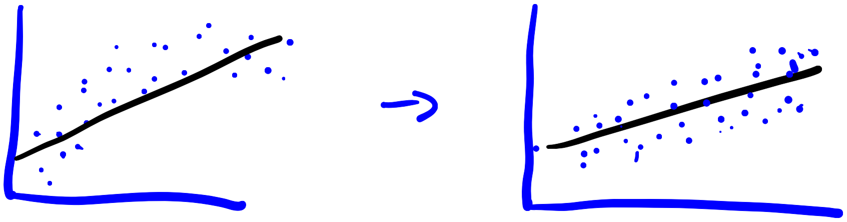


fit linear model to non-linear relationship

generate data from model - does it look like the real data!

Investigating models using simulation

Ben Prytherch

7/15/2021

Setup

- We typically use statistical models to analyze data and to draw inference from the results.
- Usually the workflow is:

  1. Prepare experimental design, perhaps giving some thought to how the data will later be analyzed
  2. Conduct experiement and collect data
  3. Figure out the details of the statistical analysis, which will involve fitting a model of some kind.

- There's nothing wrong with this on its own terms. We use data to fit models and then draw inferences. But, models can also work in the reverse direction: they can be used to generate fake data.
- Using models as data generating allows us to more closely assess how well the model represents the underlying data generating process. It allows us to investigate how the model will behave under various possible scenarios. It allows us to assess power under various possible scenarios. It allows us to assess probable error in our estimates. It allows us to assess the potential consequences of violating assumptions or of fitting a "wrong" model.

Example scenario (thanks to Jessica Hill for providing this)

Here is an example experimental design. I'm leaving out a lot of technical details, and just providing those which are needed to create the statistical model. I have also made some changes for simplification's sake.

- 20 C. elegans worms (all the same species) will be grown on labeled bacteria of interest: 1) P. aeruginosa PA14; or 2) the CeMbio group (contains 12 different bacteria) until adulthood (about 3 days). The PA14 will serve as a pathogenic exposure, and the CeMbio group will serve as the natural microbiome.
- 10 adult worms will be imaged per slide, per 1 bacterial exposure group (i.e. PA14 and CeMbio).
- This will be repeated 3 times. There may be a "batch effect" that results in total mean counts varying across repetition.
- Possible "effects" of interest:
  - The effect of bacterial species on bacterial count
  - The variability in the "batch effect" on overall mean counts
  - The variability in the "batch effect" on species effect

Creating a generative model

- So, where to start? First, I'd like some realistic values for means and standard deviations of the outcome variable. Dr. Hill provided a reference: CeMbio - The Caenorhabditis elegans Microbiome Resource. It seems that a mean of 10,000 and a standard deviation of ~~1~~2,000 seem like reasonable ballpark values.

- Now, I should consider the model. Here is a simple model, in which mean count only varies by species, and potential batch effects from replicates are ignored:

$$CFU_i = \beta_0 + \beta_1 PH14_i + \varepsilon_i$$

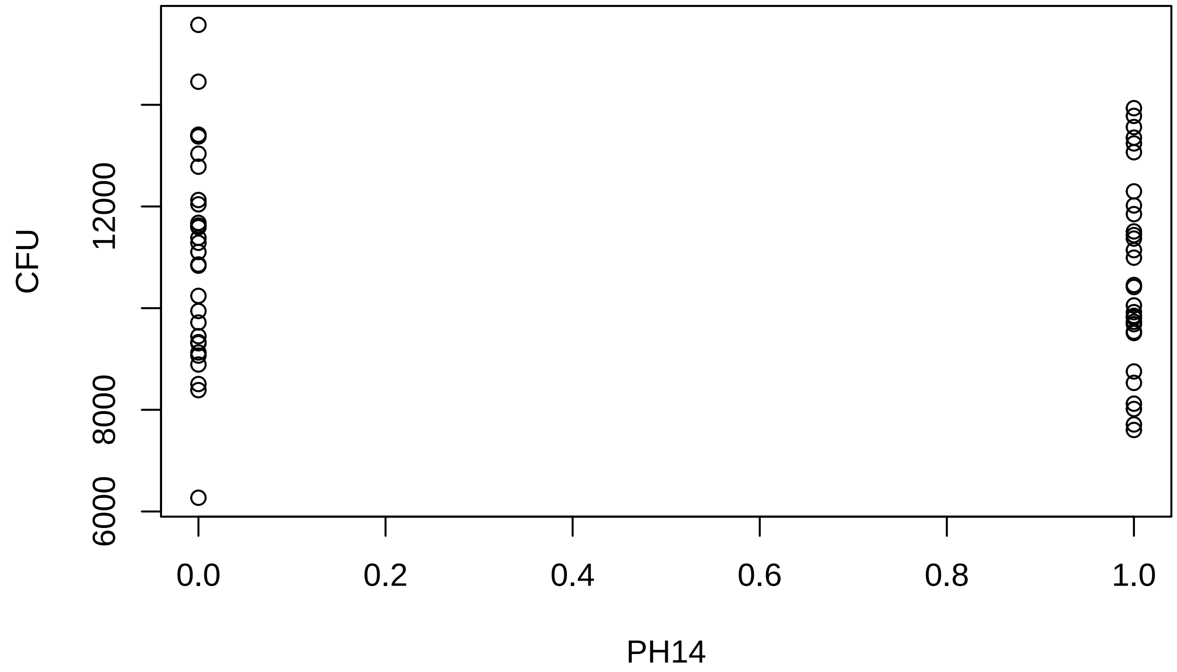
$$\varepsilon_i \sim Normal(0, \sigma_\varepsilon)$$

- Here, β_0 is mean CFU count for CeMbio, and β_1 is the difference between mean CFU counts between CeMbio and PH14.
- The "PH14" variable is 0/1 "dummy" or "indicator" variable.
- σ_ε is the standard deviation of the individual CFU counts around their means for each bacterial group. This standard deviation is assumed to be the same across groups.
- And now the fun part! Here's some code to generate fake data:

```
n <- 60 #sample size
PH14 <- rep(0:1,each=n/2) #make indicator variable
beta_0 <- 10000 #mean for CeMbio
beta_1 <- 1000 #effect of PH14
sigma_e <- 2000 #standard deviation of CFU around means
epsilon <- rnorm(0,sigma_e,n=n) #make vector of residuals taken from a normal distribution
CFU <- rep(0,n) #make empty vector to fill in with simulated data

#simulate CFU counts
for (i in 1:n) {
  CFU[i] <- beta_0+beta_1*PH14[i]+epsilon[i]
}
              10,000    1,000               "residual"

#plot data
plot(PH14,CFU)
```



```
#look at the model results
model <- lm(CFU~factor(PH14))
summary(model)
```

```
## 
## Call:
## lm(formula = CFU ~ factor(PH14))
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -4684.2 -1304.3    27.6  1148.9  4618.8 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)    10954.4      353.7  30.967   <2e-16 ***
## factor(PH14)1   -245.5      500.3  -0.491    0.625    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1938 on 58 degrees of freedom
## Multiple R-squared:  0.004136,  Adjusted R-squared:  -0.01303 
## F-statistic: 0.2409 on 1 and 58 DF,  p-value: 0.6254
```

yikes! It's negative. (only due to random chance)

We can also run a power analysis by simulating this model repeatedly and collecting the p-value each time:

```
reps <- 1000
pvals <- rep(0,reps)
for (i in 1:reps) {
  epsilon <- rnorm(0,sigma_e,n=n)

  for (j in 1:n) {
     CFU[j] <- beta_0+beta_1*PH14[j]+epsilon[j]
  }
  model <- lm(CFU~factor(PH14))
  pvals[i] <- summary(model)$coefficients[2,4]
}

sum(pvals<0.05)/reps
```

```
## [1] 0.503
```

Try changing the values of n, β_1, and σ_e to see how they affect power.

Violating an assumption

- We've seen how to set up a basic generative model, and how to use this to estimate power.
- Generative models are also useful for assessing how things can go wrong. For instance, we can violate the assumption of equal standard deviations across treatments, then fit the model that makes this assumption, and check the effect on power:

```
n <- 60
batch <- rep(1:3,each=n/3)
PH14 <- rep(rep(0:1,each=n/6),3)
beta_0 <- 10000
beta_1 <- 1000
sigma_1 <- 500 #standard deviation for CeMio
sigma_2 <- 4000
reps <- 1000
pvals <- rep(0,reps)
for (i in 1:reps) {
  epsilon <- c(rnorm(0,sigma_1,n=n/2),rnorm(0,sigma_2,n=n/2))

  for (j in 1:n) {
     CFU[j] <- beta_0+beta_1*PH14[j]+epsilon[j]
  }
  model <- lm(CFU~factor(PH14))
  pvals[i] <- summary(model)$coefficients[2,4]
}

sum(pvals<0.05)/reps
```

```
## [1] 0.289
```

← power: we have a 28.9% chance of getting $p<0.05$
this is bad! We want high power

- Violating this assumption reduced our power.

Including a post-treatment predictor variable

that are correlated w/ the outcome of interest

- As a general rule, you should not include "post-treatment" variables as predictors in a model. If the post-treatment variable is influenced by the treatment, then "controlling" for it (i.e. "holding it constant") will reduce the apparent effect of the treatment on the outcome. Let's try it here!
- Suppose we measure worm lifespan, which averages 20 days with a standard deviation of 2 days. But suppose lifespan is negatively affected by CFU count, to the extent that one standard deviation in CFU count corresponds to a roughly 3 day decrease in lifespan:

increase

```
n <- 60
PH14 <- rep(0:1,each=n/2)
beta_0 <- 10000
beta_1 <- 1000
sigma_e <- 2000
reps <- 1000
pvals <- rep(0,reps)
CFU <- rep(0,n)
lifespan <- rep(0,n) #we are including worm lifespan
beta1_original <- rep(0,reps) # save slopes for original model
beta1_biased <- rep(0,reps) # save slope for model including lifespan
for (i in 1:reps) {
  epsilon <- c(rnorm(0,sigma_e,n=n))

  for (j in 1:n) {
     CFU[j] <- beta_0+beta_1*PH14[j]+epsilon[j]
     lifespan[j] <- 20+((CFU[j]-10500)/2000)*3+rnorm(n=1,mean=0,sd=2) #generate lifespans
  }

  model_original <- lm(CFU~factor(PH14))
  model_biased <- lm(CFU~factor(PH14)+lifespan)
  beta1_original[i] <- summary(model_original)$coefficients[2,1]
  beta1_biased[i] <- summary(model_biased)$coefficients[2,1]
}
mean(beta1_original)
```

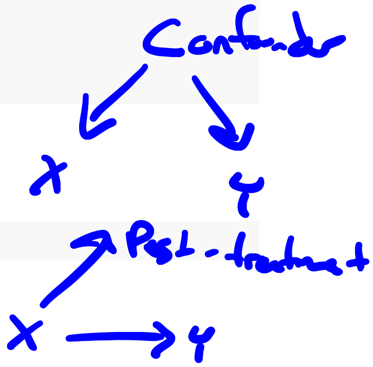
same as before

Z_{CFU}

+ add random noise

```
## [1] 994.1369
```

compare to $\beta_1 = 1000$



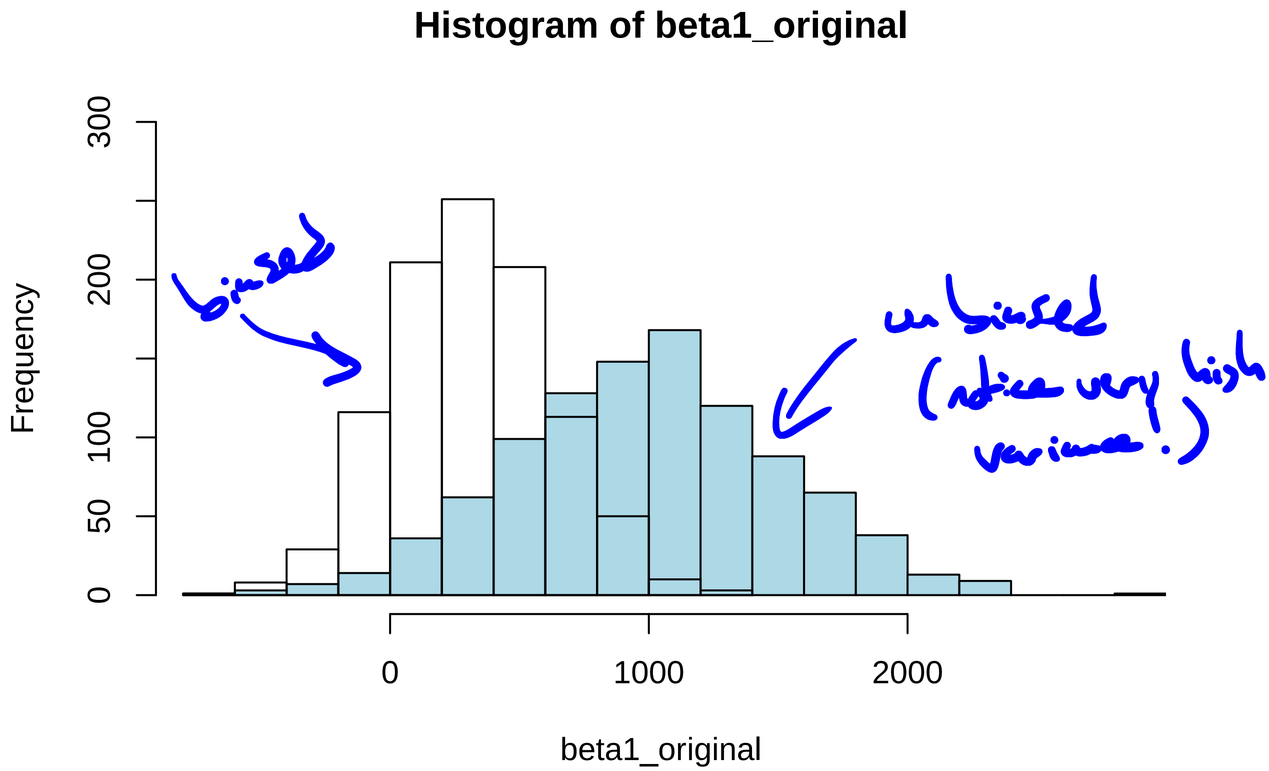
```
mean(beta1_biased)
```

```
## [1] 315.0093
```

```
hist(beta1_original,xlim=c(min(beta1_biased),max(beta1_original)),breaks=20,ylim=c(0,300),
     col="lightblue")
hist(beta1_biased,add=TRUE,breaks=10)
```

in general: confounders affect X & Y, should be included in model

post-treatment variable are affected by X, should not be included



Expanding the model

The previous model was simple. Here's an expanded model using a random "batch" effect.

$CFU_i = \beta_0 + \beta_1 PH14_i + \alpha_j + \varepsilon_i$

$\varepsilon_i \sim Normal(0, \sigma_\varepsilon)$

$\alpha_j \sim Normal(0, \sigma_\alpha)$ $j = 1,2,3$

this is a mixed model

Here, α_j is the random deviation from the overall mean for the j^{th} batch. It is drawn from a normal distribution with mean zero and standard deviation σ_α

```
n <- 60
batch <- rep(1:3,each=n/3)
PH14 <- rep(rep(0:1,each=n/6),3)
beta_0 <- 10000
beta_1 <- 1000
sigma_e <- 2000 #standard deviation for CeMio
sigma_a <- rnorm(n=3,mean=0,sd=2000)
epsilon <- rnorm(0,sigma_e,n=n)
alpha <- c(rep(sigma_a[1],n/3),rep(sigma_a[2],n/3),rep(sigma_a[3],n/3))
  for (j in 1:n) {
     CFU[j] <- beta_0+beta_1*PH14[j]+epsilon[j]+alpha[j]
  }
  model <- lmer(CFU~factor(PH14)+(1|alpha))
```

(factor(PH14)|alpha) → would fit random slopes

↑ random intercept (1)

(random thing | random across values of)

```
## boundary (singular) fit: see ?isSingular
```

not great but not fatal. Won't effect fixed effect estimates, random effect variance estimates unreliable

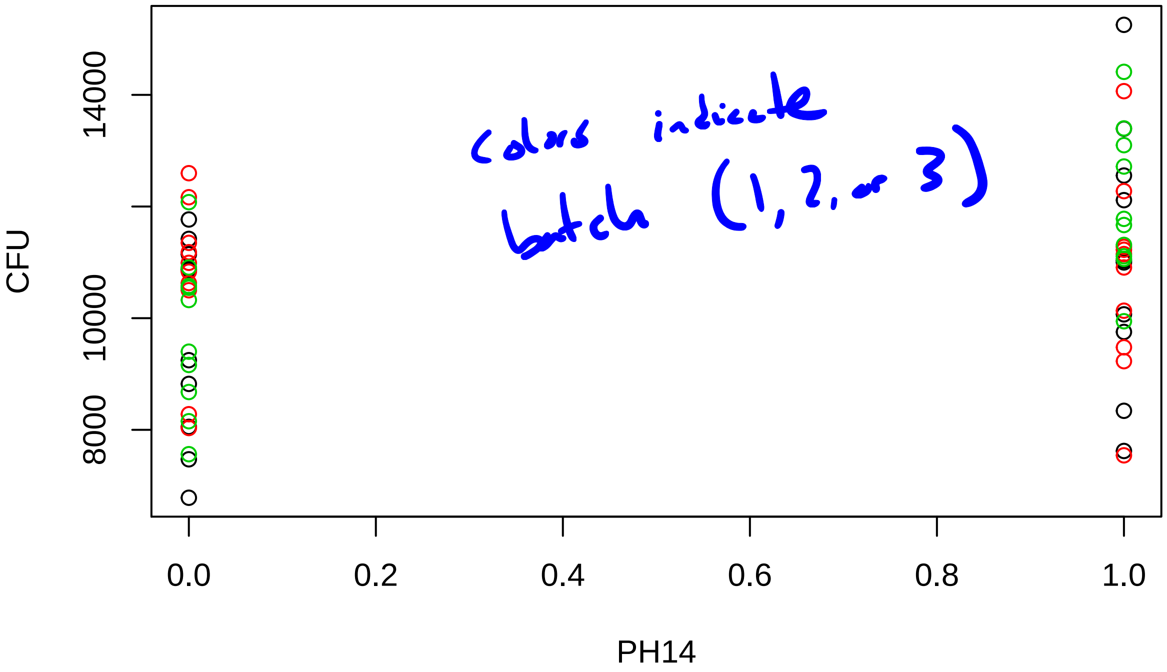
```
summary(model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: CFU ~ factor(PH14) + (1 | alpha)
##
## REML criterion at convergence: 1036.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.1650 -0.7239  0.0942  0.6575  2.2819
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  alpha    (Intercept)       0       0
##  Residual             3006779    1734
## Number of obs: 60, groups:  alpha, 3
##
## Fixed effects:
##                Estimate Std. Error t value
## (Intercept)     10013.3      316.6  31.629
## factor(PH14)1    1284.1      447.7   2.868
##
## Correlation of Fixed Effects:
##               (Intr)
## fctr(PH14)1 -0.707
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

(n=3 for estimating this variance. More replicates/ batches would help)

estimate is within the std error

```
plot(CFU~PH14,col=(batch))
```



colors indicate batch (1, 2, or 3)

- We can now to a power analysis. I'm reducing the number of reps, as running lmer is much more computationally intensive than running lm.

generative mixed model

```
n <- 60
batch <- rep(1:3,each=n/3)
PH14 <- rep(rep(0:1,each=n/6),3)
beta_0 <- 10000
beta_1 <- 1000
sigma_e <- 2000 #standard deviation for CeMio
sigma_a <- rnorm(n=3,mean=0,sd=3000)
reps <- 10
pvals <- rep(0,reps)
for (i in 1:reps) {
  epsilon <- rnorm(0,sigma_e,n=n)
  alpha <- c(rep(sigma_a[1],n/3),rep(sigma_a[2],n/3),rep(sigma_a[3],n/3))
  for (j in 1:n) {
     CFU[j] <- beta_0+beta_1*PH14[j]+epsilon[j]+alpha[j]
  }
  model <- lmer(CFU~factor(PH14)+(1|alpha))
#  pvals[i] <- round(summary(model)$coefficients[2,5],4)
}
#sum(pvals<0.05)/reps
```

remove to run power analysis