Statistics 771 - R Output for Simulations

Yao (Tony) Wang

24/04/2020

```r
library(gsDesign) #See Zhu, Ni & Yao 2011 for performance vs SAS/EAST
```

```
## Loading required package: xtable

## Loading required package: ggplot2
```

```r
#Parallelization to accelerate adaptive enrichment
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators

## Loading required package: parallel
```

```r
#Data Visualization
library(ggplot2)
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
#Initialize parallelization
#Simulations are intensive
#Will take upwards of 30 minutes to run on a dual-core laptop
cores=detectCores()
cl_cores <- makeCluster(cores[1]-1)
registerDoParallel(cl_cores)

######Efron_BCD_Randomizer######

### Efron Biased Coin Algorithm
### With consideration for accumulated results of past tosses
bcd_cumul <- function(prob=2/3, nrequired=20, oldtoss=numeric()) {

  #Vector of new tosses
  newtoss <- c(rep(9, nrequired))

  for(i in 1:length(newtoss)){
    #Take difference between sizes of assigned groups
    #Include past results as well in this difference
    diff <- (sum(oldtoss==1)+(sum(newtoss[1:i]==1))
             -(sum(oldtoss==0)+sum(newtoss[1:i]==0)))
    #Assign based on Efron's rules
    if (diff > 0) {
      newtoss[i] <- sample(x=c(0, 1), size=1, prob=c(prob, 1-prob))
    }
    else if (diff < 0) {
      newtoss[i] <- sample(x=c(0, 1), size=1, prob=c(1-prob, prob))
    }
    else {
      newtoss[i] <- sample(x=c(0, 1), size=1, prob=c(0.5, 0.5))
    }
  }
  return(newtoss)
}

######Generate_Biased_Coin_Data_Function######

simdata_bcd <- function(n_init=20, test_eff=414, placebo=718, past=numeric()){

  #Distributions and parameters guessed from Burton (2015)
  #Pre-treatment: all the same, Normal(mean=790, sd=50)
  #Post-treatment: placebo: Normal(mean=718, sd=50)
  #Post-treatment: drug: Normal(mean=414, sd=50)

  #Biased Coin Assignments and Outcomes
  #Also consider vector of past tosses
  sim <- data.frame("Subject"=c(rep(1:n_init)),
                    "Phe_Pre"=c(rnorm(n_init, mean=790, sd=50)),
                    "Treat"=c(bcd_cumul(nrequired=n_init, oldtoss=past)),
                    "Phe_Post"=c(rep(1:n_init)))

  #Experimental "Results" conditional on given treatment
  for(i in 1:dim(sim)[1]){
    if(sim[i, 3] == 1){sim[i, 4] <- rnorm(1, mean=test_eff, sd=50)} #Test
    else {sim[i, 4] <- rnorm(1, mean=placebo, sd=50)} #Ref
  }

  return(sim)
}

########################Simulate_Biased_Coin_Study########################

#Can adjust treatment/control effects plus the spending function used
bcd_simulation <- function(n_interim = 6, nsims=1000,
                           treat_eff=414, control=718, method="OF", error_type="I"){

#"Pre-study" calculation of efficacy bounds
#Choices of parameters based on Burton et al for a fixed design
#80% power, alpha=0.05, 30% difference between groups
#Not clear from Burton (2016) what the "30% difference" refers to
#Presume refers to a 30% difference in probability
#(effect size around 0.84 on a normal(0,1) scale)
#120 Patients total (close to observed count in study)

gsd <- gsDesign(k=n_interim, test.type=2, alpha=0.025, beta=0.2,
                delta = 0.84, n.fix=120, sfu=method)

#Critical values (Z-scores) from gsDesign
#Convert Z-scores to tail p-values, then from p to t-score directly
```

```r
gsd_zscore <- gsd$upper$bound
gsd_nominal_p <- pnorm(gsd_zscore, lower.tail=FALSE)

#degrees of freedom: n1+n2-2 (unequal groups with equal variances)
adjust_df <- c(1:n_interim)
for(i in 1:n_interim){
  adjust_df[i]<- (i*(120/n_interim))-2
}

#Vector of adjusted t-values for rejection
adjust_t_val <- qt(gsd_nominal_p, df=adjust_df, lower.tail=FALSE)

#Store simulated results
opchar <- data.frame("rejected"=c(rep(1:nsims)),
                     "n_used"=c(rep(1:nsims)),
                     "treat_diff"=c(rep(1:nsims)),
                     "placebo_test_ratio"=c(rep(1:nsims)))

#Simulate 1000 times
for(i in 1:dim(opchar)[1]){

#Make initial sample
dta <- simdata_bcd(test_eff = treat_eff, placebo=control)
#Run a test with up to 6 interim analyses
for(j in 1:length(adjust_t_val)){

#T-statistic (unpaired, equal variances)
t_unpaired <- t.test(x=dta[dta$Treat==1, 4],
       y=dta[dta$Treat==0, 4],
       alternative="two.sided",
       paired=FALSE, var.equal=TRUE)$statistic

#If t large enough at any point, reject null hypothesis
if(abs(t_unpaired) > adjust_t_val[j]){
  opchar[i, 1] <- 1 #1=Reject,
  opchar[i, 2] <- dim(dta)[1]
  opchar[i, 3] <- mean(dta[dta$Treat==1, 4]) - mean(dta[dta$Treat==0, 4])
  opchar[i, 4] <- sum(dta$Treat==0)/sum(dta$Treat==1) # num_placebo / num_treatment
  break
}
#If not all interim analyses are done,
#Add another 20 data points and try again
else if((abs(t_unpaired) < adjust_t_val[j]) & (j < length(adjust_t_val))){
  dta <- rbind(dta, simdata_bcd(test_eff = treat_eff, placebo=control))
}
#Else fail to reject null hypothesis
else if(j==length(adjust_t_val) & abs(t_unpaired) < adjust_t_val[j]){
  opchar[i, 1] <- 0 #0=Fail to Reject null
  opchar[i, 2] <- dim(dta)[1]
  opchar[i, 3] <- mean(dta[dta$Treat==1, 4]) - mean(dta[dta$Treat==0, 4])
  opchar[i, 4] <- sum(dta$Treat==0)/sum(dta$Treat==1) # num_placebo / num_treatment
  break
}
}

}

#Create data frame showing appropriate interpretations of results
#Differences in how rejections are interpreted based on scenario
{if (error_type=="II"){
  results_dframe <- data.frame("Method"= as.character(gsd$upper$name),
                               "Randomization"="Efron BCD",
                               "Treatment_Mean"=treat_eff, "Placebo_Mean"=control,
                               "Theoretical_Delta"=treat_eff-control,
                               "Observed_Delta"=mean(opchar$treat_diff),
                               "Bias_Delta"=(mean(opchar$treat_diff)-(treat_eff-control)),
                               "Mean_Sample_Size"=mean(opchar$n_used),
                               "Randomization_Ratio"=mean(opchar$placebo_test_ratio),
                               "Empirical_Power"=sum(opchar$rejected==1)/1000)
}
else {
  results_dframe <- data.frame("Method"= as.character(gsd$upper$name),
                               "Randomization"="Efron BCD",
                               "Treatment_Mean"=treat_eff, "Placebo_Mean"=control,
                               "Theoretical_Delta"=treat_eff-control,
                               "Observed_Delta"=mean(opchar$treat_diff),
                               "Bias_Delta"=(mean(opchar$treat_diff)-(treat_eff-control)),
                               "Mean_Sample_Size"=mean(opchar$n_used),
                               "Randomization_Ratio"=mean(opchar$placebo_test_ratio),
                               "Type_I"=sum(opchar$rejected==1)/1000)
}}

#Return results
return(results_dframe)

}

######Zhang_Rosenberger_Algorithm######

zr_cumul <- function(nrequired=20, test_eff=414,
                     placebo=718, past_data=data.frame("Subject"=numeric(),
                                                       "Phe_Pre"=numeric(),
                                                       "Treat"=numeric(),
                                                       "Phe_Post"= numeric())){
  #Load up past data and capture original dimension
  sim <- past_data
  orig_dim <- dim(past_data)[1]

  #Calculate values
  mu_ref <- mean(sim[sim$Treat==0, 4])
  sd_placebo <- sd(sim[sim$Treat==0, 4])
  mu_test <- mean(sim[sim$Treat==1, 4])
  sd_treatment <- sd(sim[sim$Treat==1, 4])

  #Loop to generate data
  for(k in 1:nrequired) {
    #r - control allocation to inferior treatment
    ratio <- (sd_treatment*sqrt(mu_ref))/(sd_placebo*sqrt(mu_test))
    #Placeholder for Randomization Ratio (start with 1/2)
    AR <- 1/2
    #Calculate allocation ratio for incoming observation
    ifelse((mu_ref!=mu_test & ratio!=1),
           AR <- (sd_treatment*sqrt(mu_ref))/((sd_treatment*sqrt(mu_ref))
                                               +(sd_placebo*sqrt(mu_test))),
           AR <- 1/2)

    #Generate imcoming observation and allocate treatment
    sample <- rnorm(1, mean=790, sd=50)
    sim[orig_dim+k, ] <- c("Subject"=orig_dim+k,
                           "Phe_Pre"=sample,
                           "Treat"=sample(c(0, 1), size=1, prob=c(1-AR, AR)),
                           "Phe_Post"= 9)
    #Get "Observed" Result and update values if group size changes
    if(sim[orig_dim+k, 3] == 1){
      sim[orig_dim+k, 4] <- rnorm(1, mean=test_eff, sd=50)
      mu_test <- mean(sim[sim$Treat==1, 4])
      sd_treatment <- sd(sim[sim$Treat==1, 4])
    }
    else {
      sim[orig_dim+k, 4] <- rnorm(1, mean=placebo, sd=50)
      mu_ref <- mean(sim[sim$Treat==0, 4])
      sd_placebo <- sd(sim[sim$Treat==0, 4])
    }


  }
  #Note: this new vector contains all old observations plus new ones
  return(sim)
}

######################Simulate_ZR_Randomization_Study######################

#Function is almost the same as before
#Incorporating both types of randomization into one function but instability occurs
zr_simulation <- function(n_interim = 6, nsims=1000,
                          treat_eff=414, control=718, method="OF", error_type="I"){

  meandiff <- treat_eff - control  #True difference between treatments

  #"Pre-study" calculation of efficacy bounds
  #Same routine and assumptions for errors, effect size, n, and number of tests

  gsd <- gsDesign(k=n_interim, test.type=2, alpha=0.025, beta=0.2,
                  delta = 0.84, n.fix=120, sfu=method)

  #Critical values (Z-scores) from gsDesign
  #Convert Z-scores to tail p-values, then from p to t-score directly
  gsd_zscore <- gsd$upper$bound
  gsd_nominal_p <- pnorm(gsd_zscore, lower.tail=FALSE)
```

```r
#degrees of freedom: n1+n2-2 (unequal groups with equal variances)
adjust_df <- c(1:n_interim)
for(i in 1:n_interim) {
  adjust_df[i]<- (i*(120/n_interim))-2
}

#Vector of adjusted t-values for rejection
adjust_t_val <- qt(gsd_nominal_p, df=adjust_df, lower.tail=FALSE)

#Store simulated results
opchar <- data.frame("rejected"=c(rep(1:nsims)),
                     "n_used"=c(rep(1:nsims)),
                     "treat_diff"=c(rep(1:nsims)),
                     "placebo_test_ratio"=c(rep(1:nsims)))

#Simulate 1000 times
for(i in 1:dim(opchar)[1]) {

  #Make initial sample
  dta <- zr_cumul(test_eff = treat_eff, placebo=control)
  #Run a test with up to 6 interim analyses
  for(j in 1:length(adjust_t_val)) {

    #T-statistic (unpaired, equal variances)
    t_unpaired <- t.test(x=dta[dta$Treat==1, 4],
                         y=dta[dta$Treat==0, 4],
                         alternative="two.sided",
                         paired=FALSE, var.equal=TRUE)$statistic
    #If t large enough at any point, reject null hypothesis
    if(abs(t_unpaired) > adjust_t_val[j]){
      opchar[i, 1] <- 1 #1=Reject,
      opchar[i, 2] <- dim(dta)[1]
      opchar[i, 3] <- mean(dta[dta$Treat==1, 4]) - mean(dta[dta$Treat==0, 4])
      opchar[i, 4] <- sum(dta$Treat==0)/sum(dta$Treat==1) # num_placebo / num_treatment
      break
    }
    #If not all interim analyses are done,
    #Add another 20 data points and try again
    else if((abs(t_unpaired) < adjust_t_val[j]) & (j < length(adjust_t_val))){
      dta <- zr_cumul(test_eff = treat_eff, placebo=control, past_data = dta)
    }
    #Else fail to reject null hypothesis
    else if(j==length(adjust_t_val) & abs(t_unpaired) < adjust_t_val[j]){
      opchar[i, 1] <- 0 #0=Fail to Reject null
      opchar[i, 2] <- dim(dta)[1]
      opchar[i, 3] <- mean(dta[dta$Treat==1, 4]) - mean(dta[dta$Treat==0, 4])
      opchar[i, 4] <- sum(dta$Treat==0)/sum(dta$Treat==1) # num_placebo / num_treatment
      break
    }
  }

}
```

```
  #Create data frame showing appropriate interpretations of results
  #Differences in how rejections are interpreted based on scenario
  {if (error_type=="II"){
  results_dframe <- data.frame("Method"= as.character(gsd$upper$name),
                              "Randomization"="Zhang-Rosenberger",
                              "Treatment_Mean"=treat_eff, "Placebo_Mean"=control,
                              "Theoretical_Delta"=treat_eff-control,
                              "Observed_Delta"=mean(opchar$treat_diff),
                              "Bias_Delta"=(mean(opchar$treat_diff)-(treat_eff-control)),
                              "Mean_Sample_Size"=mean(opchar$n_used),
                              "Randomization_Ratio"=mean(opchar$placebo_test_ratio),
                              "Empirical_Power"=sum(opchar$rejected==1)/1000)
  }
    else {
  results_dframe <- data.frame("Method"= as.character(gsd$upper$name),
                              "Randomization"="Zhang-Rosenberger",
                              "Treatment_Mean"=treat_eff, "Placebo_Mean"=control,
                              "Theoretical_Delta"=treat_eff-control,
                              "Observed_Delta"=mean(opchar$treat_diff),
                              "Bias_Delta"=(mean(opchar$treat_diff)-(treat_eff-control)),
                              "Mean_Sample_Size"=mean(opchar$n_used),
                              "Randomization_Ratio"=mean(opchar$placebo_test_ratio),
                              "Type_I"=sum(opchar$rejected==1)/1000)
    }}




  #Return results
  return(results_dframe)

}


#####################################Simulations####################################

#######Efron_BCD##########

#Set Seed for Cluster
clusterSetRNGStream(cl_cores, 12527321)

#TypeI Errors for equal means

Errors_BCD <- foreach (i=c("Pocock", "OF", "WT", sfHSD), .combine=rbind,
                      .packages = 'gsDesign') %dopar% {
                        bcd_simulation(treat_eff=718,
                                       control=718, method=i, error_type="I")}


#Power Curve and Randomization Ratio for Biased Coin Randomization and Given Rules

#Compute results for each set of stopping bounds
#Parallelization helps substantially if using a multi-core CPU
```

```r
WT_Power_BCD <- foreach (i=c(seq(415, 715, by=5)), .combine=rbind,
                        .packages = 'gsDesign') %dopar% {
     bcd_simulation(treat_eff=i, control=718, method="WT", error_type="II")}

Pocock_power_BCD <- foreach(i=c(seq(415, 715, by=5)), .combine=rbind,
                           .packages = 'gsDesign') %dopar% {
     bcd_simulation(treat_eff=i, control=718, method="Pocock", error_type="II")}


OF_power_BCD <- foreach (i=c(seq(415, 715, by=5)), .combine=rbind,
                        .packages = 'gsDesign') %dopar% {
     bcd_simulation(treat_eff=i, control=718, method="OF", error_type="II")}

sfHSD_Power_BCD <- foreach (i=c(seq(415, 715, by=5)), .combine=rbind,
                           .packages = 'gsDesign') %dopar% {
     bcd_simulation(treat_eff=i, control=718, method=sfHSD, error_type="II")}

#######Zhang-Rosenberger#########

#Set New Seed for Cluster
clusterSetRNGStream(cl_cores, 65848968)

#TypeI Errors for equal means

Errors_ZR <- foreach (i=c("Pocock", "OF", "WT", sfHSD), .combine=rbind,
                      .packages = 'gsDesign') %dopar% {
    zr_simulation(treat_eff=718, control=718, method=i, error_type="I")}


#Compute results for each set of stopping bounds
#Takes a while to run because of adaptive randomization process
#Parallelization helps substantially if using a multi-core CPU

WT_Power_ZR <- foreach (i=c(seq(415, 715, by=5)), .combine=rbind,
                        .packages = 'gsDesign') %dopar% {
   zr_simulation(treat_eff=i, control=718, method="WT", error_type="II")}

Pocock_power_ZR <- foreach(i=c(seq(415, 715, by=5)), .combine=rbind,
                           .packages = 'gsDesign') %dopar% {
   zr_simulation(treat_eff=i, control=718, method="Pocock", error_type="II")}


OF_power_ZR <- foreach (i=c(seq(415, 715, by=5)), .combine=rbind,
                        .packages = 'gsDesign') %dopar% {
    zr_simulation(treat_eff=i, control=718, method="OF", error_type="II")}

sfHSD_Power_ZR <- foreach (i=c(seq(415, 715, by=5)), .combine=rbind,
                           .packages = 'gsDesign') %dopar% {
    zr_simulation(treat_eff=i, control=718, method=sfHSD, error_type="II")}

#No intensive calculations needed afterwards
#Stop cluster
stopCluster(cl_cores)
```

```
###################################Final_Results#####################################

#Show tables for Type I error
Errors_BCD[, c(1, 2, 5, 6, 10)]
```

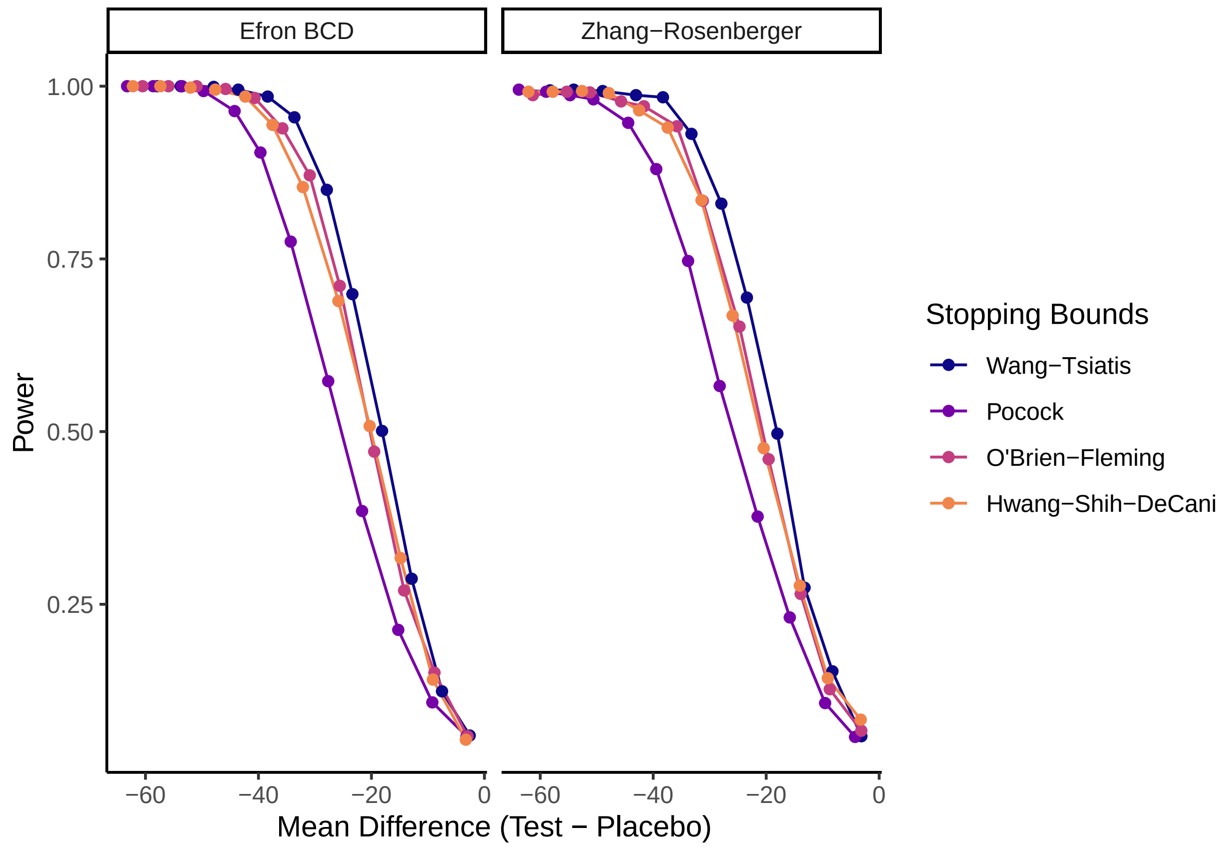
```
##                Method Randomization Theoretical_Delta Observed_Delta Type_I
## 1              Pocock     Efron BCD                 0     0.02692155  0.043
## 2                  OF     Efron BCD                 0    -0.01095718  0.046
## 3                  WT     Efron BCD                 0    -0.02380443  0.055
## 4 Hwang-Shih-DeCani     Efron BCD                 0    -0.16052285  0.053
```

```
Errors_ZR[, c(1, 2, 5, 6, 10)]
```

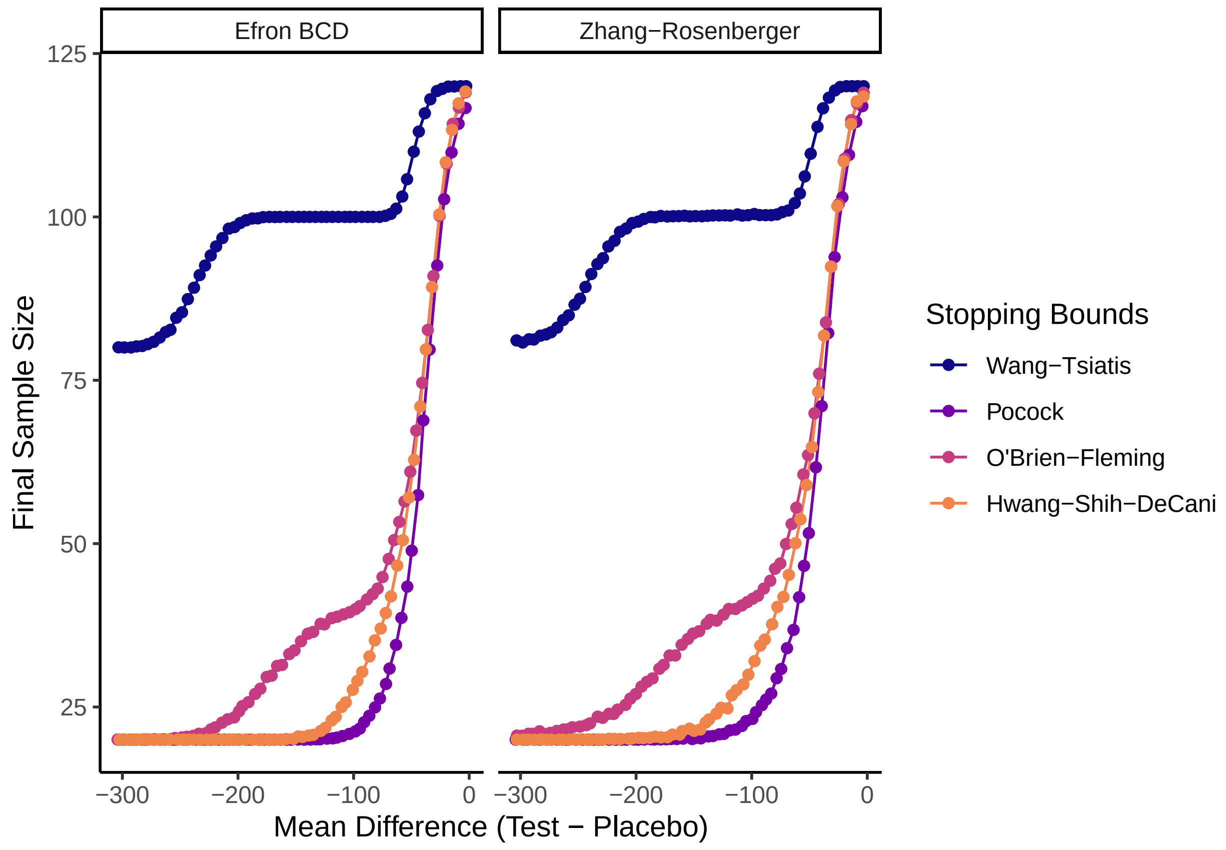
```
##                Method     Randomization Theoretical_Delta Observed_Delta Type_I
## 1              Pocock Zhang-Rosenberger                 0      0.3254817  0.043
## 2                  OF Zhang-Rosenberger                 0     -0.3191846  0.061
## 3                  WT Zhang-Rosenberger                 0      0.2795089  0.032
## 4 Hwang-Shih-DeCani Zhang-Rosenberger                 0     -0.5352344  0.053
```

```
#Data Frame of Finalized results
power_total <- rbind(WT_Power_BCD, Pocock_power_BCD, OF_power_BCD, sfHSD_Power_BCD,
                     WT_Power_ZR, Pocock_power_ZR, OF_power_ZR, sfHSD_Power_ZR)

#Power plots
power <- (ggplot(data=power_total[power_total$Treatment_Mean>=660, ],
                     aes(x=Observed_Delta, y=Empirical_Power,
                         color=Method))
              +geom_line()
              +geom_point()
              +facet_grid(cols=vars(Randomization))
              +labs(x="Mean Difference (Test - Placebo)", y="Power")
              +theme_classic()
              +scale_color_viridis(begin=0, end=0.7,
                                   option="C",discrete=TRUE,
                                   labels=c("Wang-Tsiatis", "Pocock",
                                            "O'Brien-Fleming", "Hwang-Shih-DeCani")))
power + guides(color=guide_legend(title="Stopping Bounds"))
```

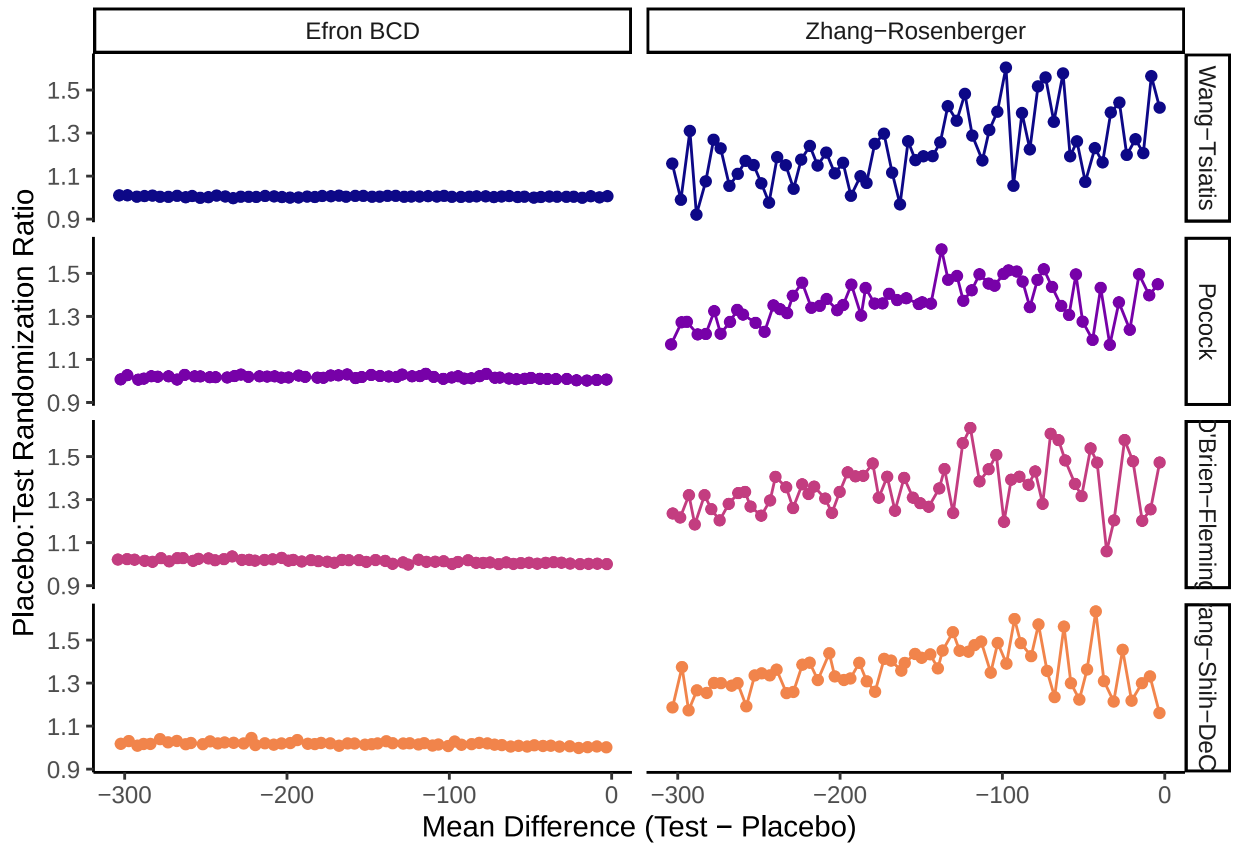


```
#Final sample sizes at end of studies
sample <- (ggplot(data=power_total,
                  aes(x=Observed_Delta, y=Mean_Sample_Size,color=Method))
             +geom_line()
             +geom_point()
             +facet_grid(cols=vars(Randomization))
             +labs(x="Mean Difference (Test - Placebo)", y="Final Sample Size")
             +theme_classic()
             +scale_color_viridis(begin=0, end=0.7,
                                  option="C",discrete=TRUE,
                                  labels=c("Wang-Tsiatis", "Pocock",
                                           "O'Brien-Fleming", "Hwang-Shih-DeCani")))
sample + guides(color=guide_legend(title="Stopping Bounds"))
```

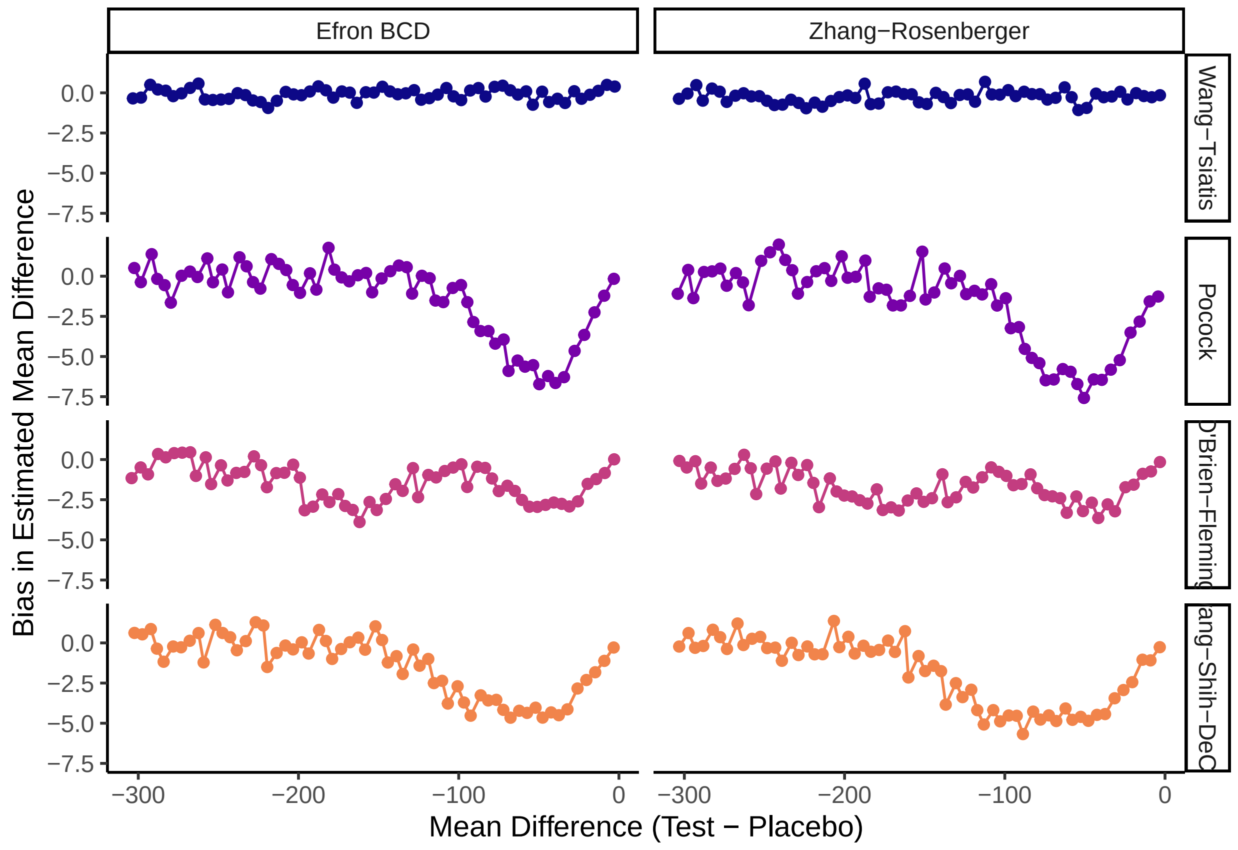


```
#Change Row Names for final facet plot
stop_rule <- c("Wang-Tsiatis", "Pocock", "O'Brien-Fleming", "Hwang-Shih-DeCani")
names(stop_rule) <- c("WT", "Pocock", "OF", "Hwang-Shih-DeCani")


rand <- (ggplot(data=power_total,
                    aes(x=Observed_Delta, y=Randomization_Ratio,color=Method))
             +geom_line()
             +geom_point()
             +facet_grid(cols=vars(Randomization), rows=vars(Method),
                          labeller=labeller(Method=stop_rule))
             +labs(x="Mean Difference (Test - Placebo)", y="Placebo:Test Randomization Ratio")
             +guides(color=FALSE)
             +theme_classic()
             +scale_color_viridis(begin=0, end=0.7,
                               option="C",discrete=TRUE))
rand
```



```
#Plots for Bias
bias <- (ggplot(data=power_total,
                aes(x=Observed_Delta, y=Bias_Delta,color=Method))
         +geom_line()
         +geom_point()
         +facet_grid(cols=vars(Randomization), rows=vars(Method),
                     labeller=labeller(Method=stop_rule))
         +labs(x="Mean Difference (Test - Placebo)", y="Bias in Estimated Mean Difference")
         +guides(color=FALSE)
         +theme_classic()
         +scale_color_viridis(begin=0, end=0.7,
                              option="C",discrete=TRUE))
bias
```



```
#Wang-Tsiatis stopping boundaries
gsDesign(k=6, test.type=2, alpha=0.025, beta=0.2,
                delta = 0.84, n.fix=120, sfu="WT")
```

```
## Symmetric two-sided group sequential design with
## 80 % power and 2.5 % Type I Error.
## Spending computations assume trial stops
## if a bound is crossed.
##
##
##   Analysis N       Z   Nominal p Spend
##          1  2 6221.99     0.000 0.000
##          2  4  274.98     0.000 0.000
##          3  6   44.35     0.000 0.000
##          4  8   12.15     0.000 0.000
##          5 10    4.45     0.000 0.000
##          6 12    1.96     0.025 0.025
##      Total                    0.0250
##
## ++ alpha spending:
##  Wang-Tsiatis boundary with Delta = -4.
##
## Boundary crossing probabilities and expected sample size
## assume any cross stops the trial
##
## Upper boundary (power or Type I Error)
##           Analysis
##   Theta 1 2 3 4      5      6 Total E{N}
##    0.00 0 0 0 0 0.0000 0.0250 0.025 11.1
##    0.84 0 0 0 0 0.0291 0.7709 0.800 11.1
##
## Lower boundary (futility or Type II Error)
##           Analysis
##   Theta 1 2 3 4 5     6 Total
##    0.00 0 0 0 0 0 0.025 0.025
##    0.84 0 0 0 0 0 0.000 0.000
```