MSiA_420_HW1

2023-01-16

Question 1

See attached

Question 2

a

```
## New names:
## Rows: 18 Columns: 5
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (2): y, x lgl (3): ...3, ...4, ...5
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`


##
## Call:
## lm(formula = y ~ x, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8103 -1.3567  0.7347  1.5796  3.0106
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.42071    0.77963   6.953 3.25e-06 ***
## x            0.48964    0.04359  11.234 5.32e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.164 on 16 degrees of freedom
## Multiple R-squared:  0.8875, Adjusted R-squared:  0.8805
## F-statistic: 126.2 on 1 and 16 DF,  p-value: 5.317e-09
```

As seen above, the coefficients $\hat{\beta_0}$ and $\hat{\beta_1}$ are 5.4207 and 0.4896, respectively. Thus, the initial guesses of the form $\hat{\gamma_0} = \frac{1}{\hat{\beta_0}} = \frac{1}{5.4207} = 0.1845$, and $\hat{\gamma_1} = \frac{\hat{\beta_1}}{\hat{\beta_0}} = \frac{0.48964}{5.4207} = 0.0903$

b

```
## [1] 28.13688 12.57428
```

```
## Nonlinear regression model
##   model: y ~ ((p1 * x)/(p2 + x))
##    data: dat
##    p1    p2
## 28.14 12.57
##  residual sum-of-squares: 4.302
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 9.921e-06
```

Question 3

a

```
##           [,1]      [,2]
## [1,]  15.37455 -13.73842
## [2,] -13.73842  13.92197
```

```
##           [,1]      [,2]
## [1,] 0.5502797 0.5430248
## [2,] 0.5430248 0.6076944
```

```
## [1] 0.7418084 0.7795475
```

b

```
##          p1        p2
## p1 0.529951 0.5202800
## p2 0.520280 0.5822471
```

Although not exactly the same, the covariance matrices are very similar.

c

```
## [1] 26.68294
```

```
## [1] 29.59083
```

```
## [1] 11.04637
```

```
## [1] 14.10219
```

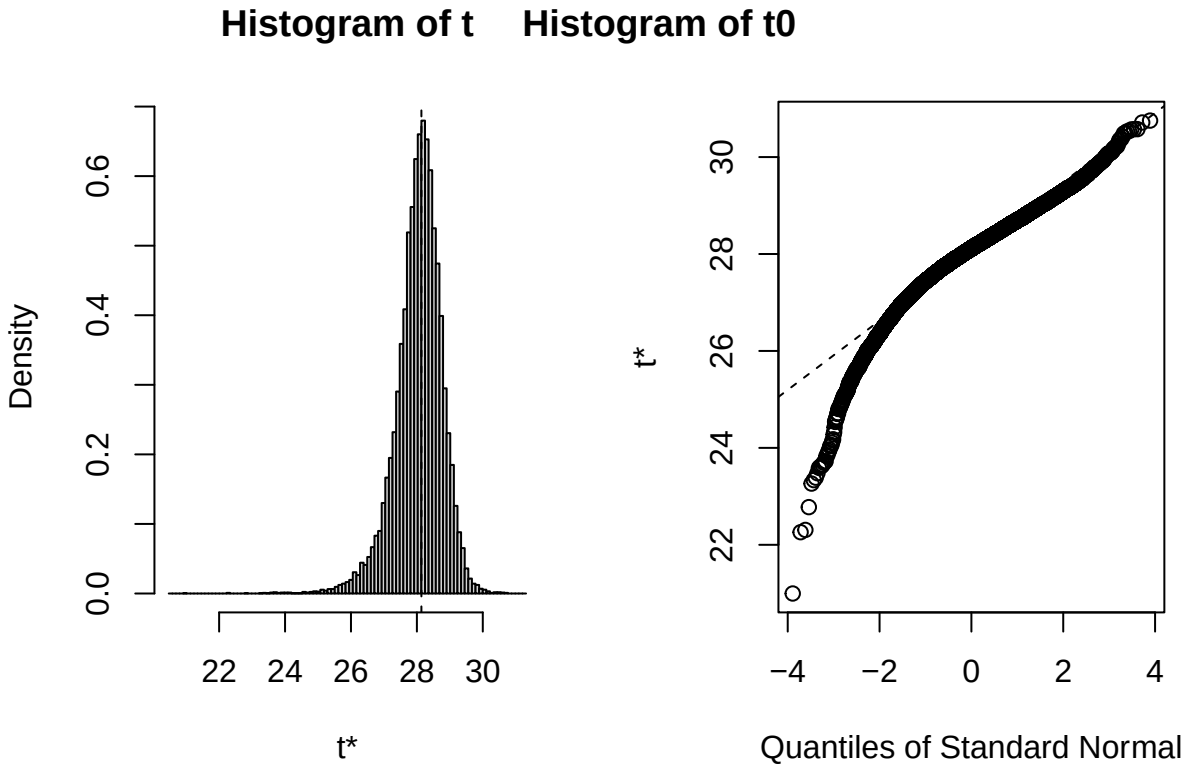
```
##       2.5 %   97.5 %
## p1 26.71021 29.56383
## p2 11.07887 14.06997
```

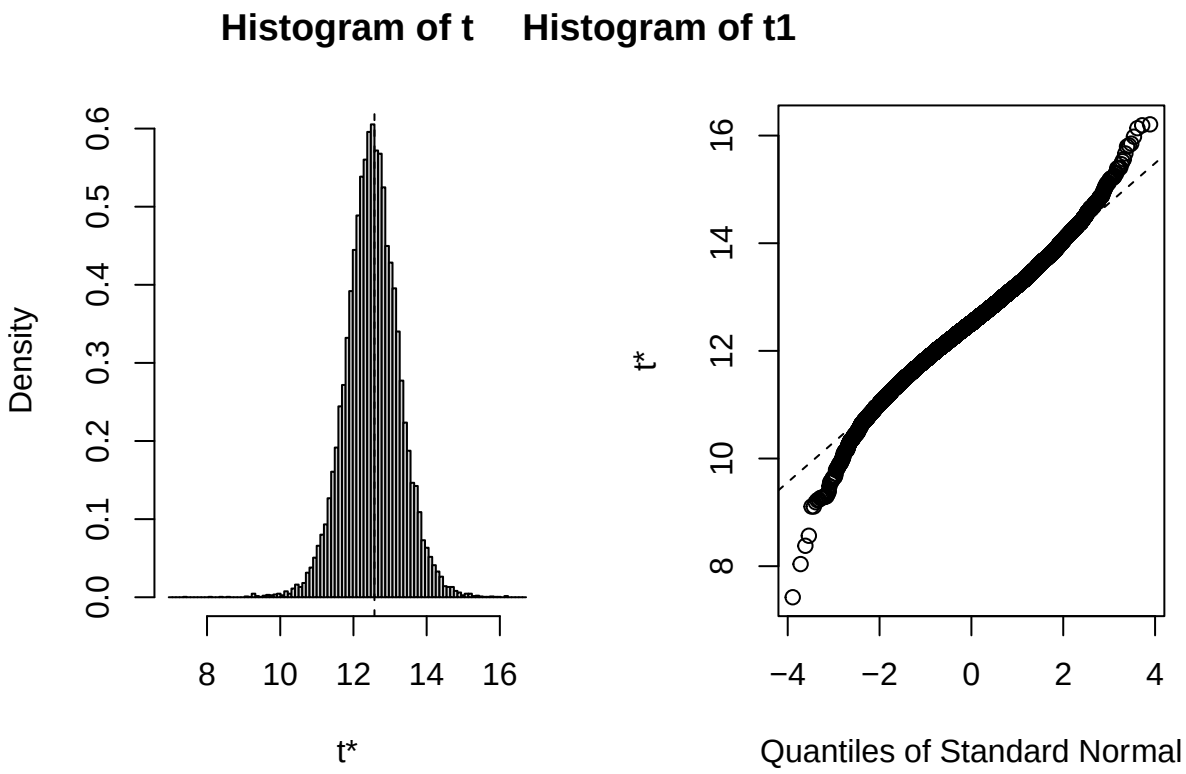
As seen from the above comparison, the "crude" confidence interval is similar to the output by R, but it is wider.

Question 4

a

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = dat, statistic = dat_fit, R = 20000, theta0 = c(0.1845,
##     0.0903))
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 28.13688 -0.08372891   0.7143639
## t2* 12.57428 -0.05167943   0.7408931
```





b

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = dat_boot, conf = 0.95, type = "norm", index = 1)
##
## Intervals :
## Level      Normal
## 95%   (26.82, 29.62 )
## Calculations and Intervals on Original Scale

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = dat_boot, conf = 0.95, type = "norm", index = 2)
##
## Intervals :
## Level      Normal
## 95%   (11.17, 14.08 )
## Calculations and Intervals on Original Scale
```

c

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = dat_boot, conf = 0.95, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   (27.01, 29.86 )
## Calculations and Intervals on Original Scale

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = dat_boot, conf = 0.95, type = "basic", index = 2)
##
## Intervals :
## Level      Basic
## 95%   (11.14, 14.09 )
## Calculations and Intervals on Original Scale
```

d

For the most part, the confidence intervals in part b and part c agree. The confidence interval for γ_1 is much closer since the histogram is pretty much completely normal. This means in part b where we use normal approximation, the confidence intervals will be much closer. For γ_0, we can see from part a that the histogram is slightly left skewed. Thus, the reflected confidence interval is slightly shifted to the right compared to the "crude" confidence interval. Note that the histogram for γ_0 is still very close to normal despite skewedness, and thus the confidence intervals do not differ by much.

Question 5

```
## [1] 18.10226 20.29115

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = dat_boot2, conf = 0.95, type = c("basic"))
##
## Intervals :
## Level      Basic
## 95%   (18.91, 19.72 )
## Calculations and Intervals on Original Scale
```

As seen above, the prediction interval for "future" Y is wider than the confidence interval. This is because of the added uncertainty from the standard errors of the prediction of the model. A prediction interval is a better representation because it not only takes into account the error of the estimated parameters, but also the error in the model that generates the prediction.

Question 6

```
## 'log Lik.' 2.836148 (df=3)
```

```
## 'log Lik.' 1.628871 (df=3)
```

The nls model is better since it has a lower AIC.

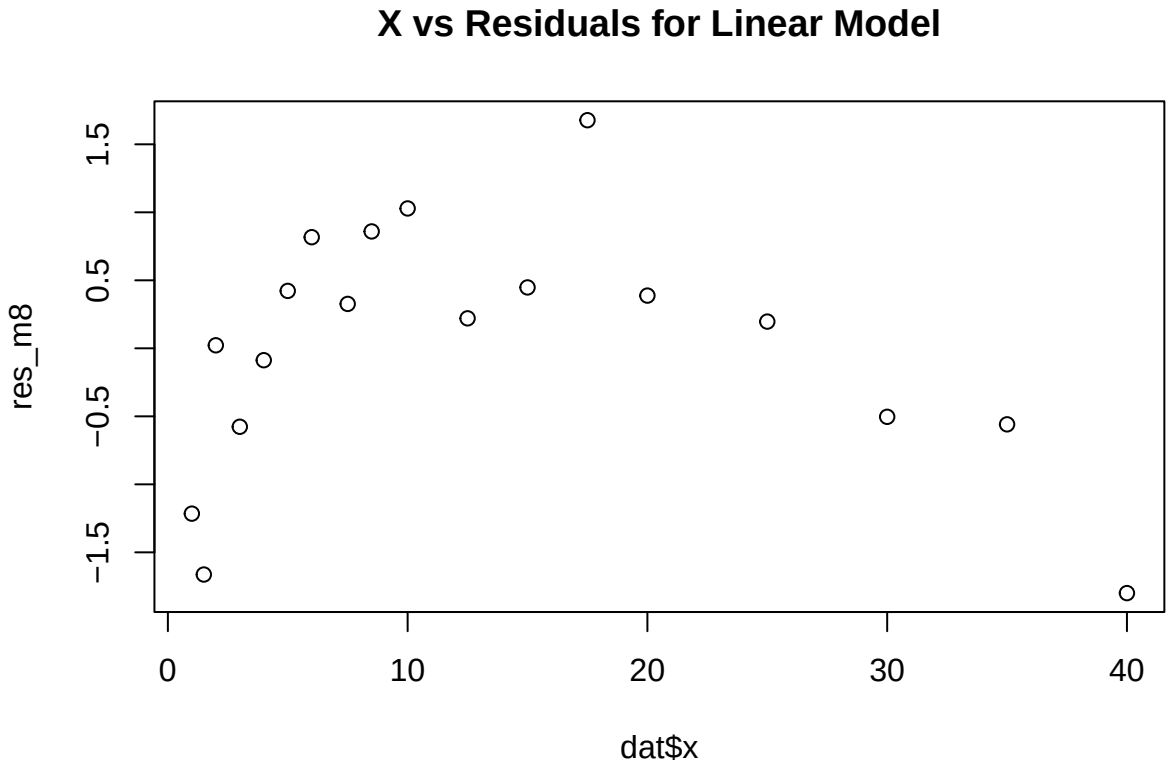
Question 7

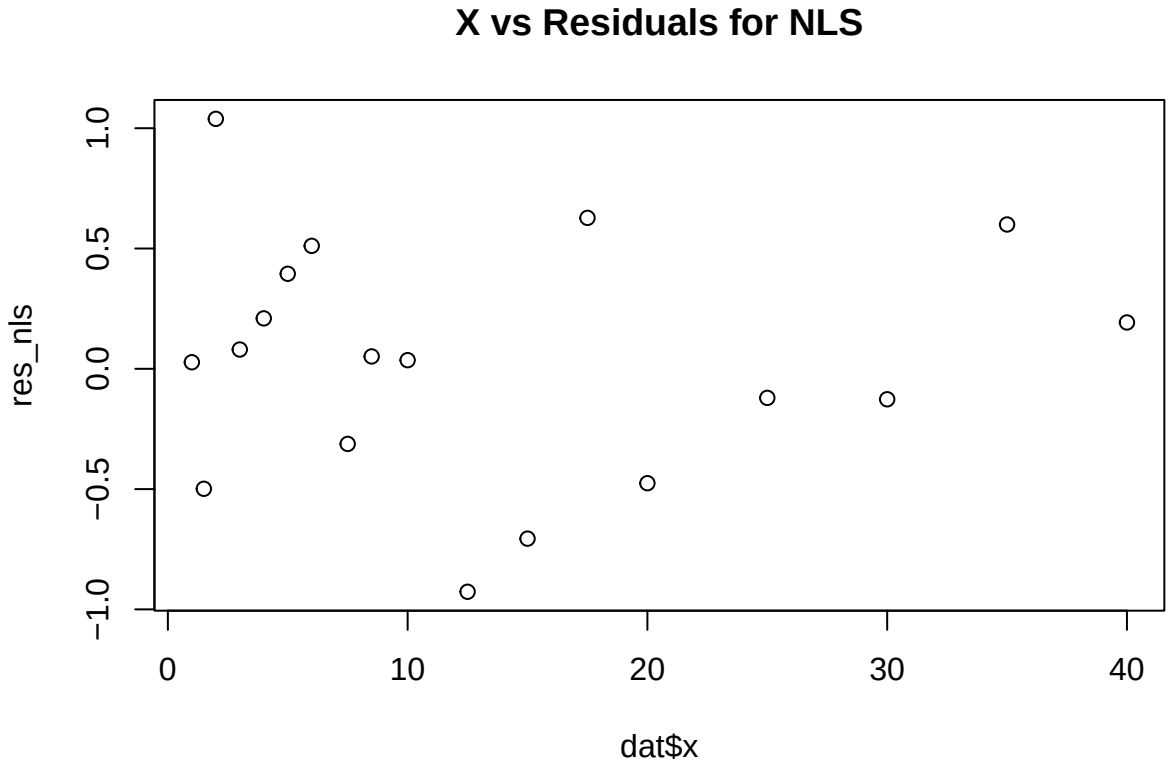
```
##           [,1]      [,2]
##  [1,] 1.175988 0.3116147
##  [2,] 1.175988 0.3116147
##  [3,] 1.175988 0.3116147
##  [4,] 1.175988 0.3116147
##  [5,] 1.175988 0.3116147
##  [6,] 1.175988 0.3116147
##  [7,] 1.175988 0.3116147
##  [8,] 1.175988 0.3116147
##  [9,] 1.175988 0.3116147
## [10,] 1.175988 0.3116147
## [11,] 1.175988 0.3116147
## [12,] 1.175988 0.3116147
## [13,] 1.175988 0.3116147
## [14,] 1.175988 0.3116147
## [15,] 1.175988 0.3116147
```

```
## [1] 1.1759878 0.3116147
```

As seen above, the better model is clearly model 2, which uses nls.

Question 8


X vs Residuals for Linear Model
res_m8
1.5
0.5
−0.5
−1.5
0
10
20
30
40
dat$x


X vs Residuals for NLS
res_nls
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
dat$x

As seen from the plots above, the X vs Residuals plot for the linear model shows non-randomly distributed residuals around the 0 line. There is a clear non-linear pattern, which means that the linear model is not a good fit for the data. On the contrary, the X vs Residuals plot for the non-linear least squares model shows randomly distributed residuals around 0, which suggests there is no further relationship that we missed. Thus, the nls model is a better choice, which is in keeping with results above.

Appendix

This section is to be used for including your R code. The following lines of code will take care of it. Please make sure to comment your code appropriately - in particular, demarcating codes belonging to different questions. Among other things, it will be easier for you to debug your own code.

```
labs = knitr::all_labels()
labs = labs[!labs %in% c("setup","getlabels", "allcode")]
```

```
########### QUESTION 2 begins here ###################
dat <- read_csv("HW1_data.csv")
dat <- dat %>%
  select(y,x)
m1 <- lm(y~x, data = dat)
summary(m1)
est <- c(0.1845, 0.0903)
x_i <- dat$x
y_i <- dat$y
fn <- function(p) {
    yhat <- (p[1] * x_i) / (p[2] + x_i)
    sum((y_i-yhat)^2)
}
out_nlm <- nlm(fn, p = est,hessian=TRUE)
theta_nlm <- out_nlm$estimate  #parameter estimates
theta_nlm
out_nls <- nls(y~((p1 * x) / (p2 + x)), data = dat, start = list(p1 = 0.1845, p2 = 0.0903))
out_nls
########### QUESTION 3 begins here ###################
mse <- out_nlm$minimum / (length(y_i) - length(theta_nlm))
info_mat <- out_nlm$hessian / 2 / mse
cov_theta <- solve(info_mat)
se <- sqrt(diag(cov_theta))
info_mat
cov_theta
se
vcov(out_nls)
p1_left <- theta_nlm[1] - 1.96 * se[1]
p1_right <- theta_nlm[1] + 1.96 * se[1]
p2_left <- theta_nlm[2] - 1.96 * se[2]
p2_right <- theta_nlm[2] + 1.96 * se[2]
p1_left
p1_right
p2_left
p2_right
confint.default(out_nls)
########### QUESTION 4 begins here ###################
```

```r
dat_fit <- function(Z, i, theta0){
  Zboot <- Z[i, ]
  x <- Zboot[[2]]
  y <- Zboot[[1]]
  fn <- function(p){
    yhat <- (p[1] * x) / (p[2] + x)
    sum((y - yhat)^2)
  }
  out <- nlm(fn, p = theta0)
  theta <- out$estimate
}
dat_boot <- boot(dat, dat_fit, R = 20000, theta0 = c(0.1845, 0.0903))
dat_boot
plot(dat_boot, index = 1)
title(main = "Histogram of t0")
plot(dat_boot, index = 2)
title(main = "Histogram of t1")
# gamma 0
boot.ci(dat_boot, conf = 0.95, type = "norm", index = 1)

#gamma 1
boot.ci(dat_boot, conf = 0.95, type = "norm", index = 2)
# gamma 0
boot.ci(dat_boot, conf = 0.95, type = "basic", index = 1)

#gamma 1
boot.ci(dat_boot, conf = 0.95, type = "basic", index = 2)
############ QUESTION 5 begins here ####################

# Bootstrap
dat_fit2 <- function(Z, i, theta0, x_pred){
  Zboot <- Z[i, ]
  x <- Zboot[[2]]
  y <- Zboot[[1]]
  fn <- function(p){
    yhat <- (p[1] * x) / (p[2] + x)
    sum((y - yhat)^2)
  }
  out <- nlm(fn, p = theta0)
  theta <- out$estimate
  y_pred <- (theta[1] * x_pred) / (theta[2] + x_pred)
}
dat_boot2 <- boot(dat, dat_fit2, R = 20000, theta0 = c(0.1845, 0.0903), x_pred = 27)
Yhat0<-dat_boot2$t0
Yhatboot<-dat_boot2$t
SEY<-sqrt(var(Yhatboot)+mse)
c(Yhat0-qnorm(.975)*SEY, Yhat0+qnorm(.975)*SEY)

# Predicted
boot.ci(dat_boot2, conf = 0.95,type=c("basic"))
m6 <- lm(y~sqrt(x), data = dat)
n <- nrow(dat)
log_lik_m6 <- logLik(m6)
```

```r
aic_m6 <- -2 * log_lik_m6 / n + 2 * 2 / n
log_lik_nls <- logLik(out_nls)
aic_nls <- -2 * log_lik_nls / n + 2 * 2 / n
aic_m6
aic_nls
Nrep <- 15 # set to 1 because we are only doing one pass
K <- nrow(dat) # set to the number of observations in the dataset
n.models = 2 #number of different models to fit and compare
FitFun1 <- function(x, p) p[1]+p[2]*sqrt(x)
FitFun2 <- function(x, p) (p[1] * x) / (p[2] + x)
n <- nrow(dat)
yhat <- matrix(0,n,n.models)
MSE <- matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  cv_idx <- cvfolds(n, K)
  for (k in cv_idx) {
    out <- lm(y~sqrt(x), data=dat[-k,])
    yhat[k,1] <- as.numeric(predict(out, dat[k,]))
    out <- nls(y ~ FitFun2(x,p), data=dat[-k,], start=list(p=c(0.1845, 0.0903)))
    yhat[k,2] <- as.numeric(predict(out, dat[k,]))
  }
  y <- dat$y
  MSE[j,] <- apply(yhat,2,function(x) sum((y-x)^2))/(n-1)
}

MSE
MSEAve <- apply(MSE,2,mean); MSEAve #averaged mean square CV error
m8 <- lm(y~sqrt(x), data = dat)
pred_m8 <- predict(m8, dat)
res_m8 <- dat$y - pred_m8

pred_nls <- predict(out_nls, dat)
res_nls <- dat$y - pred_nls

plot(dat$x, res_m8, main = "X vs Residuals for Linear Model")
plot(dat$x, res_nls, main = "X vs Residuals for NLS")
```