PS1, Economics C142, Spring 2019

Vinay Maruri

January 28, 2019

Problem 1: Case 1: n=30, p=0.05

```
means_1 <- vector('numeric')
sds_1 <- vector('numeric')
upperci_1 <- vector('numeric')
lowerci_1 <- vector('numeric')
length_1 <- vector('numeric')
intervalcapture_1 <- vector('numeric')
true_mean <- 0.05
for (i in 1:1000) {
  x_1 <- rbinom(30, 1, 0.05)
  temp_mean <- mean(x_1)
  means_1 <- c(means_1, temp_mean)
  std_dev <- sd(x_1)
  sds_1 <- c(sds_1, std_dev)
  upperbound <- temp_mean + 1.96*(std_dev/(30^(1/2)))
  upperci_1 <- c(upperci_1, upperbound)
  lowerbound <- temp_mean - 1.96*(std_dev/(30^(1/2)))
  lowerci_1 <- c(lowerci_1, lowerbound)
  interval_length <- upperbound - lowerbound
  length_1 <- c(length_1, interval_length)
  if (true_mean >= lowerbound && true_mean <= upperbound) {
    intervalcapture_1 <- c(intervalcapture_1, TRUE)
  } else {
    intervalcapture_1 <- c(intervalcapture_1, FALSE)
  }

  }
```

```
print(mean(means_1))
```

```
## [1] 0.05086667
```

```
#above is mean estimate of p for case 1.
```

```
print(mean(sds_1))
```

```
## [1] 0.1916313
```

```
#above is mean estimate of true standard deviation for case 1.
```

```
print(sum(intervalcapture_1)/length(intervalcapture_1))
```

```
## [1] 0.801
```

```
#above is coverage rate for case 1.
```

Case 2: n=30, p=0.25

```
means_2 <- vector('numeric')
sds_2 <- vector('numeric')
upperci_2 <- vector('numeric')
lowerci_2 <- vector('numeric')
length_2 <- vector('numeric')
intervalcapture_2 <- vector('numeric')
true_mean2 <- 0.25
for (i in 1:1000) {
  x_2 <- rbinom(30, 1, 0.25)
  temp_mean <- mean(x_2)
  means_2 <- c(means_2, temp_mean)
  std_dev <- sd(x_2)
  sds_2 <- c(sds_2, std_dev)
  upperbound <- temp_mean + 1.96*(std_dev/(30^(1/2)))
  upperci_2 <- c(upperci_2, upperbound)
  lowerbound <- temp_mean - 1.96*(std_dev/(30^(1/2)))
  lowerci_2 <- c(lowerci_2, lowerbound)
  interval_length <- upperbound - lowerbound
  if (true_mean2 >= lowerbound && true_mean2 <= upperbound) {
    intervalcapture_2 <- c(intervalcapture_2, TRUE)
  } else {
    intervalcapture_2 <- c(intervalcapture_2, FALSE)
  }

  }
```

```
print(mean(means_2))
```

```
## [1] 0.2514667
```

```
#above is mean estimate of p for case 2.
```

```
print(mean(sds_2))
```

```
## [1] 0.4308419
```

```
#above is mean estimate of true standard deviation for case 2.
```

```
print(sum(intervalcapture_2)/length(intervalcapture_2))
```

```
## [1] 0.942
```

```
#above is coverage rate for case 2.
```

Case 3: n = 60, p = 0.05

```r
means_3 <- vector('numeric')
sds_3 <- vector('numeric')
upperci_3 <- vector('numeric')
lowerci_3 <- vector('numeric')
length_3 <- vector('numeric')
intervalcapture_3 <- vector('numeric')
true_mean3 <- 0.05
for (i in 1:1000) {
  x_3 <- rbinom(60, 1, 0.05)
  temp_mean <- mean(x_3)
  means_3 <- c(means_3, temp_mean)
  std_dev <- sd(x_3)
  sds_3 <- c(sds_3, std_dev)
  upperbound <- temp_mean + 1.96*(std_dev/(30^(1/2)))
  upperci_3 <- c(upperci_3, upperbound)
  lowerbound <- temp_mean - 1.96*(std_dev/(30^(1/2)))
  lowerci_3 <- c(lowerci_3, lowerbound)
  interval_length <- upperbound - lowerbound
  if (true_mean3 >= lowerbound && true_mean3 <= upperbound) {
    intervalcapture_3 <- c(intervalcapture_3, TRUE)
  } else {
    intervalcapture_3 <- c(intervalcapture_3, FALSE)
  }

  }
```

```r
print(mean(means_3))
```

```
## [1] 0.04948333
```

```r
#above is mean estimate of p for case 3.
```

```r
print(mean(sds_3))
```

```
## [1] 0.2054559
```

```r
#above is mean estimate for true standard deviation for case 3.
```

```r
print(sum(intervalcapture_3)/length(intervalcapture_3))
```

```
## [1] 0.952
```

```r
#above is coverage rate for case 3.
```

Case 4: n = 60, p = 0.25

```r
means_4 <- vector('numeric')
sds_4 <- vector('numeric')
upperci_4 <- vector('numeric')
lowerci_4 <- vector('numeric')
```

```
length_4 <- vector('numeric')
intervalcapture_4 <- vector('numeric')
true_mean4 <- 0.25
for (i in 1:1000) {
  x_4 <- rbinom(60, 1, 0.25)
  temp_mean <- mean(x_4)
  means_4 <- c(means_4, temp_mean)
  std_dev <- sd(x_4)
  sds_4 <- c(sds_4, std_dev)
  upperbound <- temp_mean + 1.96*(std_dev/(30^(1/2)))
  upperci_4 <- c(upperci_4, upperbound)
  lowerbound <- temp_mean - 1.96*(std_dev/(30^(1/2)))
  lowerci_4 <- c(lowerci_4, lowerbound)
  interval_length <- upperbound - lowerbound
  if (true_mean4 >= lowerbound && true_mean4 <= upperbound) {
    intervalcapture_4 <- c(intervalcapture_4, TRUE)
  } else {
    intervalcapture_4 <- c(intervalcapture_4, FALSE)
  }

  }
```

```
print(mean(means_4))
```

```
## [1] 0.25205
```

```
#above is mean estimate of p for case 4.
```

```
print(mean(sds_4))
```

```
## [1] 0.4327388
```

```
#above is mean estimate of true standard deviation for case 4.
```

```
print(sum(intervalcapture_4)/length(intervalcapture_4))
```

```
## [1] 0.993
```

```
#above is coverage rate for case 4.
```

I do not agree that n of 30 or larger is enough to ensure that asymptotic confidence intervals work well. For both n=30 cases, and especially when p = 0.05, the coverage rate does not converge to 95%. It is consistently below 95%, meaning that the asymptotic behavior is not observed. However, for n=60, the coverage rate does converge in both cases.

Problem 2:

```
library(ggplot2)
library(haven)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

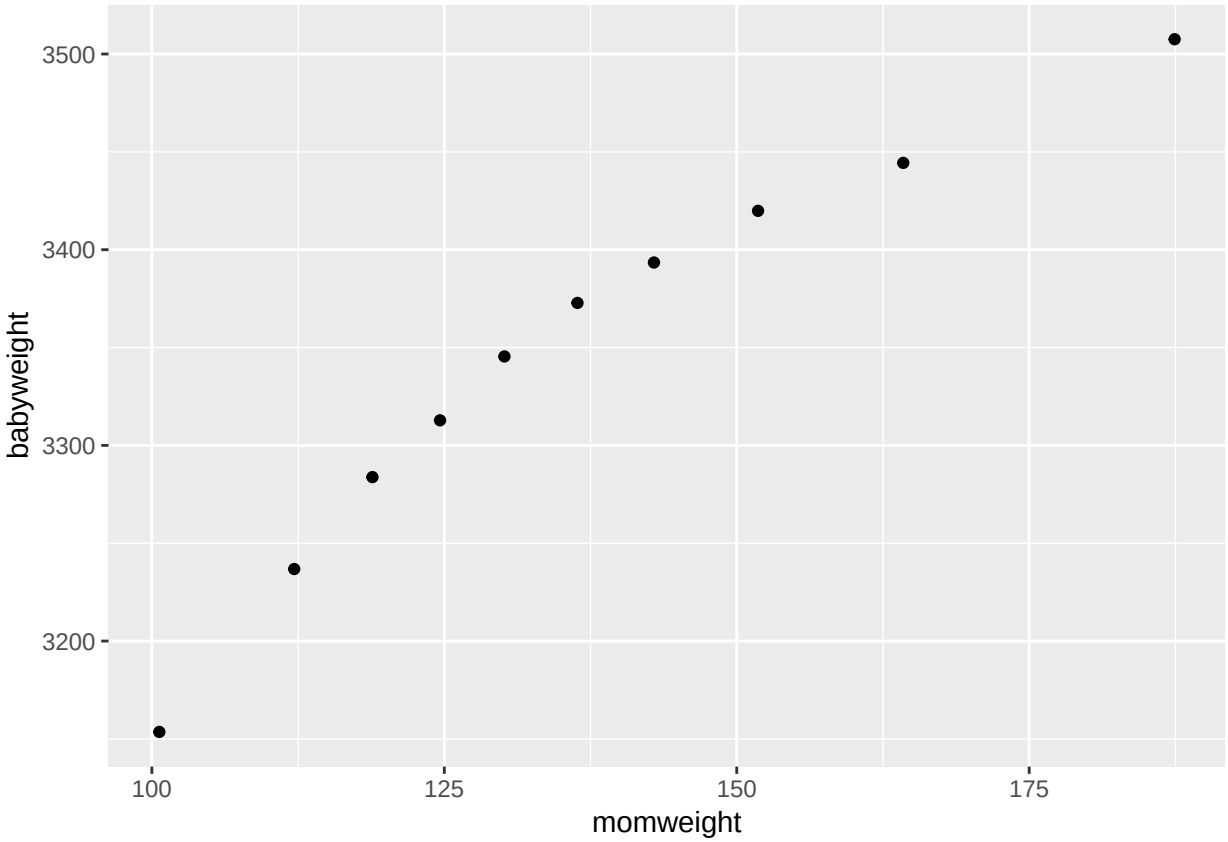
```
library(statar)
```

```
# Load dataset
library(readr)
ps1 <- read.csv("ps1.csv")
head(ps1,4)
```

```
##   momweight babyweight momheight
## 1       120       3365        62
## 2       128       3033        62
## 3       171       3005        68
## 4       112       2948        63
```

(a) i.

```
plot1 <- ggplot(ps1, aes(x = momweight, y = babyweight)) + stat_binmean(n=10)
plot1
```

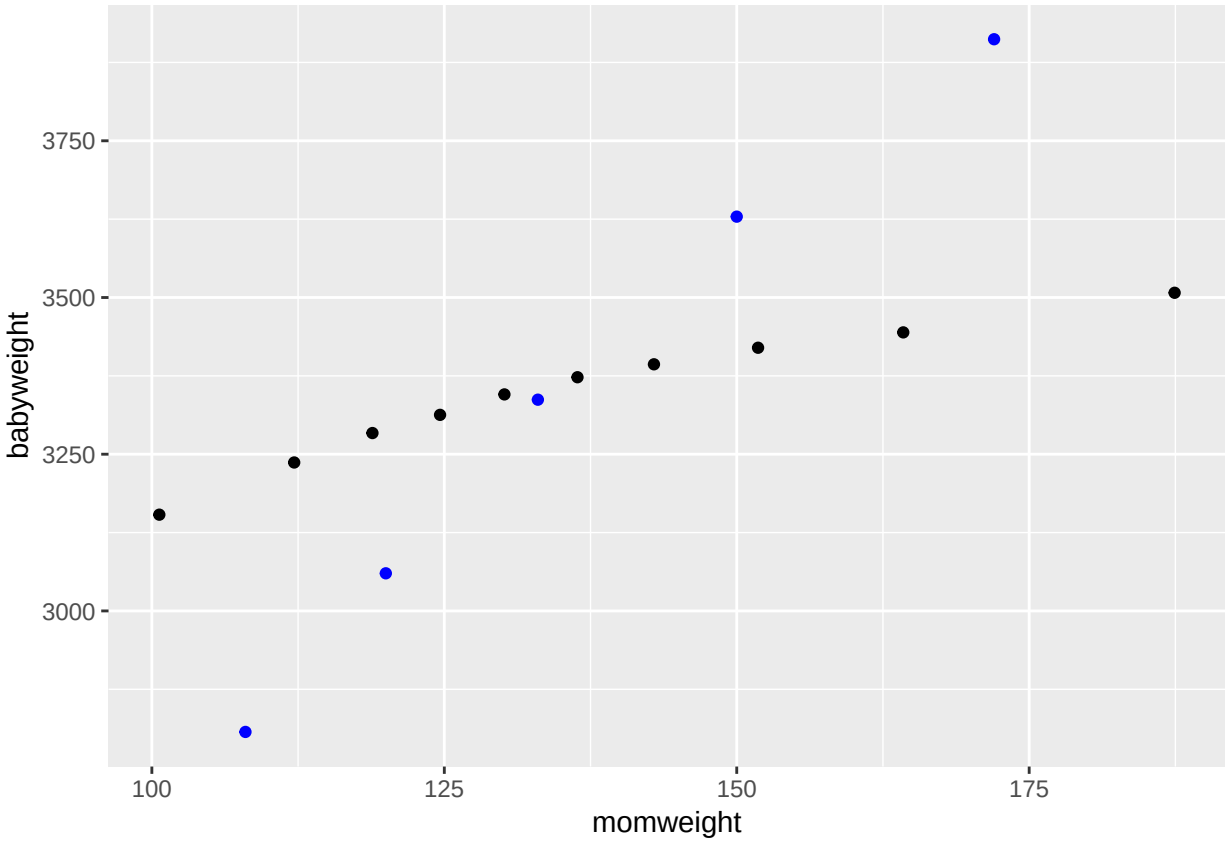



```
#binscatter plot for mean baby weight and against mean mother birthweight in each decile.
```

ii. It appears that there is a positive linear relationship between the mean baby weight specified percentiles and the mean mother's weight for each decile. Most of the points in the binned scatterplot (mean baby weight vs mean mother birthweight in each decile) are between the 25th and 75th percentiles for the pairs of mother's birthweight and their corresponding baby's birthweight. The datapoints appear to follow a Gaussian distribution.

```
babypercents <- quantile(ps1$babyweight, probs = c(0.1,0.25,0.5,0.75,0.9))
mompercents <- quantile(ps1$momweight, probs = c(0.1, 0.25, 0.5, 0.75, 0.9))
combinedpercents <- data.frame(momweight = mompercents, babyweight = babypercents)
```

```
plot2 <- ggplot(ps1, aes(x = momweight, y = babyweight)) + stat_binmean(n=10) +
  geom_point(data = combinedpercents, mapping = aes(x = momweight, y = babyweight), col = 'blue')
plot2
```



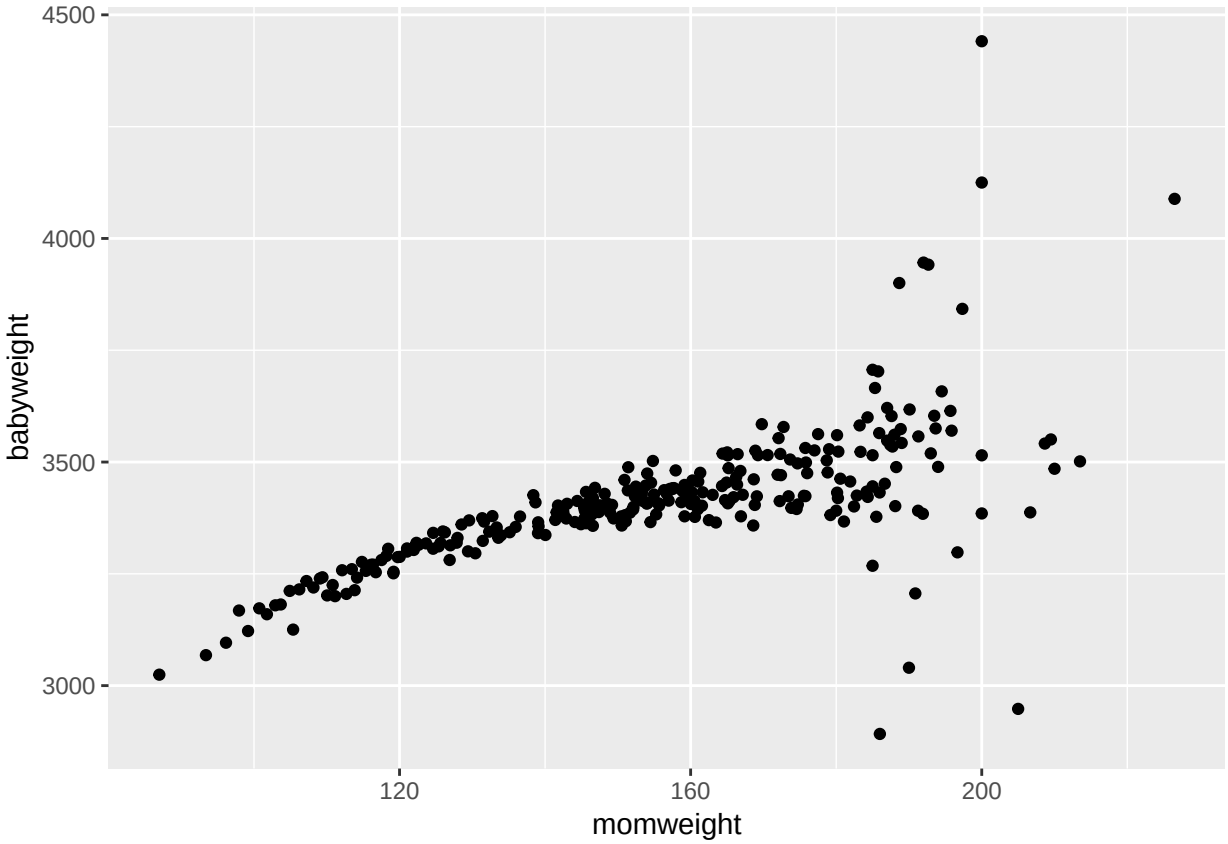

```
#adding 10th, 25th, 50th, 75th and 90th percentiles for baby weight
#plotted against mean mother's weight for mothers in each decile.
```

iii. From 72 pounds until roughly 180 pounds, this plot follows a very similar trend to the bin-scatter plot from part (i). Steadily increasing baby weights in a non-linear fashion for increases in mother's birth weight. However, above 180 pounds, this plot diverges, as there appears to be a lot of noise in the data.

```
plot3 <- ggplot(ps1, aes(x=momweight, y=babyweight)) + stat_binmean(n = unique(ps1$momweight))
plot3
```

```
## Warning in if (n == 0) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in n * (row_number(x) - 1): longer object length is not a multiple
## of shorter object length
```




```
#binscatter using individual values of mother's weight instead of 10 deciles of mother's weight.
```

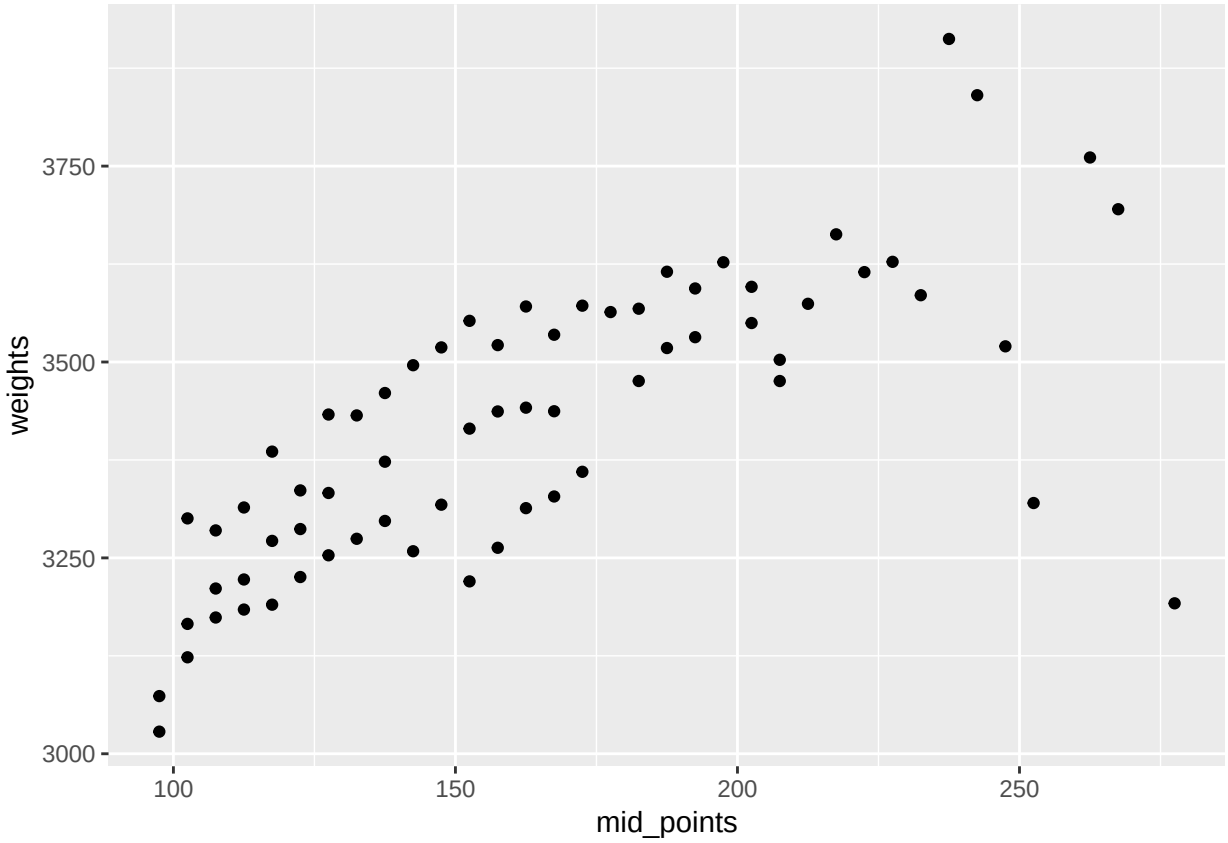
2 (b)

```
init_weight <- 95
weights <- vector('numeric')
mid_points <- vector('numeric')
while (init_weight < 280) {
  temp <- init_weight + 5
  matching <- ps1[ps1$momweight >= init_weight, ]
  matching <- matching[matching$momweight < temp, ]
  sixty_or_under <- matching[matching$momheight <= 60, ]
  sixone_to_sixthree <- matching[matching$momheight >= 61 && matching$momheight <= 63, ]
  sixfour_to_sixsix <- matching[matching$momheight >= 64 && matching$momheight <= 66, ]
  six67_above <- matching[matching$momheight >= 67, ]
  mid <- (init_weight + temp)/2
  point1_mean <- mean(sixty_or_under$babyweight)
  point2_mean <- mean(sixone_to_sixthree$babyweight)
  point3_mean <- mean(sixfour_to_sixsix$babyweight)
  point4_mean <- mean(six67_above$babyweight)
  if (!is.na(point1_mean)) {
  weights <- c(weights, point1_mean)
  mid_points <- c(mid_points, mid)
  }
  if (!is.na(point2_mean)) {
```

```
  weights <- c(weights, point2_mean)
  mid_points <- c(mid_points, mid)
  }
  if(!is.na(point3_mean)) {
  weights <- c(weights, point3_mean)
  mid_points <- c(mid_points, mid)
  }
  if(!is.na(point4_mean)) {
  weights <- c(weights, point4_mean)
  mid_points <- c(mid_points, mid)
  }
  init_weight <- init_weight + 5
}
#code above iteratively considers each mother weight 5 pound bucket,
#and calculates baby weights separately for each momheight bucket.
#Vectors are created for mean baby weights for the 4 groups, and a vector
#of mid-point weights in each bucket.
```

```
pointvector <- data.frame(weights, mid_points)
#dataframe to merge the 2 vectors together.
```

```
qplot(x = mid_points, y = weights, data = pointvector)
```



```
#produces stacked plot, where there are multiple y-values for a given x-value.
#In other words, for the mid-point weight in a mother's weight bucket (x-values),
#there are multiple mean baby weights for the different mother height buckets.
#Points at the same x, but higher y are indicative of taller mothers.
```

We can conclude that there is a positive, linear relationship between mother's weight and the baby's weight, and that there is some positive relationship between a mother's height and a baby's weight.

(c) I have constructed a 3D binscatter, where mother's height and weight have been put into 25 bins, then plotted against baby's weight.

```
library(scatterplot3d)
bin_x <- quantile(ps1$momweight, probs = seq(0, 1, 0.04))
bin_z <- quantile(ps1$momheight, probs = seq(0, 1, 0.04))
bin_y <- quantile(ps1$babyweight, probs = seq(0, 1, 0.04))
all_bins <- data.frame(bin_x, bin_y, bin_z)
#x variable is mother's weight, y variable is baby's weight, and z variable is mother's height.
plot5 <- scatterplot3d(all_bins, main="3D Bin Scatter",
              xlab = "Mother's Birthweight (LB)",
              ylab = "Baby's birthweight (grams)",
              zlab = "Mother's Height (in.)")
```