20201023-p8133_computingassignment_jsg2145

Jared Garfinkel

10/23/2020

Question 1

Part a

```
typeIerr = function(n = 10000) {
  sim_dat = rbinom(n, 20, .25)
  output = NULL

  for (i in 1:n) {
    if (sim_dat[[i]] > 5) {
      output = rbind(output, sim_dat[[i]])
    }
  }

  typeIerror = length(output)/n
  return(typeIerror)
}

typeIerror = typeIerr()
```

The type I error rate under the null, response rate = 0.25, is 0.3823.

```
output = NULL
pow = function(n = 10000) {
  sim_dat = rbinom(n, 20, .4)

  for (i in 1:n) {
    if (sim_dat[[i]] > 5) {
      output = rbind(output, sim_dat[[i]])
    }
  }

  power = nrow(output)/n
  return(power)
}

power = pow()
```

The power under the alternative, response rate = 0.4, is 0.8749.

```
# The expected sample size is equal to 20 times the proportion of times the futility trial threshold is

exp_ss = 20*typeIerror + 71*(1-typeIerror)
```

The expected sample size under these simulations is 51.5027, or 52.

Part b

```
gonogo = function(stage1, stage2, null, alt, n1, n2) {
  res = vector(mode = "list", length = stage1)
  for(i in 1:stage1) {
    for(j in 1:stage2){
      res[[c(i,j)]] = bind_cols("n1_response" = i, "n2_response" = j)
    }
    res[[i]] = map(res[[i]], ~bind_rows(.))
  }

  sum_df = bind_rows(res) %>%
    mutate(errorI = dbinom(n1_response, stage1, null) * dbinom(n2_response, stage2, null),
           power = dbinom(n1_response, stage1, alt) * dbinom(n2_response, stage2, alt)) %>%
    filter(n1_response > n1-1,
           n1_response + n2_response > n2-1) %>%
    summarize(sum_error = sum(errorI),
              sum_power = sum(power))

  return(sum_df)
}
```

Expected number of drugs identified to be effective

```
sum_df = gonogo(stage1 = 20, stage2 = 51, null = .25, alt = 0.4, n1 = 6, n2 = 24)
```

The type I error rate under the null is 0.0489.

The power under the alternative is 0.8025

The expected number of drugs given a go decision is the number of drugs that work times the power (the probability of rejecting the null under the alternative) plus the number of drugs that don't work times the type I error rate (the probability of rejecting the null when the null is true).

```
exp_godrug = 5*pull(sum_df, sum_power) + 95*pull(sum_df, sum_error)
```

So the expected number of drugs given a go decision can be calculated to be 8.6623 or 9.

The expected number of patients enrolled to the trial is the number of expected patients in each trial times 100.

Expected total number of patients enrolled to the 100 trials

The expected number of patients in each trial is the size of the futility trial times the type I error plus the size of the two-stage trial when the futility threshold is reached.

```
exp_ssnull = 20*pbinom(5, 20, .25) + 71*(1-pbinom(5, 20, .25))
exp_ssalt = 20*pbinom(5, 20, .4) + 71*(1-pbinom(5, 20, .4))
exp_ss2 = 5/100*exp_ssalt + 95/100*exp_ssnull
```

The expected sample size of each trial is 40.7777 or 41.

So, the expected number of patients for all trials is about 4077.

Expected number of patients not having an MRI response among the enrolled.

The number of patients expected to not have an MRI response among the enrolled will be the number of patients enrolled in each trial times the true response rate for the drug in that trial.

```
norespratenull = 95 * .25 * exp_ssnull
norespratealt = 5 * .4 * exp_ssalt
norespratetot = norespratenull + norespratealt
```

The number of patients expected not to have a response is 1067.8885272.

Part c

Expected number of drugs identified to be effective

```
fixed = function(n, go, null, alt) {
  errorI = NULL
  power = NULL
  res = NULL
  for(i in 1:n) {
    errorI[[i]] = dbinom(i, n, null)
    power[[i]] = dbinom(i, n, alt)
    res[[i]] = bind_cols("i" = i, "errorI" = errorI[[i]], "power" = power[[i]])
  }

  fixed_df = res %>%
    bind_rows() %>%
    filter(i > go - 1) %>%
    summarize(sum_error = sum(errorI),
              sum_power = sum(power))

  return(fixed_df)
}
```

```
fixed_df = fixed(n = 62, go = 22, null = 0.25, alt = 0.4)
fixed_df
```

```
## # A tibble: 1 x 2
##   sum_error sum_power
##       <dbl>     <dbl>
## 1    0.0428     0.803
```

```
exp_eff = 5*pull(fixed_df, sum_power) + 95 * pull(fixed_df, sum_error)
```

The expected number of drugs to be considered effective will be 8.0861 or 9.

Expected total number of patients enrolled to the 100 trials

The expected number of patients in the trial is 62 x 100 since there are always the same number of patients enrolled in each trial in a fixed design study.

There are 6200 patients enrolled in the fixed design study.

Expected number of patients not having an MRI response among the enrolled.

```
norespratenull2 = 95 * .25 * 62
norespratealt2 = 5 * .4 * 62
norespratetot2 = norespratenull2 + norespratealt2
```

The expected number of patients with no response is 1596.5 in the fixed design study. Compare to 1067.8885272 in the two-stage design.

Question 2

```
output = vector(mode = "list", length = 20)
for (i in 1:20) {
  for (j in 1:51) {
    output[[c(i, j)]] = bind_cols("n1_response" = i, "n2_response" = j)
    output[[c(i, j)]]$x = list(seq(from = 0.01, to = 0.4, by = 0.003))
  }
  output[[i]] = map(output[[i]], ~bind_rows(.))
}


output = bind_rows(output) %>%
  unnest(x) %>%
  nest(x)

# output %>%
#   unnest() %>%
#   unnest(x)
```

```
prior = function(df = output, null = 0.25, stage1n = 20, stage2n = 71, n1 = 6, n2 = 24) {
  res = df %>%
    unnest(data) %>%
```

```
    mutate(pE1 = qbeta(null, n1_response, (stage1n-n1_response)),
           pE2 = qbeta(null, (n1_response+n2_response), (stage2n-n1_response-n2_response)),
           pS = qbeta(null, 25, 75))

  res  = res %>%
    filter(pE1>pS + x,
           pE2>pS + x) %>%
    # group_by(x) %>%
    # mutate(alpha = 1 - length(x)/nrow(goutput)) %>%
    # select(x, alpha) %>%
    distinct()
  return(res)
}
```

```
res = prior()
```

```
res
```

```
## # A tibble: 48,547 x 6
##    n1_response n2_response     x   pE1   pE2    pS
##          <int>       <int> <dbl> <dbl> <dbl> <dbl>
##  1           7          12 0.01  0.275 0.231 0.220
##  2           7          13 0.01  0.275 0.245 0.220
##  3           7          13 0.013 0.275 0.245 0.220
##  4           7          13 0.016 0.275 0.245 0.220
##  5           7          13 0.019 0.275 0.245 0.220
##  6           7          13 0.022 0.275 0.245 0.220
##  7           7          14 0.01  0.275 0.258 0.220
##  8           7          14 0.013 0.275 0.258 0.220
##  9           7          14 0.016 0.275 0.258 0.220
## 10           7          14 0.019 0.275 0.258 0.220
## # ... with 48,537 more rows
```

According to my calculations, a delta of 0.151 and an alpha of approximately 0.154 will result in a go decision when the 1st stage trial has a response rate equal to or greater than 6.

```
set.seed(1)
n_sim = 10e5
sum(rbeta(n_sim, shape1 = 6.5, shape2 = 15.5) > (rbeta(n_sim, shape1 = 25, shape2 = 75) + 0.151)) / n_si
```

```
## [1] 0.158776
```

```
sum(rbeta(n_sim, shape1 = 5.5, shape2 = 16.5) > (rbeta(n_sim, shape1 = 25, shape2 = 75) + 0.151)) / n_si
```

```
## [1] 0.074492
```

A simulation confirms that when the response rate is 6, the alpha is greater than .154, and when the response rate is 5, the alpha is less than 0.154.

Question 3

The priors can be updated at n = 21, n = 41, and n = 61. There should be a probability that the futility response of less than 6 out of 20 respond.

```
n_sim = 10e5
futility = sum(rbeta(n_sim, .5, 1.5) > rbeta(n_sim, 25, 75) + 0.15)/n_sim
futility
```

```
## [1] 0.253996
```

```
n_sim = 10e5
futility2 = sum(rbeta(n_sim, 6.5, 15.5) > rbeta(n_sim, 25, 75) + 0.15)/n_sim
futility2
```

```
## [1] 0.16094
```

```
n_sim = 10e5
futility3 = sum(rbeta(n_sim, 12.5, 29.5) > rbeta(n_sim, 25, 75) + 0.15)/n_sim
futility3
```

```
## [1] 0.108629
```

```
n_sim = 10e5
futility4 = sum(rbeta(n_sim, 18.5, 43.5) > rbeta(n_sim, 25, 75) + 0.15)/n_sim
futility4
```

```
## [1] 0.080481
```

```
n_sim = 10e5
futility5 = sum(rbeta(n_sim, 24.5, 47.5) > rbeta(n_sim, 25, 75) + 0.15)/n_sim
futility5
```

```
## [1] 0.197761
```

When the null is true the type I error rate is x2 and when the alternative is true, the power is y2.

Question 4

The stopping boundary should be set such that given a delta and alpha pair, the response rate at or above the boundary will result in a go decision and below the boundary will result in a no-go decision. The first stopping boundary is given as $S_{20} = 6$. The decision rule should be such that the delta and alpha can be applied to each interim analysis and achieve the final go decision of $S_{71} = 24$

Given:

pE ~ beta(0.5, 1.5) pS ~ beta(25, 75) $\delta = 0.15$ $\alpha = 0.2$

After the first 20 patients, pE ~ beta(0.5 + n_1, 1.5 + 20 - n_1), where n_1 is the number of responses in the first stage.

After 40 patients, pE ~ beta(0.5 + $n_1 + n_2$, 1.5 + 40 - $(n_1 + n_2)$).

pE is updated at 60 patients and 71 patients as such.

```
df4_maker = function(n1 = 7) {
  output4 = vector(mode = "list", length = 20)
  for (i in 1:20) {
    for(j in 1:20) {
      output4[[c(i, j)]] = bind_cols("n1_response" = i, "n2_response" = j)
      output4[[c(i, j)]]$k = list(seq(from = 1, to = 20))
      output4[[c(i, j)]]$l = list(seq(from = 1, to = 11))
      output4[[c(i, j)]]$x = list(seq(from = .15, to = 0.35, by = 0.02))
      output4[[c(i, j)]]$n1 = n1
      # output4[[c(i, j)]]$n2 = list(seq(13, 14))
      # output4[[c(i, j)]]$n3 = list(seq(19, 21))
      output4[[c(i, j)]]$n4 = 24
    }
    output4[[i]] = map(output4[[i]], ~bind_rows(.))
  }
  df4 = bind_rows(output4) %>%
    rename("n3_response" = k,
           "n4_response" = l) %>%
    unnest(x) %>%
    unnest(n3_response) %>%
    unnest(n4_response) %>%
    nest(-x)
  return(df4)
}
```

By creating a dataframe of all possible outcomes it is possible to calculate the probability of each response rate for each interim analysis. This function creates the responses, there are 20 possible response rates in the first trial, 20 potential response rates in the 2nd trial, 20 potential response rates in the 3rd trial and 11 in the 4th and final trial. This means there are 20x20x20x11 potential outcomes, or 88,000.

```
df5 = df4_maker()
```

```
df5 = df5 %>%
  mutate(n3 = list(19:21),
         n2 = list(13:14)) %>%
  unnest(n2) %>%
  unnest(n3)

# df4
```

I hypothesized that the cutoff rates for futility could be 6 or 7 in each interim analysis to reduce the number of potential futility response rates. With a futility response of 7 in the first trial, there can be a cutoff of 13 (7 + 6) or 14 (7 + 7) in the 2nd round. In the 3rd round there should be between 19 (7 + 6 + 6) and 21 (7 + 7 + 7) responses to avoid futility. There must be 24 responses ($S_{71} = 24$) to avoid futility in the final check.

```
prior5 = function(df = df5, null = 0.25, stage1n = 20, stage2n = 40, stage3n = 60, stage4n = 71) {
  res5 = df %>%
    mutate(pE1 = qbeta(null, (0.5+n1_response), (1.5+stage1n-n1_response)),
           pE2 = qbeta(null, (0.5+n1_response+n2_response), (1.5+stage2n-n1_response-n2_response)),
           pE3 = qbeta(null, (0.5+n1_response + n2_response + n3_response), (1.5+stage3n-n1_response-n2
           pE4 = qbeta(null, (0.5+n1_response + n2_response + n3_response + n4_response), (1.5+stage4n-
```

```
            # pS = dbeta(null+pull(df, x), 25, 75))
    # filter(n1_response > n1-1,
    #        (n1_response + n2_response) > n2-1,
    #        (n1_response + n2_response + n3_response) > n3-1,
    #        (n1_response + n2_response + n3_response + n4_response) > n4-1)
  return(res5)
}
```

I calculated the quantile of each response using an updated beta distribution for that number of responses in each trial. Then, I filtered on only the response rates that would pass the futility interim analysis.

When the null is true, there is a type I error rate of x1, and when the alternative is true, there is a power of y1.

```
map_df5 = df5 %>%
  mutate(data = map(data, prior5))
```

```
map_df5_nested = map_df5 %>%
  unnest() %>%
  mutate(pS = qbeta(.25, shape1 = 25, shape2 = 75)) %>%
  group_by(x, n1_response, n2_response, n3_response, n4_response) %>%
  filter(pE1 > pS + x,
         pE2 > pS + x,
         pE3 > pS + x,
         pE4 > pS + x) %>%
  # ungroup() %>%
  # group_by(n1_response, n2_response, n3_response, n4_response) %>%
  # filter(n1_response > n1-1,
  #        (n1_response + n2_response) > n2-1,
  #        (n1_response + n2_response + n3_response) > n3-1,
  #        (n1_response + n2_response + n3_response + n4_response) > n4-1)
  nest()
```

```
# map_df51_nested = map_df5_nested %>%
#   mutate(data = map(data, ~filter(., alpha1 > 0.2,
#                                   alpha2 > 0.2,
#                                   alpha3 > 0.2,
#                                   alpha4 > 0.2)))
```

```
map_df5_nested %>%
  unnest() %>%
  select(n1_response, n2_response, n3_response, n4_response, x, pE1, pE2, pE3, pE4, pS) %>%
  distinct() %>%
  filter(x == 0.15) %>%
  head()
```

```
## # A tibble: 6 x 10
## # Groups:   x, n1_response, n2_response, n3_response, n4_response [6]
##   n1_response n2_response n3_response n4_response     x   pE1   pE2   pE3   pE4
##         <int>       <int>       <int>       <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1          10           8           8           4  0.15 0.405 0.388 0.385 0.378
## 2          10           8           8           5  0.15 0.405 0.388 0.385 0.392
```

```
## 3          10           8           8           6  0.15 0.405 0.388 0.385 0.406
## 4          10           8           8           7  0.15 0.405 0.388 0.385 0.419
## 5          10           8           8           8  0.15 0.405 0.388 0.385 0.433
## 6          10           8           8           9  0.15 0.405 0.388 0.385 0.447
## # ... with 1 more variable: pS <dbl>
```

Then I added the posterior distribution, pS (given), to the dataframe.

The alpha values can be calculated to be the average of the values of pE that are greater than $p_S + \delta$ for all values of potential responses. For instance, if the response in the first trial is below the futility decision no more participants are recruited. Further, if the futility decision is reached in an earlier trial, then the stopping time is recorded at the end of that trial (n = 20, 40, 60 or 71).

So, only if the probability of observing a response rate for the prior based on the updated prior distribution (pE) is greater than the probability of choosing the same value under the posterior (pS) with a buffer of δ in less than α proportion of cases then the conditions are satisfied.

Using the direct calculation method, it is possible to determine that response rates in each of the 4 interim analyses of 10, 18, 26, and 30 will satisfy the conditions. So, we can use these as the futility cutoffs.

alternative is true

```
df4 = df4_maker()

# df4 = df4 %>%
#   mutate(n2 = list(12:13),
#          n3 = list(18:20)) %>%
#   unnest(n2) %>%
#   unnest(n3)
```

```
prior4 = function(df= df4, null = 0.25, stage1n = 20, stage2n = 40, stage3n = 60, stage4n = 71) {
  # res = vector(mode = "list", length = 2)
  # for (i in 1:2) {
  res4 = df %>%
    group_by(n1_response, n2_response, n3_response, n4_response) %>%
    mutate(pE1 = qbeta(null, (0.5+n1_response), (1.5+stage1n-n1_response)),
           pE2 = qbeta(null, (0.5+n1_response+n2_response), (1.5+stage2n-n1_response-n2_response)),
           pE3 = qbeta(null, (0.5+n1_response + n2_response + n3_response), (1.5+stage3n-n1_response-n2
           pE4 = qbeta(null, (0.5+n1_response + n2_response + n3_response + n4_response), (1.5+stage4n-
           # pS = dbeta(null+pull(df, x), 25, 75))
    # filter(n1_response > n1-1,
    #        (n1_response + n2_response) > n2-1,
    #        (n1_response + n2_response + n3_response) > n3-1,
    #        (n1_response + n2_response + n3_response + n4_response) > n4-1)
  return(res4)
}
```

```
map_df4 = df4 %>%
  mutate(data = map(data, ~prior4(df = ., null = 0.4, stage1n = 20, stage2n = 40, stage3n = 60, stage4n
```

```
dist4 = map_df4 %>%
  unnest() %>%
  mutate(pS = qbeta(.25, shape1 = 25, shape2 = 75)) %>%
  # filter((n1_response + n2_response) > n2-1,
  #        (n1_response + n2_response + n3_response) > n3-1) %>%
  group_by(x, n1_response, n2_response, n3_response, n4_response) %>%
  filter(pE1 > pS + x,
         pE2 > pS + x,
         pE3 > pS + x,
         pE4 > pS + x)
# group_by(n2, n3) %>%
# nest()
```

```
dist4
```

```
## # A tibble: 326,056 x 12
## # Groups:   x, n1_response, n2_response, n3_response, n4_response [326,056]
##        x n1_response n2_response n3_response n4_response    n1    n4   pE1   pE2
##    <dbl>       <int>       <int>       <int>       <int> <dbl> <dbl> <dbl> <dbl>
##  1  0.15           9           7           8           4     7    24 0.403 0.372
##  2  0.15           9           7           8           5     7    24 0.403 0.372
##  3  0.15           9           7           8           6     7    24 0.403 0.372
##  4  0.15           9           7           8           7     7    24 0.403 0.372
##  5  0.15           9           7           8           8     7    24 0.403 0.372
##  6  0.15           9           7           8           9     7    24 0.403 0.372
##  7  0.15           9           7           8          10     7    24 0.403 0.372
##  8  0.15           9           7           8          11     7    24 0.403 0.372
##  9  0.15           9           7           9           3     7    24 0.403 0.372
## 10  0.15           9           7           9           4     7    24 0.403 0.372
## # ... with 326,046 more rows, and 3 more variables: pE3 <dbl>, pE4 <dbl>,
## #   pS <dbl>
```

As can be seen from the list of potential outcomes that satisfy the interim analyses,

```
# gg4 = dist4 %>%
# #   group_by(x) %>%
# # # filter(pE1>pS,
# # #        pE2>pS,
# # #        pE3>pS,
# # #        pE4>pS) %>%
#   # summarize(n = n(),
#   #           alpha1 = 1 - sum(pE1 > pS)/n,
#   #           alpha2 = 1 - sum(pE2 > pS)/n,
#   #           alpha3 = 1 - sum(pE3 > pS)/n,
#   #           alpha4 = 1 - sum(pE4 > pS)/n) %>%
#   # filter(alpha1 < 0.2,
#   #        alpha2 < 0.2,
#   #        alpha3 < 0.2,
#   #        alpha4 < 0.2) %>%
#   ggplot(aes(x = x)) +
#   geom_line(aes(y = alpha1), color = "red") +
#   geom_line(aes(y = alpha2), color = "orange") +
#   geom_line(aes(y = alpha3), color = "yellow") +
```

```
#   geom_line(aes(y = alpha4), color = "green") +
#   facet_grid(n2~n3)
```

```
# head(map_df4) %>%
#   unnest()
#
# dist4 %>%
#   filter(alpha4 > alpha1)
#
# rbeta(10, 6.5, 14.5)
# qbeta(.25, 5.5, 14.5)
# qbeta(.25, 25, 75)
#
# gg4
```

```
set.seed(719)
n_sim = 10e5
x = 0.15
dat = tibble(means = c(sum(rbeta(n_sim, shape1 = 10.5, shape2 = 11.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 9.5, shape2 = 12.5) > (rbeta(n_sim, shape1 = 25, shape2
                       sum(rbeta(n_sim, shape1 = 18.5, shape2 = 23.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 17.5, shape2 = 24.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 26.5, shape2 = 35.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 25.5, shape2 = 36.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 30.5, shape2 = 41.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 29.5, shape2 = 42.5) > (rbeta(n_sim, shape1 = 25, shape

dat
```

```
## # A tibble: 8 x 1
##   means
##   <dbl>
## 1 0.749
## 2 0.604
## 3 0.678
## 4 0.573
## 5 0.641
## 6 0.559
## 7 0.629
## 8 0.553
```

Under the null, an alpha of 0.61 with a delta of 0.15 can be used to mirror adaptive designs "go-no-go" described in lecture using the futility decision rules described above $(S_{20} = 10, S_{40} = 18, S_{60} = 26, S_{71} = 30).

```
set.seed(719)
n_sim = 10e5
x = 0.15
dat = tibble(means = c(sum(rbeta(n_sim, shape1 = 9.5, shape2 = 12.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 8.5, shape2 = 13.5) > (rbeta(n_sim, shape1 = 25, shape2
                       sum(rbeta(n_sim, shape1 = 16.5, shape2 = 25.5) > (rbeta(n_sim, shape1 = 25, shape
                       sum(rbeta(n_sim, shape1 = 15.5, shape2 = 26.5) > (rbeta(n_sim, shape1 = 25, shape
```

```
                      sum(rbeta(n_sim, shape1 = 24.5, shape2 = 37.5) > (rbeta(n_sim, shape1 = 25, shape
                      sum(rbeta(n_sim, shape1 = 23.5, shape2 = 38.5) > (rbeta(n_sim, shape1 = 25, shape
                      sum(rbeta(n_sim, shape1 = 28.5, shape2 = 43.5) > (rbeta(n_sim, shape1 = 25, shape
                      sum(rbeta(n_sim, shape1 = 27.5, shape2 = 44.5) > (rbeta(n_sim, shape1 = 25, shape

dat
```

```
## # A tibble: 8 x 1
##   means
##   <dbl>
## 1 0.605
## 2 0.442
## 3 0.464
## 4 0.355
## 5 0.473
## 6 0.389
## 7 0.478
## 8 0.399
```

Under the alternative, an alpha of 0.45 and a delta of 0.15 can satisfy the adaptive design "go-no-go" described in lecture.

```
res7 = vector(mode = "list", length = 72)
sum_alphas = vector(mode = "list", length = 72)
n_sim = 10e2
for (i in 1:72) {
  for (j in 1:71) {
    sum_alphas[[c(i, j)]] = sum(rbeta(n_sim, shape1 = .5 + i - 1, shape2 = 1.5 + j - i - 1) > rbeta(n_si
    res7[[c(i, j)]] = bind_cols("alpha" = sum_alphas[[c(i, j)]], "i" = i, "j" = j)
  }
  res7[[i]] = map(res7[[i]], ~bind_rows(.))
}
```

```
res7 = bind_rows(res7)
```

```
res7 %>%
  tibble() %>%
  group_by(i, j) %>%
  summarize(n = n(),
            alpha = alpha) %>%
  filter(alpha >= 0.9) %>%
  group_by(i, j) %>%
  summarize(nj = n(),
            nm = nj/n,
            alpha = alpha) %>%
  arrange(desc(j), i)
```

```
## # A tibble: 1,638 x 5
## # Groups:   i [70]
##        i     j    nj    nm alpha
##    <int> <int> <int> <dbl> <dbl>
##  1    25    71     1     1 0.905
```

```
##  2    26    71     1     1 0.936
##  3    27    71     1     1 0.959
##  4    28    71     1     1 0.975
##  5    29    71     1     1 0.98
##  6    30    71     1     1 0.99
##  7    31    71     1     1 0.991
##  8    32    71     1     1 0.997
##  9    33    71     1     1 0.999
## 10    34    71     1     1 1
## # ... with 1,628 more rows
```

The alpha required to achieve the beta prior based on the available data can be calculated.

A response rate of at least 25 is required to achieve the interim analysis $Pr(S_{71} \geq 24|S_m = s)$.

```
set.seed(1)
n_sim = 10e4
x = seq(0.1, 0.19, by = 0.01)
outputx = vector(mode = "list", length = 20)
res = vector(mode = "list", length = 20)
for (i in 1:20) {
  for (j in 1:10) {
    outputx[[c(i, j)]] = sum(rbeta(n_sim, shape1 = .5+i, shape2 =1.5+20-i) > (rbeta(n_sim, shape1 = 25,
    res[[c(i, j)]] = bind_cols("alpha" = outputx[[c(i, j)]], "i" = i, "j" = j)
  }
  res[[i]] = map(res[[i]], ~bind_rows(.))
}
# outputx = bind_rows(outputx)
# outputx
res = bind_rows(res)
```

```
res %>%
  filter(alpha >= 0.9)
```

```
## # A tibble: 91 x 3
##    alpha     i     j
##    <dbl> <int> <int>
##  1 0.935    11     1
##  2 0.923    11     2
##  3 0.910    11     3
##  4 0.972    12     1
##  5 0.966    12     2
##  6 0.959    12     3
##  7 0.950    12     4
##  8 0.940    12     5
##  9 0.931    12     6
## 10 0.917    12     7
## # ... with 81 more rows
```

Question 5

I explored the beta distribution using beta distribution functions in R.

I simulated values of response rates and compared them to those response rates in a posterior distribution given by ~beta(25, 75).

After this assignment I learned a way to plan an interim analysis study design using Bayesian formulation.

I updated the beta function for each potential outcome and compared it to the probability of the posterior distribution.

Burn in and interim analyses can improve the expected number of patients enrolled in the study for drugs that can be assumed to have no effect.