myName<- “Priam Dinesh Vyas”

yes

title: “HW 1 Solutions” date: “9/7/2020” output: pdf_document —

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.

```
library("rstanarm")
set.seed(1234)
x = runif(100,0,50)
error = rnorm(100)
y1 = 5 + (7*x) + error
fake <- data.frame(x,y1)
dim(fake)
```

```
## [1] 100   2
```

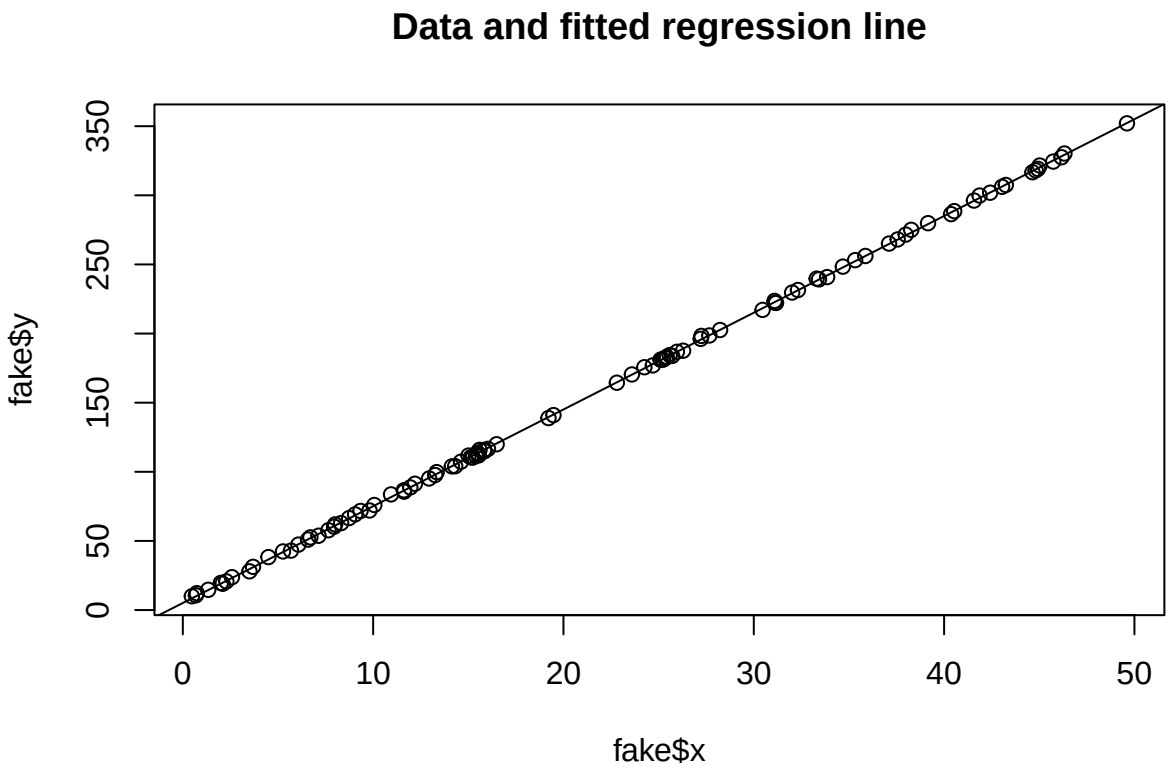
```
fit_1 <- lm(y1 ~ x, data=fake)
display(fit_1)
```

```
## lm(formula = y1 ~ x, data = fake)
##             coef.est coef.se
## (Intercept) 5.10     0.18
## x           7.00     0.01
## ---
## n = 100, k = 2
## residual sd = 0.96, R-Squared = 1.00
```

7.2b

Graph a scatterplot of the data and the regression line.

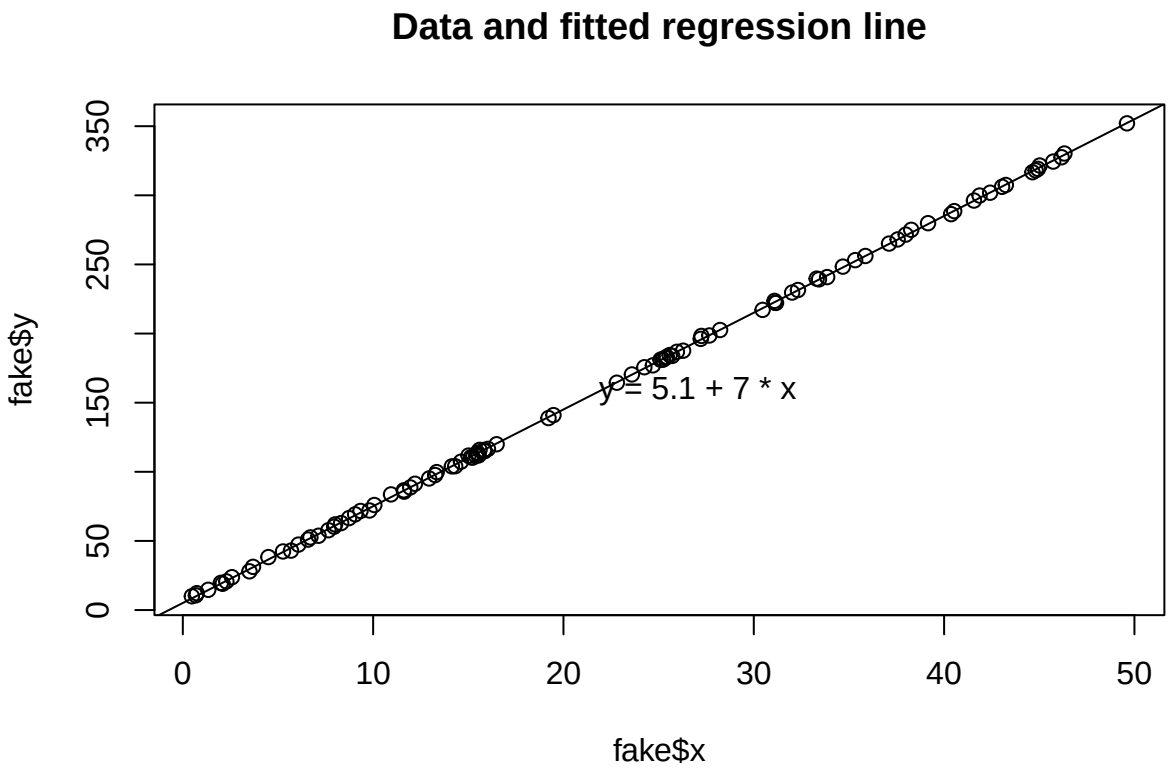
```
plot(fake$x, fake$y, main="Data and fitted regression line")
a_hat <- coef(fit_1)[1]
b_hat <- coef(fit_1)[2]
abline(a_hat, b_hat)
```



7.2c

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

```
plot(fake$x, fake$y, main="Data and fitted regression line")
a_hat <- coef(fit_1)[1]
b_hat <- coef(fit_1)[2]
abline(a_hat, b_hat)
x_bar <- mean(fake$x)
text(x_bar, a_hat + b_hat*x_bar, paste("y =", round(a_hat,2), "+", round(b_hat, 2), "* x"), adj=0)
```



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.

```
x = runif(100,0,50)
error = rnorm(100,0,3)
y = 1 + 2*x + 3*x^2 + error
fake_2 <- data.frame(x,y)
dim(fake_2)
```

```
## [1] 100   2
```

```
fit_2 <- stan_glm(y ~ x, data=fake_2)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000115 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.15 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: 0.445728 seconds (Warm-up)
## Chain 1:                0.09357 seconds (Sampling)
## Chain 1:                0.539298 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 0.694127 seconds (Warm-up)
## Chain 2:                0.07861 seconds (Sampling)
## Chain 2:                0.772737 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.18 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: 1.4401 seconds (Warm-up)
## Chain 3:                0.076095 seconds (Sampling)
## Chain 3:                1.51619 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.17 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: 3.29961 seconds (Warm-up)
## Chain 4:                0.075384 seconds (Sampling)
## Chain 4:                3.375 seconds (Total)
## Chain 4:
```

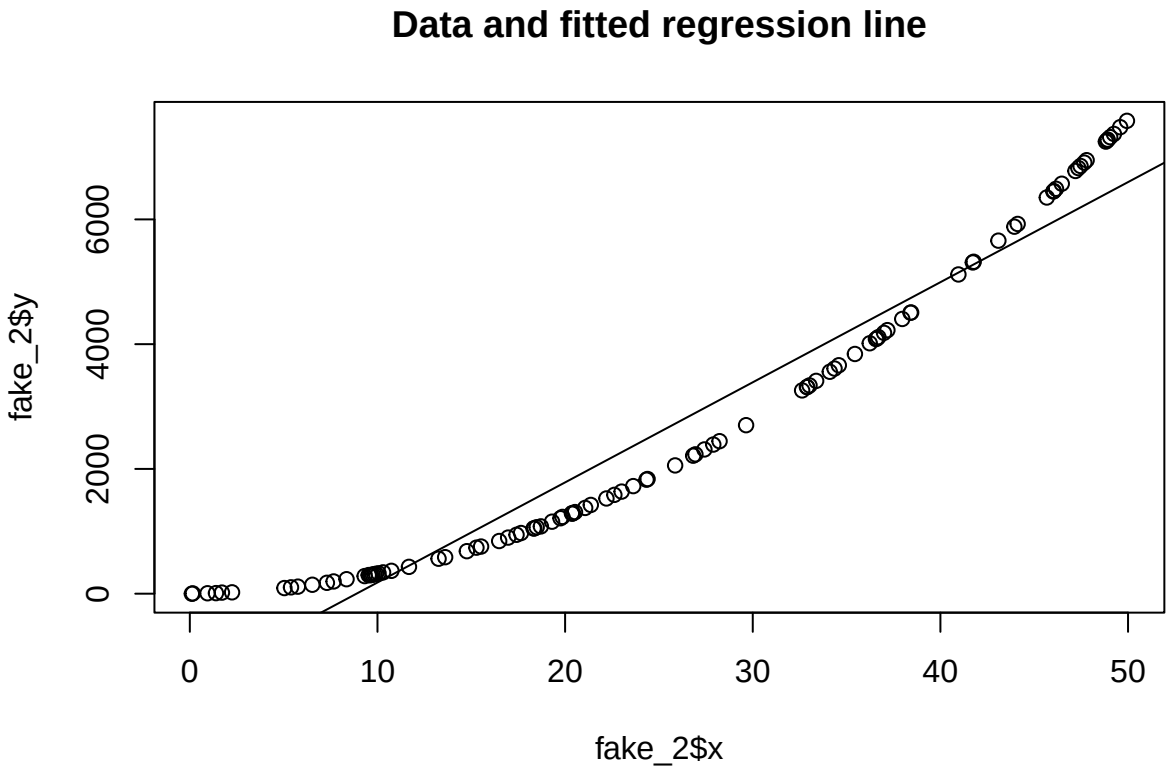
```
fit_2
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x
##  observations: 100
##  predictors:   2
## ------
##             Median  MAD_SD
## (Intercept) -1425.3   121.2
## x             160.5     4.3
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 597.3   42.2
##
## ------
## * 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_2$x, fake_2$y, main="Data and fitted regression line")
a_hat <- coef(fit_2)[1]
b_hat <- coef(fit_2)[2]
abline(a_hat, b_hat)
```




```
#The best fit line in the is the closest straight line that can be made such that the errors are lowest
```

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("/Users/priamvyas/Desktop/MSSP/678 Applied Statistics Modeling/HW Data/hibbs.dat.tx
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
```

```
hibbs <- as.data.frame(hibbs)

hibbs["binary_pred"] <- ifelse(hibbs["growth"]>2.0, 1.0, 0.0)
head(hibbs)
```

```
##   year growth  vote inc_party_candidate other_candidate growth
## 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
```

7.6a

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.

```
good <- hibbs[which(hibbs$binary_pred==1), "vote"]
bad <- hibbs[which(hibbs$binary_pred==0), "vote"]

se_good <- sd(good)/sqrt(length(good))
se_bad <- sd(bad)/sqrt(length(bad))

print(se <- sqrt(se_good^2 + se_bad^2))
```

```
## [1] 2.502052
```

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.

```
head(hibbs)
```

```
##   year growth  vote inc_party_candidate other_candidate growth
## 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
```

```
fit_7b <- lm(vote ~ binary_pred, data = hibbs)
display(fit_7b)
```

```
## lm(formula = vote ~ binary_pred, data = hibbs)
##             coef.est coef.se
## (Intercept) 49.30     1.77
## binary_pred  5.51     2.50
## ---
## n = 16, k = 2
## residual sd = 5.00, R-Squared = 0.26
```

8.8 Comparing lm and stan_glm:

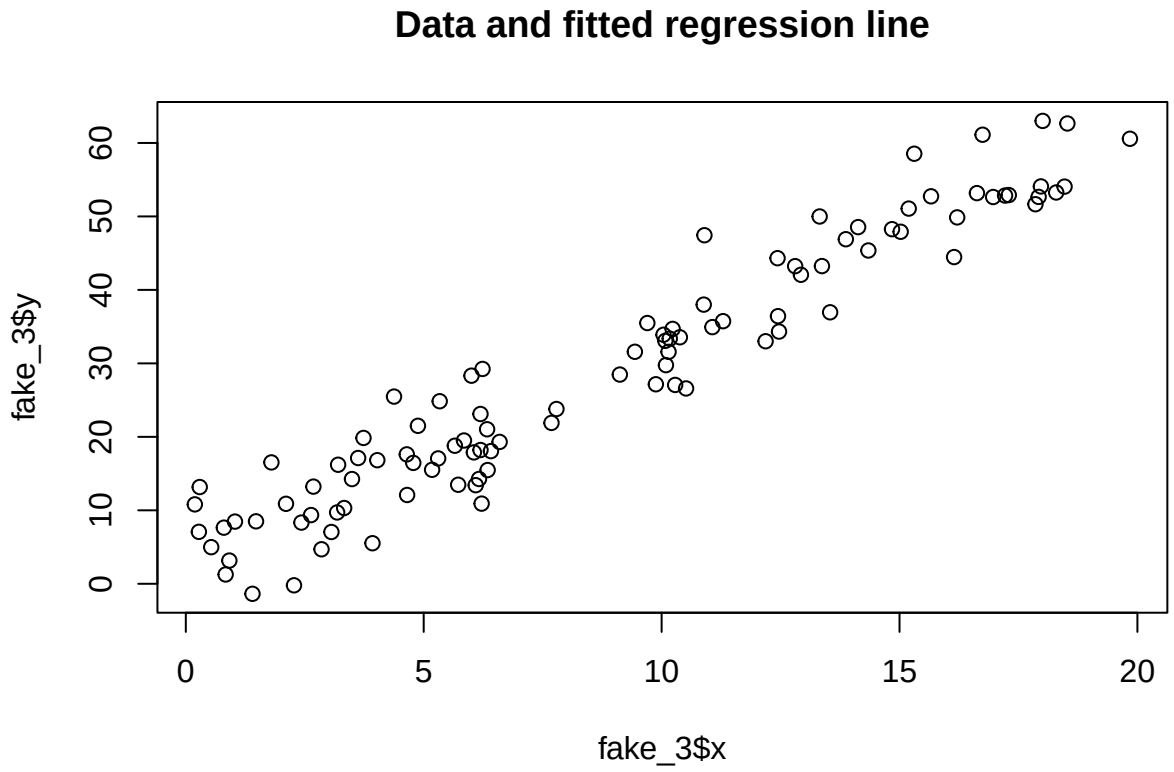
Use simulated data to compare least squares estimation to default Bayesian regression:

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.

```
set.seed(1234)
x = runif(100,0,20)
error = rnorm(100,0,5)
y = 2 + 3*x + error
fake_3 <- data.frame(x,y)

plot(fake_3$x, fake_3$y, main="Data and fitted regression line")
```




```
fit_3 <- lm(y ~ x, data=fake_3)
fit_4 <- stan_glm(y ~ x, data=fake_3)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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)
```

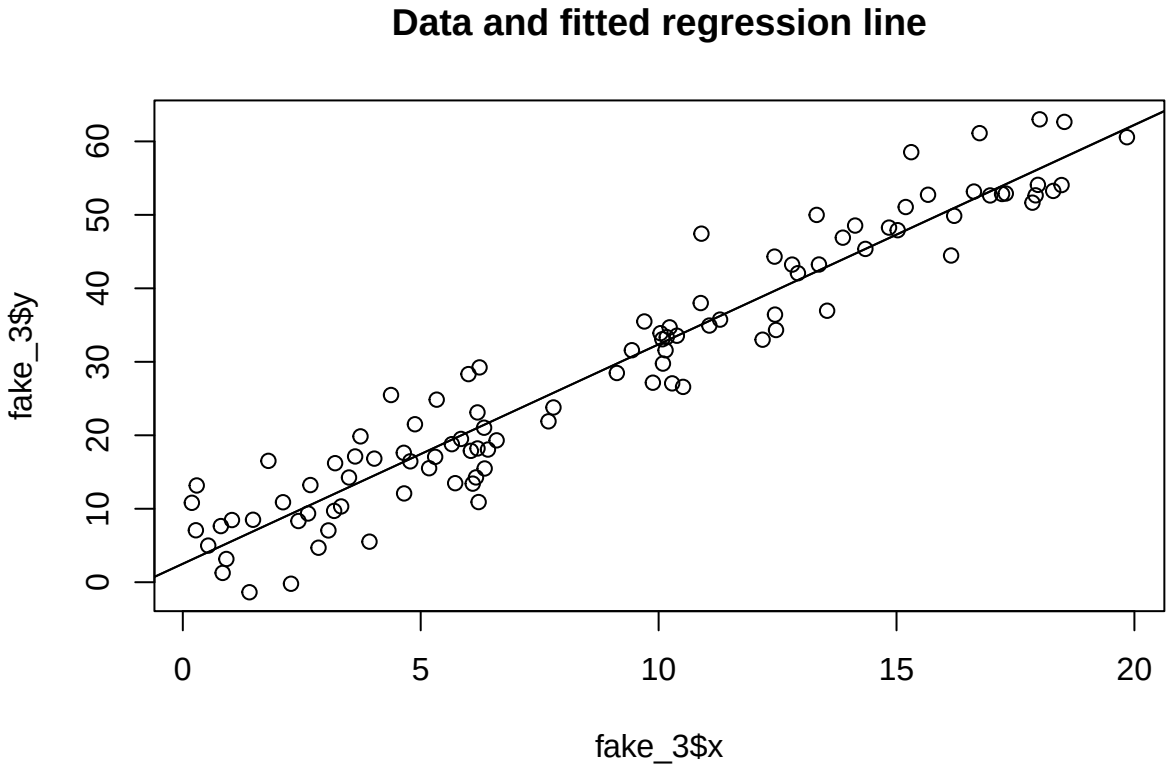
```
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.072678 seconds (Warm-up)
## Chain 1:                0.073982 seconds (Sampling)
## Chain 1:                0.14666 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.087464 seconds (Warm-up)
## Chain 2:                0.077834 seconds (Sampling)
## Chain 2:                0.165298 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.33 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: 0.124134 seconds (Warm-up)
## Chain 3:                0.091785 seconds (Sampling)
## Chain 3:                0.215919 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.9e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.29 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: 0.091591 seconds (Warm-up)
## Chain 4:                0.077394 seconds (Sampling)
## Chain 4:                0.168985 seconds (Total)
## Chain 4:
```

8.8b

Plot the simulated data and the two fitted regression lines.

```
plot(fake_3$x, fake_3$y, main="Data and fitted regression line")
a_hat_1 <- coef(fit_3)[1]
b_hat_1 <- coef(fit_3)[2]
a_hat_2 <- coef(fit_4)[1]
b_hat_2 <- coef(fit_4)[2]

abline(a_hat_1, b_hat_1)
abline(a_hat_2, b_hat_2)
```



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.

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: No variance estimation is
## Chain 1:          performed for num_warmup < 20
## Chain 1:
## Chain 1: Iteration: 1 / 5 [ 20%]  (Warmup)
## Chain 1: Iteration: 2 / 5 [ 40%]  (Warmup)
## Chain 1: Iteration: 3 / 5 [ 60%]  (Sampling)
## Chain 1: Iteration: 4 / 5 [ 80%]  (Sampling)
## Chain 1: Iteration: 5 / 5 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 7.7e-05 seconds (Warm-up)
## Chain 1:                0.000127 seconds (Sampling)
## Chain 1:                0.000204 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.1e-05 seconds
```

```
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: WARNING: No variance estimation is
## Chain 2:          performed for num_warmup < 20
## Chain 2:
## Chain 2: Iteration: 1 / 5 [ 20%]  (Warmup)
## Chain 2: Iteration: 2 / 5 [ 40%]  (Warmup)
## Chain 2: Iteration: 3 / 5 [ 60%]  (Sampling)
## Chain 2: Iteration: 4 / 5 [ 80%]  (Sampling)
## Chain 2: Iteration: 5 / 5 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 7.2e-05 seconds (Warm-up)
## Chain 2:                0.000121 seconds (Sampling)
## Chain 2:                0.000193 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: WARNING: No variance estimation is
## Chain 3:          performed for num_warmup < 20
## Chain 3:
## Chain 3: Iteration: 1 / 5 [ 20%]  (Warmup)
## Chain 3: Iteration: 2 / 5 [ 40%]  (Warmup)
## Chain 3: Iteration: 3 / 5 [ 60%]  (Sampling)
## Chain 3: Iteration: 4 / 5 [ 80%]  (Sampling)
## Chain 3: Iteration: 5 / 5 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 5.8e-05 seconds (Warm-up)
## Chain 3:                0.000132 seconds (Sampling)
## Chain 3:                0.00019 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: WARNING: No variance estimation is
## Chain 4:          performed for num_warmup < 20
## Chain 4:
## Chain 4: Iteration: 1 / 5 [ 20%]  (Warmup)
## Chain 4: Iteration: 2 / 5 [ 40%]  (Warmup)
## Chain 4: Iteration: 3 / 5 [ 60%]  (Sampling)
## Chain 4: Iteration: 4 / 5 [ 80%]  (Sampling)
## Chain 4: Iteration: 5 / 5 [100%]  (Sampling)
```

```
## Chain 4:
## Chain 4:  Elapsed Time: 9.8e-05 seconds (Warm-up)
## Chain 4:                0.000179 seconds (Sampling)
## Chain 4:                0.000277 seconds (Total)
## Chain 4:
```

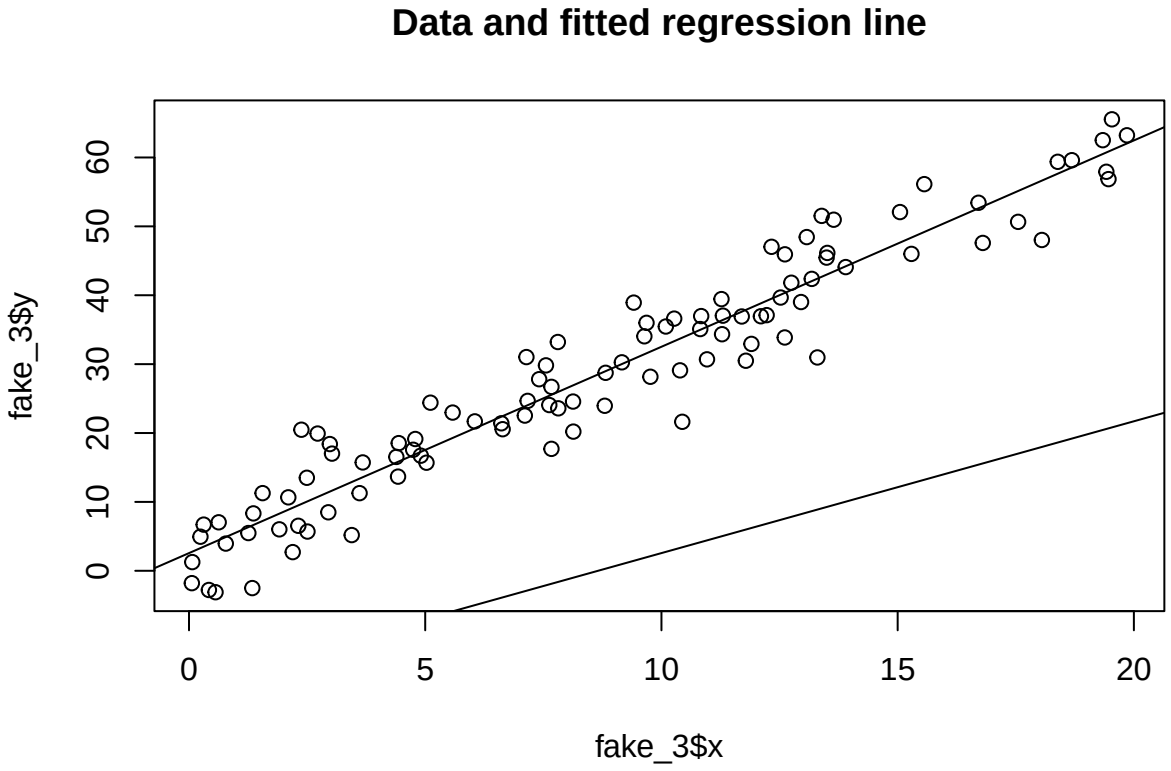
```
## Warning: There were 12 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
## Warning: Markov chains did not converge! Do not analyze results!
```



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.

```
z1 = c(1,0)
z = sample(z1, prob = c(0.5, 0.5))
x = rnorm(100,z,1)
xz = x*z


error = rnorm(100,0,3)
y = 1 + 2*x - 1*z - 2*x*z + error
fake_4 <- data.frame(x,y,z,xz)
dim(fake_4)
```

```
## [1] 100   4
```

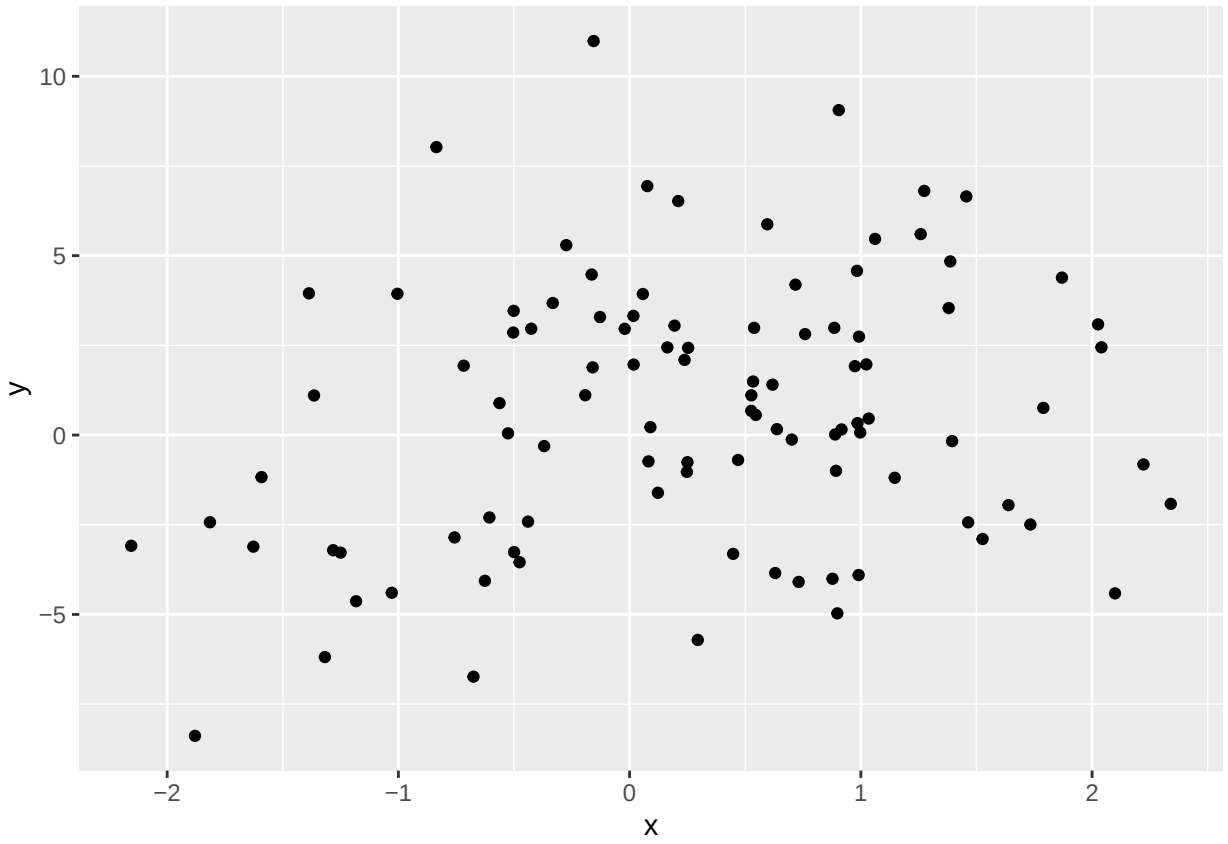
```
head(fake_4)
```

```
##             x         y z         xz
## 1  1.25891372  5.600983 1  1.2589137
## 2  0.90458938  9.058087 0  0.0000000
## 3 -0.50126271  3.462462 1 -0.5012627
## 4  0.01643887  3.322412 0  0.0000000
## 5  0.97438018  1.920452 1  0.9743802
## 6 -1.81487896 -2.431151 0  0.0000000
```

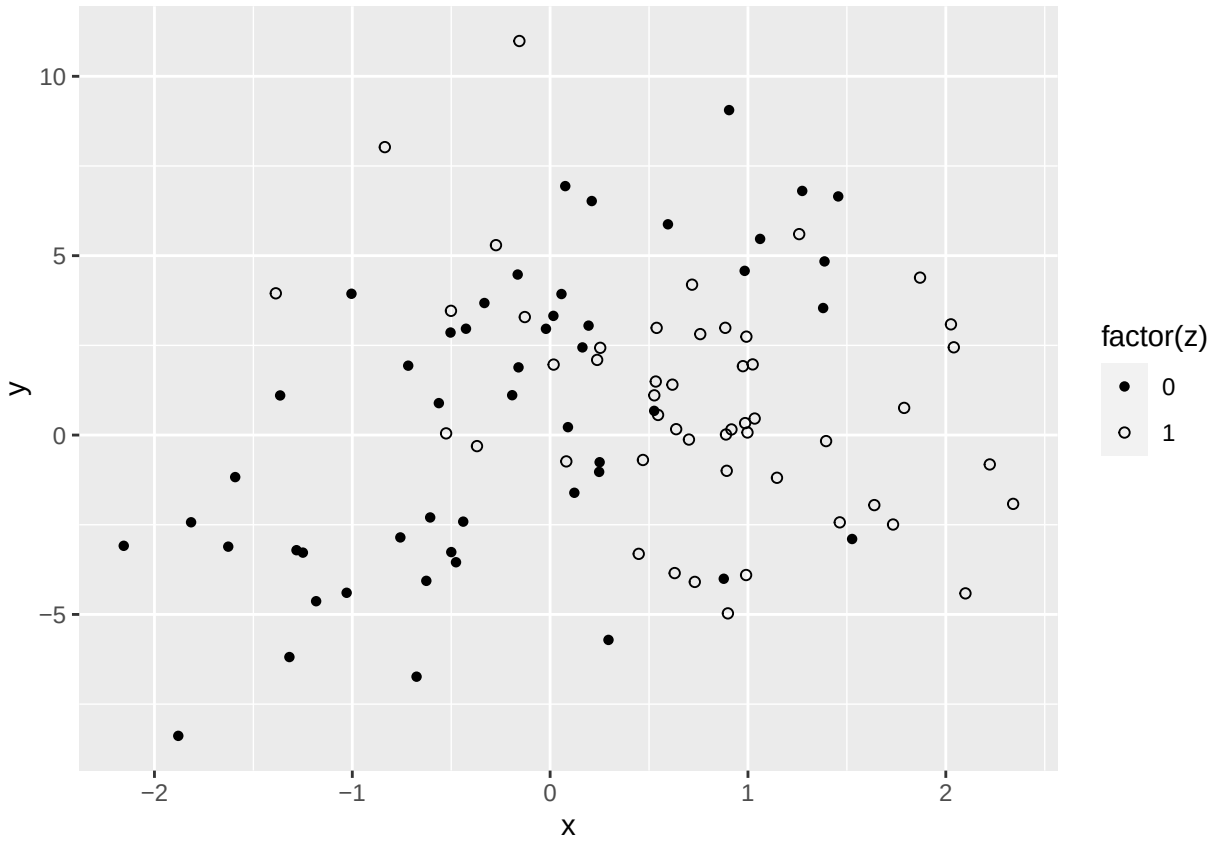
```
head(fake_4)
```

```
##             x         y z         xz
## 1  1.25891372  5.600983 1  1.2589137
## 2  0.90458938  9.058087 0  0.0000000
## 3 -0.50126271  3.462462 1 -0.5012627
## 4  0.01643887  3.322412 0  0.0000000
## 5  0.97438018  1.920452 1  0.9743802
## 6 -1.81487896 -2.431151 0  0.0000000
```

```
p <- ggplot(fake_4, aes(x, y))
p + geom_point()
```




```
p + geom_point(aes(shape = factor(z))) + scale_shape_manual(values=c(16, 1))
```



10.1b

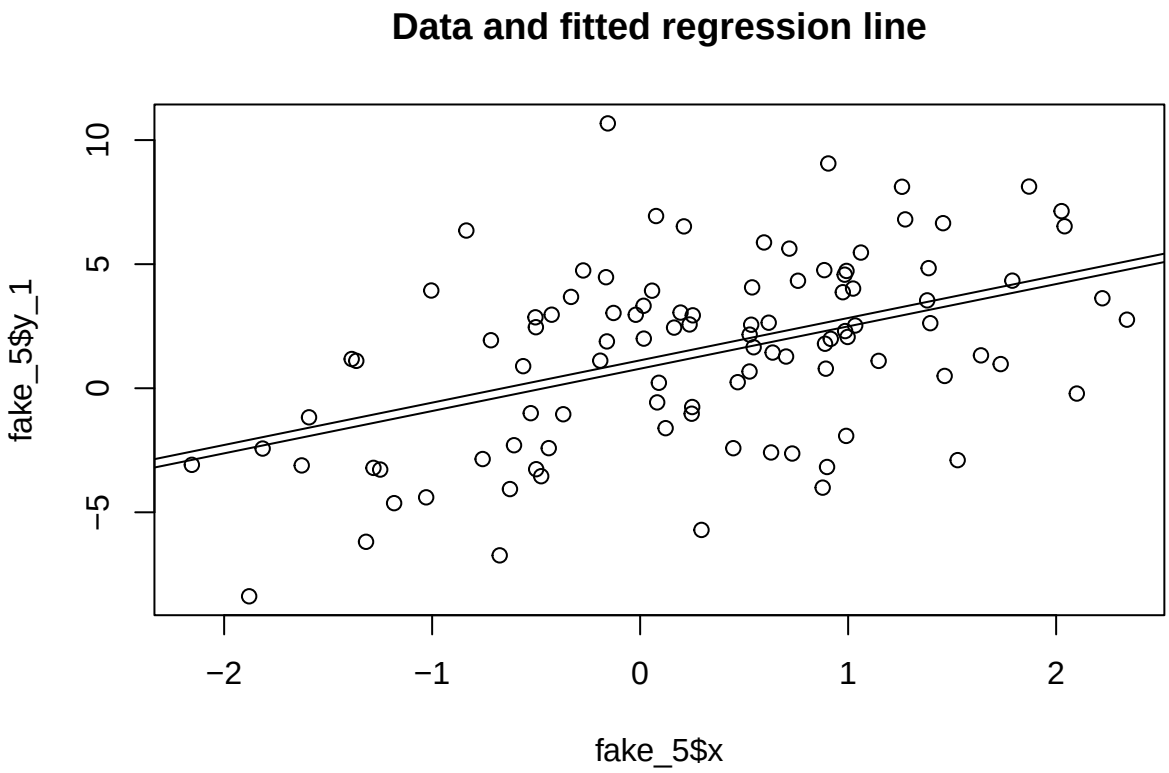
Fit a regression predicting y from x and z with no interaction. Make a graph with the data and two parallel lines showing the fitted model.

```
y_1 = 1 + 2*x - 1*z + error
fake_5 <- data.frame(x,y_1,z,xz)

plot(fake_5$x, fake_5$y_1, main="Data and fitted regression line")

fit_7 <- lm(y_1~x+z,data=fake_5)
fit_7
```

```
##
## Call:
## lm(formula = y_1 ~ x + z, data = fake_5)
##
## Coefficients:
## (Intercept)            x            z
##      0.7870       1.7056       0.3379
```

```
a_hat_1 <- coef(fit_7)[1]
b_hat_1 <- coef(fit_7)[2]

a_hat_2 <- coef(fit_7)[1] + coef(fit_7)[3]
b_hat_2 <- coef(fit_7)[2]

abline(a_hat_1, b_hat_1)
```

```
abline(a_hat_2, b_hat_2)
```



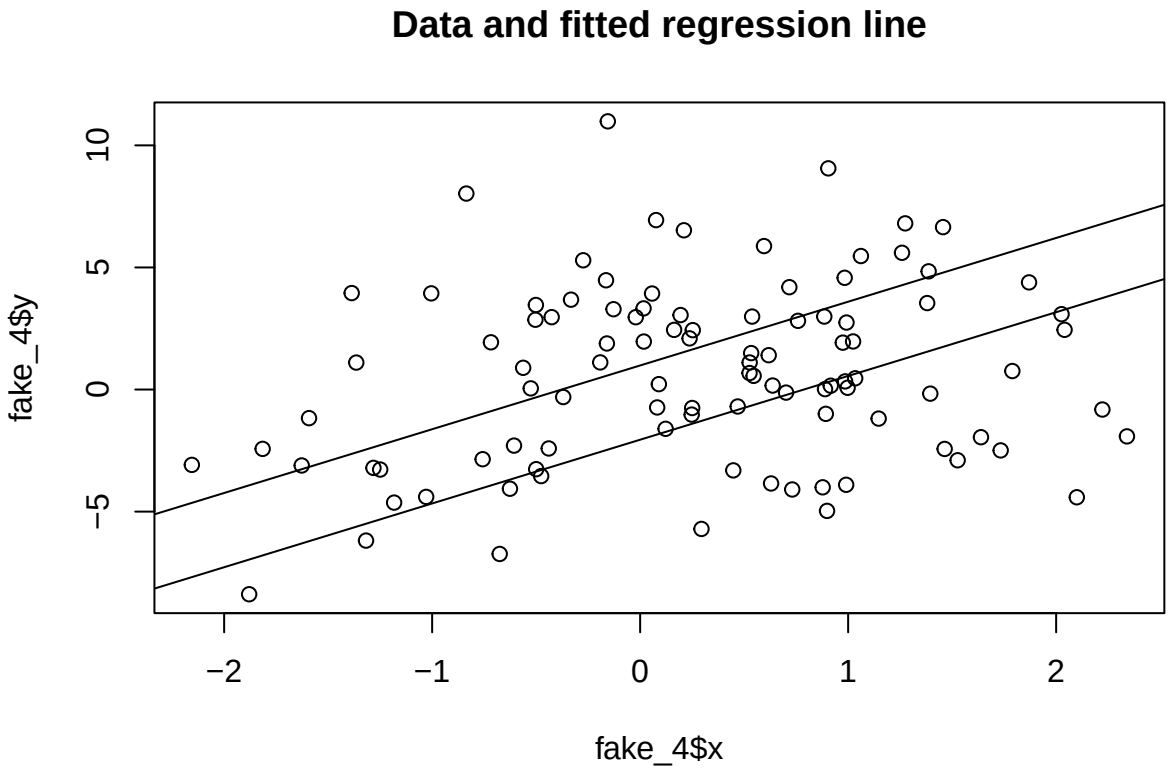
10.1c

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

```
plot(fake_4$x, fake_4$y, main="Data and fitted regression line")

fit_8 <- lm(y ~ x + z + x*z, data=fake_4)
fit_8
```

```
##
## Call:
## lm(formula = y ~ x + z + x * z, data = fake_4)
##
## Coefficients:
## (Intercept)            x            z          x:z
##      0.9856       2.6114       1.0888      -4.1302
```

```
a_hat_1 <- coef(fit_8)[1]
b_hat_1 <- coef(fit_8)[2]


a_hat_2 <- coef(fit_8)[1] + coef(fit_8)[3] + coef(fit_8)[4]
b_hat_2 <- coef(fit_8)[2]


abline(a_hat_1, b_hat_1)
abline(a_hat_2, b_hat_2)
```



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:

```
            Mediam MAD_SD
(Intercept) 1.2    0.2
x           1.6    0.4
z           2.7    0.3
x:z         0.7    0.5

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.4    0.0
```

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.

```
#Treatment Group
#y = 3.9 + 2.3x

#Control Group
#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.

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.

```
KidIQ <- read.csv("/Users/priamvyas/Desktop/MSSP/678 Applied Statistics Modeling/HW Data/child_iq.csv")
head(KidIQ)
```

```
##   ppvt educ_cat momage
## 1  120        2     21
## 2   89        1     17
## 3   78        2     19
## 4   42        1     20
## 5  115        4     26
## 6   97        1     20
```

```
dim(KidIQ)
```

```
## [1] 400   3
```

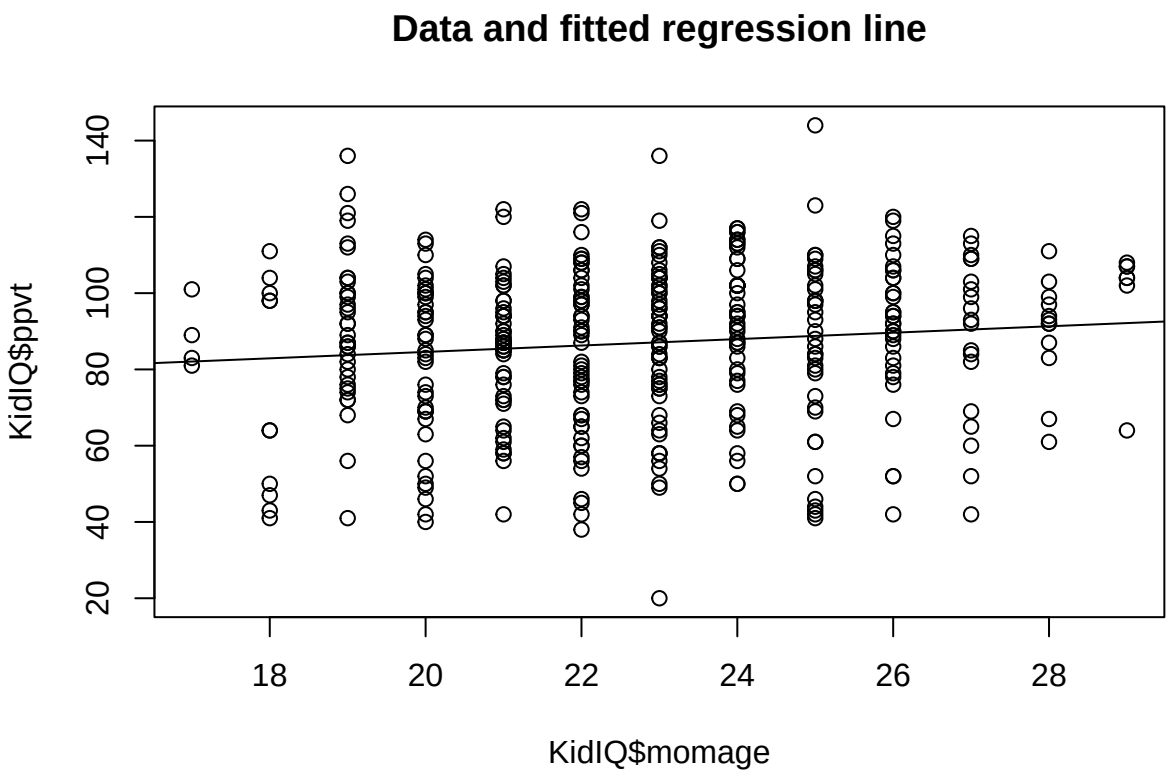
10.5a

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?

```
fit_kiq <- lm(ppvt ~ momage, data = KidIQ)
fit_kiq
```

```
##
## Call:
## lm(formula = ppvt ~ momage, data = KidIQ)
##
## Coefficients:
## (Intercept)       momage
##     67.7827       0.8403
```

```
plot(KidIQ$momage, KidIQ$ppvt, main="Data and fitted regression line")

a_hat_2 <- coef(fit_kiq)[1]
b_hat_2 <- coef(fit_kiq)[2]
abline(a_hat_2, b_hat_2)
```

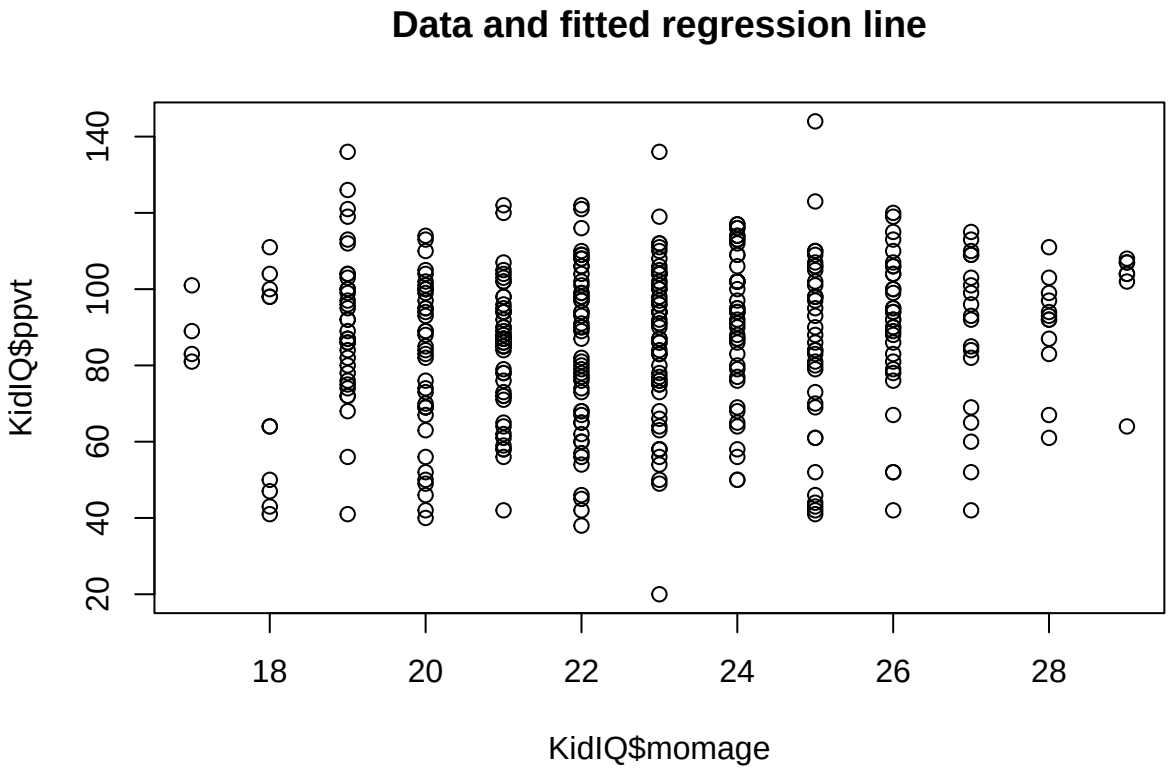



```
#The line is nearly flat, which means there really isn't any specific age that the mother should give b
```

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?

```
plot(KidIQ$momage, KidIQ$ppvt, main="Data and fitted regression line")
```




```
fit_kiq_2 <- lm(ppvt ~ momage+educ_cat, data = KidIQ)
fit_kiq_2
```

```
##
## Call:
## lm(formula = ppvt ~ momage + educ_cat, data = KidIQ)
##
## Coefficients:
## (Intercept)       momage     educ_cat
##     69.1554       0.3433       4.7114
```

```
dim(KidIQ)
```

```
## [1] 400   3
```

```
#The conclusion about not have any specfic age to birth children still holds as even after adding the m
```

10.5c

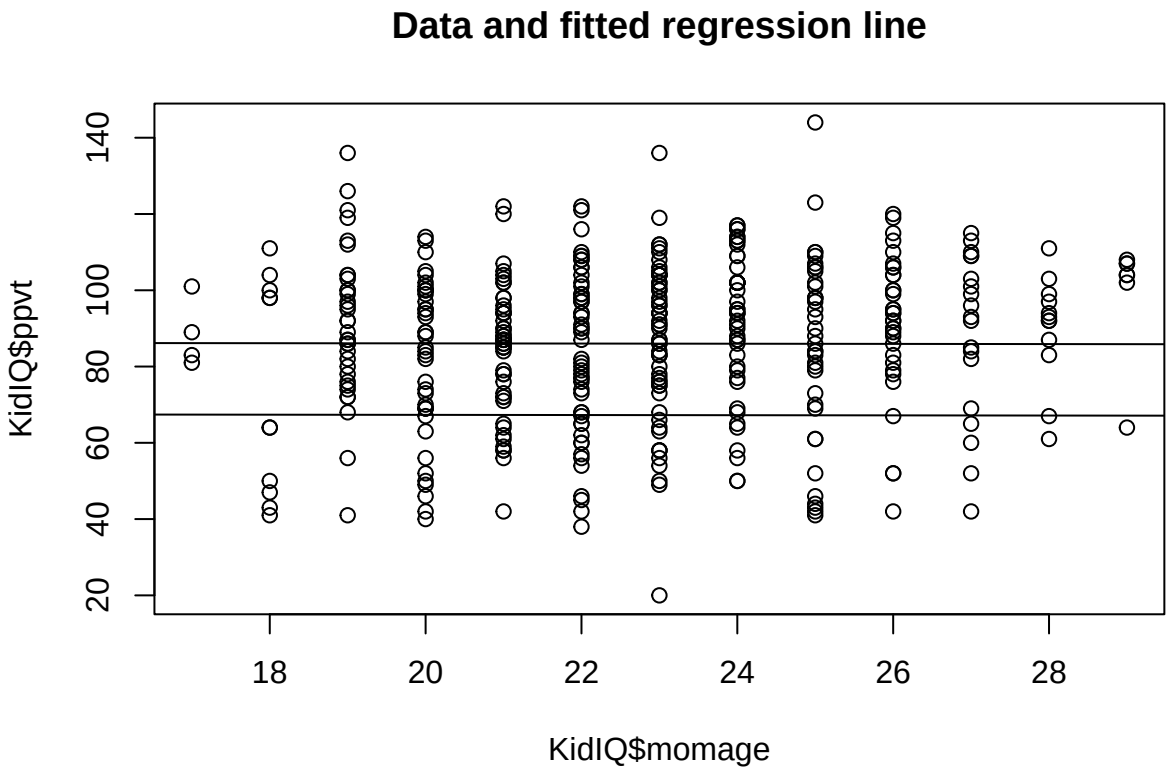
Now create an indicator variable reflecting whether the mother has completed high school or not. Consider interactions between high school completion and mother's age. Also create a plot that shows the separate regression lines for each high school completion status group.

```
KidIQ["mom_hs"] <- ifelse(KidIQ["educ_cat"]>=2.0, 1.0, 0.0)

plot(KidIQ$momage, KidIQ$ppvt, main="Data and fitted regression line")


KidIQ$mom_hs <- sample(c(1,0), size = 400, replace = TRUE)

fit_kiq_3 <- lm(ppvt ~ momage + mom_hs + mom_hs*momage, data = KidIQ)
```

```
fit_kiq_3
```

```
##
## Call:
## lm(formula = ppvt ~ momage + mom_hs + mom_hs * momage, data = KidIQ)
##
## Coefficients:
##   (Intercept)         momage         mom_hs  momage:mom_hs
##      86.52638       -0.02259      -40.78573        1.88710
```

```
#mom_hs = 0
a_hat_1 <- coef(fit_kiq_3)[1]
b_hat_1 <- coef(fit_kiq_3)[2]


#mom_hs = 1
b_hat_2 <- coef(fit_kiq_3)[1] + coef(fit_kiq_3)[3] + + coef(fit_kiq_3)[4]
b_hat_2 <- coef(fit_kiq_3)[2]

abline(a_hat_1, b_hat_1)
abline(a_hat_2, b_hat_2)
```



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.

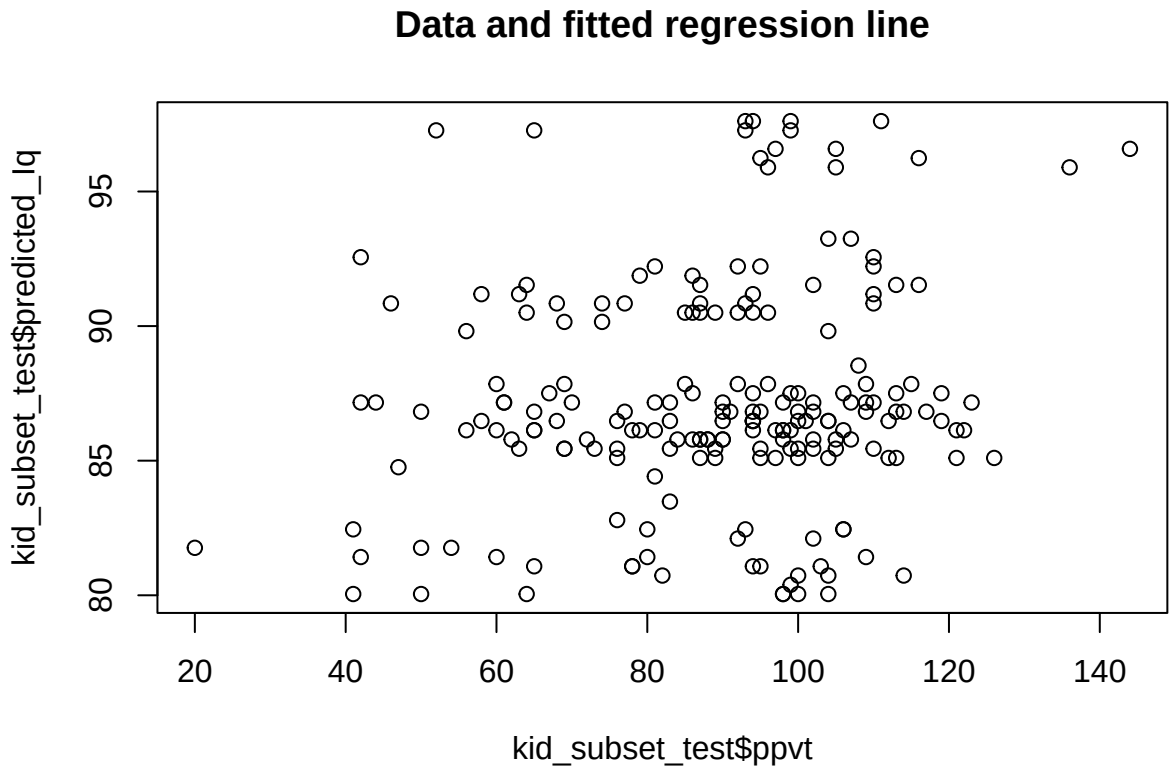
```
library(tidyr)
```

```
##
```

```
## Attaching package: 'tidyr'
```

```
## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack
```

```
library(ggplot2)

kid_subset_train <- head(KidIQ, 200)
kid_subset_test <- tail(KidIQ, 200)

kidq_fit <- lm(ppvt ~ momage+educ_cat, data = KidIQ)

kid_subset_test$predicted_Iq <- predict(kidq_fit, newdata = kid_subset_test)


plot(kid_subset_test$ppvt, kid_subset_test$predicted_Iq, main="Data and fitted regression line")
```



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.

```
beauty <- read.csv("/Users/priamvyas/Desktop/MSSP/678 Applied Statistics Modeling/HW data/beauty.csv")
head(beauty)
```

```
##   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
```

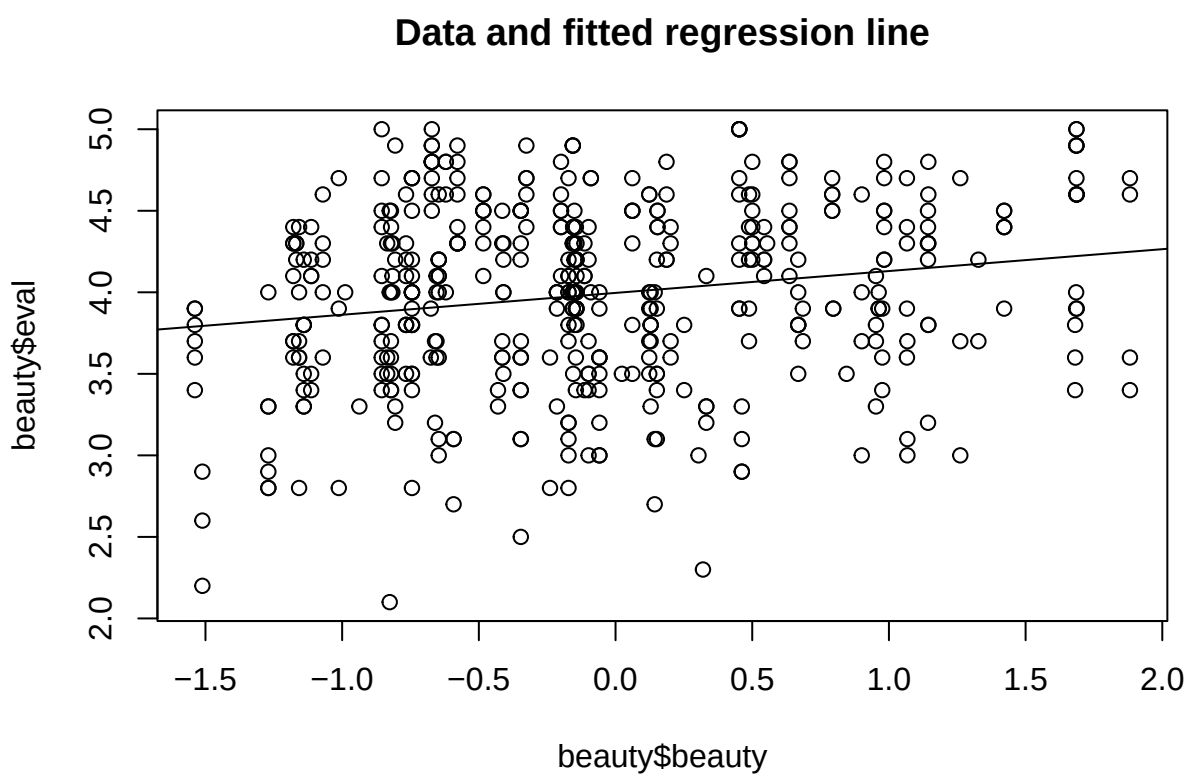
```
dim(beauty)
```

```
## [1] 463   8
```

```
fit_10 <- lm(eval ~ beauty + age, data = beauty)
fit_10
```

```
##
## Call:
## lm(formula = eval ~ beauty + age, data = beauty)
##
## Coefficients:
## (Intercept)       beauty          age
##   3.9962457    0.1340634    0.0002868
```

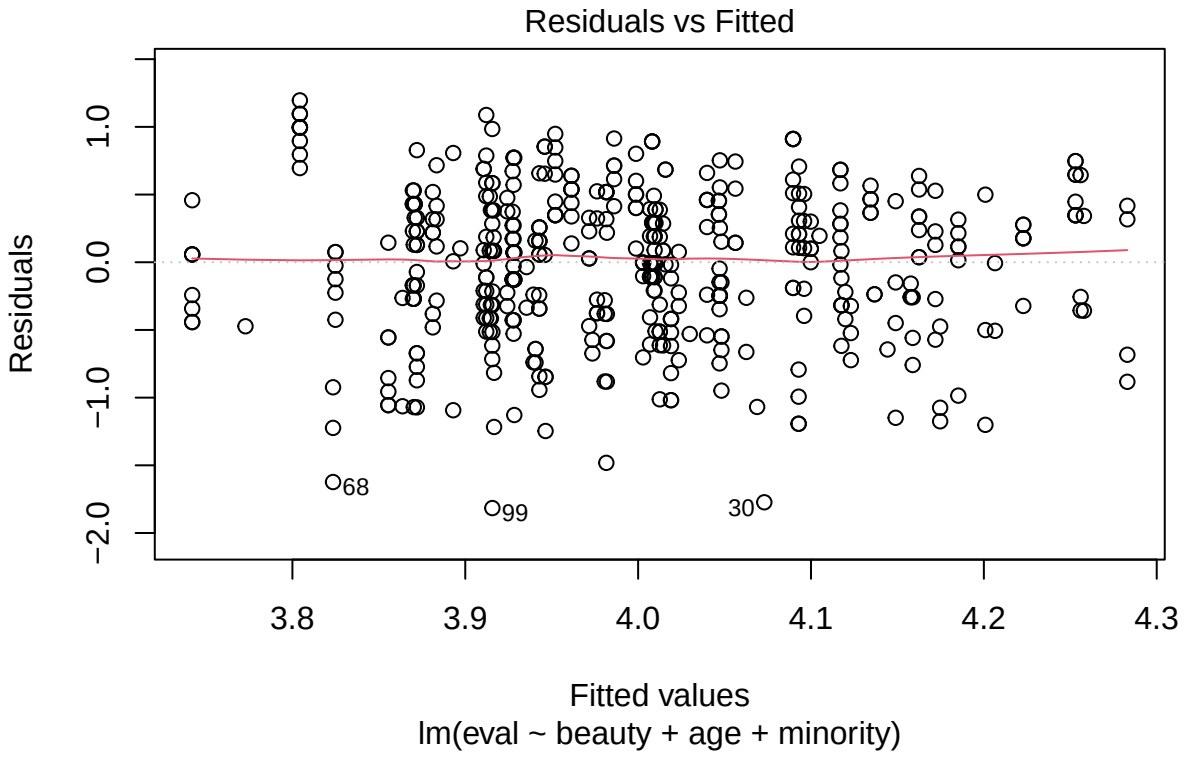
```
#The mean eval score when beauty and age are 0 is 3.99. For each increment of 1 in the beauty variable,

plot(beauty$beauty, beauty$eval, main="Data and fitted regression line")

a_hat_2 <- coef(fit_10)[1]
b_hat_2 <- coef(fit_10)[2]
abline(a_hat_2, b_hat_2)
```

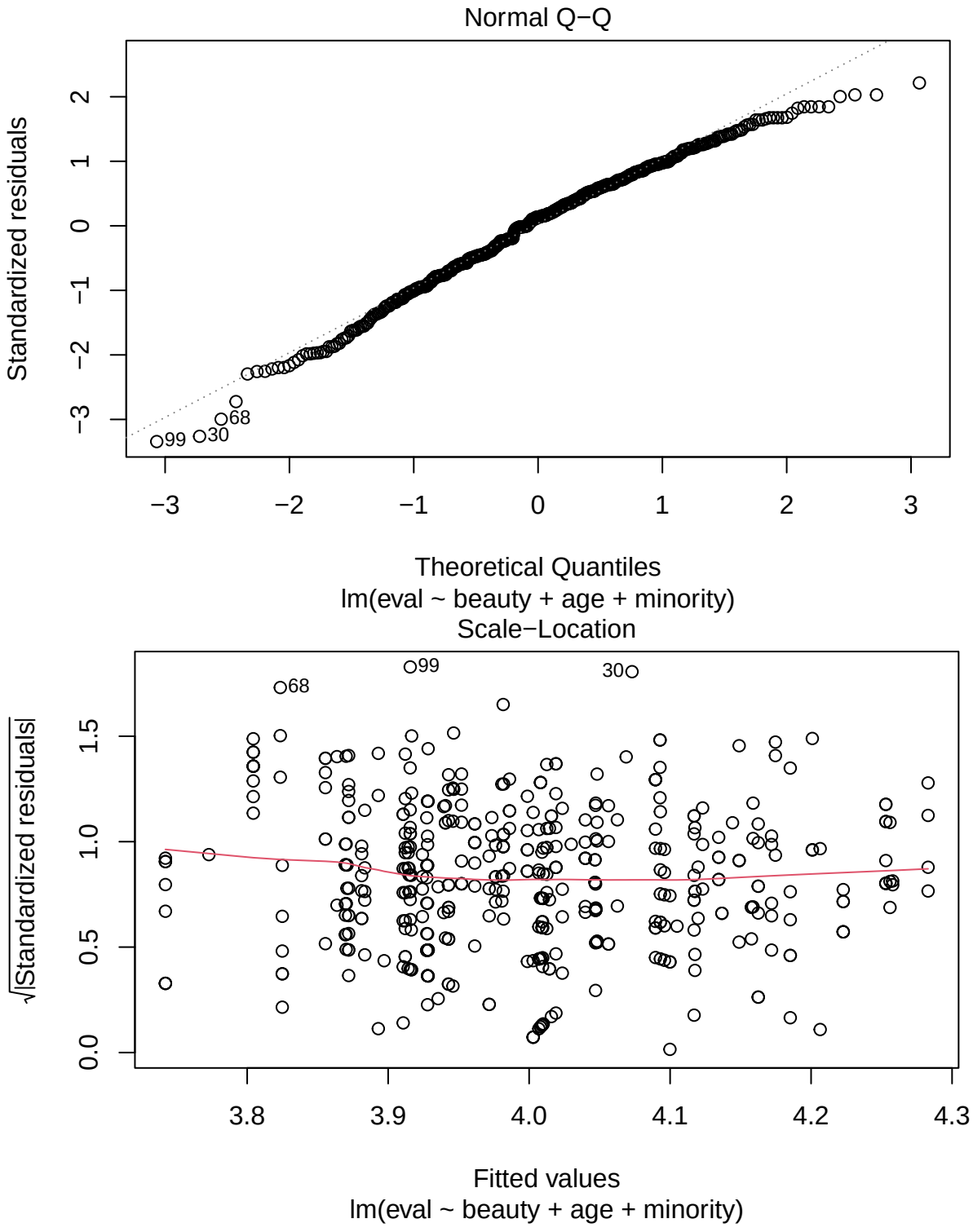


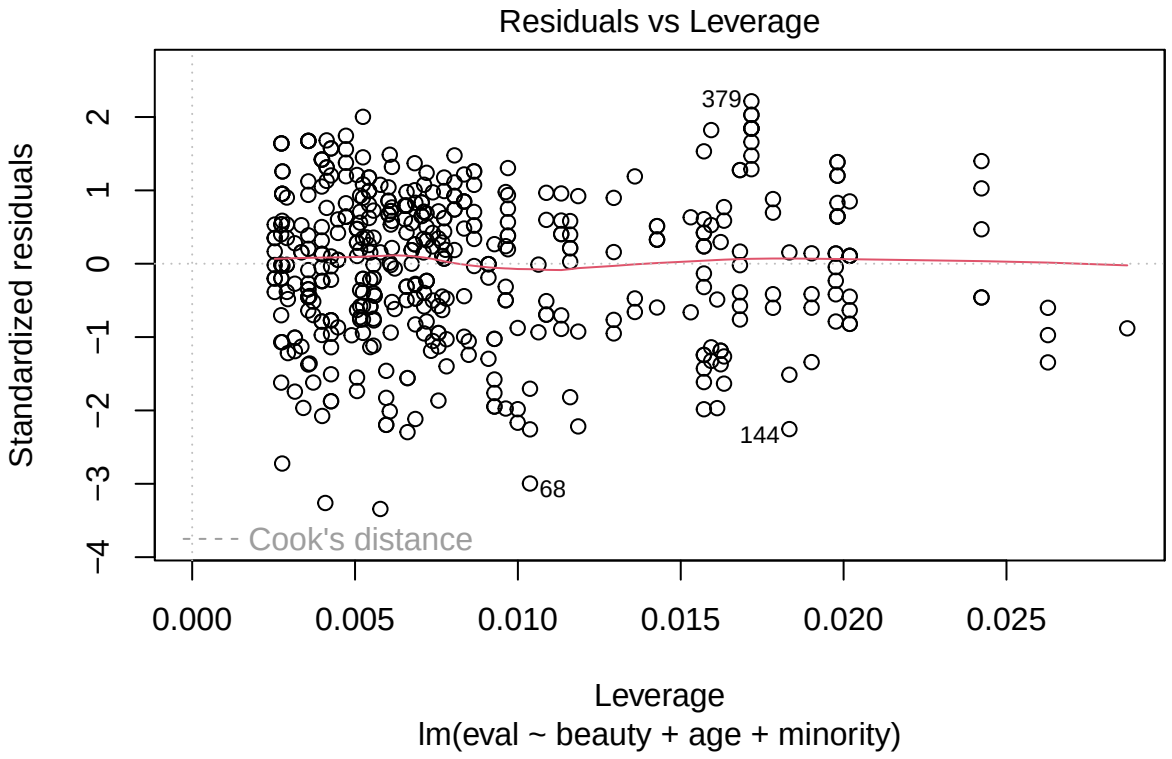
10.6a

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.

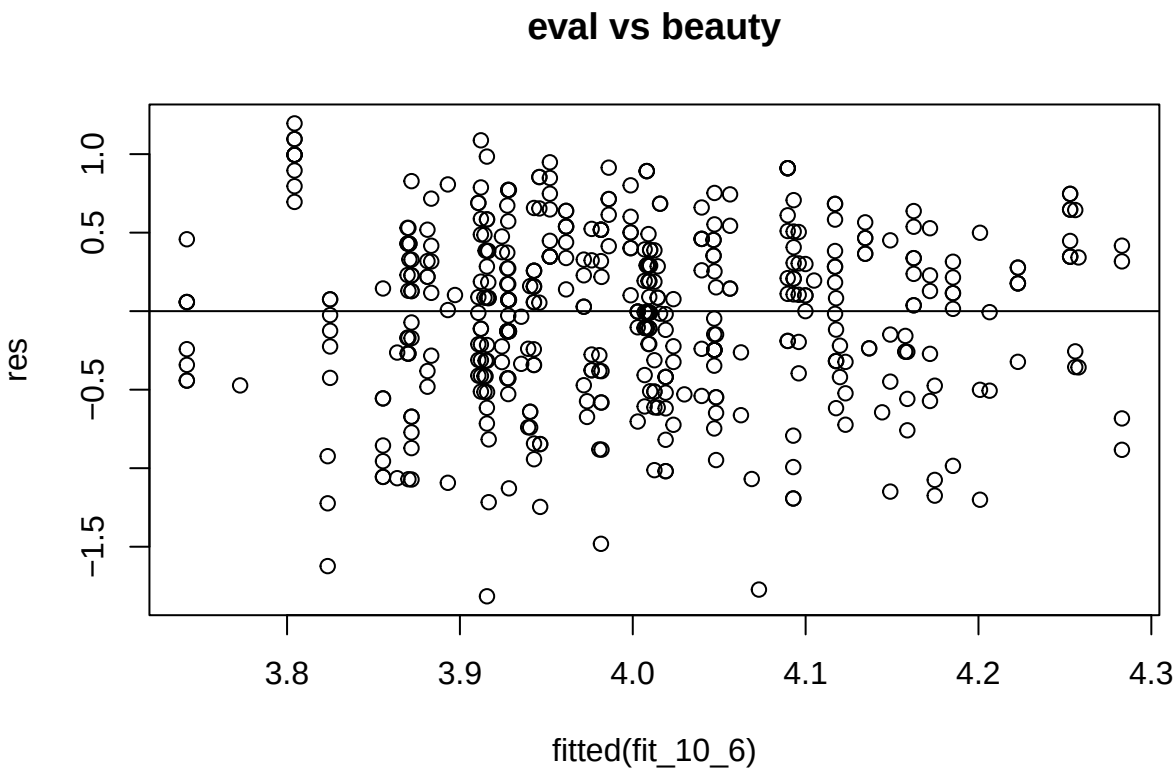
```
fit_10_6 <- lm(eval~beauty+age+minority, data=beauty)
plot(fit_10_6)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(eval ~ beauty + age + minority)
Scale−Location
√|Standardized residuals|
Fitted values
lm(eval ~ beauty + age + minority)
99
30
68




```
res <- resid(fit_10_6)
plot(fitted(fit_10_6), res, main="eval vs beauty")
abline(0,0)
```



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.

```
fit_11 <- stan_glm(eval ~ beauty + age, data = beauty)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 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: 0.078843 seconds (Warm-up)
## Chain 1:                0.116477 seconds (Sampling)
## Chain 1:                0.19532 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.9e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.070017 seconds (Warm-up)
## Chain 2:                0.10741 seconds (Sampling)
```

```
## Chain 2:                0.177427 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.24 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: 0.069545 seconds (Warm-up)
## Chain 3:                0.115606 seconds (Sampling)
## Chain 3:                0.185151 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.25 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: 0.088264 seconds (Warm-up)
## Chain 4:                0.1492 seconds (Sampling)
## Chain 4:                0.237464 seconds (Total)
## Chain 4:
```

```
fit_11
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + age
##  observations: 463
##  predictors:   3
## ------
##             Median MAD_SD
## (Intercept) 4.0    0.1
## beauty      0.1    0.0
## age         0.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
#The mean eval score when beauty and age are 0 is 3.99. For each increment of 1 in the beauty variable,

fit_12 <- stan_glm(eval ~ beauty + age + nonenglish, data = beauty)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 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: 0.068134 seconds (Warm-up)
## Chain 1:                0.118189 seconds (Sampling)
## Chain 1:                0.186323 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.076772 seconds (Warm-up)
## Chain 2:                0.123037 seconds (Sampling)
## Chain 2:                0.199809 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.9e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.069721 seconds (Warm-up)
## Chain 3:                0.115415 seconds (Sampling)
## Chain 3:                0.185136 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.32 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: 0.077802 seconds (Warm-up)
## Chain 4:                0.12353 seconds (Sampling)
## Chain 4:                0.201332 seconds (Total)
## Chain 4:
```

```
fit_12
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + age + nonenglish
##  observations: 463
##  predictors:   4
## ------
##             Median MAD_SD
## (Intercept)  4.0    0.1
## beauty       0.1    0.0
## age          0.0    0.0
## nonenglish  -0.3    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
#When the nonenglish variable is added to the regression, the mean age which is the intercept for when
```

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.

```
print(fit_12)
```

```
## stan_glm
##  family:       gaussian [identity]
```

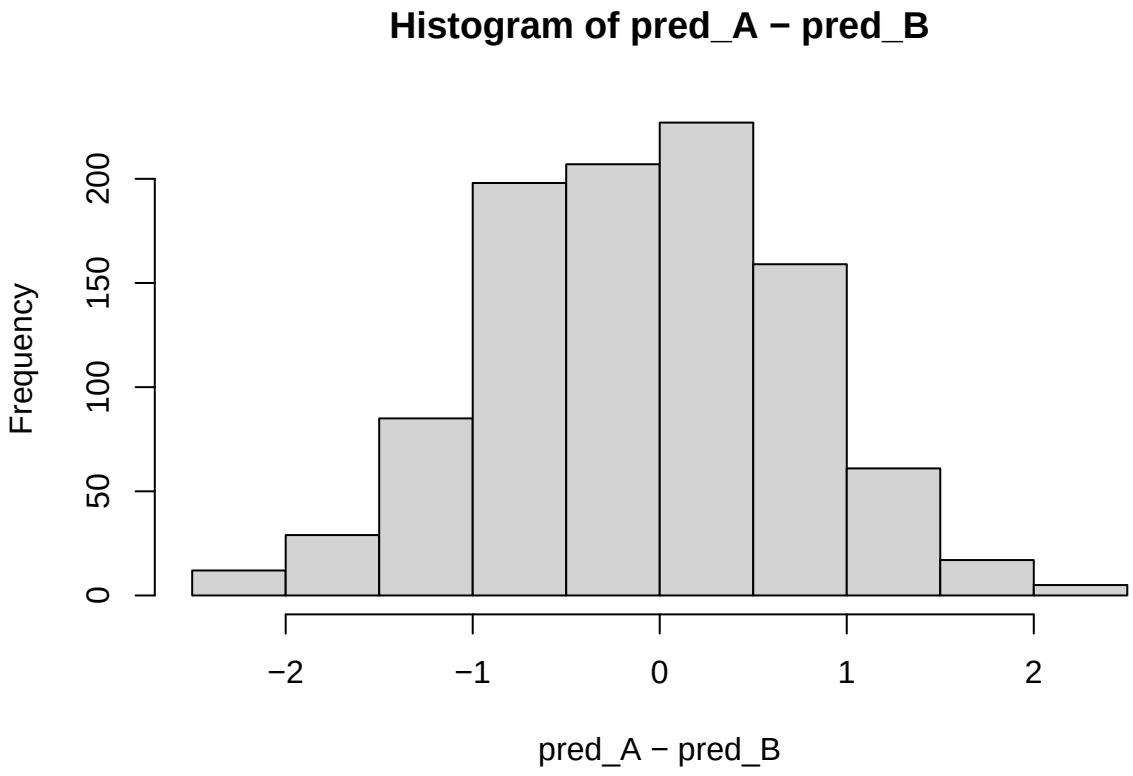
```
##  formula:      eval ~ beauty + age + nonenglish
##  observations: 463
##  predictors:   4
## ------
##             Median MAD_SD
## (Intercept)  4.0    0.1
## beauty       0.1    0.0
## age          0.0    0.0
## nonenglish  -0.3    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
Instructor_A <- data.frame(beauty=-1,age=50,female=1,nonenglish=0)
pred_A <- posterior_predict(fit_12,newdata = Instructor_A,draws=1000)

Instructor_B <- data.frame(beauty=-0.5,age=60,female=0,nonenglish=0)
pred_B <- posterior_predict(fit_12,newdata = Instructor_B,draws=1000)
```

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?

```
hist(pred_A - pred_B)
```



```
#The probability that A will have a higher evaluation is 0.
```

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.

```
print(fit_12)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + age + nonenglish
##  observations: 463
##  predictors:   4
## ------
##             Median MAD_SD
## (Intercept)  4.0    0.1
## beauty       0.1    0.0
## age          0.0    0.0
## nonenglish  -0.3    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
fit_12
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + age + nonenglish
##  observations: 463
##  predictors:   4
## ------
##             Median MAD_SD
## (Intercept)  4.0    0.1
## beauty       0.1    0.0
## age          0.0    0.0
## nonenglish  -0.3    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
#The estimate if 0.1 for beauty that mean the eval score rises by 0.1 for each 1 points rise in 1 point
```

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.

postmed(x, s, w = 0.5, prior = "laplace", a = 0.5)

package.remove("brms")

```
#library("brms")

#post <- posterior_samples(fit_12)
#med_beauty <- median(post[,2])
#se_beauty <- mad(post[,2])
#fit_post <- stan_glm(post)
#print(fit_post)
```

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.

```
#fit_post_1000 <- stan_glm(post,iter=1000)
#print(fit_post_1000)
```

10.8d

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

```
#fit_post_2100 <- stan_glm(post,iter=100)
#print(fit_post_2100)


#fit_post_10 <- stan_glm(post,iter=10)
#print(fit_post_10)
```

10.8e

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

#1000 simulations were needed to give a good approximation to the mean and se for the coefficient of beauty.