Power Analysis of Significance Tests in Metabolomics

Created by: Huaxu Yu

March 16, 2021

This document contains all the code that is mentioned in the method section. The document is an RMarkdown document which means that it can be compiled, along with the code chunks thus executing and capturing the output of the code within the document. To read more about RMarkdown see the website for the package, as well as the Get Started guide.

Dependencies

Input serial diluted urine metabolomics data. Set the working directory containing file "model_selection.csv"

```
urine_data = read.csv("model_selection.csv")
```

This document comes with a list of required packages.

1. `polynom`: polynomial regression.
2. `e1071`: skewness calculation
3. `effsize`: effect size calculation
4. `pwr`: t test power calculation
5. `coin`: permutation test

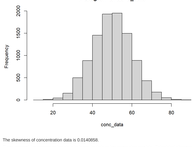
Skewness Calculation

First, 10000 data points are sampled from a normal distribution (mean = 50, sd = 10)

```
data_number = 10000
conc_data = rnorm(data_number,50,10)
```

Check the data normality of concentration data

```
hist(conc_data, breaks = 20)
```



The skewness of concentration data is 0.0140656.

Then, convert the concentration data to intensity data using regression models. Calculated skewness values for all metabolic features individually:

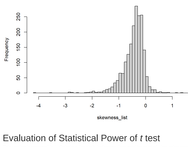
```
skewness_list = c()
int_data = c()
all_conc_seq = seq(10,100,10)

for (i in 1:nrow(urine_data)) {
  #m_model = lm(as.numeric(urine_data[i,6:15]) ~ poly(all_conc_seq, urine_data$Mode[i], raw = T))
  #m_coeff = #m_model$coefficients

  int_data = predict(polynomial(#m_coeff),conc_data)
  skewness_list[i] = skewness(int_data)
}
```

Check the distribution of skewness values

```
hist(skewness_list, breaks = 50)
```



Evaluation of Statistical Power of *t* test

This section is prepared to 1. Calculate the statistical power (true positive rate) of t-test using concentration data. The concentration data follow normal distribution. 2. Assess the statistical power of t-test using converted intensity data by Monte Carlo method.

```
input_file = read.csv("model_selection.csv")
#Column 1-5: Alignment number, retention time, m/z value, metabolite name, adduct type
#Column 6-15: MS intensities of serially diluted urine samples.
#Column 16: Regression model, 2 for quadratic, 3 for cubic.
#1941 features in total

#Parameter settings
#Mean values of two normally distributed populations (concentration data)
mean_1 = 50
mean_2 = 60

#Number of data in each group (sample size). In manuscript, we investigated 5, 15 and 30
sample_size = 15

#Repeated simulation times in Monte Carlo method section
repeat_times = 1000

#Effect sizes to be tested
eff_seq = seq(0.2,2.0,1)

#Relation concentrations of serial urine samples
all_conc_seq = seq(10,100,10)
```

Calculate the statistical power (true positive rate) of *t* test using concentration data. By default, the power is calculated using a value cutoff at 0.01.

```
#Generate a vector to store statistical power
TPR_conc = rep(0,length(eff_seq))

#Calculate statistical power for all effect sizes, in %
for (i in 1:length(eff_seq)) {
  TPR_conc[i]=pwr.t.test(d = eff_seq[i], sig.level = 0.01, n = sample_size, type = c("two.sample"))$power*100
}
```

Generate a data matrix to store the statistical power after data simulation. Each row represents one metabolic feature, which shows a unique nonlinear ESI response pattern. Each column represents one effect size from 0.2 to 2.0.

```
TPR_vs_Efffsize = data.frame(matrix(nrow = 1941, ncol = length(eff_seq)))
colnames(TPR_vs_Efffsize) = eff_seq

#Calculate the statistical power of t-test using converted intensity data for all metabolic features (1941 in tot
al), in %
for (model_number in 1:1941) {
  #Regression models
  #m_model = poly(as.numeric(input_file[model_number,6:15]) ~ poly(all_conc_seq, input_file$Mode[model_number], raw =
T))
  #m_coeff = #m_model$coefficients

  #Run data simulation for each effect size
  for (i in 1:length(eff_seq)) {
    #Find the standard deviation based on mean value and effect size
    sd_sam2 = (mean_2 - mean_1)/eff_seq[i]

    #Store p values from data simulation
    p_int = c()
    #Store statistical power from data simulation
    TPR_trans = rep(0,15)

    #Repeat the entire data simulation for 15 times to acquire reproducible result
    for (k in 1:15) {
      #Monte Carlo data simulation module
      for (r in 1:repeat_times) {
        #Sample two data sets from two normal distributions. These two data sets are recognized as concentration
data
        data1 = rnorm(sample_size,mean_1,sd_sam2)
        data2 = rnorm(sample_size,mean_2,sd_sam2)

        #Convert concentration data to intensity data using nonlinear regression model
        trans_data1 = predict(polynomial(#m_coeff),data1)
        trans_data2 = predict(polynomial(#m_coeff),data2)

        #Run t-test and get p value
        p_int_trans[r] = t.test(trans_data1,trans_data2,var.equal = T, paired = F)$p.value

        #After each round of Monte Carlo data simulation, the statistical power is calculated and stored.
        TPR_trans[k] = mean(p_int_trans < 0.01)
      }
    }
    #Statistical power is calculated by taking average of 15 replicates
    TPR_vs_Efffsize[model_number,i] = mean(TPR_trans)*100
  }

  #Show the progress of data simulation
  print(paste0("Feature number ", model_number, " has been finished. 1941 in total"))
}
```

```
write.csv(TPR_vs_Efffsize,"TPR_vs_Efffsize.csv")
```

To calculate the maximum difference of statistical power (Figure 3B, 3D and 3E)

```
#Generate a vector to store the max difference of statistical power
max_diff_seq = rep(0,1941)

#Do calculation for all features
for (i in 1:nrow(TPR_vs_Efffsize)) {
  diff = TPR_conc - TPR_vs_Efffsize[i,]
  seq_number = match(max(abs(diff)),abs(diff))
  max_diff_seq[i] = as.numeric(diff[seq_number])
}


write.csv(max_diff_seq,"max_diff_seq.csv")
```

Correlation Between Reduced Statistical Power and Nonlinear ESI Response

This section is prepared to

1. Calculate W value and squared skewness value (skewness^2) for each metabolic feature after converting concentration data to intensity data
2. Calculate the correlation between maximum TPR difference and W
3. Calculate the correlation between maximum TPR difference and skewness^2.

Calculate W value and squared skewness value (skewness^2)

```
input_file = read.csv("model_selection.csv")

# 10000 data points are sampled from a normal distribution (mean = 50, sd = 10)
data_number = 10000
all_conc_seq = seq(10,100,10)
conc_data = rnorm(data_number,50,10)

#Generate vectors to store the W values and squared skewness values
W_list = c()
skewness2_list = c()

int_data = vector(length = data_number)

for (i in 1:nrow(input_file)) {
  #Obtain regression model
  #m_model = lm(as.numeric(input_file[i,6:15]) ~ poly(all_conc_seq, input_file$[i], raw = T))
  #m_coeff = #m_model$coefficients

  int_data = predict(polynomial(#m_coeff),conc_data)
  #Run Shapiro-Wilk normality test and record sample size between 3 to 5000, we randomly sampled 5000 data
  # from 10000 for the W value calculation
  W_list[i] = shapiro.test(sample(int_data, 5000))$statistic
  skewness2_list = skewness(int_data)
}

write.csv(W_list,"W-value.csv")
write.csv(skewness2_list,"squared skewness value.csv")
```

Calculate correlations

```
# to W value
W_cor = cor(max_diff_seq, W_list)

# to skewness^2 value
skewness2_cor = cor(max_diff_seq, skewness2_list)
```

Evaluation of the Power of *U* test

This section is prepared to

1. Evaluate the statistical power of U test for all data regression models

```
#Read MS intensity table of serially diluted urine samples
input_file = read.csv("model_selection.csv", stringsAsFactors = F)
#Column 1-5: Alignment number, retention time, m/z value, metabolite name, adduct type
#Column 6-15: MS intensities of serially diluted urine samples.
#Column 16: Regression model, 2 for quadratic, 3 for cubic.
#1941 features in total

##############################################################################################################
#Parameter settings

#Mean values of two normally distributed populations (concentration data)
mean_1 = 50
mean_2 = 60

#Number of data in each group (sample size)
data_per_group = 15

#Repeated simulation times in Monte Carlo method section
repeat_times = 1000

#Effect sizes to be tested
eff_seq = seq(0.2,2.0,1)

#Relation concentrations of serial urine samples
all_conc_seq = seq(10,100,10)

##############################################################################################################
#Generate TWO data tables to store the statistical power of U-test before and after data simulation
#Each row represents one metabolic feature, which shows a unique nonlinear ESI response pattern.
#Each column represents one effect size from 0.2 to 2.0
TPR_vs_Efffsize = data.frame(matrix(nrow = 1941, ncol = length(eff_seq)))
colnames(TPR_vs_Efffsize) = eff_seq
TPR_vs_Efffsize_trans = TPR_vs_Efffsize

#Calculate the statistical power of U-test for all metabolic features (1941 in total), in %
for (model_number in 1:1941) {
  #Regression models
  #m_model = poly(as.numeric(input_file[model_number,6:15]) ~ poly(all_conc_seq, input_file$[model_number], raw =
T))
  #m_coeff = #m_model$coefficients

  #Run data simulation for each effect size
  for (i in 1:length(eff_seq)) {
    #Find the standard deviation based on mean value and effect size
    sd_sam2 = (mean_2 - mean_1)/eff_seq[i]

    #Store p values from data simulation
    p_int = c()
    p_int_trans = c()
    #Store statistical power from data simulation
    TPR = rep(0,15)
    TPR_trans = rep(0,15)

    #Repeat the entire data simulation for 15 times to acquire reproducible result
    for (k in 1:15) {
      #Monte Carlo data simulation module
      for (r in 1:repeat_times) {
        #Sample two data sets from two normal distributions. These two data sets are recognized as concentration
data
        data1 = rnorm(sample_size,mean_1,sd_sam2)
        data2 = rnorm(sample_size,mean_2,sd_sam2)

        #Convert concentration data to intensity data using nonlinear regression model
        trans_data1 = predict(polynomial(#m_coeff),data1)
        trans_data2 = predict(polynomial(#m_coeff),data2)

        #Run U-test and get p value
        p_int[r] = wilcox.test(data1,data2,alternative = "two.sided")$p.value
        p_int_trans[r] = wilcox.test(trans_data1,trans_data2,alternative = "two.sided")$p.value
      }
      #After each round of Monte Carlo data simulation, the statistical power is calculated and stored.
      TPR[k] = mean(p_int < 0.01)
      TPR_trans[k] = mean(p_int_trans < 0.01)
    }
    #Statistical power is calculated by taking average of 15 replicates
    TPR_vs_Efffsize[model_number,i] = mean(TPR)*100
    TPR_vs_Efffsize_trans[model_number,i] = mean(TPR_trans)*100
  }

  #Show the progress of data simulation
  print(paste0("Feature number ", model_number, " has been finished. 1941 in total"))
}

write.csv(TPR_vs_Efffsize,"TPR_vs_Efffsize.csv")
write.csv(TPR_vs_Efffsize_trans,"TPR_vs_Efffsize_trans.csv")
```

Evaluation of the Power of Permutation test

This section is prepared to

1. pick one extreme data regression model (model showed in Figure 2B in manuscript)
2. investigate the power of permutation test for concentration data and intensity data in different effect sizes

```
library(coin)

#Parameter settings

mean_1 = 50   # mean of group 1
mean_2 = 60   # mean of group 2


eff_seq = seq(0.2,2.0,1)
data_per_group = 15

model_number = 692   #Take an extreme model (model showed in Figure 2B)
all_conc_seq = seq(10,100,10)
#m_model = lm(as.numeric(input_file[model_number,6:15]) ~ poly(all_conc_seq, input_file$[model_number], raw =
T))
#m_coeff = #m_model$coefficients

TPR = TPR_trans = matrix(nrow = 15, ncol = 10)

for (t in 1:15) {
  for (i in 1:length(eff_seq)) {
    #Find the SD
    sd_sam2 <- (mean_2 - mean_1)/eff_seq[eff]

    #initialize the counter
    p_int = c()
    trans_true_num = 0
    p_int = p_int_trans = rep(0,repeat_times)

    for (r in 1:repeat_times) {
      data1 = rnorm(data_per_group,mean_1,sd_sam2)
      data2 = rnorm(data_per_group,mean_2,sd_sam2)

      trans_data1 = predict(polynomial(#m_coeff),data1)
      trans_data2 = predict(polynomial(#m_coeff),data2)

      DV1 <- c(data1, data2)
      IV1 <- factor(rep(c("A", "B"), c(length(data1), length(data2))))

      DV2 <- c(trans_data1, trans_data2)
      IV2 <- factor(rep(c("A", "B"), c(length(trans_data1), length(trans_data2))))
      pvalue(oneway_test(DV ~ IV))


      p_int[r] = pvalue(oneway_test(DV1 ~ IV1))
      p_int_trans[r] = pvalue(oneway_test(DV2 ~ IV2))

    }
    TPR[i,eff] = mean(p_int < 0.01)
    TPR_trans[i,eff] = mean(p_int_trans < 0.01)

    print(i)
    print(eff)
  }
}
write.csv(TPR, "TPR.csv")                    #TPR of concentration data
write.csv(TPR_trans, "TPR_trans.csv") #TPR of intensity data
```