Supplementary material for "Prioritization of omics features using Beta distributions on Montecarlo p-values"

March 13, 2021

Contents

Chapter 1

Microarrays

1.1 Packages

Loading the needed packages.

```
pacman::p_load(Biobase,OMICfpp2,ggplot2,hgu133plus2.db,hgu133a.db,
               hgu133a2.db,hgu219.db,locfit,ReportingTools,limma,ggpubr,
               gganimate)
```

The working directory is defined.

```
progDir = "/home/gag/Nextcloud/BetaMonteCarlo20/prog/"
setwd(progDir)
```

Some functions to generate results.

```
source(paste0(progDir,"fun-BetaMonteCarlo20.R"))
```

1.2 Reading data

```
dirArrayData = "/home/gag/Nextcloud/DATOS_ANGELA/Array/"
```

File `Array2.csv` contains the metadata of the experiments.

```
id = read.csv(paste0(progDir,"Array2.csv"),
              header=FALSE,sep=";",stringsAsFactors=FALSE)
id[,2] = as.factor(id[,2])
pathData =  paste0(dirArrayData,id[,1],".rda")
```

Now we obtain the universe of genes that are considered at least one experiment.

```
dbs = unique(id[,3])
pacman::p_load(unique(id[,3]),character.only=TRUE)
entrez = NULL
## To replace will all databases
for(i  in 1:23){
```

```
    load(pathData[i])
    x0 = get(id[i,1])
    a = AnnotationDbi::select(get(id[i,3]),
                              keys=featureNames(x0),
                              columns=c("ENTREZID","SYMBOL"),keytype="PROBEID")
    entrez = union(entrez,a[,"ENTREZID"])
}
entrez = na.omit(entrez)
save(entrez,file="entrez.rda")
```

```
load("entrez.rda")
```

Let us indicate paired and non paired data sets.

```
experiments = c(1:19,21,23)
experimentsPaired = intersect(which(id[,2] == "paired"),experiments)
experimentsNonPaired = intersect(which(id[,2] == "non-paired"),experiments)
chips = setdiff(unique(id[,3]),c("hgu219.db","hgu133a2.db"))
nsims = c(100,300,500)
```

Let us indicate the directory to save different files.

```
outputDir = "~/Nextcloud/BetaMonteCarlo20/prog/results/"
```

1.3 Limma

In this section the p-values using the limma method [1] are calculated.
First, the non-paired studies are considered.

```
for(i in experimentsNonPaired){
  cat("Experiment ",i,"\n")
  load(pathData[i])
  x0 = get(id[i,1])
  design =  model.matrix(~pData(x0)[,"Type"])
  fit = lmFit(x0, design)
  eBayesfit = eBayes(fit)
  x1 = topTable(eBayesfit,sort.by="none",number=nrow(x0),p.value=1,
                coef=ncol(design))
  save(x1,file=paste0(outputDir,"x1/x1_limma_patient_",i,".rda"))
}
```

Secondly, we will considered the paired studies.

```
for(i in experimentsPaired){
  cat("Experiment ",i,"\n")
  load(pathData[i])
  x0 = get(id[i,1])
  pData(x0)[,"Pair"] = as.factor(pData(x0)[,"Pair"])
  design =  model.matrix(~pData(x0)[,"Pair"]+pData(x0)[,"Type"])
  fit = lmFit(x0, design)
  eBayesfit = eBayes(fit)
```

```
  x1 = topTable(eBayesfit,sort.by="none",number=nrow(x0),p.value=1,
                coef=ncol(design))
  save(x1,file=paste0(outputDir,"x1/x1_limma_patient_",i,".rda"))
}
```

Let us merge results for all experiments using the limma p-values.

```
load("entrez.rda")
res = matrix(NA,nrow=length(entrez),ncol=length(experiments))
for(i in 1:length(experiments)){
  experiment = experiments[i]
  cat("Experiment ",experiment,"\n")
  load(paste0(outputDir,"x1/x1_limma_patient_",experiment,".rda"))
  a = AnnotationDbi::select(get(id[i,3]),
                            keys=x1[,"PROBEID"],
                            columns="ENTREZID",keytype="PROBEID")
  c1 = match(unique(a[,1]),a[,1])
  a1 = a[c1,]
  c2 = match(unique(a1[,2]),a1[,2])
  a2 = a1[c2,]
  a2 = na.omit(a2)
  res[match(a2[,"ENTREZID"],entrez),i] =
    x1[match(a2[,"PROBEID"],x1[,"PROBEID"]),"P.Value"]
}
save(res,file=paste0(outputDir,"res/res_limma.rda"))
```

1.4 Generating p-values and scores

Different scores are generated for **paired** data sets.

```
load("entrez.rda")
for(nsim in nsims){
  cat("nsim ",nsim,"\n")
  for(dts in c("normal","tstudent")){
    for(type in c("between-pair","complete")){
      type = "complete"
      doResults(experiments = experimentsPaired,nsim=nsim,id=id,dts=dts,
                type=type,alpha=.95,pathData=pathData,
                outputDir = paste0(outputDir,"x1/"),nboot=2,fboot=.5)
    }
  }
}
```

Different scores are generated for **non-paired** data sets.

```
load("entrez.rda")
type = "complete"
for(nsim in nsims){
  cat("nsim ",nsim,"\n")
  for(dts in c("normal","tstudent")){
    doResults(experiments = experimentsNonPaired,nsim=nsim,id=id,dts=dts,
```

```
              type=type,alpha=.95,pathData=pathData,
              outputDir = "~/Nextcloud/BetaMonteCarlo20/prog/results/x1/",
              nboot=10,fboot=.9)
  }
}
```

1.5 Merging p-values

This section contains how to merge the different p-values for different cumulative distribution functions and methods.

```
nsims = c(100,300,500)
experiments = experimentsPaired
for(experiment in experiments){
  df = matrix(NA,ncol=length(nsims)*2*2+2,nrow=length(entrez))
  df = as.data.frame(df)
  rm(x1)
  cat("Experiment ",experiment,"\n")
  load("entrez.rda")
  df[,1] = entrez
  ncol0 = 1
  names0 = "entrez"
  for(dts in c("normal","tstudent")){
    cat("dts ",dts,"\n")
    for(nsim in nsims){
      cat("nsim ",nsim,"\n")
      for(type in c("complete","between-pair")){
        rm(x1)
        load(paste0(outputDir,"x1/x1_",dts,"_",type,"_",nsim,"_",experiment,
                    ".rda"))
        res0 = matrix(NA,nrow=length(entrez),ncol=5)
        load(pathData[experiment])
        x0 = get(id[experiment,1])
        a = AnnotationDbi::select(get(id[experiment,3]),
                                  keys=featureNames(x0),
                                  columns="ENTREZID",keytype="PROBEID")
        c1 = match(unique(a[,1]),a[,1])
        a1 = a[c1,]
        c2 = match(unique(a1[,2]),a1[,2])
        a2 = a1[c2,]
        a2 = na.omit(a2)
        u = match(a2[,"ENTREZID"],entrez)
        v = match(a2[,1],featureNames(x0))
        df[u,(ncol0+1):(ncol0+2)]= as.matrix(x1[v,1:2])
        names0 = c(names0,paste0(paste0(c("mc","beta"),dts),type,nsim))
        rm(x1)
        ncol0 = ncol0 + 2
      }
    }
  }
  load(paste0(outputDir,"x1/x1_","limma_patient_",experiment,".rda"))
  v = match(a2[,1],x1[,"PROBEID"])
```

```
  df[u,ncol(df)] = as.matrix(x1[v,"P.Value"])
  names(df) = names0
  save(df,file=paste0("results/Experiments/Experiment",experiment,".rda"))
}
```

```
experiments = experimentsNonPaired
for(experiment in experiments){
  df = matrix(NA,ncol=length(nsims)*2*2+2,nrow=length(entrez))
  df = as.data.frame(df)
  rm(x1)
  cat("Experiment ",experiment,"\n")
  load("entrez.rda")
  df[,1] = entrez
  ncol0 = 1
  names0 = "entrez"
  for(dts in c("normal","tstudent")){
    cat("dts ",dts,"\n")
    for(nsim in nsims){
      cat("nsim ",nsim,"\n")
      for(type in "complete"){
        rm(x1)
        load(paste0(outputDir,"x1/x1_",dts,"_",type,"_",nsim,"_",experiment,
                    ".rda"))
        res0 = matrix(NA,nrow=length(entrez),ncol=5)
        load(pathData[experiment])
        x0 = get(id[experiment,1])
        a = AnnotationDbi::select(get(id[experiment,3]),
                                  keys=featureNames(x0),
                                  columns="ENTREZID",keytype="PROBEID")
        c1 = match(unique(a[,1]),a[,1])
        a1 = a[c1,]
        c2 = match(unique(a1[,2]),a1[,2])
        a2 = a1[c2,]
        a2 = na.omit(a2)
        u = match(a2[,"ENTREZID"],entrez)
        v = match(a2[,1],featureNames(x0))
        df[u,(ncol0+1):(ncol0+2)]= as.matrix(x1[v,1:2])
        names0 = c(names0,paste0(paste0(c("mc","beta"),dts),type,nsim))
        rm(x1)
        ncol0 = ncol0 + 2
      }
    }
  }
  load(paste0(outputDir,"x1/x1_","limma_patient_",experiment,".rda"))
  v = match(a2[,1],x1[,"PROBEID"])
  df[u,ncol(df)] = as.matrix(x1[v,"P.Value"])
  names(df) = names0
  save(df,file=paste0("results/Experiments/Experiment",experiment,".rda"))
}
```

1.6 Selected genes using previous literature

Let us consider some selected genes using literature. First, some plots for given genes.

```
plotting = function(entrezid,df,which_plot=1,
                    dts=c("normal","tstudent"),
                    type=c("complete","between-pair"),
                    method=c("beta","mc")){
  ## Plot 1
  if(which_plot == 1){
    s0 = grep(paste0(paste0(method,dts),type),names(df))
    u0 =  df[which(entrezid==df$entrez),s0]
    df1 = data.frame(nsims,t(u0))
    names(df1) = c("nsim","p")
    p = ggplot(df1,aes(x=nsim,y=p))+geom_point() + ylim(0,1)
    return(p)
    }

  ## Plot 2
  if(which_plot == 2){
    s0 = grep(paste0(paste0("beta",dts),type),names(df))
    s1 = grep(paste0(paste0("mc",dts),type),names(df))
    u0 =  df[which(entrezid==df$entrez),s0]
    u1 = df[which(entrezid==df$entrez),s1]
    df1 = data.frame(c(nsims,nsims),c(t(u0),t(u1)),
                     rep(c("beta","mc"),each=length(nsims)))
    names(df1) = c("nsim","p","method")
    p = ggplot(df1,aes(x=nsim,y=p,color=method))+geom_point() +  ylim(0,1)
    return(p)
  }
}
```

```
nsims=seq(10,1090,10)
dtss=c("normal","tstudent")
types=c("complete","between-pair")
methods=c("beta","mc")

## Selected genes: entrez identifiers
entrezids = c(960,10562,3934,7076,4233,50506,4609,27299,2920,2919,
              140803,343,2980,760,10351,759,6387,125)

## Paired experiments
for(experiment in experimentsPaired){
  load(paste0("results/Experiments/Experiment",experiment,".rda"))
  for(entrezid in entrezids){
    for(dts in dtss){
      for(type in types){
        for(method in methods){
          for(which_plot in 2){
            p = plotting(df=df,entrezid=entrezid,which_plot=which_plot,
                         dts=dts,type=type,
```

```
                          method=method)
            name0 = paste0(paste("figures/selectedGenes/experiment",
                                 experiment,dts,type,
                                 method,entrezid,which_plot,sep="_"),".png")
            ggsave(name0,p)
          }
        }
      }
    }
  }
}

## Non paired experiments
for(experiment in experimentsNonPaired){
  load(paste0("results/Experiments/Experiment",experiment,".rda"))
  for(entrezid in entrezids){
    for(dts in dtss){
      type = "complete"
      for(method in methods){
        for(which_plot in 2){
          p = plotting(df=df,entrezid=entrezid,which_plot=which_plot,
                       dts=dts,type=type,
                       method=method)
          name0 = paste0(paste("figures/selectedGenes/experiment",
                               experiment,dts,type,
                               method,entrezid,which_plot,sep="_"),".png")
          ggsave(name0,p)
        }
      }
    }
  }
}
```

We choose a given number of simulations and generate a html report with the scores values for all experiments. We choose the complete distribution in order to use all the experiments.

```
library(ReportingTools)
nsims=seq(10,1090,10)
dtss=c("normal","tstudent")
types=c("complete","between-pair")
methods=c("beta","mc")

nsim=300

df_all = matrix(NA,nrow=length(entrez),
                ncol=length(experiments)*length(dtss)*length(methods))
j = 0
names0 = NULL
for(experiment in experiments){
  load(paste0("results/Experiments/Experiment",experiment,".rda"))
  dim(df)
  for(dts in dtss){
    type = "complete"
```

```
    for(method in methods){
      j = j+1
      cat("j ",j,"\n")
      s0 = grep(paste0(paste0(method,dts),type,nsim),names(df))[1]
      df_all[,j] = df[,s0]
      names0 = c(names0,
                 paste0(paste0(method,dts),type,nsim,"_",experiment))
    }
  }
}
df_all = data.frame(entrezid2url(entrez),df_all)
names(df_all) = c("entrez",names0)
foutput = paste0("p-values_",nsim)
htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                      reportDirectory = "./reports")
publish(df_all,htmlRep1)
finish(htmlRep1)
```

1.7 Evaluating parameters

Now, we evaluate the use of the score or the original Montecarlo p-value. We use diffferent number of simulations (from 100 to 1000) and we compare the score with the Montecarlo p-value for all genes studied. Each line corresponds with a different experiment.

```
nsims=seq(10,1090,10)
dtss=c("normal","tstudent")
types=c("complete","between-pair")
methods=c("beta","mc")

type = "complete"
for(nsim in seq(100,1000,100)){
  for(dts in dtss){
    df2 = NULL
    for(experiment in experiments){
      load(paste0("results/Experiments/Experiment",experiment,".rda"))
      s0 = grep(paste0("beta",dts,type,nsim),names(df))[1]
      s1 = grep(paste0("mc",dts,type,nsim),names(df))[1]
      df1 = data.frame(beta= df[,s0],mc = df[,s1],rep(experiment,
                                                      length(entrez)))
      df2 = rbind(df2,df1)
  }
    names(df2) = c("beta","mc","experiment")
    (p = ggplot(df2,aes(x=beta,y=mc,color=factor(experiment)))+
       geom_smooth(method="locfit") +
       geom_abline(intercept=0,slope=1))
    ggsave(file=paste0("figures/comparisons/methods",dts,type,nsim,".png"),p)
  }
}
```

We compare for the score the differences using the normal of t-Student distribution functions.

```
nsims=seq(10,1090,10)
dtss=c("normal","tstudent")
types=c("complete","between-pair")
methods=c("beta","mc")

type = "complete"
for(nsim in seq(100,1000,100)){
  for(method in methods){
    df2 = NULL
    for(experiment in experiments){
      load(paste0("results/Experiments/Experiment",experiment,".rda"))
      s0 = grep(paste0(method,"normal",type,nsim),names(df))[1]
      s1 = grep(paste0(method,"tstudent",type,nsim),names(df))[1]
      df1 = data.frame(beta= df[,s0],mc = df[,s1],rep(experiment,
                                                      length(entrez)))
      df2 = rbind(df2,df1)
  }
    names(df2) = c("normal","tstudent","experiment")
    (p = ggplot(df2,aes(x=normal,y=tstudent,color=factor(experiment)))+
       geom_smooth(method="locfit") +
       geom_abline(intercept=0,slope=1))
    ggsave(file=paste0("figures/comparisons/dtss",method,type,nsim,".png"),p)
  }
}
```

```
## Error in eval(expr, envir, enclos): objeto 'experiments'
no encontrado
```

1.8 Meta-analysis

We choose our score with `normal`, `complete` and 100, 300 and 500 simulations for all experiments.

```
nsims=seq(10,1090,10)
dtss=c("normal","tstudent")
types=c("complete","between-pair")
methods=c("beta","mc")

method = "beta"
dts = "normal"
type = "complete"
nsim = 300

for(nsim in c(100,300,500)){
  names0 = NULL
  df1 = matrix(NA,nrow=length(entrez),ncol=length(experiments))
  j = 0
  for(experiment in experiments){
    j = j+1
    load(paste0("results/Experiments/Experiment",experiment,".rda"))
    s0 = grep(paste0(method,dts,type,nsim),names(df))[1]
    df1[,j] = df[,s0]
    names0 = c(names0,paste0("experiment",experiment))
```

```
  }
  names(df1) = names0
  View(df1)
  dim(df1)
  length(entrez)
  head(df1)
  df2 = data.frame(df1,apply(df1,1,min,na.rm=TRUE),
                   apply(df1,1,median,na.rm=TRUE),
                   apply(df1,1,max,na.rm=TRUE))
  names0 = c(names0,"min","median","max")

  df2 = data.frame(entrezid2url(entrez),df2)
  names(df2) = c("entrez",names0)

  foutput = paste0("Meta_beta_normal_complete_",nsim)
  htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                        reportDirectory = "./reports")
  publish(df2,htmlRep1)
  finish(htmlRep1)
}
```

1.9 Simulation study

In this section we present a simulation study with the following steps.

1. A previous false discovery rate (FDR) will be previously given.
2. For a chosen model we will generate a realization.
3. The Montecarlo p-value, our score and the p-value of the moderated t-test of Limma will be calculated. We will use 100 simulations for Montecarlo p-value and the score.
4. The Benjamini-Hochberg correction will be applied to each p-value.
5. The declared significant features will be compared with the (real) significant features.
6. Steps 2 to 5 will be repeated `nrep` times.

We simulate a model using **normal distributions**.

```
outputDir = "~/Nextcloud/BetaMonteCarlo20/prog/results/"
.mu0=20
## ¿Utilizando los arrays observar las desviaciones estándar?
.sd0=.sd1=.sd2=1
.N1=100
.N2=100
.N3=800
.nrep = 100

n1s = 10:20
deltas = seq(.01,4,.02)
mu1s = .mu0 + deltas
```

```
nsims = seq(100,1000,100)
alphas = .alphas = seq(.001,.05,.001)

for(.nsim in nsims){
  cat("nsim ",.nsim,"\n")
  df = matrix(NA,nrow=length(alphas)*length(deltas)*length(n1s),ncol=9)
  countRow = 0
  for(delta in deltas){
    cat("delta ",delta,"\n")
    .mu1 = .mu0 + delta
    .mu2 = .mu0 + delta
    for(.n1 in n1s){
      mu0=.mu0;sd0=.sd0;mu1=.mu1;sd1=.sd1;mu2=.mu2;
      sd2=.sd2;n1=.n1;n2=.n1;N1=.N1;N2=.N2;N3=.N3;
      alphas=.alphas;nsim=.nsim;nrep=.nrep
      res = doReplication(mu0=.mu0,sd0=.sd0,mu1=.mu1,sd1=.sd1,mu2=.mu2,
                          sd2=.sd2,n1=.n1,n2=.n1,N1=.N1,N2=.N2,N3=.N3,
                          alphas=.alphas,nsim=.nsim,nrep=.nrep,model="Normal")
      toInsert = 1:length(alphas) + countRow * length(alphas)
      df[toInsert,1] = rep(delta,length(alphas))
      df[toInsert,2] = rep(.n1,length(alphas))
      df[toInsert,3:9] = res
      countRow = countRow + 1
    }
  }
  df = as.data.frame(df)
  names(df) = c("delta","n1","alpha","mc1","mc2","score1","score2","l1","l2")
  save(df,file=paste0(outputDir,
                      "simulationStudy/Normal/df_Normal_nsim_",
                      .nsim,".rda"))
}
```

Now a model using **gamma distributions**.

```
outputDir = "~/Nextcloud/BetaMonteCarlo20/prog/results/"
.mu0=20
.sd0=.sd1=.sd2=1
.N1=100
.N2=100
.N3=800
.nrep = 100

n1s = 10:20
deltas = seq(.01,4,.02)
mu1s = .mu0 + deltas
nsims = seq(100,1000,100)
alphas = .alphas = seq(.001,.05,.001)

for(.nsim in nsims){
  cat("nsim ",.nsim,"\n")
  df = matrix(NA,nrow=length(.alphas)*length(deltas)*length(n1s),ncol=9)
  countRow = 0
  for(delta in deltas){
    cat("delta ",delta,"\n")
```

```
    .mu1 = .mu0 + delta
    .mu2 = .mu0 + delta
    for(.n1 in n1s){
      mu0=.mu0;sd0=.sd0;mu1=.mu1;sd1=.sd1;mu2=.mu2;
      sd2=.sd2;n1=.n1;n2=.n1;N1=.N1;N2=.N2;N3=.N3;
      alphas=.alphas;nsim=.nsim;nrep=.nrep
      res = doReplication(mu0=.mu0,sd0=.sd0,mu1=.mu1,sd1=.sd1,mu2=.mu2,
                          sd2=.sd2,n1=.n1,n2=.n1,N1=.N1,N2=.N2,N3=.N3,
                          alphas=.alphas,nsim=.nsim,nrep=.nrep,model="Gamma")
      toInsert = 1:length(alphas) + countRow * length(alphas)
      df[toInsert,1] = rep(delta,length(alphas))
      df[toInsert,2] = rep(.n1,length(alphas))
      df[toInsert,3:9] = res
      countRow = countRow + 1
    }
  }
  df = as.data.frame(df)
  names(df) = c("delta","n1","alpha","mc1","mc2","score1","score2","l1","l2")
  save(df,file=paste0(outputDir,"simulationStudy/Gamma/df_Gamma_nsim_",
                      .nsim,".rda"))
}
```

```
load(paste0(outputDir,"simulationStudy/Gamma/df_Gamma_nsim_100.rda"))
```

```
nsim = 100
type = "Normal"
for(nsim in c(100,200)){
  for(type in c("Normal","Gamma")){
    rm(df)
    load(paste0("results/simulationStudy/",type,"/df_",type,"_nsim_",
                nsim,".rda"))
    deltas = sort(unique(df[,"delta"]))
    n1s = sort(unique(df[,"n1"]))
    alphas = sort(unique(df[,"alpha"]))

    alphas0 = alphas[c(1,length(alphas))]
    deltas0 = deltas[1:5]
    n1s0 = n1s[c(1,length(n1s))]
    ## delta, alpha given
    for(delta in deltas0){
      for(alpha in alphas0){
        df1 = df[df$delta == delta & df$alpha == alpha,]
        df2 = df1[,c("n1","mc1","score1","l1")]
        df3 = reshape2::melt(df2,id.vars="n1")
        p = ggplot(df3,aes(x=n1,y=value,color=variable))+geom_line()
        ggsave(paste0("figures/simulationStudy/type1_",type,"_",nsim,
                      "_delta_",100*delta,"_alpha_",1000*alpha,"_n1_all.png"),p)

        df2 = df1[,c("n1","mc2","score2","l2")]
        df3 = reshape2::melt(df2,id.vars="n1")
        p = ggplot(df3,aes(x=n1,y=value,color=variable))+geom_line()
        ggsave(paste0("figures/simulationStudy/type2_",type,"_",nsim,
```

```
                    "_delta_",100*delta,"_alpha_",1000*alpha,"_n1_all.png"),p)
      }
    }

    ## delta, n1 given
    for(delta in deltas0){
      for(n1 in n1s0){
        df1 = df[df$delta == delta & df$n1 == n1,]
        df2 = df1[,c("alpha","mc1","score1","l1")]
        df3 = reshape2::melt(df2,id.vars="alpha")
        p= ggplot(df3,aes(x=alpha,y=value,color=variable))+geom_line()
        ggsave(paste0("figures/simulationStudy/type1_",type,"_",nsim,
                    "_delta_",100*delta,"_alpha_all","_n1_",n1,".png"),p)

        df2 = df1[,c("alpha","mc2","score2","l2")]
        df3 = reshape2::melt(df2,id.vars="alpha")
        p = ggplot(df3,aes(x=alpha,y=value,color=variable))+geom_line()
        ggsave(paste0("figures/simulationStudy/type2_",type,"_",nsim,
                    "_delta_",100*delta,"_alpha_all","_n1_",n1,".png"),p)
      }
    }
    ## alpha, n1 given
    for(alpha in alphas0){
      for(n1 in n1s0){
        df1 = df[df$alpha == alpha & df$n1 == n1,]
        df2 = df1[,c("delta","mc1","score1","l1")]
        df3 = reshape2::melt(df2,id.vars="delta")
        p=ggplot(df3,aes(x=delta,y=value,color=variable))+geom_line()
        ggsave(paste0("figures/simulationStudy/type1_",type,"_",nsim,
                    "_delta_all_alpha_",1000*alpha,"_n1_",n1,".png"),p)

        df2 = df1[,c("delta","mc2","score2","l2")]
        df3 = reshape2::melt(df2,id.vars="delta")
        p=ggplot(df3,aes(x=delta,y=value,color=variable))+geom_line()
        ggsave(paste0("figures/simulationStudy/type2_",type,"_",nsim,
                    "_delta_all_alpha_",1000*alpha,"_n1_",n1,".png"),p)
      }
    }

  }
}
```

1.9.1 Animations

First, some animations for the simulation study using normal distributions.

```
doGif = FALSE
outputDir = "~/Nextcloud/BetaMonteCarlo20/prog/results/"
n1s = 10:20
deltas = seq(.01,4,.02)
alphas = .alphas = seq(.001,.05,.001)
```

```
df3 = NULL
for(nsim in c(100,200)){
  for(model0 in c("Normal","Gamma")){
    load(paste0(outputDir,"simulationStudy/",model0,"/df_",model0,"_nsim_",
                nsim,".rda"))
    for(n10 in n1s){
      df0 = df[df$n1 == n10,]
      ## Type 1 error
      p = ggplot(df0,aes(x=delta,y=score1,colour=alpha))+ geom_point()
      p1 = p + transition_time(alpha)
      filename = paste0("df_",model0,"_nsim_",nsim,"_n1_",n10,
                        "_score_",1,".gif")
      file0 = paste0(outputDir,"simulationStudy/Normal/",filename)
      if(doGif) anim_save(file=file0,p1)
      df3 = rbind(df3,c(model0,nsim,n10,1,filename))


      ## Type 2 error
      p = ggplot(df0,aes(x=delta,y=score2,colour=alpha))+ geom_point()
      p1 = p + transition_time(alpha)
      filename = paste0("df_",model0,"_nsim_",nsim,"_n1_",n10,
                        "_score_",2,".gif")
      file0 = paste0(outputDir,"simulationStudy/Normal/",filename)
      if(doGif) anim_save(file=file0,p1)
      df3 = rbind(df3,c(model0,nsim,n10,2,filename))


      ## Type 1 and 2 errors
      p = ggplot(df0,aes(x=delta,y=score2,colour=alpha))+ geom_point()
      p = p + geom_point(aes(x=delta,y=score1,colour=alpha))
      p1 = p + transition_time(alpha)
      filename = paste0("df_",model0,"_nsim_",nsim,"_n1_",n10,
                        "_score_",3,".gif")
      file0 = paste0(outputDir,"simulationStudy/Normal/",filename)
      if(doGif) anim_save(file=file0,p1)
      df3 = rbind(df3,c(model0,nsim,n10,3,filename))

      }
      ## Type 1 errors for all n1s
      df0 = df
      p = ggplot(df0,aes(x=delta,y=score1,color=n1))+ geom_point()
      p1 = p + transition_time(alpha)
      filename = paste0("df_",model0,"_nsim_",nsim,"_score_",1,".gif")
      file0 = paste0(outputDir,"simulationStudy/Normal/",filename)
      if(doGif) anim_save(file = file0,p1)
      df3 = rbind(df3,c(model0,nsim,NA,1,filename))


    ## Type 2 errors for all n1s
      df0 = df
      p = ggplot(df0,aes(x=delta,y=score2,color=n1))+ geom_point()
      p1 = p + transition_time(alpha)
      filename = paste0("df_",model0,"_nsim_",nsim,"_score_",2,".gif")
      file0 = paste0(outputDir,"simulationStudy/Normal/",filename)
```

```
      if(doGif) anim_save(file=file0,p1)
    df3 = rbind(df3,c(model0,nsim,NA,2,filename))
  }
}
save(df3,file="df3.rda")
```

```
load("df3.rda")
df3[,5] = ifelse(df3[,5] == "NA",NA,paste("<a href='file:",df3[,5],"'>",
                                          df3[,5],"</a>",sep=""))
foutput = "zz"
df3 = data.frame(df3)
htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
reportDirectory = "./reports/SimulationStudy")
publish(df3,htmlRep1)
finish(htmlRep1)
```

Now we compare score with Montecarlo p-values.

```
outputDir = "~/Nextcloud/BetaMonteCarlo20/prog/results/"
n1s = 10:20
deltas = seq(.01,4,.02)
alphas = .alphas = seq(.001,.05,.001)

df4 = NULL
for(nsim in c(100,200)){
  for(model0 in c("Normal","Gamma")){
    load(paste0(outputDir,"simulationStudy/",model0,"/df_",model0,"_nsim_",
                nsim,".rda"))
    df0 = df
    ## Type 1 error
    p = ggplot(df0,aes(x=delta,y=score1,colour="red"))+ geom_point()
    p = p + geom_point(aes(x=delta,y=mc1,colour="blue"))
    p1 = p + transition_time(alpha)
    file0 = paste0(outputDir,"simulationStudy/Normal/df_",
                   model0,"_nsim_",nsim,"_score_",1,"_mc_",1,".gif")
    anim_save(file = file0,p1)
    df4 = rbind(df4,c(model0,nsim,1,file0))

    ## Type 2 error
    p = ggplot(df0,aes(x=delta,y=score2,colour="red"))+ geom_point()
    p = p + geom_point(aes(x=delta,y=mc2,colour="blue"))
    p1 = p + transition_time(alpha)
    file0 = paste0(outputDir,"simulationStudy/Normal/df_",
                   model0,"_nsim_",nsim,"_score_",2,"_mc_",2,".gif")
    anim_save(file=file0,p1)
    df4 = rbind(df4,c(model0,nsim,1,file0))
  }
}
```

Chapter 2

RNA-Seq

```
pacman::p_load(OMICfpp2,SummarizedExperiment,ReportingTools)
```

```
data(TCGA,package="OMICfpp2")
for(nsim in c(100,300,500)){
  TCGA_score = intp(x=as.matrix(limma::voom(assay(TCGA))),
                    y=data.frame(colData(TCGA)),
                    type="complete",paired=TRUE,nsim=nsim,
                    nboots=10,fboot=.6)
  save(TCGA_score,file=paste0("TCGA_score_complete_",nsim,".rda"))
}
```

```
data(PRJNA218851,package="OMICfpp2")
for(nsim in c(100,300,500)){
  PRJNA218851_score = intp(x=as.matrix(limma::voom(assay(PRJNA218851))),
                           y=data.frame(colData(PRJNA218851)),
                           type="complete",paired=TRUE,nsim=nsim)
  save(PRJNA218851_score,file=paste0("PRJNA218851_score_complete_",nsim,".rda"))
}
```

```
data(PRJNA413956,package="OMICfpp2")
for(nsim in c(100,300,500)){
  PRJNA413956_score = intp(x=as.matrix(limma::voom(assay(PRJNA413956))),
                           y=data.frame(colData(PRJNA413956)),
                           type="complete",paired=TRUE,nsim=nsim)
  save(PRJNA413956_score,file=paste0("PRJNA413956_score_complete_",nsim,".rda"))
}
```

```
data(TCGA,package="OMICfpp2")
for(nsim in c(100,300,500)){
  load(paste0("TCGA_score_complete_",nsim,".rda"))
  df = data.frame(ensembl = ensembl2url(rowData(TCGA)$X),
                  TCGA_score)
  foutput = paste0("TCGA_score_complete_",nsim)
  htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                        reportDirectory = "./reports")
```

```
  publish(df,htmlRep1)
  finish(htmlRep1)
}
```

```
data(PRJNA218851,package="OMICfpp2")
for(nsim in c(100,300,500)){
  load(paste0("PRJNA218851_score_complete_",nsim,".rda"))
  x = PRJNA218851_score$BetaNormal
  df = data.frame(ensembl = ensembl2url(rowData(PRJNA218851)$X),
                  PRJNA218851_score)
  foutput = paste0("PRJNA218851_score_complete_",nsim)
  htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                        reportDirectory = "./reports")
  publish(df,htmlRep1)
  finish(htmlRep1)
}
```

```
data(PRJNA413956,package="OMICfpp2")
for(nsim in c(100,300,500)){
  load(paste0("PRJNA413956_score_complete_",nsim,".rda"))
  x = PRJNA413956_score$BetaNormal
  df = data.frame(ensembl = ensembl2url(rowData(PRJNA413956)$X),
                  PRJNA413956_score)
  foutput = paste0("PRJNA413956_score_complete_",nsim)
  htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                        reportDirectory = "./reports")
  publish(df,htmlRep1)
  finish(htmlRep1)
}
```

Chapter 3

Methylation

We can perform a differential methylation analysis.

```
pacman::p_load(minfi,SummarizedExperiment,IlluminaHumanMethylationEPICmanifest)
dirTamiData = "/home/gag/ownCloud/alltami/tami-data/"
```

```
gcel = GEOquery::getGEOSuppFiles("GSE149282")
path = paste0(dirTamiData,"GSE149282_RAW/")
(targetsBasename = read.csv(paste0(path,"idats.csv"),head=TRUE,sep=","))
targetsBasename = DataFrame(targetsBasename)
targets = paste0(path,as.character(targetsBasename$File_name))
rgset = minfi::read.metharray(targets, verbose = TRUE, force=TRUE)
colData(rgset) = targetsBasename
mset = preprocessIllumina(rgset)
GSE149282 = mapToGenome(mset)
save(GSE149282,file=paste0(dirTamiData,"GSE149282.rda"))
```

```
pacman::p_load(limma)
load(paste0(dirTamiData,"GSE149282nFlt.rda"))
## calculate M-values for statistical analysis
mVals = getM(GSE149282nFlt)
load(paste0(dirTamiData,"GSE149282.rda"))
design = model.matrix(~factor(GSE149282$Pair) + factor(GSE149282$Status))
# fit the linear model
fit = limma::lmFit(mVals, design)
# Empirical Bayes
fit2 = limma::eBayes(fit)
DMPs = limma::topTable(fit2,number=nrow(GSE149282),p.value=1,coef=ncol(design),
                       sort.by="none")
save(DMPs,file=paste0(dirTamiData,"DMPs.rda"))
```

```
load(paste0(dirTamiData,"GSE149282nFlt.rda"))
mVals = getM(GSE149282nFlt)
load(paste0(dirTamiData,"DMPs.rda"))
```

```
pacman::p_load(OMICfpp2,ggplot2)
for(nsim in c(100,300,500)){
  y0 = colData(GSE149282nFlt)[,c("Pair","Status")]
  y = cbind(as.numeric(y0[,1]),(y0[,2] == "cancer")*1)
  GSE149282_score  = intp(x=mVals,y=y,paired=TRUE,nsim=nsim)
  save(GSE149282_score,file=paste0("GSE149282_score_complete_",nsim,".rda"))
}
```

```
for(nsim in c(100,300,500)){
  load(paste0("GSE149282_score_complete_",nsim,".rda"))
  summary(GSE149282_score$BetaNormal)
  df = data.frame(p.limma = DMPs[,"P.Value"],
                  p.intp = GSE149282_score$BetaNormal)
  p = ggplot(df,aes(x=p.limma,y=p.intp))+geom_point(alpha=1/100)
  cor(df)
  ggsave(paste0("figures/methylation/GSE149282_limma_intp_",nsim,".png"),p)
  }
```

3.1 Ties

Now we evaluate the proportion of ties.

```
library(ReportingTools)
data(TCGA,package="OMICfpp2")
data(PRJNA218851,package="OMICfpp2")
data(PRJNA413956,package="OMICfpp2")
experimentsDir = "results/Experiments/"


dfTies = data.frame(experiment=rep(NA,(length(experiments)+4)*3),
                    nsim=rep(NA,(length(experiments)+4)*3),
                    unique = rep(NA,(length(experiments)+4)*3),
                    total = rep(NA,(length(experiments)+4)*3),
                    zeros = rep(NA,(length(experiments)+4)*3))
```

```
## Error in data.frame(experiment = rep(NA, (length(experiments)
+ 4) * 3), : objeto 'experiments' no encontrado
```

```
cont = 0
for(nsim in c(10,25,50)){
  for(experiment in experiments){
    cont = cont + 1
    load(paste0(experimentsDir,"Experiment",experiment,".rda"))
    s0 = grep(paste0("beta","normal","complete",nsim),names(df))[1]
    x1 = df[,s0]
    x1 = na.omit(x1)
    dfTies[cont,] = c(paste0("experiment",experiment),nsim,length(x1),
                      length(unique(x1)),sum(x1==0))
  }
  load(paste0(experimentsDir,"TCGA_score_complete_",nsim,".rda"))
  cont = cont + 1
  x1 = TCGA_score$BetaNormal
```

```
  x1 = na.omit(x1)
  dfTies[cont,] = c("TCGA",nsim,length(x1),length(unique(x1)),sum(x1==0))
  load(paste0(experimentsDir,"PRJNA218851_score_complete_",nsim,".rda"))
  cont = cont + 1
  x1 = PRJNA218851_score$BetaNormal
  x1 = na.omit(x1)
  dfTies[cont,] = c("PRJNA218851",nsim,length(x1),length(unique(x1)),sum(x1==0))
  load(paste0(experimentsDir,"PRJNA413956_score_complete_",nsim,".rda"))
  cont = cont + 1
  x1 = PRJNA413956_score$BetaNormal
  x1 = na.omit(x1)
  dfTies[cont,] = c("PRJNA413956",nsim,length(x1),length(unique(x1)),sum(x1==0))
  load(paste0(experimentsDir,"GSE149282_score_complete_",nsim,".rda"))
  cont = cont + 1
  x1 = GSE149282_score$BetaNormal
  x1 = na.omit(x1)
  dfTies[cont,] = c("GSE149282",nsim,length(x1),length(unique(x1)),sum(x1==0))
}
```

```
## Error in eval(expr, envir, enclos): objeto 'experiments'
no encontrado
```

```
foutput = "ties"
htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                      reportDirectory = "./reports")
publish(dfTies,htmlRep1)
```

```
## Error in h(simpleError(msg, call)): error in evaluating
the argument 'object' in selecting a method for function 'publish':
objeto 'dfTies' no encontrado
```

```
finish(htmlRep1)
```

```
## [1] "./reports/ties.html"
```

Bibliography

[1] Gordon Smyth et al. *limma: Linear Models for Microarray Data*. R package version 3.38.3. 2018. URL: http://bioinf.wehi.edu.au/limma.