Interpreting logistic models

Frank Edwards

3/10/2021

- Remember that logistic regression is a GLM with a logit link function

- Remember that logistic regression is a GLM with a logit link function
- A GLM takes the form: $g(y) = \mathbf{X}\beta$

- Remember that logistic regression is a GLM with a logit link function
- A GLM takes the form: $g(y) = \mathrm{X}\beta$
- Logistic regression is the special case where g is the logit function:
  $$\mathrm{logit}(y) = \mathrm{X}\beta$$

- Remember that logistic regression is a GLM with a logit link function
- A GLM takes the form: $g(y) = \mathrm{X}\beta$
- Logistic regression is the special case where g is the logit function: $\mathrm{logit}(y) = \mathrm{X}\beta$
- A logistic regression model returns $\mathrm{X}\beta$ on the logit scale

- Remember that logistic regression is a GLM with a logit link function
- A GLM takes the form: $g(y) = \mathrm{X}\beta$
- Logistic regression is the special case where g is the logit function: $\mathrm{logit}(y) = \mathrm{X}\beta$
- A logistic regression model returns $\mathrm{X}\beta$ on the logit scale
- How can we convert $\mathrm{x}\beta$ to something useful?

Let's return to the grad school admission example

```
admits<-read_csv("./data/binary.csv")
summary(admits)

##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

3

Huh, all this tells us is mean(admits) = 0.3175

```
hist(admits$admit)
```
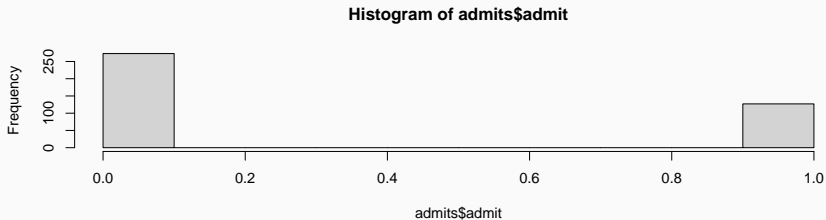
Histogram of admits$admit

4

- First, fit an intercept-only logistic regression model

```
m0<-glm(admit ~ 1, data = admits, family = "binomial")
m0_est<-tidy(m0)
```

- What does this model tell us?

```
m0_est$estimate ## log odds
```

```
## [1] -0.7652847
```

```
exp(m0_est$estimate) ## odds
```

```
## [1] 0.4652015
```

```
invlogit(m0_est$estimate) ## probability
```

```
## [1] 0.3175
```

```
mean(admits$admit) ## mean admission probability
```
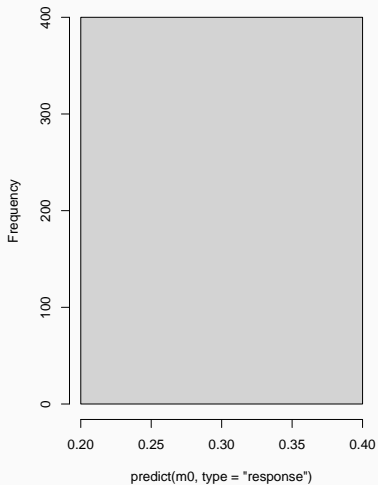
```
## [1] 0.3175
```

Let's add a predictor

```r
m1<-glm(admit ~ 1 + gre, data = admits, family = "binomial")
m1
```
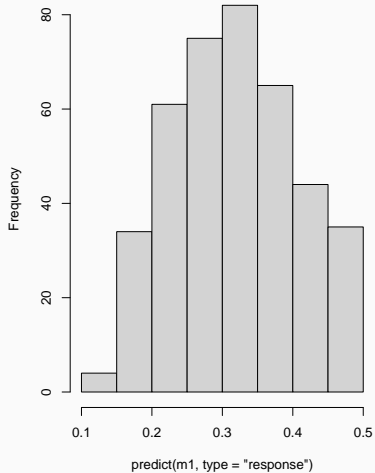
```
##
## Call:  glm(formula = admit ~ 1 + gre, family = "binomial", data
##
## Coefficients:
## (Intercept)          gre
##   -2.901344     0.003582
##
## Degrees of Freedom: 399 Total (i.e. Null);  398 Residual
## Null Deviance:        500
## Residual Deviance: 486.1     AIC: 490.1
```

Before and after - what's going on?

Intercept only

With GRE predictor
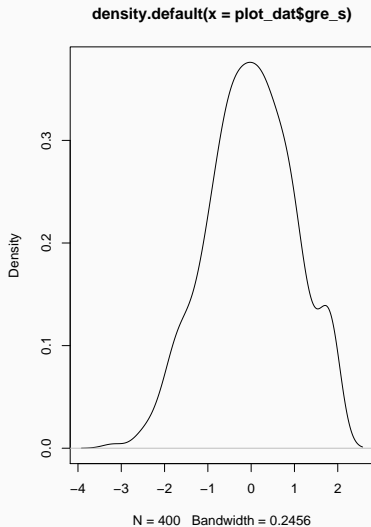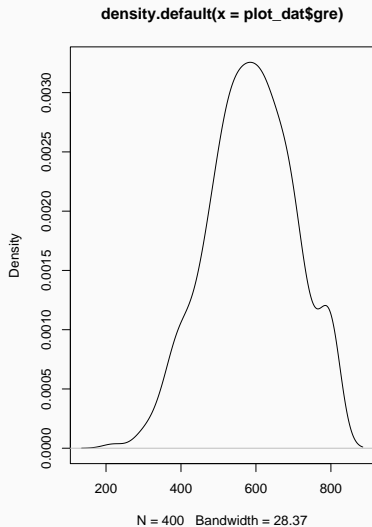
Why do these generate such different predictions?

Intercept only model (m0): $p(admit) = \beta_0$ With GRE predictor (m1): $p(admit) = \beta_0 + \beta_1 GRE_i$

Scale mean-centers and SD scales variables: $\mathrm{scale}(x_i) = \frac{x_i - \bar{x}}{sd(x)}$

Re-estimate the model: much nicer to look at

```
admits<-admits%>%
  mutate(gre_s = as.numeric(scale(gre)))
m1<-glm(admit ~ 1 + gre_s, data = admits, family = "binomial")
m1_est<-tidy(m1)
m1_est

## # A tibble: 2 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    -0.796     0.111     -7.20 6.01e-13
## 2 gre_s           0.414     0.114      3.63 2.80e- 4
```

Interpret the model

```
m1_est
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -0.796     0.111     -7.20 6.01e-13
## 2 gre_s          0.414     0.114      3.63 2.80e- 4
```

Remember: $\text{logit}(y) = X\beta = \log\left(\frac{y}{1-y}\right)$

So: $y = \text{logit}^{-1}(X\beta) = \frac{\exp(X\beta)}{\exp(X\beta)+1}$

```
m1_est
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -0.796     0.111     -7.20 6.01e-13
## 2 gre_s          0.414     0.114      3.63 2.80e- 4
```

Remember: $\text{logit}(y) = \text{X}\beta = \log\left(\frac{y}{1-y}\right)$

So: $y = \text{logit}^{-1}(\text{X}\beta) = \frac{\exp(\text{X}\beta)}{\exp(\text{X}\beta)+1}$

- What is β_0?

Interpret the model

```
m1_est
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -0.796     0.111     -7.20 6.01e-13
## 2 gre_s          0.414     0.114      3.63 2.80e- 4
```

Remember: $\text{logit}(y) = \mathrm{X}\beta = \log\left(\frac{y}{1-y}\right)$

So: $y = \text{logit}^{-1}(\mathrm{X}\beta) = \frac{\exp(\mathrm{X}\beta)}{\exp(\mathrm{X}\beta)+1}$

- What is β_0?
- What is β_1?

$$e^{y_1+y_2} = e^{y_1}e^{y_2}$$

Refresher on exponentials

$$e^{y_1 + y_2} = e^{y_1} e^{y_2}$$

and

$$e^{y_1 - y_2} = \frac{e^{y_1}}{e_2^y}$$

$$e^{y_1+y_2} = e^{y_1}e^{y_2}$$

and

$$e^{y_1-y_2} = \frac{e^{y_1}}{e_2^y}$$

so how can we rewrite:

$$\exp(\text{logit}(y)) = \frac{y}{1-y} = e^{\beta_0+\beta_1 x_1}$$

On the log scale, β_0 and β_1 are related to y multiplicatively because

$$e^{\beta_0 + \beta_1 x_1} = e^{\beta_0} e^{\beta_1 x_1}$$

Odds are defined as the probability of the event occurring divided by the probability of probability of the event not occurring. To obtain odds in a logistic regression, we exponentiate both sides:

$$\frac{y}{1-y} = e^{\beta_0 + \beta_1 x_1}$$

Odds are defined as the probability of the event occurring divided by the probability of probability of the event not occurring. To obtain odds in a logistic regression, we exponentiate both sides:

$$\frac{y}{1-y} = e^{\beta_0 + \beta_1 x_1}$$

The odds of $y == 1$ are simply $e^{X\beta}$

The odds ratio is the ratio of two odds - or the proportional change in odds. We can obtain an isolated estimate for the relationship between $\beta_1 x_{1i}$ and y this way:

$$\frac{Odds(y|x_1 = 1)}{Odds(y|x_1 = 0)} = \frac{e^{X\beta + \beta_1}}{e^{X\beta}} = \frac{e^{X\beta} \times e^{\beta_1}}{e^{X\beta}} = e^{\beta_1}$$

The odds ratio can be interpreted as the change in odds of $y == 1$ for a one-unit change in x_1.

- Odds ratios appear convenient - e^{β_1} is a percent change in y for a one-unit change in x_1

- Odds ratios appear convenient - e^{β_1} is a percent change in y for a one-unit change in x_1

How do they work?

In our example: what do these figures mean?

```
new_dat<-c(1,0) # for scale(gre) == 0, mean score
odds_0<-exp(new_dat%*%m1_est$estimate)
odds_0

##          [,1]
## [1,] 0.4510945

new_dat1<-c(1,1)
odds_1<-exp(new_dat1%*%m1_est$estimate)
odds_1

##          [,1]
## [1,] 0.6823082

odds_1/odds_0 # odds ratio

##          [,1]
## [1,] 1.512562

exp(m1_est$estimate[2]) # exp(beta_1)

## [1] 1.512562
```

The odds of admission are `exp(m1_est$estimate[2])` times higher for a student with a GRE score one standard deviation above the mean than they are for a student with a mean GRE score.

The odds of admission are `exp(m1_est$estimate[2])` times higher for a student with a GRE score one standard deviation above the mean than they are for a student with a mean GRE score.

Any trouble you can anticipate here?

The odds of admission are `exp(m1_est$estimate[2])` times higher for a student with a GRE score one standard deviation above the mean than they are for a student with a mean GRE score.

Any trouble you can anticipate here?

A visual example: the "effect" of 1 SD increase in GRE scores on Pr(admit==1)
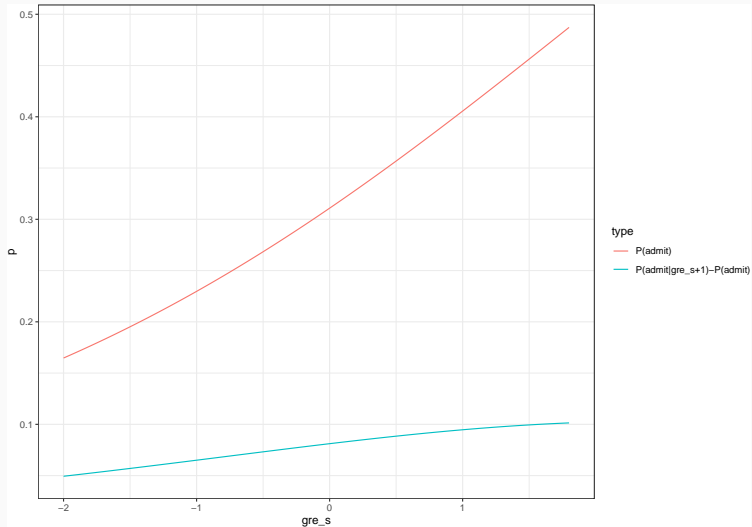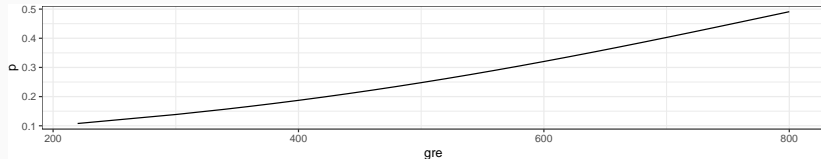
It is easy enough to work on the probability scale

To obtain predicted probabilities of the observed:

- p_hat<-invlogit(predict(m1))
- p_hat<-predict(m1, type = "response")

```
preds<-predict(m1, type = "response")
p_hat<-data.frame(gre = admits$gre,
                  p = preds)

ggplot(p_hat, aes(x = gre, y = p)) +
  geom_line()
```

1. Choose scenarios of theoretical interest
2. Define these in terms of "counterfactual" (fake) data
3. Plug these fake data into the linear predictor (regression equation)
4. Visualize!

Reminder: our model is

$$logit(p(admit_i)) = \beta_0 + \beta_1 GRE_i$$

$$admit_i \sim Binomial(1, p)$$

```
m1<-glm(admit ~ gre, data = admits, family = "binomial")
```

1. Choose scenarios of theoretical interest

Low GRE, average GRE, high GRE

2. Define these in terms of "counterfactual" (fake) data

```
## Look at the distribution of the data to think about scenarios
mean(admits$gre)
```

```
## [1] 587.7
```

```
sd(admits$gre)
```

```
## [1] 115.5165
```

2. Define these in terms of "counterfactual" (fake) data

```
## Look at the distribution of the data to think about scenarios
mean(admits$gre)
```

```
## [1] 587.7
```

```
sd(admits$gre)
```

```
## [1] 115.5165
```

Let's define scenarios at the mean, 1 SD below the mean, and 1 SD above
the mean

```
fake_data<-data.frame(gre = c(
  mean(admits$gre),
  mean(admits$gre) - sd(admits$gre),
  mean(admits$gre) + sd(admits$gre)
))

fake_data
```

```
##        gre
## 1 587.7000
## 2 472.1835
## 3 703.2165
```

3. Plug these fake data into the linear predictor (regression equation)

Because $logit(p(admit_i)) = \beta_0 + \beta_1 x_i$, we can compute the expected probability of admission for a student with mean GRE scores as

```
coef(m1)
```

```
## (Intercept)          gre
## -2.901344270  0.003582212
```

```
### mean GRE scenario: linear predictor
-2.9 + 0.0036 * 587.7
```

```
## [1] -0.78428
```

3. Plug these fake data into the linear predictor (regression equation)

Because $logit(p(admit_i)) = \beta_0 + \beta_1 x_i$, we can compute the expected probability of admission for a student with mean GRE scores as

```
coef(m1)

##  (Intercept)          gre
## -2.901344270  0.003582212

### mean GRE scenario: linear predictor
-2.9 + 0.0036 * 587.7

## [1] -0.78428

### on probability scale
invlogit(-2.9 + 0.0036 * 587.7)

## [1] 0.3133982
```

3. Plug these fake data into the linear predictor (regression equation)

The `predict()` function makes life very easy here

```
## linear predictor
predict(m1, newdata = fake_data)


##          1          2          3
## -0.7960785 -1.2098832 -0.3822738

## probability scale (inverse logit)
predict(m1, newdata = fake_data, type = "response")


##         1         2         3
## 0.3108650 0.2297217 0.4055786
```

4. Visualize!
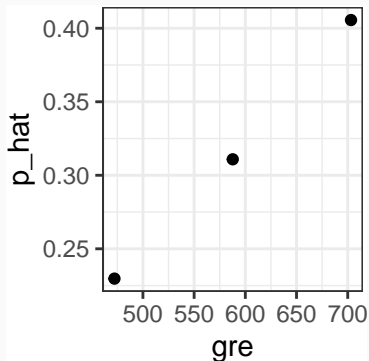
```
### set up our data frame with predictions for plotting
fake_data<-fake_data %>%
  mutate(p_hat = predict(m1, newdata = fake_data, type = "response"))

ggplot(fake_data,
       aes(x = gre, y = p_hat)) +
  geom_point()
```

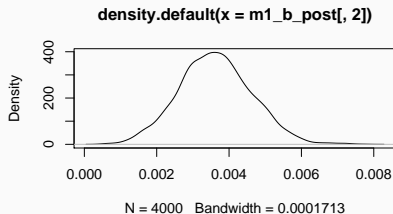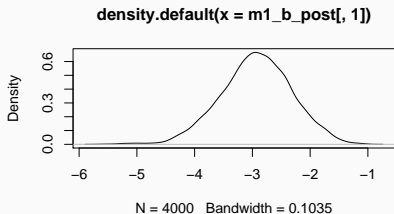## Break

```
m1_b<-stan_glm(admit ~ gre,
               family = "binomial",
               data = admits)
```

Examining our fit

The posterior samples *are* our parameter estimates

```
m1_b_post<-as.data.frame(m1_b)
str(m1_b_post)
```

```
## 'data.frame':    4000 obs. of  2 variables:
## $ (Intercept): num  -2.97 -2.79 -1.88 -2.26 -3.18 ...
## $ gre        : num  0.00373 0.0035 0.00202 0.00271 0.00422 ...
```

density.default(x = m1_b_post[, 1])

density.default(x = m1_b_post[, 2])

N = 4000 Bandwidth = 0.1035

N = 4000 Bandwidth = 0.0001713

Posterior parameter estimates and uncertainty

90 percent of parameter values that are compatible with our data and priors fall between

```
quantile(m1_b_post$`(Intercept)`, probs = c(0.05, 0.95))
```

```
##        5%       95%
## -3.961082 -1.928628
```

```
quantile(m1_b_post$gre, probs = c(0.05, 0.95))
```

```
##          5%         95%
## 0.002003697 0.005274139
```

90 percent of parameter values that are compatible with our data and priors fall between

```
quantile(m1_b_post$`(Intercept)`, probs = c(0.05, 0.95))
```

```
##        5%        95%
## -3.961082 -1.928628
```

```
quantile(m1_b_post$gre, probs = c(0.05, 0.95))
```

```
##          5%          95%
## 0.002003697 0.005274139
```

How do we summarize uncertainty in p?

We have two sources of uncertainty in our linear predictor

$logit(p) = \beta_0 + \beta_1 x_1$

We have two sources of uncertainty in our linear predictor

$$logit(p) = \beta_0 + \beta_1 x_1$$

If an applicant had a GRE score of 600, our posterior expected value of $logit(p)$ is

```
### evaluate the linear equation at all draws of the posterior para
logit_p_hat<-m1_b_post$`(Intercept)` + m1_b_post$gre * 600
summary(logit_p_hat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.1557 -0.8336 -0.7585 -0.7577 -0.6816 -0.3701
```

```
### on the probability scale using inverse logit
summary(invlogit(logit_p_hat))
```
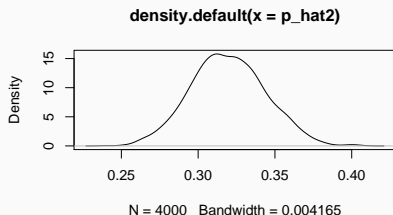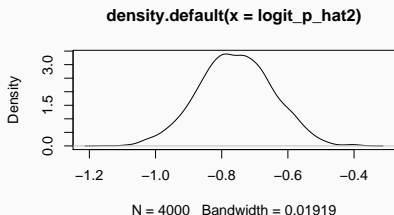
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2394  0.3029  0.3190  0.3196  0.3359  0.4085
```

The same operation can be performed (more easily!) with
`posterior_linpred()` for the linear predictor (logit) scale, and
`posterior_epred()` for the original scale (probability)

```
fake_data<-data.frame(gre = 600)
logit_p_hat2<-posterior_linpred(m1_b, newdata = fake_data)
p_hat2<-posterior_epred(m1_b, newdata = fake_data)
```


density.default(x = logit_p_hat2)

N = 4000   Bandwidth = 0.01919

density.default(x = p_hat2)

N = 4000   Bandwidth = 0.004165

```
fake_data<-data.frame(gre = 400:800)
p_hat<-posterior_epred(m1_b, newdata = fake_data)
### This produces a 4000 row x 401 column matrix, 4000 simulated dr
dim(p_hat)


## [1] 4000  401
```

Let's compute 90 percent intervals for each scenario, then plot the results

```
### convert to data frame and make it long for plotting
p_hat<-as_tibble(p_hat)
p_hat<-p_hat %>%
  pivot_longer(cols = everything(),
               names_to = "scenario",
               values_to = "p_hat")

## compute the uncertainty interval and posterior mean
p_hat<-p_hat %>%
  mutate(scenario = as.numeric(scenario)) %>%
  group_by(scenario) %>%
  summarise(y_lwr = quantile(p_hat, 0.05),
            y_upr = quantile(p_hat, 0.95),
            y_mn = mean(p_hat))

head(p_hat)
```
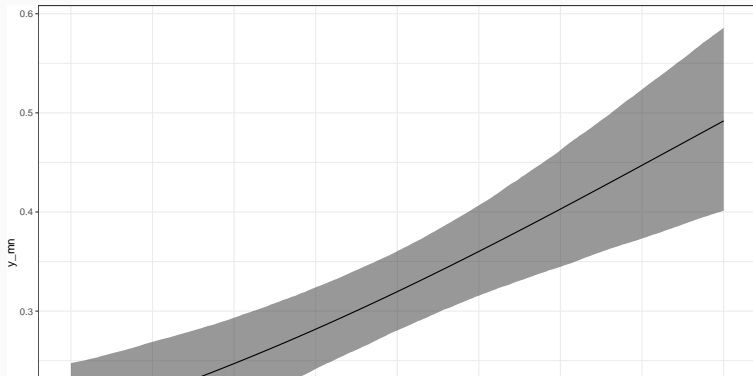
```
## attach the GRE scores using scenario number (row number)
fake_data<-fake_data %>%
  mutate(scenario = 1:n())
### join to p_hat
p_hat<-p_hat %>%
  left_join(fake_data)
## plot with uncertainty interval
ggplot(p_hat,
       aes(x = gre, y = y_mn, ymin = y_lwr, ymax = y_upr)) +
  geom_ribbon(alpha = 0.5)+
  geom_line()
```

p_hat describes our uncertainty in *p*, driven by our estimated uncertainty in β_0 and β_1.

Does it describe our uncertainty in admit?

p_hat describes our uncertainty in *p*, driven by our estimated uncertainty in β_0 and β_1.

Does it describe our uncertainty in admit?

Recall that our model is:

$$logit(p(admit_i)) = \beta_0 + \beta_1 GRE_i$$

$$admit_i \sim Binomial(1, p)$$

p_hat describes our uncertainty in *p*, driven by our estimated uncertainty in β_0 and β_1.

Does it describe our uncertainty in admit?

Recall that our model is:

$$logit(p(admit_i)) = \beta_0 + \beta_1 GRE_i$$

$$admit_i \sim Binomial(1, p)$$

Uncertainty in admit is driven by the binomial distribution

The posterior predictive distribution

We can now take our posterior estimates for *p*, and draw predictions from the *posterior predictive distribution*. This approach averages over our uncertainty in both the parameters, and in sampling the outcome.

```
p_hat<-posterior_epred(m1_b, newdata = fake_data)
### first few draws of p for scenario 1, GRE = 400
head(p_hat[,1])
```

```
## [1] 0.1853545 0.1999156 0.2545779 0.2356202 0.1842336 0.1423152
```

We can now take our posterior estimates for *p*, and draw predictions from the *posterior predictive distribution*. This approach averages over our uncertainty in both the parameters, and in sampling the outcome.

```
p_hat<-posterior_epred(m1_b, newdata = fake_data)
### first few draws of p for scenario 1, GRE = 400
head(p_hat[,1])
```

```
## [1] 0.1853545 0.1999156 0.2545779 0.2356202 0.1842336 0.1423152
```

We can use our uncertainty in *p* to estimate uncertainty in `admit` for new applicants

```
### simulate admissions for each value of p_hat
admit_hat_scen1<-rbinom(4000, 1, p_hat[,1])
mean(admit_hat_scen1)
```

```
## [1] 0.18575
```

```
sd(admit_hat_scen1)
```

```
## [1] 0.3889534
```

```
### compare admit_hat to p_hat
mean(p_hat[,1])
```