Ch. 15: Regression

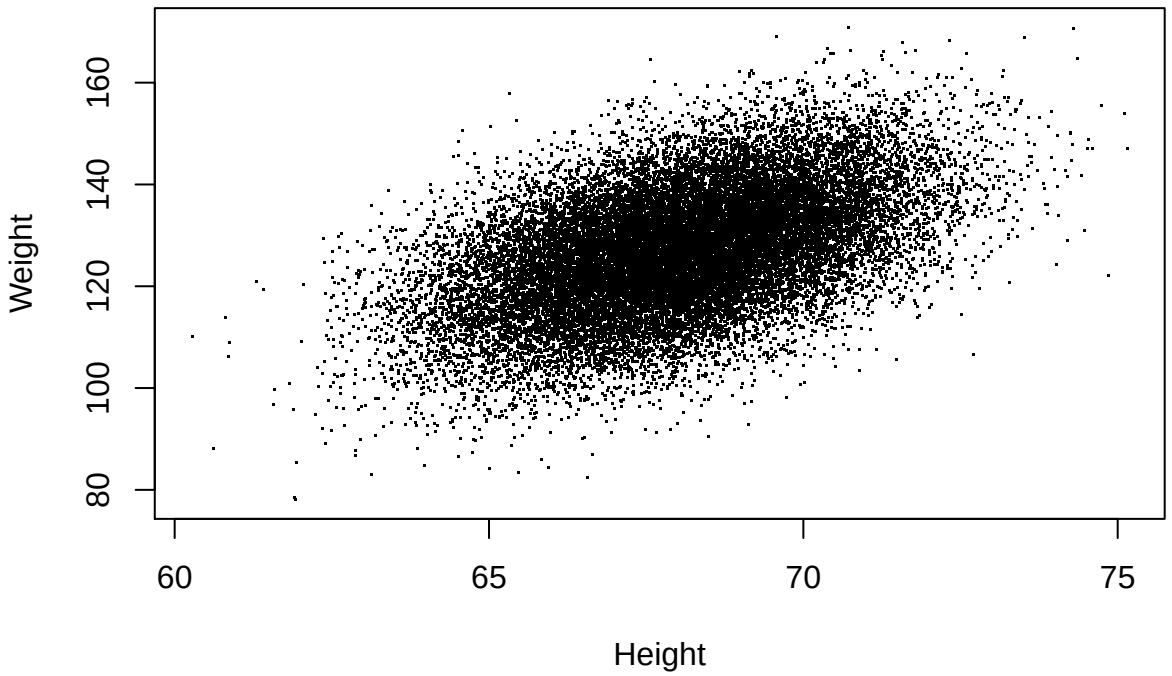
S520

These notes are written to accompany Trosset chapter 15 (omitting 15.3.)

Motivation

Let's study some data from UCLA's statistics wiki:

```
data = read.table("hw-ucla.txt", header=TRUE)
Height = data$Height
Weight = data$Weight
plot(Height, Weight, pch=".")
```



These are reportedly the heights and weights of 25,000 children in Hong Kong. We can look at the summary statistics:

```
summary(data)
```

```
##      Height          Weight
##  Min.   :60.28   Min.   : 78.01
##  1st Qu.:66.70   1st Qu.:119.31
##  Median :68.00   Median :127.16
##  Mean   :67.99   Mean   :127.08
##  3rd Qu.:69.27   3rd Qu.:134.89
##  Max.   :75.15   Max.   :170.92
```

```
sd(Height)
```

```
## [1] 1.901679
```

```
sd(Weight)
```

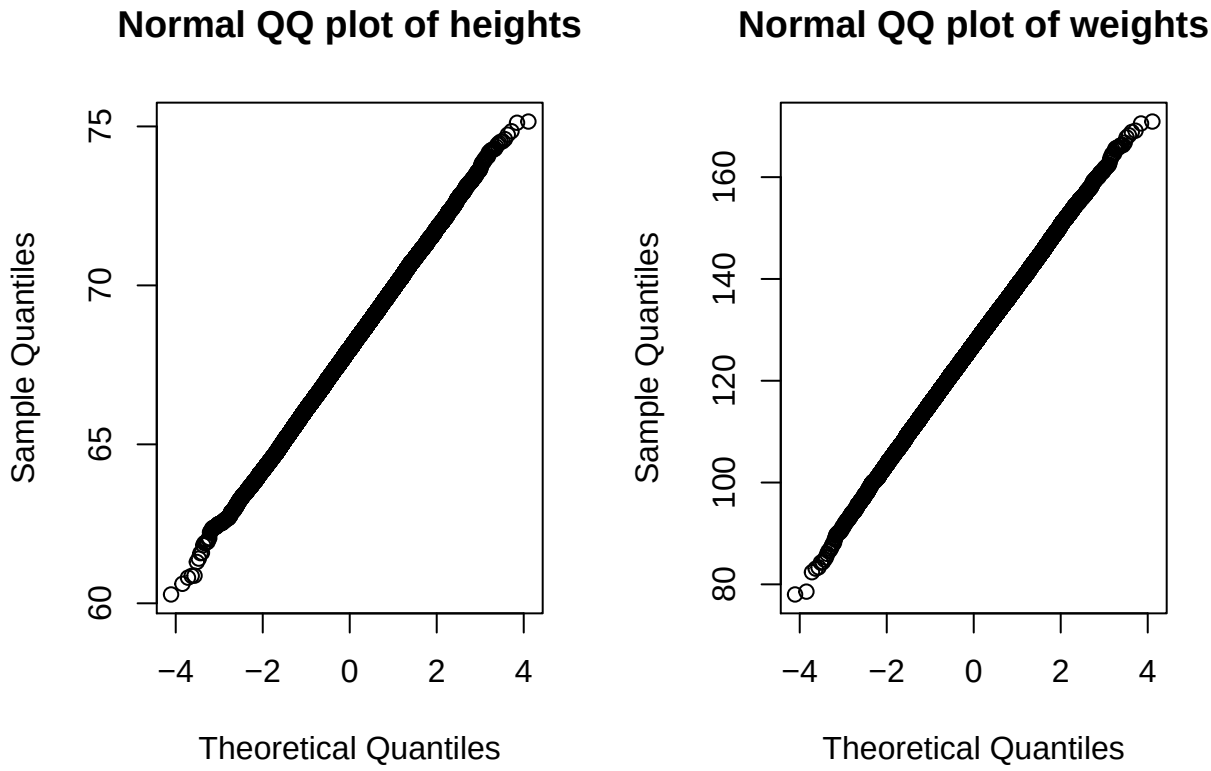
```
## [1] 11.6609
```

```
cor(Height, Weight)
```

```
## [1] 0.5028585
```

Is the data bivariate normal? We draw QQ plots:

```
par(mfrow=c(1,2))
qqnorm(Height, main="Normal QQ plot of heights")
qqnorm(Weight, main="Normal QQ plot of weights")
```



This is almost too good to be true (leading me to suspect that the data is fake, but I digress.) We could also use Trosset's binorm.scatter() function, but by this point I'm satisfied that the data is very close to bivariate normal.

How heavy are six foot kids?

Obviously the answer varies from kid to kid – some tall kids have a thin build, others have a heavier build. But we can find an *average* answer to this question from the data as follows:

- Pick out all the kids who are about six foot, say between 71.5 and 72.5 inches;
- Find their weights;
- Find the average of their weights.

```
sixfoot = which(Height>71.5 & Height<72.5)
mean(Weight[sixfoot])
```

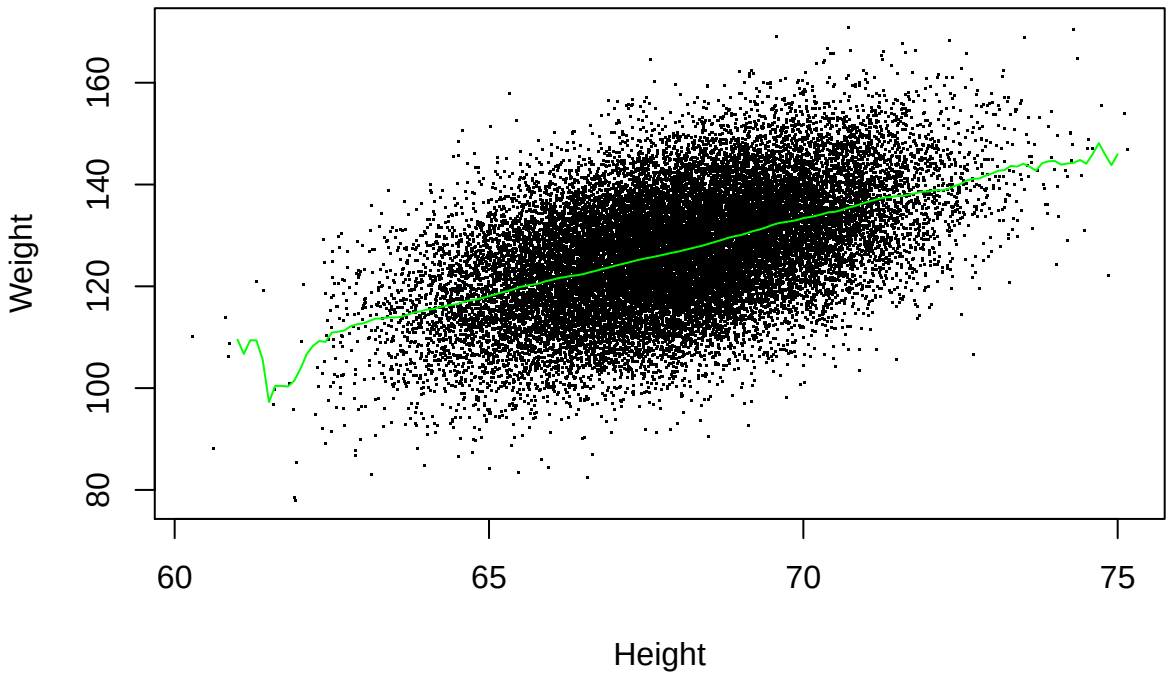
```
## [1] 138.6346
```

We can write a function to make similar predictions for *any* specified height (within reason – we don't have any seven foot kids in the data set, so we shouldn't make predictions for seven foot kids.)

```
predict.weight = function(h){
  kids = which(Height>(h-0.5) & Height<(h+0.5))
  mean(Weight[kids])
}
```

Now let's find these predictions for all heights in our data set.

```
height.list = seq(61, 75, 0.1)
weight.list = rep(NA, length(height.list))
for(J in 1:length(height.list)){
  weight.list[J] = predict.weight(height.list[J])
}
par(mfrow=c(1,1))
plot(Height, Weight, pch=".")
lines(height.list, weight.list, col="green")
```



This gives us pretty good predictions! The only problem is that the green curve gets messy for very short and very tall kids because there isn't much data. We guess that a straight line might give us even prediction than the green curve.

To define a straight line, we need to know:

- The slope;
- At least one point on the line.

Well, in the context of this data, it makes sense to think that kids of average height are (on average) of about average weight.

```
predict.weight(mean(Height))
```

```
## [1] 126.8058
```

```
mean(Weight)
```

```
## [1] 127.0794
```

Close enough. So that gives us one point on the line.

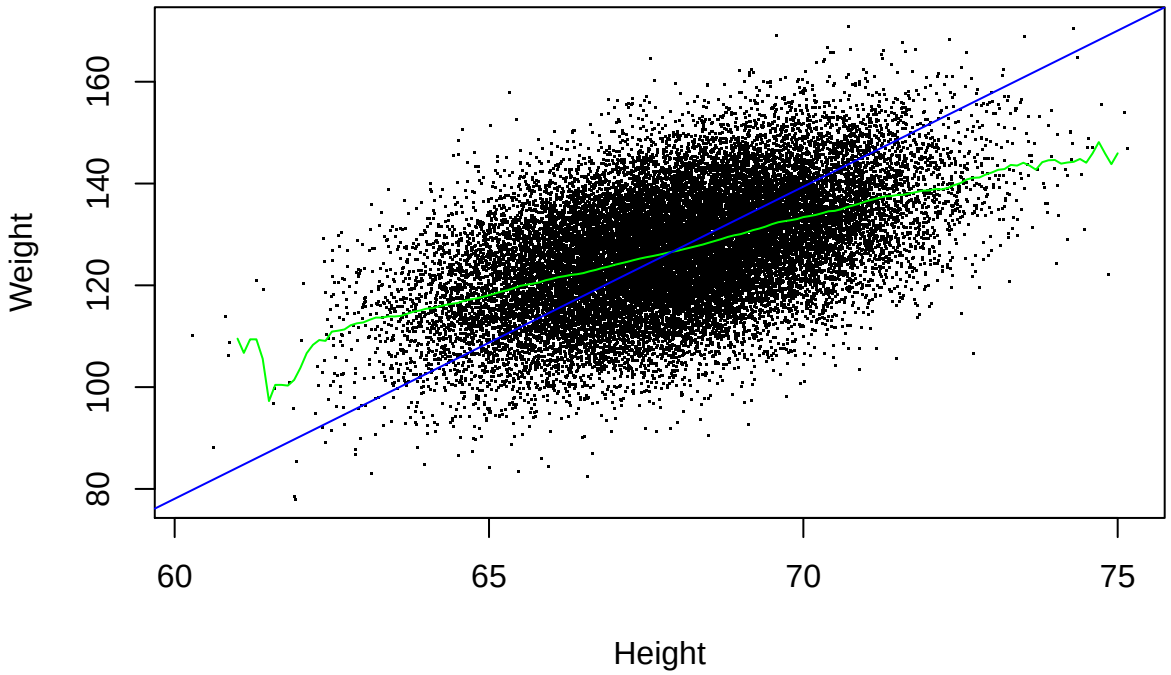
What about the slope? You might suggest the following argument:

- When you go across by one SD in height, you should also go up by one SD in weight.
- Since the slope is rise over run, it should be

SD of weight / SD of height

However, you would be wrong.

```
# abline(a, b) graphs a line with intercept a
# and slope b
b = sd(Weight) / sd(Height) ### THIS IS WRONG
a = mean(Weight) - b * mean(Height)
plot(Height, Weight, pch=".")
lines(height.list, weight.list, col="green")
abline(a, b, col="blue")
```

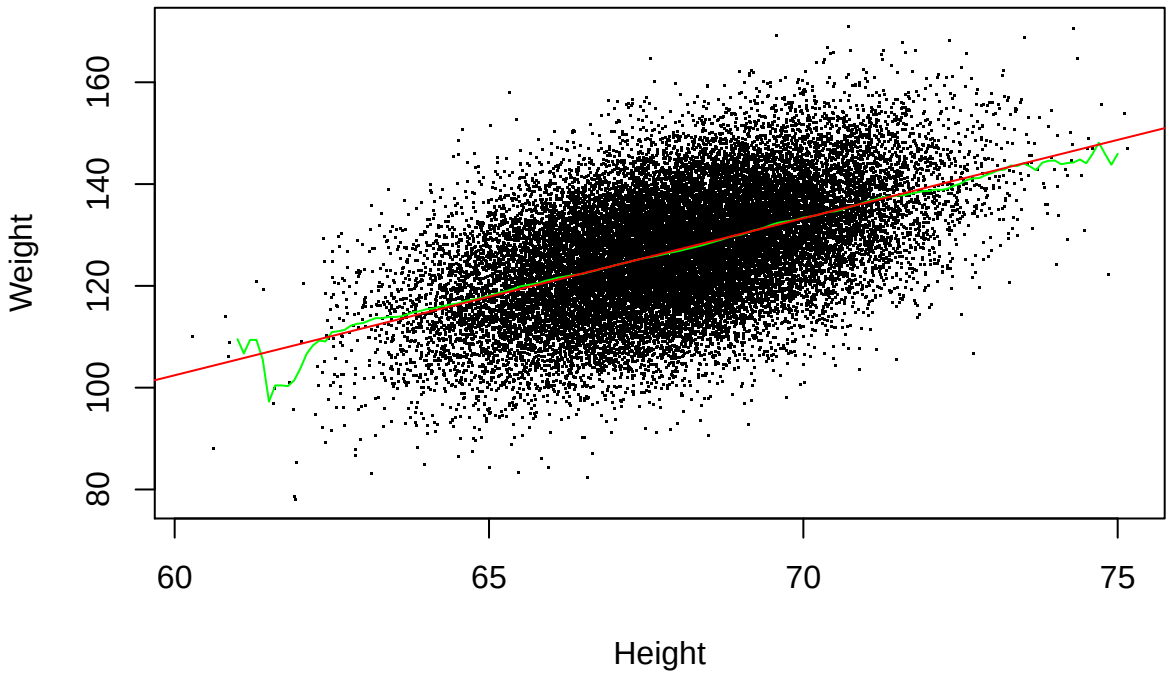


This is a completely different line from the one we found earlier! This is called the **SD line**. It describes the data well but is terrible at prediction.

The correct answer is not at all obvious until you do some calculus or linear algebra. We'll skip that for now and just give you the answer. The correct slope to use is:

Correlation * SD of weight / SD of height

```
b = cor(Height, Weight) * sd(Weight) / sd(Height)
a = mean(Weight) - b * mean(Height)
plot(Height, Weight, pch=".")
lines(height.list, weight.list, col="green")
abline(a, b, col="red")
```



The intuition behind this is that kids who are 99th percentile in height, say, are generally *not* 99th percentile in weight – usually (not always) they're a bit shorter. This is called **regression to the mean** or the **regression effect**.

Regression to the mean

What height corresponds to the 95th percentile?

We can answer this empirically:

```
quantile(Height, 0.95)
```

```
##     95%
## 71.1102
```

What weight corresponds to the 95th percentile?

```
quantile(Weight, 0.95)
```

```
##      95%
## 146.1938
```

Okay, let's now consider people who are at the 95th percentile in height. What does the regression line predict for their height?

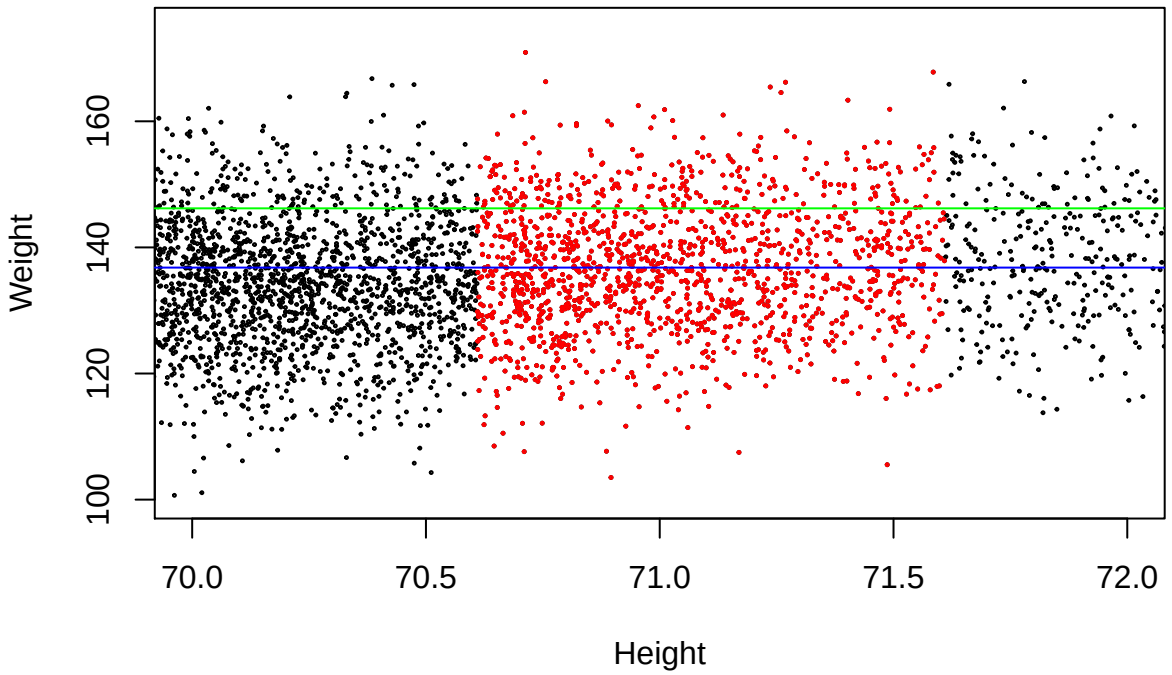
```
h = quantile(Height, 0.95)
a + b * h
```

```
##      95%
## 136.6909
```

This is much less than the 95th percentile of weight! In fact, it's more like the 80th percentile. So we're saying that very tall kids are predicted to only be somewhat heavy.

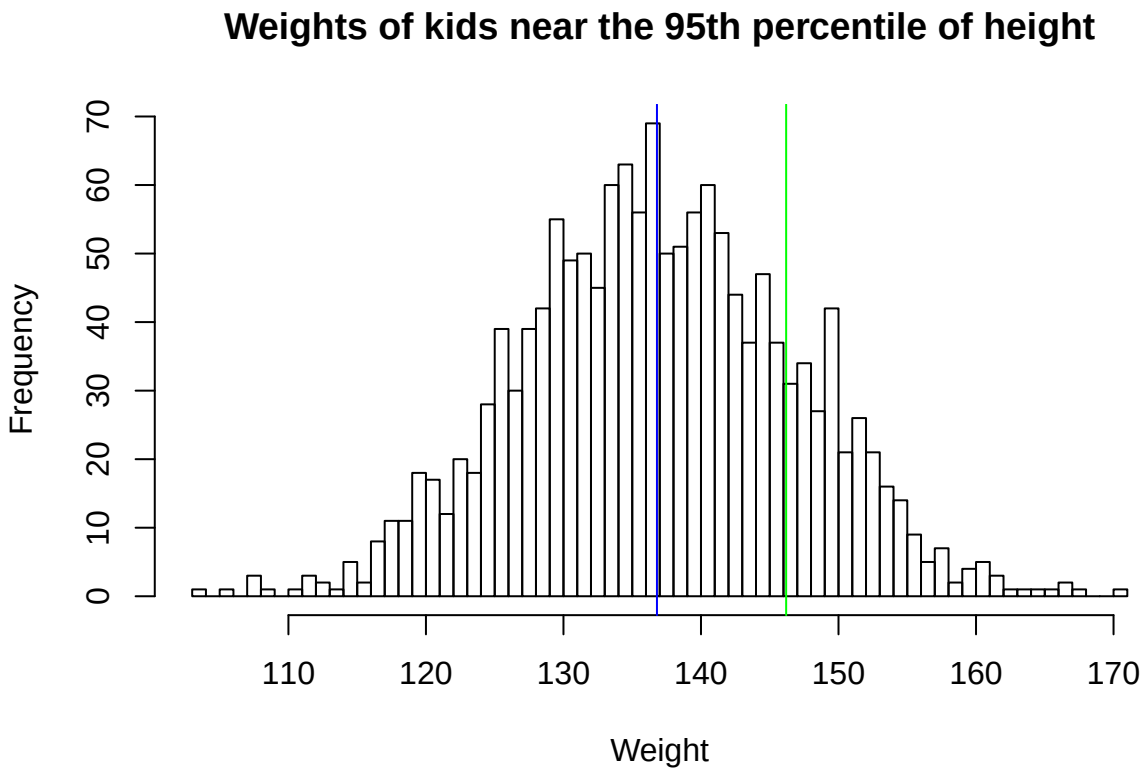
Again, this is regression to the mean. We can see what's happening by coloring the points on the scatterplot corresponding to kids around the 95th percentile of height.

```
tallkids = which(Height>(h-0.5) & Height<(h+0.5))
plot(Height, Weight, cex=0.2, xlim=c(70,72), ylim=c(100,175))
points(Height[tallkids], Weight[tallkids], cex=0.2, col="red")
abline(h=quantile(Weight, 0.95), col="green")
abline(h=quantile(Weight, 0.8), col="blue")
```



The green line is the 95th percentile of weights, while the blue is the 80th percentile of weights. The green is higher than most of the red dots, while the blue line goes approximately through the middle. This is even more apparent looking at a histogram:

```
hist(Weight[tallkids],
     main="Weights of kids near the 95th percentile of height",
     xlab="Weight",
     breaks=103:171)
abline(v=quantile(Weight, 0.95), col="green")
abline(v=quantile(Weight, 0.8), col="blue")
```



Why are kids most kids near the 95th percentile of heights lower than the 95th percentile on weights? The answer is simply that in general, *not many* kids are above the 95th percentile in weight (otherwise it wouldn't be the 95th percentile.) If you pick a kid at random, there's only a 5% chance they're above the 95th weight percentile. If you pick a tall kid, this probability goes up (because there's a positive correlation between height and weight), but only to about 15 to 20%, not to 95% (because the relationship is far from perfect.)

In general, most people – perhaps even most statisticians – have difficulty recognizing the regression effect. Consider the following parable.

When children at a certain school begin a grade, they're given an aptitude test. The top 20% go to class A, the next 20% to class B, and so on down to the bottom 20%, who go to class E. At the end of the year, they're given a similar aptitude test. Suppose the Board of Education wants to measure teacher performance by looking at the average change (second test minus first test.) They get the following results:

- Class A: Average change: -2.15
- Class B: Average change: 3.45
- Class C: Average change: 7.6
- Class D: Average change: 14.5
- Class E: Average change: 21.9

This data could lead to all kinds of arguments. "Class A's scores actually dropped! The teacher must be fired!" "No, it's the students fault! They got overconfident and slacked off during the year!" The point is no causal explanations are necessary – it could just be regression. Scores on different tests are never perfectly correlated, so student who do very well on the first test will tend to be closer to the mean on the second – maybe they were just lucky the first time. Similarly, students who do poorly on the first time will generally do better the second time, perhaps because the got out o the wrong side on bed on the first morning.

The regression line

Correlation summarizes the linear association between variables X and Y.

If X and Y are linearly associated, they have the relationship

Y = slope * X + intercept

for some slope and intercept.

Suppose we know X. To **predict** Y from X, we need to:

- Estimate the intercept and slope from data. Call the estimates a^* and b^*.
- Call the predictions $\hat{y}$, where

$$\hat{Y} = a^* + b^* X$$

Linear regression gives a method for getting the "best" estimates.

$$b^* = r\frac{s_y}{s_x}$$

$$a^* = \bar{y} - b^*\bar{x}$$

where:

- r is the (Pearson's product-moment) correlation;
- s_x and s_y are the sample SDs of x and y;
- $\bar{x}$ and $\bar{y}$ are the sample means of x and y.

Later we'll see *why* this works.

For now, we'll see *that* it works.

Examples

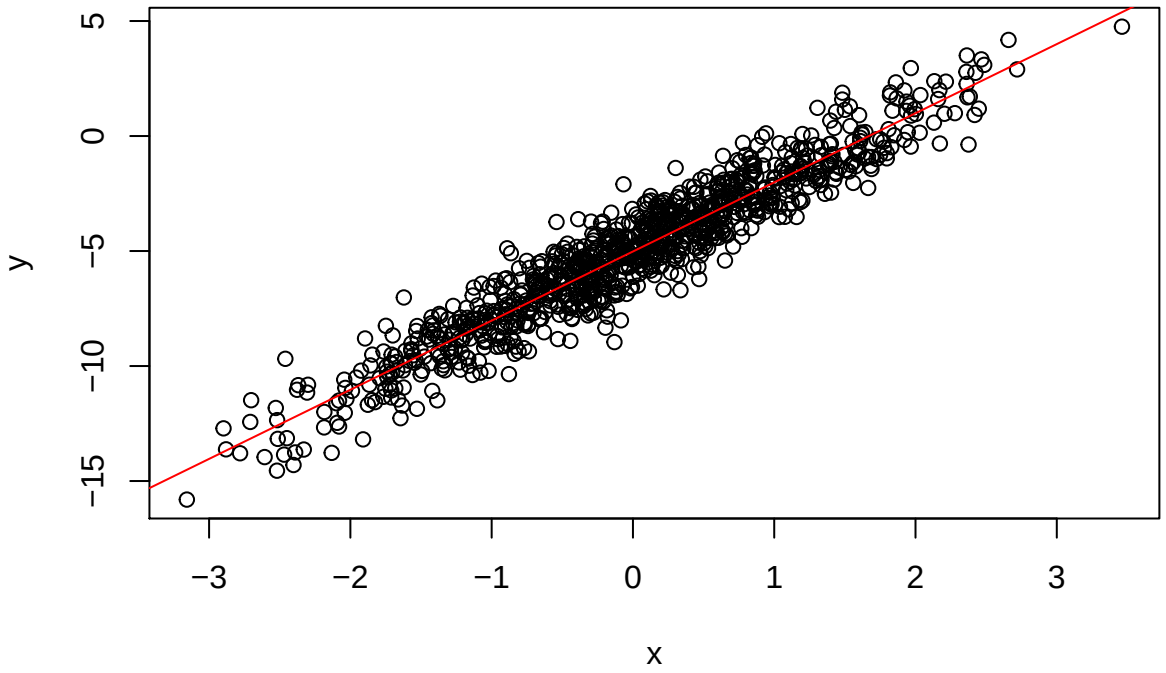
```
x = rnorm(1000)
y = - 5 + 3 * x + rnorm(1000)
# Intercept = 3, slope = -5: can we reproduce these?
r = cor(x, y)
print(r)
```

```
## [1] 0.9520533
```

```
b = r * sd(y) / sd(x)
a = mean(y) - b * mean(x)
print(c(a,b))
```

```
## [1] -5.02011  3.00583
```

```
# Compare the regression line to the data:
plot(x, y)
abline(a, b, col="red")
```



That example was pretty easy, as the position of the straight line was pretty clear. Let's try a harder one.

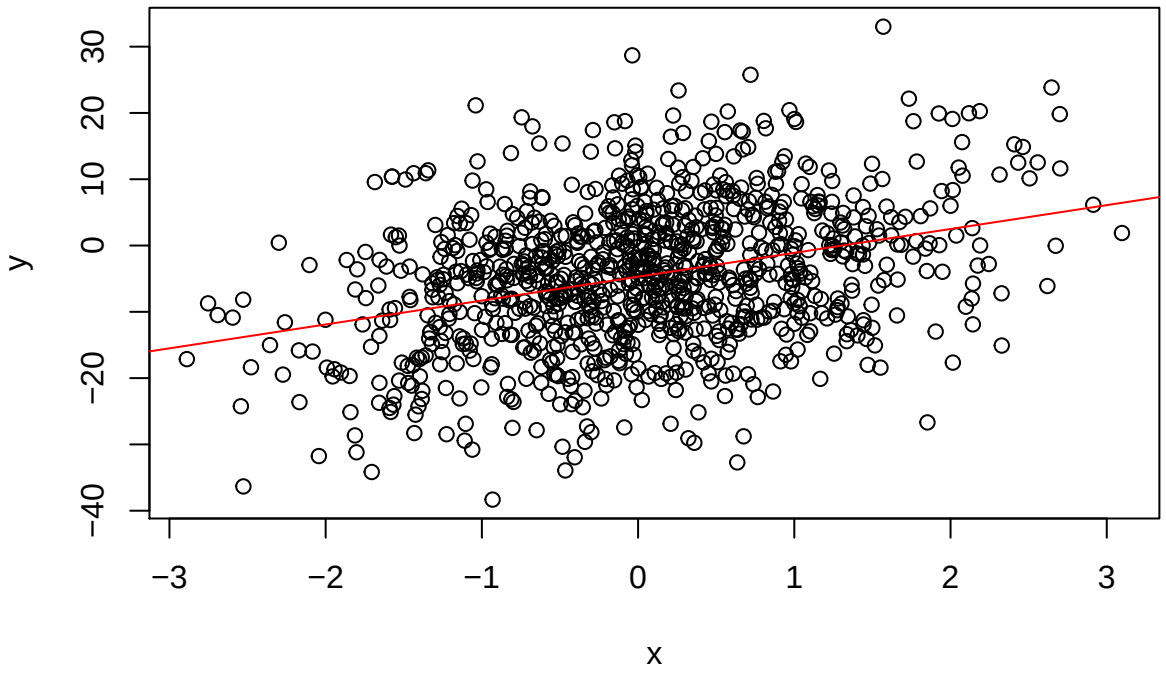
```
x = rnorm(1000)
y = - 5 + 3 * x + rnorm(1000) * 10 # more error
r = cor(x, y)
print(r)
```

```
## [1] 0.3344711
```

```
b = r * sd(y) / sd(x)
a = mean(y) - b * mean(x)
print(c(a, b))
```

```
## [1] -4.706558  3.600124
```

```
plot(x, y)
abline(a, b, col="red")
```



How good are the predictions?

```
errors = y - (a + b * x) # "residuals"
# Proportion of variance "unexplained"
var(errors) / var(y)
```

```
## [1] 0.8881291
```

```
# Proportion of variance "explained"
1 - var(errors) / var(y)
```

```
## [1] 0.1118709
```

```
r^2
```

```
## [1] 0.1118709
```

The regression formulae

The regression line to predict y from x goes through $(\bar{x}, \bar{y})$ and has slope $r \cdot s_y/s_x$. Where does this come from mathematically?

Our predictions are of the form

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

We want to find values of a and b that minimize the **error sum of squares**:

$$\begin{aligned} SS_E &= \sum (y - \hat{y})^2 \\ &= \sum (y - a - bx)^2 \end{aligned}$$

To minimize something, differentiate and set to zero.

$$\frac{\partial}{\partial a} SS_E = \sum -2(y - a - bx) = 0$$
$$\sum (y - a - bx) = 0$$
$$\sum y = \sum (a + bx)$$
$$\frac{\sum y}{n} = \frac{\sum (a + bx)}{n}$$
$$\bar{y} = a + b\bar{x}$$

Substituting $\bar{x}$ into the regression line gives $\bar{y}$. In other words, the regression line goes through the point of averages.

Similarly,

$$\frac{\partial}{\partial b} SS_E = \sum -2x(y - a - bx) = 0$$

After substituting in our equation for a above and doing quite a lot of algebra, we can make b the subject.

$$b = \frac{\sum xy - n\bar{x}\bar{y}}{\sum x^2 - n\bar{x}^2}$$
$$= \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

I'll leave it as an exercise to verify this is the same thing as $r \cdot s_y / s_x$.

Using the bivariate normal for probability calculations

Recall that we use pnorm() to calculate normal distribution probabilities.

e.g. Suppose X is Normal with mean 5 and SD 2. What's P(X < 3)?

```
pnorm(3, mean=5, sd=2)
```

```
## [1] 0.1586553
```

With bivariate normal data, we might want to answer questions like:

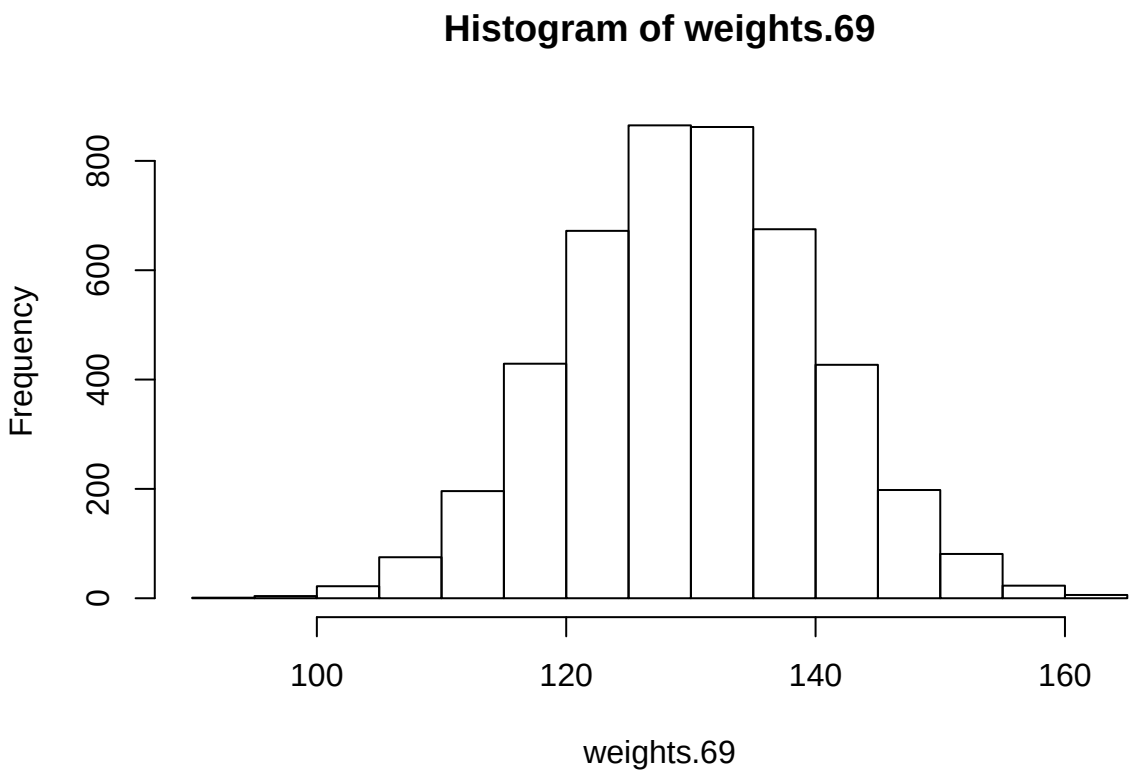
"If a Hong Kong kid is 69 inches tall, what's the probability they weigh less than 140 pounds?"

Can we use the normal distribution? Looking at the original data, we pick out the weights of kids who are about 69 inches tall.

```
kids.69 = which(Height>68.5 & Height<69.5)
weights.69 = Weight[kids.69]
```

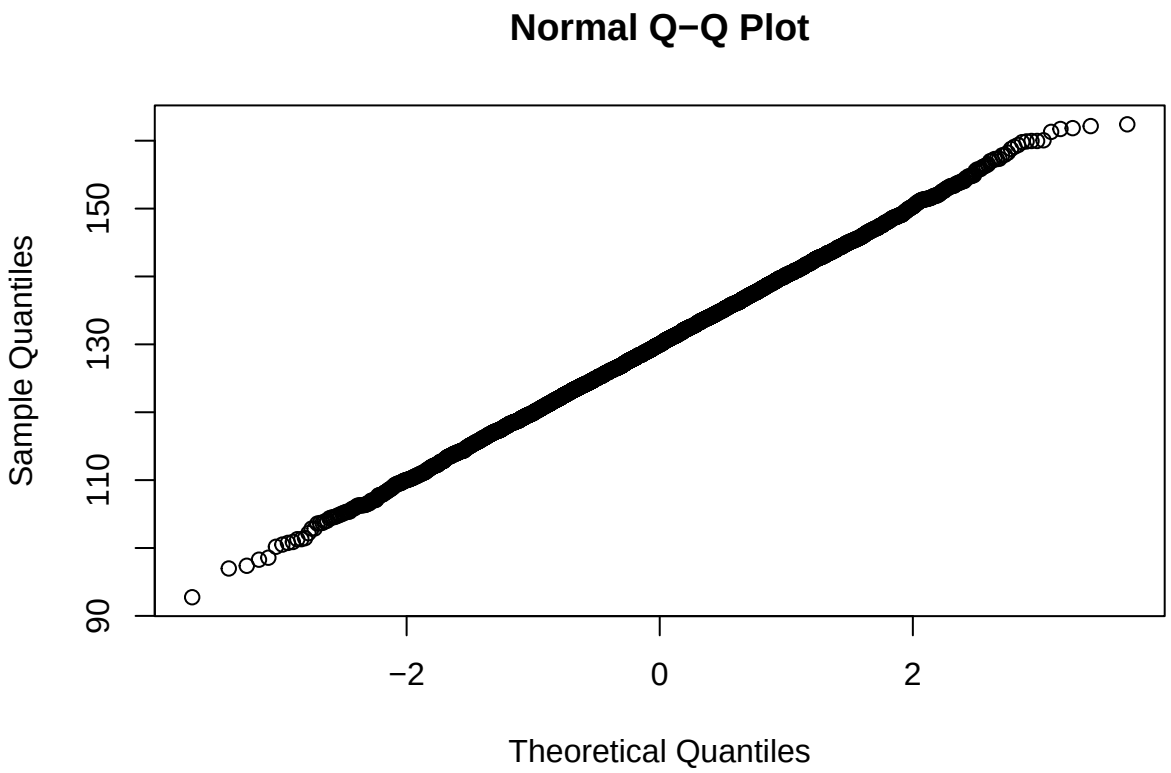
First we draw a histogram:

```
hist(weights.69)
```



and then a QQ plot:

```
qqnorm(weights.69)
```



So the weights of kids that are 69 inches tall adhere very closely to a normal distribution. This is a general fact about the bivariate normal.

Suppose X and Y are bivariate normal. Then given a specific value of X, the values of Y follow a normal distribution.

We say that Y has a *conditional normal distribution given X*. We're using "conditional" in the same sense here as when we were talking about conditional probability.

To use the normal distribution, we need a mean and variance. However, we **can't** just use the mean and variance of Y. Knowing X changes the distribution – the weights of 69-inch tall kids don't have the same distribution as the weights of kids in general. For one thing, 69-inch tall kids are taller than average, so they should typically be a bit heavier than avergae. For another, the *spread* of their weights should be lower than for the population as a whole. This is the point of regression predictions: Knowing height lets you predict weight a little bit more accurately because the spread is reduced. The spread of weights given a height can then be thought of as *prediction error*.

What's the new mean? This is exactly what the regression line tells us.

```
slope = cor(Height, Weight) * sd(Weight) / sd(Height)
intercept = mean(Weight) - slope * mean(Height)
predict.69 = intercept + 69 * slope
print(predict.69)
```

```
## [1] 130.1841
```

What's the prediction error? This isn't obvious, so let's just tell you the answer(s):

- If you have the population values of the means, SDs, and correlation, the prediction error is
$$\sigma_Y \sqrt{1-\rho^2}$$
where ρ is the population correlation.
- If you have sample values of the means, SDs, and correlation, and a large sample, a good approximation for the prediction error is
$$s_Y \sqrt{1-r^2}$$
where r is the sample correlation.
- A marginally more accurate approximation for the prediction error is
$$\sqrt{\frac{SS_E}{n-2}}$$
but the difference between this and the previous formula will be negligible unless your sample size is small.

Example. If a Hong Kong kid is 69 inches tall, what's the probability they weigh less than 140 pounds?

1. Get the regression prediction:

```
predict.69 = intercept + 69 * slope
```

2. Estimate the prediction error. We have a sample of size 25,000, which is large, so we can do this the easy way.

```
r = cor(Height, Weight)
pred.error = sd(Weight) * sqrt(1 - r^2)
```

3. Use the normal distribution:

```
pnorm(140, mean=predict.69, sd=pred.error)
```

```
## [1] 0.8349376
```

The probability is 83.5%. We can check this is sensible empirically:

```
mean(weights.69 < 140)
```

```
## [1] 0.837963
```

Close enough. Note that as with many techniques based on the normal, this may fail at the tails of the distribution (e.g. for 75-inch tall kids.)

Variance explained

If we just made predictions based on the mean, our squared prediction error would be s_Y^2. Thanks to the regression model, we've reduced the squared prediction error to $s_Y^2(1-r^2)$. In other words, the model has reduced squared prediction error by a proportion r^2. We call r^2 the *coefficient of determination* or the *proportion of variance explained*. This terminology has never made much sense to me (it's not clear that the model is necessarily explaining anything) but some people like it and it is widely used.

Regression tests and confidence intervals

The two major types of regression test are: * t-tests for the coefficient; * F-tests of the whole model.

For more complicated regression models, these two tests are testing different hypotheses, so may give different results. For simple linear regression (one predictor), they're equivalent: Both do a two-tailed test of the null that the slope is 0. We'll just do the t-test.

Regression t-test

The model we use is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

The ϵ_i terms are error terms. We assume they follow a normal distribution.

Let β_1 be the true slope and $\hat{\beta}_1$ be the estimated slope. The standard error of $\hat{\beta}_1$ is

$$se(\hat{\beta}_1) = \frac{s_y}{s_x}\sqrt{\frac{1-r^2}{n-2}}$$

where r is the correlation and n is the sample size.

Let

$$T = \frac{\hat{\beta}_1 - \beta_1}{se(\hat{\beta}_1)}$$

Then T has a t-distribution with $n-2$ degrees of freedom. Thus we can use pt to find t-test P-values.

A 95% confidence interval for β_1 is:

$$\hat{\beta}_1 \pm qt(0.975, df = n-2) * se(\hat{\beta}_1)$$

We can do this the long way, but at this stage of the semester let's skip to the shortcut.

```
model = lm(Weight ~ Height)
summary(model)
```

```
##
## Call:
## lm(formula = Weight ~ Height)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.302  -6.711  -0.052   6.814  39.093
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -82.57574    2.28022  -36.21   <2e-16 ***
## Height        3.08348    0.03352   91.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.08 on 24998 degrees of freedom
## Multiple R-squared:  0.2529, Adjusted R-squared:  0.2528
## F-statistic:  8461 on 1 and 24998 DF,  p-value: < 2.2e-16
```

The P-value for "Height" is minuscule. Yes, using height gives us better predictions for weight (obviously.)

WARNING. The inference here requires the model to be very very very close to true. If the model is not literally true, you can still fit a regression line, but the inference and interpretation will be different. In this case, ask a statistician to teach you how to do the paired-data bootstrap.