Colon project statistical analyses

Analyses performed in this document:

1. Which signatures are ubiquitous and which are sporadic?
2. How does the mutation burden vary with age and with site in the colon for ubiquitous signatures?
3. When do copy number changes occur?
4. Driver discovery by dNdScv.
5. Test whether the global frequency of driver mutations is different between cancer and normal.

Load packages

```
suppressMessages(library("GenomicRanges"))
suppressMessages(library("nrmisc"))
suppressMessages(library("Biostrings"))
suppressMessages(library("lme4"))
suppressMessages(library("RColorBrewer"))
suppressMessages(library("seqinr"))
suppressMessages(library("MASS"))
suppressMessages(library("dndscv"))
```

```
## Warning: replacing previous import 'Biostrings::translate' by
## 'seqinr::translate' when loading 'dndscv'
```

```
suppressMessages(library("MutationTimeR"))
```

1) Which signatures are common and which are rare?

Read in input file of number of mutations due to each signature in each sample.

```
cc <- read.csv("model_input.txt", sep="\t", header=T, stringsAsFactors = T)
cc$sitecol <- as.character(cc$sitecol)
sigs <- colnames(cc)[13:38]

mycols <- c("grey", brewer.pal(n=9, name="Set1")[-6], brewer.pal(n=12, name="Set3"
)[-c(2,12)], brewer.pal(n=8, name="Dark2"))
mycols <- mycols[!grepl("X0", sigs)]
sigs <- sigs[!grepl("X0", sigs)] # remove the residual.
mycols <- mycols[1:length(sigs)]

sbssigs <- grep("sbs",sigs, value=T)
dbssigs <- grep("dbs",sigs, value=T)
idsigs <- grep("id",sigs, value=T)
large <- sigs[!sigs %in% c(sbssigs, dbssigs, idsigs)]
str(cc)
```

```
## 'data.frame':    445 obs. of  43 variables:
##  $ crypt     : Factor w/ 445 levels "HLS_1C_30_B5",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ site      : Factor w/ 4 levels "Ileum","Left",..: 3 3 3 3 3 3 3 3 4 4 ...
##  $ sex       : Factor w/ 3 levels "female","Female",..: 1 1 1 1 1 1 3 3 3 3 ...
##  $ age       : int  60 60 60 60 60 60 79 79 79 79 ...
##  $ med_vafs  : num  0.44 0.41 0.43 0.45 0.46 0.45 0.4 0.44 0.38 0.44 ...
##  $ med_depths: num  12 10 12 9 8 9 17 18 17 16 ...
##  $ vafdep    : num  5.28 4.1 5.16 4.05 3.68 4.05 6.8 7.92 6.46 7.04 ...
##  $ expt      : Factor w/ 18 levels "cancerNL","extrapaeds",..: 3 3 3 3 3 3 3 3
3 3 ...
##  $ library   : Factor w/ 2 levels "fragmentation",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ sitecol   : chr  "firebrick3" "firebrick3" "firebrick3" "firebrick3" ...
##  $ cancer.pt : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ patient   : Factor w/ 42 levels "300","312","HLS",..: 3 3 3 3 3 3 35 35 35 3
5 ...
##  $ sbs.X0    : num  140.2 77.5 53.3 30.8 84 ...
##  $ sbs.SBS1  : num  1240 1124 1037 1225 1273 ...
##  $ sbs.SBS5  : num  1666 1552 1319 1687 1910 ...
##  $ sbs.SBS18 : num  651 818 561 675 682 ...
##  $ sbs.SBS2  : num  0.00739 0.03818 0.012 0.00417 0.03133 ...
##  $ sbs.SBS13 : num  0.01331 0.00182 0.012 0.00209 1.18824 ...
##  $ sbs.N1    : num  0.0458 0.22 0.093 0.0751 0.0157 ...
##  $ sbs.N2    : num  227 455 485 403 616 ...
##  $ sbs.N3    : num  0.25 0.867 0.243 2.997 0.779 ...
##  $ sbs.N4    : num  0.3298 0.0582 1.4065 0.486 0.2685 ...
##  $ dbs.X0    : num  0.0037 0.01091 0.00675 0.00313 0.01678 ...
##  $ dbs.DBS2  : num  0.26 1.921 1.587 0.736 0.782 ...
##  $ dbs.DBS4  : num  0.277 2.042 1.688 0.783 0.832 ...
##  $ dbs.DBS6  : num  0.148 1.095 0.905 0.42 0.446 ...
##  $ dbs.DBS8  : num  0.00148 0.02545 0.00825 0.00313 0.0235 ...
##  $ dbs.DBS9  : num  0.207 1.53 1.264 0.586 0.623 ...
##  $ dbs.DBS11 : num  0.17 1.25 1.03 0.48 0.51 ...
##  $ id.X0     : num  0 0 0 0 0 ...
##  $ id.ID1    : num  0 0 0 0 0 ...
##  $ id.ID2    : num  0 0 0 0 0 ...
##  $ id.ID5    : num  0 0 0 0 0 ...
##  $ id.N2     : num  0 0 0 0 0 ...
##  $ id.N3     : num  0 0 0 0 0 ...
##  $ svs       : int  1 0 0 0 2 0 0 0 0 1 ...
##  $ chromamps : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ loh       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total     : num  3927 4036 3464 4028 4572 ...
##  $ sbstotal  : num  3784 3950 3404 3994 4483 ...
##  $ dbstotal  : num  1.06 7.86 6.49 3.01 3.22 ...
##  $ idtotal   : num  0 0 0 0 0 ...
##  $ largetotal: int  1 0 0 0 2 0 0 0 0 1 ...
```

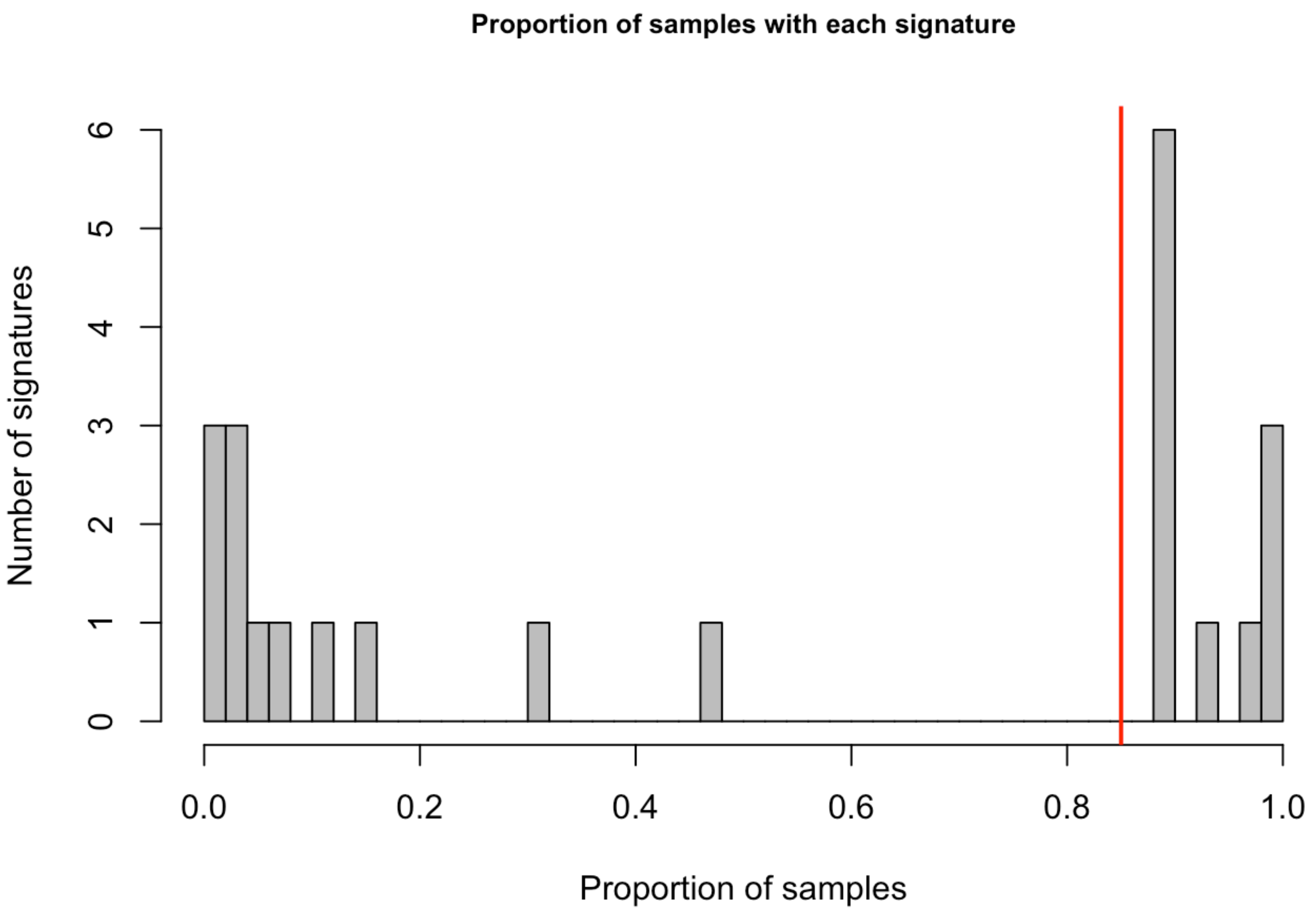
Define the presence or absence of a signature to count which samples it is in.

```
propn_with_sig <- c()
for (sig in sigs) {
  if (sig %in% large) {
    tc <- cc[,c(sig, "largetotal")]
  }
  if (sig %in% sbssigs) {
    tc <- cc[,c(sig, "sbstotal")]
  }
  if (sig %in% dbssigs) {
    tc <- cc[,c(sig, "dbstotal")]
  }
  if (sig %in% idsigs) {
    tc <- cc[,c(sig, "idtotal")]
  }
  tc <- tc[complete.cases(tc),]
  tc$prop <- tc[,1]/tc[,2]

  proppres <- nrow(tc[(tc[,1]>100 | tc$prop>0.05) & tc[,1]>0,])/nrow(tc)
  propn_with_sig <- c(propn_with_sig, proppres)
}
ps <- data.frame(cbind(sigs, propn_with_sig),stringsAsFactors = F)

ps
```

```
##          sigs      propn_with_sig
## 1   sbs.SBS1                   1
## 2   sbs.SBS5                   1
## 3  sbs.SBS18   0.997752808988764
## 4   sbs.SBS2 0.00449438202247191
## 5  sbs.SBS13 0.00449438202247191
## 6     sbs.N1  0.0269662921348315
## 7     sbs.N2    0.30561797752809
## 8     sbs.N3   0.051685393258427
## 9     sbs.N4  0.0269662921348315
## 10  dbs.DBS2   0.898876404494382
## 11  dbs.DBS4   0.898876404494382
## 12  dbs.DBS6   0.887640449438202
## 13  dbs.DBS8   0.112359550561798
## 14  dbs.DBS9    0.89438202247191
## 15 dbs.DBS11   0.889887640449438
## 16    id.ID1   0.937078651685393
## 17    id.ID2   0.889887640449438
## 18    id.ID5   0.964044943820225
## 19     id.N2   0.467415730337079
## 20     id.N3  0.0651685393258427
## 21       svs   0.148314606741573
## 22 chromamps 0.00674157303370787
## 23       loh  0.0224719101123595
```

Define those in which >85% of the crypts have the sig by these criteria as common, and the rest as rare.

```
hist(as.numeric(ps$propn_with_sig), col="grey", xlim=c(0,1), 40, main="Proportion
of samples with each signature",
     xlab="Proportion of samples", ylab="Number of signatures", cex.main=0.8)
abline(v=0.85, col="red", lwd=2)
```



```
ps[ps$propn_with_sig>0.85,]
```

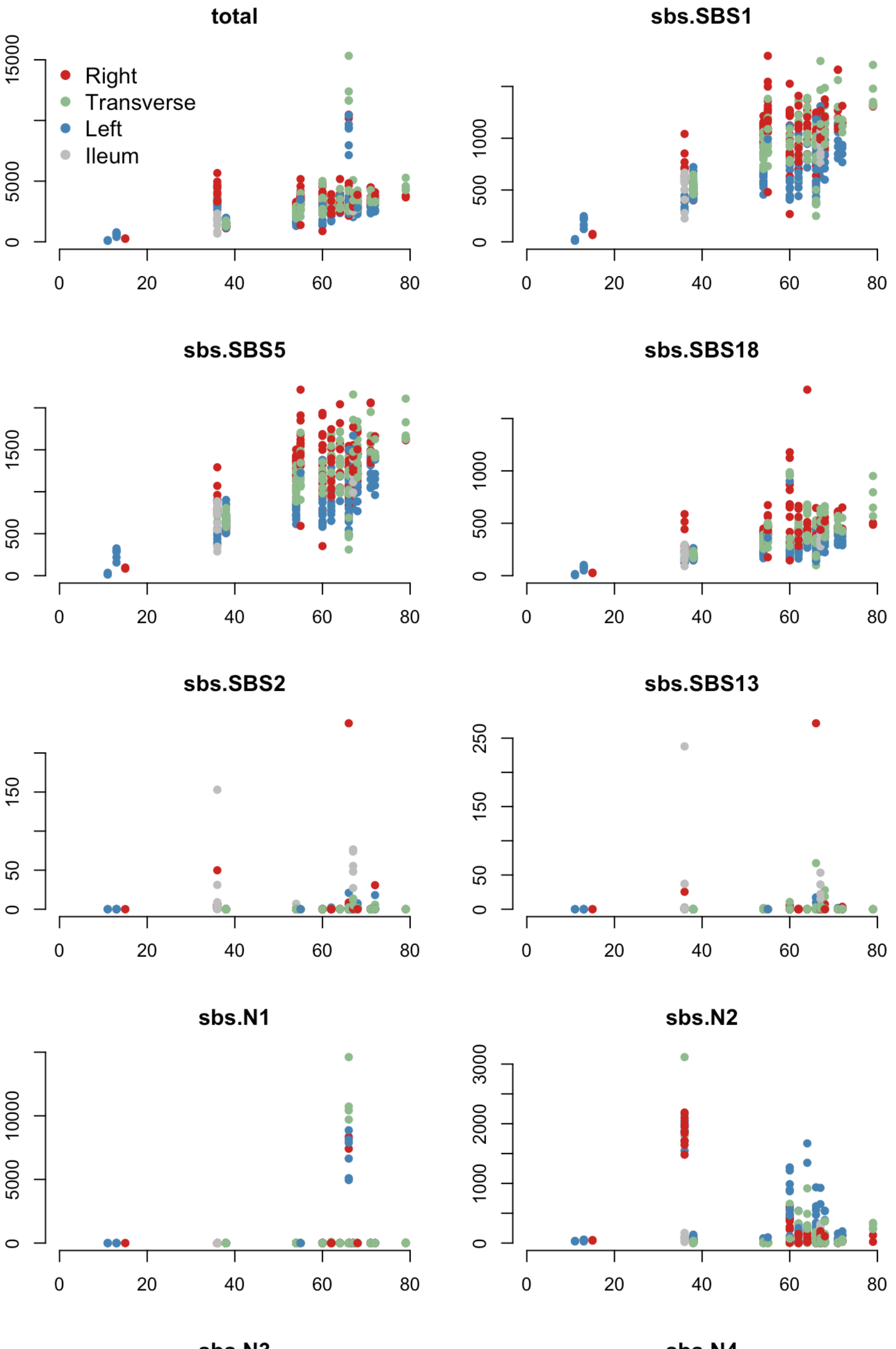
```
##          sigs    propn_with_sig
## 1    sbs.SBS1                 1
## 2    sbs.SBS5                 1
## 3   sbs.SBS18 0.997752808988764
## 10   dbs.DBS2 0.898876404494382
## 11   dbs.DBS4 0.898876404494382
## 12   dbs.DBS6 0.887640449438202
## 14   dbs.DBS9  0.89438202247191
## 15  dbs.DBS11 0.889887640449438
## 16     id.ID1 0.937078651685393
## 17     id.ID2 0.889887640449438
## 18     id.ID5 0.964044943820225
```

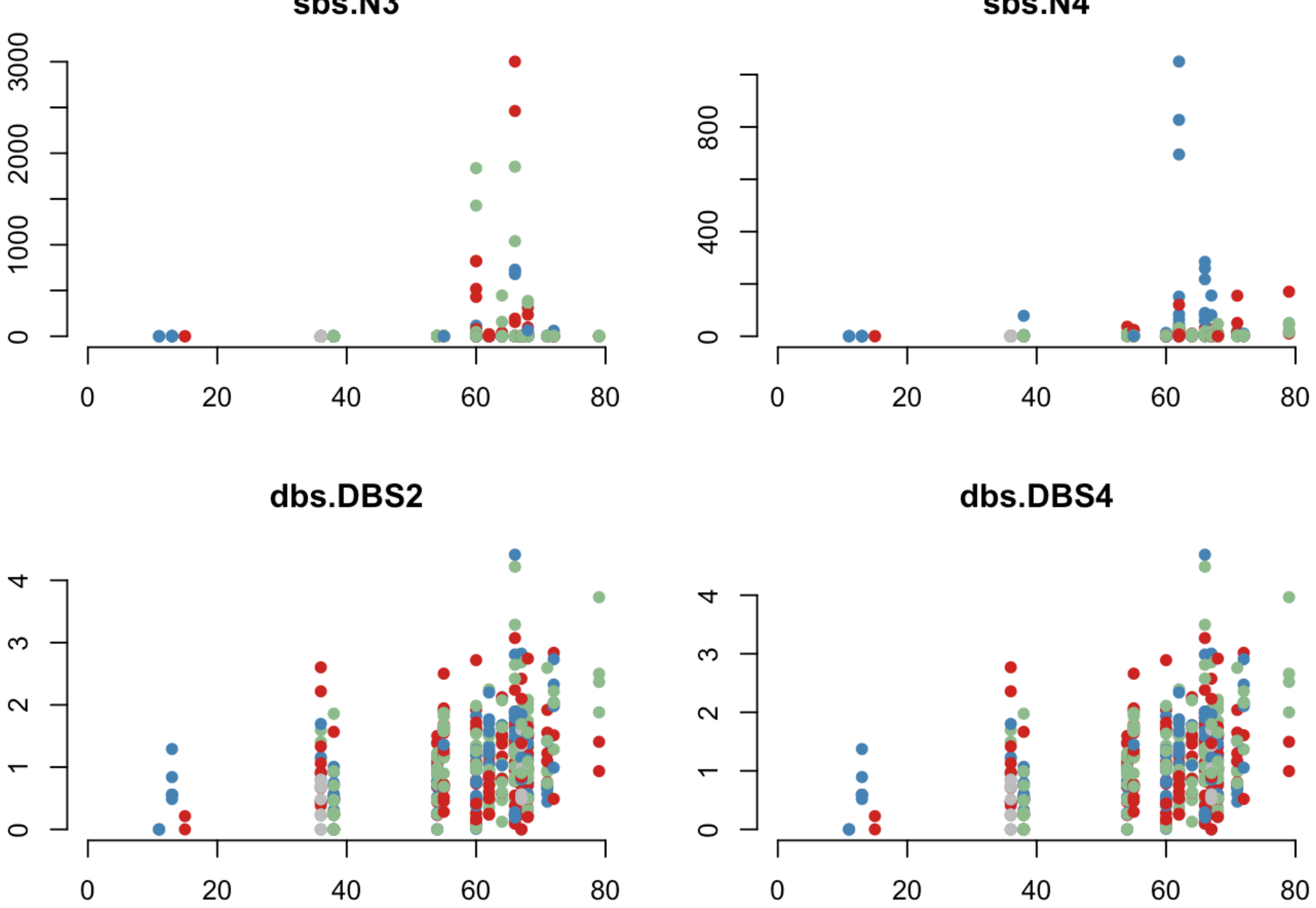
```
ubiquitous <- ps[ps$propn_with_sig>0.85,"sigs"]
```

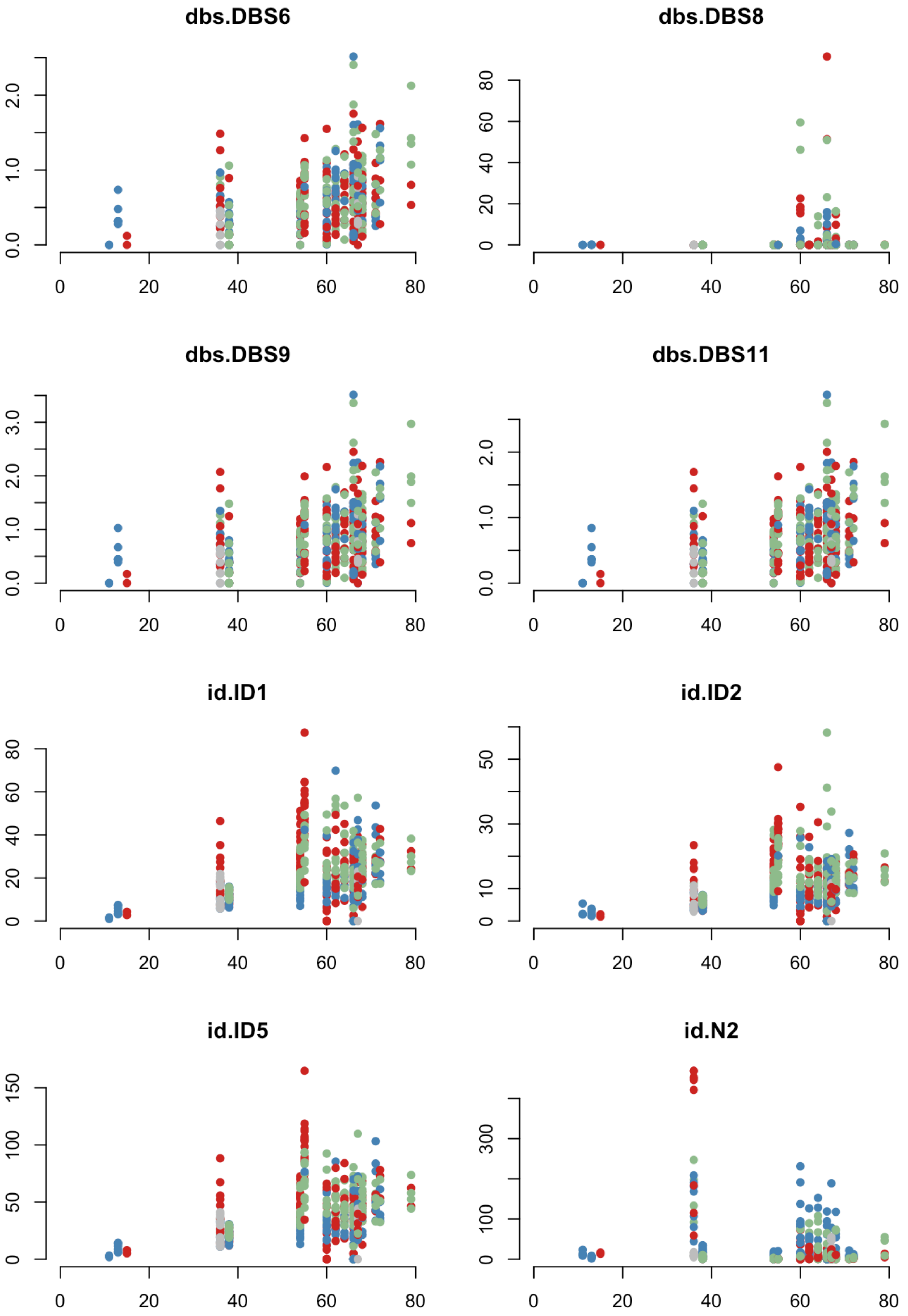
2) Mutation burden vs age and site for ubiquitous signatures with large mutation numbers: SBS1, SBS5, SBS18, ID1, ID2, ID5

Plot the burden of each signature in every crypt versus age, colouring by site.

```
par(mfrow=c(2,2))
par(mar=c(3,3,3,1))
i <- 1
for (sig in c("total", sigs)) {
  plot(cc[,sig] ~ cc[,"age"], col=cc$sitecol, pch=16, xlim=c(0,80), ylim=c(0, max(
cc[,sig])),
       ylab="Mutations", xlab="Patient age", main=sig, frame.plot=F)
  if (i==1) {
    legend("topleft", legend=c(unique(as.character(cc$site))), pch=16, col=unique(
cc$sitecol), bty="n", cex=1.2)
  }
  i <- i + 1
}
```


total
Right
Transverse
Left
Ileum
0
5000
15000
0
20
40
60
80
sbs.SBS1
0
500
1000
0
20
40
60
80
sbs.SBS5
0
500
1500
0
20
40
60
80
sbs.SBS18
0
500
1000
0
20
40
60
80
sbs.SBS2
0
50
150
0
20
40
60
80
sbs.SBS13
0
50
150
250
0
20
40
60
80
sbs.N1
0
5000
10000
0
20
40
60
80
sbs.N2
0
1000
2000
3000
0
20
40
60
80
sbs.N3
sbs.N4


sbs.N3
3000
2000
1000
0
0
20
40
60
80
sbs.N4
800
400
0
0
20
40
60
80
dbs.DBS2
4
3
2
1
0
0
20
40
60
80
dbs.DBS4
4
3
2
1
0
0
20
40
60
80



dbs.DBS6

dbs.DBS8

dbs.DBS9

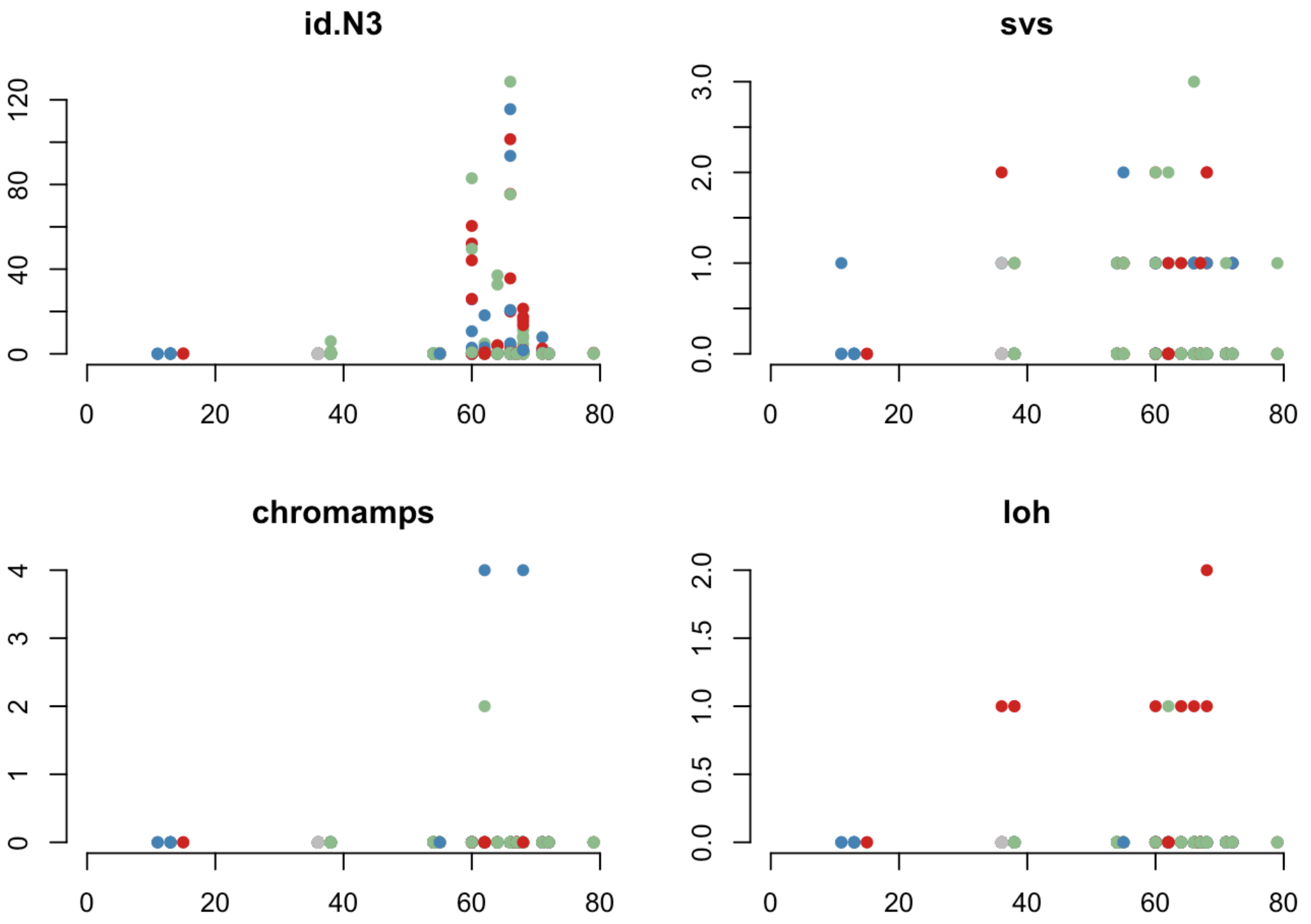
dbs.DBS11

id.ID1

id.ID2

id.ID5

id.N2



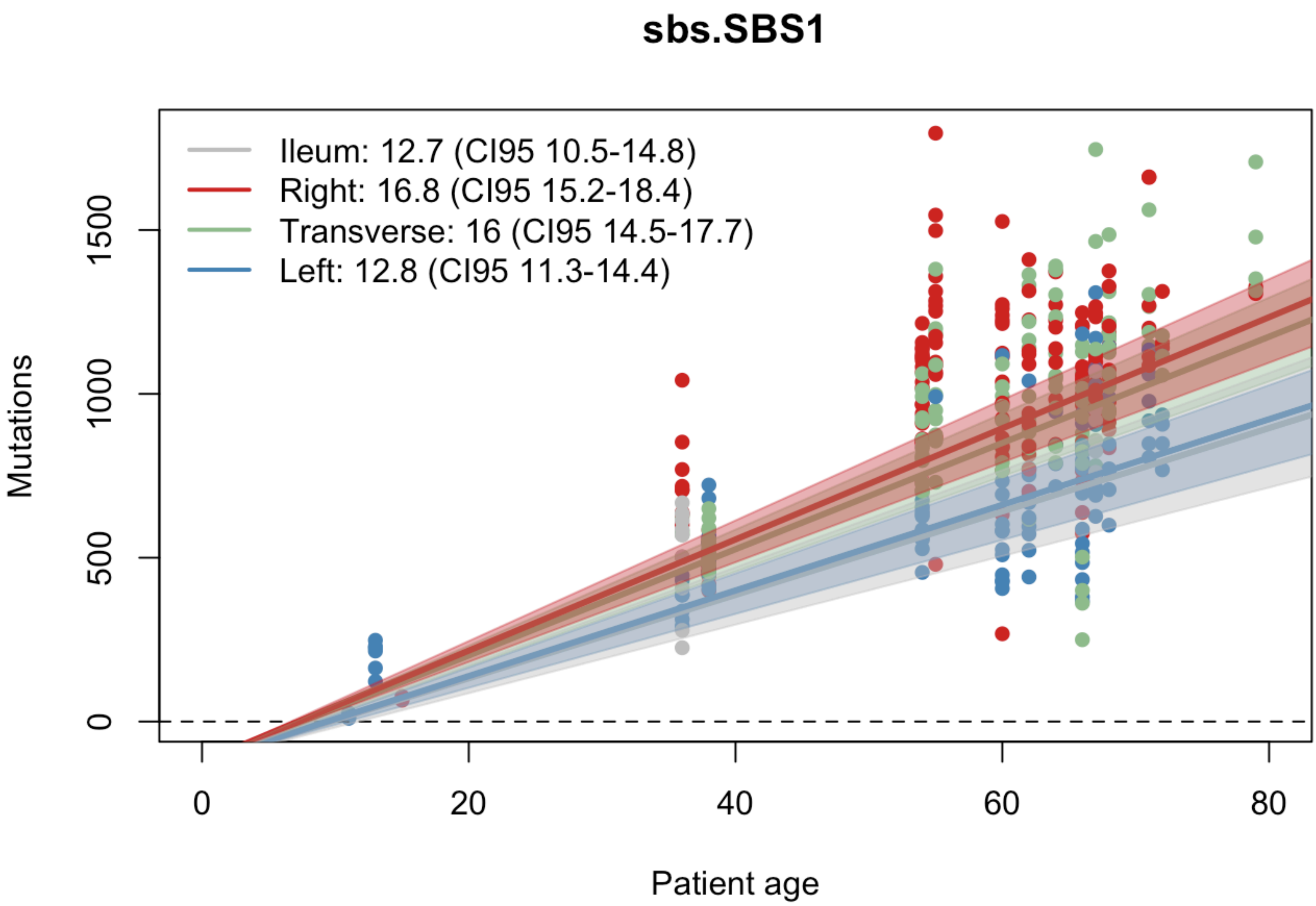
For the common signatures that have large numbers of mutations in most crypts I can fit a linear model.

Functions to fit each model:

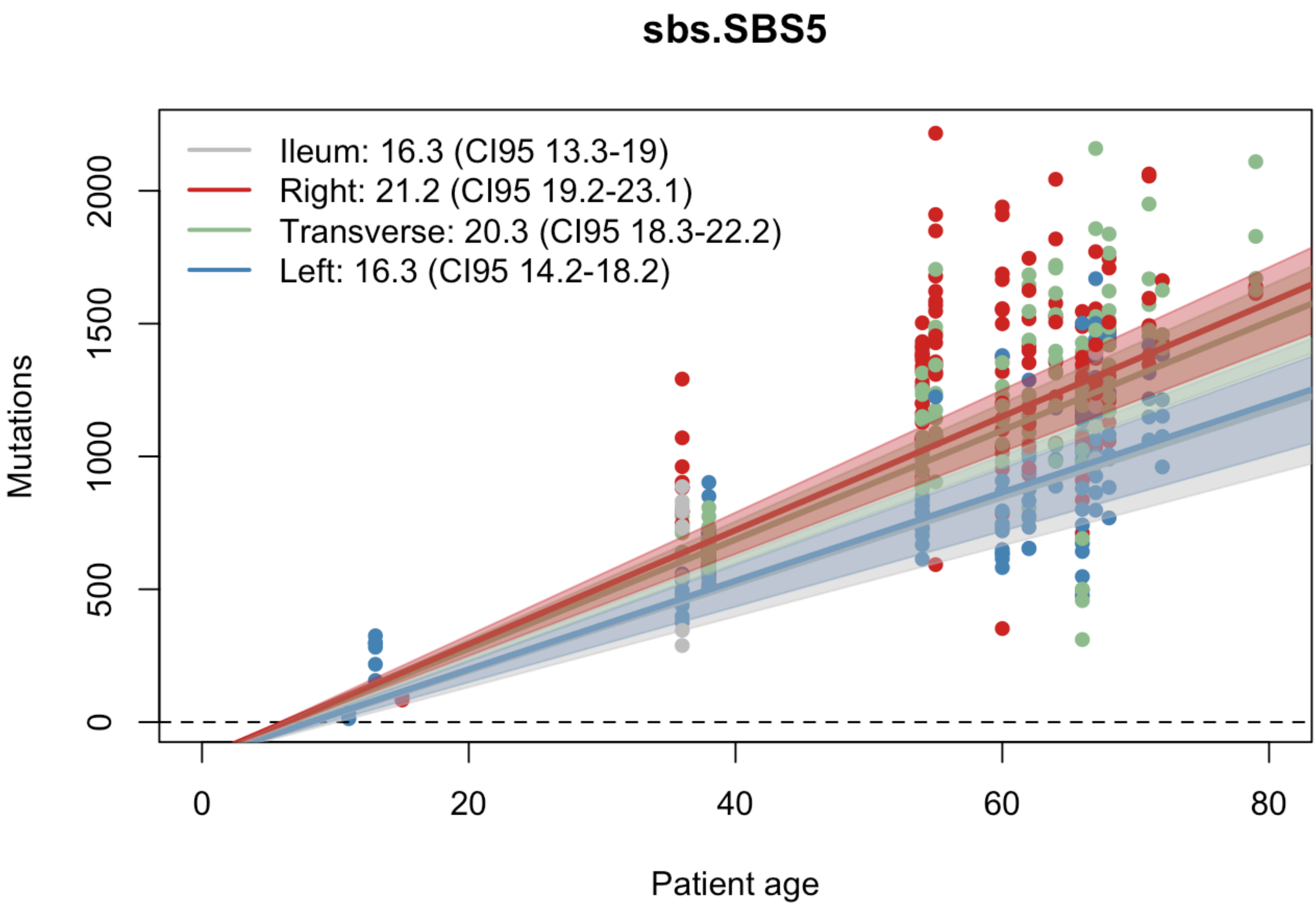
```
cc$patsite <- paste0(cc$patient, "_", cc$site)

# function needed for bootstrapping:
site_mutrate.fn <- function(model) {
  return(as.numeric(fixef(model)[c("age:siteIleum", "age:siteRight", "age:siteTransverse", "age:siteLeft")]))
}

# function to make colours transparent:
makeTransparent = function(..., alpha=0.4) {
  if(alpha<0 | alpha>1) stop("alpha must be between 0 and 1")
  alpha = floor(255*alpha)
  newColor = col2rgb(col=unlist(list(...)), alpha=FALSE)
  .makeTransparent = function(col, alpha) {
    rgb(red=col[1], green=col[2], blue=col[3], alpha=alpha, maxColorValue=255)
  }
  newColor = apply(newColor, 2, .makeTransparent, alpha=alpha)
  return(newColor)
}

fit_linear_model_fixed_site.fn <- function(signature) {
  print(signature)
  if (grepl("sbs", signature)) {
    formula <- paste0(signature, " ~ age:site + vafdep + ((age-1)|patient)")
  } else {
    formula <- paste0(signature, " ~ age:site + ((age-1)|patient)")  # don't include vafdep in indel model as accounted for it better.
  }

  tlmer <- lmer(formula, data=cc, REML=F)

  if (grepl("sbs", signature)) {
    formula_no_site <- paste0(signature, " ~ age + vafdep + ((age-1)|patient)")
  } else {
    formula_no_site <- paste0(signature, " ~ age + ((age-1)|patient)")  # don't include vafdep in indel model as accounted for it better.
  }

  tlmer_null <- lmer(formula_no_site, data=cc, REML=F)

  print(anova(tlmer, tlmer_null, test="Chisq"))

  # bootstrap
  bootlm1 <- bootMer(tlmer, site_mutrate.fn, nsim=1000, use.u=TRUE, type="parametric")
  cis <- as.data.frame(apply(bootlm1$t, 2, function(col) quantile(col, probs=c(0.025, 0.5, 0.975))))
  colnames(cis) <- c("age:siteIleum", "age:siteRight", "age:siteTransverse", "age:siteLeft")

  # plot
  plot(cc[,signature] ~ cc[,"age"], col=cc$sitecol, pch=16, xlim=c(0,80), ylab="Mutations", xlab="Patient age", main=signature)
  abline(h=0, lty=2)
  intercept <- summary(tlmer)$coef["(Intercept)", "Estimate"]
  abline(intercept, summary(tlmer)$coef["age:siteIleum", "Estimate"], col=cc$sitecol[cc$site=="Ileum"][1], lwd=3)
  abline(intercept, summary(tlmer)$coef["age:siteLeft", "Estimate"], col=cc$sitecol[cc$site=="Left"][1], lwd=3)
  abline(intercept, summary(tlmer)$coef["age:siteTransverse", "Estimate"], col=cc$sitecol[cc$site=="Transverse"][1], lwd=3)
  abline(intercept, summary(tlmer)$coef["age:siteRight", "Estimate"], col=cc$sitecol[cc$site=="Right"][1], lwd=3)
  # add on confidence intervals
  lciage90 <- cis["2.5%","age:siteTransverse"]*90 + intercept
  uciage90 <- cis["97.5%","age:siteTransverse"]*90 + intercept
  tcol=makeTransparent(cc$sitecol[cc$site=="Transverse"][1])
  polygon(x=c(0,90,90,0), y=c(intercept,lciage90, uciage90, intercept), col=tcol, border=tcol)
  lciage90 <- cis["2.5%","age:siteLeft"]*90 + intercept
  uciage90 <- cis["97.5%","age:siteLeft"]*90 + intercept
  tcol=makeTransparent(cc$sitecol[cc$site=="Left"][1])
  polygon(x=c(0,90,90,0), y=c(intercept,lciage90, uciage90, intercept), col=tcol, border=tcol)
  lciage90 <- cis["2.5%","age:siteRight"]*90 + intercept
  uciage90 <- cis["97.5%","age:siteRight"]*90 + intercept
  tcol=makeTransparent(cc$sitecol[cc$site=="Right"][1])
  polygon(x=c(0,90,90,0), y=c(intercept,lciage90, uciage90, intercept), col=tcol, border=tcol)
  lciage90 <- cis["2.5%","age:siteIleum"]*90 + intercept
  uciage90 <- cis["97.5%","age:siteIleum"]*90 + intercept
  tcol=makeTransparent(cc$sitecol[cc$site=="Ileum"][1])
  polygon(x=c(0,90,90,0), y=c(intercept,lciage90, uciage90, intercept), col=tcol, border=tcol)
  legend("topleft", lwd=2, col=c(cc$sitecol[cc$site=="Ileum"][1], cc$sitecol[cc$site=="Right"][1], cc$sitecol[cc$site=="Transverse"][1], cc$sitecol[cc$site=="Left"][1]),
  legend=c(paste0("Ileum: ", signif(cis["50%","age:siteIleum"], digits=3), " (CI95 ", signif(cis["2.5%","age:siteIleum"], digits=3), "-", signif(cis["97.5%","age:siteIleum"], digits=3), ")"),
         paste0("Right: ", signif(cis["50%","age:siteRight"], digits=3), " (CI95 ", signif(cis["2.5%","age:siteRight"], digits=3), "-", signif(cis["97.5%","age:siteRight"], digits=3), ")"),
         paste0("Transverse: ", signif(cis["50%","age:siteTransverse"], digits=3), " (CI95 ", signif(cis["2.5%","age:siteTransverse"], digits=3), "-", signif(cis["97.5%","age:siteTransverse"], digits=3), ")"),
         paste0("Left: ", signif(cis["50%","age:siteLeft"], digits=3), " (CI95 ", signif(cis["2.5%","age:siteLeft"], digits=3), "-", signif(cis["97.5%","age:siteLeft"], digits=3), ")")), bty="n")
}
```

```
totest <- c("sbs.SBS1", "sbs.SBS5", "sbs.SBS18", "id.ID1", "id.ID2", "id.ID5")

for (sig in totest) {
  fit_linear_model_fixed_site.fn(sig)
}
```

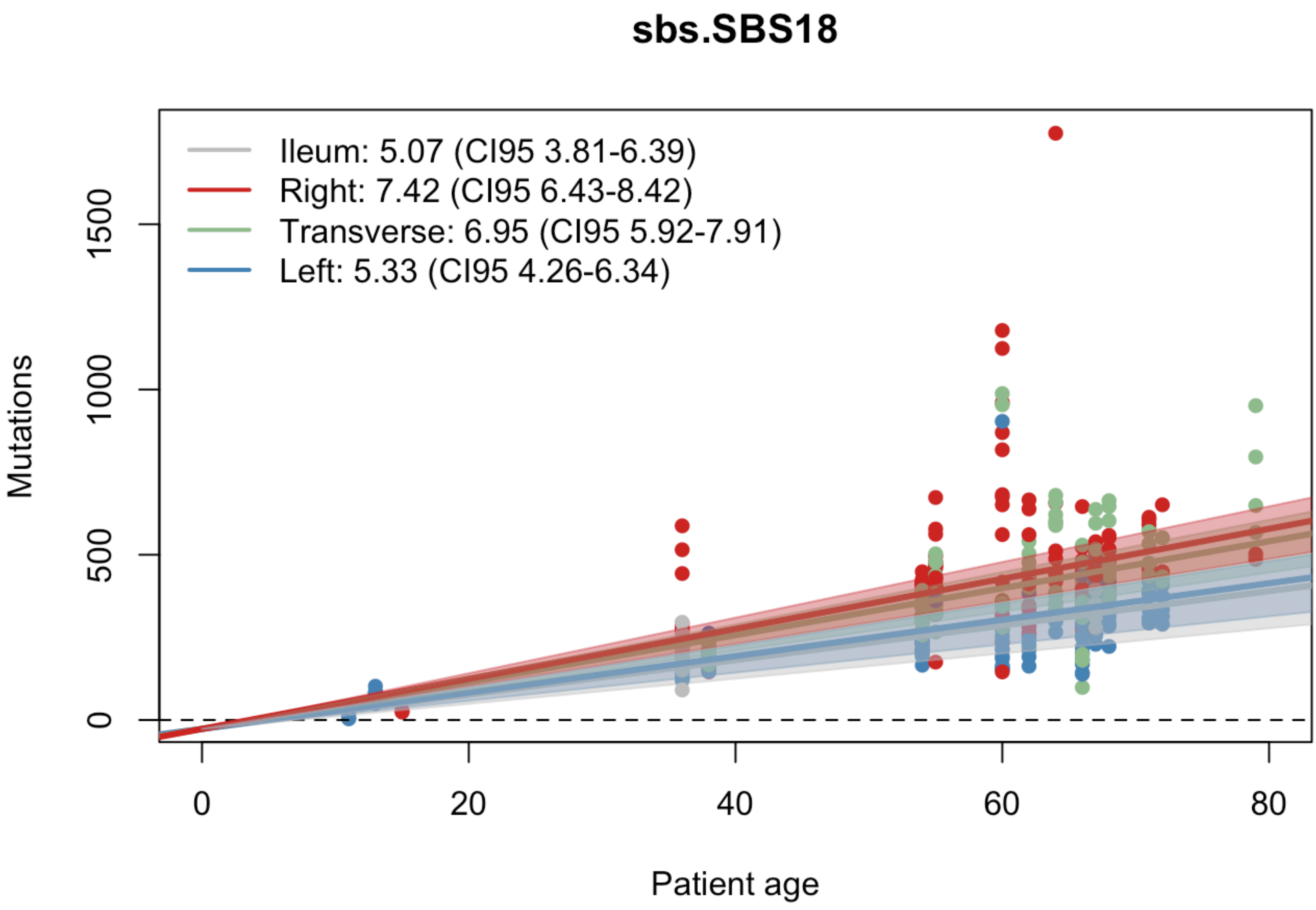
```
## [1] "sbs.SBS1"
## Data: cc
## Models:
## tlmer_null: sbs.SBS1 ~ age + vafdep + ((age - 1) | patient)
## tlmer: sbs.SBS1 ~ age:site + vafdep + ((age - 1) | patient)
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## tlmer_null  5 5970.0 5990.5 -2980.0   5960.0
## tlmer       8 5845.3 5878.1 -2914.7   5829.3 130.74      3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



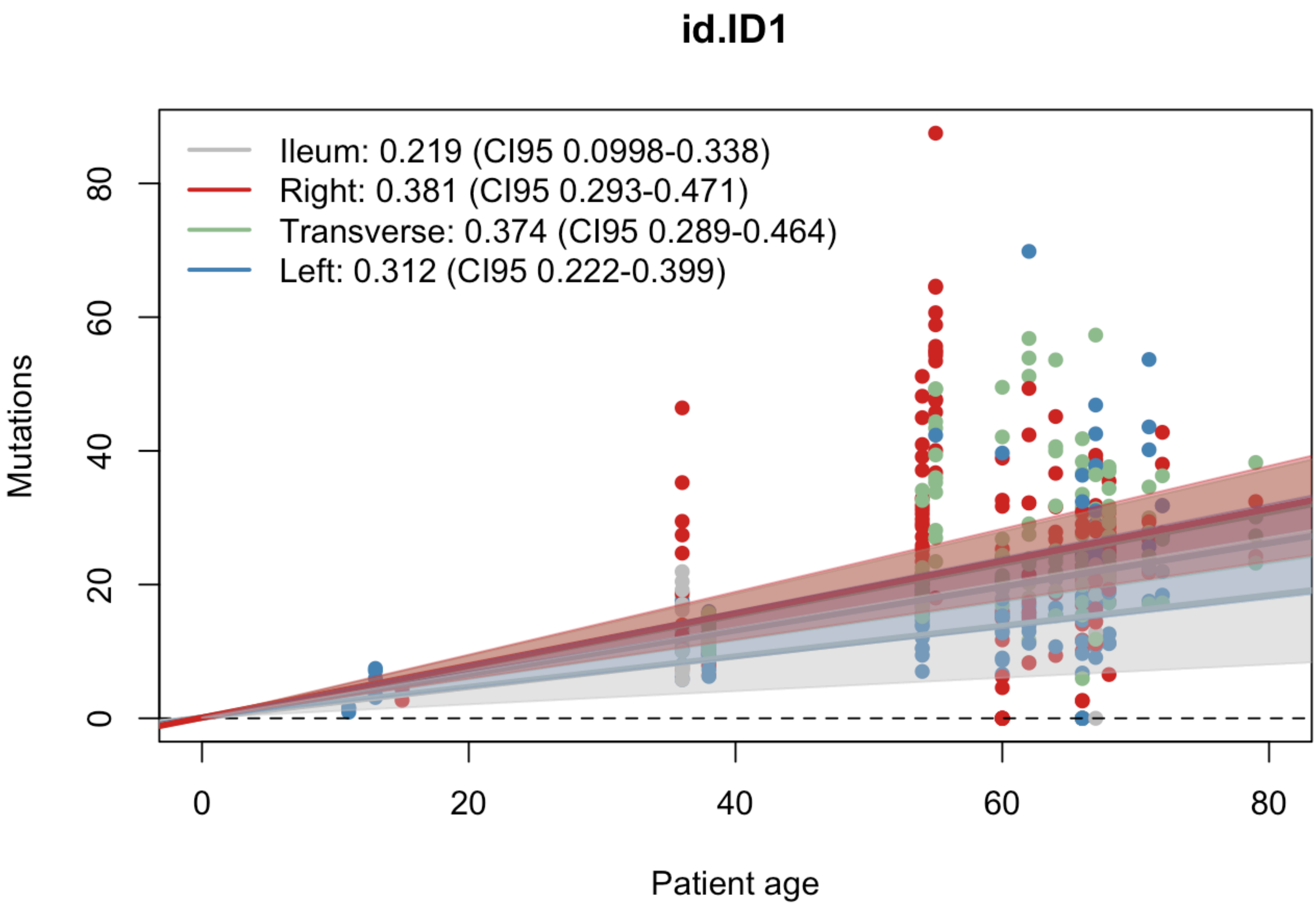
```
## [1] "sbs.SBS5"
## Data: cc
## Models:
## tlmer_null: sbs.SBS5 ~ age + vafdep + ((age - 1) | patient)
## tlmer: sbs.SBS5 ~ age:site + vafdep + ((age - 1) | patient)
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## tlmer_null  5 6174.3 6194.8 -3082.2   6164.3
## tlmer       8 6057.6 6090.4 -3020.8   6041.6 122.74      3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



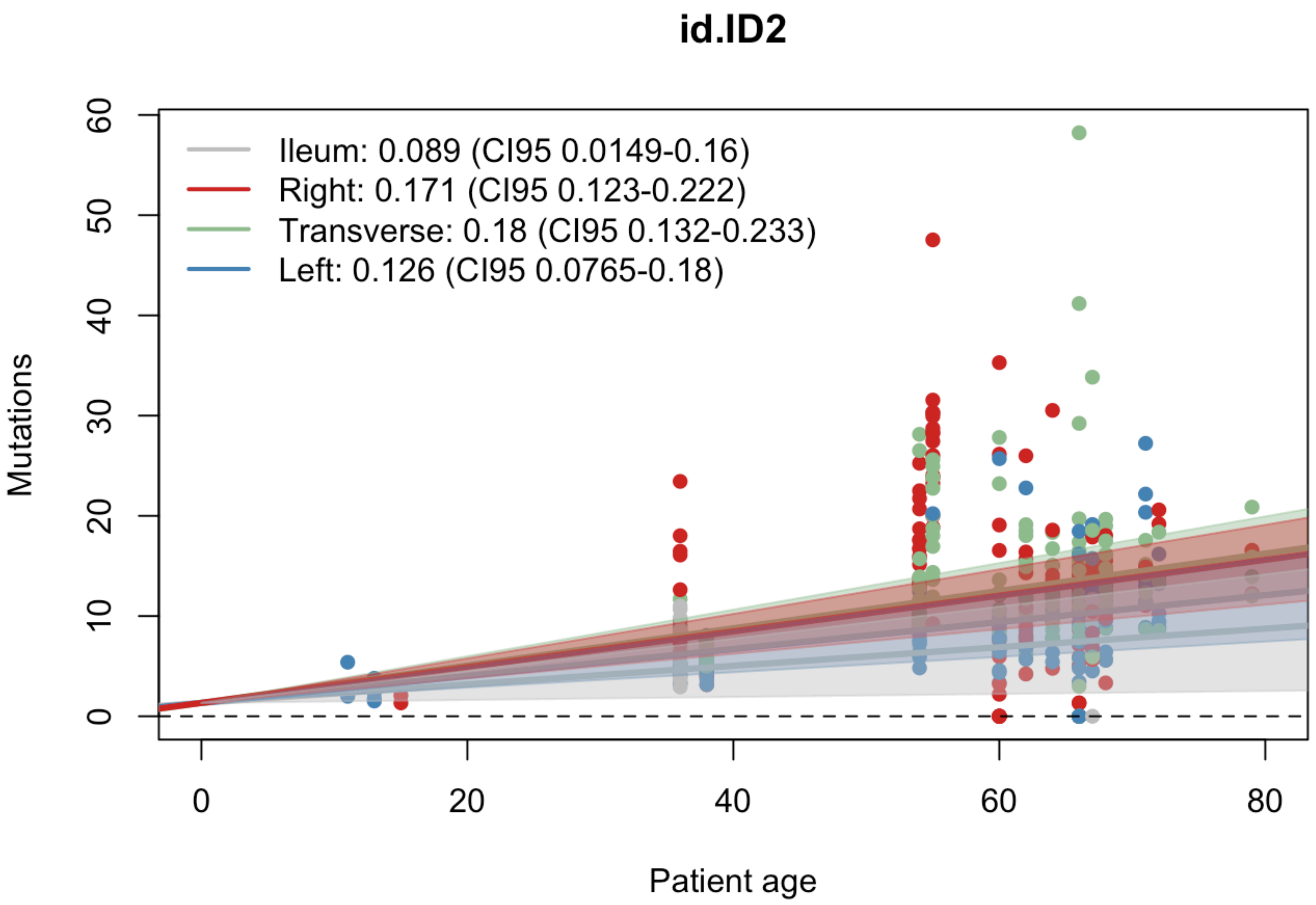
```
## [1] "sbs.SBS18"
## Data: cc
## Models:
## tlmer_null: sbs.SBS18 ~ age + vafdep + ((age - 1) | patient)
## tlmer: sbs.SBS18 ~ age:site + vafdep + ((age - 1) | patient)
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## tlmer_null  5 5512.9 5533.4 -2751.4   5502.9
## tlmer       8 5414.0 5446.8 -2699.0   5398.0 104.92      3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



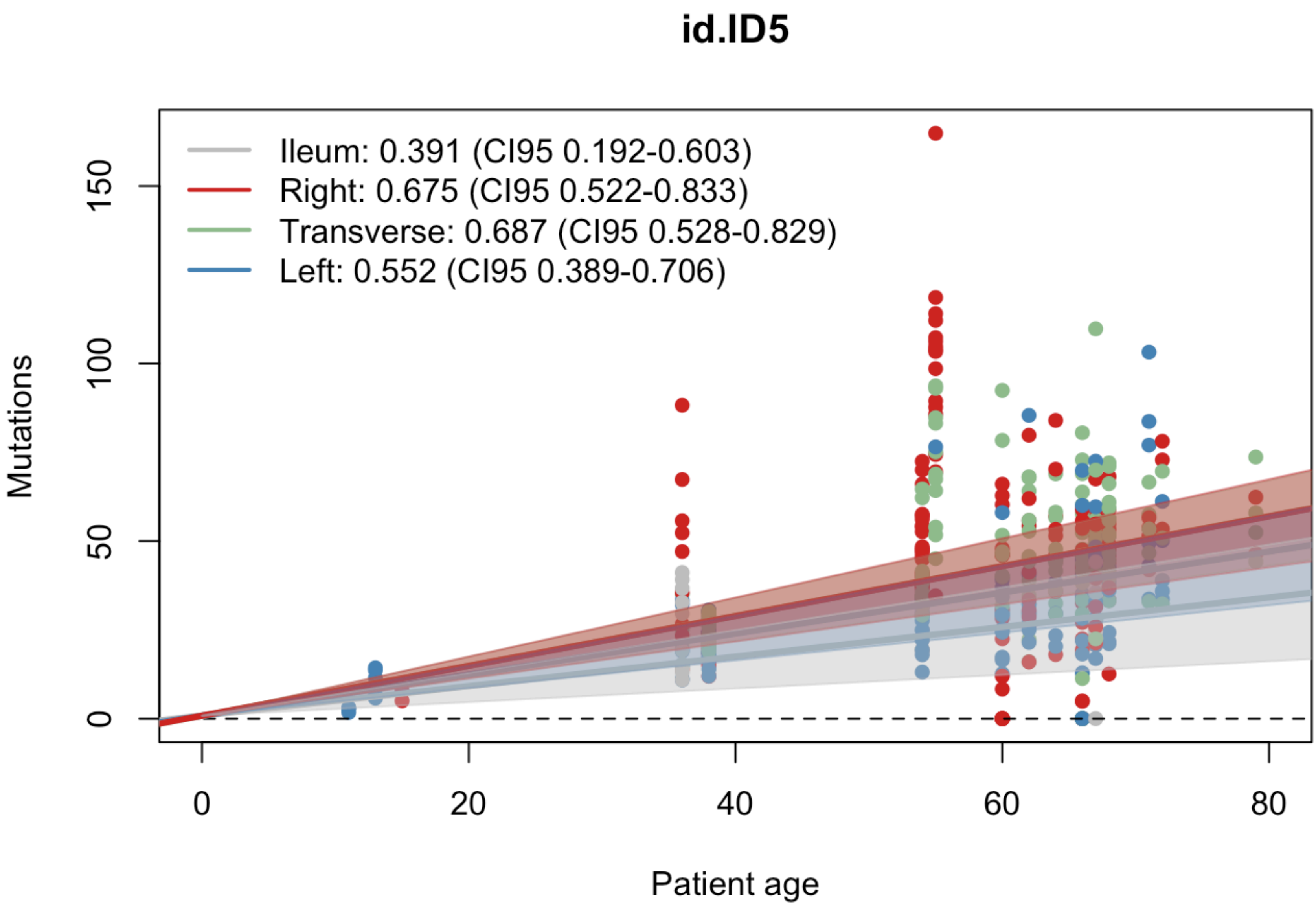
```
## [1] "id.ID1"
## Data: cc
## Models:
## tlmer_null: id.ID1 ~ age + ((age - 1) | patient)
## tlmer: id.ID1 ~ age:site + ((age - 1) | patient)
##             Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## tlmer_null   4 3265.7 3282.1 -1628.8   3257.7
## tlmer        7 3246.8 3275.5 -1616.4   3232.8 24.845      3  1.664e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
## [1] "id.ID2"
## Data: cc
## Models:
## tlmer_null: id.ID2 ~ age + ((age - 1) | patient)
## tlmer: id.ID2 ~ age:site + ((age - 1) | patient)
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## tlmer_null  4 2839.8 2856.1 -1415.9   2831.8
## tlmer       7 2816.8 2845.5 -1401.4   2802.8 28.977      3  2.265e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
## [1] "id.ID5"
## Data: cc
## Models:
## tlmer_null: id.ID5 ~ age + ((age - 1) | patient)
## tlmer: id.ID5 ~ age:site + ((age - 1) | patient)
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## tlmer_null  4 3764.4 3780.8 -1878.2   3756.4
## tlmer       7 3741.1 3769.8 -1863.5   3727.1 29.294      3  1.942e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



3) Time copy number changes.

Give path to copy number files.

```
cnpath <- "/Users/hl11/Documents/Projects/normal_colon/WGS_results/2018.07.02_sign
ature_results/hdp_sigs_to_pcawg_sigs_by_em/modelling_august_2018/MutationTimer/"

ascats <- list.files(path=paste0(cnpath, "/ascat_files"), pattern="ascat_ngs.summa
ry.csv")
```

For each of these ascat (copy number files), read in the relevant file of substitutions with their allele fractions. Then use the MutationTimeR package to estimate the time of the copy number change.

```r
dd <- data.frame()
for (asc in ascats) {
  patient <- sapply(strsplit(asc, '\\.'), "[[", 1)
  cav <- list.files(path=paste0(cnpath, "/filtered_caveman_files/"), pattern=paste0(patient, ".caveman_c.annot.vcf"))

  vcf <- readVcf(paste0(cnpath, "/filtered_caveman_files/", cav))
  bbin <- read.csv(paste0(cnpath, "/ascat_files/", asc), stringsAsFactors = F, header=F, row.names = 1)
  colnames(bbin) <- c("Chrom", "Start", "End", "Norm_tot", "Norm_min", "Tum_tot", "Tum_min")
  bbin$Norm_tot <- NULL
  bbin$Norm_min <- NULL
  bbin$Tum_maj <- bbin$Tum_tot - bbin$Tum_min

  vv <- read.csv(paste0(cnpath, "/filtered_caveman_files/", cav), sep="\t", header=T, skip = 136) # work out the clonality of the sample
  vv$pm <- as.numeric(sapply(strsplit(as.character(vv$TUMOUR), ":"), "[[", 10))

  dip <- bbin[bbin$Tum_tot==2 & bbin$Tum_maj==1,]  # only look at the diploid segments.
  dipsubs <- data.frame()
  for (j in 1:nrow(dip)) {
    td <- dip[j,]
    vv$X.CHROM<-as.character(vv$X.CHROM)
    vv$POS<-as.numeric(as.character(vv$POS))
    vd <- vv[vv$X.CHROM==td$Chrom & vv$POS >= td$Start & vv$POS<= td$End,]
    dipsubs <- as.data.frame(rbind(dipsubs, vd))
  }
  bbin$purity <- median(dipsubs$pm, col="grey", 50)*2

  bb <- GRanges(bbin$Chrom, IRanges(bbin$Start, bbin$End), major_cn=bbin$Tum_maj , minor_cn=bbin$Tum_min, clonal_frequency=bbin$purity)

  mt <- MutationTimeR::mutationTime(vcf, bb)

  aa <- as.data.frame(cbind(bbin, mt$T))
  aa <- aa[!is.na(aa$type) & !is.na(aa$time),]

  if (nrow(aa)>0) {
      aa$crypt <- patient
      dd <- as.data.frame(rbind(dd, aa))
      write.table(aa, paste0(cnpath, "/output/", patient, "_segment_times.txt"), sep="\t", col.names = T, row.names=F)
  }
}
```

Only keep the ones with >100 mutations per segment that we can time with high confidence.

```r
ee <- dd[dd$n.snv_mnv>100 & !is.na(dd$n.snv_mnv),]
```

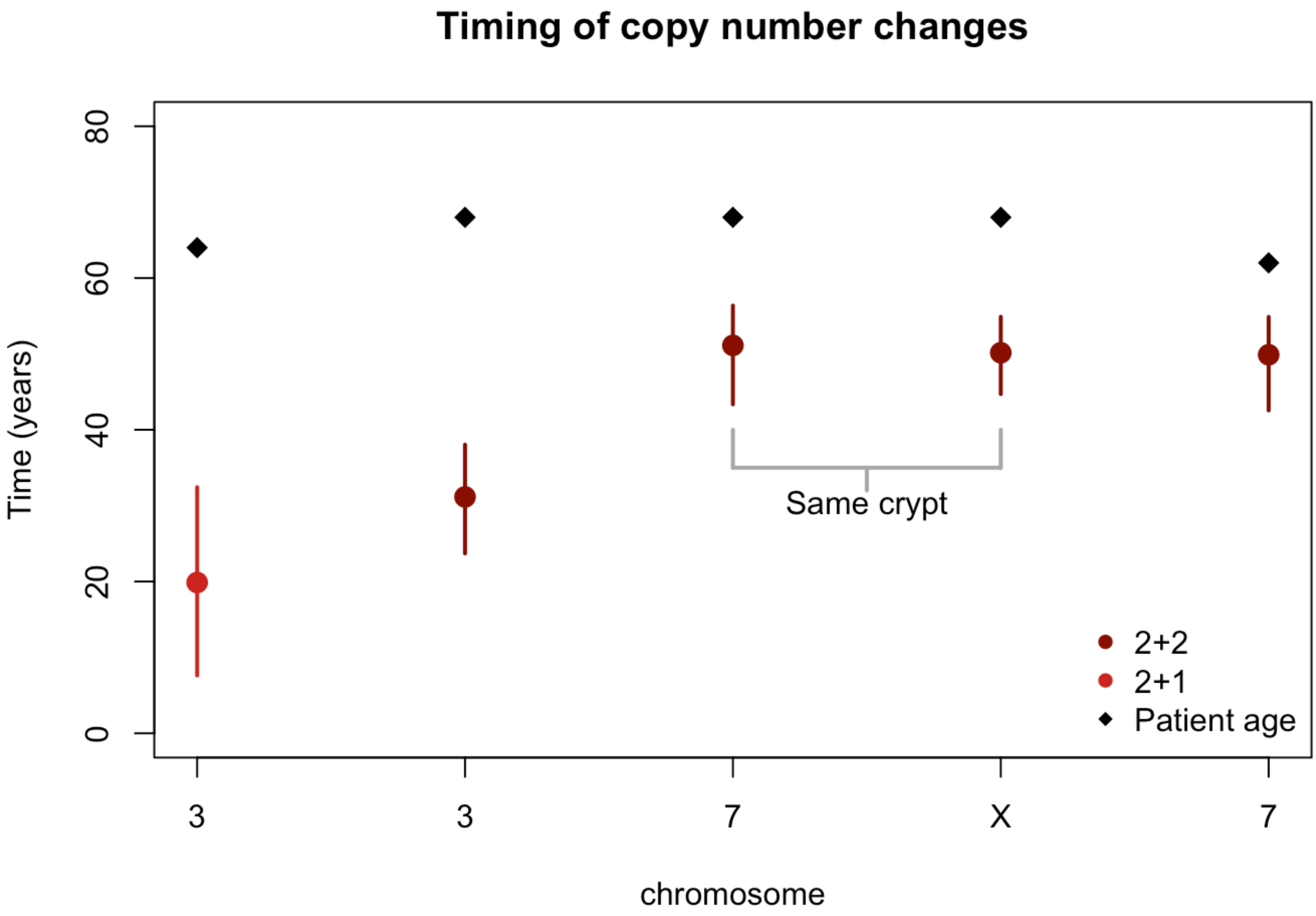
Now need to add on the ages.

```
ages <- read.csv(paste0(cnpath, "/age_info.txt"), sep="\t", header=T, stringsAsFac
tors = F)

ff <- merge(ee, ages, all.x=T, all.y=F)
ff$cnage <- ff$age*ff$time
ff$cnage_lo <- ff$age*ff$time.lo
ff$cnage_up <- ff$age*ff$time.up

ff[,c('crypt', "Chrom", "Start", "End", "Tum_tot", "Tum_min", "Tum_maj", "purity",
"type", "age", "cnage", "cnage_lo", "cnage_up", "n.snv_mnv")]
```

```
##       crypt Chrom   Start       End Tum_tot Tum_min Tum_maj purity
## 1 PD36814c6     3   60197 197908615       3       1       2   0.58
## 2 PD36814l7     3   60197 197908615       4       2       2   1.00
## 3 PD37226e5     7   24220 159127076       4       2       2   0.86
## 4 PD37226e5     X 2699555 154929412       4       2       2   0.86
## 5 PD37226m5     7   24220 159127076       4       2       2   0.86
##                     type age    cnage  cnage_lo cnage_up n.snv_mnv
## 1     Mono-allelic Gain  64 19.87235  7.623973 32.43707       184
## 2 Bi-allelic Gain (WGD)  68 31.15109 23.702204 38.04505       283
## 3 Bi-allelic Gain (WGD)  68 51.11889 43.349747 56.37321       164
## 4 Bi-allelic Gain (WGD)  68 50.15287 44.699405 54.89750       251
## 5 Bi-allelic Gain (WGD)  62 49.88222 42.542479 54.88502       127
```

Plot the age at whicht the copy number change occurs, showing the 95% confidence interval.

```
ff$colour <- "firebrick3"
ff$colour[ff$Tum_min==2] <- "darkred"

par(mfrow=c(1,1))
par(mar=c(4,4,4,1))
plot(ff$cnage, xaxt="n", xlab="chromosome", ylim=c(0, 80), ylab="Time (years)",
     col=ff$colour, pch=16, cex=1.5, main="Timing of copy number changes")
for (i in 1:nrow(ff)) {
  segments(x0=i , y0=ff$cnage_lo[i], x1=i , y1=ff$cnage_up[i], col=ff$colour[i], l
wd=2)
}
points(ff$age, col="black", pch=18, cex=1.5)
axis(side=1, at=c(1:nrow(ff)), labels = ff$Chrom)
legend('bottomright', pch=c(16,16,18), col=c("darkred", "firebrick3", "black"), le
gend=c("2+2", "2+1", "Patient age"), bty="n")
text(x=3.5, y=30, "Same crypt", col="black")
segments(x0 = 3, x1=4, y0=35, y1=35, col="darkgrey", lwd=2)
segments(x0 = 3, x1=3, y0=35, y1=40, col="darkgrey", lwd=2)
segments(x0 = 4, x1=4, y0=35, y1=40, col="darkgrey", lwd=2)
segments(x0 = 3.5, x1=3.5, y0=32, y1=35, col="darkgrey", lwd=2)
```



4) dNdS for driver mutation discovery.

Two analyses: a) Whole genome sequencing samples only, considering the whole genome. b) Combined genomes and targeted data, considering only 90 colorectal cancer genes curated from the literature that were included in our baitset.

Read in data.

```
comb <- read.csv("/Users/hl11/Documents/Projects/normal_colon/WGS_results/2018.03.
19_drivers/combined_dnds_input_targ_wgs.txt", sep="\t", header=T, stringsAsFactors
= F)
str(comb)
```

```
## 'data.frame':    23039 obs. of  10 variables:
##  $ sampleID: chr  "HLS_1C_30_B5" "HLS_1C_30_B5" "HLS_1C_30_B5" "HLS_1C_30_B5" .
..
##  $ chr     : chr  "1" "1" "1" "1" ...
##  $ pos     : int  65330632 155883970 158262062 158669869 200378047 203683366 24
8129571 6422622 14990481 60563086 ...
##  $ ref     : chr  "T" "TG" "G" "G" ...
##  $ mut     : chr  "G" "T" "A" "A" ...
##  $ source  : chr  "wgs" "wgs" "wgs" "wgs" ...
##  $ patient : chr  "HL" "HL" "HL" "HL" ...
##  $ patkey  : chr  "HL_1_65330632_T_G" "HL_1_155883970_TG_T" "HL_1_158262062_G_A
" "HL_1_158669869_G_A" ...
##  $ plus1   : int  65330633 155883971 158262063 158669870 200378048 203683367 24
8129572 6422623 14990482 60563087 ...
##  $ tonext  : int  155883970 158262062 158669869 200378047 203683366 248129571 6
422622 14990481 60563086 64039174 ...
```

```
wgs <- comb[comb$source=="wgs",]
```

Read in the set of colorectal cancer genes covered by the baitset.

```
genes <- read.csv("/Users/hl11/Documents/Projects/normal_colon/WGS_results/2018.03
.19_drivers/driver_genes_in_baitset.txt", sep="\t", header=F, stringsAsFactors = F
)
genes <- genes$V1
genes
```

```
##  [1] "ACVR1B"  "ACVR2A"  "APC"     "ARID1A"  "ARID2"   "ASXL1"   "ATM"    
##  [8] "ATR"     "ATRX"    "AXIN2"   "BCOR"    "BRAF"    "BRCA2"   "CARD11" 
## [15] "CBL"     "CDC73"   "CDH1"    "CDK12"   "CREBBP"  "CTNNB1"  "EGFR"   
## [22] "ELF3"    "EP300"   "ERBB2"   "ERBB3"   "EZH2"    "AMER1"   "FBXW7"  
## [29] "FGFR1"   "FGFR2"   "FLT1"    "GATA3"   "GNAS"    "H3F3A"   "H3F3B"  
## [36] "JAK2"    "KDM6A"   "KDR"     "KIT"     "KRAS"    "MET"     "MGA"    
## [43] "MLH1"    "KMT2D"   "KMT2C"   "MSH2"    "MSH6"    "NF1"     "NF2"    
## [50] "NRAS"    "PDGFRA"  "PIK3CA"  "PIK3R1"  "POLE"    "PTCH1"   "PTEN"   
## [57] "PTPN11"  "RB1"     "RBM10"   "RET"     "RNF43"   "SETD2"   "SMAD2"  
## [64] "SMAD4"   "SOX9"    "STAG2"   "STK11"   "TBX3"    "TCF7L2"  "TGFBR2" 
## [71] "TP53"    "TSHR"    "VHL"     "WT1"     "AKT1"    "AXIN1"   "B2M"    
## [78] "CDKN1B"  "CSF1R"   "CUX1"    "GRIN2A"  "MAP2K1"  "MAP2K4"  "MAX"    
## [85] "QKI"     "RAD21"   "ROBO2"   "SRC"     "TBL1XR1" "UBR5"
```

a. Whole genome analysis

```
wgsout <- dndscv(wgs)
```

```
## [1] Loading the environment...
```

```
## [2] Annotating the mutations...
## Warning in dndscv(wgs): Same mutations observed in different sampleIDs.
## Please verify that these are independent events and remove duplicates
## otherwise.
```

```
##     Note: 6 mutations removed for exceeding the limit of mutations per gene per
sample
```

```
##     51 %...
```

```
## [3] Estimating global rates...
```

```
## [4] Running dNdSloc...
```

```
## [5] Running dNdScv...
```

```
##     Regression model for substitutions: all covariates were used (theta = 5.28)
.
```

```
##     Regression model for indels: all covariates were used (theta = 1.4)
```

```
print(head(wgsout$sel_cv), digits = 3)
```

```
##       gene_name n_syn n_mis n_non n_spl n_ind wmis_cv wnon_cv wspl_cv
## 6146    FAM196B     0     5     2     0     0   12.20    63.2    63.2
## 16954   SULT2B1     0     2     0     0     2    5.15     0.0     0.0
## 13340     POTEE     3    11     0     0     0    8.38     0.0     0.0
## 16925   STXBP5L     2     0     3     0     1    0.00    10.2    10.2
## 11870     OLFM4     0     0     0     0     2    0.00     0.0     0.0
## 2710   C6orf118     1     8     0     0     0   11.73     0.0     0.0
##       wind_cv  pmis_cv ptrunc_cv pallsubs_cv  pind_cv qmis_cv qtrunc_cv
## 6146        0 4.41e-04  0.000573    1.71e-05 1.000000   0.790     0.952
## 16954     123 9.24e-02  0.821319    2.35e-01 0.000220   0.790     0.952
## 13340       0 1.20e-05  0.674260    5.71e-05 1.000000   0.208     0.952
## 16925      26 5.21e-02  0.006017    1.58e-03 0.037204   0.790     0.952
## 11870     146 2.94e-01  0.766392    5.54e-01 0.000157   0.790     0.952
## 2710        0 2.07e-05  0.704545    9.87e-05 1.000000   0.208     0.952
##       qallsubs_cv pglobal_cv qglobal_cv
## 6146        0.343   0.000205          1
## 16954       0.969   0.000561          1
## 13340       0.573   0.000615          1
## 16925       0.969   0.000630          1
## 11870       0.969   0.000902          1
## 2710        0.661   0.001009          1
```

```
wgs_signif_genes = wgsout$sel_cv[wgsout$sel_cv$qglobal_cv<0.1, c("gene_name","qglo
bal_cv")]
rownames(wgs_signif_genes) = NULL
print(wgs_signif_genes)
```

```
## [1] gene_name  qglobal_cv
## <0 rows> (or 0-length row.names)
```

```
print(wgsout$globaldnds)
```

```
##      name       mle     cilow   cihigh
## wmis wmis 0.9770452 0.9457840 1.009340
## wnon wnon 0.9846980 0.9082206 1.067615
## wspl wspl 0.9355197 0.8311536 1.052991
## wtru wtru 0.9691727 0.9047088 1.038230
## wall wall 0.9767169 0.9458699 1.008570
```

Thus there is no evidence of selection at the gene level, nor globally in the dataset.

b. Combined wgs and targeted analysis over all genes in baitset.

```
length(genes)
```

```
## [1] 90
```

```
combout <- dndscv(comb, gene_list = genes)
```

```
## [1] Loading the environment...
```

```
## [2] Annotating the mutations...
```

```
## Warning in dndscv(comb, gene_list = genes): Same mutations observed in
## different sampleIDs. Please verify that these are independent events and
## remove duplicates otherwise.
```

```
## [3] Estimating global rates...
```

```
## [4] Running dNdSloc...
```

```
## [5] Running dNdScv...
```

```
##     Regression model for substitutions: no covariates were used (theta = 1.66).
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
##     Regression model for indels: no covariates were used (theta = 1.6e+04)
```

```
print(head(combout$sel_cv), digits = 3)
```

```
##    gene_name n_syn n_mis n_non n_spl n_ind wmis_cv wnon_cv wspl_cv wind_cv
## 13     AXIN2     1     1     2     0     3   0.588   18.07   18.07   29.31
## 80     STAG2     0     9     2     1     2  10.558   24.55   24.55   13.00
## 7      ARID2     3     0     2     0     1   0.000    4.47    4.47    4.49
## 86      TP53     0     1     0     0     2   2.250    0.00    0.00   41.86
## 73     RNF43     0     7     1     0     1  10.563   16.41   16.41   10.52
## 44     KDM6A     0     1     2     1     0   1.251   26.79   26.79    0.00
##    pmis_cv ptrunc_cv pallsubs_cv  pind_cv qmis_cv qtrunc_cv qallsubs_cv
## 13  0.6702   0.01236     0.02057 0.000166   0.850     0.278       0.370
## 80  0.0104   0.00497     0.00762 0.010692   0.413     0.149       0.185
## 7   0.0117   0.12441     0.00326 0.199585   0.413     0.874       0.185
## 86  0.6045   0.78393     0.82585 0.001106   0.850     0.874       0.891
## 73  0.0138   0.09296     0.03332 0.090686   0.413     0.869       0.500
## 44  0.8872   0.00393     0.00662 1.000000   0.939     0.149       0.185
##    pglobal_cv qglobal_cv
## 13   4.63e-05    0.00416
## 80   8.48e-04    0.03817
## 7    5.42e-03    0.16262
## 86   7.30e-03    0.16432
## 73   2.06e-02    0.36996
## 44   3.98e-02    0.54241
```

```
comb_signif_genes = combout$sel_cv[combout$sel_cv$qglobal_cv<0.1, c("gene_name","q
global_cv")]
rownames(comb_signif_genes) = NULL
print(comb_signif_genes)
```

```
##   gene_name  qglobal_cv
## 1     AXIN2 0.004163356
## 2     STAG2 0.038174299
```

```
print(combout$globaldnds)
```

```
##      name       mle     cilow   cihigh
## wmis wmis 0.8838778 0.7102556 1.099942
## wnon wnon 0.9415812 0.5825645 1.521849
## wspl wspl 1.0600806 0.5265670 2.134146
## wtru wtru 0.9751772 0.6465682 1.470797
## wall wall 0.8915288 0.7190344 1.105404
```

Thus two genes show evidence of positive selection. Global measures of positive and negative selection do not differ significantly from the null hypothesis of neutral mutations.

5) Test whether the global spectrum of driver mutations is different between cancer and normal.

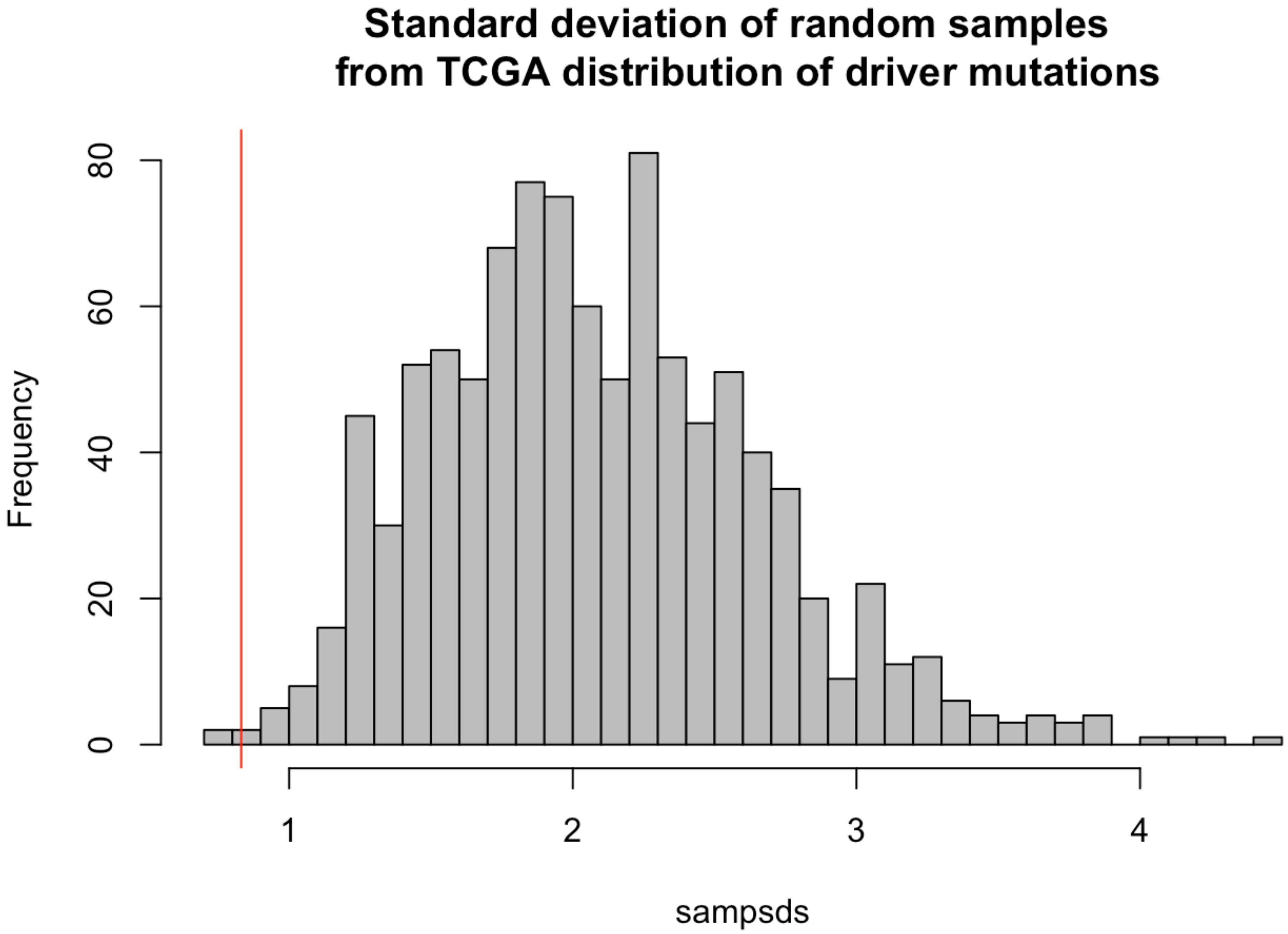
Read in the table of gene driver mutation frequencies in TCGA cancers and in normal crypts.

```
comp <- read.csv('/Users/hl11/Documents/Projects/normal_colon/WGS_results/2018.08.
02_tcga_crc_drivers/frequency_of_driver_mutations_in_tcga_vs_normal_crypts.txt', s
ep='\t', header = T, stringsAsFactors = F)
comp[,c("gene", "tcga", "normal", "tcgaprop", "normprop", "poistestpval")]
```

```
##       gene tcga normal    tcgaprop     normprop  poistestpval
## 1      APC  157      0 0.688596491 0.0000000000 5.848065e-127
## 2     TP53  119      1 0.521929825 0.0008166599  2.117820e-94
## 3     KRAS   79      0 0.346491228 0.0000000000  3.030332e-64
## 4   PIK3CA   27      2 0.118421053 0.0016333197  5.715771e-20
## 5     BRAF   20      0 0.087719298 0.0000000000  8.305478e-17
## 6     NRAS   20      0 0.087719298 0.0000000000  8.305478e-17
## 7    AMER1   19      0 0.083333333 0.0000000000  5.289278e-16
## 8    SMAD4   17      0 0.074561404 0.0000000000  2.145161e-14
## 9      ATM    9      2 0.039473684 0.0016333197  2.354268e-06
## 10   FBXW7    9      2 0.039473684 0.0016333197  2.354268e-06
## 11    SOX9    9      0 0.039473684 0.0000000000  5.803822e-08
## 12   ERBB3    7      2 0.030701754 0.0016333197  6.307790e-05
## 13    PTEN    7      0 0.030701754 0.0000000000  2.353844e-06
## 14  TCF7L2    7      0 0.030701754 0.0000000000  2.353844e-06
## 15  ACVR2A    6      0 0.026315789 0.0000000000  1.499027e-05
## 16    MSH6    6      0 0.026315789 0.0000000000  1.499027e-05
## 17  PIK3R1    6      0 0.026315789 0.0000000000  1.499027e-05
## 18   ARID2    5      1 0.021929825 0.0008166599  4.978347e-04
## 19   ASXL1    5      0 0.021929825 0.0000000000  9.546434e-05
## 20    BCOR    5      0 0.021929825 0.0000000000  9.546434e-05
## 21    ELF3    5      0 0.021929825 0.0000000000  9.546434e-05
## 22   RBM10    5      0 0.021929825 0.0000000000  9.546434e-05
## 23   SMAD2    5      0 0.021929825 0.0000000000  9.546434e-05
## 24     B2M    4      0 0.017543860 0.0000000000  6.079571e-04
## 25   BRCA2    4      1 0.017543860 0.0008166599  2.657928e-03
## 26   KMT2D    4      0 0.017543860 0.0000000000  6.079571e-04
## 27     NF1    4      0 0.017543860 0.0000000000  6.079571e-04
## 28    POLE    4      0 0.017543860 0.0000000000  6.079571e-04
## 29   ROBO2    4      0 0.017543860 0.0000000000  6.079571e-04
## 30   SETD2    4      0 0.017543860 0.0000000000  6.079571e-04
## 31   AXIN2    3      3 0.013157895 0.0024499796  5.336328e-02
## 32  CARD11    3      0 0.013157895 0.0000000000  3.871727e-03
## 33  CDKN1B    3      1 0.013157895 0.0008166599  1.366304e-02
## 34  MAP2K4    3      0 0.013157895 0.0000000000  3.871727e-03
## 35     MGA    3      0 0.013157895 0.0000000000  3.871727e-03
## 36     RB1    3      0 0.013157895 0.0000000000  3.871727e-03
## 37     ATR    2      1 0.008771930 0.0008166599  6.622690e-02
## 38    CDH1    2      0 0.008771930 0.0000000000  2.465679e-02
## 39   CDK12    2      2 0.008771930 0.0016333197  1.187908e-01
## 40   FGFR2    2      0 0.008771930 0.0000000000  2.465679e-02
## 41  MAP2K1    2      0 0.008771930 0.0000000000  2.465679e-02
## 42  PTPN11    2      0 0.008771930 0.0000000000  2.465679e-02
## 43     QKI    2      0 0.008771930 0.0000000000  2.465679e-02
## 44   RNF43    2      2 0.008771930 0.0016333197  1.187908e-01
## 45    UBR5    2      0 0.008771930 0.0000000000  2.465679e-02
## 46   AXIN1    1      0 0.004385965 0.0000000000  1.570248e-01
## 47  CTNNB1    1      0 0.004385965 0.0000000000  1.570248e-01
## 48    EGFR    1      0 0.004385965 0.0000000000  1.570248e-01
## 49   EP300    1      0 0.004385965 0.0000000000  1.570248e-01
## 50   H3F3B    1      0 0.004385965 0.0000000000  1.570248e-01
## 51     KDR    1      0 0.004385965 0.0000000000  1.570248e-01
## 52    MSH2    1      0 0.004385965 0.0000000000  1.570248e-01
## 53     NF2    1      0 0.004385965 0.0000000000  1.570248e-01
## 54   PTCH1    1      0 0.004385965 0.0000000000  1.570248e-01
## 55   RAD21    1      0 0.004385965 0.0000000000  1.570248e-01
## 56   STAG2    1      2 0.004385965 0.0016333197  4.009757e-01
## 57    TBX3    1      0 0.004385965 0.0000000000  1.570248e-01
## 58  TGFBR2    1      0 0.004385965 0.0000000000  1.570248e-01
## 59   ERBB2    0      3 0.000000000 0.0024499796  1.000000e+00
## 60 TBL1XR1    0      1 0.000000000 0.0008166599  1.000000e+00
```

Sample from the tcga distribution 1,000 times, and every time take a measure of the flatness of the distribution.

```
tcgagenes <- as.character(unlist(apply(comp,1,function(row) rep(row['gene'], row['
tcga']))))

sampsds <- c()
for (i in 1:1000) {
  tsamp <- sample(tcgagenes, sum(comp$normal), replace=T)
  tcnts <- as.numeric(table(tsamp))
  sampsds <- c(sampsds, sd(tcnts))
}
hist(sampsds, col="grey", 50, main="Standard deviation of random samples \n from T
CGA distribution of driver mutations")
abline(v=sd(comp$normal), col="red")
```



Now print the probability of getting such a flat distribution as we observe in the normal tissues if normal drivers were just resampling from cancer drivers.

```
length(which(sampsds<=sd(comp$normal)))/length(sampsds)
```

```
## [1] 0.003
```