Supplement: Splice Expression Variation Analysis (SEVA): Variability Analysis to Detect Significant Alternative Splicing Events

Bahman Afsari

Sunday, June 12, 2016

1 Preperations

1.1 Loading Library

First, we load the libraries:

```
library('Homo.sapiens')
library('org.Hs.eg.db')
library('GenomicRanges')
library("GSReg")
library(EBSeq)
library(limma)
library('gplots')
library('ROCR')
library(Matrix)
library(limma)

sessionInfo()
```

```
## R version 3.3.3 (2017-03-06)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] Matrix_1.2-11
##  [2] ROCR_1.0-7
##  [3] limma_3.30.13
##  [4] EBSeq_1.14.0
##  [5] testthat_1.0.2
##  [6] gplots_3.0.1
##  [7] blockmodeling_0.1.9
##  [8] GSReg_1.9.2
##  [9] Homo.sapiens_1.3.1
## [10] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
## [11] org.Hs.eg.db_3.4.0
## [12] GO.db_3.4.0
## [13] OrganismDbi_1.16.0
## [14] GenomicFeatures_1.26.4
## [15] GenomicRanges_1.26.4
## [16] GenomeInfoDb_1.10.3
## [17] AnnotationDbi_1.36.2
## [18] IRanges_2.8.2
## [19] S4Vectors_0.12.2
## [20] Biobase_2.34.0
## [21] BiocGenerics_0.20.0
##
## loaded via a namespace (and not attached):
##  [1] SummarizedExperiment_1.4.0 gtools_3.5.0
##  [3] lattice_0.20-35            htmltools_0.3.6
##  [5] rtracklayer_1.34.2         yaml_2.1.14
##  [7] blob_1.1.0                 XML_3.98-1.9
##  [9] RBGL_1.50.0                rlang_0.1.2
## [11] DBI_0.7                    BiocParallel_1.8.2
## [13] bit64_0.9-7                stringr_1.2.0
## [15] zlibbioc_1.20.0            Biostrings_2.42.1
## [17] caTools_1.17.1             memoise_1.1.0
## [19] evaluate_0.10.1            knitr_1.17
## [21] biomaRt_2.30.0             BiocInstaller_1.24.0
## [23] Rcpp_0.12.13               KernSmooth_2.23-15
## [25] backports_1.1.1            gdata_2.18.0
## [27] graph_1.52.0               XVector_0.14.1
## [29] bit_1.1-12                 Rsamtools_1.26.2
## [31] digest_0.6.12              stringi_1.1.5
## [33] rprojroot_1.2              grid_3.3.3
## [35] tools_3.3.3                bitops_1.0-6
## [37] magrittr_1.5               RCurl_1.95-4.8
## [39] tibble_1.3.4               RSQLite_2.0
## [41] crayon_1.3.4               pkgconfig_2.0.1
## [43] rmarkdown_1.6              R6_2.2.2
## [45] GenomicAlignments_1.10.1
```

1.2 Loading Data Joe's Data:

```
source("../Scripts/functions.R") #loading the functions for analysis

### loading Joe's data


## loading gene expression of Joe data
load("../Data/JoeData/CalifanoHPVOP_RSEM_28Jul2014.RDa")

### loading junction expression data
load("../Data/JoeData/juncRPM.rda")

# loading isoform expression
load("../Data/JoeData/isoforms.rda")
```

```
#loading the map of the isoform names to genes
load("../Results/Simulation/SecondTryJan13/isos2genesvect.rda")
```

Now, we preprocess data to get the sample phenotypes from the data:

```
# Normal Samples Names
NormalSamp <- pheno[which(pheno["classes"]=="Normal"),"junctionSample"]
# Tumor Samples Names
TumorSamp <- pheno[which(pheno["classes"]=="Tumor"),"junctionSample"]

# Generating a vector maps the sample names to phenotypes
phenoVect <- c(rep(x= "Normal",length(NormalSamp)),rep(x="Tumor",length(TumorSamp)))
names(phenoVect) <- c(NormalSamp,TumorSamp)

#gene exp removing duplicated names
geneexp <- HPVOPRSEMData[which(duplicated(sapply(strsplit(rownames(HPVOPRSEMData),
                                                          split = "[|]"),
                                                 FUN = function(x) x[[1]]))==F),]
#correct colname (Sample) name
colnames(geneexp) <- gsub(pattern = "[.]",replacement = "-" ,
                          x = sapply(strsplit(colnames(HPVOPRSEMData),split = "_"),
                                     function(x) x[2]))
#correct gene names
rownames(geneexp)<- sapply(strsplit(rownames(geneexp),split = "[|]"),
                           FUN = function(x) x[[1]])

#gene expression of only phenoVect
geneexp <- geneexp[,names(phenoVect)]

#logscale geneexp
loggeneExp <- log2(geneexp+1)
```

2 Generating subplots for Figure 2

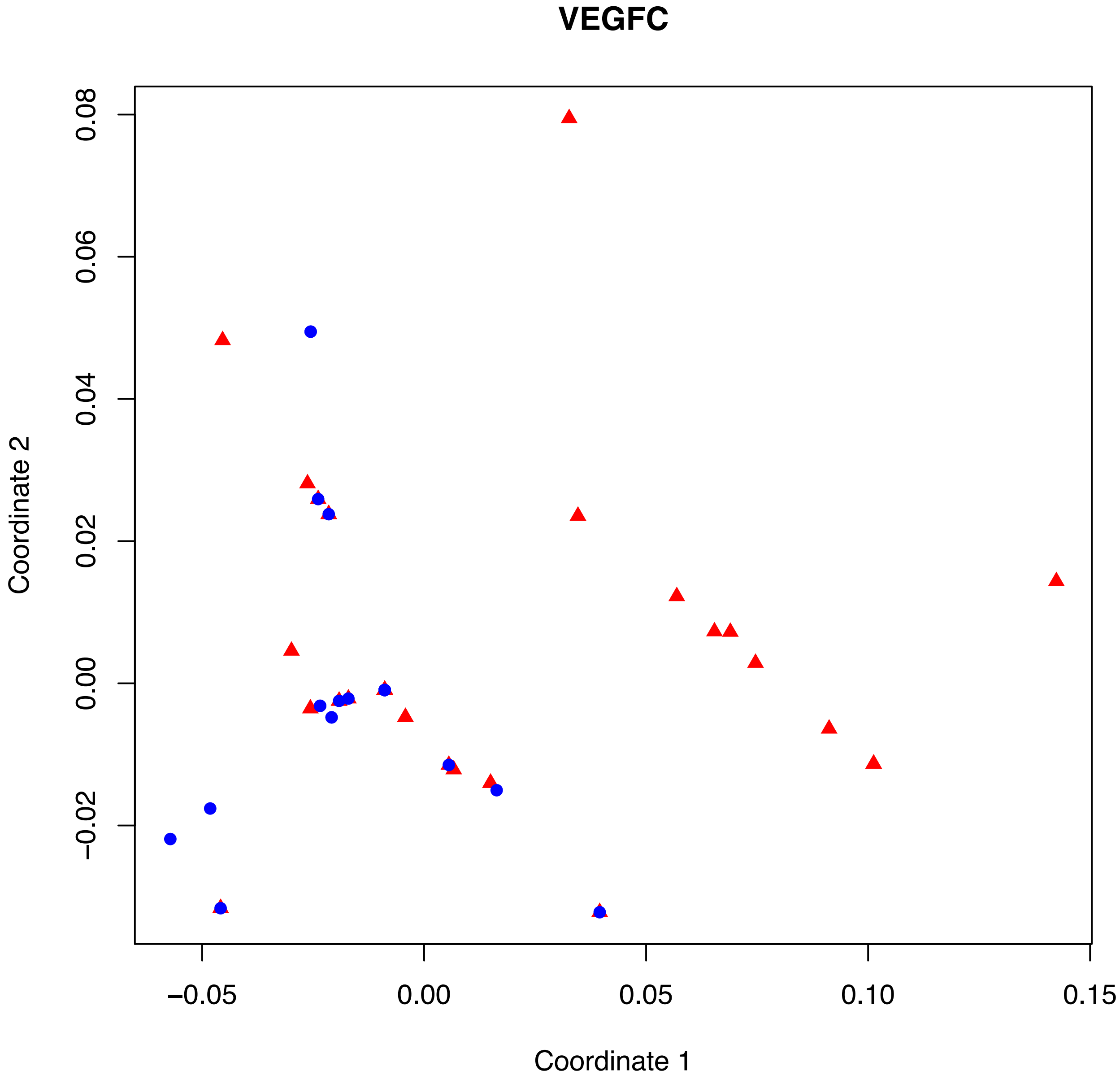
We studied genes from a previous study. We calculated the modified Kendall-tau distance only on those genes and we plotted MDS to visualize the samples.

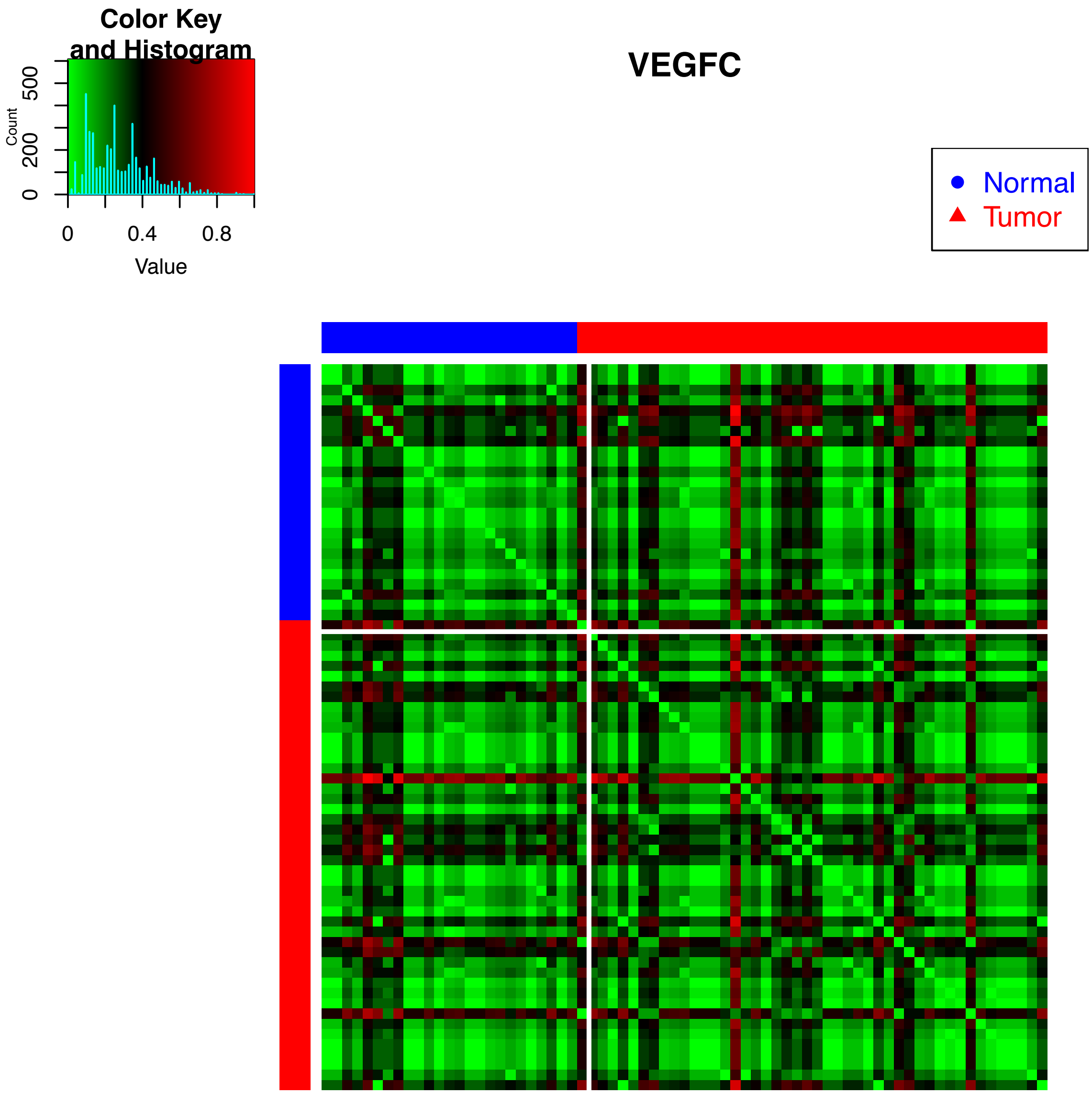
```
# Genes from PLOS one paper I. Smith et al from 2013 PLoS one
# Coordinated Activation of Candidate Proto-Oncogenes
# and Cancer Testes Antigens via Promoter Demethylation
# in Head and Neck Cancer and Lung Cancer

GenestoStudy <- c("VEGFC","DST","LAMA3","SDHA","TP63","RASIP1")

## Find the overlaps for all junctions.
z <-  GSReg.overlapJunction(juncExprs = junc.RPM,
                              GenestoStudy = GenestoStudy)
```

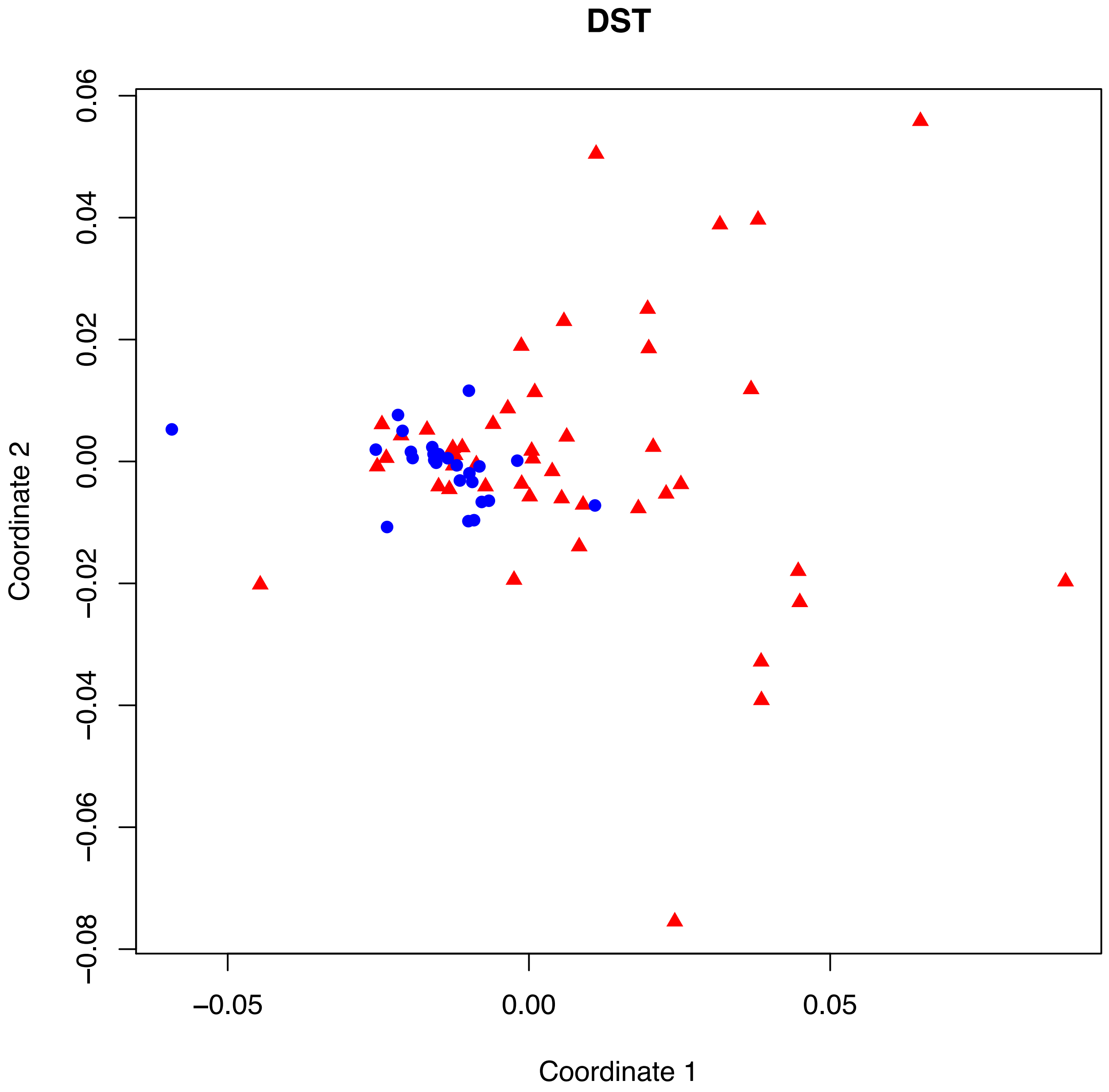
```
## 'select()' returned 1:1 mapping between keys and columns
```

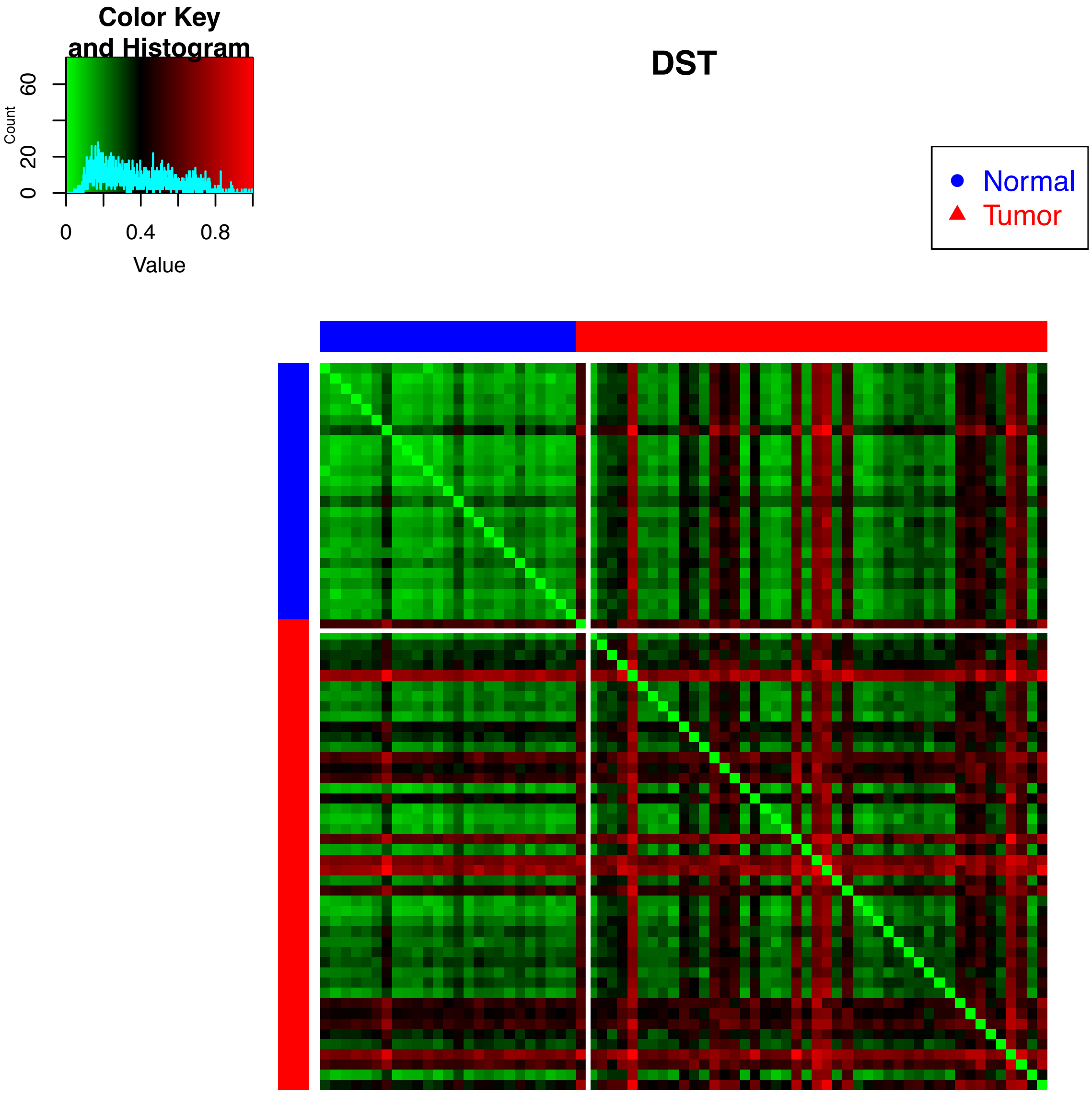
```
MyRest <- z$Rest

## Tumor Samples
```

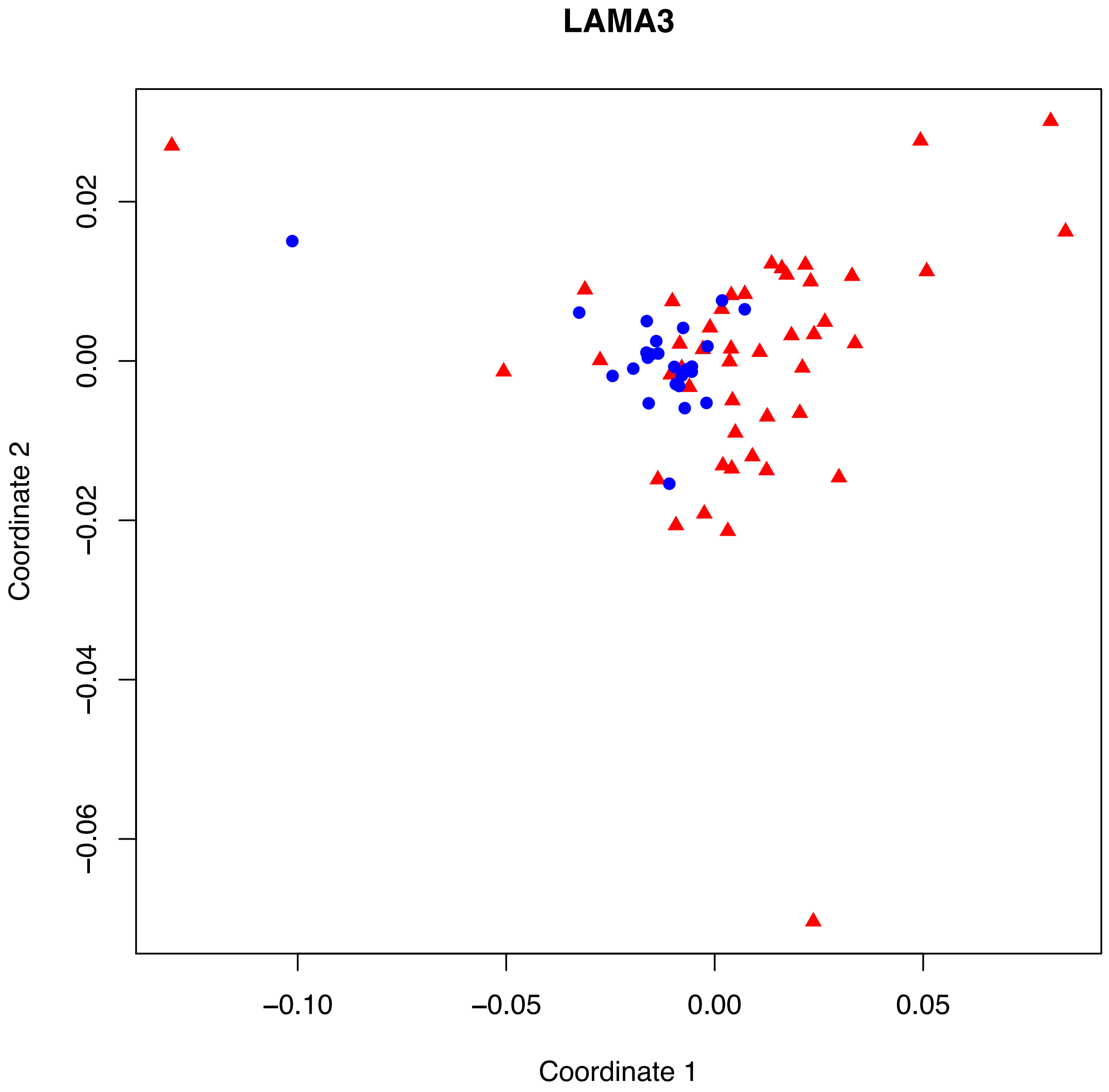
```
TumorSamp <- pheno[which(pheno[,"classes"]=="Tumor"),"junctionSample"]#Tumor samples
NormSamp <- pheno[which(pheno[,"classes"]=="Normal"),"junctionSample"]#Normal samples

### only Tumor and Normal samples
junc.RPM.NT <- cbind(junc.RPM[,TumorSamp],junc.RPM[,NormSamp])#only tumor and normal

for( i in seq_along(GenestoStudy)){
  GenetoStudy <- GenestoStudy[i]
  GeneRestMat <- MyRest[[GenetoStudy]]

  # calculating the distance
  dist <- GSReg.kendall.tau.distance.restricted(
                            V = junc.RPM.NT[rownames(GeneRestMat),],
                            RestMat = GeneRestMat)


  # Calculating the mds plot
  fit <- cmdscale(dist,eig=TRUE, k=2) # k is the number of dim

  # coordinations
  x <- fit$points[,1]
  y <- fit$points[,2]

  # plotting mds
  plot(x = x[TumorSamp], y = y[TumorSamp],
       xlab="Coordinate 1", ylab="Coordinate 2",
       xlim =range(x),ylim = range(y),
       main= GenetoStudy,   type="p", col = 'red',pch = 17)

  lines(x[NormSamp], y[NormSamp],   main=GenetoStudy,   type="p", col="blue", pch = 16)
  maxdist <- max(dist[c(NormSamp,TumorSamp),c(NormSamp,TumorSamp)])
  breaks = seq(0,1,length.out=1000)
  gradient1 = colorpanel( sum( breaks[-1]<= 0.4 ), "green", "black" )
  gradient2 = colorpanel( sum( breaks[-1]> 0.4 ), "black", "red" )
  hm.colors = c(gradient1,gradient2)

  ## heatmap of distances
  heatmap.2(x = dist[c(NormSamp,TumorSamp),c(NormSamp,TumorSamp)]/maxdist, main = GenetoStudy,
            Rowv = FALSE, Colv = FALSE,
            colsep= length(NormSamp)+1,
            rowsep = length(NormSamp)+1,
            sepcolor = "white",
            sepwidth = c(0.3,0.3),
            RowSideColors = c(rep("blue",length(NormSamp)),rep("red",length(TumorSamp))),
            ColSideColors = c(rep("blue",length(NormSamp)),rep("red",length(TumorSamp))),
            dendrogram = "none",scale="none",
            na.rm = T,col = hm.colors,
            labRow = "",labCol = "",trace="none")



  legend("topright",      # location of the legend on the heatmap plot
         legend = c("Normal", "Tumor"), # category labels
         col = c("blue", "red"),  # color key
         text.col = c("blue", "red"),  # color key
         lty= 0,             # line style
         pch = c(16,17)           # line width
  )
}
```

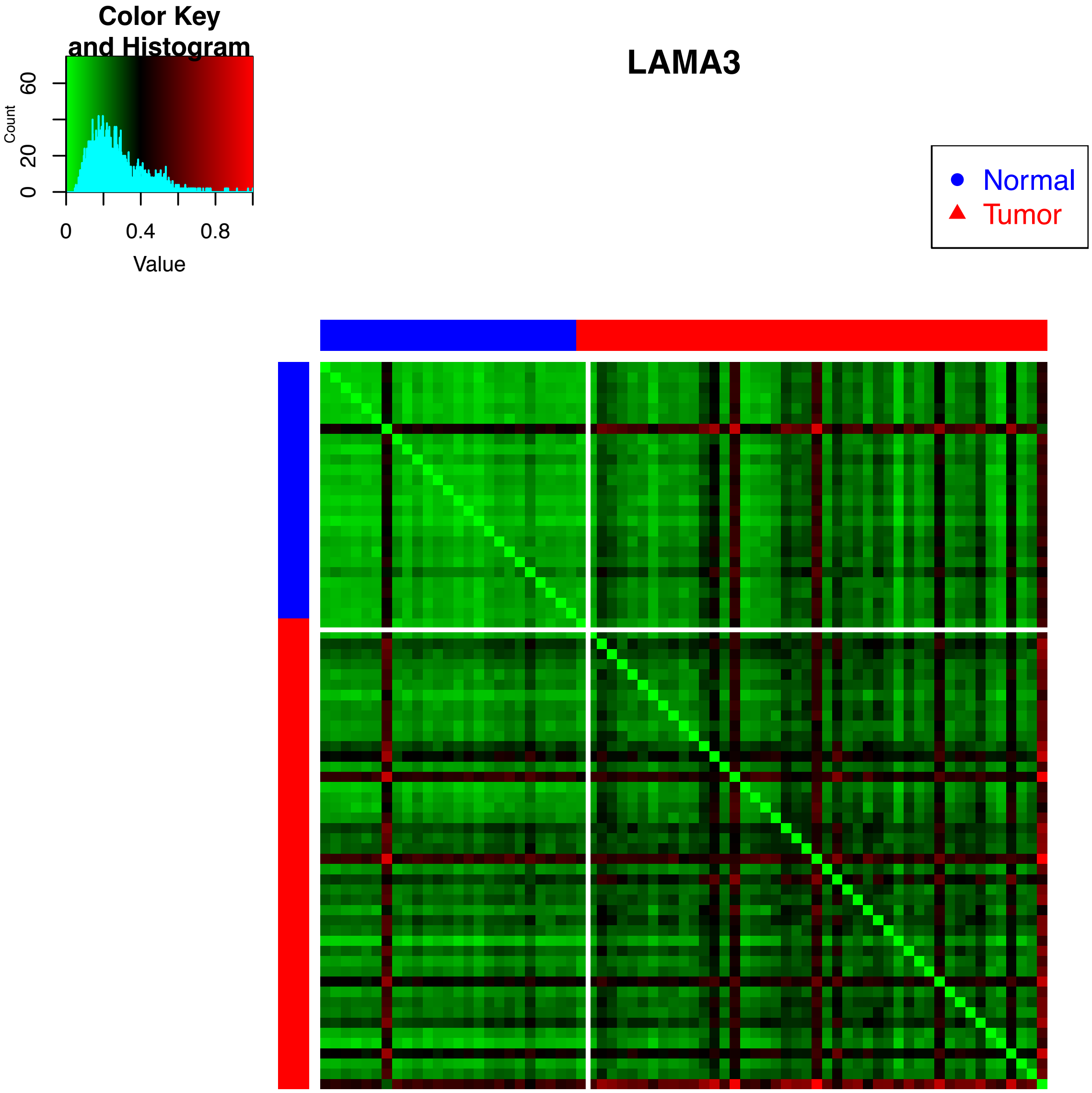


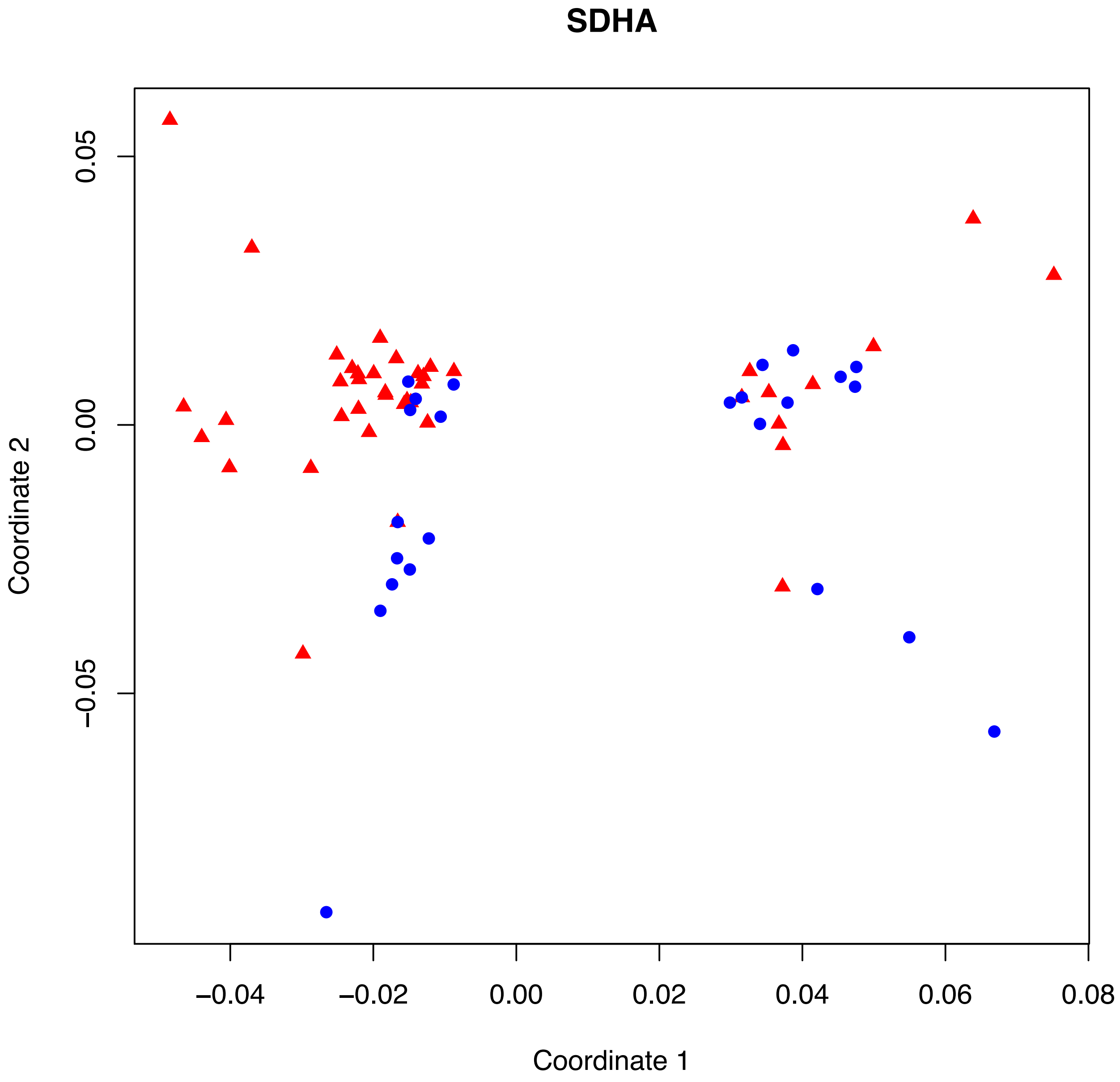

Color Key
and Histogram
Count
0
200
500
0
0.4
0.8
Value
VEGFC
● Normal
▲ Tumor

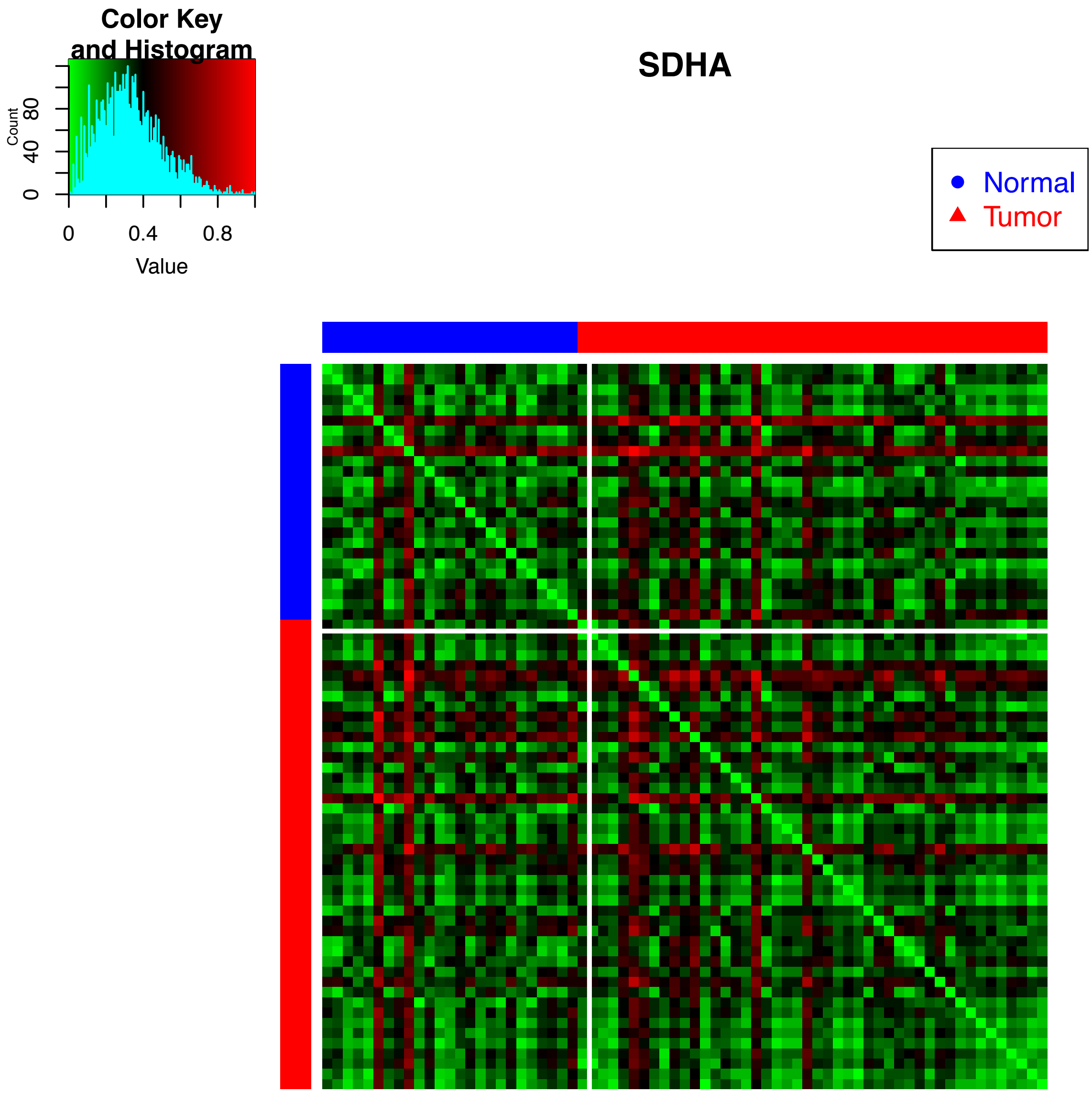

DST
Coordinate 2
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
Coordinate 1
−0.05
0.00
0.05

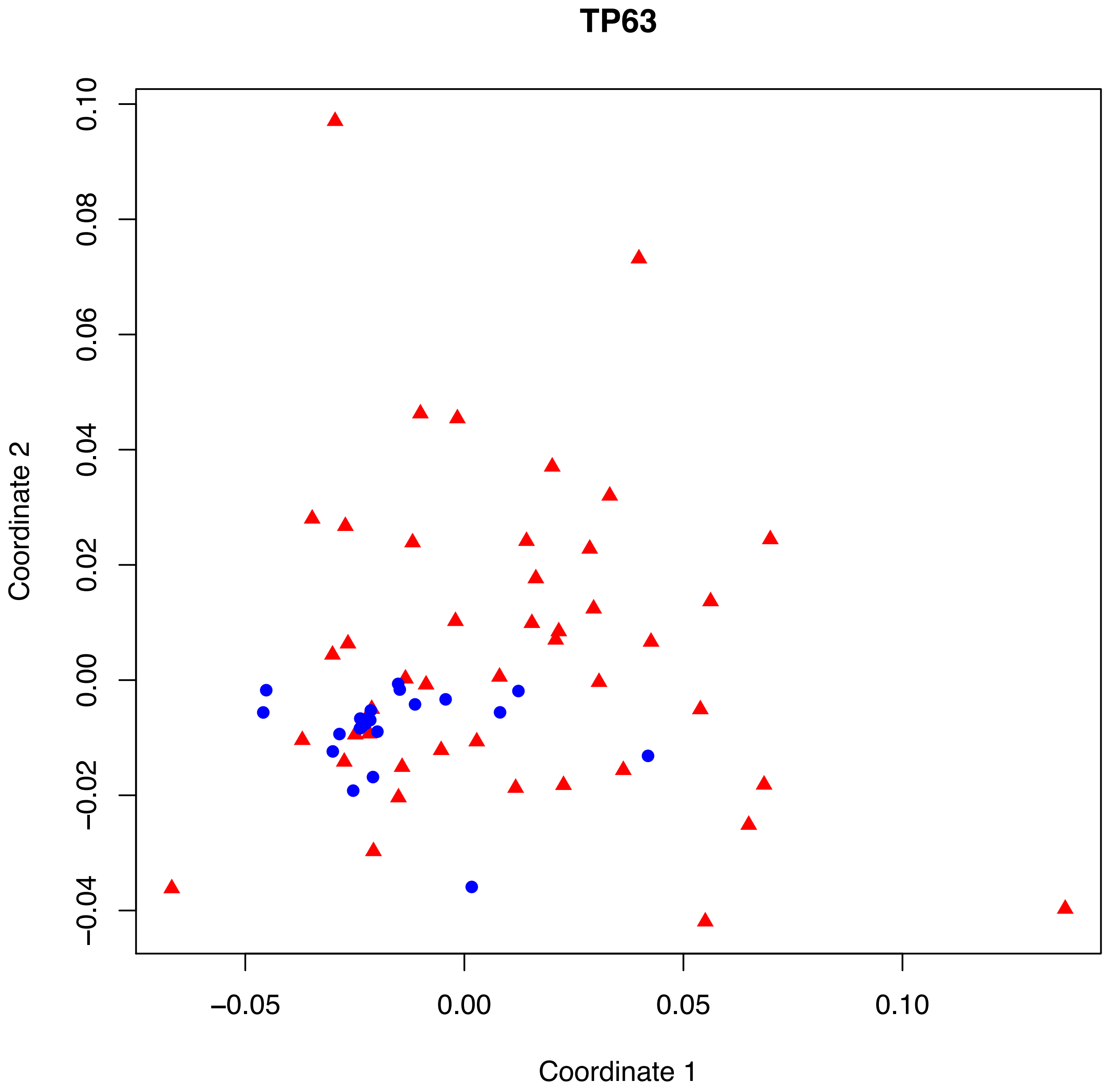

Color Key
and Histogram
Count
0
20
60
0
0.4
0.8
Value
DST
● Normal
▲ Tumor

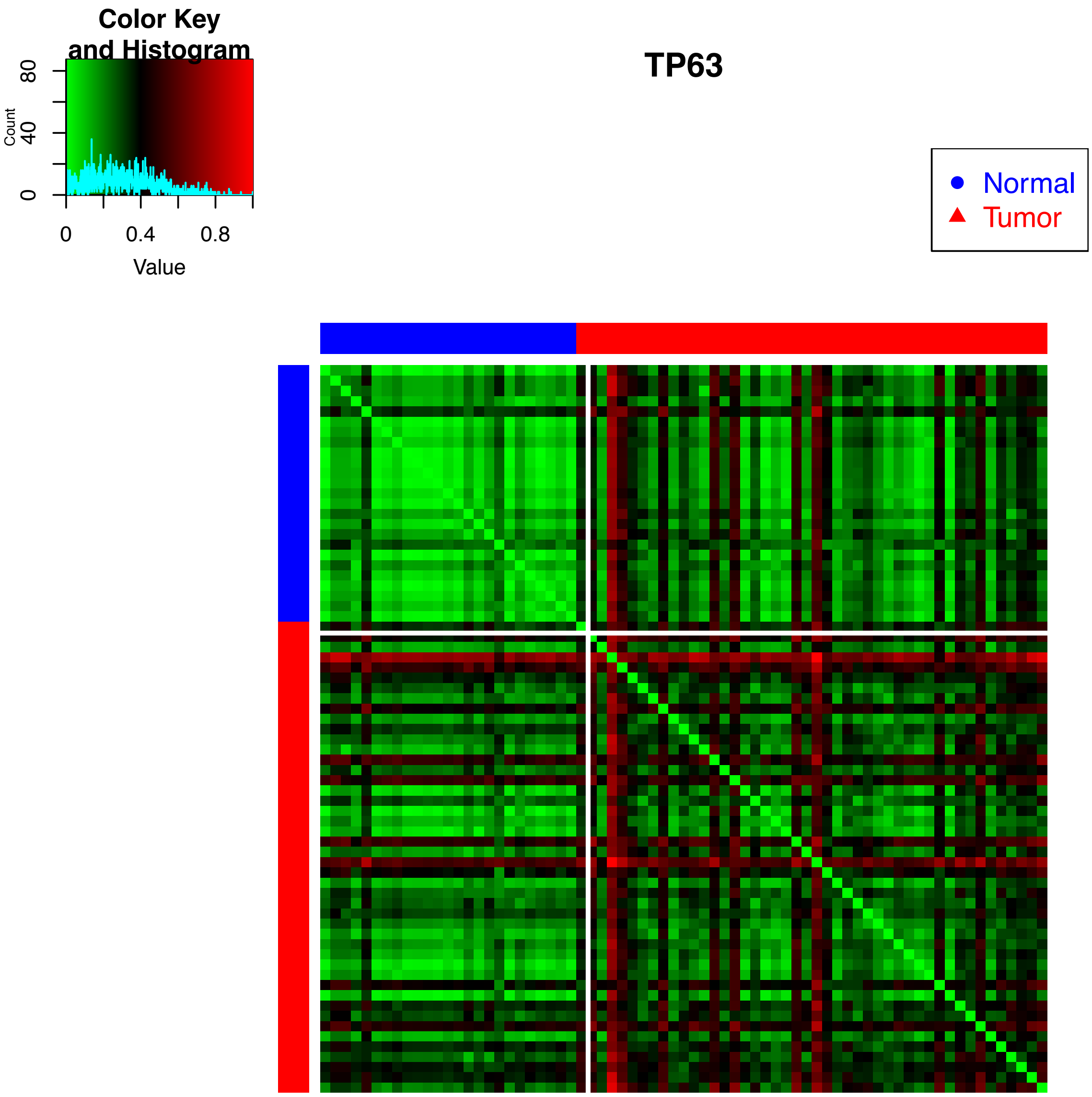

LAMA3
Coordinate 2
0.02
0.00
−0.02
−0.04
−0.06
−0.10
−0.05
0.00
0.05
Coordinate 1

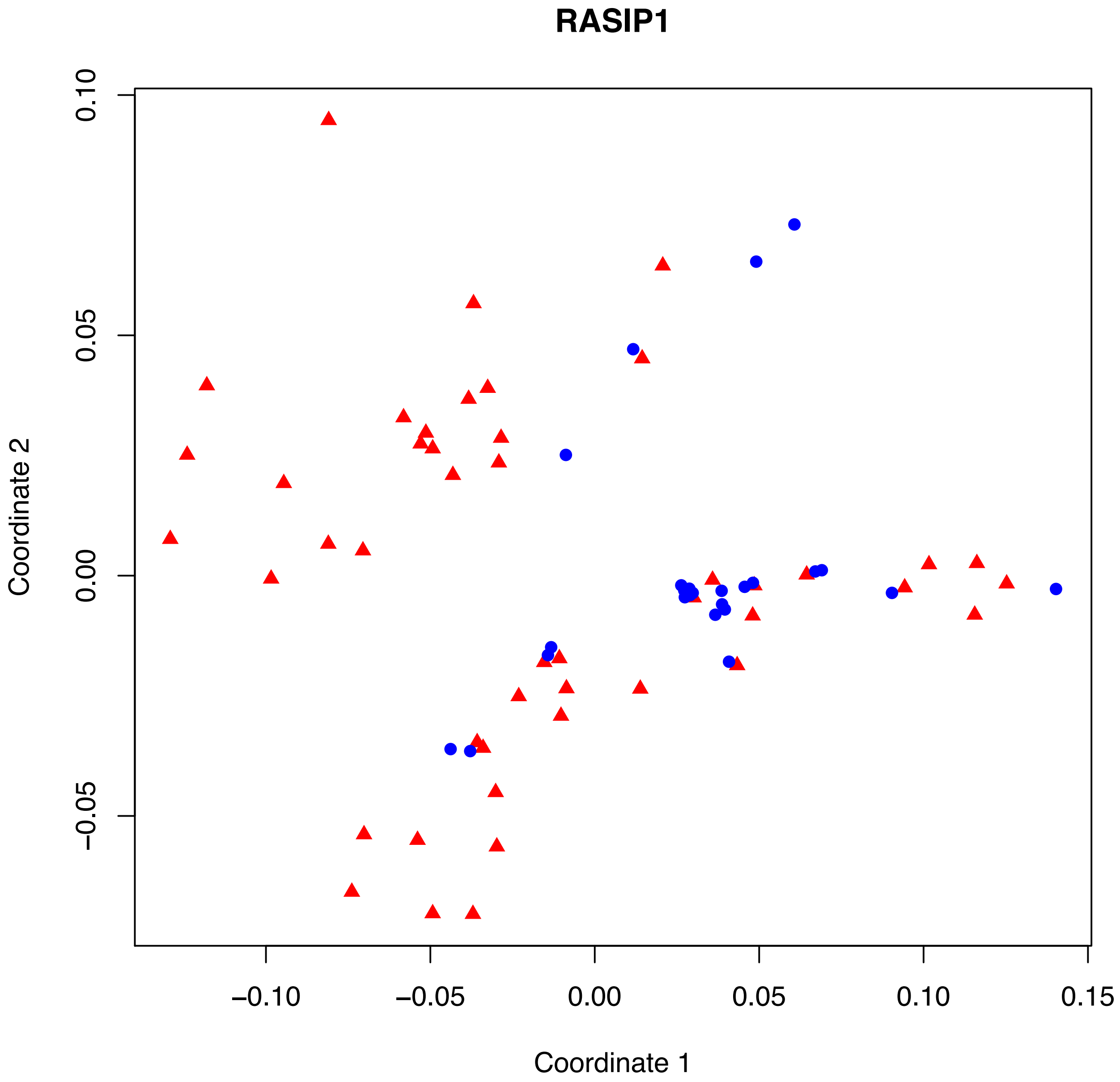

Color Key
and Histogram
Count
0
20
60
0
0.4
0.8
Value
LAMA3
● Normal
▲ Tumor

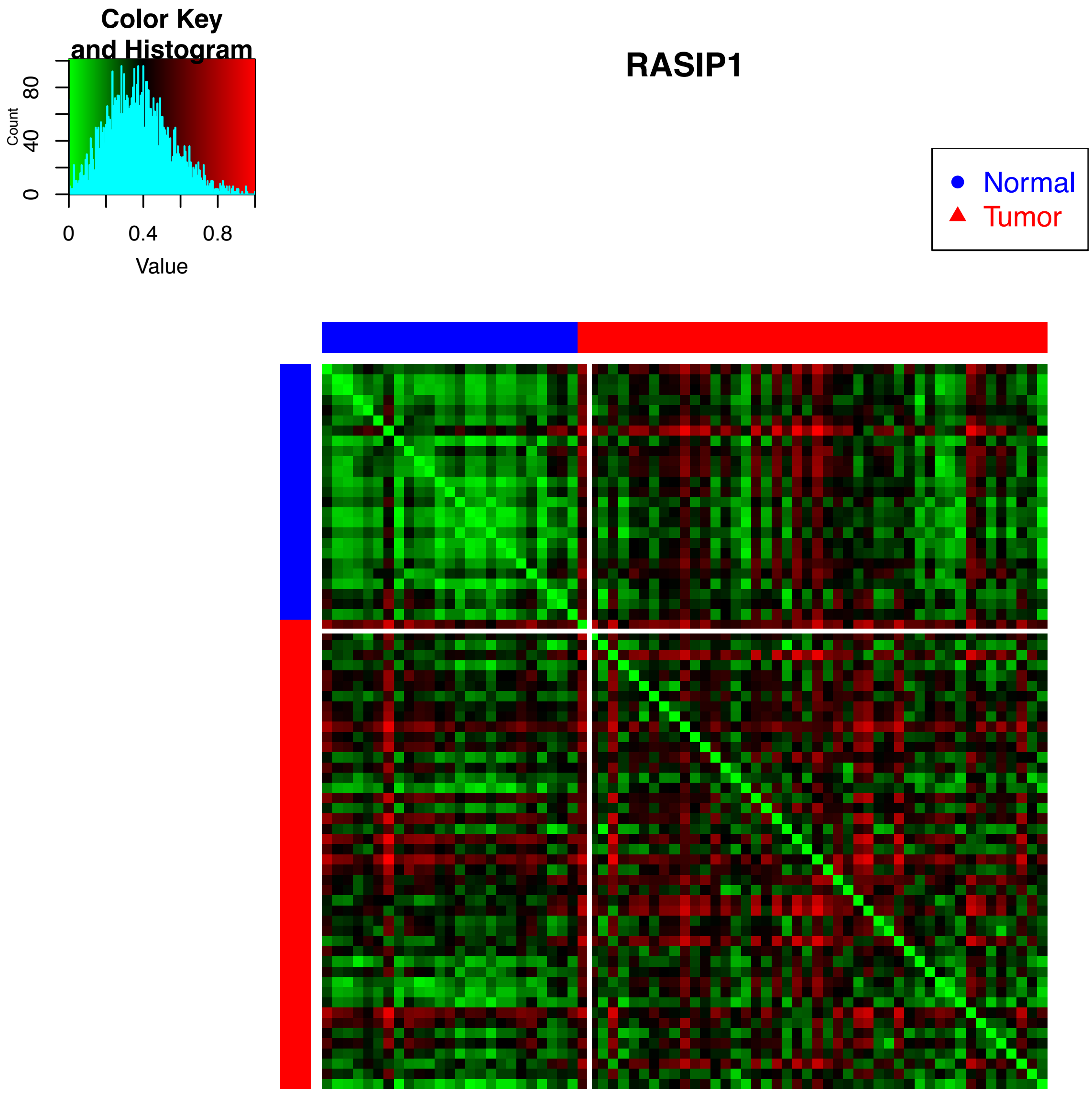

SDHA
Coordinate 1
Coordinate 2
0.05
0.00
−0.05
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08


Color Key
and Histogram
Count
0
40
80
0
0.4
0.8
Value
SDHA
● Normal
▲ Tumor


TP63
Coordinate 1
Coordinate 2
−0.05
0.00
0.05
0.10
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
0.10


Color Key
and Histogram
Count
0
40
80
0
0.4
0.8
Value
TP63
● Normal
▲ Tumor


RASIP1
Coordinate 1
Coordinate 2
−0.10
−0.05
0.00
0.05
0.10
0.15
−0.05
0.00
0.05
0.10



```
rm(list="junc.RPM.NT")
```

3 Generating Figures 4 and Table 1

First, we define the functions to find differential expression genes

```
### DEgenes
findDEGenes <- function(geneexp,phenoVect,pvaluecorrected=0.01){
  geneexpr <- as.matrix(geneexp)

  Sizes = MedianNorm(geneexpr)
  EBOut = EBTest(Data = geneexpr,
               Conditions = as.factor(phenoVect),
                sizeFactors = Sizes, maxround = 5)


  PP = GetPPMat(EBOut)
  DEGenes_EBesq <- as.character(names(which(PP[,"PPDE"]>1-pvaluecorrected)))
  return(list(DEGenes_EBesq=DEGenes_EBesq,pvaluescorrected=PP[,"PPDE"]))
}
```

Also, we write a wrapper for differential splicing algorithm using EBSeq:

```
### EBSEQ isoform DS
findDSEBSEQ<-function(isos,phenoVect, isos2genesvectsimplified, pvaluecorrected= 0.05 )
{
  isoexpressed <- names(which(apply(isos,MARGIN = 1, FUN = sum)>0.0000001))
  Sizes = MedianNorm(isos[isoexpressed,])
  IsoEBOut = EBTest(Data = isos[isoexpressed,],
                    Conditions = as.factor(phenoVect),
                    sizeFactors = Sizes, maxround = 5)


  PPISO = GetPPMat(IsoEBOut)
  ebseqpvalueGenes <- tapply(  X = PPISO[,"PPDE"] ,
                               INDEX = isos2genesvectsimplified[rownames(PPISO)],
                               FUN = function(x) mean(x,na.rm=T))
  DSEBseq <- names(which(ebseqpvalueGenes>1-pvaluecorrected))
  return(list(DSEBseq=DSEBseq, pvaluescorrected=ebseqpvalueGenes))

}
```

Now, we use EBSeq to find differential splicing.

```
#finding DE genes
DEGenes_EBesq_outcome <- findDEGenes(geneexp = geneexp,phenoVect=phenoVect)
```

```
## Removing transcripts with 100 th quantile < = 0
## 20355 transcripts will be tested
```

```
DEGenes_EBesq <- DEGenes_EBesq_outcome$DEGenes_EBesq

## EBSeq requires isoforms
samplesIsos <- intersect(names(phenoVect),colnames(isos))## we do not have all isos expression in all g
isos2genesvectsimplified <- sapply(strsplit(isos2genesvect,split = '[|]'),FUN = function(x) x[1])

ptm <- proc.time()
## EBSeq for Differential Splicing
DSEBseq_outcome <- findDSEBSEQ(isos[,samplesIsos], ### DSEBSEQ
                               as.factor(phenoVect)[samplesIsos],
                               isos2genesvectsimplified = isos2genesvectsimplified)
cat("EBSEQ time:",proc.time() - ptm,"\n")
```

```
## EBSEQ time: 788.143 231.205 1024.165 0 0
```

```
DSEBseq_Genes <- DSEBseq_outcome$DSEBseq



### Make the Venn matrix with EBSEQ and DiffSplice for further application
```

```
VennMatrix <- matrix(data=0,ncol = 4,nrow= nrow(geneexp),
                     dimnames = list(rownames=rownames(geneexp),colnames=c("DE","SEVA","EBSEQ","DiffSpli

VennMatrix[DEGenes_EBesq,"DE"] <- 1# DE genes
VennMatrix[intersect(rownames(VennMatrix),DSEBseq_Genes),"EBSEQ"] <- 1 #EBSEQ genes
```

DiffSplice analysis takes a longer time and requires more resources to run. So, we have applied it seperately and we load its outcome.

```
### loading genes
gn <- genes(TxDb.Hsapiens.UCSC.hg19.knownGene)
gSymbol <- select(org.Hs.eg.db,keys=as.character(gn$gene_id),
                  columns=c('SYMBOL'),keytype='ENTREZID')
gn$SYMBOL <- gSymbol$SYMBOL

#Read the DiffSplice genes (the analysis has been done in a different computer)
diffsplicefile <- read.csv("../Data/JoeData/differential_transcription_sig.csv")

junctionsDiffSplice <- GRanges(seqnames = Rle(diffsplicefile[,"chromosome"]),
                              ranges = IRanges(
                                start = as.numeric(diffsplicefile[,"position_start"]),
                                end = as.numeric(diffsplicefile[,"position_end"])))

DiffSplicehits <- findOverlaps(junctionsDiffSplice,gn)
DiffSplicegenes <- unique(gn$SYMBOL[subjectHits(DiffSplicehits)])
VennMatrix[intersect(rownames(VennMatrix),DiffSplicegenes),"DiffSplice"] <- 1


##################################### END of DE, EBSEQ and DiffSplice
```

Now, we apply the SEVA analysis for the genes:

```
### Applying SEVA
geneexpr <- as.matrix(geneexp)

ptm <- proc.time()
junctionPValue <- SEVA.meangeneFilter(junc.RPM = junc.RPM,
                             phenoVect=phenoVect,
                             minmeanloggeneexp= 3,
                             geneexpr=geneexpr)
```

```
## 100
## 200
## 300
## 400
## 500
## 600
## 700
## 800
## 900
## 1000
## 1100
## 1200
## 1300
## 1400
## 1500
## 1600
## 1700
## 1800
## 1900
## 2000
## 2100
## 2200
## 2300
## 2400
## 2500
## 2600
## 2700
## 2800
## 2900
## 3000
## 3100
## 3200
## 3300
## 3400
## 3500
## 3600
## 3700
## 3800
## 3900
## 4000
## 4100
## 4200
## 4300
## 4400
## 4500
## 4600
## 4700
## 4800
## 4900
## 5000
## 5100
## 5200
## 5300
## 5400
## 5500
## 5600
## 5700
## 5800
## 5900
## 6000
## 6100
## 6200
## 6300
## 6400
## 6500
## 6600
## 6700
## 6800
## 6900
## 7000
## 7100
## 7200
## 7300
## 7400
## 7500
## 7600
## 7700
## 7800
## 7900
## 8000
## 8100
## 8200
## 8300
## 8400
## 8500
## 8600
## 8700
## 8800
## 8900
## 9000
## 9100
## 9200
## 9300
## 9400
## 9500
## 9600
## 9700
## 9800
## 9900
## 10000
## 10100
## 10200
## 10300
## 10400
## 10500
## 10600
## 10700
## 10800
## 10900
## 11000
## 11100
## 11200
## 11300
## 11400
## 11500
## 11600
## 11700
## 11800
## 11900
## 12000
## 12100
## 12200
## 12300
## 12400
## 12500
## 12600
## 12700
## 12800
## 12900
## 13000
## 13100
## 13200
## 13300
## 13400
## 13500
## 13600
## 13700
## 13800
## 13900
## 14000
```

```r
cat("SEVA time:",proc.time() - ptm,"\n")
```

```
## SEVA time: 2374.558 214.317 2593.342 0 0
```

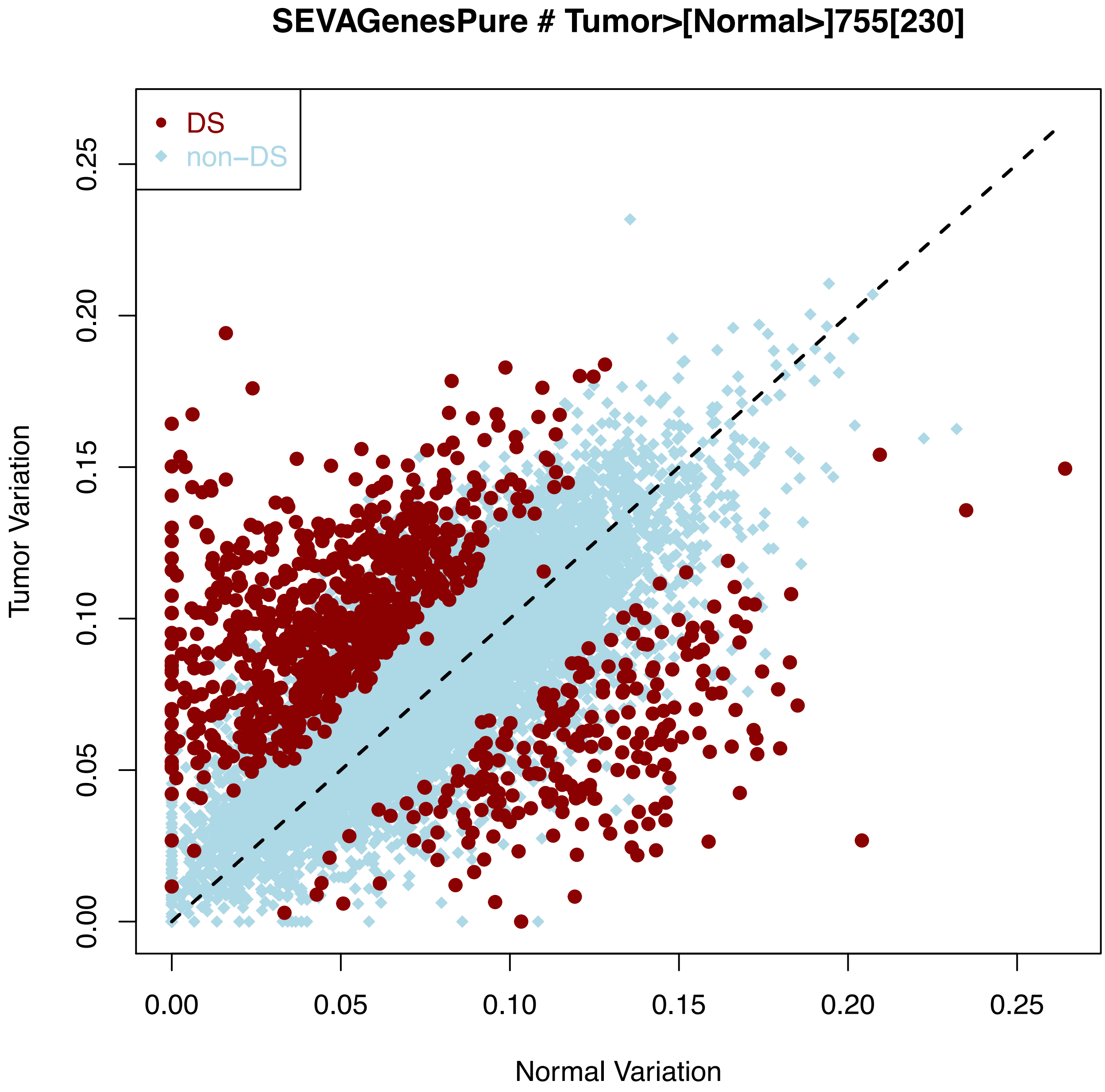
```r
#Doing offline
#load("../Cache/SEVAJoe.rda")

#SEVA pvalue
SEVApvaluePure <- sapply(junctionPValue,FUN =  function(x) x$pvalue)
#FDR adjusted
SEVAGenesPure <- names(which(p.adjust(SEVApvaluePure,"BH")<0.01))
#Bonferonni correction (Not used)
#names(which(SEVApvaluePure<0.01/length(junctionPValue)))
save(list=ls(),file = "../Cache/SEVAJoe.rda")
```

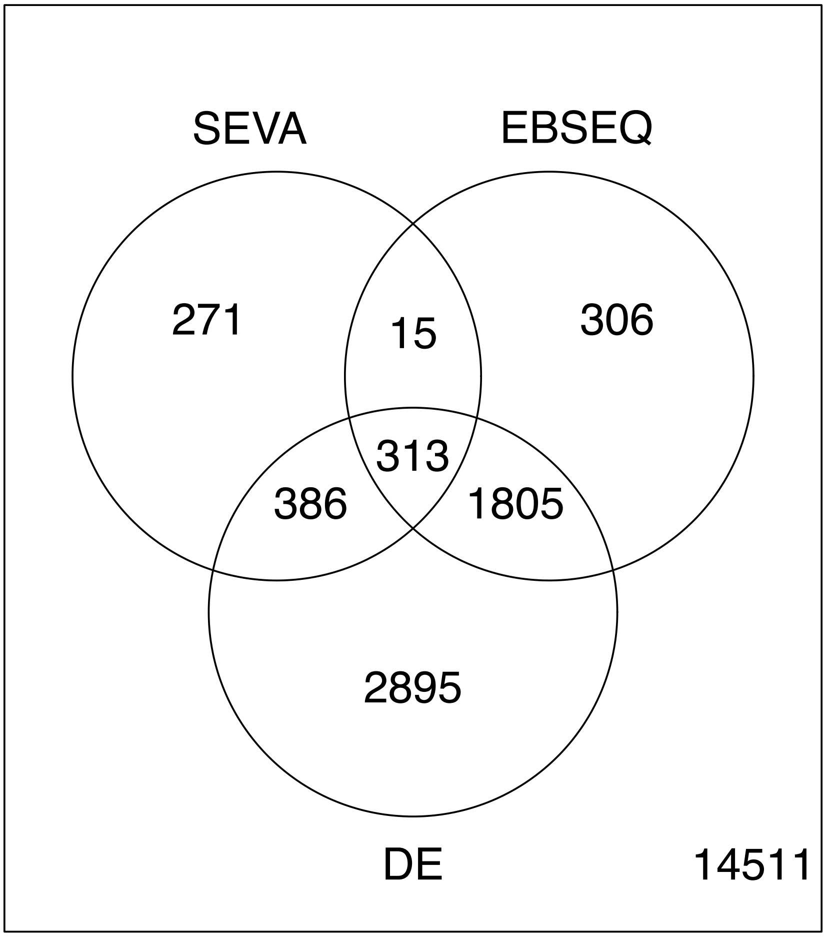
Plotting Figure 4:

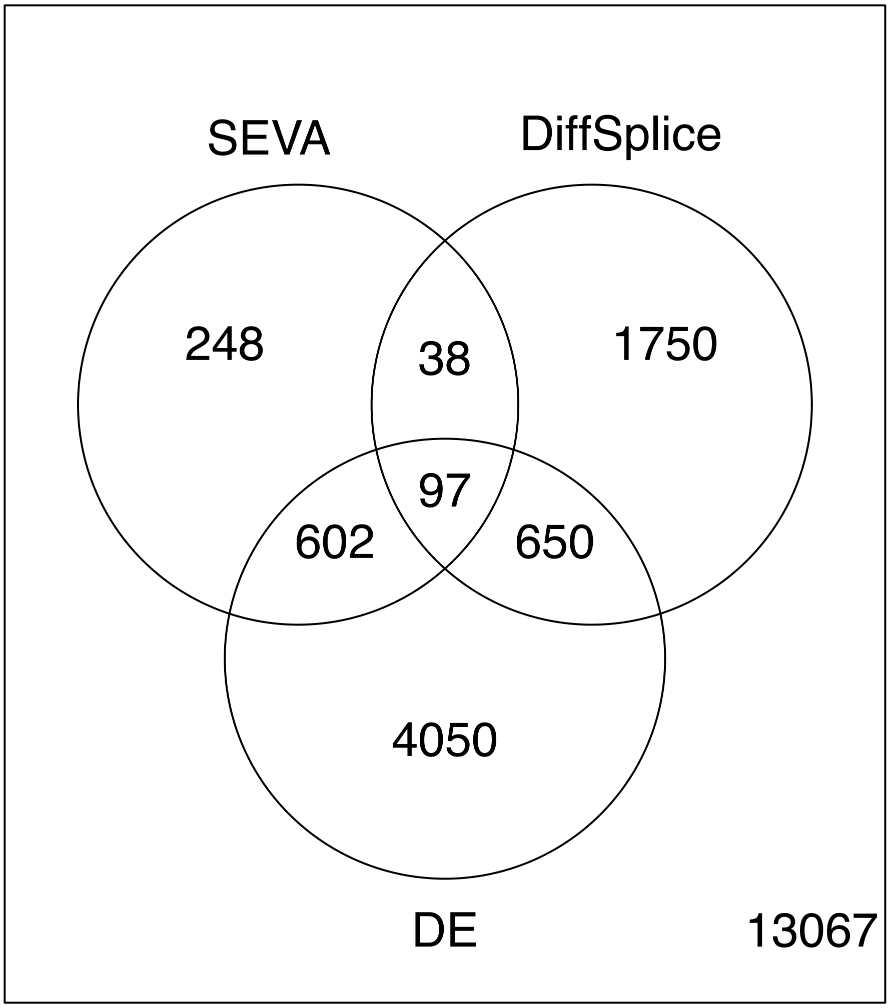
```r
#dispersions
E1 <- sapply(junctionPValue,FUN = function(x) x$E1)
E2 <- sapply(junctionPValue,FUN = function(x) x$E2)


### plot variation diagram and if VennMatrix is available it plots venn diagram as well
#Venn columns must be DE, EBSEQ, DiffSplice and SEVA
plotVariation <- function(ENormal,ETumor,
                          DSgenes,mainname = deparse(substitute(DSgenes)),
                          VennMatrix){
  #mainname with number of identified genes with higher variation in tumore and in cancer
  if(missing(VennMatrix)){
    AllGenes <- names(ENormal)
  }else{
    AllGenes <- intersect(names(ENormal),rownames(VennMatrix))
  }

  DSgenesIntersect <- intersect(AllGenes,DSgenes)

  mainename_variation_count <- paste0(mainname, " # Tumor>[Normal>]", #main naime
                                      sum(ETumor[DSgenesIntersect] > ENormal[DSgenesIntersect]), "[",# 
                                      sum(ETumor[DSgenesIntersect] < ENormal[DSgenesIntersect]), "]") #




  Erange <- range(c(ENormal,ETumor)) #range of variation
  plot(x=ENormal[setdiff(names(ENormal),DSgenesIntersect)], #plot non-DS
       y=ETumor[setdiff(names(ETumor),DSgenesIntersect)], main = mainename_variation_count,
       col="light blue", xlab = "Normal Variation", ylab="Tumor Variation",
       xlim = Erange,ylim = Erange,pch = 18)
  lines(x=ENormal[DSgenesIntersect],y=ETumor[DSgenesIntersect], #plot DS
        col="dark red",type = "p",pch = 19)
  lines(x=Erange,y=Erange,col="black",type = "l", lty = 2,lwd = 2) #45 degree line
  legend("topleft", legend = c("DS", "non-DS"),pch = c(20,18), #legend
         col = c("dark red","light blue"),
         text.col = c("dark red","light blue"))

  if(!missing(VennMatrix)){
    VennMatrixCopy <- VennMatrix
    VennMatrixCopy[,"SEVA"] <- 0
    VennMatrixCopy[DSgenesIntersect,"SEVA"] <- 1
    vennDiagram(VennMatrixCopy[,c("SEVA","EBSEQ","DE")])
    vennDiagram(VennMatrixCopy[,c("SEVA","DiffSplice","DE")])
  }

}



pdf(file = "Figure4.pdf")
plotVariation(ENormal = E1,ETumor = E2, DSgenes = SEVAGenesPure,VennMatrix = VennMatrix)
dev.off()
```

```
## pdf
##   2
```

```
plotVariation(ENormal = E1,ETumor = E2, DSgenes = SEVAGenesPure,VennMatrix = VennMatrix)
```


SEVAGenesPure # Tumor>[Normal>]755[230]
● DS
◆ non−DS
Tumor Variation
0.00
0.05
0.10
0.15
0.20
0.25
Normal Variation
0.00
0.05
0.10
0.15
0.20
0.25


SEVA
EBSEQ
271
15
306
313
386
1805
2895
DE
14511



Now, we generate Table 1, i.e. the six genes corrected p-values from the previous study:

```
### Genes to study from PLoS one paper
GenestoStudy <- c("VEGFC","DST","LAMA3","SDHA","TP63","RASIP1")
#EVA without correction
print("SEVA (without correction)")
```

```
## [1] "SEVA (without correction)"
```

```
print(sapply(SEVApvaluePure[GenestoStudy],FUN = function(x) x))
```

```
##        VEGFC          DST        LAMA3         SDHA         TP63
## 2.164749e-01 4.800871e-11 1.026985e-05 8.544950e-01 5.775838e-10
##       RASIP1
## 4.756504e-07
```

```
print("SEVA (with correction)")
```

```
## [1] "SEVA (with correction)"
```

```
#Bonferonni (old)
#print(sapply(SEVApvaluePure[GenestoStudy]*length(GenestoStudy),FUN = function(x) min(c(x,1))))
#FDR
print(p.adjust(SEVApvaluePure[GenestoStudy],"BH"))
```

```
##        VEGFC          DST        LAMA3         SDHA         TP63
## 2.597699e-01 2.880522e-10 1.540477e-05 8.544950e-01 1.732751e-09
##       RASIP1
## 9.513007e-07
```

```
#### FDR
### VEGFC          DST        LAMA3         SDHA         TP63       RASIP1
#### 2.597699e-01 2.880522e-10 1.540477e-05 8.544950e-01 1.732751e-09 9.513007e-07

#cat("Genes survive the 0.01 threshold (Bon-feronni corrected)",
#    names(which(SEVApvaluePure[GenestoStudy]<0.01/length(GenestoStudy))))

cat("Genes survive the 0.01 threshold (FDR corrected)",
    names(which(p.adjust(SEVApvaluePure[GenestoStudy],"BH")<0.01)))
```

```
## Genes survive the 0.01 threshold (FDR corrected) DST LAMA3 TP63 RASIP1
```

```
DSEBSeqOnlyGenesofStudy <- names(isos2genesvectsimplified)[which(isos2genesvectsimplified %in% GenestoS

ptm <- proc.time()
DSEBseq_outcome_GenesofStudy <- findDSEBSEQ(isos[DSEBSeqOnlyGenesofStudy,samplesIsos], ### DSEBSEQ
                               as.factor(phenoVect)[samplesIsos],
                               isos2genesvectsimplified = isos2genesvectsimplified)

cat("EBSEQ time:",proc.time() - ptm,"\n")
```

```
## EBSEQ time: 1.032 0.177 1.213 0 0
```

```
cat("Genes survive the 0.01 threshold (EBSeq)",
    names(which(DSEBseq_outcome_GenesofStudy$pvaluescorrected>0.99)))
```

```
## Genes survive the 0.01 threshold (EBSeq) VEGFC
```

```
print("EBSeq p-value")
```

```
## [1] "EBSeq p-value"
```

```
print(1-DSEBseq_outcome_GenesofStudy$pvaluescorrected)
```

```
##          DST        LAMA3       RASIP1         SDHA         TP63
## 2.665608e-01 4.636929e-01 1.279767e-01 1.847109e-01 1.084451e-01
##        VEGFC
## 2.664457e-06
```

```
### Free some memory
print("Genes of interest identified by DiffSplice: (TRUE found and FALSE not identified)")
```

```
## [1] "Genes of interest identified by DiffSplice: (TRUE found and FALSE not identified)"
```

```
print(GenestoStudy)
```

```
## [1] "VEGFC"  "DST"    "LAMA3"  "SDHA"   "TP63"   "RASIP1"
```

```
print(GenestoStudy %in% DiffSplicegenes)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
save(list=c("junctionPValue","SEVAGenesPure","VennMatrix"),file = "../Cache/ForTCGAAnalysis.rda")
```

4. Cross-study Validation with TCGA

Now, we cross-study the genes we identified in TCGA.

```
rm(list="junc.RPM")
gc()

TCGA.RSEM <- as.matrix(TCGA.RSEM)

junctionPValueTCGA <- SEVA.meangeneFilter(junc.RPM = junc.RPM.TCGA,
                                          phenoVect=phenoVect.TCGA,
                                          geneexpr=TCGA.RSEM,
                                          minmeanloggeneexp= 3,
                                          GenestoStudy = intersect(SEVAGenesPure,
                                                                   rownames(TCGA.RSEM)))

# junctionPValueTCGA <- GSReg.SEVA(juncExprs=junc.RPM.TCGA,
#                                 phenoVect=as.factor(phenoVect.TCGA),
#                                 verbose = F,
#                                 geneexpr=TCGA.RSEM,
#                                 minmeanloggeneexp= 3,
#                                       GenestoStudy = intersect(SEVAGenesPure,
#                                                                    rownames(TCGA.RSEM)))

#Only consider the genes both analyzed in TCGA and Joe Data
SEVATCGAGenes <- intersect(names(junctionPValueTCGA),
                                      SEVAGenesPure)

tcgapval <- sapply(junctionPValueTCGA[SEVATCGAGenes],function(x) x$pvalue)

### Bonferonni calls (old)
#cat("percentage that survived",
#    mean(tcgapval[SEVATCGAGenes] <0.01/length(tcgapval)))

### FDR calls
cat("percentage that survived",
    mean(p.adjust(tcgapval[SEVATCGAGenes],"BH") <0.01))




hist(x = tcgapval,
     xlab="P-Value",main="P-Value from TCGA for SEVA Identified")




cat("Quatile of the p-value distribution SEVA genes using TCGA data")
print(quantile(tcgapval))
```

Now, checking a random set genes.

```
originaldatapval <- sapply(junctionPValue,function(x) x$pvalue)

plot(originaldatapval[names(tcgapval)],
     tcgapval,
     ylab="Based on original data",
     xlab= "Based on TCGA",
     main="Cross-study P-Values")


hist(x = originaldatapval,
     xlab="P-Value",main="P-Value from original data")






set.seed(1)
randomgenes <- sample(names(junctionPValue),size = length(SEVATCGAGenes))



junctionPValueRandom <- SEVA.meangeneFilter(junc.RPM = junc.RPM.TCGA,
                                          phenoVect=phenoVect.TCGA,
                                          geneexpr=TCGA.RSEM,
                                          minmeanloggeneexp= 3,
                                          GenestoStudy = randomgenes)



# junctionPValueRandom <- SEVA.meangeneFilter(juncExprs=junc.RPM.TCGA,
#                                           phenoVect=as.factor(phenoVect.TCGA),
#                                           geneexpr=TCGA.RSEM,
#                                           verbose = F,
#                                           minmeanloggeneexp= 3,
#                                           GenestoStudy = randomgenes)
randompvalue <- sapply(junctionPValueRandom,FUN = function(x) x$pvalue)
cat("Quatile of the p-value distribution random genes using TCGA data")
print(quantile(randompvalue))

print(wilcox.test(x=tcgapval,y=randompvalue,alternative = "less"))

z <- c(tcgapval,randompvalue)
print(cor.test(z,originaldatapval[names(z)],method = "spearman"))

save(list=ls(),file = "C:/Users/bahman/Dropbox/SEVApaper/PaperSuppl/Cache/SEVATCGA.rda")
```

3. Generating Figure 3

First, we load aligned simulated isofrom, junction, gene expression data. Since simulating requires more computational power than a laptop.

```
### plotting a data figures.
load("../Results/Simulation/FourthTryFeb5/VennInf_functionR.rda")
source("../Scripts/functions.R")
#loading data
load("../Results/Simulation/SecondTryJan13/juncRPMExp.Rdata")
#loading groundtruth
load("../Results/Simulation/SecondTryJan13/groundtruthSimplified.rda")
#isos 2 gene names
load("../Results/Simulation/SecondTryJan13/isos2genesvect.rda")
#load the percentages
```

```r
load("../Results/Simulation/PercentageData/PercentageDataFinal.rda")

myisoforms <- names(which(isos2genesvectsimplified == names(neutralgenes)[3] ))[1:4]
```

No, we choose one of the genes and apply the processes in the simulation parts.

```r
PerturbedNum <- 15

neutralmat <- log2(isoexprext[myisoforms,1:50]+1)
pdf(file = "Neutral.pdf",width = 7,height=7)
matplot(t(neutralmat), main="Neutral (Not affected gene)",
        pch = c(10,11,12,13),lty=2,lwd = 1, type= "b",xaxt= "n",
        xlab = "Samples",
        ylab = "isoform expression (log2)")
lines(x=c(25.5,25.5),y=c(0,10),type = "l",lty =5, lwd =3)
#lines(x=c(50.5-PerturbedNum,50.5-PerturbedNum),y=c(0,10),type = "l",lty =3, lwd =1)
axis(side = 1,at = c(15,35),labels = c("Normal", "Cancer"))
dev.off()
```

```
## pdf
##   2
```

```r
pdf(file = "DS.pdf",width = 7,height=7)
DSmat <- neutralmat
MyPermutation <- c(3,4,2,1)
DSmat[,-(1:(ncol(DSmat)-PerturbedNum))] <- DSmat[MyPermutation ,
                                                 -(1:(ncol(DSmat)-PerturbedNum))]

matplot(t(DSmat), main="Differentially  Spliced (DS) gene",
        pch = c(10,11,12,13),lty=2,lwd = 1, type= "b",xaxt= "n",
        xlab = "Samples",
        ylab = "isoform expression (log2)")
lines(x=c(25.5,25.5),y=c(0,10),type = "l",lty =5, lwd =3)
lines(x=c(50.5-PerturbedNum,50.5-PerturbedNum),y=c(0,10),type = "l",lty =6, lwd =1)
axis(side = 1,at = c(15,30, 43),labels = c("Normal", "non-disrupted\n Cancer","disrupted\n Cancer"))
dev.off()
```

```
## pdf
##   2
```

```r
DEmat <- neutralmat
DEmat[,-(1:(ncol(DSmat)-PerturbedNum))] <- DEmat[,-(1:(ncol(DSmat)-PerturbedNum))]+1

pdf(file = "DEonly.pdf",width = 7,height=7)
matplot(t(DEmat), main= "Differentially Expressed (DE) gene",
        pch = c(10,11,12,13),lty=2,lwd = 1, type= "b",xaxt= "n",
        xlab = "Samples",
        ylab = "isoform expression (log2)")
lines(x=c(25.5,25.5),y=c(0,10),type = "l",lty =5, lwd =3)
lines(x=c(50.5-PerturbedNum,50.5-PerturbedNum),y=c(0,10),type = "l",lty =6, lwd =1)
axis(side = 1,at = c(15,30, 43),labels = c("Normal", "non-disrupted\n Cancer","disrupted\n Cancer"))
dev.off()
```

```
## pdf
##   2
```

```
DEDSmat <- DSmat
DEDSmat[,-(1:(ncol(DSmat)-PerturbedNum))] <- DEDSmat[ ,
                                                 -(1:(ncol(DSmat)-PerturbedNum))]+1
pdf(file = "DS-DE.pdf",width = 7,height=7)
matplot(t(DEDSmat), main =" DS-DE gene",
        pch = c(10,11,12,13),lty=2,lwd = 1, type= "b",xaxt= "n",
        xlab = "Samples",
        ylab = "isoform expression (log2)")
lines(x=c(25.5,25.5),y=c(0,10),type = "l",lty =5, lwd =3)
lines(x=c(50.5-PerturbedNum,50.5-PerturbedNum),y=c(0,10),type = "l",lty =6, lwd =1)
axis(side = 1,at = c(15,30, 43),labels = c("Normal", "non-disrupted\n Cancer","disrupted\n Cancer"))
dev.off()
```

```
## pdf
##   2
```

Now, we generate the last two figures 3. First, we load the ground truth:

```
#### gene type
DEDSGenes <- names(DEDS)
DEnonDSGenes <- names(DEnonDS)
nonDEDSGenes <- names(nonDEDS)
neutralgenes <- names(neutralgenes)
### DSgenes
DSGenes <- union(DEDSGenes, nonDEDSGenes)
#DEGenes
DEGenes <- union(DEDSGenes, DEnonDSGenes)
OnlyGenesGroundTruth <- union(union(DEDSGenes,nonDEDSGenes),union(DEnonDSGenes,neutralgenes))
```

We generate labels for simulated data:

```
#PHENOTYPES
phenotypes <- as.numeric(sapply(strsplit(colnames(junc.RPM),split = "_"),function(x) x[2]))<=25

names(phenotypes) <- colnames(junc.RPM)

#Tumor and Normal Sample names
TumorSamples <- names(which(phenotypes==TRUE))
NormalSamples <- names(which(phenotypes==FALSE))



#Median of median expression
medT <- log2(apply(X = geneexpr[,TumorSamples],MARGIN = 1, FUN = median)+1)
medN <- log2(apply(X = geneexpr[,NormalSamples],MARGIN = 1, FUN = median)+1)
```

Preprocessing of the simulated data:

```
#Filtering genes
#genes_withfoldchange <- names(which(abs(medT-medN)>1))

#GeneMat.small <- geneExp[genes_withfoldchange,]
genesCHR1 <- names(which(apply(geneexpr,MARGIN = 1,sum)>0))
genesChr1 <- sapply(strsplit(genesCHR1,split = "[|]"),function(x) x[1])



DEGenes_EBesq_outcome <- findDEGenes(geneexp = geneexpr,phenoVect=as.factor(phenotypes))
```

```
## Removing transcripts with 100 th quantile < = 0
## 655 transcripts will be tested
```

```
DEGenes_EBesq <- sapply(strsplit(DEGenes_EBesq_outcome$DEGenes_EBesq,"[|]"),FUN = function(x) x[1])


isos2genesvectsimplified <- sapply(strsplit(isos2genesvect,split = '[|]'),FUN = function(x) x[1])
```

EBSeq analysis for simulated data:

```
DSEBseq_outcome <- findDSEBSEQ(isoexpr, ### DSEBSEQ
                               as.factor(phenotypes),
                               isos2genesvectsimplified = isos2genesvectsimplified)
DSEBseq <- intersect(DSEBseq_outcome$DSEBseq,genesChr1)
```

SEVA analysis for simulated data:

```
# ###SEVA #################################
#
# #geneexpr <- as.matrix(geneexp)
# ### simplify gene expression names and remove duplicated genes
# exprsimiplifiednames <- sapply(strsplit(rownames(geneexpr),split = "[|]"),FUN = function(x) x[1])
# notduplicatedgenes <- which(!duplicated(exprsimiplifiednames))#not duplicatred genes
# geneexp <- geneexpr[notduplicatedgenes,]#removing duplicated genes
# rownames(geneexp)<- exprsimiplifiednames[notduplicatedgenes]
#
#
# #SEVA pvalue calculation
# #junctionPValue <- SEVA.meangeneFilter(juncExprs=junc.RPM,phenoVect=as.factor(phenotypes),
# #                                geneexpr=geneexp,minmeanloggeneexp= 0)
#
# junctionPValue <- SEVA.meangeneFilter(junc.RPM = junc.RPM,
#                              phenoVect=phenotypes,
#                              geneexpr=geneexp,
#                              minmeanloggeneexp= 0)
#
#
# SEVA <- names(which(p.adjust(sapply(junctionPValue,function(x) x$pvalue),"BH")<0.01))
#
#
# # SEVA <- names(which((apply(rbind(sapply(junctionPValue,function(x) x$pvalue),
# #                                    sapply(junctionPValue,function(x) x$pvalueD12D1),
# #                                    sapply(junctionPValue,function(x) x$pvalueD12D2)),MARGIN = 2,min)
#
# VennDiag <- matrix(0,nrow = length(genesChr1),ncol = 5,
#                    dimnames = list(genesChr1,list("DE","DS","EBSEQ","SEVA","DiffSplice")))
# #DE genes
# VennDiag[DEGenes,"DE"] <- 1
# #DS genes
# VennDiag[DSGenes,"DS"] <- 1
#
# #DE genes
# #VennDiag[DEGenes_EBesq,"EBSEQ"] <- 1
# #DS genes
# VennDiag[DSEBseq,"EBSEQ"] <- 1
#
```

```
#
# #OnlyGenesGroundTruth <- c(names(neutralgenes),names(DEnonDS),names(DEDS),names(nonDEDS))
# vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DE","DS","EBSEQ")])
# vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DS","EBSEQ")])
# vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DE","DS","EBSEQ")])
# #vennDiagram(VennDiag)
#
#
#
#
# VennDiag[intersect(SEVA,genesChr1),"SEVA"] <-  1
# vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DE","DS","SEVA")])
#
# vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DE","DS","EBSEQ")])

#Venn4Percentage <- vector(mode = "list",length = length(experimentSamplesTumors))
print("Skipped")
```

```
## [1] "Skipped"
```

Applying DiffSplice requires a lot of time and recources. So, we applied them offline to the simulated data.

```
diffspliceFiles <- dir("../Results/Simulation/DiffSplice/")

#ebseqpvalueAll <- vector(mode = "list",length = length(experimentSamplesTumors))
ebseqpvalueGenesAll <- vector(mode = "list",length = length(experimentSamplesTumors))
```

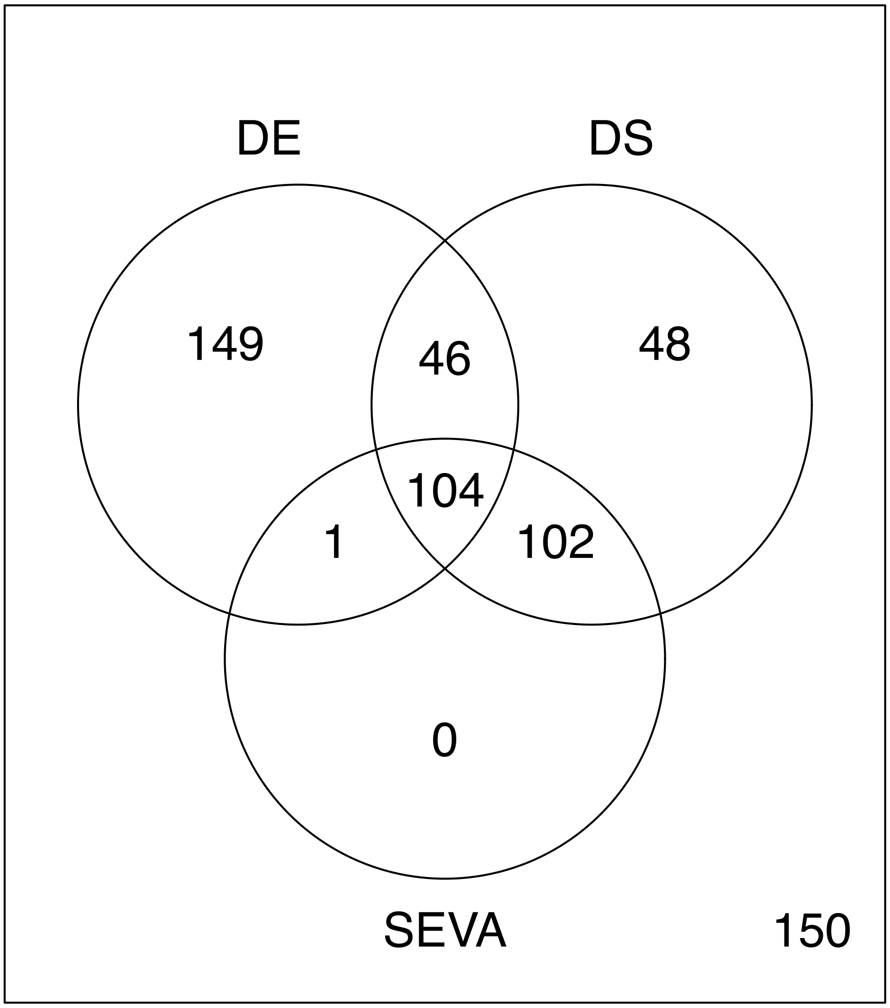
We apply with all three to different number of disrupted samples.

```
for( i in seq_along(experimentSamplesTumors)){
  #current samples: Normals as nomral with a mixture of normal and cancerous as the cancer samples
  samplescur <- c(NormalLabels,experimentSamplesTumors[[i]])
  #phenotype
  phenotypescur <- sapply(strsplit(samplescur,split = "_"),FUN = function(x) x[3])
  names(phenotypescur) <- samplescur[names(phenotypescur)]
#   junctionPValue <- GSReg.GeneSets.EVA(geneexpres = junc.RPMext[,samplescur],
#                                         phenotypes = as.factor(phenotypescur),
#                                         minGeneNum = 2,
#                                         pathways = genesJunction[intersect(names(which(sapply(genesJu
#                                         distFunc = GSReg.kendall.tau.distance.Restricted,
#                                         distparamPathways = MyRest  )

  junctionPValue <- SEVA.meangeneFilter(junc.RPM = junc.RPMext[,samplescur],
                                        phenoVect=as.factor(phenotypescur),
                                        geneexpr=geneexp,
                                        minmeanloggeneexp= 0)

   # junctionPValue <- GSReg.SEVA(juncExprs=junc.RPMext[,samplescur],
   #                                      phenoVect=as.factor(phenotypescur),
   #                                      verbose = F,
   #                                      geneexpr=geneexp,minmeanloggeneexp= 0)

  SEVA <- names(which(p.adjust(sapply(junctionPValue,function(x) x$pvalue),"BH")<0.01))
  zscoresSEVA <- sapply(junctionPValue,FUN = function(x) abs(x$zscore))

#  zscoresSEVA <- sapply(junctionPValue,FUN = function(x) max(abs(c(x$zscore,x$zscoreD12D1,x$zscoreD12D

  DSEBseq_outcome <- findDSEBSEQ(isoexprext[,samplescur],
                                 isos2genesvectsimplified,
                                 phenoVect =as.factor(phenotypescur) )

  DSEBseq <- DSEBseq_outcome$DSEBseq


  ebseqpvalueGenes <- DSEBseq_outcome$pvaluescorrected
  ebseqpvalueGenesAll[[i]] <- ebseqpvalueGenes







  VennDiag <- matrix(0,nrow = length(OnlyGenesGroundTruth),ncol = 5,
                     dimnames = list(OnlyGenesGroundTruth,c("DE","DS","EBSEQ","SEVA","DiffSplice")))

  #DE genes
  VennDiag[DEGenes,"DE"] <- 1
  #DS genes
  VennDiag[DSGenes,"DS"] <- 1
  #DSStatus[DSGenes] <- 1

  VennDiag[intersect(SEVA,OnlyGenesGroundTruth),"SEVA"] <-  1
  VennDiag[intersect(OnlyGenesGroundTruth,DSEBseq),"EBSEQ"] <- 1



  ###Diffsplice Results
  tumorsampleNum <- strsplit(x = names(experimentSamplesTumors)[i], split = " perturbed samples")[[1]][
  transDiffSplice <- read.delim(
    paste("../Results/Simulation/DiffSplice/ResultsSim",
          tumorsampleNum,"/differential_transcription.txt",sep = ""))

  signtransDiffSplice <- which(transDiffSplice[,"significant"]=="yes")

  transDiffSpliceGRanges <- GRanges(seqnames = transDiffSplice[signtransDiffSplice,"chromosome"],
                                    ranges = IRanges(start = transDiffSplice[signtransDiffSplice,"positi
                                                     end = transDiffSplice[signtransDiffSplice,"position

  overlapDiffSplice <- findOverlaps(transDiffSpliceGRanges,gn)

# VennDiag[,"DiffSplice"] <- 0
  VennDiag[intersect(rownames(VennDiag),
                     unique(na.omit(gn$SYMBOL[subjectHits(overlapDiffSplice)]))),
         "DiffSplice"] <- 1
```

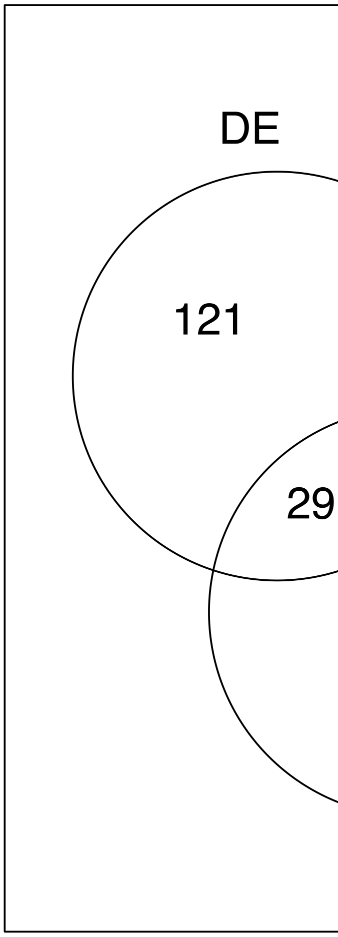
```r
Venn4Percentage[[i]] <- VennDiag
vennDiagram(VennDiag[,c("DE","DS","SEVA")])
title(paste(names(experimentSamplesTumors)[i],
            "\nNull: DE and SEVA are independent\n (p-value ",
          signif(fisher.test(VennDiag[,"DE"],
                             VennDiag[,"SEVA"])$"p.value",
                 digits = 2),")"))

vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DE","DS","EBSEQ")])

title(paste(names(experimentSamplesTumors)[i],
            "\nNull: DE and EBSEQ are independent\n (p-value ",
          signif(fisher.test(VennDiag[OnlyGenesGroundTruth,"DE"],
                             VennDiag[OnlyGenesGroundTruth,"EBSEQ"])$"p.value",
                 digits = 2),")"))





vennDiagram(VennDiag[OnlyGenesGroundTruth,c("DE","DS","DiffSplice")])
title(paste(names(experimentSamplesTumors)[i],
            "\nNull: DE and DiffSplice are independent\n (p-value ",
          signif(fisher.test(VennDiag[OnlyGenesGroundTruth,"DE"],
                             VennDiag[OnlyGenesGroundTruth,"DiffSplice"])$"p.value",
                 digits = 2),")"))




#required for precision recall curve
DSStatus <- vector(mode = "numeric",length = length(OnlyGenesGroundTruth))
names(DSStatus) <- OnlyGenesGroundTruth
DSStatus[DSGenes] <- 1



DSStatusofDE <- vector(mode = "numeric",length = length(DEGenes))
names(DSStatusofDE) <- DEGenes
DSStatusofDE[DEDSGenes] <- 1


diffcurves <- list(precrec = c("prec","rec","bottomleft"),
         tfpr = c("rec","fpr","bottomright"),
         senspec = c("rec","tnr","bottomleft"))#different types of curves
for( j in seq_along(diffcurves)){

  myz <- vector(mode = "numeric",length = length(DSStatus))
  names(myz) <- names(DSStatus)
  myz[intersect(names(DSStatus),names(zscoresSEVA))] <- zscoresSEVA[intersect(names(DSStatus),names(zs

  pred1 <- prediction( myz, DSStatus)
  perf1 <- performance(pred1, diffcurves[[j]][1], diffcurves[[j]][2])
```

```
    plot(perf1, lty =1, col="dark red")

    myz <- vector(mode = "numeric",length = length(DSStatusofDE))
    names(myz) <- names(DSStatusofDE)
    myz[intersect(names(DSStatusofDE),names(zscoresSEVA))] <- zscoresSEVA[intersect(names(DSStatusofDE)


    pred2 <- prediction( myz, DSStatusofDE)
    perf2 <- performance(pred2,  diffcurves[[j]][1], diffcurves[[j]][2])
    lines(perf2@x.values[[1]],perf2@y.values[[1]], lty =2, col="dark red")


    myz <- vector(mode = "numeric",length = length(DSStatus))
    names(myz) <- names(DSStatus)
    myz[intersect(names(DSStatus),names(ebseqpvalueGenes))] <- ebseqpvalueGenes[intersect(names(DSStatu


    pred3 <- prediction( myz, DSStatus)
    perf3 <- performance(pred3,  diffcurves[[j]][1], diffcurves[[j]][2])
    lines(perf3@x.values[[1]],perf3@y.values[[1]], lty =1, col="blue")

    myz <- vector(mode = "numeric",length = length(DSStatusofDE))
    names(myz) <- names(DSStatusofDE)
    myz[intersect(names(DSStatusofDE),names(ebseqpvalueGenes))] <- ebseqpvalueGenes[intersect(names(DSS



    pred4 <- prediction( myz, DSStatusofDE)
    perf4 <- performance(pred4,  diffcurves[[j]][1], diffcurves[[j]][2])
    lines(perf4@x.values[[1]],perf4@y.values[[1]], lty =2, col="blue")

    legend(diffcurves[[j]][3],legend = c("SEVA","EBSEQ","SEVA DE","EBSEQ DE"),
           col=c("dark red","blue","dark red","blue"),
           text.col = c("dark red","blue","dark red","blue"),
           lty = c(1,1,2,2)     )



    }









}
```

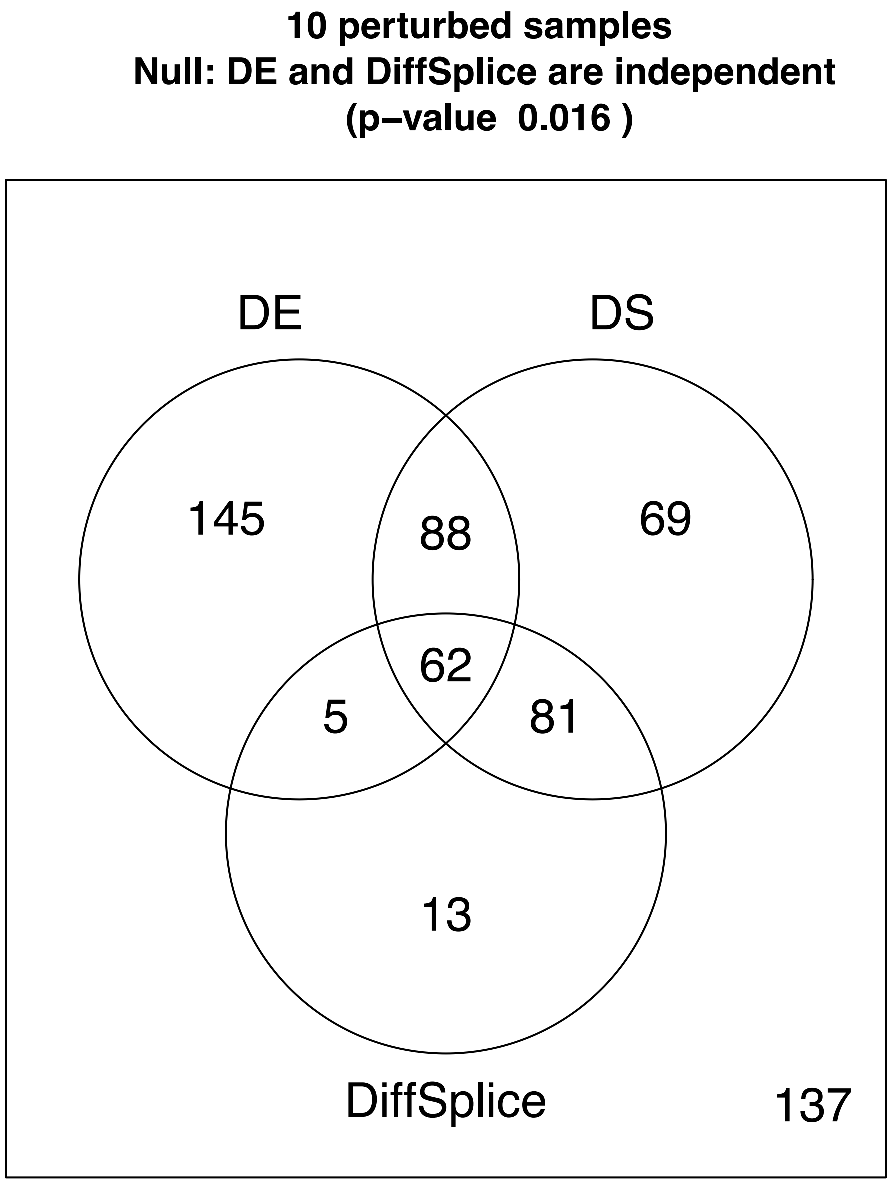
```
## 100
## 200
## 300
## 400
## 500
```

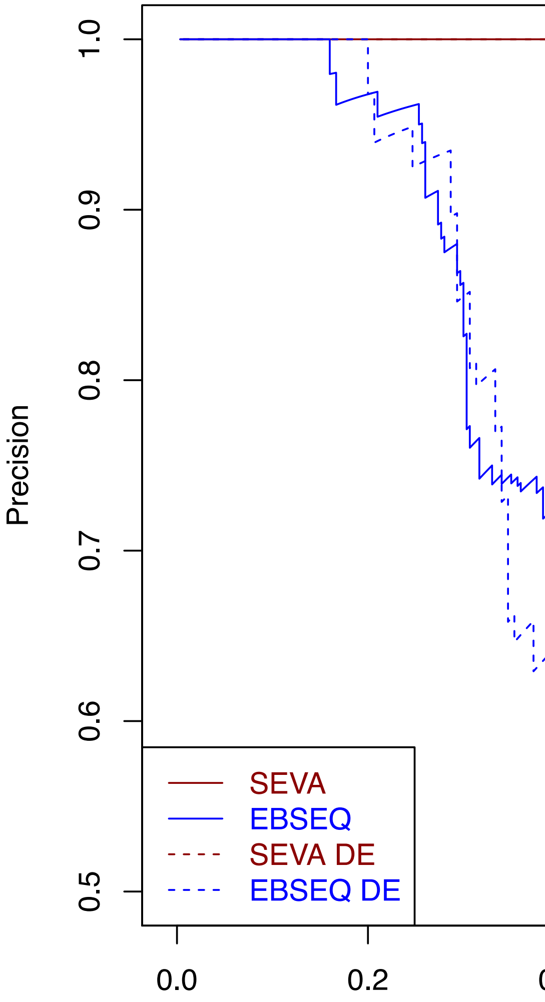


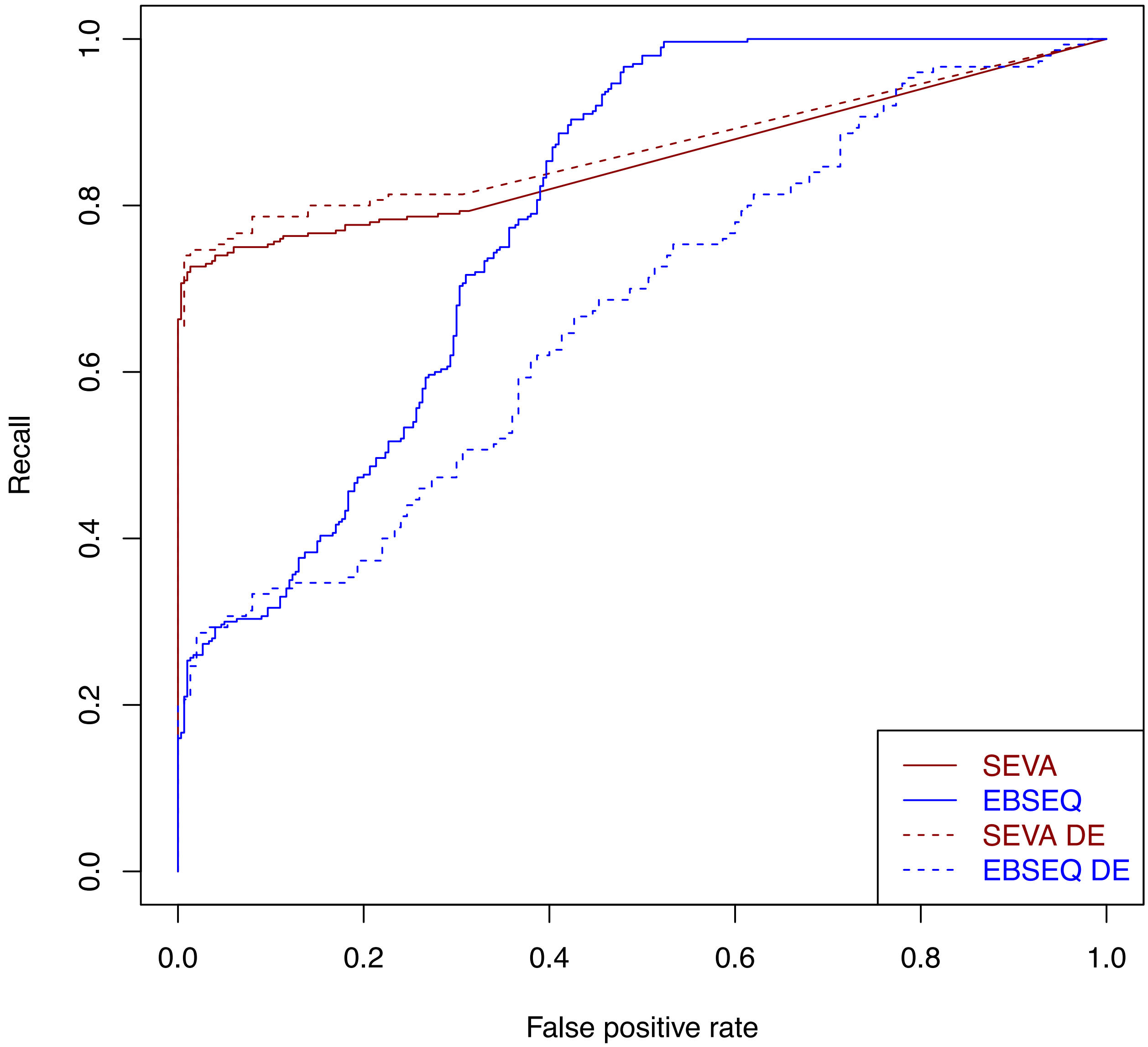
10 perturbed samples
Null: DE and SEVA are independent
(p–value 0.86)



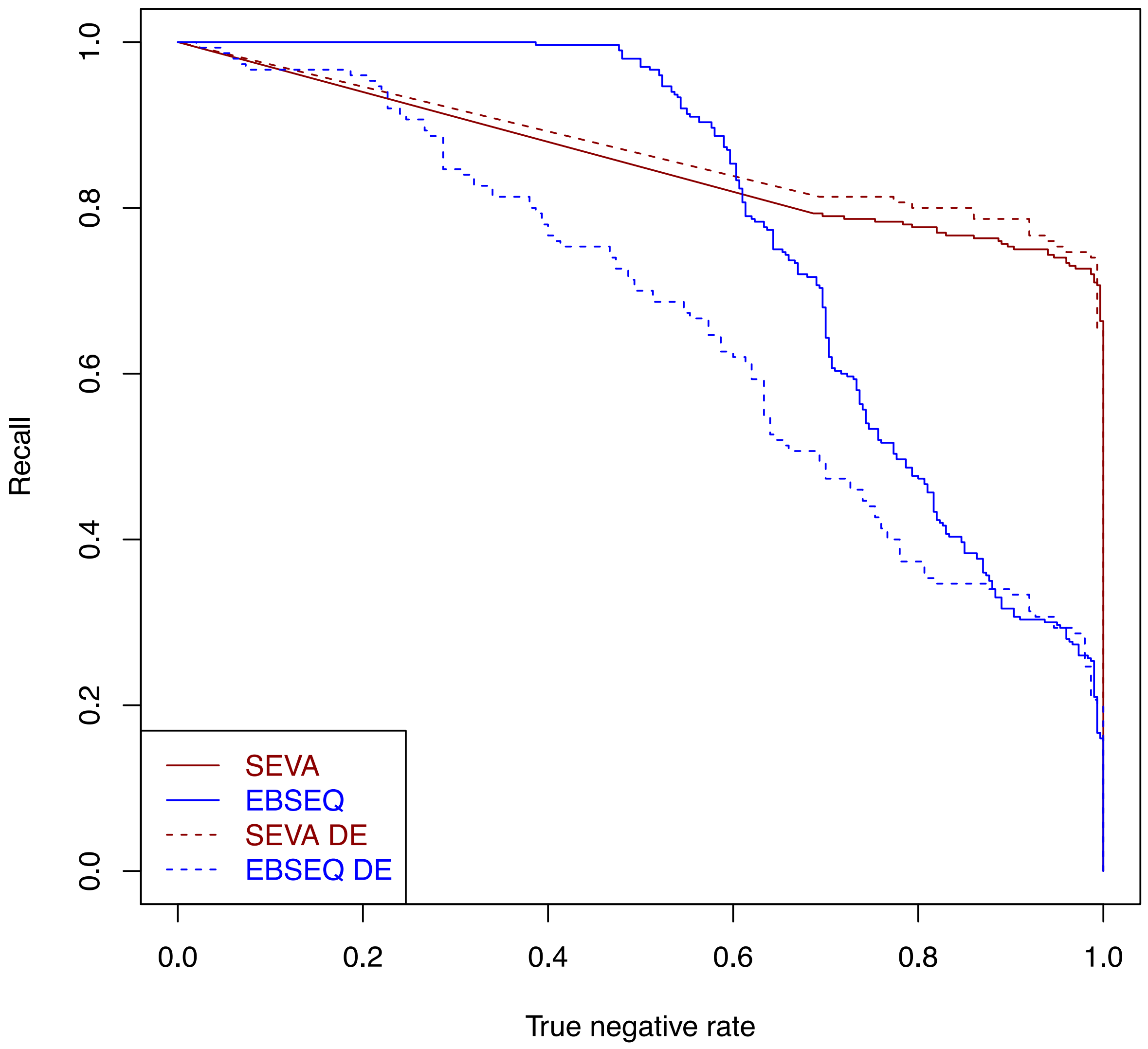
10 pe
Null: DE and
(p–


10 perturbed samples
Null: DE and DiffSplice are independent
(p–value 0.016)
DE
DS
DiffSplice
145
88
69
62
5
81
13
137


Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
SEVA
EBSEQ
SEVA DE
EBSEQ DE

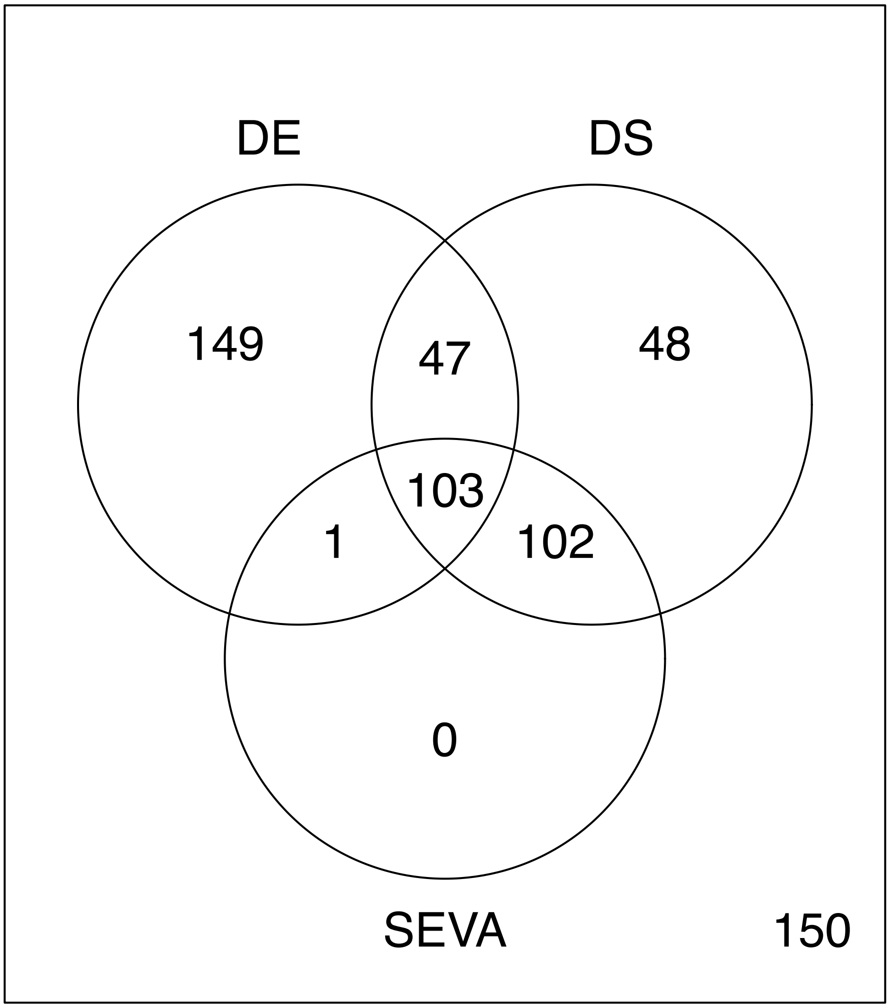


1.0
0.8
0.6
0.4
0.2
0.0
Recall
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
SEVA
EBSEQ
SEVA DE
EBSEQ DE

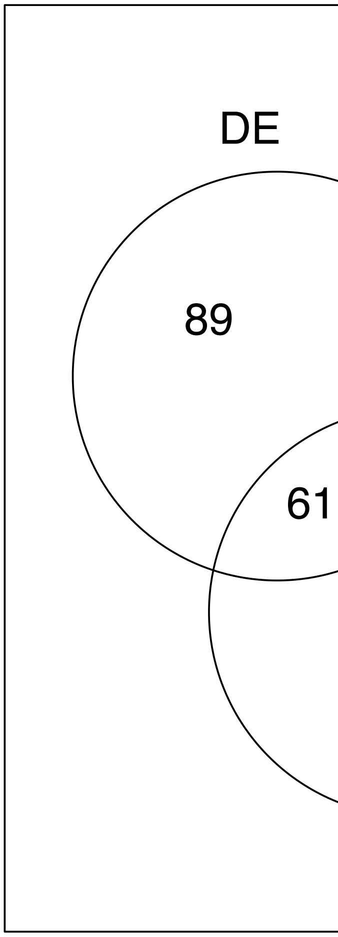


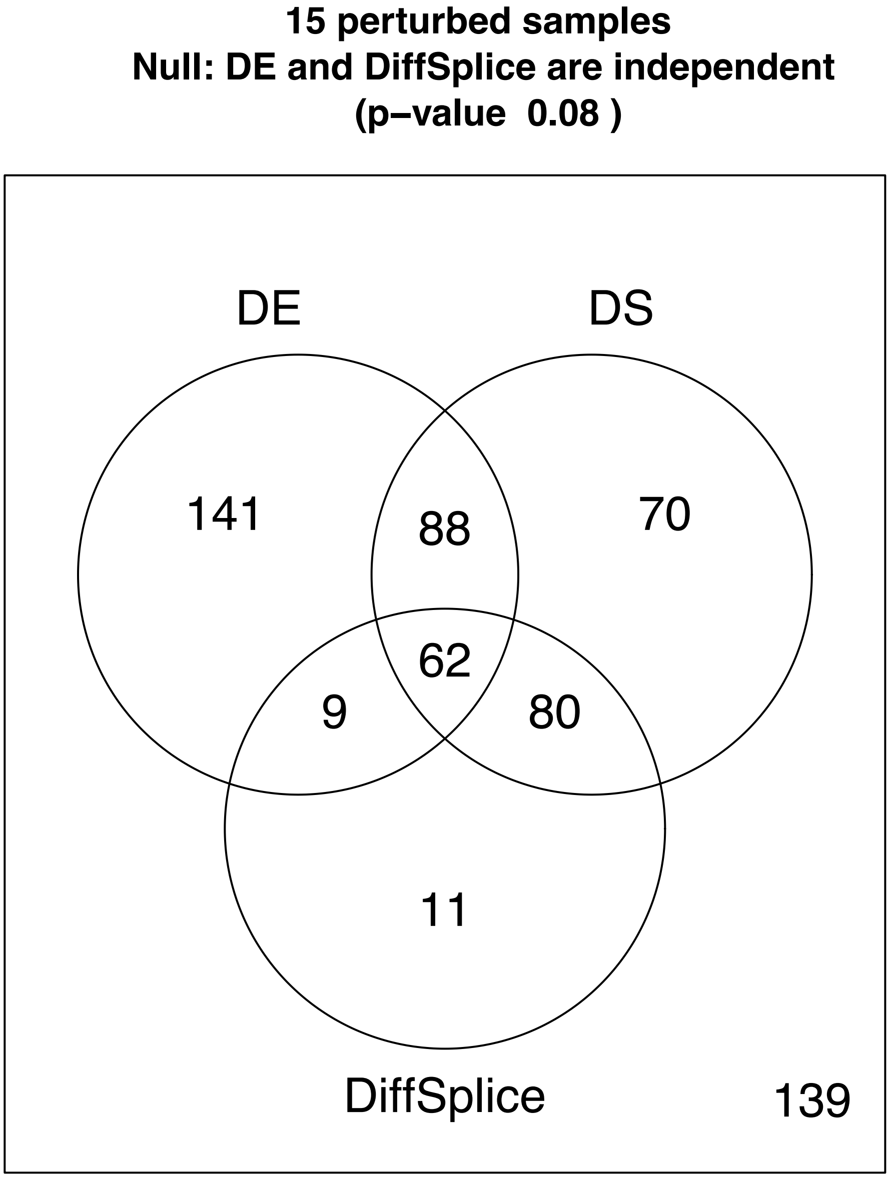
```
## 100
## 200
## 300
## 400
## 500
```

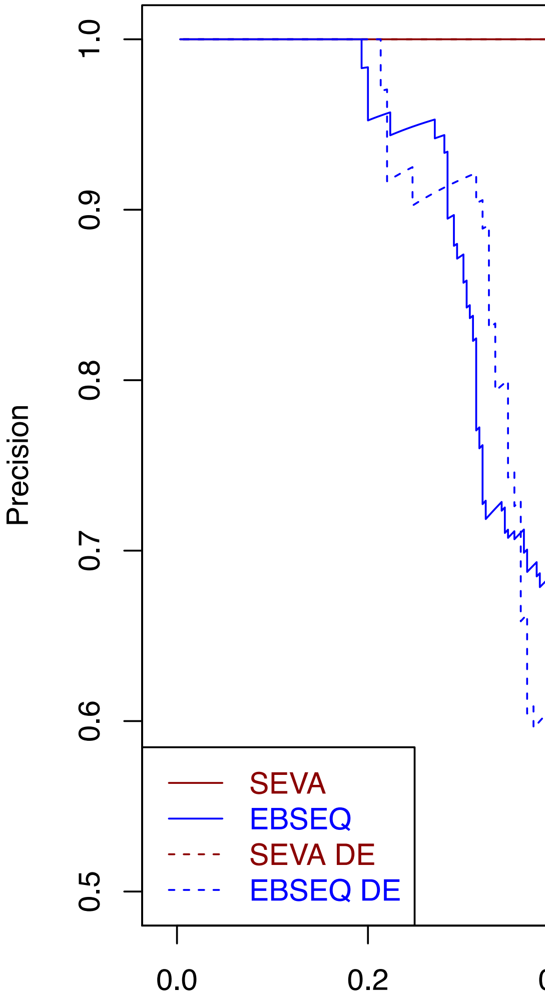
15 perturbed samples
Null: DE and SEVA are independent
(p–value 0.93)

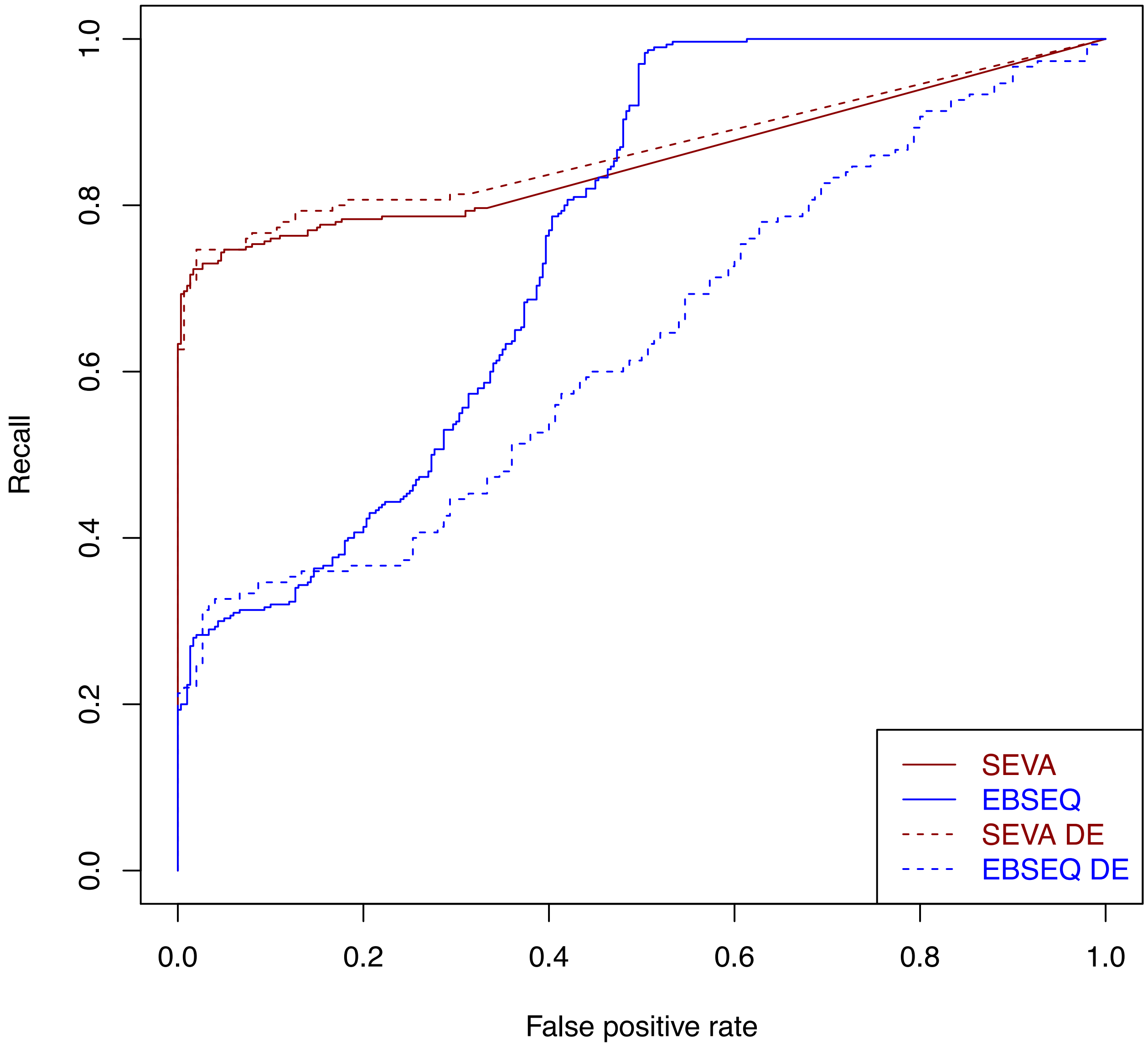


15 pe
Null: DE and
(p-

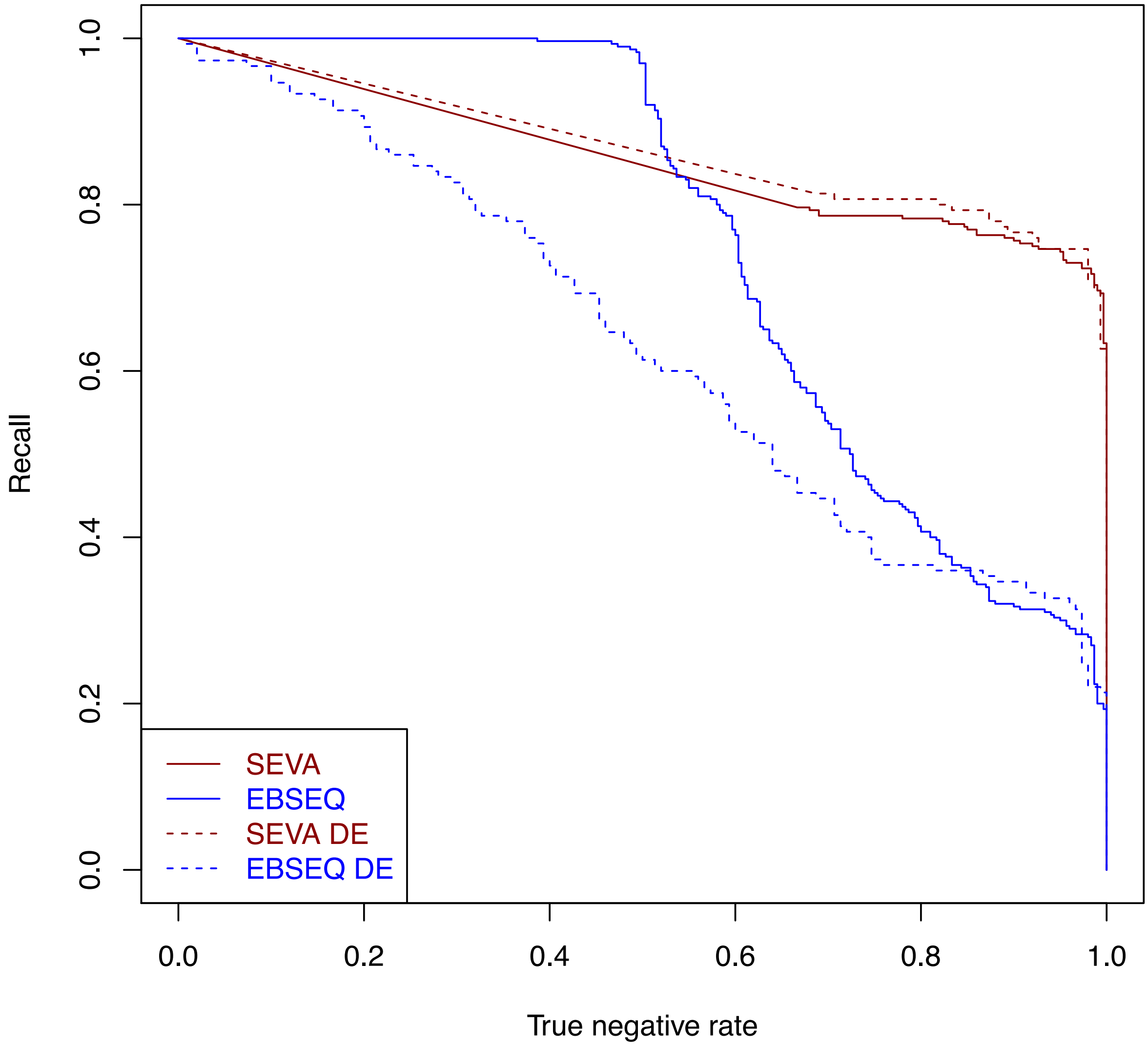



15 perturbed samples
Null: DE and DiffSplice are independent
(p−value 0.08)
DE
DS
DiffSplice
141
88
70
62
9
80
11
139

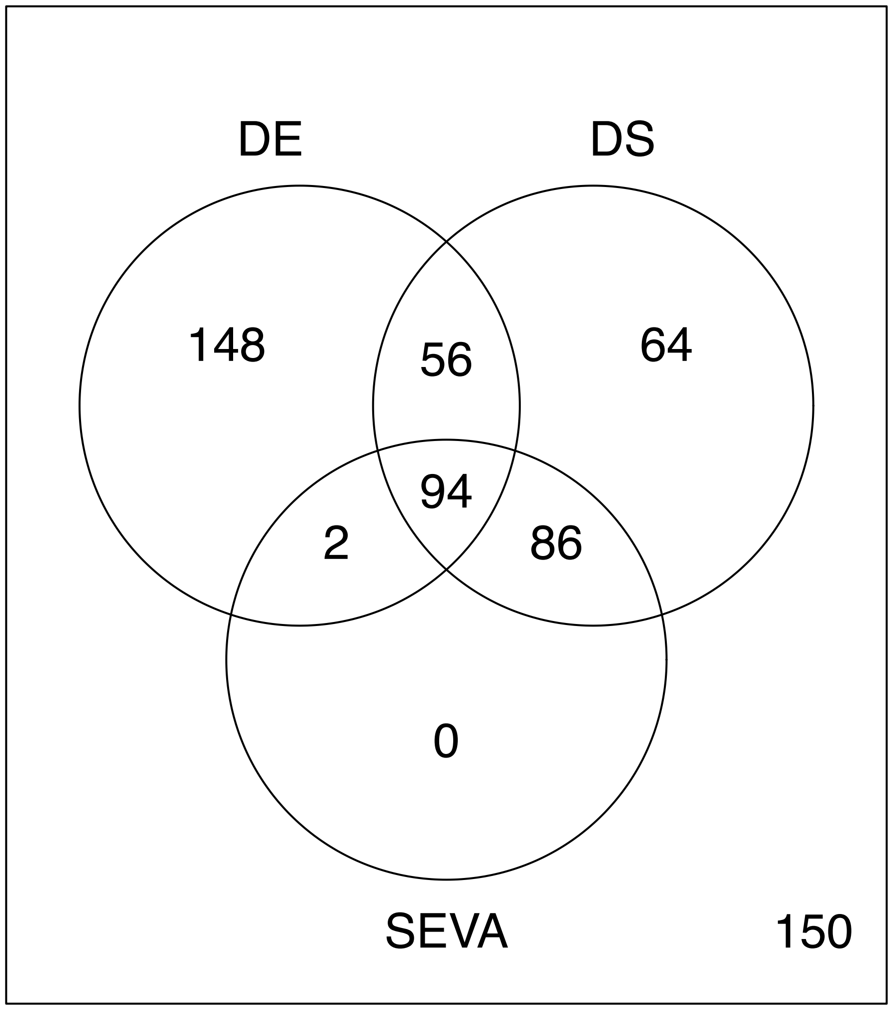

Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
SEVA
EBSEQ
SEVA DE
EBSEQ DE



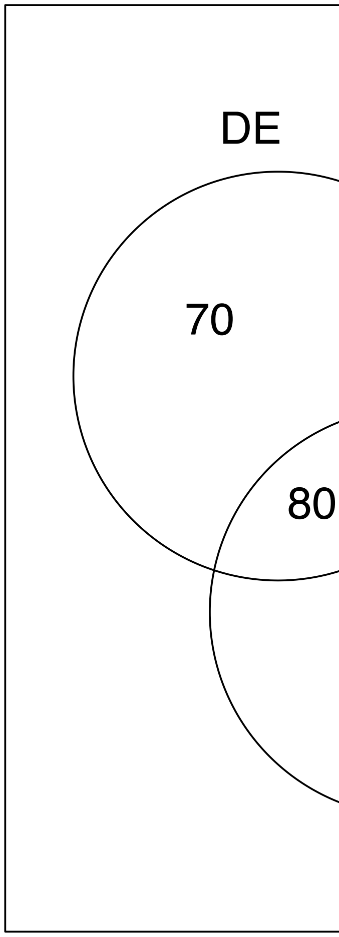
Recall
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
SEVA
EBSEQ
SEVA DE
EBSEQ DE



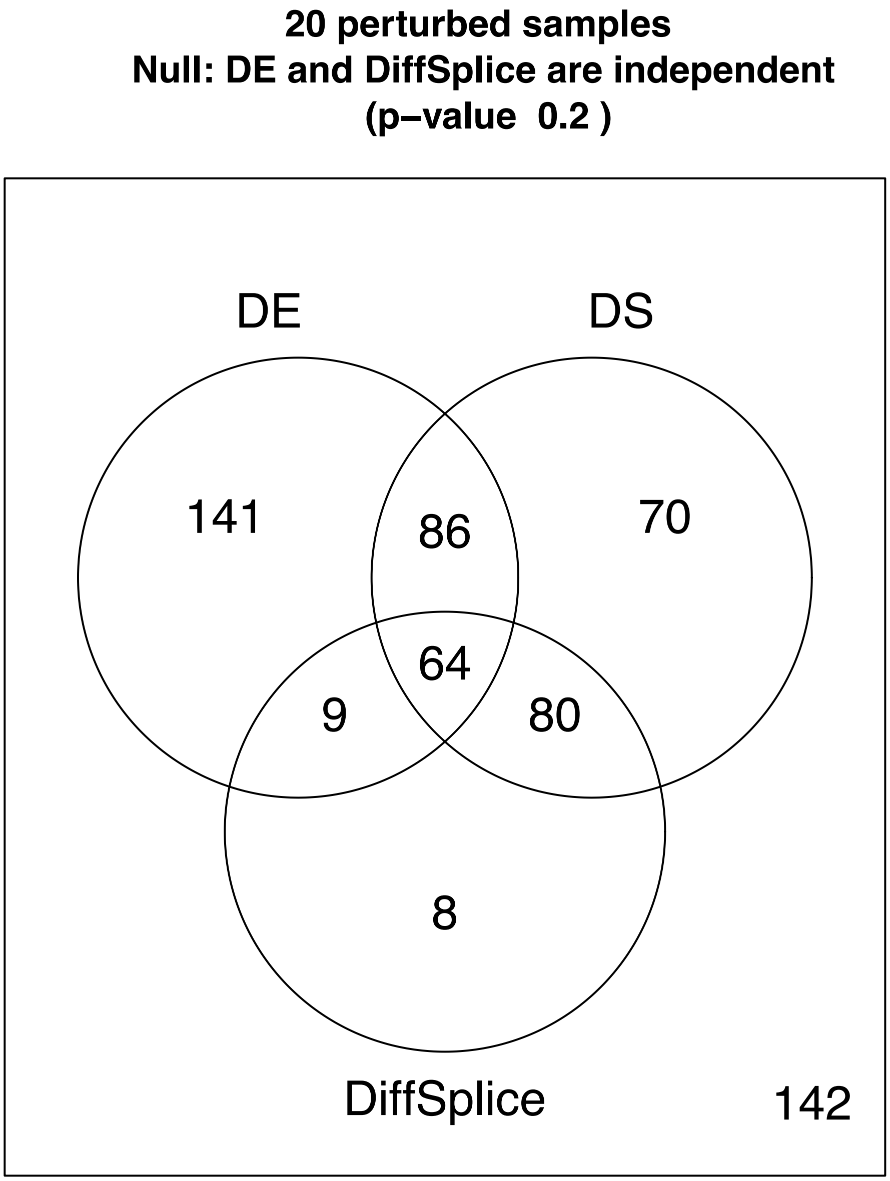
```
## 100
## 200
## 300
## 400
## 500
```

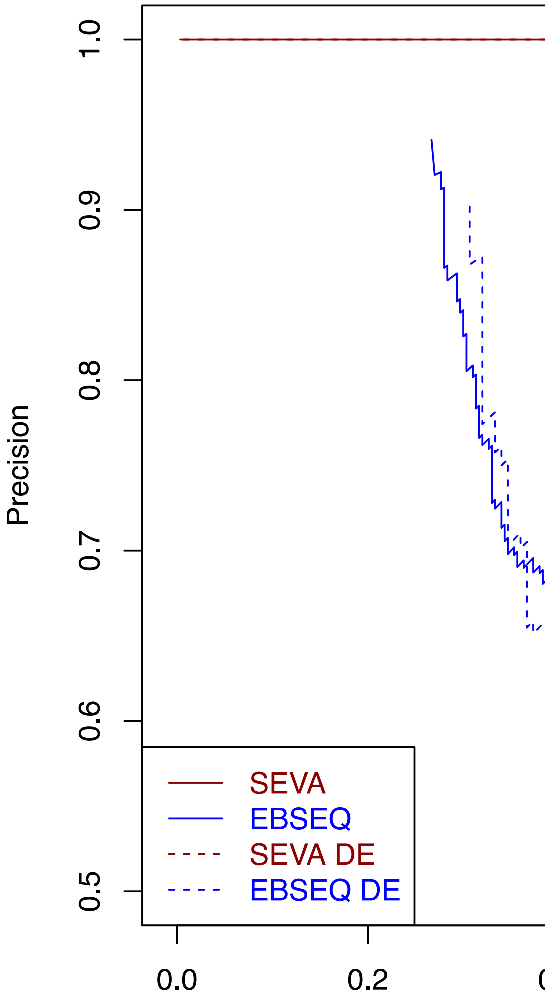


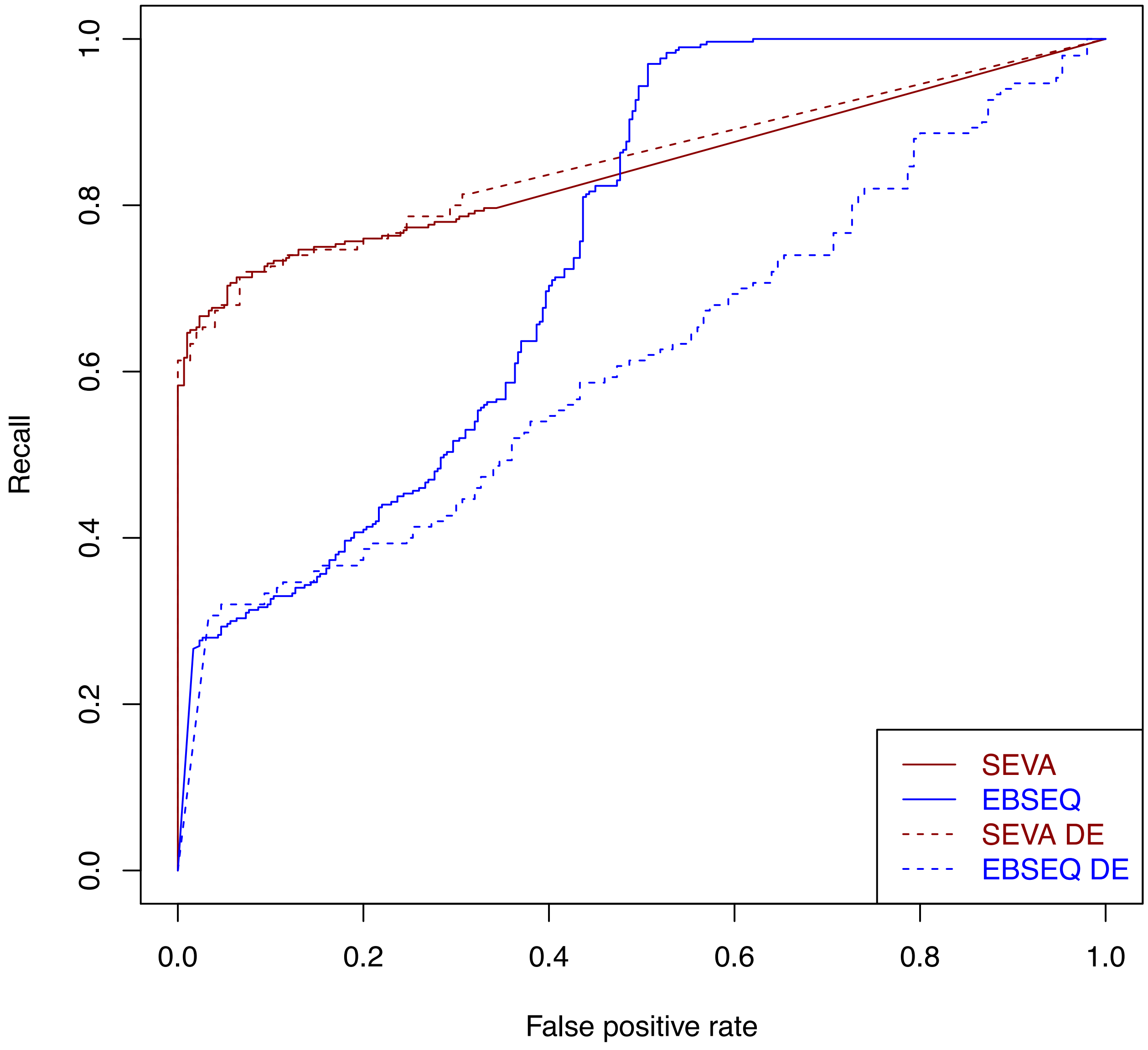
20 perturbed samples
Null: DE and SEVA are independent
(p–value 0.42)

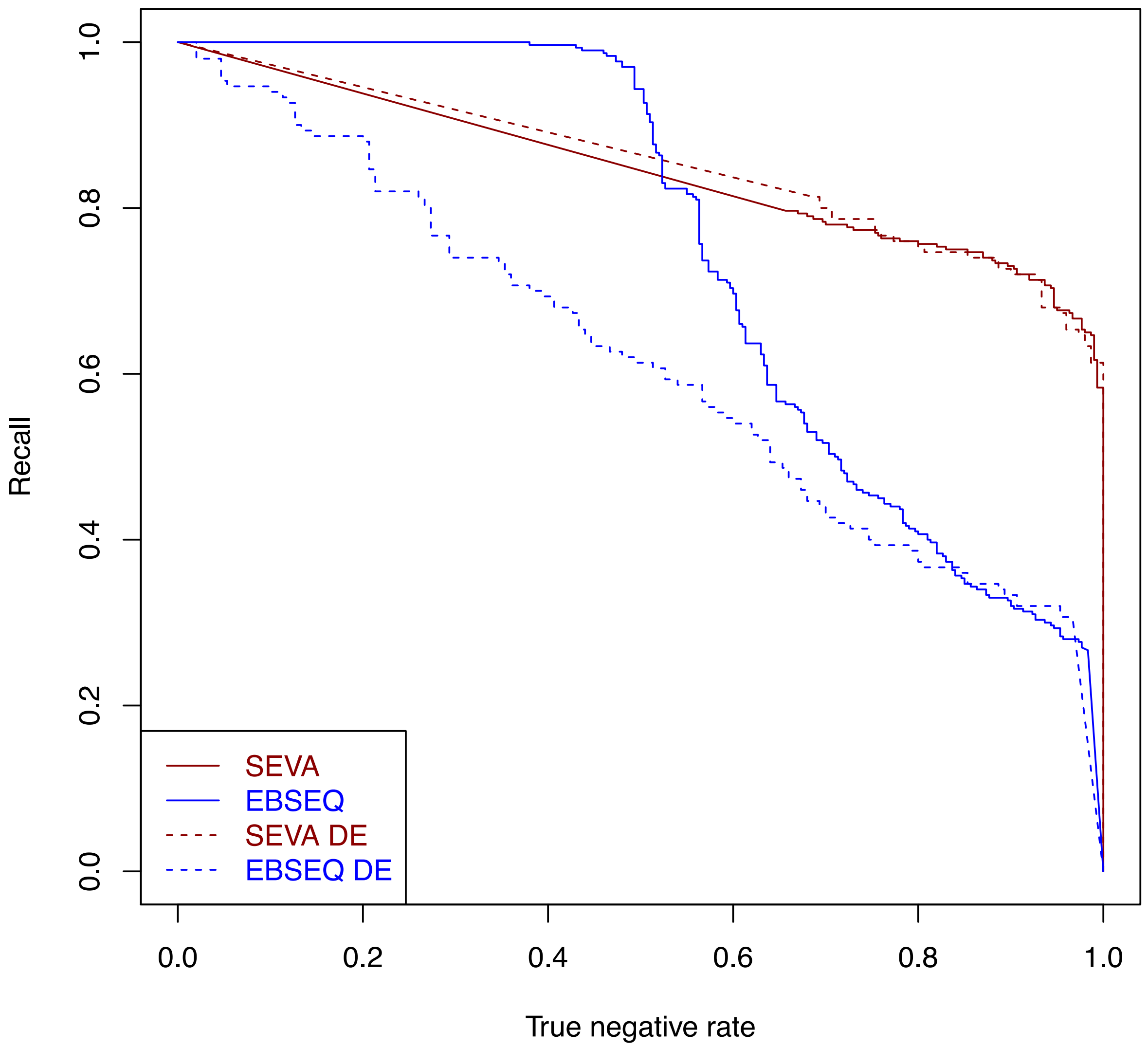


20 pe
Null: DE and
(p–


20 perturbed samples
Null: DE and DiffSplice are independent
(p−value 0.2)
DE
DS
DiffSplice
141
86
70
64
9
80
8
142

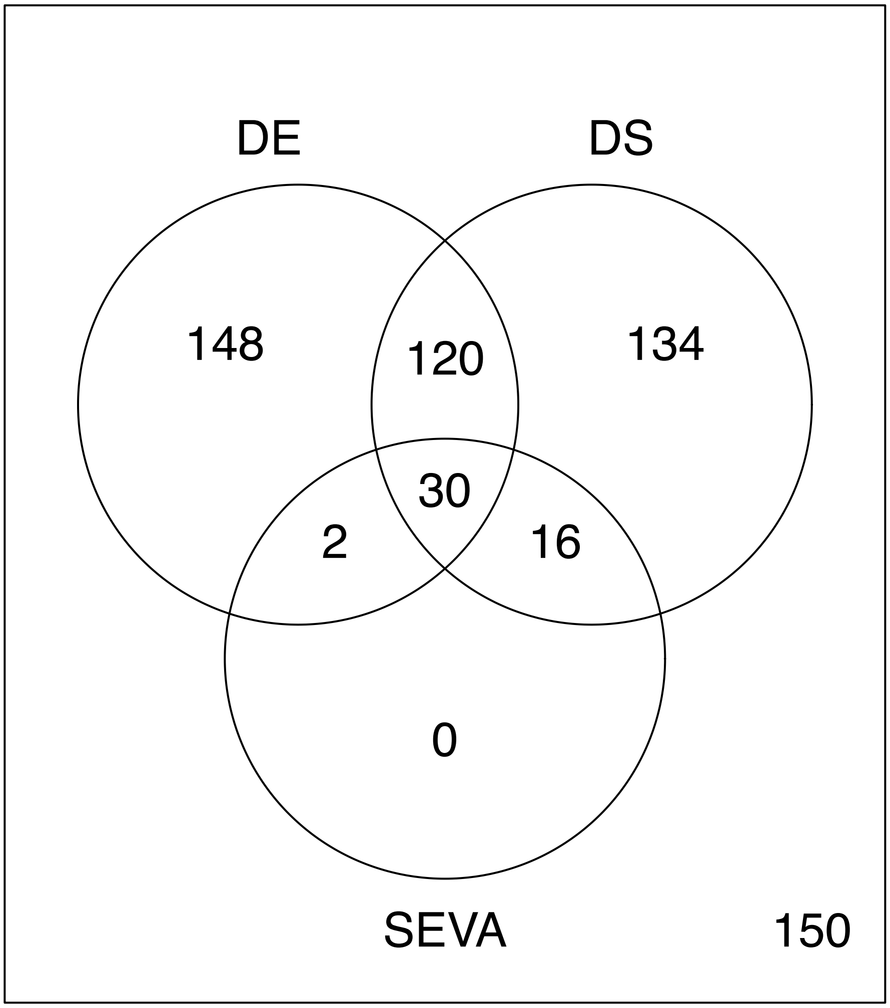

Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
SEVA
EBSEQ
SEVA DE
EBSEQ DE


Recall
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
SEVA
EBSEQ
SEVA DE
EBSEQ DE

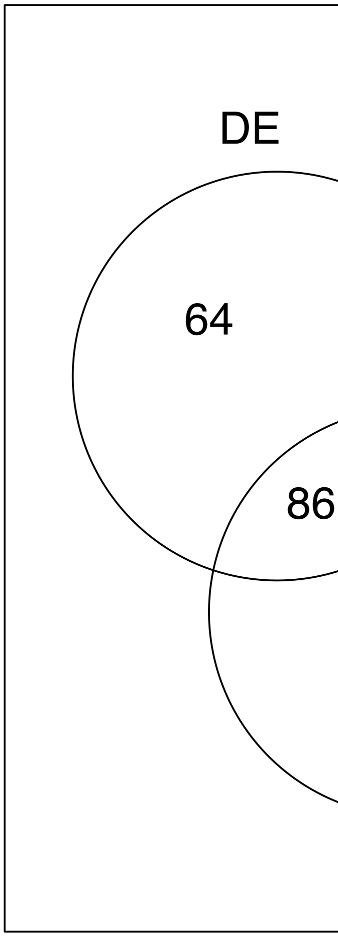


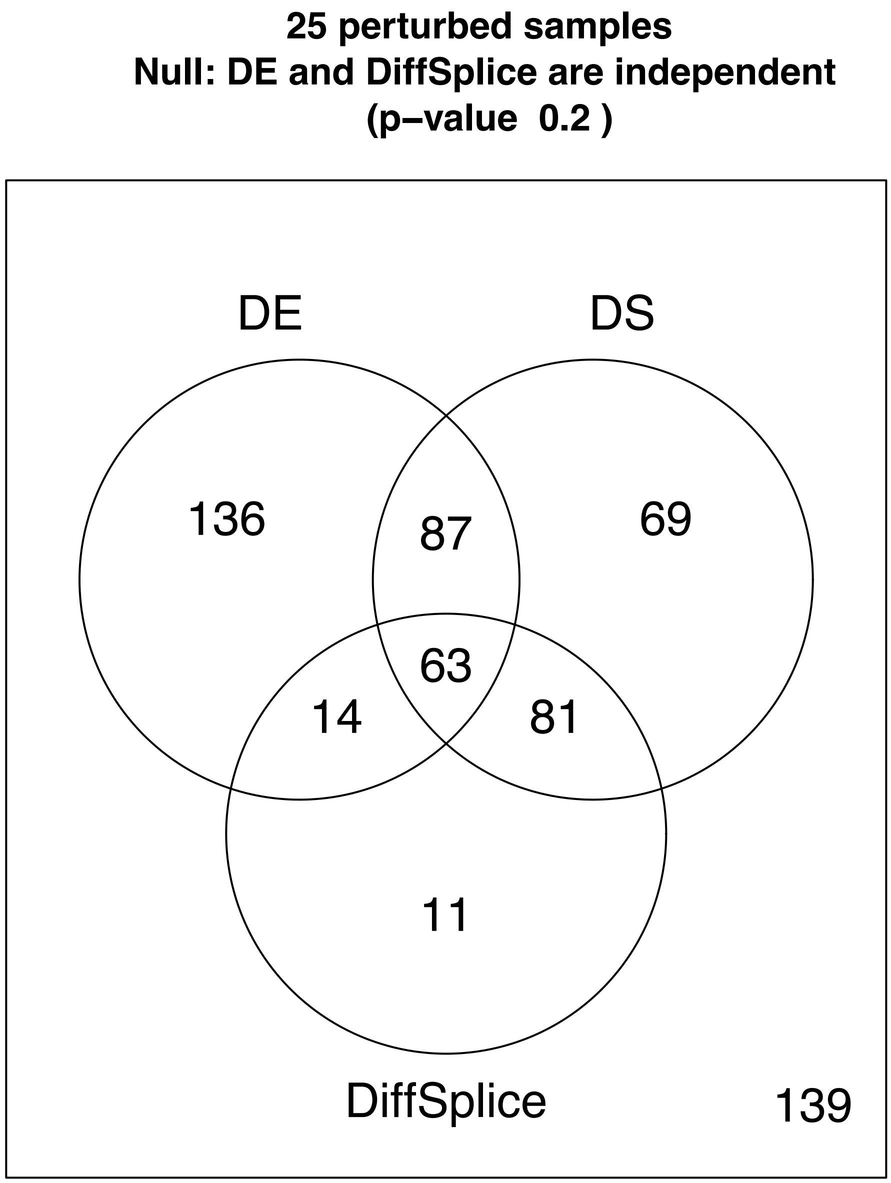
```
## 100
## 200
## 300
## 400
## 500
```

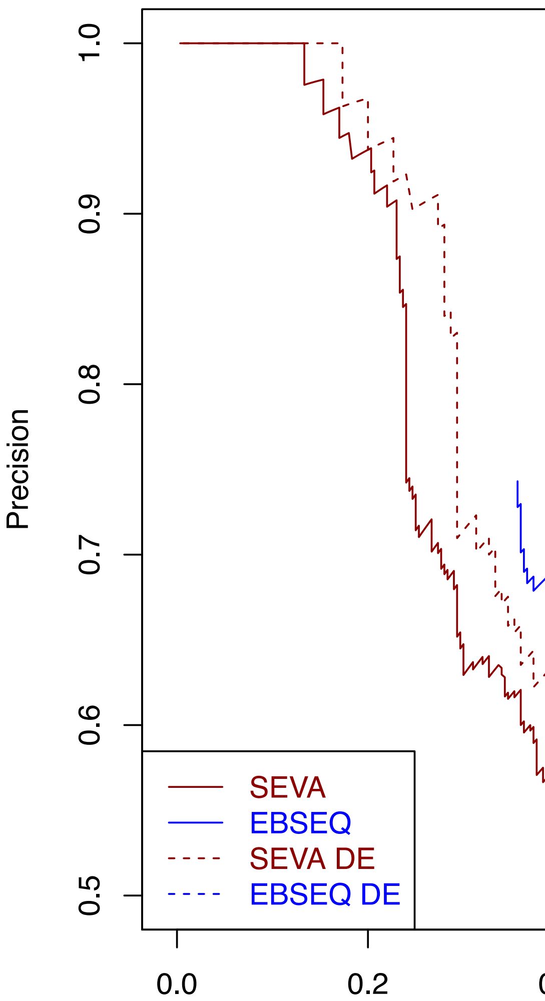
25 perturbed samples
Null: DE and SEVA are independent
(p–value 0.023)

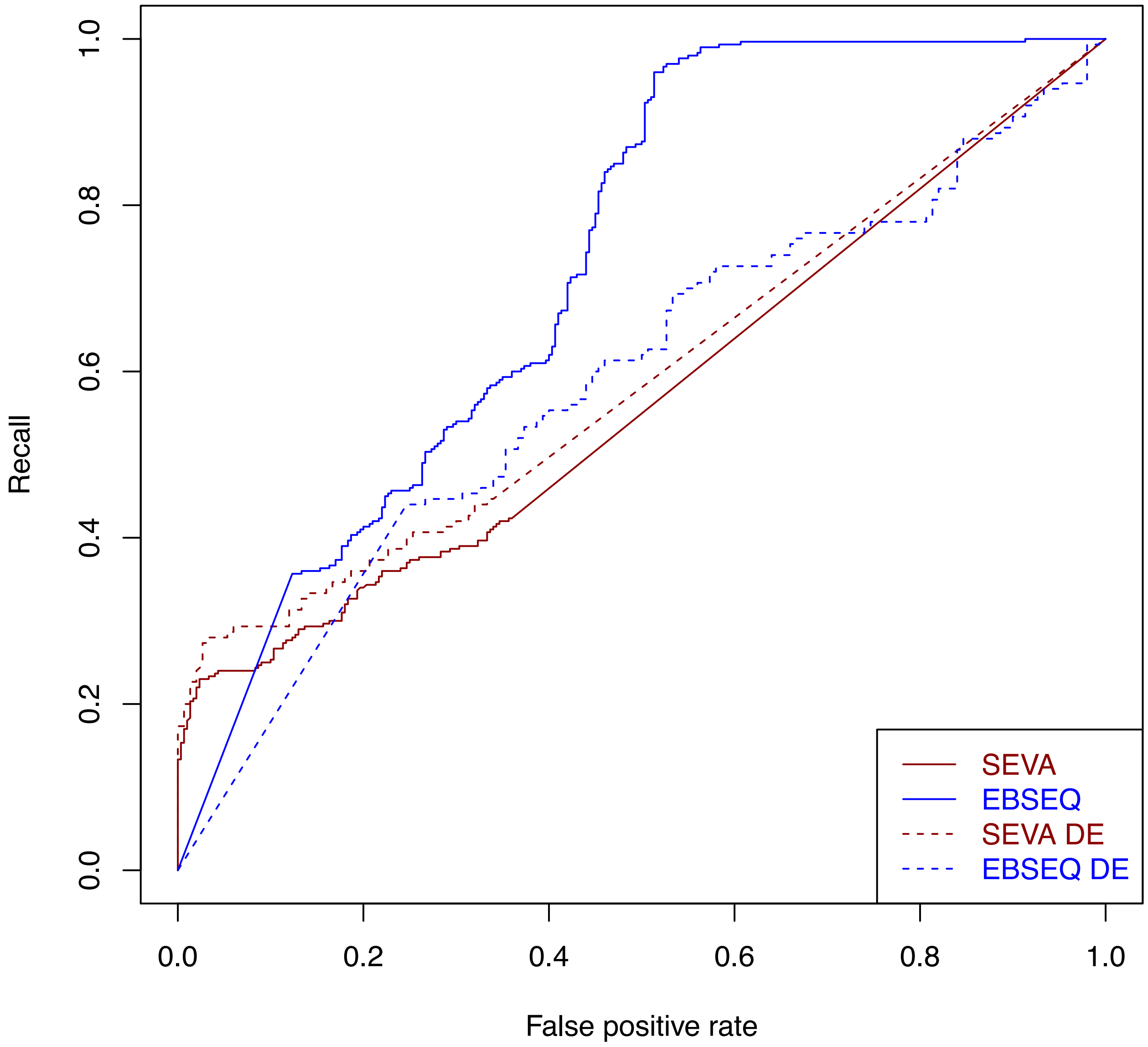


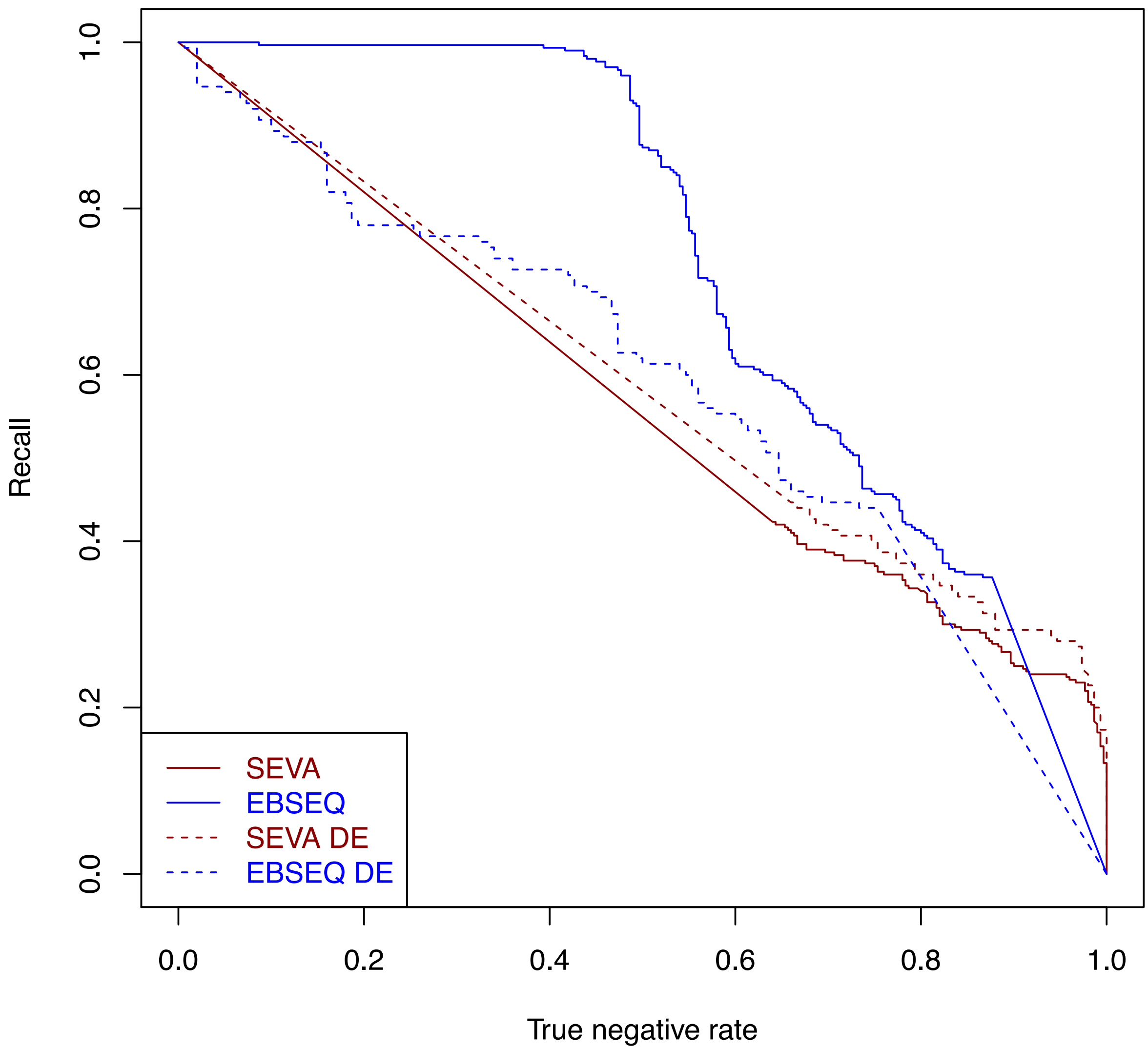
25 pe
Null: DE and
(p–




25 perturbed samples
Null: DE and DiffSplice are independent
(p−value 0.2)
DE
DS
136
87
69
63
14
81
11
DiffSplice
139

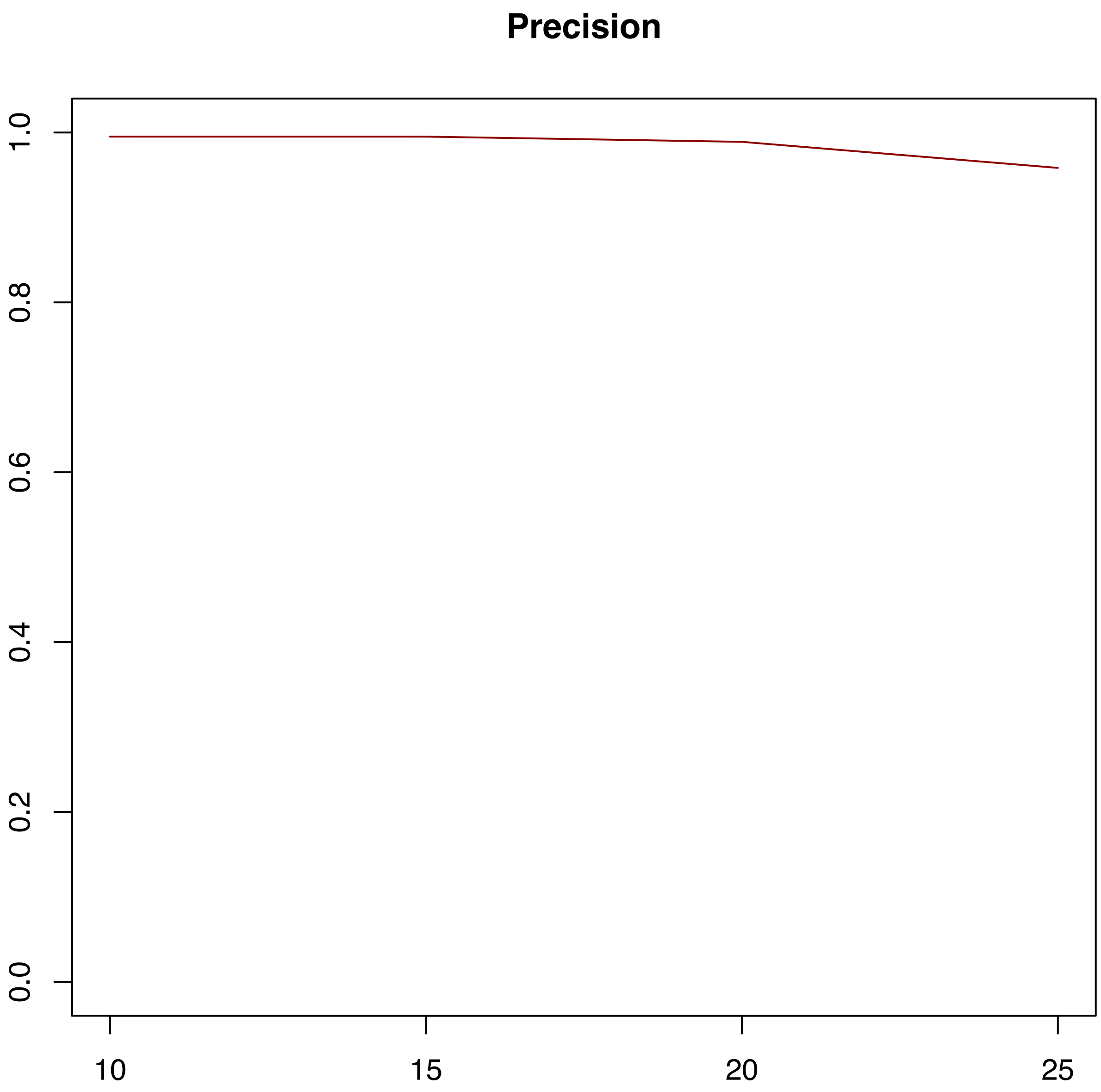

Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
SEVA
EBSEQ
SEVA DE
EBSEQ DE


Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
SEVA
EBSEQ
SEVA DE
EBSEQ DE

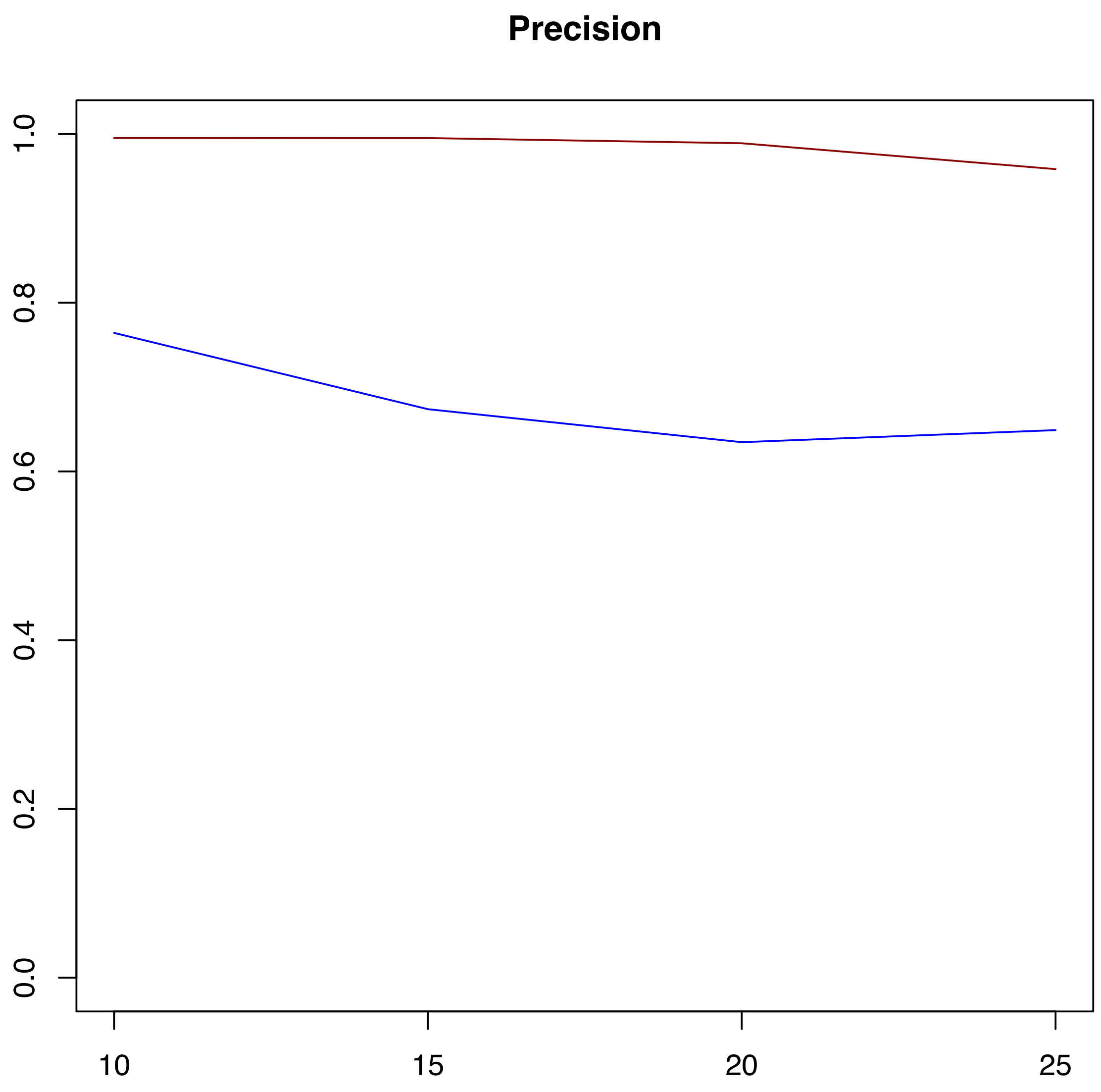


```
#ploting sens and spec vs samp size for different methods
sampNum <- as.numeric(sapply(strsplit(names(experimentSamplesTumors),split = " "), function(x) x[1]))
DSMethods <- c("SEVA","EBSEQ","DiffSplice")
precisionall <- vector(mode = "list",length = length(DSMethods))
names(precisionall) <- DSMethods
recallall <- precisionall
precisionDEall <- precisionall
recallDEall <- recallall
specall <- recallall
specDEall <- recallall
```

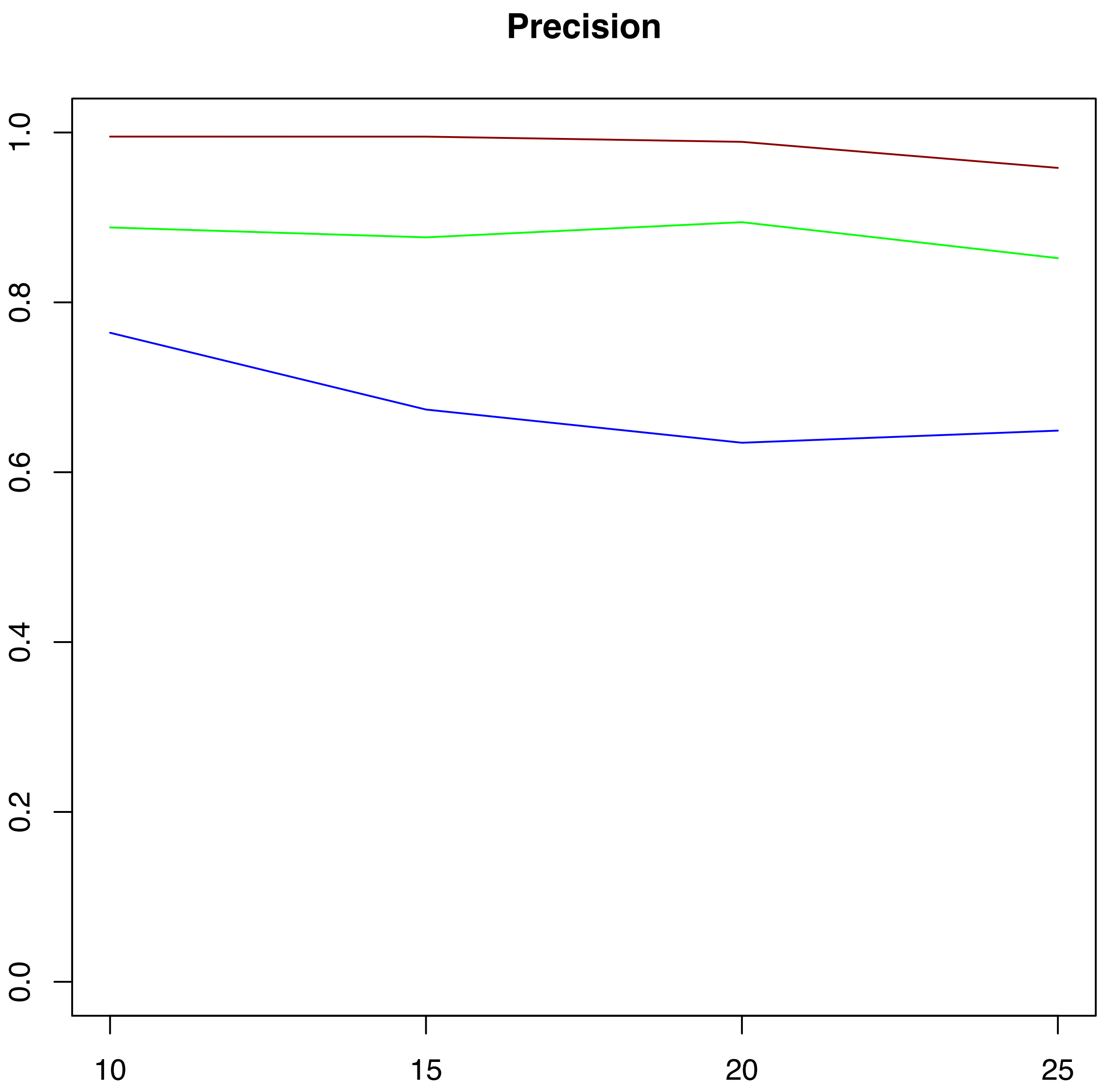
```
for( j in seq_along(DSMethods)){
  precision <- vector(mode = "numeric", length(experimentSamplesTumors))
  recall <- vector(mode = "numeric", length(experimentSamplesTumors))
  precisionDE <- vector(mode = "numeric", length(experimentSamplesTumors))
  recallDE <- vector(mode = "numeric", length(experimentSamplesTumors))
  spec <- vector(mode = "numeric", length(experimentSamplesTumors))
  specDE <- vector(mode = "numeric", length(experimentSamplesTumors))


    for( i in seq_along(experimentSamplesTumors))
    {




      DSDEGenes <- names(DEDS)

      recall[i] <- sum(Venn4Percentage[[i]][DSGenes,DSMethods[j]])/length(DSGenes)
      precision[i] <- sum(Venn4Percentage[[i]][DSGenes,DSMethods[j]])/sum(Venn4Percentage[[i]][OnlyGenes
      spec[i] <- 1-mean(Venn4Percentage[[i]][setdiff(OnlyGenesGroundTruth,DSGenes),DSMethods[j]])

      recallDE[i]<- sum(Venn4Percentage[[i]][DSDEGenes,DSMethods[j]])/length(DEDSGenes)
      precisionDE[i] <- sum(Venn4Percentage[[i]][DSDEGenes,DSMethods[j]])/sum(Venn4Percentage[[i]][DEGen
      specDE[i] <- 1-mean(Venn4Percentage[[i]][setdiff(DEGenes,DSGenes),DSMethods[j]])



    }
  precisionall[[DSMethods[j]]] <- precision
  recallall[[DSMethods[j]]] <- recall
  specall[[DSMethods[j]]] <- spec

  precisionDEall[[DSMethods[j]]] <- precisionDE
  recallDEall[[DSMethods[j]]] <- recallDE
  specDEall[[DSMethods[j]]] <- specDE




}
plot(x= sampNum, y= precisionall[["SEVA"]], xlab="", ylab= "", main = "Precision", type="l",col="dark r
```



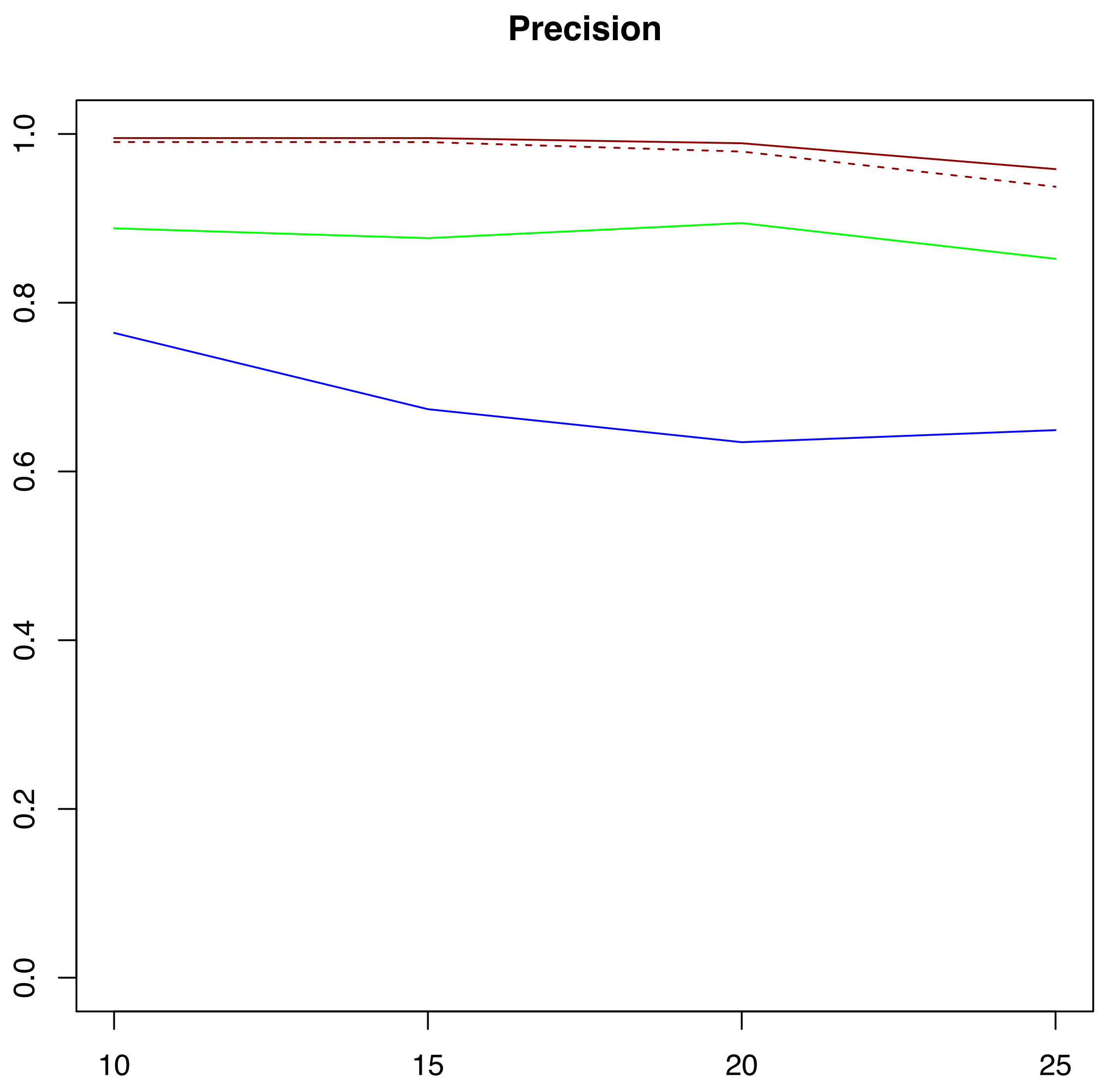
```r
lines(x= sampNum, y= precisionall[["EBSEQ"]],col="blue",lty=1,pch=1)
```



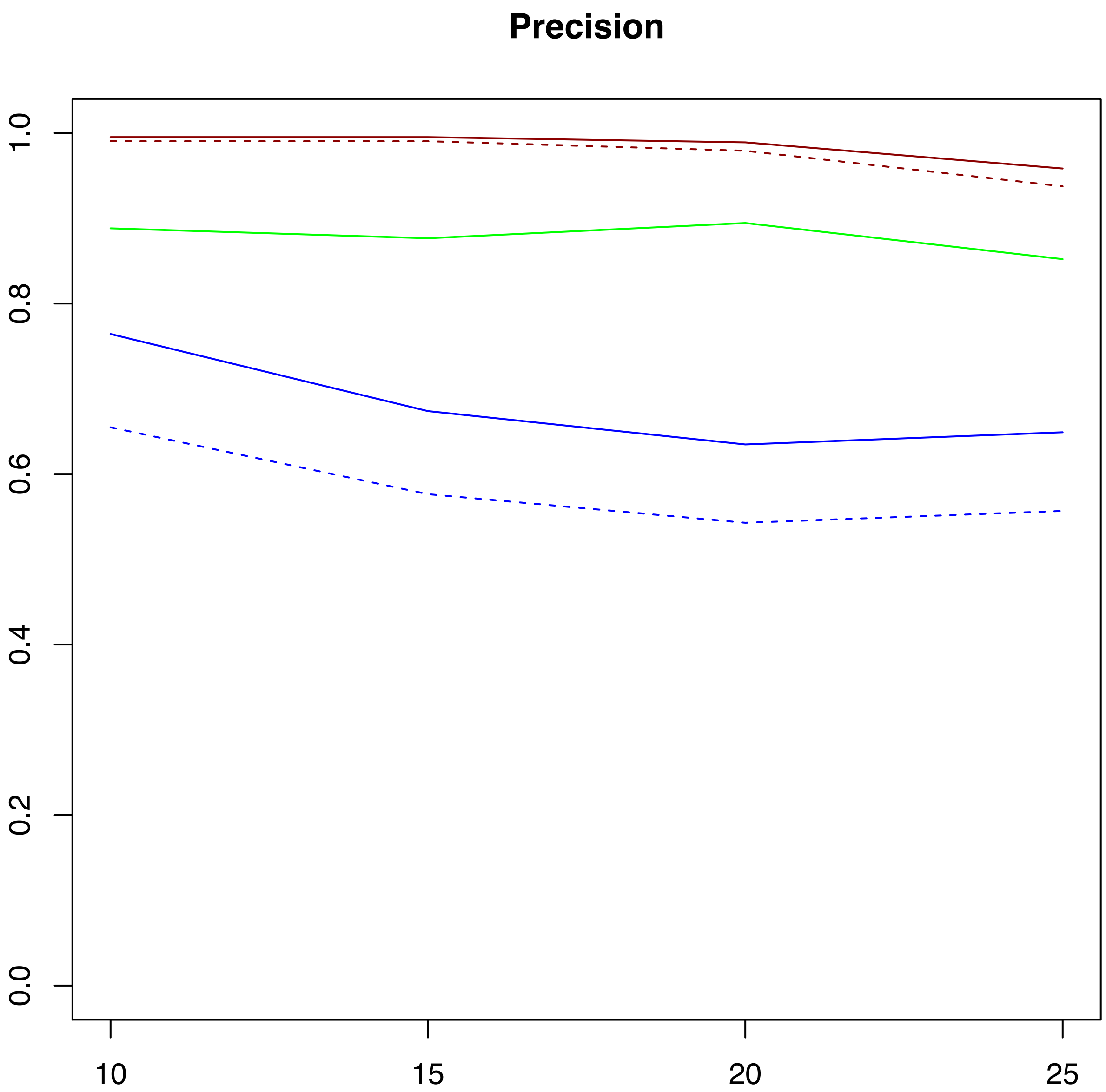
```r
lines(x= sampNum, y= precisionall[["DiffSplice"]],col="green",lty=1,pch=1)
```



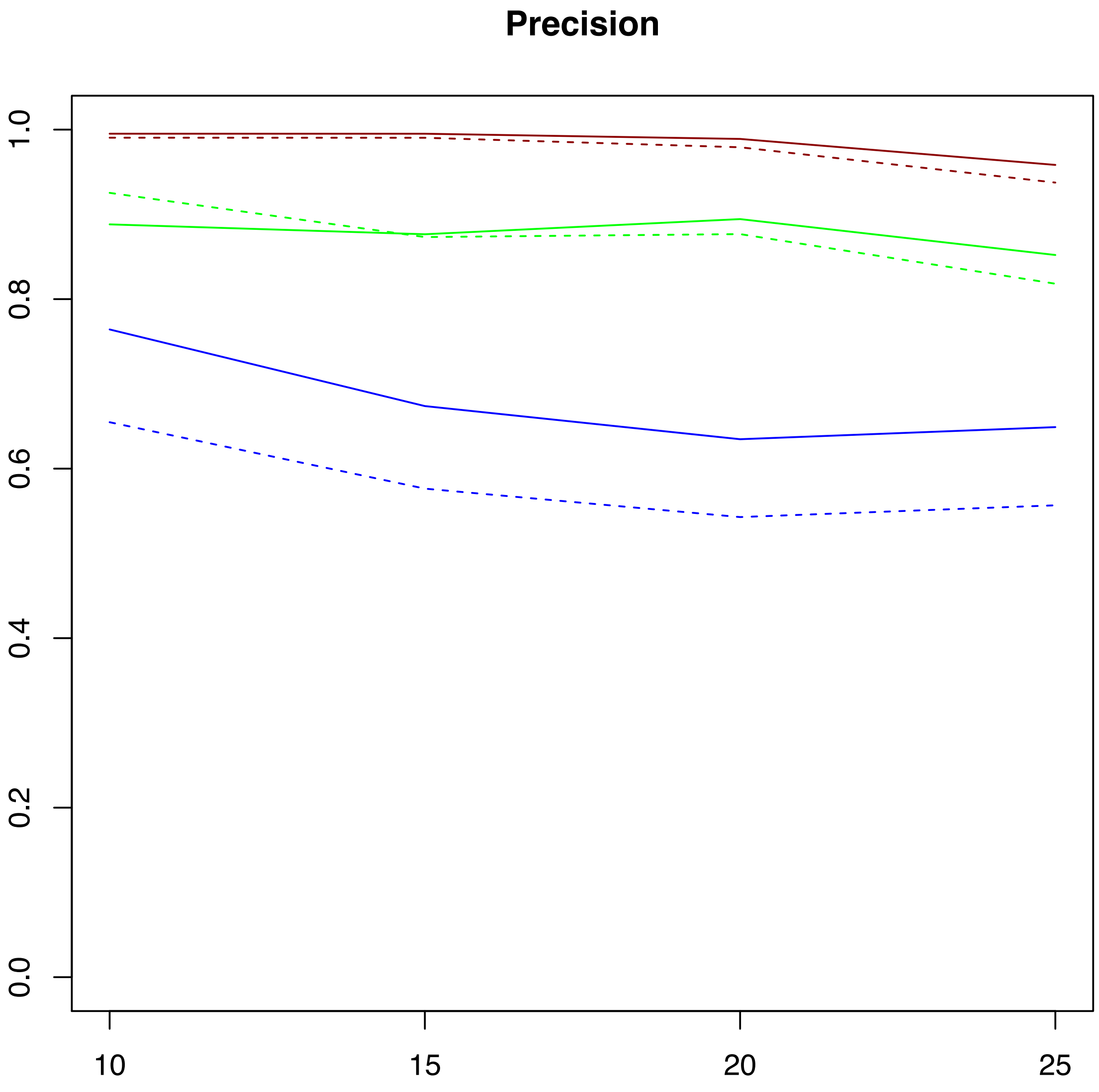
```
lines(x= sampNum, y= precisionDEall[["SEVA"]],col="dark red",lty=2,pch=2)
```



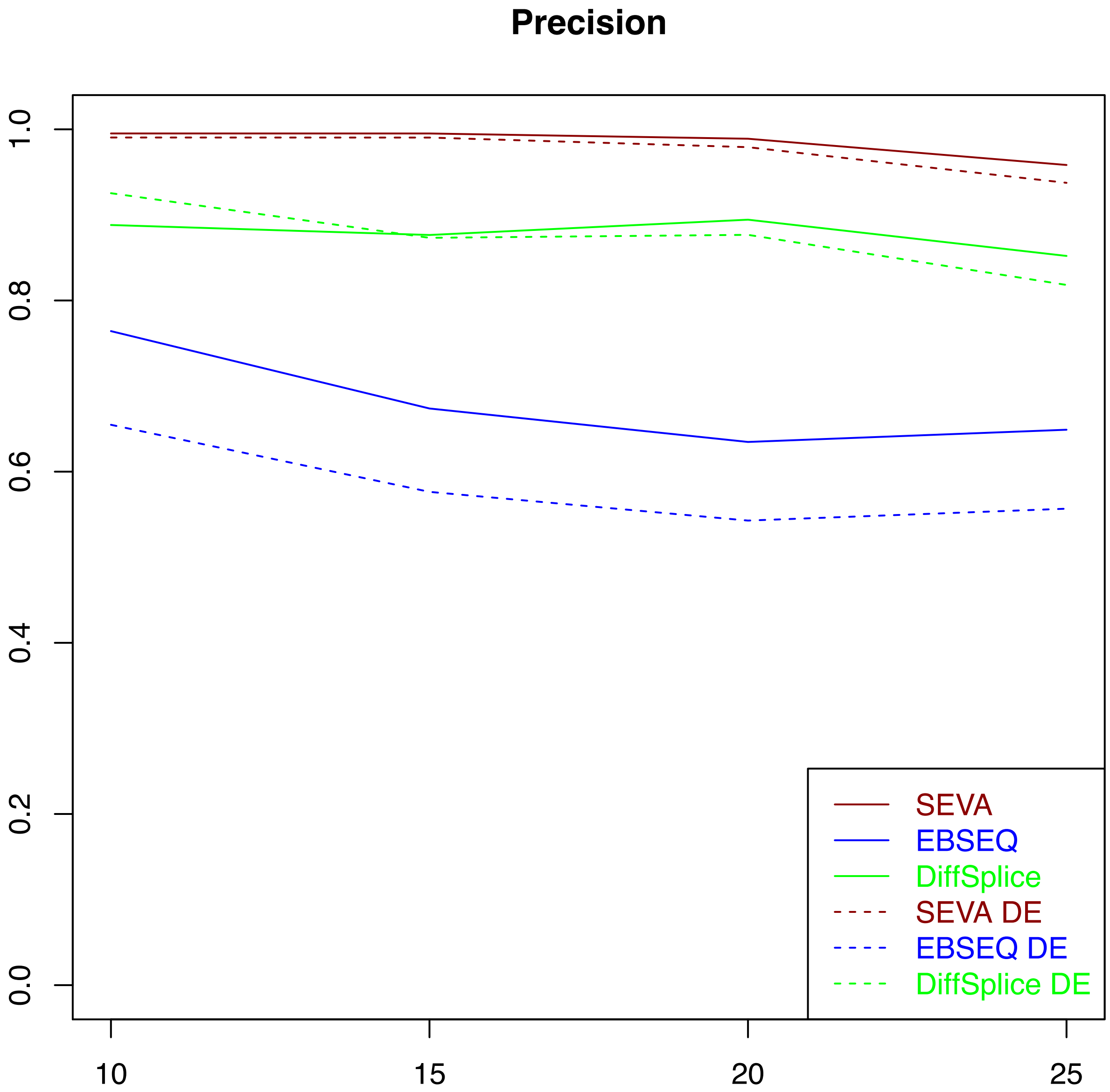
```
lines(x= sampNum, y= precisionDEall[["EBSEQ"]],col="blue",lty=2,pch=2)
```



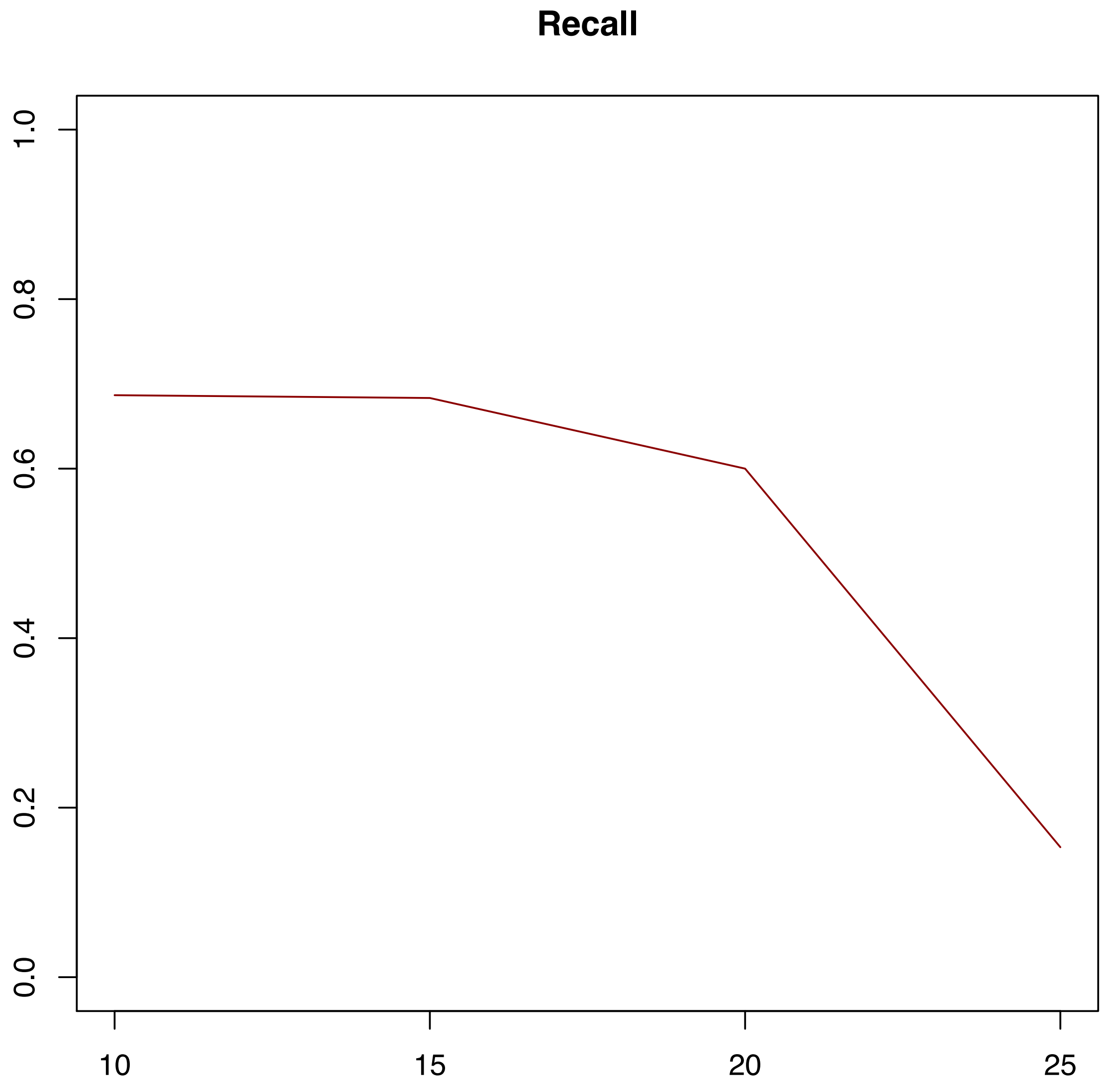
```
lines(x= sampNum, y= precisionDEall[["DiffSplice"]],col="green",lty=2,pch=2)
```



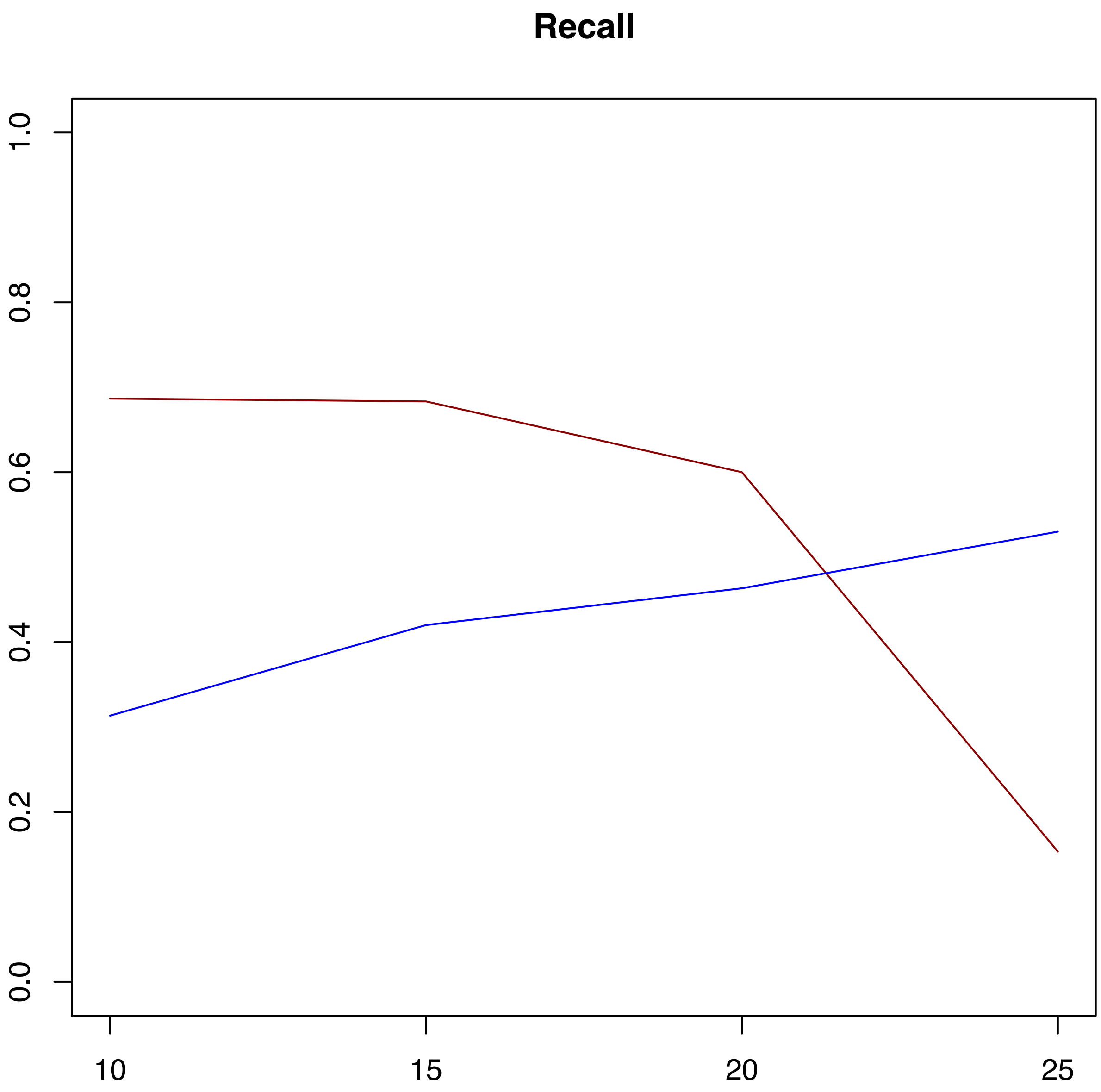
```
legend("bottomright",legend = c("SEVA","EBSEQ","DiffSplice","SEVA DE","EBSEQ DE","DiffSplice DE"),
            col=c("dark red","blue","green","dark red","blue","green"),
            text.col = c("dark red","blue","green","dark red","blue","green"),
       lty = c(1,1,1,2,2,2)     )
```



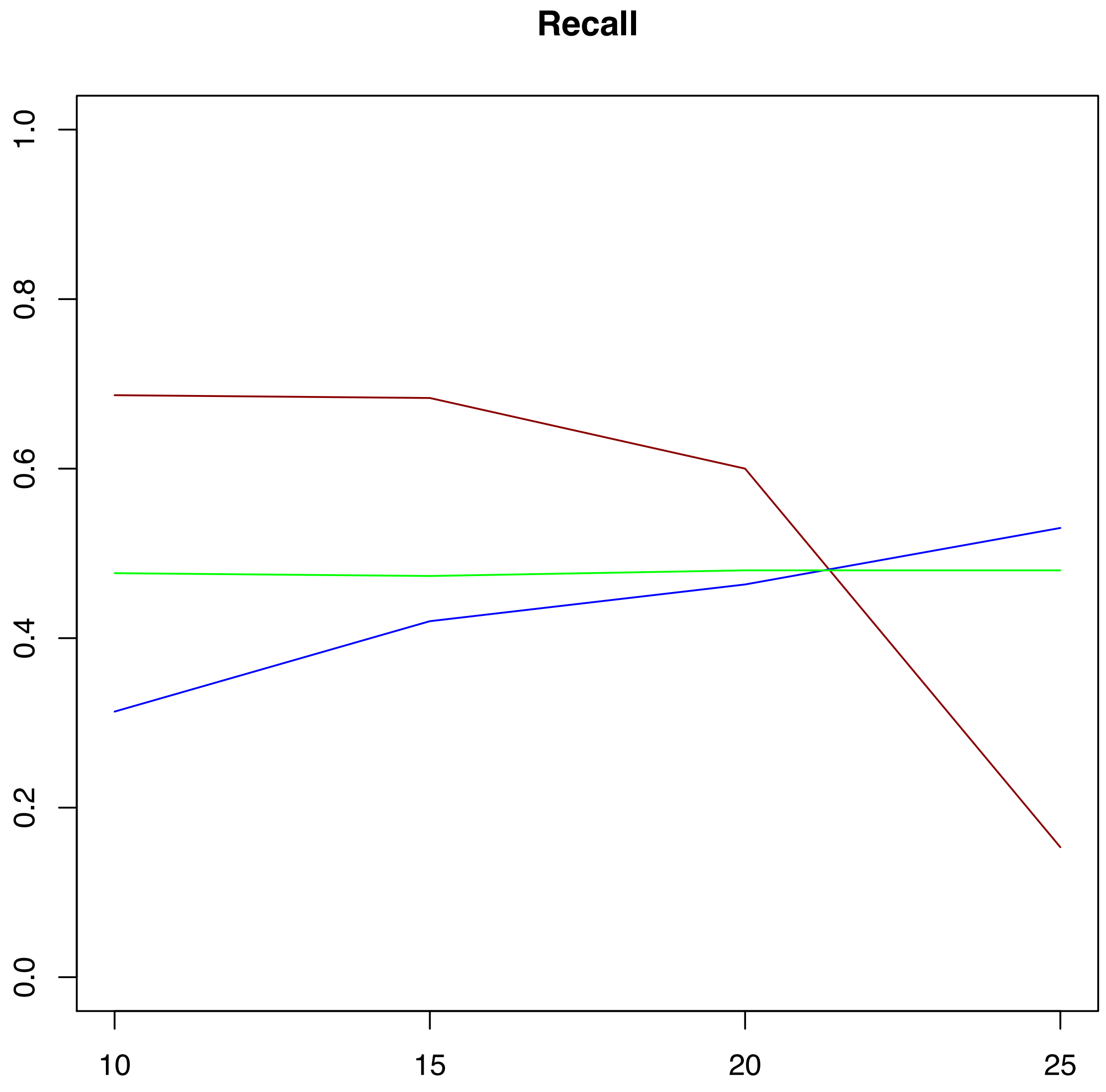
```
plot(x= sampNum, y= recallall[["SEVA"]], xlab="", ylab= "", main = "Recall", type="l",col="dark red", yl
```



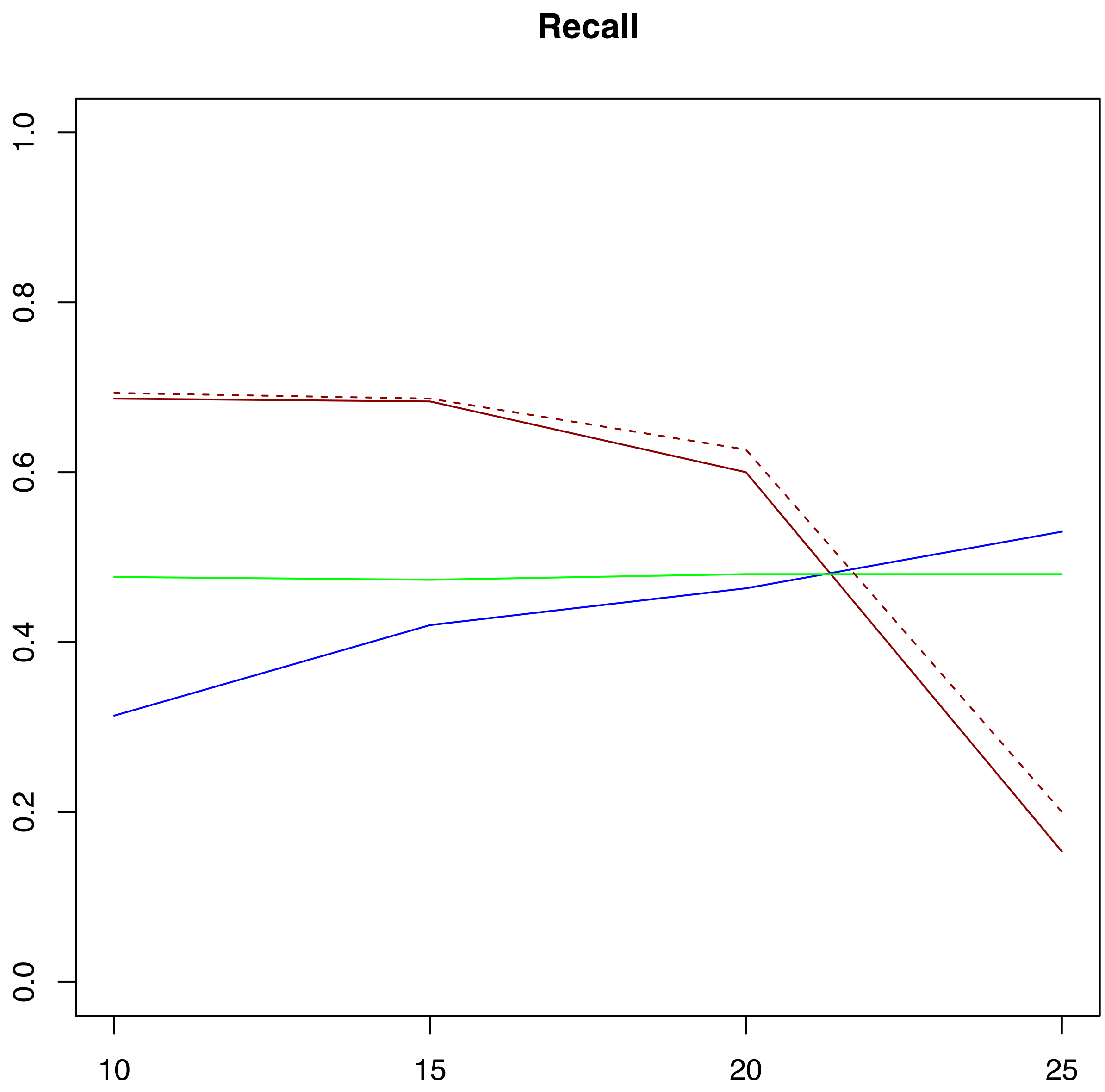
```r
lines(x= sampNum, y= recallall[["EBSEQ"]],col="blue",lty=1,pch=1)
```



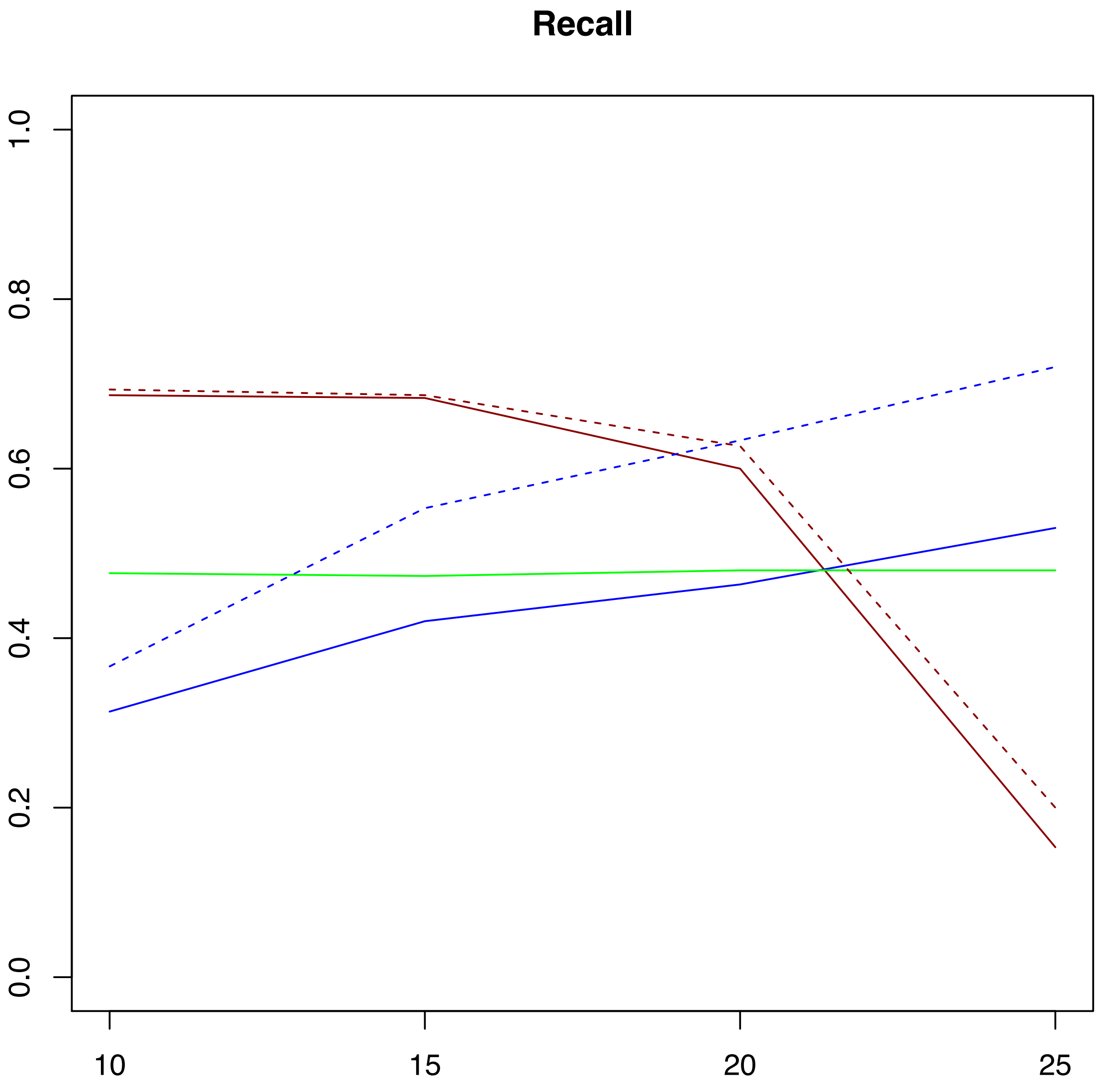
```
lines(x= sampNum, y= recallall[["DiffSplice"]],col="green",lty=1,pch=1)
```



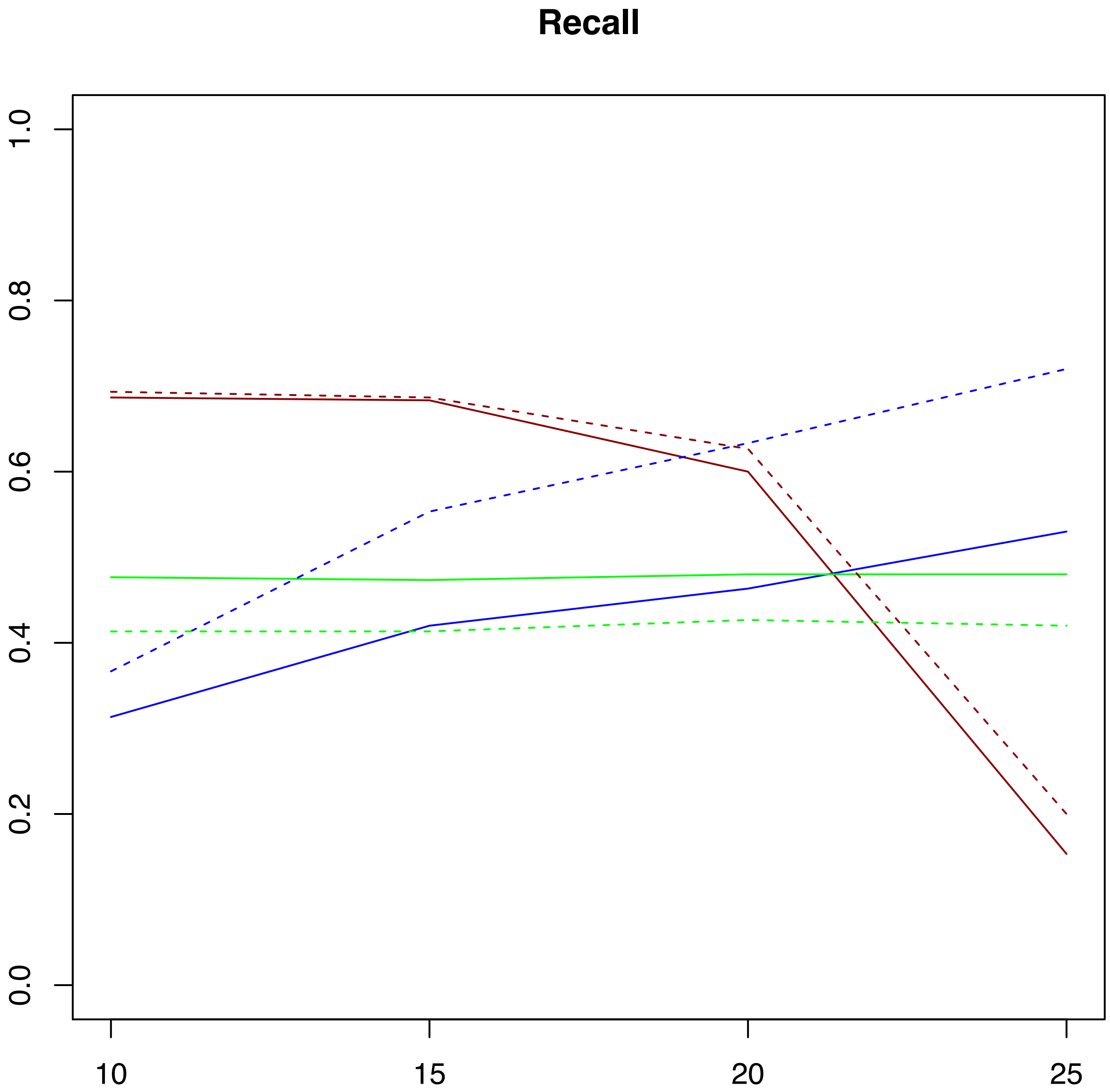
```
lines(x= sampNum, y= recallDEall[["SEVA"]],col="dark red",lty=2,pch=2)
```



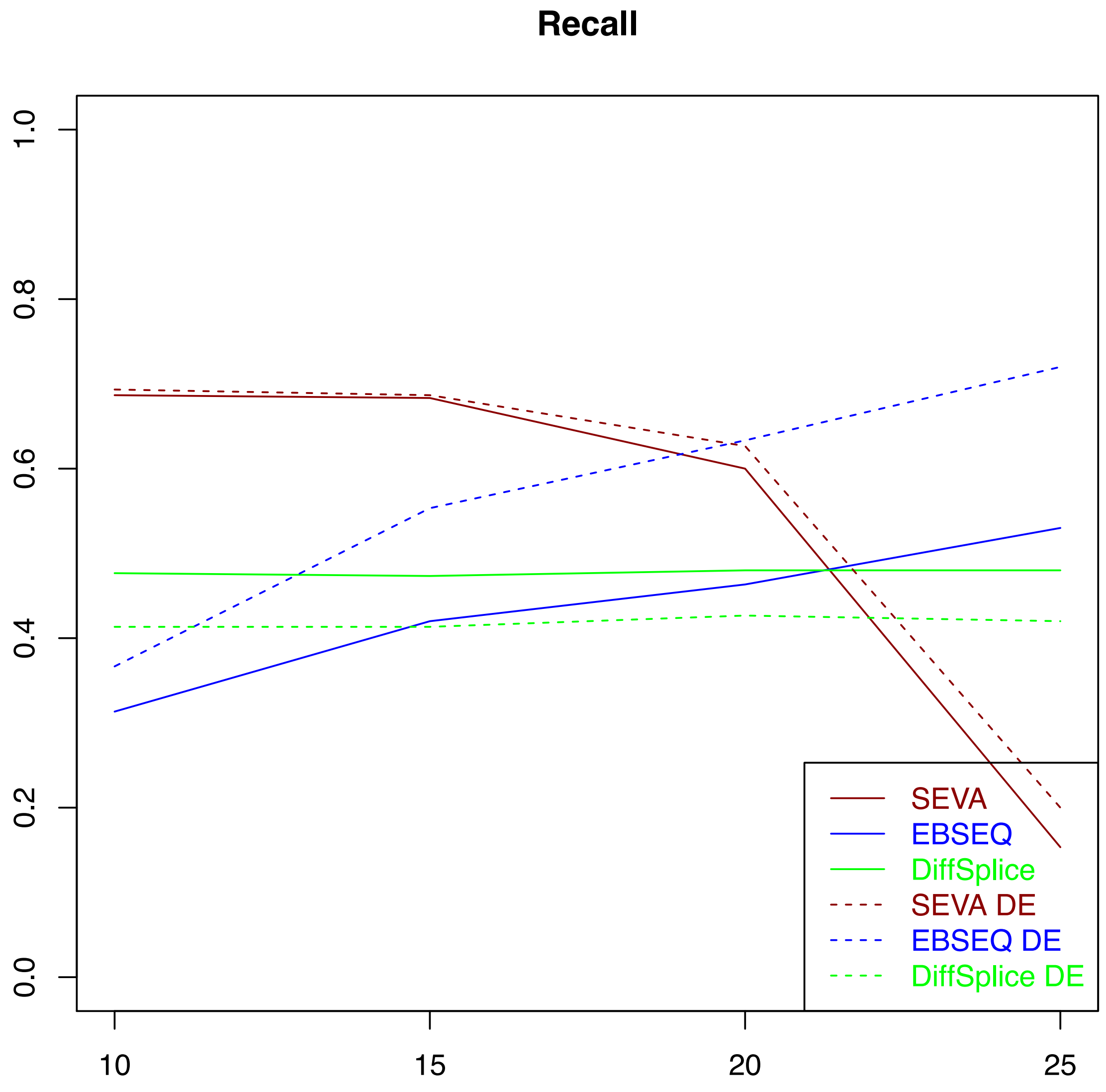
```
lines(x= sampNum, y= recallDEall[["EBSEQ"]],col="blue",lty=2,pch=2)
```



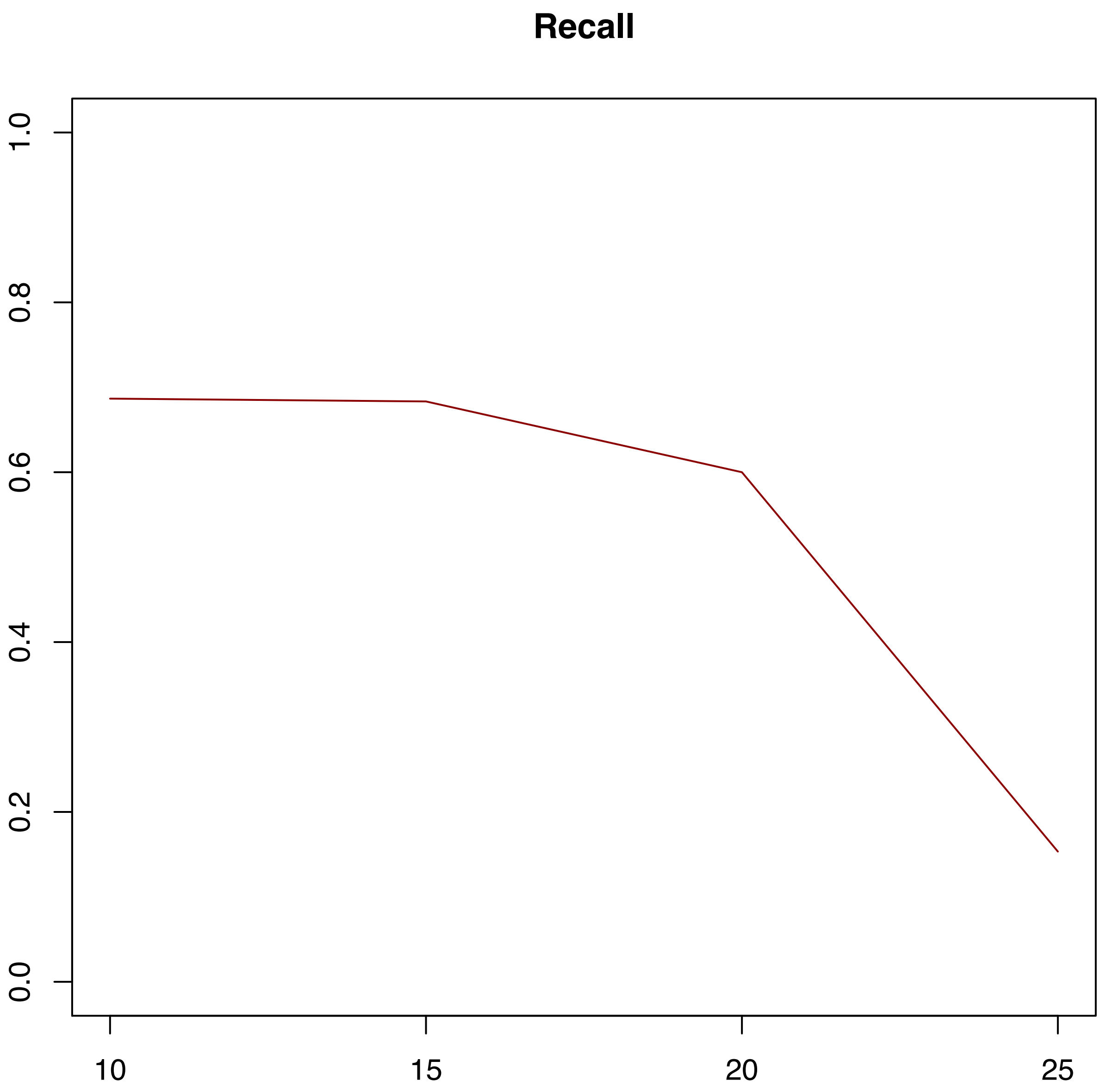
```
lines(x= sampNum, y= recallDEall[["DiffSplice"]],col="green",lty=2,pch=2)
```



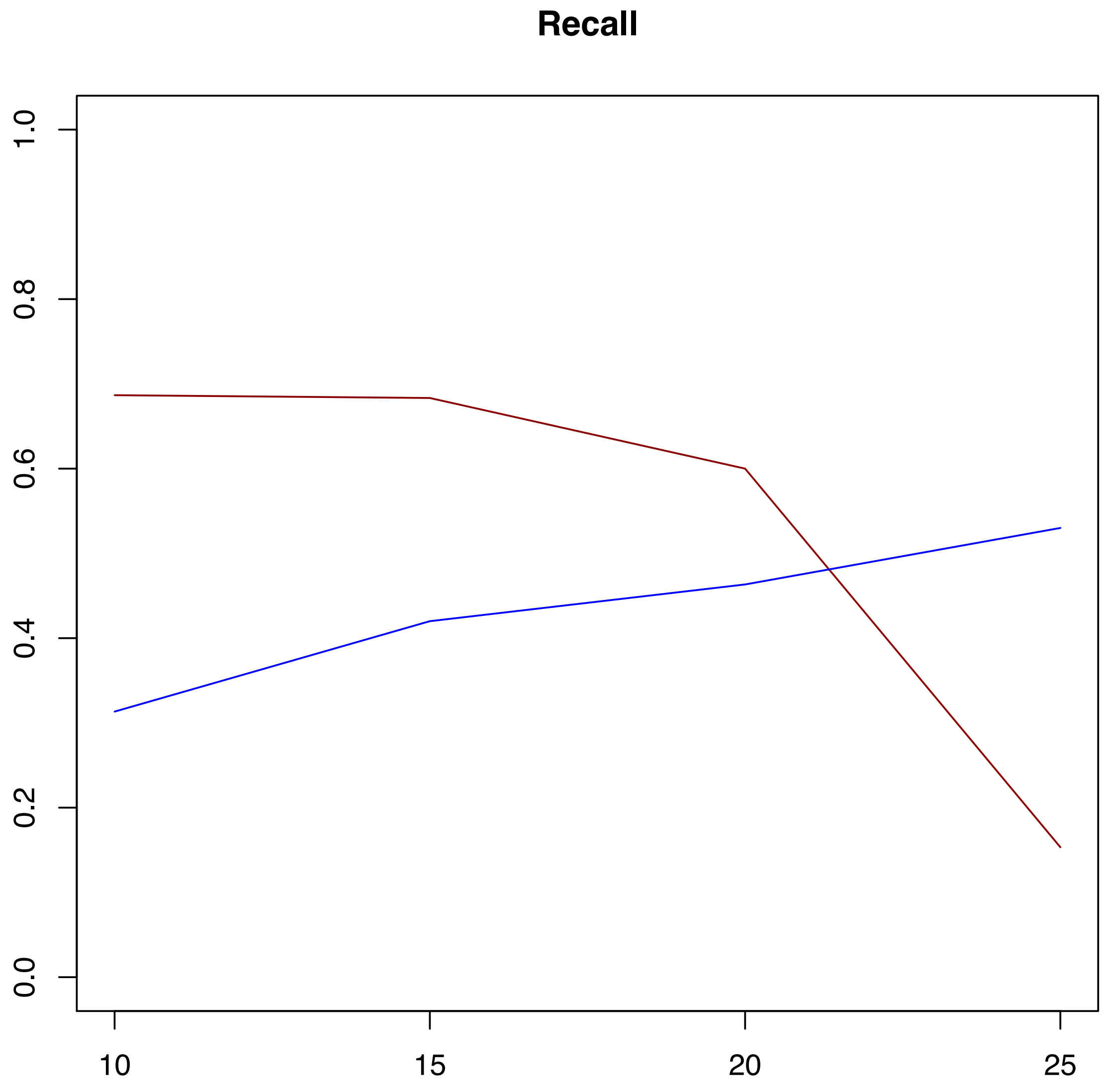
```
legend("bottomright",legend = c("SEVA","EBSEQ","DiffSplice","SEVA DE","EBSEQ DE","DiffSplice DE"),
       col=c("dark red","blue","green","dark red","blue","green"),
       text.col = c("dark red","blue","green","dark red","blue","green"),
       lty = c(1,1,1,2,2,2)      )
```



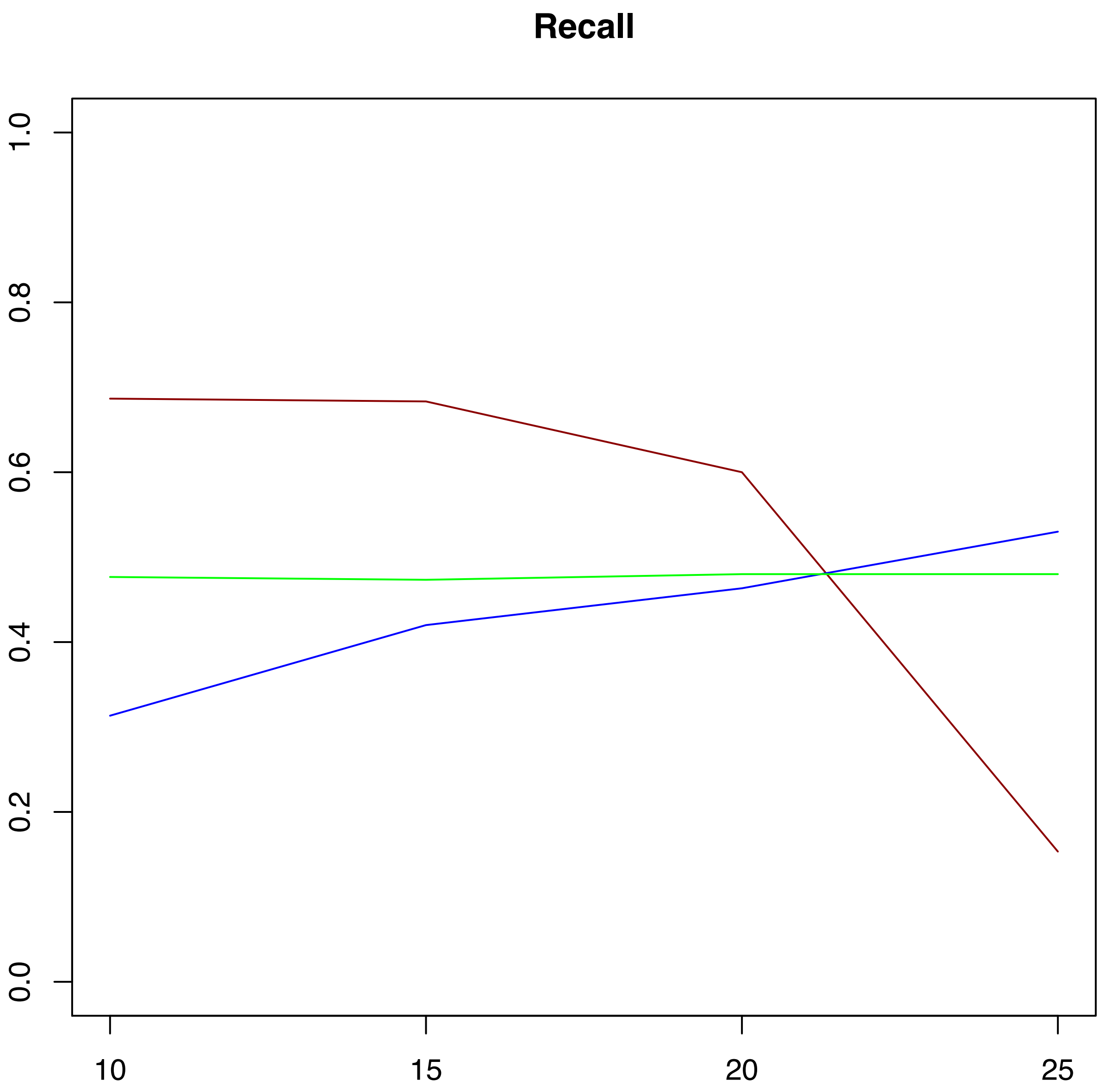
```
plot(x= sampNum, y= recallall[["SEVA"]], xlab="", ylab= "", main = "Recall", type="l",col="dark red", yl
```



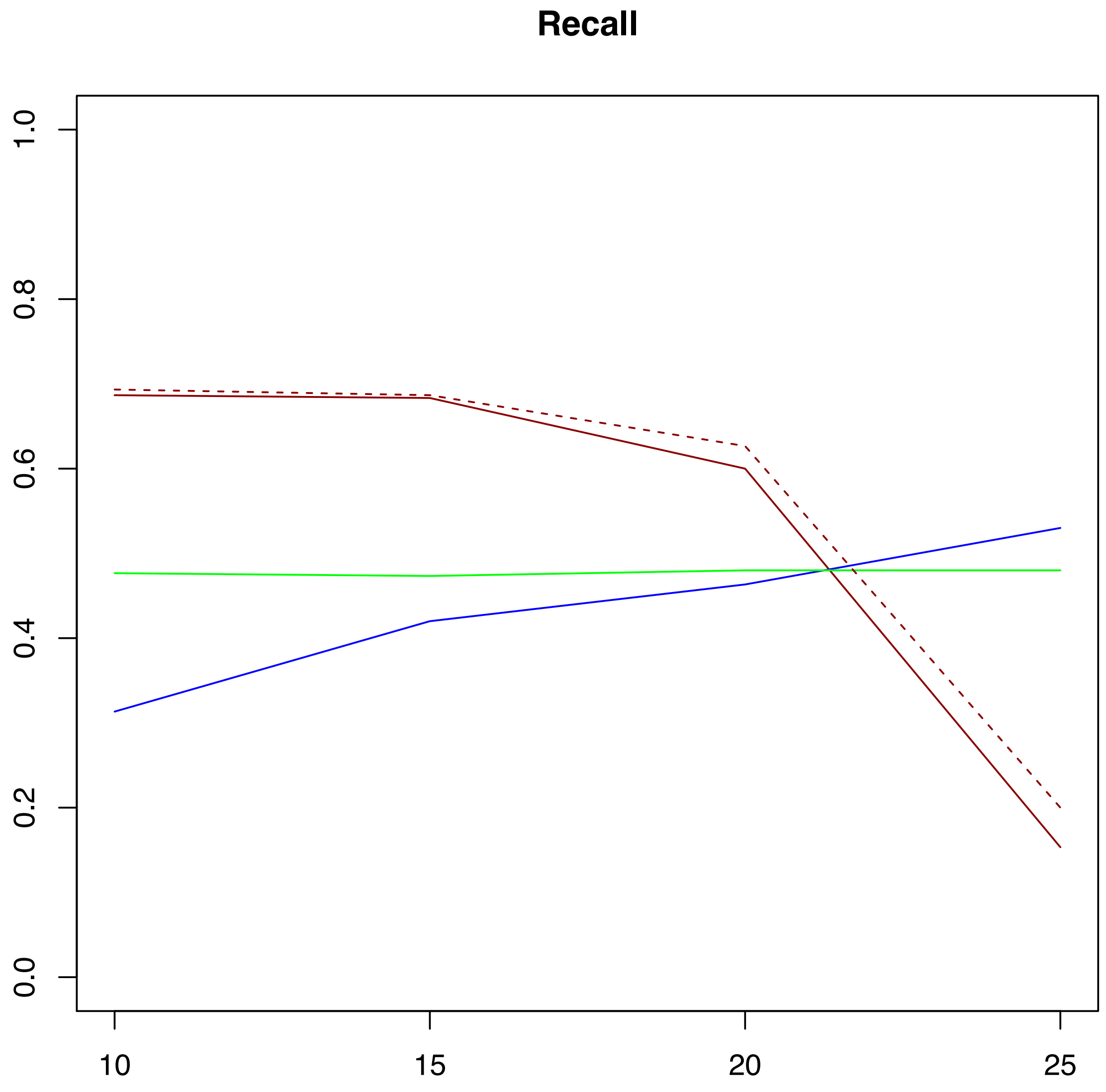
```
lines(x= sampNum, y= recallall[["EBSEQ"]],col="blue",lty=1,pch=1)
```



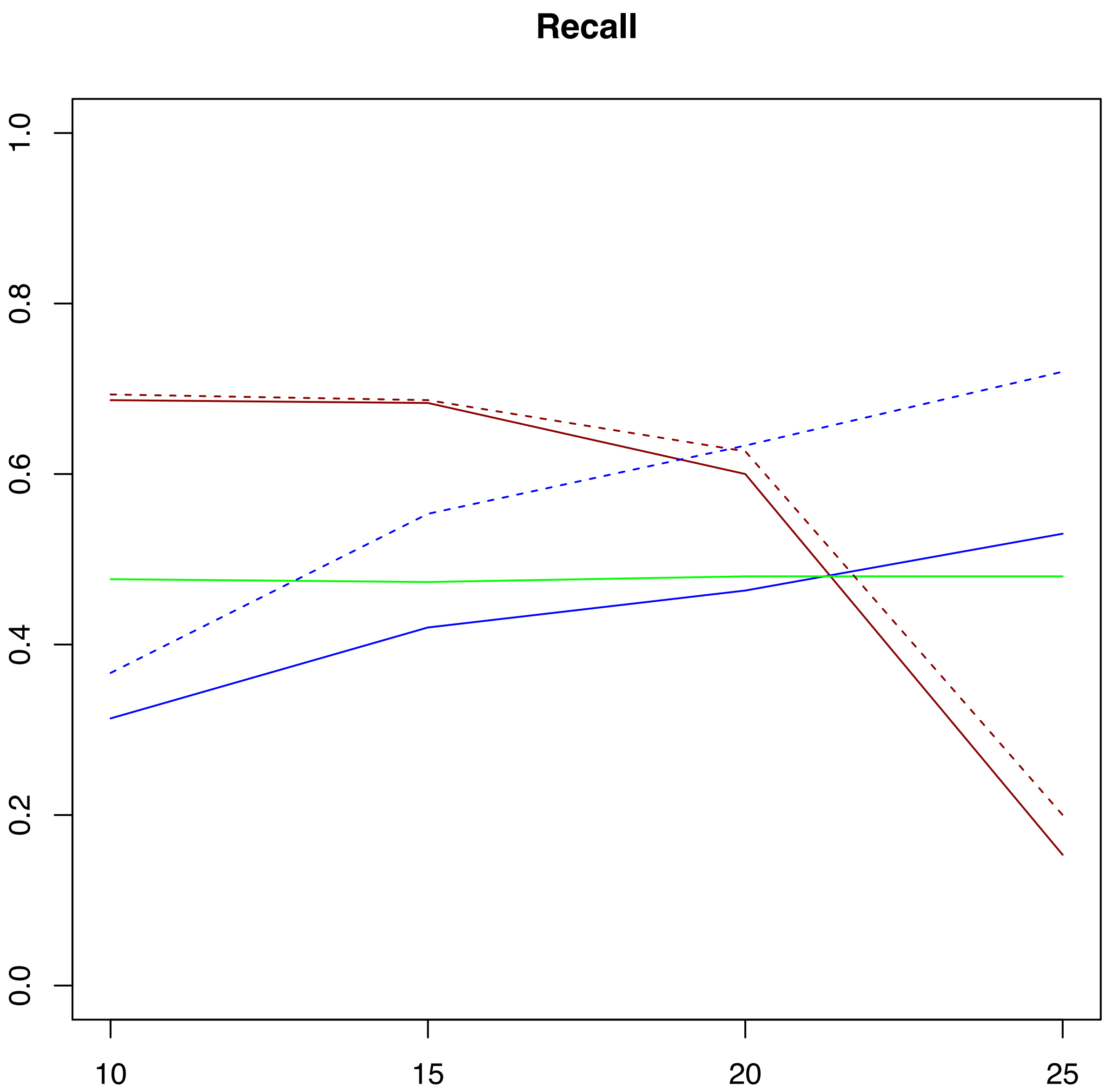
```
lines(x= sampNum, y= recallall[["DiffSplice"]],col="green",lty=1,pch=1)
```



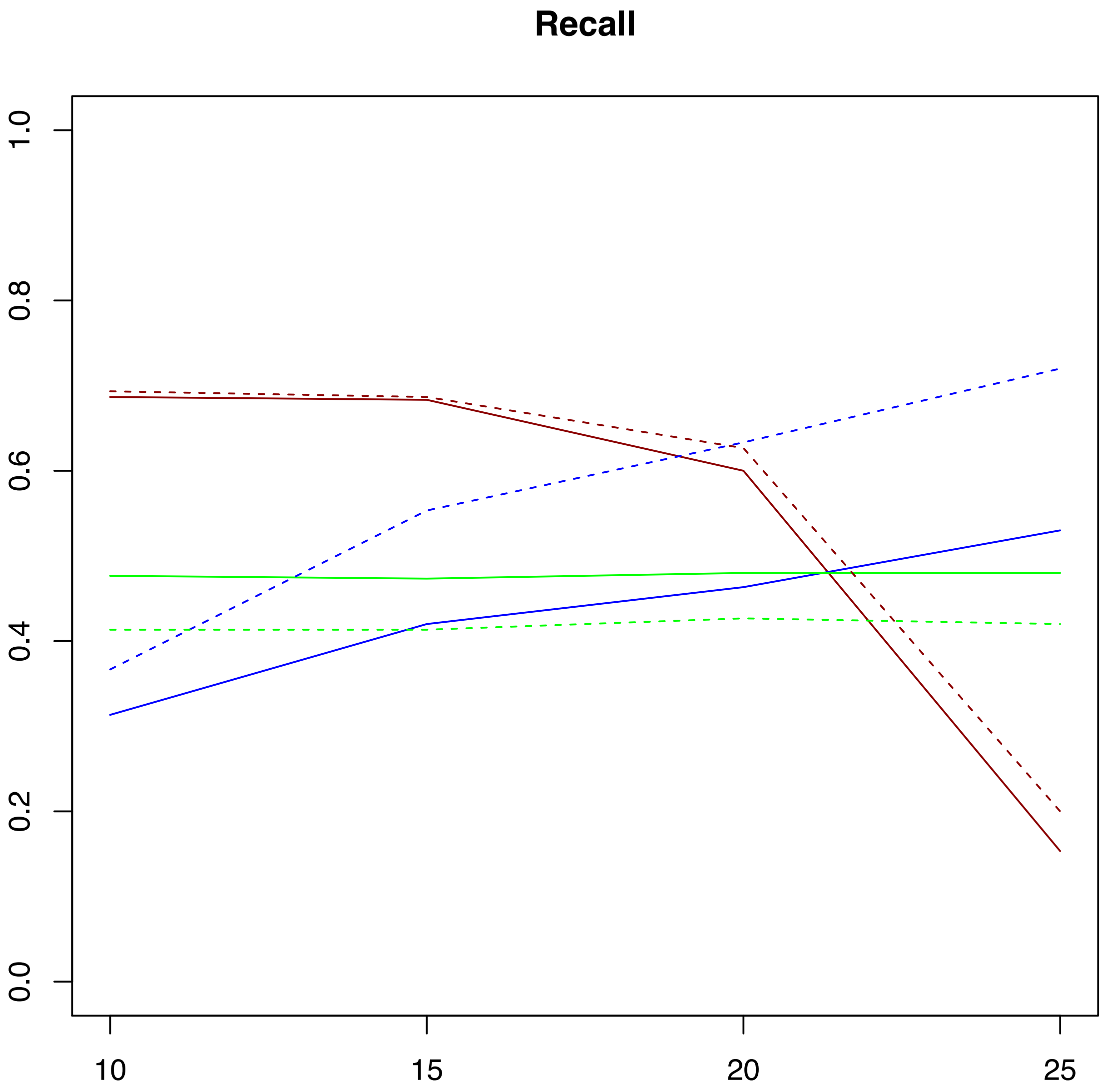
```
lines(x= sampNum, y= recallDEall[["SEVA"]],col="dark red",lty=2,pch=2)
```



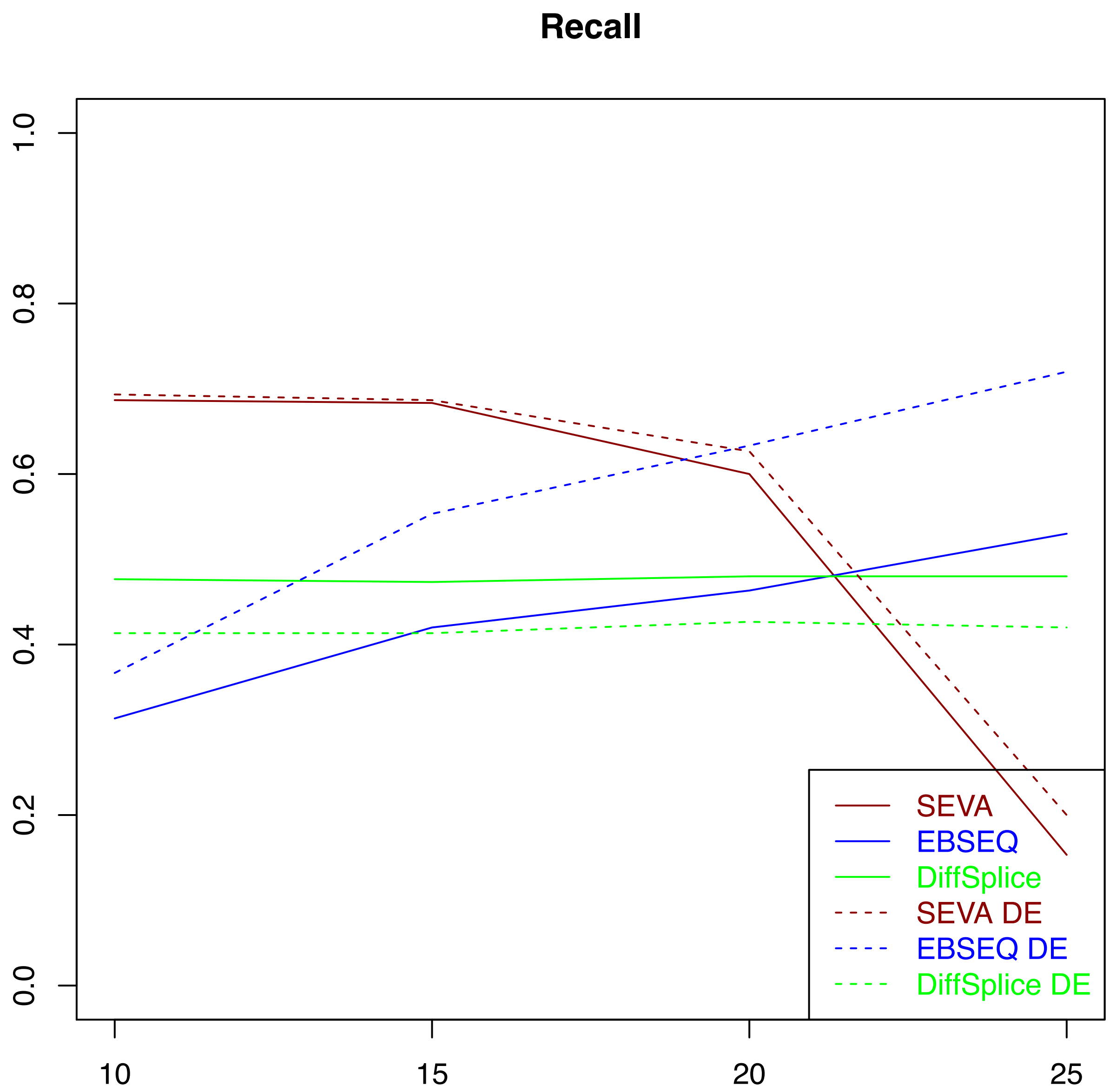
```
lines(x= sampNum, y= recallDEall[["EBSEQ"]],col="blue",lty=2,pch=2)
```



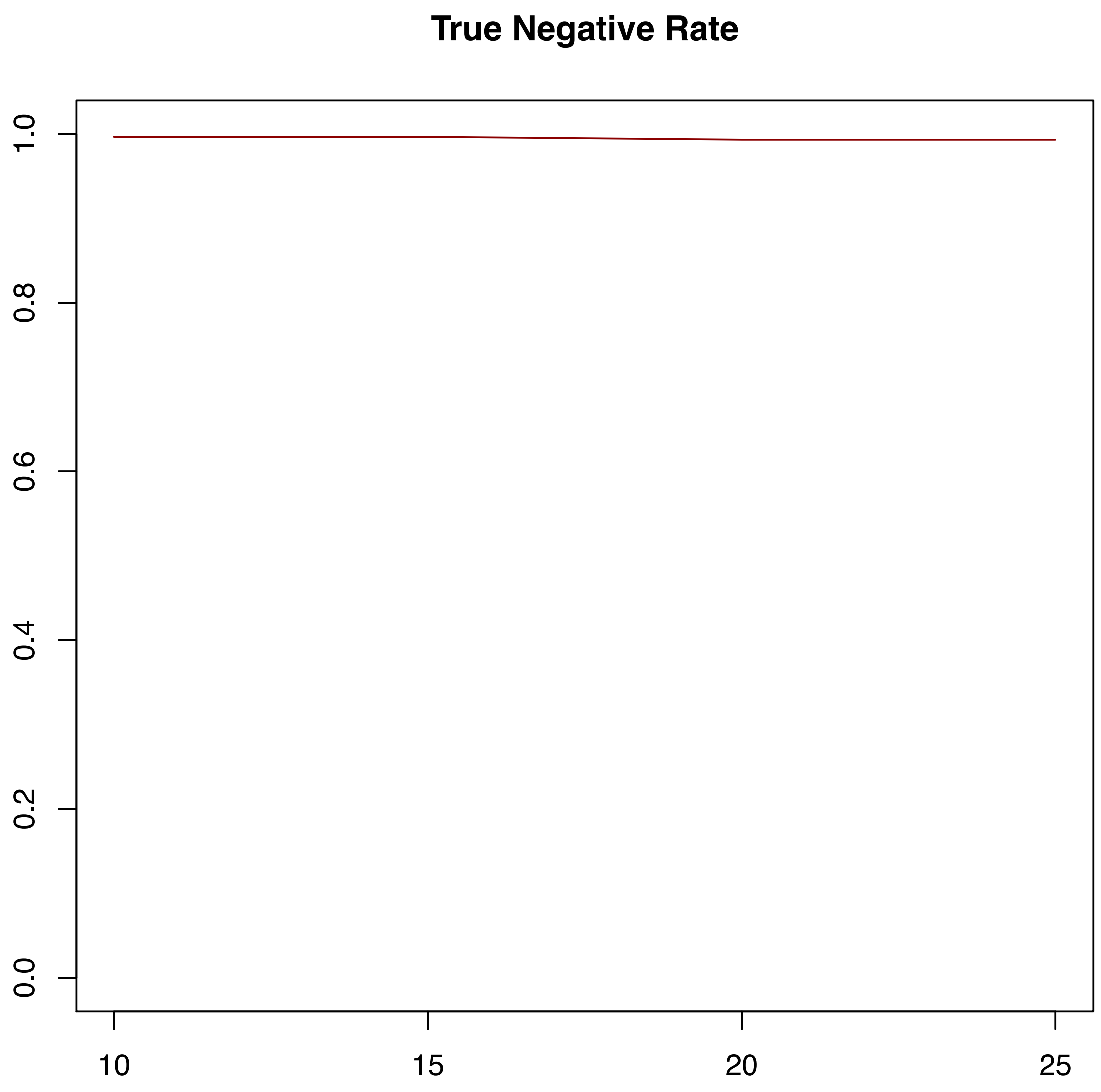
```
lines(x= sampNum, y= recallDEall[["DiffSplice"]],col="green",lty=2,pch=2)
```



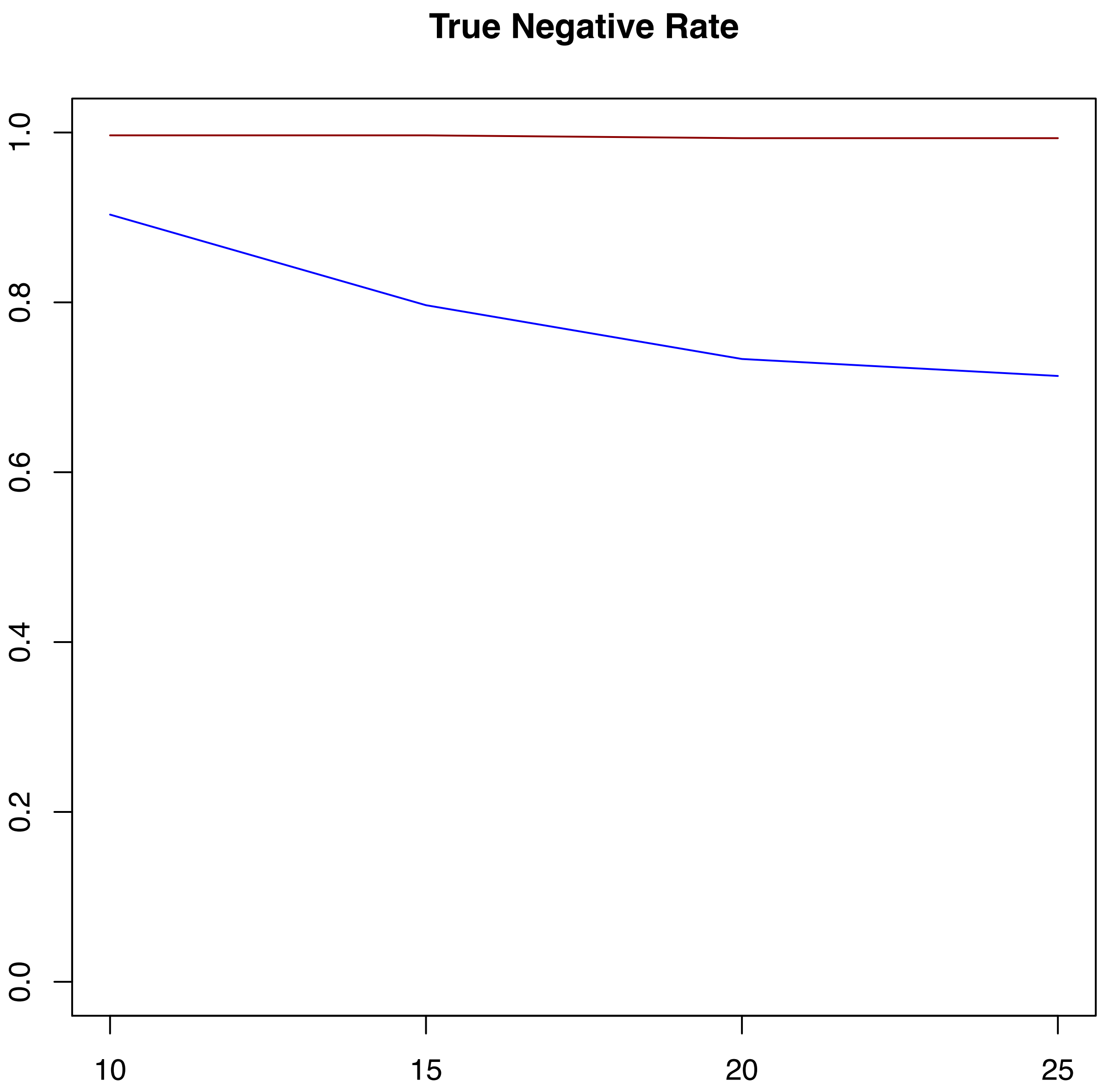
```
legend("bottomright",legend = c("SEVA","EBSEQ","DiffSplice","SEVA DE","EBSEQ DE","DiffSplice DE"),
       col=c("dark red","blue","green","dark red","blue","green"),
       text.col = c("dark red","blue","green","dark red","blue","green"),
       lty = c(1,1,1,2,2,2)     )
```



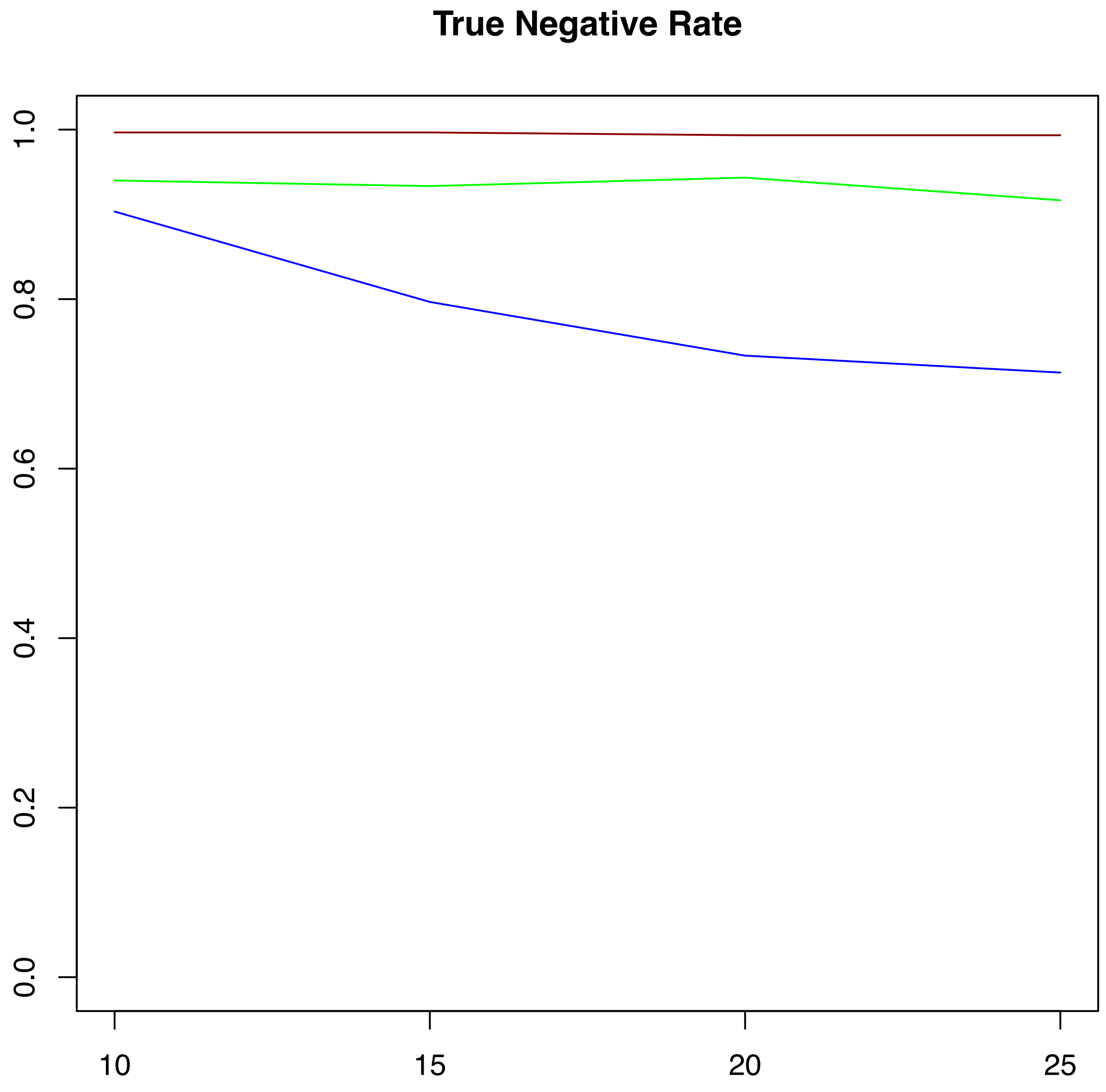
```
plot(x= sampNum, y= specall[["SEVA"]], xlab="", ylab= "", main = "True Negative Rate", type="l",col="da
```



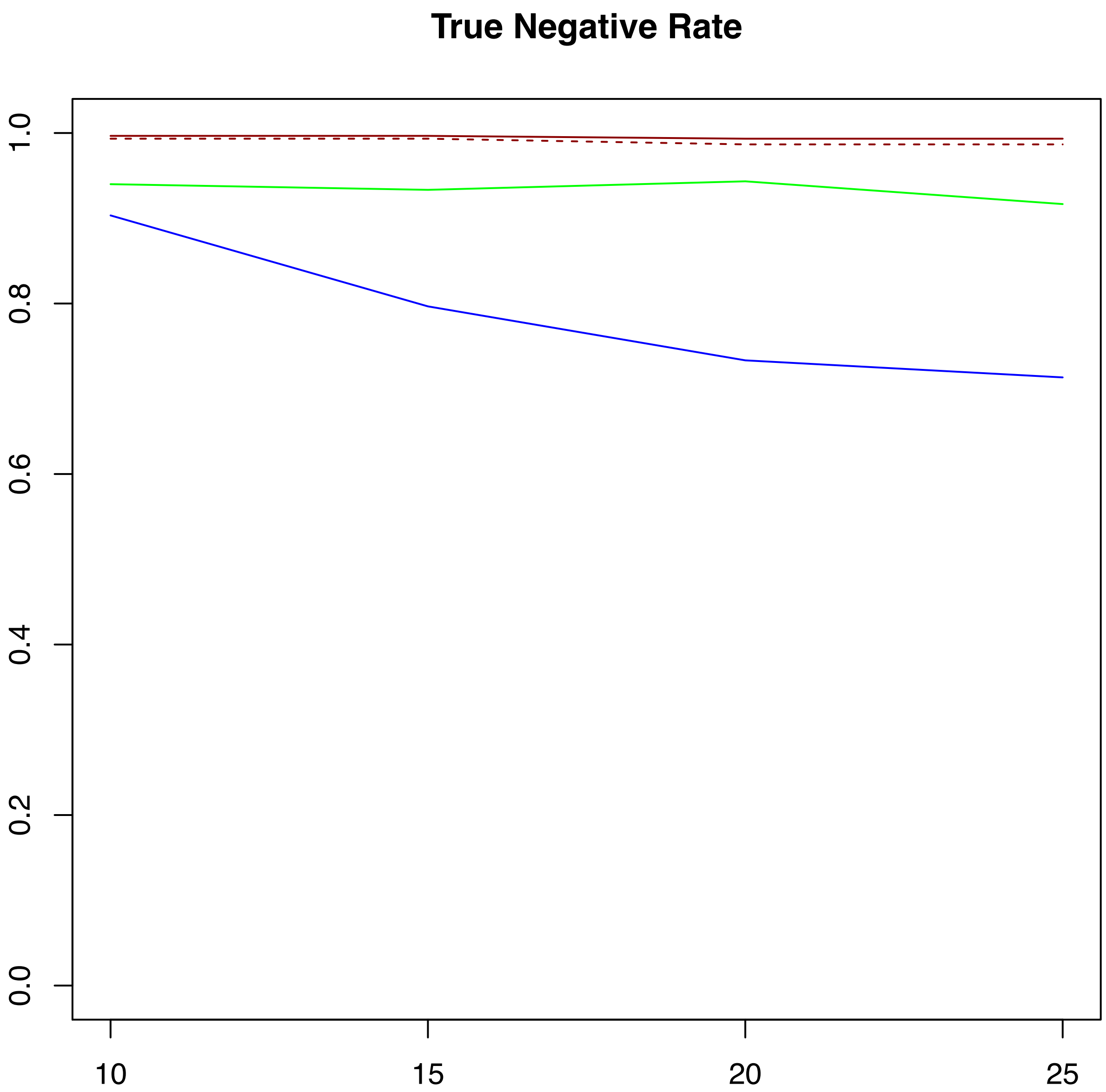
```
lines(x= sampNum, y= specall[["EBSEQ"]],col="blue",lty=1,pch=1)
```



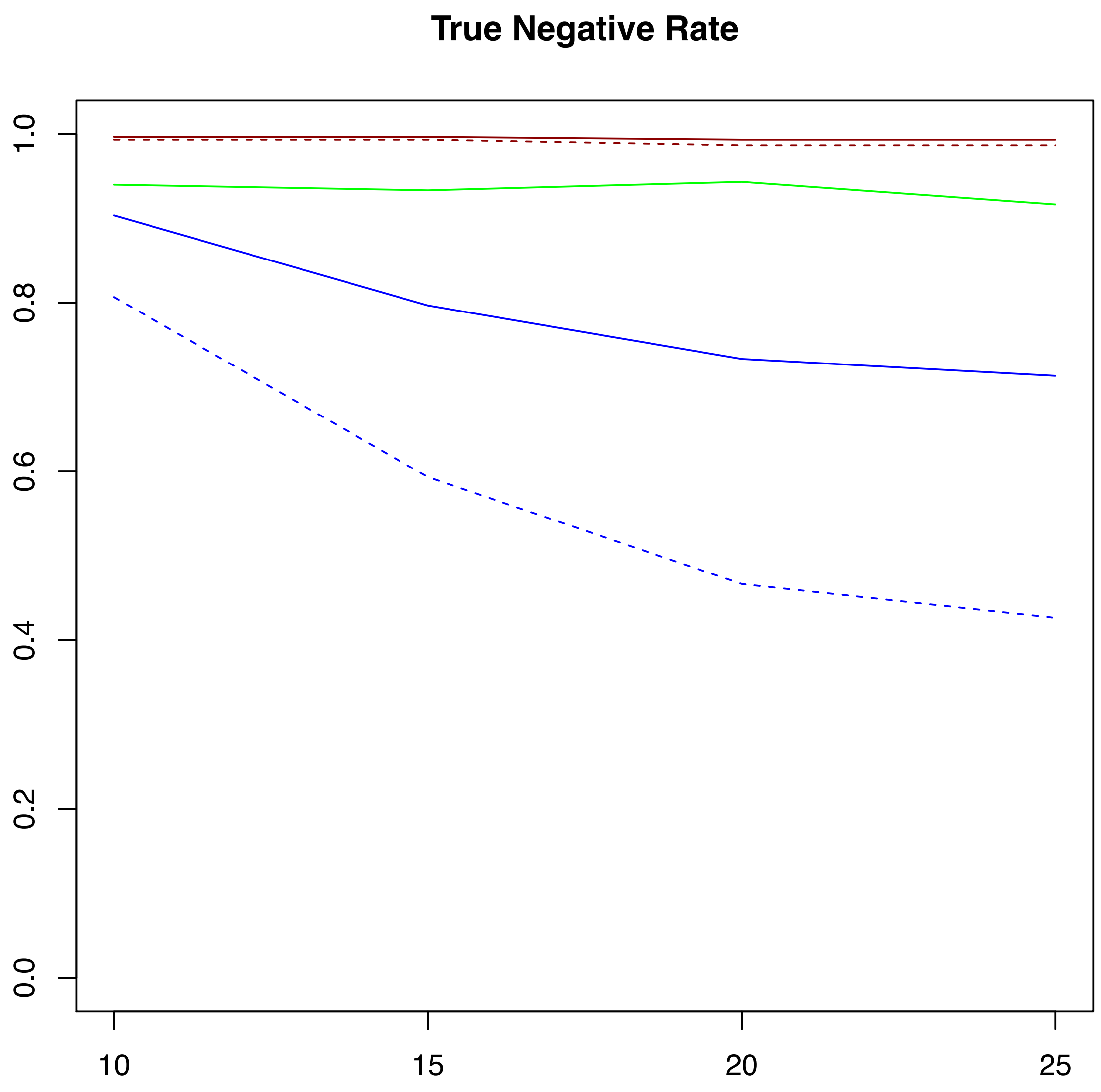
```
lines(x= sampNum, y= specall[["DiffSplice"]],col="green",lty=1,pch=1)
```



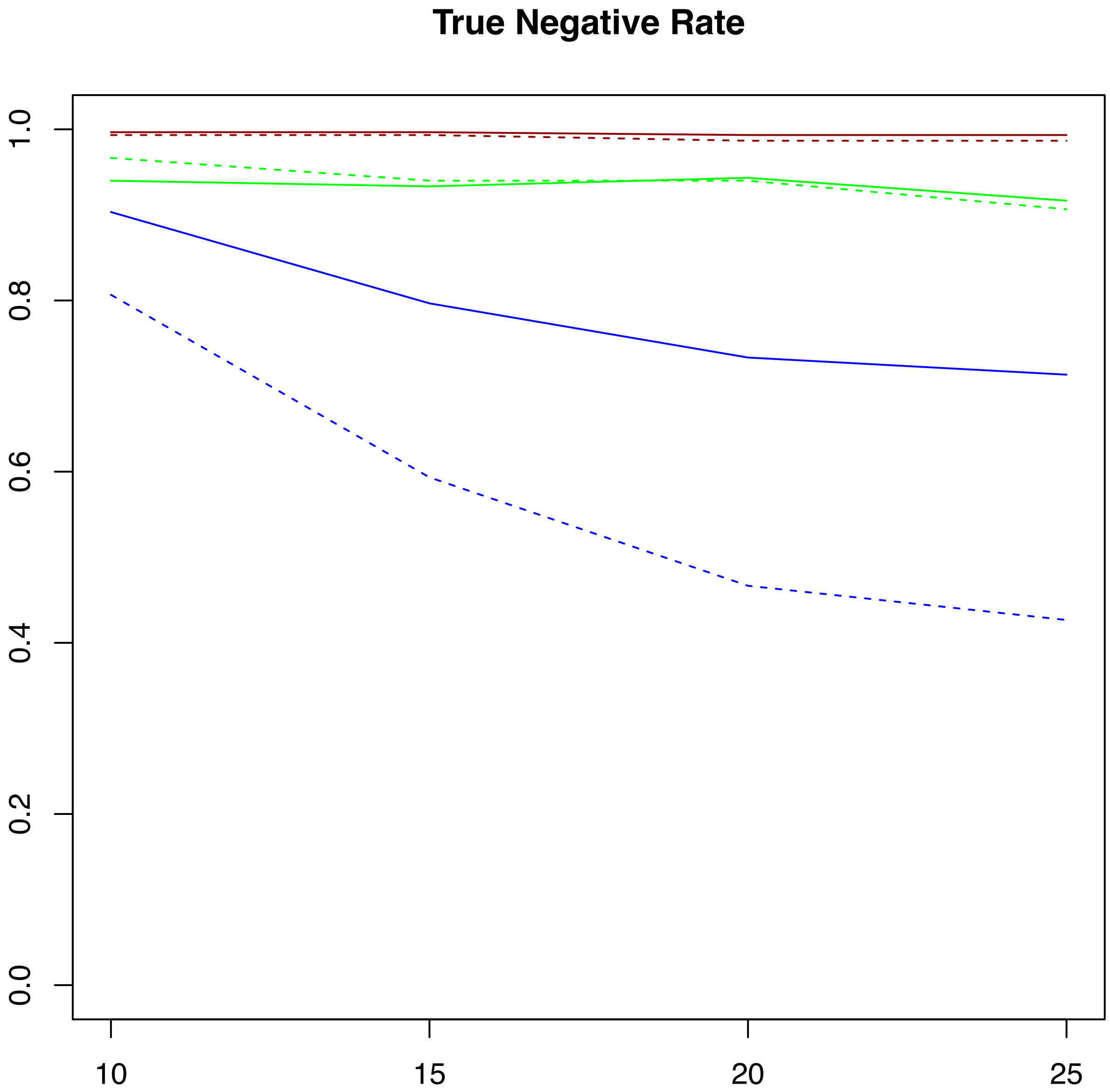
```r
lines(x= sampNum, y= specDEall[["SEVA"]],col="dark red",lty=2,pch=2)
```



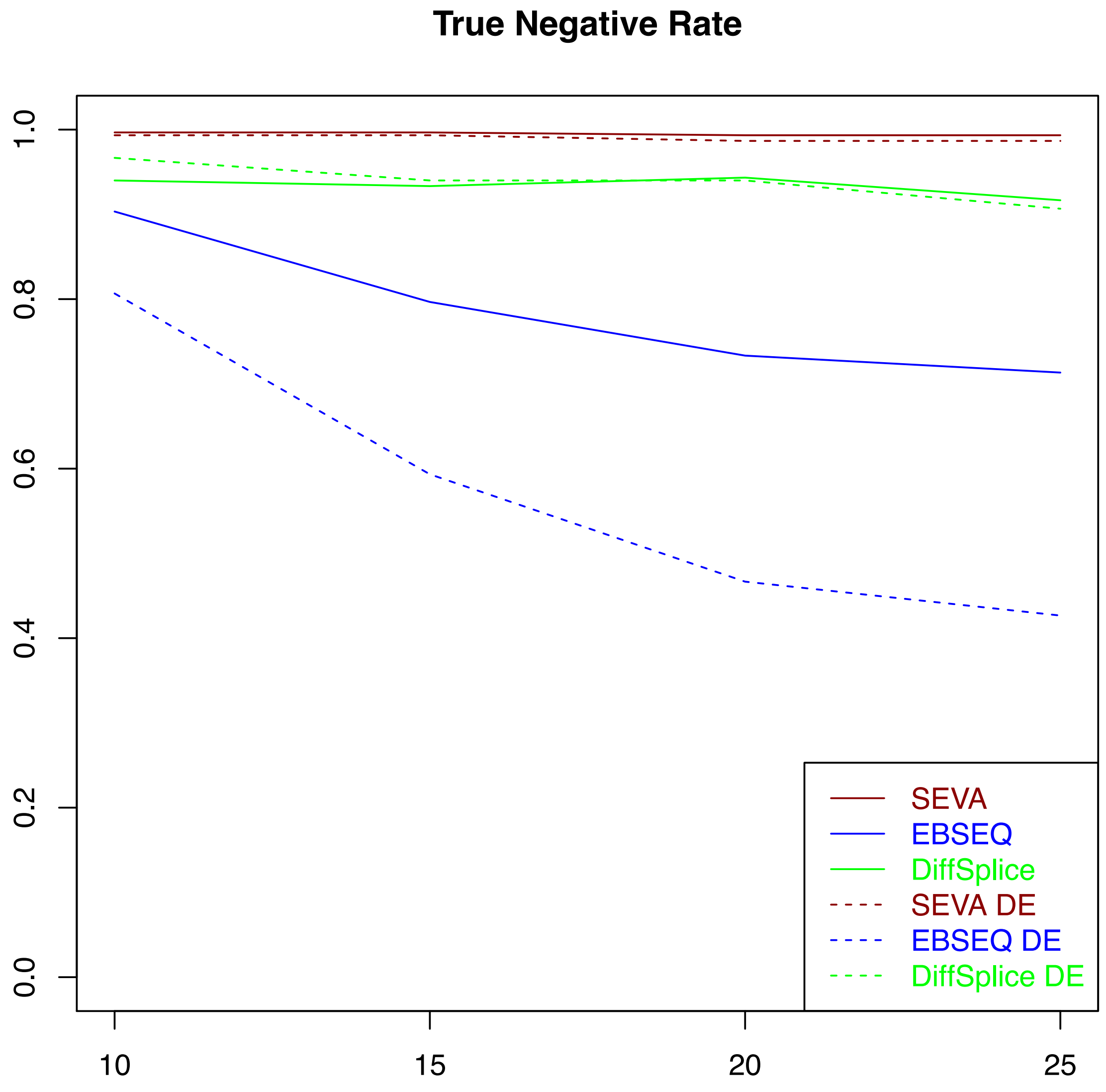
```r
lines(x= sampNum, y= specDEall[["EBSEQ"]],col="blue",lty=2,pch=2)
```



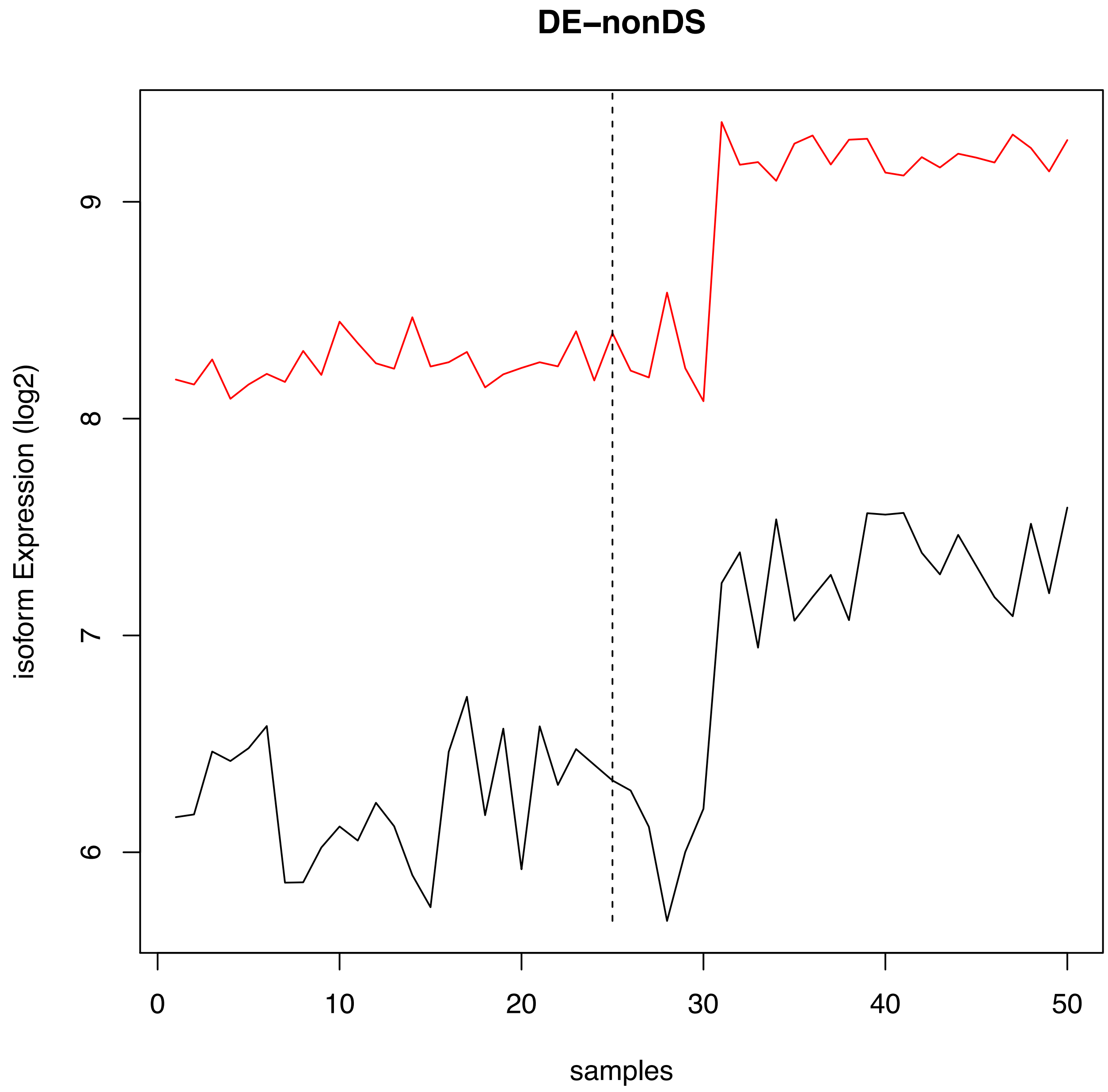
```
lines(x= sampNum, y= specDEall[["DiffSplice"]],col="green",lty=2,pch=2)
```

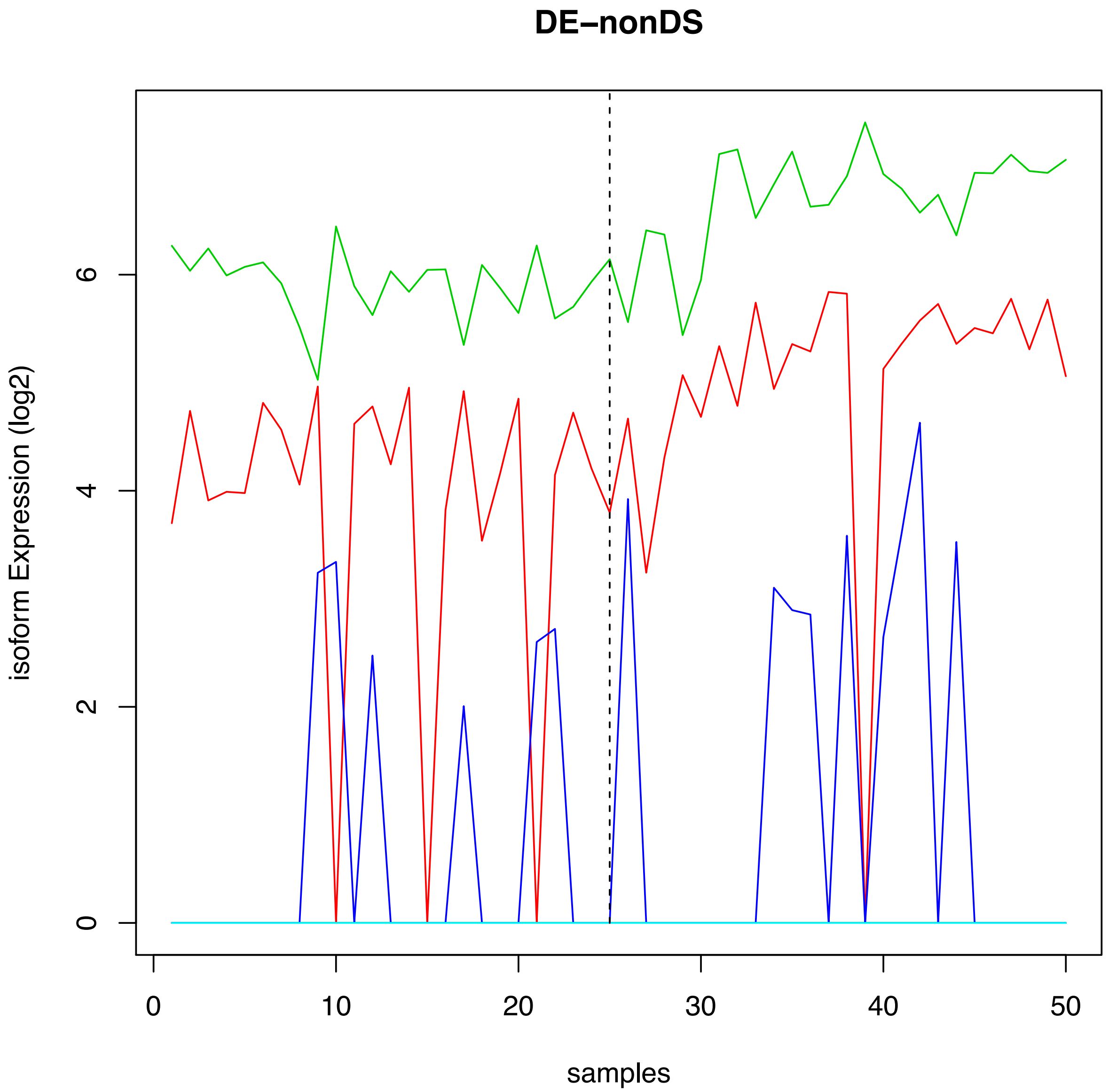


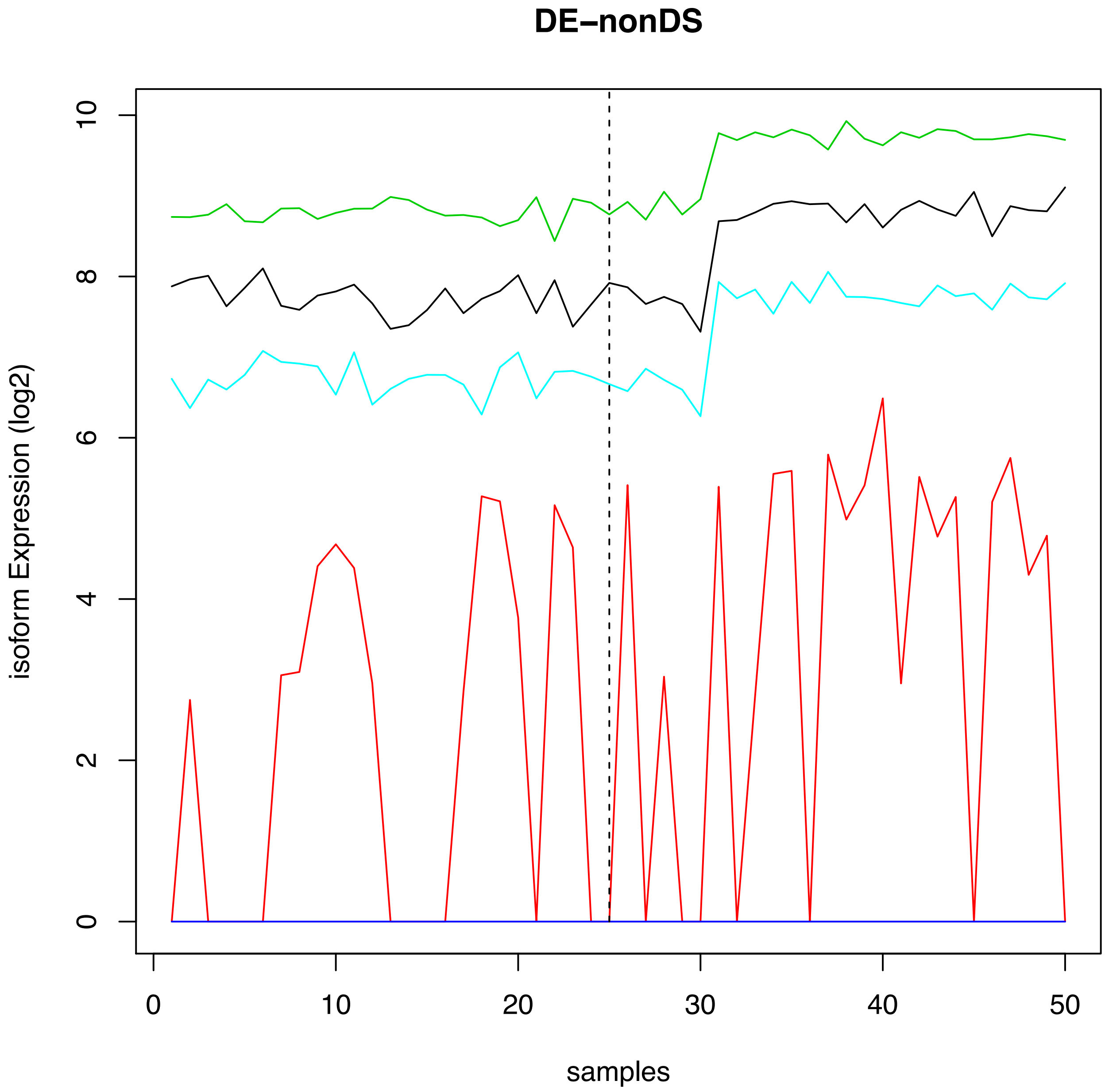
```
legend("bottomright",legend = c("SEVA","EBSEQ","DiffSplice","SEVA DE","EBSEQ DE","DiffSplice DE"),
       col=c("dark red","blue","green","dark red","blue","green"),
       text.col = c("dark red","blue","green","dark red","blue","green"),
       lty = c(1,1,1,2,2,2)      )
```

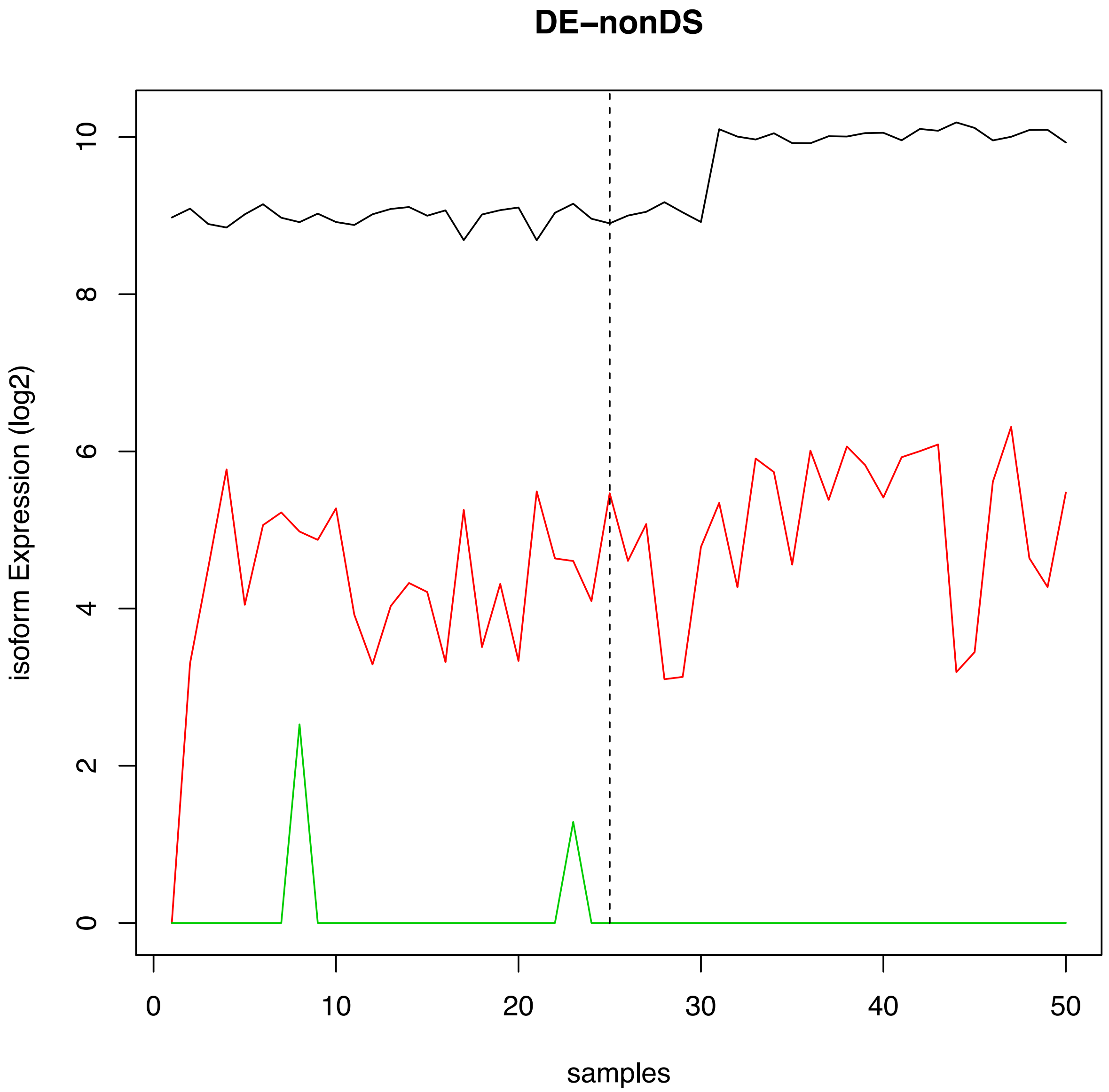


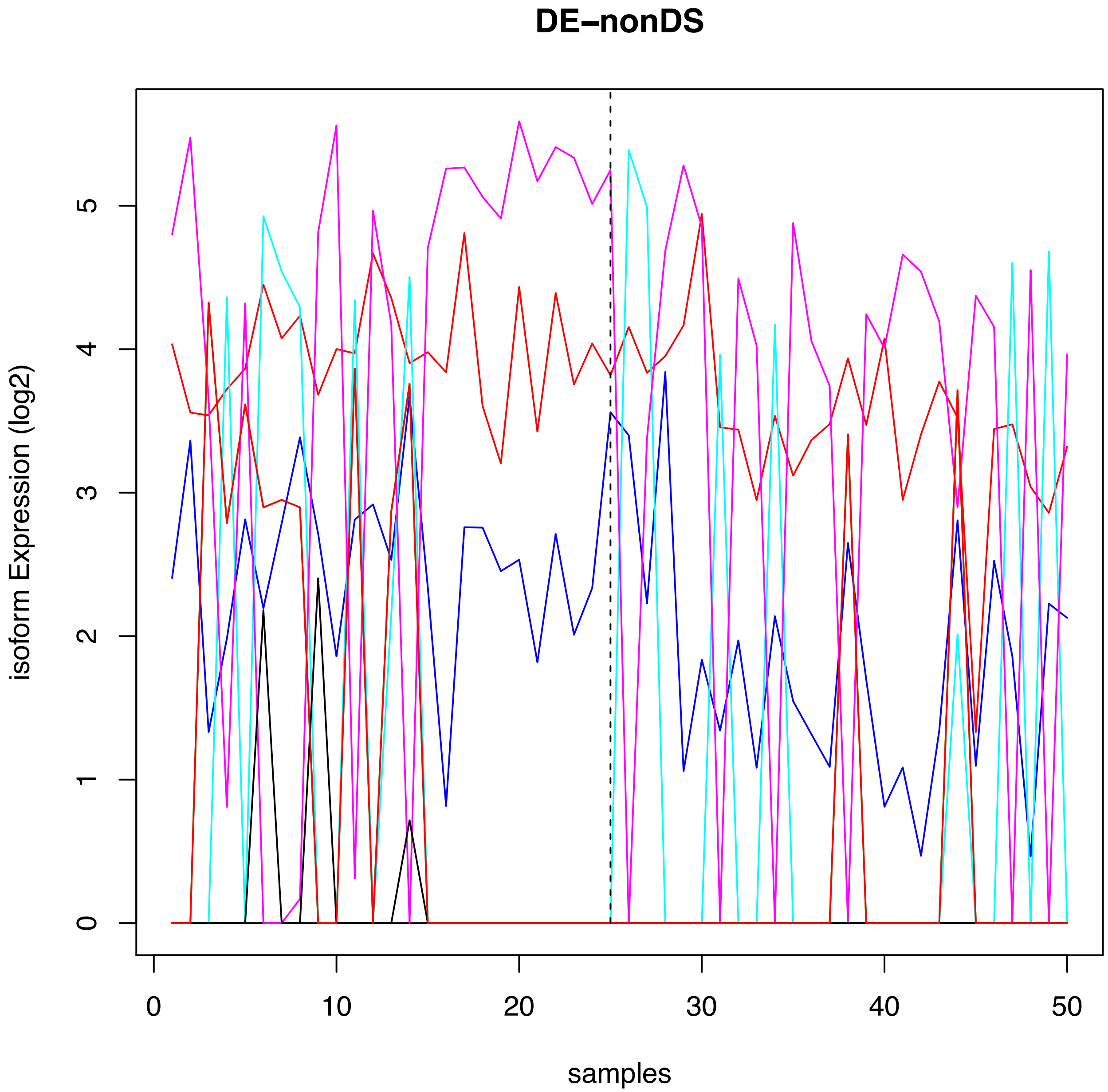
```
genesToVisualize <- c(names(which(DSStatusofDE==0))[1:10],
           names(which(DSStatusofDE==1))[1:10],
           names(which(DSStatus[setdiff(names(DSStatus),names(DSStatusofDE))]==0))[1:10],
           names(which(DSStatus[setdiff(names(DSStatus),names(DSStatusofDE))]==1))[1:10])

genesType <- c(rep("DE-nonDS",10),
               rep("DE-DS",10),
               rep("neutral",10),
               rep("nonDE-DS",10))

samplescur <- c(NormalLabels,experimentSamplesTumors[[3]])


for( i in seq_along(genesToVisualize)){
  matplot( log2(t(isoexprext[names(which(isos2genesvectsimplified==genesToVisualize[i])),samplescur])+1
           lty=1,type = "l",
           ylab = "isoform Expression (log2)",
           xlab="samples",
           #main=paste(genesToVisualize[i]," (",genesType[i],")",sep = ""))
           main=genesType[i])
  lines(x=c(25,25),y=range(log2(t(isoexprext[names(which(isos2genesvectsimplified==genesToVisualize[i]))

}
```

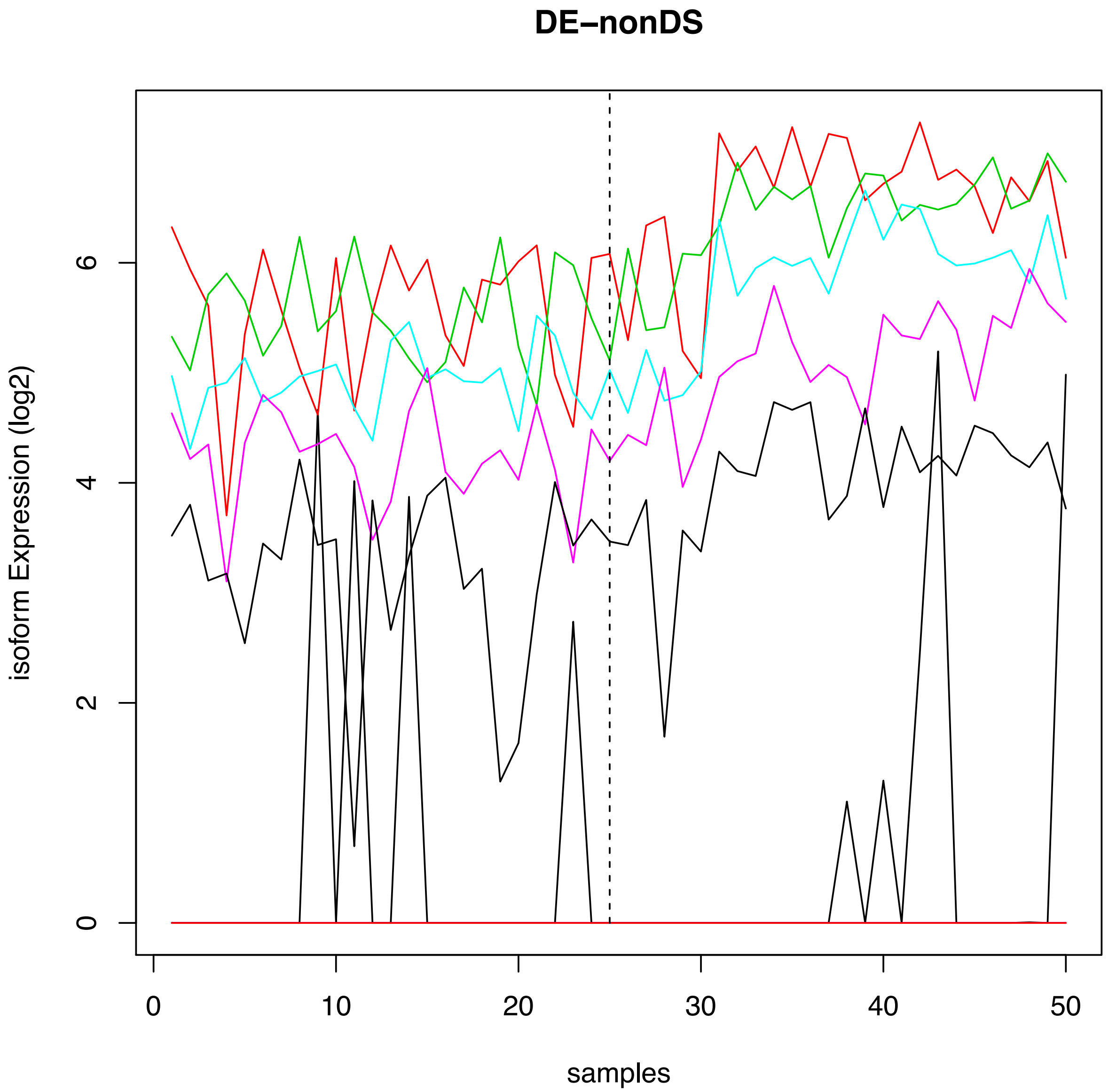


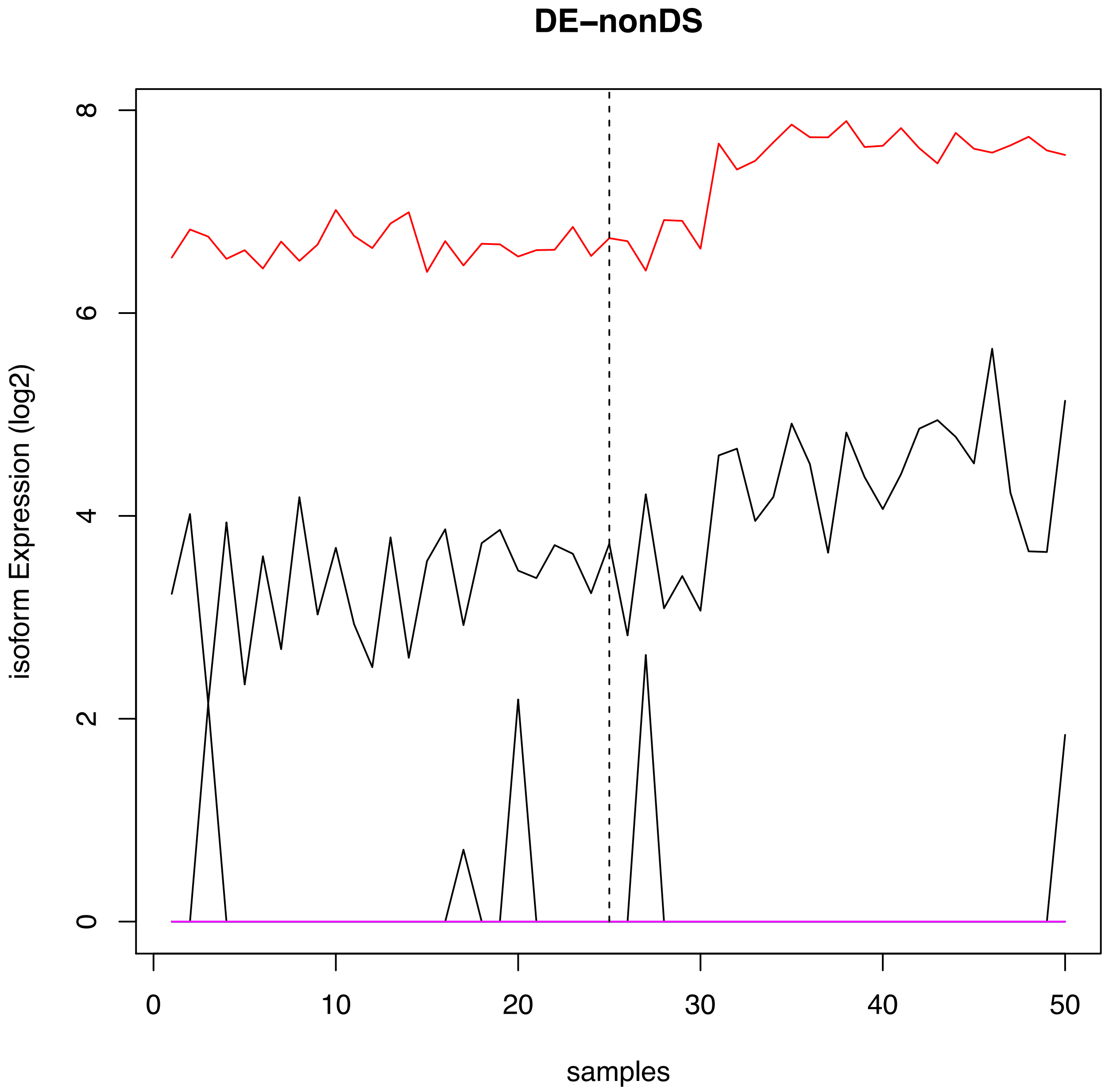

DE−nonDS
isoform Expression (log2)
0
2
4
6
0
10
20
30
40
50
samples

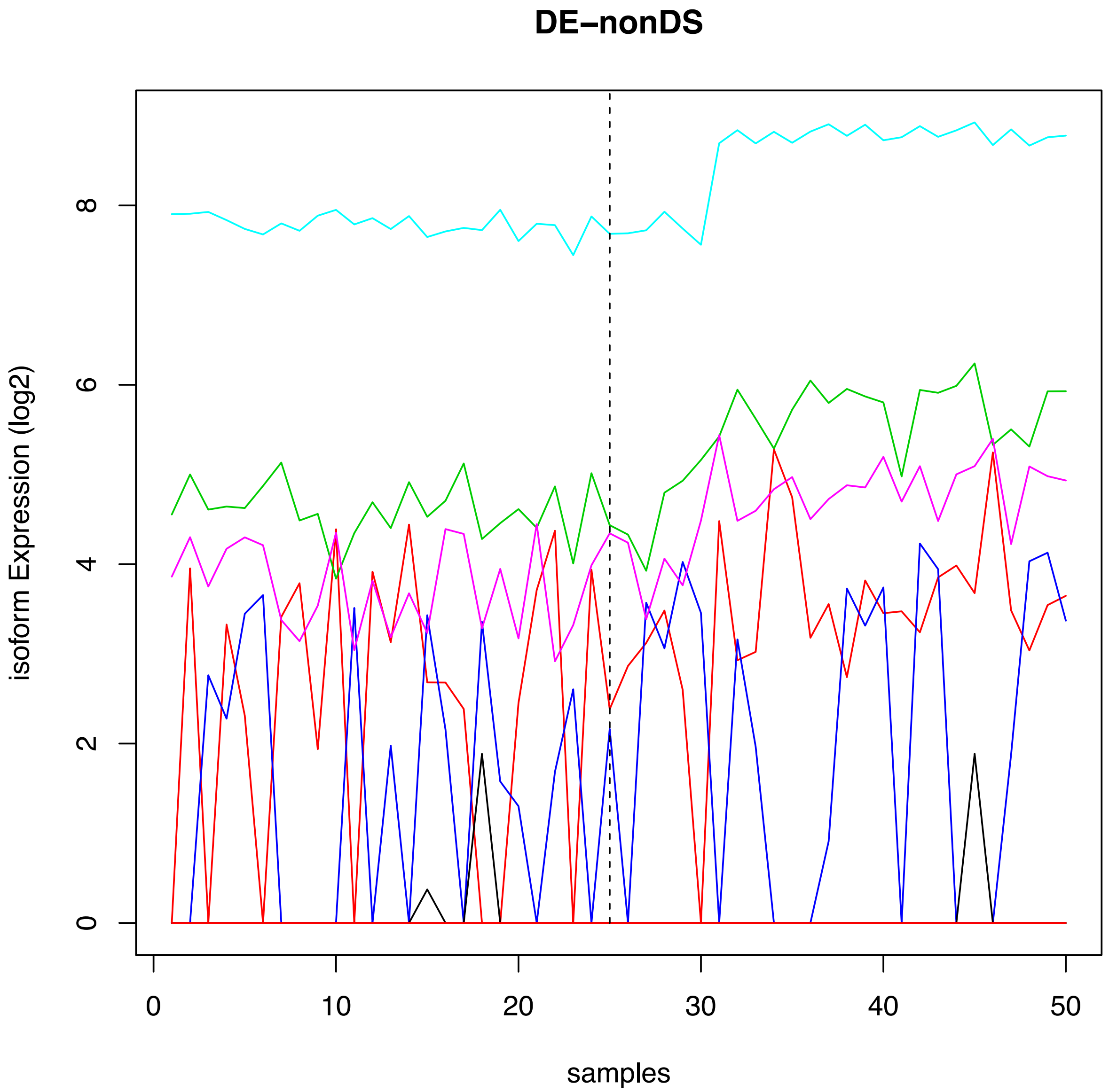

DE−nonDS
isoform Expression (log2)
0
2
4
6
8
10
samples
0
10
20
30
40
50

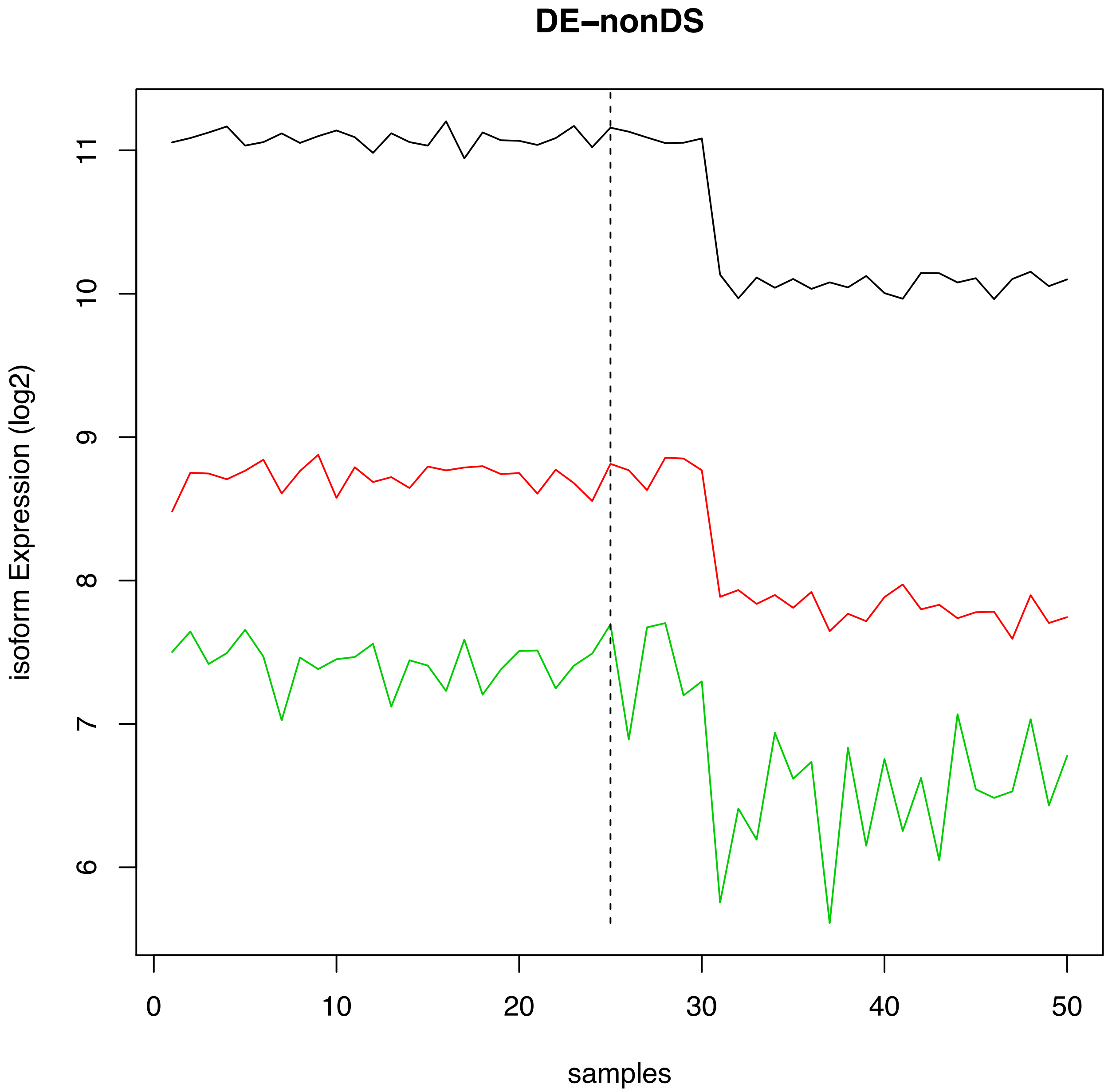

DE−nonDS
isoform Expression (log2)
0
2
4
6
8
10
samples
0
10
20
30
40
50

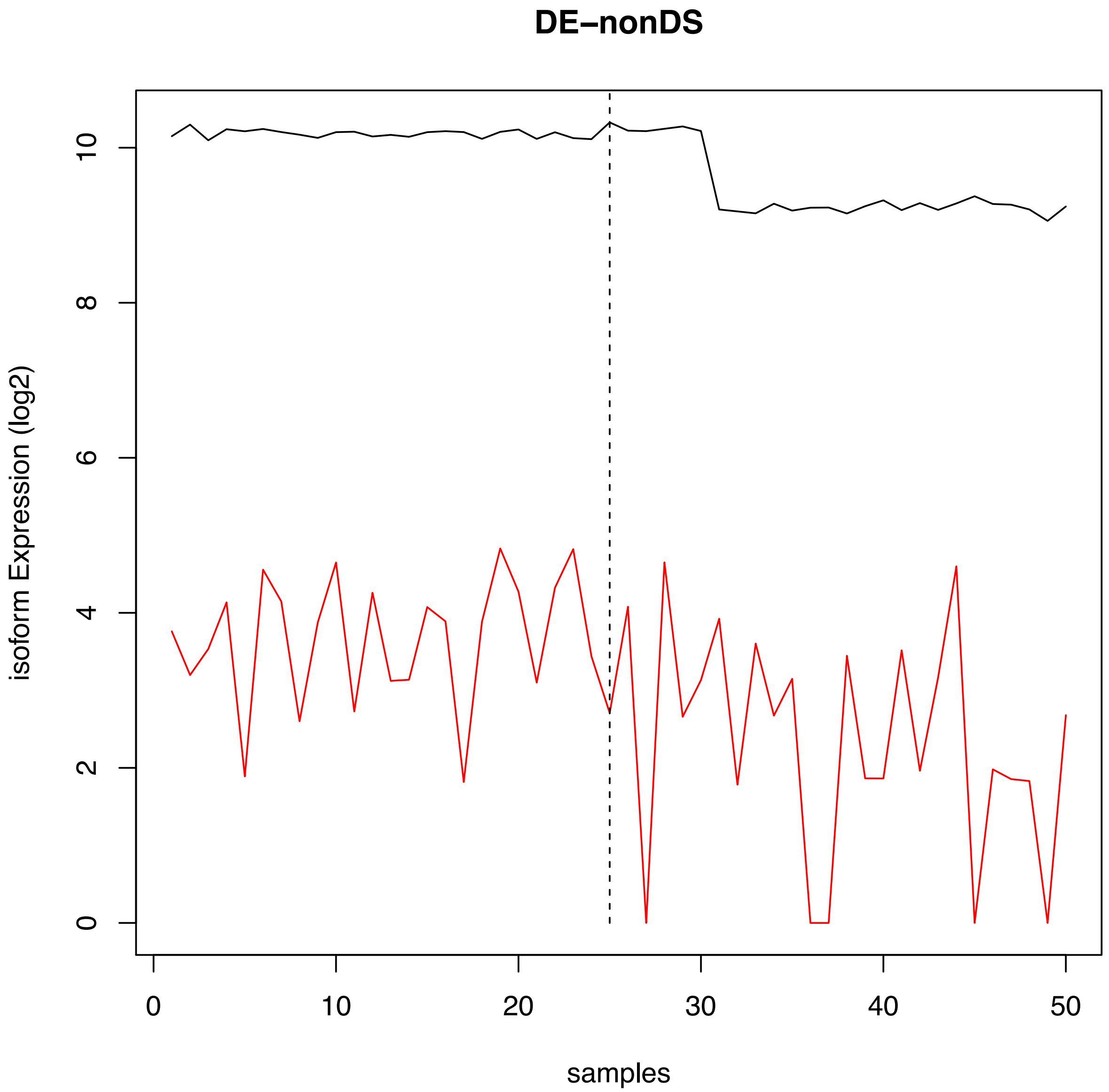

DE–nonDS
isoform Expression (log2)
0
1
2
3
4
5
0
10
20
30
40
50
samples

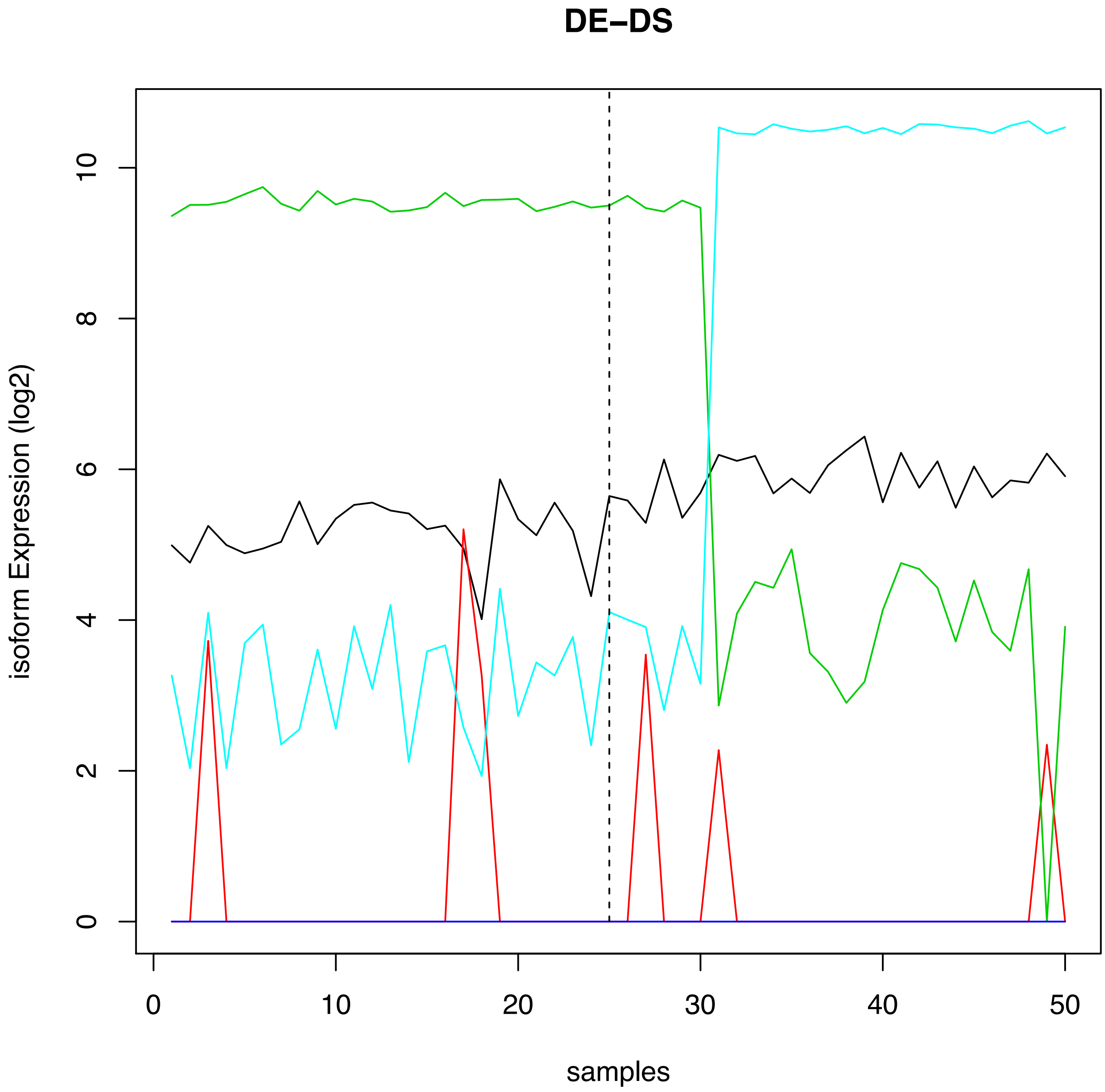

DE−nonDS
isoform Expression (log2)
0
2
4
6
0
10
20
30
40
50
samples

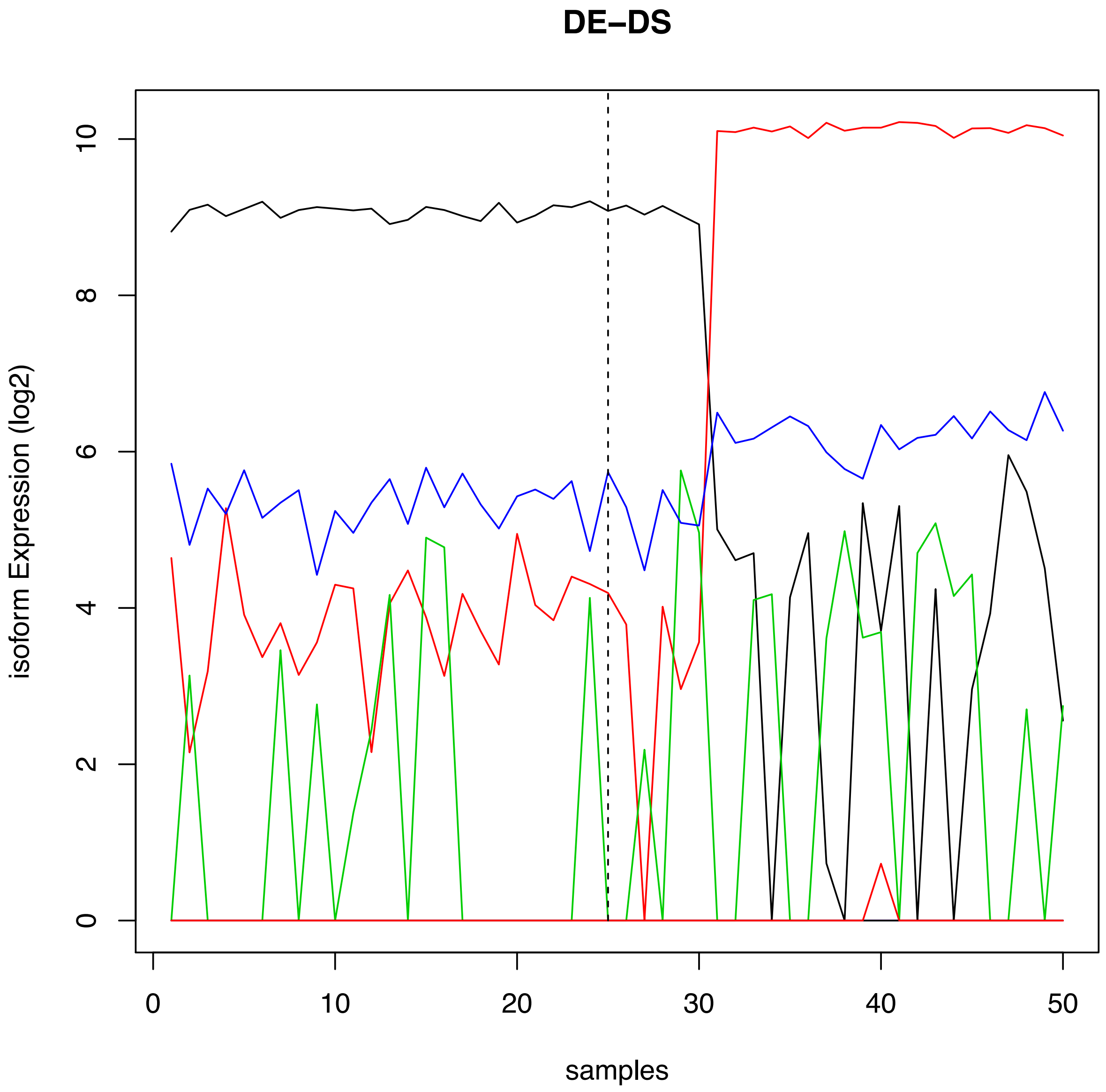

DE−nonDS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

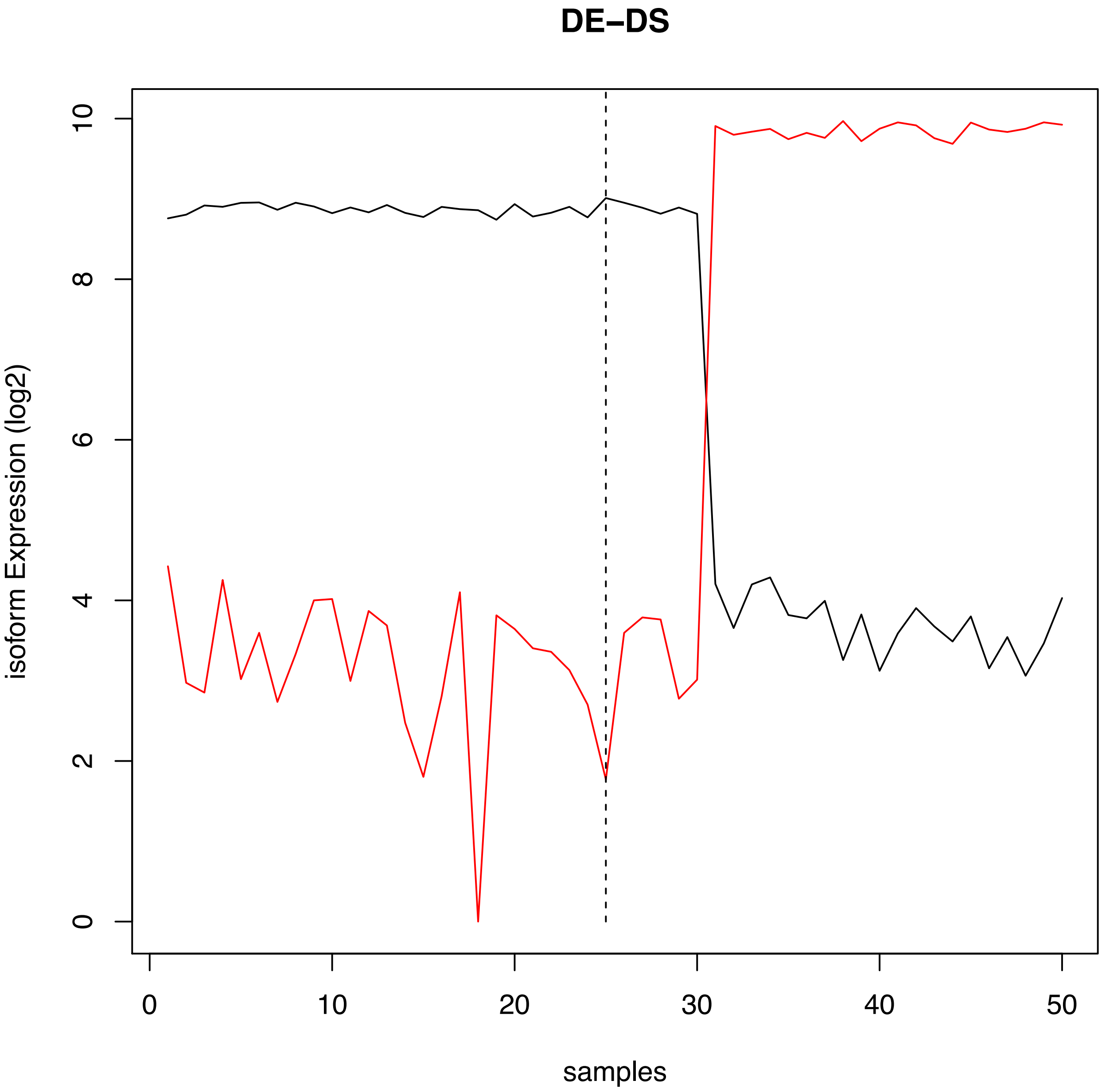

DE−nonDS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

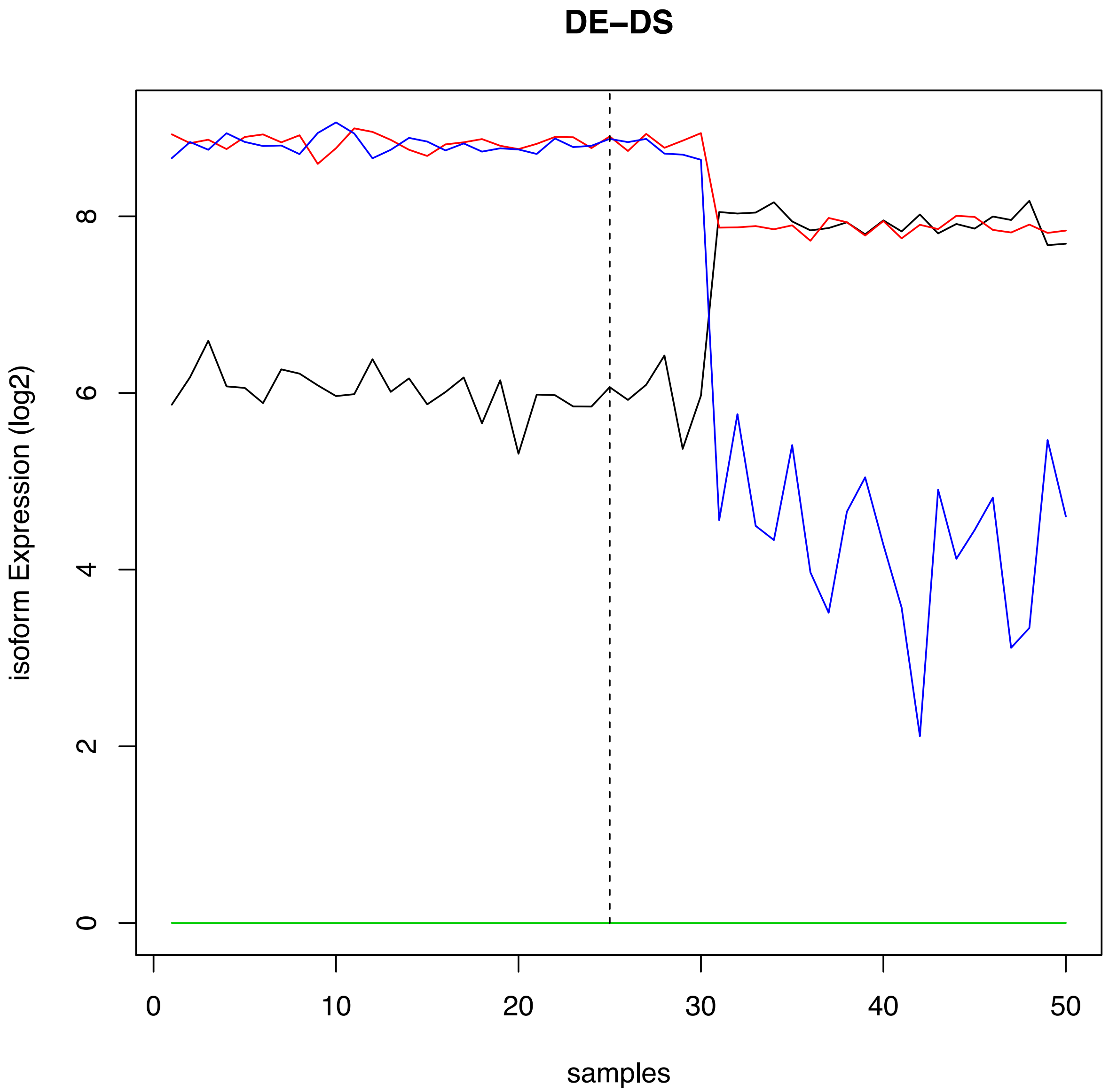

DE–nonDS
isoform Expression (log2)
11
10
9
8
7
6
0
10
20
30
40
50
samples

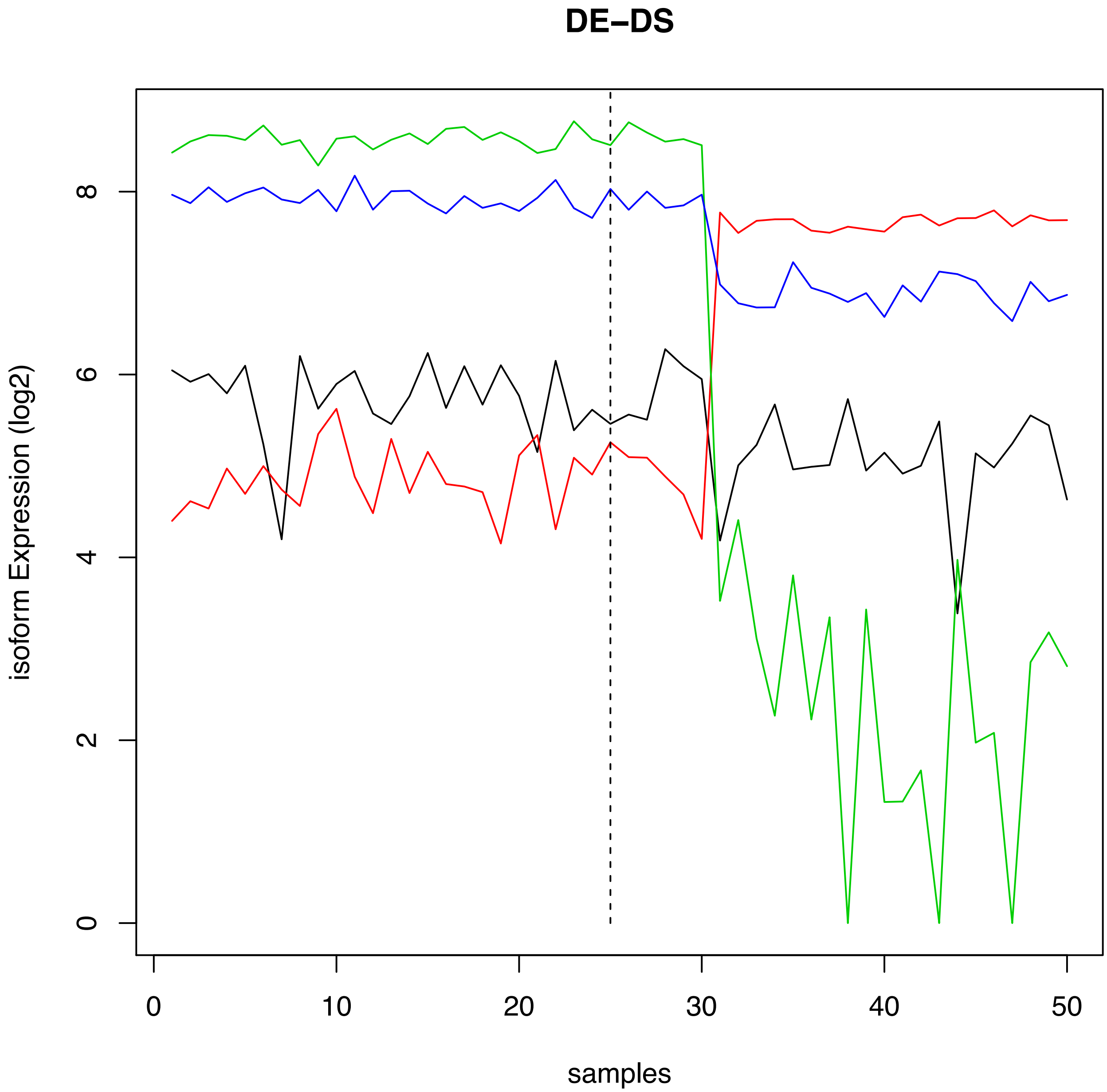

DE−nonDS
isoform Expression (log2)
0
2
4
6
8
10
0
10
20
30
40
50
samples

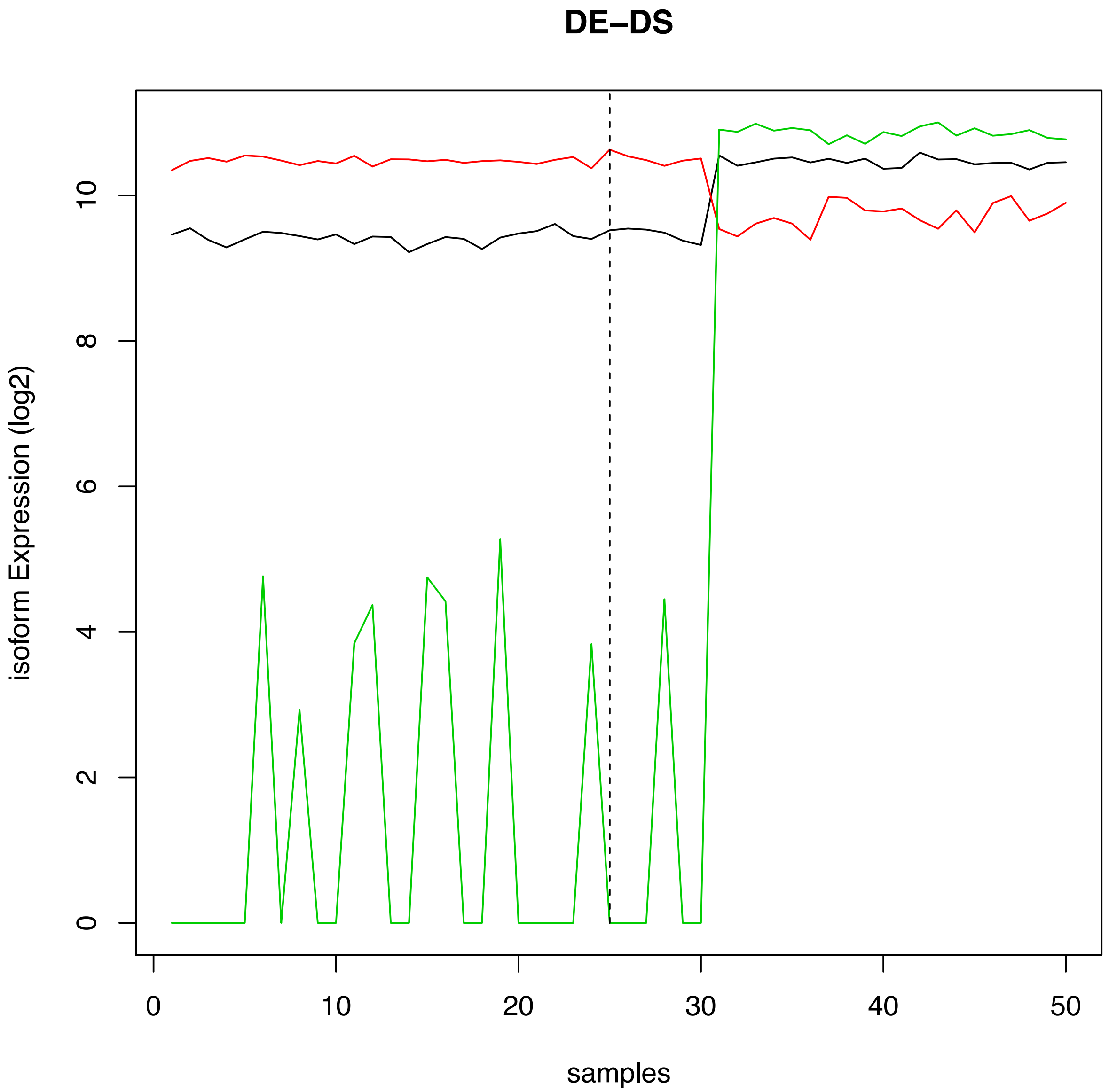

DE–DS
isoform Expression (log2)
0
2
4
6
8
10
samples
0
10
20
30
40
50

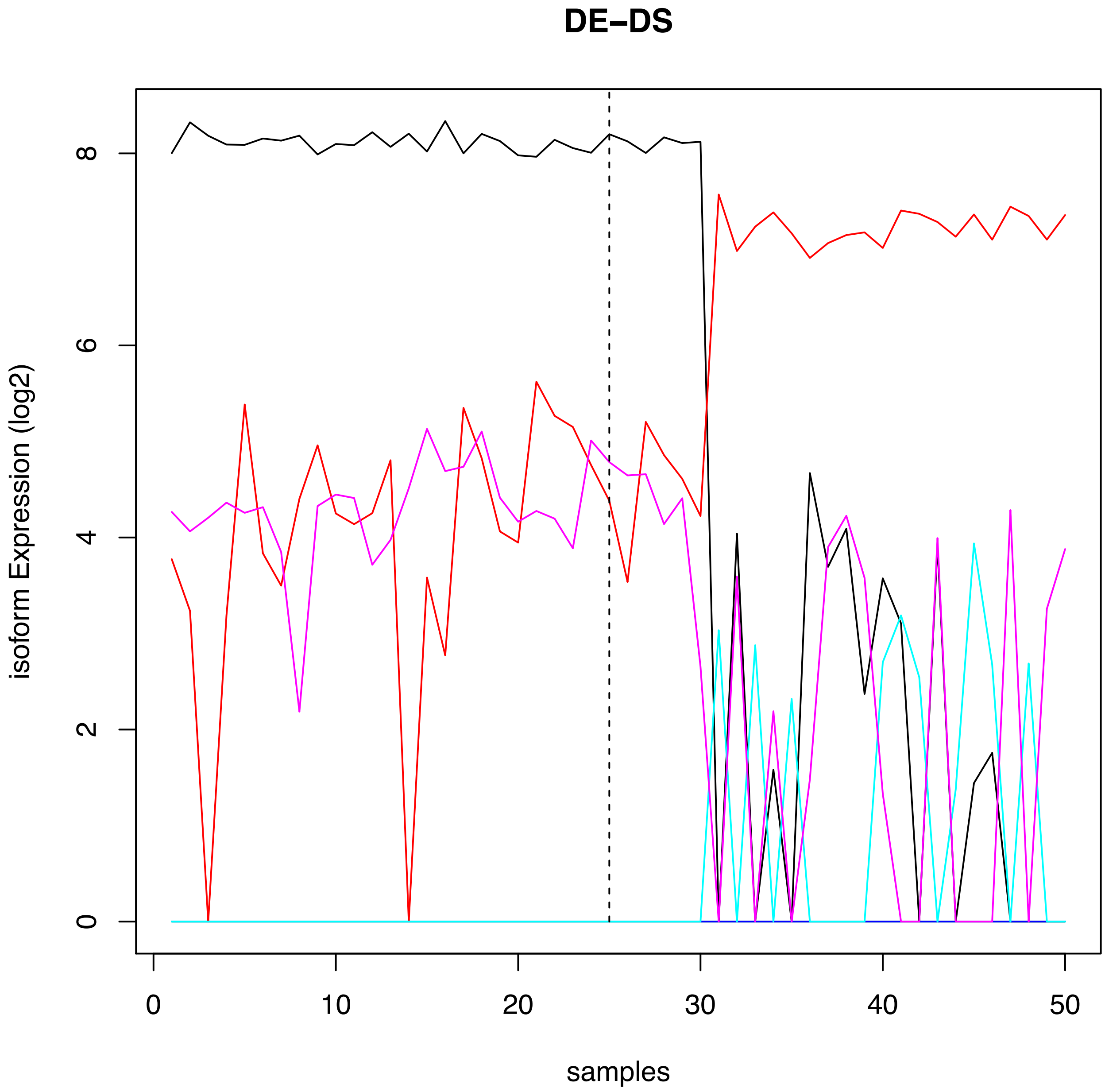

DE–DS
isoform Expression (log2)
0
2
4
6
8
10
samples
0
10
20
30
40
50

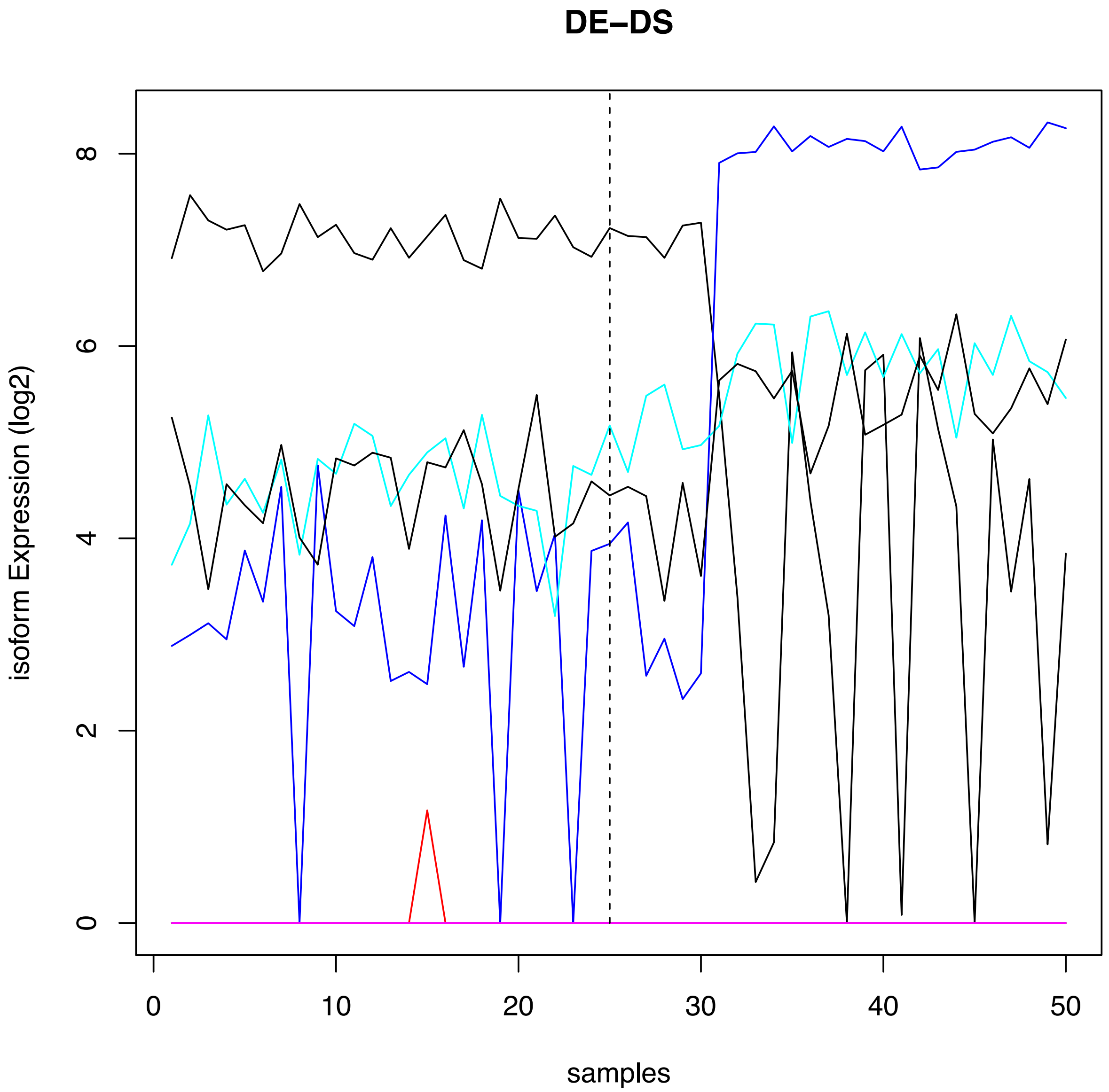

DE–DS
isoform Expression (log2)
0
2
4
6
8
10
0
10
20
30
40
50
samples

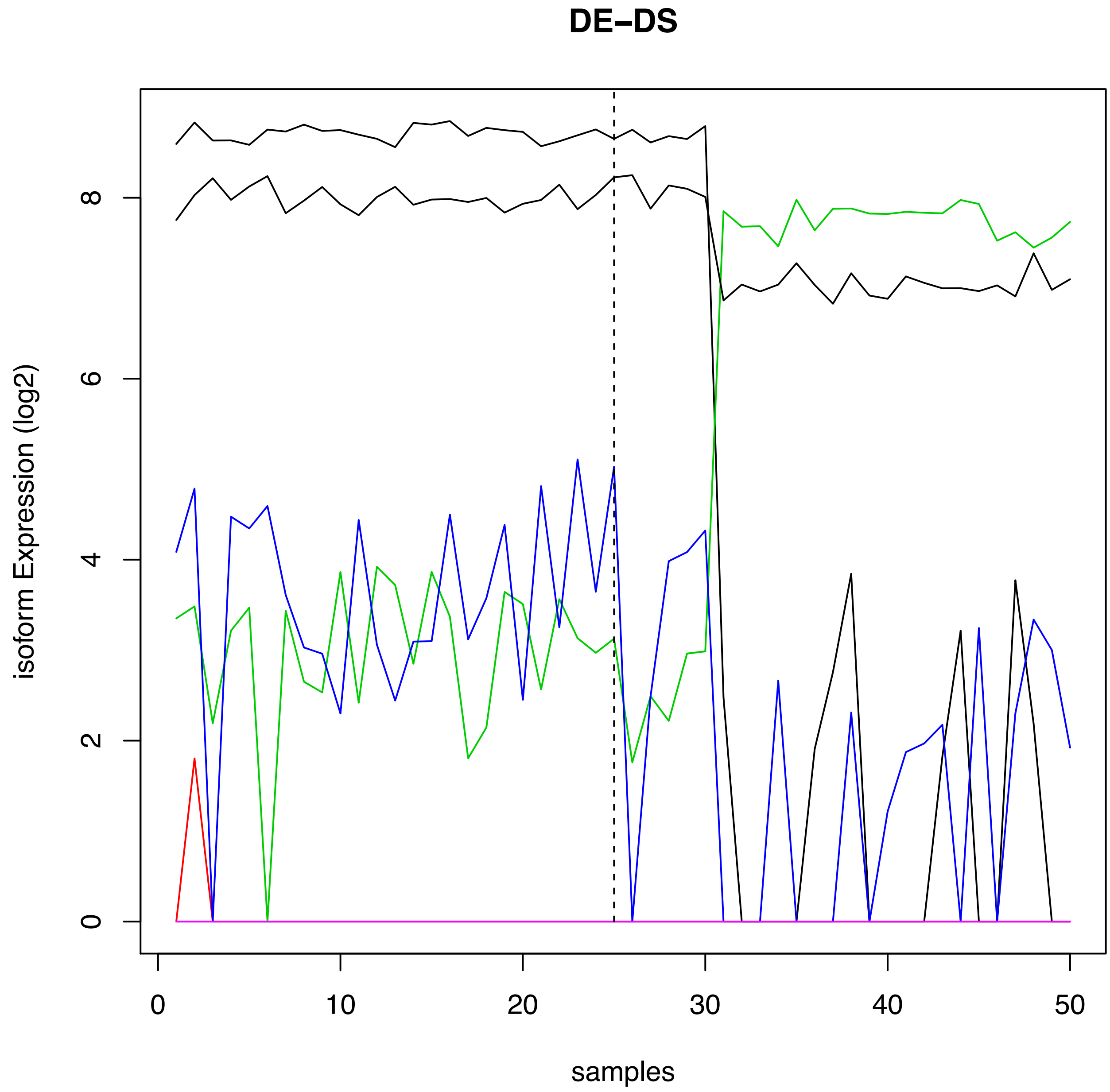

DE–DS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

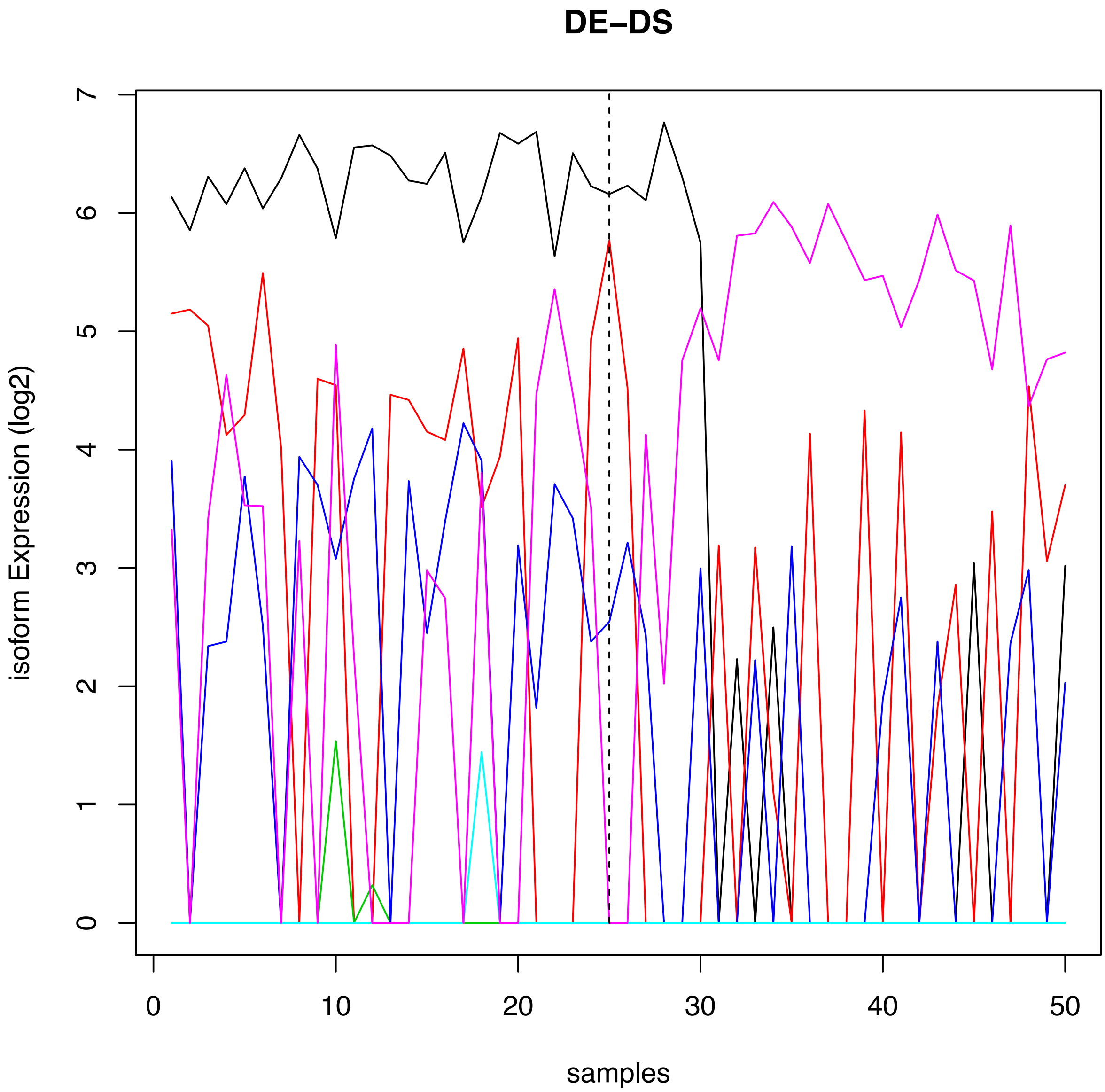

DE−DS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

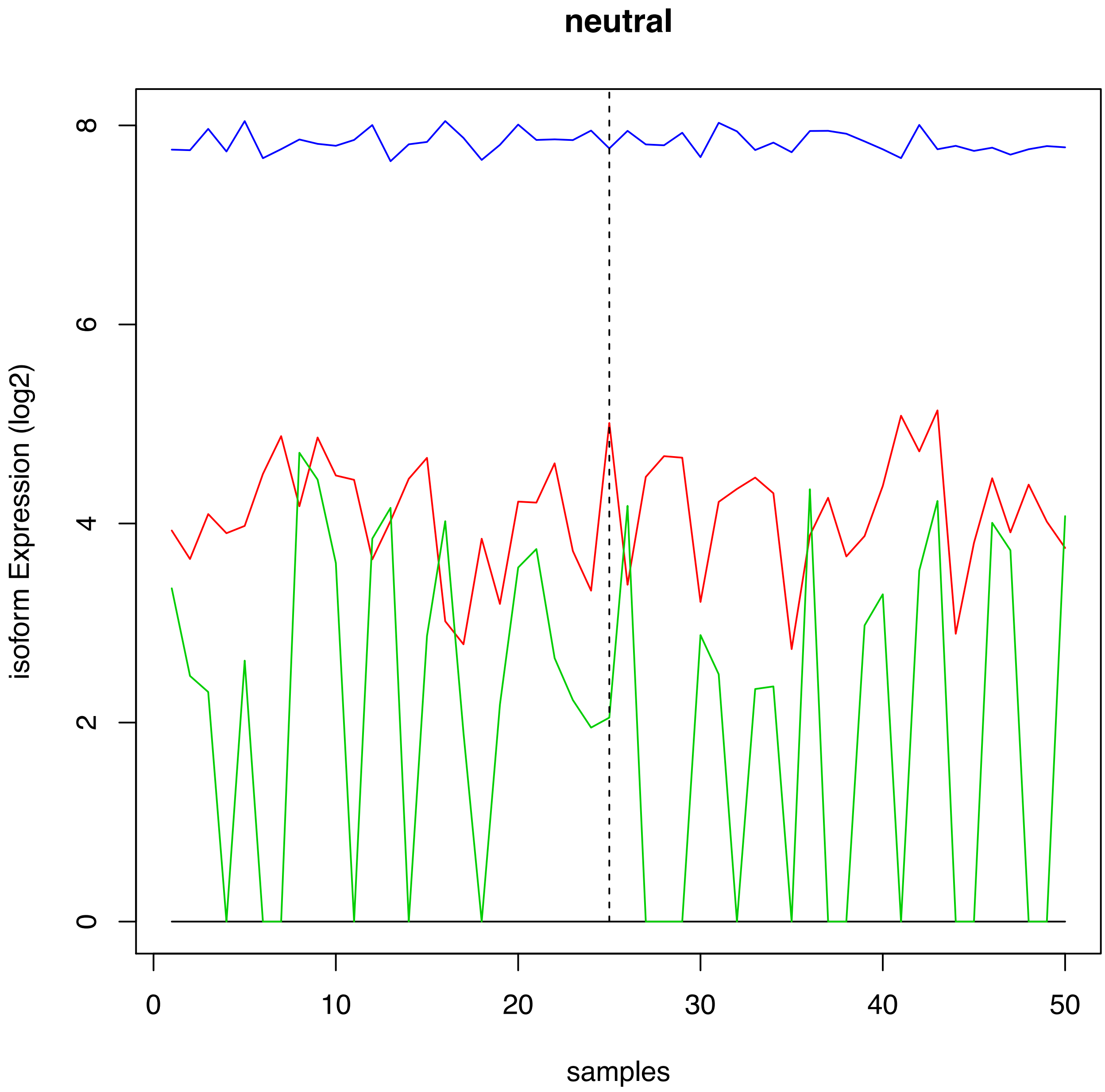

DE−DS
isoform Expression (log2)
0
2
4
6
8
10
0
10
20
30
40
50
samples

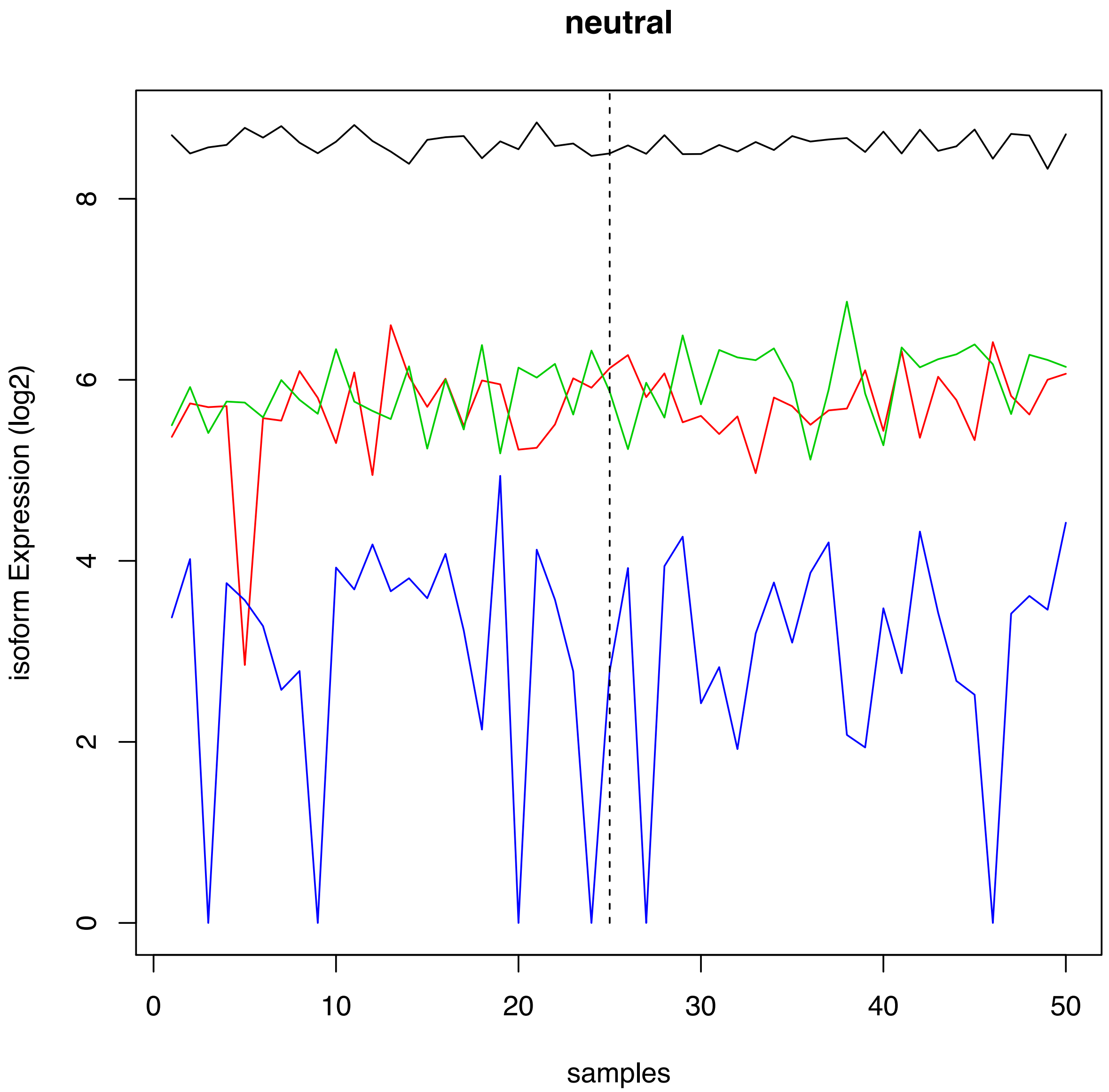

DE–DS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

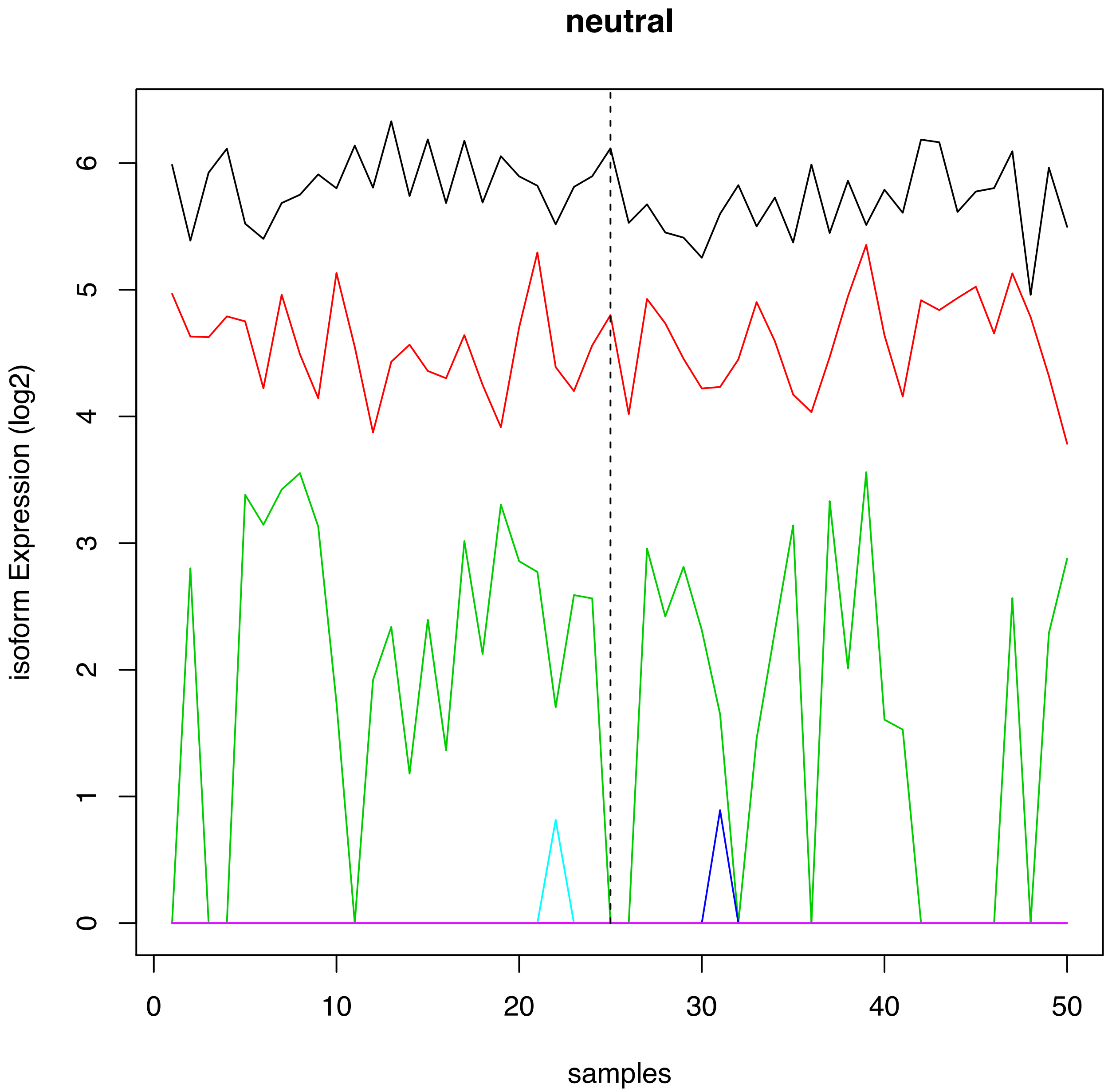

DE−DS
isoform Expression (log2)
0
2
4
6
8
samples
0
10
20
30
40
50

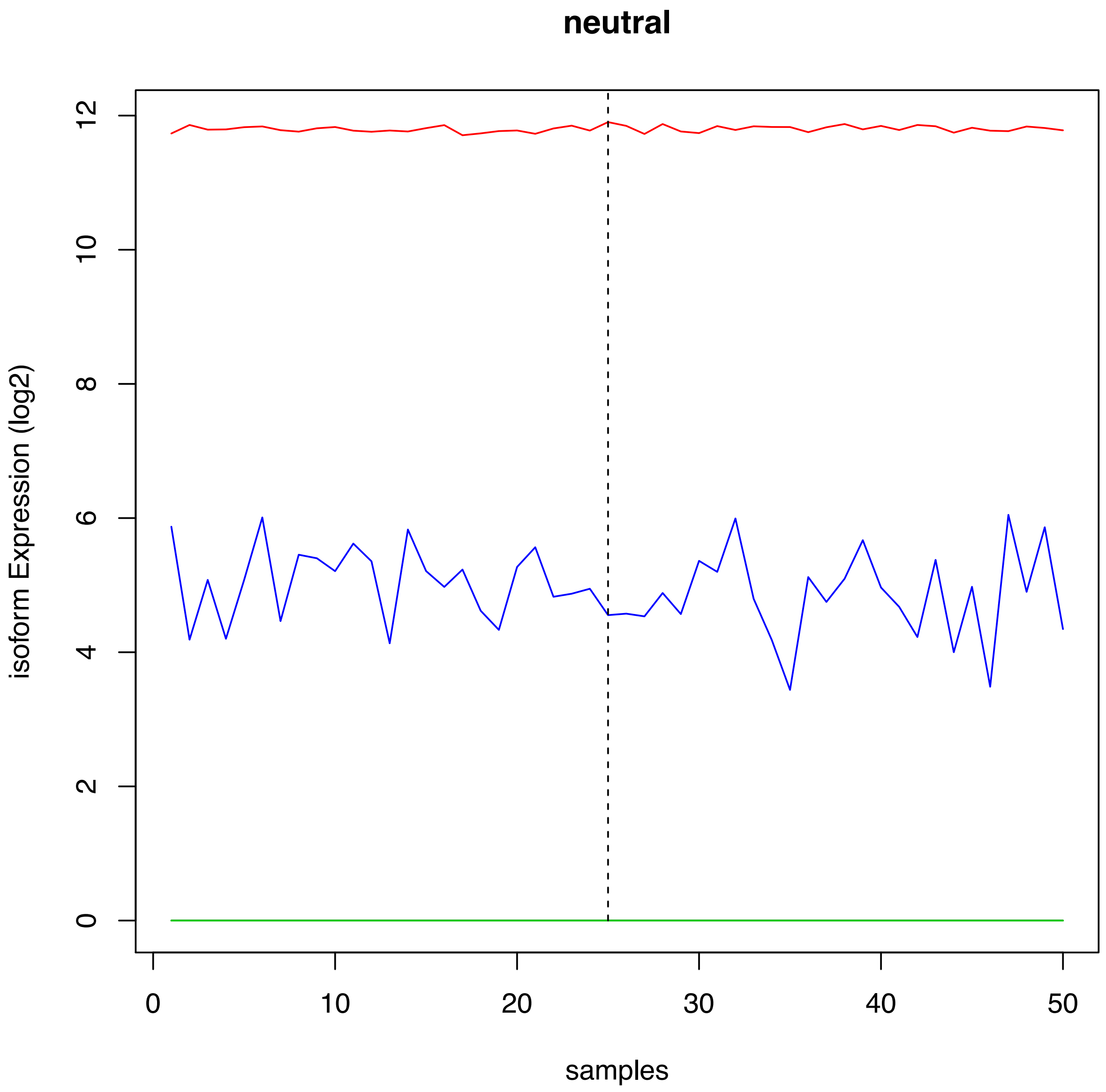

DE–DS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

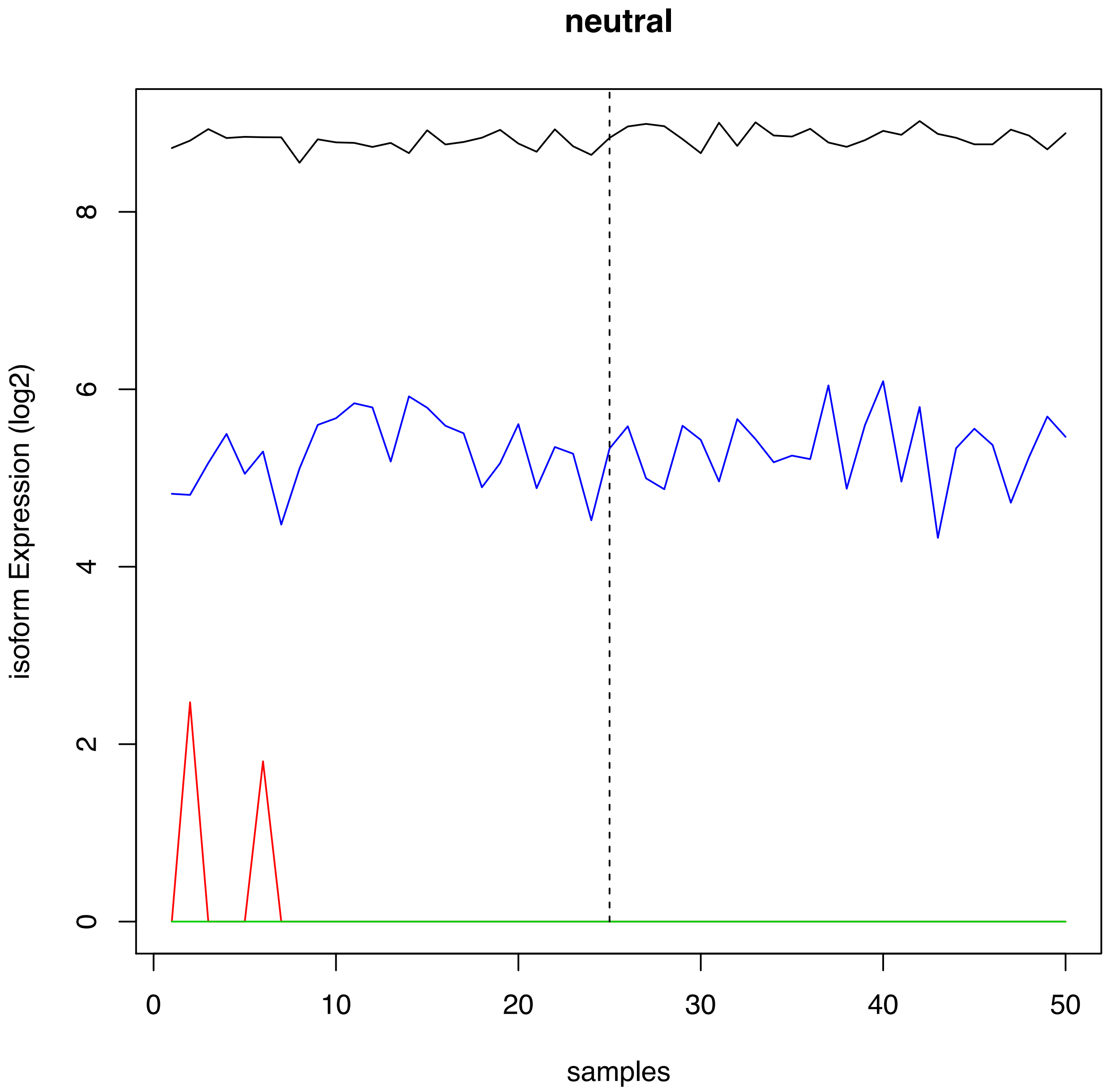

DE−DS
isoform Expression (log2)
0
1
2
3
4
5
6
7
samples
0
10
20
30
40
50

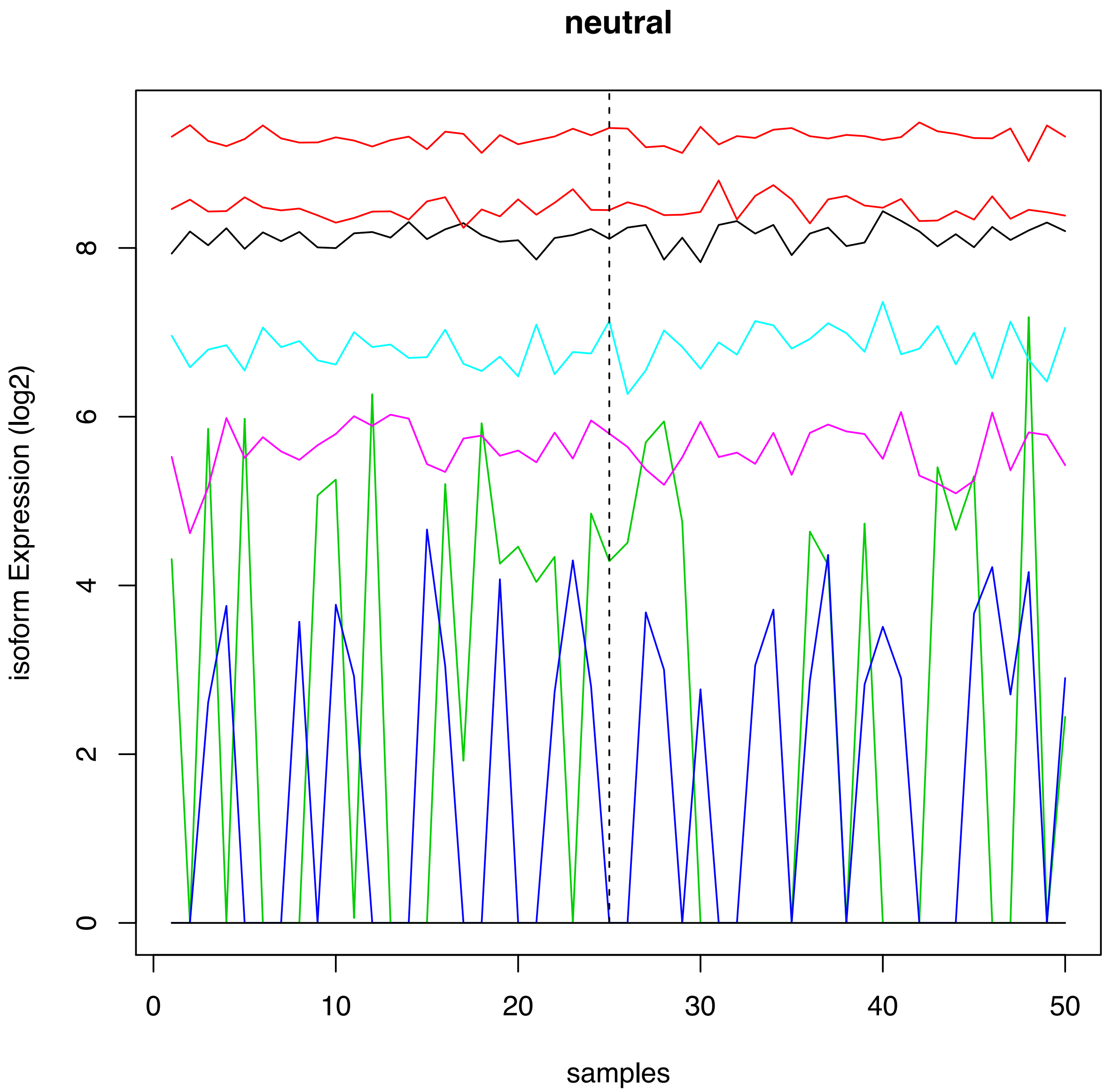

neutral
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

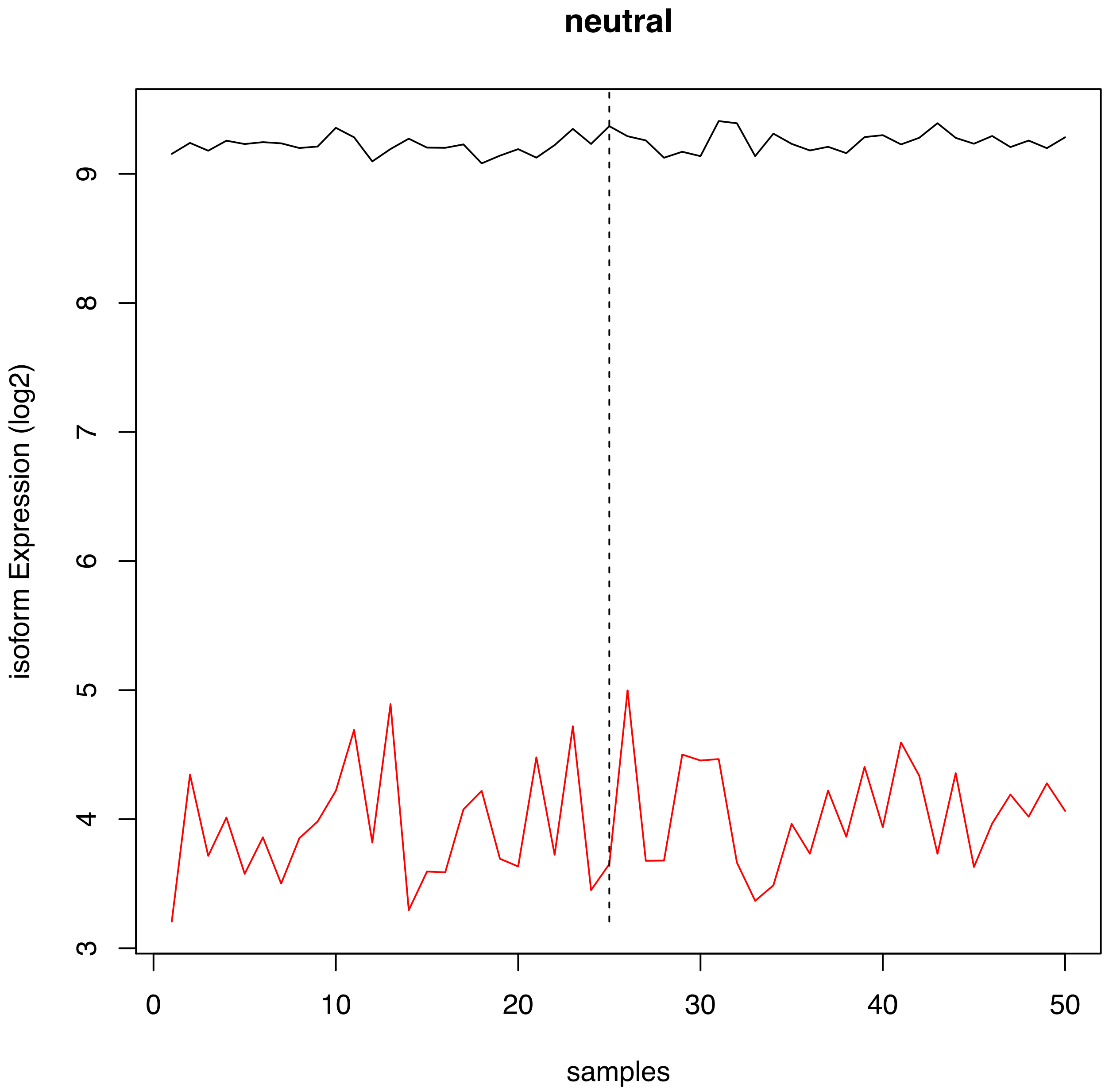

neutral
isoform Expression (log2)
0
2
4
6
8
samples
0
10
20
30
40
50

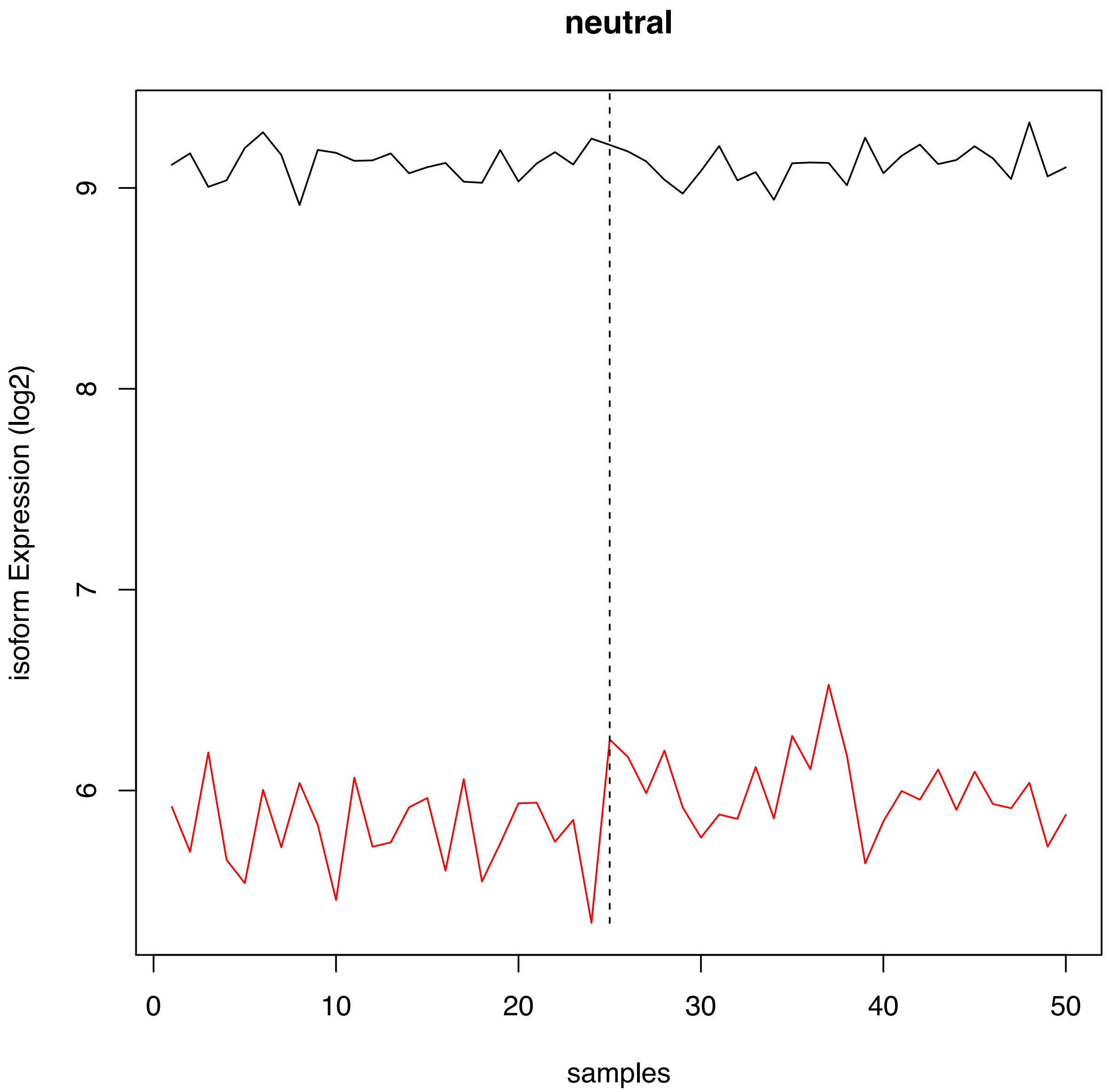

neutral
isoform Expression (log2)
0
1
2
3
4
5
6
samples
0
10
20
30
40
50

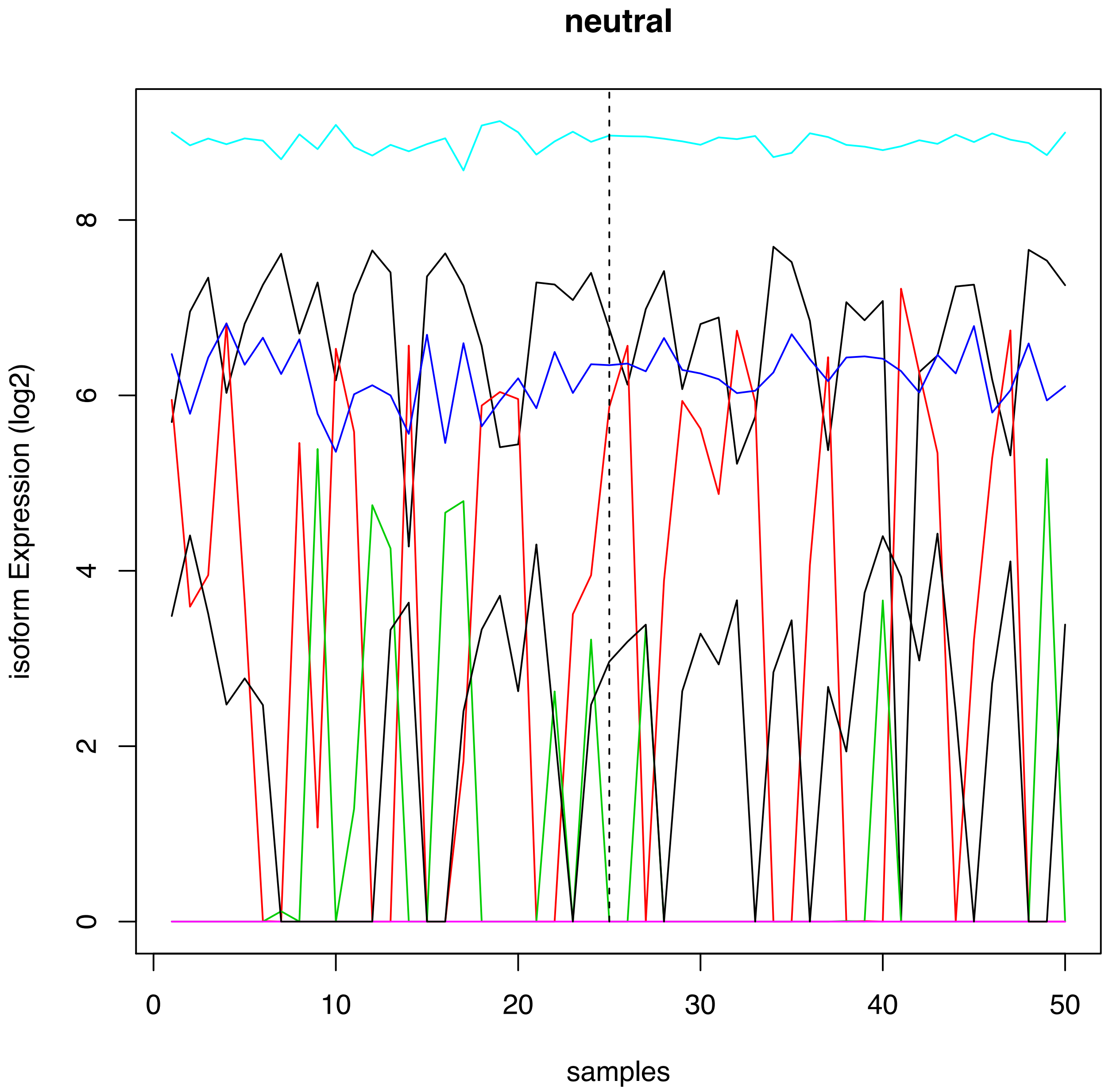

neutral
isoform Expression (log2)
0
2
4
6
8
10
12
samples
0
10
20
30
40
50

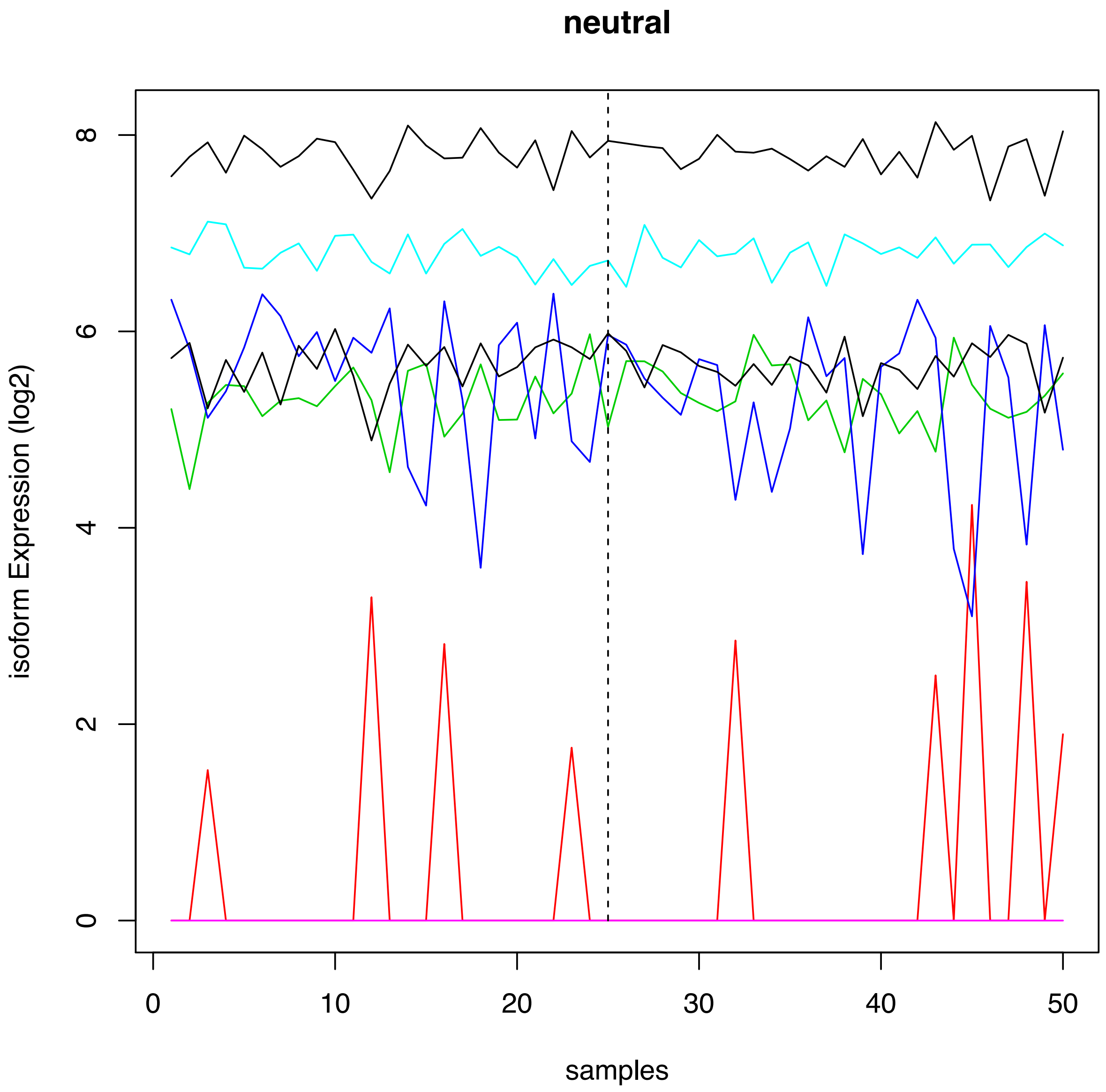

neutral
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

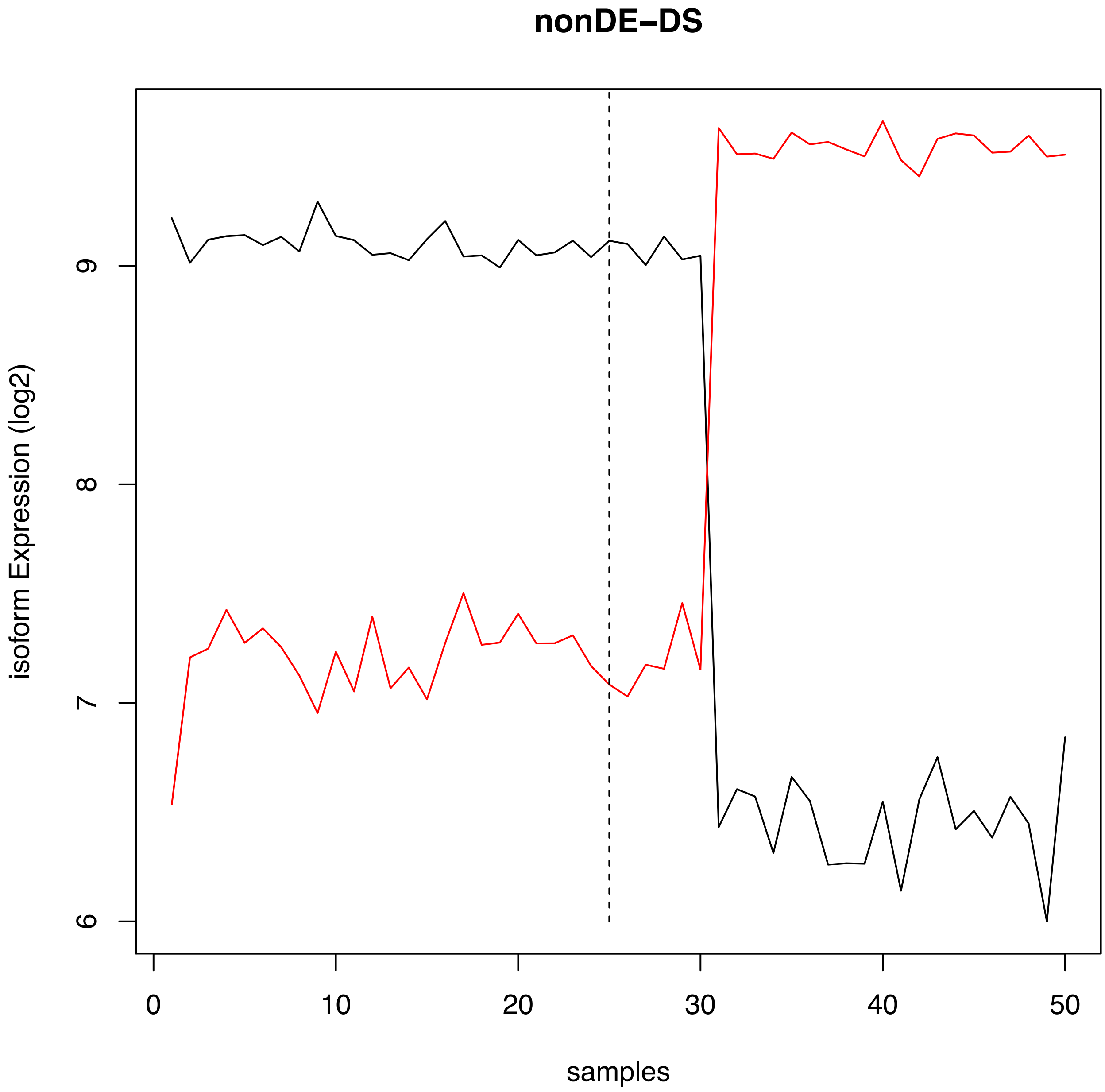

neutral
isoform Expression (log2)
0
2
4
6
8
samples
0
10
20
30
40
50

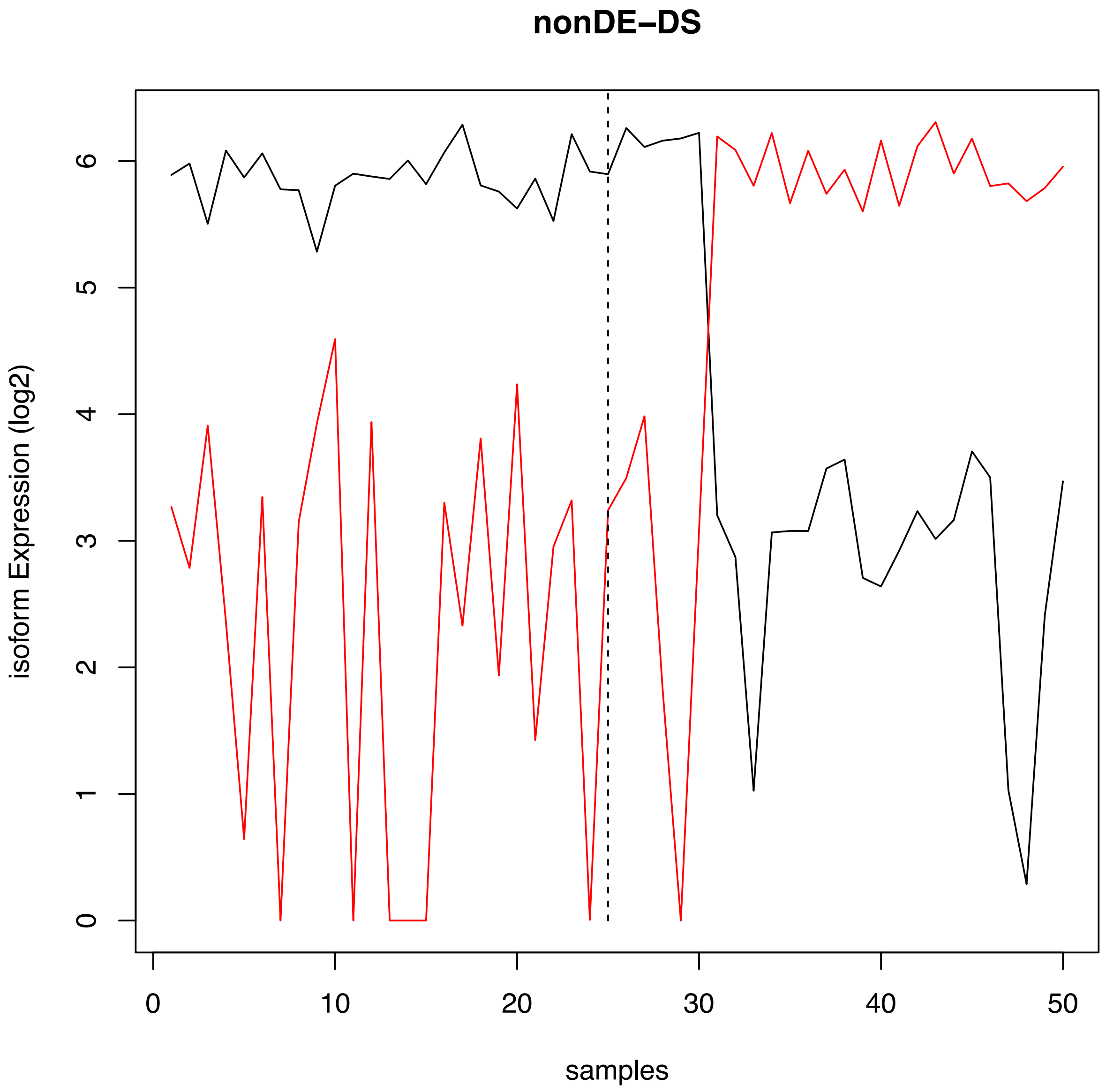

neutral
isoform Expression (log2)
3
4
5
6
7
8
9
0
10
20
30
40
50
samples

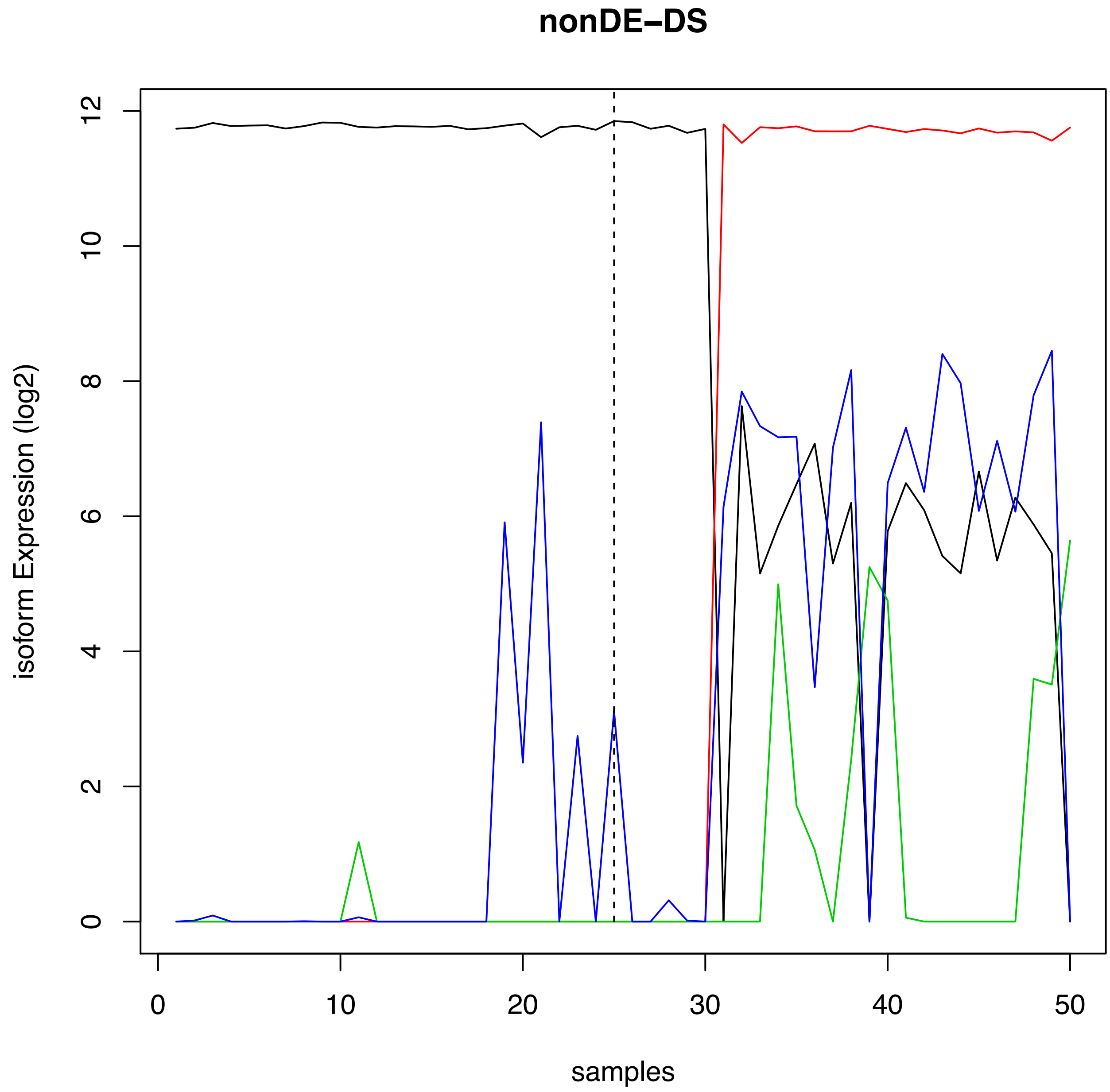

neutral
isoform Expression (log2)
9
8
7
6
0
10
20
30
40
50
samples

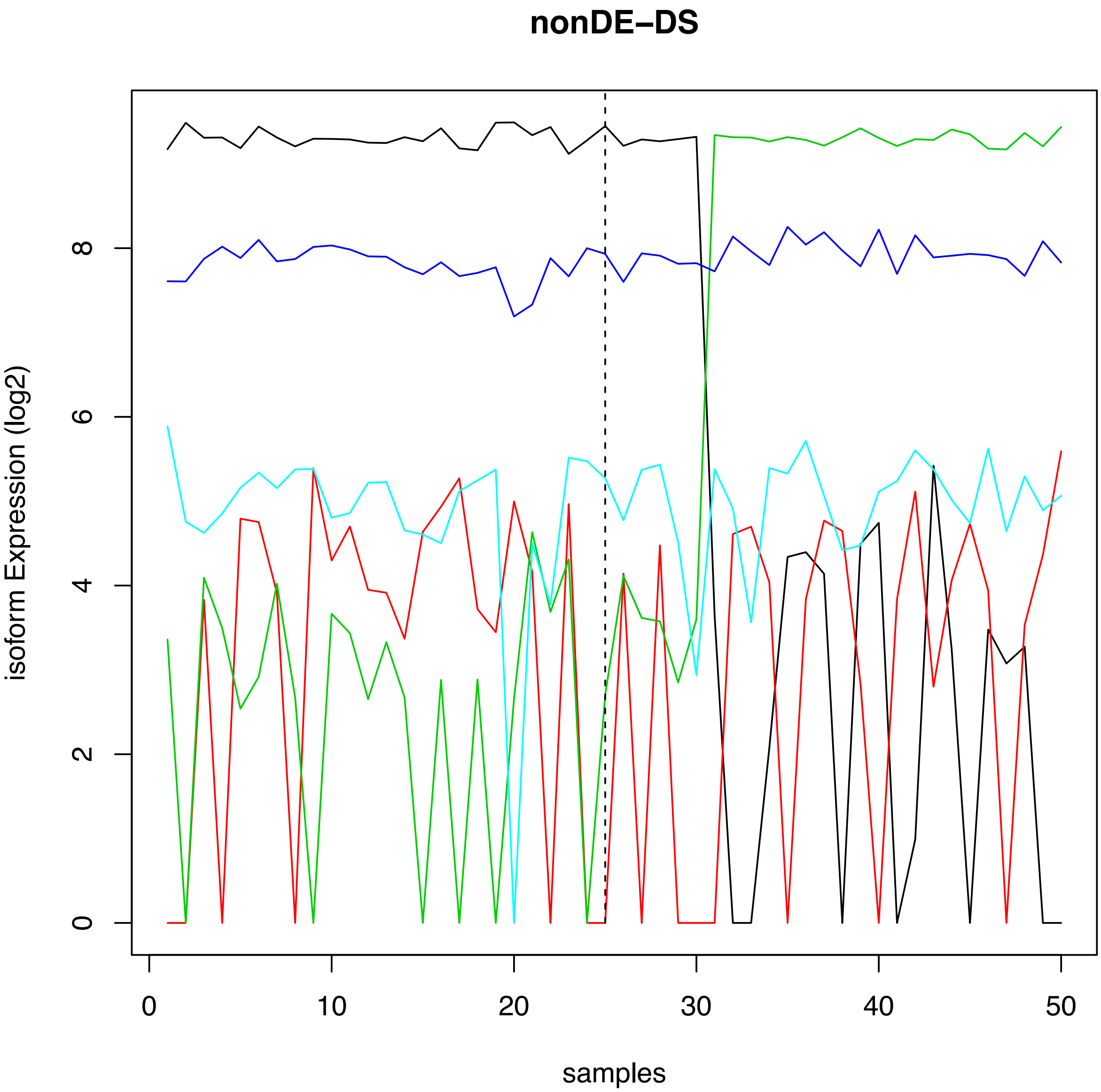

neutral
isoform Expression (log2)
0
2
4
6
8
samples
0
10
20
30
40
50

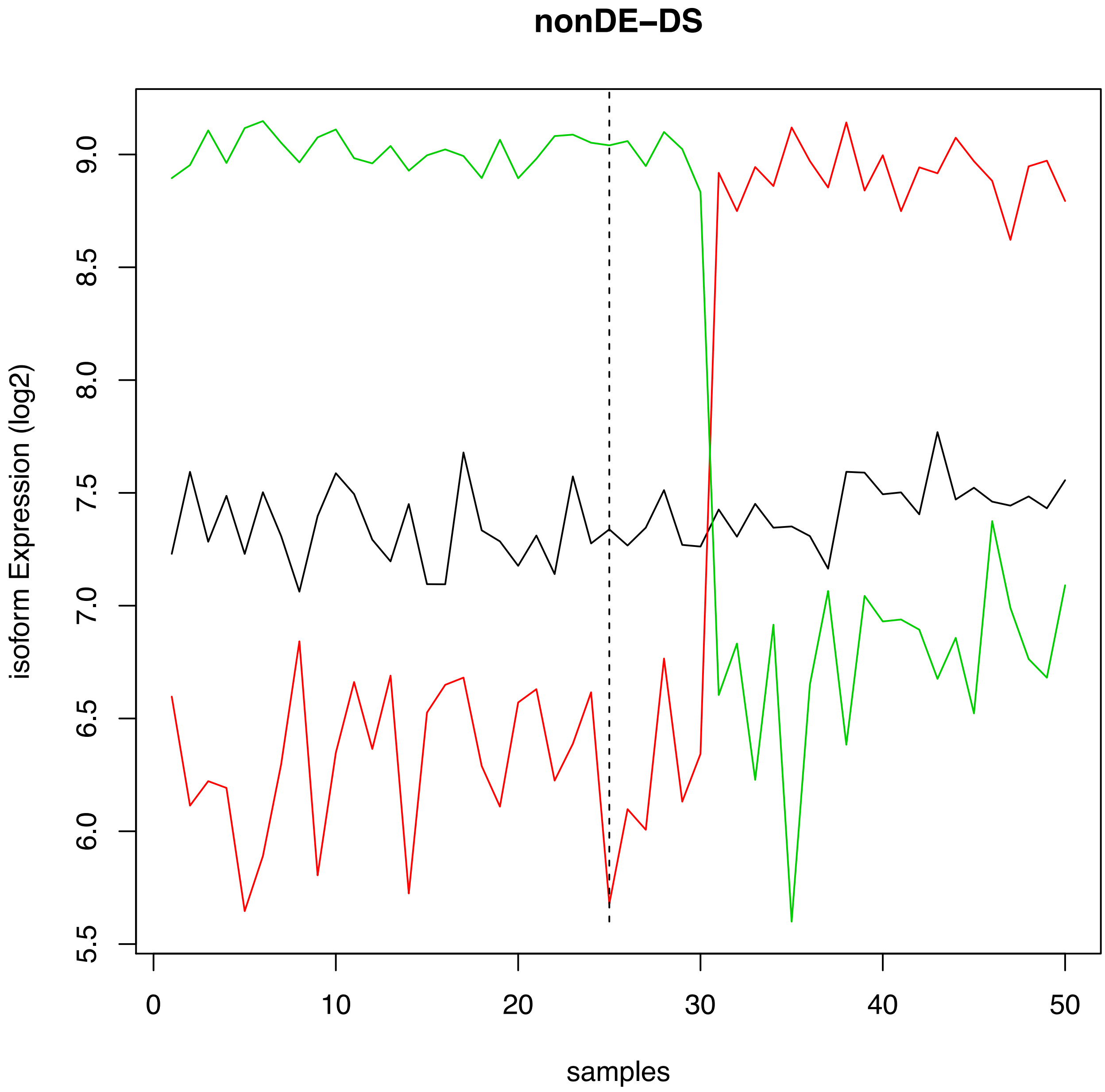

neutral
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

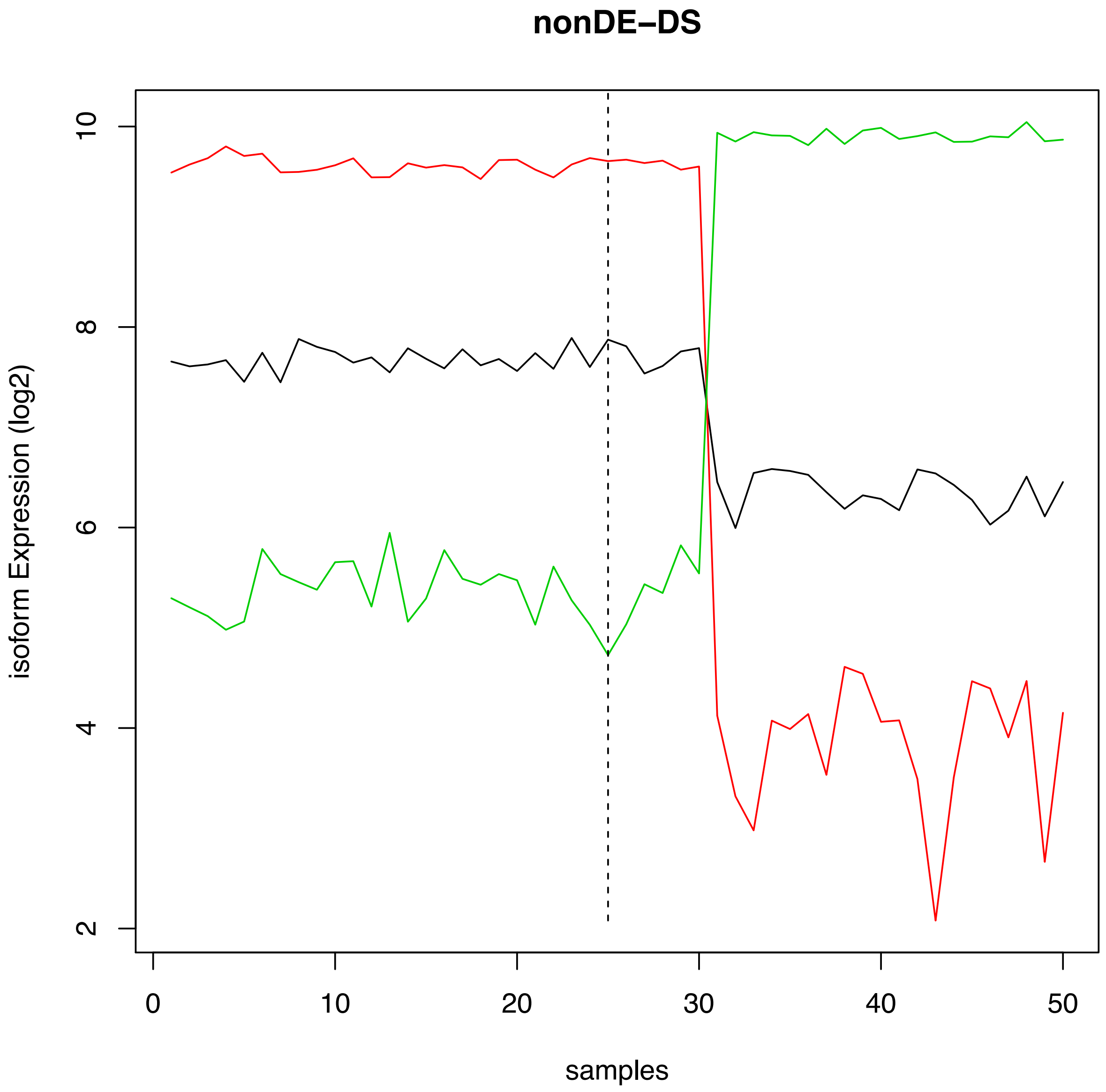

nonDE−DS
isoform Expression (log2)
6
7
8
9
0
10
20
30
40
50
samples

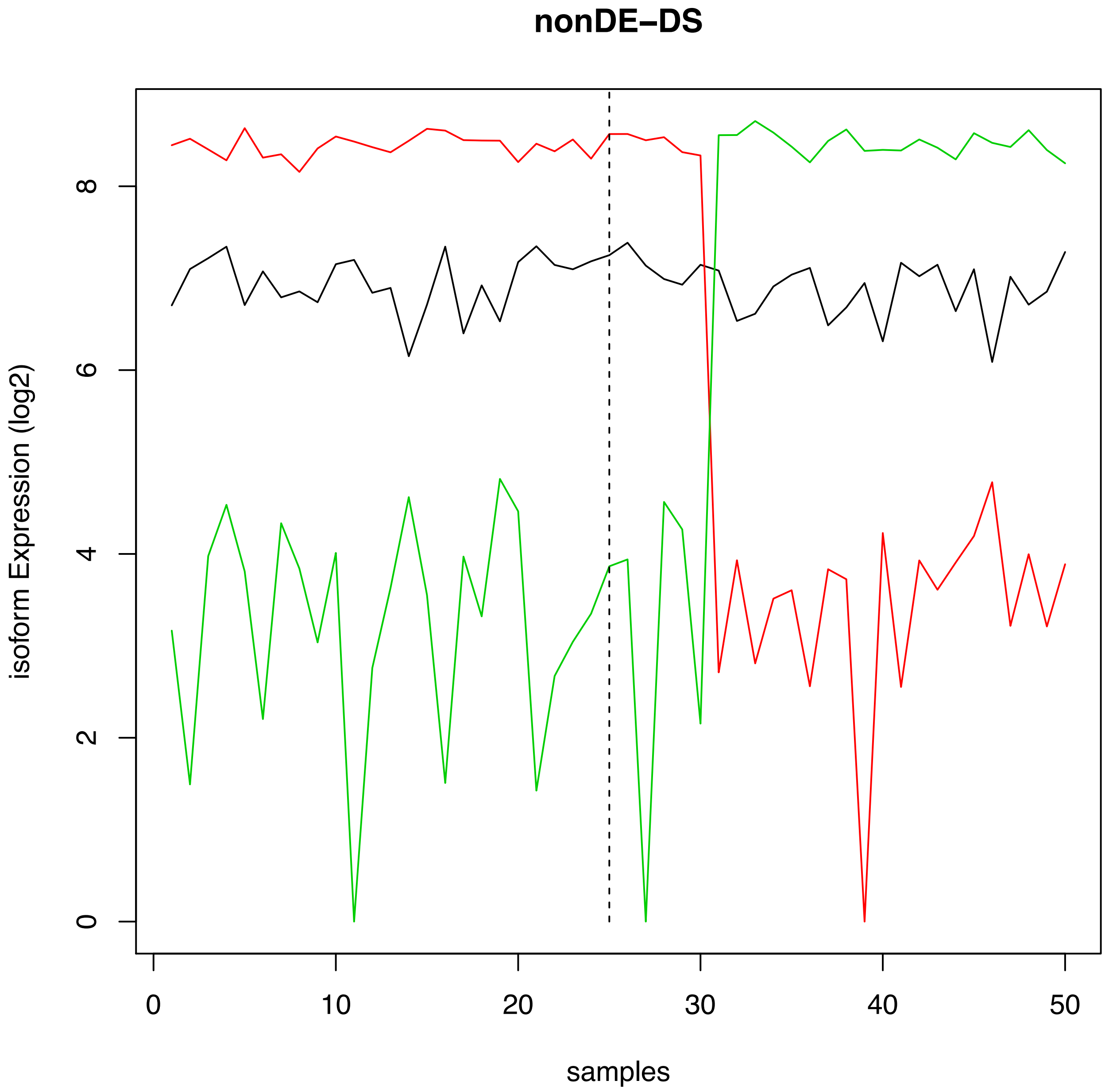

nonDE−DS
isoform Expression (log2)
0
1
2
3
4
5
6
0
10
20
30
40
50
samples

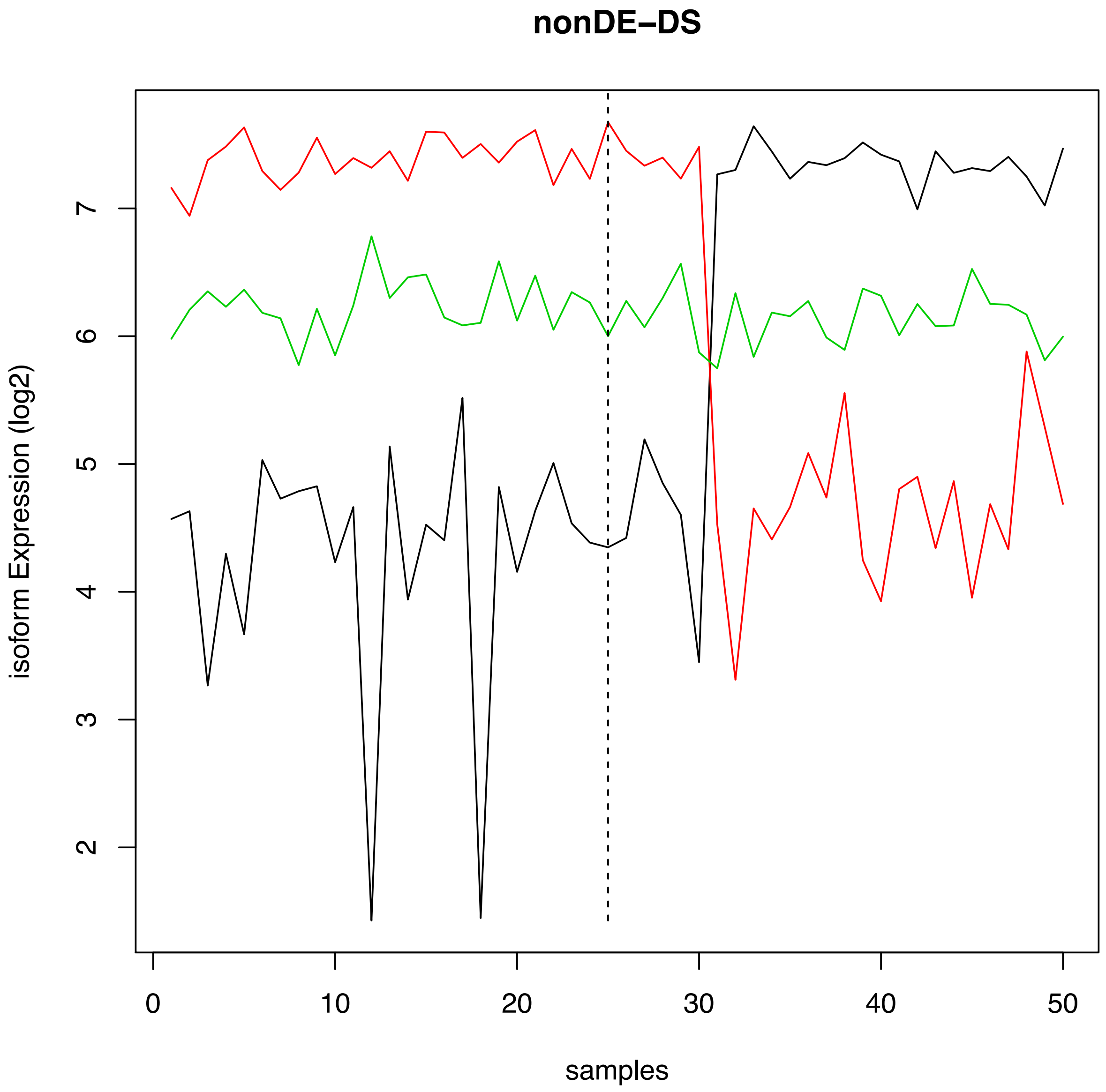

nonDE–DS
isoform Expression (log2)
0
2
4
6
8
10
12
samples
0
10
20
30
40
50

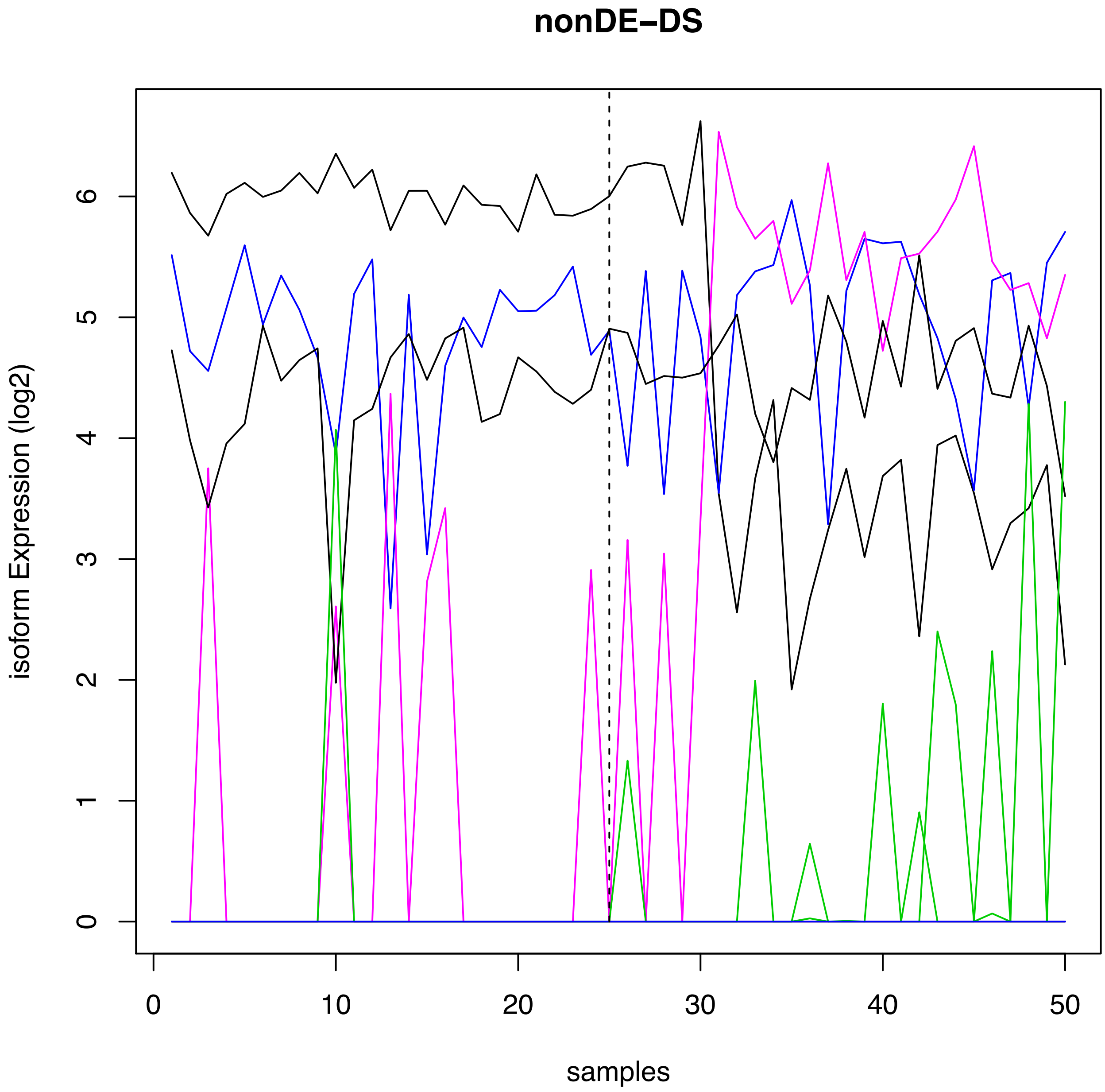

nonDE−DS
isoform Expression (log2)
0
2
4
6
8
0
10
20
30
40
50
samples

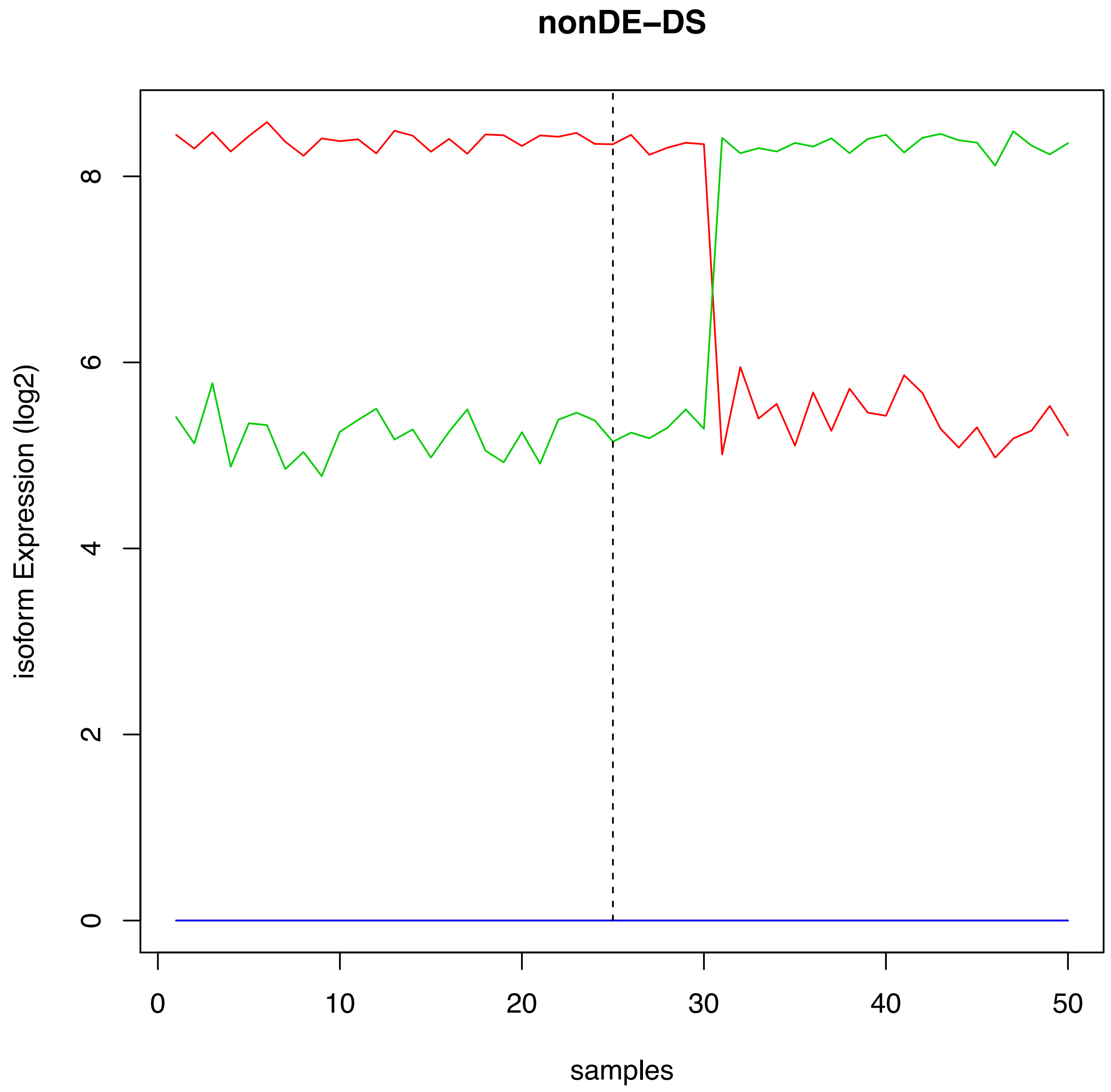

nonDE–DS
isoform Expression (log2)
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
0
10
20
30
40
50
samples


nonDE−DS
isoform Expression (log2)
2
4
6
8
10
0
10
20
30
40
50
samples


nonDE–DS
isoform Expression (log2)
0
2
4
6
8
samples
0
10
20
30
40
50


nonDE−DS
isoform Expression (log2)
2
3
4
5
6
7
0
10
20
30
40
50
samples


nonDE–DS
isoform Expression (log2)
0
1
2
3
4
5
6
0
10
20
30
40
50
samples



```
save(list=ls(),file = "../Cache/All.rda")
```

Now, we analyze the second simulations:

```
library(dplyr)
library(tibble)
library(reshape2)
library(ggplot2)


source("../Scripts/functions.R")



### loading genes
gn <- genes(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

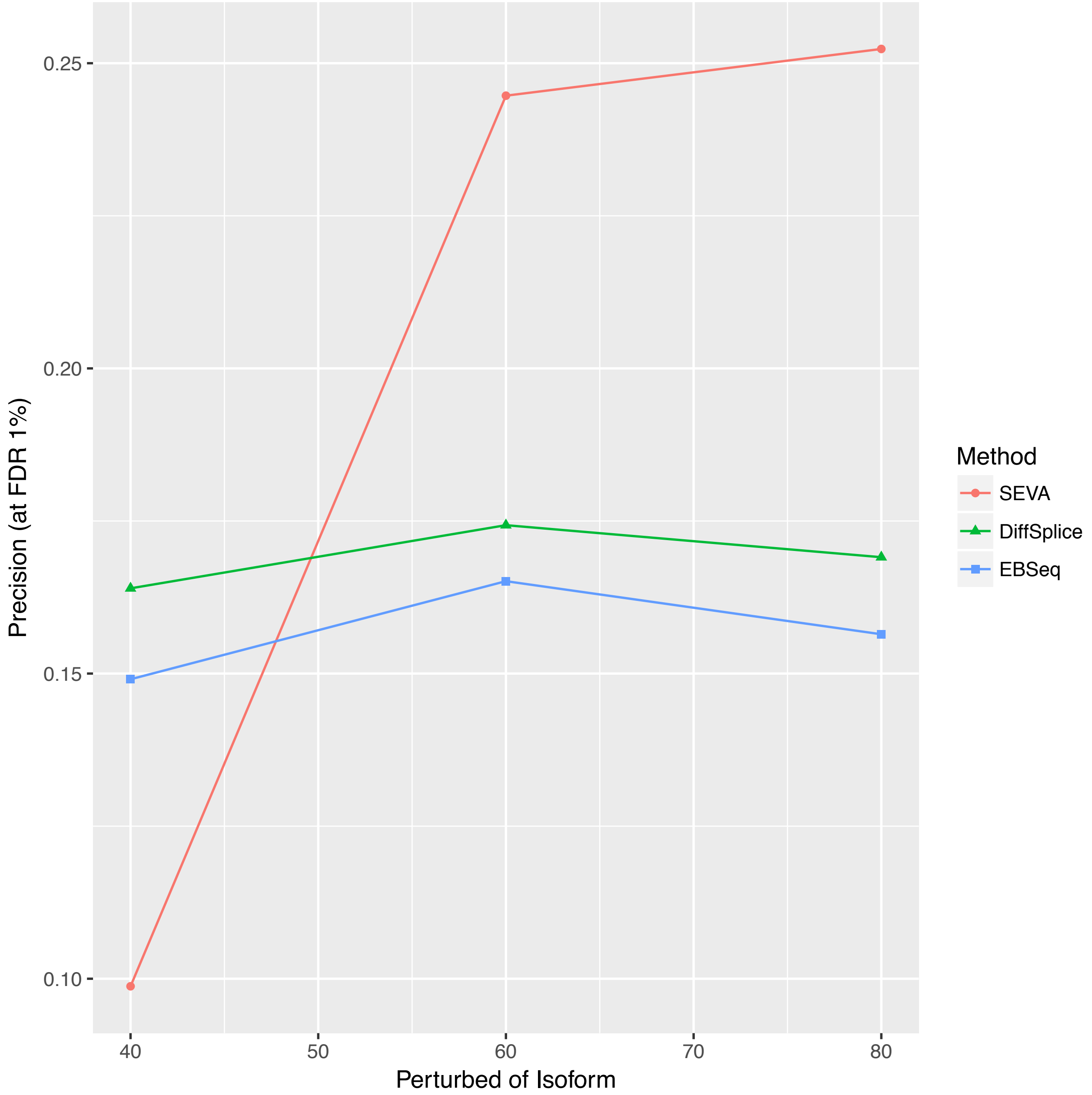
```
gSymbol <- AnnotationDbi::select(org.Hs.eg.db,
                  keys=as.character(gn$gene_id),
                  columns=c('SYMBOL'),keytype='ENTREZID')
gn$SYMBOL <- gSymbol$SYMBOL

#loading the Truth
ASTruth <- (read.delim("../../NewSimData/Sim2/Data/Truth/AS_genes_list.txt",sep = "\t",header = F,string

load("../Results/Simulation/SecondTryJan13/isos2genesvect.rda")
isos2genesvectsimplified <- sapply(strsplit(isos2genesvect,split = '[|]'),FUN = function(x) x[1])


load("../Data/Data4Simulation/Sim2.Rdata")

phenotypes <- setNames(grepl(pattern = "tumor",
colnames(allJunctionsMat)),
colnames(allJunctionsMat))
## Scenarios

Scenarios <- vector(mode = "list",length = 4)
Scenarios[[1]] <- c(paste("normalSim_",1:20,sep = ""),
                    paste("tumorSim20_",1:20,sep = ""))
Scenarios[[2]] <- c(paste("normalSim_",1:20,sep = ""),
                    paste("tumorSim20_",1:10,sep = ""),
                    paste("tumorSim40_",11:20,sep = ""))
Scenarios[[3]] <- c(paste("normalSim_",1:20,sep = ""),
                    paste("tumorSim20_",1:10,sep = ""),
                    paste("tumorSim60_",11:20,sep = ""))
Scenarios[[4]] <- c(paste("normalSim_",1:20,sep = ""),
                    paste("tumorSim20_",1:10,sep = ""),
                    paste("tumorSim80_",11:20,sep = ""))


diffspliceFiles <- dir("../../PaperSuppl/Results/Simulation/DiffSplice/")

ebseqpvalueGenesAll <- vector(mode = "list",length = length(Scenarios))

DSEBseqAll <- ebseqpvalueGenesAll
SEVAAll <- ebseqpvalueGenesAll
DEEBSeqAll <- ebseqpvalueGenesAll
AllResults <- ebseqpvalueGenesAll
for( i in seq_along(Scenarios)){
  #current samples: Normals as nomral with a mixture of normal and cancerous as the cancer samples
  samplescur <- Scenarios[[i]]
  #phenotype
  phenotypescur <- setNames(grepl(pattern = "tumor",x=samplescur),samplescur)

  junctionPValue <- SEVA.meangeneFilter(junc.RPM=allJunctionsMat[,samplescur],
                                        phenoVect=as.factor(phenotypescur),
                                        geneexpr=allGeneExpressionMat[,samplescur],
                                        minmeanloggeneexp= 0)

  SEVA <- names(which(p.adjust(sapply(junctionPValue,function(x) x$pvalueTotal))<0.01))
```

```
SEVAAll[[i]] <- SEVA

zscoresSEVA <- sapply(junctionPValue,FUN = function(x) abs(x$zscore))


DSEBseq_outcome <- findDSEBSEQ(allIsoformsMat[names(isos2genesvectsimplified)[which(isos2genesvectsim
                               isos2genesvectsimplified,
                               phenoVect =as.factor(phenotypescur) )

DSEBseq <- DSEBseq_outcome$DSEBseq
DSEBseqAll[[i]] <- DSEBseq

DEEBSeq <- findDEGenes(geneexp = allGeneExpressionMat[names(junctionPValue),samplescur],

            phenoVect =as.factor(phenotypescur) ,pvaluecorrected = 0.05)

DEEBSeqAll[[i]] <- DEEBSeq$DEGenes_EBesq

if( i== 2){
  diffsplicefile <- read.delim("../../NewSimData/Sim2/ResultsSimN40/result/differential_transcription

}else if(i==3){
  diffsplicefile <- read.delim("../../NewSimData/Sim2/ResultsSimN60/result/differential_transcription
}else if(i==4){
  diffsplicefile <- read.delim("../../NewSimData/Sim2/ResultsSimN80/result/differential_transcription

}

if(i == 1){
  DiffSplicegenes = c()
}else{
  junctionsDiffSplice <- GRanges(seqnames = Rle(diffsplicefile[,"chromosome"]),
                                 ranges = IRanges(start = as.numeric(diffsplicefile[,"position_start"]
                                                  end = as.numeric(diffsplicefile[,"position_end"])))

  DiffSplicehits <- findOverlaps(junctionsDiffSplice,gn)
  DiffSplicegenes <- unique(gn$SYMBOL[subjectHits(DiffSplicehits)])
}
AllResults[[i]] <- inner_join(data.frame(Gene = names(junctionPValue)) %>%
                                mutate(DS = Gene %in% ASTruth,
                                       SEVA = Gene %in% SEVA,
                                       EBSeq = Gene %in% DEEBSeq$DEGenes_EBesq,
                                       DE = Gene %in% DEEBSeq$DEGenes_EBesq,
                                       DiffSplice = Gene %in% DiffSplicegenes ),
                              data.frame(LogMeanExp =
                                           (allGeneExpressionMat %>% rowMeans()+1)%>%
                                           log2) %>%
                                rownames_to_column("Gene" ) %>%
                                transmute(Gene=Gene,ExpressionLevel = cut(LogMeanExp,c(-Inf,4,9,Inf))
}
```

```
## 100
## 200
## 300
## 400
## 500
## 600
## 700
## 800
## 900
## 1000
## 100
## 200
## 300
## 400
## 500
## 600
## 700
## 800
## 900
## 1000
## 100
## 200
## 300
## 400
## 500
## 600
## 700
## 800
## 900
## 1000
## 100
## 200
## 300
## 400
## 500
## 600
## 700
## 800
## 900
## 1000
```

```
names(AllResults) <- paste(c(20,40,60,80))
NewSimulations <-
  sapply(AllResults[2:4],function(x)
    setNames(c(x %>% filter(SEVA==T)  %>% summarise(mean(DS)) %>% unlist ,
               x %>% filter(DiffSplice==T)  %>% summarise(mean(DS))%>% unlist,
               x %>% filter(EBSeq==T)  %>% summarise(mean(DS))%>% unlist),c("SEVA","DiffSplice","EBSeq"
NewSimulationsForPlot <- melt(NewSimulations)
colnames(NewSimulationsForPlot) <- c("Method","Perturbed of Isoform","Precision (at FDR 1%)")

NewSimulationsForPlot$`Perturbed of Isoform`<-
  as.numeric(NewSimulationsForPlot$`Perturbed of Isoform`)
```

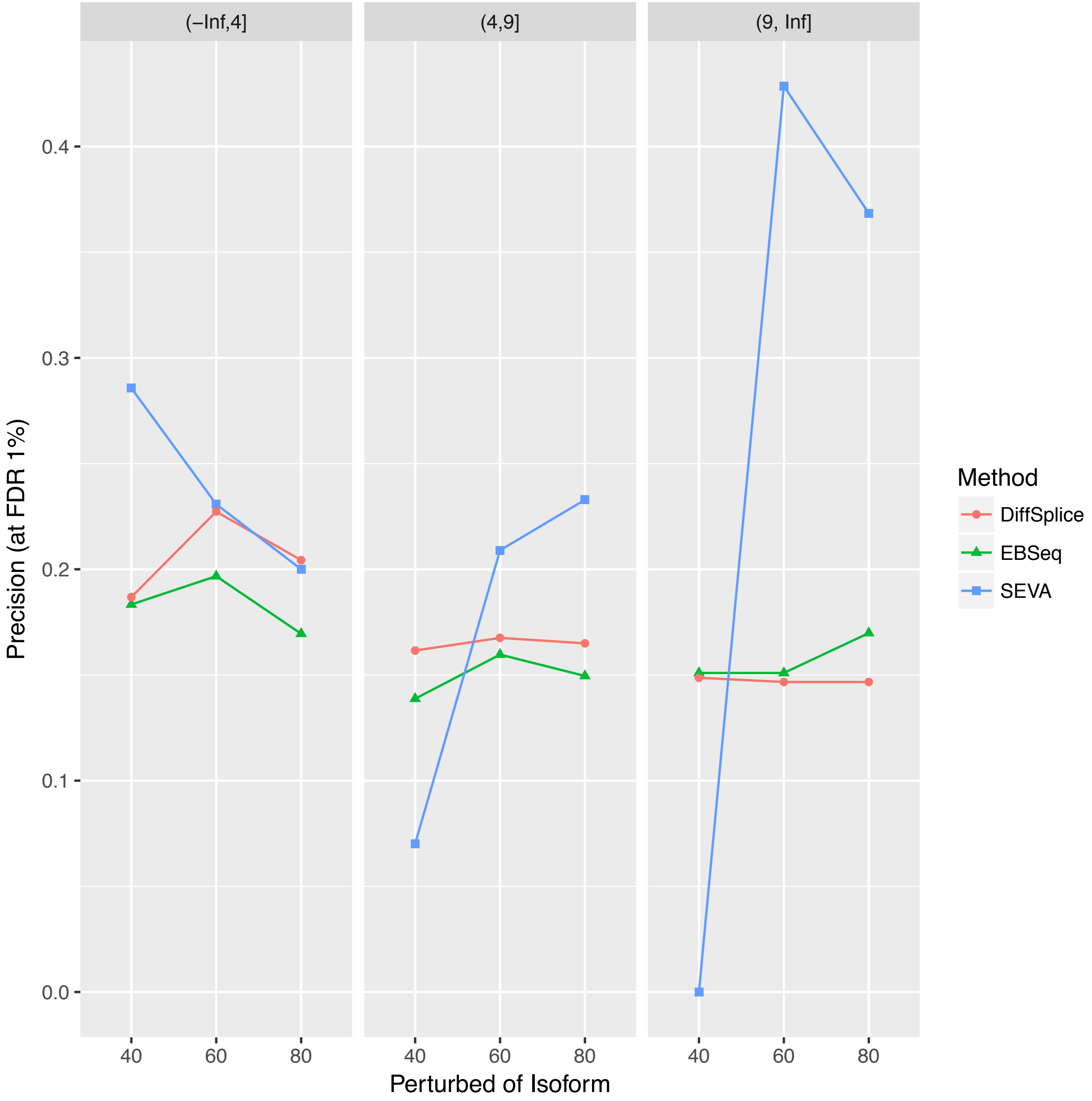
```
print(ggplot( NewSimulationsForPlot,
      aes(x=`Perturbed of Isoform`,
          y=`Precision (at FDR 1%)`,
          shape=Method ,
          color=Method))+
geom_point()+geom_line())
```



```
pdf("../../NewSimData/Sim2/SecondSimulations.pdf")
print(ggplot( NewSimulationsForPlot,
        aes(x=`Perturbed of Isoform`,
            y=`Precision (at FDR 1%)`,
            shape=Method ,
            color=Method))+
  geom_point()+geom_line())

dev.off()
```

```
## pdf
##   2
```

```
NewSimulationsLevels <- bind_rows(sapply(names(AllResults)[2:4],
                                         function(x) AllResults[[x]] %>%
                                           mutate(`Perturbed of Isoform`=x),simplify = F))

NewSimulationsLevelsPlot <-
  rbind(NewSimulationsLevels %>% group_by(ExpressionLevel,`Perturbed of Isoform`)%>%filter(EBSeq==T) %>
        NewSimulationsLevels %>% group_by(ExpressionLevel,`Perturbed of Isoform`)%>%filter(SEVA==T) %>%
        NewSimulationsLevels %>% group_by(ExpressionLevel,`Perturbed of Isoform`)%>%filter(DiffSplice==
colnames(NewSimulationsLevelsPlot)[3] <- "Precision (at FDR 1%)"

print( ggplot(NewSimulationsLevelsPlot,
              aes(y=`Precision (at FDR 1%)`,
                  x=`Perturbed of Isoform`,
                  shape=Method,
                  group =Method,
                  color=Method))+
         geom_point()+
         geom_line()+
         facet_wrap(~ExpressionLevel))
```



```
pdf("../../NewSimData/Sim2/SecondSimulationsForDifferentExpressionValue.pdf")
print( ggplot(NewSimulationsLevelsPlot,
              aes(y=`Precision (at FDR 1%)`,
                  x=`Perturbed of Isoform`,
                  shape=Method,
                  group =Method,
                  color=Method))+
         geom_point()+
         geom_line()+
         facet_wrap(~ExpressionLevel))
dev.off()
```

```
## pdf
##   2
```