PHP 2514 HW #5

Blain Morin

April 16, 2018

Question 1: Gellman Hill Chapter 7 #2

According to the textbook, 52% of the United States is female. So, we will run our simulation using this proportion.

```
### Choose our number of simulations:
n.sims = 1000000

### Initiate a results vector:
totalweight = c()


### Run simulation using a for loop:


for (i in 1:n.sims) {

  sex = rbinom(10, 1, .52) ### Determines gender of the ten people (1 = Female)

  weight = ifelse(sex == 1,
                  rnorm(10, mean = 4.96, sd = .2), ### Weight distribution for female
                  rnorm(10, mean = 5.13, sd = .17)) ### Weight distribution for male

  result = sum(exp(weight)) ### Get weight in pounds by exponentiating and add them all t
ogether

  totalweight[i] = result ### Tally our simulation results

}

breaks = sum(totalweight >1750) ### Count the number of cases where the elevator breaks

breaks / n.sims ### Probability of failure
```

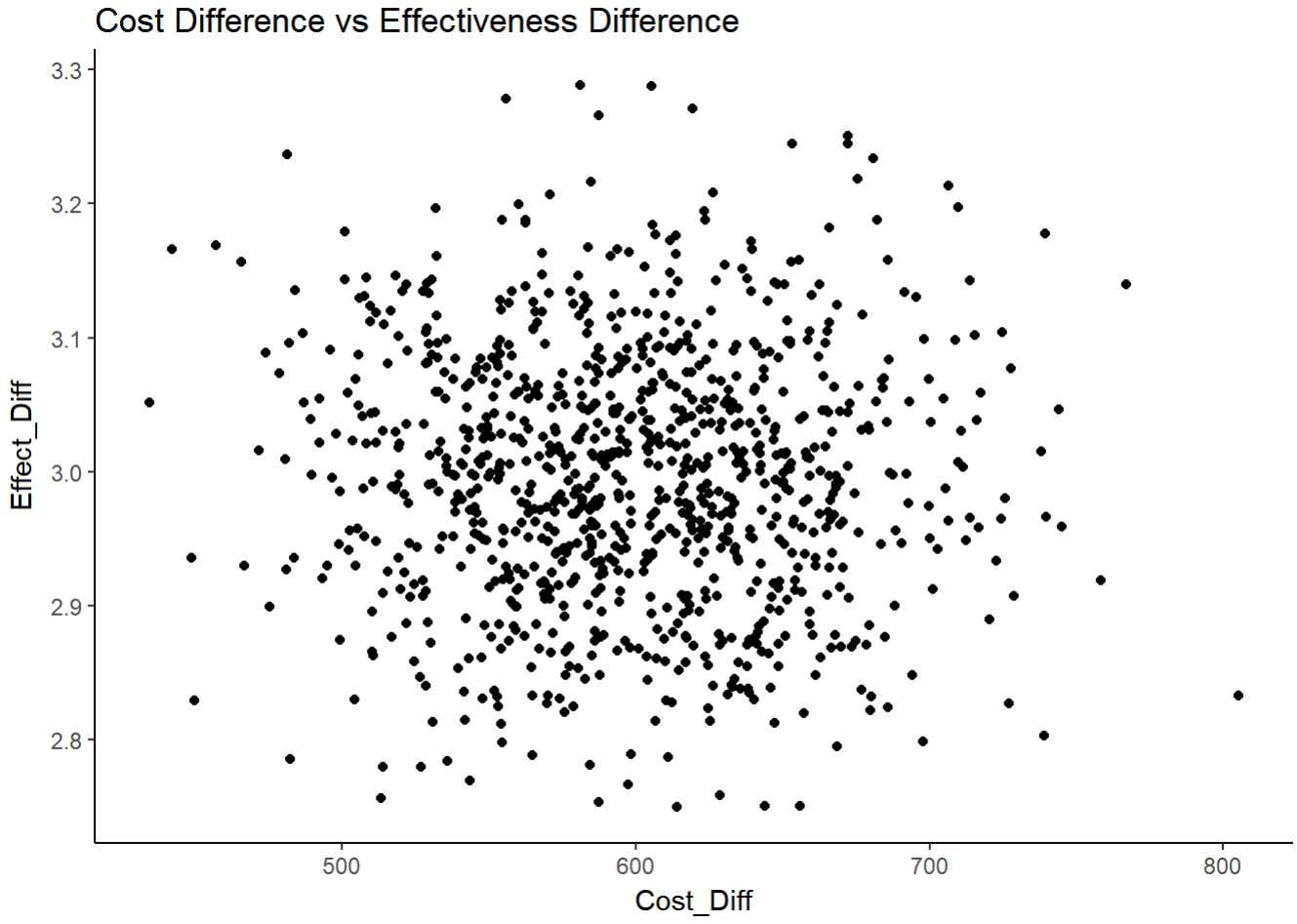
```
## [1] 0.050352
```

After simulating a million times, we find that the probability that the elevator breaks is .05.

2: Gelman Hill Chapter 7 #8

(a)

```
library(dplyr)
library(ggplot2)

### 1000 Cost difference draws
cost.difference = rnorm(1000, 600, sd = 400/(52**.5))

### 1000 Effectiveness difference draws
effect.diff <- rnorm(1000, 3, sd = 1/(102**.5))

### Create data frame
data.2a = data.frame(Cost_Diff = cost.difference, Effect_Diff = effect.diff)


### Create Scatterplot
data.2a %>% ggplot(aes(x = Cost_Diff, y = Effect_Diff)) + geom_point() +
  theme_classic() + ggtitle("Cost Difference vs Effectiveness Difference")
```



(b)

```
### Initiate matrices
costs.diff = c()
effects.diff = c()

### Draw cost and effectiveness
for(i in 1:1000) {
  costs.diff = rbind(costs.diff, rnorm(1000, 600, sd = 400/(52**.5)))
  effects.diff = rbind(effects.diff, rnorm(1000, 3, sd = 1/(102**.5)))
}

ratio = costs.diff/effects.diff

### Point estimate
estimate = mean(ratio)


### 50% CI
confidence.50 = quantile(ratio, c(.25, .75))


### 95% CI
confidence.95 = quantile(ratio, c(.05, .95))


estimate
```

```
## [1] 200.2194
```

```
confidence.50
```

```
##      25%      75%
## 186.8737 213.3105
```

```
confidence.95
```

```
##       5%      95%
## 168.2455 232.9244
```

3: Gelman Hill Chapter 8 #1

(a)

```
### Create predictors
x1 = 1:100
x2 = rbinom(100, 1, .5)

### This is the true model
y = 3 + .1 * x1 + .5 * x2 + rnorm(100, 0, 1)

### Regress y on our created predictors
model.3a = lm(y ~ x1 + x2)
summary.3a = summary(model.3a)

### Extract betas and se of betas
betas.3a = summary.3a$coefficients[ , 1]
beta.error.3a = summary.3a$coefficients[ , 2]

### Create CI
confidence.68 = cbind(betas.3a - beta.error.3a, betas.3a + beta.error.3a)

confidence.68
```

```
##                   [,1]      [,2]
## (Intercept) 2.63466893 3.0730517
## x1          0.09918198 0.1060648
## x2          0.57077470 0.9682160
```

In this run, the true parameters are all within the 68% intervals.

(b)

```
### Set number of simulations to 1000
n.sims = 1000

### Initialize a matrix for the parameter estimates
b.cover = matrix(nrow = 1000, ncol = 3)

### Run the simulation
for (i in 1:n.sims) {

  b.x1 = 1:100
  b.x2 = rbinom(100, 1, .5)
  y.b = 3 + .1 * b.x1 + .5 * b.x2 + rnorm(100, 0, 1)
  b.model = lm(y.b ~ b.x1 +b.x2)
  summary.3b = summary(b.model)
  est.3b = summary.3b$coefficients[ , 1]
  err.3b = summary.3b$coefficients[ , 2]

  ### Set the 68% bounds
  lower.bound = est.3b - err.3b
  upper.bound = est.3b + err.3b

  ### These statements test if the true parameters are within the bounds
  int.b = ifelse(3 >= lower.bound[1] && 3 <= upper.bound[1], TRUE, FALSE)
  x1.beta = ifelse(.1 >= lower.bound[2] && .1 <= upper.bound[2], TRUE, FALSE)
  x2.beta = ifelse(.5 >= lower.bound[3] && .5 <= upper.bound[3], TRUE, FALSE)

  ### Add the test results to the coverage matrix
  b.cover[i , 1] = int.b
  b.cover[i , 2] = x1.beta
  b.cover[i , 3] = x2.beta

}

### The coverage probability is the mean
inter.prob = mean(b.cover[ , 1])
x1.prob = mean(b.cover[ , 2])
x2.prob = mean(b.cover[, 3])

### Report results in a table
probs = cbind(inter.prob, x1.prob, x2.prob)

kable(probs)
```

inter.prob	x1.prob	x2.prob
0.673	0.68	0.706

Thus, the probability that the 68% confidence interval covers the true parameter is .68 for the intercept, .67 for the beta on x1, and .69 for the beta on x2. These are all very close to the expected probability of .68.

(c)

```
### Set number of simulations to 1000
n.sims = 1000

### Initialize a matrix for the parameter estimates
c.cover = matrix(nrow = 1000, ncol = 3)

### Run the simulation
for (i in 1:n.sims) {

  c.x1 = 1:100
  c.x2 = rbinom(100, 1, .5)
  y.c = 3 + .1 * c.x1 + .5 * c.x2 + rt.scaled(100, df = 4, mean = 0, sd = 5)
  c.model = lm(y.c ~ c.x1 +c.x2)
  summary.3c = summary(c.model)
  est.3c = summary.3c$coefficients[ , 1]
  err.3c = summary.3c$coefficients[ , 2]

  ### Set the 68% bounds
  lower.bound = est.3c - err.3c
  upper.bound = est.3c + err.3c

  ### These statements test if the true parameters are within the bounds
  int.c = ifelse(3 >= lower.bound[1] && 3 <= upper.bound[1], TRUE, FALSE)
  x1.c = ifelse(.1 >= lower.bound[2] && .1 <= upper.bound[2], TRUE, FALSE)
  x2.c = ifelse(.5 >= lower.bound[3] && .5 <= upper.bound[3], TRUE, FALSE)

  ### Add the test results to the coverage matrix
  c.cover[i , 1] = int.c
  c.cover[i , 2] = x1.c
  c.cover[i , 3] = x2.c

}

### The coverage probability is the mean
inter.prob.c = mean(c.cover[ , 1])
x1.prob.c = mean(c.cover[ , 2])
x2.prob.c = mean(c.cover[, 3])

### Report results in a table
probs.c = cbind(inter.prob.c, x1.prob.c, x2.prob.c)

kable(probs.c)
```

inter.prob.c	x1.prob.c	x2.prob.c
0.663	0.647	0.677

All of the coverage probabilities are near .68. Thus, having errors with a t distribution does not affect our coverage probabilities.

4: Gelman Hill Chapter 8 #4

(a)

```
library(readr)
risky = read_csv("risky.csv")[-1]
risky$fupacts = round(risky$fupacts)
attach(risky)

### Run poisson regression
pois.model = glm(fupacts ~ bs_hiv, family = poisson)

### Set number of simulations
n.sims = 1000

y = risky$fupacts
n = length(y)
x = cbind(as.numeric(1), as.factor(bs_hiv))


### Run simulation
sim1 = sim(pois.model, n.sims)
y.rep = array(NA, c(n.sims, n))

for (i in 1:n.sims) {

  yhat = exp(x %*% sim1@coef[i,])
  y.rep[i,] = rpois(n, yhat)

}


### Function for proportion of 0 counts
test.0 = function(x) {
  mean(x==0)
}


### Function for proportion of counts greater than 10
test.10 = function(x) {
  mean(x >10)
}


### Simulated proportion of 0s
rep.0 = rep(NA, n.sims)
for(i in 1:n.sims) {
  rep.0[i] = test.0(y.rep[i])
}

pred.0 = mean(rep.0)
```

```
### Simulated proportion of greater than 10
rep.10 = rep(NA, n.sims)
for(i in 1:n.sims) {
  rep.10[i] = test.10(y.rep[i])
}

pred.10 = mean(rep.10)

### Observed proportion of counts = 0
obs.0 = test.0(fupacts)

### Observed proportion of counts greater than 10
obs.10 = test.10(fupacts)


### Make table
predicteds = c(pred.0, pred.10)
observed = c(obs.0, obs.10)

answer = data.frame(predicteds, observed)

kable(answer)
```

predicteds	observed
0.000	0.2926267
0.417	0.3617512

We see from the above table that the model predicts no 0 counts, but 29% of the observed data is 0. This is a signal that there may be something wrong with our model.

(b)

```
### Run quasipoisson regression
qpois.model = glm(fupacts ~ bs_hiv, family = quasipoisson)

### Set number of simulations
n.sims = 1000

y = risky$fupacts
n = length(y)
x = cbind(as.numeric(1), as.factor(bs_hiv))


### Run simulation
sim1 = sim(qpois.model, n.sims)
y.rep = array(NA, c(n.sims, n))

for (i in 1:n.sims) {

  yhat = exp(x %*% sim1@coef[i,])
  y.rep[i,] = rpois(n, yhat)

}


### Function for proportion of 0 counts
test.0 = function(x) {
  mean(x==0)
}


### Function for proportion of counts greater than 10
test.10 = function(x) {
  mean(x >10)
}


### Simulated proportion of 0s
rep.0 = rep(NA, n.sims)
for(i in 1:n.sims) {
  rep.0[i] = test.0(y.rep[i])
}

pred.0 = mean(rep.0)


### Simulated proportion of greater than 10
rep.10 = rep(NA, n.sims)
for(i in 1:n.sims) {
  rep.10[i] = test.10(y.rep[i])
```

```
}

pred.10 = mean(rep.10)

### Observed proportion of counts = 0
obs.0 = test.0(fupacts)

### Observed proportion of counts greater than 10
obs.10 = test.10(fupacts)


### Make table
predicteds = c(pred.0, pred.10)
observed = c(obs.0, obs.10)

answer = data.frame(predicteds, observed)

kable(answer)
```

predicteds	observed
0.000	0.2926267
0.414	0.3617512

As you can see in the table above, the quasipoisson model does not seem to correct our lack of zeros in our predictions.

(c)

```
### Run quasipoisson regression with additional predictors
qpois.model2 = glm(fupacts ~ bs_hiv + bupacts, family = quasipoisson)

### Set number of simulations
n.sims = 1000

y = risky$fupacts
n = length(y)
x = cbind(as.numeric(1), as.factor(bs_hiv), bupacts)


### Run simulation
sim1 = sim(qpois.model2, n.sims)
y.rep = array(NA, c(n.sims, n))

for (i in 1:n.sims) {

  yhat = exp(x %*% sim1@coef[i,])
  y.rep[i,] = rpois(n, yhat)

}


### Function for proportion of 0 counts
test.0 = function(x) {
  mean(x==0)
}


### Function for proportion of counts greater than 10
test.10 = function(x) {
  mean(x >10)
}


### Simulated proportion of 0s
rep.0 = rep(NA, n.sims)
for(i in 1:n.sims) {
  rep.0[i] = test.0(y.rep[i])
}

pred.0 = mean(rep.0)


### Simulated proportion of greater than 10
rep.10 = rep(NA, n.sims)
for(i in 1:n.sims) {
  rep.10[i] = test.10(y.rep[i])
```

```
}

pred.10 = mean(rep.10)

### Observed proportion of counts = 0
obs.0 = test.0(fupacts)

### Observed proportion of counts greater than 10
obs.10 = test.10(fupacts)


### Make table
predicteds = c(pred.0, pred.10)
observed = c(obs.0, obs.10)

answer = data.frame(predicteds, observed)

kable(answer)
```

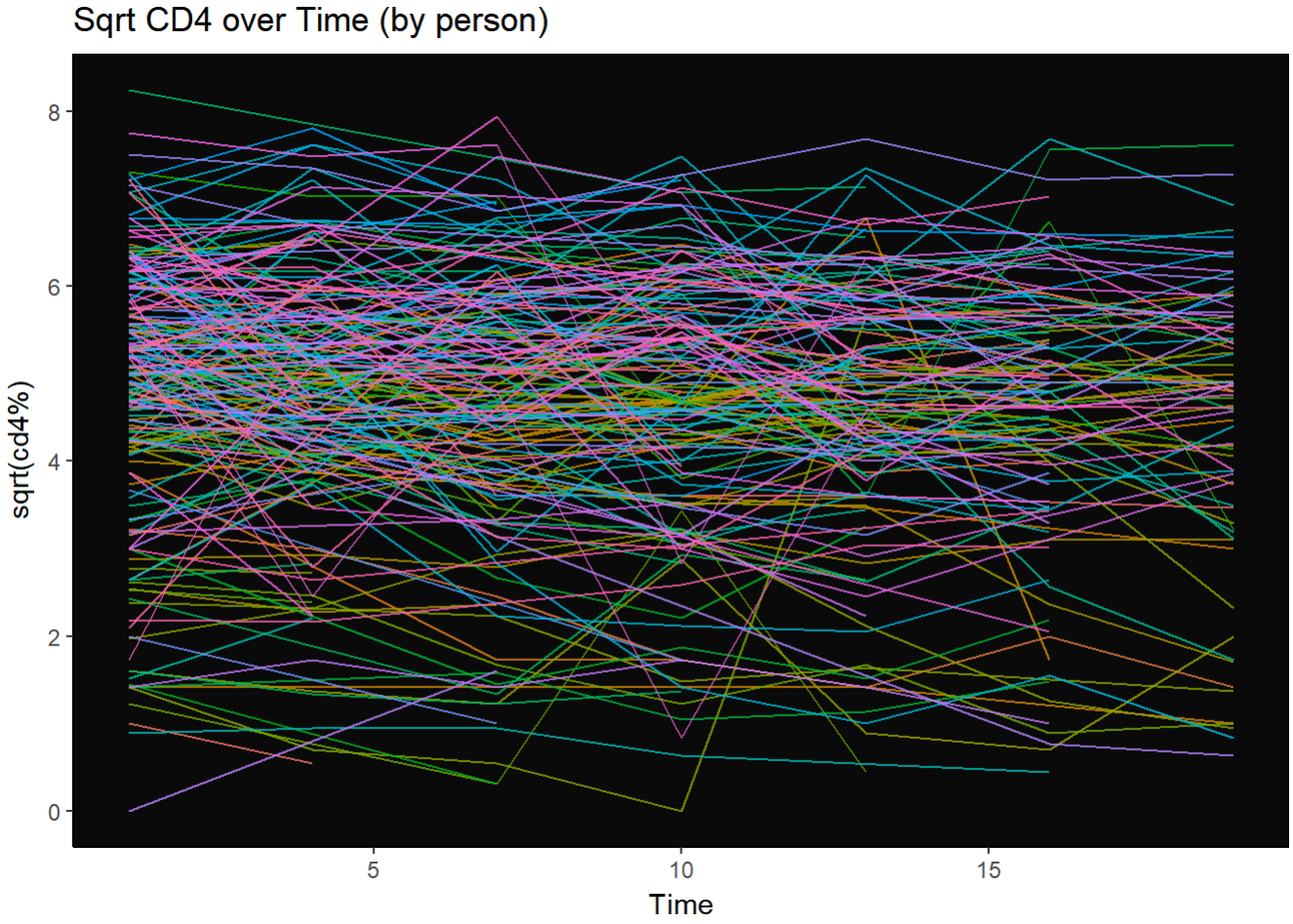
predicteds	observed
0.003	0.2926267
0.227	0.3617512

Adding the additional predictor did not help improve the lack of zeros in our predictions.

5: Gelman Hill Chapter 11 #4

(a)

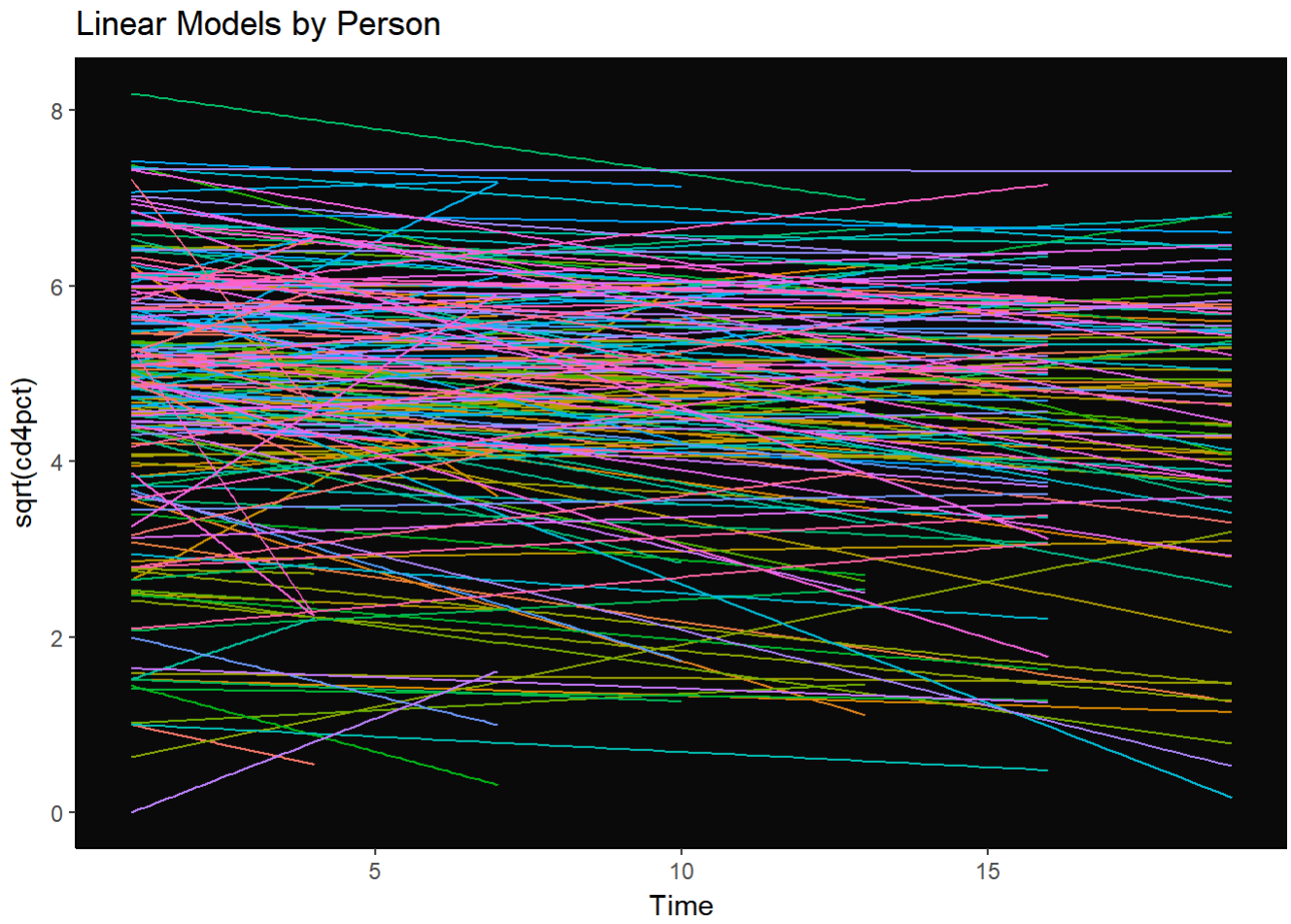
```
library(readr)
library(lubridate)
library(ggplot2)
library(dplyr)

### Load data and change the string to a date
cd4 = read_csv("cd4.csv")
cd4$vdate = mdy(cd4$vdate)
cd4.complete = cd4[complete.cases(cd4),]

### Create the plot
cd4.complete %>%
  ggplot(aes(x = visitno, y = cd4pct**.5)) + geom_line(aes(color = as.factor(newpid)), al
pha = .8)+ ylab("sqrt(cd4%)") +
  xlab("Time") + theme_classic() + theme(legend.position = "none") + ggtitle("Sqrt CD4 ov
er Time (by person)") +
  theme(panel.background = element_rect(fill = "gray4"))
```



We see from the above plot that there is lots of variation in the starting CD4 percentages and many different kinds of trajectories over time.

(b)

```
cd4.complete %>%
  ggplot(aes(x = visitno, y = cd4pct**.5)) +
  geom_smooth(method = "lm", se = FALSE, aes(color = as.factor(newpid)), alpha = .6, size
 = .5) +
  theme_classic() + theme(panel.background = element_rect(fill = "gray4")) +
  theme(legend.position = "none") +
  ylab("sqrt(cd4pct)") +
  xlab("Time") +
  ggtitle("Linear Models by Person")
```



The above plot shows the variation in the slopes for the individual regressions. We see that some are strongly negative while others are strongly positive.

(c)

```
library(lattice)
library(lme4)
library(sjPlot)
attach(cd4.complete)

###remove conflicting package
detach(package:metRology)

### Run no pooling regressions for each individual
no.pool = lmList(cd4pct**.5 ~ visitno | newpid, data = cd4.complete)

detach(cd4.complete)

### Condense data so we have one line per individual
cd4.slice = cd4.complete %>% group_by(newpid) %>% slice(1)

### Combine into df with slope, intercepts, baseage, and treatment
no.pool.df = as.data.frame(coef(no.pool))
no.pool.df = cbind(no.pool.df, cd4.slice$baseage)
no.pool.df = cbind(no.pool.df, as.factor(cd4.slice$treatmnt))
names(no.pool.df) = c("intercept", "slope", "baseage", "treatmnt")
attach(no.pool.df)

### Run model for intercept
intercept.model = lm(intercept ~ baseage + treatmnt)

### Run model for slope
slope.model = lm(slope ~ baseage + treatmnt)

sjt.lm(intercept.model, slope.model)
```

	intercept			slope		
	B	*CI*	*p*	*B*	*CI*	*p*
(Intercept)	5.12	4.72 – 5.52	<.001	-0.03	-0.07 – 0.01	.184
baseage	-0.12	-0.21 – -0.04	.006	-0.00	-0.01 – 0.01	.777
treatmnt (2)	0.25	-0.15 – 0.65	.214	-0.01	-0.05 – 0.04	.804
Observations	226			202		
R^2 / adj. R^2	.040 / .031			.001 / -.009		

```
detach(no.pool.df)
```

We see from the above table that base age has a significant effect on the intercept, but treatment group does not. We also see that neither variable has a significant effect on the slope.

6: Gelman Hill Chapter 12 #2:

(a)

```
attach(cd4.complete)

vary.intercept = lmer(cd4pct ~ 1 + visitno + (1 | newpid))
sjt.lmer(vary.intercept)
```

	cd4pct		
	B	*CI*	*p*
Fixed Parts			
(Intercept)	25.73	24.04 – 27.41	<.001
visitno	-0.27	-0.35 – -0.18	<.001
Random Parts			
σ^2	50.482		
$\tau_{00,\ newpid}$	130.331		
N_{newpid}	226		
ICC_{newpid}	0.721		
Observations	978		
R^2 / $\Omega_0{}^2$	.781 / .776		

The coefficient for time is -0.27. This result means that for each additional follow up, average cd4 percentage decreases by .27 points.

(b)

```
vary.intercept2 = lmer(cd4pct ~ 1 + baseage + treatmnt + visitno + (1|newpid))
sjt.lmer(vary.intercept2)
```

	cd4pct		
	B	*CI*	*p*
Fixed Parts			
(Intercept)	25.97	20.63 – 31.32	<.001
baseage	-0.92	-1.61 – -0.24	.009
treatmnt	1.90	-1.21 – 5.01	.233

visitno	-0.26	-0.35 – -0.18	<.001
Random Parts			
σ^2	50.517		
$\tau_{00,\ newpid}$	126.001		
N_{newpid}	226		
ICC_{newpid}	0.714		
Observations	978		
R^2 / $\Omega_0{}^2$	.780 / .775		

The interpretation for time is the same as in part (a). For every year in base age, the initial cd4 percentage decreases by .92. The beta on treatment group means that relative to treatment group 1, treatment group 2 sees an average cd4 percentage that is initially 1.9 points higher.

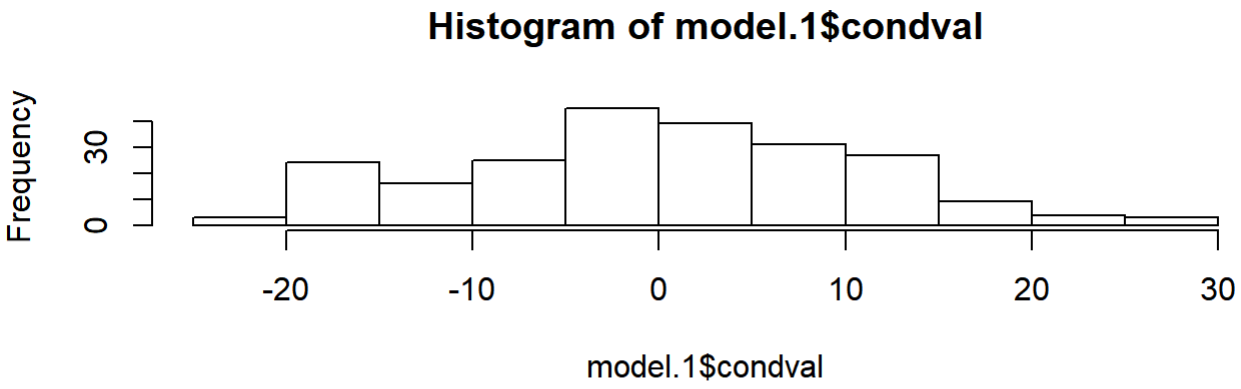
(c)

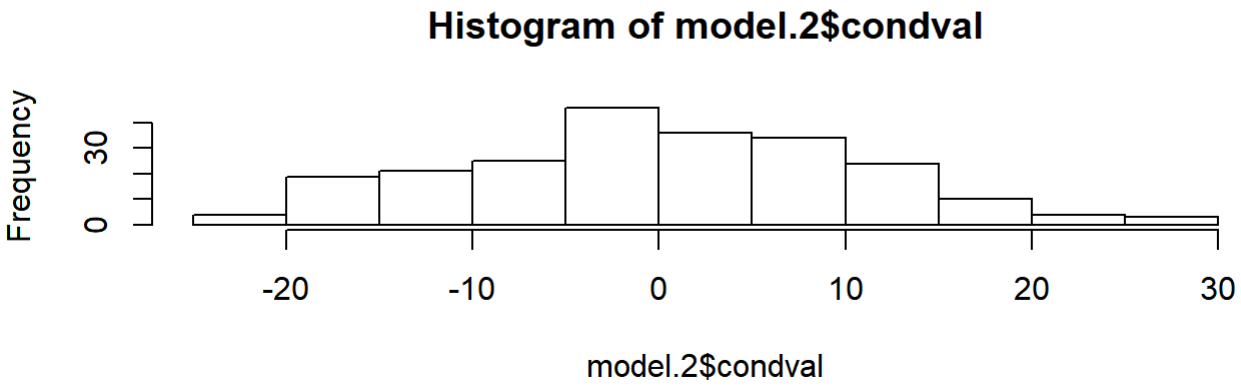
```
sjt.lmer(vary.intercept, vary.intercept2)
```

	cd4pct			cd4pct		
	B	*CI*	*p*	*B*	*CI*	*p*
Fixed Parts						
(Intercept)	25.73	24.04 – 27.41	<.001	25.97	20.63 – 31.32	<.001
visitno	-0.27	-0.35 – -0.18	<.001	-0.26	-0.35 – -0.18	<.001
baseage				-0.92	-1.61 – -0.24	.009
treatmnt				1.90	-1.21 – 5.01	.233
Random Parts						
σ^2	50.482			50.517		
$\tau_{00,\ newpid}$	130.331			126.001		
N_{newpid}	226			226		
ICC_{newpid}	0.721			0.714		
Observations	978			978		
R^2 / $\Omega_0{}^2$	.781 / .776			.780 / .775		

We see from the table above that the pooling difference is small between the two models. The random intercept's variance changes very little (sigma squared in the table).

```
### Make histograms of the random effects
par(mfrow=c(2,1))

model.1 = as.data.frame(ranef(vary.intercept))
hist(model.1$condval)

model.2 = as.data.frame(ranef(vary.intercept2))
hist(model.2$condval)
```





The above histograms show that the random intercepts of the two models have similar distributions. Since they are the only random effects, we can expect that there is not a large difference in the partial pooling.

(d)

```
anova(vary.intercept, vary.intercept2)
```

```
## Data: NULL
## Models:
## vary.intercept: cd4pct ~ 1 + visitno + (1 | newpid)
## vary.intercept2: cd4pct ~ 1 + baseage + treatmnt + visitno + (1 | newpid)
##                 Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## vary.intercept   4 7155.5 7175.1 -3573.8   7147.5
## vary.intercept2  6 7151.2 7180.6 -3569.6   7139.2 8.2773      2    0.01594
##
## vary.intercept
## vary.intercept2 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since our p value is .01, there is evidence to suggest that the more complicated model with the group level predictors gives us a better fit.

7: Gelman Hill Chapter 12 #3

(a)

```
### Going to predict for a follow up visit #20


cd4.slice.pred = cd4.slice
cd4.slice.pred$visitno = 20

preds = predict(vary.intercept2, cd4.slice.pred)

head(preds)
```

```
##          1          2          3          4          5          6
## 15.8836588  0.2217388 28.4102169 25.6122005 14.1212680 23.1636571
```

```
summary(preds)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.033  13.704  20.542  20.441  27.433  49.699
```

Since there is no bound at 0, some of our predictions are negative. This may suggest that using a logistic regression would be a better model.

(b)

```
detach(cd4.complete)

### Create a data frame with all possible combinations of independent variables, for each
 time period

combos = data.frame(visitno = rep(1:19, each = 2))
combos = cbind(combos, baseage = 4)
combos = cbind(combos, treatmnt = as.factor(rep(1:2)))
combos = cbind(combos, newpid = rep(1:2))

### Predict for each row, this gives us a prediction for a 4 year old child
### for every date and treatment group

preds2 = predict(vary.intercept2, combos)

head(preds2)
```

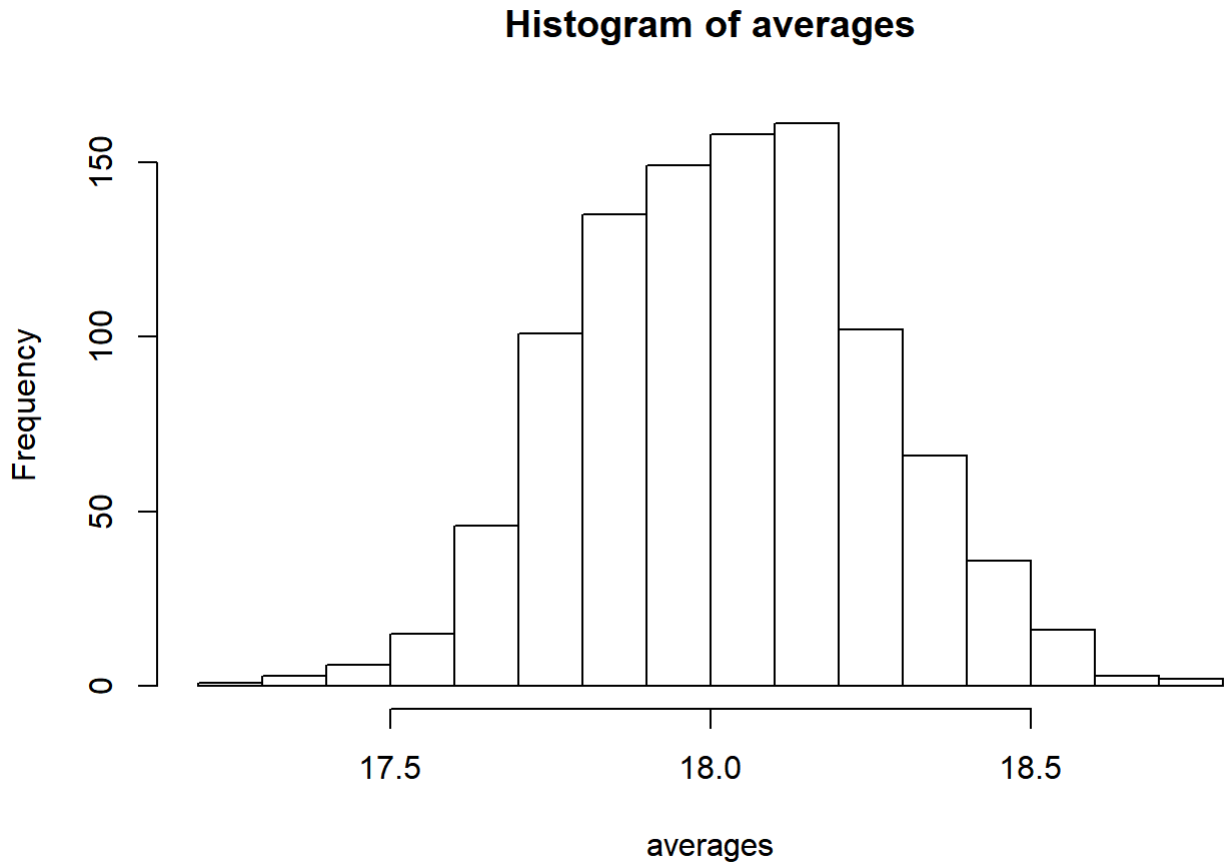
```
##         1         2         3         4         5         6
## 18.929369  2.948779 18.664757  2.684167 18.400145  2.419555
```

```
summary(preds2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.8142  0.6334  8.5576  8.5576 16.4817 18.9294
```

8: Gelman Hill Chapter 12 #4

```
### Find observed mean at final time

final = cd4.complete %>% filter(visitno == 19)

obs.final.mean = mean(final$cd4pct)


### Simulate 1000 new data sets

averages = c()

n.sims = 1000
n = nrow(final)

for(i in 1:n.sims){

  tempdata = data.frame(
    baseage = rnorm(n = n, mean = mean(final$baseage), sd = sd(final$baseage)),
    treatmnt = rbinom(n = n, size = 1, prob = (mean(final$treatmnt) - 1)),
    visitno = 19,
    newpid = 1:n
  )

  tempdata$treatmnt = tempdata$treatmnt + 1

  ### Predict CD4 level using our varying intercept model
  temp.predict = predict(vary.intercept2, tempdata, allow.new.levels = TRUE)

  ### Tally our mean CD4 level
  averages[i] = mean(temp.predict)


}

hist(averages)
```




```
obs.final.mean
```

```
## [1] 22.96026
```

Our simulated data has a mean cd4 percentage around 18. As you can see from the histogram, it is pretty tight around 18, and does not contain the observed mean of 22.96. 18 is not horribly far off, but it does suggest that our model has some systematic problem. I would guess that our averages are lower than the observed because our model can predict negative percentages, which would pull the mean down. Thus, logistic regression may help our model fit.