Normality Assumption in Large Data Settings

Abstract

The normality assumption is one of the common misunderstandings in all statistics concepts. In linear regression, the assumption requiring a normal distribution only applicable to the disturbance term due to the nature of randomness, which is also called errors. For satisfying this assumption, various transformation approaches are invented to both dependent and independent variables. However, Schmidt in his research claims such transformations are usually unnecessary but may bias the final estimation (2018). In this study, I would design an experiment to verify the reproducibility and replicability of Schmidt's project. Also, answer the question, whether the normality assumption is still predominant in large data settings.

Introduction and Research Questions

The normality assumption is one of the common misunderstandings in all statistics concepts. In linear regression, the assumption requiring a normal distribution only applicable to the disturbance term, which is also called errors. From the Gauss-Markov theorem, the ordinary least squares ensure the lowest sampling variances with a sort of best linear unbiased (BLUE) estimators under several assumptions, including there is no correlation among regressors, errors from the linear regression model are uncorrelated with equal variances, and the expectation value of zero (Henderson, 1975). Although the classic linear regression model (CLRM) does not include assumptions concerning the disturbances' distribution, Due to the nature inherent in randomness and statistical testing purpose, the normality for the error term is highly desired.

Despite the extensive research about sample size and disturbance that have proved that increasing sample size ensures the validity of hypothesis testing (Ghasemi & Zahediasl,

2012), whether we need the normality assumption in large data settings still open to debate.

In 2018, Schmidt in his research claims that there are a bunch of studies that apply arbitrary transformations to the regression outcome, which is unnecessary and problematic (Schmidt & Finan, 2018). Here, I could briefly introduce his logic behind this conclusion. Set a bivariate linear regression as follow:

$$\hat{y} = \widehat{\beta_0} + \widehat{\beta_1}x + \varepsilon$$

In this equation, $\hat{y}$ stands for the dependent variable from the sample, x is the regressor used for building the linear model, estimator $\widehat{\beta_0}$ is the intercept, which equals to $\bar{y}$ when $x = 0$, $\widehat{\beta_1}$ is the slope which equals to $\frac{\widehat{y_{x+\delta}} - \widehat{y_x}}{\delta}$ with δ indicating infinite minors, and ε represents the disturbances that ideally follow a normal distribution. Use $f()$ to denote the transformation function they applied to the outcome carriable $\hat{y}$, the equation becomes:

$$f(\hat{y}) = f(\hat{y} = \widehat{\beta_0} + \widehat{\beta_1}x + \varepsilon)$$

And the estimated slope $\widehat{\beta_1}$ becomes $\widehat{\beta_{1,t}} = \frac{f(\widehat{y_{x+\delta}}) - f(\widehat{y_x})}{\delta}$, with t indicating "transformed". We could easily conclude that $\widehat{\beta_{1,t}} \neq \widehat{\beta_1}$ unless $f(x) = x + c$.

However, this statement is rooted in the assumption that the underline data generation process is clear. In other words, the population estimator β_1 is given, which is usually violated in the realistic. Since the $\widehat{\beta_1}$ is not equivalent to the population slope β_1, $\widehat{\beta_{1,t}} \neq \widehat{\beta_1}$ does not bring any troubles to our modeling. No matter what kind of transformation we apply to the dependent or independent variables, the Gauss-Markov confirms the efficiency of the evaluated parameters. But the key is, we need to make inferences from the sampling data to the population, we want to test the significance of the estimated regressor to answer our research questions. Therefore, the most important things we need to examine are: (1) Whether the assumption about the linear relationship among

dependent and independent variables holds? (2) Whether the evaluated value is significant enough to reject the none-hypothesis? Thomas Lumley makes effort to confirm the validity of the t-test and linear regression for any outcome distributions (Lumley et al., 2002) , which is an inspiring study but I would not elaborate too much here.

Also, Schmidt proposed another theory that the normality assumption is not necessary to the big data settings. For testing this statement and answering the research question that what is the role of normality assumption for large size of data modeling, in this study, 16 illustrative datasets are simulated from 4*4 different scenarios to address how different distribution of errors would affect our regression modeling.

Methodology

In this section, I will illustrate my framework for data simulating, regression modeling, and model diagnostics in detail.

Data simulating

To examine the impacts of disturbance from different distribution, I generate 4 sets of errors, following : (1) the standard normal distribution, (2) a uniform distribution with minimum as -1 and maximum as 1, (3) a beta distribution with $\alpha = 2$ and $\beta = 10$, (4) heterocedasticity error which composed of 3 independent normal distribution with equal mean but different variance. Fig.1 demonstrates the normality of 10000 descriptive errors from those four scenarios. It shows the obvious deviations in scenarios 2, 3, and 4.

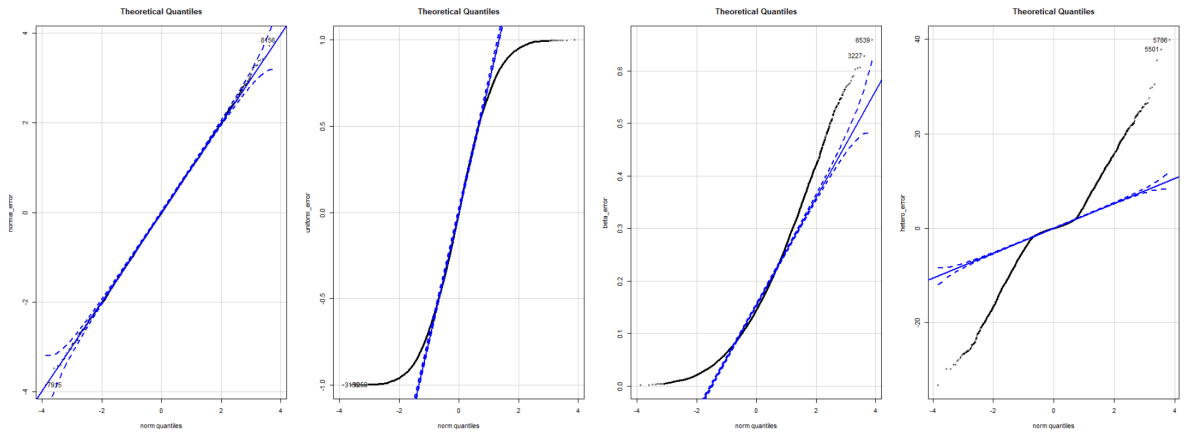


Figure 1. The normality of four errors distribution

For completely investigating the impacts from errors distribution, I also simulate the independent variable x_i from 4 distinct scenarios, including: (1) a uniform distribution with minimum as -20 and maximum as 20, (2) a normal distribution with mean as 50 and variance as 25, (3) the square of a normal distribution with mean as 50 and variance as 25, (4) a normal distribution with mean as 50 and variance as 5. Fig.2 depicts the distribution of 4 simulated independent variables by histogram.

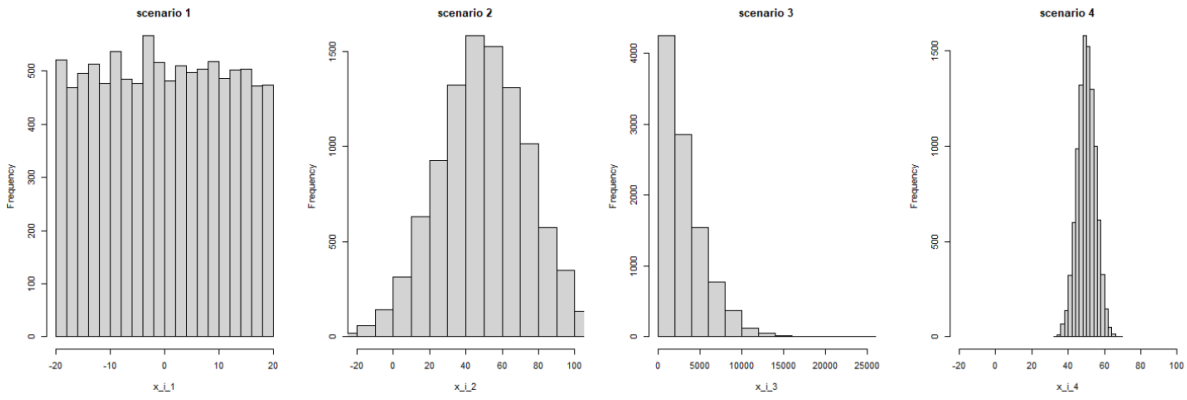


Figure 2. The distribution of simulated variables x_i from 4 scenarios

In the end, the dependent variable y_i is yield by the equation $y_i = 20 + \beta_1 x_{i,s} + \varepsilon_k$. The slope parameter of the population β_1 is randomly picked by a uniform distribution $U(-100,100)$. Since both x_i and ε_k come from 4 different scenarios , by the combination of them, a total of 16 types of y_i is simulated with each has 10000 observations.

Regression modeling and model diagnostics

For mimicking the stochastic sampling approach, I randomly select data from the simulated population with sample size n = 3, 10, 100, 500, 1000, 2000 for 3000 times. And then perform the bivariate linear model on it the get the sample estimator $\widehat{\beta_1}$. Again, our predominant purpose is to test the significance of the linear regressor. Therefore, I use the t-test to get the p-value between the sample estimator $\widehat{\beta_1}$ with the underlying population slope β_1 as the measure metrics. And the final accuracy ratio is evaluated by the equation:

$$accuracy_n = 1 - \frac{\sum_{i=1}^{k}\{p\_value \leq 0.05\}}{k}$$

k is the iteration round for the stochastic sampling approach, which equals 3000 in this study. And the p-value is estimated by equation:

$$p\_value = 2 * \min\{\Pr\left(T \geq \frac{\widehat{\beta_1} - \beta_1}{SE}\right), \Pr\left(T \leq \frac{\widehat{\beta_1} - \beta_1}{SE}\right)\}$$

Therefore, the accuracy ratio implies, what is the percentage of the evaluated estimator $\widehat{\beta_1}$ not significantly different from the underlying population slope β_1. Thus, a higher accuracy ratio is desirable.

First four scenarios are, $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim U(-20,20)$ and $\varepsilon \sim N(0,1)$, $U(-1,1)$, $B(2,10)$, and heteroscedasticity, respectively. As shown in Fig.3, the normality of error distribution does make some help for inference aspects. Conversely, the benefit from the large size of sampling seems to correlate with the distribution of the error term.

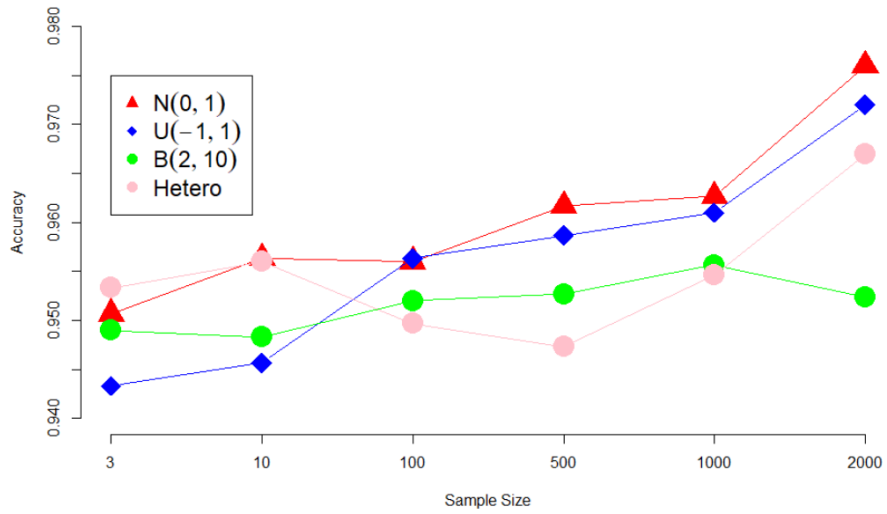


Figure3. Accuracy ratio for $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim U(-20,20)$

The next scenarios are $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim N(50,25)$ and $\varepsilon \sim N(0,1)$, $U(-1,1)$, $B(2,10)$, and heteroscedasticity, respectively. From Fig. 4, as the sampling size increasing, the accuracy ratio of heteroscedasticity regression drops dramatically. The regression with normally distributed regression also overperformance than other 3 models in those scenarios.

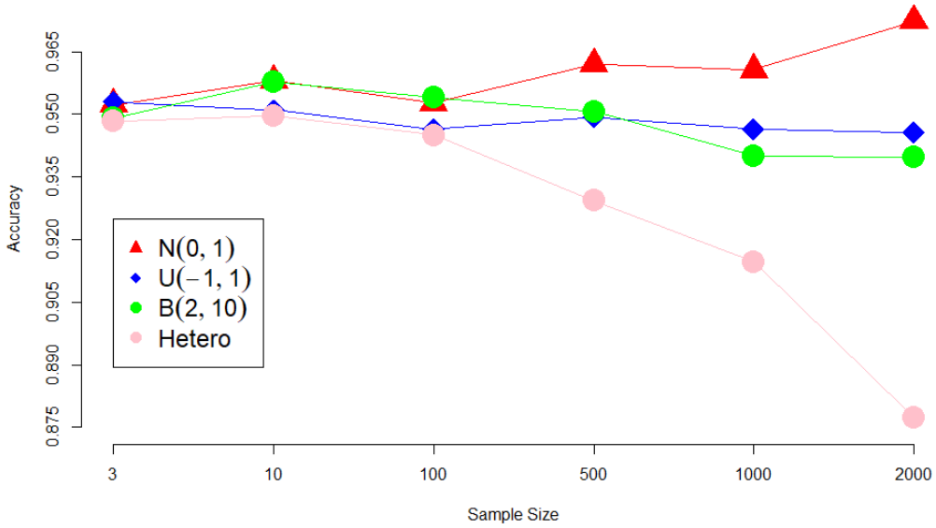


Figure4. Accuracy ratio for $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim N(50,25)$

The next scenarios are $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim N(50,25)^2$. The independent variable is highly positively skewed due to the quadratic nature. From Fig. 5, as the sampling size increasing, the accuracy ratio of uniform errors regression drops dramatically.

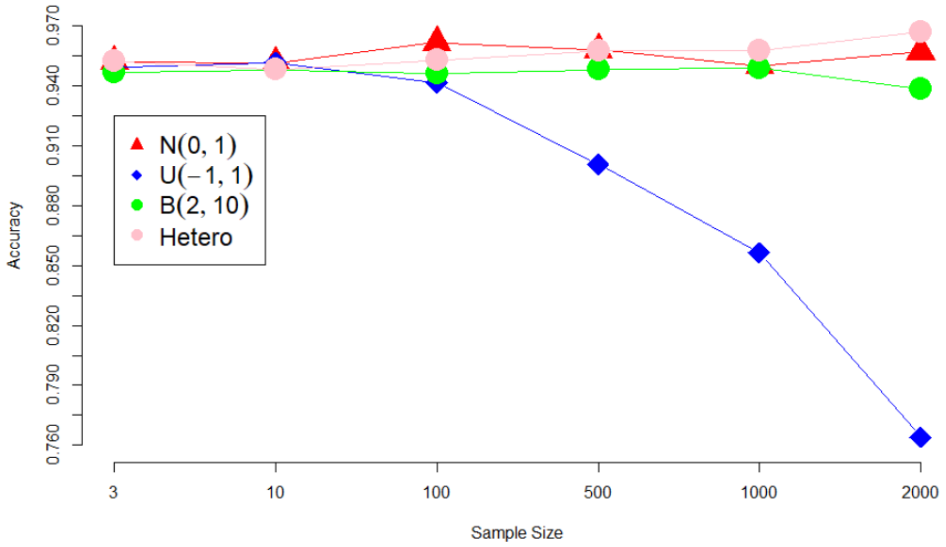


Figure5. Accuracy ratio for $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim N(50,25)^2$

The last four scenarios are $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim N(50,5)$. All four types of regression perform very well with near 95% accuracy. With the sampling size increasing, the highest accuracy from the normally distributed errors regression reaches 97%.

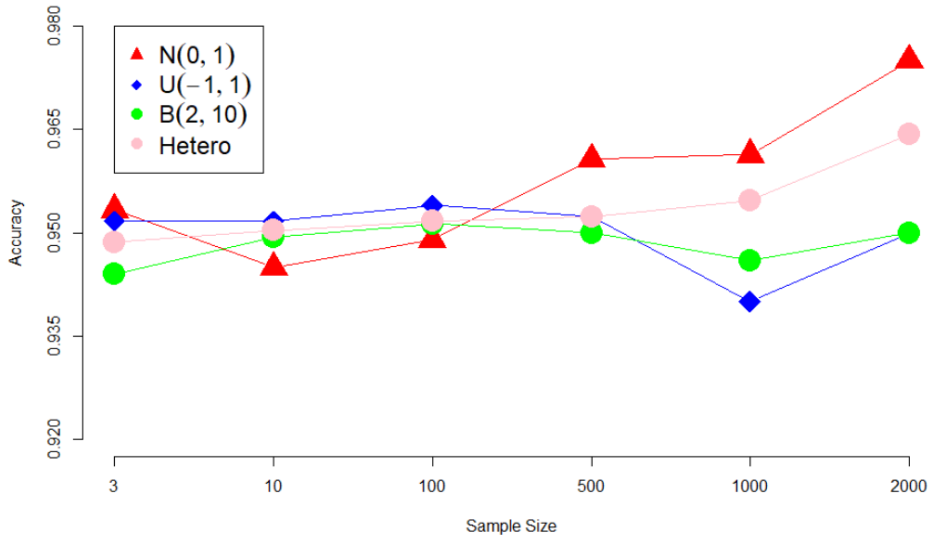


Figure6. Accuracy ratio for $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ with $x_i \sim N(25,5)$

Conclusion and future work

In this project, by briefly simulated errors with 4 different types of distributions, we could address the impacts from disturbances and conclude that normality assumption in large data settings still beneficial for statistical inference. Therefore, the transformation on both independent and dependent variables for fulfilling the normality assumption is recommended for regression modeling. Again, the most predominant purpose for applying statistical modeling is making inferences from samples, so we should pay more attention to the significance of estimated parameters and whether they can answer our hypotheses. The T-test with normality assumption is just one of the multiple statistical models. you can forget about it if you have the rationale about true-parameter distribution.

On the contrary, by multiple data simulations, it is hard to find the significant advantages of enlarging the sample size. For some specific circumstances, the large number of observations makes the model prediction worse. It is not clear those situations happen occasionally or not. More experiments and mathematical derivations are required to explore the logic behind them.

More appealing distributions, such as the Poisson distribution, the exponential distribution, the gamma distribution are not included in this study. Those are also worth exploring.

Reference

Lumley, T., Diehr, P., Emerson, S. and Chen, L., 2002. The importance of the normality assumption in large public health data sets. Annual review of public health, 23(1), pp.151-169.

Ghasemi, A., & Zahediasl, S. (2012). Normality tests for statistical analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism*, *10*(2), 486–489. https://doi.org/10.5812/ijem.3505

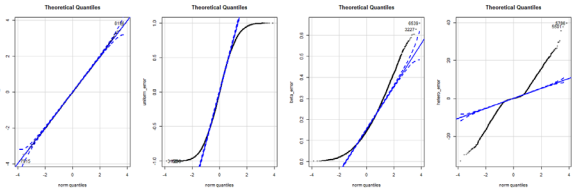
Henderson, C. R. (1975). Best Linear Unbiased Estimation and Prediction under a Selection Model. *Biometrics*, *31*(2), 423. https://doi.org/10.2307/2529430

Schmidt, A. F., & Finan, C. (2018). Linear regression and the normality assumption. *Journal of Clinical Epidemiology*, *98*, 146–151. https://doi.org/10.1016/j.jclinepi.2017.12.006

Wang, J. Z., Lu, X., & Lin, Z. (2018). *I Nformative a Ssumption in C Hannel P Runing*. *2017*, 1–11. https://doi.org/10.1146/annurev.publheath.23.100901.140546

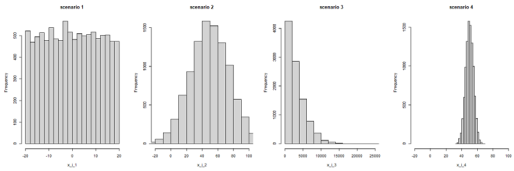
Appendix

```
library(car)
set.seed(123)
normal_error <- rnorm(10000,0,1)
uniform_error <- runif(10000,-1,1)
beta_error <- rbeta(10000,2,10)
hetero_error <- c(rnorm(3500,0,1),rnorm(3500,0,10),rnorm(1000,0,5))
par(mfrow = c(1,4))
qqPlot(normal_error,main = 'Theoretical Quantiles')
## [1] 8156 7915
qqPlot(uniform_error,main = 'Theoretical Quantiles')
## [1] 3156 1256
qqPlot(beta_error,main = 'Theoretical Quantiles')
## [1] 6539 3227
qqPlot(hetero_error,main = 'Theoretical Quantiles')
```



```
## [1] 5786 5501
```

```
set.seed(123)
x_i_1 <- runif(10000,-20,20)
x_i_2 <- rnorm(10000,50,25)
x_i_3 <- rnorm(10000,50,25)**2
x_i_4 <- rnorm(10000,50,5)
par(mfrow = c(1,4))
hist(x_i_1,main = 'scenario 1')
hist(x_i_2,xlim = c(-20,100),main = 'scenario 2')
hist(x_i_3,main = 'scenario 3')
hist(x_i_4,xlim = c(-20,100),main = 'scenario 4')
```



```
set.seed(123)

beta_1 <- runif(1,-100,100)
y1_1 <- 20 + beta_1*x_i_1 + normal_error
y1_2 <- 20 + beta_1*x_i_1 + uniform_error
y1_3 <- 20 + beta_1*x_i_1 + beta_error
y1_4 <- 20 + beta_1*x_i_1 + hetero_error
print(beta_1)
```

```
## [1] -42.4845
```

```
coverage_calculating <- function(n,y,x,beta_1,repeat_time = 3000){
 set.seed(123)
 p.value = c()
 for(i in c(1:repeat_time)){
  index <- sample(1:length(y),n)
  sample_y <- y[index]
  sample_x <- x[index]
  mod1 <- lm(sample_y~sample_x)
  t_value <- (mod1$coefficients[2] - beta_1) / coef(summary(mod1))[, "Std. Error"][2]
  p_value1 = pt(-t_value,df = n-2)
  p_value2 = pt(t_value,df = n-2)
  p.value = c(p.value,2*min(p_value1,p_value2))
 }
 acc = 1 - sum(p.value <=0.05) / repeat_time
```

```
  return(t(result <- data.frame(acc = acc , n = n)))
}
library(foreach)

run_time <- system.time(
 result_y1_1 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
 coverage_calculating(n,y1_1,x_i_1,beta_1))

(run_time)
##    user  system elapsed
##   13.36    0.08   13.46
result_y1_1 <- as.data.frame(t(result_y1_1))
result_y1_1
##         acc    n
## 1 0.9506667    3
## 2 0.9563333   10
## 3 0.9560000  100
## 4 0.9616667  500
## 5 0.9626667 1000
## 6 0.9760000 2000
run_time <- system.time(
 result_y1_2 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
 coverage_calculating(n,y1_2,x_i_1,beta_1))
(run_time)
##    user  system elapsed
##   12.96    0.08   13.03
result_y1_2 <- as.data.frame(t(result_y1_2))
result_y1_2
##         acc    n
## 1 0.9433333    3
## 2 0.9456667   10
## 3 0.9563333  100
## 4 0.9586667  500
## 5 0.9610000 1000
## 6 0.9720000 2000
run_time <- system.time(
 result_y1_3 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
 coverage_calculating(n,y1_3,x_i_1,beta_1))

(run_time)
##    user  system elapsed
##   12.97    0.08   13.05
result_y1_3 <- as.data.frame(t(result_y1_3))
result_y1_3
```

```
##         acc    n
## 1 0.9490000    3
## 2 0.9483333   10
## 3 0.9520000  100
## 4 0.9526667  500
## 5 0.9556667 1000
## 6 0.9523333 2000
```

```
run_time <- system.time(
 result_y1_4 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
 coverage_calculating(n,y1_4,x_i_1,beta_1))
```

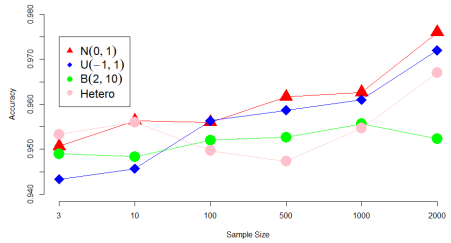
```
(run_time)
```

```
##    user  system elapsed
##   12.83    0.04   12.88
```

```
result_y1_4 <- as.data.frame(t(result_y1_4))
result_y1_4
```

```
##         acc    n
## 1 0.9533333    3
## 2 0.9560000   10
## 3 0.9496667  100
## 4 0.9473333  500
## 5 0.9546667 1000
## 6 0.9670000 2000
```

```
plot.default(x = as.factor(result_y1_1$n),y = result_y1_1$acc,type = 'b',pch = 17,col = 'red',
cex = 3, axes = FALSE,ylim = c(0.940,0.980),ylab = "Accuracy", xlab = "Sample Size")
lines(x = c(1:6), y = result_y1_2$acc,type = 'b',pch = 18,col = 'blue', cex = 3)
lines(x = c(1:6), y = result_y1_3$acc,type = 'b',pch = 19,col = 'green', cex = 3)
lines(x = c(1:6), y = result_y1_4$acc,type = 'b',pch = 19,col = 'pink', cex = 3)
axis(side = 1, at = as.factor(result_y1_1$n),labels =
c("3","10","100","500","1000","2000"))
axis(side = 2, at = seq(from = 0.940,to = 0.98,by = 0.005))
legend(1,0.975,pch = c(17,18,19,19), col = c('red','blue','green','pink'),legend =
c(expression(N(0,1)),expression(U(-1,1)),expression(B(2,10)),'Hetero'),cex = 1.5)
```



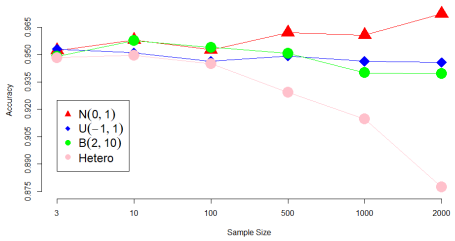

```
set.seed(456)
```

```
beta_1 <- runif(1,-100,100)
y2_1 <- 20 + beta_1*x_i_2 + normal_error
y2_2 <- 20 + beta_1*x_i_2 + uniform_error
```

```
y2_3 <- 20 + beta_1*x_i_2 + beta_error
y2_4 <- 20 + beta_1*x_i_2 + hetero_error
beta_1
## [1] -82.08968
run_time <- system.time(
 result_y2_1 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
 coverage_calculating(n,y2_1,x_i_2,beta_1))

(run_time)
##    user  system elapsed
##   13.83    0.13   13.99
result_y2_1 <- as.data.frame(t(result_y2_1))
result_y2_1
##         acc    n
## 1 0.9523333    3
## 2 0.9580000   10
## 3 0.9526667  100
## 4 0.9620000  500
## 5 0.9606667 1000
## 6 0.9723333 2000
result_y2_2 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y2_2,x_i_2,beta_1)
result_y2_2 <- as.data.frame(t(result_y2_2))
result_y2_2
##         acc    n
## 1 0.9530000    3
## 2 0.9510000   10
## 3 0.9463333  100
## 4 0.9493333  500
## 5 0.9463333 1000
## 6 0.9456667 2000
result_y2_3 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y2_3,x_i_2,beta_1)
result_y2_3 <- as.data.frame(t(result_y2_3))
result_y2_3
##         acc    n
## 1 0.9490000    3
## 2 0.9576667   10
## 3 0.9540000  100
## 4 0.9506667  500
## 5 0.9400000 1000
## 6 0.9396667 2000
result_y2_4 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y2_4,x_i_2,beta_1)
```

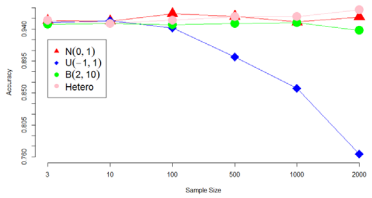
```
result_y2_4 <- as.data.frame(t(result_y2_4))
result_y2_4
##          acc    n
## 1 0.9483333    3
## 2 0.9496667   10
## 3 0.9450000  100
## 4 0.9293333  500
## 5 0.9146667 1000
## 6 0.8773333 2000
plot.default(x = as.factor(result_y2_1$n),y = result_y2_1$acc,type = 'b',pch = 17,col = 'red',
cex = 3, axes = FALSE,ylim = c(0.875,0.975),ylab = "Accuracy", xlab = "Sample Size")
lines(x = c(1:6), y = result_y2_2$acc,type = 'b',pch = 18,col = 'blue', cex = 3)
lines(x = c(1:6), y = result_y2_3$acc,type = 'b',pch = 19,col = 'green', cex = 3)
lines(x = c(1:6), y = result_y2_4$acc,type = 'b',pch = 19,col = 'pink', cex = 3)
axis(side = 1, at = as.factor(result_y1_1$n),labels =
c("3","10","100","500","1000","2000"))
axis(side = 2, at = seq(from = 0.875,to = 0.975,by = 0.015))
legend(1,0.925,pch = c(17,18,19,19), col = c('red','blue','green','pink'),legend =
c(expression(N(0,1)),expression(U(-1,1)),expression(B(2,10)),'Hetero'),cex = 1.5)
```




```
set.seed(789)

beta_1 <- runif(1,-100,100)
y3_1 <- 20 + beta_1*x_i_3 + normal_error
y3_2 <- 20 + beta_1*x_i_3 + uniform_error
y3_3 <- 20 + beta_1*x_i_3 + beta_error
y3_4 <- 20 + beta_1*x_i_3 + hetero_error
beta_1
## [1] 39.97887
result_y3_1 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y3_1,x_i_3,beta_1)
result_y3_1 <- as.data.frame(t(result_y3_1))
result_y3_1
##          acc    n
## 1 0.9520000    3
## 2 0.9513333   10
## 3 0.9616667  100
## 4 0.9580000  500
```

```
## 5 0.9500000 1000
## 6 0.9570000 2000
result_y3_2 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y3_2,x_i_3,beta_1)
result_y3_2 <- as.data.frame(t(result_y3_2))
result_y3_2
##         acc    n
## 1 0.9493333    3
## 2 0.9516667   10
## 3 0.9416667  100
## 4 0.9006667  500
## 5 0.8563333 1000
## 6 0.7640000 2000
result_y3_3 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y3_3,x_i_3,beta_1)
result_y3_3 <- as.data.frame(t(result_y3_3))
result_y3_3
##         acc    n
## 1 0.9466667    3
## 2 0.9480000   10
## 3 0.9460000  100
## 4 0.9480000  500
## 5 0.9490000 1000
## 6 0.9383333 2000
result_y3_4 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y3_4,x_i_3,beta_1)
result_y3_4 <- as.data.frame(t(result_y3_4))
result_y3_4
##         acc    n
## 1 0.9523333    3
## 2 0.9480000   10
## 3 0.9526667  100
## 4 0.9573333  500
## 5 0.9576667 1000
## 6 0.9670000 2000
plot.default(x = as.factor(result_y3_1$n),y = result_y3_1$acc,type = 'b',pch = 17,col = 'red',
cex = 3, axes = FALSE,ylim = c(0.760,0.970),ylab = "Accuracy", xlab = "Sample Size")
lines(x = c(1:6), y = result_y3_2$acc,type = 'b',pch = 18,col = 'blue', cex = 3)
lines(x = c(1:6), y = result_y3_3$acc,type = 'b',pch = 19,col = 'green', cex = 3)
lines(x = c(1:6), y = result_y3_4$acc,type = 'b',pch = 19,col = 'pink', cex = 3)
axis(side = 1, at = as.factor(result_y1_1$n),labels =
c("3","10","100","500","1000","2000"))
axis(side = 2, at = seq(from = 0.76,to = 0.97,by = 0.015))
legend(1,0.925,pch = c(17,18,19,19), col = c('red','blue','green','pink'),legend =
c(expression(N(0,1)),expression(U(-1,1)),expression(B(2,10)),'Hetero'),cex = 1.5)
```




```
set.seed(321)

beta_1 <- runif(1,-100,100)
y4_1 <- 20 + beta_1*x_i_4 + normal_error
y4_2 <- 20 + beta_1*x_i_4 + uniform_error
y4_3 <- 20 + beta_1*x_i_4 + beta_error
y4_4 <- 20 + beta_1*x_i_4 + hetero_error
beta_1
## [1] 91.17875
result_y4_1 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y4_1,x_i_4,beta_1)
result_y4_1 <- as.data.frame(t(result_y4_1))
result_y4_1
##         acc    n
## 1 0.9533333    3
## 2 0.9450000   10
## 3 0.9490000  100
## 4 0.9606667  500
## 5 0.9613333 1000
## 6 0.9750000 2000
result_y4_2 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y4_2,x_i_4,beta_1)
result_y4_2 <- as.data.frame(t(result_y4_2))
result_y4_2
##         acc    n
## 1 0.9516667    3
## 2 0.9516667   10
## 3 0.9540000  100
## 4 0.9523333  500
## 5 0.9400000 1000
## 6 0.9500000 2000
result_y4_3 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y4_3,x_i_4,beta_1)
result_y4_3 <- as.data.frame(t(result_y4_3))
result_y4_3
##         acc    n
## 1 0.9440000    3
## 2 0.9493333   10
## 3 0.9513333  100
```

```
## 4 0.9500000  500
## 5 0.9460000 1000
## 6 0.9500000 2000
result_y4_4 <- foreach(n = c(3,10,100,500,1000,2000),.combine = cbind ) %dopar%
coverage_calculating(n,y4_4,x_i_4,beta_1)
result_y4_4 <- as.data.frame(t(result_y4_4))
result_y4_4
##         acc    n
## 1 0.9486667    3
## 2 0.9503333   10
## 3 0.9516667  100
## 4 0.9523333  500
## 5 0.9546667 1000
## 6 0.9643333 2000
plot.default(x = as.factor(result_y4_1$n),y = result_y4_1$acc,type = 'b',pch = 17,col = 'red',
cex = 3, axes = FALSE,ylim = c(0.92,0.980),ylab = "Accuracy", xlab = "Sample Size")
lines(x = c(1:6), y = result_y4_2$acc,type = 'b',pch = 18,col = 'blue', cex = 3)
lines(x = c(1:6), y = result_y4_3$acc,type = 'b',pch = 19,col = 'green', cex = 3)
lines(x = c(1:6), y = result_y4_4$acc,type = 'b',pch = 19,col = 'pink', cex = 3)
axis(side = 1, at = as.factor(result_y1_1$n),labels =
c("3","10","100","500","1000","2000"))
axis(side = 2, at = seq(from = 0.92,to = 0.98,by = 0.015))
legend(1,0.980,pch = c(17,18,19,19), col = c('red','blue','green','pink'),legend =
c(expression(N(0,1)),expression(U(-1,1)),expression(B(2,10)),'Hetero'),cex = 1.5)
```