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 \( x_1, \ldots, x_n \) are drawn independently from a normal distribution with mean \( \mu \) and standard deviation \( \sigma \). We want to know how the sample size \( n \), mean \( \mu \) and standard deviation \( \sigma \) 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. \( \mu \) takes value from 0.2. \( \sigma \) takes value from 1, 3, 5.

```r
> library(ezsim)
> ezsim_basic<-ezsim(
+ m = 50,
+ run = TRUE,
+ 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<-summary(ezsim_basic)
> head(summary_ezsim_basic,16)

<table>
<thead>
<tr>
<th>estimator</th>
<th>n</th>
<th>sigma</th>
<th>mu</th>
<th>Mean</th>
<th>TV</th>
<th>Bias</th>
<th>SD</th>
<th>rmse</th>
<th>BiasPercentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>0</td>
<td>0.0299</td>
<td>0.0299</td>
<td>0.2338</td>
<td>0.2357</td>
<td>Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>1</td>
<td>0.678</td>
<td>0.678</td>
<td>0.6636</td>
<td>0.6670</td>
<td>-0.0095</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>2</td>
<td>1.9387</td>
<td>2.0143</td>
<td>1.9298</td>
<td>1.9299</td>
<td>0.0067</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>0.1364</td>
<td>0.1364</td>
<td>0.9308</td>
<td>0.9407</td>
<td>-Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>2.0190</td>
<td>2.0190</td>
<td>2.0190</td>
<td>2.0190</td>
<td>0.0095</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>0</td>
<td>0.096</td>
<td>0.096</td>
<td>0.1796</td>
<td>0.1799</td>
<td>-Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>1</td>
<td>2.0190</td>
<td>2.0190</td>
<td>2.0190</td>
<td>2.0190</td>
<td>0.0095</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>3</td>
<td>0</td>
<td>0.1201</td>
<td>0.1201</td>
<td>0.5126</td>
<td>0.5265</td>
<td>-Inf</td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>2</td>
<td>1.9199</td>
<td>2.0190</td>
<td>1.9199</td>
<td>1.9199</td>
<td>-Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>0</td>
<td>0.0469</td>
<td>0.0469</td>
<td>0.8682</td>
<td>0.8695</td>
<td>-Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>5</td>
<td>1.9183</td>
<td>2.0190</td>
<td>1.9183</td>
<td>1.9183</td>
<td>-Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>60</td>
<td>0</td>
<td>0.0389</td>
<td>0.0389</td>
<td>0.1452</td>
<td>0.1504</td>
<td>Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>60</td>
<td>1</td>
<td>1.9828</td>
<td>2.0190</td>
<td>1.9828</td>
<td>1.9828</td>
<td>-Inf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>60</td>
<td>3</td>
<td>0</td>
<td>0.0024</td>
<td>0.0024</td>
<td>0.3946</td>
<td>0.3946</td>
<td>Inf</td>
<td></td>
</tr>
<tr>
<td>hat(mu)</td>
<td>60</td>
<td>2</td>
<td>2.0755</td>
<td>2.0190</td>
<td>2.0755</td>
<td>2.0755</td>
<td>0.0378</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> plot(ezsim_basic)
> plot(ezsim_basic,"density")
```
Summary of $\hat{\mu}$

<table>
<thead>
<tr>
<th>$\sigma = 1$</th>
<th>$\sigma = 3$</th>
<th>$\sigma = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary Statistics</td>
<td>Mean</td>
<td>Median</td>
</tr>
</tbody>
</table>

Confidence Bound | 95% | 50% |
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. number_of_workers : The number of CPU core to be used in the simulation.

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, selection (the first argument) and banker. selection are parameters may vary in the parameters set. Any vectors or matrix are regarded as a sequence of the same parameter. banker are fixed parameters in the parameters set. It can be any data type.

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

```r
> par_def<-createParDef(selection=list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5)))
> par_def
Selection Parameters:
$n
[1] 20 40 60 80

$mu
[1] 0 2

$sigma
[1] 1 3 5

Banker Parameters:
list()
```

Since we have 4 different values of n, 2 different values of \( \mu \) and 3 different values of \( \sigma \), 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)

```r
> 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
```

setSelection and setBanker change the value of a parameterDef object. Different from createParDef, the parameters don't have to be store in a list.

Example: Suppose we want to generate \( n \) sample from a bivariate normal distribution with parameter \( \mu_1, \mu_2 \) and a variance-covariance matrix \( \Sigma \).
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 \( \mu \) and sd \( \sigma \).

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

> evalFunctionOnParameterDef(par_def,dgp,index=1)

[1] 1.09163329 -0.12732104 0.90500054 0.71105309 0.03198997 0.06962448
[7] 0.88976157 -1.08826694 -0.08457396 1.11738408 -1.11814315 -0.48246969
[13] -1.74319570 0.86629124 0.01853387 0.31468978 1.22678791 0.91200281
[19] 0.19367642 0.44374683

> evalFunctionOnParameterDef(par_def,dgp,index=2)

[1] -0.02931632 0.35926976 0.93921957 0.24636143 -0.30217446 0.27336888
[7] 0.22112788 -2.02915775 0.66936042 -1.06527117 0.06136985 1.32876292
[13] -0.38135852 -0.33175638 -0.30250330 -1.58018363 1.24221369 0.60834922
[19] 0.81693055 0.29709227 0.90894098 2.23569332 -0.69198998 -1.52226422
[25] 0.82466679 0.63093696 -0.95528778 -1.04701992 1.73963681 0.01681215
[31] -1.52461818 -0.58453456 0.22541851 -0.81015739 -0.22136172 0.23686525
[37] -0.24824754 2.08124703 0.87418554 -1.22408171
```

```r
> 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 ))
+
> evalFunctionOnParameterDef(par_def2,dgp_2)
   x1       x2
[1,] 4.966375 4.007011
[2,] 4.934019 3.311238
[3,] 3.994684 3.842224
[4,] 5.275246 4.379373
[5,] 4.077053 2.213060
[6,] 4.433213 1.522050
[7,] 7.456260 2.376825
[8,] 6.089728 2.008078
[9,] 4.987722 2.652864
[10,] 5.667685 3.296310
>
2.3 Estimators
It computes the estimates from the data generated by dgp. The return value of estimators must be a numeric vector. Don’t forget to specify the name of estimators. You can use the evalFunctionOnParameterDef 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(evalFunctionOnParameterDef(par_def,dgp,index=1))
   mean_hat       sd_mean_hat
[1,] -0.1128466 0.2269429

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))
> evalFunctionOnParameterDef(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.

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

<table>
<thead>
<tr>
<th>estimator</th>
<th>n</th>
<th>sigma</th>
<th>mu</th>
<th>Mean</th>
<th>TV Bias</th>
<th>SD</th>
<th>rmse</th>
<th>BiasPercentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0.0299</td>
<td>0</td>
<td>0.0299</td>
<td>0.2338</td>
<td>0.2357</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>1</td>
<td>2</td>
<td>1.9810</td>
<td>2</td>
<td>-0.0190</td>
<td>0.2525</td>
<td>0.2532</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>3</td>
<td>0</td>
<td>0.0678</td>
<td>0</td>
<td>0.0678</td>
<td>0.6636</td>
<td>0.6670</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>3</td>
<td>2</td>
<td>1.9387</td>
<td>2</td>
<td>-0.0613</td>
<td>0.6373</td>
<td>0.6402</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>-0.0096</td>
<td>0</td>
<td>-0.0096</td>
<td>0.1796</td>
<td>0.1799</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>1</td>
<td>2</td>
<td>2.0190</td>
<td>2</td>
<td>0.0190</td>
<td>0.1593</td>
<td>0.1604</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>3</td>
<td>0</td>
<td>-0.1201</td>
<td>2</td>
<td>-0.1201</td>
<td>0.5126</td>
<td>0.5265</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>3</td>
<td>2</td>
<td>1.9199</td>
<td>2</td>
<td>-0.0801</td>
<td>0.5105</td>
<td>0.5167</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>estimator</th>
<th>n</th>
<th>sigma</th>
<th>mu</th>
<th>Mean</th>
<th>TV Bias</th>
<th>SD</th>
<th>rmse</th>
<th>BiasPercentage</th>
<th>Q25</th>
<th>Median</th>
<th>Q75</th>
<th>Max</th>
<th>JB_test</th>
</tr>
</thead>
<tbody>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0.0299</td>
<td>0</td>
<td>0.0299</td>
<td>0.2338</td>
<td>0.2357</td>
<td>-0.1612</td>
<td>0.0147</td>
<td>0.1969</td>
<td>0.6353</td>
<td>0.7354</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>1</td>
<td>2</td>
<td>1.9810</td>
<td>2</td>
<td>-0.0190</td>
<td>0.2525</td>
<td>0.2532</td>
<td>1.8071</td>
<td>2.0282</td>
<td>2.1837</td>
<td>2.3837</td>
<td>0.0437</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>3</td>
<td>0</td>
<td>0.0678</td>
<td>0</td>
<td>0.0678</td>
<td>0.6636</td>
<td>0.6670</td>
<td>-0.2427</td>
<td>0.0885</td>
<td>0.5906</td>
<td>1.2132</td>
<td>0.4039</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>20</td>
<td>3</td>
<td>2</td>
<td>1.9387</td>
<td>2</td>
<td>-0.0613</td>
<td>0.6373</td>
<td>0.6402</td>
<td>1.6421</td>
<td>1.9827</td>
<td>2.3108</td>
<td>3.3743</td>
<td>0.0160</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>-0.0096</td>
<td>0</td>
<td>-0.0096</td>
<td>0.1796</td>
<td>0.1799</td>
<td>-0.1492</td>
<td>0.0105</td>
<td>0.1172</td>
<td>0.3199</td>
<td>0.5343</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>1</td>
<td>2</td>
<td>2.0190</td>
<td>2</td>
<td>0.0190</td>
<td>0.1593</td>
<td>0.1604</td>
<td>1.9390</td>
<td>2.0390</td>
<td>2.1150</td>
<td>2.3437</td>
<td>0.7997</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>3</td>
<td>0</td>
<td>-0.1201</td>
<td>2</td>
<td>-0.1201</td>
<td>0.5126</td>
<td>0.5265</td>
<td>-0.4511</td>
<td>-0.0925</td>
<td>0.1980</td>
<td>0.7946</td>
<td>0.6315</td>
</tr>
<tr>
<td>hat(mu)</td>
<td>40</td>
<td>3</td>
<td>2</td>
<td>1.9199</td>
<td>2</td>
<td>-0.0801</td>
<td>0.5105</td>
<td>0.5167</td>
<td>1.6301</td>
<td>1.9728</td>
<td>2.2067</td>
<td>3.2630</td>
<td>0.9848</td>
</tr>
</tbody>
</table>
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.

```r
> 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)))
```

<table>
<thead>
<tr>
<th>estimator</th>
<th>n</th>
<th>sigma</th>
<th>mu</th>
<th>Q25</th>
<th>Median</th>
<th>Q75</th>
<th>Q025</th>
<th>Q975</th>
</tr>
</thead>
<tbody>
<tr>
<td>hat(mu) 20</td>
<td>1</td>
<td>0</td>
<td>-0.1612</td>
<td>0.0147</td>
<td>0.1969</td>
<td>-0.3578</td>
<td>0.4207</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 20</td>
<td>1</td>
<td>2</td>
<td>1.8071</td>
<td>2.0282</td>
<td>2.1837</td>
<td>1.4959</td>
<td>2.3318</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 20</td>
<td>3</td>
<td>0</td>
<td>-0.2427</td>
<td>0.0885</td>
<td>0.5906</td>
<td>-1.3746</td>
<td>1.0568</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 20</td>
<td>3</td>
<td>2</td>
<td>1.6421</td>
<td>1.9827</td>
<td>2.3108</td>
<td>0.6669</td>
<td>3.0554</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 40</td>
<td>1</td>
<td>0</td>
<td>-0.1492</td>
<td>0.0105</td>
<td>0.1172</td>
<td>-0.3865</td>
<td>0.3049</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 40</td>
<td>1</td>
<td>2</td>
<td>1.9390</td>
<td>2.0390</td>
<td>2.1150</td>
<td>1.7179</td>
<td>2.3157</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 40</td>
<td>3</td>
<td>0</td>
<td>-0.4511</td>
<td>-0.0925</td>
<td>0.1980</td>
<td>-1.1463</td>
<td>0.7683</td>
<td></td>
</tr>
<tr>
<td>hat(mu) 40</td>
<td>3</td>
<td>2</td>
<td>1.6301</td>
<td>1.9728</td>
<td>2.2067</td>
<td>1.0408</td>
<td>2.8636</td>
<td></td>
</tr>
</tbody>
</table>

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.

```r
> plot(ezsim_basic, subset=list(estimator="sd_mean_hat",mu=3))
```
Summary of $\hat{\sigma}_\mu$ ($\mu = 0$)

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>Summary Statistics</th>
<th>95% Confidence Bound</th>
<th>50% Confidence Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mean: 0.49 Median: 0.50 True Value: 0</td>
<td># grey shade</td>
<td># grey shade</td>
</tr>
<tr>
<td>3</td>
<td>Mean: 1.49 Median: 1.50 True Value: 0</td>
<td># grey shade</td>
<td># grey shade</td>
</tr>
<tr>
<td>5</td>
<td>Mean: 2.49 Median: 2.50 True Value: 0</td>
<td># grey shade</td>
<td># grey shade</td>
</tr>
</tbody>
</table>

> `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. dnorm can be replaced by other density function. See section 1 for examples.

```r
> plot(ezsim_basic,"density",subset=list(estimator="mean_hat",sigma=3),parameters_priority="n",benchmark=dnorm)
```
Density of $\hat{\mu}$ ($\sigma = 3$)

> 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")))
\[ \sigma = 1 \quad \sigma = 3 \quad \sigma = 5 \]

Summary Statistics

\[ \mu = 0 \quad \mu = 2 \]

\[ > \text{plot(summary(ezsim_basic,c("median"),subset=list(estimator="sd_mean_hat")))} \]
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.

```r
> ez_powerfun<-ezsim(
+   m = 100,
+   run = TRUE,
+   display_name = c(b="beta",es="sigma[e]^2",xs="sigma[x]^2"),
+   parameter_def = createParDef(selection=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(ex_powerfun,"powerfun",null_hypothesis=0)

Power Function (true_value = ., $\sigma_x^2 = 1$, $n = 50$, $\sigma_e^2 = 5$)

Estimators - Mean of z.test - Mean of t.test