Supplemental material for Decomposition of mutational context signatures using QP methods

Andy Lynch
24 May 2016

Contents

Load the libraries ................................................................. 1
Define the QP function: ....................................................... 1
Create some data objects .................................................... 2
An example where the QP approach is substantially faster .............. 3
A flat combination of all 27 signatures .................................... 4
Combinations of two signatures ............................................. 6
Specific examples of problematic signature pairs .......................... 8
WGS data example ............................................................... 11
Details of the packages used. .................................................. 13

Load the libraries

We consider two methods for decomposing a table of mutational contexts into signatures, that of Rosenthal et al, and the quadratic programming approach. Here we load in the two libraries required for these approaches:

library(deconstructSigs)
library(quadprog)

Define the QP function:

The QPsig function is designed to take input in similar format to the whichSignatures function in the deconstructSigs library for easy interchange between the two. The output is limited to the equivalent of the ‘weights’ slot in the output of whichSignatures.

QPsig<-function(tumour.ref = NA,samplerow,signatures.ref = signatures.nature2013){

  # we normalize the observations so that they sum to 1
  obs<-as.numeric(tumour.ref[samplerow,]/sum(tumour.ref[samplerow,]))

  # to allow use of the deconstructSigs objects we convert to matrices
  signatures.ref<-as.matrix(signatures.ref)

  # we use the factorized version of solve.QP -

# we normalize the observations so that they sum to 1
obs<-as.numeric(tumour.ref[samplerow,]/sum(tumour.ref[samplerow,]))

# to allow use of the deconstructSigs objects we convert to matrices
signatures.ref<-as.matrix(signatures.ref)

# we use the factorized version of solve.QP -
# see the helpfile of that function for details of the required form
# otherwise we would set Dmat = signatures.ref %*% t(signatures.ref) as indicated
# in the article
Rinverse <- backsolve(chol(signatures.ref %*% t(signatures.ref)),
                      diag(dim(signatures.ref)[1]))

# we also need to define the linear part of the function that we are minimizing
dvec <- (obs) %*% t(signatures.ref)

# we have one constraint that the sum of weights is equal to 1
# we have more constraints that each weight is positive
Amat <- cbind(rep(1,dim(Rinverse)[1]), diag(dim(Rinverse)[1]))
bvec <- c(1,rep(0,dim(Rinverse)[1]))

# we now call the solve.QP function from the quadprog library
myQP<-quadprog::solve.QP(Dmat = Rinverse, dvec = dvec, Amat = Amat, bvec = bvec, meq = 1,
                          factorized = TRUE)
return(myQP$solution)
}

The Rinverse object is a function of the signature matrix, so the same for all samples and could be precompiled,
but this doesn’t save time in practice.

Create some data objects

The first row is an equal mix of signatures 1A, 1B, 2 and 13 that (in this test setup at least) takes over 2000
iterations and considerable time for whichSignatures to reach the solution.
The second row is a flat combination of all 27 signatures.
The next 351 rows give all combinations of two signatures.
The final row gives the data for an oesophageal adenocarcinoma example (SS6003314) from the Weaver et al
paper “Ordering of mutations in preinvasive disease stages of esophageal carcinogenesis” Nature Genetics
2014.

# Using the PMID of Rosenthal et al. as a seed.
set.seed(26899170)
signatures.ref<-.as.matrix(signatures.nature2013)
datatemplate<-randomly.generated.tumors[1,]
datatemplate[,1]<-(500*(matrix(0.25*signatures.ref[1,]+0.25*signatures.ref[2,]
                        +0.25*signatures.ref[3,]+0.25*signatures.ref[14,],ncol=1)))
datatemplate[2,]<-round(500*(runif(96,0.95,1.05)*apply(signatures.ref,2,sum)))
dataguide<-.rep("",351)
index<-3
for(i in 2:27){
  for(j in 1:(i-1)){
    datatemplate[index,]<-round(runif(96,0.95,1.05)*250*
                                (signatures.ref[i,]+signatures.ref[j,]))
  }
}


An example where the QP approach is substantially faster

We use equal mixtures of signatures 1A, 1B, 2 and 13. Both methods get the right answer for this combination of signatures, but the ILM approach requires thousands of iterations and is thousands of times slower than the QP approach.

```r
system.time(
QPres1<-QPsig(tumour.ref=datatemplate,1))
## user  system elapsed
## 0.020 0.000  0.017

system.time(
Linres1 <- whichSignatures(
tumor.ref = datatemplate,
signatures.ref = signatures.nature2013,
sample.id =1, contexts.needed = TRUE,
tri.counts.method = 'default'))
## user  system elapsed
## 230.814 0.000 231.286
```

To confirm that both methods correctly assign weights of 0.25 to the four contributing signatures we plot the two sets of contributions.

```r
plot(1:27-0.1,as.numeric(Linres1$weights),pch=16,axes=F,xlab="signature",ylab="contribution")
abline(h=.25,lwd=2)
points(1:27+0.1,QPres1,pch=16,col="red")
axis(2)
axis(1, at=1:27, labels=sapply(rownames(signatures.nature2013),
  function(x){unlist(strsplit(x,".",fixed=T))[2]}),las=2)
box()
legend("right",fill=c("black","red"),legend=c("ILM","QP"),inset=0.1)
```
A flat combination of all 27 signatures

The second example combines all signatures. This gives most scope for discovering a combination of signatures that cause problems, but doesn’t allow for the possibility of erroneously including a signature in the model.

```r
system.time(
QPres2<-QPsig(tumour.ref=datatemplate,2))
```

```r
## user system elapsed
## 0.016 0.000 0.015
```

```r
system.time(Linres2 <- whichSignatures(tumor.ref = datatemplate,
signatures.ref = signatures.nature2013,
sample.id =2, contexts.needed = TRUE,
signature.cutoff=0.01))
```

```r
## user system elapsed
## 39.090 0.000 39.167
```

The two sets of contributions are illustrated below. Both methods generally do well in estimating contributions of approximately 1/27, but the four biggest outliers are all from the ILM method, and Signature 3 is not included in the ILM solution.

```r
plot(1:27-0.1,as.numeric(Linres2$weights),pch=16,axes=F,xlab="signature",ylab="contribution",ylim=c(0,0.1))
points(1:27+0.1,QPres2,pch=16,col="red")
axis(2,at=seq(0,3,0.5)/27,labels=c("0","1/54","1/27","3/54","2/27","5/54","3/27"),las=2)
axis(1,at=1:27,labels=sapply(rownames(signatures.nature2013),
function(x){unlist(strsplit(x,".",fixed=T))[2]}),las=2)
box()
```
Since we know the truth, we can compare the estimated weights to the true weights:

```r
sum((Linres2$weights-1/27)^2)
```

## [1] 0.007079487

```r
sum((QPres2-1/27)^2)
```

## [1] 0.002204034

Also we can see how the likelihoods of the two two solutions compare given the ‘observations’:

```r
# log-likelihood for QP (to within additive constant)
sum(datatemplate[2,]*log(QPres2 %*% as.matrix(signatures.ref)))
```

## [1] -57561.15

```r
# log-likelihood for ILM (to within additive constant)
sum(datatemplate[2,]*log(as.numeric(Linres2$weights) %*% as.matrix(signatures.ref)))
```

## [1] -57647.05

```r
# Improved likelihood for QP solution (+ve is good for QP)
(sum(datatemplate[2,]*log(QPres2 %*% as.matrix(signatures.ref)))) -
(sum(datatemplate[2,]*log(as.numeric(Linres2$weights) %*% as.matrix(signatures.ref))))
```

## [1] 85.90378
Combinations of two signatures

We now create combinations of all pairs of signatures, apply both methods, and record the weights/contributions assigned to the two relevant signatures in each case.

```r
proplin <- matrix(NA, nrow=351, ncol=2)

pt1 <- proc.time()
index <- 1
for (i in 2:27){
  for (j in 1:(i-1)){
    tmpLinres <- whichSignatures(tumor.ref = datatemplate,
                                signatures.ref = signatures.nature2013,
                                sample.id = index+2, contexts.needed = TRUE)
    proplin[index,] <- as.numeric(tmpLinres$weights[c(i,j)])
    index <- index+1
  }
}
pt2 <- proc.time()
pt2 - pt1

propQP <- matrix(NA, nrow=351, ncol=2)

pt3 <- proc.time()
index <- 1
for (i in 2:27){
  for (j in 1:(i-1)){
    myQP <- QPsig(tumour.ref = datatemplate, index+2)
    propQP[index,] <- myQP[c(i,j)]
    index <- index+1
  }
}
pt4 <- proc.time()
pt4 - pt3

The QP method is substantially faster.

We see that in general, both methods do well in recovering the signal from the pairs of signatures, but for a handful of combinations the iterative linear model approach does noticably worse.

pairlabels <- rep("",351)
index <- 1
for (i in 2:27){
  for (j in 1:(i-1)){
    myQP <- QPsig(tumour.ref = datatemplate, index+2)
    pairlabels[index] <- myQP[c(i,j)]
    index <- index+1
  }
}
pairlabels[index] <- paste(sapply(rownames(signatures.nature2013),
   function(x){unlist(strsplit(x, ".", fixed=T))[2]}[c(i,j)],
   collapse="-")

index <- index + 1
}
}

#pdf("plot1b.pdf", height=6, width=6)
par(mar=c(4.1,4.1,1.1,1.1))
plot(proplin[,1],proplin[,2],pch=20,xlab="estimated contribution of first signature",
   ylab="estimated contribution of second signature")
abline(h=0.5,lty=2,col="grey40",lwd=2)
abline(v=0.5,lty=2,col="grey40",lwd=2)
points(propQP[,1],propQP[,2],pch=20,col="red")
for(i in 1:351){
   lines(c(proplin[i,1],propQP[i,1]),c(proplin[i,2],propQP[i,2]),lwd=1)}

text(proplin[13,1],proplin[13,2],pairlabels[13],adj=c(.5,1.5))
text(proplin[349,1],proplin[349,2],pairlabels[349],adj=c(.5,1.5))
text(proplin[84,1],proplin[84,2],pairlabels[84],adj=c(1.2,.5))
text(proplin[255,1],proplin[255,2],pairlabels[255],adj=c(.5,1.5))
text(proplin[51,1],proplin[51,2],pairlabels[51],adj=c(.5,1.5))
text(propQP[5,1],propQP[5,2],pairlabels[5],adj=c(.5,1.5))
text(propQP[259,1],propQP[259,2],pairlabels[259],adj=c(1.2,.5))
text(propQP[155,1],propQP[155,2],pairlabels[155],adj=c(0,.5))
text(propQP[3,1],propQP[3,2],pairlabels[3],adj=c(0,.5))
text(propQP[138,1],propQP[138,2],pairlabels[138],adj=c(1.2,.5))
legend("bottomleft",fill=c("black","red"),legend=c("ILM","QP"),inset=.1)
Specific examples of problematic signature pairs

Next we revisit some of those outlying cases. First we need to fit the models.

QPres3<-QPsig(tumour.ref=datatemplate,86)
Linres3 <- whichSignatures(tumor.ref = datatemplate, 
  signatures.ref = signatures.nature2013,sample.id =86, 
  contexts.needed = TRUE, tri.counts.method = 'default')

QPres4<-QPsig(tumour.ref=datatemplate,15)
Linres4 <- whichSignatures(tumor.ref = datatemplate, 
  signatures.ref = signatures.nature2013,sample.id =15, 
  contexts.needed = TRUE, tri.counts.method = 'default')

QPres5<-QPsig(tumour.ref=datatemplate,257)

#dev.off()
Linres5 <- whichSignatures(tumor.ref = datatemplate,
                         signatures.ref = signatures.nature2013,sample.id = 257,
                         contexts.needed = TRUE, tri.counts.method = 'default')

QPres6 <- QPsig(tumour.ref = datatemplate, 351)
Linres6 <- whichSignatures(tumor.ref = datatemplate,
                         signatures.ref = signatures.nature2013,sample.id = 351,
                         contexts.needed = TRUE, tri.counts.method = 'default')

When the data are generated from signatures 5 and 13, the contribution of signature 5 is underestimated
(dramatically so for the ILM method). ILM sees contributions from Signatures 3 and 16, while QP has weight
for U2.

When the data are generated from signatures 2 and 5, the contribution of signature 5 is underestimated
(more so for the ILM method which sees a contribution from U2).
When the data are generated from signatures 1B and R2, the contribution of signature 1B is underestimated (more so for the ILM method). Perhaps unsurprisingly a contribution of 1A is seen to compensate.

When the data are generated from signatures U2 and R2, the contribution of signature U2 is underestimated (more so for the ILM method). R2 is slightly overestimated.
WGS data example

We now produce estimates for the oesophageal adenocarcinoma sample taken from Weaver et al. (Whole-genome sequencing data are available at the European Genome-phenome Archive (EGA; accession EGAD00001000704); however, data access approval is required via the ICGC data portal.)

```r
system.time(
QPresSS6003314<-QPsig(tumour.ref=datatemplate,354))
## user  system elapsed
##   0.016   0.000   0.016

system.time(LinresSS6003314 <- whichSignatures(tumor.ref = datatemplate,
signatures.ref = signatures.nature2013,
sample.id =354, contexts.needed = TRUE,
signature.cutoff=0.01))
## user  system elapsed
##  5.092   0.004  5.109

system.time(
QPresSS6003314b<-QPsig(tumour.ref=datatemplate,354,signatures.ref = signatures.cosmic))
## user  system elapsed
##   0.016   0.000   0.017
```
system.time(LinresSS6003314b <- whichSignatures(tumor.ref = datatemplate,
                                           signatures.ref = signatures.cosmic,
                                           sample.id = 354, contexts.needed = TRUE,
                                           signature.cutoff = 0.01))

## user  system elapsed
## 4.861 0.000  4.868

Nature 2013 Signatures

contribution

0
0.1
0.2
0.3
0.4
0.5
0.6
1A
1B
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
R1
R2
R3
U1
U2
Details of the packages used.

```r
sessionInfo()
```

```
## R version 3.2.4 Revised (2016-03-16 r70336)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu precise (12.04.5 LTS)
##
## locale:
## [1] LC_CTYPE=en_GB.UTF-8 LC_NUMERIC=C
## [3] LC_TIME=en_GB.UTF-8 LC_COLLATE=en_GB.UTF-8
## [5] LC_MONETARY=en_GB.UTF-8 LC_MESSAGES=en_GB.UTF-8
## [7] LC_PAPER=en_GB.UTF-8 LC_NAME=C
## [9] LC_ADDRESS=C LC_TELEPHONE=C
##[11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats graphics grDevices utils datasets methods base
##
## other attached packages:
## [1] quadprog_1.5-5 deconstructSigs_1.6.0
##
## loaded via a namespace (and not attached):
## [1] magrittr_1.5 formatR_1.4 tools_3.2.4 htmltools_0.3.5
## [5] yaml_2.1.13 Rcpp_0.12.5 stringi_1.0-1 rmarkdown_0.9.6
## [9] knitr_1.13 stringr_1.0.0 digest_0.6.9 evaluate_0.9
```
if(Sys.info()$"sysname" == "Linux") scan("/proc/cpuinfo", n=4, what="character", sep="\n", skip=1)

## [1] "vendor_id	: GenuineIntel"
## [2] "cpu family	: 6"
## [3] "model	: 45"
## [4] "model name	: Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz"

if(Sys.info()$"sysname" == "Linux") scan("/proc/meminfo", n=6, what="character", sep="\n")

## [1] "MemTotal:	65902308 kB" "MemFree:	57185292 kB"
## [3] "Buffers:	625728 kB" "Cached:	2300004 kB"
## [5] "SwapCached:	0 kB" "Active:	5252668 kB"