Econometrics, Exercise 1

Nikolas Kuschnig, Daran Demirol & Casper Engelen

14/03/2018

Exercise 1.1

i)

First we create X and beta for calculating Y.

```
X = matrix(c(rep(1, 15), seq(1:15)), nrow = 15)
beta = c(1, 0.25)
```

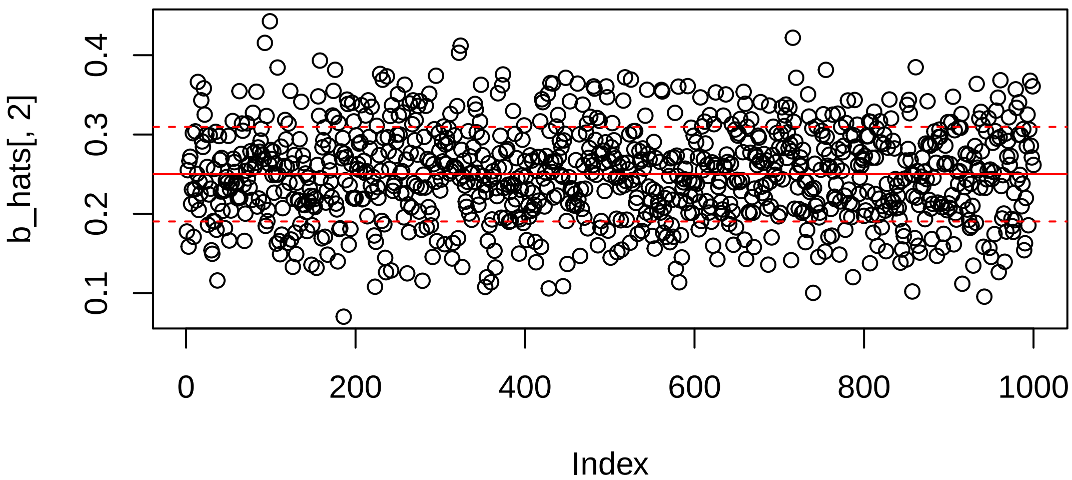
Now we initialize our beta-hat matrix and loop over it, inserting the coefficients of the linear model estimated by each iteration.

```
# Initialize matrix to store each b_hat
b_hats = matrix(ncol = 2, nrow = 1000)

for (i in 1:1000) {
  # Generate our error term
  e = rnorm(15, 0, 1)

  # Calculate Y from our X, beta and e
  Y = X %*% beta + e

  # Store the coefficients of the current iteration
  b_hats[i, ] = lm(Y ~ X)$coefficients[2:3]

  # Store one summary of our linear model for later
  if (i == 027) summ = summary(lm(Y ~ X))
}

# Comply to homework specifications
b_hat = b_hats
```

ii) & iii)

We calculate the mean and standard deviation of our beta-hats and plot the observations including lines for mean and +/- the standard deviation.

```
# Calculate the mean and standard deviation of our b_hats
mu = mean(b_hats[, 2])
std = sd(b_hats[, 2])

# Plot the b_hat observations and add lines for mean, +/- std
plot(b_hats[, 2])
abline(mu, 0, col = "red")
abline(mu + std, 0, col = "red", lty = 2)
abline(mu - std, 0, col = "red", lty = 2)
```



Then we print the previously stored summary of one linear model and interpret its output.

```
summm
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0988 -0.5606  0.2093  0.6796  1.7056
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.79040    0.58746   1.345  0.20147
## X1                NA         NA      NA       NA
## X2           0.22442    0.06461   3.473  0.00412 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.081 on 13 degrees of freedom
## Multiple R-squared:  0.4813, Adjusted R-squared:  0.4414
## F-statistic: 12.06 on 1 and 13 DF,  p-value: 0.004119
```

In our observation (it will differ from the compiled one) we have an estimated intercept of 0.7 with a standard error of 0.66 that is not significant. Contrarily our X2 is highly significant with an estimated coefficient of 0.32 at a standard error of 0.07. These values are rather far from the mean and std we computed from our beta hats, but not far enough to worry us.

Exercise 1.2

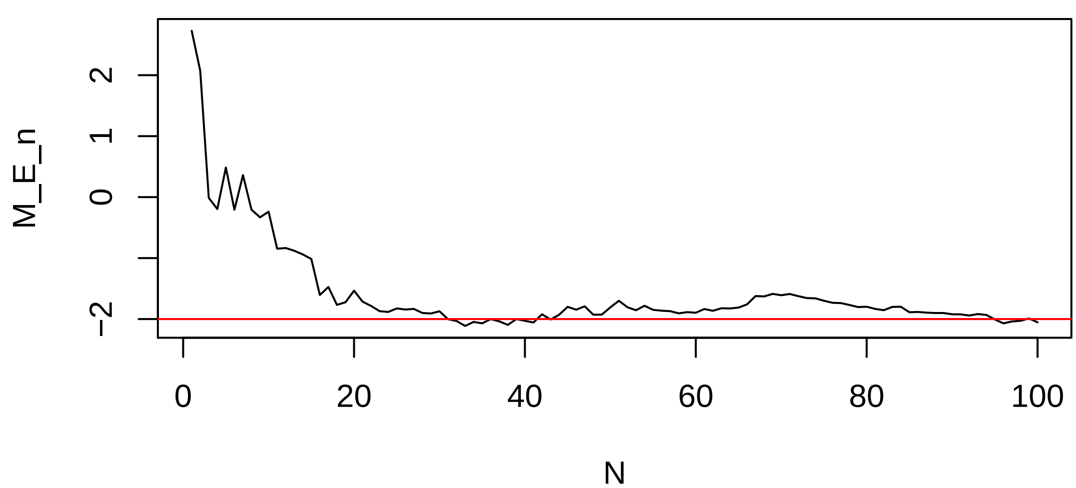
i) & ii)

We create a function to show convergence for vectors with length of 100 & 10000.

```r
do_stuff = function(len, mu = -2, va = 9, cauchy = FALSE) {

  # Create normally distributed vector
  E_n = rnorm(len, mu, sd = sqrt(va))

  # Create the cauchy vector if applicable
  if (cauchy) E_n = E_n / rnorm(len, mu, sd = sqrt(va))


  # Initialize vector to store the mean up to i
  M_E_n = vector("numeric", length = len)

  for (i in 1:len) {
    M_E_n[i] = mean(E_n[1:i])
  }

  # Create the vector from ii) b.
  N = 1:len

  # Visualize the thingy with code from ii) c.
  #ts.plot(cbind(N, M_E_n))

  # Return the visualization without the ugly N = N-line
  plot(x = N, y = M_E_n, type = "l")
  abline(mu, 0, col = "red")
}
```

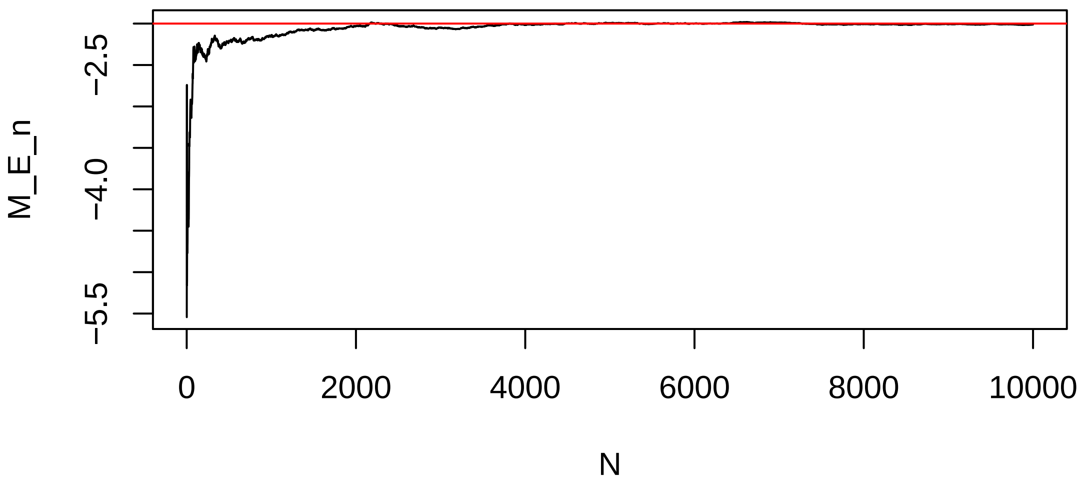
The output for a length of 100 already shows convergence.

```r
do_stuff(100)
```



Which is confirmed by the output for a length of 10000.

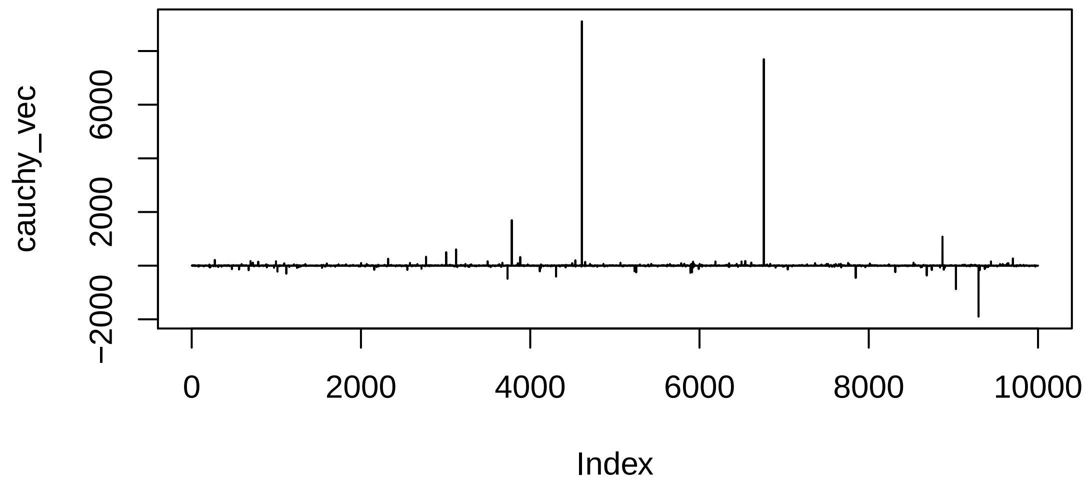
```r
do_stuff(10000)
```



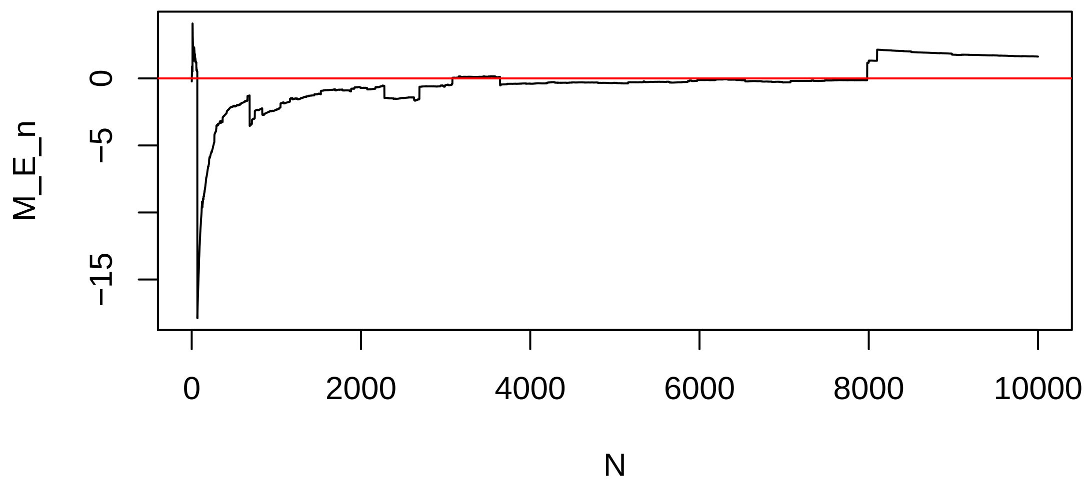
iii) & iv)

We create a vector of 10000 Cauchy-distributed variables and take a look at it graphically. We take note of the extreme extremes and the concentration of values. Then we observe no convergence to the mean when calling our function from before.

```r
# Create a vector of 10000 Cauchy-distributed variables
cauchy_vec = rnorm(10000, 0, 1) / rnorm(10000, 0, 1)

# Have a look at it graphically
plot(cauchy_vec, type = "l")
```

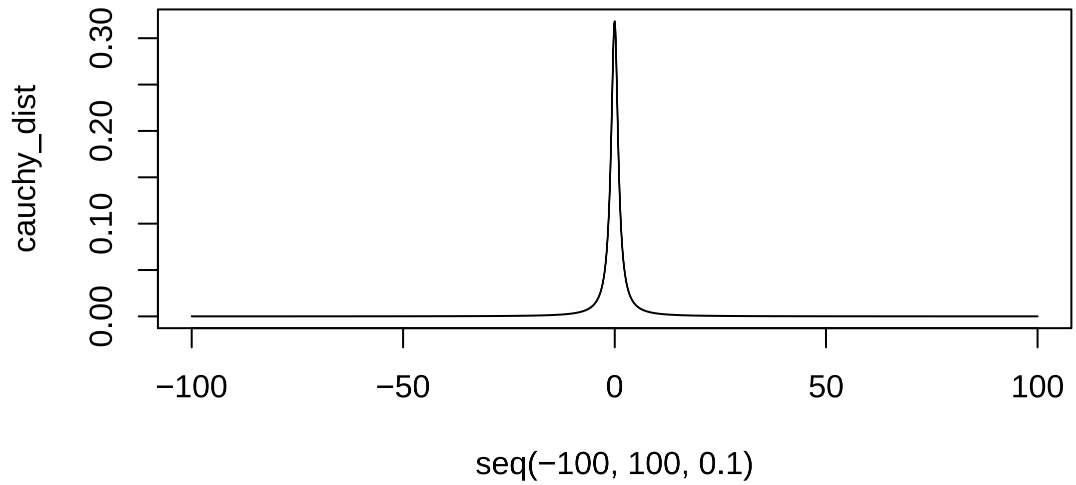



```
# Plot the (usually) missing convergence
do_stuff(10000, 0, 1, cauchy = TRUE)
```



To conclude we try illustrating the Cauchy-density with graphics.

```
# Try illustrating the cauchy density (thanks dcauchy) with graphics
cauchy_dist = dcauchy(seq(-100, 100, 0.1))
plot(y = cauchy_dist, x = seq(-100, 100, 0.1), type = "l")
```




```r
# Plot our custom-made vector
plot(density(cauchy_vec), main = "")
```