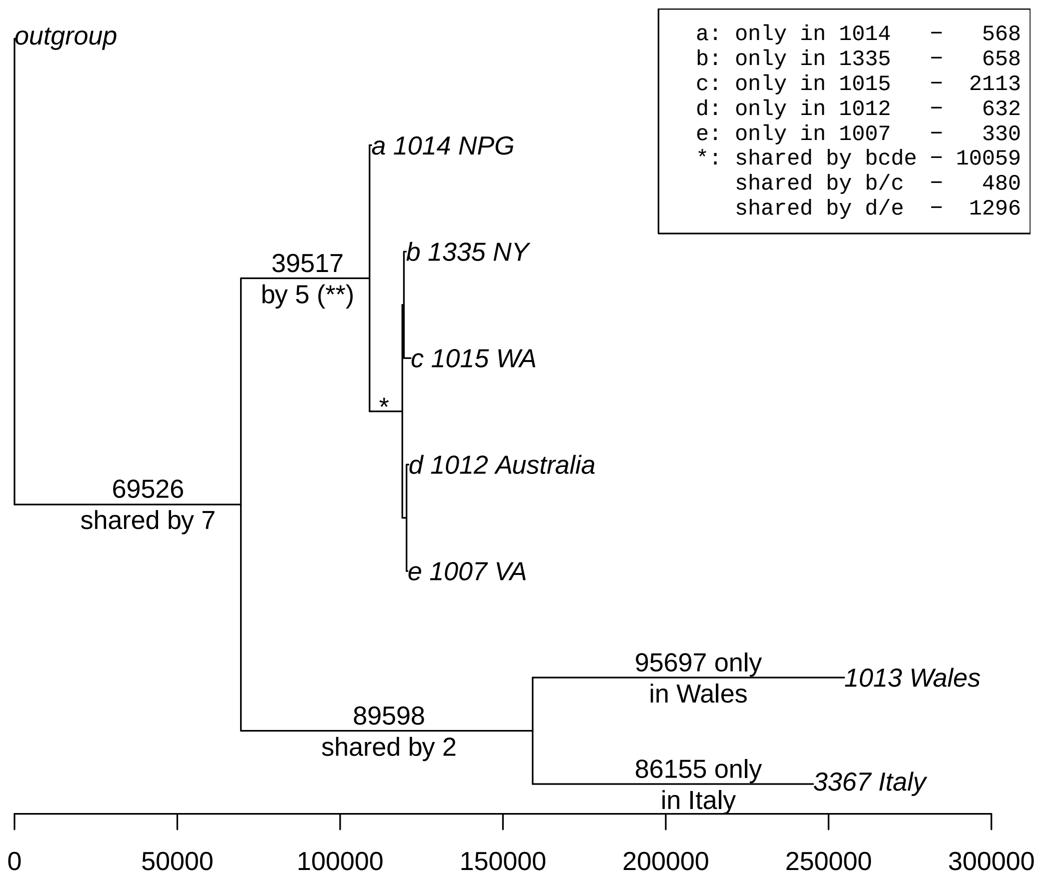


Figure 9: Tree based on qfiltered reads and medium SNP filters. "Lengths" are numbers of shared/private SNPs all Chrs. (no edge labels, no legend, short scale bar)

```
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:

```
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]])
}
```



```
a: only in 1014   -   568
b: only in 1335   -   658
c: only in 1015   -  2113
d: only in 1012   -   632
e: only in 1007   -   330
*: shared by bcde - 10059
   shared by b/c  -   480
   shared by d/e  -  1296
```

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