**A guide to modeling proportions with Bayesian beta and zero-inflated beta regression models**

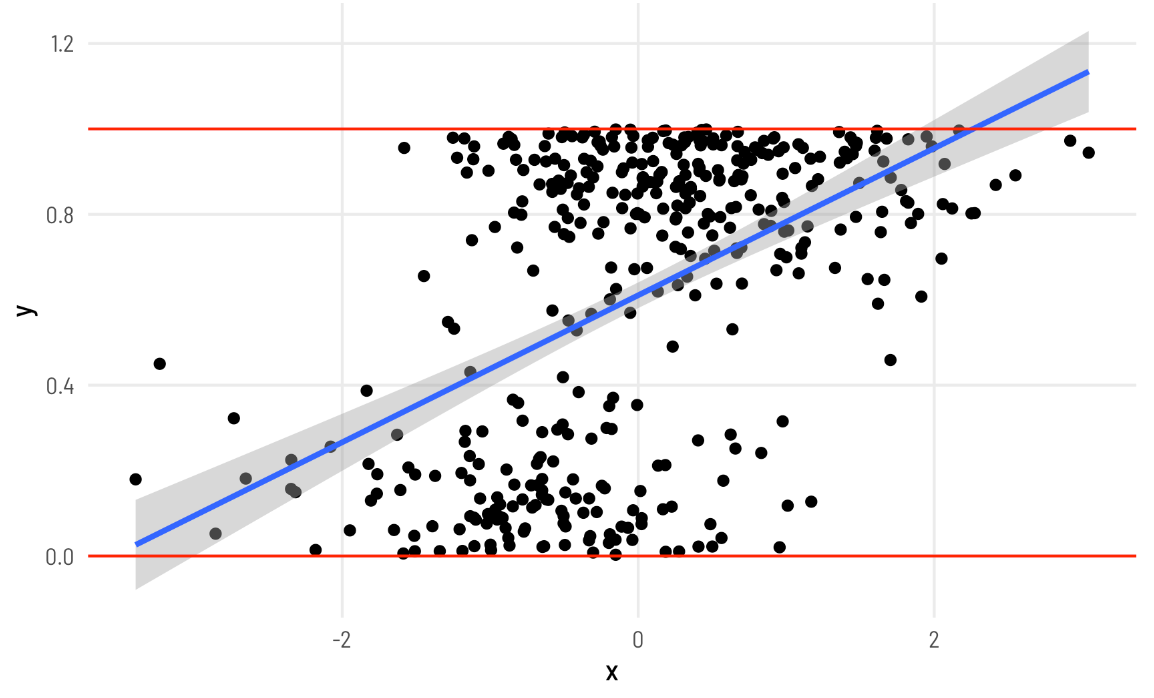
https://www.andrewheiss.com/blog/2021/11/08/beta-regression-guide/

In the data I work with, it’s really common to come across data that’s measured as proportions: the percent of women in the public sector workforce, the amount of foreign aid a country receives as a percent of its GDP, the percent of religious organizations in a state’s nonprofit sector, and so on.

在我使用的数据中，很常见的数据是以比例来衡量的：公共部门劳动力中女性的百分比，一个国家获得的外国援助金额占其GDP的百分比，以及一个州非营利部门宗教组织的百分比，等等。

When working with this kind of data as an outcome variable (or dependent variable) in a model, analysis gets tricky if you use standard models like OLS regression. For instance, look at this imaginary simulated dataset of the relationship between some hypothetical x and y:

当将这类数据作为模型中的结果变量（或因变量）处理时，如果使用OLS回归等标准模型，分析就会变得棘手。例如，看看这个假想的x和y之间关系的模拟数据集：



Y here is measured as a percentage and ranges between 0 and 1 (i.e. between the red lines), but the fitted line from a linear model creates predictions that exceed the 0–1 range (see the blue line in the top right corner). Since we’re thinking about proportions, we typically can’t exceed 0% and 100%, so it would be great if our modeling approach took those limits into account.

这里的Y是以百分比的形式测量的，范围在0和1之间（即红线之间），但线性模型的拟合线产生的预测超过了0–1范围（请参见右上角的蓝线）。由于我们考虑的是比例，我们通常不能超过0%和100%，所以如果我们的建模方法考虑到这些限制，那就太好了。

There are a bunch of different ways to model proportional data that vary substantially in complexity. In this post, I’ll explore four, but I’ll mostly focus on beta and zero-inflated beta regression:

有很多不同的方法来对比例数据进行建模，这些方法的复杂性差异很大。在这篇文章中，我将探讨四个，但我将主要关注贝塔和零膨胀贝塔回归：

**1.Linear probability models**

**2.Fractional logistic regression**

**3.Beta regression**

**4.Zero-inflated beta regression**

Throughout this example, we’ll use data from the Varieties of Democracy project (V-Dem) to answer a question common in comparative politics: do countries with parliamentary gender quotas (i.e. laws and regulations that require political parties to have a minimum proportion of women members of parliament (MPs)) have more women MPs in their respective parliaments? This question is inspired by existing research that looks at the effect of quotas on the proportion of women MPs, and a paper by Tripp and Kang (2008) was my first exposure to fractional logistic regression as a way to handle proportional data. The original data from that paper is available at Alice Kang’s website, but to simplify the post here, we won’t use it—we’ll just use a subset of equivalent data from V-Dem.

在整个例子中，我们将使用民主多样性项目（V-Dem）的数据来回答比较政治中常见的一个问题：有议会性别配额的国家（即要求政党在议会中女性议员比例最低的法律和法规）在各自的议会中有更多的女议员吗？这个问题的灵感来自于现有的研究，该研究着眼于配额对女议员比例的影响，Tripp和Kang（2008）的一篇论文是我第一次接触到分数逻辑回归作为处理比例数据的一种方法。该论文的原始数据可在Alice Kang的网站上获得，但为了简化本文，我们不会使用它——我们只使用V-Dem的等效数据子集。

There’s a convenient R package for accessing V-Dem data, so we’ll use that to make a smaller panel of countries between 2010 and 2020. Let’s load all the libraries we need, clean up the data, and get started!

有一个方便的R包用于访问V-Dem数据，因此我们将在2010年至2020年间使用它来创建一个较小的国家小组。让我们加载所有需要的库，清理数据，然后开始吧！

library(tidyverse) # ggplot, dplyr, %>%, and friends

library(brms) # Bayesian modeling through Stan

library(tidybayes) # Manipulate Stan objects in a tidy way

library(broom) # Convert model objects to data frames

library(broom.mixed) # Convert brms model objects to data frames

library(vdemdata) # Use data from the Varieties of Democracy (V-Dem) project

library(betareg) # Run beta regression models

library(extraDistr) # Use extra distributions like dprop()

library(ggdist) # Special geoms for posterior distributions

library(gghalves) # Special half geoms

library(ggbeeswarm) # Special distribution-shaped point jittering

library(ggrepel) # Automatically position labels

library(patchwork) # Combine ggplot objects

library(scales) # Format numbers in nice ways

library(marginaleffects) # Calculate marginal effects for regression models

library(modelsummary) # Create side-by-side regression tables

set.seed(1234) # Make everything reproducible

# Define the goodness-of-fit stats to include in modelsummary()

gof\_stuff <- tribble(

~raw, ~clean, ~fmt,

"nobs", "N", 0,

"r.squared", "R²", 3

)

# Custom ggplot theme to make pretty plots

# Get the font at https://fonts.google.com/specimen/Barlow+Semi+Condensed

theme\_clean <- function() {

theme\_minimal(base\_family = "Barlow Semi Condensed") +

theme(panel.grid.minor = element\_blank(),

plot.title = element\_text(family = "BarlowSemiCondensed-Bold"),

axis.title = element\_text(family = "BarlowSemiCondensed-Medium"),

strip.text = element\_text(family = "BarlowSemiCondensed-Bold",

size = rel(1), hjust = 0),

strip.background = element\_rect(fill = "grey80", color = NA))

}

# Make labels use Barlow by default

update\_geom\_defaults("label\_repel", list(family = "Barlow Semi Condensed"))

# Format things as percentage points

label\_pp <- label\_number(accuracy = 1, scale = 100,

suffix = " pp.", style\_negative = "minus")

label\_pp\_tiny <- label\_number(accuracy = 0.01, scale = 100,

suffix = " pp.", style\_negative = "minus")

V-Dem covers all countries since 1789 and includes hundreds of different variables. We’ll make a subset of some of the columns here and only look at years from 2010–2020. We’ll also make a subset of that and have a dataset for just 2015.

V-Dem涵盖1789年以来的所有国家，包括数百个不同的变量。我们将在这里对一些专栏做一个子集，只关注2010-2020年。我们还将制作其中的一个子集，并为2015年提供一个数据集。

# Make a subset of the full V-Dem data

vdem\_clean <- vdem %>%

select(country\_name, country\_text\_id, year, region = e\_regionpol\_6C,

polyarchy = v2x\_polyarchy, corruption = v2x\_corr,

civil\_liberties = v2x\_civlib, prop\_fem = v2lgfemleg, v2lgqugen) %>%

filter(year >= 2010, year < 2020) %>%

drop\_na(v2lgqugen, prop\_fem) %>%

mutate(quota = v2lgqugen > 0,

prop\_fem = prop\_fem / 100,

polyarchy = polyarchy \* 100)

vdem\_2015 <- vdem\_clean %>%

filter(year == 2015) %>%

# Sweden and Denmark are tied for the highest polyarchy score (91.5), and R's

# max() doesn't deal with ties, so we cheat a little and add a tiny random

# amount of noise to each polyarchy score, mark the min and max of that

# perturbed score, and then remove that temporary column

mutate(polyarchy\_noise = polyarchy + rnorm(n(), 0, sd = 0.01)) %>%

mutate(highlight = polyarchy\_noise == max(polyarchy\_noise) |

polyarchy\_noise == min(polyarchy\_noise)) %>%

select(-polyarchy\_noise)

**Explore the data**

Before trying to model this data, we’ll look at it really quick first. Here’s the distribution of prop\_fem across the two different values of quota. In general, it seems that countries without a gender-based quota have fewer women MPs, which isn’t all that surprising, since quotas were designed to boost the number of women MPs in the first place.

在尝试对这些数据进行建模之前，我们将首先快速查看它。以下是prop\_fem在两个不同的配额值之间的分布。总的来说，没有基于性别的配额的国家似乎女议员更少，这并不奇怪，因为配额最初是为了增加女议员的数量。

quota\_halves <- ggplot(vdem\_2015, aes(x = quota, y = prop\_fem)) +

geom\_half\_point(aes(color = quota),

transformation = position\_quasirandom(width = 0.1),

side = "l", size = 0.5, alpha = 0.5) +

geom\_half\_boxplot(aes(fill = quota), side = "r") +

scale\_y\_continuous(labels = label\_percent()) +

scale\_fill\_viridis\_d(option = "plasma", end = 0.8) +

scale\_color\_viridis\_d(option = "plasma", end = 0.8) +

guides(color = "none", fill = "none") +

labs(x = "Quota", y = "Proportion of women in parliament") +

theme\_clean()

quota\_densities <- ggplot(vdem\_2015, aes(x = prop\_fem, fill = quota)) +

geom\_density(alpha = 0.6) +

scale\_x\_continuous(labels = label\_percent()) +

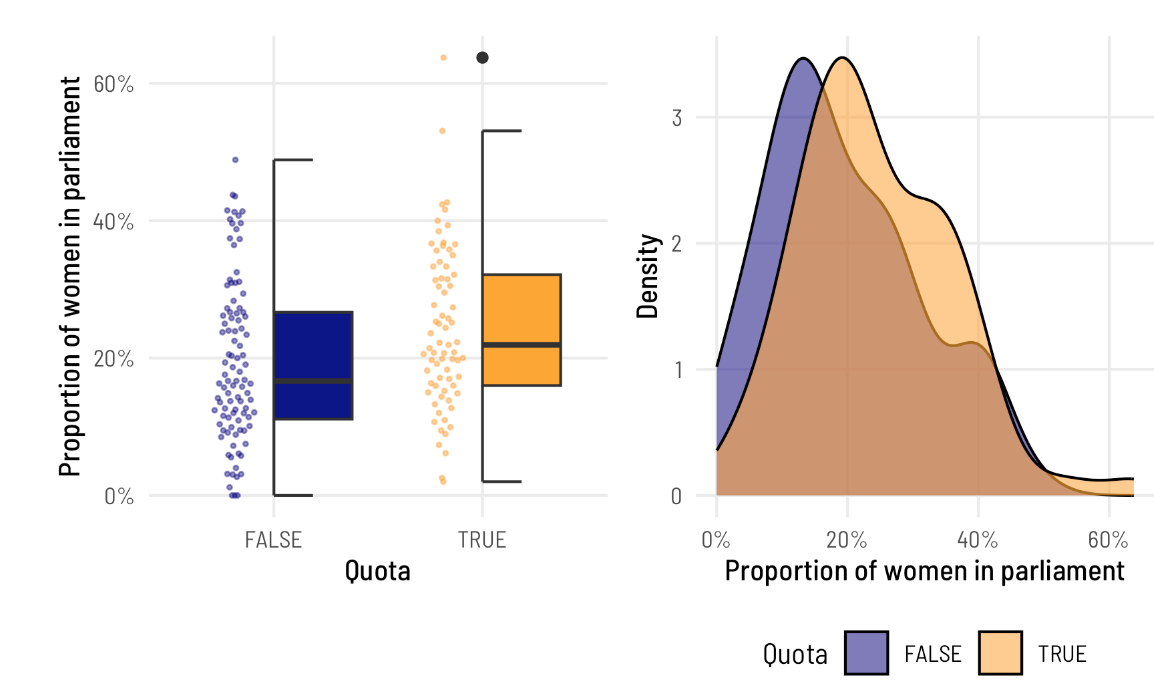
scale\_fill\_viridis\_d(option = "plasma", end = 0.8) +

labs(x = "Proportion of women in parliament", y = "Density", fill = "Quota") +

theme\_clean() +

theme(legend.position = "bottom")

quota\_halves | quota\_densities



And here’s what the proportion of women MPs looks like across different levels of electoral democracy. V-Dem’s polyarchy index measures the extent of electoral democracy in different countries, incorporating measures of electoral, freedom of association, and universal suffrage, among other factors. It ranges from 0 to 1, but to make it easier to interpret in regression, we’ll multiply it by 100 so we can talk about unit changes in democracy. To help with the intuition of the index, we’ll highlight the countries with the minimum and maximum values of democracy. In general, the proportion of women MPs increases as democracy increases.

以下是不同选举民主级别的女议员比例。V-Dem的多元民主指数衡量不同国家的选举民主程度，包括选举、结社自由和普选等因素。它的范围从0到1，但为了更容易在回归中解释，我们将它乘以100，这样我们就可以讨论民主中的单位变化。为了有助于该指数的直觉，我们将重点介绍民主价值最低和最高的国家。总的来说，女议员的比例随着民主的增加而增加。

ggplot(vdem\_2015, aes(x = polyarchy, y = prop\_fem)) +

geom\_point(aes(color = highlight), size = 1) +

geom\_smooth(method = "lm") +

geom\_label\_repel(data = filter(vdem\_2015, highlight == TRUE),

aes(label = country\_name),

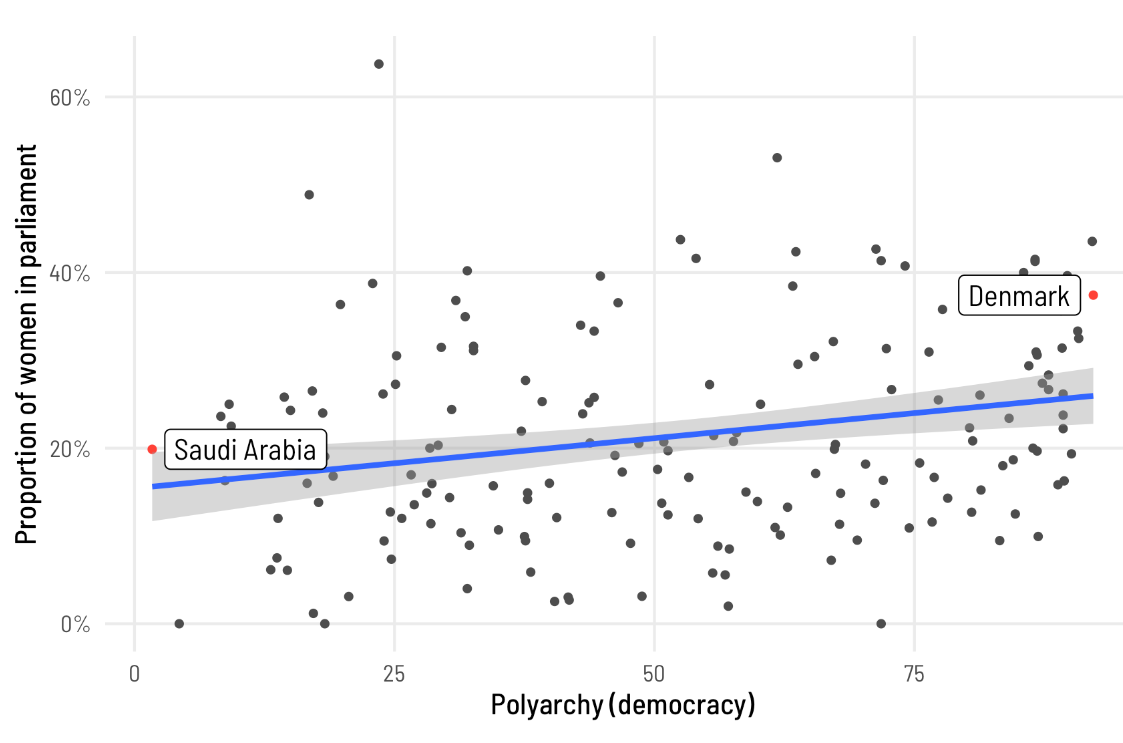
seed = 1234) +

scale\_y\_continuous(labels = label\_percent()) +

scale\_color\_manual(values = c("grey30", "#FF4136"), guide = "none") +

labs(x = "Polyarchy (democracy)", y = "Proportion of women in parliament") +

theme\_clean()



**1: Linear probability models**

In the world of econometrics, having the fitted regression line go outside the 0–1 range (like in the plot at the very beginning of this post) is totally fine and not a problem. Economists love to use a thing called a linear probability model (LPM) to model data like this, and for values that aren’t too extreme (generally if the outcome ranges between 0.2 and 0.8), the results from these weirdly fitting linear models are generally equivalent to fancier non-linear models. LPMs are even arguably best for experimental data. But it’s still a weird way to think about data and I don’t like them. As Alex Hayes says,

在计量经济学的世界里，让拟合的回归线超出0–1的范围（就像本文开头的图中一样）是完全可以的，不是问题。经济学家喜欢使用一种叫做线性概率模型（LPM）的东西来对这样的数据进行建模，对于不太极端的值（通常情况下，如果结果在0.2到0.8之间），这些奇怪拟合的线性模型的结果通常相当于更花哨的非线性模型。LPM甚至可以说是最适合实验数据的。但这仍然是一种奇怪的看待数据的方式，我不喜欢它们。正如Alex Hayes所说，

[T]he thing [LPM] is generally inconsistent and aesthetically offensive, but whatever, it works on occasion.

An LPM is really just regular old OLS applied to a binary or proportional outcome, so we’ll use trusty old lm().

LPM实际上只是应用于二进制或比例结果的常规旧OLS，所以我们将使用可靠的旧lm（）。

# Linear probability models

model\_ols1 <- lm(prop\_fem ~ quota,

data = vdem\_2015)

model\_ols2 <- lm(prop\_fem ~ polyarchy,

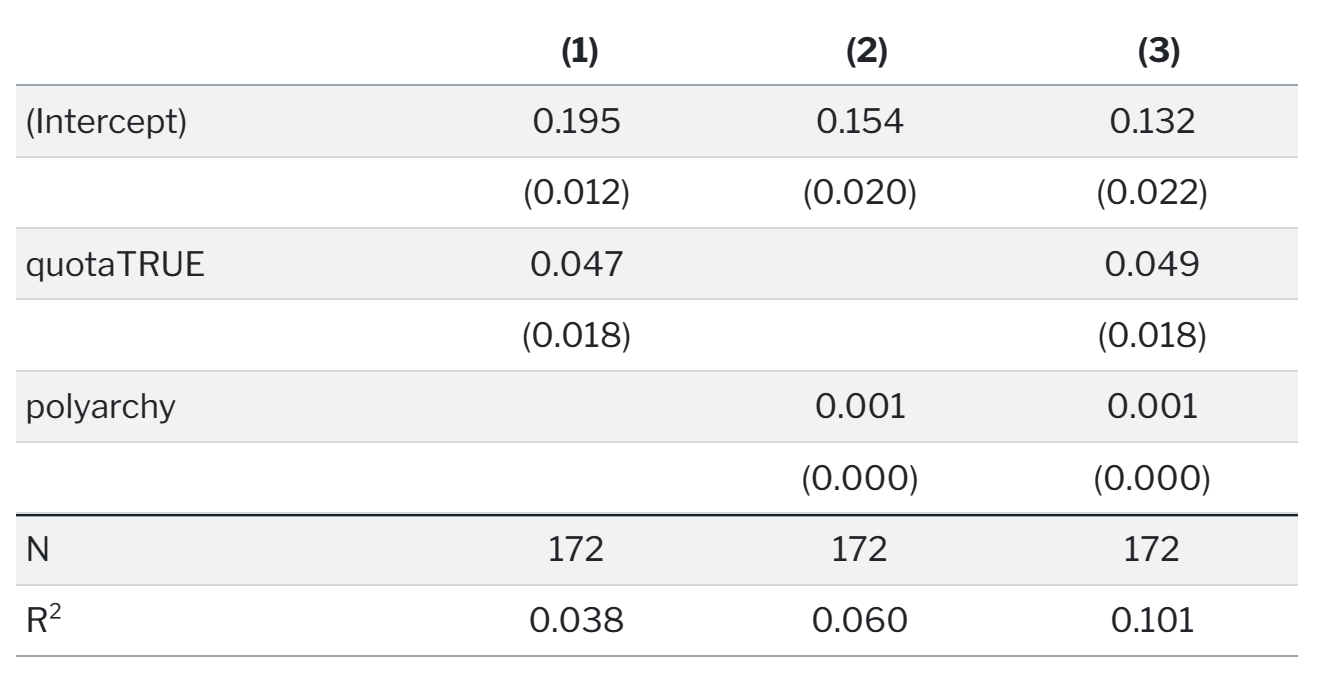
data = vdem\_2015)

model\_ols3 <- lm(prop\_fem ~ quota + polyarchy,

data = vdem\_2015)

modelsummary(list(model\_ols1, model\_ols2, model\_ols3),

gof\_map = gof\_stuff)



Based on this, having a quota is associated with a 4.7 percentage point increase in the proportion of women MPs on average, and the coefficient is statistically significant. If we control for polyarchy, the effect changes to 4.9 percentage points. Great.

基于此，拥有配额意味着女性议员的比例平均增加4.7个百分点，这一系数具有统计学意义。如果我们控制多元民主，效果会变为4.9个百分点。

**2: Fractional logistic regression**

Logistic regression is normally used for binary outcomes, but surprisingly you can actually use it for proportional data too! This kind of model is called fractional logistic regression, and though it feels weird to use logistic regression with non-binary data, it’s legal! The Stata documentation actually recommends it, and there are tutorials (like this and this) papers about how to do it with Stata. This really in-depth post by Michael Clark shows several ways to run these models with R.

逻辑回归通常用于二元结果，但令人惊讶的是，你实际上也可以将其用于比例数据！这种模型被称为分数逻辑回归，尽管对非二进制数据使用逻辑回归感觉很奇怪，但它是合法的！Stata文档实际上推荐了它，并且有关于如何使用Stata的教程（像这样和这样）。Michael Clark的这篇非常深入的帖子展示了用R运行这些模型的几种方法。

Basically use regular logistic regression with glm(..., family = binomial(link = "logit")) with an outcome variable that ranges from 0 to 1, and you’re done. R will give you a warning when you use family = binomial() with a non-binary outcome variable, but it will still work. If you want to suppress that warning, use family = quasibinomial() instead.

基本上使用glm（…，family=二项式（link=“logit”））的正则逻辑回归，结果变量范围从0到1，就完成了。当您将family=binomial（）与非二进制结果变量一起使用时，R会向您发出警告，但它仍然有效。如果要取消显示该警告，请改用family=qua-binomical（）。

model\_frac\_logit1 <- glm(prop\_fem ~ quota,

data = vdem\_2015,

family = quasibinomial())

model\_frac\_logit2 <- glm(prop\_fem ~ polyarchy,

data = vdem\_2015,

family = quasibinomial())

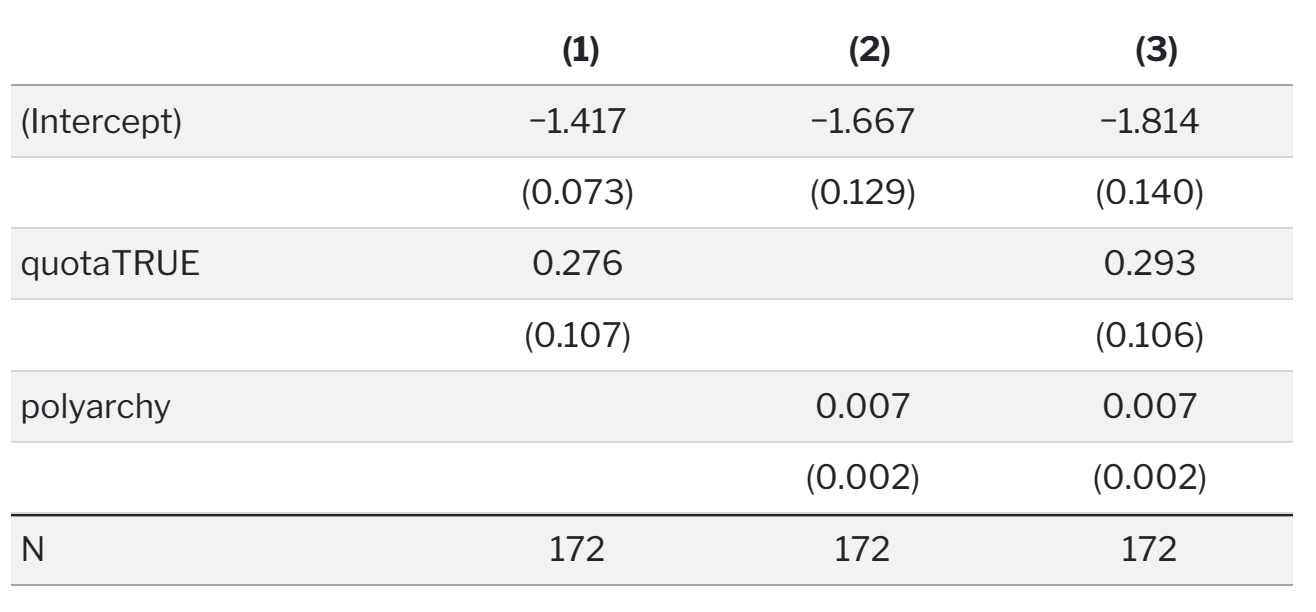
model\_frac\_logit3 <- glm(prop\_fem ~ quota + polyarchy,

data = vdem\_2015,

family = quasibinomial())

modelsummary(list(model\_frac\_logit1, model\_frac\_logit2, model\_frac\_logit3),

gof\_map = gof\_stuff)



Interpreting this model is a little more involved than interpreting OLS. The coefficients in logistic regression are provided on different scale than the original data. With OLS (and the linear probability model), coefficients show the change in the probability (or proportion in this case, since we’re working with proportion data) that is associated with a one-unit change in an explanatory variable. With logistic regression, coefficients show the change in the natural logged odds of the outcome, also known as “logits.”

解释这个模型比解释OLS要复杂一些。逻辑回归中的系数以不同于原始数据的尺度提供。对于OLS（和线性概率模型），系数显示了与解释变量的一个单位变化相关的概率（在这种情况下是比例，因为我们使用的是比例数据）的变化。通过逻辑回归，系数显示了结果的自然记录几率的变化，也称为“logits”

This log odds scale is weird and not very intuitive normally, so often people will convert these log odds into odds ratios by exponentiating them. I’ve done this my whole statistical-knowing-and-doing life. If a model provides a coefficient of 0.4, you can exponentiate that with , resulting in an odds ratio of 1.49. You’d interpret this by saying something like “a one unit change in x increases the likelihood of y by 49%.” That sounds neat and soundbite-y and you see stuff like this all the time in science reporting. But thinking about percent changes in probability is still a weird way of thinking (though admittedly less non-intuitive than thinking about logits/logged odds).

这种对数赔率表很奇怪，通常不太直观，所以人们通常会通过指数化将这些对数赔率转换为赔率比。我一生都在做这件事。如果一个模型提供了0.4的系数，你可以用来表示，得到1.49的比值比。你可以通过说“x的一个单位变化会使y的可能性增加49%”来解释这一点。这听起来很巧妙，很有道理，你在科学报道中经常看到这样的东西。但是，考虑概率的百分比变化仍然是一种奇怪的思维方式（尽管无可否认，这不如考虑logits/loged赔率那么直观）。

For instance, here the coefficient for quota in Model 1 is 0.276. If we exponentiate that () we get 1.32. If our outcome were binary, we could say that having a quota makes it 32% more likely that the outcome would occur. But our outcome isn’t binary—it’s a proportion—so we need to think about changes in probabilities or proportions instead, and odds ratios make that really really hard.

例如，模型1中的配额系数为0.276。如果我们对（）取幂，我们得到1.32。如果我们的结果是二元的，我们可以说，有一个配额会使结果发生的可能性增加32%。但我们的结果不是二元的，而是一个比例，所以我们需要考虑概率或比例的变化，而比值比让这变得非常困难。

Instead of odds ratios, we need to work directly with these logit-scale values. For a phenomenally excellent overview of binary logistic regression and how to interpret coefficients, see Steven Miller’s most excellent lab script here—because of that post I’m now far less afraid of dealing with logit-scale coefficients!

我们需要直接使用这些logit量表值，而不是比值比。关于二元逻辑回归以及如何解释系数的极好概述，请参阅Steven Miller最优秀的实验室脚本——因为这篇文章，我现在不再害怕处理logit尺度系数了！

We can invert a logit value and convert it to a probability using the plogis() function in R. We’ll illustrate this a couple different ways, starting with a super simple intercept-only model:

我们可以使用R中的plogis（）函数反转logit值并将其转换为概率。我们将以几种不同的方式对此进行说明，从一个超简单的仅截距模型开始：

model\_frac\_logit0 <- glm(prop\_fem ~ 1,

data = vdem\_2015,

family = quasibinomial())

tidy(model\_frac\_logit0)

## # A tibble: 1 × 5

## term estimate std.error statistic p.value

## <chr> <dbl> <dbl> <dbl> <dbl>

## 1 (Intercept) -1.29 0.0540 -24.0 1.24e-56

logit0\_intercept <- model\_frac\_logit0 %>%

tidy() %>%

filter(term == "(Intercept)") %>%

pull(estimate)

# The intercept in logits. Who even knows what this means.

logit0\_intercept

## [1] -1.29

The intercept of this intercept-only model is -1.295 in logit units, whatever that means. We can convert this to a probability though:

这种仅截距模型的截距为-1.295，单位为logit，不管这意味着什么。我们可以将其转换为概率：

# Convert logit to a probability (or proportion in this case)

plogis(logit0\_intercept)

## [1] 0.215

According to this probability/proportion, the average proportion of women MPs in the dataset is 0.215. Let’s see if that’s really the case:

根据这一概率/比例，数据集中女性议员的平均比例为0.215。让我们看看是否真的是这样：

mean(vdem\_2015$prop\_fem)

## [1] 0.215

It’s the same! That’s so neat!

Inverting logit coefficients to probabilities gets a little trickier when there are multiple moving parts, though. With OLS, each coefficient shows the marginal change in the outcome for each unit change in the explanatory variable. With logistic regression, that’s not the case—we have to incorporate information from the intercept in order to get marginal effects.

不过，当有多个运动部件时，将logit系数转化为概率会变得有点棘手。对于OLS，每个系数显示解释变量中每个单位变化的结果的边际变化。对于逻辑回归，情况并非如此——我们必须结合截距中的信息才能获得边际效应。

For example, in Model 1, the log odds coefficient for quota is 0.276. If we invert that with plogis(), we end up with a really big number:

例如，在模型1中，配额的对数优势系数为0.276。如果我们用plogis（）将其反转，我们最终会得到一个非常大的数字：

logit1\_intercept <- model\_frac\_logit1 %>%

tidy() %>%

filter(term == "(Intercept)") %>%

pull(estimate)

logit1\_quota <- model\_frac\_logit1 %>%

tidy() %>%

filter(term == "quotaTRUE") %>%

pull(estimate)

# Incorrect marginal effect of quota

plogis(logit1\_quota)

## [1] 0.568

If that were true, we could say that having a quota increases the proportion of women MPs by nearly 60%! But that’s wrong. We have to incorporate information from the intercept, as well as any other coefficients in the model, like so:

如果这是真的，我们可以说，有了配额，女议员的比例增加了近60%！但这是错误的。我们必须将来自截距的信息以及模型中的任何其他系数合并在一起，如下所示：

plogis(logit1\_intercept + logit1\_quota) - plogis(logit1\_intercept)

## [1] 0.0469

This is the effect of having a quota on the proportion of women MPs. Having a quota increases the proportion by 4.7 percentage points, on average.

这是配额对女议员比例的影响。有了配额，这一比例平均提高了4.7个百分点。

The math of combining these different logit coefficients and converting them to probabilities gets really tricky once there are more than one covariate, like in Model 3 here where we include both quota and polyarchy. Rather than try to figure out, we can instead calculate the marginal effects of specific variables while holding others constant. In practice this entails plugging a custom hypothetical dataset into our model and estimating the predicted values of the outcome. For instance, we can create a small dataset with just two rows: in one row, quota is FALSE and in the other quota is TRUE, while we hold polyarchy at its mean value. The marginaleffects package makes this really easy to do:

一旦有多个协变量，将这些不同的logit系数组合起来并将其转换为概率的数学运算就会变得非常棘手，比如在这里的模型3中，我们同时包括配额和多配偶制。我们可以计算特定变量的边际效应，而不是试图弄清楚，同时保持其他变量不变。在实践中，这需要将自定义的假设数据集插入我们的模型中，并估计结果的预测值。例如，我们可以创建一个只有两行的小数据集：在一行中，配额为FALSE，在另一行中配额为TRUE，而我们将多元民主保持在其平均值。marginalcoffects软件包让这项工作变得非常容易：

# Calculate the typical values for all variables in Model 3, except for quota,

# which we'll force to be FALSE and TRUE

data\_to\_predict <- datagrid(model = model\_frac\_logit3,

quota = c(FALSE, TRUE))

data\_to\_predict

## prop\_fem polyarchy quota

## 1 0.215 53.2 FALSE

## 2 0.215 53.2 TRUE

Next we can use predictions() from the marginaleffects package to plug this data into Model 3 and figure out the fitted values for each of the rows:

接下来，我们可以使用marginaleffects包中的predictions（）将这些数据插入模型3，并计算出每行的拟合值：

# Plug this new dataset into the model

frac\_logit\_pred <- predictions(model\_frac\_logit3, newdata = data\_to\_predict)

frac\_logit\_pred

##

## Estimate Pr(>|z|) 2.5 % 97.5 % polyarchy quota

## 0.192 <0.001 0.171 0.215 53.2 FALSE

## 0.242 <0.001 0.215 0.271 53.2 TRUE

##

## Columns: rowid, estimate, p.value, conf.low, conf.high, prop\_fem, polyarchy, quota

The estimate column here tells us the predicted or fitted value. When countries don’t have a quota, the average proportion of women MPs is 0.192; when they do have a quota, the average predicted proportion is 0.192. That’s a 4.96 percentage point difference, which is really similar to what we found from Model 1. We can plot these marginal effects too and see that this difference is statistically significant:

此处的估计列告诉我们预测值或拟合值。当国家没有配额时，女性议员的平均比例为0.192；当他们确实有配额时，平均预测比例为0.192。这是4.96个百分点的差异，与我们在模型1中发现的非常相似。我们也可以绘制这些边际效应，并看到这种差异在统计学上是显著的：

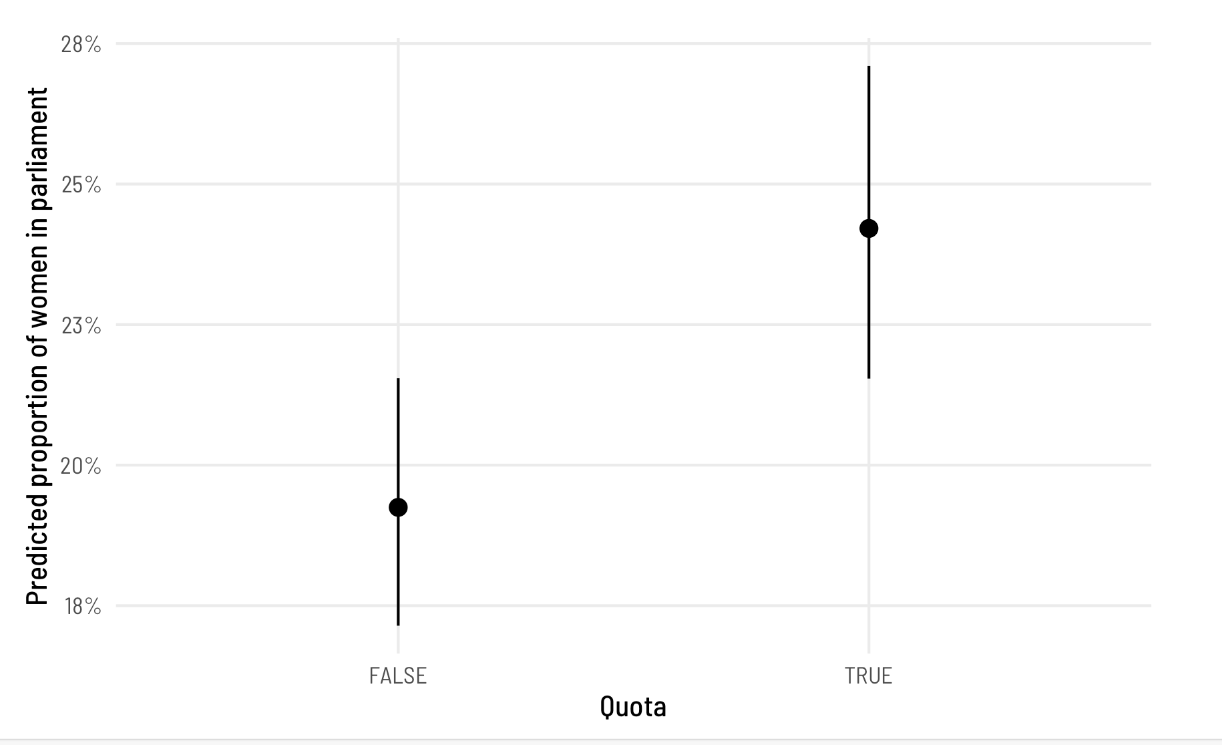
ggplot(frac\_logit\_pred, aes(x = quota, y = estimate)) +

geom\_pointrange(aes(ymin = conf.low, ymax = conf.high)) +

scale\_y\_continuous(labels = label\_percent(accuracy = 1)) +

labs(x = "Quota", y = "Predicted proportion of women in parliament") +

theme\_clean()



We can also compute marginal effects for polyarchy, holding quota at FALSE (its most common value) and ranging polyarchy from 0 to 100. And we technically don’t need to make a separate data frame first—we can use datagrid() inside predictions():

我们还可以计算多元民主的边际效应，将配额保持在FALSE（其最常见的值），并将多元民主从0到100。从技术上讲，我们不需要先制作一个单独的数据帧——我们可以在predictions（）中使用datagrid（）：

frac\_logit\_pred <- predictions(model\_frac\_logit3,

newdata = datagrid(polyarchy = seq(0, 100, by = 1)))

ggplot(frac\_logit\_pred, aes(x = polyarchy, y = estimate)) +

geom\_ribbon(aes(ymin = conf.low, ymax = conf.high),

alpha = 0.4, fill = "#480B6A") +

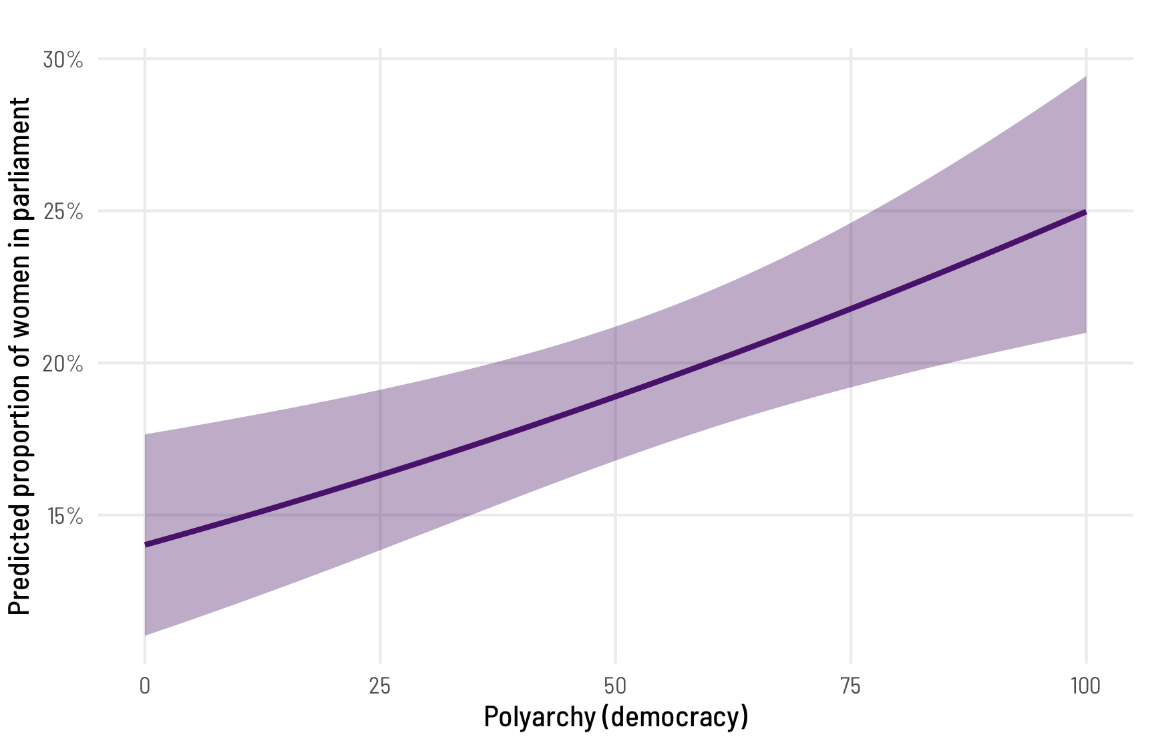
geom\_line(linewidth = 1, color = "#480B6A") +

scale\_y\_continuous(labels = label\_percent()) +

labs(x = "Polyarchy (democracy)",

y = "Predicted proportion of women in parliament") +

theme\_clean()



The predicted proportion of women MPs increases as democracy increases, which follows from the model results. But what’s the marginal effect of a one-unit increase in polyarchy? The coefficient from Model 3 was really small (0.007 logits). We can convert this to a probability while correctly accounting for the intercept and all other coefficients with the avg\_slopes() function:

根据模型结果，预测的女议员比例随着民主程度的提高而增加。但是，增加一个单位的多元民主的边际效应是什么？模型3的系数非常小（0.007 logits）。我们可以将其转换为概率，同时使用avg\_slopes（）函数正确计算截距和所有其他系数：

avg\_slopes(model\_frac\_logit3, variables = "polyarchy")

##

## Term Estimate Std. Error z Pr(>|z|) 2.5 % 97.5 %

## polyarchy 0.00119 0.000349 3.42 <0.001 0.00051 0.00188

##

## Columns: term, estimate, std.error, statistic, p.value, conf.low, conf.high

The effect here is still small: a one-unit change in polyarchy (i.e. moving from a 56 to a 57) is associated with a 0.119 percentage point increase in the proportion of women MPs, on average. That’s not huge, but a one-unit change in polyarchy also isn’t huge. If a country moves 10 points from 56 to 66, for instance, there’d be a ≈10x increase in the effect (0.001 × 10, or 0.012). This is apparent in the predicted proportion when polyarchy is set to 56 and 66: 0.207 - 0.196 = 0.011

这里的影响仍然很小：多元民主的一个单位变化（即从56岁变为57岁）与女性议员比例平均增加0.119个百分点有关。这并不算大，但多元民主中的一个单位的变化也不算大。例如，如果一个国家从56个百分点上升到66个百分点，其影响将增加≈10倍（0.001×10，或0.012）。这一点在多元民主设定为56和66时的预测比例中很明显：0.207-0.196=0.011

predictions(model\_frac\_logit3,

newdata = datagrid(polyarchy = c(56, 66)))

##

## Estimate Pr(>|z|) 2.5 % 97.5 % quota polyarchy

## 0.196 <0.001 0.174 0.219 FALSE 56

## 0.207 <0.001 0.184 0.232 FALSE 66

##

## Columns: rowid, estimate, p.value, conf.low, conf.high, prop\_fem, quota, polyarchy

Finally, how do these fractional logistic regression coefficients compare to what we found with the linear probability model? We can look at them side-by-side:

最后，这些分数逻辑回归系数与我们在线性概率模型中的发现相比如何？我们可以并排来看：

modelsummary(list("OLS" = model\_ols3,

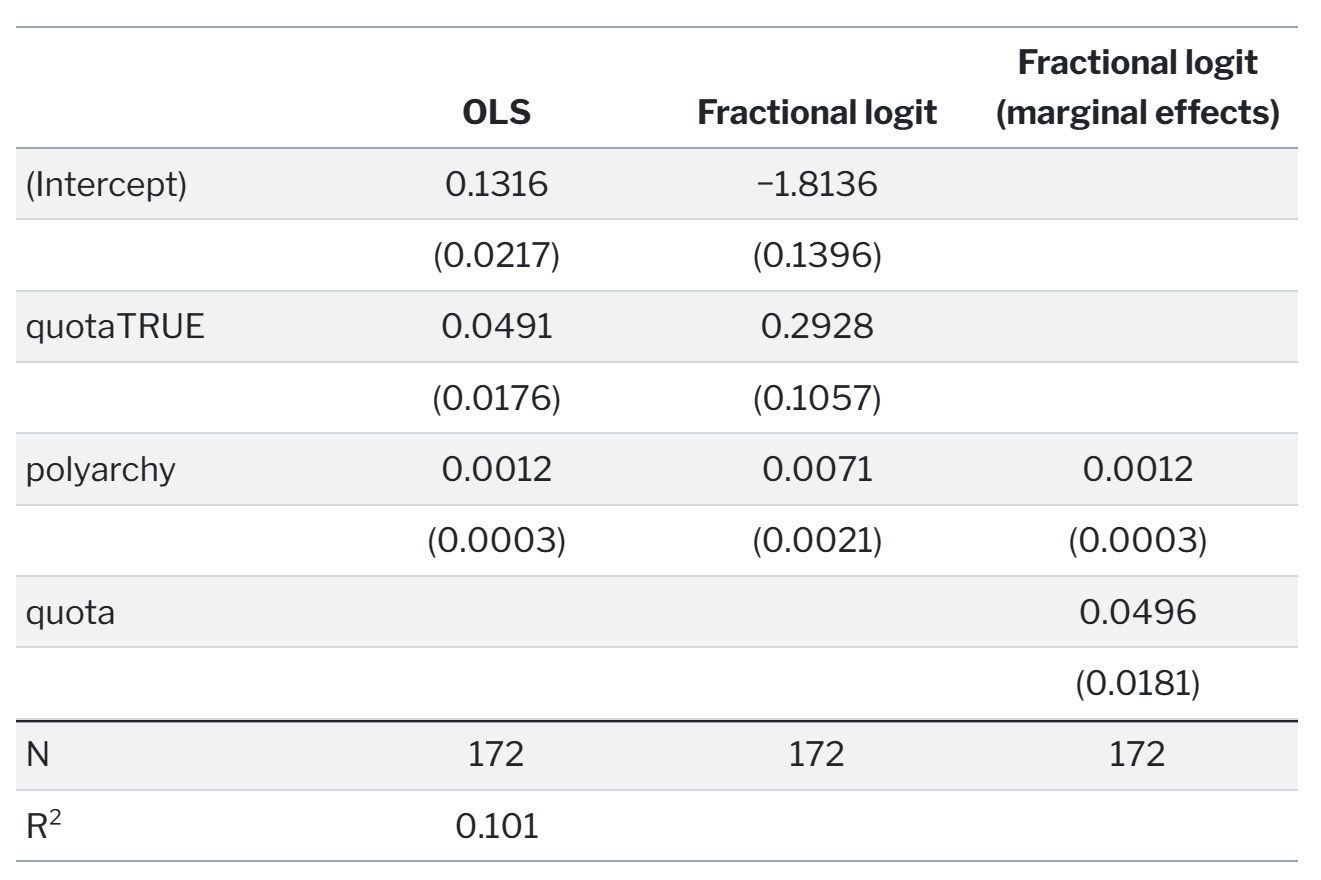
"Fractional logit" = model\_frac\_logit3,

"Fractional logit<br>(marginal effects)" = marginaleffects(model\_frac\_logit3)),

gof\_map = gof\_stuff,

fmt = 4,

escape = FALSE)



They’re basically the same! Having a quota is associated with a 4.96 percentage point increase in the proportion of women MPs, on average, in both OLS and the fractional logit model. That’s so neat!

它们基本上是一样的！在OLS和分数logit模型中，拥有配额与女性议员比例平均增加4.96个百分点有关。太整洁了！

**Interlude: The beta distribution and distributional regression**

The OLS-based linear probability model works (begrudgingly), and fractional logistic regression works (but it feels kinda weird to use logistic regression on proportions like that). The third method we’ll explore is beta regression, which is a little different from linear regression. With regular old linear regression, we’re essentially fitting a line to data. We find an intercept, we find a slope, and we draw a line. We can add some mathematical trickery to make sure the line stays within specific bounds (like using logits and logged odds), but fundamentally everything just has intercepts and slopes.

基于OLS的线性概率模型有效（令人不快），分数逻辑回归有效（但在这样的比例上使用逻辑回归感觉有点奇怪）。我们将探索的第三种方法是贝塔回归，它与线性回归有点不同。使用常规的旧线性回归，我们本质上是在拟合数据的一条线。我们找到一个截距，找到一个斜率，然后画一条线。我们可以添加一些数学技巧来确保线保持在特定的范围内（比如使用logits和logged赔率），但从根本上讲，一切都只有截距和斜率。

An alternative to this kind of mean-focused regression is to use something called distributional regression (Kneib, Silbersdorff, and Säfken 2021) to estimate not lines and averages, but the overall shapes of statistical distributions.

这种以均值为中心的回归的另一种选择是使用所谓的分布回归（Kneib、Silbersdorff和Säfken 2021）来估计统计分布的总体形状，而不是直线和平均值。

Beta regression is one type of distributional regression, but before we explore how it works, we have to briefly review how distributions work.

贝塔回归是分布回归的一种，但在我们探索它的工作原理之前，我们必须简要回顾一下分布是如何工作的。

In statistics, there are all sorts of probability distributions that can represent the general shape and properties of a variable (I have a whole guide for generating data using the most common distributions here). For instance, in a normal distribution, most of the values are clustered around a mean, and values are spread out based on some amount of variance, or standard deviation.

在统计学中，有各种各样的概率分布可以表示变量的一般形状和属性（我在这里有一个使用最常见分布生成数据的完整指南）。例如，在正态分布中，大多数值都聚集在平均值周围，并且值是基于一定量的方差或标准差分布的。

Here are two normal distributions defined by different means and standard deviations. One is centered at 5 with a fairly wide standard deviation of 4 (so 95% of its values range from 5 ± (4 × 1.96), or from −3ish to 13ish), while the other is centered at 10 with a narrower standard deviation of 2 (so 95% of its values range from 10 ± (2 × 1.96), or from 6 to 14):

这里有两个由不同均值和标准差定义的正态分布。一个以5为中心，标准偏差相当大，为4（因此其95%的值范围为5±（4×1.96），或−3ish至13ish），而另一个以10为中心，其标准偏差较窄，为2（因此其值的95%范围为10±（2×1.96，或6至14）：

ggplot(data = tibble(x = seq(-10, 20)), aes(x = x)) +

geom\_function(fun = dnorm, args = list(mean = 5, sd = 4),

aes(color = "Normal(mean = 5, sd = 4)"), linewidth = 1) +

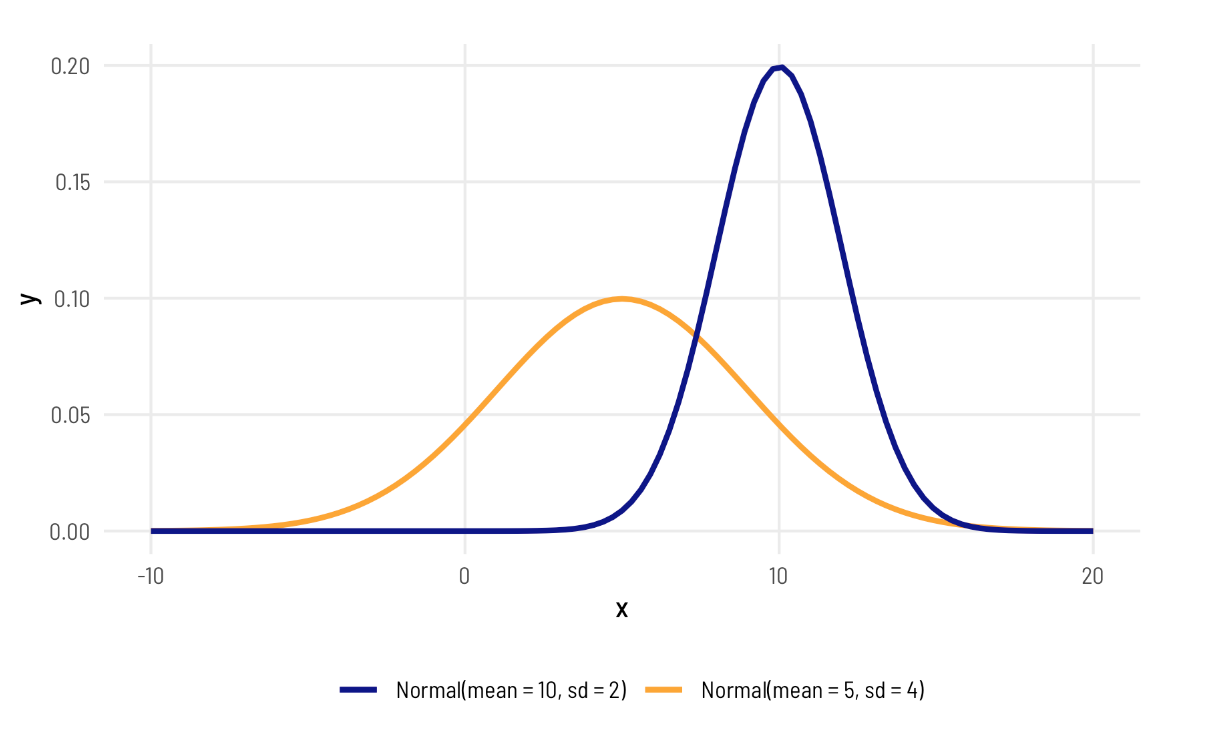
geom\_function(fun = dnorm, args = list(mean = 10, sd = 2),

aes(color = "Normal(mean = 10, sd = 2)"), linewidth = 1) +

scale\_color\_viridis\_d(option = "plasma", end = 0.8, name = "") +

theme\_clean() +

theme(legend.position = "bottom")



Most other distributions are defined in similar ways: there’s a central tendency (often the mean), and there’s some spread or variation around that center.

大多数其他分布都是以类似的方式定义的：有一个中心趋势（通常是均值），并且在这个中心周围有一些传播或变化。

**Beta distributions and shape parameters**

One really common distribution that’s perfect for percentages and proportions is the beta distribution, which is naturally limited to numbers between 0 and 1 (but importantly doesn’t include 0 or 1). The beta distribution is an extremely flexible distribution and can take all sorts of different shapes and forms (stare at this amazing animated GIF for a while to see all the different shapes!)

一个非常适合百分比和比例的真正常见的分布是贝塔分布，它自然局限于0到1之间的数字（但重要的是不包括0或1）。贝塔分布是一个非常灵活的分布，可以采取各种不同的形状和形式（盯着这个令人惊叹的动画GIF看一会儿，可以看到所有不同的形状！）

Unlike a normal distribution, where you use the mean and standard deviation as the distributional parameters, beta distributions take two non-intuitive parameters: (1) shape1 and (2) shape2, often abbreviated as

and

. This answer at Cross Validated does an excellent job of explaining the intuition behind beta distributions and it’d be worth it to read it. Go do that—I’ll wait.

与使用均值和标准差作为分布参数的正态分布不同，贝塔分布采用两个非直观参数：（1）shape1和（2）shape2，通常缩写为

和

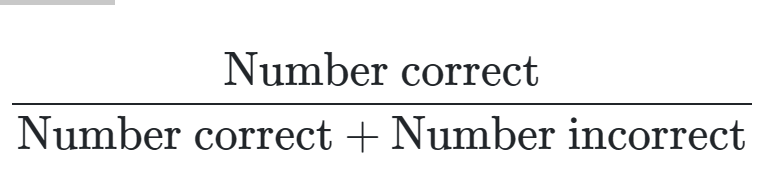
。Cross Validated的这个答案很好地解释了测试版发行版背后的直觉，值得一读。去做吧，我等着。

Basically beta distributions are good at modeling the probabilities of things, and shape1 and shape2 represent specific parts of a formula for probabilities and proportions.

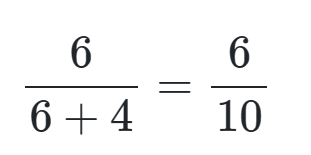
基本上，贝塔分布善于对事物的概率进行建模，shape1和shape2代表概率和比例公式的特定部分。

Let’s say that there’s an exam with 10 points where most people score a 6/10. Another way to think about this is that an exam is a collection of correct answers and incorrect answers, and that the percent correct follows this equation:

假设有一个10分的考试，大多数人的分数是6/10。另一种思考方式是，考试是正确答案和错误答案的集合，正确率遵循以下等式：



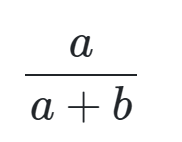
If you scored a 6, you could write that as:



To make this formula more general, we can use variable names:

a for the number correct and

b for the number incorrect, leaving us with this:



In a beta distribution, the a

and the b

in that equation correspond to the shape1 and shape2 parameters. If we want to look at the distribution of scores for this test where most people get 6/10, or 60%, we can use 6 and 4 as parameters. Most people score around 60%, and the distribution isn’t centered—it’s asymmetric. Neat!

在β分布中，该方程中的a和b对应于shape1和shape2参数。如果我们想看看这个测试的分数分布，大多数人得到6/10，或60%，我们可以使用6和4作为参数。大多数人的得分在60%左右，而且分布不是集中的，而是不对称的。整洁的

ggplot() +

geom\_function(fun = dbeta, args = list(shape1 = 6, shape2 = 4),

aes(color = "Beta(shape1 = 6, shape2 = 4)"),

size = 1) +

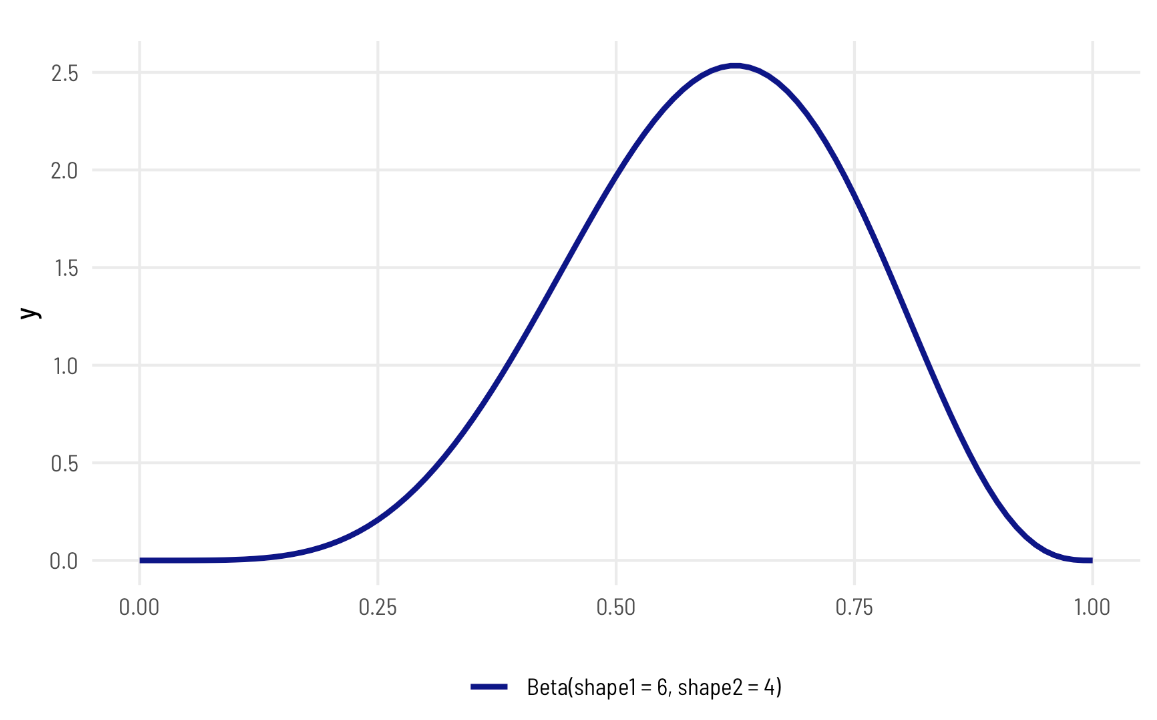
scale\_color\_viridis\_d(option = "plasma", name = "") +

theme\_clean() +

theme(legend.position = "bottom")

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.

## ℹ Please use `linewidth` instead.



The magic of—and most confusing part about—beta distributions is that you can get all sorts of curves by just changing the shape parameters. To make this easier to see, we can make a bunch of different beta distributions.

贝塔分布的神奇之处——也是最令人困惑的地方——是你只需改变形状参数就可以得到各种曲线。为了更容易看到这一点，我们可以制作一系列不同的测试版。

ggplot() +

geom\_function(fun = dbeta, args = list(shape1 = 6, shape2 = 4),

aes(color = "Beta(shape1 = 6, shape2 = 4)"),

size = 1) +

geom\_function(fun = dbeta, args = list(shape1 = 60, shape2 = 40),

aes(color = "Beta(shape1 = 60, shape2 = 40)"),

size = 1) +

geom\_function(fun = dbeta, args = list(shape1 = 9, shape2 = 1),

aes(color = "Beta(shape1 = 9, shape2 = 1)"),

size = 1) +

geom\_function(fun = dbeta, args = list(shape1 = 2, shape2 = 11),

aes(color = "Beta(shape1 = 2, shape2 = 11)"),

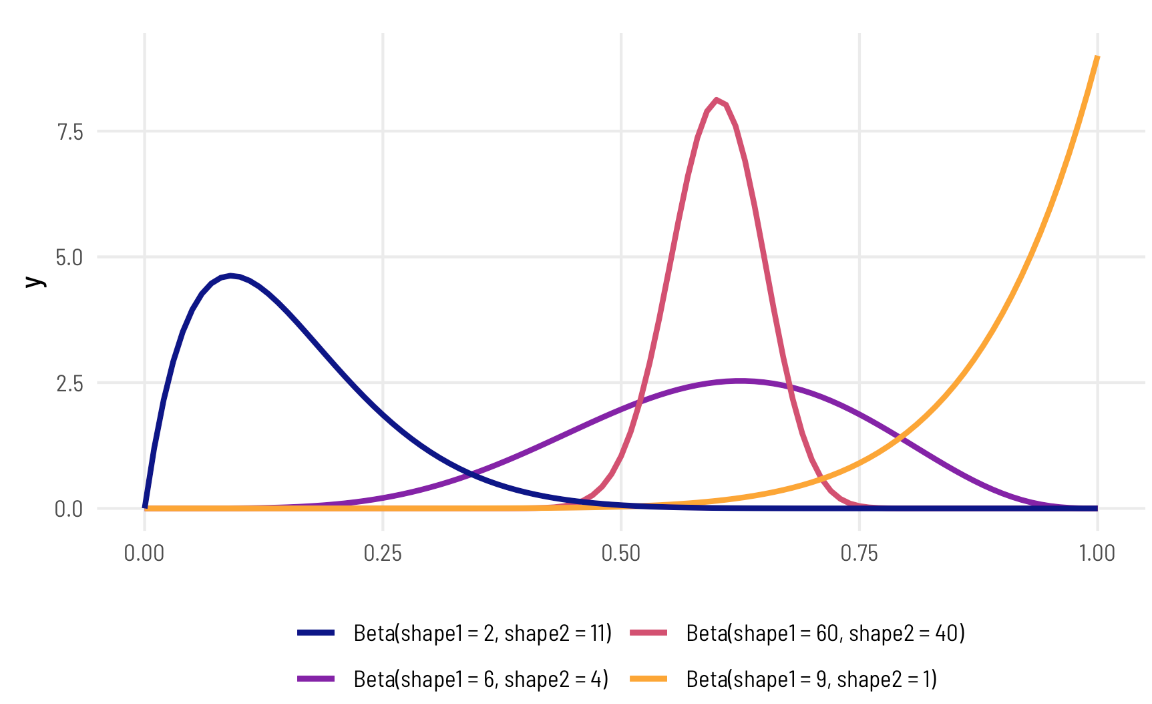
size = 1) +

scale\_color\_viridis\_d(option = "plasma", end = 0.8, name = "",

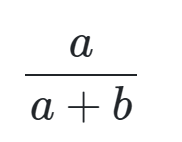
guide = guide\_legend(nrow = 2)) +

theme\_clean() +

theme(legend.position = "bottom")



To figure out the center of each of these distributions, think of the

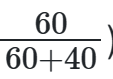


formula. For the blue distribution on the far left, for instance, it’s



or 0.154. The orange distribution on the far right is centered at

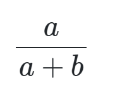


, or 0.9. The tall pink-ish distribution is centered at 0.6 (

), just like the distribution, but it’s much narrower and less spread out. When working with these two shape parameters, you control the variance or spread of the distribution by scaling the values up or down.

粉红色的高分布集中在0.6（ )，就像 分布一样，但它更窄，分布更少。使用这两个形状参数时，可以通过向上或向下缩放值来控制分布的方差或扩散。

**Mean and precision instead of shapes**

But thinking about these shapes and manually doing the 

calculation in your head is hard! It’s even harder to get a specific amount of spread. Most other distributions can be defined with a center and some amount of spread or variance, but with beta distributions you’re stuck with these weirdly interacting shape parameters.

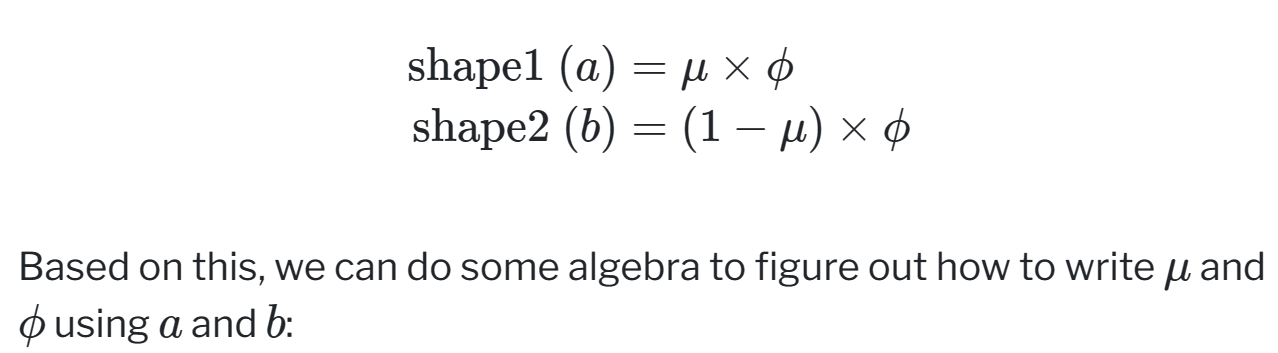
Fortunately there’s an alternative way of parameterizing the beta distribution that uses a mean μ and precision  (the same idea as variance) instead of these strange shapes.

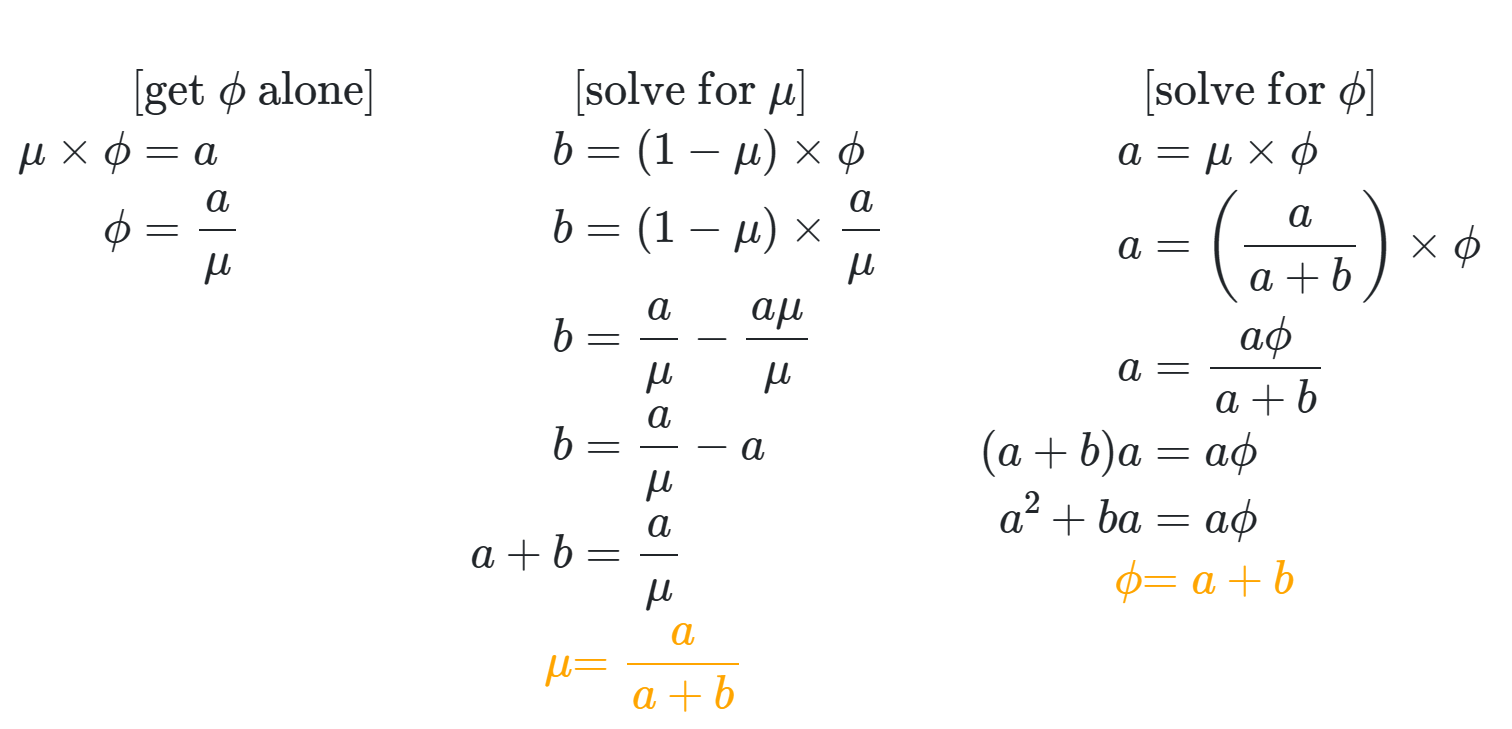
These shapes and the μ and parameters are mathematically related and interchangeable. Formally, the two shapes can be defined using μ and 

like so:

但是考虑这些形状并手动执行 你脑子里的计算很难！更难获得特定的传播量。大多数其他分布可以用一个中心和一些扩散或方差来定义，但对于贝塔分布，你会被这些奇怪的相互作用形状参数所困扰。

幸运的是，有一种替代方法可以参数化β分布，它使用平均值μ和精度（与方差的概念相同），而不是这些奇怪的形状。 这些形状、μ和参数在数学上是相关的，并且可以互换。形式上，这两种形状可以使用μ和 就像这样：





It’s thus possible to translate between these two parameterizations:



To help with the intuition, we can make a couple little functions to switch between them.

shapes\_to\_muphi <- function(shape1, shape2) {

mu <- shape1 / (shape1 + shape2)

phi <- shape1 + shape2

return(list(mu = mu, phi = phi))

}

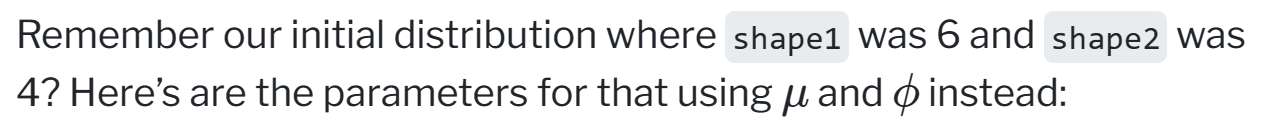
muphi\_to\_shapes <- function(mu, phi) {

shape1 <- mu \* phi

shape2 <- (1 - mu) \* phi

return(list(shape1 = shape1, shape2 = shape2))

}



shapes\_to\_muphi(6, 4)

## $mu

## [1] 0.6

##

## $phi

## [1] 10

It has a mean of 0.6 and a precision of 10. That more precise and taller distribution where shape1 was 60 and shape2 was 40?

shapes\_to\_muphi(60, 40)

## $mu

## [1] 0.6

##

## $phi

## [1] 100

It has the same mean of 0.6, but a much higher precision (100 now instead of 10).

R has built-in support for the shape-based beta distribution with things like dbeta(), rbeta(), etc. We can work with this reparameterized μ - and -based beta distribution using the dprop() (and rprop(), etc.) from the extraDistr package. It takes two arguments: size for  and mean for μ .

它的平均值为0.6，但精度要高得多（现在是100，而不是10）。

R内置了对基于形状的beta分布的支持，比如dbeta（）、rbeta（等。我们可以使用extraDistr包中的dprop（和rprop（）等）来处理这种重新参数化的μ和基于μ的beta分布。它需要两个参数：的大小和μ的平均值

beta\_shapes <- ggplot() +

geom\_function(fun = dbeta, args = list(shape1 = 6, shape2 = 4),

aes(color = "dbeta(shape1 = 6, shape2 = 4)"),

size = 1) +

geom\_function(fun = dbeta, args = list(shape1 = 60, shape2 = 40),

aes(color = "dbeta(shape1 = 60, shape2 = 40)"),

size = 1) +

geom\_function(fun = dbeta, args = list(shape1 = 9, shape2 = 1),

aes(color = "dbeta(shape1 = 9, shape2 = 1)"),

size = 1) +

geom\_function(fun = dbeta, args = list(shape1 = 2, shape2 = 11),

aes(color = "dbeta(shape1 = 2, shape2 = 11)"),

size = 1) +

scale\_color\_viridis\_d(option = "plasma", end = 0.8, name = "",

guide = guide\_legend(ncol = 1)) +

labs(title = "Shape-based beta distributions") +

theme\_clean() +

theme(legend.position = "bottom")

beta\_mu\_phi <- ggplot() +

geom\_function(fun = dprop, args = list(mean = 0.6, size = 10),

aes(color = "dprop(mean = 0.6, size = 10)"),

size = 1) +

geom\_function(fun = dprop, args = list(mean = 0.6, size = 100),

aes(color = "dprop(mean = 0.6, size = 100)"),

size = 1) +

geom\_function(fun = dprop, args = list(mean = 0.9, size = 10),

aes(color = "dprop(mean = 0.9, size = 10)"),

size = 1) +

geom\_function(fun = dprop, args = list(mean = 0.154, size = 13),

aes(color = "dprop(mean = 0.154, size = 13)"),

size = 1) +

scale\_color\_viridis\_d(option = "plasma", end = 0.8, name = "",

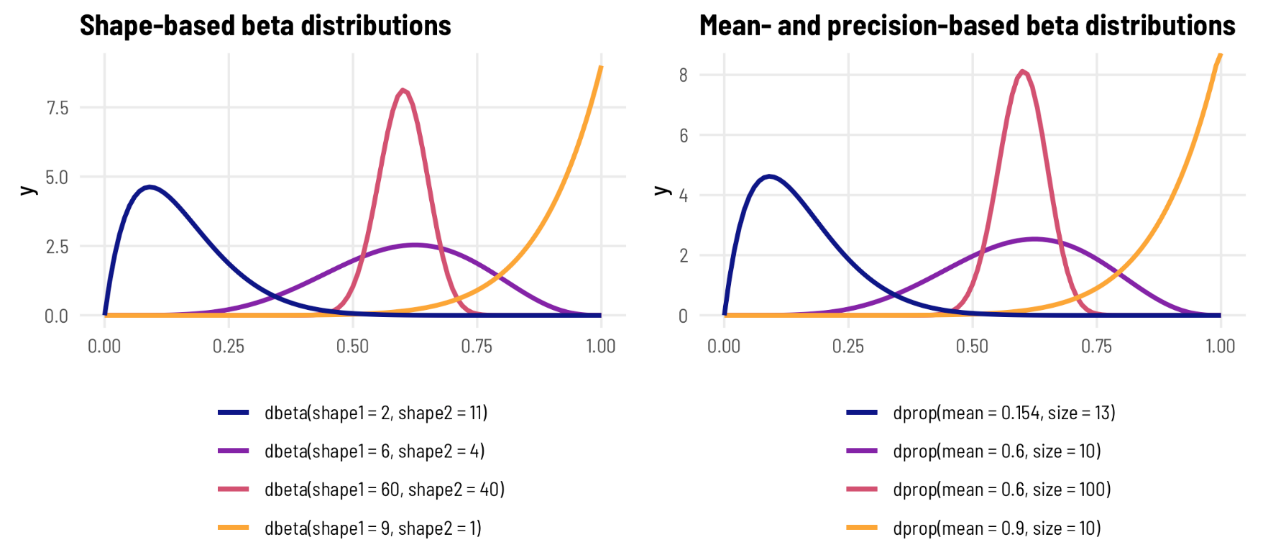
guide = guide\_legend(ncol = 1)) +

labs(title = "Mean- and precision-based beta distributions") +

theme\_clean() +

theme(legend.position = "bottom")

beta\_shapes | beta\_mu\_phi



Phew. That’s a lot of background to basically say that you don’t have to think about the shape parameters for beta distributions—you can think of mean (μ

) and precision (

) instead, just like you do with a normal distribution and its mean and standard deviation.

Phew。这是一个很大的背景，基本上说你不必考虑β分布的形状参数——你可以考虑平均值（μ)和精度（)相反，就像处理正态分布及其均值和标准差一样。

**3a: Beta regression**

So, with that quick background on how beta distributions work, we can now explore how beta regression lets us model outcomes that range between 0 and 1. Again, beta regression is a distributional regression, which means we’re ultimately modeling

μ and Φ and not just a slope and intercept. With this regression, we’ll be able to fit separate models for both parameters and see how the overall distribution shifts based on different covariates.

因此，有了关于贝塔分布如何工作的快速背景，我们现在可以探索贝塔回归如何让我们对0到1之间的结果进行建模。同样，贝塔回归是一种分布回归，这意味着我们最终要建模μ和Φ，而不仅仅是斜率和截距。通过这种回归，我们将能够为这两个参数拟合单独的模型，并了解总体分布如何基于不同的协变量而变化。

The betareg package provides the betareg() function for doing frequentist beta regression. For the sake of simplicity, we’ll just look at prop\_fem ~ quota and ignore polyarchy for a bit.

betareg包提供了betareg（）函数，用于进行频率学派的贝塔回归。为了简单起见，我们只看prop\_fem~quota，暂时忽略多元民主。

The syntax is just like all other formula-based regression functions, but with an added bit: the first part of the equation (prop\_fem ~ quota) models the mean, or μ, while anything that comes after a | in the formula will explain variation in the precision, or Φ.

语法与所有其他基于公式的回归函数一样，但增加了一点：方程的第一部分（prop\_fem~quota）对平均值或μ进行建模，而公式中a|之后的任何内容都将解释精度或Φ的变化。

Let’s try it out!

model\_beta <- betareg(prop\_fem ~ quota | quota,

data = vdem\_2015,

link = "logit")

## Error in betareg(prop\_fem ~ quota | quota, data = vdem\_2015, link = "logit"): invalid dependent variable, all observations must be in (0, 1)

Oh no! We have an error. Beta regression can only handle outcome values that range between 0 and 1—it cannot deal with values that are exactly 0 or exactly 1.

哦，不！我们有一个错误。Beta回归只能处理介于0和1之间的结果值，而不能处理恰好为0或1的值。

We’ll show how to deal with 0s and 1s later. For now we can do some trickery and add 0.001 to all the 0s so that they’re not actually 0. This is cheating, but it’s fine for now :)

稍后我们将展示如何处理0和1。现在我们可以做一些小把戏，在所有的0上加0.001，这样它们实际上就不是0。这是作弊，但目前还可以：）

vdem\_2015\_fake0 <- vdem\_2015 %>%

mutate(prop\_fem = ifelse(prop\_fem == 0, 0.001, prop\_fem))

model\_beta <- betareg(prop\_fem ~ quota | quota,

data = vdem\_2015\_fake0,

link = "logit")

tidy(model\_beta)

## # A tibble: 4 × 6

## component term estimate std.error statistic p.value

## <chr> <chr> <dbl> <dbl> <dbl> <dbl>

## 1 mean (Intercept) -1.44 0.0861 -16.7 7.80e-63

## 2 mean quotaTRUE 0.296 0.115 2.58 9.79e- 3

## 3 precision (Intercept) 2.04 0.140 14.6 2.42e-48

## 4 precision quotaTRUE 0.440 0.214 2.06 3.93e- 2

**Interpreting coefficients**

We now have two sets of coefficients, one set for each parameter (the mean and precision). The parameters for the mean are measured on the logit scale, just like with logistic regression previously, and we can calculate the marginal effect of having a quota by using plogis() and piecing together the coefficient and the intercept:

我们现在有两组系数，每个参数（平均值和精度）一组。平均值的参数是在logit量表上测量的，就像以前的逻辑回归一样，我们可以通过使用plogis（）并将系数和截距拼凑在一起来计算配额的边际效应：

beta\_mu\_intercept <- model\_beta %>%

tidy() %>%

filter(component == "mean", term == "(Intercept)") %>%

pull(estimate)

beta\_mu\_quota <- model\_beta %>%

tidy() %>%

filter(component == "mean", term == "quotaTRUE") %>%

pull(estimate)

plogis(beta\_mu\_intercept + beta\_mu\_quota) - plogis(beta\_mu\_intercept)

## [1] 0.0501

Having a quota thus increases the average of the distribution of women MPs by 5.006 percentage points, on average. That’s the same value that we found with OLS and with fractional logistic regression!

因此，有了配额，女议员的平均分布平均增加了5.006个百分点。这与我们在OLS和分数逻辑回归中发现的值相同！

**Working with the precision parameter**

But what about those precision parameters—what can we do with those? For mathy reasons, these are not measured on a logit scale. Instead, they’re logged values. We can invert them by exponentiating them with exp().

但那些精度参数呢？我们能用它们做什么？出于数学原因，这些都不是用logit量表来衡量的。相反，它们是记录值。我们可以通过用exp（）对它们进行幂运算来反转它们。

The phi intercept represents the precision of the distribution of the proportion of women MPs in countries without a quota, while the phi coefficient for quota represents the change in that precision when quota is TRUE. That means we have all the parameters to draw two different distributions of our outcome, split by whether countries have quotas. Let’s plot these two predicted distributions on top of the true underlying data and see how well they fit:

phi截距表示没有配额的国家中女议员比例分布的精度，而配额的phi系数表示配额为TRUE时精度的变化。这意味着我们有所有的参数来绘制我们结果的两种不同分布，根据国家是否有配额来划分。让我们在真实的基础数据之上绘制这两个预测分布，看看它们的拟合程度：

beta\_phi\_intercept <- model\_beta %>%

tidy() %>%

filter(component == "precision", term == "(Intercept)") %>%

pull(estimate)

beta\_phi\_quota <- model\_beta %>%

tidy() %>%

filter(component == "precision", term == "quotaTRUE") %>%

pull(estimate)

no\_quota\_title <- paste0("dprop(mean = plogis(", round(beta\_mu\_intercept, 2),

"), size = exp(", round(beta\_phi\_intercept, 2), "))")

quota\_title <- paste0("dprop(mean = plogis(", round(beta\_mu\_intercept, 2),

" + ", round(beta\_mu\_quota, 2),

"), size = exp(", round(beta\_phi\_intercept, 2),

" + ", round(beta\_phi\_quota, 2), "))")

ggplot(data = tibble(x = 0:1), aes(x = x)) +

geom\_density(data = vdem\_2015\_fake0,

aes(x = prop\_fem, fill = quota), alpha = 0.5, color = NA) +

stat\_function(fun = dprop, size = 1,

args = list(size = exp(beta\_phi\_intercept),

mean = plogis(beta\_mu\_intercept)),

aes(color = no\_quota\_title)) +

stat\_function(fun = dprop, size = 1,

args = list(size = exp(beta\_phi\_intercept + beta\_phi\_quota),

mean = plogis(beta\_mu\_intercept + beta\_mu\_quota)),

aes(color = quota\_title)) +

scale\_x\_continuous(labels = label\_percent()) +

scale\_fill\_viridis\_d(option = "plasma", end = 0.8, name = "Quota",

guide = guide\_legend(ncol = 1, order = 1)) +

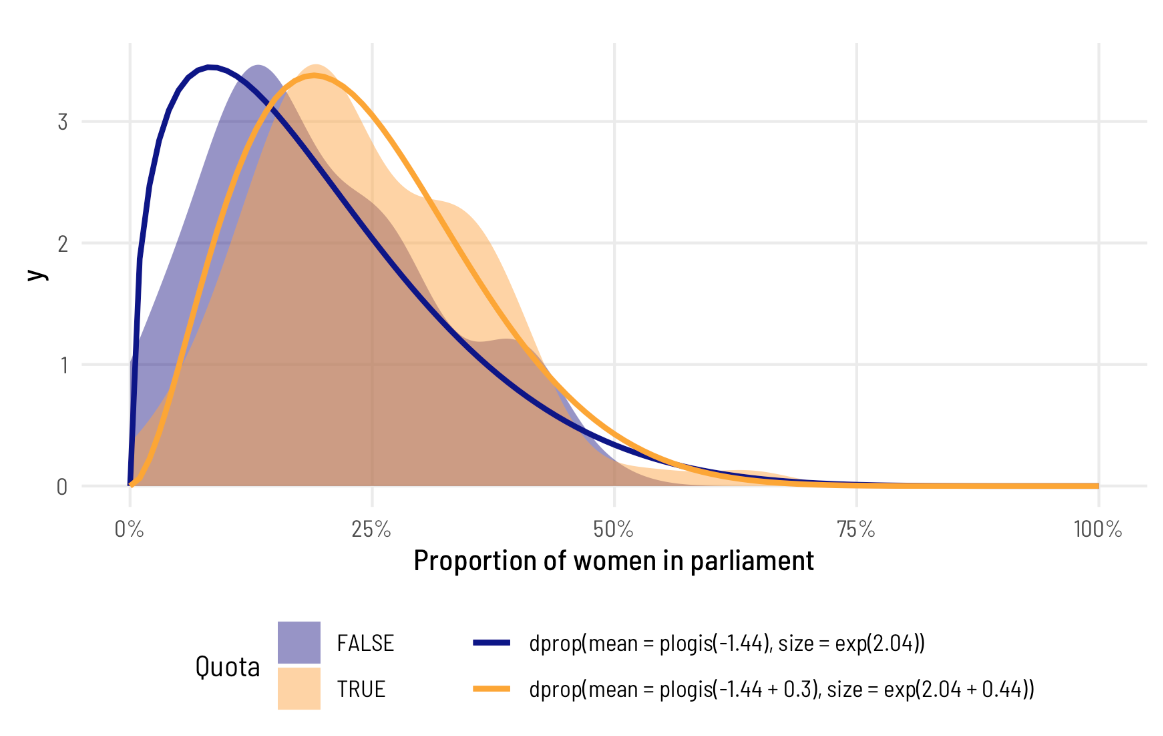
scale\_color\_viridis\_d(option = "plasma", end = 0.8, direction = -1, name = "",

guide = guide\_legend(reverse = TRUE, ncol = 1, order = 2)) +

labs(x = "Proportion of women in parliament") +

theme\_clean() +

theme(legend.position = "bottom")



Note how the whole distribution changes when there’s a quota. The mean doesn’t just shift rightward—the precision of the distribution changes too and the distribution where quota is TRUE has a whole new shape. The model actually captures it really well too! Look at how closely the predicted orange line aligns with the the underlying data. The blue predicted line is a little off, but that’s likely due to the 0s that we cheated with.

请注意，当存在配额时，整个分布是如何变化的。平均值不仅向右移动，分布的精度也发生了变化，配额为TRUE的分布有了一个全新的形状。这个模型实际上也很好地捕捉到了它！看看预测的橙色线和底层数据的对齐程度。蓝色预测线有点偏离，但这可能是由于我们欺骗了0。

You don’t have to model the Φ if you don’t want to. A model like this will work just fine too:

如果你不想的话，你不必为Φ建模。这样的模型也可以很好地工作：

model\_beta\_no\_phi <- betareg(prop\_fem ~ quota,

data = vdem\_2015\_fake0,

link = "logit")

tidy(model\_beta\_no\_phi)

## # A tibble: 3 × 6

## component term estimate std.error statistic p.value

## <chr> <chr> <dbl> <dbl> <dbl> <dbl>

## 1 mean (Intercept) -1.48 0.0785 -18.8 3.70e-79

## 2 mean quotaTRUE 0.370 0.111 3.33 8.76e- 4

## 3 precision (phi) 9.12 0.962 9.48 2.62e-21

In this case, you still get a precision component, but it’s universal across all the different coefficients—it doesn’t vary across quota (or any other variables, had we included any others in the model). Also, for whatever mathy reasons, when you don’t explicitly model the precision, the resulting coefficient in the table isn’t on the log scale—it’s a regular non-logged number, so there’s no need to exponentiate. We can plot the two distributions like before. The mean part is all the same! The only difference now is that phi is constant across the two groups

在这种情况下，你仍然会得到一个精度分量，但它在所有不同的系数中都是通用的——它不会因配额（或任何其他变量，如果我们在模型中包括任何其他变量的话）而变化。此外，无论出于何种数学原因，当你没有显式地对精度建模时，表中得到的系数不在对数范围内——它是一个规则的非对数，所以没有必要进行幂运算。我们可以像以前一样绘制这两种分布。卑鄙的部分都一样！现在唯一的区别是phi在两组中是常数

beta\_phi <- model\_beta\_no\_phi %>%

tidy() %>%

filter(component == "precision") %>%

pull(estimate)

no\_quota\_title <- paste0("dprop(mean = plogis(", round(beta\_mu\_intercept, 2),

"), size = ", round(beta\_phi, 2), ")")

quota\_title <- paste0("dprop(mean = plogis(", round(beta\_mu\_intercept, 2),

" + ", round(beta\_mu\_quota, 2),

"), size = ", round(beta\_phi, 2), ")")

ggplot(data = tibble(x = 0:1), aes(x = x)) +

geom\_density(data = vdem\_2015\_fake0,

aes(x = prop\_fem, fill = quota), alpha = 0.5, color = NA) +

stat\_function(fun = dprop, size = 1,

args = list(size = beta\_phi,

mean = plogis(beta\_mu\_intercept)),

aes(color = no\_quota\_title)) +

stat\_function(fun = dprop, size = 1,

args = list(size = beta\_phi,

mean = plogis(beta\_mu\_intercept + beta\_mu\_quota)),

aes(color = quota\_title)) +

scale\_x\_continuous(labels = label\_percent()) +

scale\_fill\_viridis\_d(option = "plasma", end = 0.8, name = "Quota",

guide = guide\_legend(ncol = 1, order = 1)) +

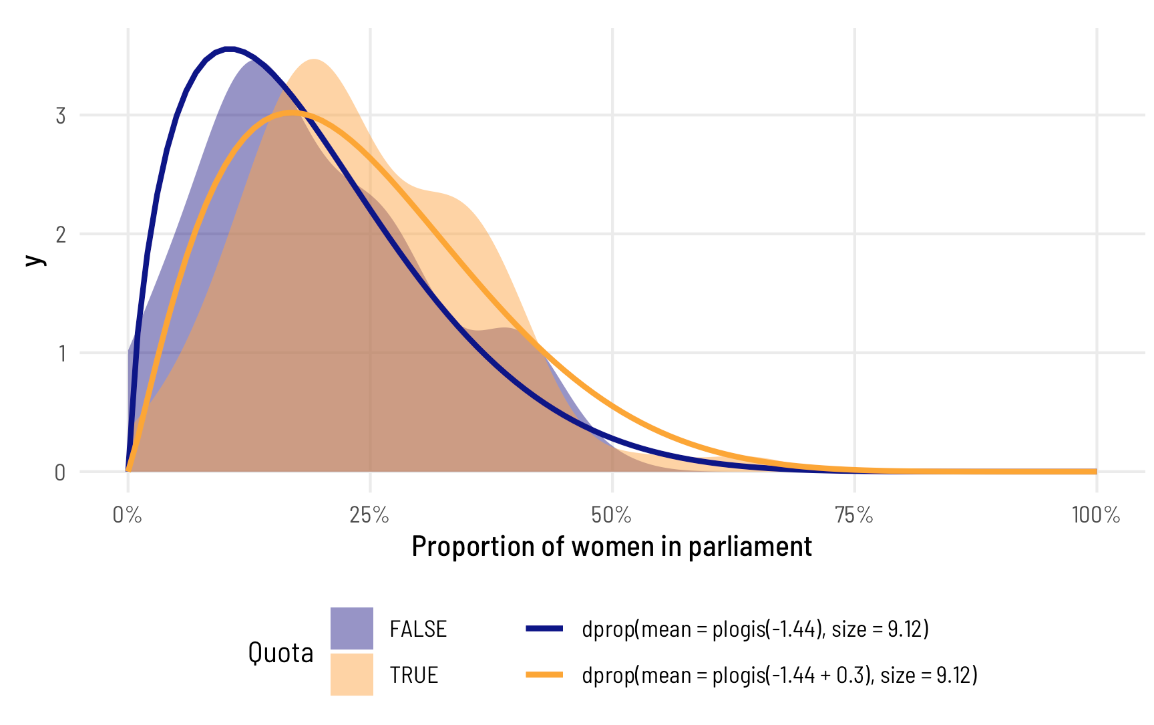
scale\_color\_viridis\_d(option = "plasma", end = 0.8, direction = -1, name = "",

guide = guide\_legend(reverse = TRUE, ncol = 1, order = 2)) +

labs(x = "Proportion of women in parliament") +

theme\_clean() +

theme(legend.position = "bottom")



Again, the means are the same, so the average difference between having a quota and not having a quota is the same, but because the precision is constant, the distributions don’t fit the data as well.

同样，平均值是相同的，因此有配额和没有配额之间的平均差异是相同的。但由于精度是恒定的，因此分布也不符合数据。

**Average marginal effects**

Because beta regression coefficients are all in logit units (with the exception of the Φ part), we can interpret their marginal effects just like we did with logistic regression. It’s tricky to piece together all the different parts and feed them through plogis(), so it’s best to calculate the slope with hypothetical data as before.

因为贝塔回归系数都以logit为单位（Φ部分除外），我们可以像逻辑回归一样解释它们的边际效应。将所有不同的部分拼凑在一起并通过plogis（）进行输入是很棘手的，所以最好像以前一样使用假设数据来计算斜率。

avg\_slopes(model\_beta,

newdata = datagrid(quota = c(FALSE, TRUE)))

##

## Term Contrast Estimate Std. Error z Pr(>|z|) 2.5 % 97.5 %

## quota TRUE - FALSE 0.0501 0.0192 2.6 0.00926 0.0124 0.0878

##

## Columns: term, contrast, estimate, std.error, statistic, p.value, conf.low, conf.high

After incorporating all the moving parts of the model, we find that having a quota is associated with an increase of 5.006 percentage points of the proportion of women MPs, on average, which is what we found earlier by piecing together the intercept and coefficient. If we had more covariates, this avg\_slopes() approach would be necessary, since it’s tricky to incorporate all the different model pieces by hand.

在纳入该模型的所有移动部分后，我们发现，拥有配额与女性议员比例平均增加5.006个百分点有关，这是我们早些时候通过拼凑截距和系数发现的。如果我们有更多的协变量，这种avg\_slopes（）方法将是必要的，因为手工合并所有不同的模型片段是很棘手的。

**3b: Beta regression, Bayesian style**

Uncertainty is wonderful. We’re working with distributions already with distributional regression like beta regression—it would be great if we could quantify and model our uncertainty directly and treat all the parameters as distributions.

Fortunately Bayesian regression lets us do just that. We can generate thousands of plausible estimates based on the combination of our prior beliefs and the likelihood of the data, and then we can explore the uncertainty and the distributions of all those estimates.

不确定性是美妙的。我们已经在使用分布回归（如贝塔回归）来处理分布——如果我们能够直接量化和建模我们的不确定性，并将所有参数视为分布，那就太好了。幸运的是，贝叶斯回归让我们做到了这一点。根据我们之前的信念和数据的可能性，我们可以生成数千个合理的估计，然后我们可以探索所有这些估计的不确定性和分布。

We’ll also get to interpret the results Bayesianly. [Goodbye confidence intervals; hello credible intervals.](https://evalf21.classes.andrewheiss.com/resource/bayes/#confidence-intervals-vs-credible-intervals)

The [incredible **brms** package](https://paul-buerkner.github.io/brms/) has built-in support for beta regression. I won’t go into details here about how to use **brms**—there are all sorts of tutorials and examples online for that ([here’s a quick basic one](https://evalf21.classes.andrewheiss.com/resource/bayes/#super-short-example)).

我们还将对结果进行贝叶斯解释。再见信心区间；你好，可信的间隔。

令人难以置信的brms包内置了对beta回归的支持。我不会在这里详细介绍如何使用brms——网上有各种各样的教程和例子（这里有一个快速的基本教程）。

Let’s recreate the model we made with [betareg()](https://rdrr.io/pkg/betareg/man/betareg.html) earlier, but now with a Bayesian flavor. The formula syntax is a little different with **brms**. Instead of using | to divide the μ part from the Φ part, we specify two separate formulas: one for prop\_fem for the μ part, and one for phi for the Φ  part. We’ll use the default priors here (but in real life, you’d want to set them to something sensible), and we’ll generate 2,000 Monte Carlo Markov Chain (MCMC) samples across 4 chains.

让我们重新创建我们之前使用betareg（）创建的模型，但现在使用贝叶斯风格。brms的公式语法有点不同。我们没有使用|将μ部分与Φ部分分开，而是指定了两个单独的公式：一个用于μ部分的prop\_fem，另一个用于Φ部分的phi。我们将在这里使用默认先验（但在现实生活中，您可能希望将它们设置为合理的值），我们将在4个链上生成2000个蒙特卡罗马尔可夫链（MCMC）样本。

model\_beta\_bayes <- brm(

bf(prop\_fem ~ quota,

phi ~ quota),

data = vdem\_2015\_fake0,

family = Beta(),

chains = 4, iter = 2000, warmup = 1000,

cores = 4, seed = 1234,

# Use the cmdstanr backend for Stan because it's faster and more modern than

# the default rstan You need to install the cmdstanr package first

# (https://mc-stan.org/cmdstanr/) and then run cmdstanr::install\_cmdstan() to

# install cmdstan on your computer.

backend = "cmdstanr",

file = "model\_beta\_bayes" # Save this so it doesn't have to always rerun

)

Check out the results—they’re basically the same that we found with [betareg()](https://rdrr.io/pkg/betareg/man/betareg.html) in model\_beta.

看看结果——它们与我们在model\_beta中使用betareg（）发现的结果基本相同。

tidy(model\_beta\_bayes, effects = "fixed")

## # A tibble: 4 × 7

## effect component term estimate std.error conf.low conf.high

## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>

## 1 fixed cond (Intercept) -1.44 0.0864 -1.61 -1.27

## 2 fixed cond phi\_(Intercept) 2.02 0.140 1.74 2.29

## 3 fixed cond quotaTRUE 0.296 0.114 0.0713 0.518

## 4 fixed cond phi\_quotaTRUE 0.436 0.211 0.0159 0.837

**Working with the posterior**

The estimates are the same, but richer—we have a complete posterior distribution for each coefficient, which we can visualize in neat ways:

估计是相同的，但更丰富——我们对每个系数都有一个完整的后验分布，我们可以用简洁的方式将其可视化：

posterior\_beta <- model\_beta\_bayes %>%

gather\_draws(`b\_.\*`, regex = TRUE) %>%

mutate(component = ifelse(str\_detect(.variable, "phi\_"), "Precision", "Mean"),

intercept = str\_detect(.variable, "Intercept"))

ggplot(posterior\_beta, aes(x = .value, y = fct\_rev(.variable), fill = component)) +

geom\_vline(xintercept = 0) +

stat\_halfeye(aes(slab\_alpha = intercept),

.width = c(0.8, 0.95), point\_interval = "median\_hdi") +

scale\_fill\_viridis\_d(option = "viridis", end = 0.6) +

scale\_slab\_alpha\_discrete(range = c(1, 0.4)) +

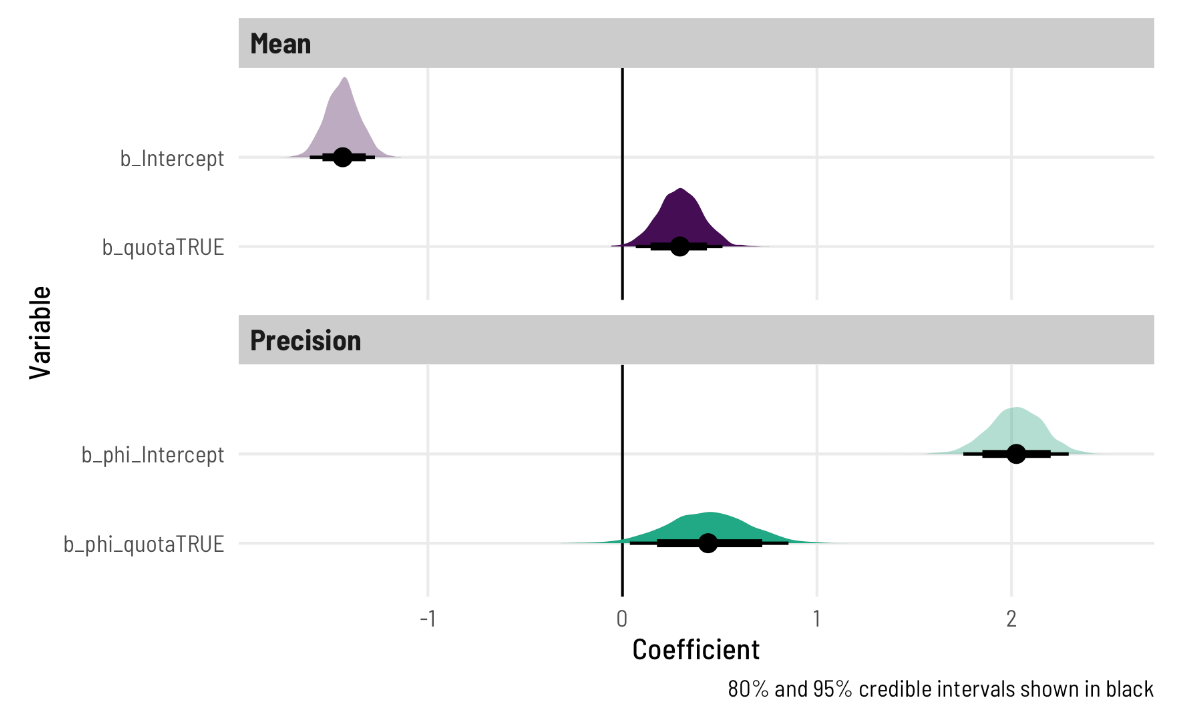
guides(fill = "none", slab\_alpha = "none") +

labs(x = "Coefficient", y = "Variable",

caption = "80% and 95% credible intervals shown in black") +

facet\_wrap(vars(component), ncol = 1, scales = "free\_y") +

theme\_clean()



That shows the coefficients in their transformed scale (logits for the mean, logs for the precision), but we can transform them to the response scale too by feeding the Φ parts to [exp()](https://rdrr.io/r/base/Log.html) and the μ parts to [plogis()](https://rdrr.io/r/stats/Logistic.html):

这显示了它们变换后的尺度中的系数（对数表示平均值，对数表示精度），但我们也可以通过将Φ部分输入exp（）和μ部分输入plogis（）将它们变换为响应尺度：

model\_beta\_bayes %>%

spread\_draws(`b\_.\*`, regex = TRUE) %>%

mutate(across(starts\_with("b\_phi"), ~exp(.))) %>%

mutate(across((!starts\_with(".") & !starts\_with("b\_phi")), ~plogis(.))) %>%

gather\_variables() %>%

median\_hdi()

## # A tibble: 4 × 7

## .variable .value .lower .upper .width .point .interval

## <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>

## 1 b\_Intercept 0.192 0.167 0.219 0.95 median hdi

## 2 b\_phi\_Intercept 7.57 5.66 9.80 0.95 median hdi

## 3 b\_phi\_quotaTRUE 1.55 0.997 2.27 0.95 median hdi

## 4 b\_quotaTRUE 0.573 0.522 0.630 0.95 median hdi

But beyond the intercepts here, this isn’t actually that helpful. The median back-transformed intercept for the μ part of the regression is 0.192, which means countries without a quota have an average proportion of women MPs of 19.2%. That’s all fine. The quota effect here, though, is 0.573. That does *not* mean that having a quota increases the proportion of women by 57 percentage points! In order to find the marginal effect of having a quota, we need to also incorporate the intercept. Remember when we ran plogis(intercept + coefficient) - plogis(intercept) earlier? We have to do that again.

但除了这里的拦截之外，这实际上并没有那么大帮助。回归μ部分的中位反变换截距为0.192，这意味着没有配额的国家女性议员的平均比例为19.2%。这一切都很好。不过，这里的配额效应是0.573。这并不意味着拥有配额会使女性比例增加57个百分点！为了找出配额的边际效应，我们还需要纳入截距。还记得我们早些时候运行plogis（截距+系数）-plogis（截距）吗？我们必须再次这样做。

**Posterior average marginal effects**

**Binary predictor**

Once again, the math for combining these coefficients can get hairy, especially when we’re working with more than one explanatory variable, so instead of figuring out that math, we’ll calculate the marginal effects across different values of quota.

同样，组合这些系数的数学运算可能会变得棘手，尤其是当我们使用多个解释变量时，所以我们将计算不同配额值的边际效应，而不是计算数学运算。

**2023-08-18 edit**

The **marginaleffects** package *does* work with **brms** models now and I like its interface and its approach to estimating things a lot better than **emmeans**, so I’ve changed all the code here to use it instead. If you want to see the **emmeans** version of everything, [look at this past version at GitHub](https://github.com/andrewheiss/ath-quarto/blob/d8628ba05ee4e4dbec3c571e0656461e2f737594/blog/2021/11/08/beta-regression-guide/index.qmd#L834).

marginalcoffects包现在确实适用于brms模型，我喜欢它的界面和它的估计方法，比emmeans好得多，所以我更改了这里的所有代码，改为使用它。如果你想看看所有东西的emmeans版本，可以在GitHub上看看这个过去的版本。

Also, I’m pretty loosey-goosey about the term “marginal effect” throughout this post because this was early in my journey to actually understanding what all these different estimands are. See [my “Marginalia” post](https://www.andrewheiss.com/blog/2022/05/20/marginalia/), published a few months after this one, for a ton more about what “marginal effects” actually mean.

此外，在这篇文章中，我对“边际效应”这个词感到非常不安，因为这是我真正理解这些不同估计的早期阶段。请参阅我在这篇文章几个月后发表的“边际效应”帖子，了解更多关于“边际影响”实际含义的信息。

We can calculate the pairwise marginal effects across different values of quota, which conveniently shows us the difference, or the marginal effect of having a quota:

我们可以计算不同配额值之间的成对边际效应，这很方便地向我们展示了配额的差异或边际效应：

model\_beta\_bayes %>%

avg\_comparisons(variables = "quota")

##

## Term Contrast Estimate 2.5 % 97.5 %

## quota TRUE - FALSE 0.05 0.0121 0.0875

##

## Columns: term, contrast, estimate, conf.low, conf.high

This summary information is helpful—we have our difference in predicted outcomes of 0.05, which is roughly what we found in earlier models. We also have a 95% highest posterior density interval for the difference.

这些汇总信息很有帮助——我们的预测结果相差0.05，这大致是我们在早期模型中发现的。我们也有95%的最高后验密度区间的差异。

But it would be great if we could visualize these predictions! What do the posteriors for quota and non-quota countries look like, and what does the distribution of differences look like? We can work with the original model and MCMC chains to visualize this, letting **brms** take care of the hard work of predicting the coefficients we need and back-transforming them into probabilities.

但如果我们能将这些预测可视化，那就太好了！配额国家和非配额国家的后验结果是什么样子的，差异的分布是什么样子？我们可以使用原始模型和MCMC链来可视化这一点，让brms承担预测我们需要的系数并将其反向转换为概率的艰巨工作。

**marginaleffects**

**marginaleffects** can also handle all this with its [comparisons()](https://marginaleffects.com/articles/comparisons.html) and [hypotheses()](https://marginaleffects.com/articles/hypothesis.html) functions—[even with Bayesian models](https://marginaleffects.com/articles/brms.html)!

边际效应也可以通过它的comparisons（）和假设（）函数来处理所有这些——即使是贝叶斯模型！

We’ll use [epred\_draws()](http://mjskay.github.io/tidybayes/reference/add_predicted_draws.html) from **tidybayes** to plug in a hypothetical dataset and generate predictions, as we did with previous models. Only this time, instead of getting a single predicted value for each level of quota, we’ll get 4,000. The [epred\_draws()](http://mjskay.github.io/tidybayes/reference/add_predicted_draws.html) (like all **tidybayes** functions) returns a long tidy data frame, so it’s really easy to plot with ggplot:

我们将使用tidybayes中的epred\_draws（）来插入一个假设的数据集并生成预测，就像我们对以前的模型所做的那样。只是这一次，我们将获得4000，而不是每个级别配额的单个预测值。epred\_draws（）（像所有tidybayes函数一样）返回一个长而整洁的数据帧，因此使用ggplot绘制非常容易：

# Plug a dataset where quota is FALSE and TRUE into the model

beta\_bayes\_pred <- model\_beta\_bayes %>%

epred\_draws(newdata = tibble(quota = c(FALSE, TRUE)))

ggplot(beta\_bayes\_pred, aes(x = .epred, y = quota, fill = quota)) +

stat\_halfeye(.width = c(0.8, 0.95), point\_interval = "median\_hdi") +

scale\_x\_continuous(labels = label\_percent()) +

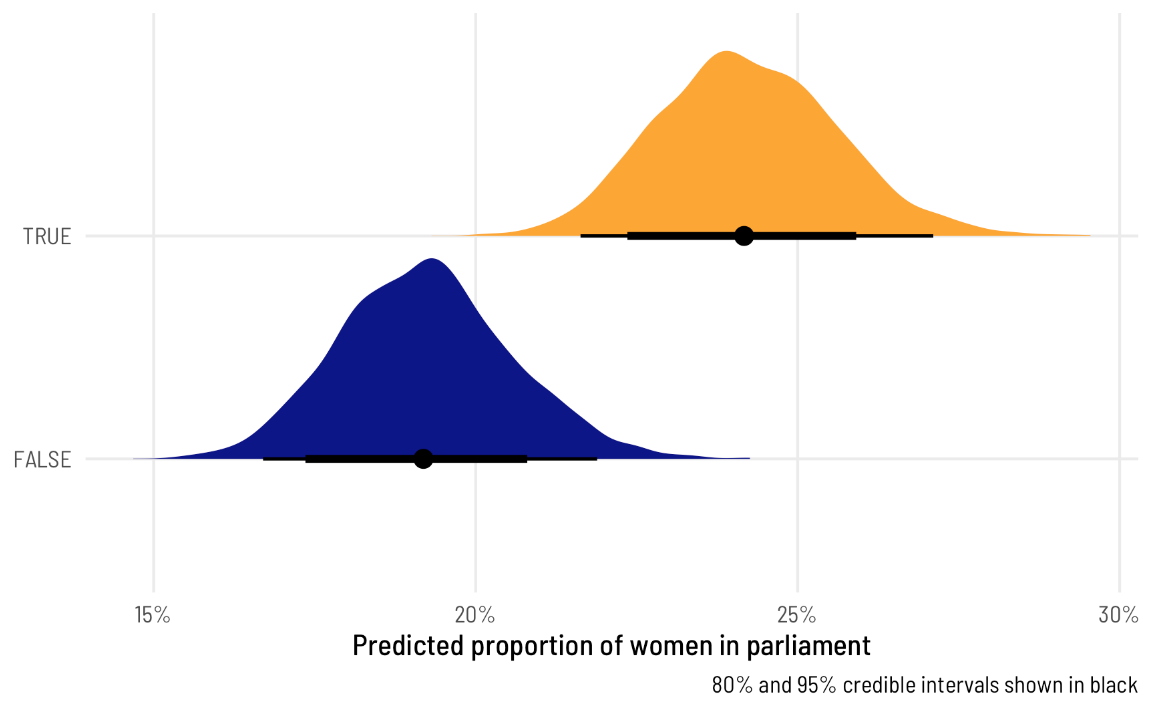
scale\_fill\_viridis\_d(option = "plasma", end = 0.8) +

guides(fill = "none") +

labs(x = "Predicted proportion of women in parliament", y = NULL,

caption = "80% and 95% credible intervals shown in black") +

theme\_clean()



That’s super neat! We can easily see that countries with quotas have a higher predicted proportion of women in parliament, and we can see the uncertainty in those estimates. We can also work with these posterior predictions to calculate the difference in proportions, or the marginal effect of having a quota, which is the main thing we’re interested in.

真是太整洁了！我们可以很容易地看到，有配额的国家议会中女性的预测比例更高，我们可以看到这些估计中的不确定性。我们也可以使用这些后验预测来计算比例的差异，或者配额的边际效应，这是我们感兴趣的主要事情。

We can use compare\_levels() from tidybayes to calculate the difference between these two posteriors:

我们可以使用tidybayes中的compare\_levels（）来计算这两个后验之间的差异：

beta\_bayes\_pred\_diff <- beta\_bayes\_pred %>%

compare\_levels(variable = .epred, by = quota)

beta\_bayes\_pred\_diff %>% median\_hdi()

## # A tibble: 1 × 7

## quota .epred .lower .upper .width .point .interval

## <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>

## 1 TRUE - FALSE 0.0500 0.0136 0.0889 0.95 median hdi

ggplot(beta\_bayes\_pred\_diff, aes(x = .epred)) +

stat\_halfeye(.width = c(0.8, 0.95), point\_interval = "median\_hdi",

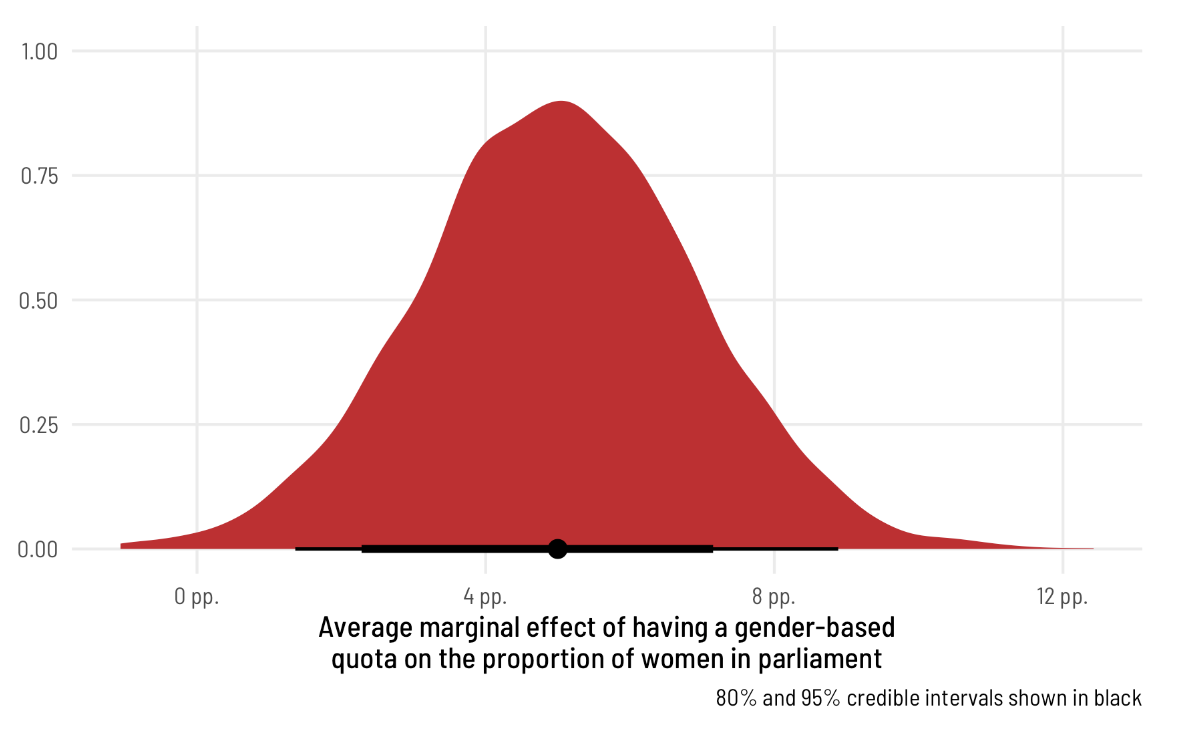
fill = "#bc3032") +

scale\_x\_continuous(labels = label\_pp) +

labs(x = "Average marginal effect of having a gender-based\nquota on the proportion of women in parliament",

y = NULL, caption = "80% and 95% credible intervals shown in black") +

theme\_clean()



Perfect! The median marginal effect of having a quota is 6 percentage points, but it can range from 2 to 10.

完美的配额的边际效应中位数为6个百分点，但可能在2到10之间。

**Continuous predictor**

The same approach works for continuous predictors too, just like with previous methods. Let’s build a model that explains both the mean and the precision with quota and polyarchy. We don’t necessarily have to model the precision with the same variables that we use for the mean—that can be a completely separate process—but we’ll do it just for fun.

同样的方法也适用于连续预测，就像以前的方法一样。让我们建立一个模型来解释配额和多元民主的平均值和精度。我们不一定要用与平均值相同的变量来建模精度——这可能是一个完全独立的过程——但我们这样做只是为了好玩。

# We need to specify init here because the beta precision parameter can never

# be negative, but Stan will generate random initial values from -2 to 2, even

# for beta's precision, which leads to rejected chains and slower performance.

# For even fancier init handling, see Solomon Kurz's post here:

# https://solomonkurz.netlify.app/post/2021-06-05-don-t-forget-your-inits/

model\_beta\_bayes\_1 <- brm(

bf(prop\_fem ~ quota + polyarchy,

phi ~ quota + polyarchy),

data = vdem\_2015\_fake0,

family = Beta(),

init = 0,

chains = 4, iter = 2000, warmup = 1000,

cores = 4, seed = 1234,

backend = "cmdstanr",

file = "model\_beta\_bayes\_1"

)

We’ll forgo interpreting all these different coefficients, since we’d need to piece together the different parts and back-transform them with either [plogis()](https://rdrr.io/r/stats/Logistic.html) (for the μ parts) or [exp()](https://rdrr.io/r/base/Log.html) (for the Φ parts). Instead, we’ll skip to the marginal effect of polyarchy on the proportion of women MPs. First we can look at the posterior prediction of the outcome across the whole range of democracy scores:

我们将放弃解释所有这些不同的系数，因为我们需要将不同的部分拼凑在一起，并使用plogis（）（对于μ部分）或exp（））（对于Φ部分）对它们进行反变换。相反，我们将跳过多元民主对女议员比例的边际影响。首先，我们可以看看整个民主评分范围内对结果的后验预测：

# Use a dataset where quota is FALSE and polyarchy is a range

beta\_bayes\_pred\_1 <- model\_beta\_bayes\_1 %>%

epred\_draws(newdata = expand\_grid(quota = FALSE,

polyarchy = seq(0, 100, by = 1)))

ggplot(beta\_bayes\_pred\_1, aes(x = polyarchy, y = .epred)) +

stat\_lineribbon() +

scale\_y\_continuous(labels = label\_percent()) +

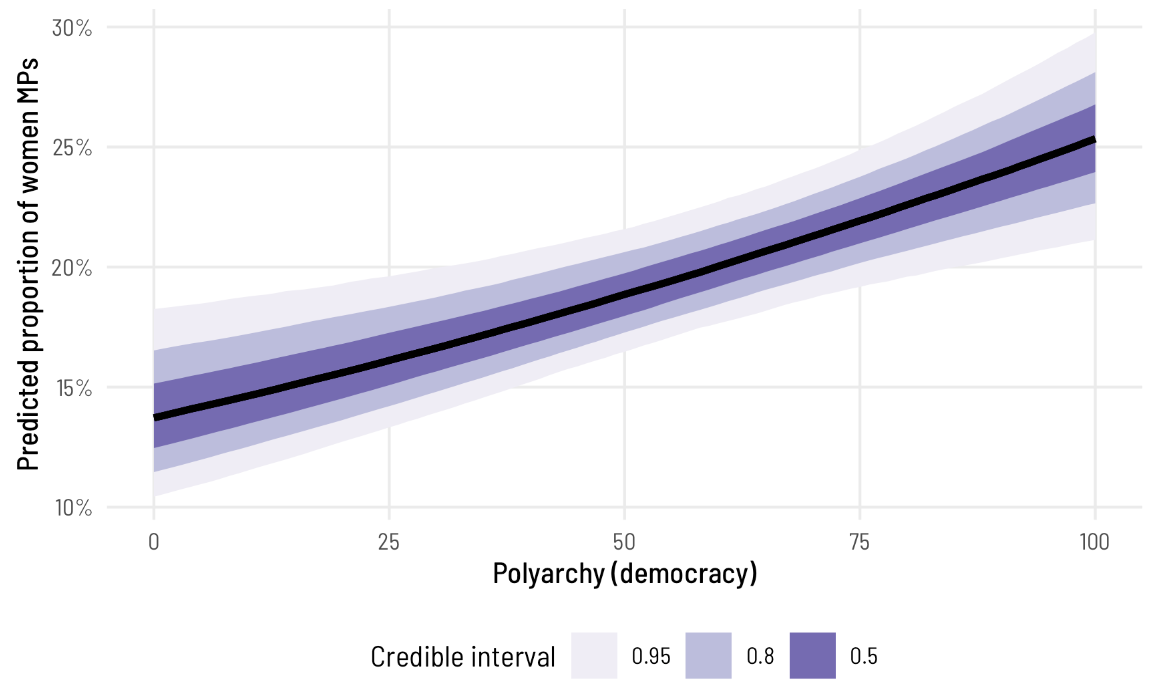
scale\_fill\_brewer(palette = "Purples") +

labs(x = "Polyarchy (democracy)", y = "Predicted proportion of women MPs",

fill = "Credible interval") +

theme\_clean() +

theme(legend.position = "bottom")



The predicted proportion of women MPs increases as democracy increases, as we’ve seen before. Note how the slope here isn’t constant, though—the line is slightly curved. That’s because this is a nonlinear regression model, making it so the marginal effect of democracy is different depending on its level. We can see this if we look at the instantaneous slope (or first derivative) at different values of polyarchy. **marginaleffects** makes this really easy with the aavg\_slopes() function. If we ask it for the slope of the line, it will provide it at the average value of polyarchy, or 53.5:

正如我们以前看到的那样，随着民主的加强，预计女议员的比例也会增加。请注意，这里的斜率不是恒定的——这条线有点弯曲。这是因为这是一个非线性回归模型，因此民主的边际效应因其水平而异。如果我们观察不同值的多元民主下的瞬时斜率（或一阶导数），我们可以看到这一点。marginalcoffects通过使用aavg\_slopes（）函数可以非常容易地实现这一点。如果我们问它这条线的斜率，它会以多元民主的平均值提供，即53.5：

avg\_slopes(model\_beta\_bayes\_1, variables = "polyarchy")

##

## Term Estimate 2.5 % 97.5 %

## polyarchy 0.00126 0.000502 0.00196

##

## Columns: term, estimate, conf.low, conf.high

Our effect is thus 0.001, or 0.126 percentage points, but only for countries in the middle range of polyarchy. The slope is shallower and steeper depending on the level of democracy. We can check the slope (or effect) at different hypothetical levels

因此，我们的影响是0.001，即0.126个百分点，但仅适用于处于多配偶制中间范围的国家。根据民主程度的不同，坡度会越来越浅、越来越陡。我们可以在不同的假设水平上检查斜率（或效果）

slopes(model\_beta\_bayes\_1,

variables = "polyarchy",

newdata = datagrid(polyarchy = c(20, 50, 80)))

##

## Term Estimate 2.5 % 97.5 % quota polyarchy

## polyarchy 0.000991 0.000438 0.00140 FALSE 20

## polyarchy 0.001153 0.000464 0.00178 FALSE 50

## polyarchy 0.001314 0.000489 0.00219 FALSE 80

##

## Columns: rowid, term, estimate, conf.low, conf.high, predicted, predicted\_hi, predicted\_lo, tmp\_idx, prop\_fem, quota, polyarchy

Neat! For observations with low democracy scores like 20, a one-unit increase in polyarchy is associated with a 0.099 percentage point increase in the proportion of women MPs, while countries with high scores like 80 see a 0.131 percentage point increase.

整洁的对于民主得分较低（如20分）的观察结果，多元民主增加一个单位与女议员比例增加0.099个百分点有关，而得分较高（如80分）的国家则增加0.131个百分点。

We can visualize the distribution of these marginal effects too by using [slopes()](https://rdrr.io/pkg/marginaleffects/man/slopes.html) to calculate the instantaneous slopes across all the MCMC chains, and then using [posteriordraws()](https://rdrr.io/pkg/marginaleffects/man/posteriordraws.html) to extract the draws as tidy, plottable data.

我们也可以通过使用slopes（）来计算所有MCMC链上的瞬时斜率，然后使用posteriordraws（）来提取作为整洁、可绘制数据的绘图，来可视化这些边际效应的分布。

(Side note: You could technically do this without **marginaleffects**. The way **marginaleffects** calculates the instantaneous slope is by finding the predicted average for some value, finding the predicted average for that same value plus a *tiny* amount extra, subtracting the two predictions, and dividing by that tiny extra amount. If you really didn’t want to use **marginaleffects**, you could generate predictions for polyarchy = 20 and polyarchy = 20.001, subtract them by hand, and then divide by 0.001. That’s miserable though—don’t do it. Use **marginaleffects** instead.)

（旁注：从技术上讲，你可以在没有边际效应的情况下做到这一点。边际效应计算瞬时斜率的方法是找到某个值的预测平均值，找到同一值的预测均值加上一个微小的额外量，减去两个预测值，然后除以这个微小的额外值。如果你真的不想使用边际效应，你可以生成p多配偶=20和多配偶=20.001的预测，用手减去它们，然后除以0.001。不过，这很痛苦——不要这样做。改为使用边际效应。）

ame\_beta\_bayes\_1 <- model\_beta\_bayes\_1 %>%

slopes(variables = "polyarchy",

newdata = datagrid(polyarchy = c(20, 50, 80))) %>%

posterior\_draws()

ggplot(ame\_beta\_bayes\_1, aes(x = draw, fill = factor(polyarchy))) +

geom\_vline(xintercept = 0) +

stat\_halfeye(.width = c(0.8, 0.95), point\_interval = "median\_hdi",

slab\_alpha = 0.75) +

scale\_x\_continuous(labels = label\_pp\_tiny) +

scale\_fill\_viridis\_d(option = "viridis", end = 0.6) +

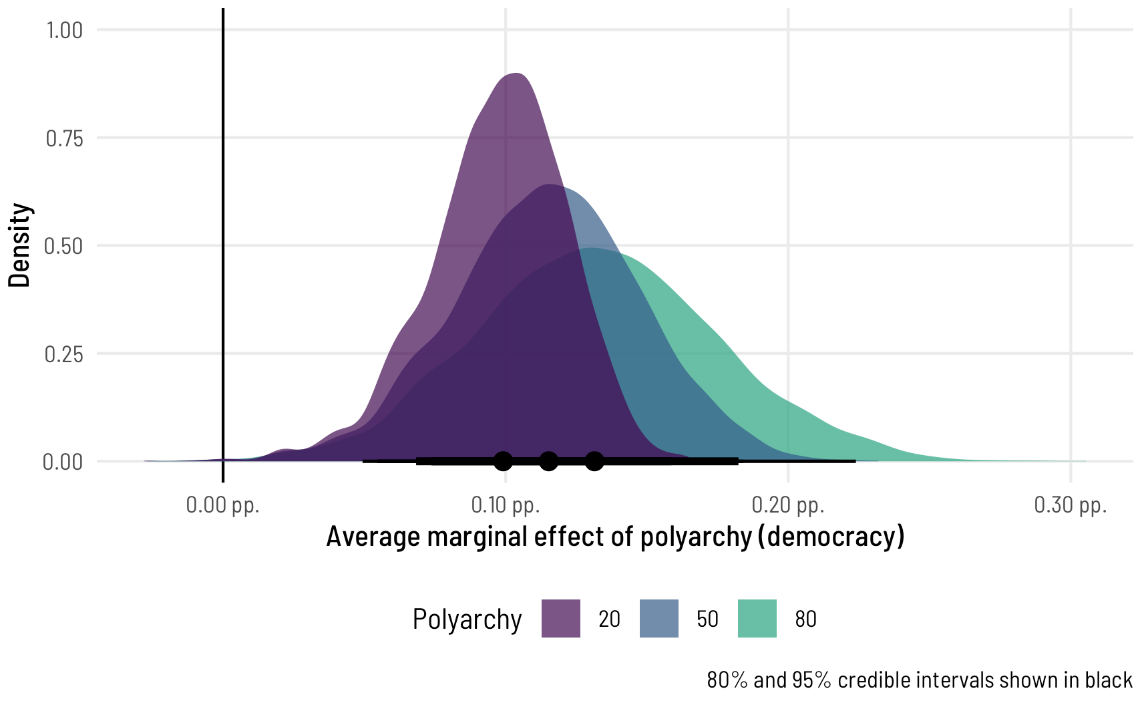
labs(x = "Average marginal effect of polyarchy (democracy)",

y = "Density", fill = "Polyarchy",

caption = "80% and 95% credible intervals shown in black") +

theme\_clean() +

theme(legend.position = "bottom")



To me this is absolutely incredible! Check out how the shapes of these distributions change as polyarchy increases! That’s because we modeled both the μ and the Φ parameters of the beta distribution, so we get to see changes in both the mean and precision of the outcome. Had we left the phi part alone, doing something like phi ~ 1, the marginal effect would look the same and would just be shifted to the right.

对我来说，这简直不可思议！看看这些分布的形状是如何随着多元民主的增加而变化的！这是因为我们对β分布的μ和Φ参数进行了建模，因此我们可以看到结果的平均值和精度的变化。如果我们把phi部分放在一边，做一些类似phi~1的事情，边际效应看起来是一样的，只是向右移动。

**4: Zero-inflated beta regression, Bayesian style**

Beta regression is really neat and modeling different components of the beta regression is a fascinating and rich way of thinking about the outcome variable. Plus the beta distribution naturally limits the outcome to the 0–1 range, so you don’t have to shoehorn the data into an OLS-based LPM or fractional logistic model.

贝塔回归真的很巧妙，对贝塔回归的不同组成部分建模是一种有趣而丰富的思考结果变量的方式。此外，贝塔分布自然将结果限制在0–1范围内，因此您不必将数据硬塞进基于OLS的LPM或分数逻辑模型中。

One problem with beta regression, though, is that the beta distribution does not include 0 and 1. As we saw earlier, if your data has has 0s or 1s, the model breaks.

但是，贝塔回归的一个问题是贝塔分布不包括0和1。正如我们前面看到的，如果您的数据有0或1，那么模型就会崩溃。

**A new parameter for modeling the zero process**

To get around this, we can use a special zero-inflated beta regression. We’ll still model the μ and Φ (or mean and precision) of the beta distribution, but we’ll also model one new special parameter α. With zero-inflated regression, we’re actually modelling a mixture of data-generating processes:

为了解决这个问题，我们可以使用一个特殊的零膨胀贝塔回归。我们仍然将对β分布的μ和Φ（或平均值和精度）进行建模，但我们也将对一个新的特殊参数α进行建模。使用零膨胀回归，我们实际上是在对数据生成过程的混合进行建模：

1. A logistic regression model that predicts if an outcome is 0 or not, defined by α
2. A beta regression model that predicts if an outcome is between 0 and 1 if it’s not zero, defined by μ and Φ

1.一个逻辑回归模型，用于预测结果是否为0，由α定义

2.一个β回归模型，如果结果不是零，则预测结果是否在0和1之间，由μ和Φ定义

Let’s see how many zeros we have in the data:

vdem\_2015 %>%

count(prop\_fem == 0) %>%

mutate(prop = n / sum(n))

## prop\_fem == 0 n prop

## 1 FALSE 169 0.9826

## 2 TRUE 3 0.0174

Hahahaha only 3, or 1.7% of the data. That’s barely anything. Perhaps our approach of just adding a tiny amount to those three 0 observations is fine. For the sake of illustrating this approach, though, we’ll still model the zero-creating process.

The only difference between regular beta regression and zero-inflated regression is that we have to specify one more parameter: zi. This corresponds to the α parameter and determines the zero/not-zero process.

To help with the intuition, we’ll first run a model where we don’t actually define a model for zi—it’ll just return the intercept for the α parameter.

哈哈哈哈只有3个，或者1.7%的数据。这算不了什么。也许我们只在这三个0的观测值上加一点就可以了。不过，为了说明这种方法，我们仍将对零创建过程进行建模。

正则贝塔回归和零膨胀回归之间的唯一区别是，我们必须再指定一个参数：zi。这与α参数相对应，并确定零/非零过程。

为了帮助直觉，我们将首先运行一个模型，在这个模型中，我们实际上并没有为zi定义一个模型——它只返回α参数的截距。

model\_beta\_zi\_int\_only <- brm(

bf(prop\_fem ~ quota,

phi ~ quota,

zi ~ 1),

data = vdem\_clean,

family = zero\_inflated\_beta(),

chains = 4, iter = 2000, warmup = 1000,

cores = 4, seed = 1234,

backend = "cmdstanr",

file = "model\_beta\_zi\_int\_only"

)

tidy(model\_beta\_zi\_int\_only, effects = "fixed")

## # A tibble: 5 × 7

## effect component term estimate std.error conf.low conf.high

## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>

## 1 fixed cond (Intercept) -1.42 0.0229 -1.47 -1.38

## 2 fixed cond phi\_(Intercept) 2.37 0.0437 2.29 2.46

## 3 fixed zi (Intercept) -4.04 0.186 -4.43 -3.70

## 4 fixed cond quotaTRUE 0.303 0.0331 0.238 0.369

## 5 fixed cond phi\_quotaTRUE 0.237 0.0677 0.105 0.368

As before, the parameters for μ ((Intercept) and quotaTRUE) are on the logit scale, while the parameters for Φ (phi\_(Intercept) and phi\_quotaTRUE) are on the log scale. The zero-inflated parameter (or α) here ((Intercept) for the zi component) is on the logit scale like the regular model coefficients. That means we can back-transform it with [plogis()](https://rdrr.io/r/stats/Logistic.html):

和以前一样，μ（（Intercept）和quotaTRUE）的参数在logit标度上，而Φ（phi\_（Intercepte）和phi\_quotaTRUE）的参数则在对数标度上。这里的零膨胀参数（或α）（zi分量的（Intercept））与正则模型系数一样在logit尺度上。这意味着我们可以使用plogis（）对其进行反向转换：

zi\_intercept <- tidy(model\_beta\_zi\_int\_only, effects = "fixed") %>%

filter(component == "zi", term == "(Intercept)") %>%

pull(estimate)

# Logit scale intercept

zi\_intercept

## b\_zi\_Intercept

## -4.04

# Transformed to a probability/proportion

plogis(zi\_intercept)

## b\_zi\_Intercept

## 0.0172

After transforming the intercept to a probability/proportion, we can see that it’s 1.72%—basically the same as the proportion of zeros in the data!

将截距转换为概率/比例后，我们可以看到它是1.72%——与数据中零的比例基本相同！

Right now, all we’ve modeled is the overall proportion of 0s. We haven’t modeled what determines those zeros. We can see that if we plot the posterior predictions for prop\_fem across quota, highlighting predicted values that are 0. In general, ≈2% of the predicted outcomes should be zero, and since we didn’t really model zi, the 0s should be equally spread across the different levels of quota. To visualize this we’ll use a histogram instead of a density plot so that we can better see the count of 0s. We’ll also cheat a little and make the 0s a negative number so that the histogram bin for the 0s appears outside of the main 0–1 range.

现在，我们所建模的只是0的总体比例。我们还没有对决定这些零的因素进行建模。我们可以看到，如果我们绘制跨配额的prop\_fem的后验预测，突出显示0的预测值。一般来说，≈2%的预测结果应该为零，由于我们没有真正对zi进行建模，0应该平均分布在不同级别的配额中。为了将其可视化，我们将使用直方图而不是密度图，这样我们可以更好地看到0的计数。我们还将进行一些欺骗，将0设为负数，以便0的直方图仓出现在主0–1范围之外。

beta\_zi\_pred\_int <- model\_beta\_zi\_int\_only %>%

predicted\_draws(newdata = tibble(quota = c(FALSE, TRUE))) %>%

mutate(is\_zero = .prediction == 0,

.prediction = ifelse(is\_zero, .prediction - 0.01, .prediction))

ggplot(beta\_zi\_pred\_int, aes(x = .prediction)) +

geom\_histogram(aes(fill = is\_zero), binwidth = 0.025,

boundary = 0, color = "white") +

geom\_vline(xintercept = 0) +

scale\_x\_continuous(labels = label\_percent()) +

scale\_fill\_viridis\_d(option = "plasma", end = 0.5,

guide = guide\_legend(reverse = TRUE)) +

labs(x = "Predicted proportion of women in parliament",

y = "Count", fill = "Is zero?") +

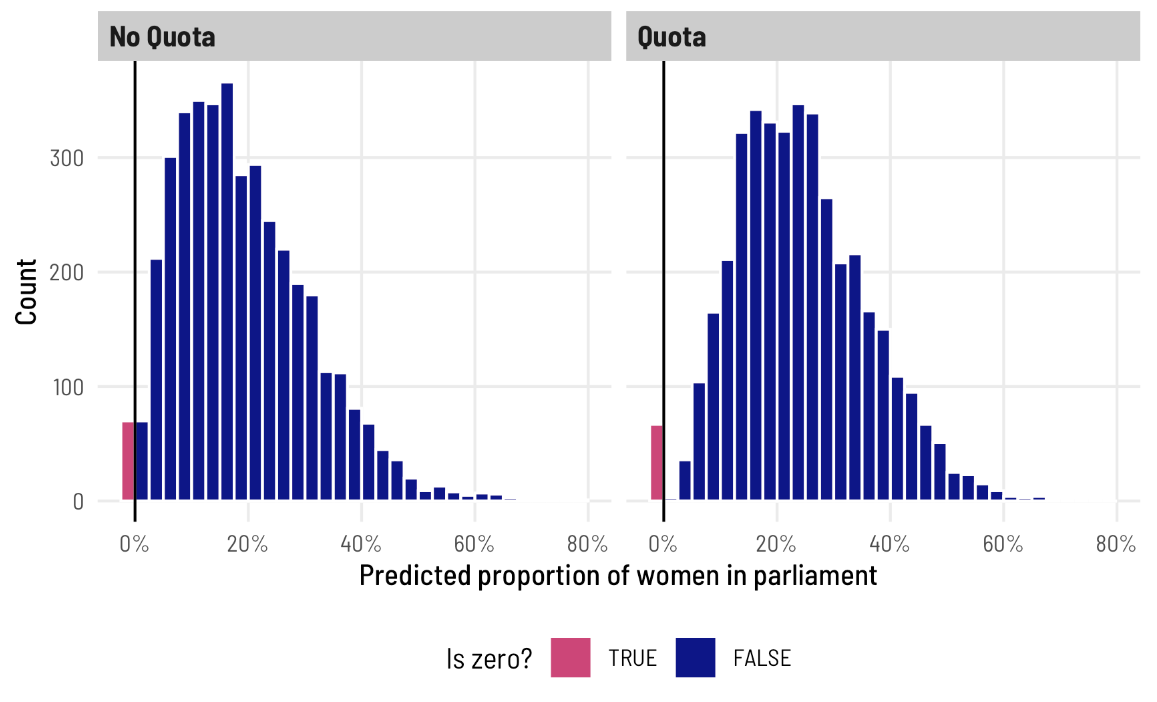
facet\_wrap(vars(quota), ncol = 2,

labeller = labeller(quota = c(`TRUE` = "Quota",

`FALSE` = "No Quota"))) +

theme\_clean() +

theme(legend.position = "bottom")



The process for deciding 0/not 0 is probably determined by different political and social factors, though. It’s likely that having a quota, for instance, should increase the probability of having at least one woman. There are probably a host of other factors—if we were doing this in real life, we could fully specify a model that explains the no women vs. at-least-one-woman split. For now, we’ll just use quota, though (in part because there are actually only 3 zeros!). This model takes a little longer to run because of all the moving parts:

然而，决定0/非0的过程可能由不同的政治和社会因素决定。例如，有一个配额很可能会增加至少有一个女人的概率。可能还有很多其他因素——如果我们在现实生活中这样做，我们可以完全指定一个模型来解释没有女性与至少有一个女性的分裂。现在，我们只使用配额（部分原因是实际上只有3个零！）。由于所有的运动部件，此模型运行需要更长的时间：

model\_beta\_zi <- brm(

bf(prop\_fem ~ quota,

phi ~ quota,

zi ~ quota),

data = vdem\_clean,

family = zero\_inflated\_beta(),

chains = 4, iter = 2000, warmup = 1000,

cores = 4, seed = 1234,

backend = "cmdstanr",

file = "model\_beta\_zi"

)

tidy(model\_beta\_zi, effects = "fixed")

## # A tibble: 6 × 7

## effect component term estimate std.error conf.low conf.high

## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>

## 1 fixed cond (Intercept) -1.42 0.0236 -1.47 -1.38

## 2 fixed cond phi\_(Intercept) 2.37 0.0433 2.29 2.45

## 3 fixed zi (Intercept) -3.53 0.192 -3.93 -3.18

## 4 fixed cond quotaTRUE 0.304 0.0334 0.239 0.368

## 5 fixed cond phi\_quotaTRUE 0.236 0.0696 0.101 0.371

## 6 fixed zi quotaTRUE -5.57 2.64 -12.2 -2.21

We now have two estimates for the zi part: an intercept and a coefficient for quota. Since these are on the logit scale, we can combine them mathematically to figure out the marginal effect of quota on the probability of being 0:

我们现在对zi部分有两个估计：截距和配额系数。由于这些都是logit量表，我们可以用数学方法将它们结合起来，计算出配额对0概率的边际影响：

zi\_intercept <- tidy(model\_beta\_zi, effects = "fixed") %>%

filter(component == "zi", term == "(Intercept)") %>%

pull(estimate)

zi\_quota <- tidy(model\_beta\_zi, effects = "fixed") %>%

filter(component == "zi", term == "quotaTRUE") %>%

pull(estimate)

plogis(zi\_intercept + zi\_quota) - plogis(zi\_intercept)

## b\_zi\_Intercept

## -0.0283

Based on this, having a quota reduces the proportion of 0s in the data by -2.83 percentage points. Unsurprisingly, we can also see this in the posterior predictions. We can look at the predicted proportion of 0s across the two levels of quota to find the marginal effect of having a quota on the 0/not 0 split. To do this with **tidybayes**, we can use the dpar argument in [epred\_draws()](http://mjskay.github.io/tidybayes/reference/add_predicted_draws.html) to also calculated the predicted zero-inflation part of the model (this is omitted by default). Here’s what that looks like:

基于此，具有配额可以将0在数据中的比例降低-2.83个百分点。不出所料，我们也可以在后验预测中看到这一点。我们可以查看两个配额级别中0的预测比例，以发现配额对0/非0划分的边际影响。为了使用tidybayes，我们可以使用epred\_draws（）中的dpar参数来计算模型的预测零通胀部分（默认情况下会省略）。以下是它的样子：

pred\_beta\_zi <- model\_beta\_zi %>%

epred\_draws(newdata = expand\_grid(quota = c(FALSE, TRUE)),

dpar = "zi")

# We now have columns for the overall prediction (.epred) and for the

# zero-inflation probability (zi)

head(pred\_beta\_zi)

## # A tibble: 6 × 7

## # Groups: quota, .row [1]

## quota .row .chain .iteration .draw .epred zi

## <lgl> <int> <int> <int> <int> <dbl> <dbl>

## 1 FALSE 1 NA NA 1 0.186 0.0358

## 2 FALSE 1 NA NA 2 0.188 0.0247

## 3 FALSE 1 NA NA 3 0.187 0.0388

## 4 FALSE 1 NA NA 4 0.187 0.0388

## 5 FALSE 1 NA NA 5 0.191 0.0274

## 6 FALSE 1 NA NA 6 0.186 0.0162

# Look at the average zero-inflation probability across quota

pred\_beta\_zi %>%

group\_by(quota) %>%

median\_hdci(zi)

## # A tibble: 2 × 7

## quota zi .lower .upper .width .point .interval

## <lgl> <dbl> <dbl> <dbl> <dbl> <chr> <chr>

## 1 FALSE 0.0286 1.91e- 2 0.0396 0.95 median hdci

## 2 TRUE 0.000220 3.91e-11 0.00233 0.95 median hdci

pred\_beta\_zi %>%

compare\_levels(variable = zi, by = quota) %>%

median\_hdci()

## # A tibble: 1 × 7

## quota zi .lower .upper .width .point .interval

## <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>

## 1 TRUE - FALSE -0.0281 -0.0389 -0.0179 0.95 median hdci

Without even plotting this, we can see a neat trend. When there’s not a quota, ≈2–3% of the the predicted values are 0; when there is a quota, almost none of the predicted values are 0. Having a quota almost completely eliminates the possibility of having no women MPs. It’s the same marginal effect that we found with [plogis()](https://rdrr.io/r/stats/Logistic.html)—the proportion of zeros drops by -2.8 percentage points when there’s a quota.

甚至不需要对此进行描绘，我们就能看到一个整洁的趋势。当并没有配额时，≈2-3%的预测值为0；当存在配额时，几乎没有一个预测值为0。有一个配额几乎完全消除了没有女议员的可能性。这与我们在plogis（）中发现的边际效应相同——当有配额时，零的比例下降了-2.8个百分点。

We can visualize this uncertainty by plotting the posterior predictions. We could either plot two distributions (for quota being TRUE or FALSE), or we can calculate the difference between quota/no quota to find the average marginal effect of having a quota on the proportion of 0s in the data:

我们可以通过绘制后验预测来可视化这种不确定性。我们可以绘制两种分布图（配额为TRUE或FALSE），也可以计算配额/无配额之间的差异，以找到有配额对数据中0的比例的平均边际影响：

mfx\_quota\_zi <- pred\_beta\_zi %>%

compare\_levels(variable = zi, by = quota)

ggplot(mfx\_quota\_zi, aes(x = zi)) +

stat\_halfeye(.width = c(0.8, 0.95), point\_interval = "median\_hdi",

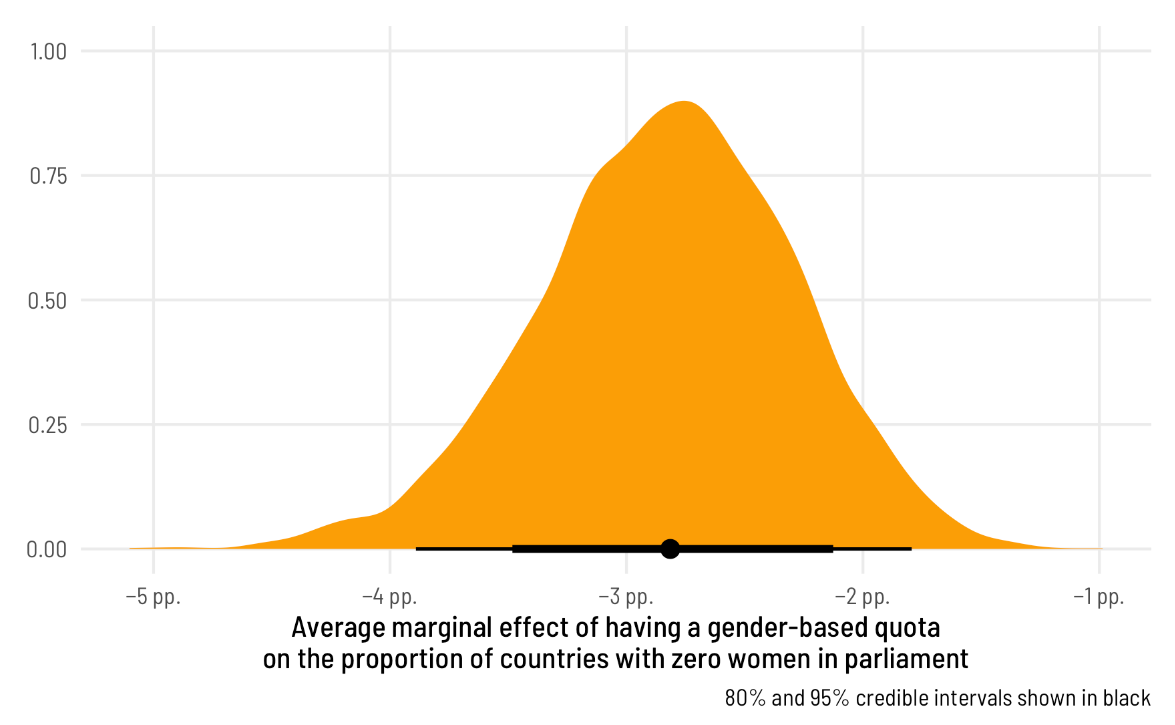
fill = "#fb9e07") +

scale\_x\_continuous(labels = label\_pp) +

labs(x = "Average marginal effect of having a gender-based quota\non the proportion of countries with zero women in parliament",

y = NULL, caption = "80% and 95% credible intervals shown in black") +

theme\_clean()



Finally, we can see how this shows up in the overall predictions from the model. Note how there are a bunch of 0s when quota is FALSE, and no 0s when it is TRUE, as expected:

最后，我们可以看到这在模型的整体预测中是如何表现出来的。请注意，当配额为FALSE时，会有一堆0，当配额是TRUE时，不会有0，正如预期的那样：

beta\_zi\_pred <- model\_beta\_zi %>%

predicted\_draws(newdata = tibble(quota = c(FALSE, TRUE))) %>%

mutate(is\_zero = .prediction == 0,

.prediction = ifelse(is\_zero, .prediction - 0.01, .prediction))

ggplot(beta\_zi\_pred, aes(x = .prediction)) +

geom\_histogram(aes(fill = is\_zero), binwidth = 0.025,

boundary = 0, color = "white") +

geom\_vline(xintercept = 0) +

scale\_x\_continuous(labels = label\_percent()) +

scale\_fill\_viridis\_d(option = "plasma", end = 0.5,

guide = guide\_legend(reverse = TRUE)) +

labs(x = "Predicted proportion of women in parliament",

y = "Count", fill = "Is zero?") +

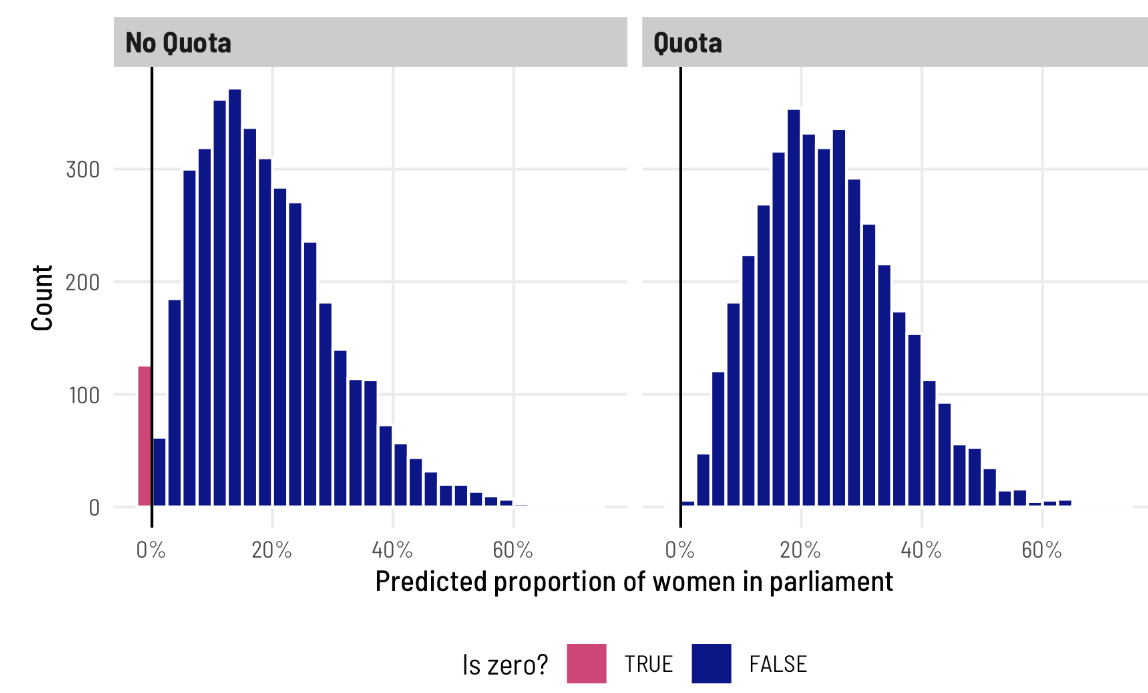
facet\_wrap(vars(quota), ncol = 2,

labeller = labeller(quota = c(`TRUE` = "Quota",

`FALSE` = "No Quota"))) +

theme\_clean() +

theme(legend.position = "bottom")



Again, I think it’s so cool that we can explore all this uncertainty for specific parameters of this model, like the 0/not 0 process! You can’t do this with OLS!

同样，我认为我们可以探索这个模型的特定参数的所有不确定性，比如0/not 0过程，这太酷了！你不能用OLS做到这一点！

**Average marginal effects, incorporating the zero process**

The main thing we’re interested in here, though, is the average marginal effect of having a quota on the proportion of women in parliament, not just the 0/not 0 process. Since we’ve incorporated quota in all the different parts of the model (the μ, the Φ, and the zero-inflated α), we should definitely once again simulate this value using **marginaleffects**—trying to get all the coefficients put together manually is going to be too tricky.

不过，我们在这里感兴趣的主要问题是，配额对议会中女性比例的平均边际影响，而不仅仅是0/not 0过程。由于我们已经在模型的所有不同部分（μ、Φ和零膨胀的α）中加入了配额，我们肯定应该再次使用边际效应来模拟这个值——试图手动将所有系数组合在一起太难了。

ame\_beta\_zi <- model\_beta\_zi %>%

avg\_comparisons(variables = "quota") %>%

posterior\_draws()

ame\_beta\_zi %>% median\_hdi(draw)

## # A tibble: 1 × 6

## draw .lower .upper .width .point .interval

## <dbl> <dbl> <dbl> <dbl> <chr> <chr>

## 1 0.0575 0.0466 0.0688 0.95 median hdi

ggplot(ame\_beta\_zi, aes(x = draw)) +

stat\_halfeye(.width = c(0.8, 0.95), point\_interval = "median\_hdi",

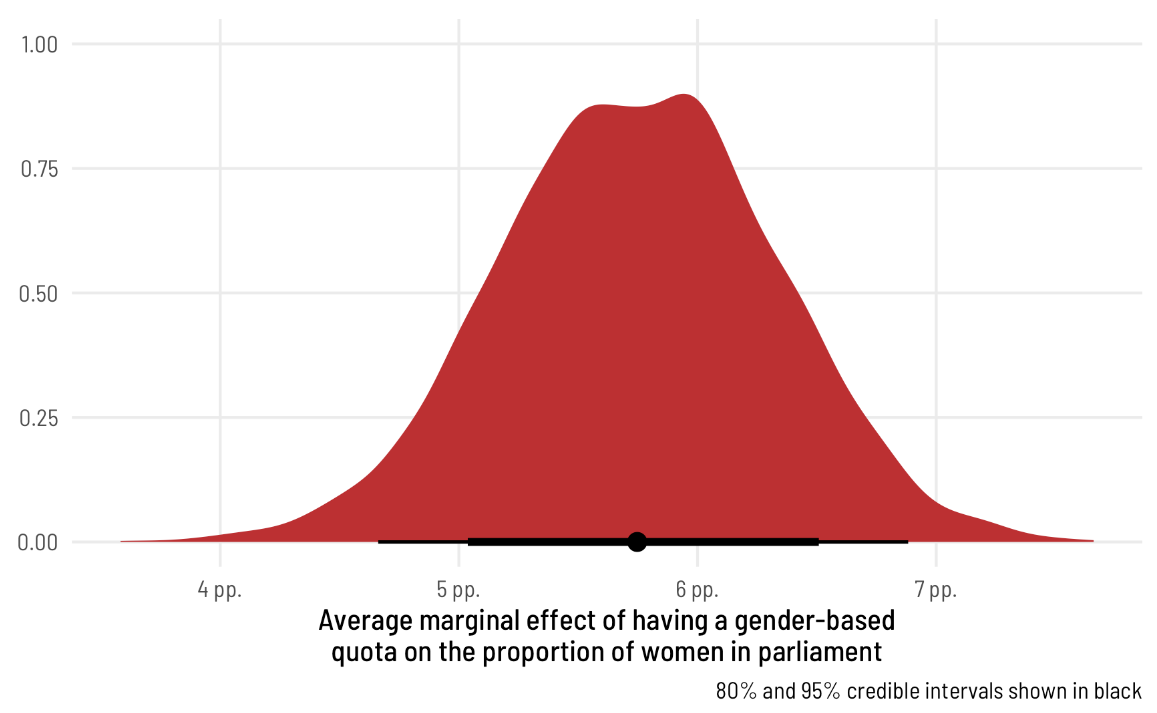
fill = "#bc3032") +

scale\_x\_continuous(labels = label\_pp) +

labs(x = "Average marginal effect of having a gender-based\nquota on the proportion of women in parliament", y = NULL,

caption = "80% and 95% credible intervals shown in black") +

theme\_clean()



The average marginal effect of having a quota on the proportion of women MPs in an average country, therefore, is 6.65 percentage points.

因此，在一个普通国家，配额对女议员比例的平均边际影响为6.65个百分点。

**Special case #1: Zero-one-inflated beta regression**

With our data here, no countries have parliaments with 100% women (cue [that RBG “when there are nine” quote](https://bookshop.org/lists/when-there-are-nine-honoring-ruth-bader-ginsburg)). But what if you have both 0s *and* 1s in your outcome? Zero-inflated beta regression handles the zeros just fine (it’s in the name of the model!), but it can’t handle both 0s and 1s.

根据我们的数据，没有哪个国家的议会中有100%的女性（暗示RBG“当有九个时”引用）。但是，如果你的结果中同时有0和1呢？零膨胀贝塔回归可以很好地处理零（这是模型的名称！），但它不能同时处理0和1。

Fortunately there’s a variation for zero-one-inflated beta (ZOIB) regression. I won’t go into the details here—[Matti Vuore has a phenomenal tutorial about how to run these models](https://mvuorre.github.io/posts/2019-02-18-analyze-analog-scale-ratings-with-zero-one-inflated-beta-models/). In short, ZOIB regression makes you model a mixture of *three* things:

根据我们的数据，没有哪个国家的议会中有100%的女性（暗示RBG“当有九个时”引用）。但是，如果你的结果中同时有0和1呢？零膨胀贝塔回归可以很好地处理零（这是模型的名称！），但它不能同时处理0和1。

1. A logistic regression model that predicts if an outcome is either 0 or 1 or not 0 or 1, defined by α (or alternatively, a model that predicts if outcomes are extreme (0 or 1) or not (between 0 and 1); thanks to [Isabella Ghement for this way of thinking about it](https://twitter.com/IsabellaGhement/status/1457789676387848205)!)

逻辑回归模型，预测结果是0还是1，或者不是0或1，由α定义（或者，预测结果是否极端（0或1）（在0和1之间）的模型；感谢Isabella Ghement的这种思考方式！）

1. A logistic regression model that predicts if any of the 0 or 1 outcomes are actually 1s, defined by γ (or alternatively, a model that predicts if the extreme values are 1)
2. A beta regression model that predicts if an outcome is between 0 and 1 if it’s not zero or not one, defined by μ and Φ (or alternatively, a model that predicts the non-extreme (0 or 1) values)

2.逻辑回归模型，预测0或1的结果中是否有任何一个实际上是1，由γ定义（或者，预测极值是否为1的模型）

3.β回归模型，预测结果是否在0和1之间，如果结果不是零或不是1，由μ和Φ定义（或者，预测非极端（0或1）值的模型）

When using **brms**, you get to model all these different parts:

brm(

bf(

outcome ~ covariates, # The mean of the 0-1 values, or mu

phi ~ covariates, # The precision of the 0-1 values, or phi

zoi ~ covariates, # The zero-or-one-inflated part, or alpha

coi ~ covariates # The one-inflated part, conditional on the 0s, or gamma

),

data = whatever,

family = zero\_one\_inflated\_beta(),

...

)

Pulling out all these different parameters, plotting their distributions, and estimating their average marginal effects looks exactly the same as what we did earlier with the zero-inflated model—all the **brms**, **tidybayes**, and **marginaleffects** functions we used for marginal effects and fancy plots work the same. All the coefficients are on the logit scale, except Φ, which is on the log scale.

提取所有这些不同的参数，绘制它们的分布，并估计它们的平均边际效应，看起来与我们之前使用零膨胀模型所做的完全相同——我们用于边际效应和幻想图的所有brms、tidybayes和边际效应函数都是一样的。所有系数都在logit标度上，除了Φ在对数标度上。

**Special case #2: One-inflated beta regression**

What if your proportion-based outcome has a bunch of 1s and no 0s? It would be neat if there were a one\_inflated\_beta() family, but there’s not ([also by design](https://github.com/paul-buerkner/brms/issues/942)). But never fear! There are two ways to do one-inflated regression:

如果你基于比例的结果有一堆1而没有0呢？如果有one\_inflated\_beta（）家族，那就太好了，但没有（也是出于设计）。但永远不要害怕！有两种方法可以进行一次膨胀回归：

1. Reverse your outcome so that it’s 1 - outcome instead of outcome and then use zero-inflated regression. In the case of the proportion of women MPs, we would instead look at the proportion of not-women MPs:

1.颠倒你的结果，使其为1-结果而不是结果，然后使用零膨胀回归。在女性议员比例的情况下，我们将转而关注非女性议员的比例：

mutate(prop\_not\_fem = 1 - prop\_fem)

1. Use zero-one-inflated beta regression and force the conditional one parameter (coi, or the γ in that model) to be one, meaning that 100% of the 0/not 0 splits would actually be 1: bf(..., coi = 1)

使用零一膨胀贝塔回归，并强制条件一参数（coi，或该模型中的γ）为一，这意味着100%的0/非0分裂实际上是1:bf（…，coi=1）

Everything else that we’ve explored in this post—posterior predictions, average marginal effects, etc.—will all work as expected.

我们在这篇文章中探索的其他一切——后验预测、平均边际效应等——都将如预期的那样发挥作用。

**Super fancy detailed model with lots of moving parts, just for fun**

Throughout this post, for the sake of simplicity we’ve really only used one or two covariates at a time (quota and/or polyarchy). Real world models are more complex: we can use more covariates, multilevel hierarchical structures, or weighting, for instance. This all works just fine with zero|one|zero-one-inflated beta regression thanks to the power of **brms**, which handles all these extra features natively.

在这篇文章中，为了简单起见，我们一次只使用了一到两个协变量（配额和/或一夫多妻制）。现实世界的模型更复杂：例如，我们可以使用更多的协变量、多级层次结构或加权。由于brms的强大功能，这一切在零|one|零壹膨胀的beta回归中都能很好地工作，它可以原生地处理所有这些额外的功能。

Just for fun here at the end, let’s run a more fully specified model with more covariates and a multilevel structure that accounts for year and region. This will let us look at the larger V-Dem dataset from 2010–2020 instead of just 2015.

最后，为了好玩，让我们运行一个更全面的模型，它包含更多的协变量和一个考虑年份和地区的多级结构。这将使我们看到2010-2020年而不仅仅是2015年的更大的V-Dem数据集。

**Set better priors**

To make sure this runs as efficiently as possible, we’ll set our own priors instead of relying on the default **brms** priors. This is especially important for beta models because of how parameters are used internally. Remember that most of our coefficients are on a logit scale, and those logits correspond to probabilities or proportions. While logits can theoretically range from −∞ to ∞, in practice they’re a lot more limited. For instance, we can convert some small percentages to the logit scale with [qlogis()](https://rdrr.io/r/stats/Logistic.html):

为了确保它尽可能高效地运行，我们将设置自己的优先级，而不是依赖于默认的brms优先级。这对于测试版模型尤其重要，因为参数是如何在内部使用的。请记住，我们的大多数系数都是在logit量表上的，这些logit对应于概率或比例。虽然logits在理论上可以从-∞到∞，但在实践中它们的范围要有限得多。例如，我们可以使用qlogis（）将一些小百分比转换为logit比例：

qlogis(c(0.01, 0.001, 0.0001))

## [1] -4.60 -6.91 -9.21

qlogis(c(0.99, 0.999, 0.9999))

## [1] 4.60 6.91 9.21

A 1%/99% value is 4.6 as a logit, while 0.01%/99.99% is 9. Most of the proportions we’ve seen in the models so far are much smaller than that—logits of like ≈0–2 at most. By default, **brms** puts a flat prior on coefficients in beta regression, so values like 10 or 15 could occur, which are excessively massive when converted to probabilities! Look at this plot to see the relationship between the logit scale and the probability scale—there are big changes in probability between −4ish and 4ish, but once you start getting into the 5s and beyond, the probability is all essentially the same.

作为logit，1%/99%的值为4.6，而0.01%/99.99%为9。到目前为止，我们在模型中看到的大多数比例都比这个小得多——logits最多大约≈0-2。默认情况下，brms在贝塔回归中对系数设置一个平坦的先验，因此可能会出现10或15这样的值，当转换为概率时，这些值太大了！看看这张图，可以看到logit量表和概率量表之间的关系——−4英寸和4英寸之间的概率有很大的变化，但一旦你开始进入5秒及以上，概率基本上是一样的。

tibble(x = seq(-8, 8, by = 0.1)) %>%

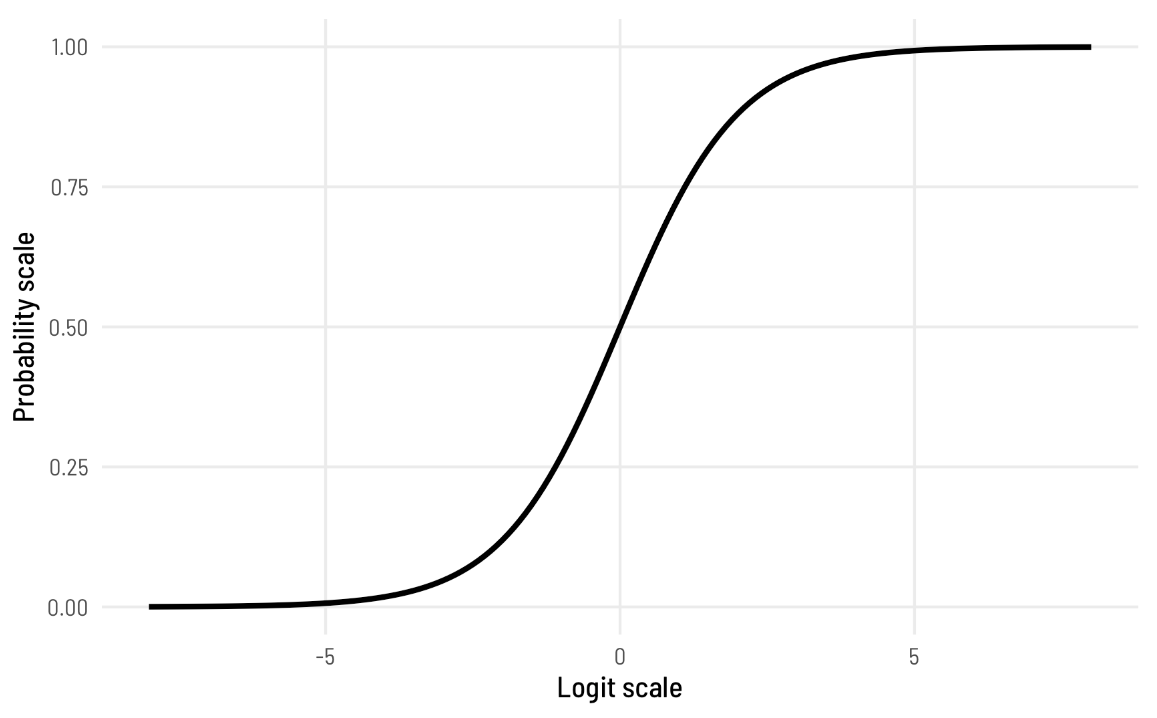
mutate(y = plogis(x)) %>%

ggplot(aes(x = x, y = y)) +

geom\_line(size = 1) +

labs(x = "Logit scale", y = "Probability scale") +

theme\_clean()



To help speed up the model (and on philosophical grounds, since we know prior information about these model coefficients), we should tell **brms** that the logit-scale parameters in the model aren’t ever going to get huge.

为了帮助加快模型的速度（从哲学角度来看，因为我们知道这些模型系数的先验信息），我们应该告诉brms，模型中的logit尺度参数永远不会变得巨大。

To do this, we first need to define the formula for our fancy complex model. I added a bunch of plausible covariates to different portions of the model, along with random effects for region. In real life this should all be driven by theory.

要做到这一点，我们首先需要定义我们的花式复杂模型的公式。我在模型的不同部分添加了一堆看似合理的协变量，以及区域的随机效应。在现实生活中，这一切都应该由理论驱动。

vdem\_clean <- vdem\_clean %>%

# Scale polyarchy back down to 0-1 values to help Stan with modeling

mutate(polyarchy = polyarchy / 100) %>%

# Make region and year factors instead of numbers

mutate(region = factor(region),

year = factor(year))

# Create the model formula

fancy\_formula <- bf(

# mu (mean) part

prop\_fem ~ quota + polyarchy + corruption +

civil\_liberties + (1 | year) + (1 | region),

# phi (precision) part

phi ~ quota + (1 | year) + (1 | region),

# alpha (zero-inflation) part

zi ~ quota + polyarchy + (1 | year) + (1 | region)

)

We can then feed that formula to [get\_prior()](https://rdrr.io/pkg/brms/man/get_prior.html) to see what the default priors are and how to change them:

然后，我们可以将该公式提供给get\_prior（），以查看默认优先级是什么以及如何更改它们：

get\_prior(

fancy\_formula,

data = vdem\_clean,

family = zero\_inflated\_beta()

)

There are a *ton* of potential priors we can set here! If we really wanted, we could set the mean and standard deviation separately for each individual coefficient, but that’s probably excessive for this case.

我们可以在这里设置大量潜在的先验！如果我们真的想的话，我们可以分别为每个系数设置平均值和标准差，但在这种情况下，这可能太过分了。

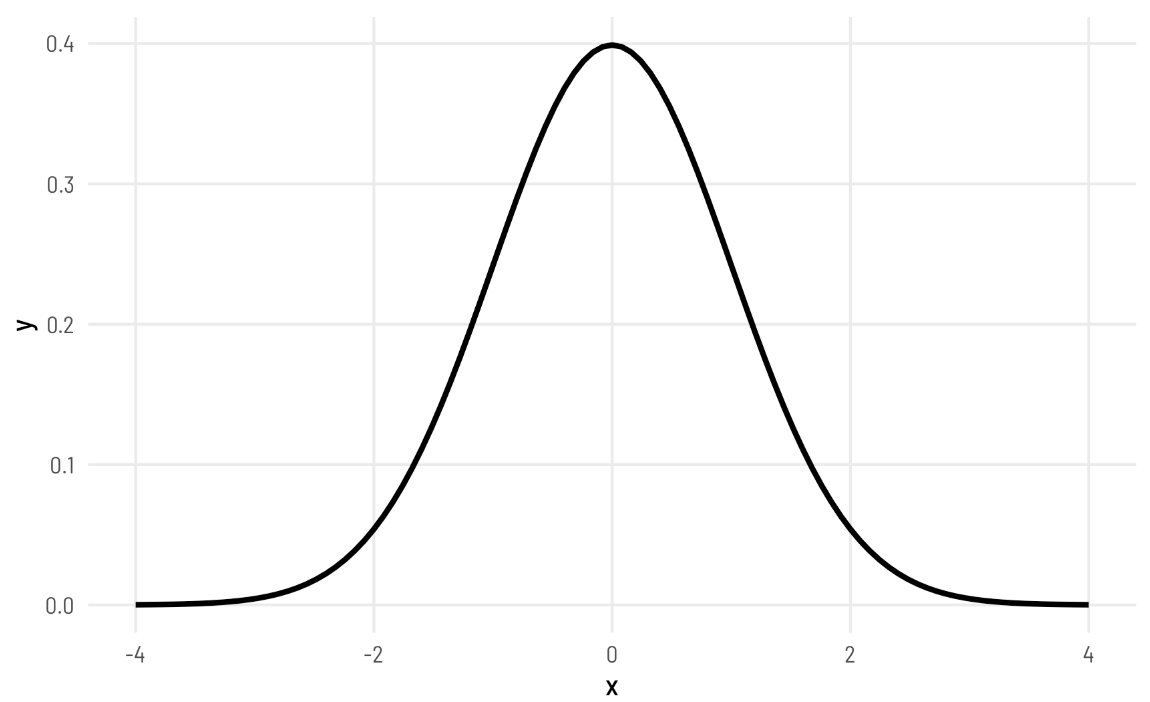
You can see that everything with class b (for β coefficient in a regression model) has a flat prior. We need to narrow that down, perhaps with something like a normal distribution centered at 0 with a standard deviation of 1. Values here won’t get too big and will mostly be between −2 and 2:

你可以看到，所有具有类b（对于回归模型中的β系数）的事物都有一个平坦的先验。我们需要缩小范围，也许可以用一个以0为中心、标准偏差为1的正态分布。这里的值不会太大，通常在−2和2之间：

ggplot(data = tibble(x = -4:4), aes(x = x)) +

geom\_function(fun = dnorm, args = list(mean = 0, sd = 1), linewidth = 1) +

theme\_clean()



To set the priors, we can create a list of priors as a separate object that we can then feed to the main [brm()](https://rdrr.io/pkg/brms/man/brm.html) function. We’ll keep the default prior for the intercepts.

要设置先验，我们可以创建一个先验列表作为一个单独的对象，然后将其提供给主brm（）函数。我们将保留拦截的默认优先级。

priors <- c(set\_prior("student\_t(3, 0, 2.5)", class = "Intercept"),

set\_prior("normal(0, 1)", class = "b"))

**Run the model**

Let’s run the model! This will probably take a (long!) while to run. On my new 2021 MacBook Pro with a 10-core M1 Max CPU, running 4 chains with two CPU cores per chain (with **cmdstanr**’s [support for within-chain threading](https://cran.r-project.org/web/packages/brms/vignettes/brms_threading.html)) takes about 6 minutes. It probably takes a ***lot*** longer on a less powerful machine. To help Stan with estimation, I also set init to help with some of the parameters of the beta distribution, and increased adapt\_delta and max\_treedepth a little. I only knew to do this because Stan complained when I used the defaults and told me to increase them :)

让我们运行模型！这可能需要一段时间才能运行。在我的新2021 MacBook Pro上，使用10核M1 Max CPU，运行4个链，每个链有两个CPU核（cmdstanr支持链内线程）大约需要6分钟。在功能较弱的机器上可能需要更长的时间。为了帮助Stan进行估计，我还设置了init来帮助处理beta分布的一些参数，并稍微增加了adapte\_delta和max\_treedepth。我之所以知道这样做，是因为当我使用默认值并告诉我增加默认值时，Stan抱怨道：）

fancy\_model <- brm(

fancy\_formula,

data = vdem\_clean,

family = zero\_inflated\_beta(),

prior = priors,

init = 0,

control = list(adapt\_delta = 0.97,

max\_treedepth = 12),

chains = 4, iter = 2000, warmup = 1000,

cores = 4, seed = 1234,

threads = threading(2),

backend = "cmdstanr",

file = "fancy\_model"

)

There are a handful of divergent chains but since this isn’t an actual model that I’d use in a publication, I’m not too worried. In real life, I’d increase the adapt\_delta or max\_treedepth parameters even more or do some other fine tuning, but we’ll just live with the warnings for now.

有一些不同的链条，但由于这不是我在出版物中使用的实际模型，我并不太担心。在现实生活中，我会更多地增加adapte\_delta或max\_treedepth参数，或者进行一些其他微调，但我们现在只能接受这些警告。

**Analyze and plot the results**

We now have a fully specified multilevel model with all sorts of rich moving parts: we account for the mean and precision of the beta distribution *and* we account for the zero-inflation process, all with different covariates with region-specific offsets. Here are the full results, with medians and 95% credible intervals for all the different coefficients:

我们现在有了一个具有各种丰富运动部分的完全指定的多级模型：我们考虑了贝塔分布的平均值和精度，我们考虑了零通货膨胀过程，所有这些都具有不同的协变量和特定区域的偏移。以下是所有不同系数的中值和95%可信区间的完整结果：

[tidy](https://generics.r-lib.org/reference/tidy.html)(fancy\_model)

We can’t really interpret any of these coefficients directly, since (1) they’re on different scales (the phi parts are all logged; all the other parts are logits), and (2) we need to combine coefficients with intercepts in order to back-transform them into percentage point values, and doing that mathematically is tricky.

我们真的不能直接解释这些系数中的任何一个，因为（1）它们在不同的尺度上（phi部分都被记录；所有其他部分都是logit），以及（2）我们需要将系数与截距相结合，以便将它们反变换为百分点值，而在数学上做到这一点很棘手。

We’ll focus on finding the marginal effect of quota, since that’s the main question we’ve been exploring throughout this post (and it was the subject of the original Tripp and Kang ([2008](https://www.andrewheiss.com/blog/2021/11/08/beta-regression-guide/#ref-TrippKang:2008)) paper). For fun, we’ll also look at polyarchy, since it’s a continuous variable.

我们将专注于寻找配额的边际效应，因为这是我们在这篇文章中一直在探索的主要问题（这也是Tripp和Kang（2008）论文的主题）。为了好玩，我们还将研究一夫多妻制，因为它是一个连续变量。

As before, **marginaleffects** makes it really easy to get posterior predictions of the difference between quota and no quota with [avg\_comparisons()](https://rdrr.io/pkg/marginaleffects/man/comparisons.html)

和以前一样，marginalseffects使得使用avg\_comparisons（）可以很容易地获得配额和无配额之间差异的后验预测

**More about these epreds and random effects**

A year after writing this post, I wrote a couple other guides about the [differences between posterior predictions, linear predictions, and epreds](https://www.andrewheiss.com/blog/2022/09/26/guide-visualizing-types-posteriors/) + [different ways of handling random effects in predictions](https://www.andrewheiss.com/blog/2022/11/29/conditional-marginal-marginaleffects/). Check those out for more details.

更多关于这些epred和随机效应的信息

写这篇文章一年后，我写了另外几本指南，介绍了后验预测、线性预测和epreds+处理预测中随机效应的不同方法之间的差异。查看这些以了解更多详细信息。

ame\_fancy\_zi\_quota <- fancy\_model %>%

avg\_comparisons(variables = "quota") %>%

posterior\_draws()

ame\_fancy\_zi\_quota %>% median\_hdi(draw)

## # A tibble: 1 × 6

## draw .lower .upper .width .point .interval

## <dbl> <dbl> <dbl> <dbl> <chr> <chr>

## 1 0.0653 0.0556 0.0749 0.95 median hdi

ggplot(ame\_fancy\_zi\_quota, aes(x = draw)) +

stat\_halfeye(.width = c(0.8, 0.95), point\_interval = "median\_hdi",

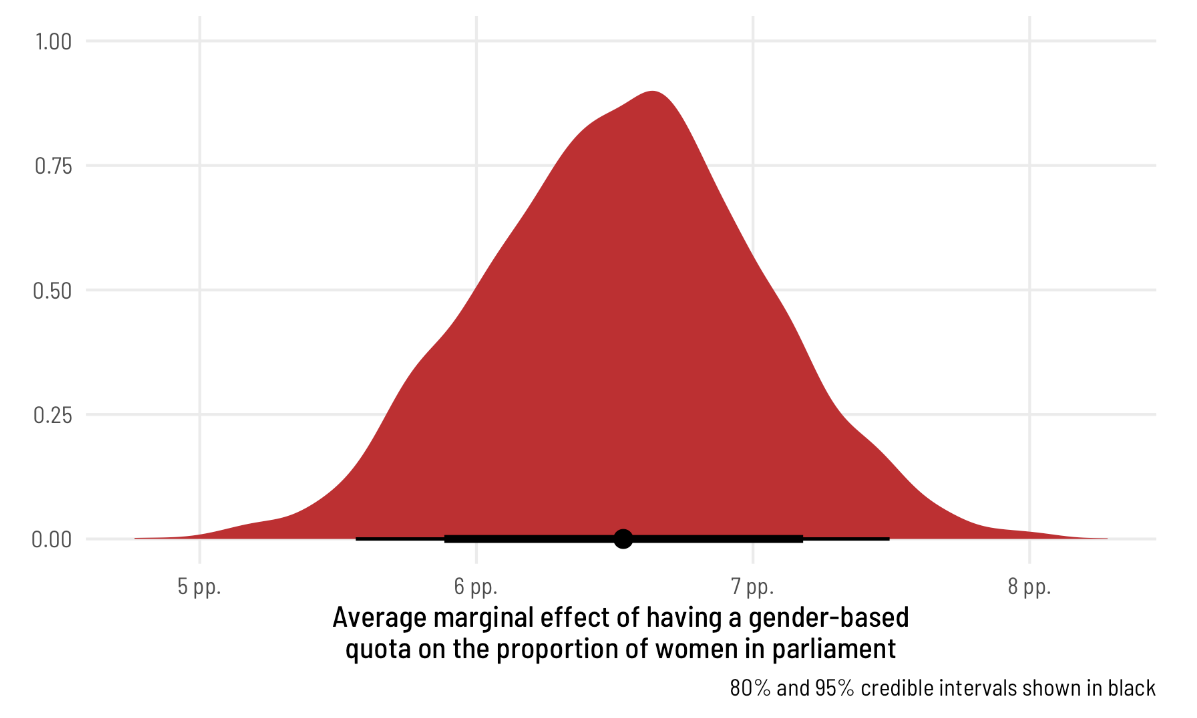
fill = "#bc3032") +

scale\_x\_continuous(labels = label\_pp) +

labs(x = "Average marginal effect of having a gender-based\nquota on the proportion of women in parliament", y = NULL,

caption = "80% and 95% credible intervals shown in black") +

theme\_clean()



r\_fancy <- ame\_fancy\_zi\_quota %>% median\_hdi(draw)

After accounting for democracy, corruption, respect for civil liberties, year, and region, the proportion of women MPs in countries with a parliamentary gender quota is 6.5 percentage points higher than in countries without a quota, on average, with a 95% credible interval ranging from 5.6 to 7.5 percentage points.

考虑到民主、腐败、尊重公民自由、年份和地区，有议会性别配额的国家的女议员比例平均比没有配额的国家高6.5个百分点，95%的可信区间为5.6至7.5个百分点。

For fun, we can look at how the predicted outcome changes across both polyarchy and quota simultaneously (and for fun we’ll do it with **marginaleffects**, though it’s also doable with tidybayes::epred\_draws() %>% compare\_levels()). Because we didn’t use any interaction terms, the slope will be the same across both levels of quota, but the plot still looks neat. Weirdly, due to the constellation of controls we included in the model, the predicted proportion of women MPs *decreases* with more democracy. But that’s not super important—if we were really only interested in the quota effect and we had included a sufficient set of variables to close backdoor paths, the coefficients for all other variables shouldn’t be interpreted ([Keele, Stevenson, and Elwert 2020](https://www.andrewheiss.com/blog/2021/11/08/beta-regression-guide/#ref-KeeleStevensonElwert:2020)). We’ll pretend that’s the case here—this code mostly just shows how you can do all sorts of neat stuff with **marginaleffects**.

为了好玩，我们可以看看预测结果如何同时在多配偶制和配额制中发生变化（为了好玩，我会用边际效应来做这件事，尽管它也可以用tidybayes来做：epred\_draws（）%>%compare\_levels（））。因为我们没有使用任何交互术语，所以两个级别的配额的斜率都是相同的，但情节看起来仍然很整洁。奇怪的是，由于我们在模型中纳入了一系列控制措施，预测的女议员比例随着民主程度的提高而下降。但这并不是非常重要——如果我们真的只对配额效应感兴趣，并且我们已经包括了一组足够的变量来关闭后门路径，那么就不应该解释所有其他变量的系数（Keele、Stevenson和Elwert 2020）。我们假设这里是这样的——这段代码主要展示了如何利用边际效应做各种巧妙的事情。

ame\_fancy\_zi\_polyarchy\_quota <- fancy\_model %>%

predictions(newdata = datagrid(quota = unique,

polyarchy = seq(0, 1, by = 0.1))) %>%

posterior\_draws() %>%

# Scale polyarchy back up for plotting

mutate(polyarchy = polyarchy \* 100)

# With tidybayes instead

# ame\_fancy\_zi\_polyarchy\_quota <- fancy\_model %>%

# epred\_draws(newdata = datagrid(model = fancy\_model,

# quota = unique,

# polyarchy = seq(0, 1, by = 0.1)))

ggplot(ame\_fancy\_zi\_polyarchy\_quota,

aes(x = polyarchy, y = draw, color = quota, fill = quota)) +

stat\_lineribbon(aes(fill\_ramp = stat(level))) +

scale\_y\_continuous(labels = label\_percent()) +

scale\_fill\_viridis\_d(option = "plasma", end = 0.8) +

scale\_color\_viridis\_d(option = "plasma", end = 0.8) +

scale\_fill\_ramp\_discrete(range = c(0.2, 0.7)) +

facet\_wrap(vars(quota), ncol = 2,

labeller = labeller(quota = c(`TRUE` = "Quota",

`FALSE` = "No Quota"))) +

labs(x = "Polyarchy (democracy)",

y = "Predicted proportion of women MPs",

fill = "Quota", color = "Quota",

fill\_ramp = "Credible interval") +

theme\_clean() +

theme(legend.position = "bottom")

