

# Power of the power command in Stata 13

Yulia Marchenko

Director of Biostatistics  
StataCorp LP

2013 Stata Conference

## Outline

- Basic functionality of the power command
- Tables of results
- Automatic graphs
- GUI
- Adding your own methods (forthcoming)

The `power` command provides power and sample-size (PSS) analysis for hypothesis tests.

You can compute

- sample size given power and effect size
- power given sample size and effect size
- effect size given power and sample size

You can also

- express the magnitude of an effect of interest (or effect size) in multiple ways
- obtain results for multiple scenarios
- display multiple results in a table
- display multiple results on a graph
- use the PSS Control Panel for point-and-click analysis
- read about PSS in the new [PSS] manual

And,

- You will soon be able to add your own methods to the `power` command and have access to its features such as multiple-scenario support, tables, and graphs.

## One-sample comparison of a

- Mean (one-sample  $t$  test)
- Proportion
- Correlation
- Variance

## Two-sample comparison of independent

- Means (two-sample  $t$  test)
- Proportions
- Correlations
- Variances

## Two-sample comparison of paired

- Means (paired  $t$  test)
- Proportions (McNemar's test)

## Example

Suppose that school officials would like to study the performance of currently-enrolled students on a standardized math test. They want to compare the average math score with the previous year's average of 25 points using a one-sample  $t$  test. Assuming a standard deviation of 6.5 points, the officials want to determine the sample size necessary to detect a hypothesized average score of 30 points with 90% power using a 5%-level two-sided test.

- Compute sample size given a power of 0.9:

```
. power onemean 25 30, power(0.9) sd(6.5)
Performing iteration ...
Estimated sample size for a one-sample mean test
t test
Ho: m = m0 versus Ha: m != m0
Study parameters:
      alpha =    0.0500
      power =    0.9000
      delta =    0.7692
      m0 =    25.0000
      ma =    30.0000
      sd =    6.5000
Estimated sample size:
      N =          20
```

- Compute power given a sample size of 15:

```
. power onemean 25 30, n(15) sd(6.5)
Estimated power for a one-sample mean test
t test
Ho: m = m0 versus Ha: m != m0
Study parameters:
      alpha =    0.0500
         N =      15
      delta =    0.7692
       m0 =   25.0000
       ma =   30.0000
       sd =    6.5000
Estimated power:
      power =    0.7911
```



- Compute effect size and a hypothesized mean given a sample size of 15 and a power of 0.9:

```
. power onemean 25, power(0.9) n(15) sd(6.5)
Performing iteration ...
Estimated target mean for a one-sample mean test
t test
Ho: m = m0 versus Ha: m != m0; ma > m0
Study parameters:
      alpha =    0.0500
      power =    0.9000
         N =      15
        m0 =   25.0000
         sd =    6.5000

Estimated effect size and target mean:
      delta =    0.9009
         ma =   30.8557
```

- Compute power for multiple sample sizes:

```
. power onemean 25 30, n(10 15 20) sd(6.5)
```

Estimated power for a one-sample mean test

t test

Ho:  $m = m_0$  versus Ha:  $m \neq m_0$

alpha	power	N	delta	m0	ma	sd
.05	.583	10	.7692	25	30	6.5
.05	.7911	15	.7692	25	30	6.5
.05	.9035	20	.7692	25	30	6.5

- Customize the table:

```
. power onemean 25 30, n(10 15 20) sd(6.5) table(alpha:"Significance level"
> N:" Sample size" power:" Power" delta:" Effect size", formats(power "%6.2f"
> delta "%6.2f"))
```

Estimated power for a one-sample mean test

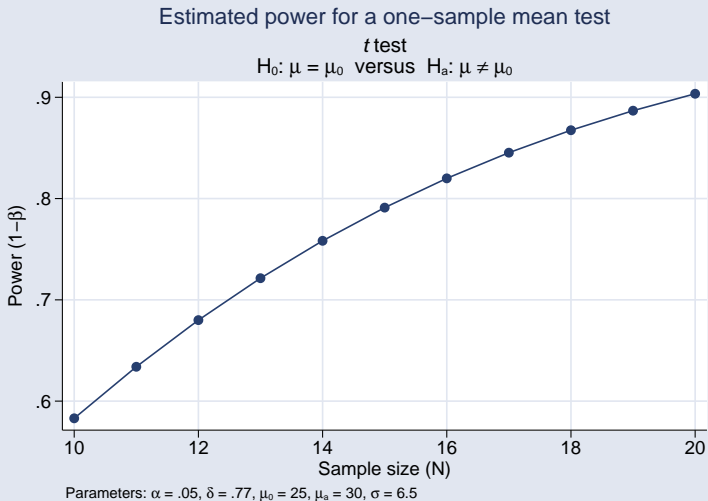
t test

Ho:  $m = m_0$  versus Ha:  $m \neq m_0$

Significance level	Sample size	Power	Effect size
.05	10	0.58	0.77
.05	15	0.79	0.77
.05	20	0.90	0.77

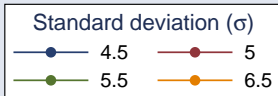
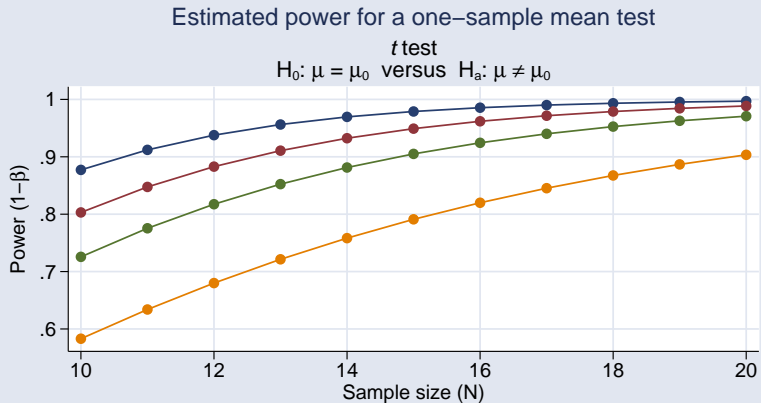
- Plot power for a range of sample sizes:

```
. power onemean 25 30, n(10(1)20) sd(6.5) graph
```



- Plot power for a range of sample sizes and standard deviations:

. power onemean 25 30, n(10(1)20) sd(4.5 5 5.5 6.5) graph



Parameters:  $\alpha = .05$ ,  $\mu_0 = 25$ ,  $\mu_a = 30$

Go to the **Statistics** > **Power and sample size** menu to launch the PSS Control Panel for point-and-click analysis:

(NEXT SLIDE)

## Methods organized by:

- [-] Population parameter
  - [+] Correlations
  - Hazard rates
  - [-] Means
    - One sample
    - Two independent samples
    - Two paired samples
  - [+] Proportions
  - Regression slope, Cox model
  - [+] Standard deviations
  - [+] Survival rates
  - [+] Variances
- [+] Outcome
- [+] Analysis type
- [+] Sample

- Test comparing one mean to a reference value
- Test comparing two independent means
- Paired test comparing two correlated means, specify correlation between paired observations
- Paired test comparing two correlated means, specify standard deviation of the differences

Suppose you want to add the `mymethod` method to `power`. Here is an outline of the steps to follow:

- create an `rclass` program defined by `power_cmd_mymethod.ado` that performs PSS computations and follows `power`'s conventions for naming options and storing results;
- optionally, create the initializer, an `sclass` program defined by `power_cmd_mymethod_init.ado`, that specifies the information about table columns, options which may allow a *numlist*, etc.;
- optionally, create a program defined by `power_cmd_mymethod_parse.ado` that checks the syntax of method-specific options.



- As an illustration, we will compute power for a one-sample  $z$  test.
- We want to add a method called `myztest` that performs this computation to the `power` command.
- We first create an `rclass` program which computes power for a  $z$  test, and store the program in a file named `power_cmd_myztest.ado`.

```

*! version 1.0.0 18jul2013
*! Power computation for a one-sample z test
program power_cmd_myzttest, rclass
    version 13

    syntax ,          n(integer)          /* parse options */
                   STDDiff(real)        /// sample size
                   [                                     ///
                   Alpha(string)         /// standardized difference
                   ONESIDEd              /// significance level
                   ]                             /// one-sided test

                                           /* compute power */
    tempname power za
    if ("`onesided'"=="") scalar `za` = invnormal(1-`alpha`/2)
    else scalar `za` = invnormal(1-`alpha`)
    scalar `power` = normal(`stddiff`*sqrt(`n`)-`za`)
                                           /* return results */

    return scalar N          = `n`
    return scalar power      = `power`
    return scalar alpha      = `alpha`
    return scalar stddiff    = `stddiff`
    return scalar onesided   = ("`onesided'"!="")
end

```

- Compute power given a sample size of 20 and a standardized difference of 1:

```
. power myztest, n(20) stddiff(1)
```

```
Estimated power
```

```
Two-sided test
```

alpha	power	N
.05	.994	20

- Compute power for a range of sample sizes:

```
. power myztest, n(10 15 20) stddiff(1)
```

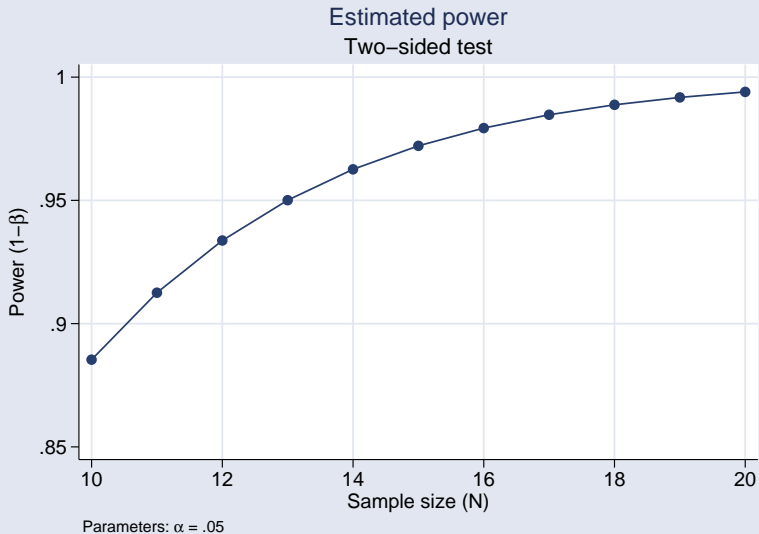
```
Estimated power
```

```
Two-sided test
```

alpha	power	N
.05	.8854	10
.05	.9721	15
.05	.994	20

- Plot power for a range of sample sizes:

```
. power myztest, n(10(1)20) stddiff(1) graph
```



- We would like to add a column containing standardized differences to our table.
- We need to somehow get this information to the power command. This is done with the initializer.
- To add a column to the displayed table, we store the name of the return scalar containing the values of that column in the `s(pss_columns)` macro in the initializer program
- `power_cmd_myztest` stores the differences in a scalar `r(stddiff)`.
- We create an `sclass` program `power_cmd_myztest_init` and store "stddiff" in `s(pss_colnames)`.

```
!* version 1.0.0 18jul2013
program power_cmd_myztest_init, sclass
    sreturn local pss_colnames "stddiff"
end
```

- The `power` command uses the convention that the name of the table column is the same as the name of the return scalar containing values of this column.
- The `stddiff` column is now displayed in the default table:

```
. power myztest, n(20) stddiff(1)
```

```
Estimated power
```

```
Two-sided test
```

alpha	power	N	stddiff
.05	.994	20	1

- If desired, we can change the default column label:

```
. power myztest, n(20) stddif(1) table(, labels(stddiff " Std. Difference"))
```

```
Estimated power
```

```
Two-sided test
```

alpha	power	N	Std. Difference
.05	.994	20	1

- We would also like to be able to specify multiple values in the `stddiff()` option.
- If we try doing this now, we will receive an error:

```
. power myztest, n(20) stddiff(0.5 1)
option stddiff() invalid
r(198);
```

- We need to let `power` know that we want to allow `stddiff()` to accept multiple values.
- We need to include the name of each option (with abbreviation) for which we wish to allow multiple values in the `s(pss_numopts)` macro in the initializer:

```
. type power_cmd_myztest_init.ado
*! version 1.0.0 18jul2013
program power_cmd_myztest_init, sclass
    sreturn local pss_colnames "stddiff"
    sreturn local pss_numopts "STDDiff"
end
```

- The `stddiff()` option now accepts multiple values:

```
. power myztest, n(20) stddiff(0.5 1)
Estimated power
Two-sided test
```

alpha	power	N	stddiff
.05	.6088	20	.5
.05	.994	20	1



- For example, we can produce results for all possible combinations of specified sample sizes and standardized differences:

```
. power myztest, n(10 20) stddiff(0.5 1)
```

Estimated power

Two-sided test

alpha	power	N	stddiff
.05	.3524	10	.5
.05	.8854	10	1
.05	.6088	20	.5
.05	.994	20	1

- or only for specific combinations:

```
. power myztest, n(10 20) stddiff(0.5 1) parallel
```

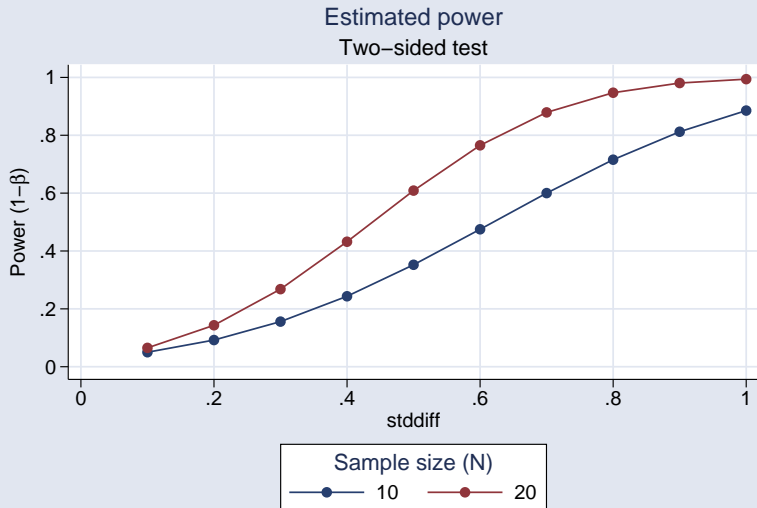
Estimated power

Two-sided test

alpha	power	N	stddiff
.05	.3524	10	.5
.05	.994	20	1

- We can also plot powers for a range of standardized differences and sample sizes

```
. power myztest, n(10 20) stddiff(0.1(0.1)1) graph(xdimension(stddiff))
```



Parameters:  $\alpha = .05$

- More power and sample-size computations
- More control for customization of user-written methods; keep an eye out for a forthcoming FAQ for more details
- Possibly other additions based on your feedback