Stat 631 Chapter 6: Background on Regression Modeling

Prof Kapitula

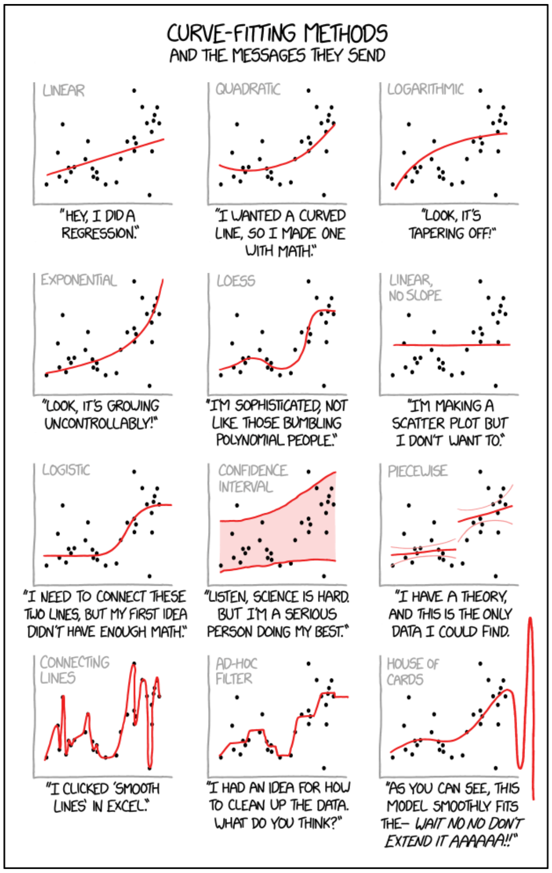

CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND
LINEAR
"HEY, I DID A REGRESSION."
QUADRATIC
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."
LOGARITHMIC
"LOOK, IT'S TAPERING OFF!"
EXPONENTIAL
"LOOK, IT'S GROWING UNCONTROLLABLY!"
LOESS
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."
LINEAR, NO SLOPE
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."
LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!"

Figure 1: curve_fitting

Regression Models

The simplest regression model is linear with one predictor

$$y = a + bx + \epsilon$$

where ϵ is error, a and b are coefficients (parameters) and y and x are variables.

Generalizations:

- Multiple Linear Regression,

$$y = \beta_o + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon$$

or in matrix form:

$$y = X\beta + \epsilon$$

- Models that allow for non-linear responses and predictors such as:

$$\log(y) = a + b\log(x) + \epsilon$$

- Models that allow for interaction, aka nonadditive models

$$y = \beta_o + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon$$

* Generalized Linear Models which are of the form,

$$g(y) = X\beta + \epsilon$$

where we can model data that does not fit the normal model well, such as binary data.

Simple Simulations

```
library("arm")
#library(rstanarm)
```

```
x <- 1:20
n <- length(x)
a <- 0.2
b <- 0.3
sigma <- 0.5
# set the random seed to get reproducible results
# change the seed to experiment with variation due to random noise
set.seed(2141)
y <- a + b*x + sigma*rnorm(n)
fake <- data.frame(x, y)
```

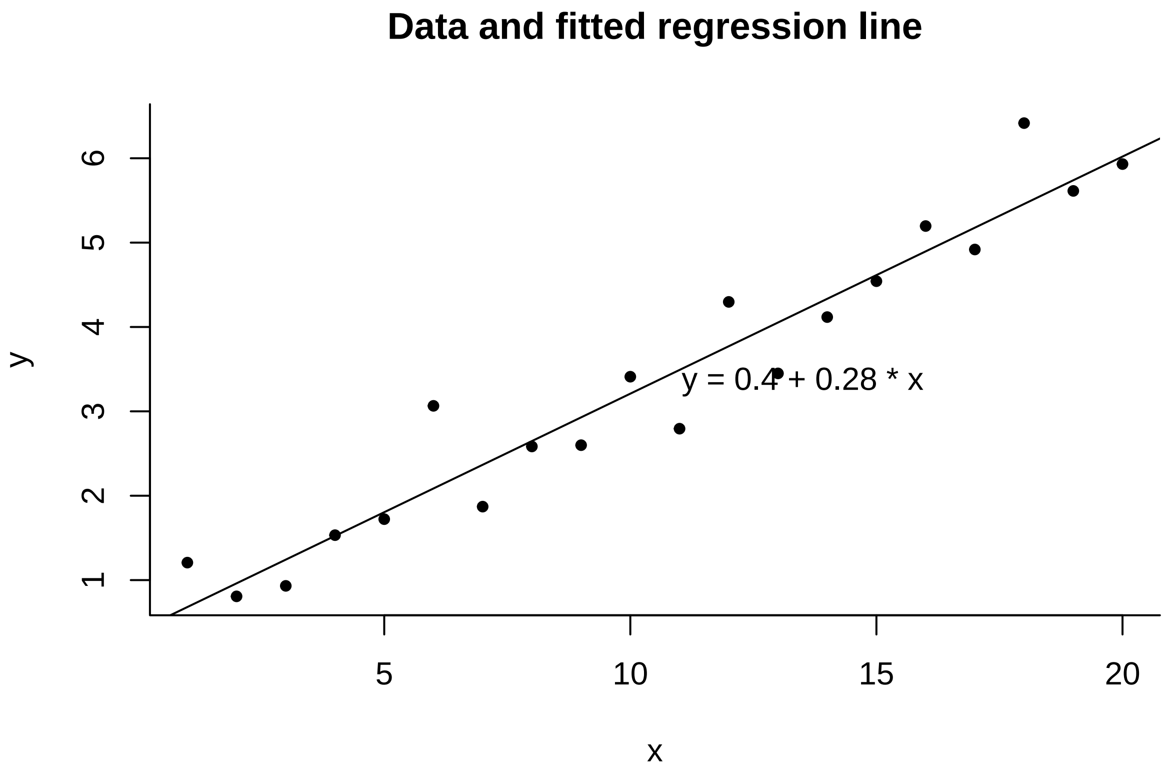
Linear least squares regression

```
fit <- lm(y ~ x, data=fake)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x, data = fake)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6965 -0.2717 -0.0858  0.2258  0.9797
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.39947    0.22008   1.815   0.0862 .
## x            0.28104    0.01837  15.297 9.28e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4738 on 18 degrees of freedom
## Multiple R-squared:  0.9286, Adjusted R-squared:  0.9246
## F-statistic:   234 on 1 and 18 DF,  p-value: 9.277e-12
```

Plot Simulated Regression

```
plot(fake$x, fake$y, main="Data and fitted regression line", bty="l", pch=20,
     xlab = "x", ylab = "y")
a_hat <- coef(fit)[1]
b_hat <- coef(fit)[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)
```



Formulating comparisons as regression models

Simulate fake data

```
n_0 <- 20
# set the random seed to get reproducible results
# change the seed to experiment with variation due to random noise
set.seed(2141)
y_0 <- rnorm(n_0, 2, 5)
y_0 <- round(y_0, 1)
round(y_0, 1)
```

```
##  [1]  9.1  2.1  0.3  3.3  2.2 12.7 -2.3  1.8 -1.0  4.1 -5.1  7.0 -4.5 -0.8  0.4
## [16]  4.0 -1.8 10.2 -0.9 -0.7
```

```
round(mean(y_0), 2)
```

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

```
round(sd(y_0)/sqrt(n), 2)
```

```
## [1] 1.06
```

Estimating the mean is the same as regressing on a constant term

```
summary(lm(y_0 ~ 1))
```

```
##
## Call:
## lm(formula = y_0 ~ 1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.105 -2.930 -0.905  2.020 10.695
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.005      1.065   1.883   0.0751 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.762 on 19 degrees of freedom
```

Simulate fake data

```
n_1 <- 30
# set the random seed to get reproducible results
# change the seed to experiment with variation due to random noise
set.seed(2141)
y_1 <- rnorm(n_1, 8, 5)
diff <- mean(y_1) - mean(y_0)
se_0 <- sd(y_0)/sqrt(n_0)
se_1 <- sd(y_1)/sqrt(n_1)
se <- sqrt(se_0^2 + se_1^2)
print(diff)
```

```
## [1] 6.68748
```

```
print(se)
```

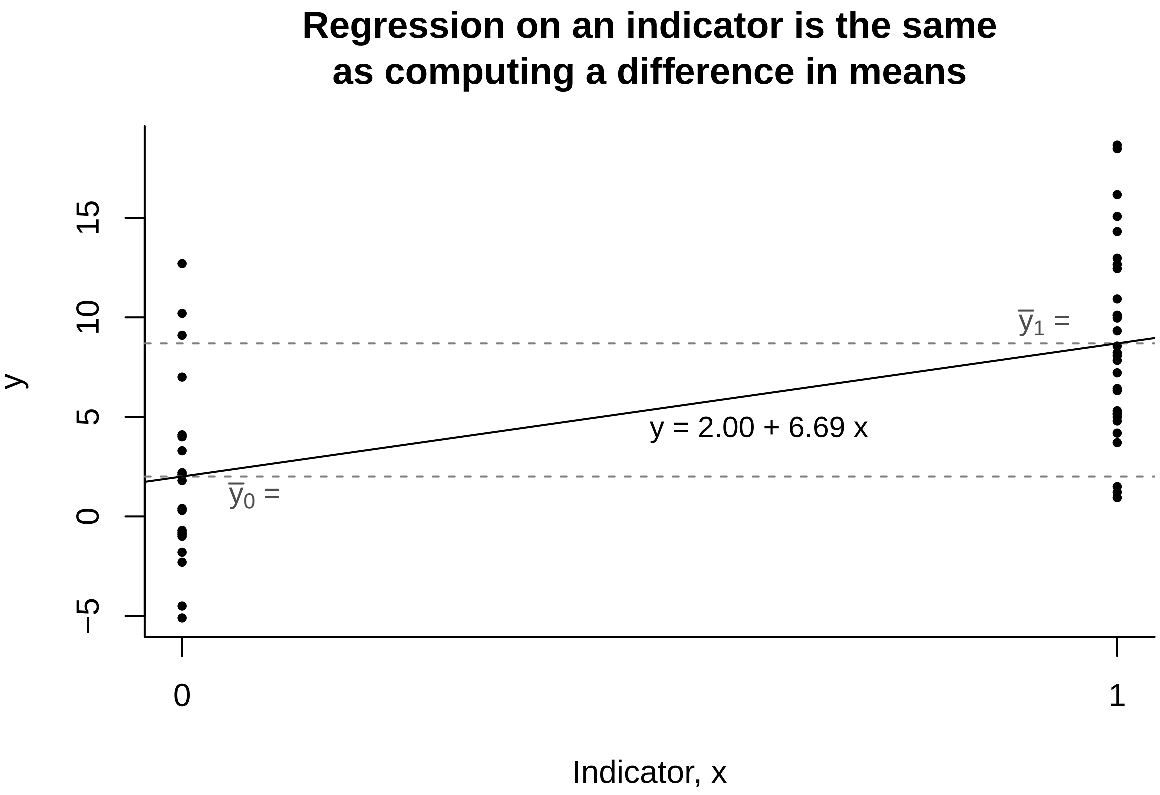
```
## [1] 1.381859
```

Estimating a difference is the same as regressing on an indicator variable

```
n <- n_0 + n_1
y <- c(y_0, y_1)
x <- c(rep(0, n_0), rep(1, n_1))
fit <- lm(y ~ x)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7485 -3.4843 -0.5439  2.1989 10.6950
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.005      1.073   1.868   0.0678 .
## x              6.687      1.385   4.827 1.45e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.799 on 48 degrees of freedom
## Multiple R-squared:  0.3268, Adjusted R-squared:  0.3127
## F-statistic:  23.3 on 1 and 48 DF,  p-value: 1.45e-05
```

```
plot(x, y, xlab="Indicator, x", ylab="y", bty="l", xaxt="n", main="Regression on an indicator is the sa
axis(1, c(0, 1))
abline(h=mean(y[x==0]), lty=2, col="gray50")
abline(h=mean(y[x==1]), lty=2, col="gray50")
abline(coef(fit)[1], coef(fit)[2])
text(.5, -1 + coef(fit)[1] + .5*coef(fit)[2], paste("y =", fround(coef(fit)[1], 2), "+", fround(coef(fi
text(.05, -1 + mean(y[x==0]), expression(paste(bar(y)[0], " =")), col="gray30", cex=.9, adj=0)
text(.95, 1 + mean(y[x==1]), expression(paste(bar(y)[1], " =")), col="gray30", cex=.9, adj=1)
```



Historical Origins and Regression to the Mean

Load packages.

```
library(ggplot2)
library(rstanarm)
library(HistData)
```

data

Import the data.

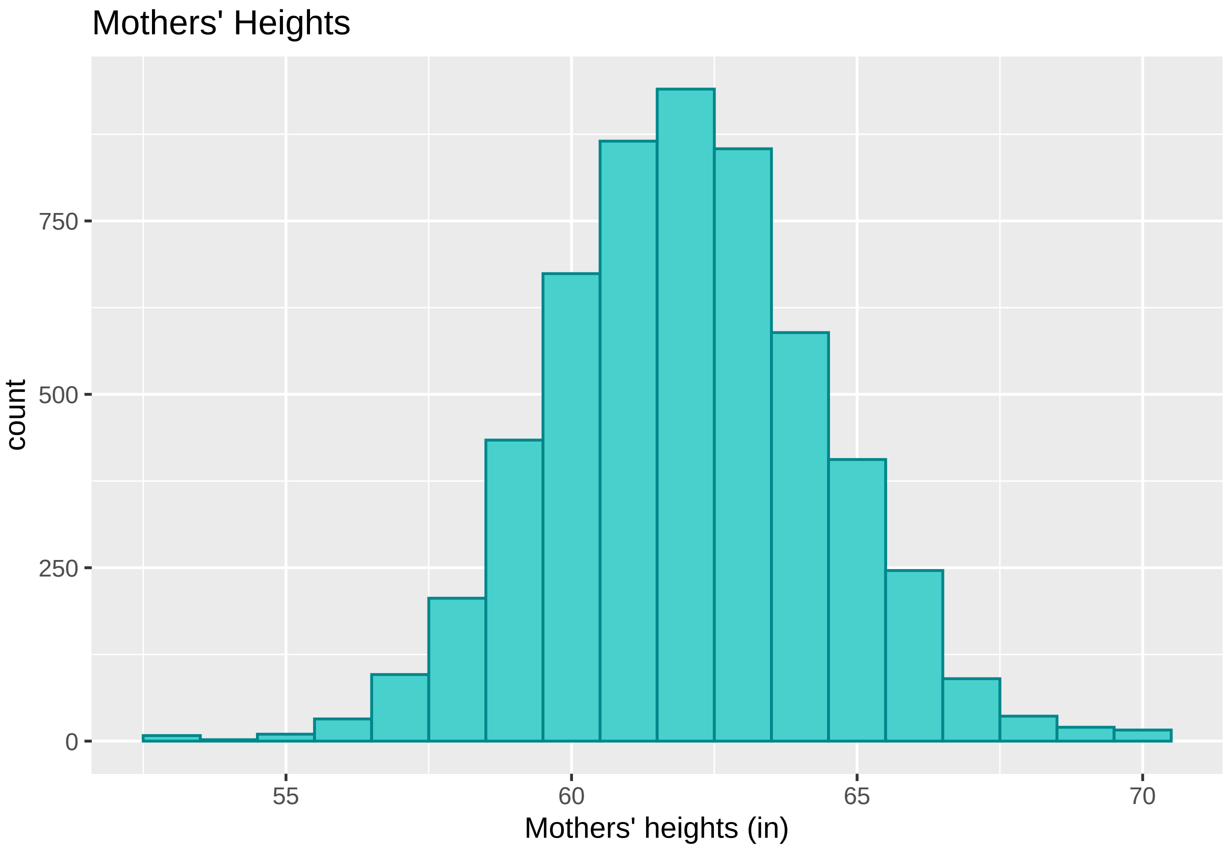
```
heights <- read.table("~/SharedProjects/Kapitula/STA631/ROS/ROSExamples/PearsonLee/data/Heights.txt", h
daughter_height <- heights$daughter_height
```

```
mother_height <- heights$mother_height
n <- length(mother_height)
```

mothers' heights.

Display the distribution of the mother's heights in a histogram, and calculate the mean and standard deviation of this distribution.

```
ggplot(heights, aes(mother_height)) +
  geom_histogram(binwidth = 1,
                 color = "turquoise4", fill = "mediumturquoise") +
  labs(x = "Mothers' heights (in)",
       title = "Mothers' Heights")
```

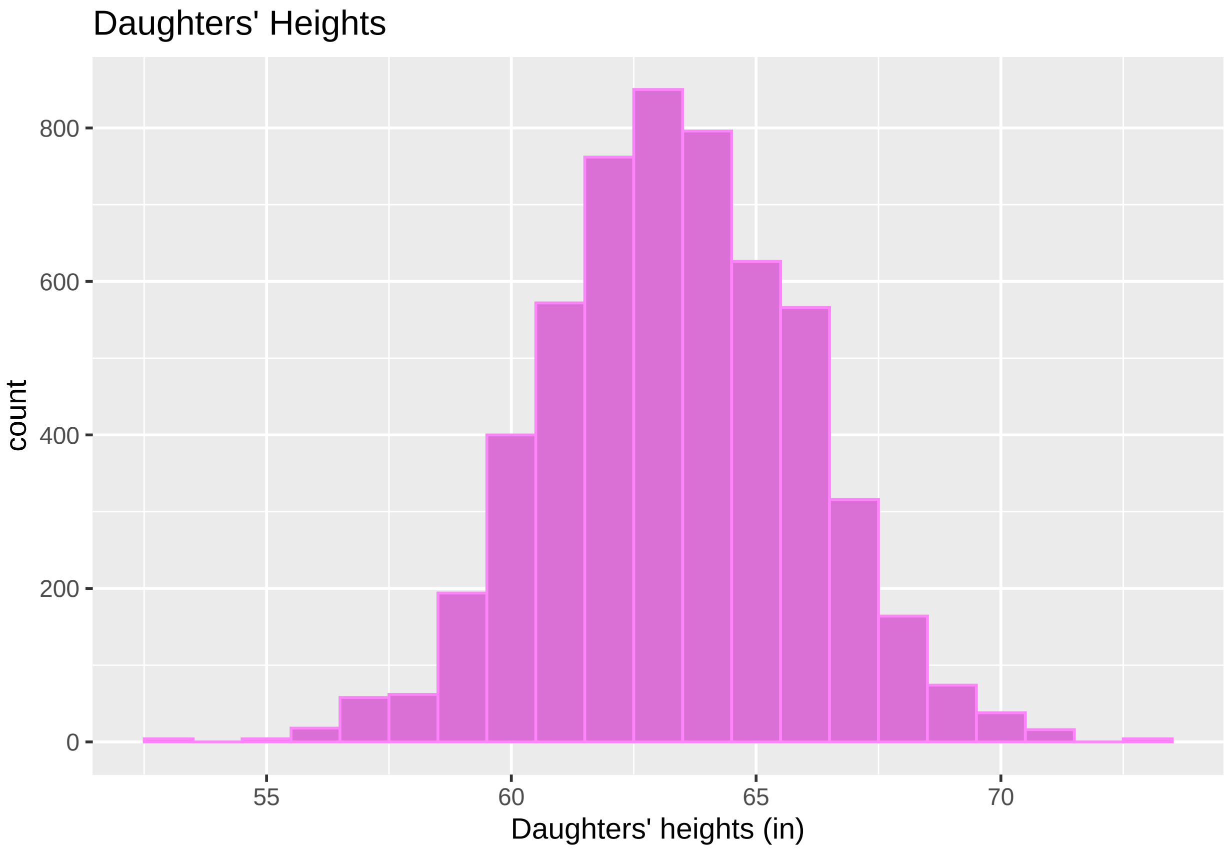


The distribution of the mother's heights has mean 62.499 and standard deviation 2.409.

daughters' heights.

Do the same for the distribution of the daughter's heights.

```
ggplot(heights, aes(daughter_height)) +
  geom_histogram(binwidth = 1,
                 color = "orchid1", fill = "orchid") +
  labs(x = "Daughters' heights (in)",
       title = "Daughters' Heights")
```

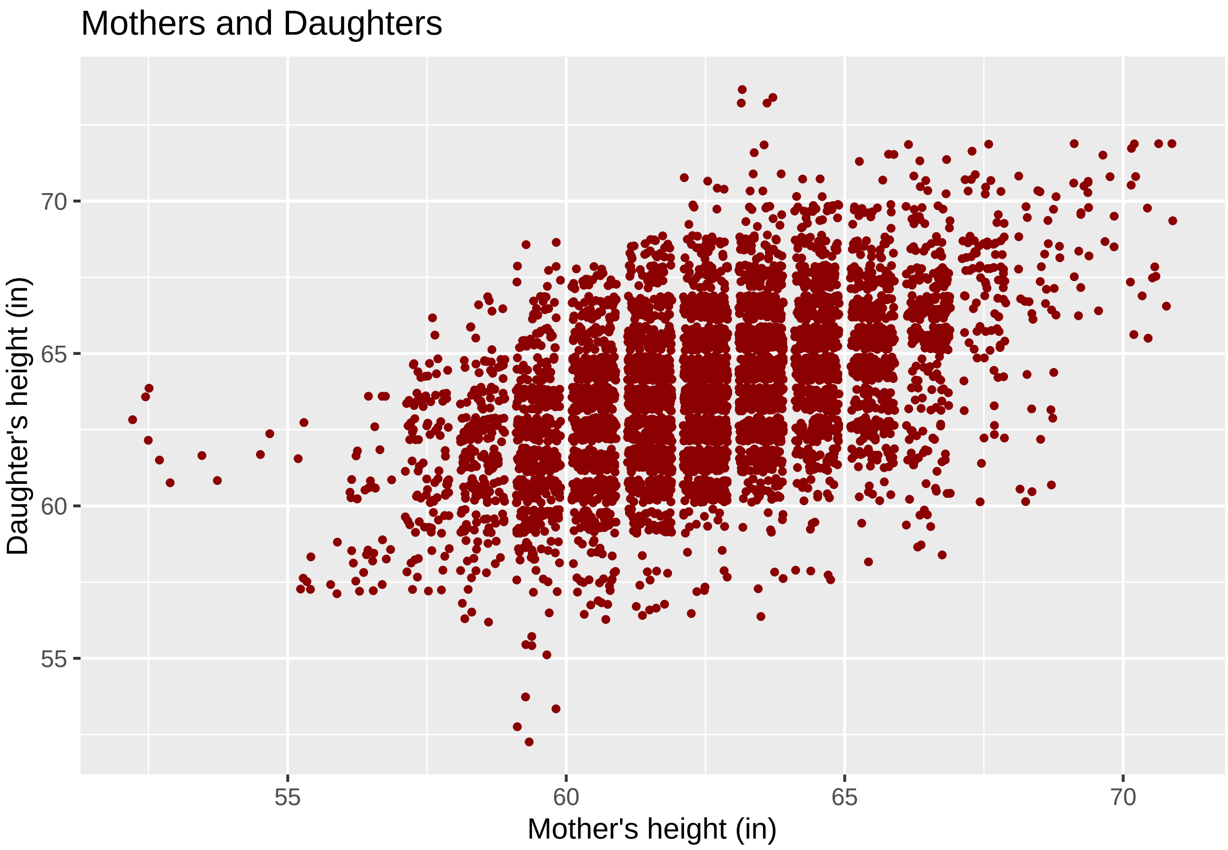


The distribution of the daughter's heights has mean 63.856 and standard deviation 2.615.

scatterplot

Pearson and Lee, 1903, were particularly interested in how these two variables were related.

```
ggplot(heights, aes(mother_height, daughter_height)) +
  geom_jitter(pch = 20, color = "darkred") +
    labs(x = "Mother's height (in)",
      y = "Daughter's height (in)",
      title = "Mothers and Daughters")
```



numerical summary

Distributions such as this one are often described by five numbers :

- the mean and standard deviation of the x data
- the mean and standard deviation of the y data, and
- the correlation of x and y, denoted r

The correlation describes the strength of the linear relationship. For this bivariate distribution, the correlation between the mother's heights and their daughter's heights is 0.502.

You can summarize such distributions in compact tables, such as

family	height	SD	r
mothers	62.5 inches	2.4 inches	
daughters	63.8 inches	2.6 inches	0.49

or

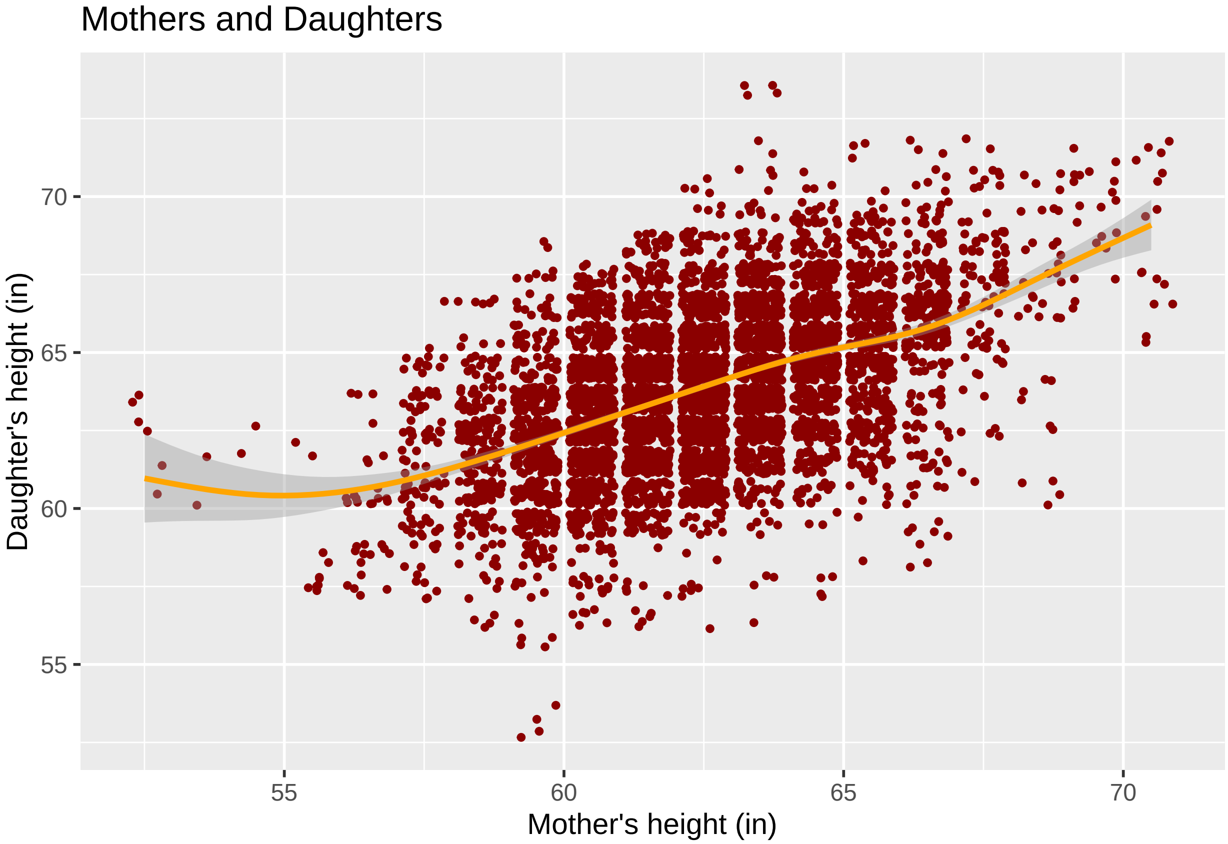
$$\begin{array}{rll} \text{mother's heights} \approx 62.5 \text{ inches} & \text{SD} \approx 2.4 \text{ inches} & \\ \text{daughter's heights} \approx 63.8 \text{ inches} & \text{SD} \approx 2.6 \text{ inches} & r \approx 0.49 \end{array}$$

regression

loess smoother

Loess curves use **local regression** to track the average value of y for each value of x.

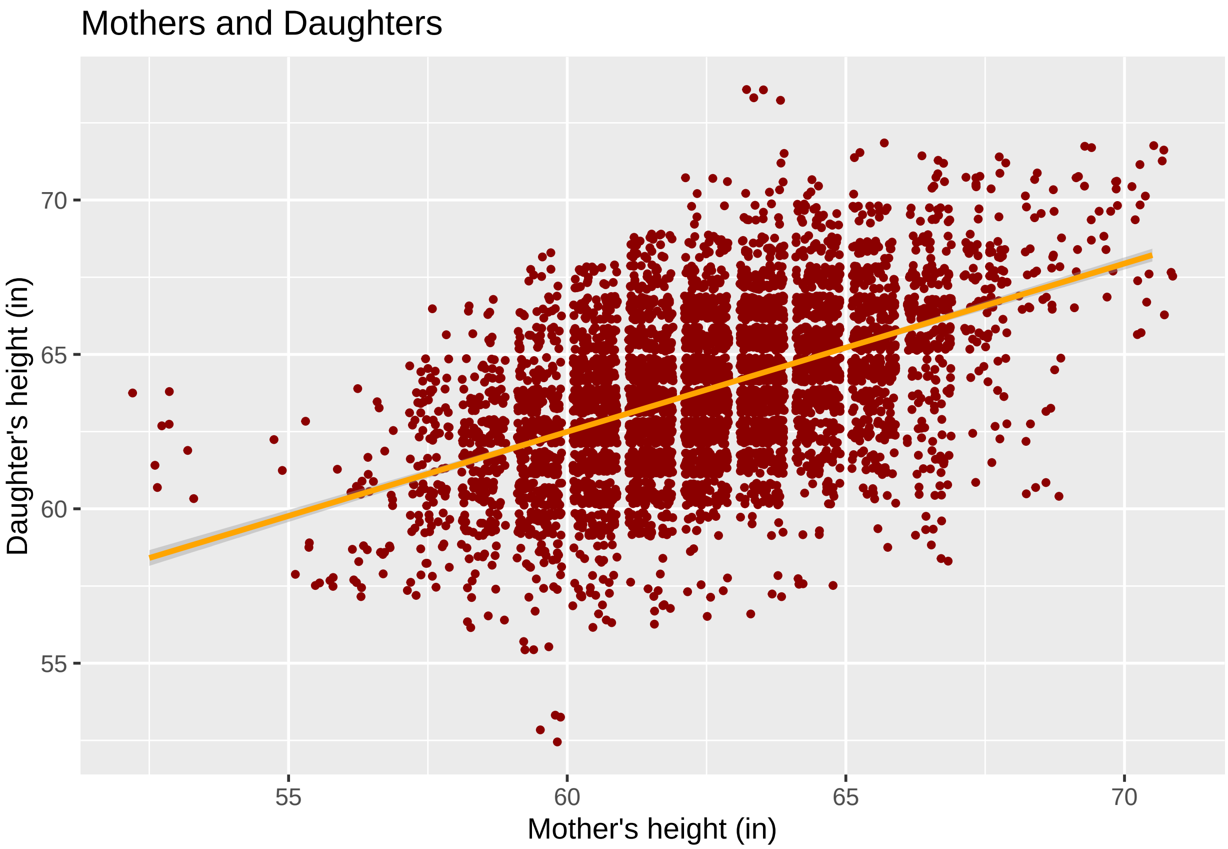
```
ggplot(heights, aes(mother_height, daughter_height)) +
  geom_jitter(pch = 20, color = "darkred") +
  geom_smooth(color = "orange") +
  labs(x = "Mother's height (in)",
       y = "Daughter's height (in)",
       title = "Mothers and Daughters")
```



regression line

The loess curve is fairly close to a straight line, except where there is very little data, suggesting that a **regression line** might fairly represent the relationship between x and y.

```
ggplot(heights, aes(mother_height, daughter_height)) +
  geom_jitter(pch = 20, color = "darkred") +
  geom_smooth(method = "lm", color = "orange") +
  labs(x = "Mother's height (in)",
       y = "Daughter's height (in)",
       title = "Mothers and Daughters")
```



equation of the regression line

Calculate the equation of the regression line.

$$\hat{y} = a + bx =$$

```
heights.lm <- lm(daughter_height ~ mother_height, data = heights)
coefficients(heights.lm)
```

```
##   (Intercept) mother_height
##    29.7984062     0.5449368
```

The intercept is meaningless because it would be the predicted height when mother's height is 0 and that is not possible. Sometimes people rescale the model so results can be seen relative to the mean height of mothers.

Calculate the predicted height of a daughter whose mother's height was the average height of all mothers.

Calculate the predicted height of a daughter whose mother's height was one standard deviation below the mean for the heights of mothers ... and do the same for two standard deviations.

Calculate the predicted height of a daughter whose mother's height was one standard deviation above the mean for the heights of mothers ... and do the same for two standard deviations.

Hint for questions above, another formula for the estimated slope of a regression line b is:

$$b = r(s_y/s_x)$$

Display your results in a table. What do you observe?

Table 2: **Mothers and Daughters**

height	-2s	-s	0	s	2s
mother					
daughter					

Use `predict` to check your numbers.

```r
new.data <- data.frame(mother_height = mean(heights$mother_height) +
                         sd(heights$mother_height) * c(-2, -1, 0, 1, 2))
data.frame(m = new.data,
           daughter_height = predict(heights.lm, new.data))
```

```
##   mother_height daughter_height
## 1      57.68122        61.23103
## 2      60.08998        62.54364
## 3      62.49873        63.85626
## 4      64.90749        65.16888
## 5      67.31624        66.48150
```

The paradox of regression to the mean

regression towards the mean

Pearson and Lee observed the same thing. This phenomenon came to be known as "regression towards the mean," and this is the origin of the modern term 'regression.'

summary

This comes into play in other situations as well. Example, suppose a teacher berates students who score in the bottom 10% on a standardized test, then the teacher has the students take a new version of the test with the same difficulty later, the students do better, so the teacher assumes berating works great. Similarly, this is why it is not always worth it to retake a standardized test if a person scores very well. Example (made up score), say a student gets a 750 out of 800 on a portion of the SAT. If the student takes another test on another day of the same difficulty as the first and there was no learning or practice effect it is expected that their score would go down.