Fig 2A for paper: Distribution of Chr 1 Deserts
(Chr1, qfiltered)

March 22, 2018

Contents

1 Intro

6/1/2017: Simple script to build fig 2A for the paper, since I can't find Tony's code: distribution of deserts across Chr 1 in all 7 strains. Some associated investigation of N/NA/gaps in reference sequence.

2 Preliminaries

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 @ D-10-18-109-56.dhcp4.washington.edu; SVN Id, I miss you.  $Id: wlr.R  2017-07-21 or later

setup.my.wd('paperfigs') # set working dir; UPDATE if this file moves, or if COPY/PASTE to new file
setup.my.knitr('Fig2A-desert-distribution-figs-knitr/') # knitr's "unnamed-chunk-nnn" figures
my.figs.dir <- 'Fig2A-desert-distribution-figs-mine/'
generic.setup(my.figs.dir)
```

```
# frequently need to add figpath to file name
fpath <- function(base, suffix='.pdf', dir=my.figs.dir){
  return(paste(dir, base, suffix, sep=''))
}
```

NOTE: A few code chunks use the knitr cache. I do NOT check for consistency of cached data with code changes and I do NOT know to what extent/whether knitr does, either. If in doubt, delete directories "cache" (knitr's) and "00common/mycache" (mine) to force rebuild.

CLEAR CACHE!!! T/F will/won't force knitr cache removal (well, actually a rename):

```
decache(FALSE)

# Cache exists, and was left alone.
```

Only using genome seq, so doesn't matter which table set, but qfiltered set is a bit smaller:

```
# see wlr.R for paths
snp.tables.chr1q <- load.snp.tables(use.chr1.tables = TRUE , data.name='full.tables.02.25.15')

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

```
chr1.len <- genome.length.constants()$chr1.length  ## 3042585
```

3 Gaps in the Reference Sequence/SNP Tables

To place the "pink" (gap) bar(s) in the fig, we need to know: How long, how many, where are the gaps in Chr 1 reference sequence?

```
# Count 'N's in the ref seq
ncount <- unlist(lapply(snp.tables.chr1q,function(x) sum(x$Ref=='N',na.rm=T)))
ncount

# 1007 1012 1013 1014 1015 3367 1335
#    9   17   16   20   12   16   10
```

```
# Repeat for NA positions (several columns in the tables are simultaneously NA)
nacount <- NULL
for(i in 1:7){
  nacount <- rbind(nacount, unlist(lapply(snp.tables.chr1q[[i]],function(x){sum(is.na(x))})))
}
row.names(nacount) <- names(snp.tables.chr1q)
nacount

#      snp   Chr   Pos   Ref Cov a g c t n .match exon indel   chr   pos rawCov
# 1007   0 11761 11761 11761   0 0 0 0 0 0      0    0     0 11761 11761      0
# 1012   0 11653 11653 11653   0 0 0 0 0 0      0    0     0 11653 11653      0
# 1013   0 11708 11708 11708   0 0 0 0 0 0      0    0     0 11708 11708      0
# 1014   0 11603 11603 11603   0 0 0 0 0 0      0    0     0 11603 11603      0
# 1015   0 11724 11724 11724   0 0 0 0 0 0      0    0     0 11724 11724      0
# 3367   0 11664 11664 11664   0 0 0 0 0 0      0    0     0 11664 11664      0
# 1335   0 11567 11567 11567   0 0 0 0 0 0      0    0     0 11567 11567      0
```

```
# Are the NA counts consistent?
nasummary <- rbind(max=apply(nacount,1,max),min=apply(nacount[,c(2,3,4,14,15)],1,min))
nasummary <- rbind(nasummary, equal=(nasummary[1,]==nasummary[2,]))
nasummary

#        1007  1012  1013  1014  1015  3367  1335
# max   11761 11653 11708 11603 11724 11664 11567
# min   11761 11653 11708 11603 11724 11664 11567
# equal     1     1     1     1     1     1     1

# Consistent? (yes):
all(nasummary[1,]==nasummary[2,])

# [1] TRUE
```

So, there are 10-20 "N" positions in the Chr 1 reference sequence in all 7 isolates and about 11,700 NA positions (with slight variability from strain to strain for each). This variability is seemingly a side effect ofTony's table-build scripts: they leave NA entries where read coverage is zero, which varies a bit, on top of the fixed gap(s) in the reference sequence. E.g., NA's are sprinkled around, but the only "N"'s are at the edges of the one big gaps. E.g., note the slightly different placement of the N/NA boundary and the drop in rawCov to zero it corresponding positions here:

```
snp.tables.chr1q[[4]][358900:358915,]

#          snp  Chr    Pos  Ref Cov a g c t n .match  exon indel  chr    pos rawCov
# 358900   0 Chr1 358900    T   5 0 0 0 0 0      5 FALSE FALSE Chr1 358900     11
# 358901   0 Chr1 358901    G   3 0 0 1 0 0      2 FALSE FALSE Chr1 358901     10
# 358902   0 Chr1 358902    T   2 0 0 0 0 0      2 FALSE FALSE Chr1 358902      9
# 358903   0 Chr1 358903    N   1 0 0 1 0 0      0 FALSE FALSE Chr1 358903      9
# 358904   0 Chr1 358904    N   1 0 1 0 0 0      0 FALSE FALSE Chr1 358904      8
# 358905   0 Chr1 358905    N   5 0 5 0 0 0      0 FALSE FALSE Chr1 358905      8
# 358906   0 Chr1 358906    N   3 0 3 0 0 0      0 FALSE FALSE Chr1 358906      8
# 358907   0 Chr1 358907    N   0 0 0 0 0 0      0 FALSE FALSE Chr1 358907      7
# 358908   0 Chr1 358908    N   1 1 0 0 0 0      0 FALSE FALSE Chr1 358908      5
# 358909   0 Chr1 358909    N   0 0 0 0 0 0      0 FALSE FALSE Chr1 358909      3
# 358910   0 Chr1 358910    N   1 1 0 0 0 0      0 FALSE FALSE Chr1 358910      1
# 358911   0 Chr1 358911    N   1 0 1 0 0 0      0 FALSE FALSE Chr1 358911      1
# 358912   0 Chr1 358912    N   1 0 0 1 0 0      0 FALSE FALSE Chr1 358912      1
# 358913   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358914   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358915   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0

snp.tables.chr1q[[5]][358900:358915,]

#          snp  Chr    Pos  Ref Cov a g c t n .match  exon indel  chr    pos rawCov
# 358900   0 Chr1 358900    T   1 0 0 0 0 0      1 FALSE FALSE Chr1 358900      3
# 358901   0 Chr1 358901    G   0 0 0 0 0 0      0 FALSE FALSE Chr1 358901      3
# 358902   0 Chr1 358902    T   0 0 0 0 0 0      0 FALSE FALSE Chr1 358902      3
# 358903   0 Chr1 358903    N   0 0 0 0 0 0      0 FALSE FALSE Chr1 358903      2
# 358904   0 Chr1 358904    N   0 0 0 0 0 0      0 FALSE FALSE Chr1 358904      2
# 358905   0 Chr1 358905    N   1 0 1 0 0 0      0 FALSE FALSE Chr1 358905      2
# 358906   0 Chr1 358906    N   1 0 1 0 0 0      0 FALSE FALSE Chr1 358906      2
# 358907   0 Chr1 358907    N   1 1 0 0 0 0      0 FALSE FALSE Chr1 358907      1
# 358908   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358909   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358910   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358911   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358912   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358913   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358914   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
# 358915   0 <NA>     NA <NA>   0 0 0 0 0 0      0 FALSE FALSE <NA>     NA      0
```

A position is NA if and only if it has zero coverage, all 7:

```
na.implies.zero <- logical(7)
nonz.implies.nonna <- logical(7)
for(i in 1:7){
  na.implies.zero[i] <- all(snp.tables.chr1q[[i]]$rawCov[is.na(snp.tables.chr1q[[i]]$Ref)]==0)
  nonz.implies.nonna[i] <- !any(is.na(snp.tables.chr1q[[i]]$Ref[snp.tables.chr1q[[i]]$rawCov>0]))
}
all(na.implies.zero)

# [1] TRUE

all(nonz.implies.nonna)

# [1] TRUE
```

A closer look. Here's a representation of positions where one but not all strains show NA; as expected, NA positions show zero coverage; non-NA positions have non-zero coverage, non-NA positions agree on the reference nuc.

```
# places where one but not all have NA:
onena <- logical(chr1.len)
for (i in 1:6){
  for(j in i:7){
    onena <- onena | xor(is.na(snp.tables.chr1q[[i]]$Ref), is.na(snp.tables.chr1q[[j]]$Ref))
  }
}
nanum1 <- sum(onena); nanum1

# [1] 663

# add the few positions that are "N" in all strains (most N's are NA in at least one strain)
for(i in 1:7){
  onena[which(snp.tables.chr1q[[i]]$Ref == 'N')] <- TRUE
}
nanum2 <- sum(onena); nanum2

# [1] 669

# and (for visualization) add a few positions before & after big gap
onena[c(358900:358902,370252:370261)] <- TRUE
nanum3 <- sum(onena); nanum3

# [1] 675

# build a data.frame to display this;
gapulate <- function(mask, snp.tab = snp.tables.chr1q){
  # first, pack ref nucs from all 7 strains together (with '-' for NA)
  refs <- character(sum(mask))
  for(i in 1:7){
    tmp <- snp.tab[[i]]$Ref[mask]
    tmp[is.na(tmp)] <- '-'
    refs <- paste(refs, tmp, sep='')
  }
  df.mask <- data.frame(ref=refs, row.names=(1:nrow(snp.tab[[1]]))[mask], stringsAsFactors=F)
  # then append coverage; zeros turn out to match NAs
  for(i in 1:7){
    df.mask <- cbind(df.mask, snp.tab[[i]]$rawCov[mask])
    names(df.mask)[i+1] <- names(snp.tab)[i]
  }
  return(df.mask)
}
summary.onena <- gapulate(onena)
# show a few (1st/last few, all N's):
show.first <- 1:22
show.gap   <- unique(sort(c(377:404,grep('N',summary.onena$ref,fixed=T))))
show.last  <- nrow(summary.onena)+(-5:0)
summary.onena[show.first,]  # 1st few

#            ref 1007 1012 1013 1014 1015 3367 1335
# 1      TTTT-TT    1    1    1    1    0    1    1
# 2      CCCC-CC    3    1    1    1    0    1    1
# 3      CCCC-CC    4    1    1    1    0    1    1
# 4      AAAA-AA    5    1    1    1    0    1    1
# 5      AAAA-AA    7    1    1    1    0    1    1
# 6      GGGG-GG    7    1    1    1    0    1    2
# 7      AAAA-AA   10    1    1    1    0    1    9
# 8      GGGG-GG   12    4    1    1    0    1   11
# 9      TTTT-TT   13    8    1    1    0    1   15
# 10     CCCC-CC   15   10    1    1    0    1   20
# 11     GGGG-GG   16   12    1    1    0    1   22
# 12     AAAA-AA   16   12    1    1    0    1   26
# 13     AAAA-AA   16   15    1    1    0    1   26
# 14     GGGG-GG   17   15    1    1    0    1   28
# 15     TTTT-TT   17   16    1    1    0    1   30
# 16     AAAA-AA   17   16    1    1    0    1   30
# 17     GGGG-GG   17   17    1    1    0    1   33
# 18     TTTT-TT   18   17    1    2    0    1   33
# 19     TTTT-TT   19   18    3    3    0    3   34
# 20     TTTT-TT   20   19    3    3    0    3   35
# 958    AAAAA-A   51   82   36   35   73    0  143
# 20602 TT-TTTT    28   43    0   21   35    1   50

summary.onena[show.gap,]    # flanks of big gap + all N's
```

```
#             ref 1007 1012 1013 1014 1015 3367 1335
# 358744 T-TTTTT    1    0    1    2    4    4    4
# 358900 TTTTTTT    8   12   13   11    3    9    4
# 358901 GGGGGGG    8   12   13   10    3    9    4
# 358902 TTTTTTT    7   11   12    9    3    9    4
# 358903 NNNNNNN    7   10   10    9    2    8    4
# 358904 NNNNNNN    7    8    9    8    2    5    4
# 358905 NNNNNNN    7    6    5    8    2    3    4
# 358906 NNNNNNN    7    4    3    8    2    3    3
# 358907 NNNNNNN    5    2    3    7    1    3    3
# 358908 NNNN-NN    3    1    3    5    0    3    1
# 358909 NNNN-N-    2    1    3    3    0    1    0
# 358910 N-NN---    1    0    1    1    0    0    0
# 358911 ---N---    0    0    0    1    0    0    0
# 358912 ---N---    0    0    0    1    0    0    0
# 370249 -N-N---    0    1    0    1    0    0    0
# 370250 -N-N-N-    0    2    0    1    0    1    0
# 370251 -NNN-N-    0    2    2    2    0    2    0
# 370252 -NNNNN-    0    3    2    3    1    2    0
# 370253 -NNNNN-    0    3    5    3    1    3    0
# 370254 -NNNNN-    0    3    5    4    1    4    0
# 370255 -NNNNNN    0    3    7    4    2    5    1
# 370256 -NNNNNN    0    6    8    4    2    7    2
# 370257 -NNNNNN    0    8    8    4    3    8    2
# 370258 NNNNNNN    1   11   10    4    5    9    6
# 370259 AAAAAAA    2   12   11    4    5   11   14
# 370260 AAAAAAA    2   13   12    4    7   11   20
# 370261 TTTTTTT    3   15   14    5    7   14   21
# 371397 ----CCC    0    0    0    0    1    4   25

summary.onena[show.last,]   # last few

#              ref 1007 1012 1013 1014 1015 3367 1335
# 2993252 --A--AA    0    0    1    0    0    3    6
# 2993253 --GGGGG    0    0    1    1    1    6   10
# 2993254 -AAAAAA    0    4    5    1    2   12   18
# 3042583 AA-AA-A    3    7    0    7    3    0   19
# 3042584 GG-GG-G    3    6    0    7    3    0   17
# 3042585 GG-GG-G    2    4    0    6    2    0   14

chr1.len

# [1] 3042585
```

Figuring out *where* the gaps are.

```
make.gap.tab <- function(nna.mask, snp.tab=snp.tables.chr1q){
  wna <- which(nna.mask)
  wna <- append(wna,wna[length(wna)]+999999999) #append "infinity"; simplifies end case in loop below
  #build a table of gaps (i.e., consecutive T's in mask)
  gap.table <- NULL
  g.start.i <- 1
  for(i in 2:length(wna)){
    if(wna[i]-wna[i-1]>1){
      g.start <- wna[g.start.i]
      g.end <- wna[i-1]
      g.len <- g.end-g.start+1
      gap.table <-rbind(gap.table, c(start=g.start, end=g.end, length=g.len))
      g.start.i <- i
    }
  }
  return(gap.table)
}
```

```
gap.tables <- vector('list',7)
gap.tables2 <- vector('list',7)
for(st in 1:7){
  gap.table <- make.gap.tab(is.na(snp.tables.chr1q[[st]]$Ref))
  gap.tables[[st]] <- gap.table
  # find largest pair of gaps
  g.max <- max(gap.table[,'length'])
  i.max <- which(gap.table[,'length'] == g.max)
  g.max2 <- max(gap.table[,'length'][-i.max])
  i.max2 <- which(gap.table[,'length'] == g.max2)
  gap.tables2[[st]] <- gap.table[c(i.max,i.max2),]
}
names(gap.tables) <- names(snp.tables.chr1q)
#gap.tables
names(gap.tables2) <- names(snp.tables.chr1q)
gap.tables2

# $`1007`
#        start    end length
# [1,] 358911 370257  11347
# [2,] 149452 149543     92
#
# $`1012`
#        start    end length
# [1,] 358910 370248  11339
# [2,] 149496 149543     48
#
# $`1013`
#        start    end length
# [1,] 358911 370250  11340
# [2,] 149478 149536     59
#
# $`1014`
#        start    end length
# [1,] 358913 370248  11336
# [2,] 149459 149546     88
#
# $`1015`
#        start    end length
# [1,] 358908 370251  11344
# [2,] 149454 149551     98
#
# $`3367`
#        start    end length
# [1,] 358910 370249  11340
# [2,] 149478 149527     50
#
# $`1335`
#        start    end length
# [1,] 358909 370254  11346
# [2,] 149461 149545     85
```

Defining "gaps" to be 1 or more consecutive NAs, the chunk above shows (on Chr 1) all 7 isolates have a single gap of about 11340, starting near 358910, while the next largest gap is less that 100 bp. Based on the earlier look at NA vs N, *none* of the short gaps have N in the ref seq; they just happened to have *coverage* gaps in that strain. The gap in the ref seq will necessarily have zero counts except at its edges (can't align to NN...N), so what we really want is runs of positions that are NA in all 7, with some border of N's.

```
nna <- ! logical(chr1.len) #initialize to TRUE
for(i in 1:7){
  nna <- nna & (is.na(snp.tables.chr1q[[i]]$Ref) | (snp.tables.chr1q[[i]]$Ref=='N'))
}
sum(nna)

# [1] 11469

#there's only one big one:
```

```
all.n.na <- make.gap.tab(nna)
all.n.na

#         start     end length
#  [1,]  149496  149527     32
#  [2,]  176529  176532      4
#  [3,]  334292  334298      7
#  [4,]  335104  335107      4
#  [5,]  358903  370258  11356
#  [6,]  371402  371426     25
#  [7,]  831869  831877      9
#  [8,] 1416105 1416133     29
#  [9,] 2587051 2587053      3

# and none of the small ones show N's at borders (big gap marked by NNNNNNN @ 358903--370258):
nnab <- nna
nnab[358904:370257] <- FALSE # omit all but 1st/last of big gap; print rest
gapulate(nnab)

#             ref 1007 1012 1013 1014 1015 3367 1335
# 149496  -------    0    0    0    0    0    0    0
# 149497  -------    0    0    0    0    0    0    0
# 149498  -------    0    0    0    0    0    0    0
# 149499  -------    0    0    0    0    0    0    0
# 149500  -------    0    0    0    0    0    0    0
# 149501  -------    0    0    0    0    0    0    0
# 149502  -------    0    0    0    0    0    0    0
# 149503  -------    0    0    0    0    0    0    0
# 149504  -------    0    0    0    0    0    0    0
# 149505  -------    0    0    0    0    0    0    0
# 149506  -------    0    0    0    0    0    0    0
# 149507  -------    0    0    0    0    0    0    0
# 149508  -------    0    0    0    0    0    0    0
# 149509  -------    0    0    0    0    0    0    0
# 149510  -------    0    0    0    0    0    0    0
# 149511  -------    0    0    0    0    0    0    0
# 149512  -------    0    0    0    0    0    0    0
# 149513  -------    0    0    0    0    0    0    0
# 149514  -------    0    0    0    0    0    0    0
# 149515  -------    0    0    0    0    0    0    0
# 149516  -------    0    0    0    0    0    0    0
# 149517  -------    0    0    0    0    0    0    0
# 149518  -------    0    0    0    0    0    0    0
# 149519  -------    0    0    0    0    0    0    0
# 149520  -------    0    0    0    0    0    0    0
# 149521  -------    0    0    0    0    0    0    0
# 149522  -------    0    0    0    0    0    0    0
# 149523  -------    0    0    0    0    0    0    0
# 149524  -------    0    0    0    0    0    0    0
# 149525  -------    0    0    0    0    0    0    0
# 149526  -------    0    0    0    0    0    0    0
# 149527  -------    0    0    0    0    0    0    0
# 176529  -------    0    0    0    0    0    0    0
# 176530  -------    0    0    0    0    0    0    0
# 176531  -------    0    0    0    0    0    0    0
# 176532  -------    0    0    0    0    0    0    0
# 334292  -------    0    0    0    0    0    0    0
# 334293  -------    0    0    0    0    0    0    0
# 334294  -------    0    0    0    0    0    0    0
# 334295  -------    0    0    0    0    0    0    0
# 334296  -------    0    0    0    0    0    0    0
# 334297  -------    0    0    0    0    0    0    0
# 334298  -------    0    0    0    0    0    0    0
# 335104  -------    0    0    0    0    0    0    0
# 335105  -------    0    0    0    0    0    0    0
# 335106  -------    0    0    0    0    0    0    0
# 335107  -------    0    0    0    0    0    0    0
# 358903  NNNNNNN    7   10   10    9    2    8    4
# 370258  NNNNNNN    1   11   10    4    5    9    6
# 371402  -------    0    0    0    0    0    0    0
# 371403  -------    0    0    0    0    0    0    0
# 371404  -------    0    0    0    0    0    0    0
# 371405  -------    0    0    0    0    0    0    0
# 371406  -------    0    0    0    0    0    0    0
# 371407  -------    0    0    0    0    0    0    0
# 371408  -------    0    0    0    0    0    0    0
# 371409  -------    0    0    0    0    0    0    0
# 371410  -------    0    0    0    0    0    0    0
# 371411  -------    0    0    0    0    0    0    0
# 371412  -------    0    0    0    0    0    0    0
# 371413  -------    0    0    0    0    0    0    0
# 371414  -------    0    0    0    0    0    0    0
# 371415  -------    0    0    0    0    0    0    0
# 371416  -------    0    0    0    0    0    0    0
# 371417  -------    0    0    0    0    0    0    0
# 371418  -------    0    0    0    0    0    0    0
# 371419  -------    0    0    0    0    0    0    0
# 371420  -------    0    0    0    0    0    0    0
# 371421  -------    0    0    0    0    0    0    0
# 371422  -------    0    0    0    0    0    0    0
# 371423  -------    0    0    0    0    0    0    0
# 371424  -------    0    0    0    0    0    0    0
# 371425  -------    0    0    0    0    0    0    0
# 371426  -------    0    0    0    0    0    0    0
# 831869  -------    0    0    0    0    0    0    0
# 831870  -------    0    0    0    0    0    0    0
# 831871  -------    0    0    0    0    0    0    0
# 831872  -------    0    0    0    0    0    0    0
# 831873  -------    0    0    0    0    0    0    0
# 831874  -------    0    0    0    0    0    0    0
# 831875  -------    0    0    0    0    0    0    0
# 831876  -------    0    0    0    0    0    0    0
# 831877  -------    0    0    0    0    0    0    0
# 1416105 -------    0    0    0    0    0    0    0
# 1416106 -------    0    0    0    0    0    0    0
# 1416107 -------    0    0    0    0    0    0    0
# 1416108 -------    0    0    0    0    0    0    0
# 1416109 -------    0    0    0    0    0    0    0
# 1416110 -------    0    0    0    0    0    0    0
# 1416111 -------    0    0    0    0    0    0    0
# 1416112 -------    0    0    0    0    0    0    0
# 1416113 -------    0    0    0    0    0    0    0
# 1416114 -------    0    0    0    0    0    0    0
# 1416115 -------    0    0    0    0    0    0    0
# 1416116 -------    0    0    0    0    0    0    0
# 1416117 -------    0    0    0    0    0    0    0
# 1416118 -------    0    0    0    0    0    0    0
# 1416119 -------    0    0    0    0    0    0    0
# 1416120 -------    0    0    0    0    0    0    0
# 1416121 -------    0    0    0    0    0    0    0
# 1416122 -------    0    0    0    0    0    0    0
# 1416123 -------    0    0    0    0    0    0    0
# 1416124 -------    0    0    0    0    0    0    0
# 1416125 -------    0    0    0    0    0    0    0
# 1416126 -------    0    0    0    0    0    0    0
# 1416127 -------    0    0    0    0    0    0    0
# 1416128 -------    0    0    0    0    0    0    0
# 1416129 -------    0    0    0    0    0    0    0
# 1416130 -------    0    0    0    0    0    0    0
# 1416131 -------    0    0    0    0    0    0    0
# 1416132 -------    0    0    0    0    0    0    0
# 1416133 -------    0    0    0    0    0    0    0
# 2587051 -------    0    0    0    0    0    0    0
# 2587052 -------    0    0    0    0    0    0    0
# 2587053 -------    0    0    0    0    0    0    0
# so our real gap is:
the.gap <- all.n.na[which(all.n.na[,'length']==max(all.n.na[,'length'])),]
the.gap

#  start    end length
# 358903 370258  11356
```

4 Deserts

Also load the desert tables:

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

Structure of desert tables:

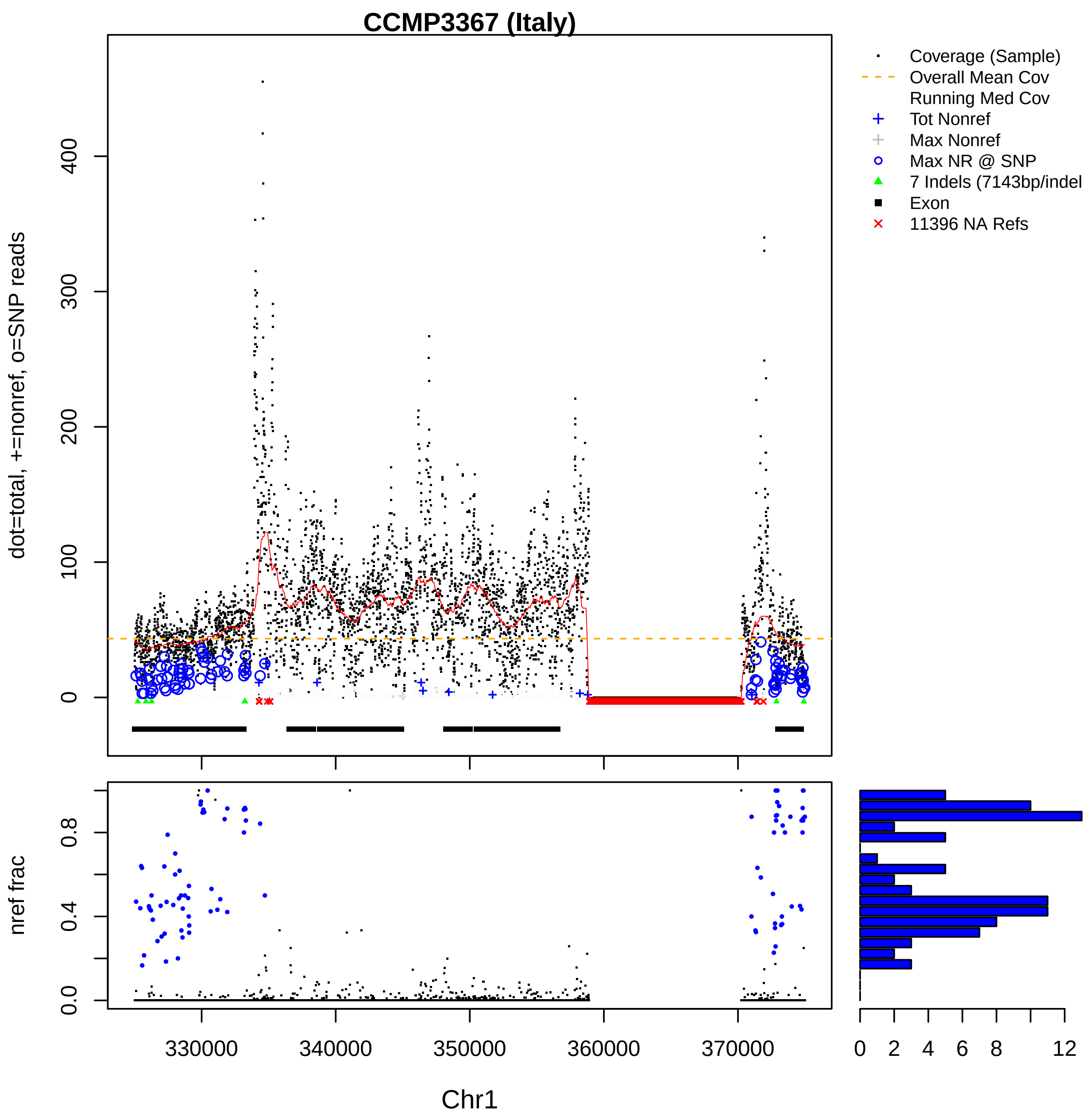
```
names(des)         # [1] "tp1007"  "tp1012"  "tp1013"  "tp1014"  "tp1015"  "thapsIT" "tp1335"

# [1] "tp1007"  "tp1012"  "tp1013"  "tp1014"  "tp1015"  "thapsIT" "tp1335"

names(des)[[6]] <- 'tp3367'  # override oldschool name
names(des[[1]])    # [1] "Chr1"   ...   "Chr24"

#  [1] "Chr1"      "Chr2"      "Chr3"      "Chr4"      "Chr5"      "Chr6"      "Chr7"
#  [8] "Chr8"      "Chr9"      "Chr10"     "Chr11a"    "Chr11b"    "Chr12"     "Chr13"
# [15] "Chr14"     "Chr15"     "Chr16a"    "Chr16b"    "Chr17"     "Chr18"     "Chr19a_19"
# [22] "Chr19b_31" "Chr19c_29" "Chr20"     "Chr22"     "Chr23"     "Chr24"

str(des[[1]][[1]])

#  num [1:74, 1:3] 1 8952 19297 91986 211997 ...
#  - attr(*, "dimnames")=List of 2
#   ..$ : NULL
#   ..$ : chr [1:3] "desert origin" "desert terminate" "Length"
```

Show desert containing the gap in all 7:

```
gapped.desert <- data.frame(id=names(des),chr='Chr1',start=0,end=0,len=0,pre.gap=0,post.gap=0,
                            stringsAsFactors=FALSE)
for(i in 1:7){
  hit <- des[[i]][[1]][,1] < the.gap['start'] & the.gap['end'] <  des[[i]][[1]][,2]
  gapped.desert[i,c('start','end','len')] <- des[[i]][[1]][hit,]
}
gapped.desert$pre.gap  <- the.gap['start'] - gapped.desert$start
gapped.desert$post.gap <- gapped.desert$end - the.gap['end']
gapped.desert

#       id  chr  start    end   len pre.gap post.gap
# 1 tp1007 Chr1 332775 371662 38886   26128     1404
# 2 tp1012 Chr1 333012 371708 38695   25891     1450
# 3 tp1013 Chr1 332712 372079 39366   26191     1821
# 4 tp1014 Chr1 331495 371708 40212   27408     1450
# 5 tp1015 Chr1 333127 371708 38580   25776     1450
# 6 tp3367 Chr1 330737 371708 40970   28166     1450
# 7 tp1335 Chr1 332426 371625 39198   26477     1367
```

The post-gap slice of the desert is short enough (< 2 Kb) that it probably would not qualify as a desert in its own right, but the pre-gap portion (> 25 Kb) certainly does, and is present in all 7. From a quick look at the "big n" table in ncsnps, it is among the largest of the 7-way shared deserts; I see only three larger ones (≈ 40 Kb each, on Chrs 9, 12, and 17). Fuzzy thought: CNVnator calls 1.5-2.0x coverage for much of the pre-gap region in Italy (but none of the others), making me wonder whether something structural like a mis-assembly, large repeat and/or recombination hotspots may have contributed to the sequencing gap and adjacent desert. In any case, while a curiosity, it doesn't seem to overturn any of our other interpretation.

```
seechunk(6,350000,25000,snp.tables=snp.tables.chr1q)
```



5 The Fig

Order of isolates, top-to-bottom in fig:

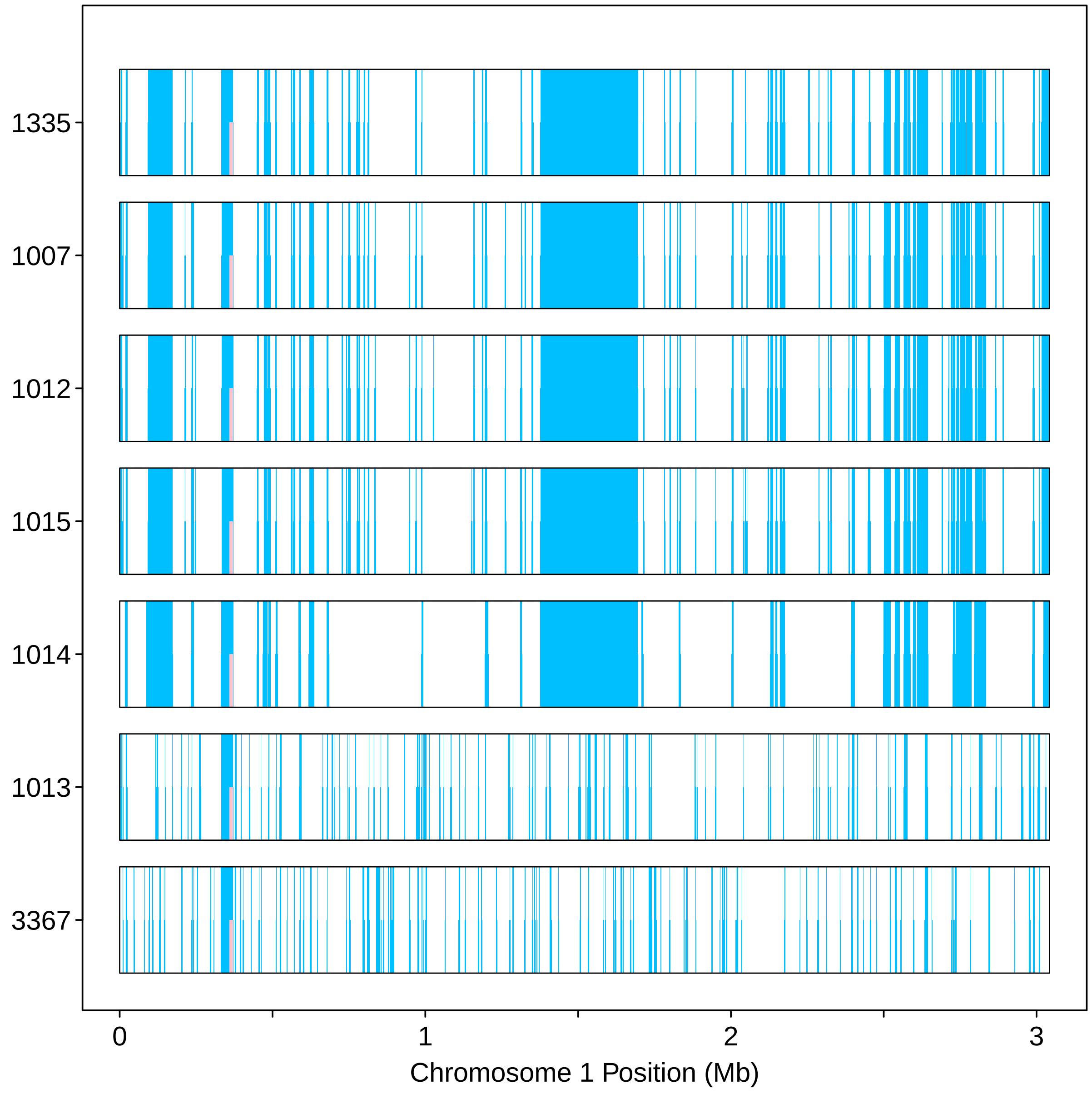
```
strain.order <- c(7,1,2,5,4,3,6)
names(des)[strain.order]

# [1] "tp1335" "tp1007" "tp1012" "tp1015" "tp1014" "tp1013" "tp3367"
```

Coordinate system: x coords are genomic positions, i.e., roughly 1..3e6 for Chr 1, printed about 6 inches wide; y coords are arbitrary, think of them as 300 units per inch. Drawing: Deserts near each other may visually merge due to the finite size of pixels. To see whether this is distorting the apparent landscape, this code can draw in two modes: first draw each bar as a wide "nondesert" (white) rectangle, overlaid by "desert" (blue) rectangles for each desert; Optionally, the top half of the bar is the reverse: white nondesert rectangles drawn over a blue background (so non-deserts separated by short deserts may blur together). This effect is strongest in H-clade where there are many short deserts (shorter than 3Kb, say), but overall I don't think it is misrepresenting the similarities/differences within/between L-/H-clade. A few sample figures illustrating this are shown below, too. Additional parameter "`min.desert`" prevents plotting of shorter deserts.

[2017-07-18: write relevant data to Fig2A-data.rda and moved draw.des.row and draw.des.fig to wlr.r so that I can generate Fig2A+B in one script. Various prototyping and exploration left here.]

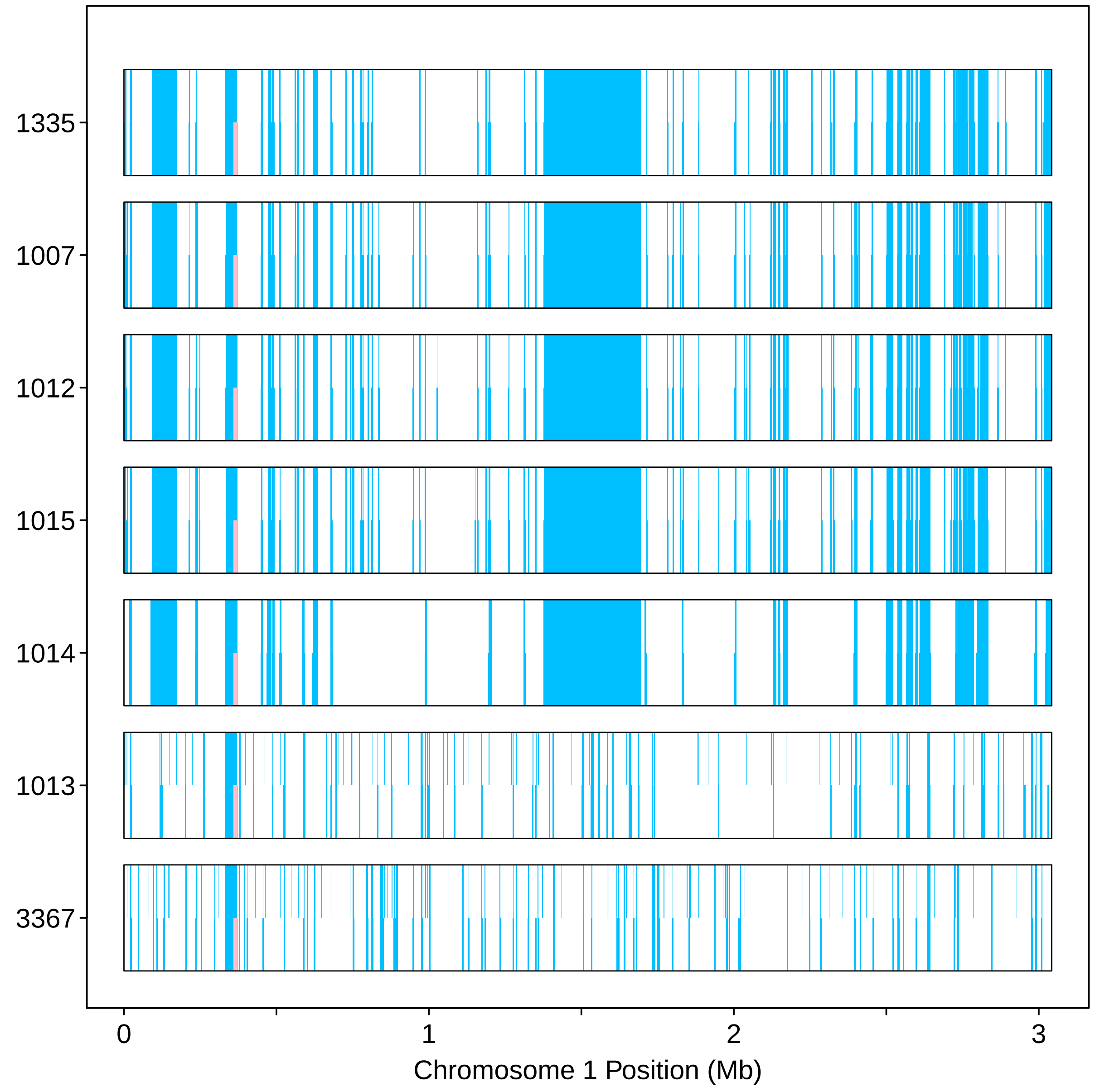
"Blur" due to fat pixels. (Again, botton half draws deserts over white background, then the gap (pink); top half draws non-des over blue background.)

```
draw.des.fig(des, all.n.na, draw.nondes=TRUE)
```



The effect is slightly ameliorated with short deserts hidden:

```
unlist(lapply(des,function(x){min(x[[1]][,'Length'])}))

# tp1007 tp1012 tp1013 tp1014 tp1015 tp3367 tp1335
#   3503   3389   2106   6330   3222   2348   3676

draw.des.fig(des, all.n.na, draw.nondes=TRUE, min.des=3000)
```



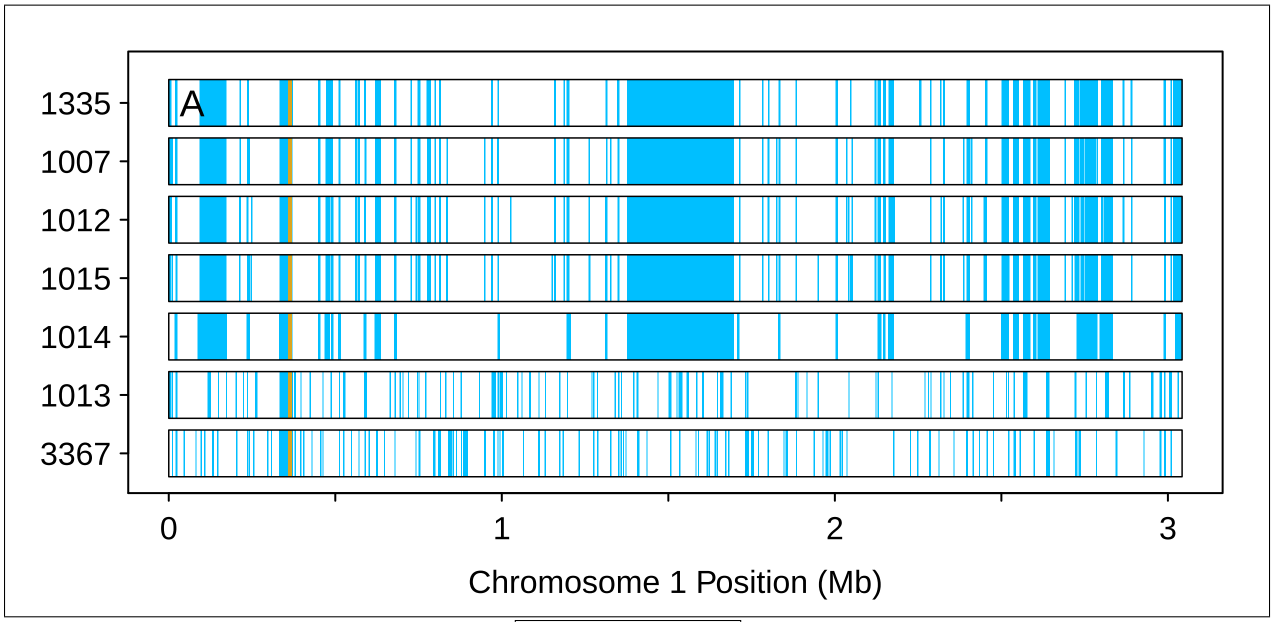
But, in summary, I think the pixel-blur efect is not so strong that I think we need to deviate from the simple draw-deserts-over-white-background model.

Figure prototype for paper:

```
if(FALSE){
  # color tests
  pdf(fpath('Fig2A-desert-distribution-figa'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'pink')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figb'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'deeppink')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figc'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'goldenrod')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figd'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'green')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-fige'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'darkgreen')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figf'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'yellow')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figg'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'orange')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figh'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'grey55')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figi'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'firebrick2')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figj'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'hotpink')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figk'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'firebrick3')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figl'), width=6.5, height=3.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'grey80',d.col='dodgerblue2')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figm'), width=6.5, height=2.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'gold',d.col='dodgerblue2')
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-fign'), width=6.5, height=2.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'gold',d.col='dodgerblue2',min.des=5000)
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figo'), width=6.5, height=2.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'gold',d.col='dodgerblue2',min.des=8000)
  dev.off()
  pdf(fpath('Fig2A-desert-distribution-figp'), width=6.5, height=2.1)
  draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'gold',d.col='dodgerblue2',min.des=10000)
  dev.off()
}
pdf(fpath('Fig2A-desert-distribution-figq'), width=6.5, height=2.1)
draw.des.fig(des, all.n.na, panel.label='A',gap.col = 'gold',d.col='dodgerblue2',min.des=10000,twotone='lightblue
dev.off()

# pdf
#   2
```

and shown juxtaposed with Fig2b for comparison as Fig 1. (Surrounding boxes just to make marginal space obvious; change fbox to mbox to remove.)
3/22/2018: Oh, Fig2B no longer exists separately, since fig2-glue.rnw now builds the full fig 2.

```
Fig2A.data.Description <- 'This .rda file contains the "gap table" all.n.na built by Fig2A-desert-distribution.rn
save(Fig2A.data.Description, all.n.na,file='Fig2A-data.rda',compress=FALSE)
```



[Panel B goes here]

Figure 1: Proposed caption: Attributes of SNP deserts for *T. pseudonana* isolates. A) SNP distributions across the 3 Mb of Chromosome 1 for the seven *T. pseudonana* isolates. Regions in blue have significantly low SNP density ("SNP deserts") based on a negative binomial model (Methods). Pink(???) region is a gap of known size in the reference sequence. The large region centered near 1.5Mb is a 320Kb SNP desert present in all L-isolates but neither H-isolate. B) SNP densities (SNP per base-pair—$\mu \pm 2\sigma$) in the 29 deserts that span at least 50Kb of the CCMP 1335 genome (blue) and the thirty regions surrounding these deserts (including deserts smaller than 50Kb; black).