# Phase II Clinical Design Using Multinomial Distribution

*Rui Qin, Yalin Zhu, Bo Gao*

*June 23, 2016*

## Contents

## 1 Two general principles of hypothesis testing

This section reviews some key principles that provide a foundation for multiple tests. It begins with two general principles, known as the principles of union-intersection testing (UIT) and intersection-union testing (IUT), that define the underlying testing problem.

### 1.1 Union-intersection testing (UIT)

Within the union-intersection framework, one rejects the global hypothesis of no effect if there is evidence of a positive effect with respect to at least one individual objective. To provide a mathematical definition, let $H_1, \ldots, H_m$ denote the hypotheses corresponding to the multiple objectives. The hypotheses are tested against the alternative hypotheses $K_1, \ldots, K_m$. The global null hypothesis $H_I$ , defined as the intersection of the hypotheses, is tested versus the union of the alternative hypotheses ($K_U$):

$$H_I : \bigcap_{i=1}^{m} H_i \quad versus \quad K_U : \bigcup_{i=1}^{m} K_i.$$

### 1.2 Intersection-union testing (IUT)

Intersection-union testing arises naturally in studies when a significant outcome with respect to two or more objectives is required in order to declare the study successful. For example, new drugs/therapies for the treatment of *Skin Cancer* are required to demonstrate their effects on both superiority (eg: partial response) and futility (eg: early progression). In other words, the intersection-union method involves testing the union of the hypotheses ($H_U$) against the intersection of the alternative hypotheses ($K_I$):

$$H_U : \bigcup_{i=1}^{m} H_i \quad versus \quad K_I : \bigcap_{i=1}^{m} K_i.$$

We differentiate between objective response and early progression. Let $p_1$ and $p_2$ be the probabilities of response and early disease progression, respectively. Note that $p_1 + p_2 \leq 1$. Then the number of objective follow the trinomial distribution $Tri(p_1, p_2, 1 - (p_1 + p_2))$. For most phase II window studies, there is interest in proceeding with further evaluation of the agent if the response rate is sufficiently high and the early progression rate is sufficiently low. Thus, the study is designed to test

$$H_U : p_1 \leq p_{01} \; or \; p_2 \leq p_{02} \quad versus \quad K_I : p_1 \geq p_{11} \; and \; p_2 \geq p_{12},$$

which belongs to IUT.

For an one-stage design, let $N$ denote a fixed sample size, $S$ denote the number of partial response and $T$ denote the number of early progressions. Then the rejection region of the null hypothesis $H_U$ can be denoted by

$$S \geq s \quad and \quad T \leq t,$$

where $s + t \leq N$. The acceptance region of the null can be denoted by

$$S < s' \quad or \quad T > t'.$$

For a two-stage design, let $N_1$ denote a fixed sample size at the first stage, $S$ denote the number of partial response and $T$ denote the number of early progressions. Then at the first stage, the rejection region of the null hypothesis $H_U$ can be denoted by

$$S \geq s_1 \quad and \quad T \leq t_1,$$

where $s_1 + t_1 \leq N_1$. The acceptance region of the null can be denoted by

$$S \leq s_1'(< s_1) \quad or \quad T \geq t_1'(> t_1).$$

Stop the trial the second stage if the number of corresponding patients satisfies the rejection or acceptance condition, enroll additional $N_2$ patients and continue to the second stage otherwise. At the second stage, the rejection region of the null hypothesis $H_U$ can be denoted by

$$S \geq s_2 \quad and \quad T \leq t_2,$$

where $s_2 + t_2 \leq N_1 + N_2$. We can also consider the acceptance region can be denoted by

$$S \geq s_2' \quad or \quad T \leq t_2',$$

althogh the original paper does not mention the acceptance region for futility.

## 2 The power function for multinomial design using IUT

### 2.1 One-stage multinomial design

#### 2.1.1 Power function validation

```
# Test whole data
s <- c(6, 8, 8, 9, 9, 8, 7, 12, 14, 15, 14, 15, 14, 17, 20, 21, 21, 19, 23,
    25, 24, 24, 24, 29, 27, 27, 29, 24)
t <- c(19, 24, 22, 21, 16, 10, 5, 23, 25, 22, 16, 12, 7, 22, 22, 18, 13, 7,
    19, 17, 12, 8, 13, 12, 7, 9, 6, 4)
n <- c(25, 36, 39, 45, 44, 39, 33, 35, 44, 47, 44, 46, 42, 39, 47, 49, 49, 44,
    42, 47, 45, 45, 37, 45, 42, 36, 39, 28)
p0.s <- unlist(mapply(rep, 1:7, 7:1)) * 0.1
p0.t <- unlist(mapply(seq, 9:3, 3)) * 0.1
p1.s <- p0.s + 0.2
p1.t <- p0.t - 0.2

sig.s1.IUT <- pmax(mapply(IUT.power, method = "s1", s2.rej = s, t2.rej = t,
    n = n, p.s = p0.s, p.t = 0, USE.NAMES = F), mapply(IUT.power, method = "s1",
    s2.rej = s, t2.rej = t, n = n, p.s = 1 - p0.t, p.t = p0.t, USE.NAMES = F))
power.s1.IUT <- mapply(IUT.power, method = "s1", s2.rej = s, t2.rej = t, n = n,
    p.s = p1.s, p.t = p1.t, USE.NAMES = F)
result.s1.IUT <- data.frame(p0.s, p0.t, s.rej = s, t.rej = t, N = n, Error = sig.s1.IUT,
    Power = power.s1.IUT)
print(result.s1.IUT, digits = 3)
```

```
##    p0.s p0.t s.rej t.rej  N  Error Power
## 1   0.1  0.9     6    19 25 0.0334 0.807
## 2   0.1  0.8     8    24 36 0.0424 0.804
## 3   0.1  0.7     8    22 39 0.0500 0.807
## 4   0.1  0.6     9    21 45 0.0483 0.833
## 5   0.1  0.5     9    16 44 0.0481 0.827
## 6   0.1  0.4     8    10 39 0.0450 0.814
## 7   0.1  0.3     7     5 33 0.0417 0.818
## 8   0.2  0.8    12    23 35 0.0344 0.805
## 9   0.2  0.7    14    25 44 0.0437 0.824
## 10  0.2  0.6    15    22 47 0.0460 0.818
## 11  0.2  0.5    14    16 44 0.0481 0.802
## 12  0.2  0.4    15    12 46 0.0354 0.801
## 13  0.2  0.3    14     7 42 0.0378 0.814
## 14  0.3  0.7    17    22 39 0.0500 0.832
## 15  0.3  0.6    20    22 47 0.0460 0.821
## 16  0.3  0.5    21    18 49 0.0427 0.811
## 17  0.3  0.4    21    13 49 0.0382 0.815
## 18  0.3  0.3    19     7 44 0.0437 0.810
## 19  0.4  0.6    23    19 42 0.0375 0.803
## 20  0.4  0.5    25    17 47 0.0460 0.808
## 21  0.4  0.4    24    12 45 0.0483 0.809
## 22  0.4  0.3    24     8 45 0.0483 0.840
## 23  0.5  0.5    24    13 37 0.0494 0.807
## 24  0.5  0.4    29    12 45 0.0446 0.808
## 25  0.5  0.3    27     7 42 0.0442 0.813
## 26  0.6  0.4    27     9 36 0.0449 0.832
## 27  0.6  0.3    29     6 39 0.0450 0.823
## 28  0.7  0.3    24     4 28 0.0474 0.858
```

### 2.1.2 Find the rejection boundary for pCR and ePD based on pre-specified type I error rate and power level.

```
# set the intervals as +-1
IUT.design(method = "s1", s2.rej = 18, t2.rej = 12, n = 80, s2.rej.delta = 1,
    t2.rej.delta = 1, n.delta = 1, p0.s = 0.15, p0.t = 0.25, p1.s = 0.3, p1.t = 0.1)
```

```
##    p0.s p0.t p1.s p1.t s.rej t.rej  N Error Power
## 17 0.15 0.25  0.3  0.1    18    13 80 0.048 0.924
```

```
##    user  system elapsed
##    0.48    0.00    0.49
```

```
# defaut do not set the intervals
IUT.design(method = "s1", s2.rej = 18, t2.rej = 12, n = 80, p0.s = 0.15, p0.t = 0.25,
    p1.s = 0.3, p1.t = 0.1)
```

```
##   p0.s p0.t p1.s p1.t s.rej t.rej  N Error Power
## 1 0.15 0.25  0.3  0.1    18    12 80 0.048 0.899
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
# output all valid outcome
IUT.design(method = "s1", s2.rej = 18, t2.rej = 12, n = 80, s2.rej.delta = 1,
    t2.rej.delta = 1, n.delta = 1, p0.s = 0.15, p0.t = 0.25, p1.s = 0.3, p1.t = 0.1,
    output.all = T)
```

```
##    p0.s p0.t p1.s p1.t s.rej t.rej  N  Error Power
## 2  0.15 0.25  0.3  0.1    18    11 79 0.0430 0.857
## 3  0.15 0.25  0.3  0.1    19    11 79 0.0228 0.825
## 5  0.15 0.25  0.3  0.1    18    12 79 0.0430 0.896
## 6  0.15 0.25  0.3  0.1    19    12 79 0.0254 0.862
## 8  0.15 0.25  0.3  0.1    18    13 79 0.0477 0.919
## 9  0.15 0.25  0.3  0.1    19    13 79 0.0477 0.882
## 11 0.15 0.25  0.3  0.1    18    11 80 0.0480 0.857
## 12 0.15 0.25  0.3  0.1    19    11 80 0.0259 0.829
## 14 0.15 0.25  0.3  0.1    18    12 80 0.0480 0.899
## 15 0.15 0.25  0.3  0.1    19    12 80 0.0259 0.869
## 17 0.15 0.25  0.3  0.1    18    13 80 0.0480 0.924
## 18 0.15 0.25  0.3  0.1    19    13 80 0.0421 0.891
## 21 0.15 0.25  0.3  0.1    19    11 81 0.0292 0.831
## 24 0.15 0.25  0.3  0.1    19    12 81 0.0292 0.874
## 27 0.15 0.25  0.3  0.1    19    13 81 0.0371 0.899
```

```
##    user  system elapsed
##    0.47    0.02    0.48
```

## 2.2 Two-stage multinomial design

### 2.2.1 Power function validation

```
## Test whole data
s1 <- c(4, 6, 6, 6, 7, 6, 6, 8, 9, 10, 10, 10, 10, 11, 13, 13, 14, 12, 15, 16,
    16, 16, 16, 17, 16, 16, 17, 14)
s2 <- c(0, 0, 1, 0, 2, 2, 1, 1, 5, 5, 3, 4, 4, 4, 7, 6, 8, 3, 9, 9, 8, 9, 9,
    8, 10, 10, 11, 10)
t1 <- c(9, 8, 7, 6, 5, 2, 0, 9, 10, 7, 4, 3, 1, 9, 8, 6, 4, 1, 6, 6, 3, 2, 3,
    3, 1, 2, 0, 0)
t2 <- c(13, 15, 15, 13, 11, 9, 5, 16, 16, 15, 12, 8, 6, 16, 14, 13, 9, 8, 12,
    12, 9, 7, 10, 11, 6, 8, 5, 4)
a1 <- c(6, 7, 8, 9, 8, 8, 7, 11, 14, 14, 14, 14, 12, 17, 20, 20, 20, 18, 22,
    25, 24, 23, 24, 27, 27, 25, 27, 24)
a2 <- c(18, 22, 22, 20, 15, 10, 5, 21, 25, 21, 16, 12, 6, 21, 22, 17, 13, 7,
    19, 17, 12, 7, 13, 11, 7, 8, 6, 4)
n1 <- c(13, 18, 20, 22, 21, 20, 17, 17, 22, 23, 22, 23, 20, 20, 24, 24, 24,
    21, 21, 24, 23, 22, 19, 21, 21, 18, 19, 14)
n2 <- c(11, 15, 19, 21, 21, 19, 16, 15, 22, 22, 22, 21, 17, 18, 23, 23, 24,
    20, 20, 23, 22, 21, 18, 21, 21, 15, 17, 14)
# show the results of IUT for two-stage with early stop for both superority
# and futility
sig.s2.sf.IUT <- pmax(mapply(IUT.power, method = "s2.sf", s1.rej = s1, s1.acc = s2,
    t1.rej = t1, t1.acc = t2, s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p0.s,
    p.t = 0, USE.NAMES = F), mapply(IUT.power, method = "s2.sf", s1.rej = s1,
    s1.acc = s2, t1.rej = t1, t1.acc = t2, s2.rej = a1, t2.rej = a2, n1 = n1,
    n2 = n2, p.s = 1 - p0.t, p.t = p0.t, USE.NAMES = F))
power.s2.sf.IUT <- mapply(IUT.power, method = "s2.sf", s1.rej = s1, s1.acc = s2,
    t1.rej = t1, t1.acc = t2, s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p1.s,
    p.t = p1.t, USE.NAMES = F)
result.s2.sf.IUT <- data.frame(p0.s, p0.t, s1.rej = s1, t1.rej = t1, s1.acc = s2,
    t1.acc = t2, s2.rej = a1, t2.rej = a2, N1 = n1, N2 = n2, Error = sig.s2.sf.IUT,
    Power = power.s2.sf.IUT)
print(result.s2.sf.IUT, digits = 3)
```

```
##    p0.s p0.t s1.rej t1.rej s1.acc t1.acc s2.rej t2.rej N1 N2  Error Power
## 1   0.1  0.9      4      9      0     13      6     18 13 11 0.0486 0.800
## 2   0.1  0.8      6      8      0     15      7     22 18 15 0.0497 0.801
## 3   0.1  0.7      6      7      1     15      8     22 20 19 0.0495 0.803
## 4   0.1  0.6      6      6      0     13      9     20 22 21 0.0483 0.800
## 5   0.1  0.5      7      5      2     11      8     15 21 21 0.0494 0.804
## 6   0.1  0.4      6      2      2      9      8     10 20 19 0.0460 0.801
## 7   0.1  0.3      6      0      1      5      7      5 17 16 0.0415 0.810
## 8   0.2  0.8      8      9      1     16     11     21 17 15 0.0448 0.800
## 9   0.2  0.7      9     10      5     16     14     25 22 22 0.0487 0.803
## 10  0.2  0.6     10      7      5     15     14     21 23 22 0.0491 0.802
## 11  0.2  0.5     10      4      3     12     14     16 22 22 0.0482 0.800
## 12  0.2  0.4     10      3      4      8     14     12 23 21 0.0495 0.806
## 13  0.2  0.3     10      1      4      6     12      6 20 17 0.0490 0.801
## 14  0.3  0.7     11      9      4     16     17     21 20 18 0.0466 0.801
## 15  0.3  0.6     13      8      7     14     20     22 24 23 0.0485 0.806
## 16  0.3  0.5     13      6      6     13     20     17 24 23 0.0497 0.801
## 17  0.3  0.4     14      4      8      9     20     13 24 24 0.0500 0.809
## 18  0.3  0.3     12      1      3      8     18      7 21 20 0.0479 0.800
## 19  0.4  0.6     15      6      9     12     22     19 21 20 0.0491 0.812
## 20  0.4  0.5     16      6      9     12     25     17 24 23 0.0477 0.804
## 21  0.4  0.4     16      3      8      9     24     12 23 22 0.0490 0.803
## 22  0.4  0.3     16      2      9      7     23      7 22 21 0.0483 0.800
## 23  0.5  0.5     16      3      9     10     24     13 19 18 0.0492 0.804
## 24  0.5  0.4     17      3      8     11     27     11 21 21 0.0497 0.800
## 25  0.5  0.3     16      1     10      6     27      7 21 21 0.0494 0.809
## 26  0.6  0.4     16      2     10      8     25      8 18 15 0.0473 0.803
## 27  0.6  0.3     17      0     11      5     27      6 19 17 0.0497 0.806
## 28  0.7  0.3     14      0     10      4     24      4 14 14 0.0487 0.852
```

#### 2.2.2 Find the rejection boundary for pCR and ePD based on pre-specified type I error rate and power level.

```
IUT.design(method = "s2.sf", s1.rej = 10, t1.rej = 3, s1.acc = 8, t1.acc = 5,
    s2.rej = 18, t2.rej = 12, n1 = 41, n2 = 41, s1.rej.delta = 1, t1.rej.delta = 1,
    s2.rej.delta = 1, t2.rej.delta = 1, p0.s = 0.15, p0.t = 0.25, p1.s = 0.3,
    p1.t = 0.1)
```

```
##    p0.s p0.t p1.s p1.t s1.rej t1.rej s1.acc t1.acc s2.rej t2.rej N1 N2
## 81 0.15 0.25  0.3  0.1     11      4      8      5     19     13 41 41
##     Error Power
## 81 0.0476 0.875
```

```
##    user  system elapsed
##   38.47    0.01   38.48
```

### 2.3 Two-stage multinomial design with futlity only

#### 2.3.1 Power function validation

```
# show the results of IUT for two-stage with early stop for only futility
sig.s2.f.IUT <- pmax(mapply(IUT.power, method = "s2.f", s1.acc = s2, t1.acc = t2,
    s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p0.s, p.t = 0, USE.NAMES = F),
    mapply(IUT.power, method = "s2.f", s1.acc = s2, t1.acc = t2, s2.rej = a1,
        t2.rej = a2, n1 = n1, n2 = n2, p.s = 1 - p0.t, p.t = p0.t, USE.NAMES = F))

nul <- mapply(IUT.power, method = "s2.f", s1.acc = s2, t1.acc = t2, s2.rej = a1,
    t2.rej = a2, n1 = n1, n2 = n2, p.s = p0.s, p.t = p0.t, output.all = TRUE,
    USE.NAMES = F)


alt <- mapply(IUT.power, method = "s2.f", s1.acc = s2, t1.acc = t2, s2.rej = a1,
    t2.rej = a2, n1 = n1, n2 = n2, p.s = p1.s, p.t = p1.t, output.all = TRUE,
    USE.NAMES = F)
```

```
result.s2.f.IUT <- data.frame(p0.s, p0.t, s1.acc = s2, t1.acc = t2, s2.rej = a1,
    t2.rej = a2, N1 = n1, N2 = n2, Error = sig.s2.f.IUT, PET.nul = nul[2, ],
    EN.nul = nul[3, ], Power = alt[1, ], PET.alt = alt[2, ], EN.alt = alt[3,
        ])
print(result.s2.f.IUT, digits = 3)
```

```
##    p0.s p0.t s1.acc t1.acc s2.rej t2.rej N1 N2  Error PET.nul EN.nul Power
## 1   0.1  0.9      0     13      6     18 13 11 0.0276   0.254   21.2 0.770
## 2   0.1  0.8      0     15      7     22 18 15 0.0497   0.521   25.2 0.801
## 3   0.1  0.7      1     15      8     22 20 19 0.0494   0.563   28.3 0.803
## 4   0.1  0.6      0     13      9     20 22 21 0.0479   0.640   29.6 0.800
## 5   0.1  0.5      2     11      8     15 21 21 0.0489   0.770   25.8 0.801
## 6   0.1  0.4      2      9      8     10 20 19 0.0449   0.766   24.4 0.800
## 7   0.1  0.3      1      5      7      5 17 16 0.0407   0.765   20.8 0.810
## 8   0.2  0.8      1     16     11     21 17 15 0.0411   0.118   30.2 0.795
## 9   0.2  0.7      5     16     14     25 22 22 0.0427   0.749   27.5 0.797
## 10  0.2  0.6      5     15     14     21 23 22 0.0480   0.725   29.1 0.801
## 11  0.2  0.5      3     12     14     16 22 22 0.0477   0.535   32.2 0.800
## 12  0.2  0.4      4      8     14     12 23 21 0.0480   0.832   26.5 0.805
## 13  0.2  0.3      4      6     12      6 20 17 0.0487   0.795   23.5 0.801
## 14  0.3  0.7      4     16     17     21 20 18 0.0387   0.238   33.7 0.791
## 15  0.3  0.6      7     14     20     22 24 23 0.0442   0.711   30.6 0.803
## 16  0.3  0.5      6     13     20     17 24 23 0.0454   0.536   34.7 0.797
## 17  0.3  0.4      8      9     20     13 24 24 0.0493   0.836   27.9 0.808
## 18  0.3  0.3      3      8     18      7 21 20 0.0458   0.314   34.7 0.798
## 19  0.4  0.6      9     12     22     19 21 20 0.0483   0.691   27.2 0.811
## 20  0.4  0.5      9     12     25     17 24 23 0.0452   0.640   32.3 0.801
## 21  0.4  0.4      8      9     24     12 23 22 0.0480   0.664   30.4 0.802
## 22  0.4  0.3      9      7     23      7 22 21 0.0479   0.726   27.8 0.799
## 23  0.5  0.5      9     10     24     13 19 18 0.0487   0.500   28.0 0.803
## 24  0.5  0.4      8     11     27     11 21 21 0.0449   0.245   36.9 0.798
## 25  0.5  0.3     10      6     27      7 21 21 0.0435   0.715   27.0 0.806
## 26  0.6  0.4     10      8     25      8 18 15 0.0443   0.437   26.5 0.800
## 27  0.6  0.3     11      5     27      6 19 17 0.0494   0.748   23.3 0.805
## 28  0.7  0.3     10      4     24      4 14 14 0.0459   0.645   19.0 0.850
##    PET.alt EN.alt
## 1  0.00969   23.9
## 2  0.03319   32.5
## 3  0.02570   38.5
## 4  0.05530   41.8
## 5  0.04945   41.0
## 6  0.04372   38.2
## 7  0.04010   32.4
## 8  0.00209   32.0
## 9  0.07689   42.3
## 10 0.05896   43.7
## 11 0.02011   43.6
## 12 0.08480   42.2
## 13 0.06002   36.0
## 14 0.00591   37.9
## 15 0.06555   45.5
## 16 0.02004   46.5
## 17 0.09835   45.6
## 18 0.00134   41.0
## 19 0.08492   39.3
## 20 0.04144   46.0
## 21 0.03575   44.2
## 22 0.05767   41.8
## 23 0.03255   36.4
## 24 0.00298   41.9
## 25 0.03692   41.2
## 26 0.01628   32.8
## 27 0.04888   35.2
## 28 0.04413   27.4
```

#### 2.3.2 Find the rejection boundary for pCR and ePD based on pre-specified type I error rate and power level.

```
IUT.design(method = "s2.f", s1.acc = 7, t1.acc = 5, s2.rej = 17, t2.rej = 13,
    n1 = 41, n2 = 41, s2.rej.delta = 1, t2.rej.delta = 1, p0.s = 0.15, p0.t = 0.25,
    p1.s = 0.3, p1.t = 0.1)
```

```
##   p0.s p0.t p1.s p1.t s1.acc t1.acc s2.rej t2.rej N1 N2  Error Power
## 9 0.15 0.25  0.3  0.1      7      5     18     14 41 41 0.0499 0.576
```

```
##    user  system elapsed
##   16.71    0.00   16.74
```

## 3 The power function for UIT

### 3.1 One-stage multinomial design

#### 3.1.1 Power function validation

```
sig.s1.UIT <- mapply(UIT.power, method = "s1", s2.rej = s, t2.rej = t, n = n,
    p.s = p0.s, p.t = p0.t, USE.NAMES = F)
power.s1.UIT <- mapply(UIT.power, method = "s1", s2.rej = s, t2.rej = t, n = n,
    p.s = p1.s, p.t = p1.t, USE.NAMES = F)
result.s1.UIT <- data.frame(p0.s, p0.t, s.rej = s, t.rej = t, N = n, Error = sig.s1.UIT,
    Power = power.s1.UIT)
print(result.s1.UIT, digits = 3)
```

```
##    p0.s p0.t s.rej t.rej  N  Error Power
## 1   0.1  0.9     6    19 25 0.0334 0.807
## 2   0.1  0.8     8    24 36 0.0450 0.884
## 3   0.1  0.7     8    22 39 0.0579 0.929
## 4   0.1  0.6     9    21 45 0.0569 0.959
## 5   0.1  0.5     9    16 44 0.0560 0.962
## 6   0.1  0.4     8    10 39 0.0560 0.963
## 7   0.1  0.3     7     5 33 0.0538 0.968
## 8   0.2  0.8    12    23 35 0.0344 0.805
## 9   0.2  0.7    14    25 44 0.0515 0.900
## 10  0.2  0.6    15    22 47 0.0555 0.926
## 11  0.2  0.5    14    16 44 0.0619 0.936
## 12  0.2  0.4    15    12 46 0.0462 0.951
## 13  0.2  0.3    14     7 42 0.0486 0.975
## 14  0.3  0.7    17    22 39 0.0500 0.832
## 15  0.3  0.6    20    22 47 0.0541 0.898
## 16  0.3  0.5    21    18 49 0.0533 0.925
## 17  0.3  0.4    21    13 49 0.0491 0.949
## 18  0.3  0.3    19     7 44 0.0439 0.968
## 19  0.4  0.6    23    19 42 0.0375 0.803
## 20  0.4  0.5    25    17 47 0.0486 0.892
## 21  0.4  0.4    24    12 45 0.0587 0.933
## 22  0.4  0.3    24     8 45 0.0642 0.980
## 23  0.5  0.5    24    13 37 0.0494 0.807
## 24  0.5  0.4    29    12 45 0.0496 0.914
## 25  0.5  0.3    27     7 42 0.0510 0.961
## 26  0.6  0.4    27     9 36 0.0449 0.832
## 27  0.6  0.3    29     6 39 0.0389 0.929
## 28  0.7  0.3    24     4 28 0.0474 0.858
```

#### 3.1.2 Find the rejection boundary for pCR and ePD based on pre-specified type I error rate and power level.

```
# set the intervals as +-1
UIT.design(method = "s1", s2.rej = 18, t2.rej = 12, n = 80, s2.rej.delta = 1,
    t2.rej.delta = 1, n.delta = 1, p0.s = 0.15, p0.t = 0.25, p1.s = 0.3, p1.t = 0.1)
```

```
##    p0.s p0.t p1.s p1.t s.rej t.rej  N  Error Power
## 23 0.15 0.25  0.3  0.1    18    12 81 0.0467 0.991
```

```
##    user  system elapsed
##    0.49    0.00    0.51
```

```
# defaut do not set the intervals
UIT.design(method = "s1", s2.rej = 18, t2.rej = 12, n = 80, p0.s = 0.15, p0.t = 0.25,
    p1.s = 0.3, p1.t = 0.1, output.all = T)
```

```
##   p0.s p0.t p1.s p1.t s.rej t.rej  N  Error Power
## 1 0.15 0.25  0.3  0.1    18    12 80 0.0462  0.99
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
# output all valid outcome
UIT.design(method = "s1", s2.rej = 18, t2.rej = 12, n = 80, s2.rej.delta = 1,
    t2.rej.delta = 1, n.delta = 1, p0.s = 0.15, p0.t = 0.25, p1.s = 0.3, p1.t = 0.1,
    output.all = T)
```

```
##     p0.s p0.t p1.s p1.t s.rej t.rej  N  Error Power
## 2  0.15 0.25  0.3  0.1    18    11 79 0.0342 0.983
## 3  0.15 0.25  0.3  0.1    19    11 79 0.0232 0.976
## 5  0.15 0.25  0.3  0.1    18    12 79 0.0464 0.990
## 6  0.15 0.25  0.3  0.1    19    12 79 0.0358 0.986
## 11 0.15 0.25  0.3  0.1    18    11 80 0.0356 0.984
## 12 0.15 0.25  0.3  0.1    19    11 80 0.0232 0.976
## 14 0.15 0.25  0.3  0.1    18    12 80 0.0462 0.990
## 15 0.15 0.25  0.3  0.1    19    12 80 0.0342 0.986
## 20 0.15 0.25  0.3  0.1    18    11 81 0.0374 0.985
## 21 0.15 0.25  0.3  0.1    19    11 81 0.0237 0.977
## 23 0.15 0.25  0.3  0.1    18    12 81 0.0467 0.991
## 24 0.15 0.25  0.3  0.1    19    12 81 0.0333 0.986
```

```
##    user  system elapsed
##    0.47    0.00    0.47
```

## 3.2 Two-stage multinomial design

### 3.2.1 Power function validation

```
sig.s2.sf.UIT <- mapply(UIT.power, method = "s2.sf", s1.rej = s1, s1.acc = s2,
    t1.rej = t1, t1.acc = t2, s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p0.s,
    p.t = p0.t, USE.NAMES = F)
power.s2.sf.UIT <- mapply(UIT.power, method = "s2.sf", s1.rej = s1, s1.acc = s2,
    t1.rej = t1, t1.acc = t2, s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p1.s,
    p.t = p1.t, USE.NAMES = F)
result.s2.sf.UIT <- data.frame(p0.s, p0.t, s1.rej = s1, t1.rej = t1, s1.acc = s2,
    t1.acc = t2, s2.rej = a1, t2.rej = a2, N1 = n1, N2 = n2, Error = sig.s2.sf.UIT,
    Power = power.s2.sf.UIT)
print(result.s2.sf.UIT, digits = 3)
```

```
##    p0.s p0.t s1.rej t1.rej s1.acc t1.acc s2.rej t2.rej N1 N2  Error Power
## 1   0.1  0.9      4      9      0     13      6     18 13 11 0.0486 0.800
## 2   0.1  0.8      6      8      0     15      7     22 18 15 0.0524 0.719
## 3   0.1  0.7      6      7      1     15      8     22 20 19 0.0476 0.753
## 4   0.1  0.6      6      6      0     13      9     20 22 21 0.0538 0.789
## 5   0.1  0.5      7      5      2     11      8     15 21 21 0.0526 0.853
## 6   0.1  0.4      6      2      2      9      8     10 20 19 0.0354 0.792
## 7   0.1  0.3      6      0      1      5      7      5 17 16 0.0408 0.815
## 8   0.2  0.8      8      9      1     16     11     21 17 15 0.0448 0.800
## 9   0.2  0.7      9     10      5     16     14     25 22 22 0.0437 0.767
## 10  0.2  0.6     10      7      5     15     14     21 23 22 0.0452 0.782
## 11  0.2  0.5     10      4      3     12     14     16 22 22 0.0548 0.815
## 12  0.2  0.4     10      3      4      8     14     12 23 21 0.0563 0.849
## 13  0.2  0.3     10      1      4      6     12      6 20 17 0.0519 0.901
## 14  0.3  0.7     11      9      4     16     17     21 20 18 0.0466 0.801
## 15  0.3  0.6     13      8      7     14     20     22 24 23 0.0488 0.797
## 16  0.3  0.5     13      6      6     13     20     17 24 23 0.0530 0.838
## 17  0.3  0.4     14      4      8      9     20     13 24 24 0.0554 0.889
## 18  0.3  0.3     12      1      3      8     18      7 21 20 0.0600 0.895
## 19  0.4  0.6     15      6       9      12      22      19 21 20 0.0491 0.812
## 20  0.4  0.5     16      6       9      12      25      17 24 23 0.0492 0.839
## 21  0.4  0.4     16      3       8       9      24      12 23 22 0.0554 0.871
## 22  0.4  0.3     16      2       9       7      23       7 22 21 0.0573 0.952
## 23  0.5  0.5     16      3       9      10      24      13 19 18 0.0492 0.804
## 24  0.5  0.4     17      3       8      11      27      11 21 21 0.0554 0.880
## 25  0.5  0.3     16      1      10       6      27       7 21 21 0.0460 0.861
## 26  0.6  0.4     16      2      10       8      25       8 18 15 0.0473 0.803
## 27  0.6  0.3     17      0      11       5      27       6 19 17 0.0519 0.815
## 28  0.7  0.3     14      0      10       4      24       4 14 14 0.0487 0.852
```

### 3.2.2 Find the rejection boundary for pCR and ePD based on pre-specified type I error rate and power level.

```
UIT.design(method = "s2.sf", s1.rej = 10, t1.rej = 3, s1.acc = 8, t1.acc = 5,
    s2.rej = 18, t2.rej = 12, n1 = 41, n2 = 41, s1.rej.delta = 1, t1.rej.delta = 1,
    p0.s = 0.15, p0.t = 0.25, p1.s = 0.3, p1.t = 0.1, output.all = TRUE)
```

```
##   p0.s p0.t p1.s p1.t s1.rej t1.rej s1.acc t1.acc s2.rej t2.rej N1 N2
## 2 0.15 0.25  0.3  0.1     10      2      8      5     18     12 41 41
## 3 0.15 0.25  0.3  0.1     11      2      8      5     18     12 41 41
## 5 0.15 0.25  0.3  0.1     10      3      8      5     18     12 41 41
## 6 0.15 0.25  0.3  0.1     11      3      8      5     18     12 41 41
## 9 0.15 0.25  0.3  0.1     11      4      8      5     18     12 41 41
##    Error Power
## 2 0.0445 0.853
## 3 0.0310 0.835
## 5 0.0487 0.900
## 6 0.0352 0.887
## 9 0.0461 0.941
```

```
##    user  system elapsed
##    4.33    0.00    4.34
```

## 3.3 Two-stage multinomial design with futlity only

### 3.3.1 Power function validation

```
sig.s2.f.UIT <- mapply(UIT.power, method = "s2.f", s1.acc = s2, t1.acc = t2,
    s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p0.s, p.t = p0.t, USE.NAMES = F)
power.s2.f.UIT <- mapply(UIT.power, method = "s2.f", s1.acc = s2, t1.acc = t2,
    s2.rej = a1, t2.rej = a2, n1 = n1, n2 = n2, p.s = p1.s, p.t = p1.t, USE.NAMES = F)
result.s2.f.UIT <- data.frame(p0.s, p0.t, s1.acc = s2, t1.acc = t2, s2.rej = a1,
    t2.rej = a2, N1 = n1, N2 = n2, Error = sig.s2.f.UIT, Power = power.s2.f.UIT)
print(result.s2.f.UIT, digits = 3)
```

```
##    p0.s p0.t s1.acc t1.acc s2.rej t2.rej N1 N2  Error Power
## 1   0.1  0.9      0     13      6     18 13 11 0.0273 0.770
## 2   0.1  0.8      0     15      7     22 18 15 0.0561 0.888
## 3   0.1  0.7      1     15      8     22 20 19 0.0575 0.929
## 4   0.1  0.6      0     13      9     20 22 21 0.0564 0.945
## 5   0.1  0.5      2     11      8     15 21 21 0.0603 0.965
## 6   0.1  0.4      2      9      8     10 20 19 0.0557 0.962
## 7   0.1  0.3      1      5      7      5 17 16 0.0533 0.967
## 8   0.2  0.8      1     16     11     21 17 15 0.0411 0.795
## 9   0.2  0.7      5     16     14     25 22 22 0.0501 0.893
## 10  0.2  0.6      5     15     14     21 23 22 0.0622 0.927
## 11  0.2  0.5      3     12     14     16 22 22 0.0617 0.936
## 12  0.2  0.4      4      8     14     12 23 21 0.0672 0.961
## 13  0.2  0.3      4      6     12      6 20 17 0.0624 0.970
## 14  0.3  0.7      4     16     17     21 20 18 0.0387 0.791
## 15  0.3  0.6      7     14     20     22 24 23 0.0524 0.892
## 16  0.3  0.5      6     13     20     17 24 23 0.0532 0.917
## 17  0.3  0.4      8      9     20     13 24 24 0.0619 0.954
## 18  0.3  0.3      3      8     18      7 21 20 0.0609 0.973
## 19  0.4  0.6      9     12     22     19 21 20 0.0483 0.811
## 20  0.4  0.5      9     12     25     17 24 23 0.0480 0.890
## 21  0.4  0.4      8      9     24     12 23 22 0.0581 0.932
## 22  0.4  0.3      9      7     23      7 22 21 0.0506 0.965
## 23  0.5  0.5      9     10     24     13 19 18 0.0487 0.803
## 24  0.5  0.4      8     11     27     11 21 21 0.0519 0.900
## 25  0.5  0.3     10      6     27      7 21 21 0.0503 0.960
## 26  0.6  0.4     10      8     25      8 18 15 0.0443 0.800
## 27  0.6  0.3     11      5     27      6 19 17 0.0573 0.941
## 28  0.7  0.3     10      4     24      4 14 14 0.0459 0.850
```

### 3.3.2 Find the rejection boundary for pCR and ePD based on pre-specified type I error rate and power level.

```r
UIT.design(method = "s2.f", s1.acc = 7, t1.acc = 5, s2.rej = 17, t2.rej = 13,
    n1 = 41, n2 = 41, s2.rej.delta = 1, t2.rej.delta = 1, p0.s = 0.15, p0.t = 0.25,
    p1.s = 0.3, p1.t = 0.1, output.all = TRUE)
```

```
##   p0.s p0.t p1.s p1.t s1.acc t1.acc s2.rej t2.rej N1 N2  Error Power
## 3 0.15 0.25  0.3  0.1      7      5     18     12 41 41 0.0368 0.968
## 6 0.15 0.25  0.3  0.1      7      5     18     13 41 41 0.0427 0.971
```

```
##    user  system elapsed
##   25.24    0.00   25.25
```