Simulations

Lauren Quesada

10/22/2021

```
###Load packages

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom)

###Define function

LFunky<- function(beta0,beta1,beta2,n_obs,n_data,n_shuffle) {

  ###Define variables

  set.seed(4747)
  sig<- 0.05


    #If there is no intercept in the fitted regression, d.o.f. = (n-1)
    #otherwise, df=n-k-1; k=the number of variables
  dof <- n_obs-1

    #addp_val<-c()

    #intp_val<-c()

  simpp_val<-c()

  simp_t<-c()

  perm_pval<-c()
```

```r
###Begin loop

for(i in 1:n_data) {               # Start of the Monte Carlo loop

###Define Variables

x1<- rnorm(n_obs)     # Artificial x series, created just once

x2<- x1+ rnorm(n_obs)     ## Did we figure out why these need to be defined in the loop?

y<- beta0 + beta1*x1 + beta2*x2 + rnorm(n_obs)   # The DGP includes an intercept

###Fit a linear regression model

simpfit<- lm(y ~ x1  -1)

###Record slope and p-value

#observed p.value for each dataset (n_data)
simpp_val<- c(simpp_val,
              simpfit %>% tidy() %>% filter(term =="x1") %>% select(p.value) %>% pull())

#observed slope for each dataset (n_data)
simp_t<- c(simp_t,
              simpfit %>% tidy() %>% filter(term =="x1") %>% select(statistic) %>% pull())

#obs value for most recent dataset
obsval<- simpfit %>% tidy() %>% filter(term =="x1") %>% select(statistic) %>% pull()

#reassign vector so it doesn't add onto the concatenation each i in 1:n_data loop
simpp_valperm<-c()

  ###Begin another loop

  for(j in 1: n_shuffle) {
# A mis-specified model is estimated (unless Beta0 = 0)
  #addfit<- lm(y ~ x1+x2  -1)

  #intfit<- lm(y ~ x1*x2  -1)


  ###Fit linear regression model to PERMUTED data

  simpfitperm<- lm(y ~ sample(x1)  -1)

  #addp_val<- c(addp_val, unname(coef(summary(addfit))[, "Pr(>|t|)"]))

  #intp_val<- c(intp_val, unname(coef(summary(intfit))[, "Pr(>|t|)"]))

  ###Record slope
  simpp_valperm<- c(simpp_valperm,
                    simpfitperm %>% tidy() %>% filter(term == "sample(x1)") %>% select(statistic) %>%
```

```
    }

  #p-value for each of my 200 datasets as compared to my permuted distribution of p-values
  perm_pval<-c(perm_pval,mean(simpp_valperm>obsval))

  }  #End of the Monte Carlo Loop

#coefficients, t-test p.values, permutation p.values
data.frame(simp_t, simpp_val, perm_pval)




}

#hist(addp_val, main="Distribution of  *ADDITIVE* p-Values", xlab="p-Value", freq=FALSE, border="black"

#hist(intp_val, main="Distribution of *INTERACTIVE* p-Values", xlab="p-Value", freq=FALSE, border="blac

#hist(simpp_val, main="Distribution of *SIMPLE* p-Values", xlab="p-Value", freq=FALSE, border="black",

#results %>% ggplot() + geom_hist(aes(x = simppval))
```

In the function above, I am repeatedly (100 times) sampling 15 values/observations from a normal distribution. I have 15 (x_i, y_i) values in each dataset. I have 100 of these datasets. I am fitting a simple linear regression to each dataset, knowing I am not including my x_2 variable in the fit, knowing my data has two explanatory variables x_1 *and* x_2. Note: x_2 $x_1 + rnorm(n_obs)$; x_2 is related to x_1, plus some noise. We *know* it is dependent.

I am asking it to then record the observed slope (β_1) of this fitted line, as well as the p-value. Is there a correlation between my x and y variables ("no" implies $\beta_1 = 0$)? Permutation tests *assume* the null hypothesis to be true. We don't have any correlation between our data, and permuting should not change this.

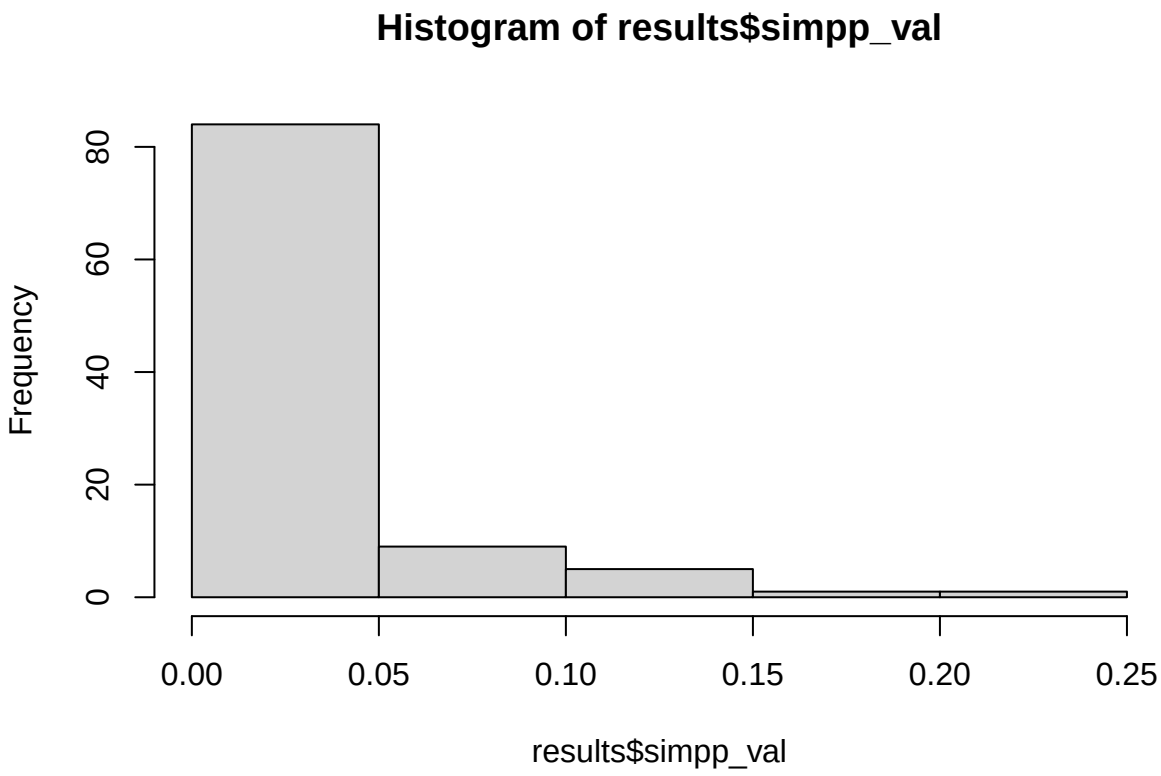
Next, I ask it to shuffle/permute my x_1 values, 200 times. Each dataset is shuffled 200 times, and each of those shuffles repeats the process described above. A line is fit to the permuted data, and the slope and p-value are recorded. Pretty soon, I have a distribution of permuted p-values and slopes.

But, I want to be able to compare my observed values to my permuted values, so I need the same size data frame for both. I take one of my 100 datasets. I shuffled it 200 times. I get a distribution of 200 permuted p-values for the 200 lines I fit (one for each permutation). I average the slopes of my lines to get 100 statistics. Then, I compare. How many permuted p-values are greater than my observed value? This is how I obtain my 100 p-values from my permuted data. If I shuffle my data, is there a correlation between my x and y variables?

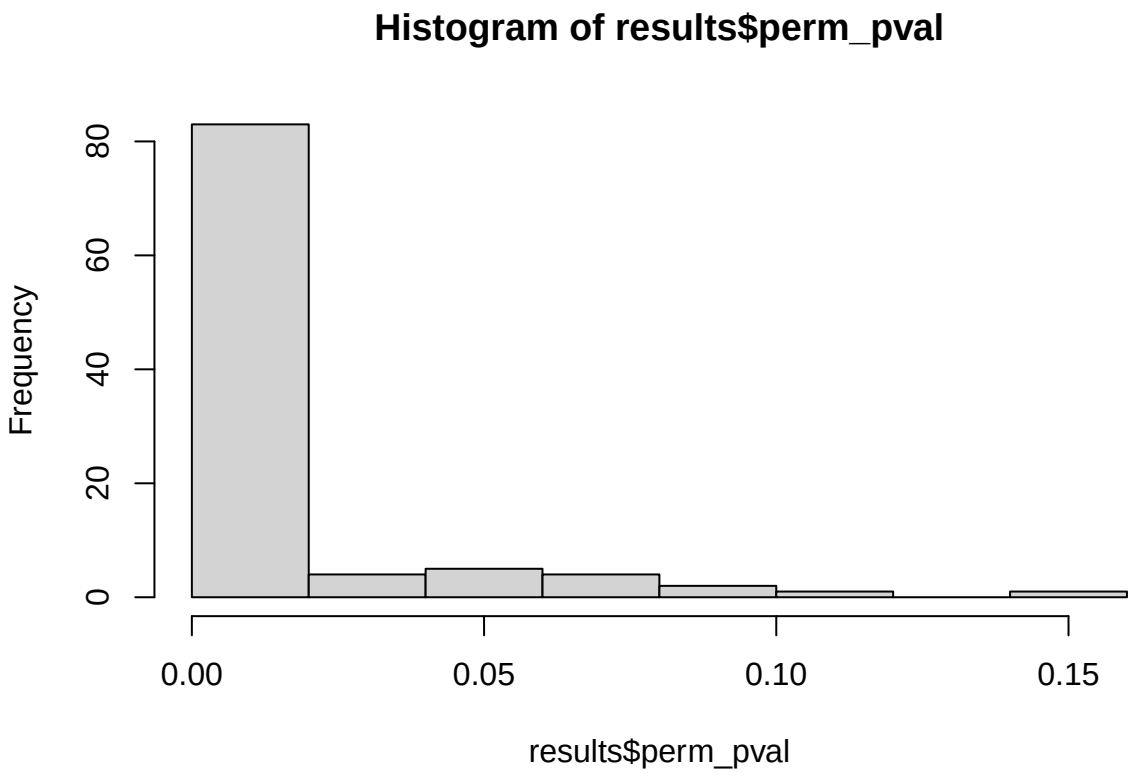
Remember, we've assumed the null hypothesis to be true. We're conducted this entire test assuming there is no correlation between my variables. So, which method is better at identifying this from the data? How do my normally distributed datasets compare to my permuted data–which is wrong *less* of the time?

$Y_i = 0 + \beta_1 * x_1 + 0 * x_2$ $Y_i = \beta_1 * x_1$ n=15, normally distributed 100 datasets 200 permutations

```
#beta0,beta1,beta2,n_obs,n_data,n_shuffle
results<-LFunky(0,1,0,15,100,200)
hist(results$simpp_val)
```




```
hist(results$perm_pval)
```



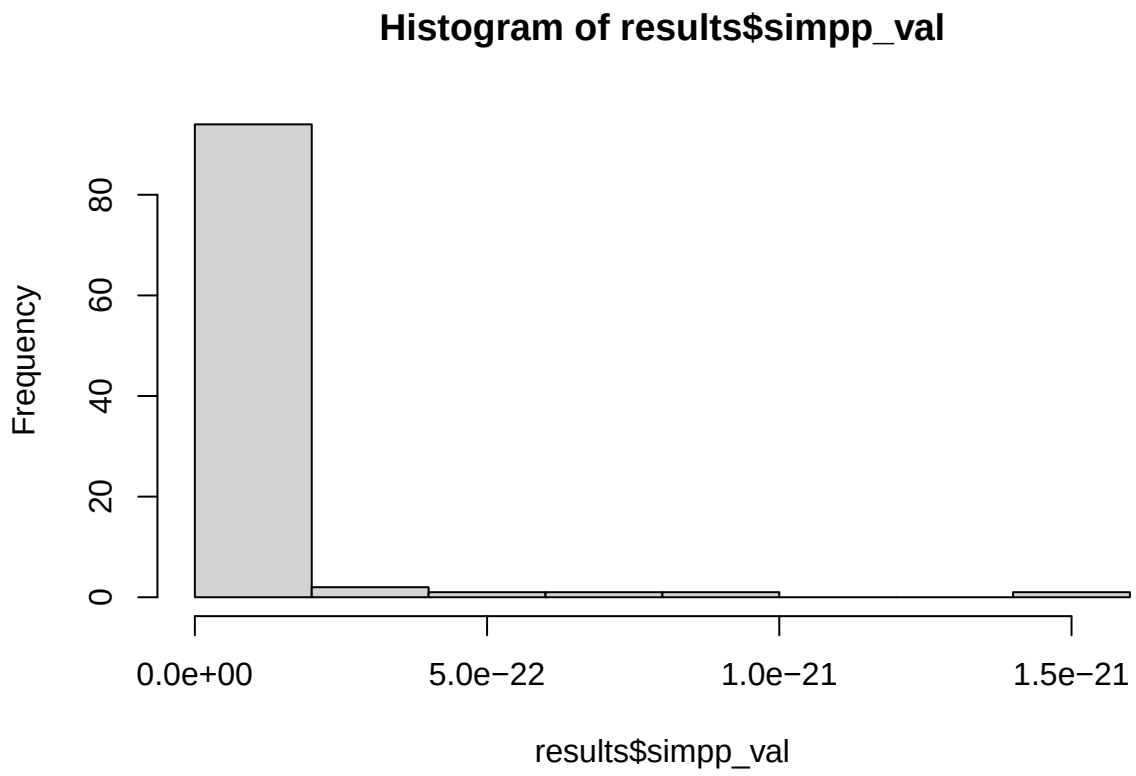
They're both right! The majority of p-values confirm there is *no* correlation between my variables ($\beta_1 = 0$), *BUT* my permuted p-values are more confidently saying this. More often and with a lower significance level, there are few permuted p-values larger than my observed value.

Let's try increasing the slope, β_1 to 50 and see if this still holds true. I assume it will.

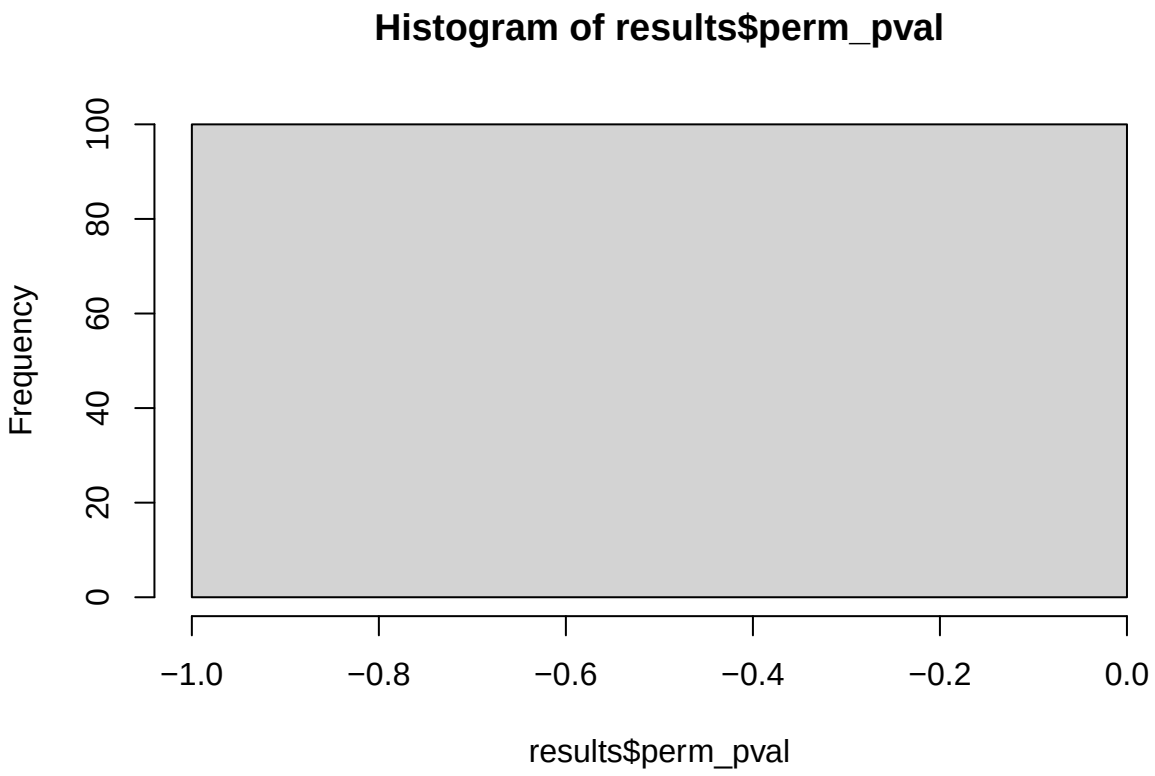
$Y_i = 0 + \beta_1 * x_1 + 0 * x_2$ $Y_i = \beta_1 * x_1$

QUESTION: Do I need to change my intercept to -50 to force it through the origin? I get an error when I do this, why? Can I remove this from my function?

```
#beta0,beta1,beta2,n_obs,n_data,n_shuffle
results<-LFunky(0,50,0,15,100,200)
hist(results$simpp_val)
```




```
hist(results$perm_pval)
```

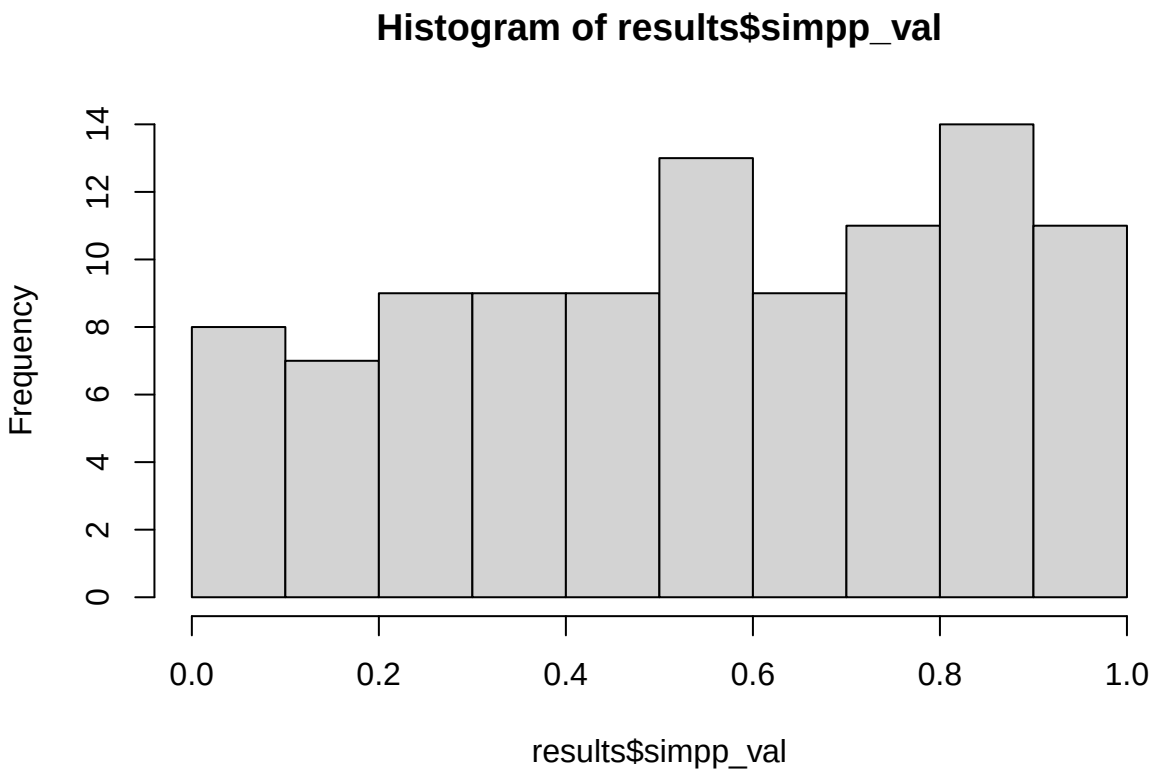


What happened? I introduced a highly correlated slope, and my simple linear model was very good at recognizing the lack of correlation. But, it seems my permuted data failed *because* my simple linear model was so good. With a p-value of nearly zero in almost all of my 100 data sets, there was almost *never* a chance for the permuted values to be greater than my observed value? I'm not sure, but my p-values can't be negative. Is there some change that needs to be made in the function to avoid this?

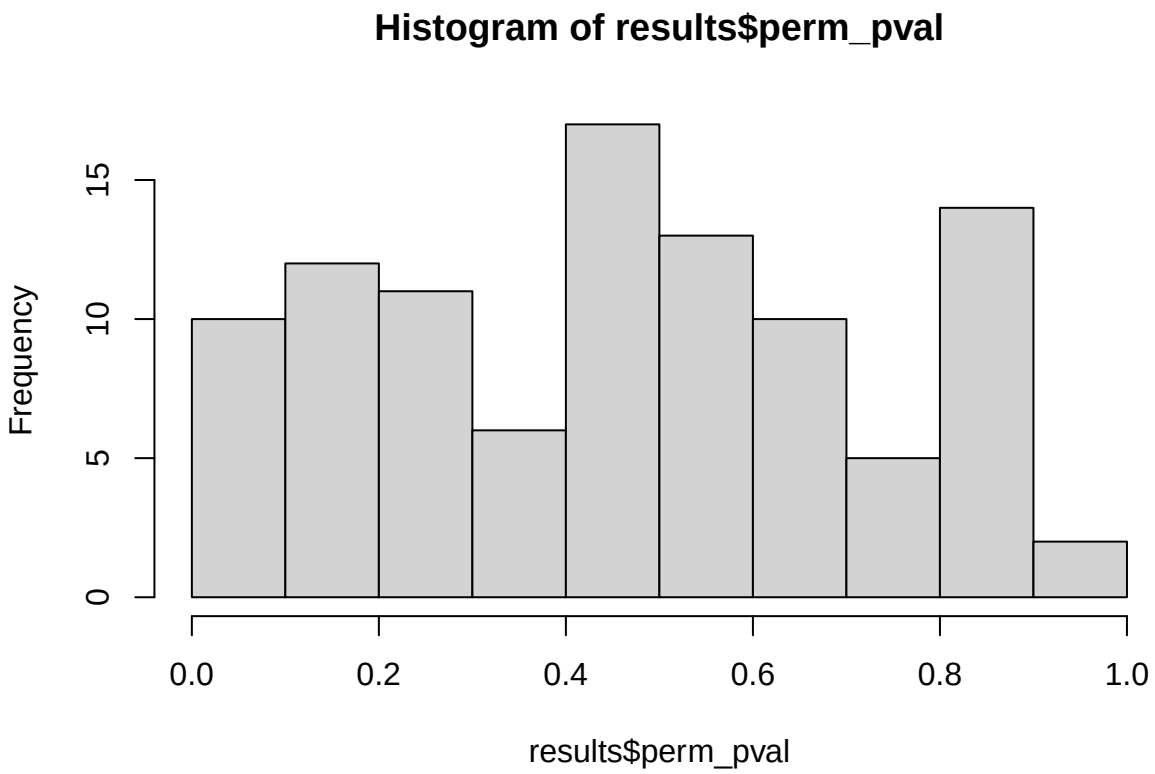
Let's try a *smaller* slope,$\beta_1 = 0.1$ and see if this still holds true. I assume it will.

$Y_i = 0 + \beta_1 * x_1 + 0 * x_2$ $Y_i = \beta_1 * x_1$

```
#beta0,beta1,beta2,n_obs,n_data,n_shuffle
results<-LFunky(0,0.1,0,15,100,200)
hist(results$simpp_val)
```




```
hist(results$perm_pval)
```



Wrong again; with a slope very close to zero, I almost get a uniform distribution for both p-values. I assume if I ran more permutations, I would find a more uniform distribution. ***QUESTION*** : I don't think uniform is the right word for the distribution since it's not [0,1], but what is? Since the slope is so close to zero, all p-values are valid. Some data sets have correlation, some don't. I think my data is almost completely randomly distributed here, so p-values for both return a lack of consistency or conclusion. While it seems the simple p-values are more consistent across their distribution, the permuted p-values have a slightly higher frequency of low proportions. I wonder what would happen if I increased the number of permutations?

Let's try it!

Okay, let's put β_1 back to 1, but let's try a different intercept. How about $\beta_0 = 1$?