week11 exercise

Xiang Li

2023/12/30

```
library(rbenchmark)
```

Exercise 1

```
set.seed(13)
n.repl = 1000
weib.shape = c(2, 3)
weib.scale = c(5, 4)
min.weibs = rep(NA, n.repl)
for (i in 1:n.repl) {
    x1 = rweibull(1, shape = weib.shape[1], scale = weib.scale[1])
    x2 = rweibull(1, shape = weib.shape[2], scale = weib.scale[2])
    min.weibs[i] = min(x1, x2)
}
```

1

```
rep_func = function(n, shape_v, scale_v) {
    result = replicate(n, expr = {
        x1 = rweibull(1, shape = shape_v[1], scale = scale_v[1])
        x2 = rweibull(1, shape = shape_v[2], scale = scale_v[2])
        min(x1, x2)
    })
    return(result)
}
```

2

```
time_table = benchmark(`for` = {
    min.weibs = rep(NA, n.repl)
    for (i in 1:n.repl) {
        x1 = rweibull(1, shape = weib.shape[1], scale = weib.scale[1])
        x2 = rweibull(1, shape = weib.shape[2], scale = weib.scale[2])
        min.weibs[i] = min(x1, x2)
```

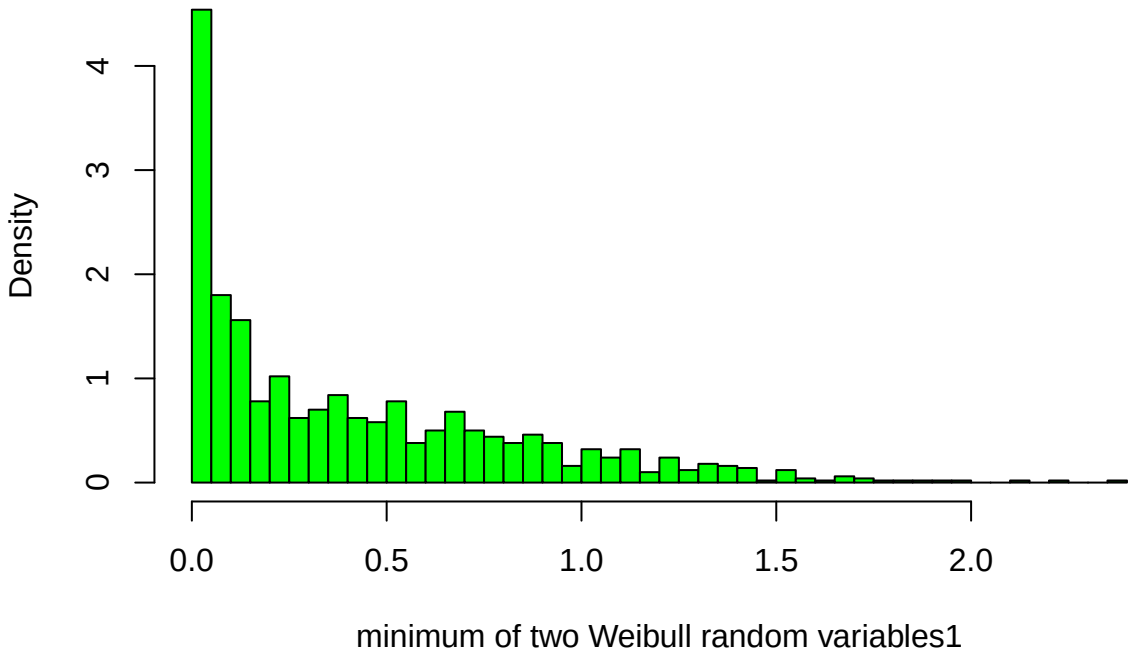
```
    }
}, rep = {
    rep_func(n.repl, weib.shape, weib.scale)
}, replications = 100)
time_table
```

```
##   test replications elapsed relative user.self sys.self user.child sys.child
## 1  for          100   1.004    1.599     0.914    0.069          0         0
## 2  rep          100   0.628    1.000     0.574    0.049          0         0
```

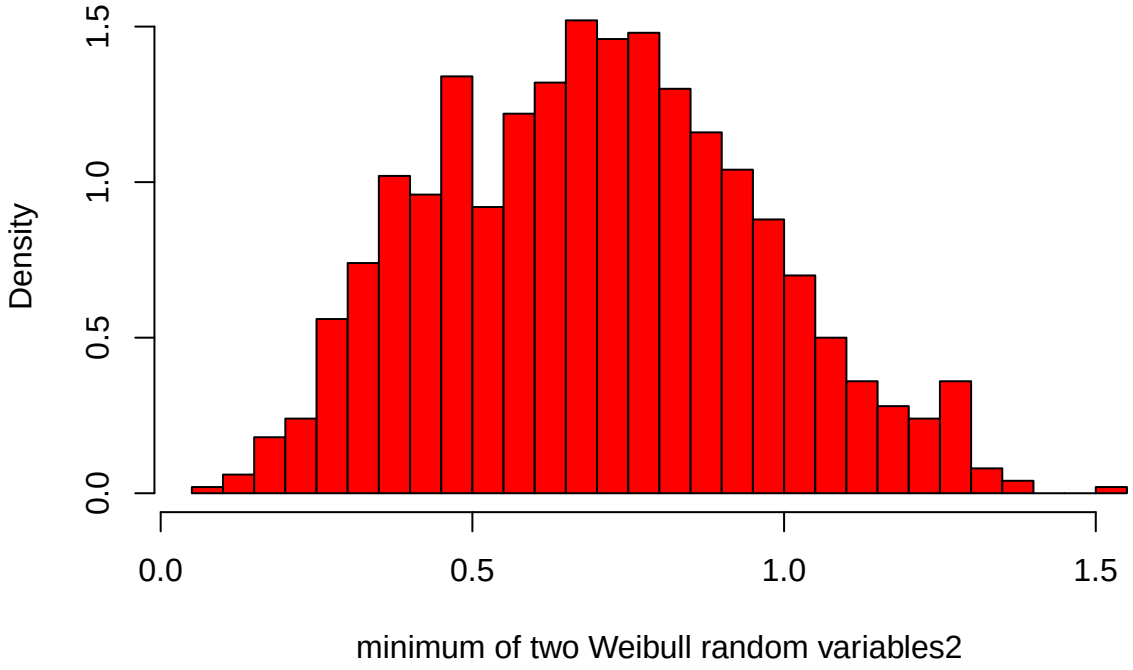
`replicate()` is faster.

3

```
min.weibs1 = rep_func(n.repl, c(0.5, 2), c(0.7, 1))
hist(min.weibs1, 50, prob = T, main = "", col = "green",
    xlab = "minimum of two Weibull random variables1")
```




```
min.weibs2 = rep_func(n.repl, c(3, 3), c(1, 1))
hist(min.weibs2, 50, prob = T, main = "", col = "red", xlab = "minimum of two Weibull random variables2
```



Exercise 2

1

```
neg_loglik = function(theta, pi1, w1, x) {
    mu1 = theta[1]
    sigma1 = exp(theta[2])
    mu2 = theta[3]
    sigma2 = exp(theta[4])
    # density of the mixture model:
    f.x1 = pi1 * dnorm(x, mu1, sigma1)
    f.x2 = (1 - pi1) * dnorm(x, mu2, sigma2)
    # negative log-likelihood:
    -sum(w1 * log(f.x1) + (1 - w1) * log(f.x2))
}
```

```
set.seed(13)
n = 2000
pi1 = 0.35
mu1 = 0.8
mu2 = 2.5
sigma1 = 0.8
sigma2 = 0.6
```

```
group = sample(1:2, n, replace = T, prob = c(pi1, 1 - pi1))
table(group)
```

```
## group
##    1    2
##  698 1302
```

```
x = rep(NA, n)
x[group == 1] = rnorm(sum(group == 1), mu1, sd = sigma1)
x[group == 2] = rnorm(sum(group == 2), mu2, sd = sigma2)
```

2

```
EM_function = function(x, n.iter) {
    # 1. preallocate objects
    n = length(x)
    pi1hat = rep(NA, n.iter)
    p1hat = matrix(NA, n.iter, n)
    thetahat = matrix(NA, n.iter, 4)
    # 2: initialize the algorithm
    range_pi1start = c(runif(1, 0.2, 0.45), runif(1, 0.55,
        0.8))
    pi1hat[1] = mean(range_pi1start)
    p1hat[1, ] = runif(n, range_pi1start[1], range_pi1start[2])
    # 3. first M step:
    thetahat[1, ] = optim(c(0, 1, 0, 1), neg_loglik, pi1 = pi1hat[1],
        w1 = p1hat[1, ], x = x)$par
    # 4. run the EM:
    for (t in 2:n.iter) {
        # E step: update individual probability
        # memberships
        p.temp = cbind(pi1hat[t - 1] * dnorm(x, thetahat[t -
            1, 1], exp(thetahat[t - 1, 2])), (1 - pi1hat[t -
            1]) * dnorm(x, thetahat[t - 1, 3], exp(thetahat[t -
            1, 4])))
        p1hat[t, ] = p.temp[, 1]/rowSums(p.temp)
        # M step: update parameter estimates
        pi1hat[t] = mean(p1hat[t, ])
        thetahat[t, ] = optim(thetahat[t - 1, ], neg_loglik,
            pi1 = pi1hat[t], w1 = p1hat[t, ], x = x)$par
    }
    # 5: compute the loglikelihood at the end of the
    # algorithm
    thetahat_ = thetahat[n.iter, ]
    pi1hat_ = pi1hat[n.iter]
    p1hat_ = p1hat[n.iter, ]
    loglikFinal = -neg_loglik(theta = thetahat_, pi1 = pi1hat_,
        w1 = p1hat_, x)
    # 6: define the exports
    out = list(mu1 = thetahat_[1], sigma1 = exp(thetahat_[2]),
```

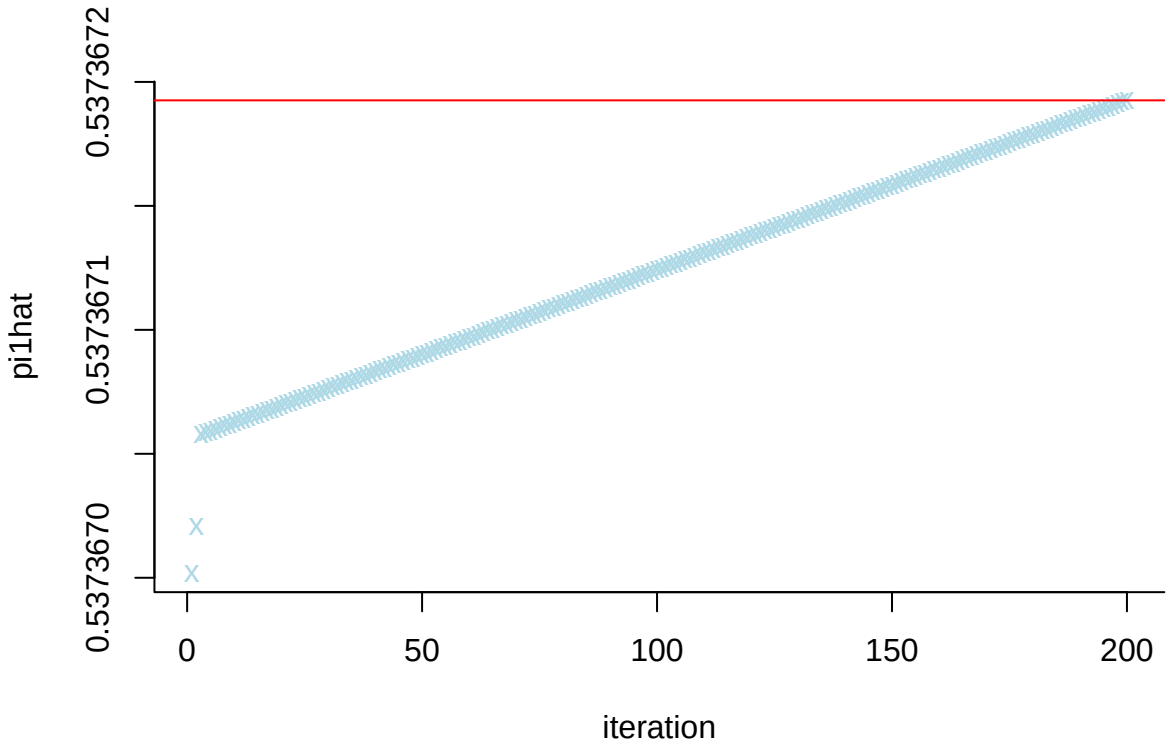
```
        mu2 = thetahat_[3], sigma2 = exp(thetahat_[4]),
        pi1hat = pi1hat, p1hat = p1hat, logl = loglikFinal)
    return(out)
}
```

```
EM_result = EM_function(x = x, n.iter = 200)
c(EM_result$mu1, EM_result$sigma1, EM_result$mu2, EM_result$sigma2)
```

```
## [1] 1.911461 1.051431 1.911668 1.051677
```

3

```
par(bty = "l")
plot(x = 1:200, y = EM_result$pi1hat, type = "p", pch = "x",
    col = "lightblue", xlab = "iteration", ylab = "pi1hat")
abline(h = EM_result$pi1hat[200], col = "red")
```



4

```
EM_result_ls = replicate(10, {
    EM_function(x, n.iter = 200)
}, simplify = F)
```

```
logl_v = rep(NA, 10)
for (i in 1:10) {
    logl_v[i] = EM_result_ls[[i]]$logl
}
best_EM_result = EM_result_ls[[which.max(logl_v)]]
c(mu1 = best_EM_result$mu1, sigma1 = best_EM_result$sigma1,
    mu2 = best_EM_result$mu2, sigma2 = best_EM_result$sigma2)
```

```
##       mu1    sigma1       mu2    sigma2
## 0.6924640 0.7193173 2.4846303 0.5961044
```

5

In my result, $\hat{\mu_1}$ and $\hat{\sigma_1}$ respectively correspond to μ_1 and σ_1. $\hat{\mu_2}$ and $\hat{\sigma_2}$ respectively correspond to μ_2 and σ_2

6

```
predicted_group = ifelse(best_EM_result$p1hat[200, ] > 0.5,
    1, 2)
conf_table = table(predicted_group, group)
miscl_rate = (conf_table[1, 2] + conf_table[2, 1])/n
miscl_rate
```

```
## [1] 0.0985
```