

A Vignette for the R package: ezsim

TszKin Julian Chan <ctszkin@gmail.com>

September 12, 2011

Contents

1	Introduction	1
2	Pre-simulation	6
2.1	Parameters	7
2.2	Data Generating Process	8
2.3	Estimators	9
2.4	True Value	9
2.5	Display Name	10
3	Post-simulation	10
3.1	Summary Table	10
3.1.1	Subset of the Summary Table	10
3.1.2	More Summary Statistics	10
3.1.3	Customize the Summary Statistics	11
3.2	Plotting the simulation	11
3.2.1	Plotting an ezsim object	11
3.2.2	Subset of the Plot	11
3.2.3	Parameters Priority of the Plot	13
3.2.4	Density Plot	15
3.2.5	Plot the summary ezsim	17
3.2.6	Plot the Power Function	20

1 Introduction

ezsim provides a handy way to run simulation and examine its results. Users dont have to work on those tedious jobs such as loop over several set of parameters, organize and summarize the simulation results,etc. Those tedious jobs are completed by ezsim. Users are only required to define some necessary information, such as data generating process, parameters and estimators. In addition, ezsim provides a flexible way to visualize the simulation results and support parallel computing. In this vignette, several examples are used to demonstrate how to create a simulation with ezsim. Our first example will give you a first glance of what ezsim can do for you. Section 2 and 3 will tell you how to use ezsim.

Suppose $x_1 \dots x_n$ are drawn independently from a normal distribution with mean μ and standard deviation σ . We want to know how the sample size n , mean μ and standard deviation σ would affect the behavior of the sample mean.

We would like to replicate the simulation for 200 times. n takes value from 20,40,60,80 . μ takes value from 0,2. σ takes value from 1,3,5.

```
> library(ezsim)
> ezsim_basic<-ezsim(
+   m           = 50,
+   run         = TRUE,
+   core        = 1,
+   display_name = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
+   parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
+   dgp         = function() rnorm(n,mu,sigma),
+   estimator    = function(x) c(mean_hat = mean(x),
+                                sd_mean_hat=sd(x)/sqrt(length(x)-1)),
+   true_value   = function() c(mu, sigma / sqrt(n-1))
+ )

> summary(ezsim_basic)
```

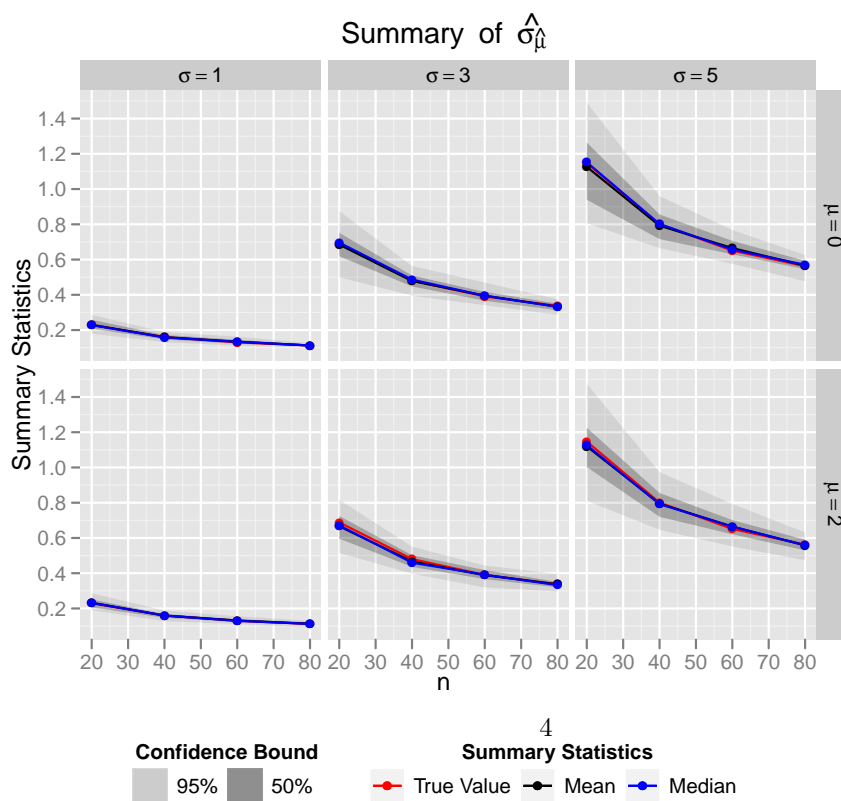
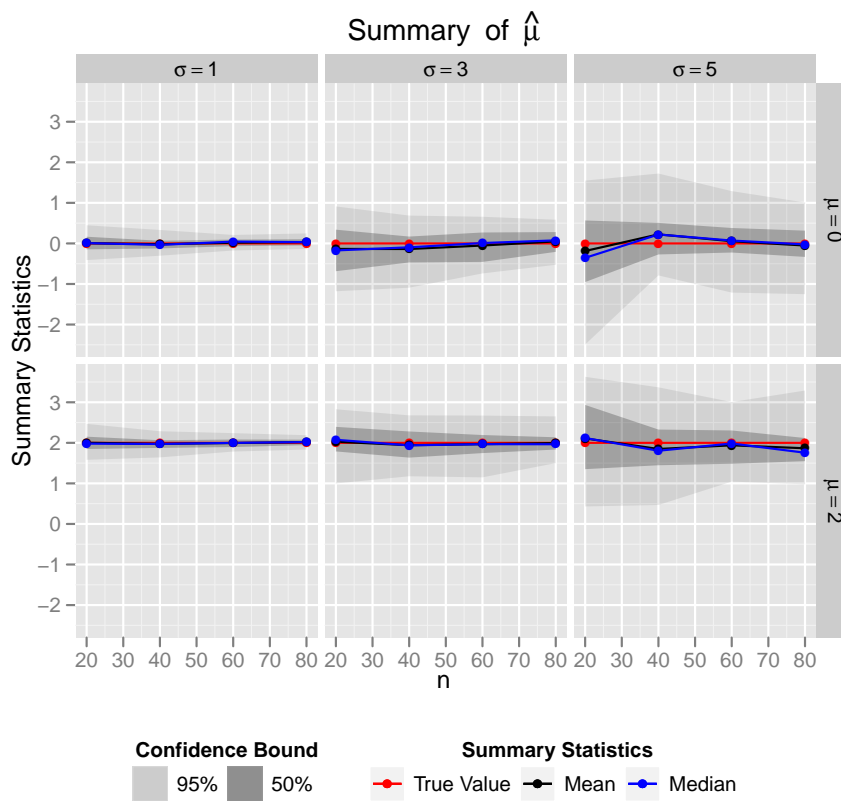
	estimator	n	sigma	mu	Mean	TV	Bias	SD	rmse	JB_test
1	hat(mu)	20	1	0	0.0058	0.0000	0.0058	0.2352	0.2353	0.9451
2	hat(mu)	20	1	2	2.0022	2.0000	0.0022	0.2232	0.2232	0.7882
3	hat(mu)	20	3	0	-0.1359	0.0000	-0.1359	0.6651	0.6788	0.7387
4	hat(mu)	20	3	2	2.0275	2.0000	0.0275	0.5665	0.5672	0.4604
5	hat(mu)	20	5	0	-0.1868	0.0000	-0.1868	1.0644	1.0806	0.6125
6	hat(mu)	20	5	2	2.1080	2.0000	0.1080	0.9716	0.9776	0.6396
7	hat(mu)	40	1	0	-0.0208	0.0000	-0.0208	0.1704	0.1716	0.7208
8	hat(mu)	40	1	2	1.9751	2.0000	-0.0249	0.1682	0.1701	0.6007
9	hat(mu)	40	3	0	-0.1376	0.0000	-0.1376	0.4712	0.4909	0.8154
10	hat(mu)	40	3	2	1.9423	2.0000	-0.0577	0.4273	0.4312	0.4237
11	hat(mu)	40	5	0	0.2230	0.0000	0.2230	0.6462	0.6836	0.1074
12	hat(mu)	40	5	2	1.8488	2.0000	-0.1512	0.7874	0.8018	0.0063
13	hat(mu)	60	1	0	0.0177	0.0000	0.0177	0.1264	0.1276	0.1313
14	hat(mu)	60	1	2	1.9959	2.0000	-0.0041	0.1321	0.1321	0.5789
15	hat(mu)	60	3	0	-0.0518	0.0000	-0.0518	0.4610	0.4640	0.6215
16	hat(mu)	60	3	2	1.9677	2.0000	-0.0323	0.3940	0.3953	0.8140
17	hat(mu)	60	5	0	0.0538	0.0000	0.0538	0.5856	0.5880	0.4370
18	hat(mu)	60	5	2	1.9392	2.0000	-0.0608	0.5690	0.5723	0.5711
19	hat(mu)	80	1	0	0.0334	0.0000	0.0334	0.1052	0.1104	0.4783
20	hat(mu)	80	1	2	2.0191	2.0000	0.0191	0.1031	0.1048	0.7141
21	hat(mu)	80	3	0	0.0493	0.0000	0.0493	0.3327	0.3363	0.9263
22	hat(mu)	80	3	2	1.9941	2.0000	-0.0059	0.2729	0.2730	0.2336
23	hat(mu)	80	5	0	-0.0518	0.0000	-0.0518	0.5772	0.5795	0.4656
24	hat(mu)	80	5	2	1.8662	2.0000	-0.1338	0.5663	0.5819	0.0862
25	hat(sigma[hat(mu)])	20	1	0	0.2295	0.2294	0.0001	0.0307	0.0307	0.7337
26	hat(sigma[hat(mu)])	20	1	2	0.2324	0.2294	0.0030	0.0296	0.0297	0.4760
27	hat(sigma[hat(mu)])	20	3	0	0.6849	0.6882	-0.0034	0.1028	0.1029	0.9082
28	hat(sigma[hat(mu)])	20	3	2	0.6664	0.6882	-0.0219	0.0840	0.0868	0.5026
29	hat(sigma[hat(mu)])	20	5	0	1.1260	1.1471	-0.0211	0.1926	0.1938	0.4244
30	hat(sigma[hat(mu)])	20	5	2	1.1197	1.1471	-0.0274	0.1765	0.1786	0.7894
31	hat(sigma[hat(mu)])	40	1	0	0.1596	0.1601	-0.0005	0.0171	0.0171	0.7279

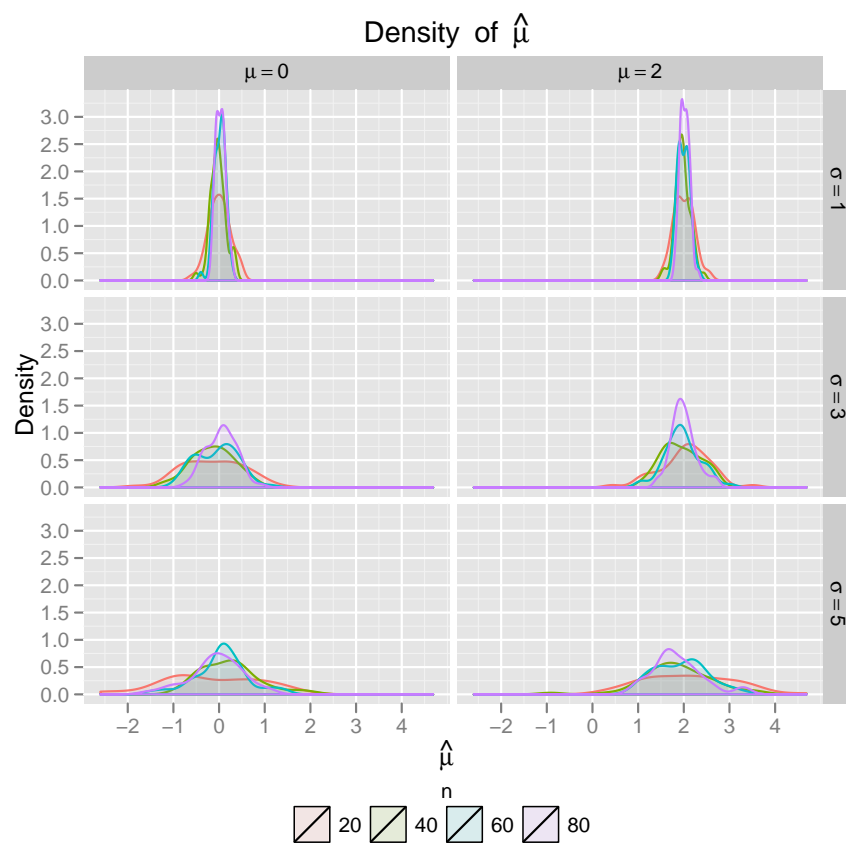
32	hat(sigma[hat(mu)])	40	1	2	0.1593	0.1601	-0.0009	0.0181	0.0181	0.1922
33	hat(sigma[hat(mu)])	40	3	0	0.4786	0.4804	-0.0018	0.0469	0.0470	0.5916
34	hat(sigma[hat(mu)])	40	3	2	0.4664	0.4804	-0.0139	0.0447	0.0468	0.5259
35	hat(sigma[hat(mu)])	40	5	0	0.7922	0.8006	-0.0085	0.0883	0.0887	0.7302
36	hat(sigma[hat(mu)])	40	5	2	0.7946	0.8006	-0.0060	0.0929	0.0931	0.7317
37	hat(sigma[hat(mu)])	60	1	0	0.1342	0.1302	0.0040	0.0140	0.0146	0.3798
38	hat(sigma[hat(mu)])	60	1	2	0.1318	0.1302	0.0016	0.0134	0.0135	0.4854
39	hat(sigma[hat(mu)])	60	3	0	0.3952	0.3906	0.0046	0.0361	0.0364	0.3849
40	hat(sigma[hat(mu)])	60	3	2	0.3886	0.3906	-0.0019	0.0369	0.0370	0.4707
41	hat(sigma[hat(mu)])	60	5	0	0.6653	0.6509	0.0144	0.0538	0.0557	0.6465
42	hat(sigma[hat(mu)])	60	5	2	0.6646	0.6509	0.0136	0.0677	0.0691	0.9729
43	hat(sigma[hat(mu)])	80	1	0	0.1125	0.1125	0.0000	0.0073	0.0073	0.8144
44	hat(sigma[hat(mu)])	80	1	2	0.1132	0.1125	0.0007	0.0104	0.0104	0.5262
45	hat(sigma[hat(mu)])	80	3	0	0.3325	0.3375	-0.0050	0.0252	0.0257	0.4173
46	hat(sigma[hat(mu)])	80	3	2	0.3372	0.3375	-0.0003	0.0234	0.0234	0.0917
47	hat(sigma[hat(mu)])	80	5	0	0.5653	0.5625	0.0027	0.0418	0.0419	0.3240
48	hat(sigma[hat(mu)])	80	5	2	0.5577	0.5625	-0.0048	0.0437	0.0439	0.6820

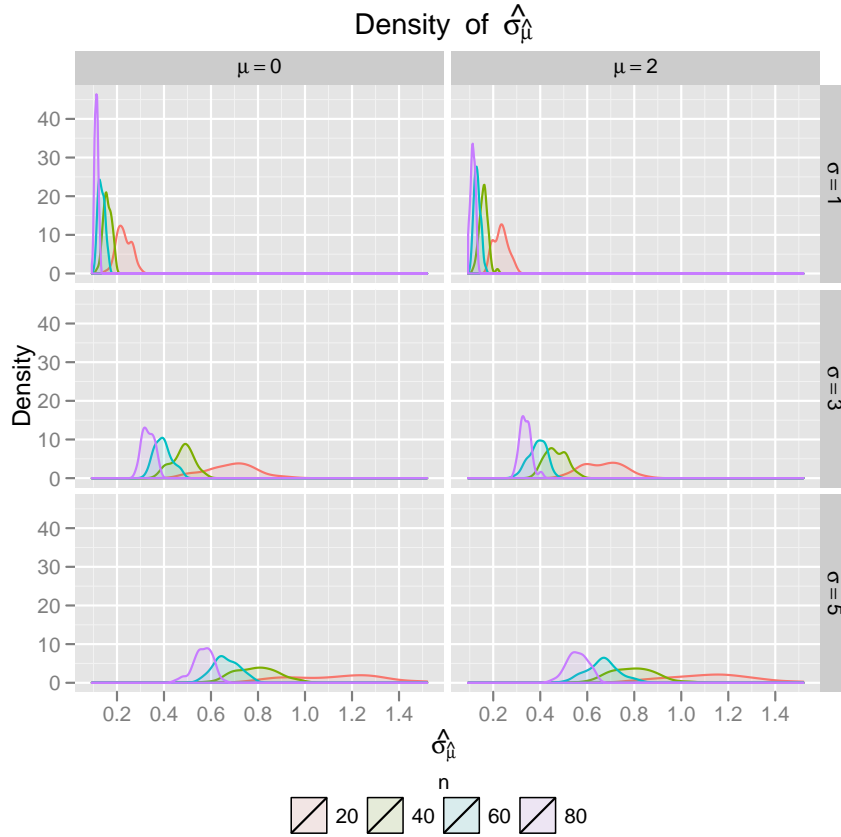
```

> plot(ezsim_basic)
> plot(ezsim_basic, "density")

```







2 Pre-simulation

There are four essential components to build an `ezsim` object. You must specify each of them to create an `ezsim` object.

1. Number of Replication m
2. Data Generating Process (dgp) : Function for generating data.
3. Parameters : dgp takes the value of parameters to generate data.
4. Estimators : It computes the estimates from the data generated by dgp.

Also there are four optional components, they are:

1. True Value (TV) : It computes the true value of estimates from dgp.
2. Display Name : It defines the display format of the name of estimators and parameters.
See `plotmath` in R manual.
3. run : If it is true, then the simulation will be ran right after the `ezsim`

4. object is created. Otherwise, you can run it manually by `run(ezsim_basic)`. Default is `TRUE`.
5. `core` : The number of CPU core to be used in the simulation. It is implemented by `foreach` and `doSNOW`. Default is 1.

If you don't specify the value of `True Value`, the value of `bias` and `rmse` will also be `NA`.

2.1 Parameters

In `ezsim`, parameters are generated by `parameterDef` object. To create a `parameterDef` object, we can use the function `createParDef`. It takes 2 arguments, `scalars` (the first argument) and `others`. `scalars` are all scalar parameters in the parameters set. Any vectors or matrix are regarded as a sequence of the same parameter. If some of your parameters are vector, matrix or other data types, you need to define it as `others`. `others` can take any data type but are fixed in the simulation.

In our example, all parameters are scalars. We can create a `parameterDef` object by:

```
> par_def <- createParDef(scalars = list(n = seq(20, 80, 20), mu = c(0,
+      2), sigma = c(1, 3, 5)))
> par_def
```

Scalar Parameters:

`$n`

```
[1] 20 40 60 80
```

`$mu`

```
[1] 0 2
```

`$sigma`

```
[1] 1 3 5
```

Others Parameters:

```
list()
```

Since we have 4 different values of n , 2 different values of μ and 3 different values of σ , there is total of $4 \times 3 \times 2 = 24$ possible combination of parameter sets. If we want to have a look of the generated parameters, we can use the function `generate`. It will return a list of parameter sets. (Only the first three will be shown in the example)

```
> generate(par_def)[1:3]
```

```
[[1]]
```

```
n=20, mu=0, sigma=1
```

```
[[2]]
```

```
n=40, mu=0, sigma=1
```

```
[[3]]
```

```
n=60, mu=0, sigma=1
```

`setScalars` and `setOthers` change the value of a `parameterDef` object. Different from `createParDef`, the parameters don't have to be stored in a list.

Example: Suppose we want to generate n sample from a bivariate normal distribution with parameter μ_1, μ_2 and a variance-covariance matrix Σ .

```
> par_def2 <- createParDef(scalars = list(mu1 = 5, mu2 = 3, n = c(10,
+ 20)), others = list(Sigma = matrix(c(1, 0.4, 0.4, 1), nrow = 2)))
> generate(par_def2)
```

```
[[1]]
Sigma :
      [,1] [,2]
[1,]  1.0  0.4
[2,]  0.4  1.0
```

```
mu1=5, mu2=3, n=10
```

```
[[2]]
Sigma :
      [,1] [,2]
[1,]  1.0  0.4
[2,]  0.4  1.0
```

```
mu1=5, mu2=3, n=20
```

2.2 Data Generating Process

The Data Generating Process generates the simulated data for `estimator` to compute the estimates. Inside this function, you can call any parameters directly. It must be a function.

In our example, the data generating process very is simple. It generate a vector of normal random variables with length n , mean μ and sd σ .

```
> dgp <- function() {
+   rnorm(n, mu, sigma)
+ }
```

```
> test(par_def, dgp, index = 1)
```

```
[1]  0.53741725  2.42495111 -0.82943977 -0.02927528  0.86855883  0.71652086
[7] -0.96449879 -1.48850268  1.72697340 -0.98152046 -0.13272934 -1.21739481
[13]  0.76860566 -0.45438887 -0.02887323 -1.19732050 -0.29381297  0.13558028
[19] -2.34612600  0.90662459
```

```
> test(par_def, dgp, index = 2)
```

```
[1]  0.23120245 -0.33812674  0.07510981  1.64745136  1.15317897 -0.50843804
[7]  0.12235509  0.28326073  1.64177493  0.71683882 -0.89201083  0.10913845
[13]  0.38262813 -1.11063359  1.58216707  0.84579952  0.84877621  1.55867923
[19]  0.02226713  0.71054968 -0.41048184 -0.08150233  0.13487307 -1.36256005
[25]  0.50068646 -0.82869989 -1.10440450 -0.28819858 -0.77566397 -0.66257286
[31] -1.38790689  0.90330925  0.51259047 -0.95404314 -0.54987508  2.25104153
[37]  0.35716337 -1.81519857 -0.71388804  1.81639821
```



```

> dgp_2 <- function() {
+   z1 <- rnorm(n)
+   z2 <- rnorm(n)
+   cbind(x1 = mu1 + z1 * Sigma[1, 1], x2 = mu2 + Sigma[2, 2] *
+         (Sigma[1, 2] * z1 + sqrt(1 - Sigma[1, 2]^2) * z2))
+ }
> test(par_def2, dgp_2)

      x1      x2
[1,] 4.175982 2.646452
[2,] 5.034060 2.388198
[3,] 6.642162 3.678985
[4,] 7.348294 4.267971
[5,] 5.658740 3.476589
[6,] 2.920541 2.159674
[7,] 5.282045 1.139881
[8,] 4.769307 1.353072
[9,] 4.552451 1.667381
[10,] 5.992440 3.394961

```

2.3 Estimators

It computes the estimates from the data generated by `dgp`. The return value of estimators must be a numeric vector. Dont forget to specify the name of estimators. You can use the `test` function to test whether the function work properly. It must be a function.

```

> estimator <- function(x) {
+   c(mean_hat = mean(x), sd_mean_hat = sd(x)/sqrt(length(x) -
+     1))
+ }
> estimator(test(par_def, dgp, index = 1))

      mean_hat sd_mean_hat
0.06138194  0.21972886

```

2.4 True Value

It computes the true value of estimates from `dgp`. The return value should have same length as the estimators. Also, the position of return value should match with estimators. Similar to `dgp`, You can call any parameters within this function. It can be a function or `NA`(bias and rmse will also be `NA`).

```

> true <- function() {
+   c(mu, sigma/sqrt(n - 1))
+ }
> test(par_def, true)

[1] 0.0000000 0.2294157

```

2.5 Display Name

It defines the display format of the name of estimators and parameters. For example, you can set the display name of "mean_hat" to "hat(mu)". See `plotmath` for details.

```
> display_name <- c(mean_hat = "hat(mu)", sd_mean_hat = "hat(sigma[hat(mu)])")
```

3 Post-simulation

3.1 Summary Table

You can create a summary table by `summary`. The default summary statistics include mean, true value, bias, standard deviation, root mean square error and p-value of Jarque-Bera test. See section 1 for example.

3.1.1 Subset of the Summary Table

You can select a subset of parameters and estimators to compute the summary statistics.

```
> summary(ezsim_basic, subset = list(estimator = "mean_hat", n = c(20,
+ 40), sigma = c(1, 3)))
```

	estimator	n	sigma	mu	Mean	TV	Bias	SD	rmse	JB_test
1	hat(mu)	20	1	0	0.0058	0	0.0058	0.2352	0.2353	0.9451
2	hat(mu)	20	1	2	2.0022	2	0.0022	0.2232	0.2232	0.7882
3	hat(mu)	20	3	0	-0.1359	0	-0.1359	0.6651	0.6788	0.7387
4	hat(mu)	20	3	2	2.0275	2	0.0275	0.5665	0.5672	0.4604
5	hat(mu)	40	1	0	-0.0208	0	-0.0208	0.1704	0.1716	0.7208
6	hat(mu)	40	1	2	1.9751	2	-0.0249	0.1682	0.1701	0.6007
7	hat(mu)	40	3	0	-0.1376	0	-0.1376	0.4712	0.4909	0.8154
8	hat(mu)	40	3	2	1.9423	2	-0.0577	0.4273	0.4312	0.4237

3.1.2 More Summary Statistics

If you want to have more summary statistics, you can set `simple=FALSE` in the argument. Then the summary statistics will also include: percentage of bias, minimum, first quartile, median, third quartile and maximum.

```
> summary(ezsim_basic, simple = FALSE, subset = list(estimator = "mean_hat",
+ n = c(20, 40), sigma = c(1, 3)))
```

	estimator	n	sigma	mu	Mean	TV	Bias	BiasPercentage	SD	rmse	Min
1	hat(mu)	20	1	0	0.0058	0	0.0058	Inf	0.2352	0.2353	-0.5979
2	hat(mu)	20	1	2	2.0022	2	0.0022	0.0011	0.2232	0.2232	1.5533
3	hat(mu)	20	3	0	-0.1359	0	-0.1359	-Inf	0.6651	0.6788	-1.8318
4	hat(mu)	20	3	2	2.0275	2	0.0275	0.0138	0.5665	0.5672	0.4349
5	hat(mu)	40	1	0	-0.0208	0	-0.0208	-Inf	0.1704	0.1716	-0.5051
6	hat(mu)	40	1	2	1.9751	2	-0.0249	-0.0124	0.1682	0.1701	1.5351
7	hat(mu)	40	3	0	-0.1376	0	-0.1376	-Inf	0.4712	0.4909	-1.2095
8	hat(mu)	40	3	2	1.9423	2	-0.0577	-0.0288	0.4273	0.4312	1.1081
	Q25	Median		Q75		Max	JB_test				

```

1 -0.1462  0.0085 0.1654 0.4565  0.9451
2  1.8479  1.9802 2.1524 2.5545  0.7882
3 -0.6852 -0.1726 0.3394 1.2760  0.7387
4  1.7877  2.0751 2.3946 3.5110  0.4604
5 -0.1241 -0.0356 0.0650 0.3501  0.7208
6  1.8749  1.9725 2.0652 2.4392  0.6007
7 -0.4693 -0.0961 0.1694 0.8674  0.8154
8  1.6358  1.9381 2.2809 2.7131  0.4237

```

3.1.3 Customize the Summary Statistics

You can choose a subset of summary statistics by specifying value in `stat`. Also you can define your own summary statistics. `value_of_estimator` is the value of estimator and `value_of_TV` is the value of true value.

```

> summary(ezsim_basic, stat = c("q25", "median", "q75"), Q025 = quantile(value_of_estimator,
+ 0.025), Q975 = quantile(value_of_estimator, 0.975), subset = list(estimator = "mean_hat",
+ n = c(20, 40), sigma = c(1, 3)))

```

	estimator	n	sigma	mu	Q25	Median	Q75	Q025	Q975
1	hat(mu)	20	1	0	-0.1462	0.0085	0.1654	-0.4112	0.4479
2	hat(mu)	20	1	2	1.8479	1.9802	2.1524	1.5812	2.4685
3	hat(mu)	20	3	0	-0.6852	-0.1726	0.3394	-1.1800	0.9125
4	hat(mu)	20	3	2	1.7877	2.0751	2.3946	0.9983	2.8279
5	hat(mu)	40	1	0	-0.1241	-0.0356	0.0650	-0.3036	0.3329
6	hat(mu)	40	1	2	1.8749	1.9725	2.0652	1.6408	2.2861
7	hat(mu)	40	3	0	-0.4693	-0.0961	0.1694	-1.0918	0.6867
8	hat(mu)	40	3	2	1.6358	1.9381	2.2809	1.1759	2.6777

3.2 Plotting the simulation

3.2.1 Plotting an ezsim object

The plot contains the mean, median, true value, 2.5th, 25th, 75th and 97.5th percentile of the estimator. The mean, median, true value are plotted as black, blue and red line respectively. 2.5th and 97.5th percentile form a 95% confidence bound and 25th and 75th percentile form a 50% confidence bound.

x-axis of the plot will be the parameter with the most number of value. Rest of them will be facets of the plot. Each estimator will occupy one plot. See section 1 for examples.

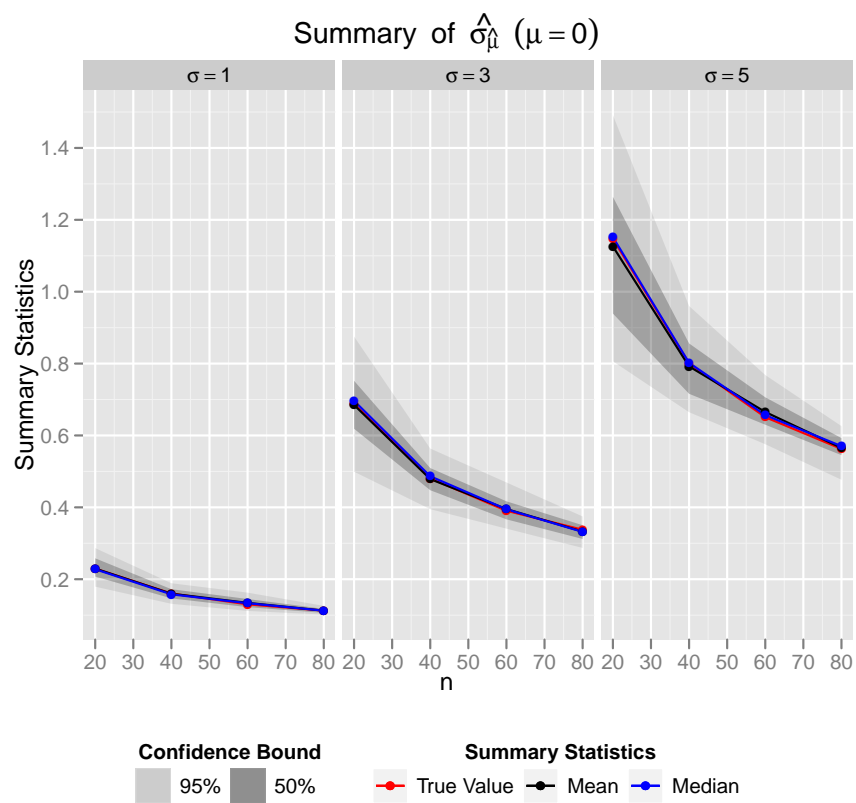
3.2.2 Subset of the Plot

The usage of `subset` is similar to `summary`. You can select a subset of `estimators` and or `parameters`.

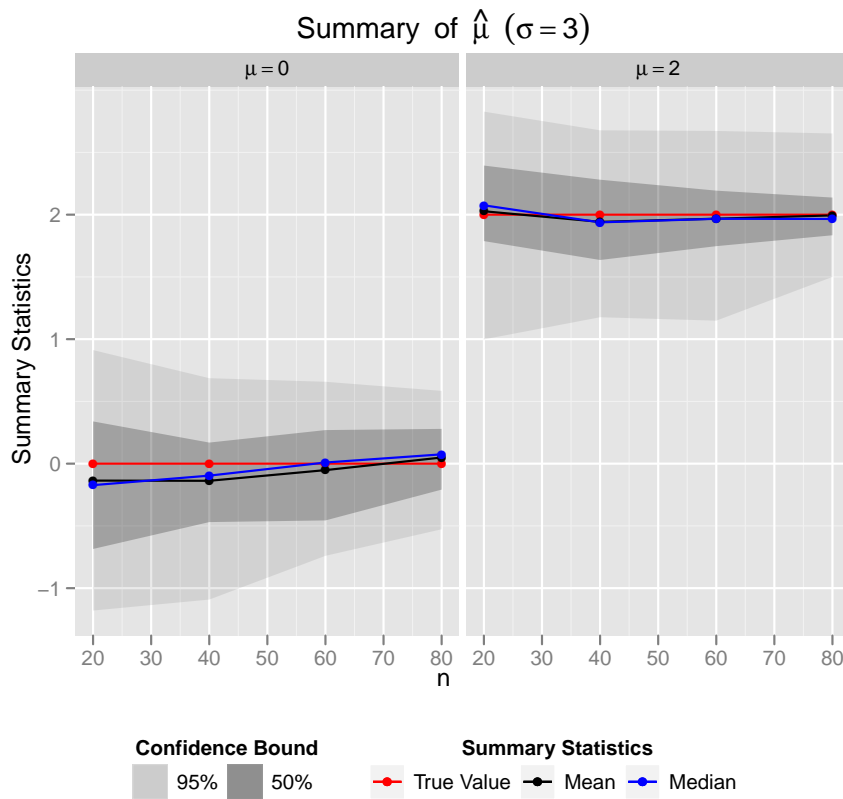
```

> plot(ezsim_basic, subset = list(estimator = "sd_mean_hat", mu = 3))

```



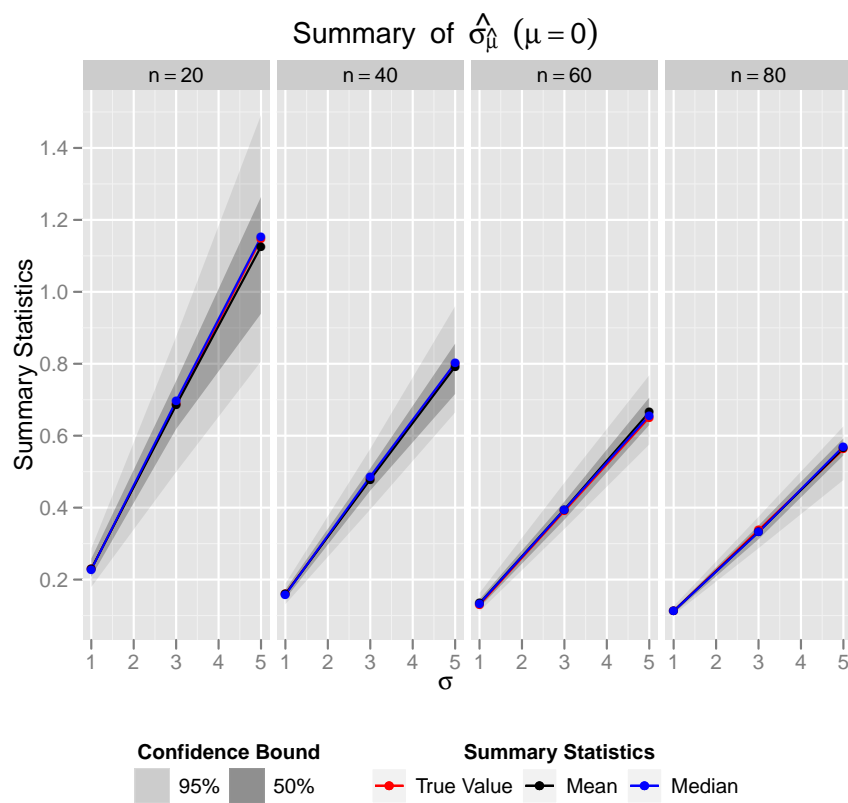
```
> plot(ezsim_basic, subset = list(estimator = "mean_hat", sigma = 3))
```



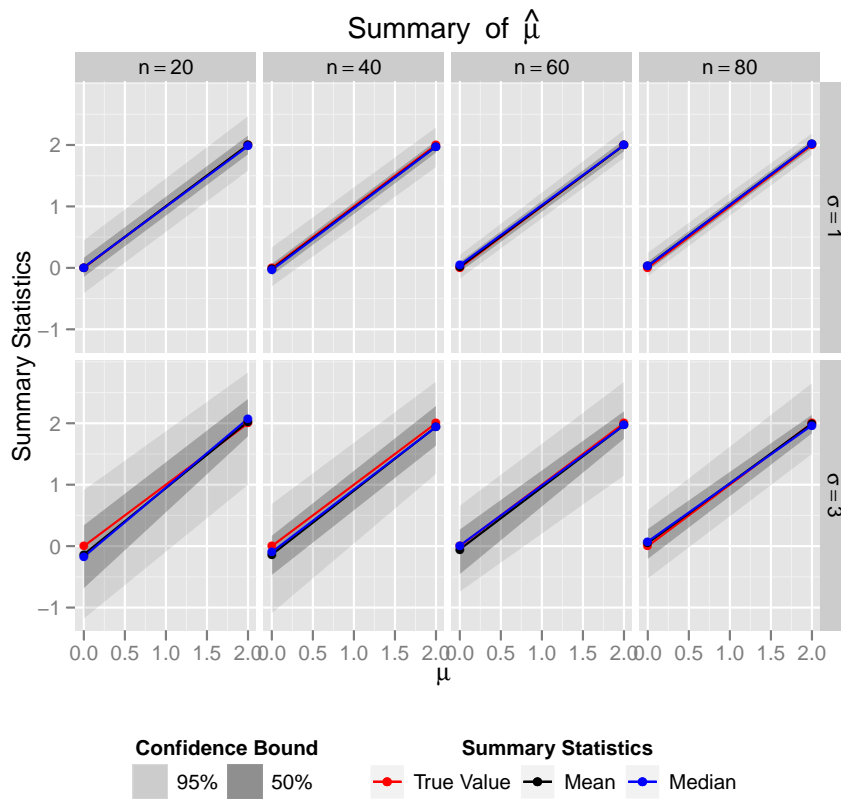
3.2.3 Parameters Priority of the Plot

The default priority of parameters is sorted by the number of value of each parameter (more to less). You can reset it by `parameter_priority`. The first parameter will have the highest priority (shown in the x-axis). You don't have to specify all parameters, the rest of them are sorted by the number of value of each of them.

```
> plot(ezsim_basic, subset = list(estimator = "sd_mean_hat", mu = 0),
+       parameters_priority = c("sigma", "n"))
```



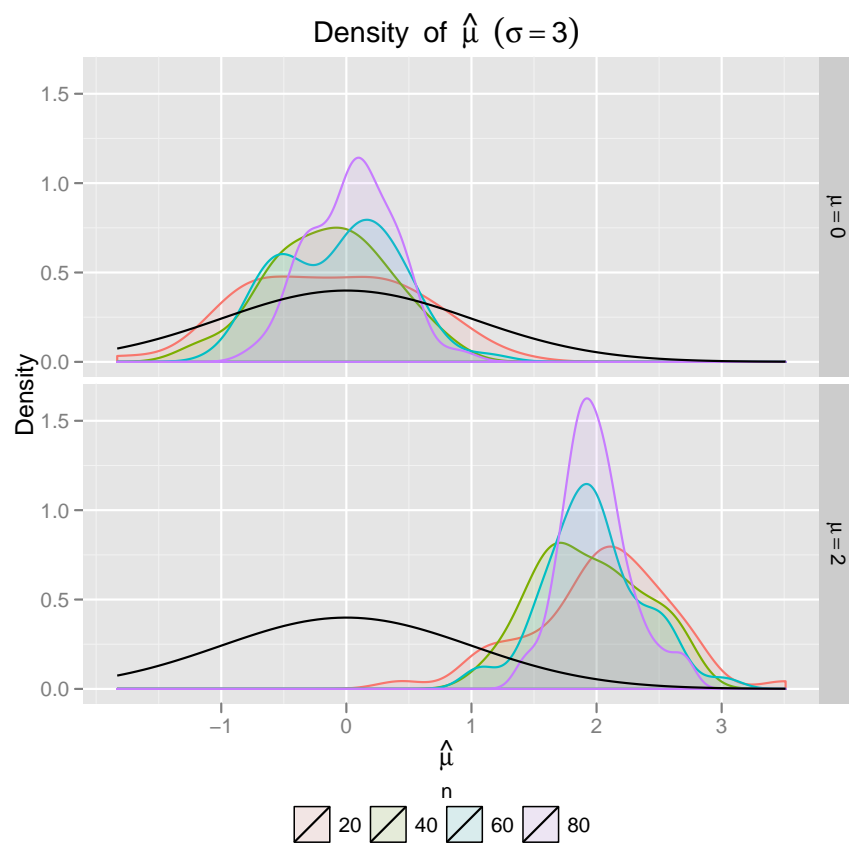
```
> plot(ezsim_basic, subset = list(estimator = "mean_hat", sigma = c(1,
+ 3)), parameters_priority = "mu")
```



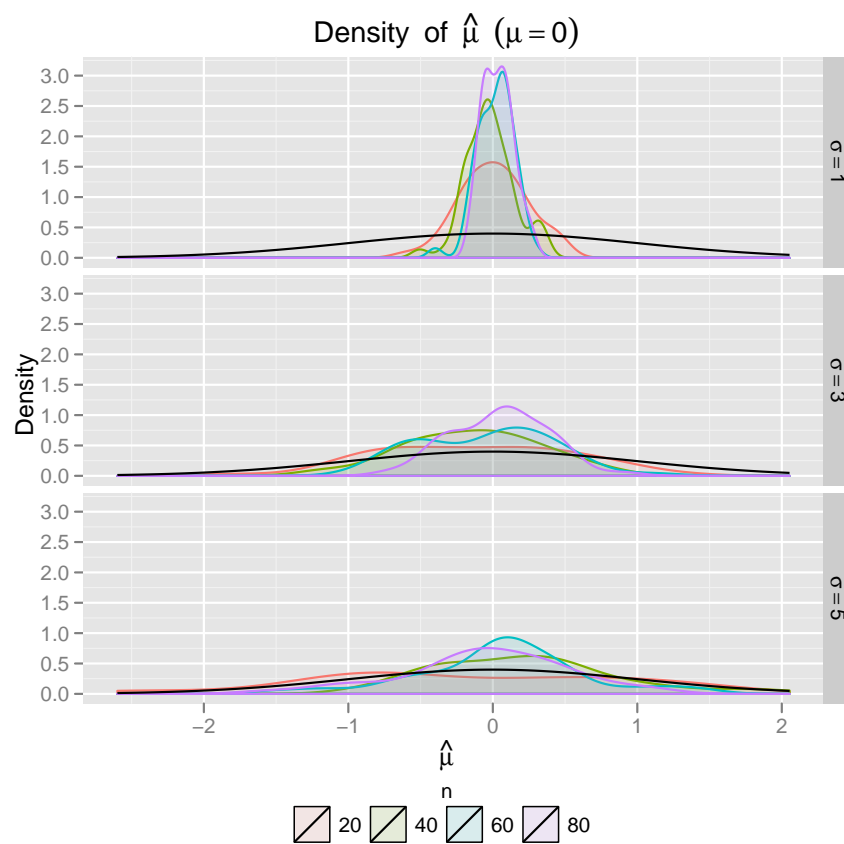
3.2.4 Density Plot

Plot the density function of the estimator. `subset` and `parameter_priority` are valid for density plot. You can specify `benchmark=dnorm` by adding a density of the standard normal distribution. `dorm` can be replaced by other density function. See section 1 for examples.

```
> plot(ezsim_basic, "density", subset = list(estimator = "mean_hat",
+     sigma = 3), parameters_priority = "n", benchmark = dnorm)
```

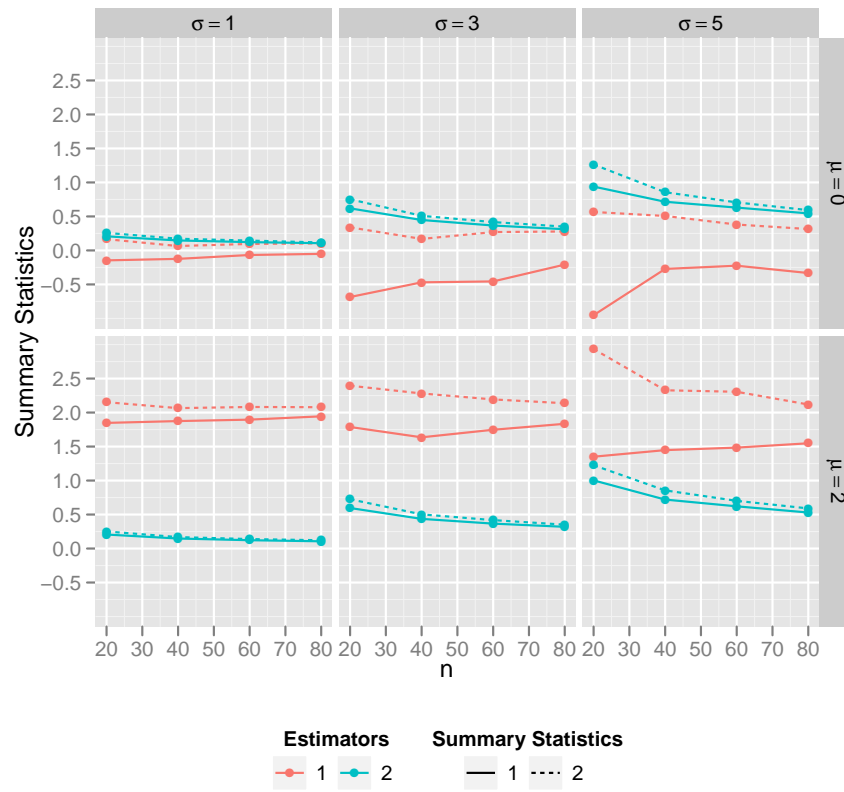


```
> plot(ezsim_basic, "density", subset = list(estimator = "mean_hat",
+      mu = 0), parameters_priority = "n", benchmark = dnorm)
```

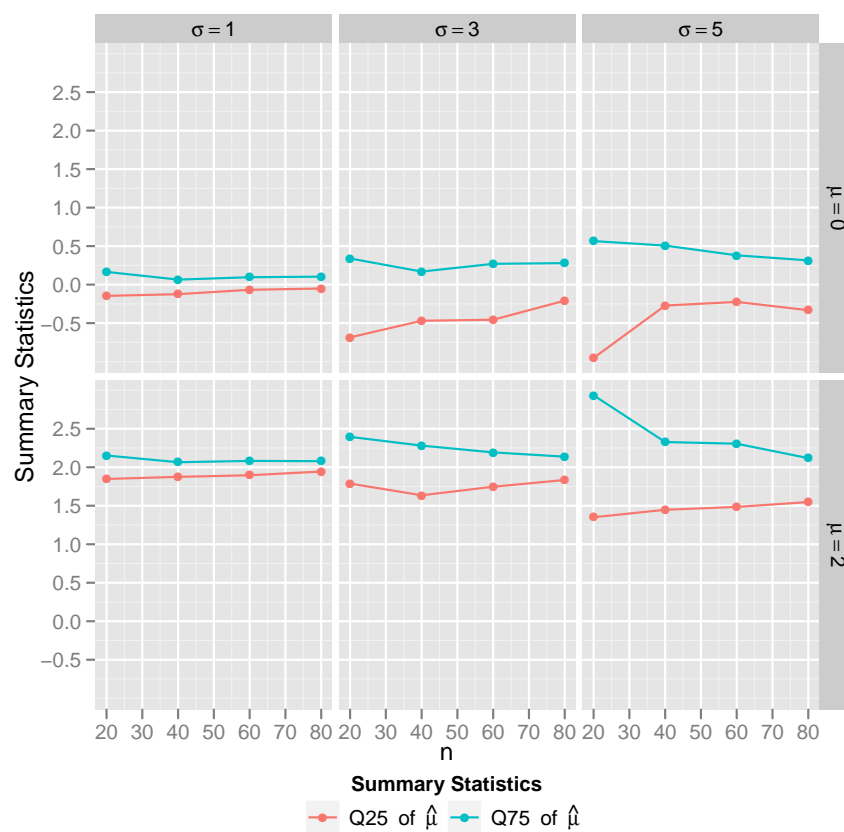



3.2.5 Plot the summary ezsim

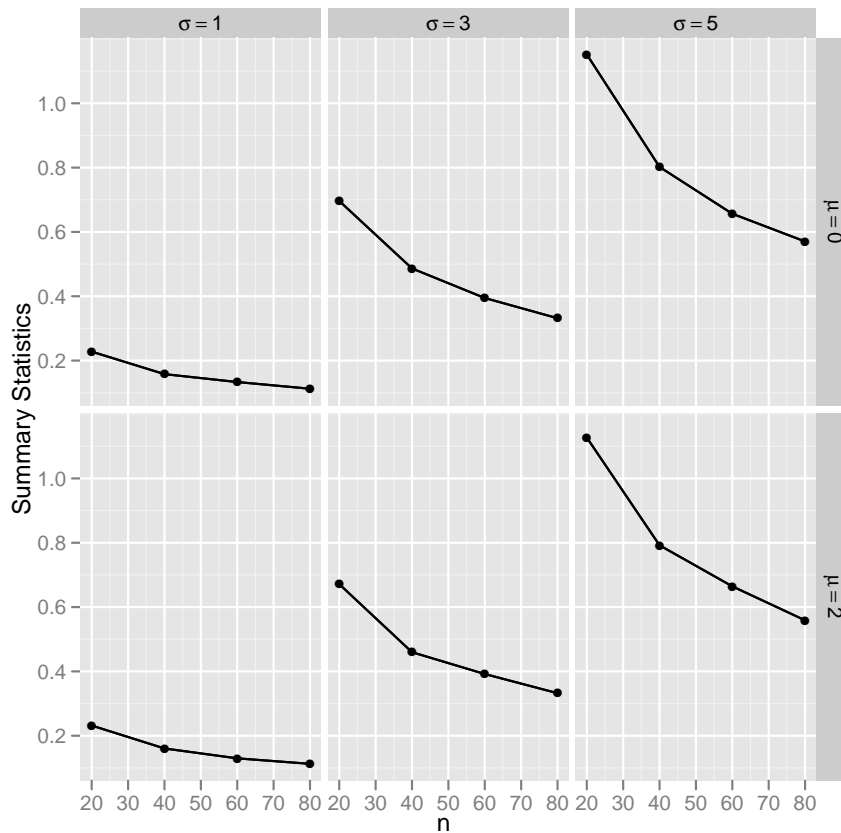
```
> plot(summary(ezsim_basic, c("q25", "q75")))
```



```
> plot(summary(ezsim_basic, c("q25", "q75"), subset = list(estimator = "mean_hat")))
```



```
> plot(summary(ezsim_basic, c("median"), subset = list(estimator = "sd_mean_hat")))
```



3.2.6 Plot the Power Function

If the estimator is an indicator of rejecting a null hypothesis(0: fail to reject null hypothesis; 1: reject null hypothesis), then we can plot the power function. A vertical line will be drawn if `null_hypothesis` is specified. The intersection of the vertical line(value of null hypothesis) and the power function is the size of the test. The following example shows the power function of testing whether the coefficient of a linear model is larger than one with t-test and z-test.

```
> ez_powerfun<-ezsim(
+   m           = 100,
+   run         = TRUE,
+   core        = 1,
+   display_name = c(b="beta",es="sigma[e]^2",xs="sigma[x]^2"),
+   parameter_def = createParDef(scalars=list(xs=1,n=50,es=5,b=seq(-1,1,0.1))),
+   dgp         = function(){
+     x<-rnorm(n,0,xs)
+     e<-rnorm(n,0,es)
+     y<-b * x + e
+     data.frame(y,x)
+   },
+   estimator    = function(d){
```

```

+                                     r<-summary(lm(y~x-1,data=d))
+                                     stat<-r$coef[,1]/r$coef[,2]
+
+                                     # test whether b > 0
+                                     # level of significance : 5%
+                                     out <- stat > c(qnorm(.95), qt(0.95,df=r$df[2]))
+                                     names(out)<-c("z-test","t-test")
+                                     out
+                                     }
+ )
> plot(ez_powerfun, "powerfun", null_hypothesis = 0)

```

