Power and Sample Size by Simulation

Alex Asher
Senior Biostatistician and Software Developer
StataCorp LLC

2025 UK Stata Conference

Outline

- Power and Sample Size Concepts
  - Null and alternative hypotheses
  - Critical values, alpha levels, power, effect size
- Power and Sample Size Using Stata
  - Built-in `power` methods
  - Power by simulation
  - Sample size by simulation
- Group Sequential Designs
  - Introduction to GSDs
  - Simulation-based GSDs

Research Question

image: Journal of Pharmacology & Clinical Research, licensed under CC BY 4.0

- Does our novel chemotherapy shrink tumors?
- Outcome: Tumor Shrinkage Rate (TSR)

  $TSR = (D_b - D_a)/(D_b \times t) \times 100\%$

  D_b is the longest diameter of the tumor before treatment

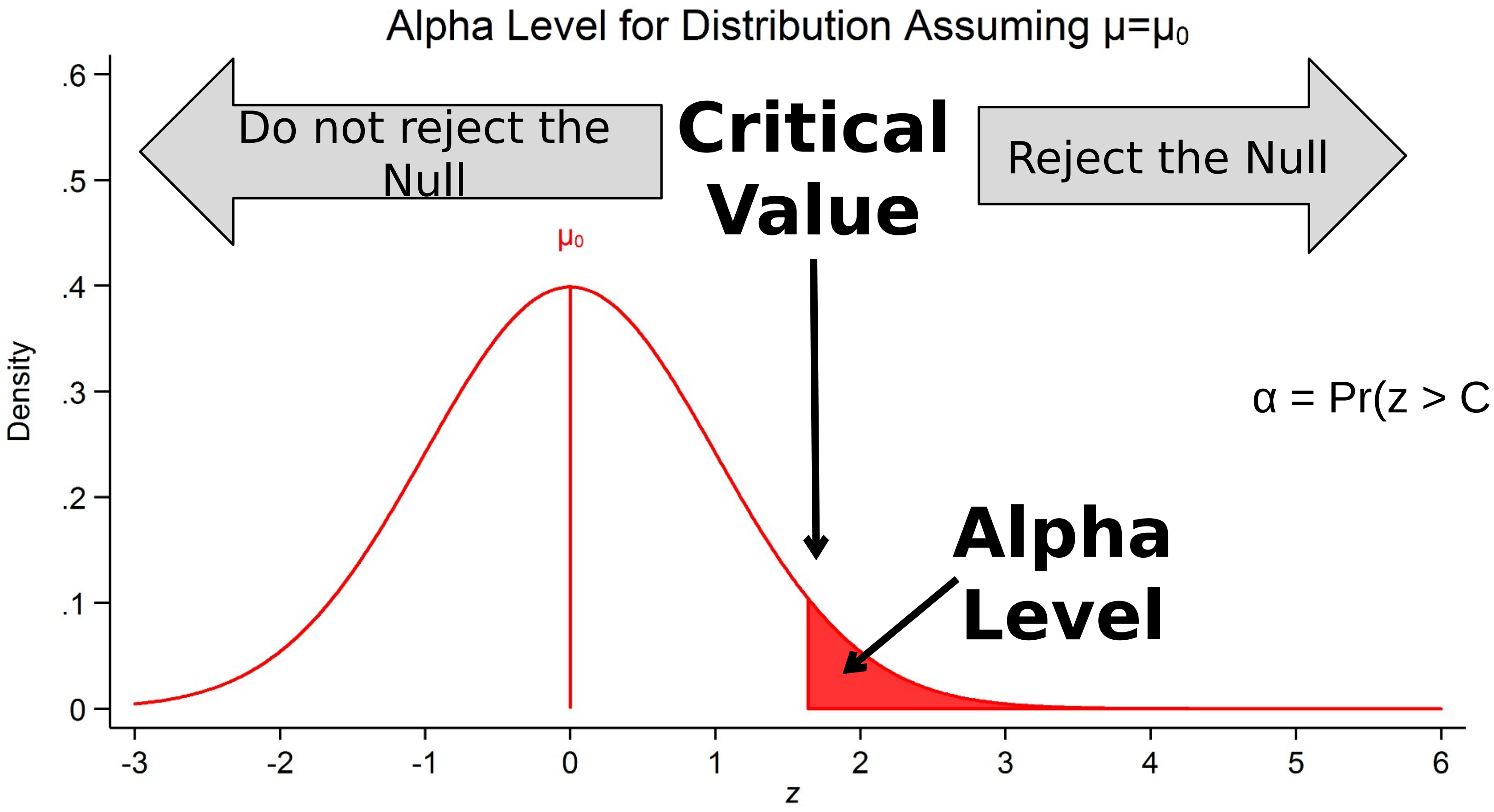
  D_a is the longest diameter after treatment

  t is the time elapsed in days

Null and alternative hypotheses

- H_0: TSR = 0, H_a: TSR > 0
- Conduct a *z*-test with known SD
- Test statistic $z = \frac{\bar{X} - \mu_0}{\sigma / \sqrt{n}}$
- x_i = observed TSR for participant i
  μ_0 = TSR under H_0
  σ = standard deviation (known)
  n = sample size

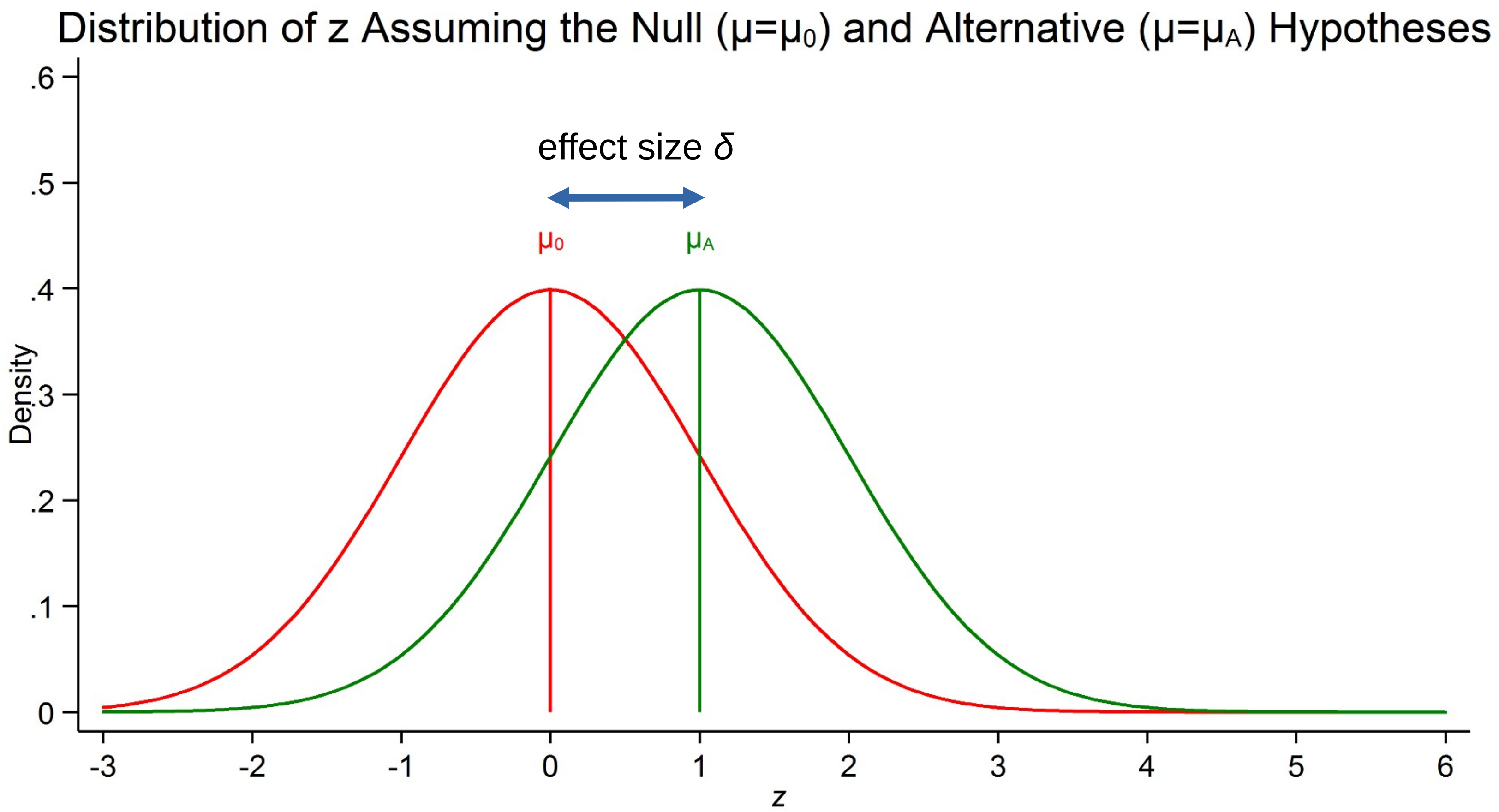
Critical Value and Alpha Level



$\alpha = Pr(z > \text{Critical Value} \mid H_0 \text{ is true})$

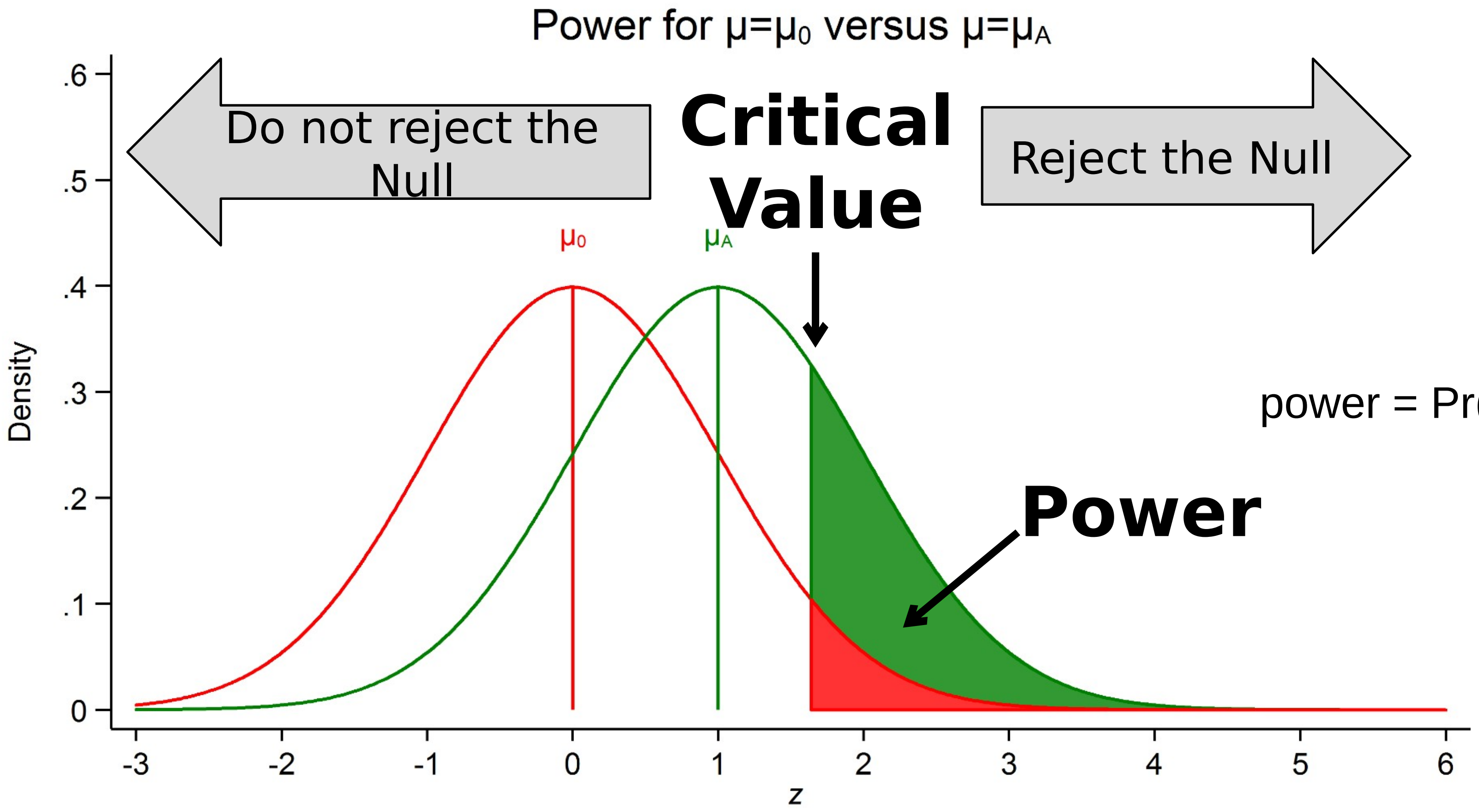
Graph courtesy of Chuck Huber, StataCorp

Distribution Under the Alternative



Graph courtesy of Chuck Huber, StataCorp

Critical Value and Power



Graph courtesy of Chuck Huber, StataCorp

power = Pr(z > Critical Value | H_a is true)

Alpha Level and Power

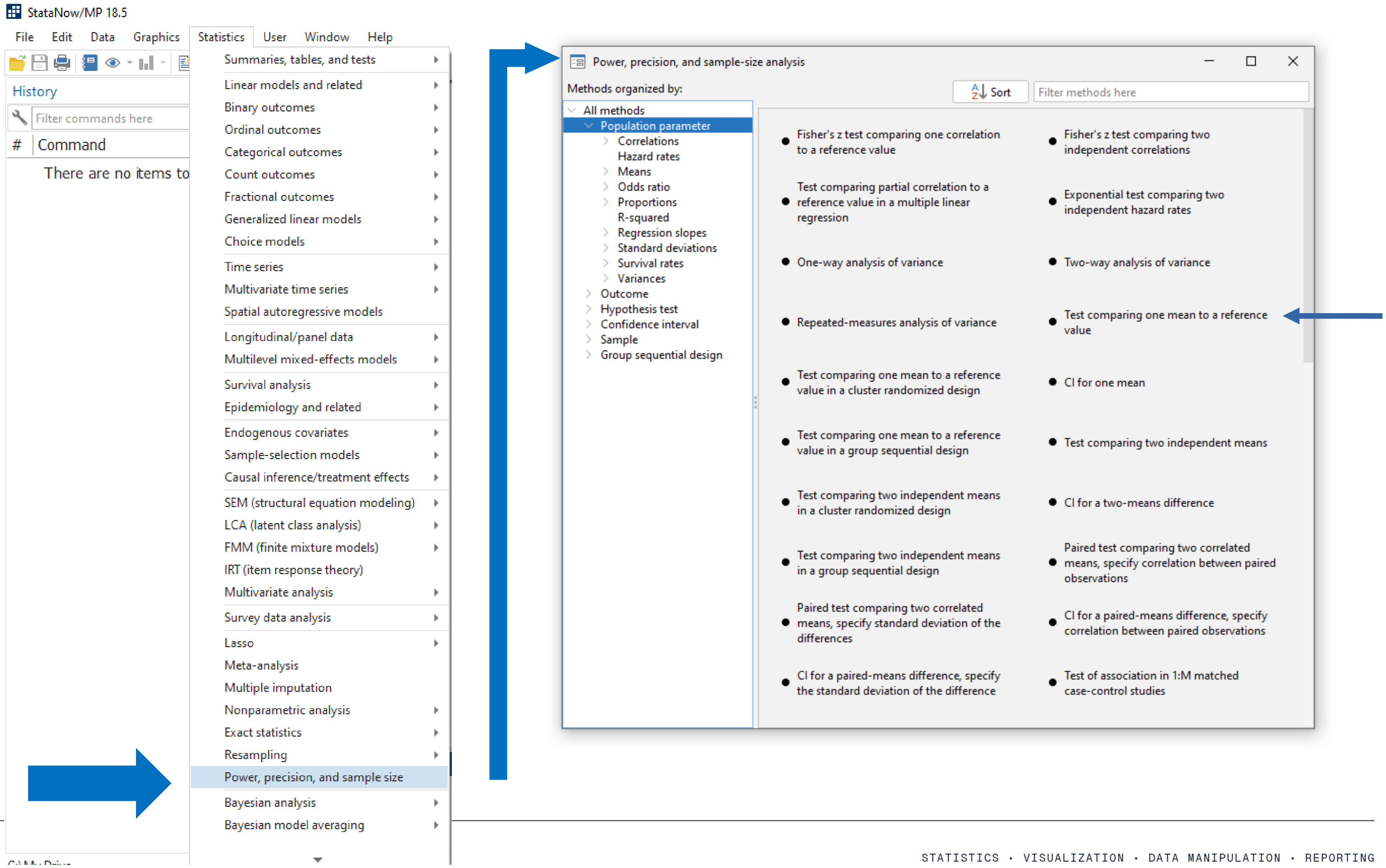
- Alpha Level:
  - The probability of rejecting the null hypothesis assuming that the null hypothesis is true.

- Statistical Power:
  - The probability of rejecting the null hypothesis assuming that the alternative hypothesis is true.

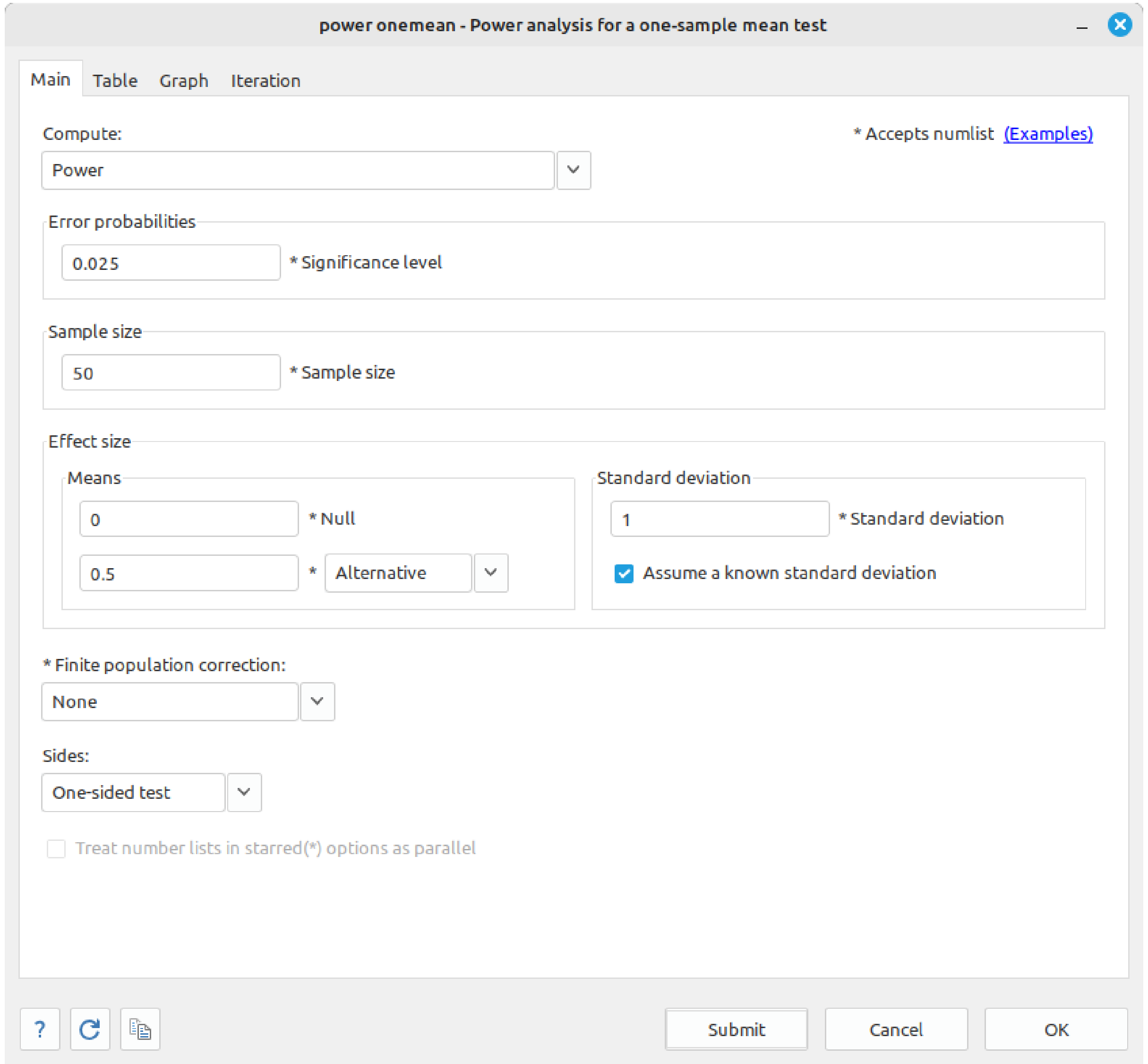
Solve for Power

- User must specify:
  - Alpha
    - One-sided or two-sided
    - Often 0.05 or 0.01 for two-sided tests
    - We will conduct an upper one-sided test at the 0.025 level
  - Sample size
    - Often dictated by funding
    - We will use n=50
  - Effect size
    - Difference between hypothesized parameter values
    - Can be scaled or unscaled (specify SD if unscaled)
    - We will use $\delta=0.5$ and $\sigma=1$

Power calculation in Stata



Power calculation in Stata




```
. power onemean 0 0.5, n(50) alpha(0.025) knownsd onesided

Estimated power for a one-sample mean test
z test
H0: m = m0  versus  Ha: m > m0

Study parameters:

        alpha =    0.0250
            N =        50
        delta =    0.5000
           m0 =    0.0000
           ma =    0.5000
           sd =    1.0000

Estimated power:

        power =    0.9424
```

Power calculation by simulation

```
**# program onemeansim: one sample z-test (upper one-sided test)
program onemeansim, rclass
        version 19
        syntax, n(integer)              ///  Sample size
                [ alpha(real 0.025)     ///  Alpha level
                  m0(real 0)            ///  Mean under the null
                  ma(real 0.5)          ///  Mean under the alternative
                ]
        // Generate a normal random variable with mean = `ma' and SD = 1
        quietly drawnorm y, n(`n') mean(`ma') sd(1) clear
        // Upper one-sided z-test
        ztest y == `m0'
        return scalar reject = (r(p_u) < `alpha')
end
```

Power calculation by simulation

```
. // Try it out
. set seed 9

. onemeansim, n(50)

One-sample z test
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. err.   Std. dev.   [95% conf. interval]
---------+--------------------------------------------------------------------
       y |      50    .6792907    .1414214           1    .4021099    .9564715
------------------------------------------------------------------------------
    mean = mean(y)                                                z =   4.8033
H0: mean = 0

    Ha: mean < 0                 Ha: mean != 0                 Ha: mean > 0
 Pr(Z < z) = 1.0000         Pr(|Z| > |z|) = 0.0000          Pr(Z > z) = 0.0000
```

Create a *power usermethod* command

```
**# program power_cmd_onemeansim: power by simulation
program power_cmd_onemeansim, rclass
        version 19
        syntax, n(integer)            ///  Sample size
                [ alpha(real 0.025)   ///  Alpha level
                  m0(real 0)          ///  Mean under the null
                  ma(real 0.5)        ///  Mean under the alternative
                  reps(integer 100)   ///  Number of replicates
                  seed(numlist max=1) ///  Random seed (no default seed)
                ]

        // Set seed, if specified
        if ("`seed'" != "") set seed `seed'

        // Simulate data and test the null hypothesis
        quietly simulate reject=r(reject), reps(`reps'):    ///
                onemeansim, n(`n') m0(`m0') ma(`ma') alpha(`alpha')
        quietly summarize reject

        // Return results
        return scalar power = r(mean)
        return scalar N     = `n'
        return scalar alpha = `alpha'
        return scalar m0    = `m0'
        return scalar ma    = `ma'
        return scalar delta = `ma' - `m0'
end
```

Run `power onemeansim`

```
. // Start with N=50. Use 1000 reps for accuracy. Set seed for reproducibility.
. power onemeansim, n(50) alpha(0.025) m0(0) ma(0.5) reps(1000) seed(9)

Estimated power
Two-sided test
```

alpha	power	N
.025	.938	50

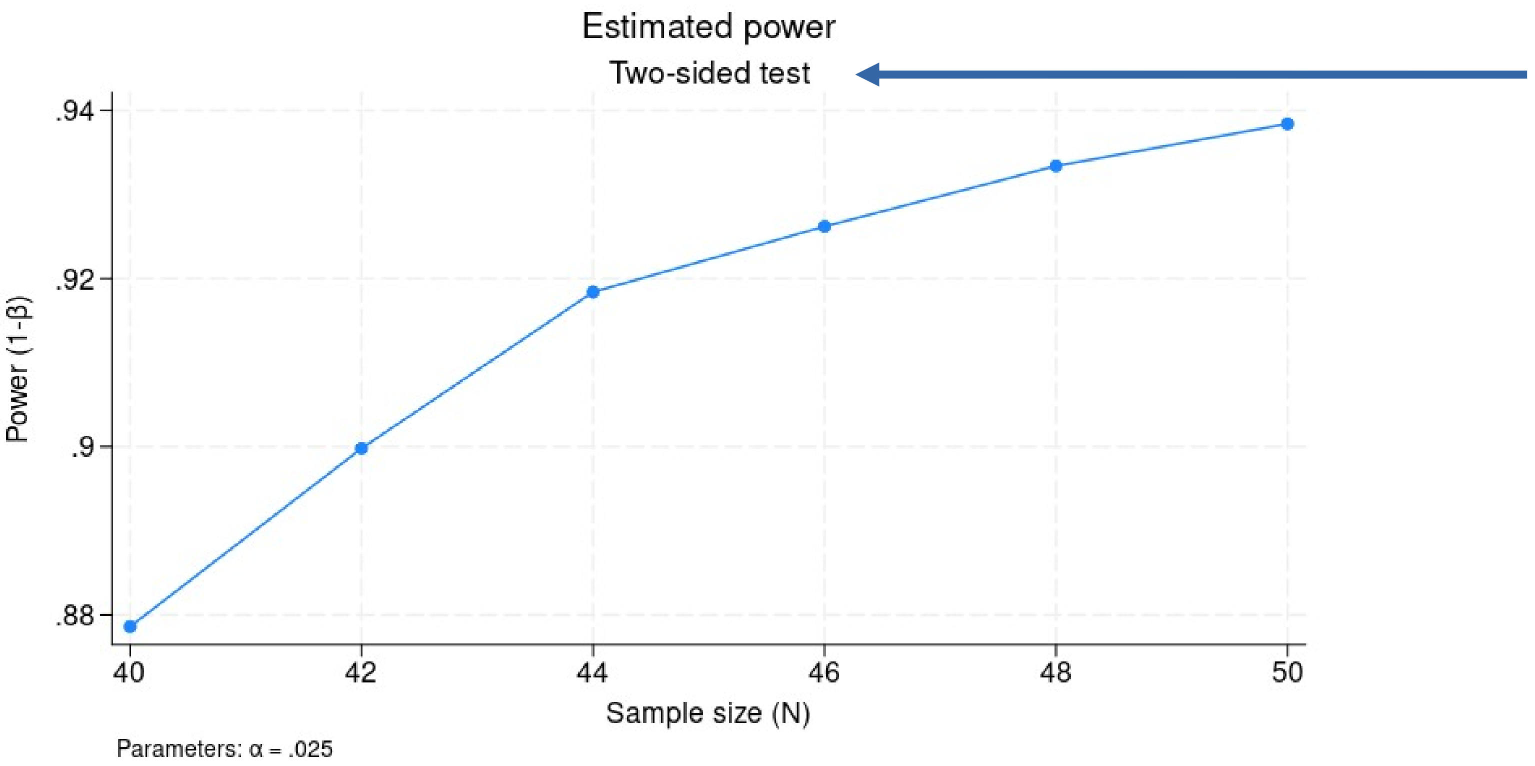
MC Error

$$SE\left(\hat{power}\right)=\sqrt{\frac{power*\left(1-power\right)}{reps}}$$

```
. // Calculate Monte Carlo error (standard error of power estimate)
. local SEpower = sqrt( r(power) * (1-r(power)) / 1000 )

. display "SE of power estimate: `SEpower'"
SE of power estimate: .007626008130077
```

Sample size calculation by simulation

```
. // Option 1: provide a numlist of sample sizes
. set seed 9  // Bonus question: Why don't we use option seed()?

. power onemeansim, n(40(2)50) alpha(0.025) m0(0) ma(0.5) reps(5000) graph
```



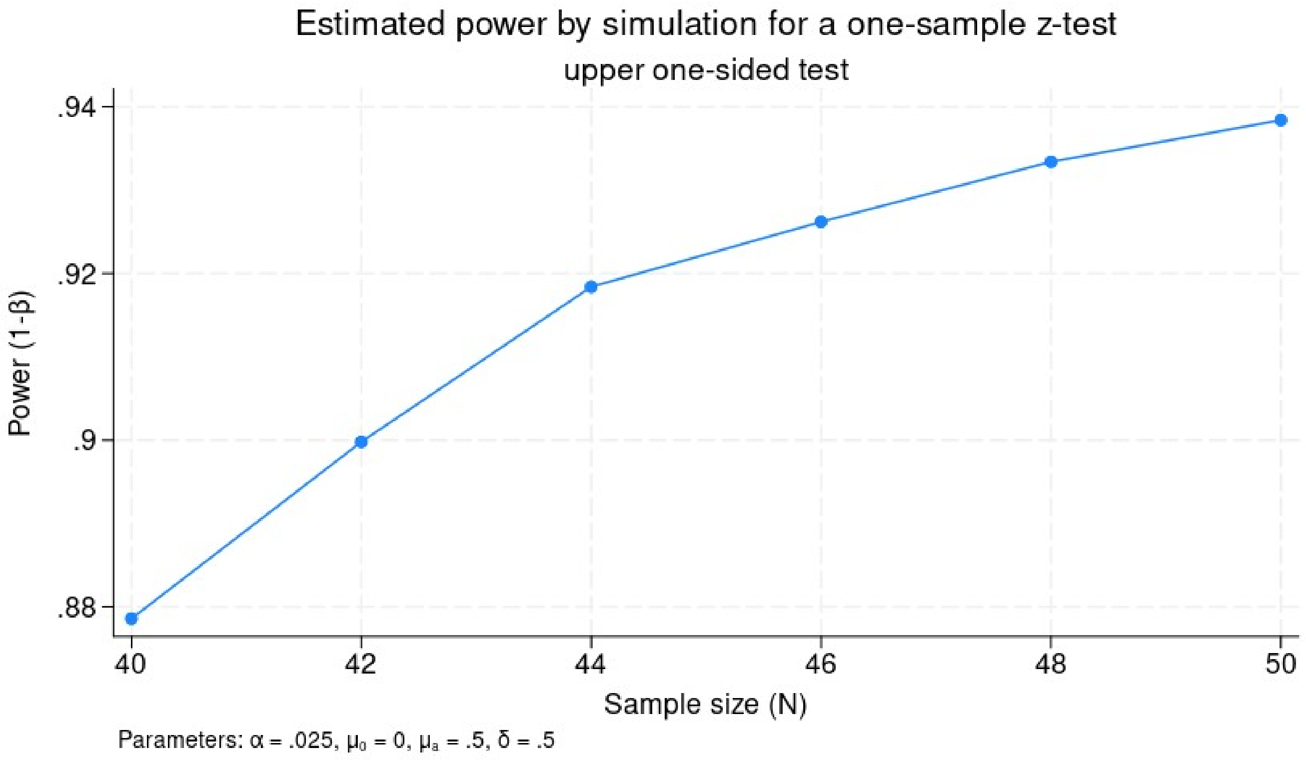
Customize output with an initializer

```
**# program power_cmd_onemeansim_init: use initializer to customize output
program power_cmd_onemeansim_init, sclass
        version 19
        sreturn clear
        sreturn local pss_numopts "m0 ma"                    // Options allowing numlists
        sreturn local pss_colnames "m0 ma delta"             // Columns to add to default table
        sreturn local pss_colgrsymbols "μ{sub:0} μ{sub:a} δ"  // Graph symbols
        sreturn local pss_title "by simulation for a one-sample z-test"
        sreturn local pss_subtitle "upper one-sided test"
end
```

Customize output with an initializer

```
. // Re-run with initializer
. set seed 9

. power onemeansim, n(40(2)50) alpha(0.025) m0(0) ma(0.5) reps(5000) graph table

Estimated power by simulation for a one-sample z-test
upper one-sided test
```

alpha	power	N	m0	ma	delta
.025	.8786	40	0	.5	.5
.025	.8998	42	0	.5	.5
.025	.9184	44	0	.5	.5
.025	.9262	46	0	.5	.5
.025	.9334	48	0	.5	.5
.025	.9384	50	0	.5	.5



Use `power` to calculate sample size by simulation

```
**# program power_cmd_onemeansampsize: simulation-based sample-size evaluator
program power_cmd_onemeansampsize, rclass
        version 19
        syntax, power(numlist >0 <1 max=1) /// Requested power
                [ alpha(real 0.05)         /// Alpha level
                  m0(real 0)               /// Mean under the null
                  ma(real 0.5)             /// Mean under the alternative
                  reps(integer 100)        /// Number of repetitions for power calculation
                  seed(numlist max=1)      /// Random seed (no default seed)
                  startval(integer 100)    /// Starting value for sample size
                  maxval(integer 1000)     /// Maximum value for sample size
                  minval(integer 1)        /// Minimum value for sample size
                  powtol(real 0.01)        /// Tolerance for power search (default 1%)
                  biter(integer 100)       /// Max iterations for bisection algorithm
                  nfractional              /// Unused, but required by -gsdesign-
                  onesided                 /// Unused, but required by -gsdesign-
                ]

        // Set seed, if specified
        if ("`seed'" != "") set seed `seed'

        // Test powtol() to ensure it's not set too low
        assert (`powtol' > 0)
```

Use `power` to calculate sample size by simulation (continued)

```
// Test maxval() to ensure it's not set too low
power_cmd_onemeansim, n(`maxval') alpha(`alpha') m0(`m0') ma(`ma') reps(`reps')
if (r(power) < `power') {
        di as error "error: {bf:maxval()} too small"
        exit 480
}
// Test -minval()- to ensure it's not set too high
power_cmd_onemeansim, n(`minval') alpha(`alpha') m0(`m0') ma(`ma') reps(`reps')
if (r(power) > `power') {
        di as error "error: {bf:minval()} too large"
        exit 480
}

// Compare requested power to power attained via simulation at the starting value
local thisN = `startval'
local lastN = .
power_cmd_onemeansim, n(`thisN') alpha(`alpha') m0(`m0') ma(`ma') reps(`reps')
local pdiff = r(power) - `power'
```

Use `power` to calculate sample size by simulation (continued)

```
// Repeat guess-n-check process using bisection algorithm
local i = 1
while (abs(`pdiff') > `powtol') {
        local ++i
        if (`i' > `biter') {
                di as error "convergence not achieved"
                exit 430
        }

        // Bisection algorithm
        local lastN = `thisN'
        if (`pdiff' > 0) { // Overpowered: decrease N
                local maxval = `thisN'
        }
        else {             // Underpowered: increase N
                local minval = `thisN'
        }
        local thisN = ceil((`maxval' + `minval') / 2)
        if (`thisN' == `lastN') { // Already converged
                continue, break
        }
        power_cmd_onemeansim, n(`thisN') alpha(`alpha') m0(`m0') ma(`ma') reps(`reps')
        local pdiff = r(power) - `power'
}
```

Use `power` to calculate sample size by simulation (continued)

```
        // Return results
        return scalar power     = r(power)
        return scalar N         = `thisN'
        return scalar alpha     = `alpha'
        return scalar m0        = `m0'
        return scalar ma        = `ma'
        return scalar delta     = `ma' - `m0'
        return local  direction = "upper"
end
```

Customize output with an initializer

```
**# program power_cmd_onemeansampsize_init: use initializer to customize output
program power_cmd_onemeansampsize_init, sclass
        version 19
        sreturn clear
        sreturn local pss_numopts "m0 ma"                      // Options allowing numlists
        sreturn local pss_colnames "m0 ma delta"               // Columns to add to default table
        sreturn local pss_colgrsymbols "μ{sub:0} μ{sub:a} δ"  // Graph symbols
        sreturn local pss_title "by simulation for a one-sample z-test"
        sreturn local pss_subtitle "upper one-sided test"
end
```

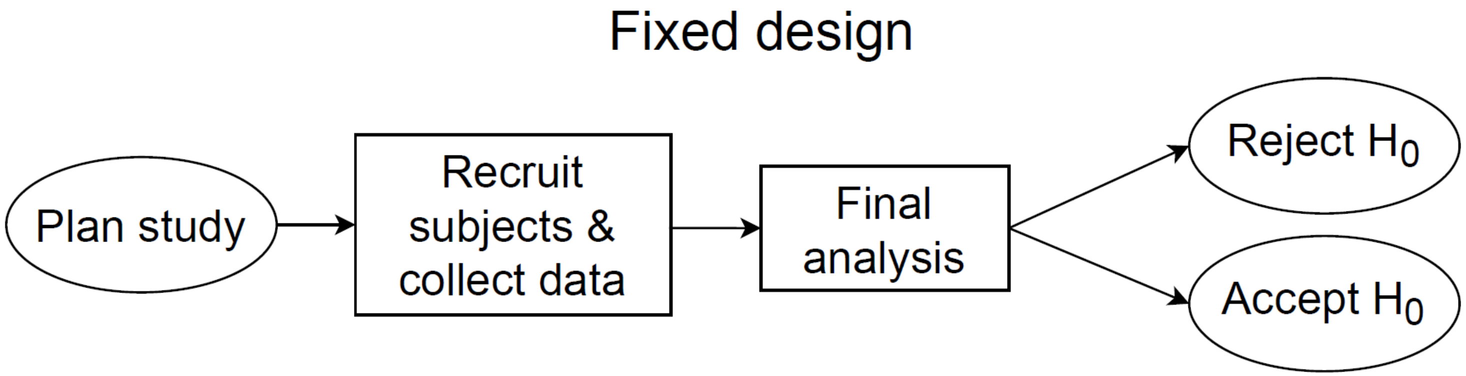
Calculate sample size by simulation

```
. // Try it as a -power- command
. power onemeansampsize, power(0.9) alpha(0.025) m0(0) ma(0.5) startval(100) reps(500) seed(9)

Estimated sample size by simulation for a one-sample z-test
upper one-sided test
```

alpha	power	N	m0	ma	delta
.025	.896	39	0	.5	.5

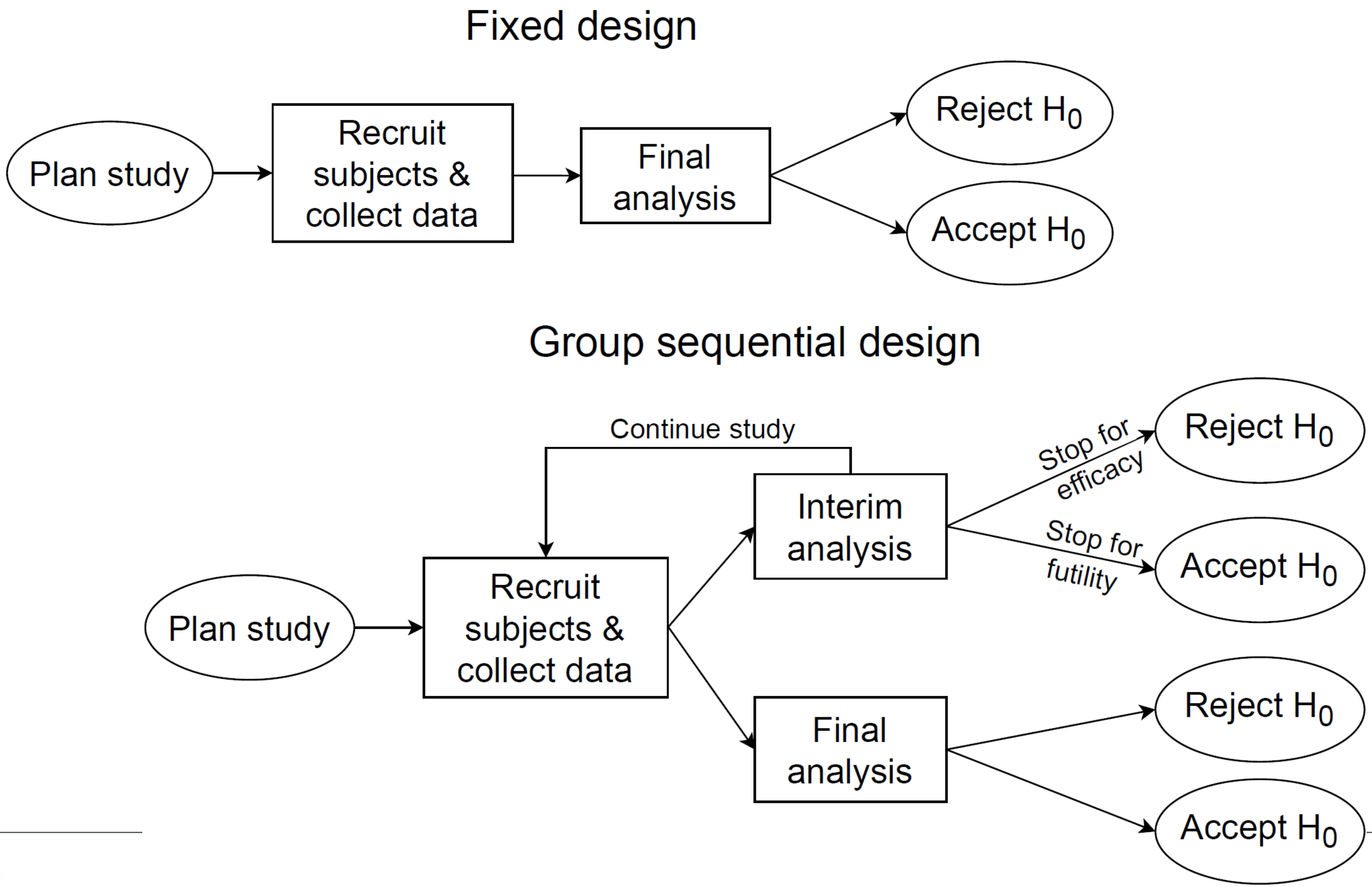
Fixed Sample Design vs Group Sequential Design

or: how I learned to stop worrying and accept the null

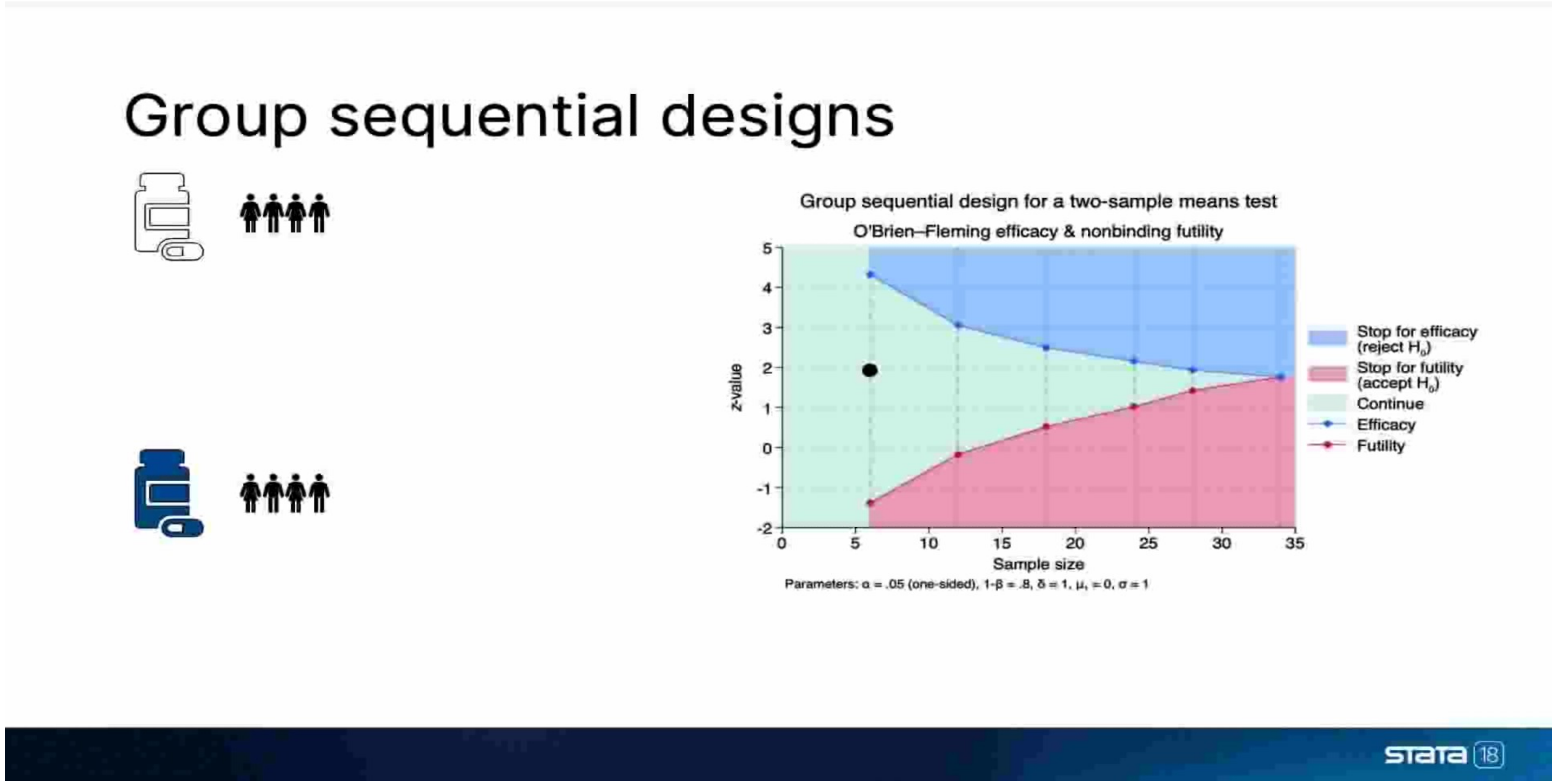


Fixed Sample Design vs Group Sequential Design

or: how I learned to stop worrying and accept the null



GSD in action



From https://youtu.be/hO2qW1NLrMk?si=gXsB8zDkAmq9K2IE by Meghan Cain, StataCorp

gsdesign syntax

gsdesign *method* . . . [, *designopts boundopts*]

where *method . . .* refers to a **power** *method* that is used for sample-size calculation, *designopts* are options controlling the sample-size calculation, and *boundopts* are options controlling the calculation of the stopping boundaries.

method	Description
onemean	GSD for one-sample mean test
twomeans	GSD for two-sample means test
oneproportion	GSD for one-sample proportion test
twoproportions	GSD for two-sample proportions test
logrank	GSD for a log-rank test
usermethod	user-defined sample-size calculation

gsdesign supports the above methods when they are used to calculate sample size with simple random sampling. To use an unsupported method, specify option **methodok**.

`gsdesign` syntax (continued)

designopts	Description
Main	
methodopts	method-specific options
`alpha(#)`	overall significance level for all tests; default is `alpha(0.05)`
`power(#)`	overall power for all tests; default is `power(0.8)`
`beta(#)`	overall probability of type II error for all tests; default is `beta(0.2)`
`onesided`	request a one-sided test; default is two-sided
`nfractional`	report fractional sample size
`force`	allow calculation with unsupported *methodopts*
`methodok`	allow calculation with unsupported *method*
`poweriteration(`*powiteropts*`)`	iteration options for the calculation of fixed-study sample size; not available with *method* `logrank`; seldom used

GSD by simulation

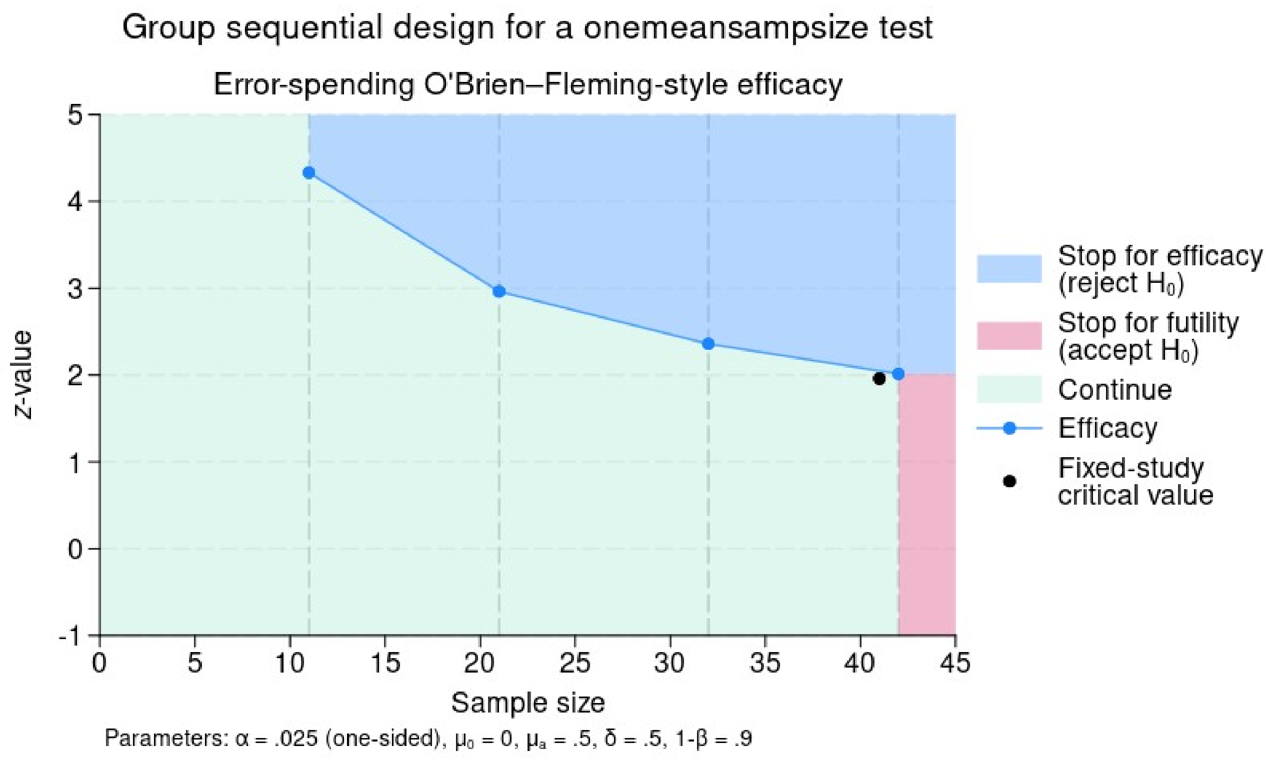
```
. gsdesign onemeansampsize, power(0.9) alpha(0.025) m0(0) ma(0.5) startval(50) reps(500) seed(9) ///
>                           onesided nlooks(4) efficacy(errobfleming) graphbounds

Group sequential design by simulation for a one-sample z-test
upper one-sided test

Efficacy: Error-spending O'Brien–Fleming style

Study parameters:
      alpha = 0.0250  (upper one-sided)
      power = 0.9000
         m0 = 0.0000
         ma = 0.5000
      delta = 0.5000

Expected sample size:
         H0 =  41.89
         Ha =  32.14

Info. ratio = 1.0183
    N fixed =     41
      N max =     42

Fixed-study crit. value = 1.9600
```

GSD by simulation (continued)

Critical values, p-values, and sample sizes
for a group sequential design

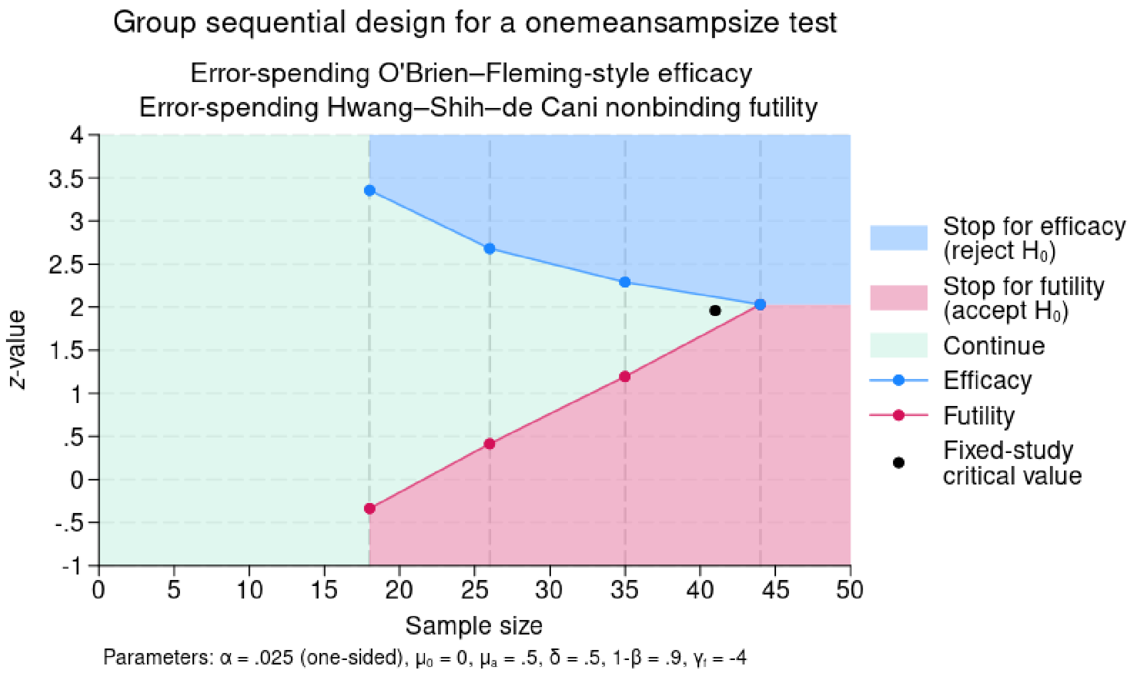
Look	Info. frac.	Efficacy Upper	Efficacy p-value	Sample size N
1	0.25	4.3326	0.0000	11
2	0.50	2.9631	0.0015	21
3	0.75	2.3590	0.0092	32
4	1.00	2.0141	0.0220	42

Notes: Critical values are for z statistics; otherwise, use p-value boundaries.
Requested information fraction not attained.



GSD with futility bounds

```
. gsdesign onemeansampsize, power(0.9) alpha(0.025) m0(0) ma(0.5) startval(50) reps(500) seed(9) ///
>                           onesided information(40 60 80 100) efficacy(errobfleming)          ///
>                           futility(hsdecani(-4)) graphbounds

Group sequential design by simulation for a one-sample z-test
upper one-sided test

Efficacy: Error-spending O'Brien-Fleming style
Futility: Error-spending Hwang-Shih-de Cani, nonbinding, gamma = -4.0000
```



Stata 19

Code from all slides available upon request

aasher@stata.com