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.

```
n0 <- 100
a <- 5
b <- 7
x0 <- runif(n0, 0, 50)

e0 <- rnorm(n0, 0, 3)
y0 <- a+ b*x0 + e0

# (a) Fit a regression line to these data and display the output.
fake_a <- data.frame(x0, y0)
fit_a <- stan_glm(y0 ~ x0, data = fake_a, refresh =0 )
print(fit_a, digits=2)
```

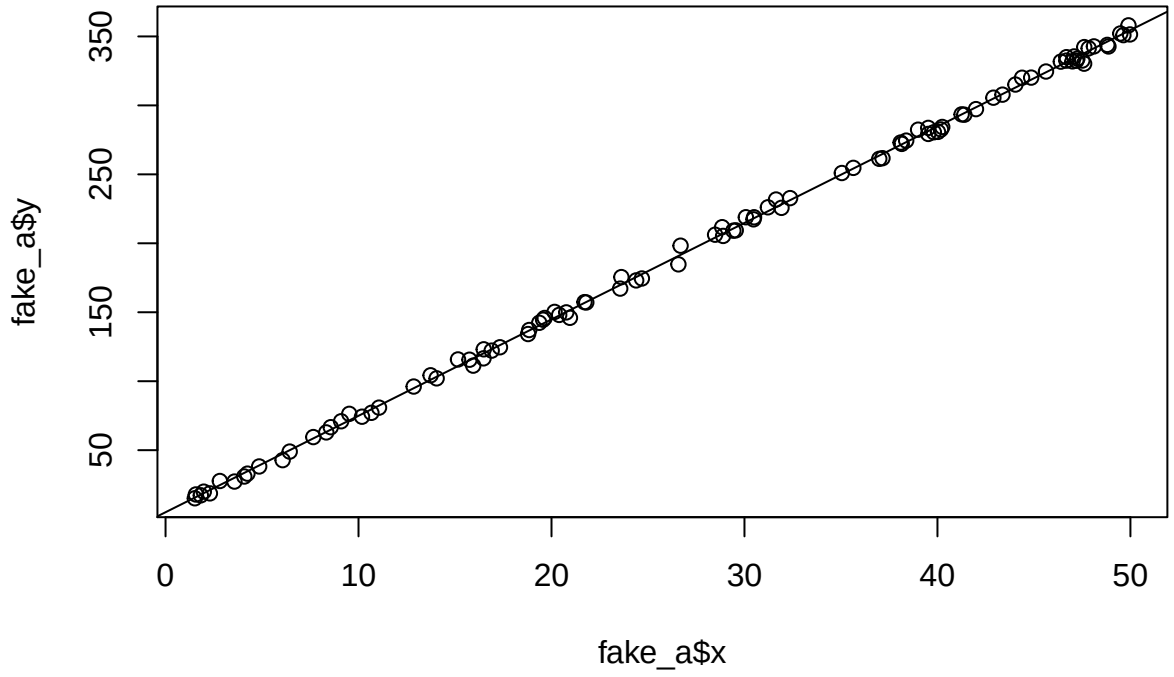
```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y0 ~ x0
##  observations: 100
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 4.98   0.59
## x0          6.99   0.02
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 2.90   0.21
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

7.2b

Graph a scatterplot of the data and the regression line.

```
# ggplot(data=fake_a)+
#   geom_point(mapping=aes(x=x, y=y))
```

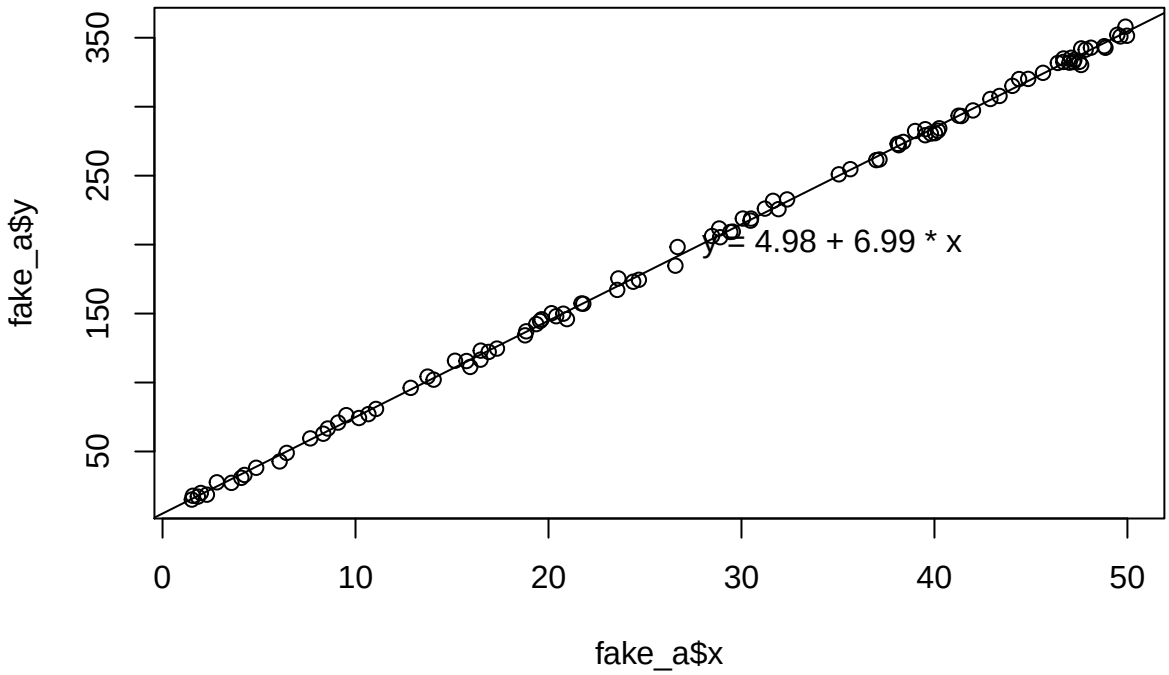
```r
plot(fake_a$x, fake_a$y)
a_hat <- coef(fit_a)[1]
b_hat <- coef(fit_a)[2]
abline(a_hat, b_hat)
```



7.2c

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

```r
plot(fake_a$x, fake_a$y)
abline(a_hat, b_hat)
x_bar <- mean(fake_a$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.

```
n1 <- 100
a_1 <- 10
b_1 <- 4
c_1 <- 1
x_1 <- runif(n1, 0, 50)
e_1 <- rnorm(n1, 0, 3)
y_1 <- a_1 + b_1*x_1 + c_1*(x_1^2)
fake_1 = data.frame(x_1, y_1)
# ggplot(data = fake_1)+
#   geom_point(mapping = aes(x=x_1, y=y_1))
fit_1 = stan_glm(y_1 ~ x_1, data = fake_1, refresh =0 )
print(fit_1, digits=2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y_1 ~ x_1
##  observations: 100
##  predictors:   2
## ------
##             Median  MAD_SD
## (Intercept) -415.07   37.18
## x_1           53.84    1.30
##
```

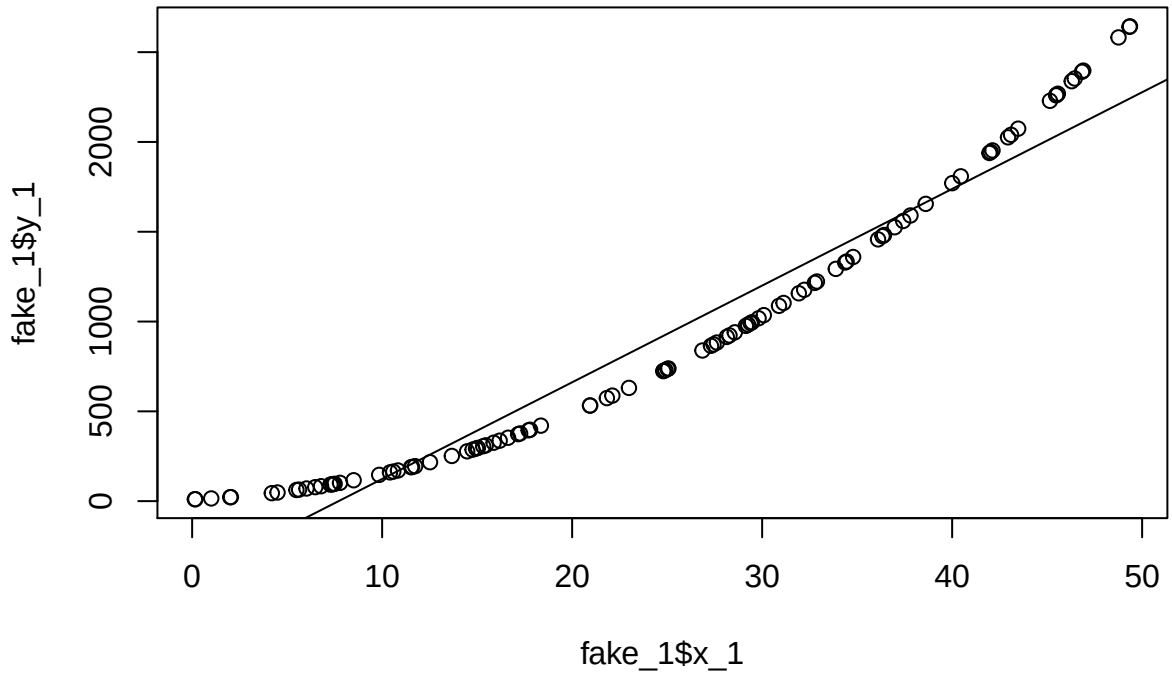
```
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 183.10  13.34
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

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?

Answer: "Best fit" means this regression model is calculated by minimizing the sum of all squared errors.

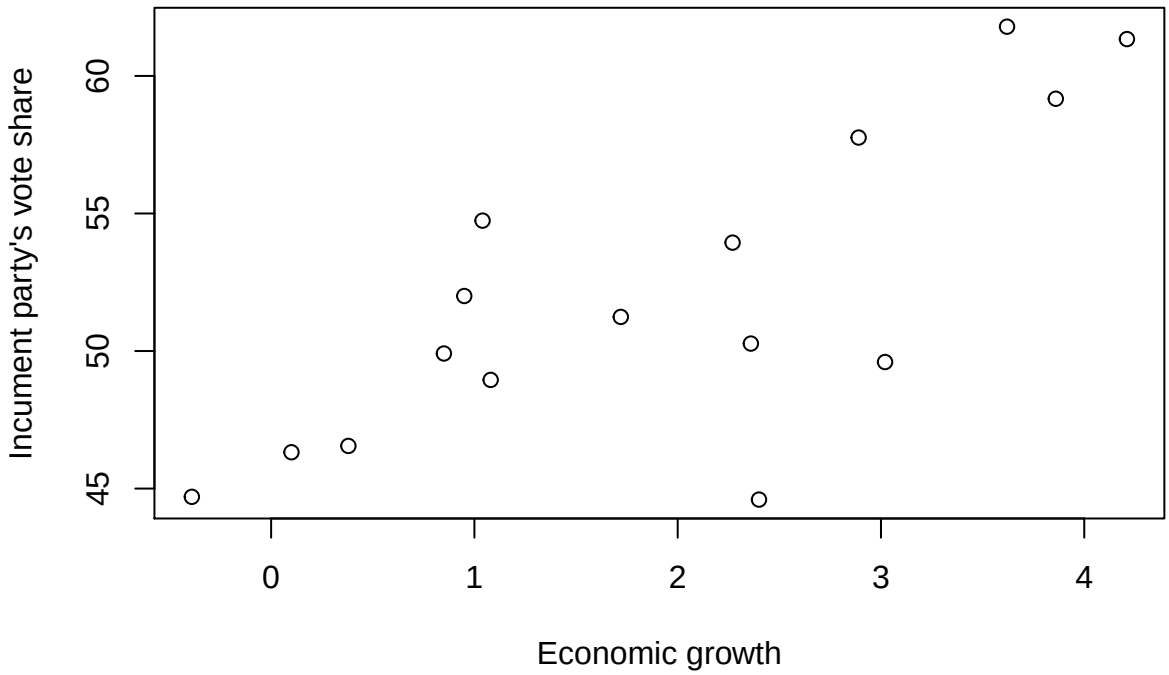
```
plot(fake_1$x_1, fake_1$y_1)
a1_hat <- coef(fit_1)[1]
b1_hat <- coef(fit_1)[2]
abline(a1_hat, b1_hat)
```



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("hibbs.dat", header = TRUE)
plot(hibbs$growth, hibbs$vote, xlab = "Economic growth",
     ylab = "Incument party's vote share")
```




```
n <- length(hibbs$growth)
growth_bi <- rep(NA, n)
vote_0 <- rep(NA, 8)
vote_1 <- rep(NA, 8)
a <- 1
b <- 1
for (i in 1:n) {
  if (hibbs$growth[i] < 2){
    growth_bi[i] <- 0
    vote_0[a] <- hibbs$vote[i]
    a <- a+1
  } else {
    growth_bi[i] <- 1
    vote_1[b] <- hibbs$vote[i]
    b <- b+1
  }
}
# print(growth_bi)
print(vote_0)
```

```
## [1] 49.91 48.95 44.70 46.55 54.74 51.24 46.32 52.00
```

```
print(vote_1)
```

```
## [1] 44.60 57.76 61.34 49.60 61.79 59.17 53.94 50.27
```

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.

```
vote_diff <- mean(vote_1) - mean(vote_0)
n_0 <- length(vote_0)
n_1 <- length(vote_1)
```

```
se_0 <- sd(vote_0)/sqrt(n_0)
se_1 <- sd(vote_1)/sqrt(n_1)
se <- sqrt(se_0^2 + se_1^2)
print(paste("difference in vote shares on average is ", vote_diff))
```

```
## [1] "difference in vote shares on average is  5.5075"
```

```
print(paste("standard error of this difference is ", se))
```

```
## [1] "standard error of this difference is  2.50205192577726"
```

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.

```
fit_hibbs <- stan_glm(vote ~ growth_bi, data = hibbs, refresh =0 )
growth_coef <- coef(fit_hibbs[2])
print(fit_hibbs, digits=3)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      vote ~ growth_bi
##  observations: 16
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 49.374  1.828
## growth_bi    5.485  2.541
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 5.123  0.922
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
print(paste("estimated coefficient of the binary predictor", ", and standard error ", " are both almost
```

```
## [1] "estimated coefficient of the binary predictor , and standard error   are both almost identical
```

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.

```
e1 <- rnorm(n, 0, 5)

x_2 <- runif(n, 0, 20)
```

```
y_2 <- 2 + 3*x_2 + e1
fake_2 = data.frame(x_2, y_2)
fit_lm = lm(y_2 ~ x_2, data = fake_2)
fit_stanlm = stan_glm(y_2 ~ x_2, data = fake_2, refresh =0 )
# print(x_2)
print(fit_lm, digits=2)
```

```
##
## Call:
## lm(formula = y_2 ~ x_2, data = fake_2)
##
## Coefficients:
## (Intercept)          x_2
##        -3.1          3.4
```

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

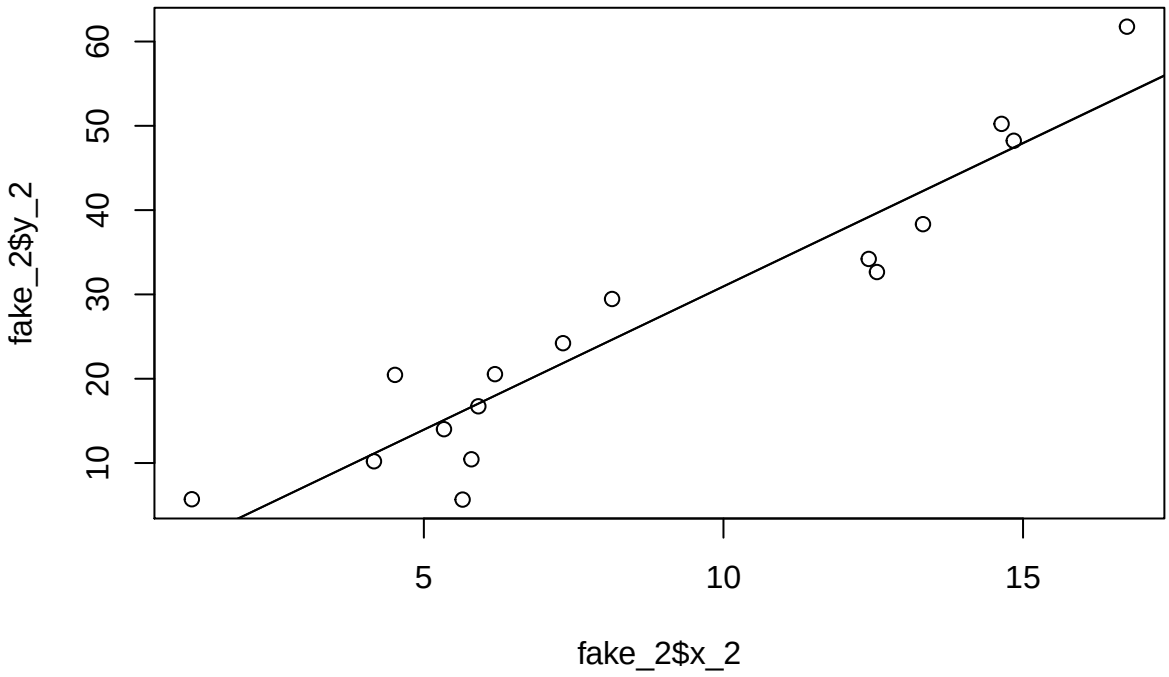
```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y_2 ~ x_2
##  observations: 16
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) -3.03   3.20
## x_2          3.40   0.32
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 5.93   1.17
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
# plot(fake_2$x_2, fake_2$y_2)
```

8.8b

Plot the simulated data and the two fitted regression lines.

```
plot(fake_2$x_2, fake_2$y_2)
a_hat_lm <- coef(fit_lm)[1]
b_hat_lm <- coef(fit_lm)[2]
a_hat_stan <- coef(fit_stanlm)[1]
b_hat_stan <- coef(fit_stanlm)[2]
abline(a = a_hat_lm, b = b_hat_lm)
abline(a = a_hat_stan, b = b_hat_stan)
```



8.8c

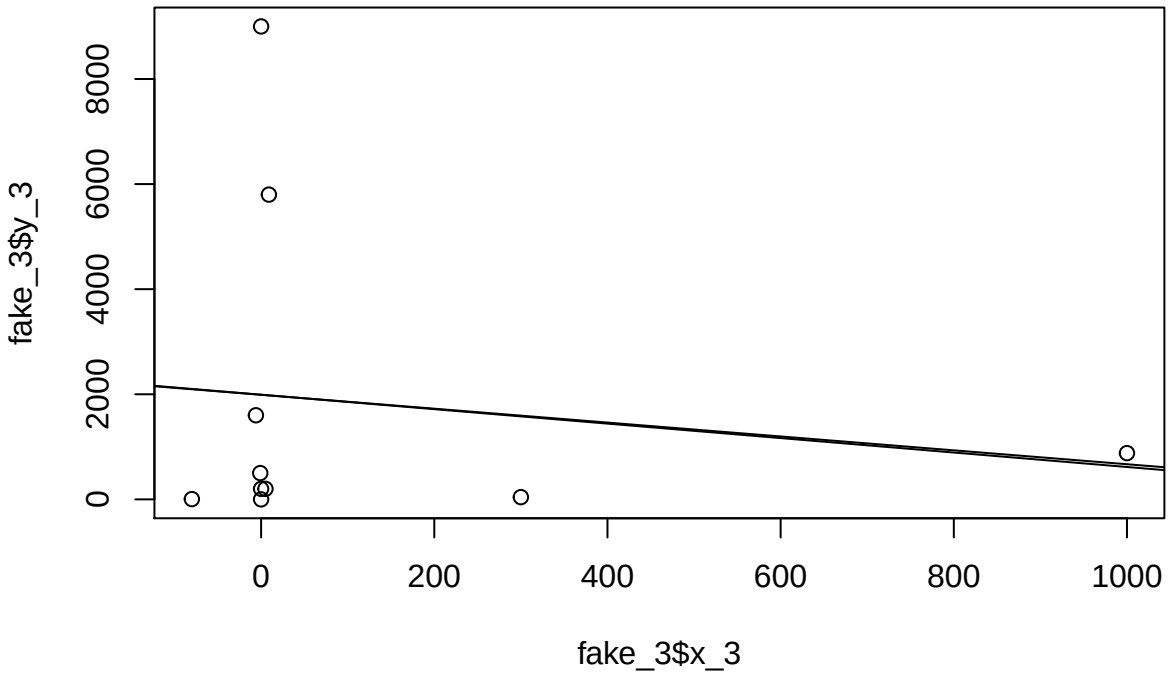
Repeat the two steps above, but try to create conditions for your simulation so that lm and stan_glm give much different results.

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'refresh' will be disregarded
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 1.963 seconds (Warm-up)
## Chain 1:                0.042586 seconds (Sampling)
## Chain 1:                2.00558 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: 1.5701 seconds (Warm-up)
## Chain 2:                0.034846 seconds (Sampling)
## Chain 2:                1.60495 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.12 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: 1.67954 seconds (Warm-up)
## Chain 3:                0.033665 seconds (Sampling)
## Chain 3:                1.71321 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.1e-05 seconds
```

```
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.11 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: 1.59326 seconds (Warm-up)
## Chain 4:                0.032591 seconds (Sampling)
## Chain 4:                1.62585 seconds (Total)
## Chain 4:
```



10.1 Regression with interactions:

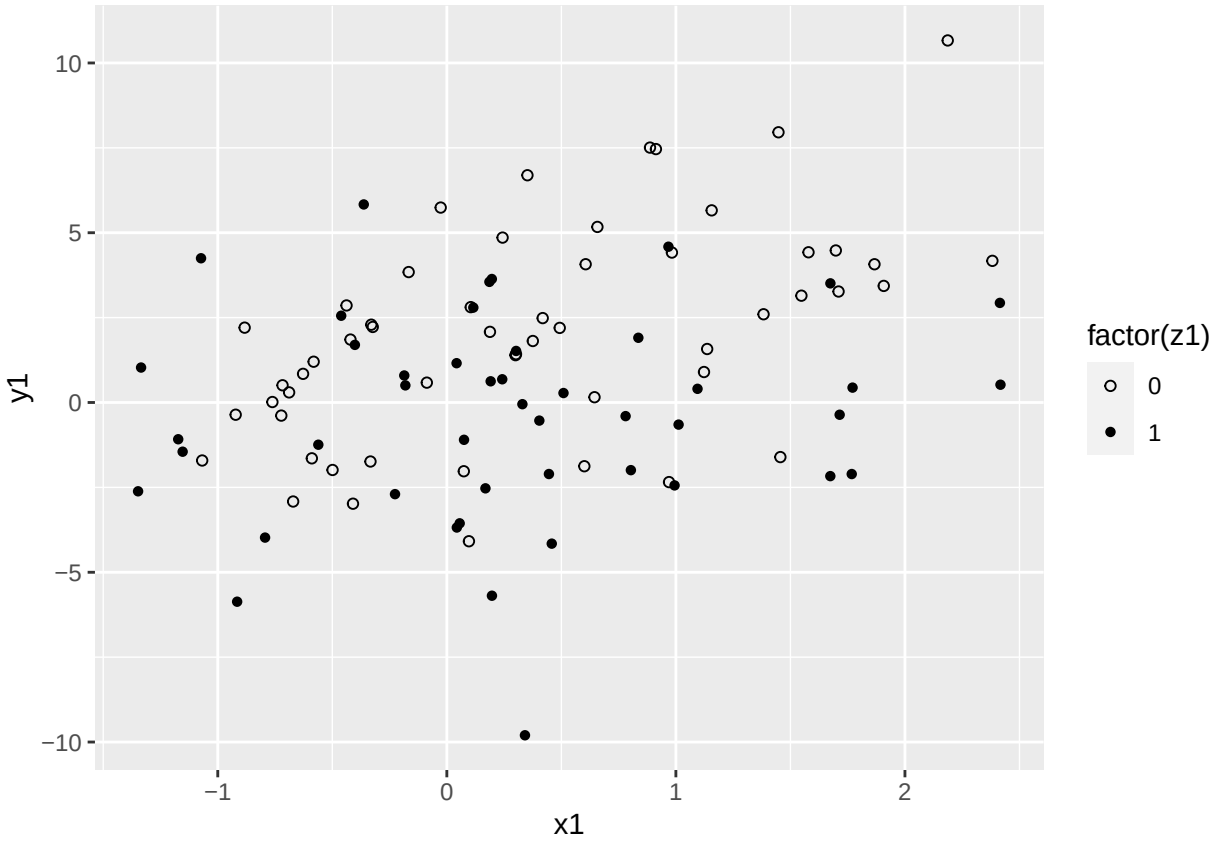
Simulate 100 data points from the model, y = b0 + b1 x + b2 z + b3 xz + error, with a continuous predictor x and a binary predictor z, coefficients b = c(1, 2, -1, -2), and errors drawn independently from a normal distribution with mean 0 and standard deviation 3, as follows. For each data point i, first 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.

```
n <- 100
z1 <- rbinom(n, 1, 0.5)

x1 <- rnorm(n, mean(z1), 1)
error <- rnorm(n, 0, 3)
y1 <- 1 + 2*x1 - z1 -2*x1*z1 + error
xyz <- data.frame(x1, z1, y1)
ggplot(data = xyz)+
  geom_point(
    mapping = aes(x=x1, y=y1, shape = factor(z1)))+
  scale_shape_manual(values = c(1, 16))
```




```
# print(z1)
```

10.1b

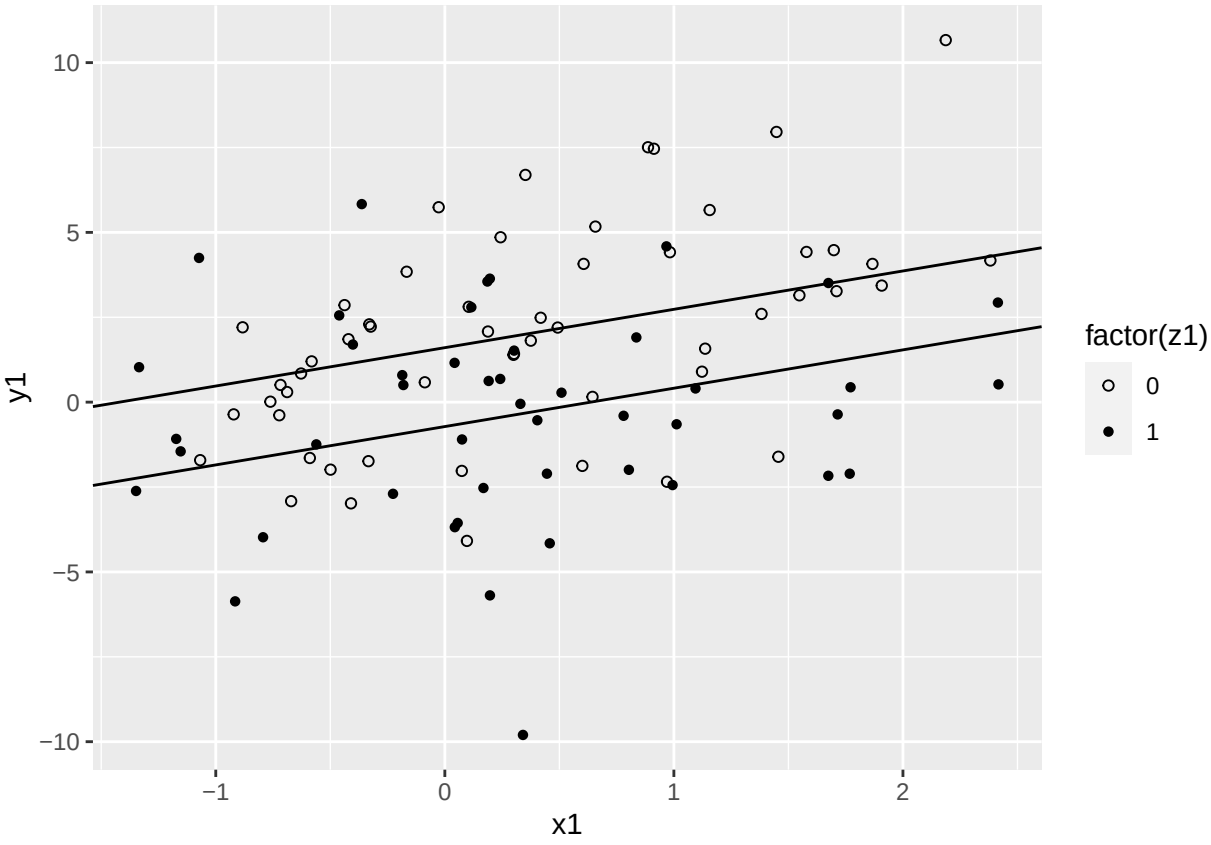
Fit a regression predicting y from x and z with no interaction. Make a graph with the data and two parallel lines showing the fitted model.

```
fit_xyz <- stan_glm(y1 ~ x1 + z1, data = xyz, refresh =0 )

int_0 <- coef(fit_xyz)[1]
slope1 <- coef(fit_xyz)[2]
z_coef <- coef(fit_xyz)[3]
```

```
int_1 <- int_0 + z_coef

ggplot(data = xyz)+
  geom_point(mapping = aes(x = x1, y = y1, shape = factor(z1) ))+
  scale_shape_manual(values = c(1, 16))+
  geom_abline(
    aes(intercept = int_0, slope = slope1)
    )+
  geom_abline(
    aes(intercept = int_1, slope = slope1)
  )
```



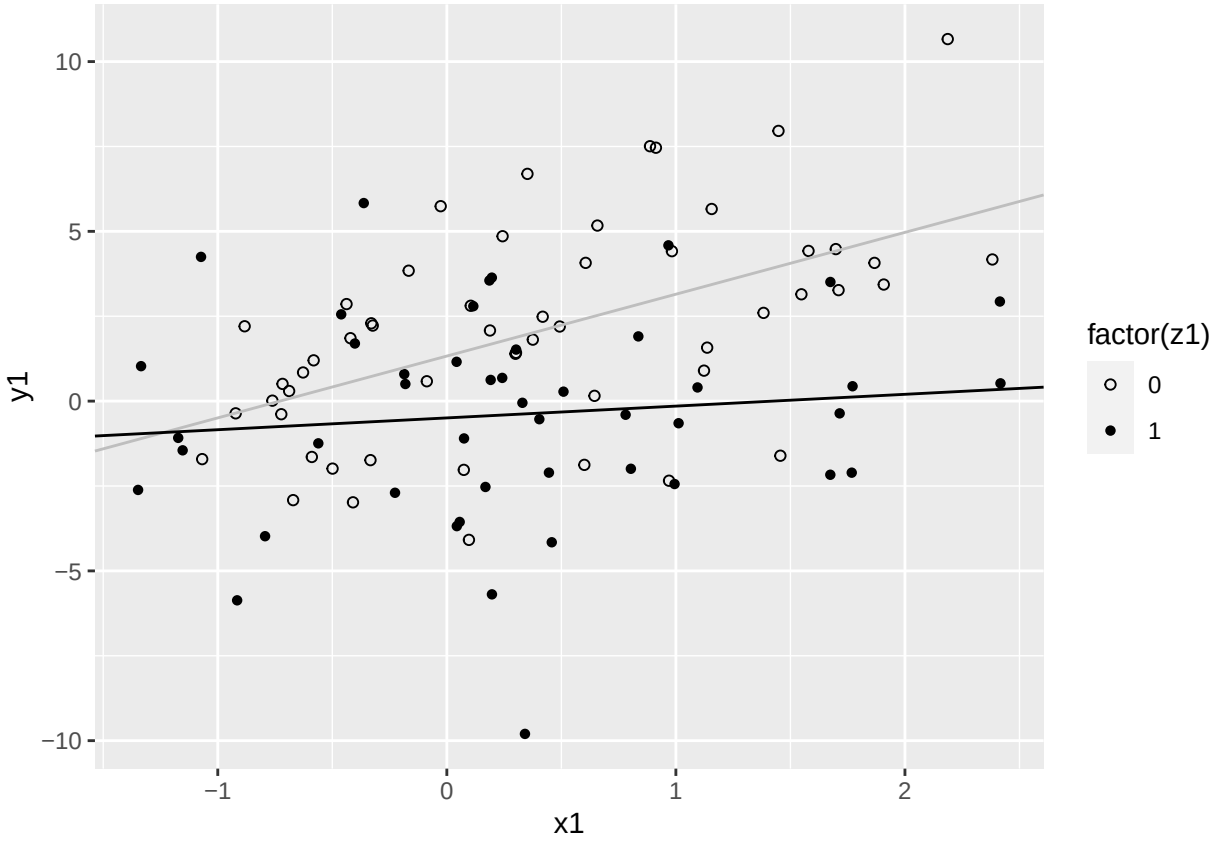
10.1c

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

```
fit_xyz2 <- stan_glm(y1 ~ x1 + z1 + x1:z1, data = xyz, refresh =0 )

int_z0 <- coef(fit_xyz2)[1]
slope_x0 <- coef(fit_xyz2)[2]
z_coef2 <- coef(fit_xyz2)[3]
xz_coef <- coef(fit_xyz2)[4]
# y0 = int2 + slope_x*x
# y1 = int2 +slope-x*x + z_coef +xz-coef*x
int_z1 <- int_z0 + z_coef2
slope_x1 = slope_x0 + xz_coef
```

```
ggplot(data = xyz)+
  geom_point(
    mapping =
      aes(x = x1, y = y1, shape = factor(z1)))+
  scale_shape_manual(values = c(1, 16))+
  geom_abline(
    aes(intercept = int_z0, slope = slope_x0), color = "grey"
  )+
  geom_abline(
    aes(intercept = int_z1, slope = slope_x1)
  )
```



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

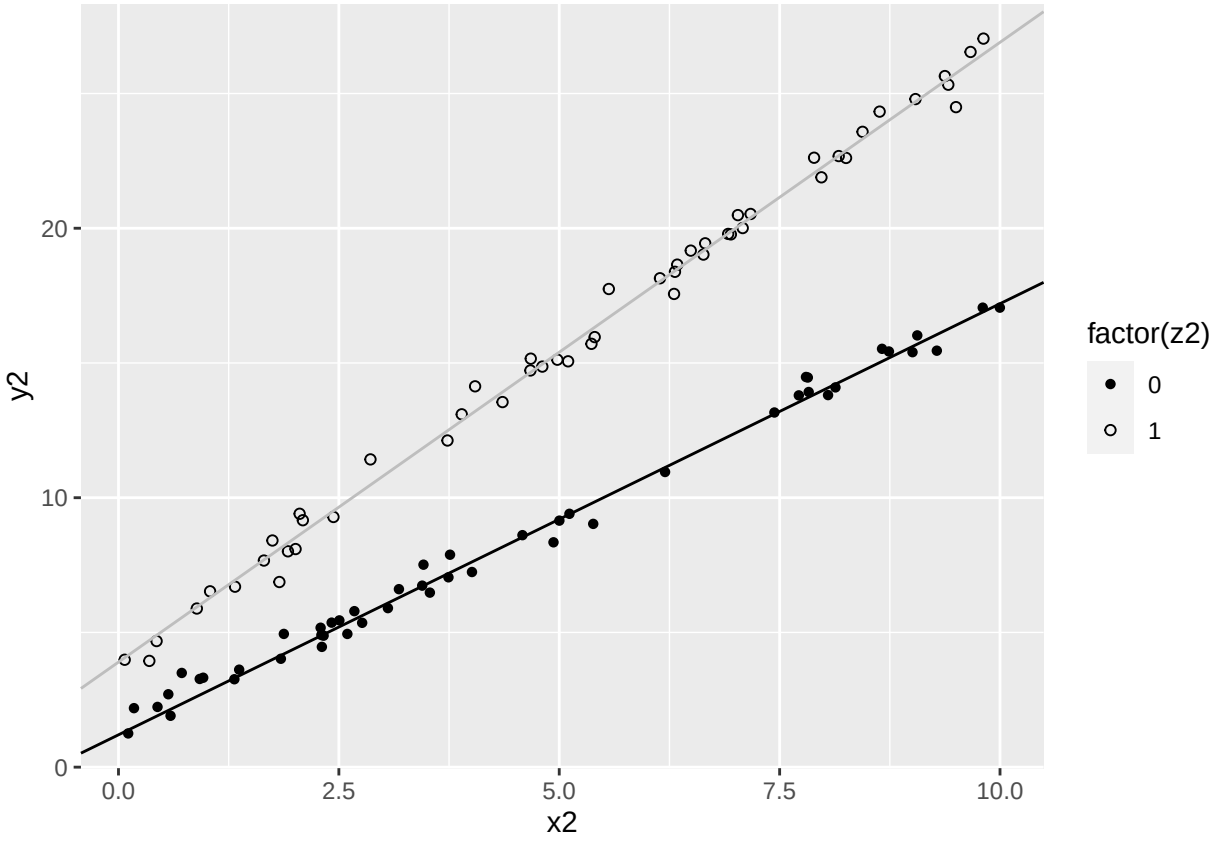
Write the equation of the estimated regression line of y on x for the treatment group and the control group, and the equation of the estimated regression line of y on x for the control group.

Answer: Equation for both treatment and control group: y = 1.2 + 1.6x + 2.7z + 0.7x*z Equation for control group(z=0): 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.

```
x2 <- runif(100, 0, 10)
e2 <- rnorm(100, 0, 0.5)
z2 <- rbinom(100, 1, 0.5)
# plot(z2)
y2 <- 1.2 + 1.6*x2 + 2.7*z2 + 0.7*x2*z2 +e2
xyz2 <- data.frame(x2, y2, z2)
ggplot(data = xyz2)+
  geom_point(
    mapping = aes(x=x2, y=y2, shape = factor(z2))
  )+
  scale_shape_manual(values = c(16, 1))+
  geom_abline(
    aes(intercept = 1.2, slope = 1.6)
  )+
  geom_abline(
    aes(intercept = 1.2+2.7, slope = 1.6+0.7), color = "grey"
  )
```



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("kidiq.csv")
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

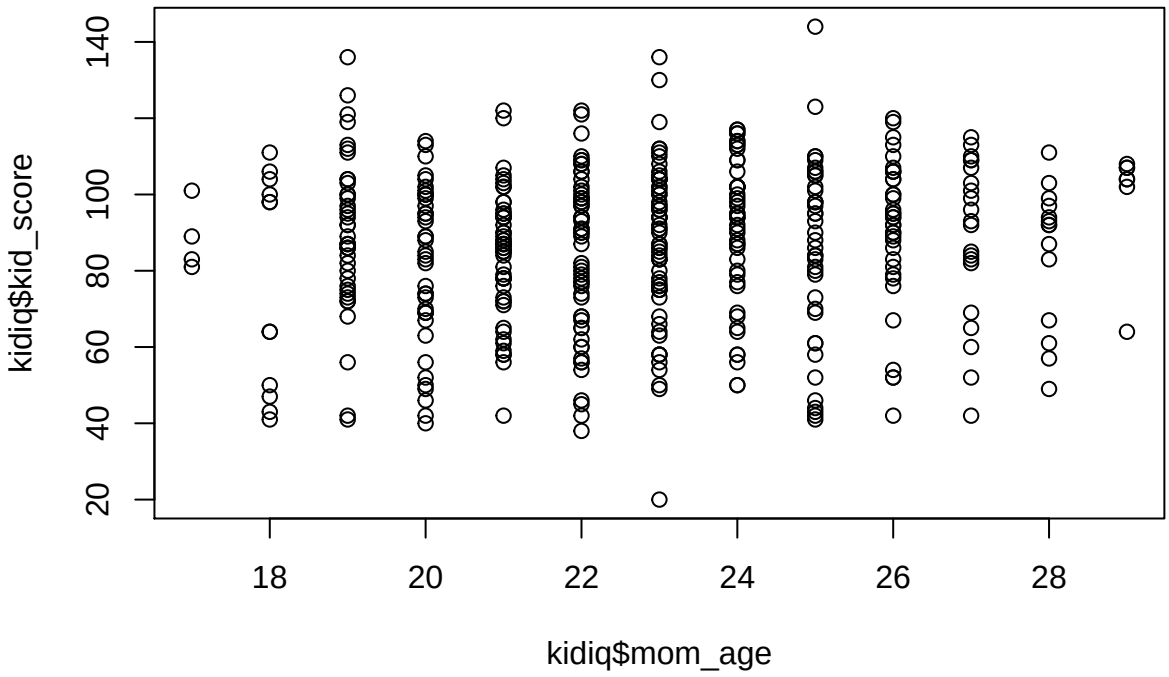
Fit a regression of child test scores on mother's age, display the data and fitted model, check assumptions, and interpret the slope coefficient. Based on this analysis, when do you recommend mothers should give birth? What are you assuming in making this recommendation?

Answer:Based on the positive coefficient(0.7) of mother's age on kid's score, assuming momthers' other conditions are the same, I would recommend mothers to give birth later.

```
fit5a <- stan_glm(kid_score ~ mom_age, data = kidiq, refresh =0)
print(fit5a)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      kid_score ~ mom_age
##  observations: 434
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 70.9    8.3
## mom_age      0.7    0.4
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 20.4    0.7
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
plot(kidiq$mom_age, kidiq$kid_score)
```



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?

Answer: After adding mother's education, the effect of age decreases, but it's still positive, so my conclusion of the timing of birth does not change.

```
fit5b <- stan_glm (kid_score ~ mom_age + mom_hs, data = kidiq, refresh =0)
print(fit5b)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      kid_score ~ mom_age + mom_hs
##  observations: 434
##  predictors:   3
## ------
##             Median MAD_SD
## (Intercept) 70.5    8.3
## mom_age      0.3    0.4
## mom_hs      11.3    2.4
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 19.9    0.7
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

10.5c

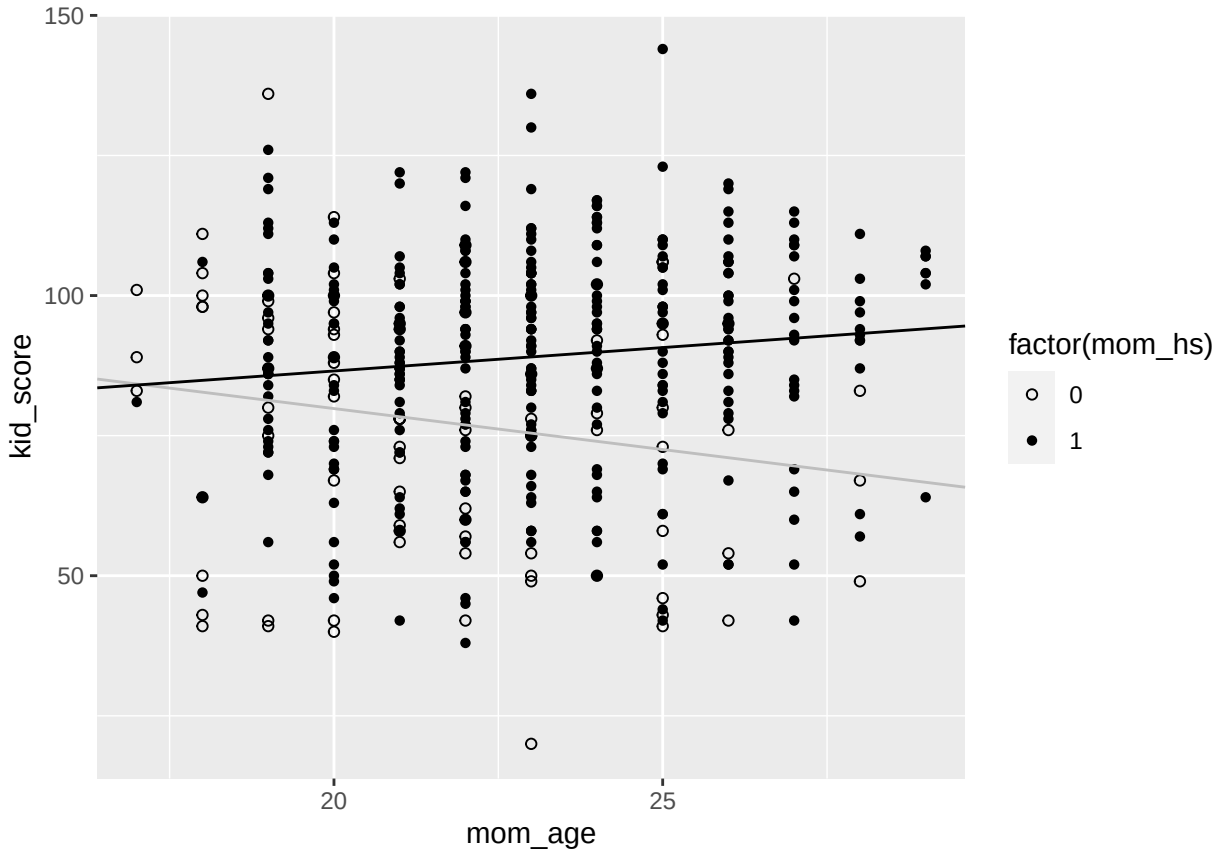
Now create an indicator variable reflecting whether the mother has completed high school or not. Consider interactions between high school completion and mother's age. Also create a plot that shows the separate

regression lines for each high school completion status group.

```
fit5c <- stan_glm (kid_score ~ mom_age + mom_hs + mom_age:mom_hs, data = kidiq, refresh =0)
print(fit5c)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      kid_score ~ mom_age + mom_hs + mom_age:mom_hs
##  observations: 434
##  predictors:   4
## ------
##                Median MAD_SD
## (Intercept)    109.1   15.4
## mom_age         -1.5    0.7
## mom_hs         -39.3   17.9
## mom_age:mom_hs   2.3    0.8
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 19.7    0.7
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
ggplot(data = kidiq) +
  geom_point(
    mapping = aes(x = mom_age, y=kid_score, shape = factor(mom_hs)))+
  scale_shape_manual(values = c(1, 16))+
  geom_abline(aes(intercept=coef(fit5c)[1], slope=coef(fit5c)[2]), color = "grey")+
  geom_abline(aes(intercept =coef(fit5c)[1]+ coef(fit5c)[3], slope=coef(fit5c)[2]+coef(fit5c)[4]))
```



10.5d

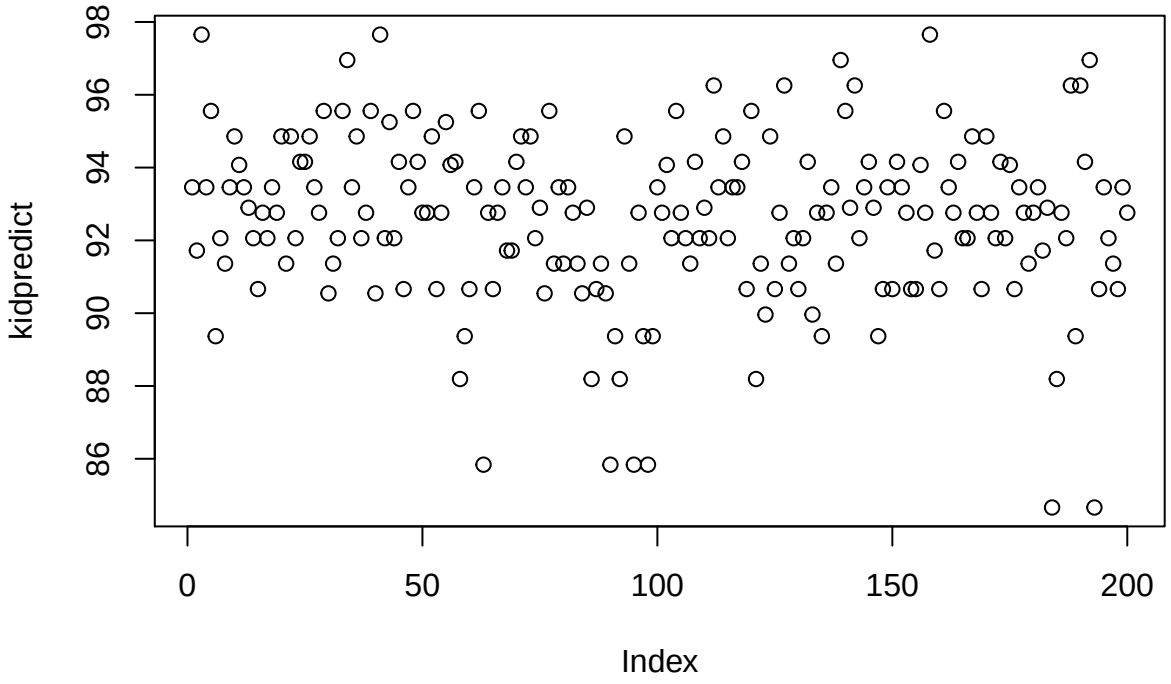
Finally, fit a regression of child test scores on mother's age and education level for the first 200 children and use this model to predict test scores for the next 200. Graphically display comparisons of the predicted and actual scores for the final 200 children.

```
kid_model <- kidiq[1:200,]
kid_test <- kidiq[201:400,]
fit5d <- stan_glm(kid_score ~ mom_hs + mom_age + mom_hs:mom_age, data = kid_model, refresh=0)
print(fit5d)
```

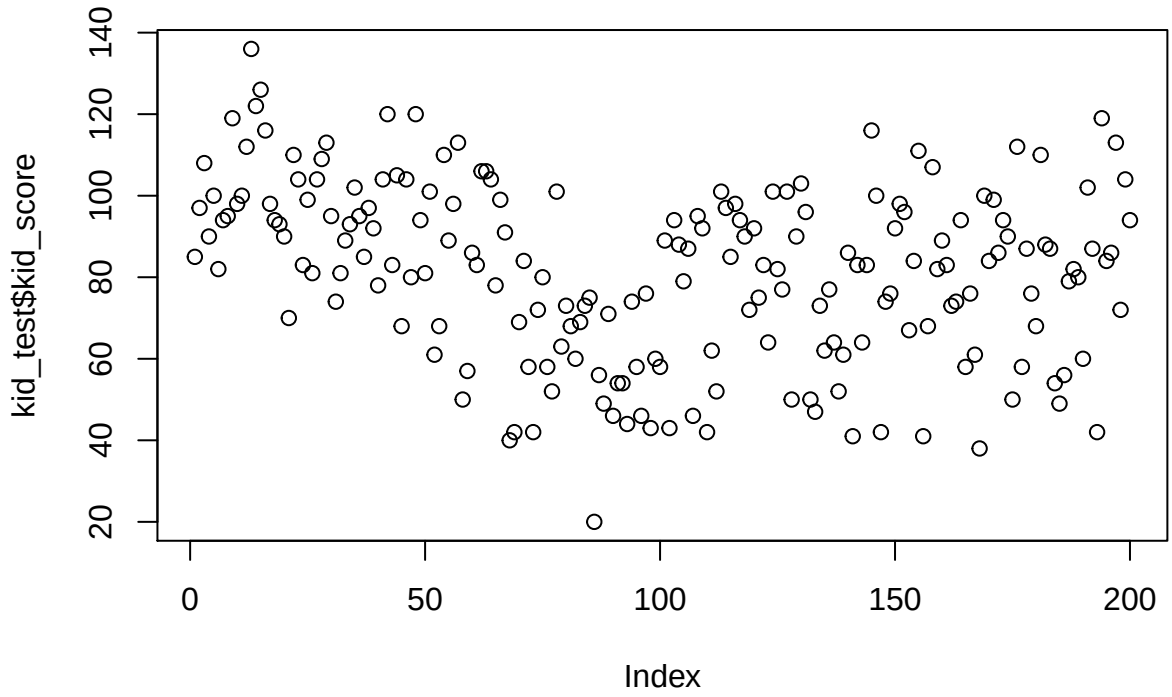
```
## stan_glm
##  family:       gaussian [identity]
##  formula:      kid_score ~ mom_hs + mom_age + mom_hs:mom_age
##  observations: 200
##  predictors:   4
## ------
##                Median MAD_SD
## (Intercept)    115.3   22.5
## mom_hs         -37.9   25.1
## mom_age         -1.2    1.0
## mom_hs:mom_age   1.9    1.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 17.6    0.9
##
```

```
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
kidpredict <- coef(fit5d)[1] + coef(fit5d)[2]*kid_test$mom_hs +
    coef(fit5d)[3]*kid_test$mom_age + coef(fit5d)[4]* kid_test$mom_hs * kid_test$mom_age
plot(kidpredict)
```




```
plot(kid_test$kid_score)
```



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("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
```

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.

```
fit6a <- stan_glm(eval ~ beauty, data = beauty, refresh = 0)
print(fit6a)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty
##  observations: 463
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 4.0    0.0
## beauty      0.1    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
```

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.

Answer: For the first model fit6b1, the intercept means for male instructors whose beauty score is 0, their average evaluation score is 4.21; the coefficient for beauty means that, for male instructors, 1 difference in beauty score corresponds to 0.19 in evaluation score, and for female instructors, 1 difference in beauty score corresponds to (0.19-0.11)=0.08 in evaluation score; the coefficient of age is close to zero, meaning age is not

a significant predictor for evaluation score; the coefficient of female and its interaction with beauty indicate that, comparing instructors with the same beauty scores but with different gender, female instructors on average correspond to (-0.22-0.11)= -0.33 evaluation score than male instructors.

For the second model fit6b2,

```
fit6b1 <- stan_glm(eval ~ beauty + age + female + female:beauty, data = beauty, refresh = 0)
print(fit6b1, digits=2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + age + female + female:beauty
##  observations: 463
##  predictors:   5
## ------
##               Median MAD_SD
## (Intercept)    4.22   0.15
## beauty         0.19   0.04
## age            0.00   0.00
## female        -0.22   0.05
## beauty:female -0.11   0.06
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.54   0.02
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
fit6b2 <- stan_glm(eval ~ beauty + minority + minority:beauty, data = beauty, refresh = 0)
print(fit6b2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + minority + minority:beauty
##  observations: 463
##  predictors:   4
## ------
##                 Median MAD_SD
## (Intercept)      4.0    0.0
## beauty           0.2    0.0
## minority        -0.1    0.1
## beauty:minority -0.2    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
```

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.

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?