Chapter 7 Simulation

With normally distributed data, the model coefficient estimates and $\mathbf{V}_\beta$ are a complete summary.
With other types of data, we could summarize with a simulation.
Sims are also used for

- prediction,
- model checking,
- and validation.

Especially useful for multi-level models.

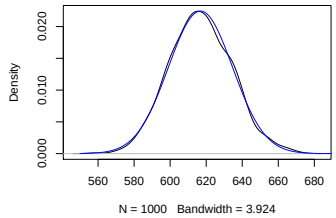
Predict Births

Bozeman Hospital delivered 1264 babies in 2011. Guess how many were girls. ($P(girl) = .488$)

Predict Births

Bozeman Hospital delivered 1264 babies in 2011. Guess how many were girls. ($P(girl) = .488$) Answer: I don't know. But we could simulate an answer AND look at the distribution we'd expect to see.

```
(n.girls <-  rbinom(1, 1264, .488))

## [1] 625

plot(density(n.girls <-  rbinom(1000, 1264, .488)), main="")
lines(550:700, dbinom(550:700,1264,.488), col=4)
```



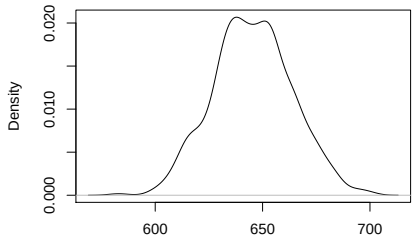
What about twins?

```
birth.type <- sample (c("frat twin", "id twin", "single"),
  size=1264, replace=TRUE, prob=c(1/25, 1/300, 1 - 13/300))
girls <- rep (NA, 1264)
for (i in 1:1264){
  if (birth.type[i] == "single"){
   girls[i] <- rbinom(1, 1, .488)}
  else if (birth.type[i]=="id twin"){
   girls[i] <- 2 * rbinom (1, 1, .495)}
  else if (birth.type[i] == "frat twin"){
   girls[i] <- rbinom (1, 2, .495)}
}
(n.girls <- sum (girls))

## [1] 631
```

`unclass(birth.type)` to select the column to use.

Distribution with twins

```
n.girls <- rep (NA, n.sims <- 1000)
for (s in 1:n.sims){
  birth.type <- sample (1:3,
   size=1264, replace=TRUE, prob=c(12, 1, 287)/300)
  girls <- cbind( rbinom(1264, 2, .495),
                  2*rbinom (1264,1, .495),
       rbinom (1264, 1, .488) )[cbind(1:1264, birth.type)]
  n.girls[s] <- sum (girls)
}
plot(density(n.girls), main="")
```



Simulation of a Continuous Variable

52% of adults are women: heights $\sim N(162, sd = 3.7)$cm
48% of adults are men: heights $\sim N(178, sd = 4.2)$cm
Average height of 30 adults (once).

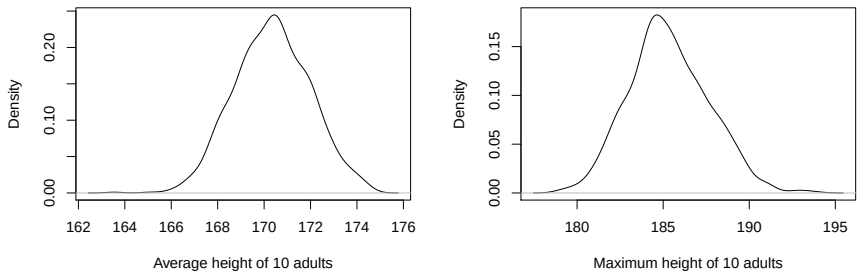
```
woman <- rbinom (30, 1, .52)  ## 30 people, men are 0's
mean(height <- ifelse (woman==0, rnorm (30,162, 3.7 ),
                                 rnorm (30,178, 4.2 )))

## [1] 170
```

Distribution – repeat 1000 times

```
avg.height <- max.height <- rep (NA, n.sims <- 1000) ## setup boxes
for (s in 1:n.sims){
  woman <- rbinom (30, 1, .52)
  height <- ifelse (woman ==0, rnorm (30, 162, 3.7 ), rnorm (30,178, 4.
  ## or  <- rnorm(30, 178 - 16* woman, 4.2 - .5*woman)
  avg.height[s] <- mean(height)
  max.height[s] <- max(height)
}
```


Densities of Avg height and Max height

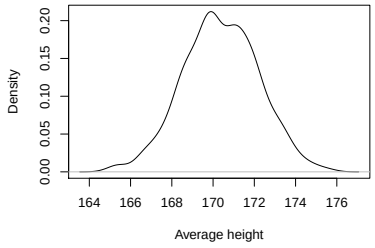
```
par(mfrow=c(1,2))
plot(density(avg.height), main="",xlab="Average height of 10 adults")
plot(density (max.height), main="",xlab="Maximum height of 10 adults")
```



With a function and replicate

A function for average of n random heights.

```
Height.sim <- function(n.adults){
  female <- rbinom (n.adults, 1, .52)
  height <- ifelse (female==0, rnorm (n.adults,162, 3.7 ), rnorm(n.adul
  return(mean(height))
}
avg.height <- replicate(1000, Height.sim(n.adults=25))
plot(density(avg.height),main="", xlab="Average height")
```



Simulation with Regression (via `predict`)

- Inferential uncertainty: SE's of coefficients and dist'n of residual error.
- Predictive uncertainty: Added variation of a new observation.

	Estimate	Std. Error	t value
(Intercept)	8.39	0.84	9.94
height	0.02	0.01	1.30
male	-0.08	1.26	-0.06
height:male	0.01	0.02	0.40

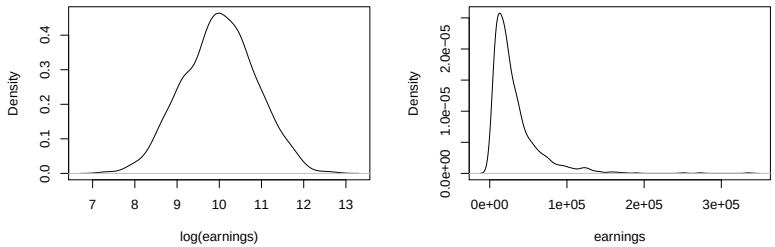
Table: n = 1192 rank = 4 resid sd = 0.881 R-Squared = 0.087

```
exp( pred.interval <- predict (logearn.fit3, newdata=
        data.frame (height=68, male=1), interval="pred"))

##     fit  lwr    upr
## 1 21436 3795 121089
```


Using a Simulation

Generate 1000 random draws from the predictive distribution with mean 9.97 and sd 0.88

```
par(mfrow=c(1,2));   randomPreds <- rnorm(1000, 9.97, 0.88)
plot(density(randomPreds), main = "", xlab="log(earnings)")
plot(density(exp(randomPreds)), main = "", xlab="earnings")
```



Note: this ignores the variability of the estimated mean, focuses on prediction error. Could also look at summaries and quantiles.

Why simulate?

For one level of random variation, the `predict` function suffices.
Assume normality, or large sample size and invoke CLT.
Then prediction intervals are based on a t_{n-k} distribution. No problem
What if we want to summarize differences in earnings between a 68" man and a 68" woman?
Create a linear combo of the coefficients, but then how to add prediction variation?

```
pred.man <- exp(rnorm(1000, 8.39 + 0.017*68 - 0.079*1 + .0074*68*1, .88
pred.woman <- exp(rnorm(1000, 8.39 + 0.017*68 - 0.079*0, .88))
pred.ratio <- pred.man/pred.woman
c(mean(pred.ratio), quantile(pred.ratio, c(.025,.25,.5,.75,.975)))

##          2.5%    25%    50%    75%  97.5%
##  3.260  0.111  0.661  1.376  3.255 17.991
```


Errors in Coefficient Estimates

We know $\beta|\sigma^2 \sim N(\widehat{\beta}, \sigma^2 \mathbf{V}_\beta)$ (if $\mathbf{X}$ is full column rank). The `sim` function in `arm` package does this for us:

1. Fit the regression to obtain $\widehat{\beta}$, $\mathbf{V}_\beta$, and s^2.
2. Draw a random χ^2_{n-k} and let $\sigma^2 = s^2(n-k)/\chi^2_{n-k}$.
3. Given that draw for variances, draw a random MVN$(\widehat{\beta}, \sigma^2 \mathbf{V}_\beta)$

```
sim.1 <- sim(logearn.fit3, 1000)
xtable(matrix(quantile( sim.1@coef[,2] + sim.1@coef[,4],
  c(.25,.25,.5,.75,.975)),1,dimnames=list(NULL, c("2.5%",
                                  "25%","50%","75%","97.5%"))
```

	2.5%	25%	50%	75%	97.5%
1	0.02	0.02	0.02	0.03	0.05

Do `str(sim.1)` to see its structure. it's an S4 object now.

Are We Bayesians Now?

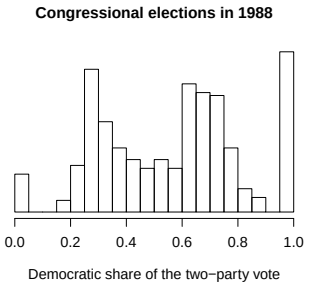
Bayesian methods simplify multilevel models.
But we're not there yet.

Bayesians view frequentist methods like regression as using a very diffuse prior. Combine prior with likelihood to get a posterior which answers all questions. With a diffuse prior, posterior is basically the likelihood. We all agree likelihood is important. It connects data to parameters.

Bayesians build "credible" intervals, not confidence intervals and can say the posterior probability that the parameter falls in this interval is 95%.

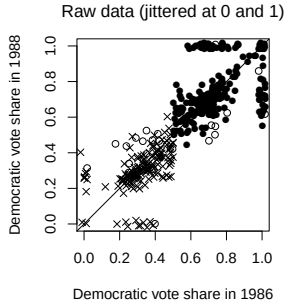
7.3 Predict Elections

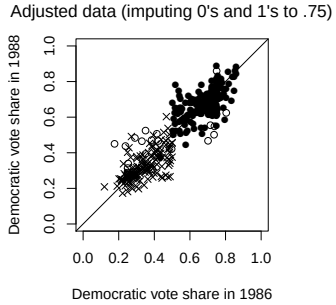
How well do 1986 election results predict 1988 election outcomes? Consider 435 races for Congressional House. Response: Vote share of Democrat in 1988. Predictors: Democrat vote share in 1986 and incumbency = 0 if neither candidate currently is the district representative, -1 if the Republican is, +1 if the Democrat is.



Zeroes and ones are uncontested races where predicting outcomes is silly.

1986 and 1988 Democrat Vote Share





Fit 1986 and 1988 Democrat Vote Share

	Estimate	Std. Error	t value
(Intercept)	0.20	0.02	11.10
vote.86	0.58	0.04	16.64
incumbency.88	0.08	0.01	10.95

Table: n = 343 rank = 3 resid sd = 0.067 R-Squared = 0.877

Change the $\mathbf{X}$ matrix to $\tilde{\mathbf{X}}$ for 1988 and simulate $\tilde{y}$ for 1990. But don't just use the $\widehat{\beta}$ above, we need to draw lots of β's from the sampling distribution.

Simulate 1990 Democrat Vote Share from 1988

If conditions in 1990 are like those of 1988, how many seats would the Dems win?
Create 1000 simulated datasets of predictions for 1990.

```
X.tilde <- cbind ( 1, v88, inc90)[complete.cases(v88),]
sim.88 <- sim (fit.88, 1000)
y.tilde <-  matrix(NA, 1000, nrow(X.tilde) )
for(s in 1:1000){
 y.tilde[s,] <- rnorm(nrow(X.tilde), X.tilde %*% sim.88@coef[s,], sim.8
summary(dems.tilde <- rowSums (y.tilde > .5, na.rm=TRUE))
sd(dems.tilde)
```


With a Function

```
X.tilde <- cbind ( 1, v88, inc90)[complete.cases(v88),]
Pred.88 <- function (X.pred, lm.fit){
  sim.88 <- sim (lm.fit, 1)
  pred <- X.pred %*% t(sim.88@coef)
  ok <- !is.na(pred)
  n.pred <- length (pred)
  y.pred <- rep (NA, n.pred)
  y.pred[ok] <- rnorm (sum(ok), pred[ok], sim.88@sigma)
  return (y.pred)
}
y.tilde <- replicate (1000, Pred.88(X.tilde, fit.88))
dems.tilde <- colSums(y.tilde > .5, na.rm = TRUE)
c(mean(dems.tilde), sd(dems.tilde))

## [1] 247.81   2.84
```


Matrix Ops are faster

```
X.tilde <- cbind ( 1, v88, inc90)[complete.cases(v88),]
sim.88 <- sim (fit.88, 1000)
ytilde.hat <- X.tilde %*% t(sim.88@coef)
ytilde <- rnorm(length(ytilde.hat), ytilde.hat, rep(sim.88@sigma, each
summary(dems.tilde <- colSums (y.tilde > .5, na.rm = TRUE))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     238     246     248     248     250     256

sd(dems.tilde)

## [1] 2.84
```

However you get there, the observed 262 seats for Democrats in 1990 are much higher than the mean of 248. That's because of national swings.

7.4 Back to a GLM Setting

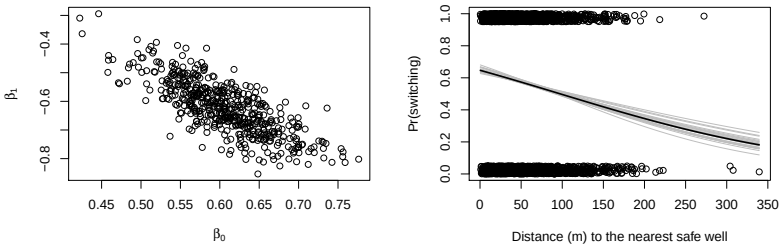
Bangladeshi well switching again. Coefficients have a normal distribution.

	Estimate	Std. Error	z value
(Intercept)	0.61	0.06	10.05
I(dist/100)	-0.62	0.10	-6.38

Table: n = 3020 rank = 2 Resid Deviance = 4076.238

Predict for GLM model

```
par(mfrow=c(1,2))
plot (sim.1@coef[,1:2], xlab=expression(beta[0]),  ylab=expression(beta
plot (wells$dist, jitt.switch, xlab="Distance (m) to the nearest safe w
for (s in 1:20){
  curve (invlogit (sim.1@coef[s,1] + sim.1@coef[s,2]*x/100), col="grey"
curve (invlogit (wells.fit1$coef[1] + wells.fit1$coef[2]*x/100), add=TR
```




```
par(mfrow=c(1,2))
plot (sim.1@coef[,1:2], xlab=expression(beta[0]),  ylab=expression(beta
plot (wells$dist, jitt.switch, xlab="Distance (m) to the nearest safe w
```


Predict 2 for GLM model

Predict the points.

```
X.tilde <- cbind (1, wells$dist/100)
y.tilde <- array (NA, c(500, 3020))
for (s in 1:500){
  p.tilde <- invlogit (X.tilde %*% sim.1@coef[s,])
  y.tilde[s,] <- rbinom (3020, 1, p.tilde)
}
output <- rbind(cbind(sim.1@coef[1:6, ], y.tilde[1:6,1:5]), c(apply(sim
2,mean), apply(y.tilde[,1:5],2,mean)))
dimnames(output) <- list( c(1:6,"means") ,c("beta_0","beta_1",paste("~y
```


Table and Latent Approach

	beta_0	beta_1	~y_1	~y_2	~y_3	~y_4	~y_5
1	0.58	-0.58	1.00	1.00	1.00	1.00	0.00
2	0.69	-0.72	0.00	1.00	0.00	1.00	1.00
3	0.71	-0.70	0.00	1.00	1.00	1.00	1.00
4	0.61	-0.66	1.00	0.00	0.00	1.00	1.00
5	0.55	-0.54	1.00	1.00	0.00	1.00	1.00
6	0.55	-0.51	1.00	1.00	0.00	1.00	1.00
means	0.61	-0.62	0.60	0.56	0.63	0.64	0.56

```
  for (s in 1:500){
  epsilon.tilde <- logit(runif (3020, 0, 1))
  z.tilde <- X.tilde %*% sim.1@coef[s,] + epsilon.tilde
  y.tilde[s,] <- ifelse (z.tilde>0, 1, 0)
}
   apply(y.tilde[,1:5],2,mean)

## [1] 0.592 0.586 0.596 0.612 0.552
```


Compound Models

	Estimate	Std. Error	z value
(Intercept)	-3.76	2.07	-1.82
height	0.08	0.03	2.48
male	1.70	0.32	5.29

Table: n = 1379 rank = 3 Resid Deviance = 989.912

	Estimate	Std. Error	t value
(Intercept)	8.15	0.60	13.53
height	0.02	0.01	2.22
male	0.42	0.07	5.84

Table: n = 1192 rank = 3 resid sd = 0.881 R-Squared = 0.087

Earnings

Try ignoring simulation uncertainty.

```
n.sims <- 1000
prob.earn.pos <- invlogit (coef(earn.fit1a) %*% x.new)
earn.pos.sim <- rbinom (n.sims, 1, prob.earn.pos)
summary(earn.sim <- ifelse (earn.pos.sim==0, 0, exp (rnorm (n.sims, coe

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0   11000   20000   31100   39700  338000

sim.1a <- sim (earn.fit1a, n.sims)
sim.1b <- sim (earn.fit1b, n.sims)
prob.earn.pos <- invlogit (sim.1a@coef %*% x.new)
earn.pos.sim <- rbinom (n.sims, 1, prob.earn.pos)
summary(earn.sim <- ifelse (earn.pos.sim==0, 0, exp (rnorm (n.sims, sim

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0   10900   20500   29700   36200  291000
```


Earning Sim with Function

```
Mean.earn <- function (height, male, sim.a, sim.b){
  x.new <- c (1, height, male)
  prob.earn.pos <- invlogit (sim.a@coef %*% x.new)
  earn.pos.sim <- rbinom (n.sims, 1, prob.earn.pos)
  earn.sim <- ifelse (earn.pos.sim==0, 0,
    exp (rnorm (n.sims, sim.b@coef %*% x.new, sim.b@sigma)))
  return (mean (earn.sim))
}
heights <- seq (60, 75, 1)
mean.earn.female <- sapply (heights, Mean.earn, male=0, sim.1a, sim.1b)
mean.earn.male <- sapply (heights, Mean.earn, male=1, sim.1a, sim.1b)
```

These are plotted in Figure 7.8 for heights 60 to 75, with 100, 1000, 10 000 simulations.