Final_code

Douglas Hannum

4/9/2019

Creating Functions for the Simulations

```
#Theoretically solved lamda for the median, in the truncated exp funct
lambda <- 0.0203821

#Gives a number of random observations (x) from the truncated exponential distr
rdist <- function(x = 1){
        rtexp(x, lambda, endpoint = 100)
}

#Random number from the truncated normal distribution
rdistn <- function(x = 1){
        rtruncnorm(x, a = 0, b = 100, mean = 50, sd = 25)
}
```

```
#A function to create the data frame, n is the sample size, p is the disease
#prevalence

#Dataframe with no censoring -> Intent to treat dataframe
dataf <- function(n,p){
        #n is the number of samples being tested
        #p is the disease rate

        #Setting up the null dataframe
        df <- as.data.frame(matrix(ncol = 2, nrow = n, rep(c(200,0), each = n)))
        colnames(df) <- c('Time','Event')

        #Using a random binomial to determine if an event has occured
        df$Event <- rbinom(n,1,p)
        #Using the truncated exponential distr. to get the time for events
        df[df$Event == 1,]$Time <- rdist(sum(df$Event))

        return (df)
}

#A dataframe using censoring to model the non-compliance of people to the trtmnt
datafc <- function(n,p){
        #n is the number of samples being tested
        #p is the disease rate
        df <- as.data.frame(matrix(ncol = 4, nrow = n,
                                   rep(c(200,0), each = n)))
        colnames(df) <- c('Time1','Event1', 'Time2','Event2')

        #Event one is getting the GVHD
        df$Event1 <- rbinom(n,1,p)
        df[df$Event1 == 1,]$Time1 <- rdist(sum(df$Event1))
```

```r
        #Event two is getting censored for non-compliance
        df$Event2 <- rbinom(n,1,.8)
        df[df$Event2 == 0,]$Time2 <- rdistn(n-sum(df$Event2))

        #Set the actual time for the minimum time between time 1 and 2
        df$Time <- NA
        df$Time <- ifelse(df$Time1 < df$Time2, df$Time <- df$Time1,
                          df$Time <- df$Time2)
        df$Event <- ifelse(df$Time ==df$Time1, df$Event1, df$Event2)

        return (df)
}
```

```r
#Seeing if the survival curve contains 0.50 in its CI at time = 100
surv_sig <- function (df, confint = 0.95){
        #Confint variable lets me vary the alpha to get the desired alpha
        #of 0.05

        #Fitting a survival curve
        fit <- survfit(Surv(Time, Event) ~1, data = df, conf.int = confint)
        #Seeing if the 95% CI passes historic threshold = 0.50
        l <- summary(fit, times = 100)$lower
        return(l > 0.50)
}
```

```r
#Power calculations for intent to treat
sample_power <- function(p = 1, samples = 1000, l = 40 , u = 210 ,
                         by = 10, perm = 1){
        #Creating sequence of sample sizes to be tested
        n <- seq(l,u, by = by)

        #Creating an empty power dataframe
        data <- as.data.frame(matrix(ncol = 1+perm, nrow = length(n),
                                     c(n, rep(0,length(n)*perm))))

        colnames(data) <- c('Sample Size',paste0('Power',1:perm))


        for(k in 1:perm){
                #Going through all the different sample sizes
                for (i in 1:length(n)){
                        #Creating an empty vector to store results
                        vect <- rep(NA,samples)
                        #Going through the designated simulations
                        for (j in 1:samples){
                                #Create df to be test
                                df <- dataf(n[i],p)
                                #Test the df for significance
                                vect[j] <- surv_sig (df)
                        }
                        #Append power resuls
                        data[i,1+k] <- round(mean(vect),4)
                }
        }
```

```r
        return (data)
}

#Power calculation with censoring
sample_power_c <- function(p = c(.40,.35,.3), samples = 10, l = 40 , u = 100,
                         by = 10, confint = 0.94, c = TRUE){
        # p is the disease rates to be tested
        # samples refers to the number of simulations
        # l is the lower bound of the sample size
        # u is the upper bound of the sample size
        # by is the interval between sample sizes to be tested
        # confint is the confidence interval applied to the logrank test


        n <- seq(l, u, by = by)

        data <- as.data.frame(matrix(ncol = 3, nrow = length(n)*length(p),0))

        colnames(data) <- c('Sample Size', 'Rate', 'Power')
        data$`Sample Size` <- rep(n,length(p))
        data$Rate <- as.factor(rep(p, each = length(n)))

        for (k in 1:length(p)){
                for (i in 1:length(n)){

                        vect <- rep(NA,samples)
                        for (j in 1:samples){
                                if (c == T){
                                        df <- datafc(n[i],p[k])
                                } else {
                                        df <- dataf(n[i], p[k])
                                }
                                vect[j] <- surv_sig(df, confint)
                        }
                        data[length(n)*(k-1) + i, 3] <- sum(vect)/samples
                }
        }
        return (data)
}

#Sample power with an interim analysis
sample_power_int <- function(p = .5, samples = 10, l = 40 , u = 100,
                         by = 10, confint1 = 0.85, confint2 = 0.85, c = TRUE){

        # two different variables to control the alpha for the interim and
        # for the final analysis
        n <- seq(l, u, by = by)

        data <- as.data.frame(matrix(ncol = 5, nrow = length(n)*length(p),0))

        colnames(data) <- c('Sample Size', 'Rate', 'Power','Rint',
                          'Rfin')
```

```r
        data$`Sample Size` <- rep(n,length(p))
        data$Rate <- as.factor(rep(p, each = length(n)))

        for (k in 1:length(p)){
                for (i in 1:length(n)){

                        vect <- rep(NA,samples)

                        #Creating empty vectors to track where trials fial
                        Rfin <- 0
                        Rint <- 0

                        #Interim setting n = 2
                        for (j in 1:samples){
                                #Testing the data halfway through enrollment
                                if (c == T){
                                        df <- datafc(n[i]/2,p[k])
                                } else {
                                        df <- dataf(n[i]/2, p[k])
                                }

                                sig <- surv_sig(df, confint1)

                                #Seeing if the interim fails
                                if (sig == 0){
                                        vect[j] <- sig
                                        Rint <- Rint + 1
                                }
                                # If the interim does not fail then do the rest
                                else{
                                        if (c == T){
                                                df <- rbind(df,
                                                            datafc(n[i]/2,p[k]))
                                        } else {
                                                df <- rbind(df,
                                                            dataf(n[i]/2, p[k]))
                                        }
                                        sig <- surv_sig(df,confint2)
                                        vect[j] <- sig
                                }
                        }
                        data[length(n)*(k-1) + i, 3] <- sum(vect)/samples
                        data[length(n)*(k-1) + i, 4] <- Rint
                        data[length(n)*(k-1) + i, 5] <- samples - Rint - sum(vect)
                }
        }
        return (data)
}

# A function to get either a dataframe or figure output
spower <- function (p = c(.40,.35,.3), samples = 1000, l = 40, u = 100, by = 10,
                    c = F, i = F, PowerOfInterest = .80, plot = T){
```

```
    if (i == T){
        if (c == T){
            sp <- sample_power_int(p = p, samples = samples,
                                   l = l, u = u,
                                   by = by, c = T)
        } else{
            sp <- sample_power_int(p = p, samples = samples,
                                   l = l, u = u,
                                   by = by, c = F)
        }
    } else{
        if (c == T){
            sp <- sample_power_c(p = p, samples = samples,
                                 l = l, u = u,
                                 by = by, c = T)
        } else{
            sp <- sample_power_c(p = p, samples = samples,
                                 l = l, u = u,
                                 by = by, c = F)
        }
    }
    sp$SSInflated <- ceiling(sp$`Sample Size`/.85)
    plt <- ggplot(data = sp, aes(x = SSInflated, y = Power,
                                 colour = Rate)) +
        geom_point() + geom_line() +
        geom_hline(yintercept = PowerOfInterest, linetype = 2) +
        xlab ('Sample Size') + theme_bw()
    if(plot == T){
        return(plt)
    }else{
        return(sp)
    }
}

#If I just want a power output, for R-Shiny app
power_only <- function (n, p = .5, samples = 10000, c = F, i = F){
    #Decreasing by the 0.15 that will dropout
    n <- n * .85

    if (i == T){
        if (c == T){
            sp <- sample_power_int(p = p, samples = samples,
                                   l = n, u = n,
                                   by = 1, c = T)
        } else{
            sp <- sample_power_int(p = p, samples = samples,
                                   l = n, u = n,
                                   by = 1, c = F)
        }
    } else{
        if (c == T){
            sp <- sample_power_c(p = p, samples = samples,
                                 l = n, u = n,
```

```
                                    by = 1, c = T)
                } else{
                        sp <- sample_power_c(p = p, samples = samples,
                                                l = n, u = n,
                                            by = 1, c = F)
                }
        }
        return(sp$Power)
}
```

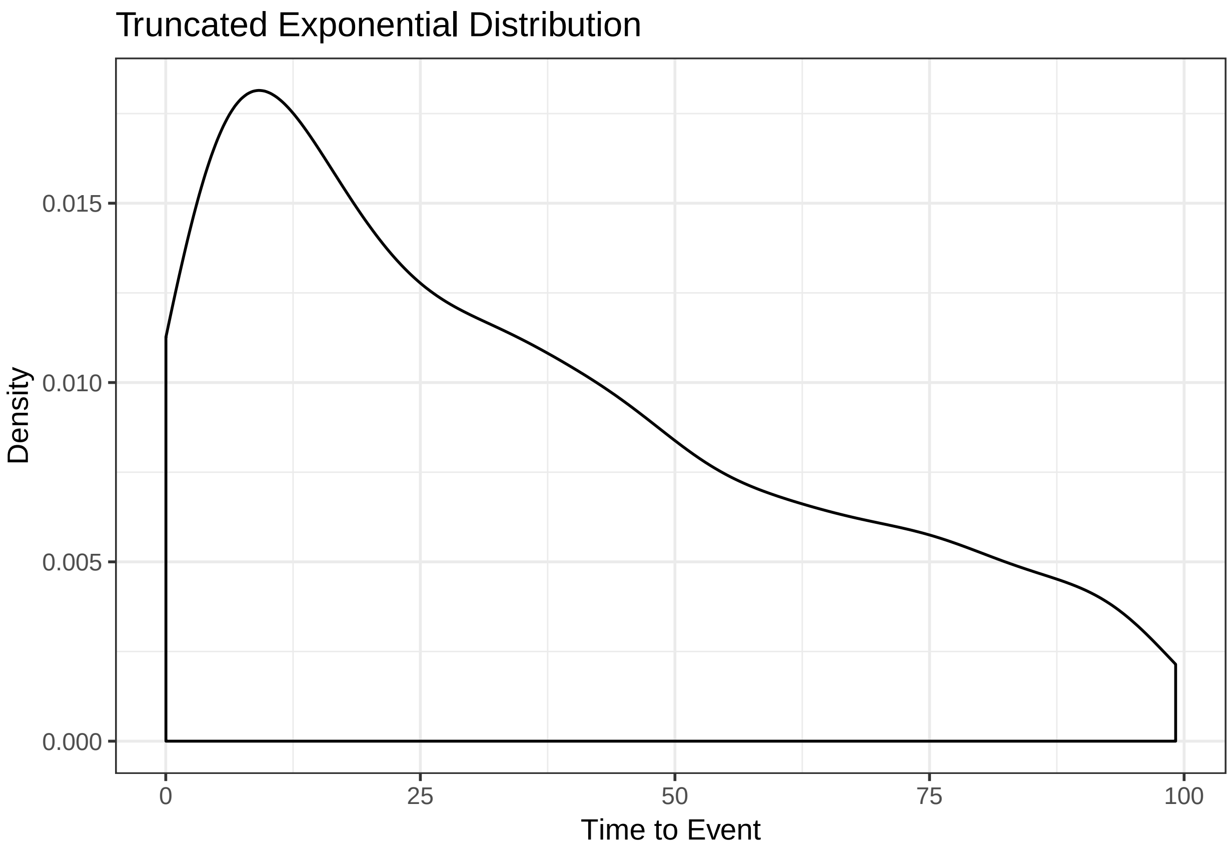
Look at the distributions

Truncated Exponential Distribution

```
#Taking a 1000 random samples
w <- rdist(1000)
summary(w)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.01513 11.33084 29.41073 34.89973 54.12814 99.16763
```

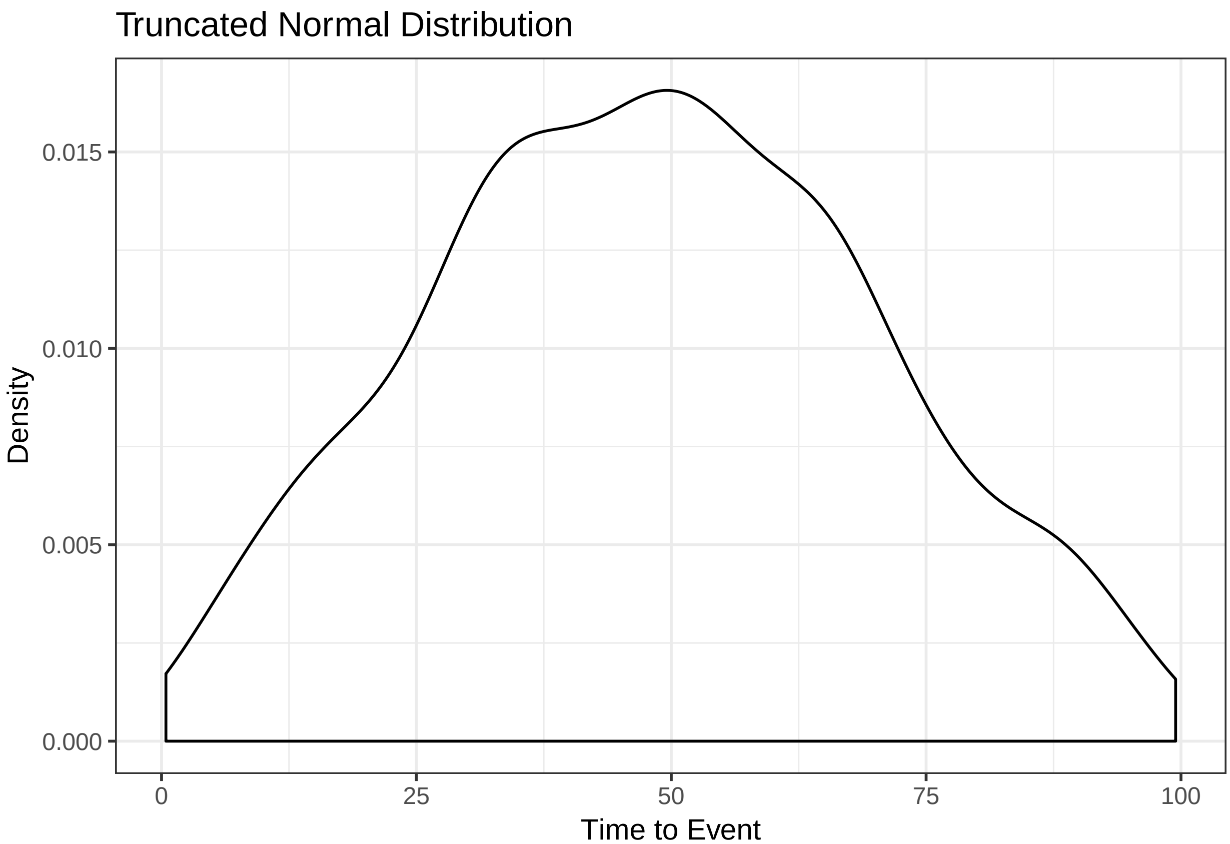
```
wd <- as.data.frame(matrix(ncol = 2, nrow = 1000, c(w, 1:1000)))
wd <- wd[order(wd$V1),]

#Plot the distribution
ggplot(data = wd, aes(x = V1)) +
        geom_density() + ylab ('Density') + xlab ('Time to Event') +
        ggtitle('Truncated Exponential Distribution') +
        theme(plot.title = element_text(hjust = 0.5)) +
        theme_bw()
```




```
#ggsave('./images/tr_exp_distr.png', device = 'png', units = 'in',
#       width = 6, height = 4)
```

Truncated Normal Distribution

```
rdistn <- function(x = 1){
        rtruncnorm(x, a = 0, b = 100, mean = 50, sd = 25)
}

wn <- rdistn(1000)
wnd <- as.data.frame(matrix(ncol = 2, nrow = 1000, c(wn, 1:1000)))
wnd <- wnd[order(wnd$V1),]

#Plot the distribution
ggplot(data = wnd, aes(x = V1)) +
        geom_density() + ylab ('Density') + xlab ('Time to Event') +
        ggtitle('Truncated Normal Distribution') +
        theme(plot.title = element_text(hjust = 0.5)) +
        theme_bw()
```

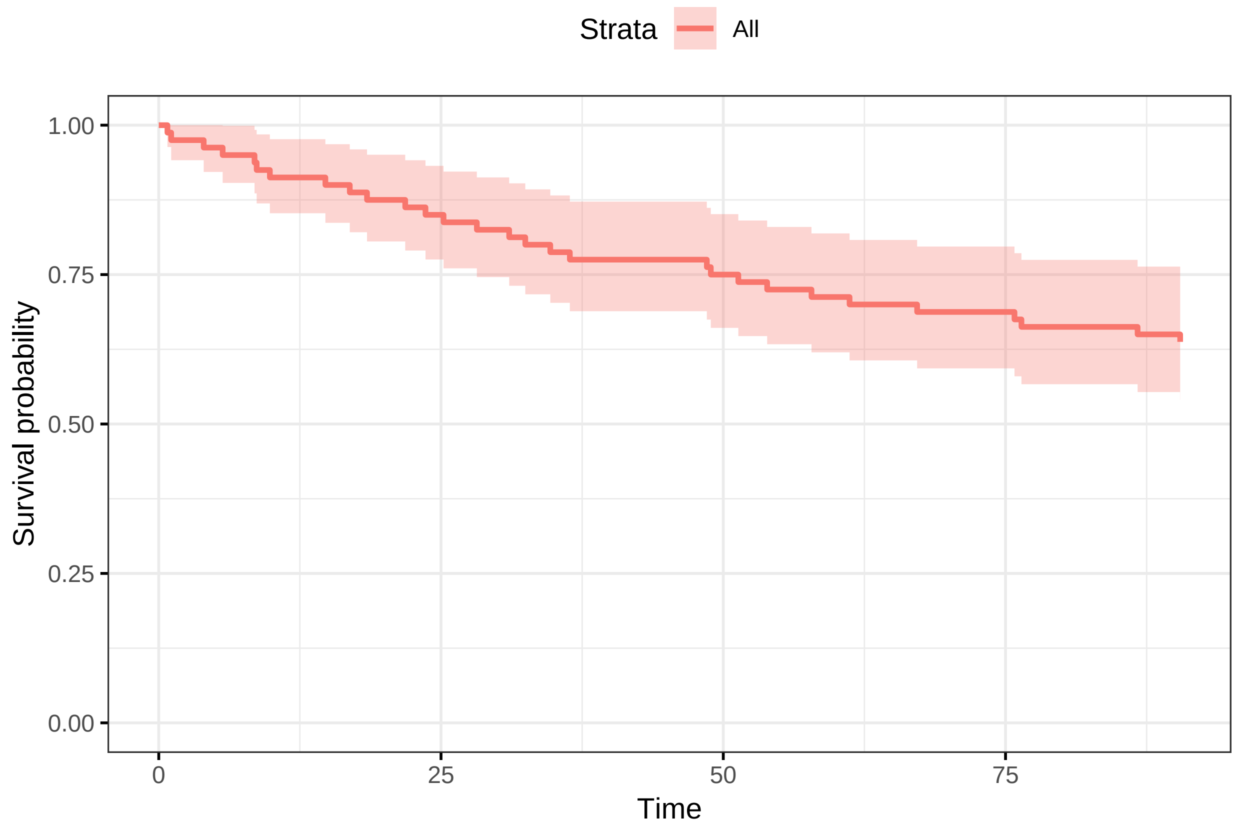



```
# ggsave('./images/tr_norm_distr.png', device = 'png', units = 'in',
#        width = 6, height = 4)
summary(wn)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4282 32.3519 48.2264 48.4764 64.0626 99.4756
```

Example Survival Plot

```
#Setting the dataframe for sample size 80 and disease rate 0.35
df1 <- dataf(80, .35)
#Fit the data to a survival curve
fit1 <- survfit(Surv(Time, Event) ~1, data = df1)

#Subset the data to only look at times less than 100
ss <- subset(surv_summary(fit1), time < 100)
#Plot the survival curve
ggsurv <- ggsurvplot (ss, main = "One Simulation (n = 80, rate = 0.35)",
                      conf.int = T, risk.table = T, risk.table.col = 'strata',
                      ggtheme = theme_bw())
ggsurv
```




```
# ggsave('./images/smple_km_plot.png', device = 'png', units = 'in',
#        height = 3, width = 6)
```

Simulations

These are the simulations I will be using for my paper. They are labeled based on whether or not they include censored data and/or an interim analysis

```
Cn_In_sample <- sample_power_c(samples = 1000, l = 30, u = 220, by = 3, c = F)
Cy_In_sample <- sample_power_c(samples = 1000, l = 30, u = 220, by = 3, c = T)
```

```
start <- Sys.time()
Cn_Iy_sample40 <- sample_power_int(samples = 1000, l = 30, u = 250, by = 4,
                                   c = F, p = .4)
Cn_Iy_sample35 <- sample_power_int(samples = 1000, l = 30, u = 250, by = 4,
                                   c = F, p = .35)
Cn_Iy_sample30 <- sample_power_int(samples = 1000, l = 30, u = 250, by = 4,
                                   c = F, p = .3)
Cy_Iy_sample40 <- sample_power_int(samples = 1000, l = 30, u = 270, by = 4,
                                   c = T, p = .4)
Cy_Iy_sample35 <- sample_power_int(samples = 1000, l = 30, u = 270, by = 4,
                                   c = T, p = .35)
Cy_Iy_sample30 <- sample_power_int(samples = 1000, l = 30, u = 270, by = 4,
                                   c = T, p = .3)

Cn_Iy_sample <- rbind(Cn_Iy_sample40, Cn_Iy_sample35, Cn_Iy_sample30)
```

```r
Cy_Iy_sample <- rbind(Cy_Iy_sample40, Cy_Iy_sample35, Cy_Iy_sample30)
end <- Sys.time()
end-start
```

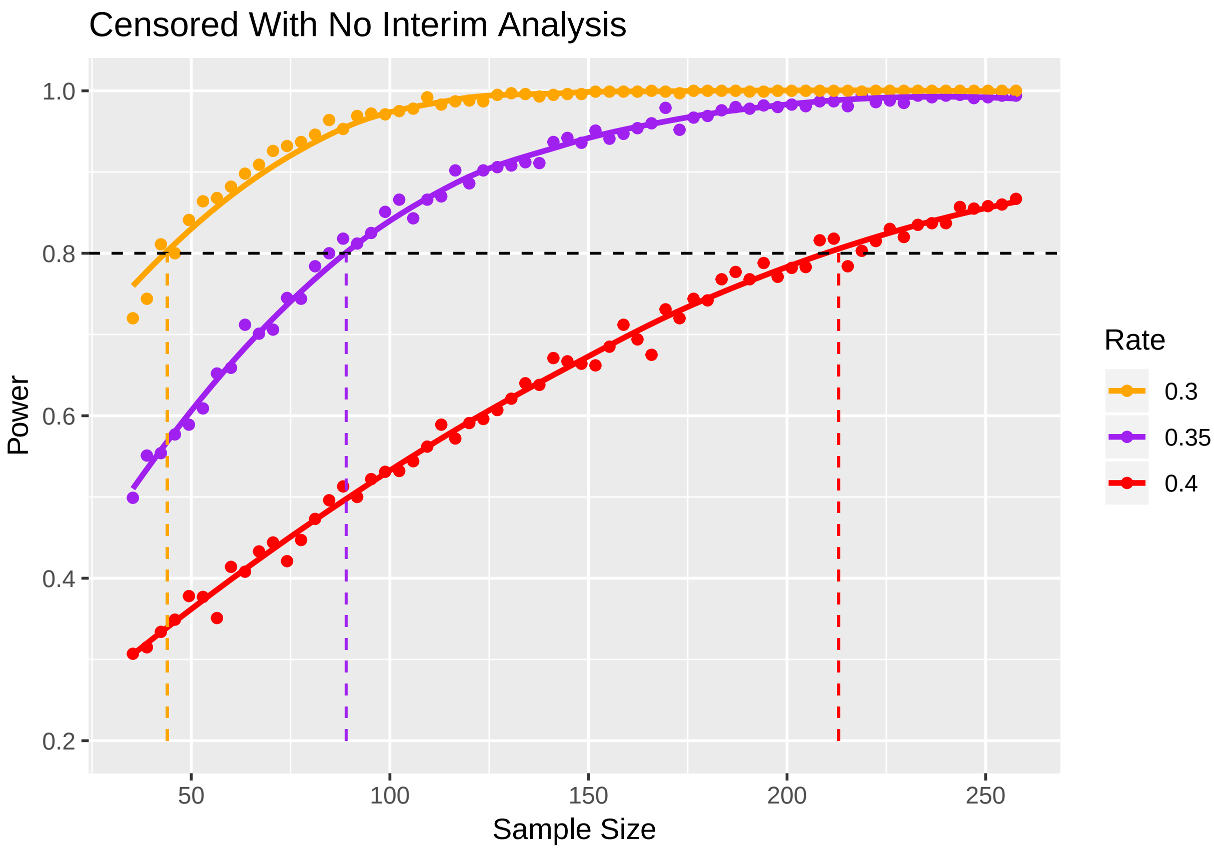
```
## Time difference of 21.13837 mins
```

```r
# GGplot function to expedite things
powerplot <- function(df, x,w,z,title = 'Such a Cool Title'){
        # Accounting for the 15% dropout rate
        df$SSInflated <- df$`Sample Size`/.85

        ggplot(data = df, aes (x = SSInflated, y = Power, color = Rate)) +
                # Getting points and a smooth curve to fit them
                geom_point() + geom_smooth(se = F) +

                # Creating a horizontal line to elucidate the 80% power
                geom_hline (yintercept = .80, linetype = 2, color = 'black') +
                ggtitle (title) + xlab ('Sample Size') +
                scale_color_manual(values = c('orange','purple','red')) +

                # Creating line segments to visualize where the sample size
                # crosses the 80% power cutoff
                geom_segment (aes(x = x, xend = x, y = .2, yend = .8),
                              linetype = 2, color = 'orange') +
                geom_segment (aes(x = w, xend = w, y = .2, yend = .8),
                              linetype = 2, color = 'purple') +
                geom_segment (aes(x = z, xend = z, y = .2, yend = .8),
                              linetype = 2, color = 'red')

}

powerplot(Cn_In_sample, 41, 81, 191, "ITT With No Interim Analysis")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

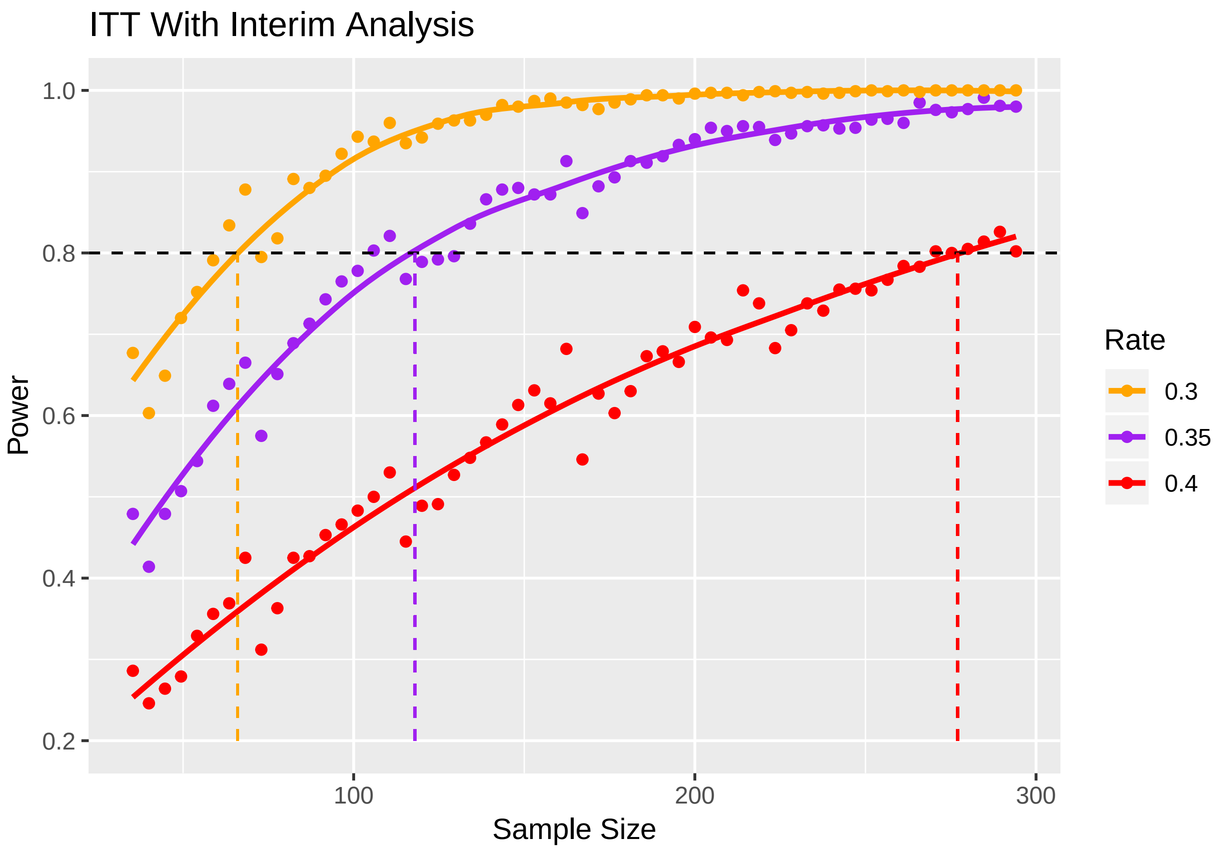

```
# x-intercepts are : 41, 81, 191
#ggsave('./images/itt_no_interim.png', device = 'png', height = 3, width = 6)

powerplot(Cy_In_sample, 44, 89, 213, "Censored With No Interim Analysis")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



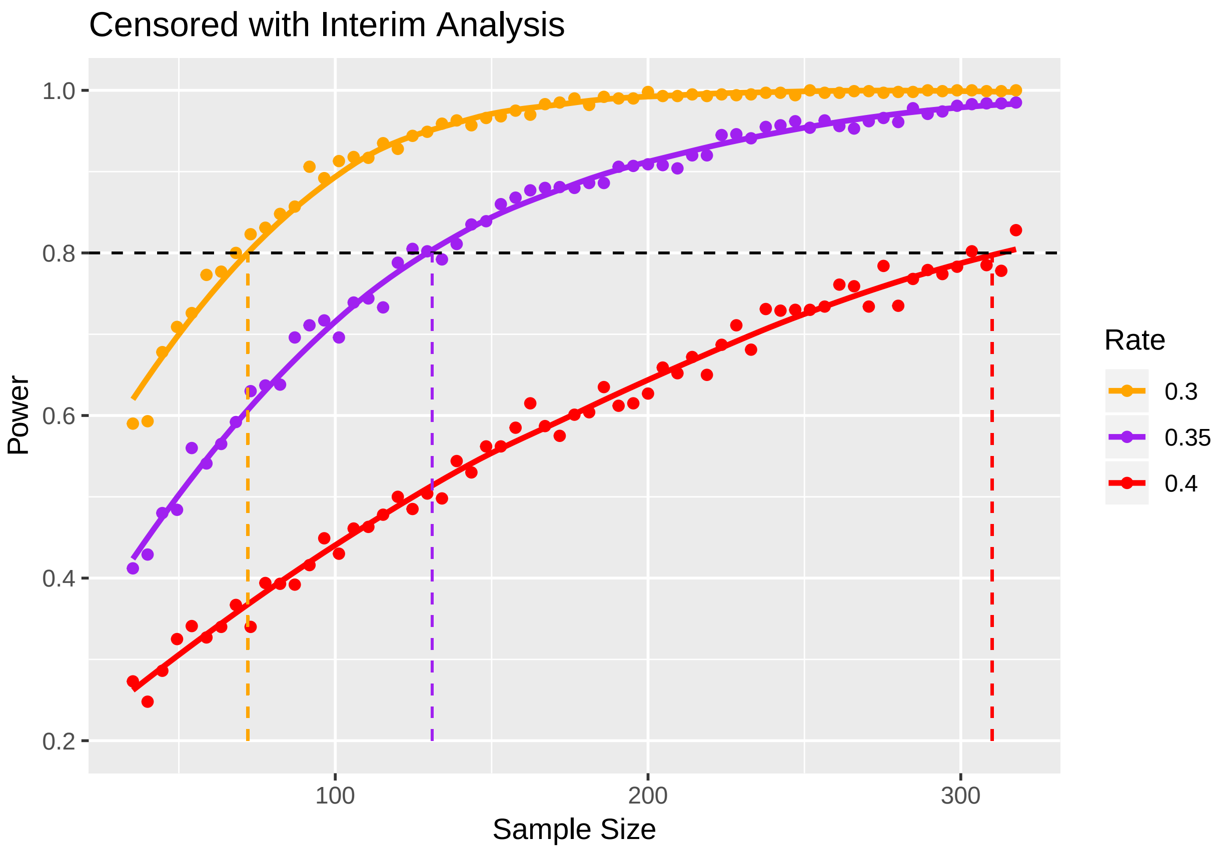

```r
# x-intercepts are : 44, 89, 213
#ggsave('./images/cen_no_interim.png', device = 'png', height = 3, width = 6)

Cn_Iy_sample$Rate <- factor(Cn_Iy_sample$Rate, levels = c(.30,.35,.4))
powerplot(Cn_Iy_sample, 66, 118, 277, "ITT With Interim Analysis")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# x-intercepts are : 66, 118, 277
#ggsave('./images/itt_yes_interim.png', device = 'png', height = 3, width = 6)

Cy_Iy_sample$Rate <- factor(Cy_Iy_sample$Rate, levels = c(.30,.35,.4))
powerplot(Cy_Iy_sample, 72, 131, 310, "Censored with Interim Analysis")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# x-intercepts are : 72, 131, 310
#ggsave('./images/cen_yes_interim.png', device = 'png', height = 3, width = 6)

# how I got the sample sizes where the smooth line crosses 80% power
p <- ggplot(data = Cy_Iy_sample, aes (x = `Sample Size`/.85, y = Power,
                                      color = Rate)) +
        geom_point() + stat_smooth()
#ggplot_build(p)$data[[2]]
```

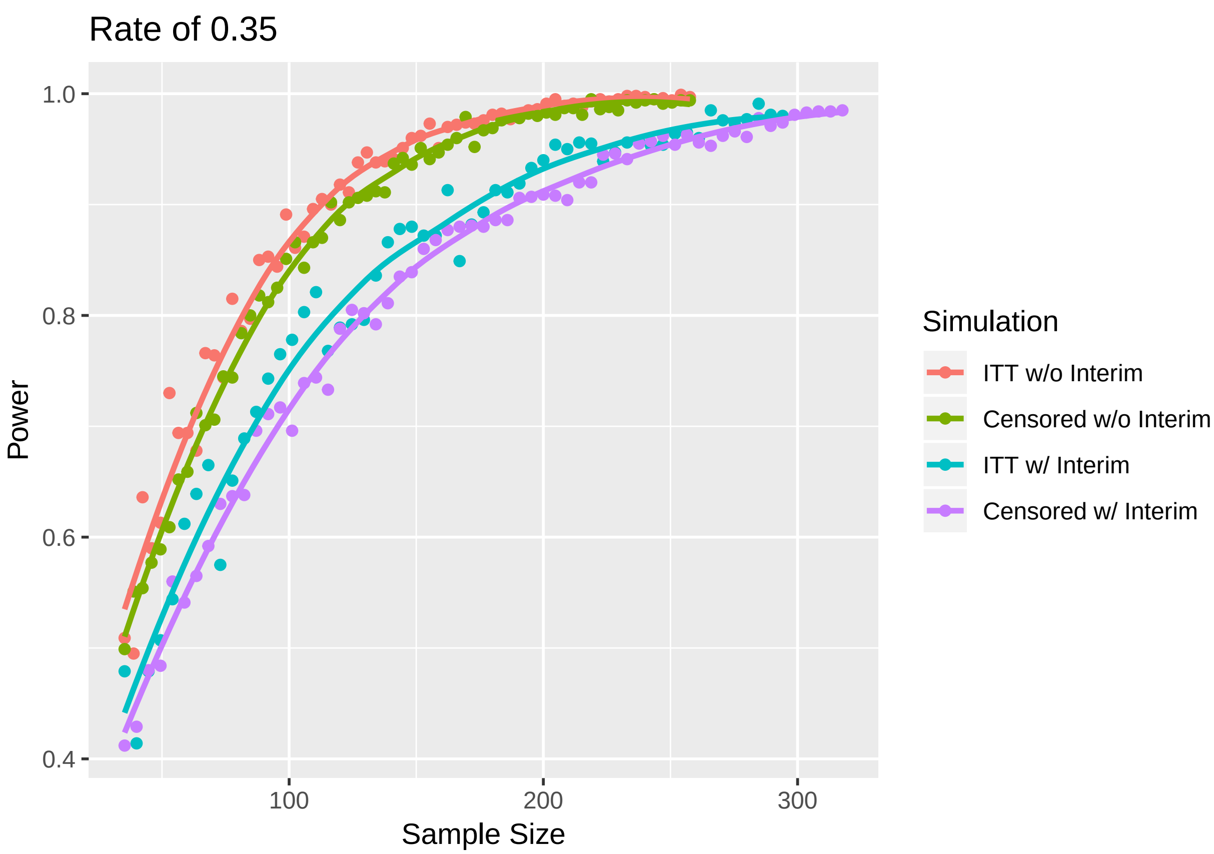
Alternative plots

Looking at building one ggplot for one rate and the four different types of power calculations, in order to get an idea of the differences in power. This was doen for each rate but only one is included in this document. All of them look relatively similar, with all the lines being proportional to one another.

```
power35 <- rbind(Cn_In_sample[Cn_In_sample$Rate == .35,],
                 Cy_In_sample[Cy_In_sample$Rate == .35,])
power35 <- rbind(power35,
                 Cn_Iy_sample[Cn_Iy_sample$Rate == .35,-c(4,5)])
power35 <- rbind(power35,
                 Cy_Iy_sample[Cy_Iy_sample$Rate == .35,-c(4,5)])
power35$SSInflated <- power35$`Sample Size`/.85

power35$Simulation <- mapvalues(power35$Sim,
                                from = c('Cn_In', 'Cy_In', 'Cn_Iy', 'Cy_Iy'),
                                to = c('ITT w/o Interim',
```

```
                                    'Censored w/o Interim', 'ITT w/ Interim',
                                    'Censored w/ Interim'))
power35$Simulation <- factor(power35$Simulation,
                             levels = c('ITT w/o Interim',
                                        'Censored w/o Interim',
                                        'ITT w/ Interim',
                                        'Censored w/ Interim'))

ggplot(power35, aes(x = SSInflated, y = Power, color = Simulation)) +
        geom_point() + geom_smooth(se = F) +
        ggtitle ('Rate of 0.35') +
        xlab ('Sample Size')
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggsave('./images/power35.png', device = 'png', width = 6, height = 3,
       units = 'in')
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Rejection in interim or final analysis

```
# For the ITT analysis
sum(Cn_Iy_sample$Rint) / sum(Cn_Iy_sample$Rfin)
```

```
## [1] 14.2398
```

```r
# For the Censored analysis
sum(Cy_Iy_sample$Rint) / sum(Cy_Iy_sample$Rfin)
```

```
## [1] 13.07041
```