Qusage: Speeding up VIF in RcppArmadillo

Timothy J. Triche, Jr, Anthony R. Colombo

11 March, 2016

Contents

1 SpeedSage Intro

qusage is published software that is slow for large runs, SpeedSage corrects for speed and efficiency at large orders. there is Bottlenecking of Functions Qusage can improve the speed of its algorithm by minimizing the cost of computaiton.

1.1 changes Armadillo C++

trading NA flexibility slows down qusage runs, but having the user input no NAs enforcing good input, this speeds up calcIndividualExpressions, as well as using C++ libraries.

2 calcVif Function

This test the local version which enforces no NA in Baseline or PostTreatment object, this reduces the flexibility. this test data is from the vignette where postTreatment was modified to be Baseline+20.4, a simple training set from the QuSAGE vignette.

```
library(inline)
library(microbenchmark)
library(Rcpp)
```

```
##
## Attaching package: 'Rcpp'

## The following object is masked from 'package:inline':
##
##     registerPlugin
```

```
library(parallel)
library(speedSage)
```

```
## Loading required package: limma
```

```r
library(qusage)
```

```
##
## Attaching package: 'qusage'

## The following objects are masked from 'package:speedSage':
##
##     aggregateGeneSet, calcBayesCI, calcVIF, getXcoords,
##     makeComparison, read.gmt
```

```r
library(ggplot2)
eset<-system.file("extdata","eset.RData",package="speedSage")
load(eset)
labels<-c(rep("t0",134),rep("t1",134))
contrast<-"t1-t0"
colnames(eset)<-c(rep("t0",134),rep("t1",134))
fileISG<-system.file("extdata","c2.cgp.v5.1.symbols.gmt",package="speedSage")
ISG.geneSet<-read.gmt(fileISG)
geneSets<-ISG.geneSet[grepl("DER_IFN_GAMMA_RESPONSE_UP",names(ISG.geneSet))]
Baseline<-eset
PostTreatment<-eset+20.4
ncol(Baseline) #not splitting up eset
```

```
## [1] 268
```

```r
i<-1  #for testing 1 gene set
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/sigmaArm.cpp")
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/sigmaSingle.cpp")
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/bayesEstimation.cpp")
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/notbayesEstimation.cp
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/calcVIFarm.cpp")
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/calcVIFarmalt.cpp")
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/calcVIFarm_nosdalphaal

pairVector<-NULL
var.equal<-FALSE
filter.genes<-FALSE
n.points<-2^12


#setting up calcVif call objects
results = makeComparisonArm(eset, labels, contrast, pairVector=pairVector,var.equal=var.equal)
```

```
## Found more than one class "QSarray" in cache; using the first, from namespace 'speedSage'
```

```r
nu = floor(min(results$dof,na.rm=T))
geneResults = aggregateGeneSet(results, geneSets, silent=F, n.points=n.points)
#eset, and results parameters for vif

useAllData<-TRUE
useCAMERA<-FALSE
```

```
##### qusage VIF calc
ogVIF<-calcVIF(eset,geneResults)

#for local code use
ogGeneSets<-geneResults$pathways
GNames<-names(geneResults$mean)[ogGeneSets[[1]]]
gs.i = which(rownames(eset)%in%GNames)
###now to compare
```

```
library(speedSage)
useCAMERA<-FALSE
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/calcVIFarmalt.cpp")
sourceCpp(file="/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/calcVIFarm_nosdalphaal


t2<-calcVIFarmalt(names(geneResults$mean),gs.i, geneResults$pathways[[i]],rownames(eset),eset,levels(ger



library(ggplot2)
speedUp<-microbenchmark(
calcVIFarmalt(names(geneResults$mean),gs.i, geneResults$pathways[[1]],rownames(eset),eset,levels(geneRes
calcVIF(eset,geneResults) )
speedUp
```

```
## Unit: microseconds
##
##  calcVIFarmalt(names(geneResults$mean), gs.i, geneResults$pathways[[1]],      rownames(eset), eset, l
##
##       min        lq      mean   median       uq      max neval cld
##   534.906  559.4425  624.8023  647.371  678.531  751.661   100  a
##  2185.080 2237.3550 2350.3630 2278.771 2320.377 5524.896   100   b
```

```
source("/home/anthonycolombo/Documents/qusage/qusage_repos/qusage_speed/R/calcVIFArm.R")
myVIF<-calcVIFArm(eset,geneResults)



### TO DO ensure that calcVIFarm matches qusage::calcVIF.R

identical(names(myVIF),names(ogVIF))
```

```
## [1] TRUE
```

```
identical(myVIF$labels,ogVIF$labels)
```

```
## [1] TRUE
```

```
 identical(myVIF$contrast,ogVIF$contrast)
```

```
## [1] TRUE
```

```
identical(myVIF$n.samples,ogVIF$n.samples)
```

```
## [1] TRUE
```

```
identical(myVIF$mean,ogVIF$mean)
```

```
## [1] TRUE
```

```
 identical(myVIF$sd.alpha,ogVIF$sd.alpha)
```

```
## [1] TRUE
```

```
 identical(myVIF$dof,ogVIF$dof)
```

```
## [1] TRUE
```

```
identical(myVIF$var.method,ogVIF$var.method)
```

```
## [1] TRUE
```

```
identical(myVIF$pathways,ogVIF$pathways)
```

```
## [1] TRUE
```

```
identical(myVIF$path.mean,ogVIF$path.mean)
```

```
## [1] TRUE
```

```
 identical(myVIF$path.size,ogVIF$path.size)
```

```
## [1] TRUE
```

```
identical(myVIF$ranges,ogVIF$ranges)
```

```
## [1] TRUE
```

```
 identical(myVIF$n.points,ogVIF$n.points)
```

```
## [1] TRUE
```

```
all.equal(myVIF$path.PDF,ogVIF$path.PDF)
```

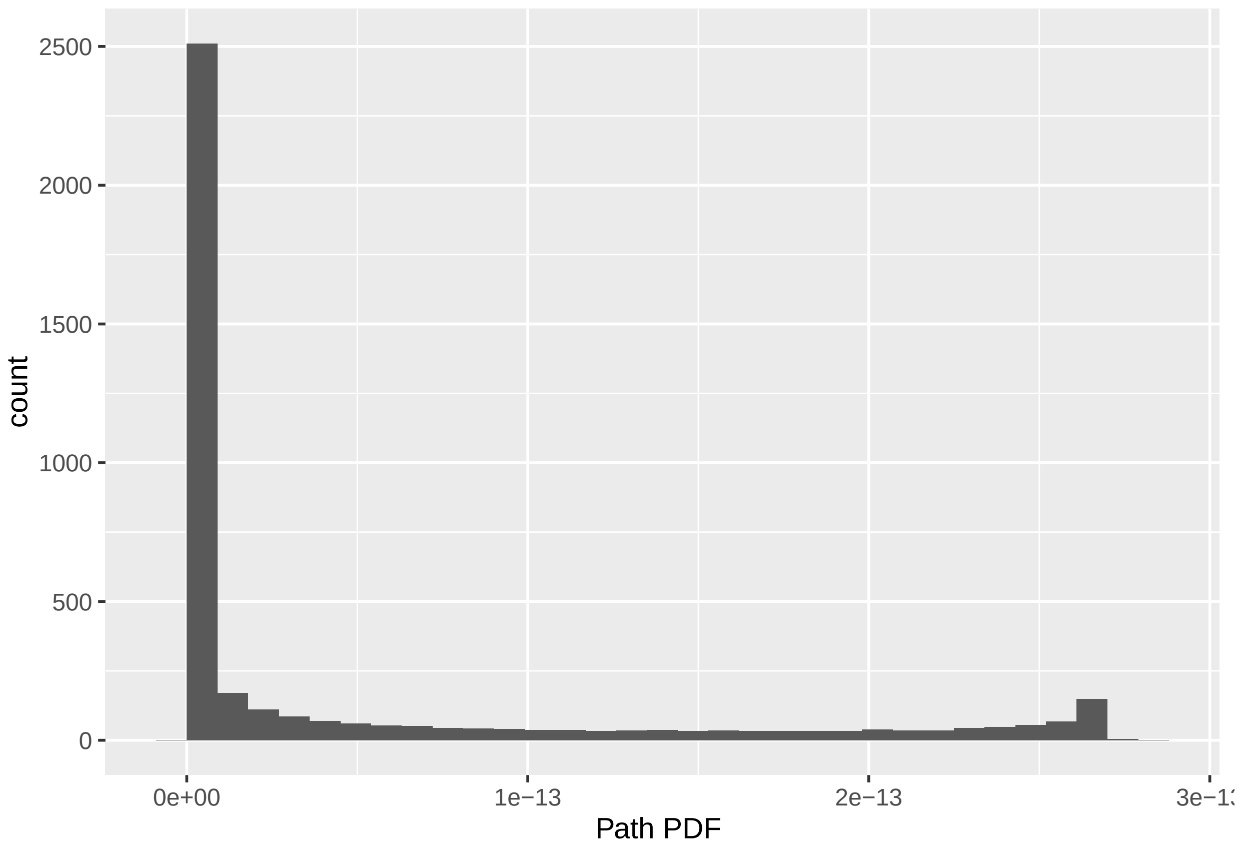
```
## [1] TRUE
```

```r
all.equal(myVIF$vif,ogVIF$vif)
```

```
## [1] TRUE
```

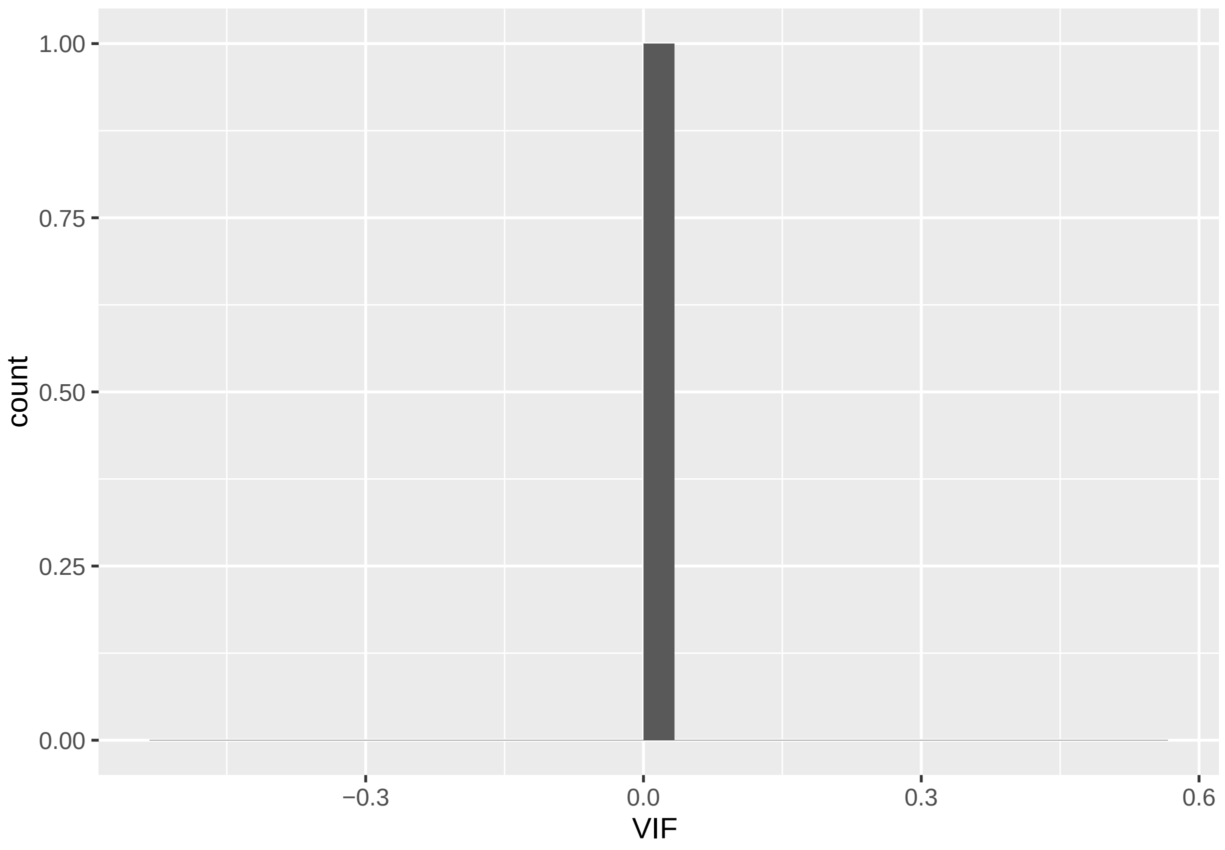
```r
qplot( as.vector(abs(myVIF$path.PDF-ogVIF$path.PDF)), xlab="Path PDF ")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



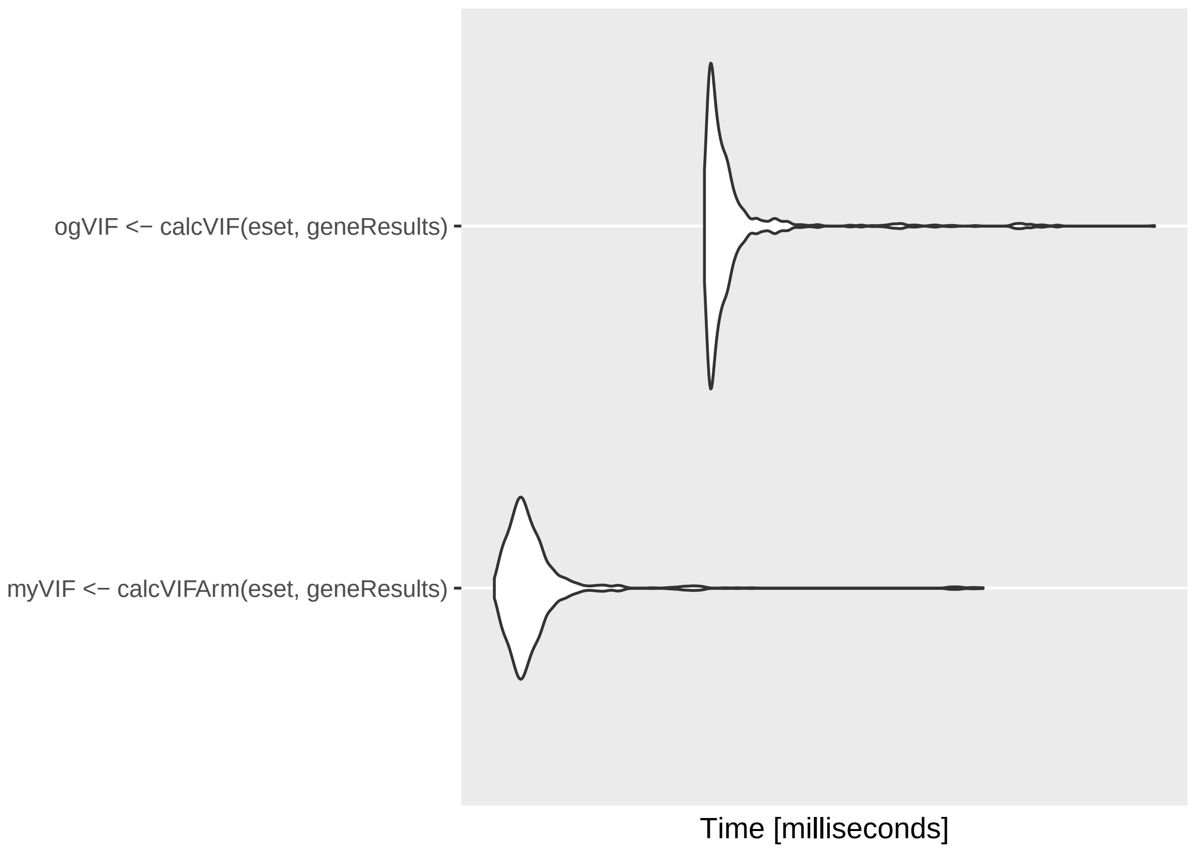
```r
qplot( as.vector(abs(myVIF$vif-ogVIF$vif)), xlab="VIF ")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
library(ggplot2)
mb<-microbenchmark(
myVIF<-calcVIFArm(eset,geneResults),
ogVIF<-calcVIF(eset,geneResults),times=1000 )

autoplot(mb)
```



```
mb
```

```
## Unit: milliseconds
##                                     expr      min       lq     mean
##  myVIF <- calcVIFArm(eset, geneResults) 1.195207 1.266857 1.366833
##     ogVIF <- calcVIF(eset, geneResults) 2.183034 2.222915 2.412804
##    median       uq      max neval cld
##  1.303616 1.358354 4.854424  1000  a
##  2.267464 2.344709 7.933272  1000   b
```