Practicing Regression with Billionaires and My Cat

Shay O'Brien

October 11, 2018

Figure 1: Arugula O'Brien

1. Lists in R

Lists are a kind of object in R. They're vectors of other objects. You can store all kinds of stuff in them- numbers, booleans, strings, lists, etc. We haven't used them much until now, but the linear regression function in R returns a list. You're going to need to know how to get information out of it.

```
## Create a list
cool <- c(0, exp(1), 1, pi)
pet <- c("cat", "arugula", "hairless", "gorgeous", "perfect")
tf <- c(TRUE, TRUE, TRUE)
bday <- as.Date("2011-08-02")
```

```
## Warning in strptime(xx, f <- "%Y-%m-%d", tz = "GMT"): unknown timezone
## 'default/America/New_York'
```

```
mylist <- list(cool, pet, tf, bday, 824)
names(mylist) <- c("Cool_Numbers", "My_Cat", "Two_truths_and_a_lie", "Birthday", "Anniversary")

mylist
```

```
## $Cool_Numbers
## [1] 0.000000 2.718282 1.000000 3.141593
##
## $My_Cat
## [1] "cat"      "arugula"  "hairless" "gorgeous" "perfect"
##
## $Two_truths_and_a_lie
## [1] TRUE TRUE TRUE
##
## $Birthday
## [1] "2011-08-02"
##
## $Anniversary
## [1] 824
```

There are lots of ways to access the cool stuff you've stored in your list. You can excerpt portions of the list either as a tinier list or as an object of whatever form best matches the things inside it. It's important to think about the form you need for whatever you're doing so that you don't get a gazillion stressful errors.

Accessing stuff in list form: We can think of this as taking slices of the list.

```
mylist[1]
```

```
## $Cool_Numbers
## [1] 0.000000 2.718282 1.000000 3.141593
```

```
mylist[2:3]
```

```
## $My_Cat
## [1] "cat"      "arugula"  "hairless" "gorgeous" "perfect"
##
## $Two_truths_and_a_lie
## [1] TRUE TRUE TRUE
```

```
mylist[c(2, 4)]
```

```
## $My_Cat
## [1] "cat"      "arugula"  "hairless" "gorgeous" "perfect"
##
## $Birthday
## [1] "2011-08-02"
```

Accessing stuff in other object types: Second, you can pull out pieces in the object-type that best matches those pieces (character, numeric, etc.)

```
mylist[[1]] ##the double bracket gets rid of the list name
```

```
## [1] 0.000000 2.718282 1.000000 3.141593
```

```
mylist$Two_truths_and_a_lie
```

```
## [1] TRUE TRUE TRUE
```

```
mylist[[2]][1]
```

```
## [1] "cat"
```

```
mylist$My_Cat[4]
```

```
## [1] "gorgeous"
```

Fun!!

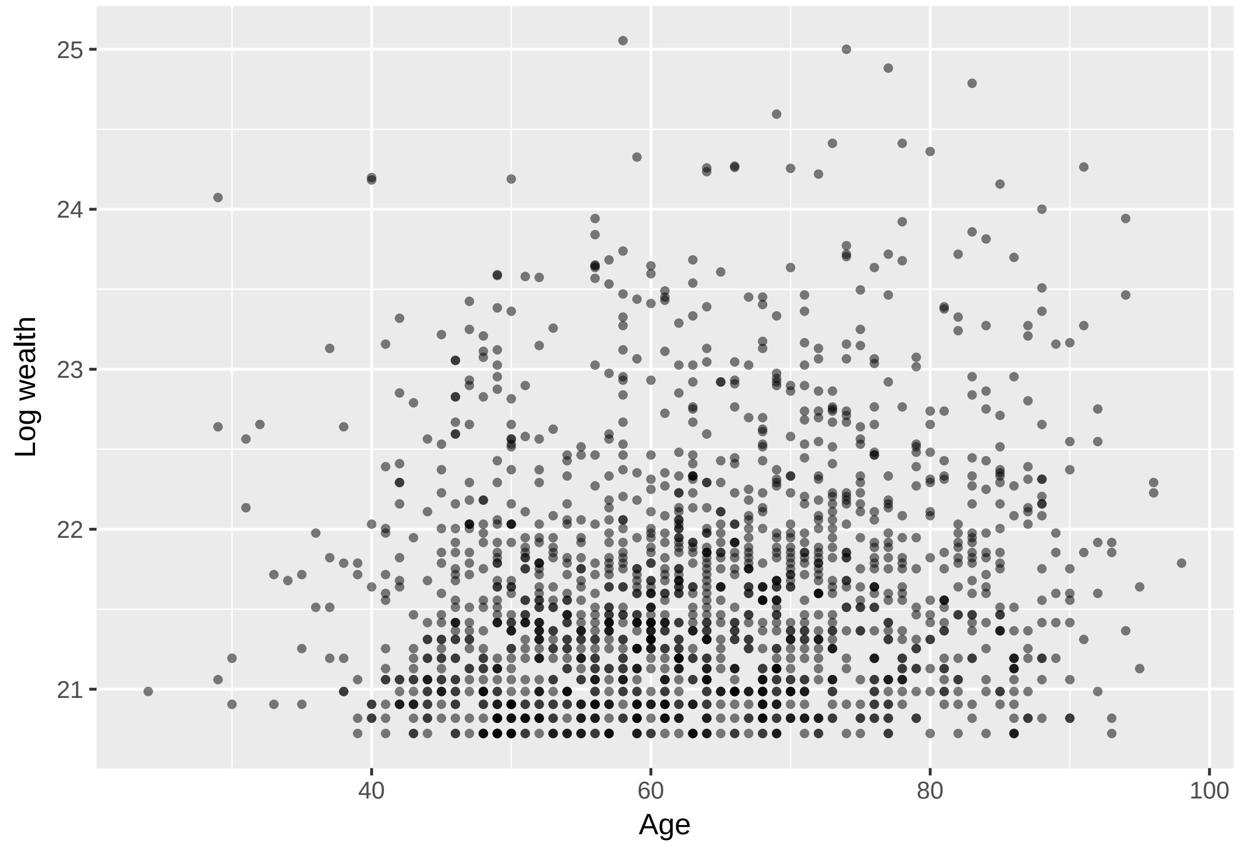
2. Simple OLS

Billionaires

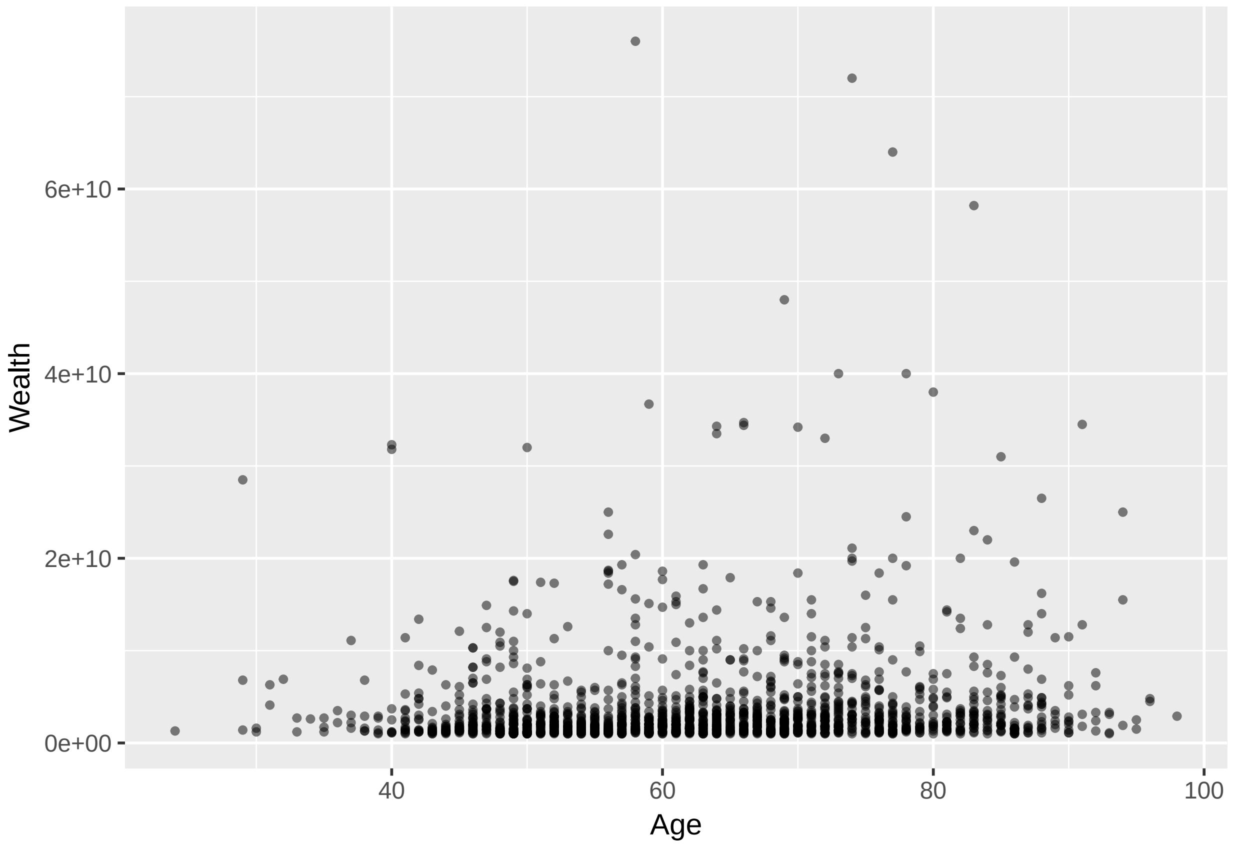
First, we'll get the data ready and look at it.

```
## Cleaning and simplifying data
bills <- billionaires %>%
  filter(year == 2014, !is.na(age), !is.na(networthusbillion)) %>%
  select(year, name, rank, citizenship, networthusbillion,
         selfmade, typeofwealth, gender, age, foundingdate) %>%
  mutate(wealth = networthusbillion*1000000000,
          logwealth = log(wealth))

##Let's look at it
#View(bills)
#summary(bills)
```

```
## Create a simple plot
ggplot(data = bills, aes(x = age, y = logwealth)) +
  geom_point(size = 1, alpha = .5) +
  xlab("Age") +
  ylab("Log wealth")
```



```r
##What does it look like if we plot wealth instead of the log of wealth?
ggplot(data = bills, aes(x = age, y = wealth)) +
  geom_point(size = 1, alpha = .5) +
  xlab("Age") +
  ylab("Wealth")
```



Hand code OLS estimators

```r
#OLS slope: sample covariance between X & Y / sample variance of X
b1 <-
#OLS intercept: sample mean of Y - (sample mean of X * OLS slope)
b0 <-
```

Now we'll run a linear regression and look at the results.

```r
## Regress logwealth on age and save the results
bilm <- lm(data = bills, logwealth ~ age)

## Looking at the linear fit list
# summary(bilm)
# names(bilm)

## Ways to extract the coefficients
coef(bilm)
```

```
##   (Intercept)           age
## 21.145690787  0.008401891
```

```
bilm$coefficients
```

```
##   (Intercept)           age
## 21.145690787  0.008401891
```

```
summary(bilm)$coefficients
```

```
##                   Estimate  Std. Error    t value    Pr(>|t|)
## (Intercept) 21.145690787 0.095810859 220.702445 0.00000e+00
## age          0.008401891 0.001481091   5.672773 1.66668e-08
```

```
## Storing the coefficients
b0 <- bilm$coefficients[[1]]
b1 <- bilm$coefficients[[2]]
```

We can print the results really nicely with a package called "stargazer."

```
stargazer(bilm, title = "Log wealth of billionaires as a function of age, 2014",
          star.cutoffs = c(0.05, 0.01, 0.001),
          header = FALSE,
          table.placement = "!h")
```

Table 1: Log wealth of billionaires as a function of age, 2014

	Dependent variable:
	logwealth
age	0.008^{***}
	(0.001)
Constant	21.146^{***}
	(0.096)
Observations	1,590
R^2	0.020
Adjusted R^2	0.019
Residual Std. Error	0.776 (df = 1588)
F Statistic	32.180^{***} (df = 1; 1588)
Note:	*p<0.05; **p<0.01; ***p<0.001

The equation for the regression line here is:

$$\text{Log of Wealth} = 21.15 + .0084 \times \text{Age}$$

Interpreting the results

What do the coefficients tell us in concrete terms?

- β_0:
- β_1:

What is the expected value of logged wealth at age:

- 0:
- 50:
- 100:

What is the **null hypothesis**?

- β_0: $\beta_0 = mean(Y)$
- β_1: $\beta_1 = 0$

What is the **alternative hypothesis**?

- β_0: $\beta_0 \neq mean(Y)$
- β_1: $\beta_1 \neq 0$

What are the **test statistics**?

```
## Using formulas to calculate the test statistics
t.int <- b0 / summary(bilm)$coefficients[1,2]
t.age <- b1 / summary(bilm)$coefficients[2,2]
t.int
```

```
## [1] 220.7024
```

```
t.age
```

```
## [1] 5.672773
```

```
## Note that we can extract these, too
summary(bilm)$coefficients[1,3]
```

```
## [1] 220.7024
```

```
summary(bilm)$coefficients[2,3]
```

```
## [1] 5.672773
```

What are the **p-values**?

Remember from lecture this week that we use a t-distribution to calculate the p-values and test statistic. When we use a t-distribution, we need to include what's called the "degrees of freedom." Usually that just means "the number of observations you have minus the number of variables you're using."

```
## Using the formulas
df <- nrow(bills) - 2
2 * pt(-abs(t.int), df = df)
```

```
## [1] 0
```

```
2 * pt(-t.age, df = df)
```

```
## [1] 1.66668e-08
```

```
## Extracting them
summary(bilm)$coefficients[1,4]
```

```
## [1] 0
```

```
summary(bilm)$coefficients[2,4]
```

```
## [1] 1.66668e-08
```

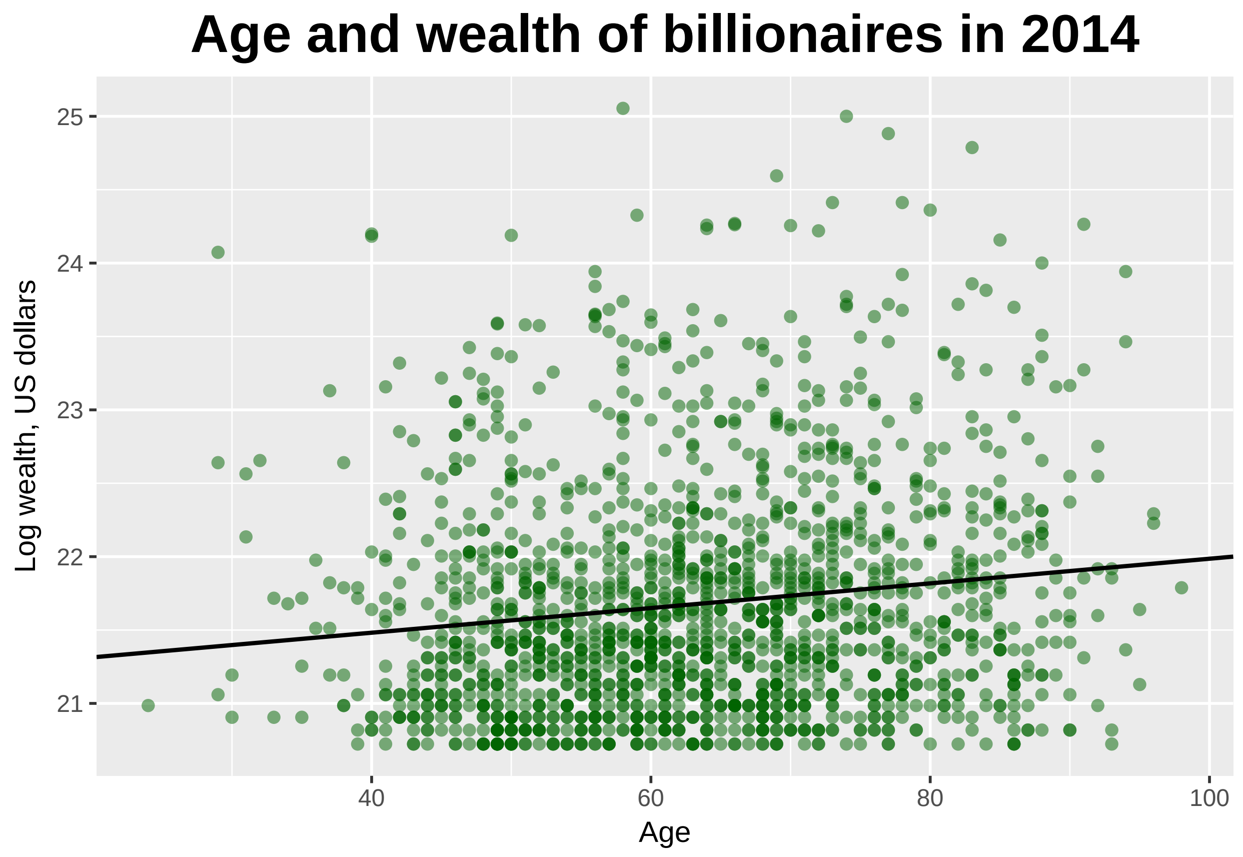
Can we reject the null at the 95% level?

- β_0:
- β_1:

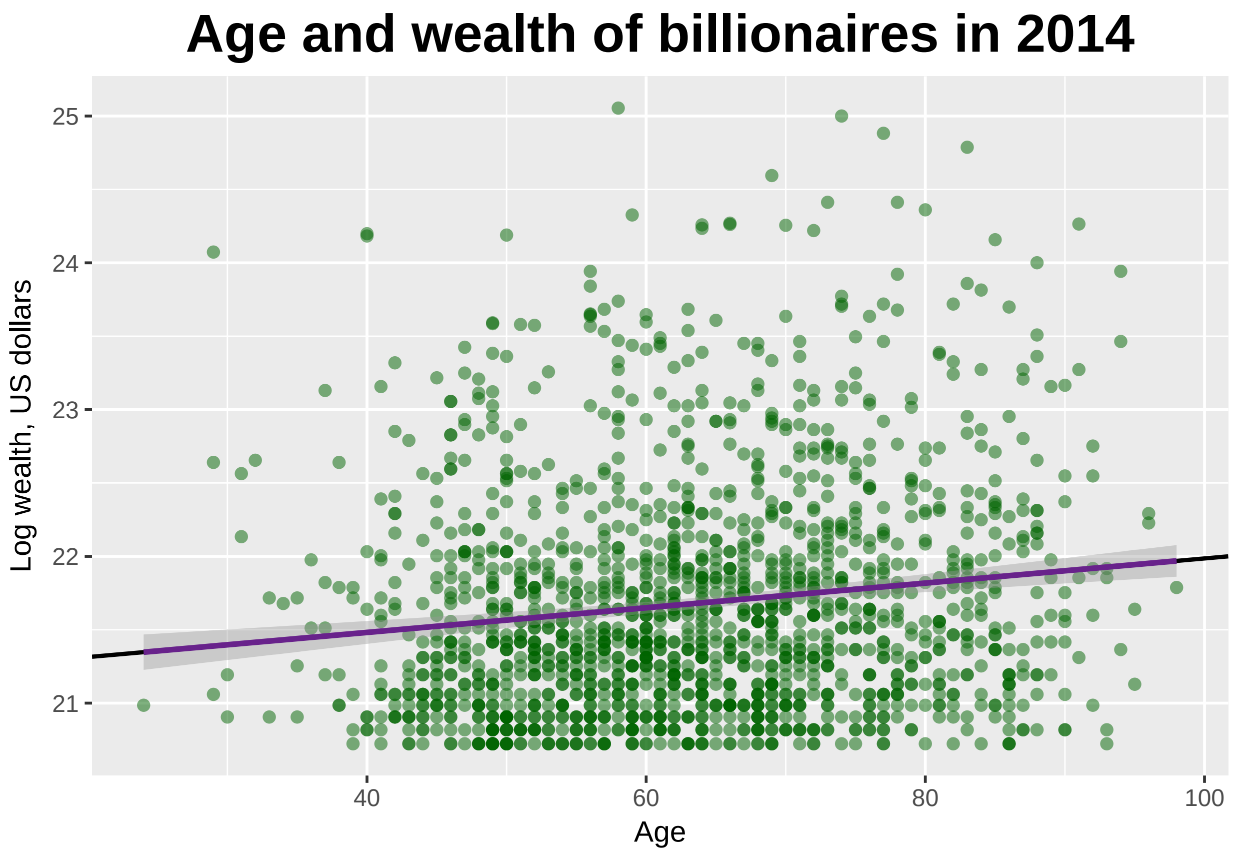
Plotting the results

[Here's a list of ggplot2 colors]{http://sape.inf.usi.ch/quick-reference/ggplot2/colour}

```
## Re-do scatterplot but add regression line and lots of other nice aesthetic features
ggplot(data = bills, aes(x = age, y = logwealth)) +
  geom_point(shape = 16, size = 2, alpha = .5, color = "darkgreen") +
  xlab("Age") +
  ylab("Log wealth, US dollars") +
  ggtitle("Age and wealth of billionaires in 2014") +
  geom_abline(intercept = b0, slope = b1, size = .75) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
        text = element_text(family = "Helvetica"))
```



```
## ggplot will plot the regression line for you if you ask
ggplot(data = bills, aes(x = age, y = logwealth)) +
  geom_point(shape = 16, size = 2, alpha = .5, color = "darkgreen") +
  xlab("Age") +
  ylab("Log wealth, US dollars") +
  ggtitle("Age and wealth of billionaires in 2014") +
  geom_abline(intercept = b0, slope = b1, size = .75) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
        text = element_text(family = "Helvetica")) +
  geom_smooth(method = "lm", se = TRUE, color = "darkorchid4")
```



Looking at the residuals

```
## let's look at the fitted values
#bilm$fitted.values

## let's take a look at the residuals,
## which R calculates as the actual values minus fitted values.
#bilm$residuals

## let's calculate our own residuals
bill_y <- bills$logwealth
bill_yhat <- bilm$fitted.values
resid <- bill_y - bill_yhat

## check residuals we calculated are the same as those from lm()
all.equal(resid, bilm$residuals)
```

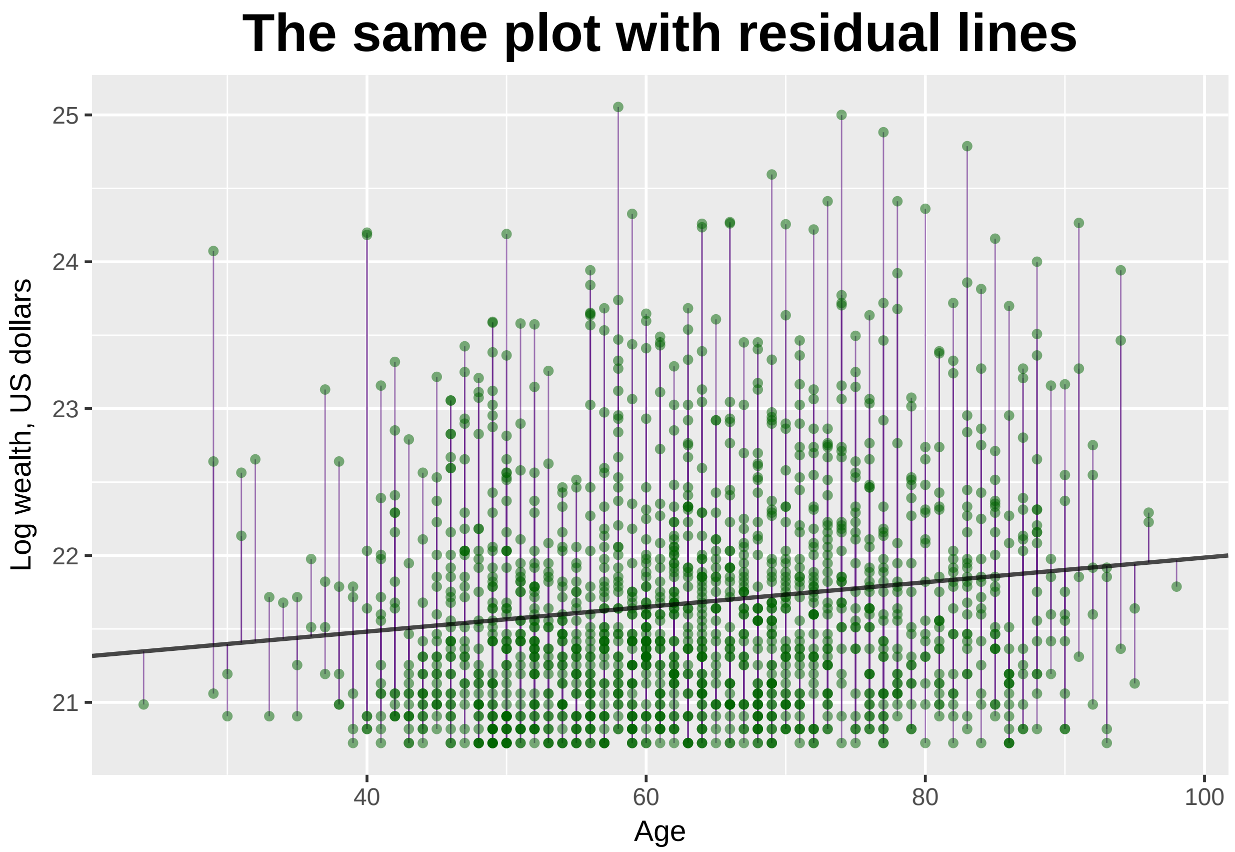
```
## [1] TRUE
```

```
## you can also sum the square of the residuals
## and calculate the R squared value
ss_res <- sum(resid ^ 2)
```

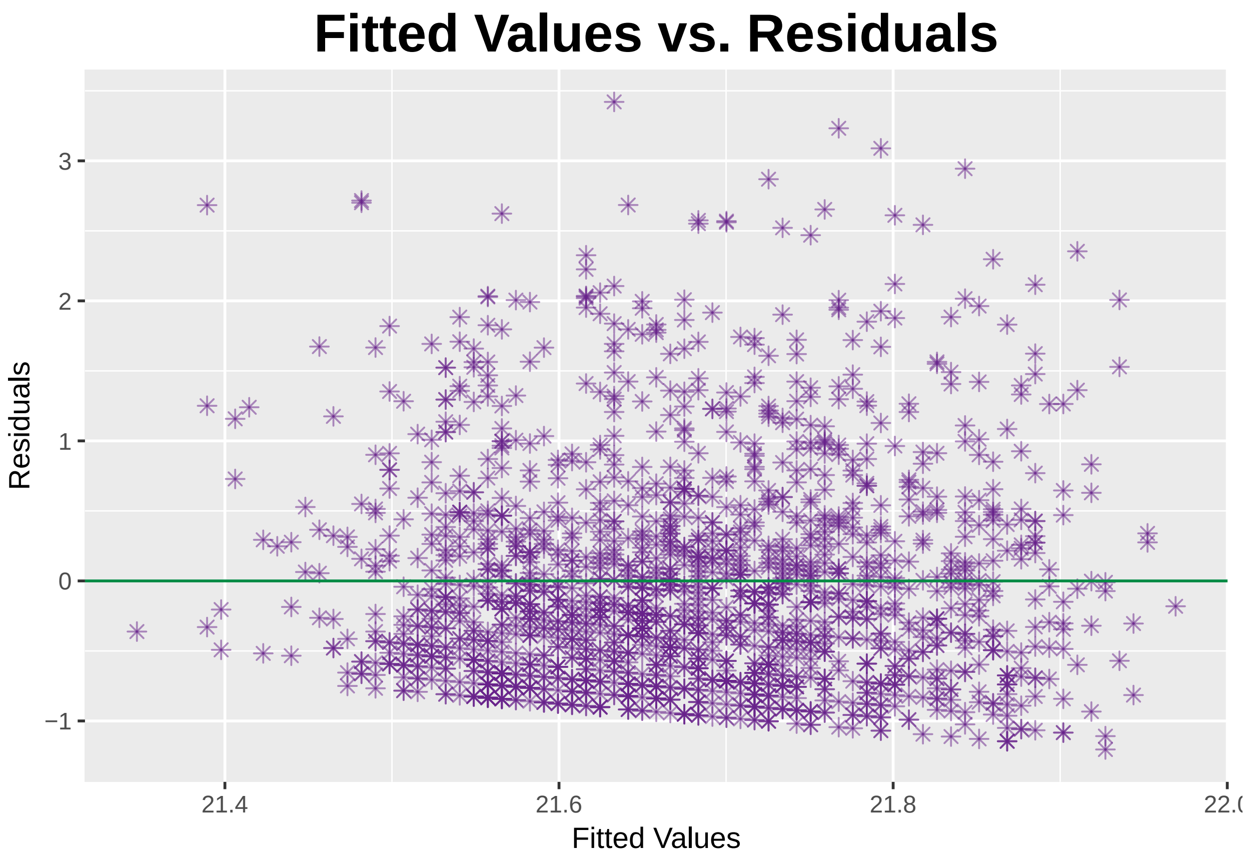
```r
ss_tot <- sum((bill_y - mean(bill_y)) ^ 2)

## R squared
1 - (ss_res / ss_tot)
```

```
## [1] 0.01986221
```

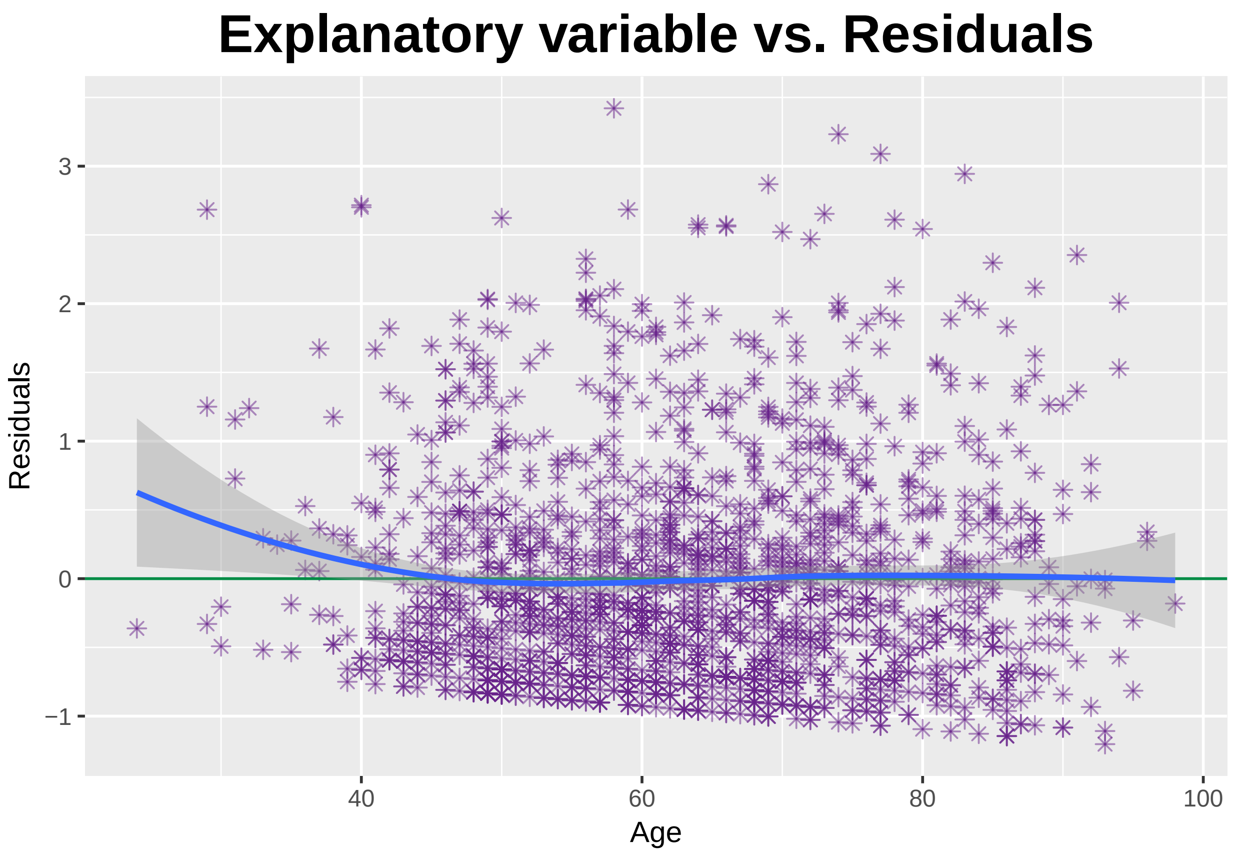
```r
## Adding in the residual lines
ggplot(data = bills, aes(x = age, y = logwealth)) +
  xlab("Age") +
  ylab("Log wealth, US dollars") +
  ggtitle("The same plot with residual lines") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
        text = element_text(family = "Helvetica")) +
  geom_segment(aes(x = age, xend = age,
                   y = logwealth, yend = bilm$fitted.values),
               color = "darkorchid4", alpha = .6, size = .25) +
  geom_point(shape = 16, size = 1.5, alpha = .5, color = "darkgreen") +
  geom_abline(intercept = b0, slope = b1, size = .75, alpha = .7)
```



```r
## We can also create a plot of residuals
## Does it look like we have homoskedasticity? Zero conditional mean?
ggplot(data = bills, aes(x = bilm$fitted.values, y = bilm$residuals)) +
  geom_point(shape = 8, size = 2, alpha = .5, color = "darkorchid4") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  ggtitle("Fitted Values vs. Residuals") +
  geom_abline(intercept = 0, slope = 0, color = "springgreen4") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
        text = element_text(family = "Helvetica"))
```



```
ggplot(data = bills, aes(x = age, y = bilm$residuals)) +
  geom_point(shape = 8, size = 2, alpha = .5, color = "darkorchid4") +
  xlab("Age") +
  ylab("Residuals") +
  ggtitle("Explanatory variable vs. Residuals") +
  geom_abline(intercept = 0, slope = 0, color = "springgreen4") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
        text = element_text(family = "Helvetica")) +
  geom_smooth(method = "loess")
```



3.Sampling and regression

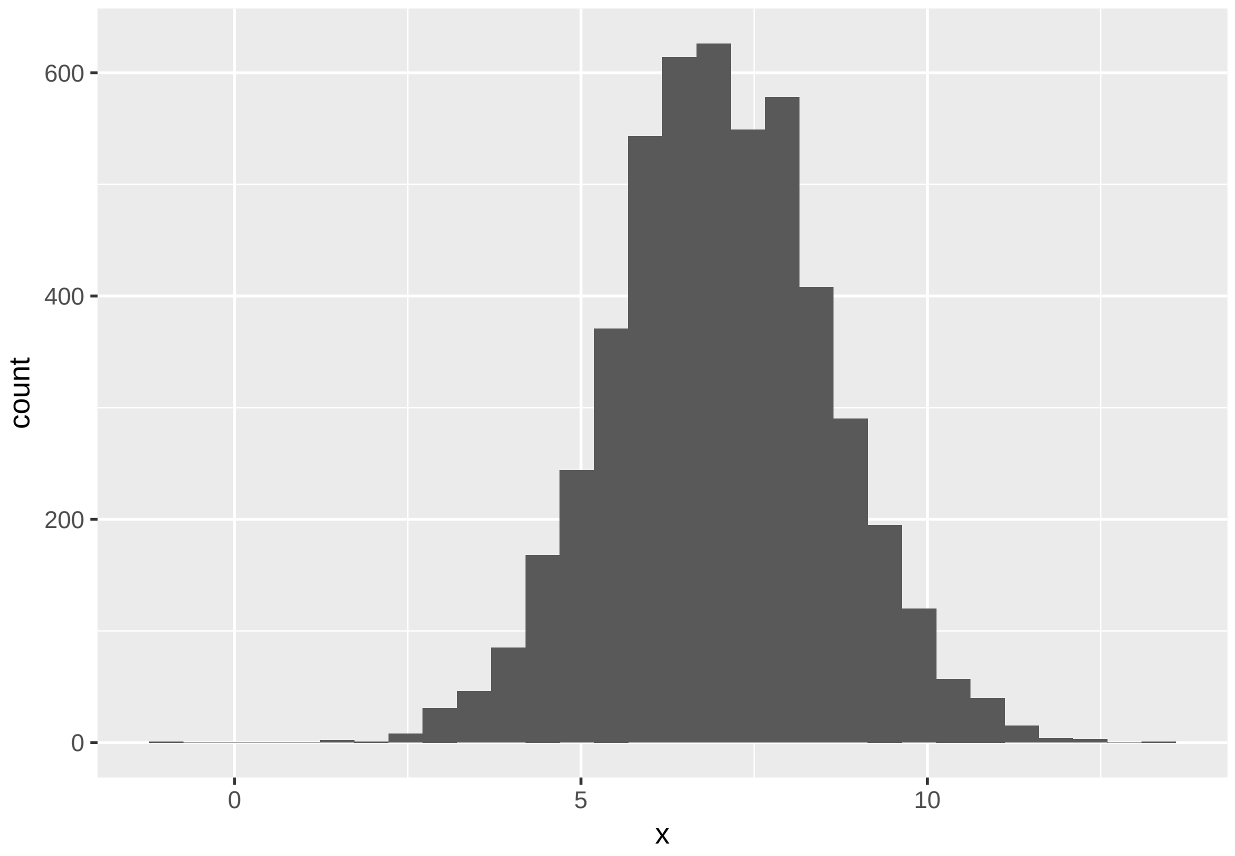
First, we're going to create a bunch of fake data and pretend that it's the true population.

```
## let's generate some data and pretend that it's the true population.
## we can do this with the rnorm function
set.seed(08544)
x <- rnorm(5000, mean = 7, sd = 1.56) # just some normally distributed data

## We're establishing here a linear relationship,
## What's this "true" linear relationship we're setting up?
## So that y = 12 - .4x + some normally distributed error values
y <- 12 - 0.4*x + rnorm(5000, mean = 0, sd = 1)

## Putting it together into a frame
data <- data.frame(x, y)
```

Visualizing our new made-up data:

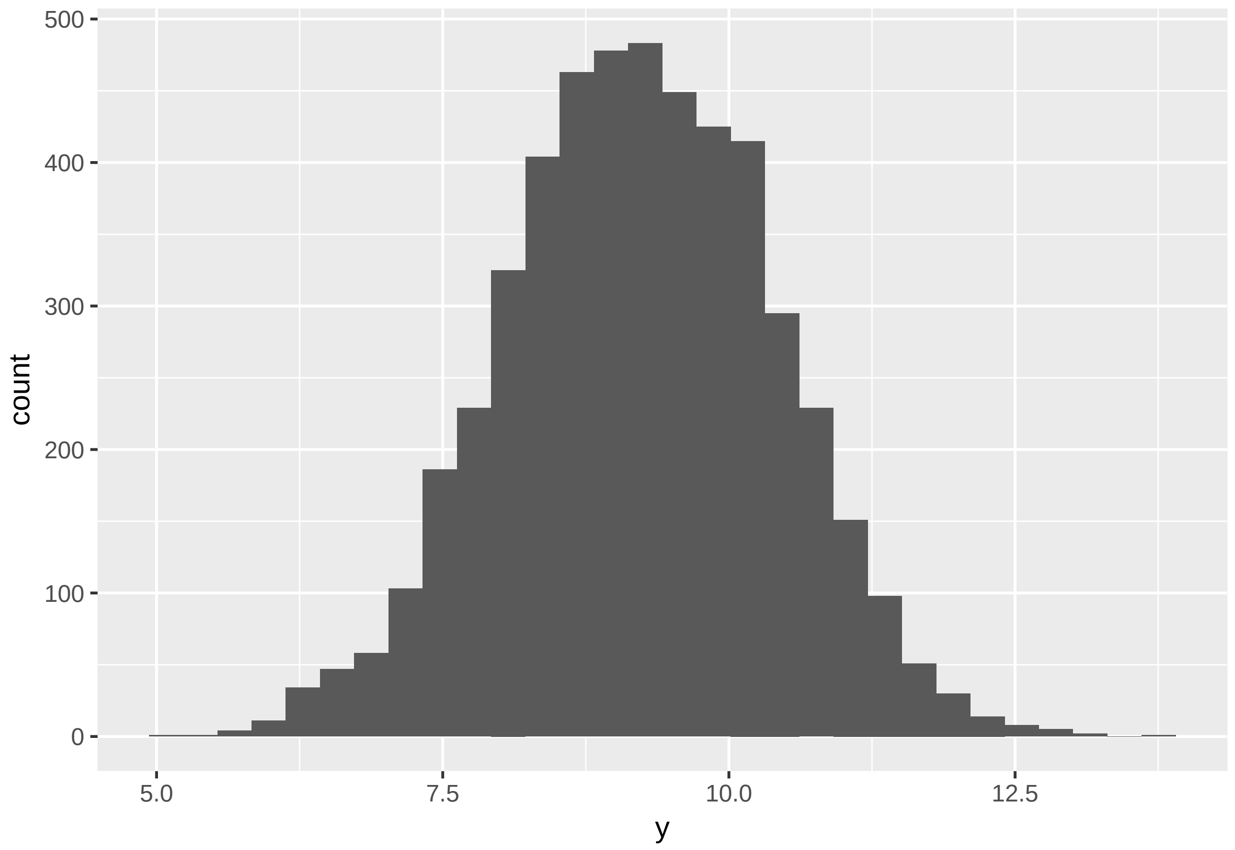
```
## Let's  check out the histograms separately
qplot(x, data = data)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

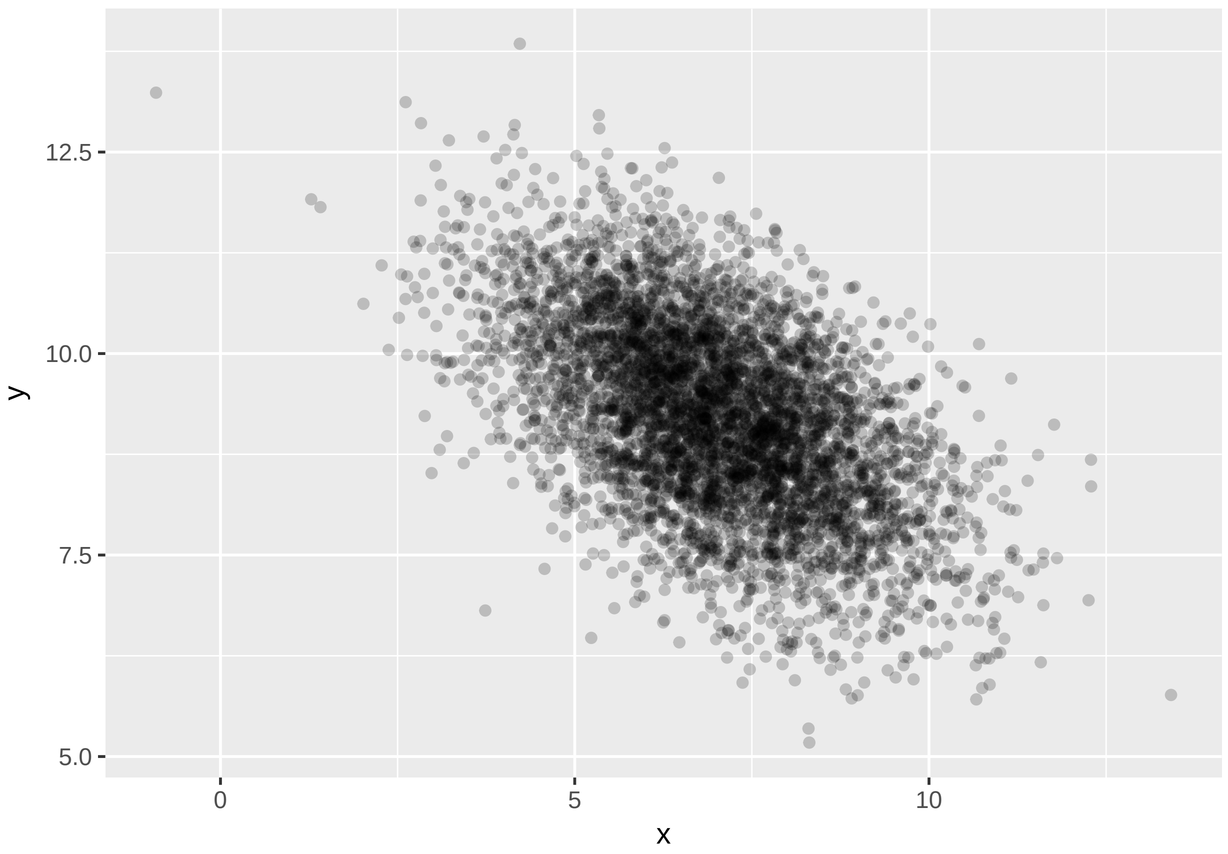


```
qplot(y, data = data)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



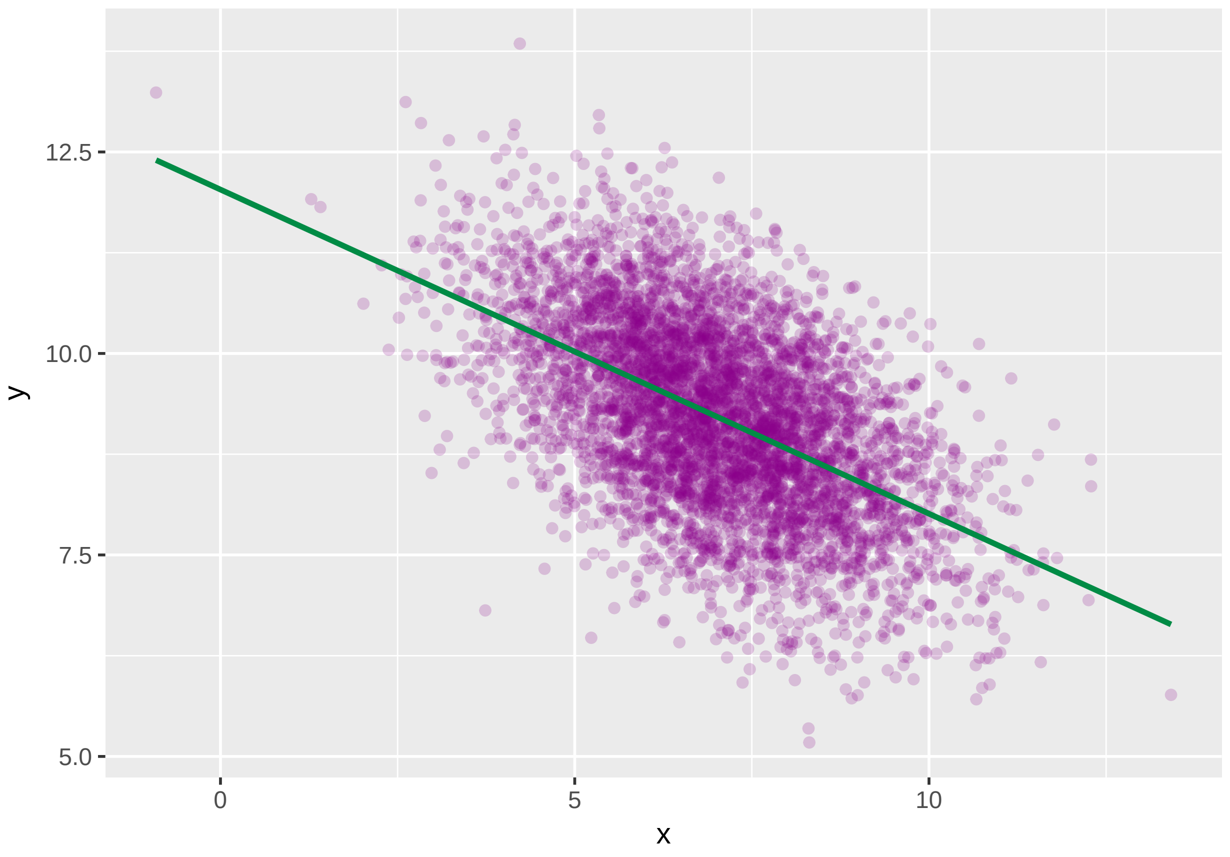
```
## You know they are normal because??

## plotting them together:
ggplot(data = data, aes(x = x, y = y)) +
  geom_point(alpha = .2)
```



```
## you see they are roughly linear

## Our regression should give us back estimates for our
## intercept and slope (remember we're at the population level)
fake.lm <- lm(y ~ x)
summary(fake.lm)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7188 -0.6658  0.0158  0.6944  3.5102
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.033221   0.064450  186.71   <2e-16 ***
## x           -0.402152   0.008986  -44.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9932 on 4998 degrees of freedom
## Multiple R-squared:  0.2861, Adjusted R-squared:  0.2859
## F-statistic:  2003 on 1 and 4998 DF,  p-value: < 2.2e-16
```

```
## Plotting the regression line
ggplot(data = data, aes(x = x, y = y)) +
  geom_point(alpha = .2, color = "darkmagenta") +
  geom_smooth(method = "lm", se = FALSE, color = "springgreen4")
```



Now we're going to start sampling from our data. This is what it looks like to take one sample and run one regression on it.

```
size <- 1000
my.samp <- sample_n(data, size)
summary(lm(my.samp$y ~ my.samp$x))
```

```
##
## Call:
## lm(formula = my.samp$y ~ my.samp$x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5419 -0.6191  0.0327  0.6868  2.9508
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.06506    0.13838   87.19   <2e-16 ***
## my.samp$x   -0.40309    0.01919  -21.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9804 on 998 degrees of freedom
## Multiple R-squared:  0.3066, Adjusted R-squared:  0.306
## F-statistic: 441.4 on 1 and 998 DF,  p-value: < 2.2e-16
```

Now we'll do that a bunch of times using two different simulation methods: a for loop and the `replicate()` function.

First, the for loop.

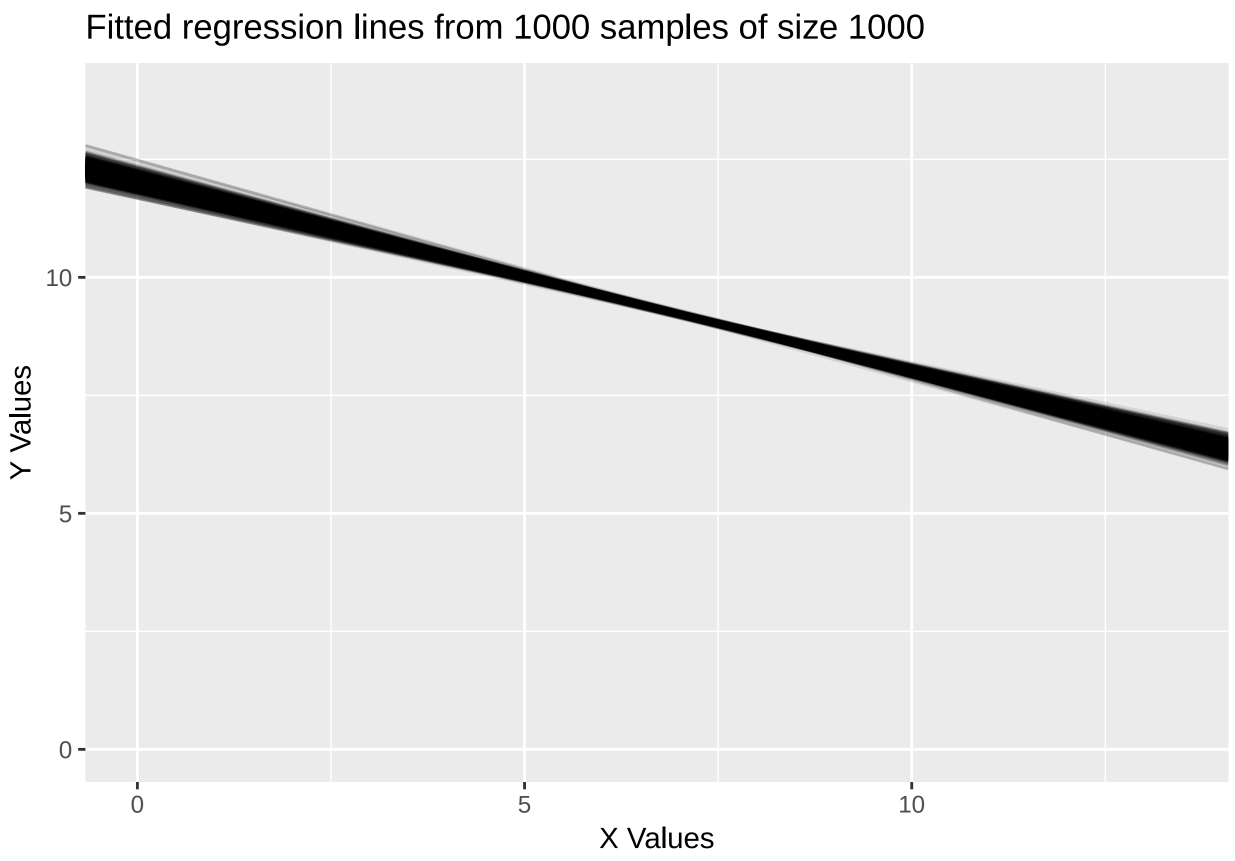
```r
## we can set up a for loop to sample a thousand times,
## and store the regression coefficients from each run
sims <- 1000
holder <- matrix(data = NA, ncol = 2, nrow = sims)
colnames(holder) <- c("intercept", "slope")

for (i in 1:sims) {
  my.samp <- sample_n(data, size)
  samp.lm <- lm(my.samp$y ~ my.samp$x)
  holder[i, 1] <- samp.lm$coefficients[1]
  holder[i, 2] <- samp.lm$coefficients[2]
}
```

Now we can try it with the `replicate()` function, which is faster.

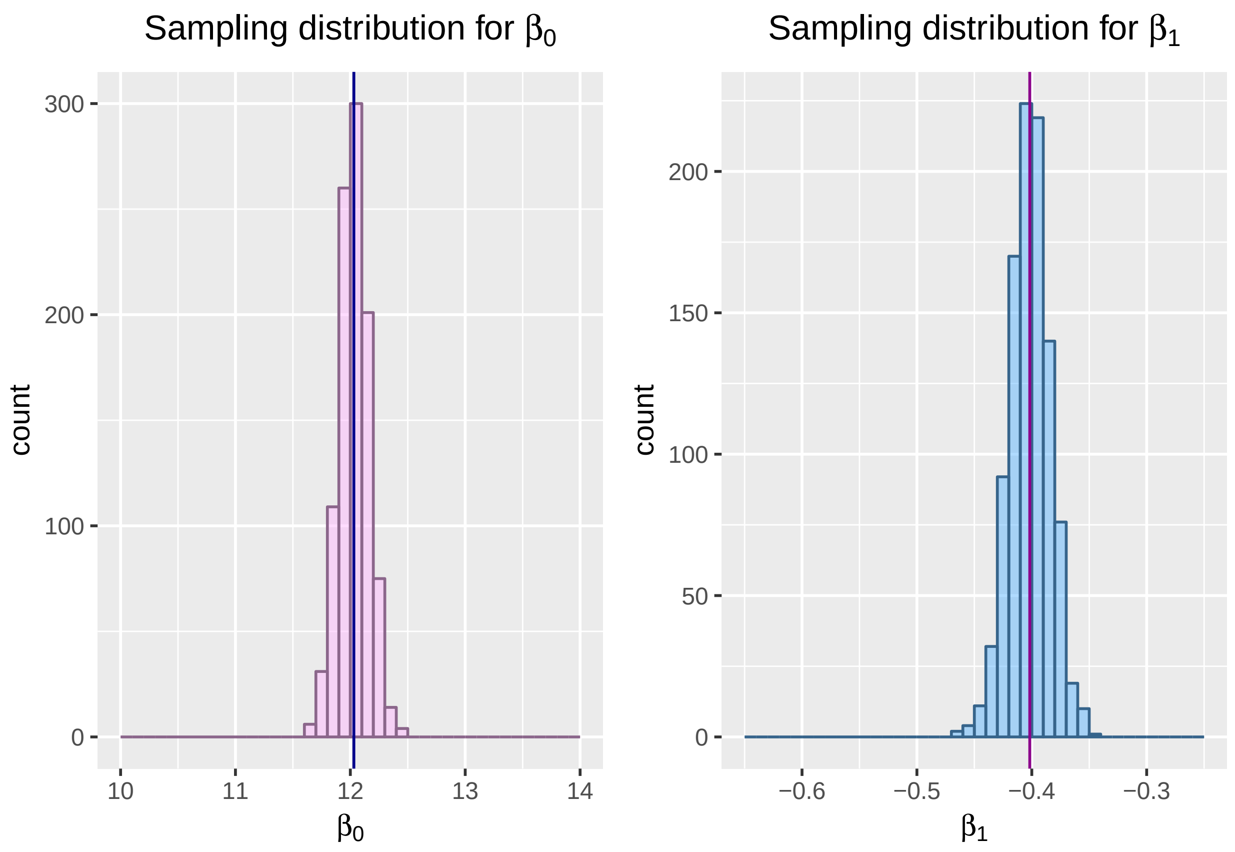
```r
## First we have to define the function replicate
# unlike most of our functions, we don't need any arguments!
my_simulation <- function() {
  my.samp <- sample_n(data, size)
  samp.lm <- lm(my.samp$y ~ my.samp$x)
  holder <- samp.lm$coefficients[1]
  holder[2] <- samp.lm$coefficients[2]
  return(holder)
}

## Now we just tell it to do that 1000 times!
results <- replicate(1000, my_simulation())

## Our results come out with the coefficients on the
## rows and simulations on the columns.
## We will change this by transposing the matrix
## to make it easier to look at.
holder <- data.frame(t(results))
colnames(holder) <- c("Intercept", "Slope")
```

Visualizing the samples

```r
## What do all these lines look like plotted together?
ggplot() +
  scale_x_continuous(name = "X Values", limits = c(0, max(x))) +
  scale_y_continuous(name = "Y Values", limits = c(0, max(y))) +
  geom_abline(data = holder, aes(intercept = Intercept, slope = Slope), alpha = .1) +
  ggtitle("Fitted regression lines from 1000 samples of size 1000")
```



```
## We can also plot the marginal distribution
## of the slope and intercept:
hist1000.b0 <- ggplot(data = holder, aes(x = Intercept)) +
  geom_histogram(breaks = seq(10, 14, by = 0.1),
                 fill = "plum1", color = "plum4", alpha = .5) +
  ggtitle(expression("Sampling distribution for " * beta[0])) +
  xlab(expression(beta[0])) +
  xlim(10,14) +
  geom_vline(xintercept = mean(holder[,1]), color = "darkblue") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        text = element_text(family = "Helvetica"))

hist1000.b1 <- ggplot(data = holder, aes(x = Slope)) +
  geom_histogram(breaks = seq(-0.65, -0.25, by = 0.01),
                 fill = "steelblue1", color = "steelblue4", alpha = .5) +
  ggtitle(expression("Sampling distribution for " * beta[1])) +
  xlab(expression(beta[1])) +
  xlim(-0.65,-0.25) +
  geom_vline(xintercept = mean(holder[,2]), color = "darkmagenta") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        text = element_text(family = "Helvetica"))

grid.arrange(hist1000.b0, hist1000.b1, nrow = 1, ncol = 2)
```



1. Why do they look normal? (And why does that make sense?)
2. What's the mean of these two distributions? (And why does that make sense?)