R Notebook

Allocation methods

Let us start by defining functions for each allocation method

Allocation methods that don't take into account covariates

Random and Restricted randomisation

```r
## Allocation functions that don't take into account covariates

randomAllocation <- function(n, T){
  results = sample(1:T, n, replace = TRUE)
  return(results)
}

biasedCoinAllocation <- function(n, T){
  Allocation = rep(0,n) # biased coin design
  Si = rep(0,n)
  q = rep(0.5,n)
  for (i in seq(from = 1, to = n, by = 1)) {
    Allocation[i] = 2*(runif(1)<= q[i]) - 1
    if(i < n){
      Si[i+1] = sum(Allocation[1:i])/(i)
      if(Si[i+1] == 0){q[i+1] = 0.5}
      if(Si[i+1] < 0){q[i+1] = 2/3}
      if(Si[i+1] > 0){q[i+1] = 1/3}
    }
  }
  Allocation = (Allocation + 3)/2
  CombinedResult = data.frame(Allocation = Allocation, Si = Si, q = q)
  return(CombinedResult)
}
```

Coin based designs

```r
doublyBiasedCoinAllocation <- function(n, n0, T, Results){
  Allocation = rep(0,n)
  q = rep(0.5,n)
  #First initialise by assigning no to each treatment
  for (t in 1:T) {
    Allocation[n0*(t-1)+1:t*n0] = t
  }

  for (t in t*no+1:n) {
    q[i] =
    Allocation[i] = 2 - (runif(1)<= q[i])
  return(Allocation)
  }
}
```

Turn based and Gittins index

```r
playTheWinner <- function(n, T, Results){
  Allocation = rep(0,n)
  Allocation[1] = sample(T,1)
  for (i in 2:n) {
    Allocation[i] = ifelse(Results[i,Allocation[i-1]]==1, Allocation[i-1], sample(T,1))
  }
  return(Allocation)
}

playTheWinnerRandomised <- function(n, T, Results){
  Allocation = rep(0,n)
}
```

Sequential estimation techniques

```r
neymanKnown <- function(n, T, Results, p){
  Allocation = sample(T, n, replace = TRUE, prob = p)
  return(Allocation)
}

neymanEstimated_binary <- function(n, n0, Results){
  Allocation = rep(0,n)
  T = 2
  #First initialise by assigning no to each treatment

   for (t in 1:T) {
    Allocation[(n0*(t-1)+1):(t*n0)] = t
  }

  for (i in (T*n0+1):n) {
    i
    pA = mean(Results[Allocation[1:(i-1)] == 1 ,1])
    qA = 1 - pA
    pB = mean(Results[Allocation[1:(i-1)] == 2 ,2])
    qB = 1 - pB
    R = sqrt((pA*qA)/(pB*qB))
    prob = R/(1+R)
    Allocation[i] = 2 - (runif(1) <= prob)

  }
  return(Allocation)
}

RSIHR_binary <- function(n, n0, Results){
  Allocation = rep(0,n)
  T = 2
  #First initialise by assigning no to each treatment

   for (t in 1:T) {
    Allocation[(n0*(t-1)+1):(t*n0)] = t
  }

  for (i in (T*n0+1):n) {
    i
    pA = mean(Results[Allocation[1:(i-1)] == 1 , 1])
    pB = mean(Results[Allocation[1:(i-1)] == 2 , 2])
    R = sqrt((pA)/(pB))
    prob = R/(1+R)
    Allocation[i] = 2 - (runif(1) <= prob)

  }
  return(Allocation)
}

ResponseAdaptive_Binary_1 <- function(n, n0, Results){
  T = 2
  Allocation = rep(0,n)
  for (t in 1:T) {
    Allocation[(n0*(t-1)+1):(t*n0)] = t
  }

  for (i in (T*n0+1):n) {
    i
    pA = mean(Results[Allocation[1:(i-1)] == 1 ,1])
    qA = 1 - pA
    pB = mean(Results[Allocation[1:(i-1)] == 2 ,2])
    qB = 1 - pB
    prob = ((pA/qA))/((pA/qA)+(pB/qB))
    Allocation[i] = 2 - (runif(1) <= prob)

  }
  return(Allocation)
}

ResponseAdaptive_Binary_2 <- function(n, n0, Results){
  Allocation = rep(0,n)
  T = 2
  #First initialise by assigning no to each treatment

   for (t in 1:T) {
    Allocation[(n0*(t-1)+1):(t*n0)] = t
  }

  for (i in (T*n0+1):n) {
    i
    pA = mean(Results[Allocation[1:(i-1)] == 1 ,1])
    qA = 1 - pA
    pB = mean(Results[Allocation[1:(i-1)] == 2 ,2])
    qB = 1 - pB
    R = sqrt((pA*qA^2)/(pB*qB^2))
    prob = R/(1+R)
    Allocation[i] = 2 - (runif(1) <= prob)

  }
  return(Allocation)
}
```

Allocation strategies that take into account covariates

Sequential estimation

```
## Allocation functions that take into account covariates
CARA1 <- function(n, n0, T, Z, Results){

}

CARA2 <- function(n, n0, T, Z, Results){

}

CARA3 <- function(n, n0, T, Z, Results){

}
```

Simulation

setting up the problem

```
n = 1000 # Number of subjects
n0 = 5

Z = rbinom(n,1,0.5) # Simulate covariates

SuccessProbs = c(0.2, 0.6)

SuccessProbs_Z = matrix(data = c(0.1, 0.3, 0.7, 0.1), nrow = 2, ncol = 2)
```

Now we can model different treatment allocation strategies

```
# Results
T = length(SuccessProbs)
Results = matrix(0, nrow = n, ncol = T)

for (i in 1:T) {
  Results[ , i] = rbinom(n, 1, SuccessProbs[i])
}


# Pure random
Allocation_random = randomAllocation(n, T)

# Biased Coin
Allocation_biasedCoin = biasedCoinAllocation(n, T)
```

```r
# Play the winner
Allocation_playTheWinner = playTheWinner(n, T, Results)

# Neyman Allocations
Allocation_neyamn_known = neymanKnown(n, T, Results, SuccessProbs)
Allocation_neyamn_estimated = neymanEstimated_binary(n, n0, Results)
Allocation_RSHIR_binary = RSIHR_binary(n, n0, Results)

Allocation_ResponseAdaptive_Binary_1 = ResponseAdaptive_Binary_1(n, n0, Results)
Allocation_ResponseAdaptive_Binary_2 = ResponseAdaptive_Binary_2(n, n0, Results)
```

```r
table(Allocation_random)
```

```
## Allocation_random
##   1   2
## 517 483
```

```r
table(Allocation_biasedCoin[,1])
```

```
##
##   1   2
## 501 499
```

```r
table(Allocation_playTheWinner)
```

```
## Allocation_playTheWinner
##   1   2
## 369 631
```

```r
table(Allocation_neyamn_known)
```

```
## Allocation_neyamn_known
##   1   2
## 245 755
```

```r
table(Allocation_neyamn_estimated)
```

```
## Allocation_neyamn_estimated
##   1   2
## 457 543
```

```r
table(Allocation_RSHIR_binary)
```

```
## Allocation_RSHIR_binary
##   1   2
## 391 609
```

```r
table(Allocation_ResponseAdaptive_Binary_1)
```

```
## Allocation_ResponseAdaptive_Binary_1
##   1   2
## 184 816
```

```r
table(Allocation_ResponseAdaptive_Binary_2)
```

```
## Allocation_ResponseAdaptive_Binary_2
##   1   2
## 514 486
```