SNV detection from tumor and normal tissues with Fisher's Exact Test and False Discovery Rate correction

Modestas Filipavicius

May 24, 2018

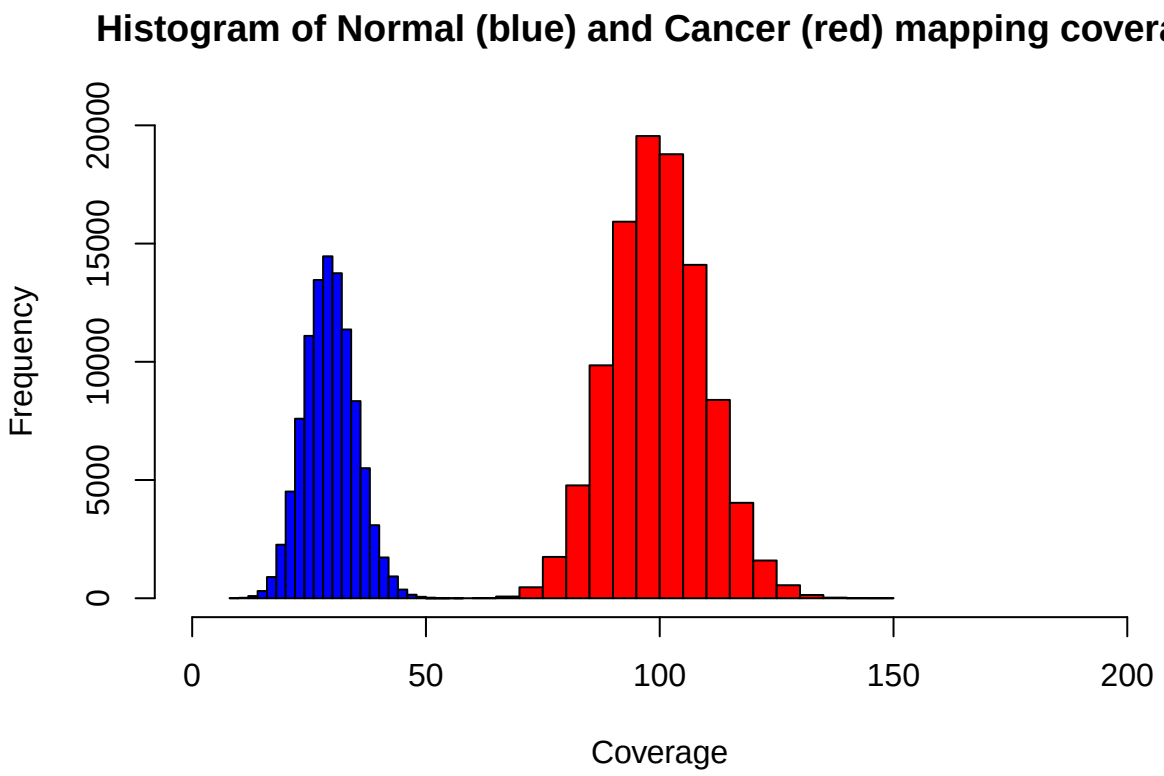
Given a simulated pileup of tumor sample and normal tissue sample, the task is to identify SNVs using the simple approach (Varscan 2) discussed in the lecture. Since the data is simulated, the number of true SNVs is known a priori and is found to be 52.

Read Normal pileup.txt and Tumor pileup.txt. Plot the histograms of coverage for each

sample.

```
library(parallel)
nor = read.table("./NGSPileup/Normal_pileup.txt", header = T)
tum = read.table("./NGSPileup/Tumor_pileup.txt", header = T)
msa_cancer = tum[, 5]
msa_control = nor[, 5]
coverage_cancer = tum[, 4]
coverage_coverage = nor[, 4]


# plot histograms
hist(nor$Coverage, col='blue', xlim=c(0, 200), ylim=c(0, 20000),
     xlab='Coverage', main='Histogram of Normal (blue) and Cancer (red) mapping coverage')
hist(tum$Coverage, col='red', add=T)
```



For each position construct the 2×2 contingency table and compute the p-Value.

First I will implement two functions: for calculating the contingency table and for calculating the Fisher's p-value.

```
# IN:   MSAs of cancer and control samples at some genomic position.
#       Character vector with mismatches
# OUT:  2D contingency table in matrix form, used for Fisher's exact test
calculate_conting_table = function(msa_cancer, msa_control, coverage_cancer,
                                   coverage_control, mismatch_nts) {

    # calculates number of variants per control and cancer msa
    variant_count_cancer = length(grep(mismatch_nts, msa_cancer, value = F,
                                       ignore.case = T, fixed = F))
    variant_count_control = length(grep(mismatch_nts, msa_control, value = F,
                                        ignore.case = T, fixed = F))

    # calculates number of other alleles
    other_count_cancer = coverage_cancer - variant_count_cancer
    other_count_control = coverage_control - variant_count_control

    contingency_table = matrix(c(variant_count_cancer, variant_count_control,
                                 other_count_cancer, other_count_control),
                               nrow = 2)

    return(contingency_table)
}

# IN:   control_df, cancer_df, id
```

```
# OUT:  Fisher's one-tailed p-value for some MSA sequence
calculate_fisher_test = function(control_df, cancer_df, id) {

    msa_cancer = unlist(strsplit(as.character(msa_cancer), ""))
    msa_control = unlist(strsplit(as.character(msa_control), ""))
    coverage_cancer = cancer_df[id,]$Coverage
    coverage_control = control_df[id,]$Coverage

    contig_table = calculate_conting_table(msa_cancer, msa_control,
                                           coverage_cancer, coverage_control,
                                           mismatch_nts)

    # right-tailed FET is used here, since we assume
    # more mismatched bases (mutations) for cancer MSAs, than controls
    f_test = fisher.test(contig_table, alternative = "greater",
                         conf.int = F) # turn-off conf.int to accelerate

    return(f_test$p.value)
}
```

Calculate p-values, using 3 methods of varying CPU efficiency: * for-loop, * mapply, and * mcmapply from 'parallel' library

```
mismatch_nts = paste(c("A", "C", "G", "T"), collapse="|")

# # FIRST CALCULATE W/OUT MAPPLY
# # calculate p-values for fisher's exact test
# p_val_vec = vector(length = nrow(nor))
# start_time = Sys.time()
# for (i in 1:nrow(nor)) {
#     p_val = calculate_fisher_test(control_df=nor, cancer_df=tum, id = i)
#     p_val_vec[i] = p_val
# }
# end_time = Sys.time()
# end_time - start_time
# # Time difference of 1.045392 mins
#
#
#
# # SECOND, CALCULATE USING MAPPLY, the fast way
# p_val_vec = vector(length = nrow(nor))
# start_time = Sys.time()
# p_val_vec = mapply(function(control5, cancer5, control4, cancer4) {
#
#     msa_cancer = unlist(strsplit(as.character(cancer5), ""))
#     msa_control = unlist(strsplit(as.character(control5), ""))
#
#     contig_table = calculate_conting_table(msa_cancer, msa_control, cancer4,
#                                            control4, mismatch_nts)
#
#     # right-tailed FET is used here, since we assume
#     # more mismatched bases (mutations) for cancer MSAs, than controls
#     f_test = fisher.test(contig_table, alternative = "greater",
#                          conf.int = F) # turn-off conf.int to accelerate
```

```
#
#     p_val_vec = f_test$p.value
#
# }, msa_control, msa_cancer, coverage_coverage, coverage_cancer)
#
# end_time = Sys.time()
# end_time - start_time
# # Time difference of 24.66781 secs



# THIRD, CALCULATE USING MCMAPPLY, the fastest way
p_val_vec = vector(length = nrow(nor))
start_time = Sys.time()
p_val_vec = mcmapply(function(control5, cancer5, control4, cancer4) {

    msa_cancer = unlist(strsplit(as.character(cancer5), ""))
    msa_control = unlist(strsplit(as.character(control5), ""))

    contig_table = calculate_conting_table(msa_cancer, msa_control,
                                           cancer4, control4, mismatch_nts)

    # right-tailed FET is used here, since we assume
    # more mismatched bases (mutations) for cancer MSAs, than controls
    f_test = fisher.test(contig_table, alternative = "greater", conf.int = F)
    # turn-off conf.int to accelerate

    p_val_vec = f_test$p.value

}, msa_control, msa_cancer, coverage_coverage, coverage_cancer)

end_time = Sys.time()
end_time - start_time
```

```
## Time difference of 14.61257 secs
```

Using a significance level of 10%, report the significant SNVs.

Without the FDR correction for multiple testing problem, we get almost 5,000 SNVs, the list of which is too long to report.

```
head(which(p_val_vec <= 0.1))
```

```
## [1]  16  17  82 130 624 750
```

```
tail(which(p_val_vec <= 0.1))
```

```
## [1] 99573 99576 99593 99673 99720 99785
```

```
length(which(p_val_vec <= 0.1))
```

```
## [1] 1525
```

```
# 4867 with two-tailed test
# 3500 with right-tailed test
```

We saw that more than 4,500 SNV were called, since we ignore the multiple testing problem, namely, testing 100,000 hypotheses simultaneusly

Perform multiple testing corrections using "fdr" method and report the significant SNVs after

correction. Use a cut-off of 0.1.

```
p_val_vec_fdr = p.adjust(p_val_vec, method = 'fdr')
snvs = which(p_val_vec_fdr <= 0.10)
length(snvs)
```

```
## [1] 16
```

```
# 15 with right-sided
# 16 with two-tailed

# These are the IDs of significant SNVs
snvs
```

```
##  [1] 22294 24221 31238 32024 34538 38974 47734 49917 60014 61554 62736
## [12] 66902 74039 76842 86126 90940
```

```
# And corresponding positions at genome assembly
nor$Position[snvs]
```

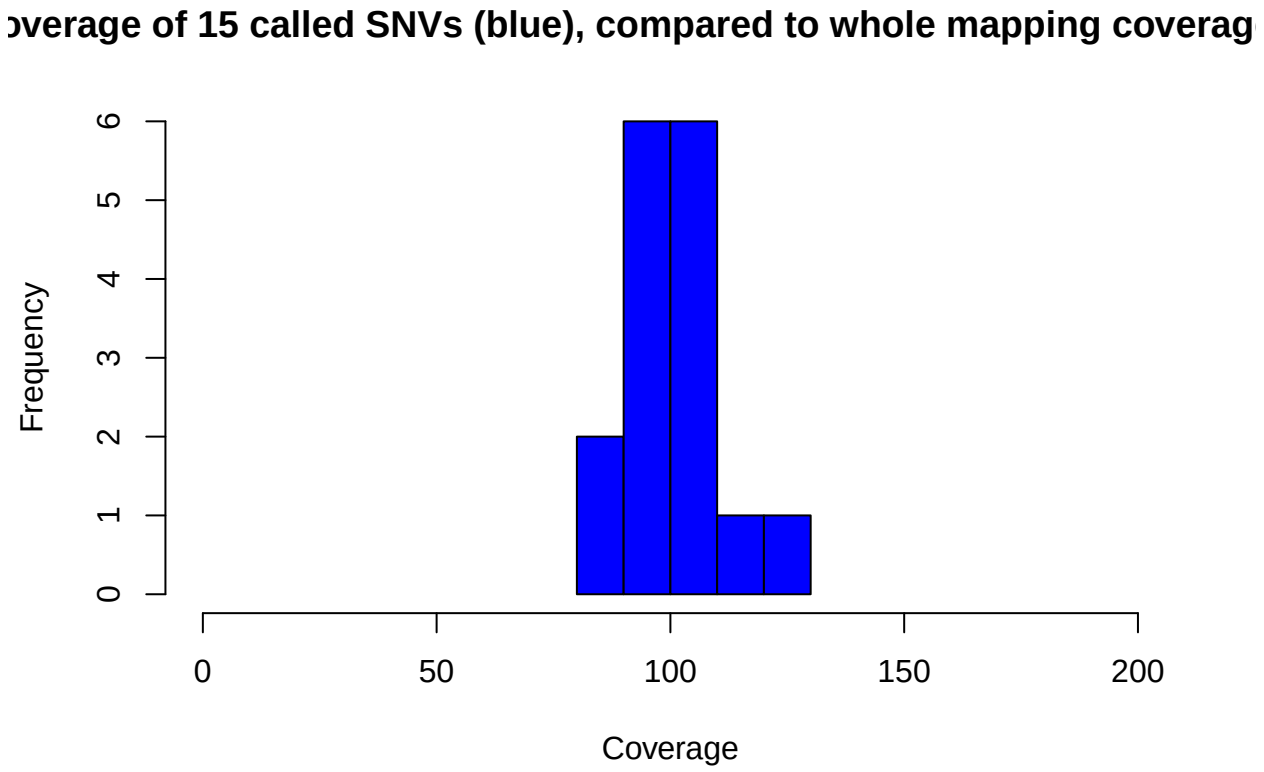
```
##  [1] 130333 589244 366758 562963 971838 856157 230973 987508 781799 496014
## [11] 519856 526679 613113  28273 420894 425943
```

Is there a difference between the number of significant SNVs after correction and the total number of known SNVs (52)?

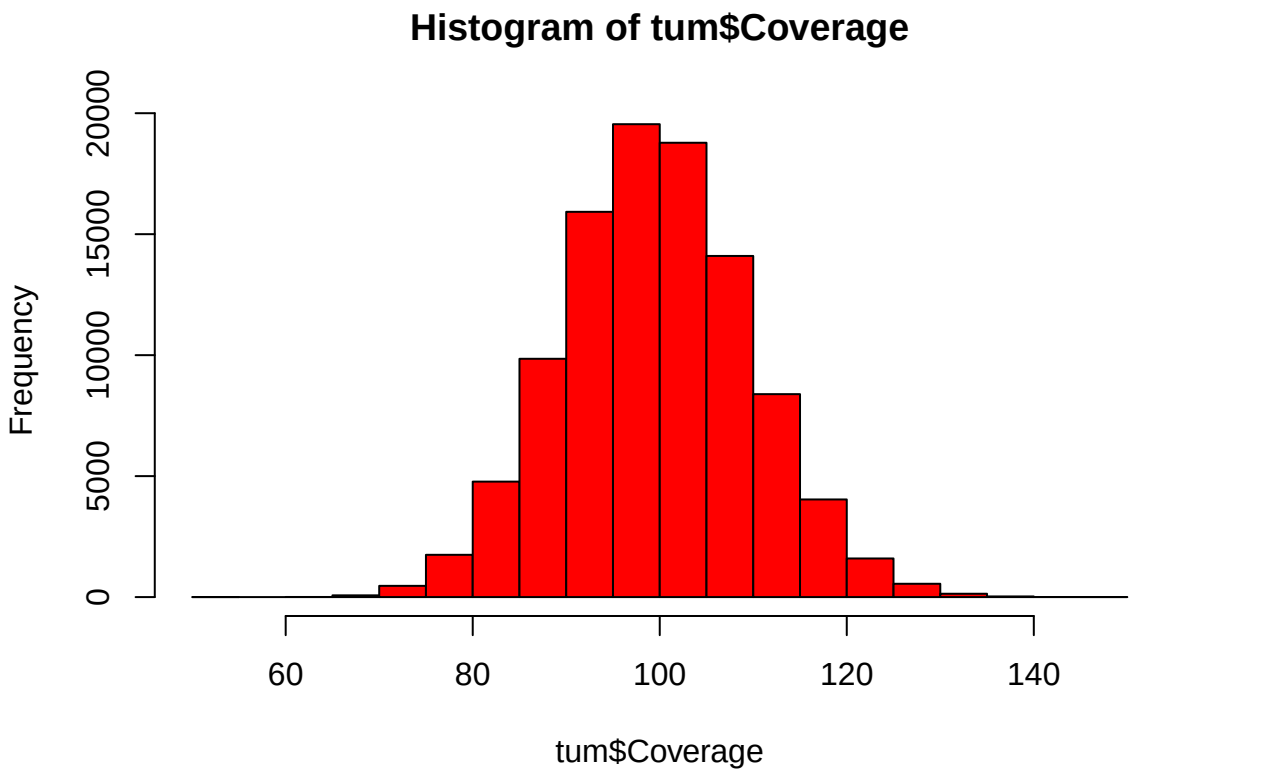
SNV calling analysis above, which has the same working principle as VarScan, yielded only 15 SNVs, thus failing to call the remaining 71% of true SNVs. Why is that?

- my first (incorrect) explanation was because the called SNVs had higher coverage, thus we missed the SNVs that are not that well covered. However, in the plots below we can see that called SNVs have similar coverage distribution to normally distributed overall data coverage:

```
hist(tum$Coverage[snvs], col='blue', xlim=c(0, 200),
     xlab='Coverage', main='Coverage of 15 called SNVs (blue), compared to whole mapping coverage (red)
```




```
hist(tum$Coverage, col='red', add=F)
```



- We didn't account for non-independent and non-identical error rates at positions, which are phased. Algorithm like "V-Phaser" should take care of that.
- Similarly, we assumed that genotypes of nearby loci are independent, which violates Linkage Disequilib-

rium theory. Thus we shouldn't calculate Fisher tests or sequencing error probabilities for a particular locus without taking into account nearby loci. Ideally, we should have used chunks of overlapping haplotypes instead of single nucleotides for SNV calling.

- We didn't take into account strand specificity. Sequencing errors at particular alignment sites usually occur on one strand. Ideally we should model the number of observed nucleotide counts on both strands separately and then attach a probability of 'being caused by a sequencing error' for that nucleotide. This is what deepSNV does, at high level.
- We specified too low p-value threshold for the fisher exact test. If the threshold was increased from 0.10 to 0.90, then 46 SNVs would be called. However, doing so makes little sense.