Sample Contamination

Lai Ping Wong

2018-02-12

Contents

1 Introduction

contPredict package identifies sample contamination base on variant allele frequency (VAF) discovered from next generation sequencing (NGS) on cancer samples. It requires 1 configuration file (tab delimited config.txt) and 2 input data files that are variant allele frequency (VAF.out) and variant coverage data (VAF_cov.out). Example input files are attached in the installation package (contPredict/inst/extdata).

2 Background

2.1 Assumption

SNPs discovered in NGS data belong to at least one of the following three classes:

a. Germline SNPs
   These SNPs are present in multiple samples of the same patient, both normal and tumor. They usually occur at about the same VAF (especially around 50% or 100%, for heterozygous and homozygous mutations, respectively), although discrepancy may exist due to tumor-specific alterations. Many of them are found in public databases (eg., NHLBI and 1000 Genomes).

b. Somatic mutations
   These SNPs are present in at least one tumor sample of the same patient, but in normal. They can occur at any VAF but mostly less than 50%. Some of them are shared by different patients if they are driver mutations of the same disease.

c. Cross-individual contamination
   These SNPs are present in at least two samples from different patients, with distinguishable VAF bias: high in source and low in target. In the source sample, they can be both germline (Class I) and somatic (Class II) mutations.

2.2 Method

Pairwise sample global commonality

Let x and y be the sets of SNPs present in two arbitrary samples, respectively.

Pairwise sample global commonality (R code)

$$globalPcommon_x = \frac{Ncomm}{Nx}$$

$$globalPcommon_y = \frac{Ncomm}{Ny}$$

Notation	Description
Nx	number of SNPs in sample X
Ny	number of SNPs in sample Y
Ncomm	number of common SNPs in a pair of sample

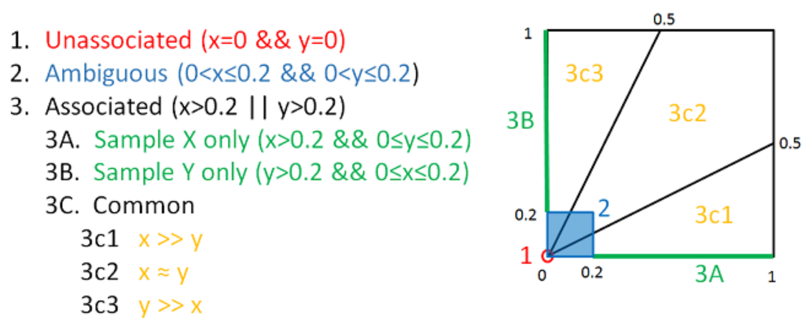
1. Unassociated (x=0 && y=0)
2. Ambiguous (0<x≤0.2 && 0<y≤0.2)
3. Associated (x>0.2 || y>0.2)
3A. Sample X only (x>0.2 && 0≤y≤0.2)
3B. Sample Y only (y>0.2 && 0≤x≤0.2)
3C. Common
3c1 x >> y
3c2 x ≈ y
3c3 y >> x
1
0.5
3c3
3c2
3B
0.5
3c1
0.2
2
1
0
0.2
3A
1

Figure 1: 7 distinct regions color coded (red, blue, green and orange) for VAF-VAF plot of 2 samples

Count number of SNPs in 7 regions

For a pair of sample X and Y, count number of SNPs in 7 distinct regions from a VAF-VAF scatter plot. SNPs in region 1 and 2 are not used (R code).

By using the following notation, pairwise sample commonality and regional commonality are calculated

Notation	Description
Na	number of SNPs in 3A
Nb	number of SNPs in 3B
Nc1	number of SNPs in 3c1
Nc2	number of SNPs in 3c2
Nc3	number of SNPs in 3c3
Nc	Nc1+Nc2+Nc3

Pairwise sample commonality

$$pcommon = \frac{Nc}{Na + Nb + Nc}$$

Regional commonality

$$Pc1 = \frac{Nc1}{Nc}$$

$$Pc2 = \frac{Nc2}{Nc}$$

$$Pc3 = \frac{Nc3}{Nc}$$

SNPs distribution within region 3c1 and 3c3

$$Qc1 = \frac{Nc1}{Nc1 + Na}$$

$$Qc3 = \frac{Nc3}{Nc3 + Nb}$$

$$Sc1 = \frac{Nc1}{Nc1 + Nc2}$$

$$Sc3 = \frac{Nc3}{Nc3 + Nc2}$$

Identification of pairwise sample relationship (R code)

Paramenter	Default	Description
VAF_cutoff1	0.050	misc. VAF threshold
VAF_cutoff	0.002	minimum VAF to be considered present in a sample
VAF_ignore	0.200	ignore ambiguous variant fall below this cutoff
pcomm_cutoff	0.100	low or high contamination
p.val_cutoff	0.050	Fish exact test p-val cutoff for multisource contamination
num_round_digit	3.000	number of decimal to be rounded off
center_cutoff	0.500	same individual threshold
source_cutoff	0.400	minimum source threshold
target_cutoff	0.100	minimum target threshold
region_cutoff	0.750	minimum region threshold for globalPcomm < pcomm_cutoff
n_sample	7.000	number of sample

By using default parameter setting, for pairwise sample with global commonality greater than pcomm_cutoff, potential contamination events:

a. Same patient contamination: Pc2 >center_cutoff
b. One-way contamination, sample X contaminates sample Y:
   Pc1 >source_cutoff && Pc2 <=center_cutoff && Pc3 <target_cutoff
c. One-way contamination, sample Y contaminates sample X:
   Pc3 >source_cutoff && Pc2 <=center_cutoff && Pc1 <target_cutoff
d. Both way contamination (X<->Y): other than all above conditions (a-c)

For paired sample with global commonality less than pcomm_cutoff, potential contamination events:

a. One-way contamination, sample X contaminates sample Y:
   (Qc1 >target_cutoff && Sc1 >region_cutoff) && !(Qc3 >target_cutoff && Sc3 >region_cutoff)
b. One-way contamination, sample Y contaminates sample X:
   !(Qc1 >target_cutoff && Sc1 >region_cutoff) && (Qc3 >target_cutoff && Sc3 >region_cutoff)
c. Both way contamination (X<->Y):
   (Qc1 >target_cutoff && Sc1 >region_cutoff) && (Sc3 >target_cutoff && Sc3 >region_cutoff)

Calculation of contamination level

For pairwise sample with relationship identified as either one-way or both way contamination, contamination level is calculated base on coverage weighted regression. SNPs used in the linear regression is confined to those in the source region (e.g. 3A+3c1 when X is the contamination source and 3B+3c3 when Y is the contamination source)

Contamination level is the intercept of linear regression (slope=0) to the ratio

$$mixR = \frac{VAF_x}{VAF_y}$$

weighted by smaller coverage between the two samples (X and Y)

3 Installation

To install this package:

```r
install_github("contPredict", quick=TRUE)
library(contPredict)
```

4 Quick Start

The purpose of this section is to provide users a general workflow of the sample contamination using test data attached within the package.

4.1 Step 1 - Setup parameters and get data

4.1.1 Parameters setting base on information from configuration file

```
currentDir <- paste(getwd(),"/inst/data",sep='')
config.file <- paste(currentDir, "/config.txt",sep="")
setParameterConfig(config.file)
vaf_file <- paste0(currentDir,"/VAF.out")
cov_file <- paste0(currentDir,"/VAF_cov.out")
```

4.1.2 Load variant allele freqeuncy (VAF) data

```
VAFdata <- inputData(vaf_file)
head(VAFdata)
```

mutationID	CAP001-Nb-K	CAP001-Td1c-K	CAP002-Nb	CAP002-Td1b-K
1.103468336.A.C	0	0	0.2667	0
1.10510320.G.A	0	0	0.4643	0.3333
1.109391557.G.A	0	0	0.5405	0.4699
1.109428144.G.C	0	0	0	0
1.109477466.G.T	0	0	0	0
1.110033736.G.A	0.4651	0.5703	0	0.05556

CAP003-Nb-K	CAP003-Td1a	CAP003-Td1b	publicDB	freq
0	0	0.08	unknown	2/7
0.1236	0	0	unknown	3/7
0.07407	0	0	unknown	3/7
0	0.15	0.09574	unknown	2/7
0	0	0.1522	unknown	1/7
0	0	0	unknown	3/7

4.1.3 Load variant coverage (COV) data

```
VAFcov <- inputData(cov_file)
head(VAFcov)
```

mutationID	CAP001-Nb-K	CAP001-Td1c-K	CAP002-Nb	CAP002-Td1b-K
1.103468336.A.C	38	30	30	26
1.10510320.G.A	52	60	56	72
1.109391557.G.A	112	93	74	83
1.109428144.G.C	102	89	63	73

mutationID	CAP001-Nb-K	CAP001-Td1c-K	CAP002-Nb	CAP002-Td1b-K
1.109477466.G.T	83	88	60	117
1.110033736.G.A	129	128	104	144

CAP003-Nb-K	CAP003-Td1a	CAP003-Td1b
46	27	50
89	44	38
108	75	86
107	60	94
106	66	46
157	90	50

4.2 Step 2 - Calculate number of SNPs for each sample

```
SNPcount <- numMutationperSample(VAFdata,VAF_cutoff,n_sample)
SNPcount
```

CAP001-Nb-K	206
CAP001-Td1c-K	244
CAP002-Nb	205
CAP002-Td1b-K	292
CAP003-Nb-K	235
CAP003-Td1a	101
CAP003-Td1b	135

4.3 Step 3 - Count number of common SNPs for pairwise samples

```
SNPshare<- pairShare(VAFdata,VAF_cutoff,n_sample)
SNPshare
```

	CAP001-Nb-K	CAP001-Td1c-K	CAP002-Nb	CAP002-Td1b-K
CAP001-Nb-K	206	140	18	118
CAP001-Td1c-K	140	244	76	201
CAP002-Nb	18	76	205	149
CAP002-Td1b-K	118	201	149	292
CAP003-Nb-K	32	75	166	147
CAP003-Td1a	43	1	1	1
CAP003-Td1b	58	1	33	4

	CAP003-Nb-K	CAP003-Td1a	CAP003-Td1b
CAP001-Nb-K	32	43	58
CAP001-Td1c-K	75	1	1
CAP002-Nb	166	1	33
CAP002-Td1b-K	147	1	4
CAP003-Nb-K	235	54	54
CAP003-Td1a	54	101	83

	CAP003-Nb-K	CAP003-Td1a	CAP003-Td1b
CAP003-Td1b	54	83	135

4.4 Step 4 - Calculate pairwise samples pcommon

```
# calculate pairwise sample global pcommon
pcomm <- pairPCommon(VAFdata,VAF_cutoff,num_round_digit,n_sample)

# generate all pairwise sample list
pairs <- pairList(VAFdata,n_sample)

# attach pcommon to each pairwise sample
Pcomm <- do.call(rbind, pairCommList(pcomm,n_sample))

# create sample pairs data frame
sample_pairs <- data.frame(pairID = pairs, Pcomm_1st = Pcomm[,1], Pcomm_2nd = Pcomm[,2])
pids <- t(sapply(as.matrix(sample_pairs$pairID), function(i) unlist(strsplit(i, "_"))))
colnames(pids) <- c("pid1", "pid2")
sample_pairs <- cbind(sample_pairs,pids)
```

4.5 Step 5 - Count number of SNPs in 7 regions

```
countPoint <- regionCountMutation(sample_pairs, VAFdata, SNPcount,
    SNPshare, VAF_cutoff, VAF_ignore, n_sample)
sample_pairs <- cbind(sample_pairs, countPoint)
```

4.6 Step 6 - Identify pairwise sample relationship

00: same patient
01/10: one way contamination
11: bothway contamination

```
rel <- pairRelation(sample_pairs, center_cutoff, source_cutoff,
    target_cutoff, localPcomm_cutoff, region_cutoff, num_round_digit,
    output_path)
sample_pairs <- cbind(sample_pairs, rel)
```

4.7 Step 7 - Check multiple contamination sources

```
final_rel <- multipleSource(sample_pairs = sample_pairs, VAFdata = VAFdata,
    VAFcov = VAFcov, VAF_cutoff, VAF_cutoff1, p.val_cutoff, output_path = output_path)
```

4.8 Step 8 - Calculate mixing ratio

```
# calculate mixing ratio
mr <- as.data.frame(mixingRatio(VAFdata = VAFdata, VAFcov = VAFcov,
    sample_pairs = sample_pairs, final_rel = final_rel, VAF_cutoff = VAF_cutoff,
    VAF_ignore = VAF_ignore, output_path = output_path))

# filter multisource contamination
final.mr <- as.data.frame(cbind(mr[, "source"], mr[, "target"],
    as.numeric(mr[, "lm_coeff"]) * 100, mr[, "rel"], mr[, "flip"]))
colnames(final.mr) <- c("source", "target", "predicted_contamination_perc",
    "rel", "flip")

printmr <- as.data.frame(cbind(source = final.mr$source, target = final.mr$target,
    predicted_contamination_perc = final.mr$predicted_contamination_perc))

result.file = paste(output_path, "/tmp/", center_cutoff, "center_",
    source_cutoff, "source_", target_cutoff, "target_", localPcomm_cutoff,
    "pcomm_", region_cutoff, "region_", n_sample, "sample_contaminationLevel.txt",
    sep = "")

write.table(unique(printmr), quote = F, sep = "\t", result.file,
    row.names = F)
```

4.9 Step 9 - Write output file and create circos plot

```
# final contamination pairs
final.pred <- as.data.frame(filterMultiSources(filename = result.file,
    output_path = output_path, VAFdata = VAFdata, VAFcov = VAFcov,
    uniq_both = 2))

# same individual pairs
usedCol <- c(4, 5)
same.ind <- sample_pairs[which(sample_pairs$rel ==
    "00"), usedCol]
colnames(same.ind) <- c("pid1", "pid2")
outF <- paste0(output_path, "/sameIndividual.txt")
write.table(same.ind, quote = F, sep = "\t", col.names = T,
    row.names = F, outF)

# sample ids
sampleID <- colnames(VAFcov)[-1]

# prepare circos plot data
same.subject <- sample_pairs[sample_pairs$rel == "00",
    ]

# attach relation
tmp = unique(merge(final.mr, final.pred, by = c("source",
    "target"))[1:4])
tmp = tmp[order(tmp$rel, decreasing = T), ]

contaminate <- tmp[!duplicated(tmp[, c("source", "target",
    "predicted_contamination_perc.x")]), ]
colnames(contaminate) <- c("source", "target", "link",
    "rel")
```

```r
contaminate$link <- as.numeric(contaminate$link)/10
contaminate[contaminate$rel == "01", "rel"] <- "10"

if (nrow(same.subject) > 0) {
    same.subject.mr <- as.data.frame(mixingRatio(VAFdata,
        VAFcov, sample_pairs, final_rel, VAF_cutoff,
        VAF_ignore, ALL_flag = TRUE, sameSubject = TRUE))

    same.subject.out <- as.data.frame(cbind(source = same.subject.mr$source,
        target = same.subject.mr$target, link = round(10 *
            as.numeric(same.subject.mr$lm_coeff), num_round_digit),
        rel = same.subject.mr$rel))
    circos.plot <- as.data.frame(rbind(same.subject.out,
        contaminate))
} else {
    circos.plot <- contaminate
}
final.circos <- circos.plot
file <- paste0(output_path, "/tmp/circos_plotdata.txt")
write.table(final.circos, file, quote = F, sep = "\t",
    row.names = F)

plot_circos = TRUE
if (plot_circos) {
    file <- paste0(output_path, "/tmp/circos_plotdata.txt")

    # contamination circos
    plot_circos_link(file, R = 200, W = 30, plotsize = 800,
        titleStr = "Contamination", seg.lab.size = 1,
        fig.file = "contamination_circos.png", contaminatedOnly = TRUE,
        sameSubjectOnly = FALSE, allSamples = sampleID,
        output_path)

    # same subject circos
    plot_circos_link(file, R = 200, W = 30, plotsize = 800,
        titleStr = "Same individual", seg.lab.size = 1,
        fig.file = "sameIndividual_circos.png", contaminatedOnly = FALSE,
        sameSubjectOnly = TRUE, allSamples = sampleID,
        output_path)

    # same individual + contamination in 1 circos
    plot_circos_link(file, R = 200, W = 30, plotsize = 800,
        titleStr = "All", seg.lab.size = 1, fig.file = "allPairs_circos.png",
        contaminatedOnly = FALSE, sameSubjectOnly = FALSE,
        allSamples = sampleID, output_path)
}

# clean up
tmp = paste0(output_path, "/tmp")
unlink(tmp, TRUE)
```

5 Output

The program will output two tab delimited file. contPredict.txt contains contamination pairs and sameIndividual.txt displays same subject pairs. Also generate three circos plot in png format depict all pairs, same individual pairs and contamination pairs.

5.1 Contamination pairs (contPredict.txt)

source	target	predicted_contamination_perc
CAP003-Td1b	CAP001-Nb-K	9.387
CAP002-Td1b-K	CAP001-Td1c-K	4.640
CAP002-Nb	CAP003-Nb-K	21.285
CAP001-Td1c-K	CAP002-Td1b-K	10.001

5.2 Same subject pairs (sameIndividual.txt)

pid1	pid2
CAP001-Nb-K	CAP001-Td1c-K
CAP002-Nb	CAP002-Td1b-K
CAP003-Nb-K	CAP003-Td1a
CAP003-Nb-K	CAP003-Td1b
CAP003-Td1a	CAP003-Td1b

5.3 Circos plot

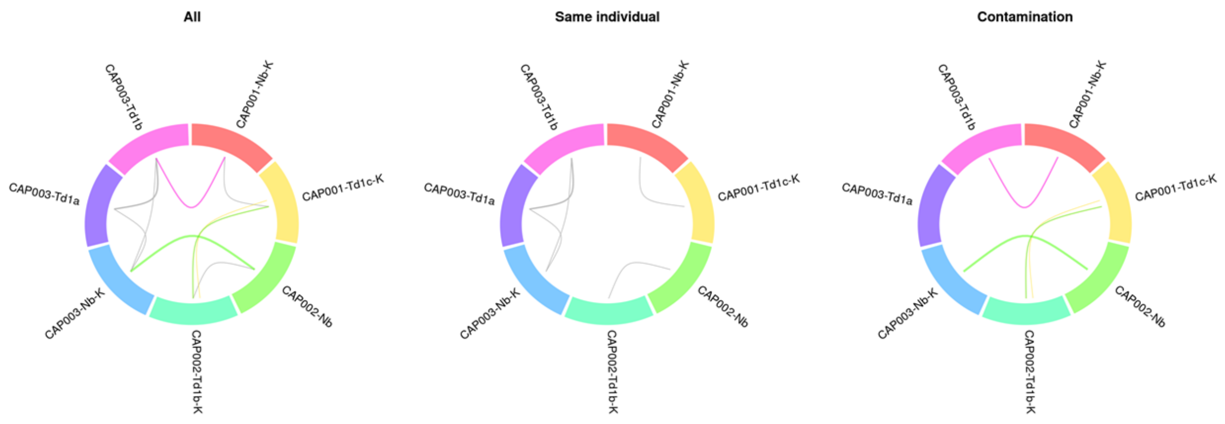

Figure 2: Circos plot