Generalized Linear Models

PSY517 Quantitative Analysis III

Derek Powell

Module 3

2021-09-21

Generalized Linear Models

PSY517 Quantitative Analysis III

Derek Powell

Module 3

Logistic regression

Logistic regression

The power of numerical discrimination

- Jevons (1871) took a handful of black beans and tossed them at a white shallow container surrounded by black cloth
- On each toss, a certain number of beans would land in the square and he would then look and try to immediately call out their number
- Then he counted how many beans there actually were in the container, and recorded the result.

I made altogether 1,027 trials, and the following table contains the complete results :—

Estimated Numbers.	Actual Numbers.												
	3	4	5	6	7	8	9	10	11	12	13	14	15
3	23												
4		65											
5			102	7									
6			4	120	18								
7			1	20	113	30	2						
8					25	76	24	6	1				
9						28	76	37	11	1			
10						1	18	46	19	4			
11							2	16	26	17	7	2	
12								2	12	19	11	3	2
13										3	6	3	1
14										1	1	4	6
15											1	2	2
Totals..	23	65	107	147	156	135	122	107	69	45	26	14	11

Figure 1: Jevons (1871) results from his bean-tossing experiment.

The power of numerical discrimination
- Jevons (1871) took a handful of black beans and tossed them at a white shallow container surrounded by black cloth
- On each toss, a certain number of beans would land in the square and he would then look and try to immediately call out their number
- Then he counted how many beans there actually were in the container, and recorded the result.
Figure 1: Jevons (1871) results from his bean-tossing experiment.

Jevons (1871) data

`jevons`

```
## # A tibble: 13 x 3
##    true_beans correct incorrect
##         <dbl>   <dbl>     <dbl>
##  1          3      23         0
##  2          4      65         0
##  3          5     102         5
##  4          6     120        27
##  5          7     113        43
##  6          8      76        59
##  7          9      76        46
##  8         10      46        61
##  9         11      26        43
## 10         12      19        26
## 11         13       6        20
## 12         14       4        10
## 13         15       0        11
```

`jevons_expanded`

```
## # A tibble: 1,027 x 2
##    true_beans correct
##         <dbl>   <dbl>
##  1          7       1
##  2          7       0
##  3          5       1
##  4          8       1
##  5          5       0
##  6         12       1
##  7         10       0
##  8          5       1
##  9          6       1
## 10          6       1
## # ... with 1,017 more rows
```

Generalized linear model: Link functions

The basic schema of a generalized linear model is:

$$
\begin{aligned}
y_i &\overset{iid}{\sim} Distribution(z_i, \ldots) \\
f(z_i) &= \alpha + \beta x_i
\end{aligned}
$$

In simple linear regression, f is the identity function: $f(x) = x$

Generalized linear model: Link functions

The basic schema of a generalized linear model is:

$y_i \overset{iid}{\sim} Distribution(z_i, \ldots)$
$f(z_i) = \alpha + \beta x_i$

In simple linear regression, f is the identity function: $f(x) = x$

Logistic regression model

$$
\begin{aligned}
y_i &\sim Bernoulli(p_i) \\
\text{logit}(p_i) &= \alpha + \beta x_i \\
\alpha &\sim Normal(0, 1.5) \\
\beta &\sim Normal(0, .5)
\end{aligned}
$$

Logistic regression model

$$
\begin{aligned}
y_i &\sim Bernoulli(p_i) \\
\text{logit}(p_i) &= \alpha + \beta x_i \\
\alpha &\sim Normal(0, 1.5) \\
\beta &\sim Normal(0, .5)
\end{aligned}
$$

The logistic function is the inverse of the logit function (also written $logit^{-1}$)

$$
\begin{aligned}
\text{logit}(p_i)) &= \alpha + \beta x_i \\
\Rightarrow p_i &= \text{logistic}(\alpha + \beta x_i)
\end{aligned}
$$

In R:

- `qlogis()` is the logit function
- `plogis()` is the logistic function

Logistic regression and the logit link

The logistic function is the inverse of the logit function (also written $logit^{-1}$)

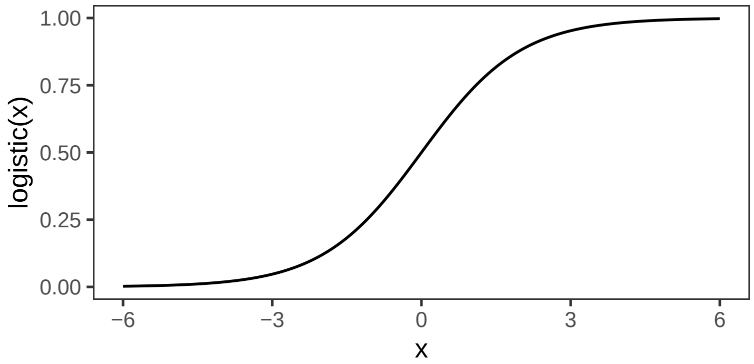
$$
\begin{aligned}
\text{logit}(p_i)) &= \alpha + \beta x_i \\
\Rightarrow p_i &= \text{logistic}(\alpha + \beta x_i)
\end{aligned}
$$

In R:

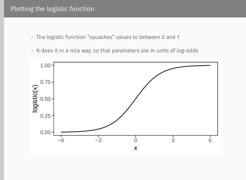
- `qlogis()` is the logit function
- `plogis()` is the logistic function

Plotting the logistic function

- The logistic function “squashes” values to between 0 and 1
- It does it in a nice way, so that parameters are in units of log-odds



2021-09-21



Odds and log odds

$$O(p) = \frac{p}{1-p}$$

Some handy equalities

- $O(.50) = 1$
- $O(0) = -\infty$ and $O(1) = \infty$
- $log(1) = 0$

Odds and log odds

$$O(p) = \frac{p}{1-p}$$

Some handy equalities

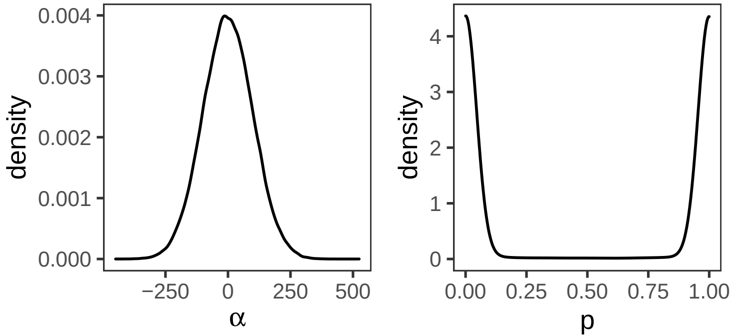
- $O(.50) = 1$
- $O(0) = -\infty$ and $O(1) = \infty$
- $log(1) = 0$

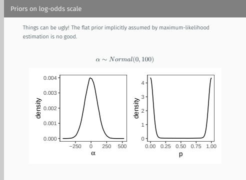
- we will have some exercises on this

Priors on log-odds scale

Things can be ugly! The flat prior implicitly assumed by maximum-likelihood estimation is no good.

$$\alpha \sim Normal(0, 100)$$





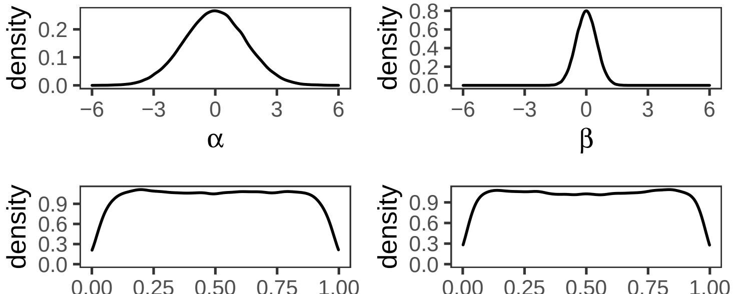
Generic priors on log-odds scale

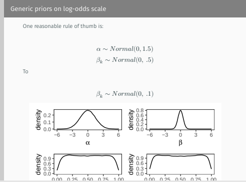
One reasonable rule of thumb is:

$$\alpha \sim Normal(0, 1.5)$$
$$\beta_k \sim Normal(0, .5)$$

To

$$\beta_k \sim Normal(0, .1)$$





Logistic regression model with Bernoulli likelihood

$$
\begin{aligned}
y_i &\sim Bernoulli(p_i) \\
\text{logit}(p_i) &= \alpha + \beta x_i \\
\alpha &\sim Normal(0, 1.5) \\
\beta &\sim Normal(0, .1)
\end{aligned}
$$

```r
fit_bern <- brm(
  correct ~ true_beans,
  family = bernoulli(),
  prior = prior(normal(0, 1.5), class="Intercept") +
    prior(normal(0, .1), class="b"),
  data = jevons_expanded
  )
```


Logistic regression model with Bernoulli likelihood

$$
\begin{aligned}
y_i &\sim Bernoulli(p_i) \\
\text{logit}(p_i) &= \alpha + \beta x_i \\
\alpha &\sim Normal(0, 1.5) \\
\beta &\sim Normal(0, .1)
\end{aligned}
$$

```r
fit_bern <- brm(
  correct ~ true_beans,
  family = bernoulli(),
  prior = prior(normal(0, 1.5), class="Intercept") +
    prior(normal(0, .1), class="b"),
  data = jevons_expanded
  )
```

Binomial logistic regression model

- We can also write our logistic regression as a Binomial regression

$$
\begin{aligned}
y_i &\sim Binomial(p_i, n_i) \\
\text{logit}(p_i) &= \alpha + \beta x_i \\
\alpha &\sim Normal(0, 1.5) \\
\beta &\sim Normal(0, .1)
\end{aligned}
$$

In BRMS we write:

```
fit_binom <- brm(
  correct | trials(correct + incorrect) ~ true_beans,
  family = binomial(),
  prior = prior(normal(0,1.5), class="Intercept") +
    prior(normal(0, .1), class="b"),
  data = jevons
  )
```


2021-09-21

Binomial logistic regression model
We can also write our logistic regression as a Binomial regression
In BRMS we write:

- When we first analyzed the Jevons' data in module 1, we used the Beta-Binomial model
- generally only makes sense to write it this way when the predictors take on a limited number of values
- will get the same parameter estimates
- but predicting etc. will work a bit differently

Bernoulli vs Binomial logistic regression

- These two models are equivalent and produce the same parameter estimates
- But the models function differently for **prediction**
- Binomial version can be helpful when all predictors are discrete

Bernoulli vs Binomial logistic regression

- These two models are equivalent and produce the same parameter estimates
- But the models function differently for **prediction**
- Binomial version can be helpful when all predictors are discrete

(Expected) posterior predictive

Expected posterior predictive

- `add_epred_draws()` will add samples from the expected posterior predictive
- Will be a **probability** for both Bernoulli and Binomial logistic regressions

Posterior predictive

- `add_predicted_draws()` will add samples from the posterior predictive
- **Bernoulli:** will be **1 or 0** for success or failure
- **Binomial:** will be a **count** of successes (out of n trials)

(Expected) posterior predictive
Expected posterior predictive
add_epred_draws() will add samples from the expected posterior predictive
Will be a probability for both Bernoulli and Binomial logistic regressions
Posterior predictive
add_predicted_draws() will add samples from the posterior predictive
Bernoulli: will be 1 or 0 for success or failure
Binomial: will be a count of successes (out of n trials)

Posterior prediction from Bernoulli and Binomial

```
jevons_expanded %>%
  add_predicted_draws(fit_bern,  ndraws=5) %>%
  select(-.chain, -.iteration, -.row)
```

```
## Adding missing grouping variables: `.row`
## # A tibble: 5,135 x 5
## # Groups:   true_beans, correct, .row [1,027]
##      .row true_beans correct .draw .prediction
##     <int>      <dbl>   <dbl> <int>       <int>
##  1      1          7       1     1           0
##  2      1          7       1     2           1
##  3      1          7       1     3           0
##  4      1          7       1     4           0
##  5      1          7       1     5           1
##  6      2          7       0     1           0
##  7      2          7       0     2           0
##  8      2          7       0     3           1
##  9      2          7       0     4           0
## 10      2          7       0     5           1
## # ... with 5,125 more rows
```

```
jevons %>%
  add_predicted_draws(fit_binom,  ndraws=5) %>%
  select(-.chain, -.iteration, -.row)
```

```
## Adding missing grouping variables: `.row`
## # A tibble: 65 x 6
## # Groups:   true_beans, correct, incorrect, .row [13]
##      .row true_beans correct incorrect .draw .prediction
##     <int>      <dbl>   <dbl>     <dbl> <int>       <int>
##  1      1          3      23         0     1          21
##  2      1          3      23         0     2          23
##  3      1          3      23         0     3          21
##  4      1          3      23         0     4          20
##  5      1          3      23         0     5          21
##  6      2          4      65         0     1          59
##  7      2          4      65         0     2          57
##  8      2          4      65         0     3          56
##  9      2          4      65         0     4          61
## 10      2          4      65         0     5          59
## # ... with 55 more rows
```

From expected posterior predictive to posterior predictive

In logistic regression, the linear model is used to predict the probability of success for each of the i observations, p_i.

$$\text{Posterior } p_i = P(p_i|y)$$

Posterior predictive distributions

$$y_i^* \sim Bernoulli(p_i|y)$$
$$y_i^* \sim Binomial(p_i|y,\ n_i)$$

Expected posterior predictive distributions

$$E(P(y_i^*)) \qquad E(X_{Bern}) = p \qquad E(X_{Binom}) = Np$$

From expected posterior predictive to posterior predictive
In logistic regression, the linear model is used to predict the probability of success for each of the i observations, p_i.
Posterior $p_i = P(p_i|y)$
Posterior predictive distributions
$y_i^* \sim Bernoulli(p_i|y)$
$y_i^* \sim Binomial(p_i|y,\ n_i)$
Expected posterior predictive distributions
$E(P(y_i^*)) \quad E(X_{Bern}) = p \quad E(X_{Binom}) = Np$

Coding posterior predictive distributions

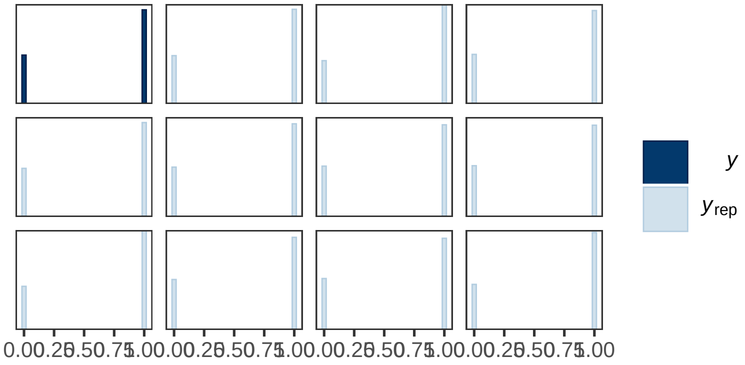
```
jevons_expanded %>%
  group_by(true_beans) %>%
  mutate(trials = n(), n_correct = sum(if_else(correct == 1, 1,0))) %>%
  add_epred_draws(fit_bern,  ndraws = 3) %>%
  mutate(
    .predbern = rbernoulli(n(), .epred),
    .predbinom = rbinom(n(), trials, .epred)
  ) %>% select(-.chain, -.iteration)
```

```
## # A tibble: 3,081 x 9
## # Groups:   true_beans, correct, trials, n_correct, .row [1,027]
##    true_beans correct trials n_correct  .row .draw .epred .predbern .predbinom
##         <dbl>   <dbl>  <int>     <dbl> <int> <int>  <dbl> <lgl>          <int>
##  1          7       1    156       113     1     1  0.760 TRUE             116
##  2          7       1    156       113     1     2  0.764 TRUE             122
##  3          7       1    156       113     1     3  0.749 TRUE             119
##  4          7       0    156       113     2     1  0.760 TRUE             113
##  5          7       0    156       113     2     2  0.764 TRUE             112
##  6          7       0    156       113     2     3  0.749 TRUE             106
##  7          5       1    107       102     3     1  0.879 TRUE              94
##  8          5       1    107       102     3     2  0.884 TRUE              90
##  9          5       1    107       102     3     3  0.870 TRUE              95
## 10          8       1    135        76     4     1  0.677 TRUE              90
## # ... with 3,071 more rows
```

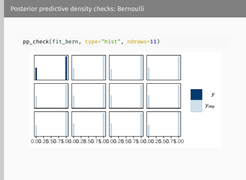

Coding posterior predictive distributions

Posterior predictive density checks: Bernoulli

```r
pp_check(fit_bern, type="hist", ndraws=11)
```

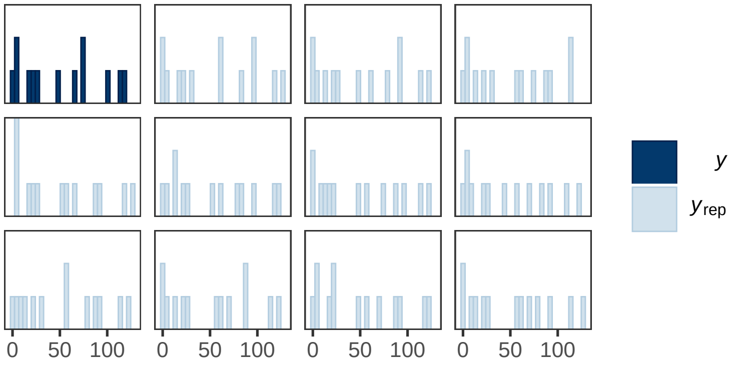


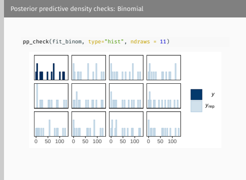
2021-09-21



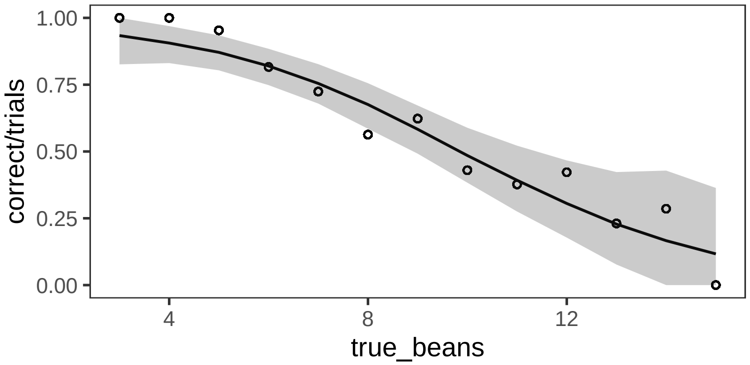
Posterior predictive density checks: Binomial

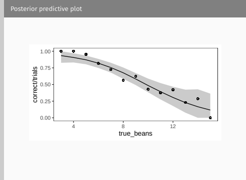
```
pp_check(fit_binom, type="hist", ndraws = 11)
```





Posterior predictive plot





Psychological model(s)

- Hypothesis is that there is a certain “power of numerical discrimination” where you have ~100% accuracy
- Nowadays, we know this power comes from the human visual system’s multi-object tracking system, which can track a limited number of objects simultaneously
- How many items can Jevon’s visual system process simultaneously?

Psychological model(s)

- Hypothesis is that there is a certain “power of numerical discrimination” where you have ~100% accuracy
- Nowadays, we know this power comes from the human visual system’s multi-object tracking system, which can track a limited number of objects simultaneously
- How many items can Jevon’s visual system process simultaneously?

Comparing different thresholds

One way to determine the most likely limit for Jevon's powers is to create models with different thresholds and compare how well they account for his data.

Comparing different thresholds

One way to determine the most likely limit for Jevon's powers is to create models with different thresholds and compare how well they account for his data.

Possible thresholds at 3, 4, 5, and 6

I created indicator variables **underT** and **overT** for thresholds at 3, 4, 5, and 6.

```
df1 <- jevons %>%
  mutate(
    under3 = if_else(true_beans <= 3, 1, 0),
    under4 = if_else(true_beans <= 4, 1, 0),
    under5 = if_else(true_beans <= 5, 1, 0),
    under6 = if_else(true_beans <= 6, 1, 0),
    over3 = if_else(true_beans > 3, 1, 0),
    over4 = if_else(true_beans > 4, 1, 0),
    over5 = if_else(true_beans > 5, 1, 0),
    over6 = if_else(true_beans > 6, 1, 0),
    beans_over3 = if_else(true_beans <= 3, 0, true_beans - 3),
    beans_over4 = if_else(true_beans <= 4, 0, true_beans - 4),
    beans_over5 = if_else(true_beans <= 5, 0, true_beans - 5),
    beans_over6 = if_else(true_beans <= 6, 0, true_beans - 6),
    )
```

A thresholded model

$$
\begin{aligned}
y_i &\sim Binomial(p_i, n) \\
p_i &= \text{logit}^{-1}(\beta_1 \text{under}_i + \beta_2 \text{over}_i + \beta_3 \text{over}_i : \text{true\_beans}_i) \\
\beta_1 &\sim Normal(3, 1.5) \\
\beta_2 &\sim Normal(0, 1.5) \\
\beta_3 &\sim Normal(0, .1)
\end{aligned}
$$

- The idea is there is a threshold number of items under which performance is unaffected by the number and is essentially perfect
- Beyond the threshold, performance can start to decline as the number of items increases

2021-09-21

A thresholded model

$$
\begin{aligned}
y_i &\sim Binomial(p_i, n) \\
p_i &= \text{logit}^{-1}(\beta_1 \text{under}_i + \beta_2 \text{over}_i + \beta_3 \text{over}_i : \text{true\_beans}_i) \\
\beta_1 &\sim Normal(3, 1.5) \\
\beta_2 &\sim Normal(0, 1.5) \\
\beta_3 &\sim Normal(0, .1)
\end{aligned}
$$

- The idea is there is a threshold number of items under which performance is unaffected by the number and is essentially perfect
- Beyond the threshold, performance can start to decline as the number of items increases

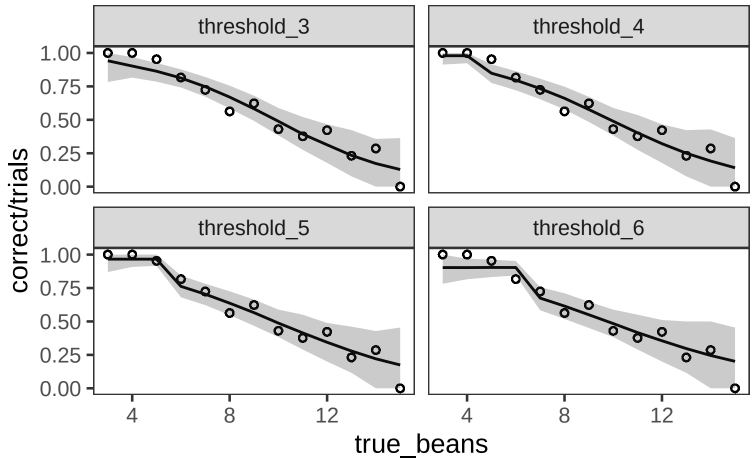
Implementing the models in **brms**

- Then, I fit four separate models using these predictors
- Below is an example for a model with a threshold of 3 objects

```
fit_t3 <- brm(
  correct | trials(correct + incorrect) ~ 0 + under3 + over3 + beans_over3,
  family = binomial(),
  prior = prior(normal(0, 1.5), coef="over3") +
    prior(normal(0, 1.5), coef = "under3") +
    prior(normal(0, .1), coef="beans_over3"),
  data = df1,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit_t3",
  file_refit = "on_change"
)
```

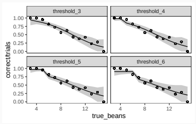

Implementing the models in brms
- Then, I fit four separate models using these predictors
- Below is an example for a model with a threshold of 3 objects
fit_t3 <- brm(
  correct | trials(correct + incorrect) ~ 0 + under3 + over3 + beans_over3,
  family = binomial(),
  prior = prior(normal(0, 1.5), coef="over3") +
    prior(normal(0, 1.5), coef = "under3") +
    prior(normal(0, .1), coef="beans_over3"),
  data = df1,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit_t3",
  file_refit = "on_change"
)

Posterior predictions for the threshold models



2021-09-21

Posterior predictions for the threshold models



Model comparison

```r
comp1 <- loo(fit_binom, fit_t3, fit_t4, fit_t5, fit_t6,
  moment_match = T
)
comp1$diffs
```

```
##            elpd_diff se_diff
## fit_t5       0.0       0.0
## fit_t4      -4.6       7.0
## fit_binom   -9.3       6.5
## fit_t3     -12.0       7.4
## fit_t6     -19.1      10.5
```


Model comparison
comp1 <- loo(fit_binom, fit_t3, fit_t4, fit_t5, fit_t6,
moment_match = T
)
comp1$diffs
elpd_diff se_diff
fit_t5 0.0 0.0
fit_t4 -4.6 7.0
fit_binom -9.3 6.5
fit_t3 -12.0 7.4
fit_t6 -19.1 10.5

- `moment_match = T` uses a fancier but slower method that produces more accurate results.
- if needed, `loo()` will warn you. So just listen to `loo()`

Model averaging

- If we can't pick one best model, we can use them all together
- Bayesian model averaging combines predictions from multiple models
- Models are weighted by the $\mathrm{elppd}_{\mathrm{loo}}$ to optimize for out-of-sample predictive accuracy

Model averaging
- If we can't pick one best model, we can use them all together
- Bayesian model averaging combines predictions from multiple models
- Models are weighted by the $\mathrm{elppd}_{\mathrm{loo}}$ to optimize for out-of-sample predictive accuracy

Model weights

Roughly: The weights tell us how much we should listen to each model if we want to make the best out-of-sample predictions.

```
loo_model_weights(comp1$loos, method="pseudobma")
```

```
## Method: pseudo-BMA+ with Bayesian bootstrap
## ------
##           weight
## fit_binom 0.018
## fit_t3    0.006
## fit_t4    0.219
## fit_t5    0.756
## fit_t6    0.002
```


Model weights
Roughly: The weights tell us how much we should listen to each model if we want to make the best out-of-sample predictions.
loo_model_weights(comp1$loos, method="pseudobma")
Method: pseudo-BMA+ with Bayesian bootstrap
------
weight
fit_binom 0.018
fit_t3   0.006
fit_t4   0.219
fit_t5   0.756
fit_t6   0.002

- The weights computed for each model can tell us how plausible each model is.

There is no one best model, but we can see from the model "pseudo-BMA" weights that the threshold=4 and threshold=5 make the best predictions, and the other models should essentially be ignored. So altogether we can affirm that the change-point is most likely at either 4 or 5 objects to be counted.

- A valuable lesson: sometimes the data just isn't there to answer a question with tons of confidence, and all the fancy statistics in the world can't change that.

Poisson regression example

Poisson regression example

NYPD’s stop-and-frisk program

- Controversial (read: bad) NYPD practice of temporarily detaining, questioning, and at times searching civilians and suspects on the street for weapons and other contraband.
- Started under Mayor Rudy Giuliani, continued under Mayor Michael Bloomberg, and finally reformed by Mayor Bill De Blasio
- At its height in 2011, NYPD officers recorded conducting 685,724 stops with 88% (603,437) resulting in no conviction.

NYPD’s stop-and-frisk program
- Controversial (read: bad) NYPD practice of temporarily detaining, questioning, and at times searching civilians and suspects on the street for weapons and other contraband.
- Started under Mayor Rudy Giuliani, continued under Mayor Michael Bloomberg, and finally reformed by Mayor Bill De Blasio
- At its height in 2011, NYPD officers recorded conducting 685,724 stops with 88% (603,437) resulting in no conviction.

685000/8129000 = enough to have stopped 8.4% of nyc population

- Reformed but not stopped—still going on in several precincts according to recent Intercept article.
- A review of the NYPD’s stops-related data shows that in 2020, the number of reported stops was at its lowest ever — 9,544, down from 13,459 in 2019 and 11,008 in 2018. Despite the drop, the racial disparity remained as stark as ever, with New Yorkers of color making up 91 percent of those stopped, roughly the same as in the two years prior.

https://theintercept.com/2021/06/10/stop-and-frisk-new-york-police-racial-disparity/

Racial bias in NYPD’s stop and frisk program

> *In total, blacks and Hispanics represented 51% and 33% of the stops, despite being only 26% and 24%, of the city population based on the 1990 Census … — Gelman, Fagan, & Kiss, 2007 (using data from 1998-1999)*

- A judge ruled in 2013 that New York City’s stop-and-frisk program was carried out in a manner that violated the U.S. Constitution
- A number of studies have found no evidence that stop-and-frisk reduced crime

Racial bias in NYPD’s stop and frisk program

In total, blacks and Hispanics represented 51% and 33% of the stops, despite being only 26% and 24%, of the city population based on the 1990 Census … — Gelman, Fagan, & Kiss, 2007 (using data from 1998-1999)

- A judge ruled in 2013 that New York City’s stop-and-frisk program was carried out in a manner that violated the U.S. Constitution
- A number of studies have found no evidence that stop-and-frisk reduced crime

Stop and frisk data

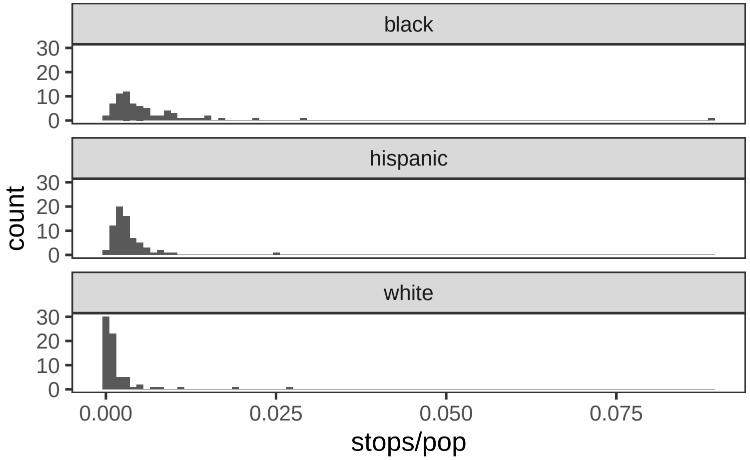
We will examine a dataset of NYPD stops from a 15mo time period from 1998-1999.

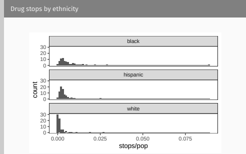
```
## # A tibble: 900 x 6
##    stops   pop arrests precinct ethnicity crime_type
##    <dbl> <dbl>   <dbl> <fct>    <chr>     <chr>
##  1    75  1720     191 1        black     violent
##  2    36  1720      57 1        black     weapons
##  3    74  1720     599 1        black     property
##  4    17  1720     133 1        black     drug
##  5    37  1368      62 1        hispanic  violent
##  6    39  1368      27 1        hispanic  weapons
##  7    23  1368     149 1        hispanic  property
##  8     3  1368      57 1        hispanic  drug
##  9    26 23854     135 1        white     violent
## 10    32 23854      16 1        white     weapons
## # ... with 890 more rows
```


the data has the

- number of stops
- total population
- arrests in the prior year
- per precinct (75 precincts)
- per suspected crime type
- for three ethnic groups: black, hispanic, and white (nypd also recorded “other”, but only rarely)

Drug stops by ethnicity





- can think it's bad from a 4th amendment perspective
- but is there a racial bias to the stops (at least in terms of outcomes)?
- looks likely

Dummy-coding factor variables

ethnicity	d1	d2
black	0	0
hispanic	1	0
white	0	1

- For n categories, create $n - 1$ binary dummy variables
- One level is the “reference” level with all zeros on the dummy variables
- `brm()` will make these variables for us automatically if we pass a factor variable into our equation (as will `lm()` and `glm()`).

Dummy-coding factor variables

ethnicity	d1	d2
black	0	0
hispanic	1	0
white	0	1

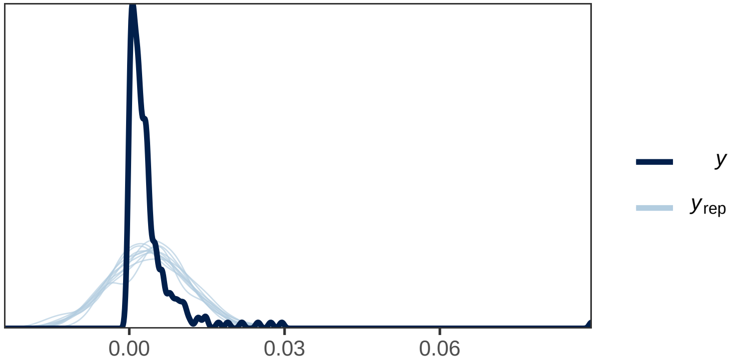
- For n categories, create $n - 1$ binary dummy variables
- One level is the “reference” level with all zeros on the dummy variables
- `brm()` will make these variables for us automatically if we pass a factor variable into our equation (as will `lm()` and `glm()`).

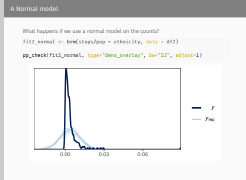
A Normal model

What happens if we use a normal model on the counts?

```r
fit2_normal <- brm(stops/pop ~ ethnicity, data = df2)
```

```r
pp_check(fit2_normal, type="dens_overlay", bw="SJ", adjust=1)
```





posterior predictive density checks look very bad.

The Poisson distribution as a model for counts

- The Poisson distribution describes the probability of counts when the total number of “trials” is unknown or uncountable.
- Described by just one parameter, λ

$$y_i \sim Poisson(\lambda_i)$$

The Poisson distribution as a model for counts
- The Poisson distribution describes the probability of counts when the total number of “trials” is unknown or uncountable.
- Described by just one parameter, λ
$$y_i \sim Poisson(\lambda_i)$$

Like the Binomial distribution, but where we don’t know the number of possible trials (or there is no maximum number of trials)

Log-link function

$$y_i \sim Poisson(\lambda_i)$$
$$\log(\lambda_i) = \alpha + \beta x_i$$

- λ_i must be a positive number
- We use the log link function to ensure this.
- A log link makes the regression coefficients **multiplicative**
  - $\log(X) + \log(Y) = \log(XY)$ and $\log(X) - \log(Y) = \log(X/Y)$
  - $\exp(\log(X) + \log(Y)) = XY$
- $exp(\beta)$ tells us: by how many times the rate λ increases with a one-unit increase in x

Log-link function

$$y_i \sim Poisson(\lambda_i)$$
$$\log(\lambda_i) = \alpha + \beta x_i$$

- λ_i must be a positive number
- We use the log link function to ensure this.
- A log link makes the regression coefficients **multiplicative**
  - $\log(X) + \log(Y) = \log(XY)$ and $\log(X) - \log(Y) = \log(X/Y)$
  - $\exp(\log(X) + \log(Y)) = XY$
- $exp(\beta)$ tells us: by how many times the rate λ increases with a one-unit increase in x

- log(0) = -Inf
- log(1) = 0
- log(< 1) = negative
- log(> 1) = positive

Poisson distribution as model for rates

- Poisson is a **discrete** probability distribution for integer counts
- But can also be used to model **rates** by including an *offset* or *exposure* predictor
- E.g. can model events per month, per population size, etc.

$$\log(\lambda/\tau) = \log(\lambda) - \log(\tau)$$

Poisson distribution as model for rates

- Poisson is a **discrete** probability distribution for integer counts
- But can also be used to model **rates** by including an *offset* or *exposure* predictor
- E.g. can model events per month, per population size, etc.

$$\log(\lambda/\tau) = \log(\lambda) - \log(\tau)$$

•

Our first Poisson regression model

$$
\begin{aligned}
y_i &\sim Poisson(\lambda_i) \\
\log(\lambda_i) &= \alpha + \beta_1 E_i^w + \beta_2 E_i^h - log(\tau) \\
\tau &= population \\
\alpha &\sim Normal(0, 5) \\
\beta_1 &\sim Normal(0, .5) \\
\beta_2 &\sim Normal(0, .5)
\end{aligned}
$$

Our first Poisson regression model

$y_i \sim Poisson(\lambda_i)$
$\log(\lambda_i) = \alpha + \beta_1 E_i^w + \beta_2 E_i^h - log(\tau)$
$\tau = population$
$\alpha \sim Normal(0, 5)$
$\beta_1 \sim Normal(0, .5)$
$\beta_2 \sim Normal(0, .5)$

Implementing the Poisson regression model

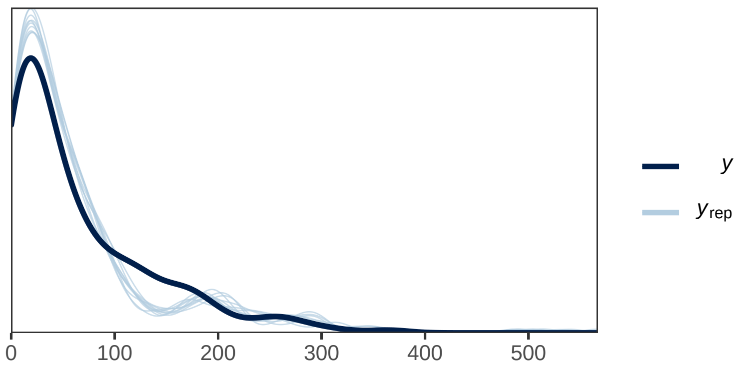
```r
fit2 <- brm(
  stops ~ ethnicity + offset(log(pop)),
  prior = prior(normal(0, 5), class="Intercept") +
    prior(normal(0, .5), class="b"),
  data = df2,
  family = poisson(),
  iter = 4000,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit2",
  file_refit = "on_change"
)
```

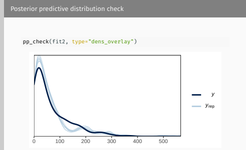

Implementing the Poisson regression model

fit2 <- brm(
  stops ~ ethnicity + offset(log(pop)),
  prior = prior(normal(0, 5), class="Intercept") +
    prior(normal(0, .5), class="b"),
  data = df2,
  family = poisson(),
  iter = 4000,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit2",
  file_refit = "on_change"
)

Posterior predictive distribution check

```
pp_check(fit2, type="dens_overlay")
```





- Poisson is discrete, but when counts are large can be ok to visualize with a density

Interpreting coefficients

```
##  Family: poisson
##   Links: mu = log
## Formula: stops ~ ethnicity + offset(log(pop))
##    Data: df2 (Number of observations: 213)
##   Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup draws = 8000
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept            -5.63      0.01    -5.66    -5.61 1.00     6353     5615
## ethnicityhispanic    -0.14      0.02    -0.17    -0.10 1.00     5294     5183
## ethnicitywhite       -1.56      0.02    -1.61    -1.51 1.00     4869     5398
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```


Interpreting coefficients

- $exp(\beta)$ is relative rate
- exp(-1.56) = .210
- black people are stopped 4.75 as many times a white people (exp(1.56))

Compared to normal model estimates

- Poisson coefficients are multiplicative
- We can multiply/divide coefficients from the normal model to compare

Normal model estimates

```
## # A tibble: 2 x 4
##   Coefficient       Estimate  Q2.5 Q97.5
## * <chr>                <dbl> <dbl> <dbl>
## 1 ethnicityhispanic     2.03  1.23  4.31
## 2 ethnicitywhite        3.53  1.69 16.3
```

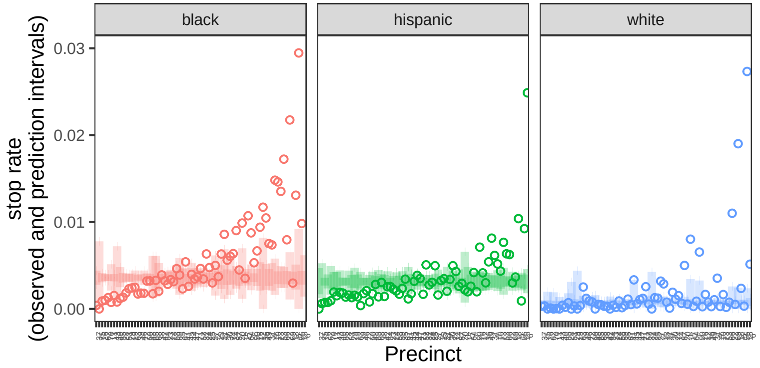
Poisson model estimates

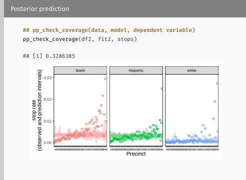
```
## # A tibble: 2 x 4
##   Coefficient       Estimate .lower .upper
##   <chr>                <dbl>  <dbl>  <dbl>
## 1 ethnicityhispanic     1.14   1.10   1.19
## 2 ethnicitywhite        4.75   4.52   4.98
```


- Using robust estimates (medians) in both cases b/c normal came out very wild
- normal model gives biased estimate for hispanic stop-rate, and has huge amounts of uncertainty

```
## pp_check_coverage(data, model, dependent variable)
pp_check_coverage(df2, fit2, stops)
```

```
## [1] 0.3286385
```





I wrote a little function called `pp_check_coverage()` to compute what proportion of the observations fell within the 95% prediction intervals

Overdispersion

- Under Poisson distribution both the variance and mean are equal to λ
- But often we observe with count data that the variance is larger than the mean
- This is called ***overdispersion***

Overdispersion

- Under Poisson distribution both the variance and mean are equal to λ
- But often we observe with count data that the variance is larger than the mean
- This is called ***overdispersion***

Negative-Binomial or Gamma-Poisson model

- One way to deal with overdispersion is to replace the Poisson distribution with the Negative-Binomial (aka Gamma-Poisson) distribution.

$$y_i \sim Negative\text{-}Binomial(\lambda_i, \phi)$$

$$\mathsf{Var}(X_{NB}) = \lambda - \frac{\lambda^2}{\phi}$$

Improve the model

- Another way is to address overdispersion is to account for structure of data better

Dealing with overdispersion

Negative-Binomial or Gamma-Poisson model
- One way to deal with overdispersion is to replace the Poisson distribution with the Negative-Binomial (aka Gamma-Poisson) distribution.

$$y_i \sim Negative\text{-}Binomial(\lambda_i, \phi)$$
$$\mathsf{Var}(X_{NB}) = \lambda - \frac{\lambda^2}{\phi}$$

Improve the model
- Another way is to address overdispersion is to account for structure of data better

- In the normal model, if we get μ wrong, then we will get a larger estimate for σ. So we will make imperfect predictions but also have appropriate uncertainty in them.
- But in Poisson we just have λ so model misspecification will result in overconfidence

Adding a predictor

If different police precincts have different leaders and policies, then they could each have their own baseline rate of stops

- Let's add predictors to code for precinct
- There are 75 precincts, so we need 74 dummy-coded predictors
- `brm()` will make these for us if we pass a factor variable into our equation (as will `lm()` and `glm()`).

```
fit3 <- brm(
  stops ~ ethnicity + precinct + offset(log(pop)),
  prior = prior(normal(0, 5), class="Intercept") +
    prior(normal(0, .5), class="b"),
  data = df2,
  family = poisson(),
  iter = 8000,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit3",
  file_refit = "on_change"
)
```


Adding a predictor

If different police precincts have different leaders and policies, then they could each have their own baseline rate of stops

- Let's add predictors to code for precinct
- There are 75 precincts, so we need 74 dummy-coded predictors
- brm() will make these for us if we pass a factor variable into our equation (as will lm() and glm()).

fit3 <- brm(
  stops ~ ethnicity + precinct + offset(log(pop)),
  prior = prior(normal(0, 5), class="Intercept") +
    prior(normal(0, .5), class="b"),
  data = df2,
  family = poisson(),
  iter = 8000,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit3",
  file_refit = "on_change"
)

Adding 74 predictors???

- Normally we would be pretty hesitant to add 74 predictor variables to a model predicitng 225 rows of data
- But with count data, the effective sample size is not purely determined by the number of rows
- The counts themselves also influence the effective sample size, larger counts will allow for more precise parameter estimates

Adding 74 predictors???

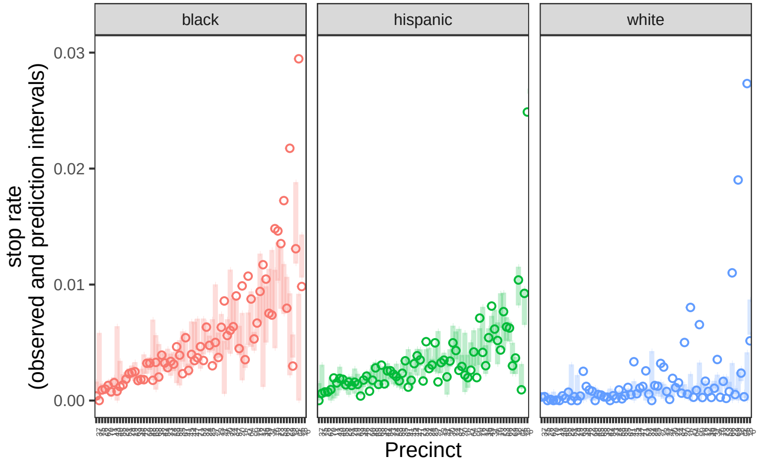
- Normally we would be pretty hesitant to add 74 predictor variables to a model predicitng 225 rows of data
- But with count data, the effective sample size is not purely determined by the number of rows
- The counts themselves also influence the effective sample size, larger counts will allow for more precise parameter estimates

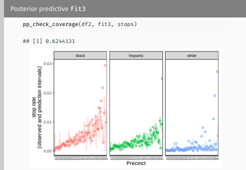
- we haven't really talked about "power" because it's an NHST thing
- but here I mean the precision of our estimates

Posterior predictive `fit3`

```
pp_check_coverage(df2, fit3, stops)
```

```
## [1] 0.6244131
```





Bias?

> *One newspaper and one news service, they just keep saying 'oh it's a disproportionate percentage of a particular ethnic group.' That may be, but it's not a disproportionate percentage of those who witnesses and victims describe as committing the [crime]. In that case, incidentally, I think we disproportionately stop whites too much and minorities too little.*
> *—Bloomberg, 2013*

Or so-claimed NYPD Mayor Michael Bloomberg in 2013.

Are the different stop-rates a result of bias? Or could they simply reflect differences in the crime rates among people from different groups?

Bias?

One newspaper and one news service, they just keep saying 'oh it's a disproportionate percentage of a particular ethnic group.' That may be, but it's not a disproportionate percentage of those who witnesses and victims describe as committing the [crime]. In that case, incidentally, I think we disproportionately stop whites too much and minorities too little.
—Bloomberg, 2013

Or so-claimed NYPD Mayor Michael Bloomberg in 2013.

Are the different stop-rates a result of bias? Or could they simply reflect differences in the crime rates among people from different groups?

“Controlling for” arrest rates

- Let’s try to address this objection by statistically controlling for the crime rate among New Yorkers of different ethnicities within each precint’s jurisdiction.
- As a proxy for the true crime rate, we will use the arrest rates for different groups (following Gelman, Fagan, & Kiss, 2007) .
- This analysis will be extremely charitable to the NYPD as it will assume there is no bias in arrest rates and that any differences in the arrests reflect differences in the true crime rate among the different ethnic groups (possibly owing to other factors like poverty, education, etc.)
- If we control for arrest counts, do we still see bias in the stops?

“Controlling for” arrest rates

- Let’s try to address this objection by statistically controlling for the crime rate among New Yorkers of different ethnicities within each precint’s jurisdiction.
- As a proxy for the true crime rate, we will use the arrest rates for different groups (following Gelman, Fagan, & Kiss, 2007) .
- This analysis will be extremely charitable to the NYPD as it will assume there is no bias in arrest rates and that any differences in the arrests reflect differences in the true crime rate among the different ethnic groups (possibly owing to other factors like poverty, education, etc.)
- If we control for arrest counts, do we still see bias in the stops?

- this may help our model’s issues as well, if the predictor improves our fit
- later we will expand on this idea of “controlling for” and give things a more rigorous causal treatment
- for now, since the assumption is essentially imaginary, we will just proceed in thinking about this in the statistical sense

[Control for arrests by adding as predictor, note that it improves fit quite a bit (what does that mean? makes stops look less biased. but clear they are still biased.)]
[arrests might be biased (it’s sort of obvious that they are.) And it wouldn’t be surprising if they were less biased than stops. b/c need SOMETHING to arrest, but nothing to stop]

Adding a predictor for arrest rates

```
fit4 <- brm(
  stops ~ ethnicity + precinct + log(arrest_rate) + offset(log(pop)),
  prior = prior(normal(0, 5), class="Intercept") +
    prior(normal(0, .5), class="b"),
  data = df2,
  family = poisson(),
  iter = 8000,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit4",
  file_refit = "on_change"
)
```

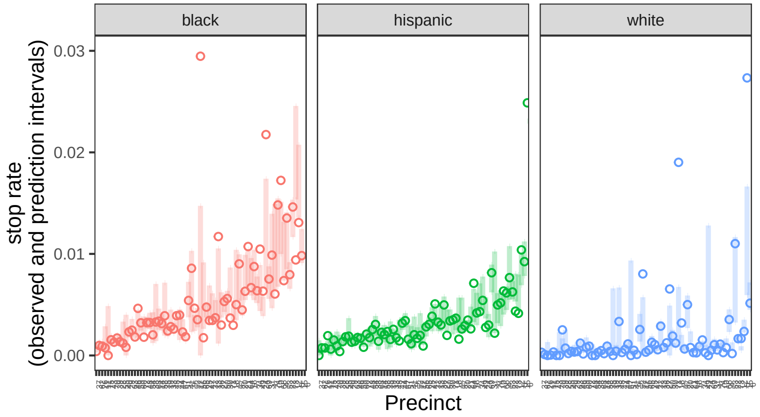

Adding a predictor for arrest rates

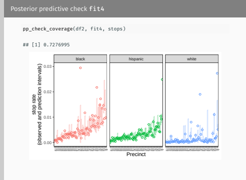
fit4 <- brm(
  stops ~ ethnicity + precinct + log(arrest_rate) + offset(log(pop)),
  prior = prior(normal(0, 5), class="Intercept") +
    prior(normal(0, .5), class="b"),
  data = df2,
  family = poisson(),
  iter = 8000,
  save_pars = save_pars(all = TRUE),
  file = "_cache/fit4",
  file_refit = "on_change"
)

Posterior predictive check `fit4`

```
pp_check_coverage(df2, fit4, stops)
```

```
## [1] 0.7276995
```





Still looks like our data are overdispersed, but predictions are considerably more accurate.

- Would need to either get more clever or change likelihood to Negative-Binomial
- Will leave that for another day
- plot omits one extreme outlier to make it easier to see others

Inspecting the parameters

```
fixef(fit4) %>%
  ...
```

```
## # A tibble: 74 x 5
##    Coefficient       Estimate Est.Error    Q2.5   Q97.5
##    <chr>                <dbl>     <dbl>   <dbl>   <dbl>
##  1 Intercept          -3.01      0.0971 -3.20   -2.82
##  2 ethnicityhispanic  -0.0199    0.0247 -0.0683  0.0279
##  3 ethnicitywhite     -0.675     0.0465 -0.766  -0.585
##  4 logarrest_rate      0.656     0.0211  0.614   0.697
##  5 precinct2          -0.389     0.143  -0.676  -0.116
##  6 precinct4           0.344     0.0868  0.173   0.513
##  7 precinct5           0.0968    0.0848 -0.0725  0.263
##  8 precinct6           1.65      0.0787  1.50    1.81
##  9 precinct7           0.427     0.0985  0.231   0.619
## 10 precinct9          -0.0724    0.184  -0.440   0.276
## # ... with 64 more rows
```


- I call `fixef()` and do some munging to clean up the output
- exp(-.675) = 0.5092 or black people are stopped 1.964 as many times a white people

Model comparison

```r
comps2 <- loo(fit2, fit3, fit4)
comps2$diffs
```

```
##      elpd_diff se_diff
## fit4     0.0       0.0
## fit3  -473.9     138.3
## fit2 -3469.8     617.7
```


Model comparison

comps2 <- loo(fit2, fit3, fit4)
comps2$diffs

elpd_diff se_diff
fit4   0.0   0.0
fit3 -473.9   138.3
fit2 -3469.8   617.7

How should we interpret this?

Should we think these latest estimates are the true ones?

- Assumed arrest-rate reflects true crime rate
- But that's not very likely to be true, especially now that we see the bias in stops persists
- Arrests must be recorded, but can we trust all stops were recorded?
- Remember: statistics is just a tool for understanding, not a replacement for thinking

How should we interpret this?
Should we think these latest estimates are the true ones?
- Assumed arrest-rate reflects true crime rate
- But that's not very likely to be true, especially now that we see the bias in stops persists
- Arrests must be recorded, but can we trust all stops were recorded?
- Remember: statistics is just a tool for understanding, not a replacement for thinking

Extra slides

Extra slides