QUAIL_simulations

Jiacheng Miao

August 22 2022

Introduction

This file provides an example of QUAIL simulations.

In the same folder, we provide two R files containing functions that implements the QUAIL and other methods used in the simulations.

- `QUAIL_FUN_Sim.R` : conduct the simulations where there is no covariates
  - Specifically, you can use `QUAIL(x, y)` to perform QUAIL, where x is a N-dimensional vector of **a single SNP** and y is a N-dimensional vector of a phenotype.
- `QUAIL_Covar_FUN_Sim.R` : conduct the simulations where there is one covariates
  - Specifically, you can use `QUAIL(x, y, covar)` to perform QUAIL, where x is a N-dimensional vector of **a single SNP** , y is a N-dimensional vector of a phenotype, covar is a N-dimensional vector of a covaraites.

An easy way to implement the QUAIL simulations is to download these two files and source it in R. The R packages required for the simulations are `car` , `quantreg` , `lmtest` , and `quantreg` .

Simulations when there is no covariates

We first have a function simulate the SNP and phenotype; You can modify any parameters in `Sim` function as you want. The detailed simulation setting can be found in QUAIL manuscripts. To have a sense about how the data looks like, we run the `Sim` function and check the head of the genotype and phenotype

```
source("QUAIL_Sim_Fun.R")
sim <- Sim(seed = 1, g_type = "mean")
geno <- sim$geno
y <- sim$y
print(head(geno))
```

```
## [1] 2 0 2 1 1 1
```

```
print(head(y))
```

```
## [1] -1.2250644 -1.6172351  0.8934900 -0.5710782 -0.6483083 -0.5570085
```

Next we compare the type-I error for different methods by replicating the simulations 500 times. We first loaded the function in `QUAIL_FUN_Sim.R` . The input data for the vQTL method is a vector.

```
source("QUAIL_Sim_Fun.R")
P_QUAIL <- c() # QUAIL P-value vector
P_DRM <- c() # DRM  P-value vector
P_Lev <- c() # Levene's test  P-value vector
P_HLMM <- c() # HLMM using raw outcome  P-value vector
```

```r
P_HLMM_INT <- c() # HLMM using inverse normal transformed outcome  P-value vector
for (rep in 1:500){
  # Simulate the genotype and phenotype
  sim <- Sim(rep, g_type = "mean") # "g_type <- mean" to access the type-I error for different vQTL met
  geno <- sim$geno
  y <- sim$y
  # Apply the vQTL methods
  P_QUAIL <- c(QUAIL(geno, y), P_QUAIL) # QUAIL
  P_DRM <- c(DRM(geno, y), P_DRM) # DRM
  P_Lev <- c(Lev(geno, y), P_Lev) # Levene's test
  P_HLMM <- c(HLMM(geno, y), P_HLMM) # HLMM using raw outcome
  P_HLMM_INT <- c(HLMM_INT(geno, y), P_HLMM_INT) # HLMM using inverse normal transformed outcome
}
# Assess the type-I error
print(sum(P_QUAIL < 0.05)/length(P_QUAIL))
```

```
## [1] 0.056
```

```r
print(sum(P_DRM < 0.05)/length(P_DRM))
```

```
## [1] 0.034
```

```r
print(sum(P_Lev < 0.05)/length(P_Lev))
```

```
## [1] 0.06
```

```r
print(sum(P_HLMM < 0.05)/length(P_HLMM))
```

```
## [1] 0.21
```

```r
print(sum(P_HLMM_INT < 0.05)/length(P_HLMM_INT))
```

```
## [1] 0.05
```

Simulations when there are covariates

We first have a function simulate the SNP, phenotype, and covariates; You can modify any parameters in `Sim_Covar` function as you want. The detailed simulation setting can be found in QUAIL manuscripts. To have a sense about how the data looks like, we run the `Sim_Covar` function and check the head of the genotype, phenotype and covaraites

```r
source("QUAIL_Sim_Covar_Fun.R")
sim <- Sim_Covar(seed = 1)
geno <- sim$geno
y <- sim$y
covar <- sim$covar
print(head(geno))
```

```
## [1] 2 0 2 1 1 1
```

```r
print(head(y))
```

```
## [1] -1.44296722 -0.33595124 -1.29767062  0.52726428 -0.15481623 -0.02426636
```

```r
print(head(covar))
```

```
## [1] 1 0 1 1 0 0
```

Next we compare the type-I error for different methods in the presence of covariates by replicating the simulations 500 times. We first loaded the function in `QUAIL_FUN_Covar_Sim.R` .The input data for the vQTL method is a vector.

```
source("QUAIL_Sim_Covar_Fun.R")
P_QUAIL <- c() # QUAIL P-value vector
P_DRM <- c() # DRM  P-value vector
P_Lev <- c() # Levene's test  P-value vector
P_HLMM <- c() # HLMM using raw outcome  P-value vector
P_HLMM_INT <- c() # HLMM using inverse normal transformed outcome  P-value vector
for (rep in 1:500){
  # Simulate the genotype and phenotype
  sim <- Sim_Covar(rep)
  geno <- sim$geno
  y <- sim$y
  covar <- sim$covar
  # Apply the vQTL methods
  P_QUAIL <- c(QUAIL_Covar(geno, y, covar), P_QUAIL) # QUAIL
  P_DRM <- c(DRM_Covar(geno, y, covar), P_DRM) # DRM
  P_Lev <- c(Lev_Covar(geno, y, covar), P_Lev) # Levene's test
  P_HLMM <- c(HLMM_Covar(geno, y, covar), P_HLMM) # HLMM using raw outcome
  P_HLMM_INT <- c(HLMM_INT_Covar(geno, y, covar), P_HLMM_INT) # HLMM using inverse normal transformed o
}
# Assess the type-I error
print(sum(P_QUAIL < 0.05)/length(P_QUAIL))
```

```
## [1] 0.046
```

```
print(sum(P_DRM < 0.05)/length(P_DRM))
```

```
## [1] 0.15
```

```
print(sum(P_Lev < 0.05)/length(P_Lev))
```

```
## [1] 0.164
```

```
print(sum(P_HLMM < 0.05)/length(P_HLMM))
```

```
## [1] 0.044
```

```
print(sum(P_HLMM_INT < 0.05)/length(P_HLMM_INT))
```

```
## [1] 0.046
```