STAT 600 Homework 1

Karissa Palmer

2024-01-19

Question 1

Make an `R` package called `SimpLin` for running a simple linear regression model that :

- Takes in numeric vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Outputs estimated regression coefficients, $\hat{\beta}_0$ and $\hat{\beta}_1$, their corresponding standard errors and 95% confidence intervals, residuals, and predicted values as a list.
- Wraps the `cpp` function (`SimpLinCpp`) in an `R` function (`SimpLinR`) that throws errors if $\boldsymbol{x}$ and $\boldsymbol{y}$ are not numeric vectors of the same length.
- Provides a description and brief vignette (using .Rmd file) demonstrating how to use the package.

Answer: Linked to Github. To download follow:
`https://github.com/KarissaPalmer/STAT600/tree/6d81cfc6393aba68d288d120456004d89d0131a5/Homework1/`
And choose the folder `SimpLin`. The associated vignette is in the folder `vignettes`.

Question 2

Connect and manage the development of your `R` package with your GitHub account. Submit link in HW. I should be able to download and install your `R` package locally.

Answer: In Question 1.

Question 3

Simulate 100 data sets with $n = 100$ observations each, where $x \sim N(0,1)$ and error terms $\epsilon \sim N(0,1)$ with true regression coefficients $\beta_0 = 1$ and $\beta_1 = -1$. Fit a linear regression model to each of the data sets using your package in (1) and the `lm()` function in `R` **in parallel**. Calculate the runtime for each of the data sets using both models. Note that you do not have to run in parallel using Rcpp.

Answer: Below is the code for the simulated data. I used 6 cores to run the simulation and it took 0.116 seconds.

```r
#First, simulate the data
RNGkind("L'Ecuyer-CMRG")
set.seed(23)

# start<-Sys.time()
registerDoParallel(6)
sim_data<- foreach(t = 1:100) %dopar% {
  x<- rnorm(100)
  eps<- rnorm(100)
  #b0=1 and b1=-1
  y<- 1-x+eps

  cbind(x,y)
}
# Sys.time()-start
#took 0.116 seconds

#Done
stopImplicitCluster()

#Save simulated data for easy loading for comparison in Q4
# saveRDS(sim_data, 'Sim_Data.RData')
```

When using applying the function `SimpLinR` and `lm()` I also used 6 cores each. The function `SimpLinR` took 0.2370 seconds to go through the simulations and the R function took 0.0957 seconds. The file on Github `Sim_Data.RData` has the simulated data whose results may be checked.

Question 4

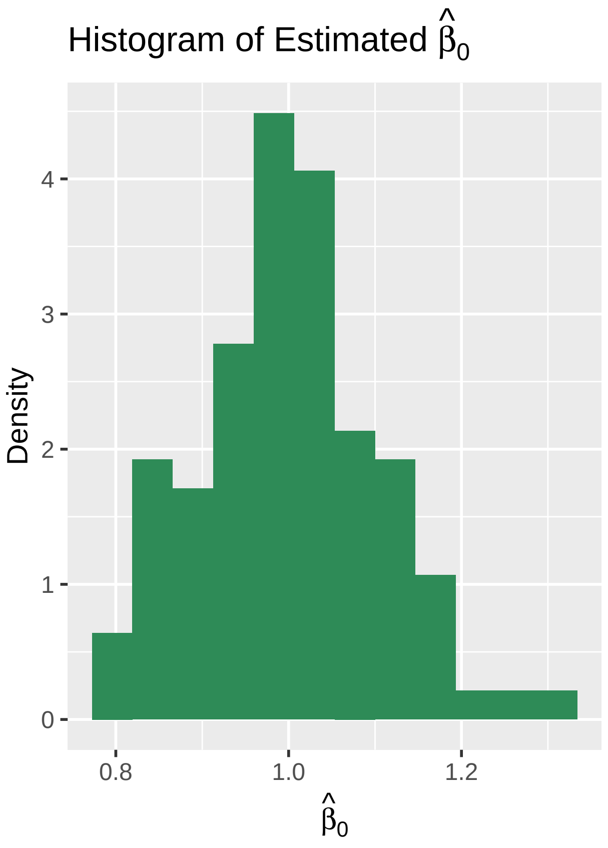
Provide a table of summary statistics for the simulations including average runtime, bias, coverage probability (proportion of 95% CIs that include the true regression coefficients), mean squared error for regression coefficients, and predictive mean squared error for $\hat{y}$ across all simulations for your model and `lm()`. Plot a histogram of the estimated regression coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ across all simulations. Comment on the performance of the methods.

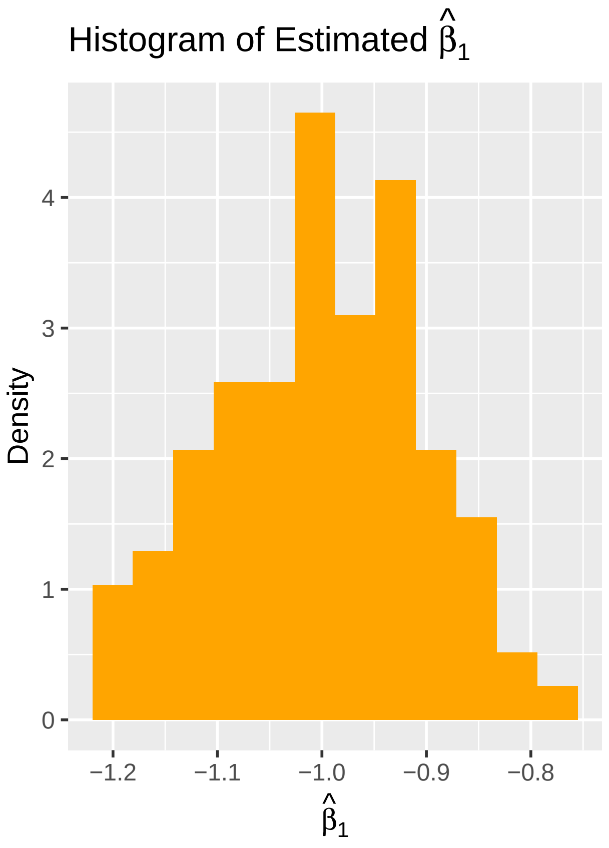
Answer:

Table 1: Simulation Results Comparison: SimpLin vs. lm

	SimpLin Package	lm
Average Runtime	1e-04 (SD = 2e-04)	9e-04 (SD = 0.0018)
Bias b0	2e-04 (SD = 0.1041)	2e-04 (SD = 0.1041)
Bias b1	-0.0031 (SD = 0.0967)	-0.0031 (SD = 0.0967)
Coverage Probability b0	0.94 (SD = 0.2387)	0.94 (SD = 0.2387)
Coverage Probability b1	0.98 (SD = 0.1407)	0.98 (SD = 0.1407)
MSE b0	0.0107 (SD = 0.0148)	0.0107 (SD = 0.0148)
MSE b1	0.0093 (SD = 0.0111)	0.0093 (SD = 0.0111)

Since the outputs of `SimpLin` and `lm()` in terms of their estimated coefficients are the same for each data set, runtime is one of the more telling measurements for differences in approach. We saw in Question 3 that the `lm()` function was approximately twice as fast, but on the individual basis our package (Rcpp) was faster than `lm()` while `lm()` had slightly more variability in the computation.





Appendix

```r
#get the data we need using SimpLin
# RNGkind("L'Ecuyer-CMRG")
# set.seed(23)
#don't need to set seeds b/c we're already using data we have.
start1<- Sys.time()

registerDoParallel(6)

summ_rcpp<- foreach(t = 1:100) %dopar% {
  #start time
  start <- Sys.time()

  mod_rcpp<- SimpLinR(sim_data[[t]][, "x"], sim_data[[t]][, "y"])
  tot_time<- difftime(Sys.time(), start)

  #Make nice assignments from output for what we need
  coef_rcpp<- t(mod_rcpp$Coefficients)
  ci_rcpp<- mod_rcpp$Conf_Ints
  pred_rcpp<- mod_rcpp$Pred_Vals

  #Put it all in a list
  list(time = tot_time, coef = coef_rcpp, ci = ci_rcpp, pred = pred_rcpp)
}

stopImplicitCluster()

difftime(Sys.time(), start1)
```

```r
#Do something similar for R

# RNGkind("L'Ecuyer-CMRG")
# set.seed(23)
start2<-Sys.time()

registerDoParallel(6)

summ_r<- foreach(t = 1:100) %dopar% {
  #start time
  start <- Sys.time()

  mod_r<- lm(sim_data[[t]][,2] ~ sim_data[[t]][,1])
  tot_time<- difftime(Sys.time(), start)

  #Make nice assignments from output for what we need
  coef_r<- mod_r$coefficients
  ci_r<- confint(mod_r)
  pred_r<- as.matrix(mod_r$fitted.values)

  #Put it all in a list
  list(time = tot_time, coef = coef_r, ci = ci_r, pred = pred_r)
}

stopImplicitCluster()

difftime(Sys.time(), start2)
```

```r
#Report mean and sd of each of the asked things in the summary statistics table.
mean_sd<- function(x){
  avg<-round(mean(x), digits = 4)
  sds<- round(sd(x), digits = 4)
  paste0(avg, ' (SD = ', sds, ')')
}
```

```r
#Get what we need for the summary statistics table.

#average runtime for each simulation
time_rcpp<- sapply(1:100, function(x){summ_rcpp[[x]]$time})
time_rcpp<- mean_sd(time_rcpp)

time_r<- sapply(1:100, function(x){summ_r[[x]]$time})
time_r<- mean_sd(time_r)

#Split the coefficients up a little bit more so easier to work with
#Make a matrix of true beta0 and beta1 so
coef_true<- matrix(c(1,-1), nrow = 100, ncol = 2, byrow = T)
coef_r<- t(sapply(1:100, function(x){summ_r[[x]]$coef}))
#rename these columns
colnames(coef_r)<- c('V1', 'V2')
coef_rcpp<- t(sapply(1:100, function(x){summ_rcpp[[x]]$coef}))

#Calculate bias from the above
bias_r<- apply(coef_r - coef_true, 2, mean_sd)
bias_rcpp<- apply(coef_rcpp - coef_true, 2, mean_sd)

#Coverage probability
cov_r<- data.frame()
cov_rcpp<- data.frame()

for (i in 1:100){
  cov_r[i,1]<- ifelse(summ_r[[i]]$ci[1,1] <= 1 & summ_r[[i]]$ci[1,2] >= 1, 1,0)
  cov_r[i,2]<- ifelse(summ_r[[i]]$ci[2,1] <= -1 & summ_r[[i]]$ci[2,2] >= -1, 1,0)

  cov_rcpp[i,1]<- ifelse(summ_rcpp[[i]]$ci[1,1] <= 1 & summ_rcpp[[i]]$ci[1,2] >= 1, 1,0)
  cov_rcpp[i,2]<- ifelse(summ_rcpp[[i]]$ci[2,1] <= -1 & summ_rcpp[[i]]$ci[2,2] >= -1, 1,0)

}

cov_r_b0<- mean_sd(cov_r[,1])
cov_r_b1<- mean_sd(cov_r[,2])

cov_rcpp_b0<- mean_sd(cov_rcpp[,1])
cov_rcpp_b1<- mean_sd(cov_rcpp[,2])
```

```r
#MSE of predictors
mse_r<- apply(((coef_r - coef_true)^2), 2, mean_sd)
mse_rcpp<- apply(((coef_rcpp - coef_true)^2), 2, mean_sd)

#Create the table
sum_stats<- rbind(c(time_rcpp, time_r), c(bias_rcpp[[1]], bias_r[[1]])
                  , c(bias_rcpp[[2]], bias_r[[2]]), c(cov_rcpp_b0, cov_r_b0)
                  , c(cov_rcpp_b1, cov_rcpp_b1), c(mse_rcpp[[1]], mse_r[[1]])
                  , c(mse_rcpp[[2]], mse_r[[2]]))

#Use kable to make the table a tad nicer
sum_stats<- data.frame(sum_stats)
rownames(sum_stats)<- c('Average Runtime', 'Bias b0', 'Bias b1'
                        , 'Coverage Probability b0', 'Coverage Probability b1'
                        , 'MSE b0', 'MSE b1')
colnames(sum_stats)<- c('SimpLin Package', 'lm')

kable(sum_stats, booktabs = T
      , caption = 'Simulation Results Comparison: SimpLin vs. lm') %>% kable_styling(full_width = T)
```

```r
#Plot histograms
coef_rcpp<- data.frame(coef_rcpp)
b0_p<- ggplot(coef_rcpp, aes(x = X1))+
                geom_histogram(bins=12, fill ="seagreen"
                               , aes(y = after_stat(density)))+
  labs(x = TeX('$\\hat{\\beta}_{0}$'), y = 'Density', title = TeX('Histogram of Estimated $\\hat{\\beta

b1_p<- ggplot(coef_rcpp, aes(x = X2))+
                geom_histogram(bins=12, fill ="orange"
                               , aes(y = after_stat(density)))+
  labs(x = TeX('$\\hat{\\beta}_{1}$'), y = 'Density', title = TeX('Histogram of Estimated $\\hat{\\beta

grid.arrange(b0_p,b1_p, ncol = 2)
```