HW 1 Solutions

9/7/2020

7.2 Fake-data simulation and regression:

Simulate 100 data points from the linear model, y = a + bx + error, with a = 5, b = 7, the values of x being sampled at random from a uniform distribution on the range [0, 50], and errors that are normally distributed with mean 0 and standard deviation 3.

7.2a

Fit a regression line to these data and display the output.

```
n  <-  100
a  <-  5
b  <-  7
x  <-  runif(n, min=0, max=50)
sigma <-  3
y <- a + b*x + rnorm(n, 0, sigma)
y
```

```
##   [1] 331.867597 263.847153  21.409822 147.665361 221.955593 211.886135
##   [7] 241.414963 300.623640 237.400392 278.677522 278.115022  46.256255
##  [13] 243.453216  75.702919 183.444783 123.861460 105.732528 352.853849
##  [19] 114.221453 141.729398 165.254041 268.091869  76.165484 355.132761
##  [25] 336.013724 324.242443  95.960717  13.749144 210.588973 343.165338
##  [31] 291.731506  66.538155 240.344448 237.568496  82.813791  26.933189
##  [37] 248.130095 249.249849  30.186638 304.298691  64.863510 322.136838
##  [43] 154.388735  65.211078 255.508385  94.858880 287.946127 315.898119
##  [49]  14.358836 186.973820 223.495962 311.162860 296.820601 155.956427
##  [55] 317.233550 252.319433 252.774704 346.515774 301.759212 233.392923
##  [61]  70.038838 213.801474 159.252644 113.714999 157.997917  68.561532
##  [67] 244.024587 101.389352  77.538114  95.886869 201.619222 245.885527
##  [73] 149.037545 143.630593 124.914005  43.551259 143.034634 257.798456
##  [79] 169.014298   9.847328  84.377370  50.759799 114.914784  46.838545
##  [85] 271.706053 127.580296 182.386872  53.411999  38.991066 195.090273
##  [91] 180.958686 143.630399 227.566787  98.296961 307.102096 149.356749
##  [97] 216.069546 318.473947 330.228874  68.712697
```

7.2b

Graph a scatterplot of the data and the regression line.

```
fake <- data.frame(x,y)
fit_1 <- stan_glm(y~x, data=fake)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000102 seconds
```

```
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.02 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.133023 seconds (Warm-up)
## Chain 1:                0.081599 seconds (Sampling)
## Chain 1:                0.214622 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.117152 seconds (Warm-up)
## Chain 2:                0.087079 seconds (Sampling)
## Chain 2:                0.204231 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
```

```
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.105082 seconds (Warm-up)
## Chain 3:                0.08451 seconds (Sampling)
## Chain 3:                0.189592 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.099053 seconds (Warm-up)
## Chain 4:                0.081402 seconds (Sampling)
## Chain 4:                0.180455 seconds (Total)
## Chain 4:
```

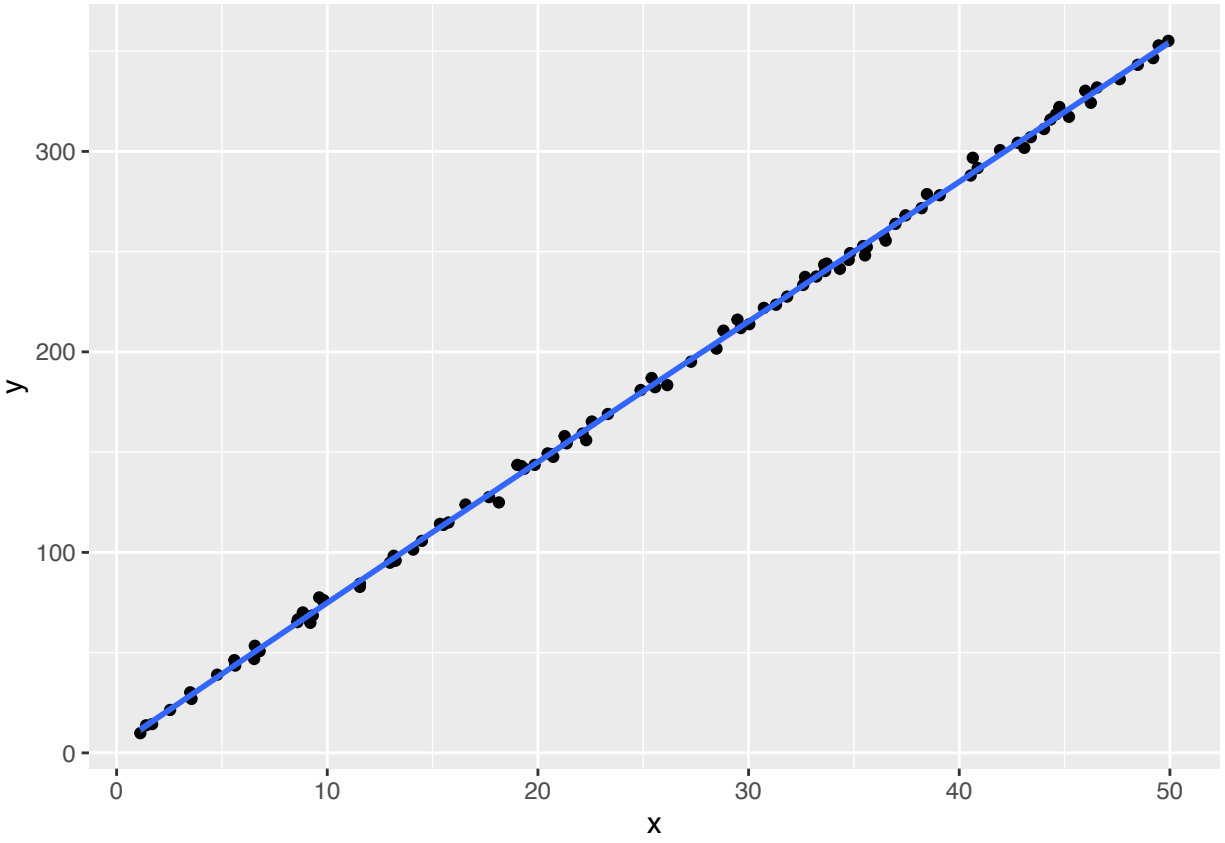
```
print(fit_1,digits=2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x
##  observations: 100
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 4.82   0.57
## x           7.00   0.02
##
```

```
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 2.74   0.19
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
ggplot(data=fake, mapping = aes(x = x,y = y))+
  geom_point() +
  geom_smooth()
```

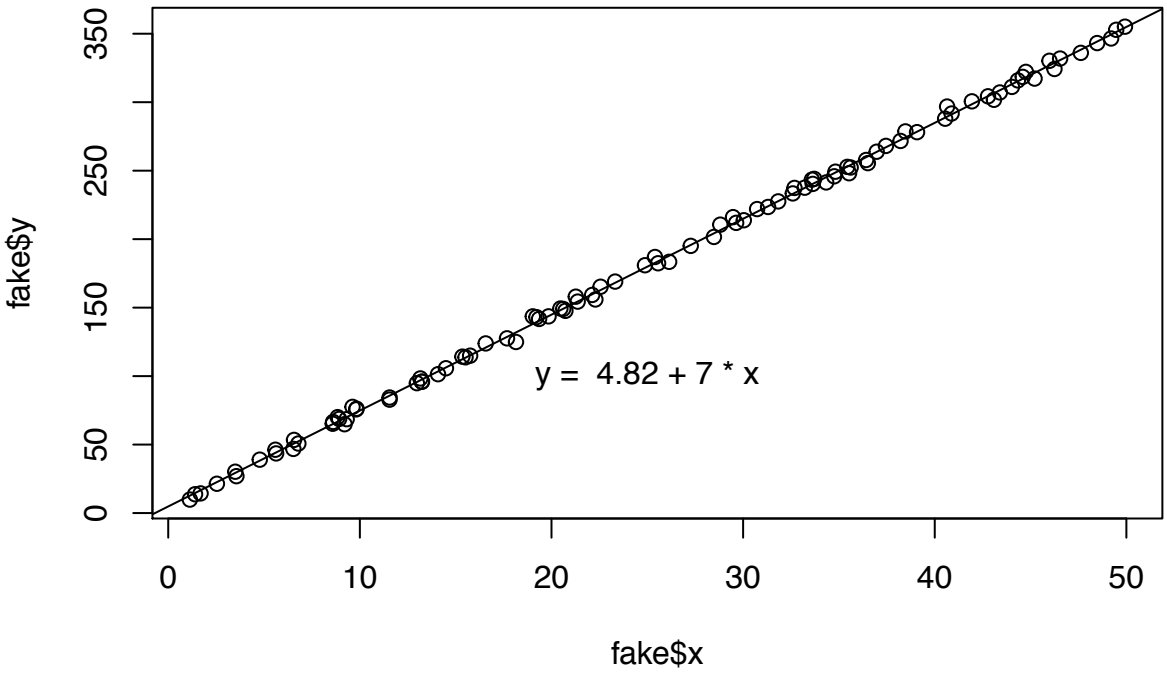
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



7.2c

Use the text function in R to add the formula of the fitted line to the graph.

```
plot(fake$x,fake$y)
a_hat <- coef(fit_1)[1]
b_hat <- coef(fit_1)[2]
abline(a_hat, b_hat)
text(25 ,100, paste("y = ", round(a_hat, 2),"+", round(b_hat,2),"* x"))
```



7.3 Fake-data simulation and fitting the wrong model:

Simulate 100 data points from the model, y = a + bx + cx2 + error, with the values of x being sampled at random from a uniform distribution on the range [0, 50], errors that are normally distributed with mean 0 and standard deviation 3, and a, b, c chosen so that a scatterplot of the data shows a clear nonlinear curve.

7.3 a

Fit a regression line stan_glm(y ~ x) to these data and display the output.

```
a <- 5; b <- 7;c <- 9
n  <-  100
x  <-  runif(n, min=0, max=50)
sigma <-  3
y <- a+b*x+c*x^2+rnorm(n,0,sigma)
y
```

```
##   [1]  5397.362183  4508.465185 19791.002270    58.717748 21684.072758
##   [6] 10652.246510  2427.811833 22293.187842  8953.458483 17074.872476
##  [11] 12490.262254  3098.597680 11069.362199   380.319581  1532.414295
##  [16]   396.134850  4033.782250  7740.215693  7776.249060 13125.206567
##  [21] 17634.967247 20178.979281  1010.596279   554.299241   315.427396
##  [26]   384.561810 17566.613965  1550.938945   505.510244  3813.912648
##  [31]   430.522169 18062.174110    81.450884   197.117942 20043.333936
##  [36]   367.106738  5889.622858     9.894689 14498.794618     6.705210
##  [41]    47.096352  8165.664723 11778.932541  3451.540810 21833.967052
##  [46]    20.782163  1024.605782 11598.620487 20857.220101  4630.955403
##  [51]  6902.954837  3515.468045  1725.219035 13393.767115 11526.434799
##  [56] 13704.123324 11561.291598  6619.060759  5285.263447   523.345644
##  [61]  9198.854577  8714.711827 22206.388370 18607.692266 17049.997066
##  [66]  9840.406371   596.868553  6756.396149 11972.137489   123.214305
##  [71] 11136.259821  3684.640161 15677.899149  1232.069526 15964.026048
##  [76] 18940.385715 14681.657652  3196.305594  3244.492058 19360.314549
```

```
##  [81] 15549.126669  6284.296324    34.111002  6875.941122 20222.936009
##  [86]  2267.589505  2113.886125     8.792940   337.839679    50.149401
##  [91] 10102.830614  1772.664178  3337.369694 17531.415413  3323.521928
##  [96] 20149.984865  8056.121940 20165.067550 19603.627266 11017.378531
```

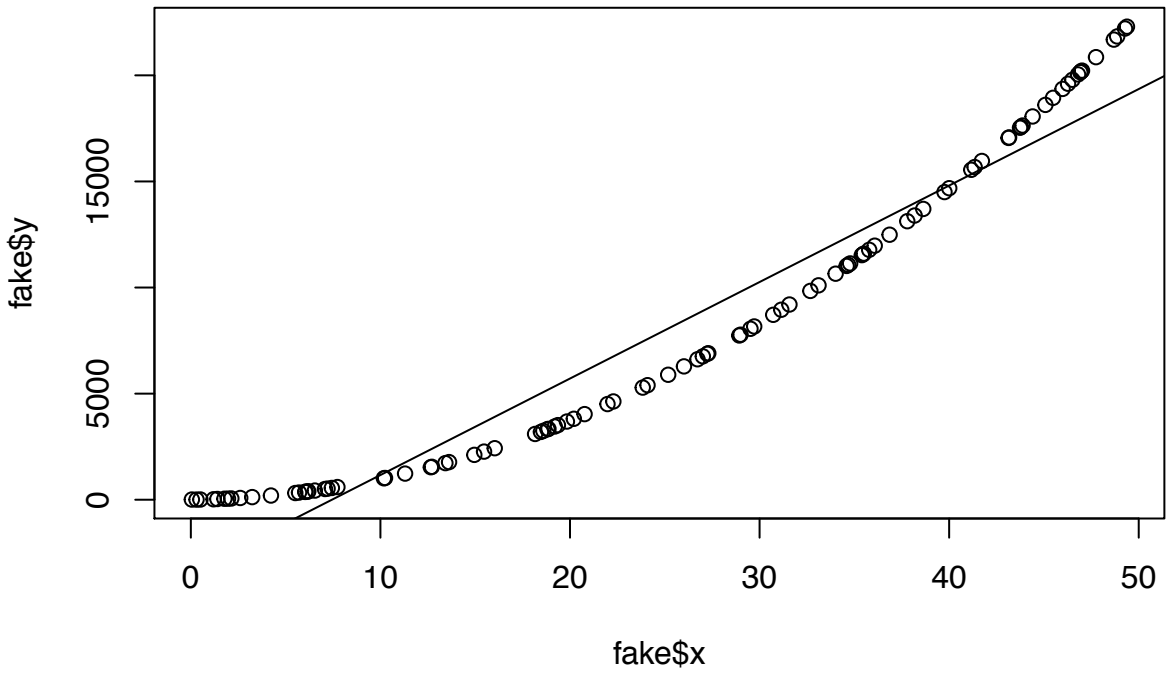
```
fake <- data.frame(x,y)
fit_2 <- stan_lm(y~x, data=fake, prior_intercept = NULL, prior = NULL,prior_aux=NULL, refresh = 0)
fit_2
```

```
## stan_lm
##  family:       gaussian [identity]
##  formula:      y ~ x
##  observations: 100
##  predictors:   2
## ------
##             Median  MAD_SD
## (Intercept) -3384.9   360.2
## x             454.8    11.7
##
## Auxiliary parameter(s):
##               Median MAD_SD
## R2               0.9    0.0
## log-fit_ratio    0.0    0.0
## sigma         1818.3  131.8
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

7.3b

Graph a scatterplot of the data and the regression line. This is the best-fit linear regression. What does "best-fit" mean in this context?

```
plot(fake$x,fake$y)
a_hat <- coef(fit_2)[1]
b_hat <- coef(fit_2)[2]
abline(a_hat, b_hat)
```



7.6 Formulating comparisons as regression models:

Take the election forecasting model and simplify it by creating a binary predictor defined as x = 0 if income growth is less than 2% and x = 1 if income growth is more than 2%.

```
hibbs <- read.table(file = "/Users/mac/Desktop/BU Mssp/MA678/ROS-Examples-master/ElectionsEconomy/data/
hibbs
```

```
##    year growth  vote inc_party_candidate other_candidate
## 1  1952   2.40 44.60           Stevenson      Eisenhower
## 2  1956   2.89 57.76          Eisenhower       Stevenson
## 3  1960   0.85 49.91               Nixon         Kennedy
## 4  1964   4.21 61.34             Johnson       Goldwater
## 5  1968   3.02 49.60            Humphrey           Nixon
## 6  1972   3.62 61.79               Nixon        McGovern
## 7  1976   1.08 48.95                Ford          Carter
## 8  1980  -0.39 44.70              Carter          Reagan
## 9  1984   3.86 59.17              Reagan         Mondale
## 10 1988   2.27 53.94          Bush, Sr.         Dukakis
## 11 1992   0.38 46.55          Bush, Sr.         Clinton
## 12 1996   1.04 54.74             Clinton            Dole
## 13 2000   2.36 50.27                Gore       Bush, Jr.
## 14 2004   1.72 51.24          Bush, Jr.           Kerry
## 15 2008   0.10 46.32              McCain           Obama
## 16 2012   0.95 52.00               Obama          Romney
```

```
head(hibbs)
```

```
##   year growth  vote inc_party_candidate other_candidate
## 1 1952   2.40 44.60           Stevenson      Eisenhower
## 2 1956   2.89 57.76          Eisenhower       Stevenson
## 3 1960   0.85 49.91               Nixon         Kennedy
## 4 1964   4.21 61.34             Johnson       Goldwater
## 5 1968   3.02 49.60            Humphrey           Nixon
```

```
## 6 1972   3.62 61.79               Nixon        McGovern
```

```
new <- 1:nrow(hibbs)
hibbs_new <- cbind(hibbs, new)
for (i in 1:nrow(hibbs_new))
  if (hibbs_new$growth[i] < 2){hibbs_new$new[i] = 0}else {hibbs_new$new[i] = 1}
hibbs_new
```

```
##    year growth  vote inc_party_candidate other_candidate new
## 1  1952   2.40 44.60           Stevenson      Eisenhower   1
## 2  1956   2.89 57.76          Eisenhower       Stevenson   1
## 3  1960   0.85 49.91               Nixon         Kennedy   0
## 4  1964   4.21 61.34             Johnson       Goldwater   1
## 5  1968   3.02 49.60            Humphrey           Nixon   1
## 6  1972   3.62 61.79               Nixon        McGovern   1
## 7  1976   1.08 48.95                Ford          Carter   0
## 8  1980  -0.39 44.70              Carter          Reagan   0
## 9  1984   3.86 59.17              Reagan         Mondale   1
## 10 1988   2.27 53.94           Bush, Sr.         Dukakis   1
## 11 1992   0.38 46.55           Bush, Sr.         Clinton   0
## 12 1996   1.04 54.74             Clinton            Dole   0
## 13 2000   2.36 50.27                Gore       Bush, Jr.   1
## 14 2004   1.72 51.24           Bush, Jr.           Kerry   0
## 15 2008   0.10 46.32              McCain           Obama   0
## 16 2012   0.95 52.00               Obama          Romney   0
```

7.6a

Compute the difference in incumbent party's vote share on average, comparing those two groups of elections, and determine the standard error for this difference.

```
#For the difference
less <- hibbs_new$vote[hibbs_new$new == 0]
more <- hibbs_new$vote[hibbs_new$new == 1]
dif <- mean(more) - mean(less)
print(paste("The difference is ", round(dif,4)))
```

```
## [1] "The difference is  5.5075"
```

```
#For the standard error
sd_less <- sd(less)/sqrt(length(less))
sd_more <- sd(more)/sqrt(length(more))
sd <- sqrt(sd_more^2 + sd_less^2)
print(paste("The standard error for this difference is ", round(sd,4)))
```

```
## [1] "The standard error for this difference is  2.5021"
```

7.6b

Regress incumbent party's vote share on the binary predictor of income growth and check that the resulting estimate and standard error are the same as above.

```
x <- hibbs_new$new
y <- hibbs_new$vote
fit <- lm(y~x)
summary(fit)
```

```
##
```

```
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.2087  -3.3706   0.1287   3.3037   6.9812
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   49.301      1.769  27.866 1.15e-13 ***
## x              5.508      2.502   2.201    0.045 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.004 on 14 degrees of freedom
## Multiple R-squared:  0.2571, Adjusted R-squared:  0.204
## F-statistic: 4.845 on 1 and 14 DF,  p-value: 0.045
```

8.8 Comparing lm and stan_glm:

Use simulated data to compare least squares estimation to default Bayesian regression:

8.8a

Simulate 100 data points from the model, y = 2 + 3x + error, with predictors x drawn from a uniform distribution from 0 to 20, and with independent errors drawn from the normal distribution with mean 0 and standard deviation 5. Fit the regression of y on x data using lm and stan_glm (using its default settings) and check that the two programs give nearly identical results.

```
#Creat data
n <- 100
x <- runif(n,min=0, max=20)
err <- rnorm(n, 0, 5)
y <- 2 + 3*x + err
fake <- data.frame(x,y)
#Fit the data with lm
lm <- lm(y~x, data = fake)
lm
```

```
##
## Call:
## lm(formula = y ~ x, data = fake)
##
## Coefficients:
## (Intercept)            x
##      0.5791       3.1027
```

```
#Fit the data with stan_lm
lm_stan <- stan_lm(y~x, data=fake, prior_intercept = NULL, prior = NULL,prior_aux=NULL, refresh = 0)
lm_stan
```

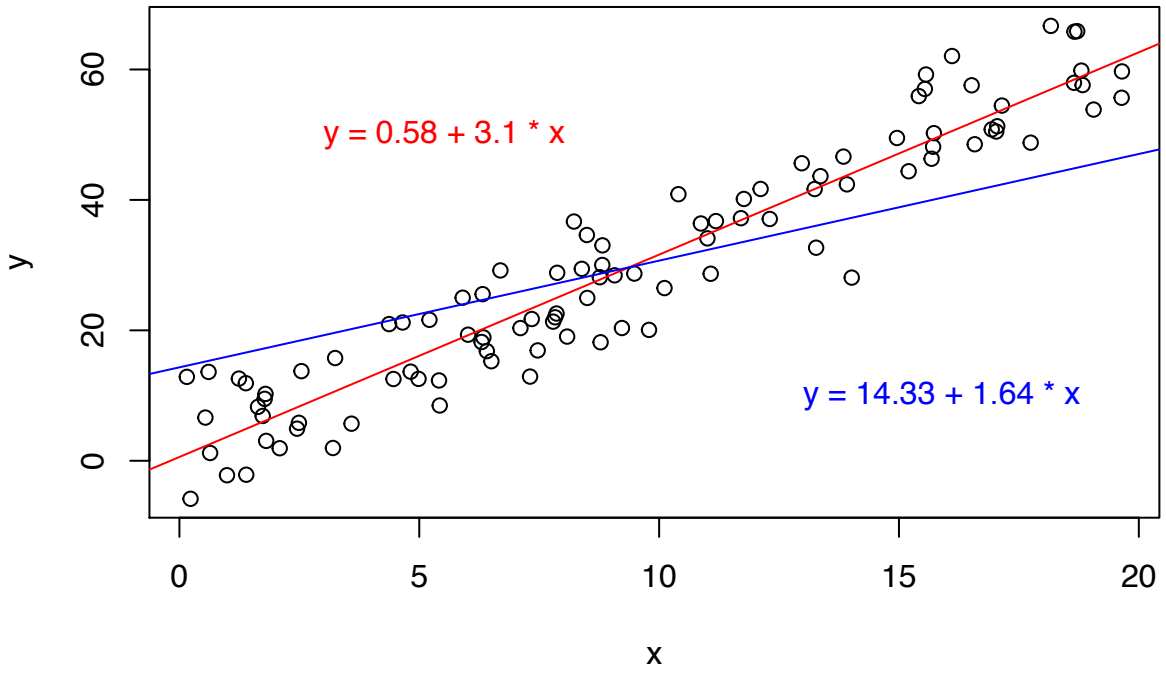
```
## stan_lm
##  family:       gaussian [identity]
##  formula:      y ~ x
##  observations: 100
##  predictors:   2
```

```
## ------
##             Median MAD_SD
## (Intercept) 14.3    1.6
## x            1.6    0.1
##
## Auxiliary parameter(s):
##               Median MAD_SD
## R2             1.0    0.0
## log-fit_ratio -0.7    0.1
## sigma         13.6    0.8
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

8.8b

Plot the simulated data and the two fitted regression lines.

```
#Calculating all the coefficient
a_hat1 <- coef(lm)[1]
b_hat1 <- coef(lm)[2]
a_hat2 <- coef(lm_stan)[1]
b_hat2 <- coef(lm_stan)[2]
#plot these two model
plot(x,y)
abline(a_hat1, b_hat1,col="red")
text(x=3, y=50,paste("y =", round(a_hat1, 2), "+", round(b_hat1, 2), "* x"), adj=0, col = "red")
abline(a_hat2, b_hat2,col="blue")
text(x=13, y= 10,paste("y =", round(a_hat2, 2), "+", round(b_hat2, 2), "* x"), adj=0, col="blue")
```

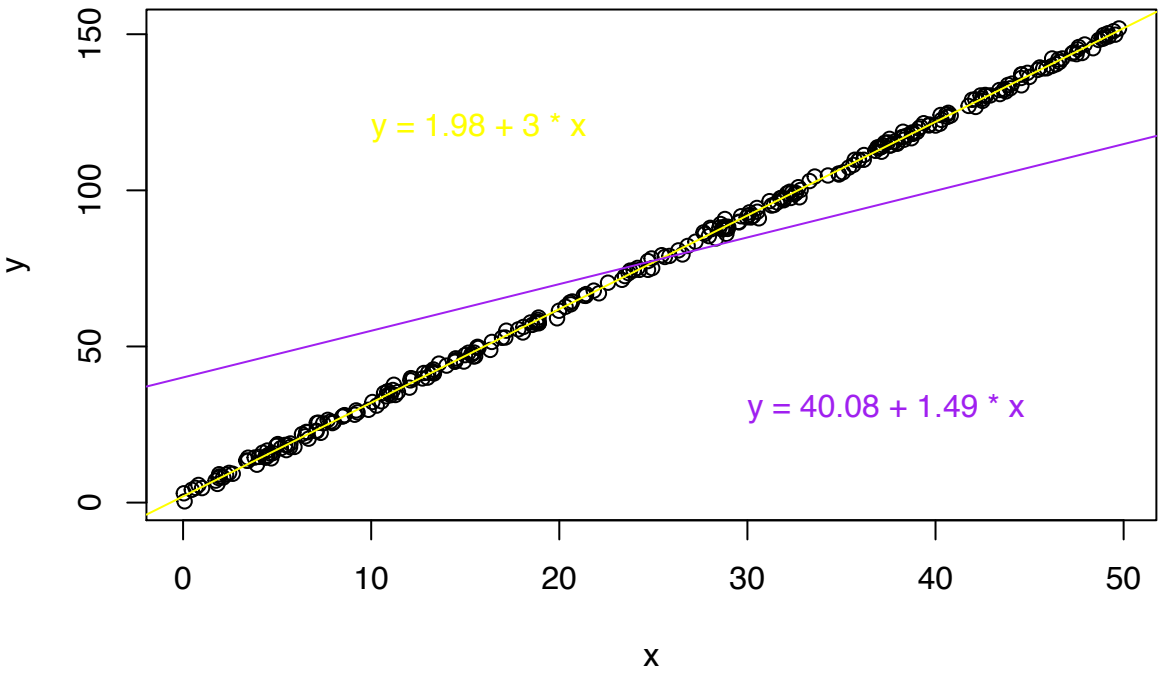


8.8c

Repeat the two steps above, but try to create conditions for your simulation so that lm and stan_glm give much different results.

```
##
## Call:
## lm(formula = y ~ x, data = fake)
##
## Coefficients:
## (Intercept)            x
##       1.983        2.999
```

```
## stan_lm
##  family:       gaussian [identity]
##  formula:      y ~ x
##  observations: 300
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 40.1    2.2
## x            1.5    0.1
##
## Auxiliary parameter(s):
##               Median MAD_SD
## R2             1.0    0.0
## log-fit_ratio -0.7    0.0
## sigma         31.4    1.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```



10.1 Regression with interactions:

Simulate 100 data points from the model, y = b0 + b1 x + b2 z + b3 xz + error, with a continuous predictor x and a binary predictor z, coefficients b = c(1, 2, -1, -2), and errors drawn independently from a normal distribution with mean 0 and standard deviation 3, as follows. For each data point i, first draw zi, equally likely to take on the values 0 and 1. Then draw xi from a normal distribution with mean zi and standard deviation 1. Then draw the error from its normal distribution and compute yi.

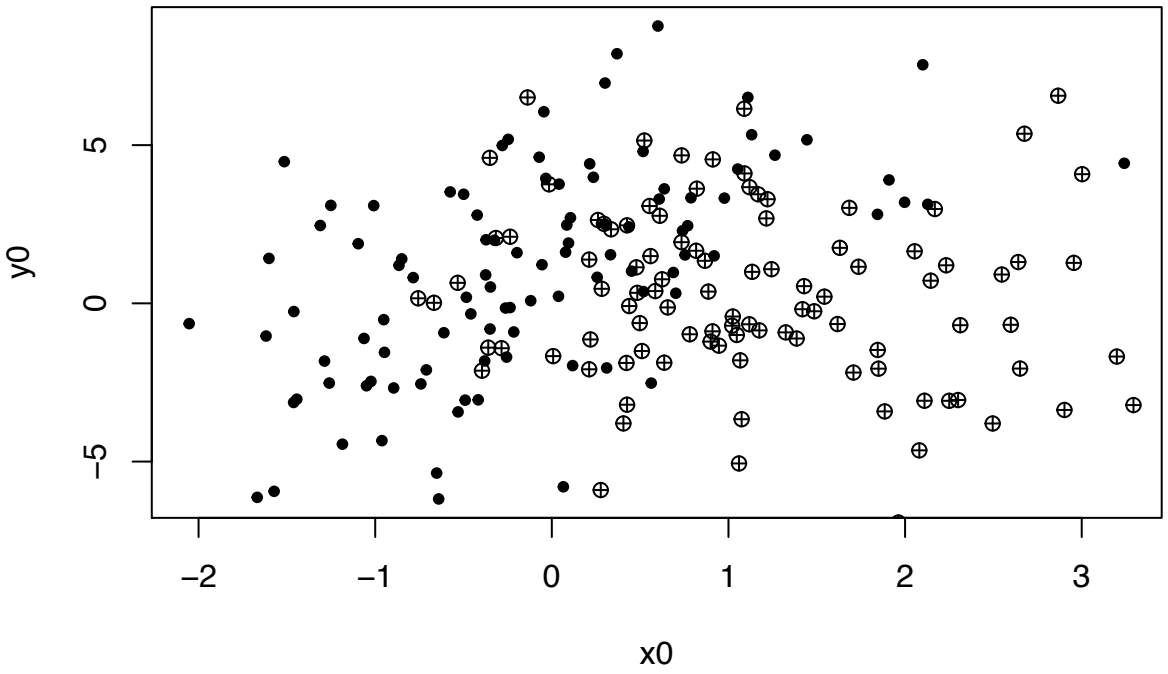
10.1a

Display your simulated data as a graph of y vs. x, using dots and circles for the points with z = 0 and 1, respectively.

```
#Stimulate the required data
set.seed(123)
n <- 100
err <- rnorm(n, 0, 3)
z0 <- rep(0,times=50)
z1 <- rep(1,times=50)
x0 <- rnorm(n, mean=mean(z0),1)
x1 <- rnorm(n, mean=mean(z1),1)
x <- c(x0,x1)

#Compute Y
y0 <- 1 + 2*x0 + err
y1 <- 1 + 2*x1 - 1 - 2*x1 + err
y <- c(y0,y1)

#Plot these data
plot(x0,y0,pch=20)
points(x1,y1,pch=10)
```



10.1b

Fit a regression predicting y from x and z with no interaction. Make a graph with the data and two parallel lines showing the fitted model.

```
#Stimulate data with no interaction.
set.seed(123)
n <- 100
err <- rnorm(n, 0, 3)
z0 <- rep(0,times=50)
z1 <- rep(1,times=50)
x0 <- rnorm(n, mean=mean(z0),1)
x1 <- rnorm(n, mean=mean(z1),1)
x <- c(x0,x1)

#Computing Y
y0 <- 1 + 2*x0 + err
y1 <- 1 + 2*x1 - 1 + err
y <- c(y0,y1)

#To fit the model
fake0 <- data.frame(x0,y0)
fake1 <- data.frame(x1,y1)
model_1 <- lm(y0~x0, data = fake0)
model_1
```

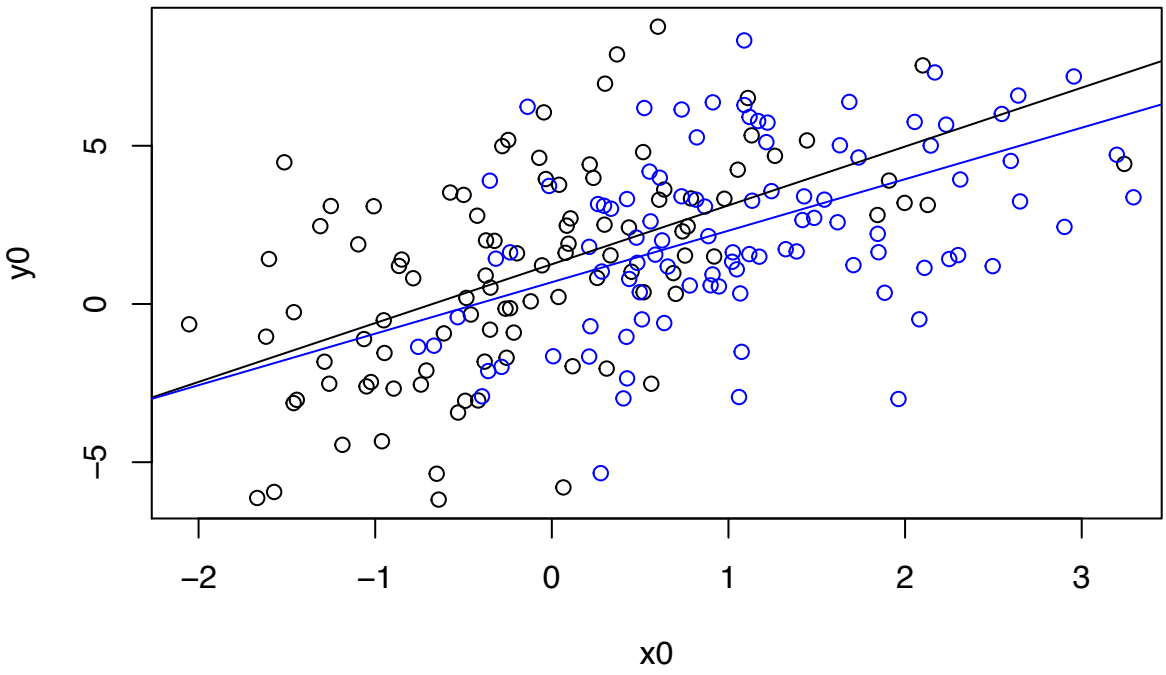
```
##
## Call:
## lm(formula = y0 ~ x0, data = fake0)
##
## Coefficients:
## (Intercept)           x0
##       1.256        1.860
```

```
model_2 <- lm(y1~x1, data = fake1)
model_2
```

```
##
## Call:
## lm(formula = y1 ~ x1, data = fake1)
##
## Coefficients:
## (Intercept)           x1
##      0.6885       1.6276
```

```
#Calculating the coefficient of data
ahat_0 <- coef(model_1)[1]
bhat_0 <- coef(model_1)[2]
ahat_1 <- coef(model_2)[1]
bhat_1 <- coef(model_2)[2]

#Making the final plot
plot(x0, y0)
points(x1,y1,col="blue")
abline(ahat_0, bhat_0)
abline(ahat_1, bhat_1,col="blue")
```



10.1c

Fit a regression predicting y from x, z, and their interaction. Make a graph with the data and two lines showing the fitted model.

```
#Stimulate data with no interaction.
set.seed(123)
n <- 100
err <- rnorm(n, 0, 3)
z0 <- rep(0,times=50)
z1 <- rep(1,times=50)
x0 <- rnorm(n, mean=mean(z0),1)
x1 <- rnorm(n, mean=mean(z1),1)
x <- c(x0,x1)

#Computing Y
y0 <- 1 + 2*x0 + err
y1 <- 1 + 2*x1 - 1 -2*x1 + err
y <- c(y0,y1)

#To fit the model
fake0 <- data.frame(x0,y0)
fake1 <- data.frame(x1,y1)
model_1 <- lm(y0~x0, data = fake0)
model_1
```

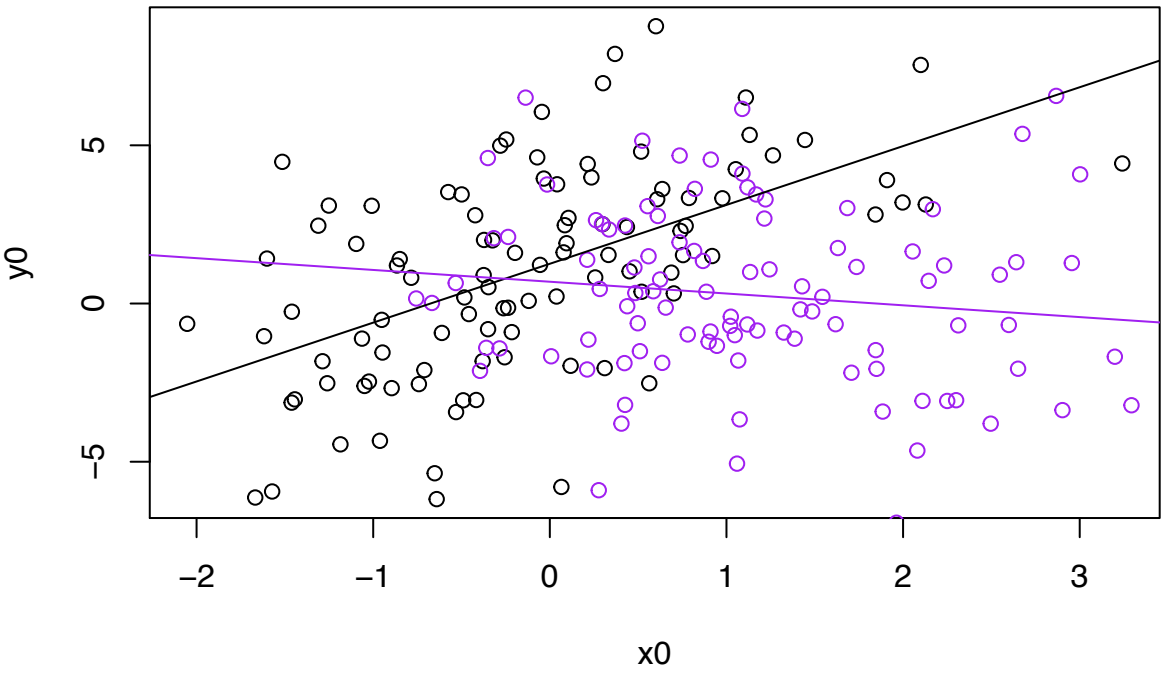
```
##
## Call:
## lm(formula = y0 ~ x0, data = fake0)
##
## Coefficients:
## (Intercept)           x0
##       1.256        1.860
```

```r
model_2 <- lm(y1~x1, data = fake1)
model_2
```

```
##
## Call:
## lm(formula = y1 ~ x1, data = fake1)
##
## Coefficients:
## (Intercept)           x1
##      0.6885      -0.3724
```

```r
#Calculating the coefficient of data
ahat_0 <- coef(model_1)[1]
bhat_0 <- coef(model_1)[2]
ahat_1 <- coef(model_2)[1]
bhat_1 <- coef(model_2)[2]

#Making the final plot
plot(x0, y0)
points(x1,y1,col="purple")
abline(ahat_0, bhat_0)
abline(ahat_1, bhat_1,col="purple")
```



10.2 Regression with interactions:

Here is the output from a fitted linear regression of outcome y on pre-treatment predictor x, treatment indicator z, and their interaction:

10.2a

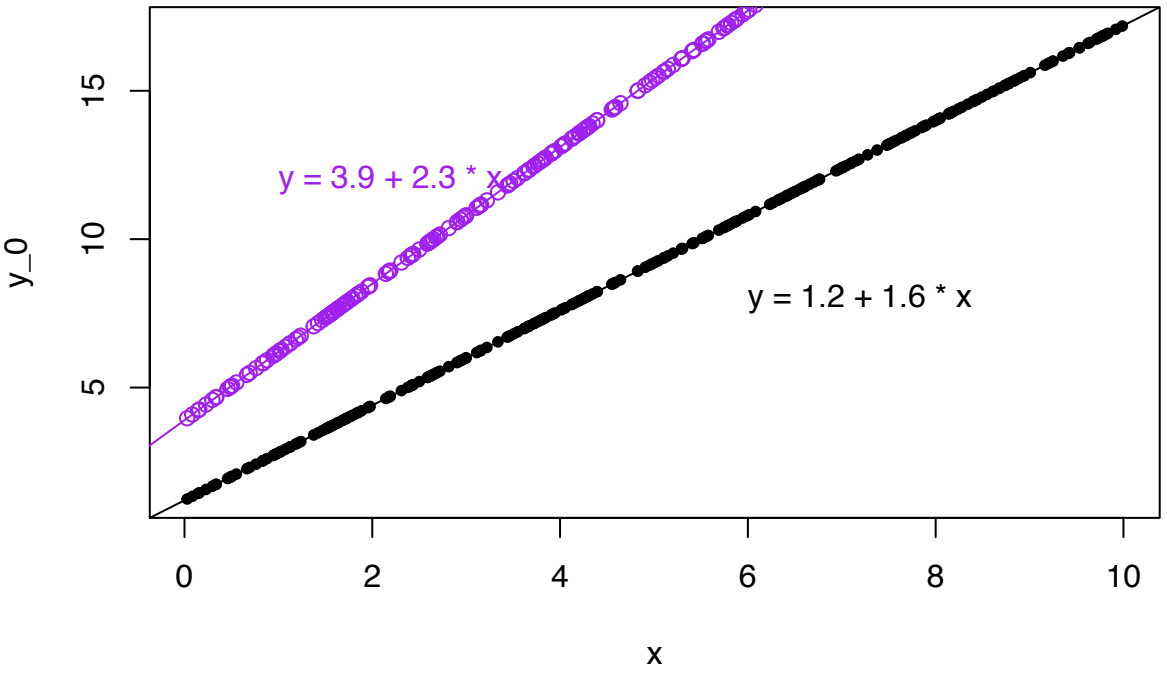
Write the equation of the estimated regression line of y on x for the treatment group and the control group, and the equation of the estimated regression line of y on x for the control group.

```
# The estimated regression line of the treatment group is:
# y = 3.9 + 2.3x
# While the estimated regression line of the control group is:
# y = 1.2 + 1.6x
```

10.2b

Graph with pen on paper the two regression lines, assuming the values of x fall in the range (0, 10). On this graph also include a scatterplot of data (using open circles for treated units and dots for controls) that are consistent with the fitted model.

```
set.seed(12)
x <-  runif(300,min=0,max=10)
y_0 <- 1.2 + 1.6*x
y_1 <- 3.9 + 2.3*x
# Plot the data
plot(x,y_0,pch=20)
points(x,y_1,pch=21,col="purple")
abline(1.2000, 1.6000)
text(x=6, y=8,paste("y =", round(1.2, 2), "+", round(1.6, 2), "* x"), adj=0)
# Plot the final graph
abline(3.9000, 2.3000,col="purple")
text(x=1, y=12,paste("y =", round(3.9, 2), "+", round(2.3, 2), "* x"), adj=0, col="purple")
```



10.5 Regression modeling and prediction:

The folder KidIQ contains a subset of the children and mother data discussed earlier in the chapter. You have access to children's test scores at age 3, mother's education, and the mother's age at the time she gave birth for a sample of 400 children.

```
# Load the required data
kidiq <- read.csv(file = "/Users/mac/Desktop/BU Mssp/MA678/ROS-Examples-master/KidIQ/data/kidiq.csv", h
head(kidiq)
```

```
##   kid_score mom_hs    mom_iq mom_work mom_age
## 1        65      1 121.11753        4      27
## 2        98      1  89.36188        4      25
## 3        85      1 115.44316        4      27
## 4        83      1  99.44964        3      25
## 5       115      1  92.74571        4      27
## 6        98      0 107.90184        1      18
```

10.5a

Fit a regression of child test scores on mother's age, display the data and fitted model, check assumptions, and interpret the slope coefficient. Based on this analysis, when do you recommend mothers should give birth? What are you assuming in making this recommendation?

```
test_age <- lm(kidiq$kid_score~kidiq$mom_age,data=kidiq)
summary(test_age)
```

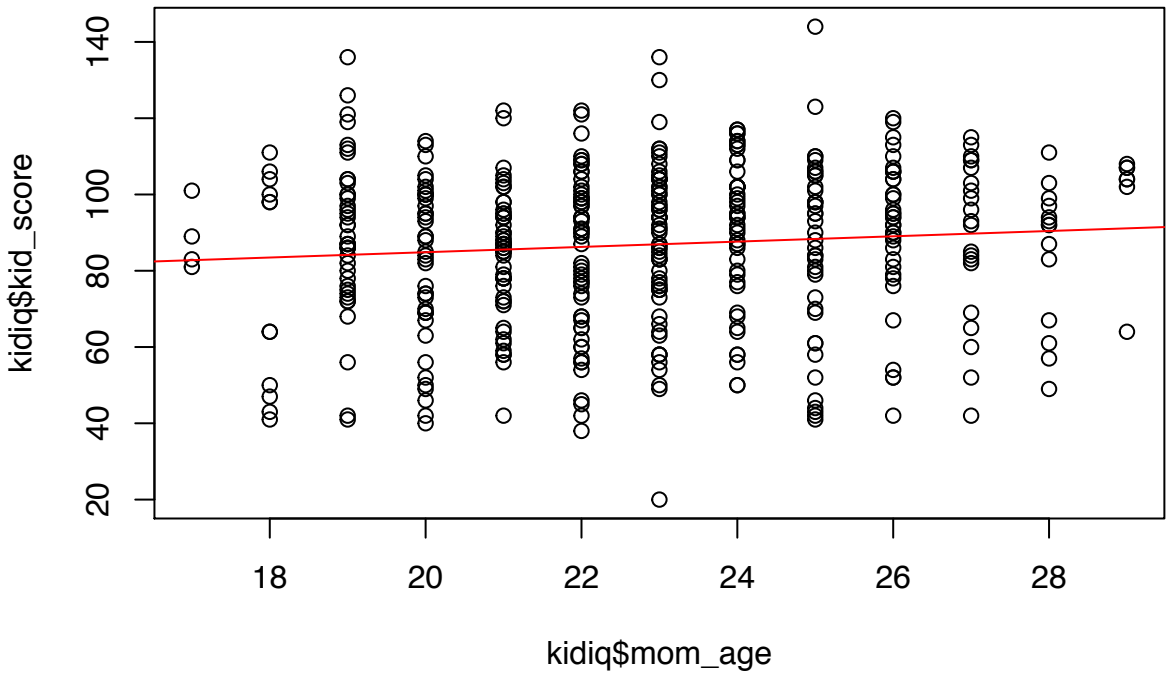
```
##
## Call:
## lm(formula = kidiq$kid_score ~ kidiq$mom_age, data = kidiq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.946 -11.925   3.097  14.694  55.663
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    70.9569     8.3065   8.542 2.28e-16 ***
## kidiq$mom_age   0.6952     0.3620   1.920   0.0555 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.35 on 432 degrees of freedom
## Multiple R-squared:  0.008464,   Adjusted R-squared:  0.006168
## F-statistic: 3.688 on 1 and 432 DF,  p-value: 0.05548
```

```
print(paste("The average of mom_age is",round(mean(kidiq$mom_age),2)))
```

```
## [1] "The average of mom_age is 22.79"
```

```
#So we can know that the coefficient of mom_age means when the age of a mother changes x units, and the

# Plot the graph
plot(x=kidiq$mom_age,y=kidiq$kid_score)
ahat <- coef(test_age)[1]
bhat <- coef(test_age)[2]
abline(ahat,bhat,col="red")
```




```
#By watching this plot, the recommended age for a mother to have a child is 23 years old.
#This assumption that can make such recommendation is that, there are no other factors that can affect
```

10.5b

Repeat this for a regression that further includes mother's education, interpreting both slope coefficients in this model. Have your conclusions about the timing of birth changed?

```
test_edu = lm(kidiq$kid_score~kidiq$mom_hs+kidiq$mom_age,data=kidiq)
summary(test_edu)
```

```
##
## Call:
## lm(formula = kidiq$kid_score ~ kidiq$mom_hs + kidiq$mom_age,
##     data = kidiq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -57.980 -12.545   2.057  14.709  59.325
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    70.4787     8.1068   8.694  < 2e-16 ***
## kidiq$mom_hs   11.3112     2.3783   4.756  2.7e-06 ***
## kidiq$mom_age   0.3261     0.3617   0.902    0.368
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.86 on 431 degrees of freedom
## Multiple R-squared:  0.05791,    Adjusted R-squared:  0.05353
## F-statistic: 13.25 on 2 and 431 DF,  p-value: 2.614e-06
```

```
# The coefficient of mom_hs means compared to mothers who don't have a high school degree and have a ki
```

```
# While in this condition, my recommendation of the age for a mother to have a baby do not change, it i
```

10.5c

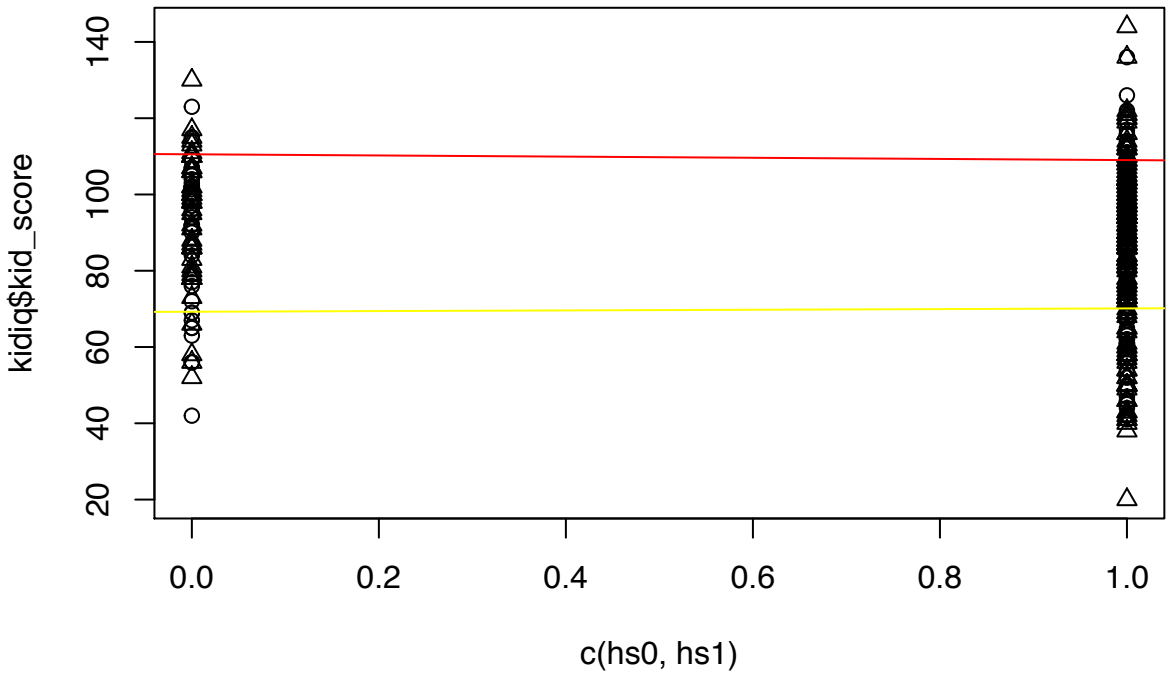
Now create an indicator variable reflecting whether the mother has completed high school or not. Consider interactions between high school completion and mother's age. Also create a plot that shows the separate regression lines for each high school completion status group.

```
test_int = lm(kidiq$kid_score~kidiq$mom_hs + kidiq$mom_age + kidiq$mom_hs: kidiq$mom_age,data=kidiq)
summary(test_int)
```

```
##
## Call:
## lm(formula = kidiq$kid_score ~ kidiq$mom_hs + kidiq$mom_age +
##     kidiq$mom_hs:kidiq$mom_age, data = kidiq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.535 -12.734   2.414  14.150  54.377
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 110.5417    16.4538   6.718 5.85e-11 ***
## kidiq$mom_hs                -41.2875    18.9920  -2.174  0.03025 *
## kidiq$mom_age                -1.5220     0.7532  -2.021  0.04391 *
## kidiq$mom_hs:kidiq$mom_age    2.3911     0.8567   2.791  0.00549 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.7 on 430 degrees of freedom
## Multiple R-squared:  0.07467,    Adjusted R-squared:  0.06822
## F-statistic: 11.57 on 3 and 430 DF,  p-value: 2.64e-07
```

```
# Coefficients
ahat <-  coef(test_int)[1]
bhat <-  coef(test_int)[2]
chat <-  coef(test_int)[3]
dhat <-  coef(test_int)[4]

# Making the final plot
hs0 = kidiq$mom_hs[kidiq$mom_hs==0]
hs1 = kidiq$mom_hs[kidiq$mom_hs==1]
plot(x=c(hs0,hs1),y=kidiq$kid_score,pch=c(1,2))
abline(ahat,chat,col="red")
abline((ahat+bhat),(chat+dhat),col="yellow")
```



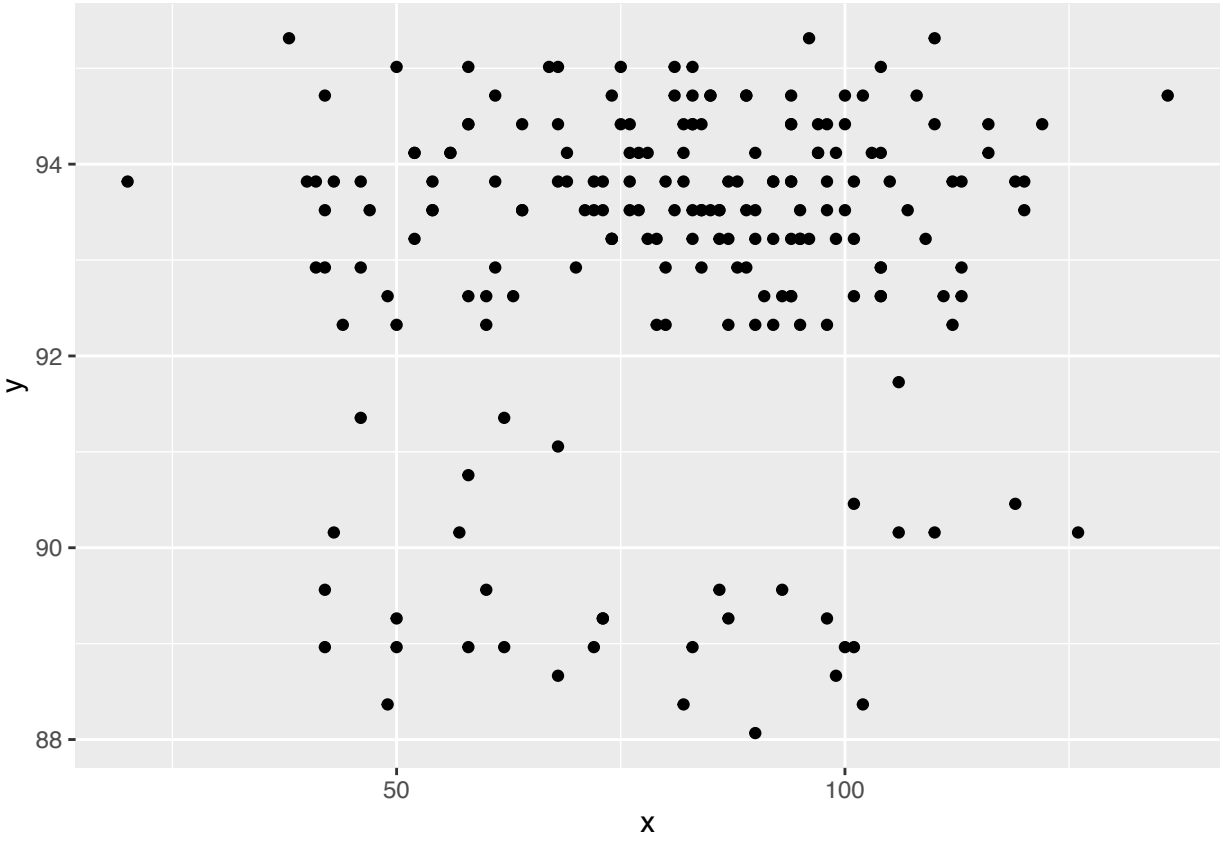
10.5d

Finally, fit a regression of child test scores on mother's age and education level for the first 200 children and use this model to predict test scores for the next 200. Graphically display comparisons of the predicted and actual scores for the final 200 children.

```
# To split the data into training and testing dataset
test_train  <-  kidiq[1:200, ]
test_test  <-  kidiq[201:400, ]
# To fit the model
mtest_train  <- lm(test_train$kid_score~test_train$mom_hs + test_train$mom_age,data=test_train)
summary(mtest_train)
```

```
##
## Call:
## lm(formula = test_train$kid_score ~ test_train$mom_hs + test_train$mom_age,
##     data = test_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.922  -8.698   3.331  11.725  49.882
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         82.9869    10.6819   7.769 4.26e-13 ***
## test_train$mom_hs    3.6596     3.4184   1.071    0.286
## test_train$mom_age   0.2988     0.4701   0.636    0.526
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.6 on 197 degrees of freedom
## Multiple R-squared:  0.01007,    Adjusted R-squared:  2.432e-05
## F-statistic: 1.002 on 2 and 197 DF,  p-value: 0.3689
```

```
# Making the final prediction
data_pre <-  data.frame(test_test$mom_hs,test_test$mom_age)
result <-  predict(mtest_train,data_pre)
data <- data.frame(x=test_test$kid_score,y=result)
ggplot(data=data, mapping = aes(x = x,y = y))+
  geom_point()
```



10.6 Regression models with interactions:

The folder Beauty contains data (use file beauty.csv) Beauty and teaching evaluations from Hamermesh and Parker (2005) on student evaluations of instructors' beauty and teaching quality for several courses at the University of Texas. The teaching evaluations were conducted at the end of the semester, and the beauty judgments were made later, by six students who had not attended the classes and were not aware of the course evaluations.

See also Felton, Mitchell, and Stinson (2003) for more on this topic.

```
# Loading the data
beau <- read.csv(file="/Users/mac/Desktop/BU Mssp/MA678/ROS-Examples-master/Beauty/data/beauty.csv", hea
head(beau)
```

```
##   eval     beauty female age minority nonenglish lower course_id
## 1  4.3  0.2015666      1  36        1          0     0         3
## 2  4.5 -0.8260813      0  59        0          0     0         0
## 3  3.7 -0.6603327      0  51        0          0     0         4
## 4  4.3 -0.7663125      1  40        0          0     0         2
## 5  4.4  1.4214450      1  31        0          0     0         0
## 6  4.2  0.5002196      0  62        0          0     0         0
```

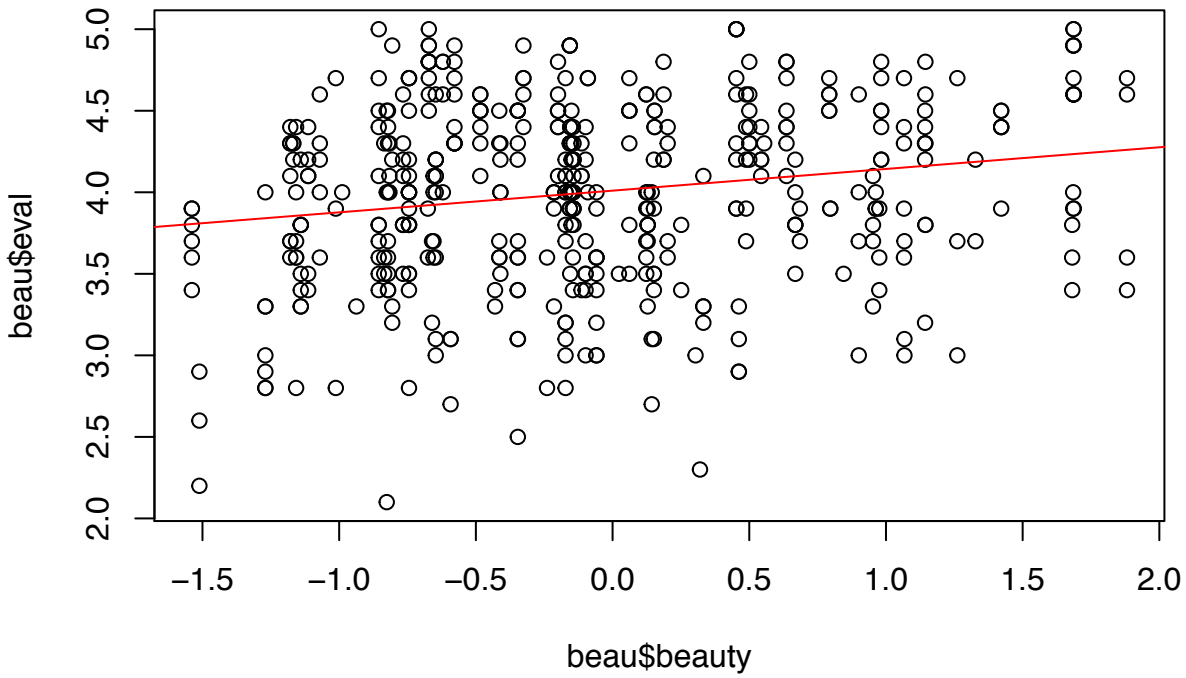
10.6a

Run a regression using beauty (the variable beauty) to predict course evaluations (eval), adjusting for various other predictors. Graph the data and fitted model, and explain the meaning of each of the coefficients along with the residual standard deviation. Plot the residuals versus fitted values.

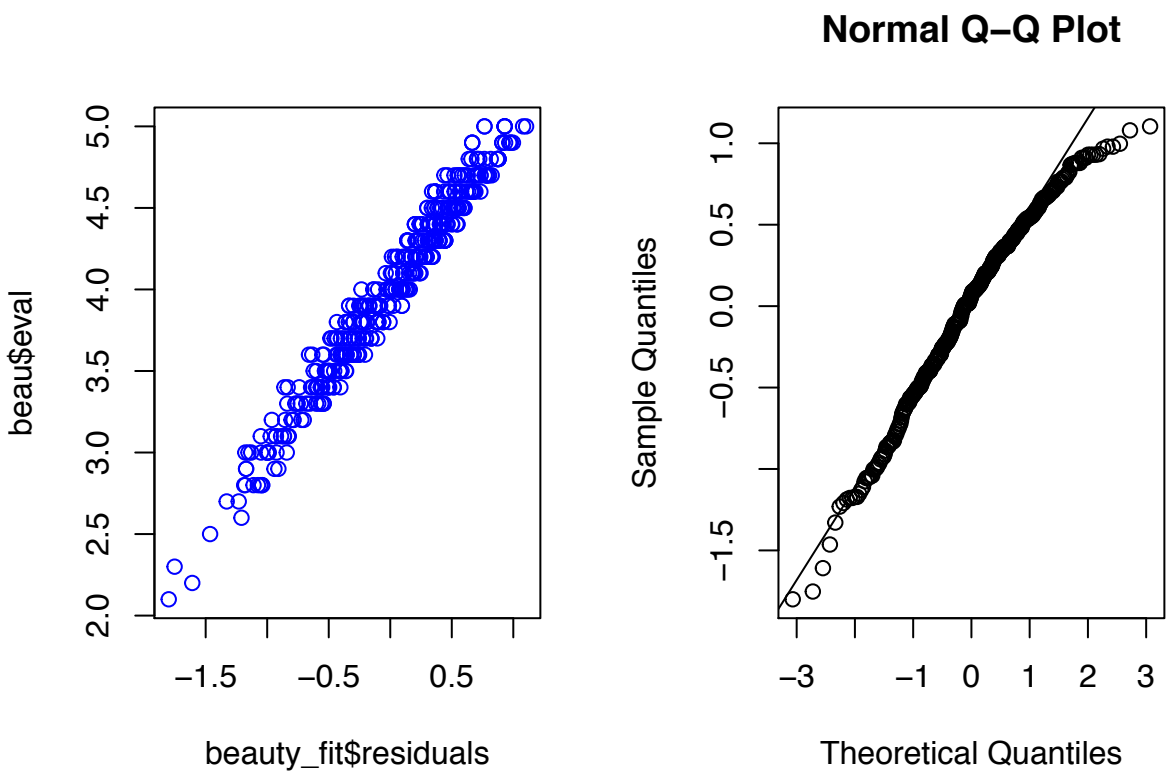
```
beauty_fit <- lm(beau$eval~ beau$beauty,data=beau)
summary(beauty_fit)
```

```
##
## Call:
## lm(formula = beau$eval ~ beau$beauty, data = beau)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.80015 -0.36304  0.07254  0.40207  1.10373
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.01002    0.02551 157.205  < 2e-16 ***
## beau$beauty  0.13300    0.03218   4.133 4.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5455 on 461 degrees of freedom
## Multiple R-squared:  0.03574,    Adjusted R-squared:  0.03364
## F-statistic: 17.08 on 1 and 461 DF,  p-value: 4.247e-05
```

```
plot(x = beau$beauty, y = beau$eval)
ahat <- coef(beauty_fit)[1]
bhat <- coef(beauty_fit)[2]
abline(ahat,bhat,col="red")
```



```r
par(mfrow=c(1,2))
plot(beauty_fit$residuals,y=beau$eval, col = "blue")
#For Q-Q plot
qqnorm(beauty_fit$residuals)
qqline(beauty_fit$residuals)
```



10.6b

Fit some other models, including beauty and also other predictors. Consider at least one model with interactions. For each model, explain the meaning of each of its estimated coefficients.

```r
# To add a variable which is the interaction of female and beauty
beauty_fit_3 = lm(beau$eval~beau$beauty +beau$female + beau$female : beau$beauty,data=beau)
summary(beauty_fit_3)
```

```
##
## Call:
## lm(formula = beau$eval ~ beau$beauty + beau$female + beau$female:beau$beauty,
##     data = beau)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83820 -0.37387  0.04551  0.39876  1.06764
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.10364    0.03359 122.158  < 2e-16 ***
## beau$beauty                 0.20027    0.04333   4.622 4.95e-06 ***
## beau$female                -0.20505    0.05103  -4.018 6.85e-05 ***
## beau$beauty:beau$female -0.11266    0.06398  -1.761   0.0789 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5361 on 459 degrees of freedom
## Multiple R-squared:  0.07256,    Adjusted R-squared:  0.0665
## F-statistic: 11.97 on 3 and 459 DF,  p-value: 1.471e-07
```

```
# Finally, this is to explain the meaning of each of its estimated  coefficients:
# The beauty coefficient refers to the average difference in evaluation scores of male teachers by abou
# The coefficient of the female is that when comparing two teachers with the same cents, the coefficien
# The coefficient of interaction means that, compared with male teachers, the average difference in eva
```

10.7 Predictive simulation for linear regression:

Take one of the models from the previous exercise.

10.7a

Instructor A is a 50-year-old woman who is a native English speaker and has a beauty score of -1. Instructor B is a 60-year-old man who is a native English speaker and has a beauty score of -0.5. Simulate 1000 random draws of the course evaluation rating of these two instructors. In your simulation, use posterior_predict to account for the uncertainty in the regression parameters as well as predictive uncertainty.

```
# First, load the previous exercise.
beauty_fit = lm(beau$eval~beau$beauty +beau$female + beau$female:beau$beauty, data=beau)
summary(beauty_fit)
```

```
##
## Call:
## lm(formula = beau$eval ~ beau$beauty + beau$female + beau$female:beau$beauty,
##     data = beau)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83820 -0.37387  0.04551  0.39876  1.06764
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                4.10364    0.03359 122.158  < 2e-16 ***
## beau$beauty                0.20027    0.04333   4.622 4.95e-06 ***
## beau$female               -0.20505    0.05103  -4.018 6.85e-05 ***
## beau$beauty:beau$female -0.11266    0.06398  -1.761   0.0789 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5361 on 459 degrees of freedom
## Multiple R-squared:  0.07256,    Adjusted R-squared:  0.0665
## F-statistic: 11.97 on 3 and 459 DF,  p-value: 1.471e-07
```

```
# Then, initiate the data
data <- data.frame(data_beauty <- c(-1,-0.5),
                   data_age <- c(50,60),
                   data_female <- c(1,0),
                   data_nonenglish <- c(0,0))
b_sample = data[sample(nrow(data),1000,replace=T),]
print(paste("The size of this data is: ",nrow(b_sample),"*",ncol(b_sample)))
```

```
## [1] "The size of this data is:  1000 * 4"
```

```
head(b_sample)
```

```
##     data_beauty....c..1...0.5. data_age....c.50..60. data_female....c.1..0.
## 1                          -1.0                    50                       1
## 1.1                        -1.0                    50                       1
## 1.2                        -1.0                    50                       1
## 1.3                        -1.0                    50                       1
## 1.4                        -1.0                    50                       1
## 2                          -0.5                    60                       0
##     data_nonenglish....c.0..0.
## 1                            0
## 1.1                          0
## 1.2                          0
## 1.3                          0
## 1.4                          0
## 2                            0
```

```
# Finally, make prediction
pre  <-  predict(beauty_fit, newdata=b_sample[1:463,])
eval_sim = as.numeric(pre)
head(eval_sim)
```

```
## [1] 3.916254 3.938201 3.971396 3.831452 4.023135 4.203825
```

10.7b

Make a histogram of the difference between the course evaluations for A and B. What is the probability that A will have a higher evaluation?

10.8 How many simulation draws:

Take the model from Exercise 10.6 that predicts course evaluations from beauty and other predictors.

10.8a

Display and discuss the fitted model. Focus on the estimate and standard error for the coefficient of beauty.

10.8b

Compute the median and mad sd of the posterior simulations of the coefficient of beauty, and check that these are the same as the output from printing the fit.

10.8c

Fit again, this time setting iter = 1000 in your stan_glm call. Do this a few times in order to get a sense of the simulation variability.

10.8d

Repeat the previous step, setting iter = 100 and then iter = 10.

10.8e

How many simulations were needed to give a good approximation to the mean and standard error for the coefficient of beauty?