Simulation

2019-04-19

Highligths

- Try 3 dimensions of the baseline covariates simulation##
- There are several different way to set the true α, with penalty or without penalty. If it is with penalty, df = 1, the results similar
- If df = 3, it is hard to find the max purity and the max purity maybe not at the true α value ($purity = f(\alpha)$)

Scenario 1. When α value is randomly chosen

Step 1: Specify true α, where α is a vector with length 3. Here I made the true value α:

```
alpha = c(1,1,1)
```

Step 2: Specify β, Γ, D for drug group (drg) and placebo group (pbo)

The outcome follows the formula:

$$\text{the outcome is } Y : \mathbf{Y} = \mathbf{X}(\beta + \mathbf{b} + \mathbf{\Gamma}(\alpha'\mathbf{x})) + \epsilon.$$

where

- The baselines come from the same distributions $\mathbf{x} = [x_1, x_2, x_3]$, the baseline covariate x_1, x_2, x_3 iid $\sim N(0,1)$
- A combination of baseline covariate w: $w = \alpha^T[x_1, x_2, x_3]$, which is $\alpha_1 * x_1 + \alpha_2 * x2 + \alpha_3 * x3$
- X: the independent covariates, $X = [1, t, t^2]$, where $t = 0, 1, 2, ..., 6$

The β:

- $\beta_{drg} = \begin{bmatrix} 0 \\ -1 \\ -3 \end{bmatrix}$, $\beta_{pbo} = \begin{bmatrix} 0 \\ 3 \\ 1 \end{bmatrix}$

The Γ:

- $\Gamma_{drg} = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}$, $\Gamma_{pbo} = \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}$

The ϵ:

- The ϵ_{drg}, $\epsilon_{pbo} \sim N(0,1)$

The bi:

- The random effect covariance matrix D_{drg}:

```r
set.seed(21)
eign1 = diag(c(1,0.4,0.5))
eign2 = matrix(runif(9,0,1),3,3)
a = eign2 %*% eign1 %*% solve(eign2)
bi_sigma = t(a) %*% (a) # make sure it is positive defined
round(bi_sigma,3)
```

```
##        [,1]   [,2]   [,3]
## [1,]  1.450 -0.110  0.203
## [2,] -0.110  0.165 -0.077
## [3,]  0.203 -0.077  0.229
```

- The random effect covariance matrix D_{pbo}:

```r
set.seed(7)
eign1 = diag(c(1,0.5,2))
eign2 = matrix(runif(9,0,2),3,3)
bi_sigma2 = eign2 %*%  t(eign2) /10 # make sure it is positive defined
round(bi_sigma2,3)
```

```
##       [,1]  [,2]  [,3]
## [1,] 0.439 0.296 0.090
## [2,] 0.296 0.465 0.160
## [3,] 0.090 0.160 0.267
```

Step 3: Simulate n observations (e.g. n = 100/treatment)

For drug group, the baseline covariates are generated from N(0,1):

```r
p = 3
baseline = as.matrix(rnorm(p,0,1),p,1)
x1 = baseline[1]; x2 = baseline[2]; x3 = baseline[3]
```

Then w is calculated as $\alpha' x$

```r
alpha = c(1, 1, 1)
w = t(alpha) %*% baseline
```

The random effect is generated from MVN, whose mean = 0 and sigma equals to the covariance matrix D

```r
bi = mvrnorm(1, c(0,0,0), bi_sigma)
```

Then the outcome is calculated

```r
yi = X%*%(beta_drg+bi+gamma_drg*w[1]) + sigma_drg*rnorm(ni,0,1)
```

The placebo group follows the same generation process.

The whole code for true data generation:

```r
true_generation = function(alpha, p, n, ni, tt, X,
                           beta_drg, gamma_drg, bi_sigma,sigma_drg,
                           beta_pbo, gamma_pbo, bi_sigma2,sigma_pbo){
  # alpha
  set.seed(123)
  alpha = as.matrix(alpha,p,1)
  dat_drg = c()
  for(i in 1:n){
    drg_temp = NULL
    drg_temp$subj = rep(paste('drg',i,sep=''),ni)
    drg_temp$trt = rep('drg',ni)
    baseline = as.matrix(rnorm(p,0,1),p,1)
    x1 = baseline[1]; x2 = baseline[2]; x3 =  baseline[3]
    w = rep(t(alpha) %*% baseline,ni)
    drg_temp$x1 = rep(x1,ni); drg_temp$x2 = rep(x2,ni); drg_temp$x3 = rep(x3,ni)
    drg_temp$w = w
    drg_temp$tt = tt
    bi = mvrnorm(1, c(0,0,0), bi_sigma)
    yi = X%*%(beta_drg+bi+gamma_drg*w[1]) + sigma_drg*rnorm(ni,0,1)
    drg_temp$y = yi
    dat_drg = rbind(dat_drg, as.data.frame(drg_temp))
  }

  dat_pbo = c()
  for(i in 1:n){
    pbo_temp = NULL
    pbo_temp$subj = rep(paste('pbo',i,sep=''),ni)
    pbo_temp$trt = rep('pbo',ni)
    baseline = as.matrix(rnorm(p),p,1)
    x1 = baseline[1]; x2 = baseline[2]; x3 =  baseline[3]
    w = rep(t(alpha) %*% baseline,ni)
    pbo_temp$x1 = rep(x1,ni); pbo_temp$x2 = rep(x2,ni); pbo_temp$x3 = rep(x3,ni)
    pbo_temp$w = w
    pbo_temp$tt = tt
    bi = mvrnorm(1, c(0,0,0), (bi_sigma2))
    yi = X%*%(beta_pbo+bi+gamma_pbo*w[1]) + sigma_pbo*rnorm(ni,0,1)
    pbo_temp$y = yi
    dat_pbo = rbind(dat_pbo, as.data.frame(pbo_temp))
  }
  print('True data generated')
  return(list(dat_drg = dat_drg, dat_pbo = dat_pbo))
}
```

Step 4: Monte carlo simulation

Since we would like to calculate the integral of

$$\int \frac{[f_1(z_i|w_i) - f_2(z_i|w_i)]^2}{f_1(z_i|w_i) + f_2(z_i|w_i)} dz_i$$

where we assume that $f_1(z_i|w_i)$ and $f_2(z_i|w_i)$ are two MVN for drug group and placebo group separately.

To calculate this, we could firstly generate a large dataset (10000) from a 2 by 2 standard multivariate normal distribution.

```
Xstart = mvrnorm(10000, c(0,0), diag(c(1,1)))
```

Then transform those points to MVN of drug and MVN of placebo separately.

- standard MVN: $X_0 \sim MVN(\mu_0, \sigma_0)$
- MVN_drg: $X_1 \sim MVN(\mu_{drg}, \sigma_{drg})$, $\sigma_{drg} = H\Lambda H^T$, which is the eigenvalue decomposition of σ_{drg}. H is the matrix of eigen vectors and Λ is the matrix whose diagnoal values are σ_{drg}'s eignevalues.
- Transformation: $X_1 = H\Lambda^{\frac{1}{2}}X + \mu_{drg}$

And then we can get the large data points sampled from the two MVN distributions. The purity can be calculated.

```
### monta
monta_carlo_pdf = function(Xstart, mu1, D1, mu2, D2){

  # transformation
  d1 = eigen(D1)
  d1 = d1$vectors %*% diag(sqrt(d1$values))

  d2 = eigen(D2)
  d2 = d2$vectors %*% diag(sqrt(d2$values))

  points1 = Xstart %*% t(d1)

  points1[,1] = points1[,1] + mu1[1]; points1[,2] = points1[,2] + mu1[2]

  points2 = Xstart %*% t(d2)
  points2[,1] = points2[,1] + mu2[1]; points2[,2] = points2[,2] + mu1[2]

  f1 = dmvnorm(points1,  mu1, D1)
  f2 = dmvnorm(points2,  mu2, D2)

  f3 = (f1 + f2) # deal with the 0s, f1 + f2 cannot be 0
  f3 = ifelse(f3 == 0, 1e-4,f3)

  purity = c((f1 - f2)^2 / f3)
  return(purity)
}
```

Step 5: Fit model using αs and compute purity

The functions

```
purity_calculation = function(dat, beta1, beta2, gamma1, gamma2, D1, D2, p=2){
  unique_dat = unique(dat[,c('subj','w')])
  purity = c()
  Mu1 = c(); Mu2 = c()
  for(i in 1:dim(unique_dat)[1]){
    #if(i %% 10 ==0) print(i)
    mu1 = beta1 + gamma1 * unique_dat$w[i] # mu1 = beta_drg + gamma_drg * unique_dat$w[i]
    mu2 = beta2 + gamma2 * unique_dat$w[i] # mu2 = beta_pbo + gamma_pbo * unique_dat$w[i]
    Mu1 = rbind(Mu1, mu1); Mu2 = rbind(Mu2, mu2)
    res = mean(monta_carlo_pdf(Xstart, mu1, D1, mu2, D2))
    purity = c(purity, res)
  }
  return(list(purity = purity, Mu1 = Mu1, Mu2 = Mu2))
}
purity_function = function(A, varname = '', times = '',
                           trt = '',
                           trtlevel = '',
                           subj = '',
                           outcome = '',
                           start = 0, data = dat){
  p = length(A)
```

```r
  alpha_est = matrix(A,p,1)
  dat_est = data
  if(sum(varname == '') == length(varname)){
    w_est = dat[(start):(start + p -1),] %*% alpha_est
  }
  if(sum(varname != '') == length(varname)){
    w_est = as.matrix(dat_est[,varname]) %*% alpha_est
  }
  dat_est$w = w_est
  if(times != ''){
    dat_est$tt = dat_est[,times]
  }
  if(subj != ''){
    dat_est$subj = dat_est[,subj]
  }
  if(trt != ''){
    dat_est$trt = dat_est[,trt]
    if(sum(trtlevel == '')==0){
      dat_est[dat_est$trt == trtlevel[1],]$trt = 'pbo'
      dat_est[dat_est$trt == trtlevel[2],]$trt = 'drg'
    }
  }
  dat_est$outcome = dat_est[,outcome]
  dat_pbo_est = dat_est[dat_est$trt == 'pbo', ]
  dat_drg_est = dat_est[dat_est$trt == 'drg', ]

  fit_drg_est = lmer(outcome ~ tt + I(tt^2) + w + w * tt +
                       w * I(tt^2) + (tt+I(tt^2)|subj),
                     data = dat_drg_est, REML = FALSE)
  fit_drg_est
  fit_pbo_est = lmer(outcome ~ tt + I(tt^2) + w + w * tt +
                       w * I(tt^2) + (tt+I(tt^2)|subj),
                     data = dat_pbo_est, REML = FALSE)
  fit_pbo_est

  beta1 = as.matrix(fixef(fit_drg_est))[2:3]
  gamma1 = as.matrix(fixef(fit_drg_est))[5:6] # true estimate = -1.9812346 -0.9895642
  D1 = as.matrix(VarCorr(fit_drg_est)$subj)[2:3, 2:3]

  beta2 = as.matrix(fixef(fit_pbo_est))[2:3]
  gamma2 = as.matrix(fixef(fit_pbo_est))[5:6] # 1.994051 1.003545
  D2 = as.matrix(VarCorr(fit_pbo_est)$subj)[2:3, 2:3]

  b = purity_calculation(dat_est, beta1, beta2, gamma1, gamma2, D1, D2, p=2)
  return(list(purity = b$purity, Mu1 = b$Mu1, Mu2 = b$Mu2, beta1 = beta1, beta2 = beta2,
              gamma1 = gamma1, gamma2 = gamma2, D1 = D1, D2 = D2, data = dat_est))
}
```

The results

The generated data set:

```r
head(dat)
```

```
##   subj trt         x1         x2       x3         w tt           y
## 1 drg1 drg -0.5604756 -0.2301775 1.558708 0.7680552  0   0.5218742
## 2 drg1 drg -0.5604756 -0.2301775 1.558708 0.7680552  1  -5.1485016
## 3 drg1 drg -0.5604756 -0.2301775 1.558708 0.7680552  2 -12.1817941
## 4 drg1 drg -0.5604756 -0.2301775 1.558708 0.7680552  3 -23.2192066
## 5 drg1 drg -0.5604756 -0.2301775 1.558708 0.7680552  4 -36.4951688
## 6 drg1 drg -0.5604756 -0.2301775 1.558708 0.7680552  5 -55.9722452
```

We can compare the purity calculated with different α values

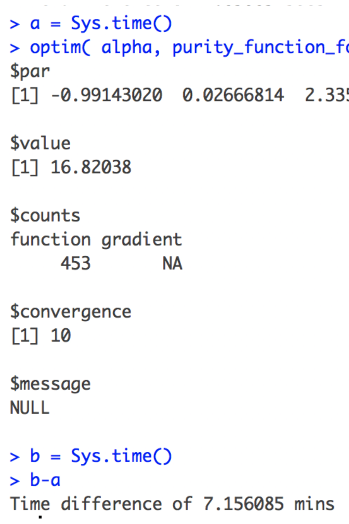
```r
A = c(1,1,1) # true value
res = purity_function(A, varname = c('x1','x2','x3'), times = '',
                      trt = '',
                      trtlevel = '',
                      subj = '',
                      outcome = 'y',
                      start = 0, data = dat)
sum(res$purity)
```

```
## [1] 295.762
```

```r
A = c(1,0,1)
res2 = purity_function(A, varname = c('x1','x2','x3'), times = '',
                      trt = '',
                      trtlevel = '',
                      subj = '',
                      outcome = 'y',
                      start = 0, data = dat)
sum(res2$purity)
```

```
## [1] 8.100446
```

Try to use *optim function* to find the max value



```
> a = Sys.time()
> optim( alpha, purity_function_for_optim)
$par
[1] -0.99143020  0.02666814  2.33597150

$value
[1] 16.82038

$counts
function gradient 
     453       NA 

$convergence
[1] 10

$message
NULL

> b = Sys.time()
> b-a
Time difference of 7.156085 mins
```

Try *genetic algoritm*

It returns a value:

```r
A = c(3.125,3.152,3.289)
res = purity_function(A, varname = c('x1','x2','x3'), times = '',
                      trt = '',
                      trtlevel = '',
                      subj = '',
                      outcome = 'y',
                      start = 0, data = dat)
sum(res$purity)
```

```
## [1] 393.7318
```

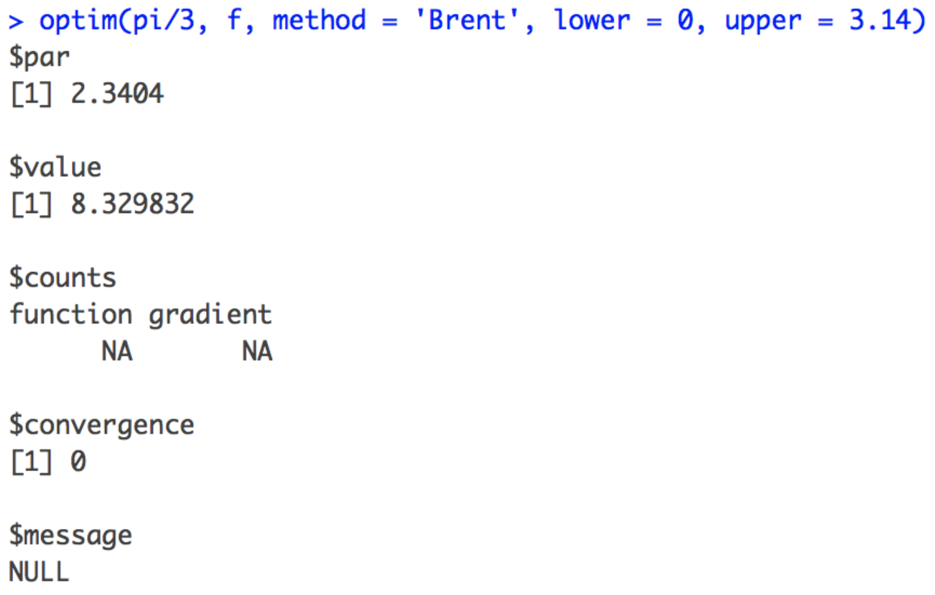
However, the value is larger than the true value.

Scenario 2. Set df = 1.

All the same with scenario 1 except the choice of *alpha*

Let $\alpha = [sin(\theta)sin(\theta), sin(\theta)cos(\theta), cos(\theta)]$, the indpendent variable will be the θ. Let's set true $\theta = \frac{\pi}{3}$.

However, this time the *optim function* doesn't work well neither.

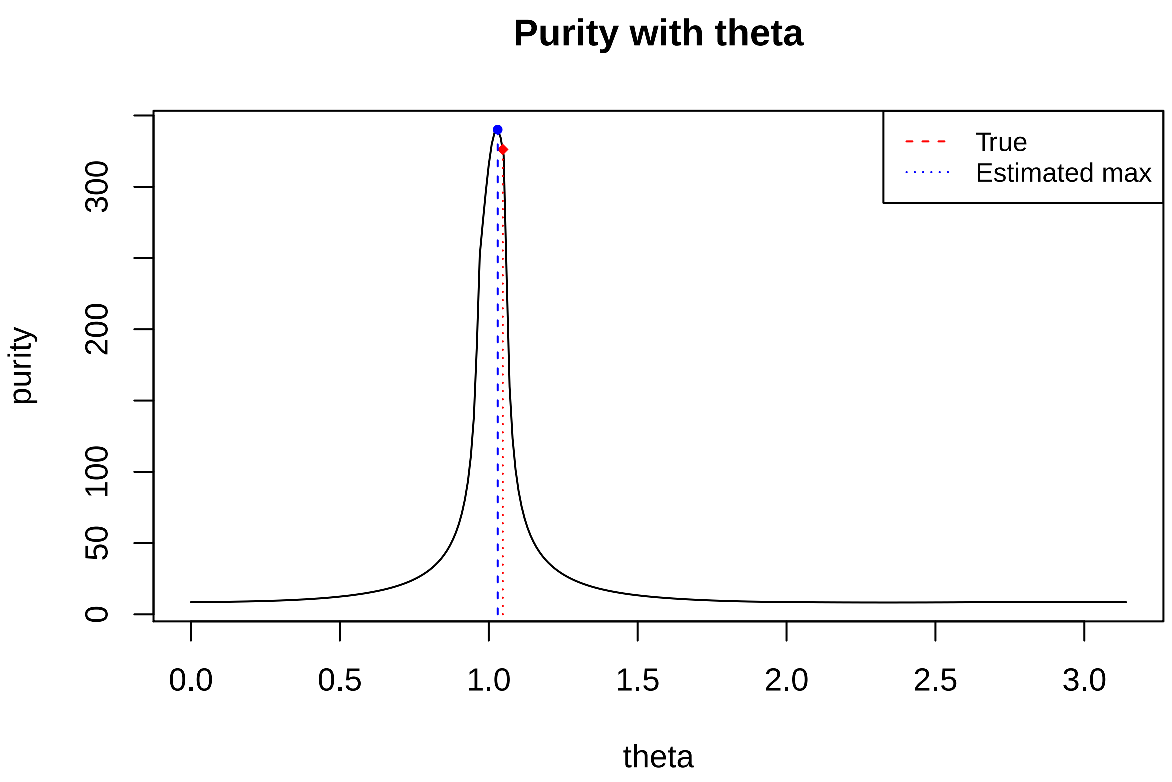


```
> optim(pi/3, f, method = 'Brent', lower = 0, upper = 3.14)
$par
[1] 2.3404

$value
[1] 8.329832

$counts
function gradient
      NA       NA

$convergence
[1] 0

$message
NULL
```

The genetic algorthm return an

- $\alpha \approx 1.01953,\ f(\alpha) \approx 338.2038$
- $\pi/3 \approx 1.0472,\ f(\pi/3) \approx 326.1763$

The purity vs α plot:


Purity with theta
True
Estimated max
purity
theta
0
50
100
200
300
0.0
0.5
1.0
1.5
2.0
2.5
3.0