‘Fragmentation duplicates’: Sweave

Andy Lynch and Mike Smith

September 2, 2015

Contents

1 Introduction: The LynchSmithEldridgeTavareFragDup Package

1.1 This Sweave

This Sweave generates the tables and data-driven figures displayed in the paper ‘A method for the empirical correction of estimated PCR duplication rates, with applications’ by Lynch, Smith, Eldridge and Taveré on behalf of the OCCAMS Consortium. It also provides greater information on some points of detail. It is dependent on the *Bioconductor* *BiocStyle* package.

This Sweave is part of the *R* package *LynchSmithEldridgeTavareFragDup* that forms the additional file for that paper and which contains all of the functions, data, and (through this Sweave) code to reproduce the results in the manuscript. This package serves no other purpose and is not part of any repository. Pains have been taken to choose a name that is unlikely to clash with any other package.

1.2 The Data

The contents of the extdata folder in the package are:

```
> list.files(system.file("extdata", package="LynchSmithEldridgeTavareFragDup"))
[1] "HetSNPDups.txt"        "HetSNPDupsHighGC.txt" "HetSNPDupsLowGC.txt"
[4] "Picard"                "SNPstoextract.txt"    "Tumour"
[7] "WeaverSuppTable1.txt" "masks"
```

’WeaverSuppTable1.txt’ reproduces data from the first Supplementary Table of Weaver et al. (2014) *Nature Genetics* **46**, 837–843, and describes the samples used to illustrate this manuscript.

The contents of ’masks’ and ’Picard’ detail the numbers of reads and duplicates within defined regions of the genome as discussed in Section~2.

’SNPstoextract.txt’ and ’HetSNPDups.txt’ give details of the SNPs described in Section~4 and used in Section~5.

The contents of ’Tumour’ are used in Section~7, where ’HetSNPDupsHighGC.txt’ and ’HetSNPDupsLowGC.txt’ are also described.

1.3 The Functions

The library defines the following functions:

- `secondbiggest`: a function to pick the second largest value from a vector.
- `propllik`: a function function that takes the empirical observations of allele patterns, and calculates the log-likelihood of the proportion of duplicates arising from fragmentation of distinct molecules, via calls to genprobs.
- `libCompNewParam`: A function to check the consistency of a library complexity estimate, the number of read-pairs and the duplicate rate.

- genprobs: Returns the probabilities of different allele patterns given the number of duplicates and the fragmentation duplication rate.
- genPL: Generate the lists of all partitions of n objects for $1 \leq n \leq N$
- genCoefs: Generates the binomialesque coefficients associated with partitions of objects.

2 Sample information

We are considering twenty-two 'normal' samples from patients forming part of the Oesophageal Adenocarcinoma ICGC study run by the OCCAMS Consortium. The full data for these samples are archived in the European Genome Archive [EGA:EGAD00001000704] and have featured in the paper Ordering of mutations in preinvasive disease stages of esophageal carcinogenesis.

Of the twenty-two samples, twelve were blood, and ten were oesophageal tissue. DNA from oesophageal tissue was extracted using the DNeasy kit (Qiagen) and from blood using the NucleonTM Genomic Extraction kit (Gen-Probe) (according to the manufacturer's instructions).

The sequencing was conducted under contract by Illumina, typically comprising five lanes of 100bp paired end sequencing. The depth of coverage ranges from 57x to 87x with a mean of 68x.

This study is a presentation of analysis methods not a presentation of the sequencing data, and raw data that could identify the patients is not provided. Sufficient data are provided to illustrate the methods presented. Further access is via the Project's Data Access Committee.

```
> library("Rsamtools")
> library("BSgenome.Hsapiens.UCSC.hg19")
> EDpath <- system.file("extdata", package="LynchSmithEldridgeTavareFragDup")
```

2.1 Picard

Picard was applied to the raw bam files, and from this output we extract data on the number of read pairs examined, the numbers of read pairs marked as duplicates, and the number of read pairs marked as optical duplicates. We keep the full results for incorporation into Table 1.

```
> unmaskedP<-read.delim(file.path(EDpath,"Picard", "samples.metrics.txt"),as.is=T)
> readPairsExamined <-
+   readPairDuplicates <-
+   OpticalPairDuplicates <- matrix(ncol=9, nrow=22,
+                               dimnames = list(unmaskedP[,1], c("Total", "X", "Y", "M",
+                                                                "Centromeres","Telomeres",
+                                                                "LowCov","HighCov",
+                                                                "Residual")))
> readPairsExamined[,1]<-unmaskedP[,3]
> readPairDuplicates[,1]<-unmaskedP[,6]
> OpticalPairDuplicates[,1]<-unmaskedP[,7]
```

2.1.1 Generate Table 1

Before generating a table, we define some vectors that allow us to write out the results in a form ready to paste into a LaTeX document.

```
> latexdiv22<-rep("&",22)
> latexend22<-rep("\\\\",22)
> latexdiv8<-rep("&",8)
> latexend8<-rep("\\\\",8)
```

We read in supplementary table 1 from Weaver et al, and extract the information of interest to us.

```
> WeaverSuppTable1<-read.delim(file.path(EDpath,"WeaverSuppTable1.txt"),sep="",header=T,as.is=T)
> WeaverSuppTable1$CaseID<-WeaverSuppTable1$CaseID-1
```

Next we collect the information for Table 1 of the main manuscript, and prepare it for insertion into the LaTeX document.

```
> Table1<-cbind(
+ WeaverSuppTable1[,1],latexdiv22,
+ WeaverSuppTable1[,2],latexdiv22,
+ WeaverSuppTable1[,12],latexdiv22,
+ round(unmaskedP[,3]/1000000000,2),latexdiv22,
+ round(100*unmaskedP[,6]/unmaskedP[,3],2),latexdiv22,
+ round(unmaskedP[,9]/1000000000,2),latexend22
+ )
> Table1[Table1=="NormalOesophagus"]<-"NormalOes"
> Table1<-rbind(c("ID", "&","Sex","&","Tissue","&","Reads","&","Dup.","&","Library", "\\\\"),
+                 c("","&","","&","Tissue","&","($10^9$)","&","Rate","&","Size","\\\\" ),
+                 c("","&","","&","","&","","&","","&","($10^9$)","\\\\"), Table1)
> write.table(Table1,sep=" ",file="Table1.tsv",row.names=F,col.names=F,quote=F)
> Table1
             latexdiv22          latexdiv22             latexdiv22            latexdiv22
 [1,] "ID"   "&"        "Sex"    "&"        "Tissue"    "&"        "Reads"    "&"
 [2,] ""     "&"        ""       "&"        "Tissue"    "&"        "($10^9$)" "&"
 [3,] ""     "&"        ""       "&"        ""          "&"        ""         "&"
 [4,] "3108" "&"        "Female" "&"        "Blood"     "&"        "1.15"     "&"
 [5,] "3110" "&"        "Male"   "&"        "Blood"     "&"        "1.16"     "&"
 [6,] "3112" "&"        "Male"   "&"        "Blood"     "&"        "1.29"     "&"
 [7,] "3114" "&"        "Male"   "&"        "Blood"     "&"        "0.88"     "&"
 [8,] "3116" "&"        "Male"   "&"        "NormalOes" "&"        "0.93"     "&"
 [9,] "3118" "&"        "Male"   "&"        "NormalOes" "&"        "1.16"     "&"
[10,] "3120" "&"        "Male"   "&"        "Blood"     "&"        "1.11"     "&"
[11,] "3124" "&"        "Male"   "&"        "Blood"     "&"        "0.99"     "&"
[12,] "3128" "&"        "Male"   "&"        "Blood"     "&"        "0.97"     "&"
[13,] "3130" "&"        "Male"   "&"        "Blood"     "&"        "1.03"     "&"
[14,] "3132" "&"        "Male"   "&"        "Blood"     "&"        "1.02"     "&"
[15,] "3134" "&"        "Male"   "&"        "Blood"     "&"        "1.1"      "&"
[16,] "3136" "&"        "Male"   "&"        "Blood"     "&"        "0.99"     "&"
[17,] "3148" "&"        "Female" "&"        "Blood"     "&"        "1.03"     "&"
[18,] "3301" "&"        "Male"   "&"        "NormalOes" "&"        "0.89"     "&"
[19,] "3304" "&"        "Male"   "&"        "NormalOes" "&"        "1.01"     "&"
[20,] "3307" "&"        "Male"   "&"        "NormalOes" "&"        "1.04"     "&"
[21,] "3310" "&"        "Female" "&"        "NormalOes" "&"        "1.03"     "&"
[22,] "3313" "&"        "Female" "&"        "NormalOes" "&"        "1"        "&"
[23,] "3316" "&"        "Male"   "&"        "NormalOes" "&"        "0.97"     "&"
[24,] "3319" "&"        "Male"   "&"        "NormalOes" "&"        "1.05"     "&"
[25,] "3322" "&"        "Male"   "&"        "NormalOes" "&"        "0.93"     "&"
              latexdiv22            latexend22
 [1,] "Dup."   "&"        "Library"  "\\\\"
 [2,] "Rate"   "&"        "Size"     "\\\\"
 [3,] ""       "&"        "($10^9$)" "\\\\"
 [4,] "6.15"   "&"        "9.03"     "\\\\"
 [5,] "7.86"   "&"        "7"        "\\\\"
 [6,] "6.94"   "&"        "8.86"     "\\\\"
 [7,] "5.05"   "&"        "8.46"     "\\\\"
 [8,] "11.92" "&"        "3.59"     "\\\\"
 [9,] "14.49" "&"        "3.61"     "\\\\"
[10,] "4.01"  "&"        "13.55"    "\\\\"
[11,] "8.07"  "&"        "5.81"     "\\\\"
[12,] "4.8"   "&"        "9.8"      "\\\\"
[13,] "8.34"  "&"        "5.83"     "\\\\"
[14,] "6.03"  "&"        "8.15"     "\\\\"
[15,] "4.38"  "&"        "12.21"    "\\\\"
[16,] "5.76"  "&"        "8.3"      "\\\\"
[17,] "3.73"  "&"        "13.55"    "\\\\"
[18,] "5.92"  "&"        "7.27"     "\\\\"
[19,] "8.37"  "&"        "5.74"     "\\\\"
[20,] "6.85"  "&"        "7.27"     "\\\\"
[21,] "10.71" "&"        "4.46"     "\\\\"
[22,] "5.22"  "&"        "9.31"     "\\\\"
[23,] "5.99"  "&"        "7.86"     "\\\\"
[24,] "8.24"  "&"        "6.01"     "\\\\"
[25,] "11.34" "&"        "3.8"      "\\\\"
```

We will define four groups based on sex and the origin of the tissue/DNA extraction kit used.

```
> table(WeaverSuppTable1[,2],WeaverSuppTable1[,12])
```

```
         Blood NormalOesophagus
  Female     2                2
  Male      10                8
```

```
> Group<-1+2*(WeaverSuppTable1[,2]=="Male")+(WeaverSuppTable1[,12]=="Blood")
> GroupN<-c("Female_Tissue","Female_Blood","Male_Tissue","Male_Blood")[Group]
```

2.1.2 Masks

The regions that we mask out of the genome for this study are defined in the '.bed' files in this package. For instance, the Centromeres are masked out by the 4th mask file. For further reference, we will record the total genome size, sizes of masked regions, and the remainder (residual).

```
> list.files(file.path(EDpath,"masks"))
```

```
[1] "Mask1-ChrX.bed"        "Mask2-ChrY.bed"        "Mask3-ChrM.bed"
[4] "Mask4-Centromeres.bed" "Mask5-Telomeres.bed"   "Mask6-lowdepth.bed"
[7] "Mask7-highdepth.bed"
```

```
> head(read.delim(file.path(EDpath, "masks", "Mask4-Centromeres.bed"),header=F,as.is=T))
```

```
    V1        V2      V3
1 chr1 120000000 1.5e+08
2 chr2  87000000 9.9e+07
3 chr3  90000000 9.4e+07
4 chr4  49000000 5.3e+07
5 chr5  46000000 5.0e+07
6 chr6  57000000 6.3e+07
```

```
> maskFiles <- list.files(file.path(EDpath,"masks"), full.names = TRUE)
> MASKS <- lapply(maskFiles, read.delim, header=FALSE, as.is=TRUE, skip=1)
> effectiveGenomeSize<-rep(0,9)
> effectiveGenomeSize[1]<-sum(as.numeric(MASKS[[5]][17:38,3]))
> +MASKS[[1]][1,3]+MASKS[[2]][1,3]+MASKS[[3]][1,3]
```

```
[1] 214676571
```

```
> for(i in 1:7){
+     effectiveGenomeSize[i+1] <- sum(MASKS[[i]][,3] - MASKS[[i]][,2])}
> effectiveGenomeSize[9]<-effectiveGenomeSize[1]-sum(effectiveGenomeSize[2:8])
```

2.1.3 Picard Output

Picard was run on bam files defined by these masks (separately for each mask) and we read these data into the matrices created earlier, before creating Table 2 in the paper.

We only use the "READ_PAIRS_EXAMINED", "READ_PAIR_DUPLICATES", and "READ_PAIR_OPTICAL_DUPLICATES" values that Picard produces (as indeed does Picard for this problem). For simplicity, we create three tables to take these values and store in them the Picard output for the entire samples, and then also for Picard output generated from the masked regions.

```
> metfiles<-list.files(file.path(EDpath,"Picard"))
> metfiles<-grep("mask",metfiles,value=T)
> for(i in 1:7){
+   temp<-read.delim(file.path(EDpath,"Picard",metfiles[i]),as.is=T)
+   temp<-temp[-(15:19),]
+   readPairsExamined[,i+1]<-temp[,3]
+   readPairDuplicates[,i+1]<-temp[,6]
+   OpticalPairDuplicates[,i+1]<-temp[,7]
+ }
```

We now calculate the 'residual' counts by subtracting the masked regions from the total.

```
> readPairsExamined[,9]<-readPairsExamined[,1]-apply(readPairsExamined[,2:8],1,sum)
> readPairDuplicates[,9]<-readPairDuplicates[,1]-apply(readPairDuplicates[,2:8],1,sum)
> OpticalPairDuplicates[,9]<-OpticalPairDuplicates[,1]-apply(OpticalPairDuplicates[,2:8],1,sum)
```

2.1.4 Generate Table 2

Now we report the residual values and the numbers for the masked regions relative to the residual values. This is Table 2 of the main manuscript.

```
> reptable<-matrix(0,ncol=4,nrow=8)
> reptable[1,]<-round(100*sapply(split(((readPairDuplicates[,9]/readPairsExamined[,9])),Group),mean),2)
> for(i in 2:8){
+   reptable[i,]<-round(sapply(split((
+     (readPairDuplicates[,i]/readPairsExamined[,i])
+     /(readPairDuplicates[,9]/readPairsExamined[,9])),Group),mean),2)
+ }
> Table2<-cbind(reptable[,1],latexdiv8,reptable[,2],latexdiv8,reptable[,3],latexdiv8,
+               reptable[,4],latexend8)
> write.table(Table2,sep="\t",file="Table2.tsv",row.names=F,col.names=F,quote=F)
> reptable[3,1:2]<-"-"
> rownames(reptable)<-c("Residual","X","Y","M","Centromeres","Telomeres","Low Cov","High Cov")
> colnames(reptable)<-c("Female Tissue","Female Blood","Male Tissue","Male Blood")
> reptable
```

```
            Female Tissue Female Blood Male Tissue Male Blood
Residual    "6.82"        "3.48"       "7.8"       "4.43"
X           "1.23"        "1.43"       "1.09"      "1.2"
Y           "-"           "-"          "4.6"       "9.19"
```

```
M           "7.41"         "14.62"      "5.62"      "5.05"
Centromeres "4.45"         "8.46"       "3.88"      "7.37"
Telomeres   "0.99"         "1.19"       "0.98"      "1.09"
Low Cov     "0.94"         "0.95"       "0.93"      "0.94"
High Cov    "1.6"          "2.76"       "1.44"      "2.45"
```

3 Methods

3.1 Case 1: A pair of duplicate fragments

Recall that if we have only observe duplicate fragments in pairs then, if we observe counts of N_{AA} fragment pairs reporting the same nucleotide, and N_{AB} reporting different nucleotides, the estimate of P_D is

$$P_D = 1 - 2 \times N_{AB}/N. \tag{1}$$

3.2 Case 2: More than two duplicate fragments

Recall that in this case we estimate P_D by maximizing the total log-likelihood

$$l(P_D) = \sum_M l_M(P_D). \tag{2}$$

where

$$l_M(P_D) = \sum_{k=1}^{Q_M} N(AP_k) \log \Pr(AP_k \mid P_D), \tag{3}$$

and

$$\Pr(AP_k \mid P_D) = \sum_i \Pr(AP_k \mid \mathsf{PART}_i) \Pr(\mathsf{PART}_i \mid P_D). \tag{4}$$

In these equations, M is the size of our set of duplicate fragments, PART_i is the i^{th} potential partition of the duplicate fragments into the original molecules, AP_k is the k^{th} observable allele pattern, and $N(AP_k)$ is the number of times that the allele pattern has been observed.

A worked example for M=4 is given in the main manuscript. An example for M=5 appears in Figure 1.

The probabilities $\Pr(\mathsf{PART}_i \mid P_D)$ are all of the form $aP_D^bF_D^c$ where a, b and c are to be determined. b is clearly the number of PCR duplicates in the partition, and $c = M - b - 1$.

The calculation of a is not always as clear. Given a partition of the M fragments into N molecules that provide f_1,f_2,...,f_N fragments in turn ($f_i > 0 \forall i, \sum_i f_i = M$), we denote as g_1,g_2,...,g_Q ($Q \leq N$) the set of unique values obtained by the f_i, and denote as ν_j the number of molecules contributing g_j fragments $1 \leq j \leq Q$. The value of a is then calculated as $a = N!/\prod_j \nu_j!$.

To give a concrete example: When there are four fragments ($M = 4$) partitioned amongst three molecules ($N = 3$) such that $f_1 = 2$, $f_2 = 1$, and $f_3 = 1$, then $g_1 = 2$ and $\nu_1 = 1$ while $g_2 = 1$ and $\nu_2 = 2$. The value of a is then $3!/(2!1!) = 3$ as is confirmed in the recursive approach that follows.

It is perhaps more clear if we derive these values recursively (as illustrated in Figure 2.), and this is the default option offered. When M=2 the partition probabilities are not controversial, being P_D if a PCR duplicate is present and F_D if two molecules are present. These probabilities sum to 1 as required. In each case we consider the addition of an additional fragment (a PCR duplicate with probability P_D or a fragmentation duplicate with probability F_D). If the new fragment is a PCR duplicate, then it may be a duplicate of any of the existing molecules with equal probability and so the probabilities are shared accordingly.

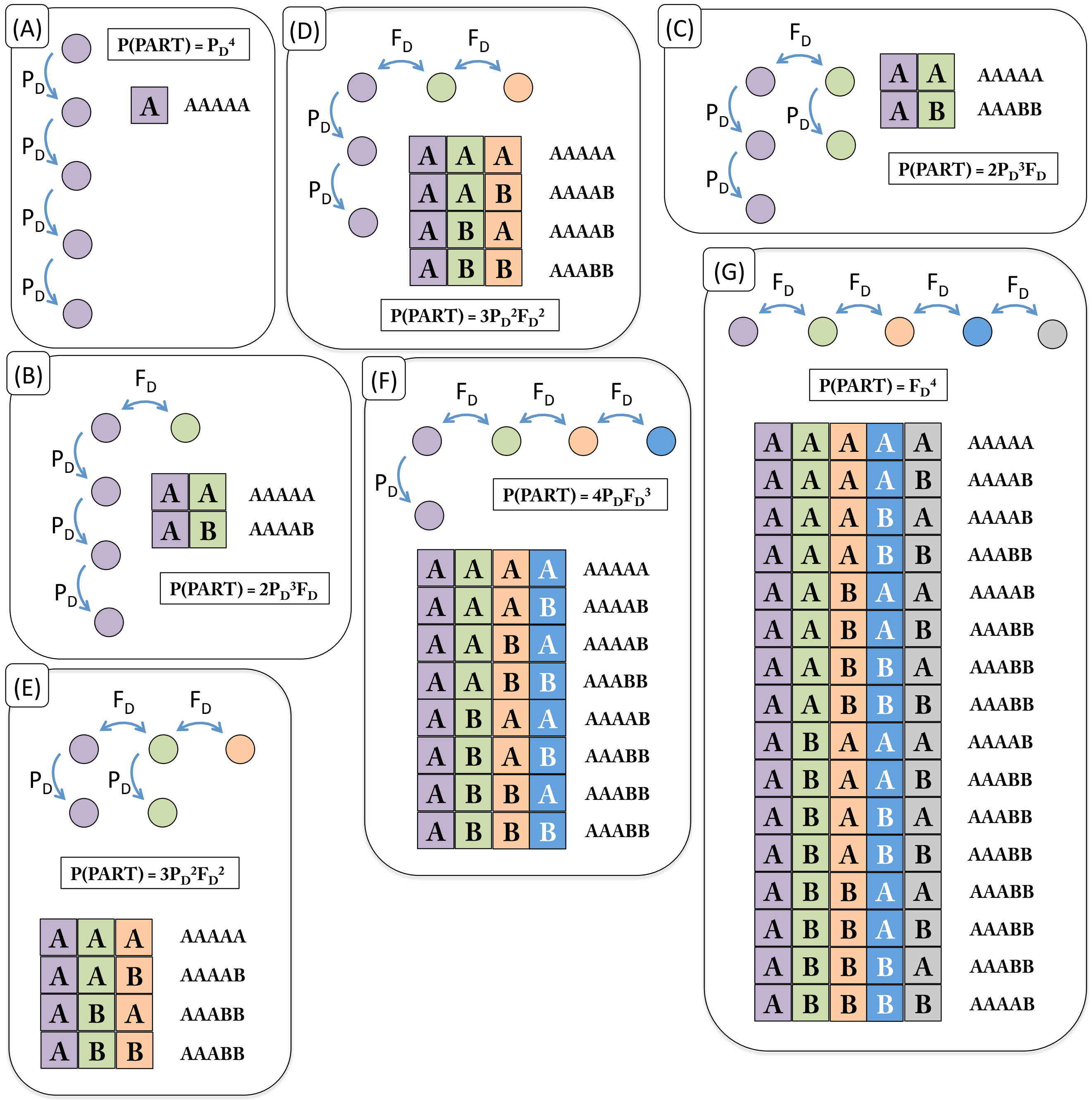


Figure 1: Worked example calculation of $\Pr(AP_k \mid P_D)$ for $M = 5$. There are seven identifiably distinct partitions of fragments into molecules. In panel A we have four PCR duplicates of a single fragment. In panel B there are two distinct duplicate fragments, one of which has three PCR duplicates, while panel C has two distinct duplicate fragments one of which has one PCR duplicate and one of which has two PCR duplicates. In panel D, there are three distinct duplicate fragments, one of which has two PCR duplicates and in panel E there are again three distinct fragments, two of which have a PCR duplicate. Panel F depicts the case where there are four distinct molecules, one of which has a PCR duplicate, and in panel G there are five distinct molecules with no PCR duplicates. The probabilities of the seven possible partitions $\Pr(\text{PART}_i \mid P_D)$ are shown, and for each partition the proportions in which the three possible observable allele patterns (AAAAA, AAAAB, AAABB) will arise are illustrated, allowing calculation of $\Pr(AP_k \mid P_D)$ For example $\Pr(\text{AAABB} \mid P_D) = P_D^3F_D + \frac{3}{4}P_D^2F_D^2 + \frac{3}{2}P_D^2F_D^2 + 2P_DF_D^3 + \frac{10}{16}F_D^4$

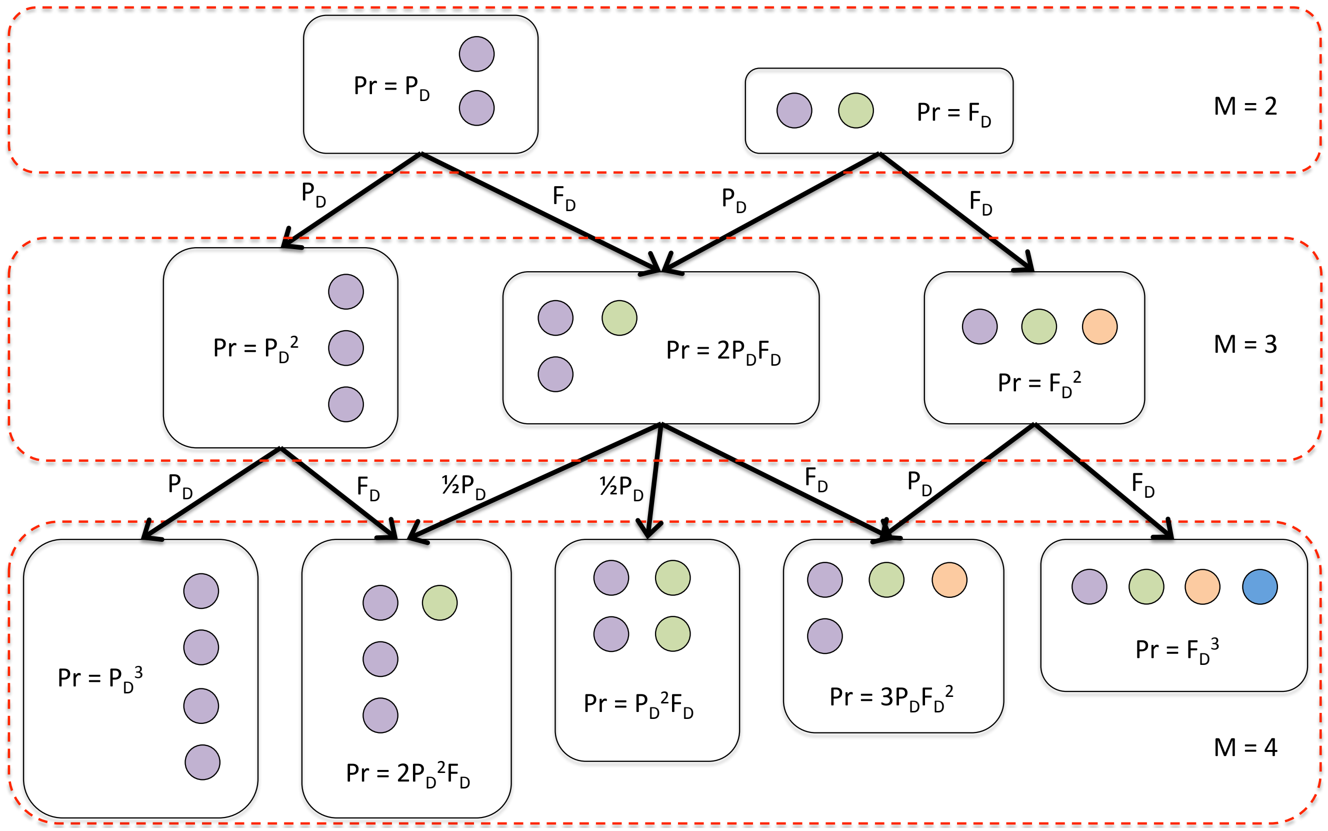


Figure 2: Recursive calculation of partition probabilities

4 SNPs to extract

4.1 Choosing the SNPs

We investigate $2,500$ common SNPs in anticipation of identifying $1,000$ heterozygous sites for each sample. In this manner we avoid the burden of having to define a bespoke set of sites for each case.

The SNPs were selected from UCSC's snp138Common table, considering only those validated by the 1000 Genomes Project. We required the reported population minor allele frequency (MAF) of selected SNPs be > 0.49 based on $> 2,500$ observations and that the SNPs were located on one of the 24 'regular' chromosomes. The $2,500$ SNPs were selected at random from all those meeting the criteria.

Despite these criteria, the true MAF can only be biased in one direction from the approximately 0.5 we nominally required, and our sequencing depth and requirements for calling heterozygous SNPs in the data mean that we will fail to identify up to ten percent of heterozygous sites because, by chance, their observed allele frequencies will be too extreme. Additionally, 35 of the randomly selected SNPs were from the sex chromosomes, and are 'unlikely' to be heterozygous in our predominantly male patients. In combination, these effects mean that we can anticipate approximately $1,000$ of the SNPs to be heterozygous in each patient.

By choosing a set of SNPs with high MAF, we can guarantee a good number of heterozygous observations in each sample, without the computational burden of defining a bespoke set of sites for each case. Extracting all reads mapping to these locations creates a BAM file that is approximately $10,000$ times smaller than the original and so allows for easy manipulation of large cohorts. We note that such a set of SNPs, and the BAM file they produce, are also useful for activities such as i) detecting sample contamination, ii) detecting sample mix ups, iii) detecting familial/ancestral relationships, and so have utility beyond our purposes in this study.

We now load in the set of SNPs and explore them.

```
> snplist<-read.delim(file.path(EDpath,"SNPstoextract.txt"),as.is=T,header=T)
```

4.2 Representativeness of the SNPs

4.2.1 Distribution across Chromosomes

We can see that the SNPs are chosen from all chromosomes (Figure 3), and that we are not far from having numbers proportional to chromosome length.

```
> par(mar=c(4.6,4.1,1.6,1.6))
> plot(table(snplist$chrom)[c(1,12,16:22,2:11,13:15,23:24)],
+      ylab="number of SNPs",xlab="chromosome",axes=F)
> axis(2)
> axis(1,labels=c(1:22,"X","Y"),at=1:24,las=2)
> box()
```

4.2.2 GC content

We take the locations of the $2,500$ SNPs in our list and place a window of 500 bases around them, before counting the number of G and C bases in that window.

```
> genome <- Hsapiens
> GCNo<-rep(NA,2500)
> for(i in 1:2500){
+   GCNo[i]<-sum(unlist(strsplit(as.character(substr(genome[[snplist$chrom[i]]],snplist$chromEnd[i]-250,snp
+ }
```

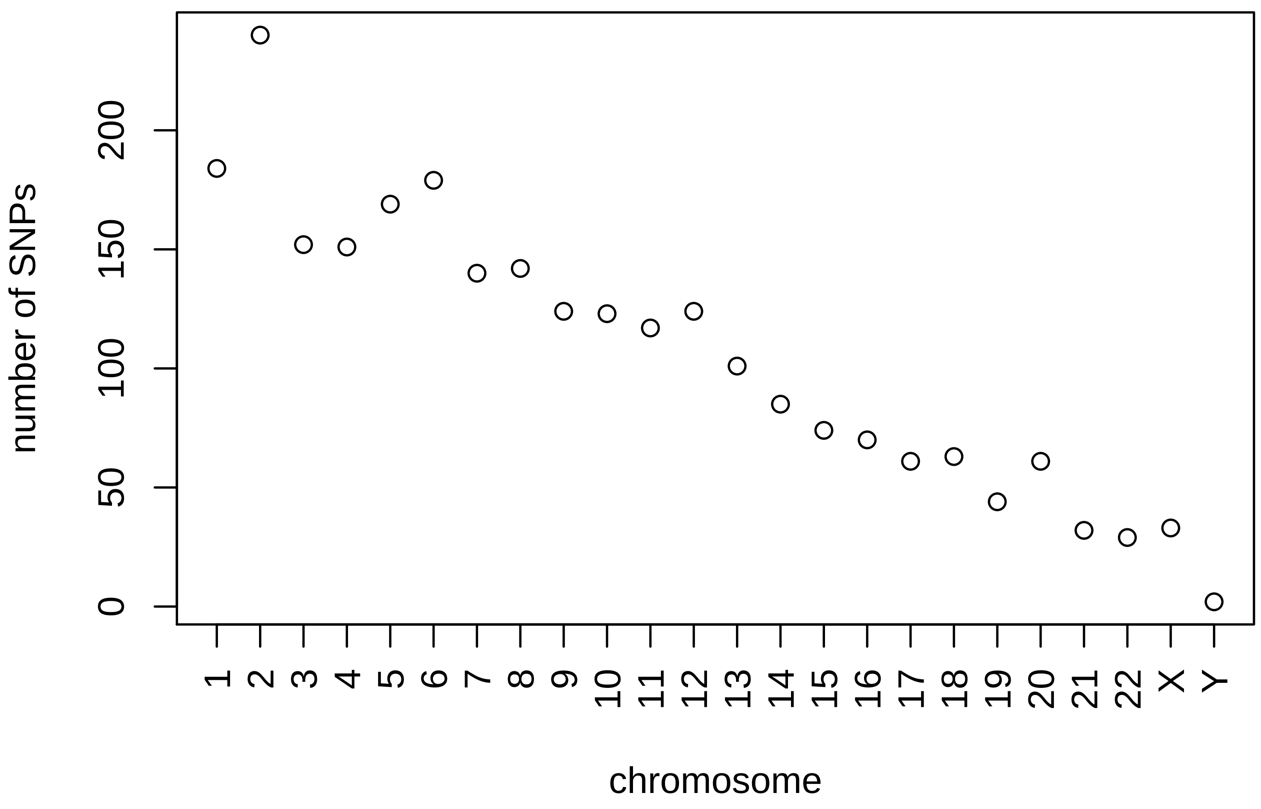


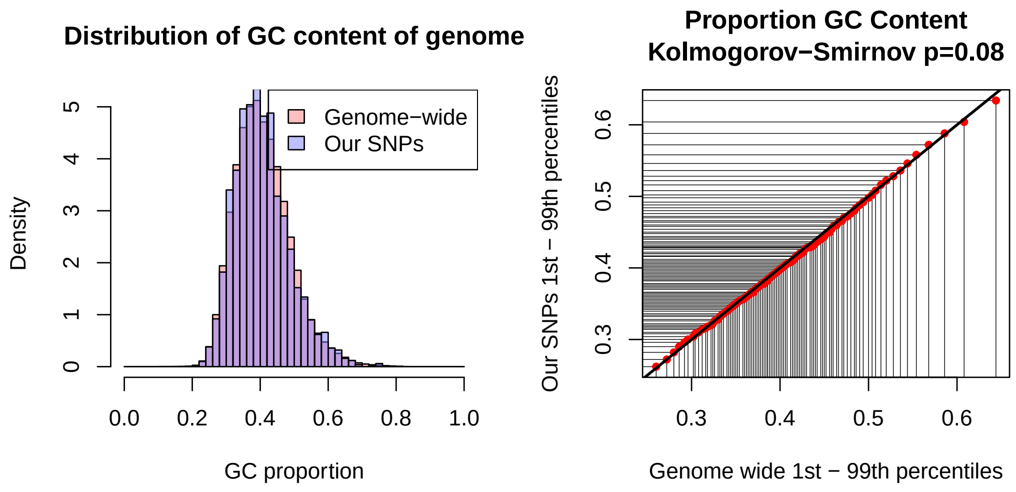
Figure 3: **SNP numbers.** The numbers of SNPs used from each chromosome.

For comparison, we take the human genome (chromosomes 1-22 and X), divide it up into bins of 500 and calculate the proportion of bases that are G or C in each bin.

```
> GCbinned<-NULL
> for(k in 1:23){
+     windowViews <- trim(Views(genome[[k]], start = seq(1, length(genome[[k]]), 500), width = 500))
+     letterFreq <- letterFrequency(windowViews, letters = c("A","C","G","T"))
+     seqGC <- rowSums(letterFreq[,2:3]) / rowSums(letterFreq)
+     GCbinned<-c(GCbinned,seqGC)
+ }
```

We see that the distribution of GC in our selection is a good representation of the GC content of the genome (with departure only in the final percentile).

```
> par(mfrow=c(1,2))
> hist(GCbinned,freq=F,col=rgb(1,0.5,0.5,0.5),breaks=seq(0,1,0.02),main="Distribution of GC content of gen
> hist(GCNo/500,freq=F,col=rgb(0.5,0.5,1,0.5),breaks=seq(0,1,0.02),add=T)
> legend("topright",fill=c(rgb(1,0.5,0.5,0.5),rgb(0.5,0.5,1,0.5)),legend=c("Genome-wide","Our SNPs"))
> plot(quantile(GCbinned,probs=seq(0.01,.99,0.01),na.rm=T),quantile(GCNo/500,probs=seq(.01,.99,0.01),na.rm
+      pch=20,col="red",xlab="Genome wide 1st - 99th percentiles",
+      ylab="Our SNPs 1st - 99th percentiles",
+      main="Proportion GC Content\nKolmogorov-Smirnov p=0.08")
> for(i in 1:99){
+   GW<-quantile(GCbinned,probs=i/100,na.rm=T)
+   OS<-quantile(GCNo/500,probs=i/100,na.rm=T)
+ lines(c(GW,GW),c(0,OS),lwd=0.5)
+ lines(c(0,GW),c(OS,OS),lwd=0.5)
+ }
> points(quantile(GCbinned,probs=seq(0.01,.99,0.01),na.rm=T),quantile(GCNo/500,probs=seq(.01,.99,0.01),na.
> abline(0,1,lwd=2)
```



Despite the power to detect deviation with this number of samples, the Kolmogorov-Smirnov test is not significant.

```
> ks.test(GCbinned,GCNo/500)

        Two-sample Kolmogorov-Smirnov test

data:  GCbinned and GCNo/500
D = 0.02552, p-value = 0.07717
alternative hypothesis: two-sided
```

4.3 Processing the SNPs

The following code will not be evaluated, as the patients' data are not distributed with this sweave, but can be obtained from the European Genome Archive (as previously noted). The summary data which are distributed are loaded in after this section, but the code here shows how they were created.

First we identify the BAMs to be examined.

```
> bamfilelist <- list.files("data/snpbams", pattern = ".bam$")
```

We know that in these data we only have to consider up to seven duplicates in a set. For other data sets it would be necessary to determine that this is sufficient. The manner in which the method could be extended to sets of 8, 9 ... etc. should be apparent.

```
> ACvec<-c("2:0", "1:1" ,
+          "3:0" ,"2:1" ,
+          "4:0" ,"3:1" ,"2:2" ,
+          "5:0" ,"4:1" ,"3:2" ,
+          "6:0" ,"5:1" ,"4:2" ,"3:3" ,
+          "7:0" ,"6:1" ,"5:2" ,"4:3")
```

We prepare a matrix to contain the results

```
> HetSNPTable<-matrix(0,nrow=length(bamfilelist),ncol=18)
> SNPnumbers<-rep(0,22)
> HSTrow<-0
```

Later, we are going to contrast the estimate obtained from SNPs in high-GC regions with that from those in low-GC regions. We prepare for this analysis now

```
> HetSNPTableHighGC<-matrix(0,nrow=length(bamfilelist),ncol=18)
> HetSNPTableLowGC<-matrix(0,nrow=length(bamfilelist),ncol=18)
> SNPnumbersHigh<-rep(0,22)
> SNPnumbersLow<-rep(0,22)
```

Now we cycle through and process each bam file

```
> for(sample in bamfilelist){
+   HSTrow<-HSTrow+1
+   cat(sample,"\n")
+   bamfile=paste("../finaldupsanalysis/data/snpbams/",sample,sep="")
+
+   # look to see which of the candidate SNPs is heterozygous in this sample.
+   # Note that this is based on all the reads not just the duplicate fragments,
+   # so should not greatly bias matters.
+
+   fls <- PileupFiles(bamfile)
+   which<-GRanges(snplist[1:2500,1],IRanges(snplist[1:2500,3],snplist[1:2500,3]))
+   PUP <- ApplyPileupsParam(which=which, yieldSize=1000000L, yieldBy="position", what="seq",maxDepth=200,
+   outres <- applyPileups(fls,(function(x){x[["seq"]]}),param=PUP)
+
+   # Our requirement is that the minor allele frequency is greater than 0.4 -
+   # quite a stringent criterion
+
+   usesnplist<-snplist[
+     which(apply((outres)[[1]],3,secondbiggest)/apply((outres)[[1]],3,sum)>0.4),]
+
+     useGCNo<-GCNo[which(apply((outres)[[1]],3,secondbiggest)/apply((outres)[[1]],3,sum)>0.4)]
+     SNPnumbersHigh[HSTrow]<-sum(useGCNo>=199)
+     SNPnumbersLow[HSTrow]<-sum(useGCNo<199)
+
+
+   SNPnumbers[HSTrow]<-dim(usesnplist)[1]
+   # now we are going to count up the numbers of duplicates that share the
+   # same allele and the numbers that do not.
+
+   truetally<-0
+   falsetally<-0
+
+   store<-rep(0,dim(usesnplist)[1])
+   # for each allele on the list
+   for(snp in 1:dim(usesnplist)[1]){
+     newwhich<-GRanges(usesnplist[snp,1], IRanges(usesnplist[snp,3], usesnplist[snp,3]))
+
+     #these are the reads that were marked as duplicates
+     ydfile<-scanBam(bamfile,
+                      param=ScanBamParam(flag=scanBamFlag(isDuplicate=T),
+                                         simpleCigar=T,what=c("pos","mpos","seq"),
+                                         which=newwhich))[[1]]
+
+     #these are the reads that were not
+     ndfile<-scanBam(bamfile,
+                     param=ScanBamParam(flag=scanBamFlag(isDuplicate=F),
+                                        simpleCigar=T,what=c("pos","mpos","seq"),
+                                        which=newwhich))[[1]]
+
+     # Any fragement marked as a duplicate must be a duplicate of a fragment
+     # that is not marked as a duplicate. We just want to keep a matched set
+     # of duplicates and the fragments of which they are duplicates
+
+     ykey<-paste(ydfile$pos,ydfile$mpos)
+     nkey<-paste(ndfile$pos,ndfile$mpos)
+
+     usekey<-unique(ykey)
+
+     # Now, assuming that we see some duplicates, we are going to go through them
+     # and compare the alleles
+     if(length(usekey)>0){
+       for(key in usekey){
+
+         storealleles<-NULL
+         for(p in which(ykey==key)){
+           storealleles<-c(storealleles, substr(ydfile$seq[p],
+                                              usesnplist[snp,3]-ydfile$pos[p]+1,
+                                              usesnplist[snp,3]-ydfile$pos[p]+1))
+         }
+         for(p in which(nkey==key)){
+           storealleles<-c(storealleles,substr(ndfile$seq[p],
+                                             usesnplist[snp,3]-ndfile$pos[p]+1,
+                                             usesnplist[snp,3]-ndfile$pos[p]+1))
+         }
+
+         ortab<-outres[[1]][,,as.numeric(rownames(usesnplist)[snp])]
+         usebase<-names(sort(ortab,decreasing=T))[1:2]
+
+
+         allelecounts<-c(sum(storealleles==usebase[1]),sum(storealleles==usebase[2]))
+         if(allelecounts[2]>allelecounts[1]){allelecounts<-allelecounts[2:1]}
+
+         HetSNPTable[HSTrow,match(paste(allelecounts,collapse=":"),ACvec)]<-
+           HetSNPTable[HSTrow,match(paste(allelecounts,collapse=":"),ACvec)]+1
+         if(useGCNo[snp]>=199){
+           HetSNPTableHighGC[HSTrow,match(paste(allelecounts,collapse=":"),ACvec)]<-
+             HetSNPTableHighGC[HSTrow,match(paste(allelecounts,collapse=":"),ACvec)]+1}
+
+           if(useGCNo[snp]<199){
+             HetSNPTableLowGC[HSTrow,match(paste(allelecounts,collapse=":"),ACvec)]<-
+               HetSNPTableLowGC[HSTrow,match(paste(allelecounts,collapse=":"),ACvec)]+1}
+       }
+     }
+   }
+ }
```

Finally we'll write out the table.

```
> HetSNPTable<-cbind(SNPnumbers,HetSNPTable)
> HetSNPTableHighGC<-cbind(SNPnumbersHigh,HetSNPTableHighGC)
> HetSNPTableLowGC<-cbind(SNPnumbersLow,HetSNPTableLowGC)
> colnames(HetSNPTable)<-c("NoSNPs", "AA", "AB",
+                            "AAA", "AAB",
+                            "AAAA", "AAAB", "AABB",
+                            "AAAAA", "AAAAB", "AAABB",
+                            "AAAAAA", "AAAAAB", "AAAABB", "AAABBB",
+                            "AAAAAAA", "AAAAAAB", "AAAAABB", "AAAABBB")
> colnames(HetSNPTableLowGC)<-c("NoSNPs", "AA", "AB",
+                          "AAA", "AAB",
+                          "AAAA", "AAAB", "AABB",
+                          "AAAAA", "AAAAB", "AAABB",
+                          "AAAAAA", "AAAAAB", "AAAABB", "AAABBB",
+                          "AAAAAAA", "AAAAAAB", "AAAAABB", "AAAABBB")
> colnames(HetSNPTableHighGC)<-c("NoSNPs", "AA", "AB",
+                          "AAA", "AAB",
+                          "AAAA", "AAAB", "AABB",
+                          "AAAAA", "AAAAB", "AAABB",
+                          "AAAAAA", "AAAAAB", "AAAABB", "AAABBB",
+                          "AAAAAAA", "AAAAAAB", "AAAAABB", "AAAABBB")
> write.table(HetSNPTable,file="HetSNPDups.txt",sep="\t")
> write.table(HetSNPTableHighGC,file="HetSNPDupsHighGC.txt",sep="\t")
> write.table(HetSNPTableLowGC,file="HetSNPDupsLowGC.txt",sep="\t")
```

4.4 Exploration of SNP numbers

4.4.1 Preliminaries

We now load in the summary of the numbers of SNPs being investigated, and generate the various results given in the section "SNP numbers and duplicate numbers" in the main manuscript.

```
> HetSNPTable<-read.delim(file.path(EDpath,"HetSNPDups.txt"),as.is=T)
```

We define the duplicate rate as the number of duplicates identified within the unmasked regions of the genome divided by the number of read-pairs examined within those regions.

```
> ResDupR<-readPairDuplicates[,9]/readPairsExamined[,9]
```

We define the residual read depth as 200 (the number of bases sequenced per read-pair) multiplied by the number of read-pairs examined, divided by the length of the unmasked genome.

```
> ResDepth<-round(200*readPairsExamined[,9]/effectiveGenomeSize[9],2)
```

4.4.2 How many SNPs do we see?

"From the 2,500 sites considered, the median number of heterozygous SNPs identified per sample is 1,009 (range 942 to 1,093)."

```
> summary(HetSNPTable[,1])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  942.0   988.5  1009.0  1015.0  1034.0  1093.0 
```

"Average read depth (as measured by the number of reads mapping to the regions of the genome that we are not masking) is correlated with the number of heterozygous SNPs (correlation = 0.52, p = 0.014): a reflection on our stringent calling criterion."

```
> summary(lm(HetSNPTable[,1]~ResDepth+ResDupR))
```

```
Call:
lm(formula = HetSNPTable[, 1] ~ ResDepth + ResDupR)

Residuals:
    Min      1Q  Median      3Q     Max 
-64.061 -17.787   1.512  25.908  50.541 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  803.774     81.198   9.899  6.2e-09 ***
ResDepth       2.714      1.033   2.627   0.0166 *  
ResDupR       -1.191    262.094  -0.005   0.9964    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 34.61 on 19 degrees of freedom
Multiple R-squared:  0.2671,	Adjusted R-squared:   0.19 
F-statistic: 3.463 on 2 and 19 DF,  p-value: 0.0522 
```

```
> cor.test(ResDepth,HetSNPTable[,1])
```

```
	Pearson's product-moment correlation

data:  ResDepth and HetSNPTable[, 1] 
```

```
t = 2.7001, df = 20, p-value = 0.01377
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1217976 0.7705574
sample estimates:
      cor
0.5168646
```

4.4.3 How many duplicates read pairs are we using?

“The number of sets of duplicate read fragments observed to overlie the heterozygous SNP sites varies from 950 to 6,740.”

```
> summary(apply(HetSNPTable[,-1],1,sum))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    950    1891    2608    2785    3516    6740
```

“The number of duplicate sets is unsurprisingly dependent on the duplicate rate ($p = 3.2x10^{-14}$) and the depth of coverage ($p = 0.001$), but not directly on the number of heterozygous sites being considered (although this is correlated with the depth of coverage).”

```
> summary(lm(apply(HetSNPTable[,-1],1,sum)~ResDupR+HetSNPTable[,1]+ResDepth))
```

```
Call:
lm(formula = apply(HetSNPTable[, -1], 1, sum) ~ ResDupR + HetSNPTable[,
    1] + ResDepth)

Residuals:
     Min       1Q   Median       3Q      Max
-1018.12   -72.60    95.91   150.38   244.39

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      -6037.302   1688.383  -3.576  0.00216 **
ResDupR          46839.067   2196.276  21.327 3.17e-14 ***
HetSNPTable[, 1]     3.012      1.922   1.567  0.13459
ResDepth            39.269     10.109   3.885  0.00109 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 290 on 18 degrees of freedom
Multiple R-squared:  0.9655,       Adjusted R-squared:  0.9597
F-statistic: 167.8 on 3 and 18 DF,  p-value: 2.415e-13
```

4.4.4 How do the duplicates read pairs break down into pairs, triples, etc.?

“Of the total of 61,272 sets of duplicate fragments identified among the 22 samples, ...”

```
> sum(HetSNPTable[,-1])
```

```
[1] 61272
```

“...the vast majority (57,883 or 94%) are duplicate pairs,”

```
> sum(HetSNPTable[,2:3])
```

```
[1] 57883
```

```
> round(100*sum(HetSNPTable[,2:3])/sum(HetSNPTable[,-1]))
```

```
[1] 94
```

"...3,189 (5%) are triples,..."

```
> sum(HetSNPTable[,4:5])
```

```
[1] 3189
```

```
> round(100*sum(HetSNPTable[,4:5])/sum(HetSNPTable[,-1]))
```

```
[1] 5
```

"...186 (0.3%) are quartets,..."

```
> sum(HetSNPTable[,6:8])
```

```
[1] 186
```

```
> round(100*sum(HetSNPTable[,6:8])/sum(HetSNPTable[,-1]),1)
```

```
[1] 0.3
```

"and the greatest number of fragments seen in a duplicate set is seven (one instance)."

```
> sum(HetSNPTable[,16:19])
```

```
[1] 1
```

5 Duplicate rate estimates

5.1 Estimating the fragmentation-duplicate proportion

To prepare for the probability calculations, we first generate the list of sets of partitions possible with each value of M (the number of fragments in a duplicate set). As previously noted, $M = 7$ is the maximum observed in our data.

```
> partitionlist<-genPL(7)
> CoefList<-genCoefs(partitionlist)
```

Note that the time taken to generate the partition list is exponential in terms of M (Figure 4). Note that we do not perform these timings within the sweave as it would be a disproportionate burden on compilation time, but rather we load in values observed previously. On a laptop of reasonable specification, with no parallelization, the list can be generated up to $M = 60$ in reasonable time (under 10 minutes). Anybody fortunate enough to have depth of coverage such that this is not sufficient may need to streamline this approach. Note also that the recursive generation of coefficients is more expensive, but we have already seen that it is not necessary to perform the calculations in this manner.

```
> PLtimes<-c(.001,.003,.007,.03,.119,.584,1.97,6.98,18.9,67.529,152.94,541.793)
> CLtimes<-c(.003,.052,1.088,16.711,212.698,NA,NA,NA,NA,NA,NA,NA)
> Mvals<-c(5,10,15,20,25,30,35,40,45,50,55,60)
> plot(Mvals,log10(PLtimes),ylab="time (seconds)",xlab="M",axes=F)
> axis(1)
> axis(2,10^seq(-3,2,1),at=seq(-3,2,1),las=2)
> box()
> points(Mvals,log10(CLtimes),pch=17)
> lm(log10(PLtimes)~Mvals)
```

```
Call:
lm(formula = log10(PLtimes) ~ Mvals)

Coefficients:
```

```
(Intercept)        Mvals
    -3.5806       0.1071
```

```
> lm(log10(CLtimes)~Mvals)
```

```
Call:
lm(formula = log10(CLtimes) ~ Mvals)

Coefficients:
(Intercept)        Mvals
    -3.7064       0.2442
```

```
> abline(-3.58,0.1071)
> abline(-3.7064,0.2442,lwd=2)
> legend("bottomright",pch=c(1,17),legend=c("Partition List","Coefficients"))
```

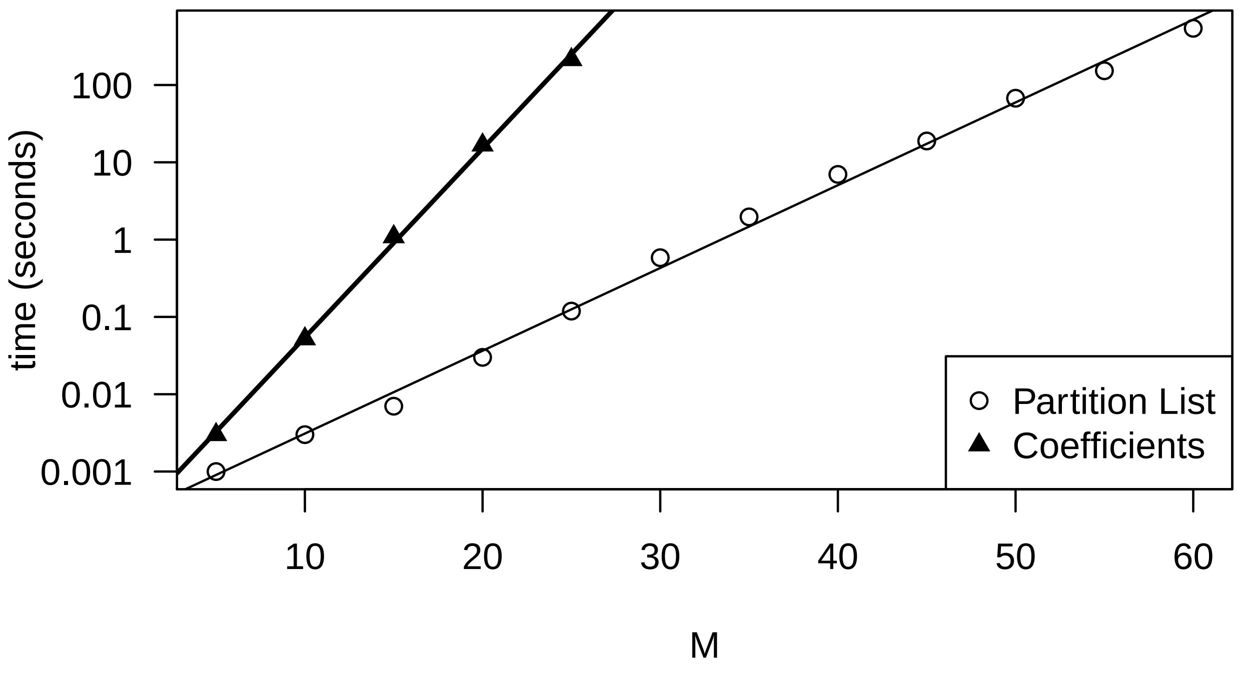


Figure 4: **Time taken for calculations on a laptop with a 1.7 GHz Intel Core i5 processor and 4 GB 1333 MHz DDR3 memory.**

We now calculate the maximum likelihood estimates of the proportions of duplicates that are fragmentation duplicates.

```
> FDvec<-rep(0,22)
> for(j in 1:endloop){
+     myseq<-seq(0,0.2,length.out=1000)
+     vals <- sapply(myseq, propllik, x = HetSNPTable[j,-1])
+     FDvec[j]<-myseq[which.max(vals)]
+ }
```

5.2 The estimates

We generate three estimates of the duplication rate: the basic estimate that ignores all of the factors we have discussed in this document, the residual estimate that adopts the basic approach, but applied only to counts from regions of the genome that are not masked, and the Fragmentation-duplicate-corrected estimate.

```
>   BasicEst<-(readPairDuplicates[,1]-OpticalPairDuplicates[,1])/
+   (readPairsExamined[,1]-OpticalPairDuplicates[,1])
>   ResidEst<-(readPairDuplicates[,9]-OpticalPairDuplicates[,9])/
+   (readPairsExamined[,9]-OpticalPairDuplicates[,9])
>   FDCorEst<-(1-FDvec)*ResidEst
```

We now generate Figure 2 from the main manuscript (Figure 5 in this document).

```
>   par(mfrow=c(1,2))
>   plot(c(1,3),c(0,0.15),type="n",ylab="duplicate rate",xlab="",axes=F,yaxs="i")
>   axis(2)
>   box()
>   for(i in seq(0,.15,.03)){
+     rect(-1,i-.015,5,i,border="grey95",col="grey95")
+   }
>   for(i in 1:22){
+     tcol<-"black"
+     if(WeaverSuppTable1[i,12]=="Blood"){tcol<-"red"}
+     lines(1:3,c(BasicEst[i],ResidEst[i],FDCorEst[i]),type="b",pch=16,col=tcol)
+   }
>   axis(1,at=1:3,labels=c("Standard\n estimate","Residual\n estimate","Corrected\n estimate"),las=3)
>   plot(ResidEst/BasicEst,FDCorEst/ResidEst, xlab="Residual/Basic estimates",
+        ylab="Corrected/Residual estimates",pch=20)
```

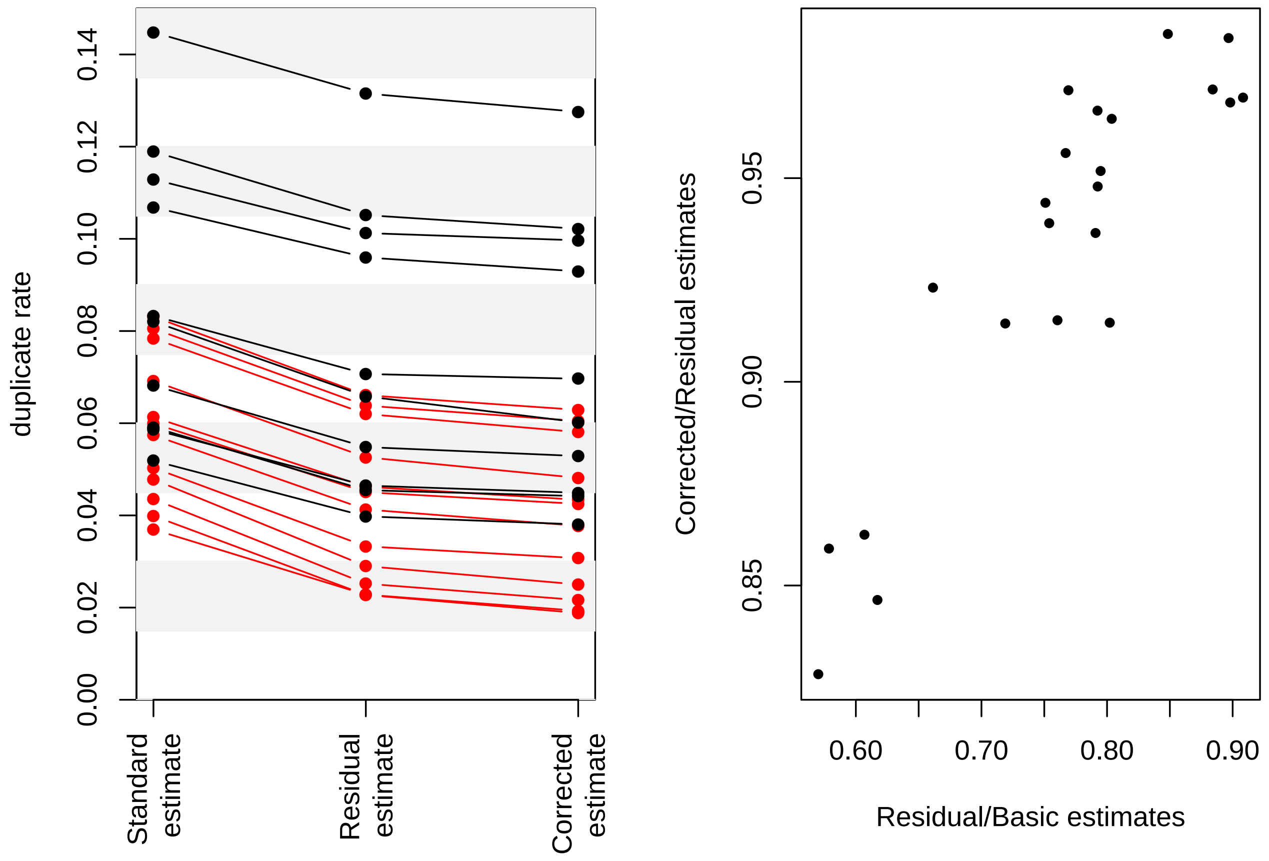


Figure 5: **The different estimates of duplication rate**

6 Observations on consistency

6.1 Confirming that the two approaches match when M equals 2

Since $M = 2$, we only have two possible allele patterns, and we can write down $\Pr(AP_k|P_D)$ in a straightforward manner.

$$P(AA \mid P_D) = \frac{1}{2}(1 + P_D) \tag{5}$$
$$P(AB \mid P_D) = \frac{1}{2}(1 - P_D) \tag{6}$$

The log-likelihood is then

$$l(P_D) = n_{AA} \log(\frac{1}{2}(1 + P_D)) + n_{AB} \log(\frac{1}{2}(1 - P_D)), \tag{7}$$

and the first derivative is

$$\frac{dl}{dP_D} = \frac{n_{AA}}{(1 + P_D)} + \frac{n_{AB}}{(1 - P_D)}. \tag{8}$$

If we look to find the MLE of P_D,

$$0 = n_{AA}(1 - \hat{P}_D) + n_{AB}(1 + \hat{P}_D), \tag{9}$$

then we find it is equal to

$$\hat{P}_D = \frac{n_{AA} + n_{AB}}{N} = 1 - \frac{2n_{AB}}{N} \tag{10}$$

as required to match equation 1.

6.2 The estimate is well-behaved when M equals 3

The case when $M = 3$ is depicted in Figure 6. In this simple case, there are only two patterns of alleles that can be observed: 'AAA' and 'AAB'.

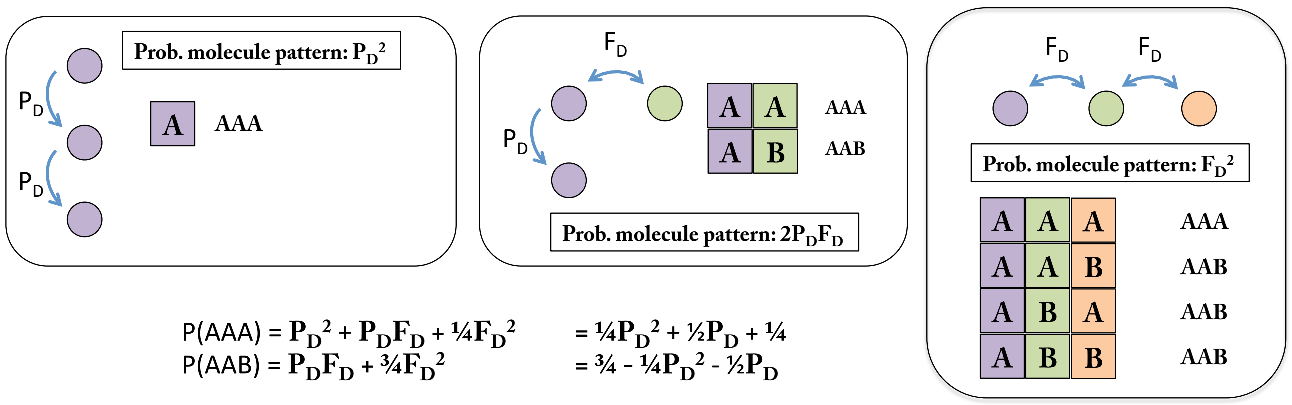

P_D
P_D
Prob. molecule pattern: P_D^2
A AAA
F_D
P_D
A A AAA
A B AAB
Prob. molecule pattern: $2P_DF_D$
F_D F_D
Prob. molecule pattern: F_D^2
A A A AAA
A A B AAB
A B A AAB
A B B AAB
$P(AAA) = P_D^2 + P_DF_D + ¼F_D^2 = ¼P_D^2 + ½P_D + ¼$
$P(AAB) = P_DF_D + ¾F_D^2 = ¾ - ¼P_D^2 - ½P_D$

Figure 6: Details of the case when $M = 3$.

The log-likelihood is then given as

$$l_3(P_D) = n_{AAA} \log(P(AAA \mid P_D)) + n_{AAB} \log(P(AAB \mid P_D)). \tag{11}$$

Since this is of the form

$$l = n \log f(P_D) + m \log(1 - f(P_D)), \tag{12}$$

we can see that the first derivative is of the form

$$nf'(P_D) - (n+m)f'(P_D)f(P_D) \tag{13}$$

So,

$$\begin{aligned}\frac{dl}{dP_D} =& n_{AAA}(\frac{1}{2}P_D + \frac{1}{2}) - \\ & (n_{AAA} + n_{AAB})(\frac{1}{2}P_D + \frac{1}{2})(\frac{1}{4}P_D^2 + \frac{1}{2}P_D + \frac{1}{4})\end{aligned} \tag{14}$$

and since P_D cannot equal -1, we have

$$0 = \hat{P}_D^2 + 2\hat{P}_D + 1 - \frac{4n_{AAA}}{n_{AAA} + n_{AAB}} \tag{15}$$

and

$$\hat{P}_D = -1 \pm 2\sqrt{\frac{n_{AAA}}{n_{AAA} + n_{AAB}}} \tag{16}$$

From Figure 6, we can see that for a large numbers of trios, $N = n_{AAA} + n_{AAB}$, n_{AAA} will be approximately equal to $N(\frac{1}{4}P_D^2 + \frac{1}{2}P_D + \frac{1}{4})$, which if substituted into the equation for $\hat{P}_D$ gives the estimate $\hat{P}_D = P_D$.

6.3 Inconsistency with a naive approach when M=3

Were we to adopt a naive approach, we might simply deal with cases where $M > 2$ by discarding all but 2 fragments in the set at random.

If $M = 3$ we would expect to see

$$n_{AA} = n_{AAA} + \frac{1}{3}n_{AAB} \tag{17}$$

$$n_{AB} = \frac{2}{3}n_{AAB} \tag{18}$$

and so the estimate of P_D will be

$$\hat{P}_D = 1 - \frac{4n_{AAB}}{3N} \tag{19}$$

which in general will not equal the estimate obtained from using all of the data (equation 16).

6.4 Agreement of estimate from M equal to 2, with estimate from M greater than 2

We can separate out the estimate when $M = 2$ from that when $M > 2$. First of all we calculate the estimates only from the $M = 2$ data.

```
> FDvec2<-rep(0,22)
> for(j in 1:endloop){
+   vals<-rep(0,1000)
+   myseq<-seq(0,0.2,length.out=1000)
+   for(i in 1:1000){
+     tmpx<-HetSNPTable[j,-1]
+     tmpx[3:18]<-0
+     vals[i]<-propllik(tmpx,myseq[i])
+   }
+
+   FDvec2[j]<-myseq[which.max(vals)]
+ }
```

and then from the $M > 2$ data.

```
> FDvecN2<-rep(0,22)
> for(j in 1:endloop){
+   vals<-rep(0,1000)
+   myseq<-seq(0,0.2,length.out=1000)
+   for(i in 1:1000){
+     tmpx<-HetSNPTable[j,-1]
+     tmpx[1:2]<-0
+     vals[i]<-propllik(tmpx,myseq[i])
+   }
+
+   FDvecN2[j]<-myseq[which.max(vals)]
+ }
```

If we do this, we see that while there is some noise (to be expected as we have seen that there is a 19:1 ratio in terms of the numbers of fragment sets from which we draw our estimates) there is no evidence of bias (Figure 7).

```
> plot(FDvec2,FDvecN2,xlab="estimates from sets of 2 duplicate fragments",
+      ylab="estimates from sets of >2 duplicate fragments")
> abline(0,1)
```

6.5 Agreement of estimate from High-GC SNPs, and Low-GC SNPs

We observe a median of $199/500$ Gs and Cs in our 2500 SNP regions and so divide up the SNPs into two sets: i) the 1242 SNP regions with GC content less than $199/500$, and ii) the 1258 SNP regions with GC content of $199/500$ or greater. We first load in the summary tables of SNP pattern counts.

```
> HetSNPTableHighGC<-read.delim(file.path(EDpath,"HetSNPDupsHighGC.txt"),as.is=T)
> HetSNPTableLowGC<-read.delim(file.path(EDpath,"HetSNPDupsLowGC.txt"),as.is=T)
```

We now process both tables using the methods developed for the main table.

```
> FDvecHigh<-rep(0,22)
> FDvecLow<-rep(0,22)
> for(j in 1:22){
+   #cat(j,"\t")
+   myseq<-seq(0,0.2,length.out=1000)
+   valsHigh <- sapply(myseq, propllik, x = HetSNPTableHighGC[j,-1])
+   valsLow <- sapply(myseq, propllik, x = HetSNPTableLowGC[j,-1])
+   FDvecHigh[j]<-myseq[which.max(valsHigh)]
+   FDvecLow[j]<-myseq[which.max(valsLow)]
+ }
```

We note that there is good agreement, and no evidence of bias, in the estimate of the proportion of fragmentation duplicates when we compare the high-GC and low-GC values.

```
>   plot(FDvecHigh,FDvecLow,xlab="estimates from duplicates in 'high GC' regions",
+        ylab="estimates from duplicates in 'low GC' regions")
> abline(0,1)
```

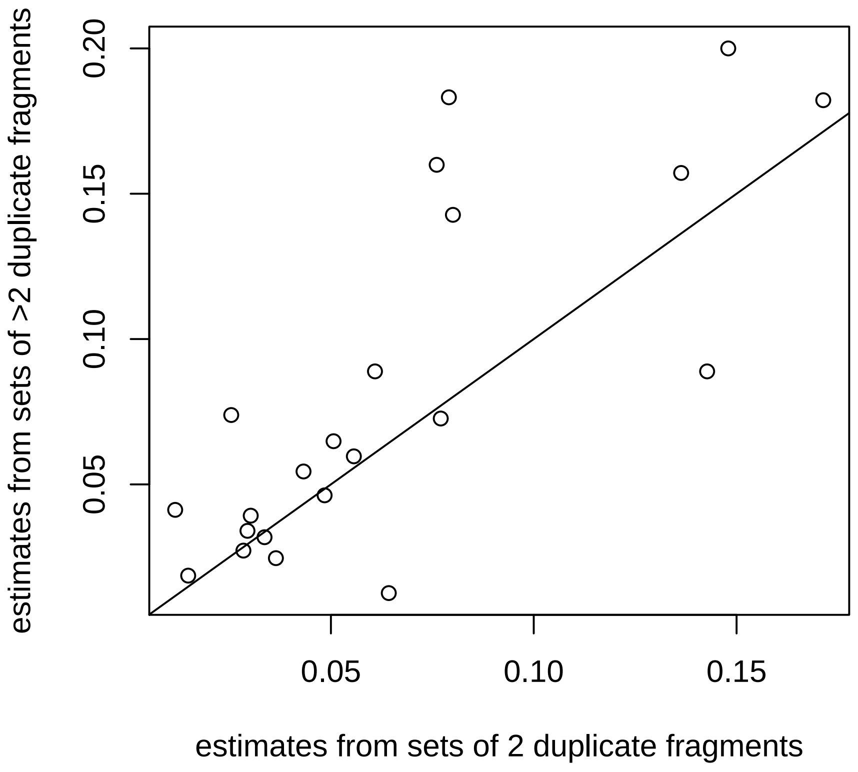
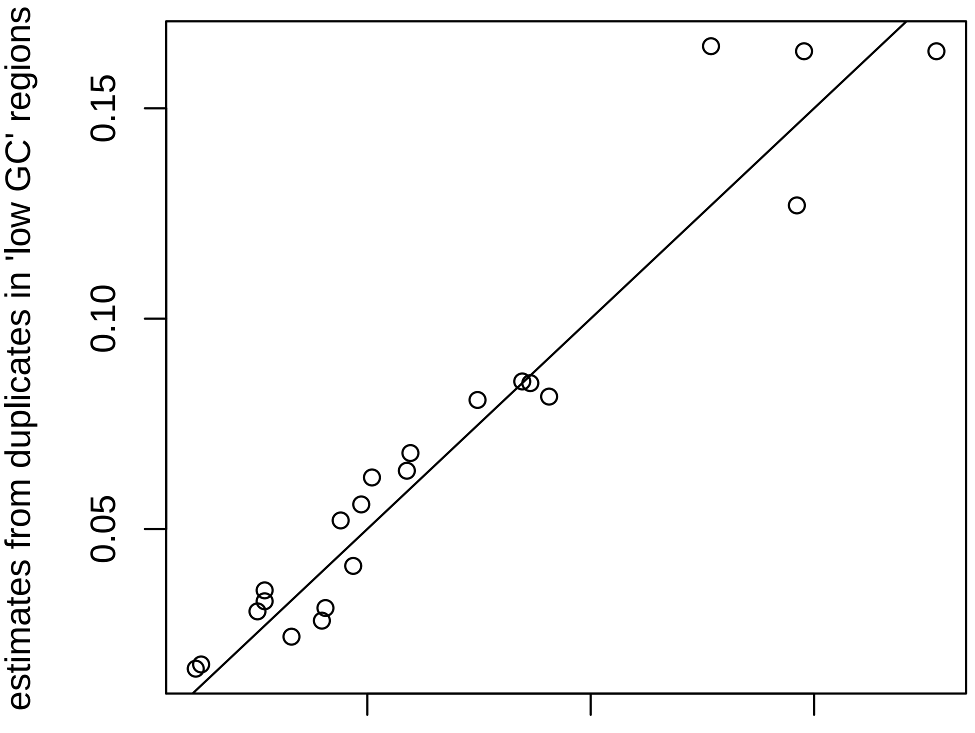


Figure 7: **Consistency of estimates from sets of 2 fragments and sets of more than 2 fragments.**



7 Example for a tumour sample

7.1 The Sample

We have, until now, been considering normal (diploid) samples. We now demonstrate the approach applied to a tumour sample: Case SS6003314 from the same paper and manuscript as the normal data. SS6003314 is a broadly tetraploid tumour with approximately 74% cellularity. It has some regions exhibiting sub-clonal copy number changes, but when we plot minor allele frequency against tumour depth 8, it notably has clear areas of diploid allelic-balance (AB), and tetraploid allelic-balance (AABB). Bed files defining these regions (`ABbed.bed` and `AABBbed.bed`) are included in the `extdata` folder of the LynchSmithEldridgeTavareFragDup package.

We load in Picard data for the entire library (duplicate rate: 5.2%), and for two sets of regions: i) The regions that are classified as AB (after removing masked regions - duplicate rate: 3.7%), and ii) The regions that are classified as AABB (after removing masked regions - duplicate rate: 3.8%). Note that this is a low duplication rate compared to those observed in libraries from normal tissue/blood. We attribute the change going from the entire library to the subsets as being due to the removal of masked regions; the AABB regions alone represent more than half of the sequencing library. We expect there to be a higher duplication rate in the AABB regions than in the AB region because of fragmentation duplicates. That it is such a small change suggests that the fragmentation duplicate rate will be small in this case.

```
> TumourPicard<-read.csv(file.path(EDpath,"Tumour","Picard.csv"),as.is=T)
> TumourPicard
```

```
         LIBRARY UNPAIRED_READS_EXAMINED READ_PAIRS_EXAMINED UNMAPPED_READS
1      SS6003314                10432535           994242462       61864339
2   SS6003314-AB                    1576              417850           1576
3 SS6003314-AABB                 2438478           588331041        2438895
  UNPAIRED_READ_DUPLICATES READ_PAIR_DUPLICATES READ_PAIR_OPTICAL_DUPLICATES
1                  7153353             51808523                       455107
2                      402                15495                          188
3                   872992             22418117                       270067
  PERCENT_DUPLICATION ESTIMATED_LIBRARY_SIZE
1            0.055415             9281651350
2            0.037493                5558009
3            0.038766             7609634846
```

```
> # and the duplicate rates
> round((TumourPicard[,6]-TumourPicard[,7])/(TumourPicard[,3]-TumourPicard[,7]),3)
```

```
[1] 0.052 0.037 0.038
```

7.2 The SNPs

Tumour samples require bespoke lists, so we restrict ourselves to sites that we know to be heterozygous. We can define two sets of heterozygous SNPs for the AB (985 SNPs), and AABB (a sample of $10,000$ SNPs) regions.

```
> snplistAABB<-read.delim(file.path(EDpath,"Tumour","usedupAABB.txt"),as.is=T,header=F)
> snplistAABB<-snplistAABB[,c(1,2,2)]
> snplistAB<-read.delim(file.path(EDpath,"Tumour","usedupAB.txt"),as.is=T,header=F)
> snplistAB<-snplistAB[,c(1,2,2)]
```

7.3 Counting the duplicates

As before, we are unable to distribute the raw data, which are archived in the European Genome Archive [EGA:EGAD00001000704]. The following code will generate the summary table in the manner we have come to expect.

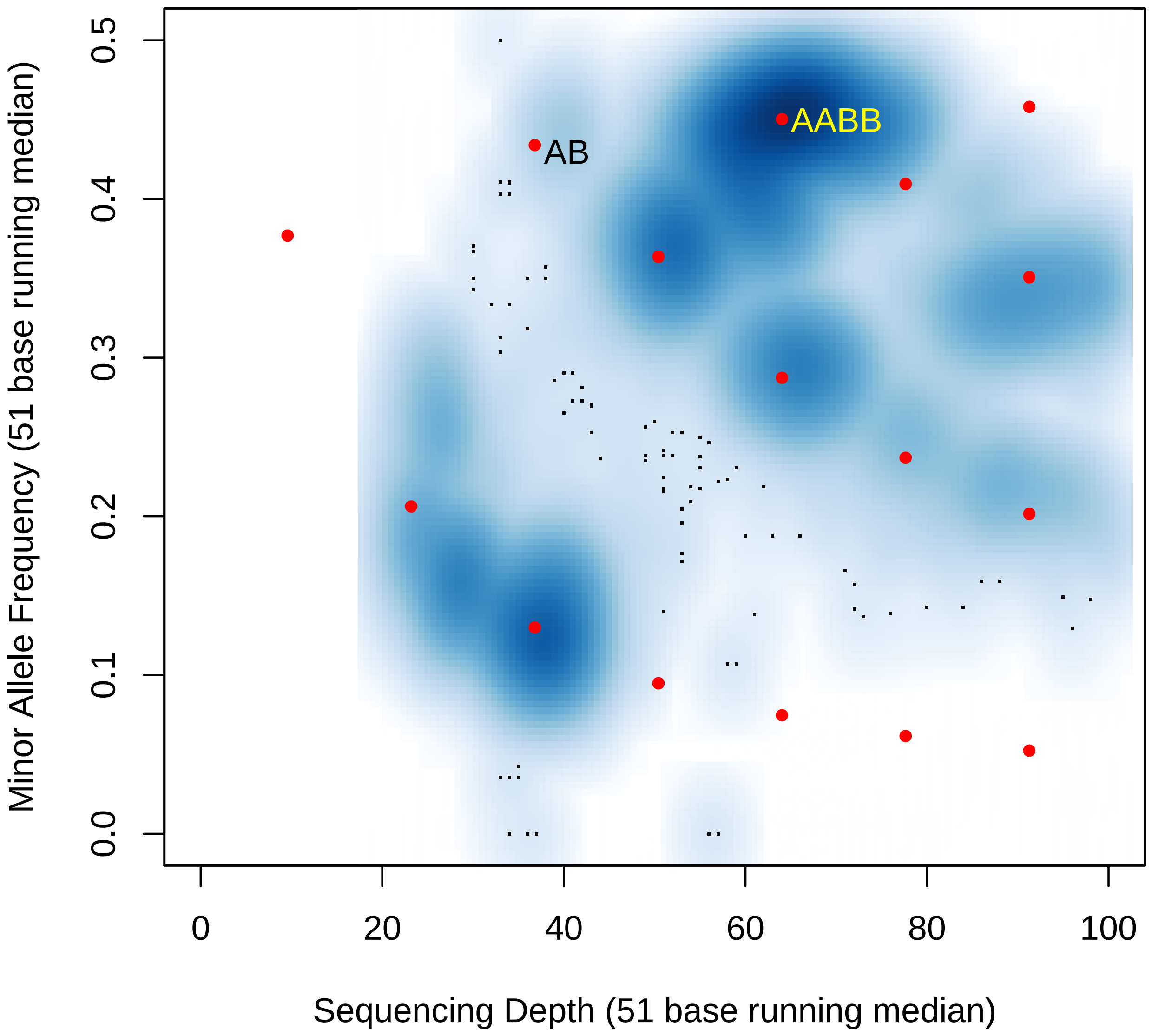


Figure 8: A plot of minor allele frequency against sequencing depth reveals regions of consistent copy number.

```
> HetSNPTableAABB<-matrix(0,nrow=2,ncol=18)
> bamfile="SS6003314.bam"
> usesnplist<-snplistAABB
> #SNPnumbers[HSTrow]<-dim(usesnplist)[1]
> # now we are going to count up the numbers of duplicates that share the
> # same allele and the numbers that do not.
>
> # for each allele on the list
> for(snp in 1:dim(usesnplist)[1]){
```

```
+   if(100*trunc(snp/100)==snp){cat(snp,"\t")}
+   newwhich<-GRanges(usesnplist[snp,1], IRanges(usesnplist[snp,3], usesnplist[snp,3]))
+
+   #these are the reads that were marked as duplicates
+   ydfile<-scanBam(bamfile,
+                   param=ScanBamParam(flag=scanBamFlag(isDuplicate=T),
+                                      simpleCigar=T,what=c("pos","mpos","seq"),
+                                      which=newwhich))[[1]]
+
+   #these are the reads that were not
+   ndfile<-scanBam(bamfile,
+                   param=ScanBamParam(flag=scanBamFlag(isDuplicate=F),
+                                      simpleCigar=T,what=c("pos","mpos","seq"),
+                                      which=newwhich))[[1]]
+
+   # Any fragement marked as a duplicate must be a duplicate of a fragment
+   # that is not marked as a duplicate. We just want to keep a matched set
+   # of duplicates and the fragments of which they are duplicates
+
+   ykey<-paste(ydfile$pos,ydfile$mpos)
+   nkey<-paste(ndfile$pos,ndfile$mpos)
+
+   usekey<-unique(ykey)
+
+   # Now, assuming that we see some duplicates, we are going to go through them
+   # and compare the alleles
+   if(length(usekey)>0){
+     for(key in usekey){
+
+       storealleles<-NULL
+       for(p in which(ykey==key)){
+         storealleles<-c(storealleles, substr(ydfile$seq[p],
+                                              usesnplist[snp,3]-ydfile$pos[p]+1,
+                                              usesnplist[snp,3]-ydfile$pos[p]+1))
+       }
+       for(p in which(nkey==key)){
+         storealleles<-c(storealleles,substr(ndfile$seq[p],
+                                             usesnplist[snp,3]-ndfile$pos[p]+1,
+                                             usesnplist[snp,3]-ndfile$pos[p]+1))
+       }
+
+       ortab<-outres[[1]][,,as.numeric(rownames(usesnplist)[snp])]
+       usebase<-names(sort(ortab,decreasing=T))[1:2]
+
+
+       allelecounts<-c(sum(storealleles==usebase[1]),sum(storealleles==usebase[2]))
+       if(allelecounts[2]>allelecounts[1]){allelecounts<-allelecounts[2:1]}
+
+       HetSNPTableAABB[1,match(paste(allelecounts,collapse=":"),ACvec)]<-
+         HetSNPTableAABB[1,match(paste(allelecounts,collapse=":"),ACvec)]+1
+     }
+   }
+ }
> usesnplist<-snplistAB
> #SNPnumbers[HSTrow]<-dim(usesnplist)[1]
> # now we are going to count up the numbers of duplicates that share the
> # same allele and the numbers that do not.
>
> # for each allele on the list
> for(snp in 1:dim(usesnplist)[1]){
+   if(100*trunc(snp/100)==snp){cat(snp,"\t")}
+   newwhich<-GRanges(usesnplist[snp,1], IRanges(usesnplist[snp,3], usesnplist[snp,3]))
+
+   #these are the reads that were marked as duplicates
+   ydfile<-scanBam(bamfile,
+                   param=ScanBamParam(flag=scanBamFlag(isDuplicate=T),
+                                      simpleCigar=T,what=c("pos","mpos","seq"),
+                                      which=newwhich))[[1]]
+
+   #these are the reads that were not
+   ndfile<-scanBam(bamfile,
+                   param=ScanBamParam(flag=scanBamFlag(isDuplicate=F),
+                                      simpleCigar=T,what=c("pos","mpos","seq"),
+                                      which=newwhich))[[1]]
+
+   # Any fragement marked as a duplicate must be a duplicate of a fragment
+   # that is not marked as a duplicate. We just want to keep a matched set
+   # of duplicates and the fragments of which they are duplicates
+
+   ykey<-paste(ydfile$pos,ydfile$mpos)
+   nkey<-paste(ndfile$pos,ndfile$mpos)
+
+   usekey<-unique(ykey)
+
+   # Now, assuming that we see some duplicates, we are going to go through them
+   # and compare the alleles
+   if(length(usekey)>0){
+     for(key in usekey){
+
+       storealleles<-NULL
+       for(p in which(ykey==key)){
+         storealleles<-c(storealleles, substr(ydfile$seq[p],
+                                              usesnplist[snp,3]-ydfile$pos[p]+1,
+                                              usesnplist[snp,3]-ydfile$pos[p]+1))
+       }
+       for(p in which(nkey==key)){
+         storealleles<-c(storealleles,substr(ndfile$seq[p],
+                                             usesnplist[snp,3]-ndfile$pos[p]+1,
+                                             usesnplist[snp,3]-ndfile$pos[p]+1))
+       }
+
+       ortab<-outres[[1]][,,as.numeric(rownames(usesnplist)[snp])]
+       usebase<-names(sort(ortab,decreasing=T))[1:2]
+
+
+       allelecounts<-c(sum(storealleles==usebase[1]),sum(storealleles==usebase[2]))
+       if(allelecounts[2]>allelecounts[1]){allelecounts<-allelecounts[2:1]}
+
+       HetSNPTableAABB[2,match(paste(allelecounts,collapse=":"),ACvec)]<-
+         HetSNPTableAABB[2,match(paste(allelecounts,collapse=":"),ACvec)]+1
+     }
+   }
+ }
> HetSNPTableAABB<-cbind(c(10000,985),HetSNPTableAABB)
> colnames(HetSNPTableAABB)<-c("NoSNPs", "AA", "AB",
+                          "AAA", "AAB",
+                          "AAAA", "AAAB", "AABB",
+                          "AAAAA", "AAAAB", "AAABB",
+                          "AAAAAA", "AAAAAB", "AAAABB", "AAABBB",
+                          "AAAAAAA", "AAAAAAB", "AAAAABB", "AAAABBB")
> rownames(HetSNPTableAABB)<-c("AABB","AB")
> write.table(HetSNPTableAABB,file="HetSNPDupsCancer.txt",sep="\t")
```

Assuming that the user hasn't obtained the raw data, we now load in the precompiled summary table.

```
> HetSNPTableAABB<-read.delim(file.path(EDpath,"Tumour","HetSNPDupsCancer.txt"),as.is=T)
```

7.4 Estimating the fragmentation duplicate proportion

Processing this in the same manner as previous allele pattern counts...

```
> FDvecCancer<-rep(0,2)
> for(j in 1:2){
+   cat(j,"\t")
+   vals<-rep(0,1000)
+   myseq<-seq(0,0.2,length.out=1000)
+   for(i in 1:1000){
+     vals[i]<-propllik(HetSNPTableAABB[j,-1],myseq[i])
+    }
+   FDvecCancer[j]<-myseq[which.max(vals)]
+ }
```

```
1          2
```

...we see that the fragmentation duplicate rate is higher in the AABB region than the AB region.

```
> FDvecCancer
```

```
[1] 0.019019019 0.005405405
```

Investigating the impact this has on our estimates of PCR duplication rate:

```
> # What were our PCR duplication rate estimates from Picard?
> BasicAABB<-(TumourPicard[3,6]-TumourPicard[3,7])/(TumourPicard[3,3]-TumourPicard[3,7])
> BasicAB<-(TumourPicard[2,6]-TumourPicard[2,7])/(TumourPicard[2,3]-TumourPicard[2,7])
> # What are our corrected estimates?
> CorAB<-(1-FDvecCancer[2])*BasicAB
> CorAABB<-(1-FDvecCancer[1])*BasicAABB
> # How much closer are they?
>
> BasicAABB-BasicAB
```

```
[1] 0.001013594
```

```
> CorAABB-CorAB
```

```
[1] 0.0004953876
```

```
> (BasicAABB-BasicAB)/(CorAABB-CorAB)
[1] 2.046062
```

The difference between the two estimates has halved (and that in the context of smaller estimates). Given the noise inherrent in the estimate from the smaller AB region, the corrected estimates from the AABB and AB regions are remarkably consistent.

7.5 Extending to regions that are not in allelic balance

We have demonstrated the approach only in genomic regions of allelic balance, but in theory this could be extended to any region where both alleles are present in the library (which due to contamination from normal cells, will still allow for regions that exhibit loss of heterozygosity in the tumour).

The added complication comes in the form of $\Pr(AP_k \mid \text{PART}_i)$ (see section 3.2). Consider the case where we have two duplicate reads from the partition into two different molecules. Without loss of generailty let the alleles at the SNP of interest be C and T. Currently we only consider two possible allele patterns `AA` representing the posibilities CC and TT and `AB` representing CT and TC. Since the probability of each allele is $\frac{1}{2}$ we say that the probability of the `AA` pattern is also $\frac{1}{2} = 0.5^2 + 0.5^2$.

Now if the allelic proportions are in fact 0.25 and 0.75, then we have two options:

- We still consider only two allele patterns, but the probability of the AA pattern is now $0.25^2 + 0.75^2 = 0.625$
- We can confidently associate probabilities with specific alleles (e.g through phasing we may be confident that it is the C allele present in 75% of molecules) in which case we would now consider three allele patterns `CC`, `CT`, `TT` with probabilities 0.5625, 0.375 and 0.0625 respectively.

The second option is more powerful than the first, but would require a not insubstantial effort to ensure that the probabilities are matched to SNPs correctly. In both cases, since fragmentation duplicates are more inclined to share the same allele than when we have allelic balance, the analysis will be less powerful than when we have allelic balance. In studies of human tumours we have thus far always been able to identify regions of the genome in allelic balance, but we acknowledge that there is the potential need to go down this route of studying regions of allelic imbalance, and that indeed it may be a prerequisite for application of these methods to other organisms.

Note that for any sizeable segment of genome in the same state, we can empirically estimate the allelic fraction with enough precision that our methods will be useful, and so this does not cause problems.

8 Complexity estimates

If the library complexity is X, the number of read-pairs sequenced is N, and the duplicate rate is denoted by R_D, then our estimate of X is obtained by solving

$$R_D = \frac{X}{N} \exp\left(-\frac{N}{X}\right) + 1 - \frac{X}{N} \tag{20}$$

using numerical methods. To obtain a starting point for the search, we use the closed form estimate based on a Maclaurin expansion $X = N/(2R_D)$.

```
> CEME<-rep(22,0)
> for(i in 1:22){
+   R<-BasicEst[i]
+   CEME[i]<-(readPairsExamined[i,1]--OpticalPairDuplicates[i,1])/(2*R)
+ }
```

Now we obtain the complexity estimate using the basic estimate of the duplicate rate,

```
> CEbasic<-rep(22,0)
> for(i in 1:22){
+   R<-BasicEst[i]
+   N<-readPairsExamined[i,1]-OpticalPairDuplicates[i,1]
+   startX<-CEME[i]
+   CEbasic[i]<-optim(startX,libCompNewParam,R=R,N=N)$par
+ }
```

and the complexity estimate using the residual estimate of the duplicate rate,

```
> CEresid<-rep(22,0)
> for(i in 1:22){
+   R<-ResidEst[i]
+   N<-readPairsExamined[i,1]-OpticalPairDuplicates[i,1]
+   startX<-CEME[i]
+   CEresid[i]<-optim(startX,libCompNewParam,R=R,N=N)$par
+ }
```

and the complexity estimate using the fragmentation-duplicate-corrected estimate of the duplicate rate. The increase in estimated complexity using the better estimate of duplicate rate is explored here. Figures will be generated after a mitochondrial interlude.

```
> CEFDCor<-rep(22,0)
> for(i in 1:22){
+   R<-FDCorEst[i]
+   N<-readPairsExamined[i,1]-OpticalPairDuplicates[i,1]
+   startX<-CEME[i]
+   CEFDCor[i]<-optim(startX,libCompNewParam,R=R,N=N)$par
+ }
```

In all cases the basic estimate is lower than our best estimate of the library complexity (with factors ranging from 1.14 to 2.15)

```
> summary(CEFDCor/CEbasic)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.144   1.308   1.373   1.465   1.521   2.149
```

9 Mitochondria

We revisit the Picard output from the mitochondrial mask discussed in the “Sample Information” section. In doing so we will create an object entitled PicardMito as described below.

```
> lf<-read.delim(file.path(EDpath,"Picard",metfiles[3]),as.is=T)
> lf<-lf[-(15:19),]
```

The Picard duplicate rate (column 4 of our PicardMito object) incorporates ‘unpaired’ reads into its calculation. These are even more affected by fragmentation duplicates than are read-pairs (see column 6 of the PicardMito object for the ‘unpaired’ read duplicate rate), and so we also record the read-pairs only duplicate rate estimate (column 5 of PicardMito). We also recycle the residual depth of coverage estimates from the “Exploration of SNP numbers” section, but adjust it for the estimate of PCR duplicates we have subsequently calculated.

```
> PicardMito<-cbind(sapply(lf[,1],substr,1,9),lf[,c(3,6,8)],
+                   as.numeric(lf[,6])/as.numeric(lf[,3]),
+                   as.numeric(lf[,5])/as.numeric(lf[,2]),ResDepth*(1-FDCorEst))
```

We have already estimated the PCR duplicate rate for these samples, so we calculate a duplicate-adjusted number of read-pairs for the mitochondrial genome.

```
> PicardMito<-cbind(PicardMito,as.numeric(PicardMito[,2])*(1-FDCorEst))
```

Since a read-pair provides 200 bases of sequence, the residual depth of coverage is for a diploid state so we have an additional factor of 2. Assuming the mitochondrial genome to be 16569 bases in length, we estimate the mitochondrial copy number without removing duplicates.

```
> PicardMito<-cbind(PicardMito,as.numeric(PicardMito[,2])*400/(as.numeric(PicardMito[,7])*16569))
```

We then estimate the mitochondrial copy number twice more, once removing all duplicates, and once adjusting for our estimated PCR duplicate rate.

```
> PicardMito<-cbind(PicardMito,
+                   (as.numeric(PicardMito[,2])-as.numeric(PicardMito[,3]))
+                   *400/(as.numeric(PicardMito[,7])*16569))
> PicardMito<-cbind(PicardMito,as.numeric(PicardMito[,8])*400/(as.numeric(PicardMito[,7])*16569))
> colnames(PicardMito)<-c("Library","ReadPairs","Duplicates","PicardRate",
+                         "BetterRate","UnpairedRate","ResCoverage",
+                         "CorrectedDuplicateReadPairs","MTCNignoredups",
+                         "MTCNremdups","MTCNcordups")
```

Now we produce Figure 3 from the paper (Figure 9 in this document), combining complexity and mitochondrial copy number results.

```
> par(mfrow=c(1,2))
> plot(c(1,3),c(3,30),type="n",ylab="Library complexity estimate (billions)",xlab="",axes=F)
> axis(2)
> box()
> for(i in seq(0,40,2)){
+   rect(-1,i-1,5,i,border="grey95",col="grey95")
+ }
> for(i in 1:22){
+   tcol<-"black"
+   if(WeaverSuppTable1[i,12]=="Blood"){tcol<-"red"}
+   lines(1:3,c(CEbasic[i],CEresid[i],CEFDCor[i])/(10^9),type="b",pch=16,col=tcol)
+ }
> axis(1,at=1:3,labels=c("Basic\nduplicate\nestimate",
+                        "Residual\nduplicate\nestimate",
+                        "Corrected\nestimate"),las=3)
> plot(c(1,3),c(60,800),type="n",ylab="MtDNA copy number",xlab="",axes=F)
> axis(2)
> box()
> for(i in seq(0,800,200)){
+   rect(-1,i-100,5,i,border="grey95",col="grey95")
+ }
> for(i in 1:22){
+   tcol<-"black"
+   if(WeaverSuppTable1[i,12]=="Blood"){tcol<-"red"}
+   lines(1:3,(as.numeric(PicardMito[i,c(9,11,10)])),type="b",pch=16,col=tcol)
+ }
> axis(1,at=1:3,labels=c("No\nduplicates\nremoved","New\nestimate","All\nduplicates\nremoved"),las=3)
```

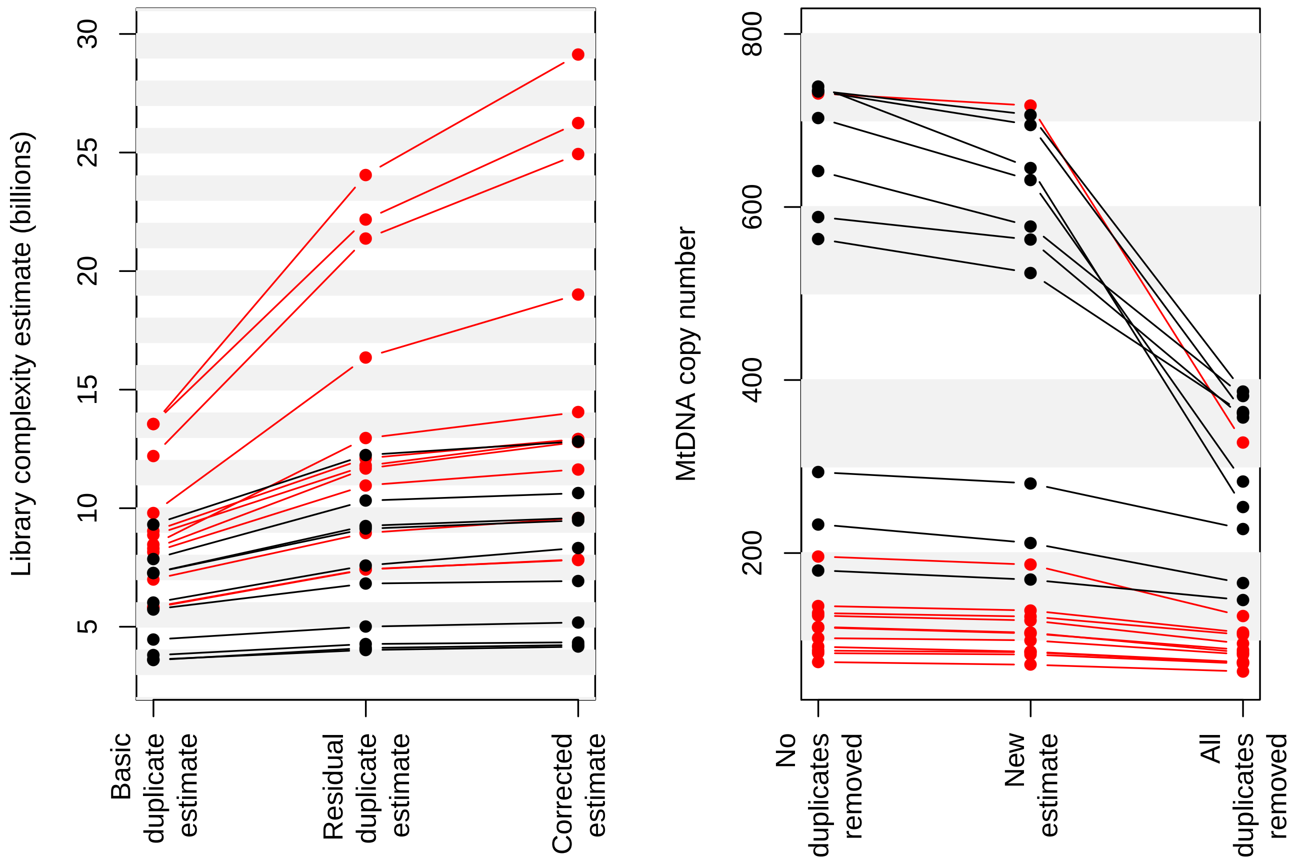


Figure 9: **Effects on complexity and mtDNA copy number estimates**

10 Session Info

For updates to this package/document, please visit www.compbio.group.cam.ac.uk.

```
> toLatex(sessionInfo())
```

- R version 3.2.1 (2015-06-18), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_GB.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_GB.UTF-8`, `LC_MESSAGES=en_GB.UTF-8`, `LC_PAPER=en_GB.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_GB.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BSgenome~1.36.3, BSgenome.Hsapiens.UCSC.hg19~1.4.0, BiocGenerics~0.14.0, Biostrings~2.36.4, GenomeInfoDb~1.4.2, GenomicRanges~1.20.5, IRanges~2.2.7, LynchSmithEldridgeTavareFragDup~1.5, Rsamtools~1.20.4, S4Vectors~0.6.3, XVector~0.8.0, rtracklayer~1.28.9
- Loaded via a namespace (and not attached): BiocParallel~1.2.20, BiocStyle~1.6.0, GenomicAlignments~1.4.1, RCurl~1.95-4.7, XML~3.98-1.3, bitops~1.0-6, futile.logger~1.4.1, futile.options~1.0.0, lambda.r~1.1.7, tools~3.2.1, zlibbioc~1.14.0