**The Parallax View**

Unveiling the relationship between UFO sightings and population, we crossed the threshold of normality underpinning linear models to construct a *generalised linear model* based on the more theoretically satisfying Poisson distribution.

On inspection, however, this model revealed itself to be less well suited to the data than we had, in our tragic ignorance, hoped. While it appeared, on visual inspection, to capture some features of the data, the predictive posterior density plot demonstrated that it still fell short of addressing the subtleties of the original.

In this post, we will seek to overcome this sad lack in two ways: firstly, we will subject our models to pitiless mathematical scrutiny to assess their ability to describe the data. With our eyes irrevocably opened to these techniques, we will construct an ever more complex armillary with which to approach the unknowable truth.

**Critical Omissions of Information**

Our previous post showed the different fit of the Poisson model to the data from the simple Gaussian linear model. When presented with a grim array of potential models, however, it is crucial to have reliable and quantitative mechanisms to select amongst them.

The eldritch procedure most suited to this purpose, *model selection*, in our framework, draws on *information criteria* that express the *relative* effectiveness of models at creating sad mockeries of the original data.

It is important to reiterate that these approaches do not speak to an absolute underlying truth; information criteria allow us to choose between models, assessing which has most closely assimilated the madness and chaos of the data. For LOO-CV, this results in an *expected log predictive density* (elpd) for each model. The model with the lowest elpd is the least-warped mirror of reality amongst those we subject to scrutiny.

There are many fragile subtleties to model selection, of which we will mention only two here. Firstly, in general, the greater the number of predictors or variables incorporated into a model, the more closely it will be able to mimic the original data. This is problematic, in that a model can become *overfit* to the original data and thus be unable to represent previously unseen data accurately — it learns to mimic the form of the observed data at the expense of uncovering its underlying reality. The LOO-CV technique avoids this trap by, in effect, withholding data from the model to assess its ability to make accurate inferences on previously unseen data.

The second consideration in model selection is that the information criteria scores of models, such as (elpd) in LOO-CV, are subject to *standard error* in their assessment; the score itself is not a perfect metric of model performance, but a cunning approximation. As such we will only consider one model to have outperformed its competitors if the difference in their relative elpd is several times greater than this standard error.

With this understanding in hand, we can now ruthlessly quantify the effectiveness of the Gaussian linear model against the Poisson generalised linear model.

**Gaussian vs. the Poisson**

The original model presented before our subsequent descent into horror was a simple linear Gaussian, produced through use of ggplot2‘s geom\_smooth function. To compare this meaningfully against the Poisson model of the previous post, we must now recreate this model using the, now hideously familar, tools of Bayesian modelling with Stan.

Show Gaussian model specification code.

population\_model\_normal.stan

data {

// Number of rows (observations)

int observations;

// Predictor (population of state)

vector[ observations ] population;

// Response (counts)

real counts[observations];

}

parameters {

// Intercept

real< lower=0 > a;

// Slope

real< lower=0 > b;

// Standard deviation

real< lower=0 > sigma;

}

model {

// Priors

a ~ normal( 0, 5 );

b ~ normal( 0, 5 );

sigma ~ cauchy( 0, 2.5 );

// Model

counts ~ normal( a + population \* b, sigma );

}

generated quantities {

// Posterior predictions

vector[observations] counts\_pred;

// Log likelihood (for LOO)

vector[observations] log\_lik;

for (n in 1:observations) {

log\_lik[n] = normal\_lpdf( counts[n] | a + population[n]\*b, sigma );

counts\_pred[n] = normal\_rng( a + population[n]\*b, sigma );

}

}

With both models straining in their different directions towards the light, we apply LOO-CV cross validation to assess their effectiveness at predicting the data.

Show LOO-CV comparison code.

LOO-CV R-code snippet. Full code is at the end of this post.

...

# Compare models with LOO

log\_lik\_normal <- extract\_log\_lik(fit\_ufo\_pop\_normal, merge\_chains = FALSE)

r\_eff\_normal <- relative\_eff(exp(log\_lik\_normal))

loo\_normal <- loo(log\_lik\_normal, r\_eff = r\_eff\_normal, cores = 2)

log\_lik\_poisson <- extract\_log\_lik(fit\_ufo\_pop\_poisson, merge\_chains = FALSE)

r\_eff\_poisson <- relative\_eff(exp(log\_lik\_poisson))

loo\_poisson <- loo(log\_lik\_poisson, r\_eff = r\_eff\_poisson, cores = 2)

...

> compare( loo\_normal, loo\_poisson )

elpd\_diff se

-8576.1 712.5

The information criterion shows that the complexity of the Poisson model does not, in fact, produce a more effective model than the false serenity of the Gaussian[2](https://www.weirddatascience.net/index.php/2019/04/17/bayes-vs-the-invaders-part-three-the-parallax-view/#easy-footnote-bottom-2-654). The negative elpd\_diff of the compare function supports the first of the two models, and the magnitude being over twelve times greater than the standard error leaves little doubt that the difference is significant. We must, it seems, look further.

With these techniques for selecting between models in hand, then, we can move on to constructing ever more complex attempts to dispel the darkness.

**Trials without End**

The Poisson distribution, whilst appropriate for many forms of count data, suffers from fundamental limits to its understanding. The single parameter of the Poisson, \(\lambda\), enforces that the mean and variance of the data are equal. When such comforting falsehoods wither in the pale light of reality, we must move beyond the gentle chains in which the Poisson binds us.

The next horrific evolution, then, is the [*negative binomial*](https://en.wikipedia.org/wiki/Negative_binomial_distribution) distribution, which similarly speaks to count data, but presents a *dispersion* parameter (\(\phi\)) that allows the variance to exceed the mean

With our arcane theoretical library suitably expanded, we can now transplant the still-beating Poisson heart of our earlier generalised linear model with the more complex machinery of the negative binomial:

$$\begin{eqnarray}  
y &\sim& \mathbf{NegBinomial}(\mu, \phi)\\  
\log(\mu) &\sim& \alpha + \beta x\\  
\alpha &\sim& \mathcal{N}(0, 1)\\  
\beta &\sim& \mathcal{N}(0, 1)\\  
\phi &\sim& \mathbf{HalfCauchy}(2)  
\end{eqnarray}$$

As with the Poisson, our negative binomial generalised linear model employs a log link function to transform the linear predictor. The Stan code for this model is given below.

Show negative binomial model specification code.

population\_model\_negbinomial.stan

data {

// Number of rows (observations)

int observations;

// Predictor (population of state)

vector[ observations ] population\_raw;

// Response (counts)

int counts[observations];

}

transformed data {

// Center and scale the predictor

vector[ observations ] population;

population = ( population\_raw - mean( population\_raw ) ) / sd( population\_raw );

}

parameters {

// Negative binomial dispersion parameter

real phi;

// Intercept

real a;

// Slope

real b;

}

transformed parameters {

vector[observations] mu;

mu = a + b\*population;

}

model {

// Priors

a ~ normal( 0, 1 );

b ~ normal( 0, 1 );

phi ~ cauchy( 0, 5 );

// Model

// Uses the log version of the neg\_binomial\_2 to avoid

// manual exponentiation of the linear predictor.

// (This avoids numerical problems in the calculations.)

counts ~ neg\_binomial\_2\_log( mu, phi );

}

generated quantities {

vector[observations] counts\_pred;

vector[observations] log\_lik;

for (n in 1:observations) {

log\_lik[n] = neg\_binomial\_2\_log\_lpmf( counts[n] | mu[n], phi );

counts\_pred[n] = neg\_binomial\_2\_log\_rng( mu[n], phi );

}

}

With this model fit, we can compare its whispered falsehoods against both the original linear Gaussian model and the Poisson GLM:

Show LOO-CV comparison code.

Code snippet for calculating the LOO-CV elpd for three models. The full R code for building and comparing all models is listed at the end of this post.

...

# Compare models with LOO

log\_lik\_normal <- extract\_log\_lik(fit\_ufo\_pop\_normal, merge\_chains = FALSE)

r\_eff\_normal <- relative\_eff(exp(log\_lik\_normal))

loo\_normal <- loo(log\_lik\_normal, r\_eff = r\_eff\_normal, cores = 2)

log\_lik\_poisson <- extract\_log\_lik(fit\_ufo\_pop\_poisson, merge\_chains = FALSE)

r\_eff\_poisson <- relative\_eff(exp(log\_lik\_poisson))

loo\_poisson <- loo(log\_lik\_poisson, r\_eff = r\_eff\_poisson, cores = 2)

log\_lik\_negbinom <- extract\_log\_lik(fit\_ufo\_pop\_negbinom, merge\_chains = FALSE)

r\_eff\_negbinom <- relative\_eff(exp(log\_lik\_negbinom))

loo\_negbinom <- loo(log\_lik\_negbinom, r\_eff = r\_eff\_negbinom, cores = 2)

...

> compare( loo\_poisson, loo\_negbinom )

elpd\_diff se

8880.8 721.9

With the first comparison, it is clear that the sinuous flexibility offered by the dispersion parameter, \(\phi\), of the negative binomial allows that model to mould itself much more effectively to the data than the Poisson. The elpd\_diff score is positive, indicating that the second of the two compared models is favoured; the difference is over twelve times the standard error, giving us confidence that the negative binomial model is meaningfully more effective than the Poisson.

Whilst superior to the Poisson, does this adaptive capacity allow the negative binomial model to render the naïve Gaussian linear model obsolete?

> compare( loo\_normal, loo\_negbinom )

elpd\_diff se

304.7 30.9

The negative binomial model subsumes the Gaussian with little effort. The elpd\_diff is almost ten times the standard error in favour of the negative binomial GLM, giving us confidence in choosing it. From here on, we will rely on the negative binomial as the core of our schemes.

**Overlapping Realities**

The improvements we have seen with the negative binomial model allow us to discard the Gaussian and Poisson models with confidence. It is not, however, sufficient to fill the gaping void induced by our belief that the sightings of abnormal aerial phenomena in differing US states vary differently with their human population.

To address this question we must ascertain whether allowing our models to unpick the individual influence of states will improve their predictive ability. This, in turn, will lead us into the gnostic insanity of *hierarchical models*, in which we group predictors in our models to account for their shadowy underlying structures.

**Limpid Pools**

The first step on this path is to allow part of the linear function underpinning our model, specifically the intercept value, \(\alpha\), to vary between different US states. In a simple linear model, this causes the line of best fit for each state to meet the y-axis at a different point, whilst maintaining a constant slope for all states. In such a model, the result is a set of parallel lines of fit, rather than a single global truth.

This varying intercept can describe a range of possible phenomena for which the rate of change remains constant, but the baseline value varies. In such *hierarchical models* we employ a concept known as *partial pooling* to extract as much forbidden knowledge from the reluctant data as possible.

A set of entirely separate models, such as the per-state set of linear regressions presented in the first post of this series, employs a *no pooling* approach: the data of each state is treated separately, with an entirely different model fit to each. This certainly considers each the uniqueness of each state, but cannot benefit from insights drawn from the broader range of data we have available, which we may reasonably assume to have some relevance.

By contrast, the global Gaussian, Poisson, and negative binomial models presented so far represent *complete pooling*, in which the entire set of data is considered a formless, protean amalgam without meaningful structure. This mindless, groping approach causes the unique features of each state to be lost amongst the anarchy and chaos.

A partial pooling approach instead builds a *global* mean intercept value across the dataset, but allows the intercept value for each individual state to deviate according to a governing probability distribution. This both accounts for the individuality of each group of observations, in our case the state, but also draws on the accumulated wisdom of the whole.

We now construct a partially-pooled varying intercept model, in which the parameters and observations for each US state in our dataset is individually indexed:

$$\begin{eqnarray}  
y &\sim& \mathbf{NegBinomial}(\mu, \phi)\\  
\log(\mu) &\sim& \alpha\_i + \beta x\\  
\alpha\_i &\sim& \mathcal{N}(\mu\_\alpha, \sigma\_\alpha)\\  
\beta &\sim& \mathcal{N}(0, 1)\\  
\phi &\sim& \mathbf{HalfCauchy}(2)  
\end{eqnarray}$$

Note that the intercept parameter, \(\alpha\), in the second line is now indexed by the state, represented here by the subscript \(i\). The slope parameter, \(\beta\), remains constant across all states.

This model can be rendered in Stan code as follows:

Show Gaussian model specification code.

population\_model\_negbinomial\_var\_intercept.stan

data {

// Number of rows (observations)

int observations;

// Number of states

int< lower=0 > states;

// Vector detailing the US state in which each observation (count of

// counts in a year) occurred

int< lower=1, upper=states > state[ observations ];

// Predictor (population of state)

vector[ observations ] population\_raw;

// Response (counts)

int counts[ observations ];

}

transformed data {

// Center and scale the predictor

vector[ observations ] population;

population = ( population\_raw - mean( population\_raw ) ) / sd( population\_raw );

}

parameters {

// Per-state intercepts

vector[ states ] a;

// Mean and SD of distribution from which per-state intercepts are drawn

real< lower=0 > mu\_a;

real< lower=0 > sigma\_a;

// Negative binomial dispersion parameter

real< lower=0 > phi;

// Slope

real b;

}

transformed parameters {

// Calculate location parameter for negative binomial incorporating

// per-state indicator.

vector[ observations ] eta;

for( i in 1:observations ) {

eta[i] = a[ state[i] ] + population[i] \* b;

}

}

model {

mu\_a ~ normal(0, 1);

sigma\_a ~ cauchy(0, 2);

// Priors

a ~ normal ( mu\_a, sigma\_a );

b ~ normal( 0, 1 );

phi ~ cauchy( 0, 2 );

// Model

counts ~ neg\_binomial\_2\_log( eta, phi );

}

generated quantities {

vector[observations] counts\_pred;

vector[observations] log\_lik;

vector[observations] mu;

mu = exp( eta );

for (n in 1:observations) {

log\_lik[n] = neg\_binomial\_2\_log\_lpmf( counts[n] | eta[n], phi );

counts\_pred[n] = neg\_binomial\_2\_log\_rng( eta[n], phi );

}

}

Once the model has twisted itself into the most appropriate form for our data, we can now compare it against our previous completely-pooled model:

Show LOO-CV comparison code.

Code snippet for comparing models via LOO-CV. Full code at the end of this post.

...

# Compare models with LOO

log\_lik\_negbinom\_var\_intercept <- extract\_log\_lik(fit\_ufo\_pop\_negbinom\_var\_intercept, merge\_chains = FALSE)

r\_eff\_negbinom\_var\_intercept <- relative\_eff(exp(log\_lik\_negbinom\_var\_intercept))

loo\_negbinom\_var\_intercept <- loo(log\_lik\_negbinom\_var\_intercept, r\_eff = r\_eff\_negbinom\_var\_intercept, cores = 2)

...

> compare( loo\_negbinom, loo\_negbinom\_var\_intercept )

elpd\_diff se

363.2 28.8

Our transcendent journey from the statistical primordial ooze continues: the varying intercept model is favoured over the completely-pooled model by a significant margin.

**Sacred Geometry**

Now that our minds have apprehended a startling glimpse of the implications of the varying intercept model, it is natural to consider taking a further terrible step and allowing both the slope and the intercept to vary[4](https://www.weirddatascience.net/index.php/2019/04/17/bayes-vs-the-invaders-part-three-the-parallax-view/#easy-footnote-bottom-4-654).

With both the intercept and slope of the underlying linear predictor varying, an additional complexity raises its head: can we safely assume that these parameters, the intercept and slope, vary independently of each other, or may there be arcane correlations between them? Do states with a higher intercept also experience a higher slope in general, or is the opposite the case? Without prior knowledge to the contrary, we must allow our model to determine these possible correlations, or we are needlessly throwing away potential information in our model.

For a varying slope and intercept model, therefore, we must now include a *correlation matrix*, \(\Omega\), between the parameters of the linear predictor for each state in our model. This correlation matrix, as with all parameters in a Bayesian framework, must be expressed with a prior distribution from which the model can begin its evaluation of the data.

$$\begin{eqnarray}  
y &\sim& \mathbf{NegBinomial}(\mu, \phi)\\  
\log(\mu) &\sim& \alpha\_i + \beta x\_i\\  
\begin{bmatrix}  
\alpha\_i\\  
\beta\_i  
\end{bmatrix} &\sim& \mathcal{N}(  
\begin{bmatrix}  
\mu\_\alpha\\  
\mu\_\beta  
\end{bmatrix}, \Omega )\\  
\Omega &\sim& \mathbf{LKJCorr}(2)\\  
\phi &\sim& \mathbf{HalfCauchy}(2)  
\end{eqnarray}$$

This model has grown and gained a somewhat twisted complexity compared with the serene austerity of our earliest linear model. Despite this, each further step in the descent has followed its own perverse logic, and the progression should clear. The corresponding Stan code follows:

Show negative binomial varying intercept and slope model specification code.

population\_model\_negbinomial\_var\_intercept\_slope.stan

data {

// Number of rows (observations)

int observations;

// Number of states

int< lower=0 > states;

// Vector detailing the US state in which each observation (count of

// counts in a year) occurred

int< lower=1, upper=states > state[ observations ];

// Predictor (population of state)

vector[ observations ] population\_raw;

// Response (counts)

int counts[ observations ];

}

transformed data {

// Center and scale the predictor

vector[ observations ] population;

population = ( population\_raw - mean( population\_raw ) ) / sd( population\_raw );

}

parameters {

// Per-state intercepts and slopes

vector[ states ] state\_intercept;

vector[ states ] state\_slope;

// Baseline intercept and slope from which each group deviates.

real pop\_intercept;

real pop\_slope;

// Per-state standard deviations for intercept and slope

vector< lower=0 >[2] state\_sigma;

// Negative binomial dispersion parameter

real< lower=0 > phi;

// Parameter correlation matrix

corr\_matrix[2] omega;

}

transformed parameters {

vector[2] vec\_intercept\_slope[ states ];

vector[2] mu\_intercept\_slope;

// Location parameter

vector[observations] eta;

// Per-state intercepts and slopes

for( i in 1:states ) {

vec\_intercept\_slope[ i, 1] = state\_intercept[i];

vec\_intercept\_slope[ i, 2] = state\_slope[i];

}

// Population slope and intercept

mu\_intercept\_slope[1] = pop\_intercept;

mu\_intercept\_slope[2] = pop\_slope;

// Calculation negbinomial location parameter

for( i in 1:observations ) {

eta[i] = state\_intercept[ state[i] ] + state\_slope[ state[i]] \* population[ i ];

}

}

model {

// Priors

omega ~ lkj\_corr(2);

phi ~ cauchy(0, 3 );

state\_sigma ~ cauchy( 0, 3 );

pop\_intercept ~ normal( 0, 1 );

pop\_slope ~ normal( 0, 1 );

vec\_intercept\_slope ~ multi\_normal( mu\_intercept\_slope, quad\_form\_diag( omega, state\_sigma ) );

// Model

counts ~ neg\_binomial\_2\_log( eta, phi );

}

generated quantities {

vector[observations] counts\_pred;

vector[observations] log\_lik;

for (n in 1:observations) {

log\_lik[n] = neg\_binomial\_2\_log\_lpmf( counts[n] | eta[n], phi );

counts\_pred[n] = neg\_binomial\_2\_log\_rng( eta[n], phi );

}

}

The ultimate test of our faith, then, is whether the added complexity of the partially-pooled varying slope, varying intercept model is justified. Once again, we turn to the ruthless judgement of the LOO-CV:

Show LOO-CV comparison code.

Code snippet for calculating the LOO-CV elpd. The full R code for building and comparing all models in this post is listed at the end.

...

log\_lik\_negbinom\_var\_intercept\_slope <- extract\_log\_lik(fit\_ufo\_pop\_negbinom\_var\_intercept\_slope, merge\_chains = FALSE)

r\_eff\_negbinom\_var\_intercept\_slope <- relative\_eff(exp(log\_lik\_negbinom\_var\_intercept\_slope))

loo\_negbinom\_var\_intercept\_slope <- loo(log\_lik\_negbinom\_var\_intercept\_slope, r\_eff = r\_eff\_negbinom\_var\_intercept\_slope, cores = 2)

...

> compare( loo\_negbinom\_var\_intercept, loo\_negbinom\_var\_intercept\_slope )

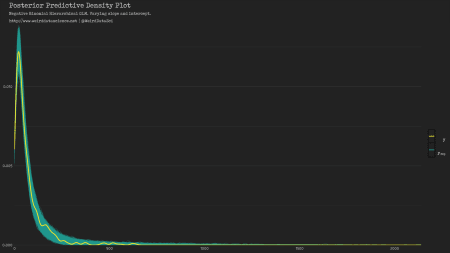
elpd\_diff se

13.3 2.4

In this final step we can see that our labours in the arcane have been rewarded. The final model is once again a significant improvement over its simpler relatives. Whilst the potential for deeper and more perfect models never ends, we will settle for now on this.

**Mortal Consequences**

With our final model built, we can now begin to examine its mortifying implications. We will leave the majority of the subjective analysis for the next, and final, post in this series. For now, however, we can reinforce our quantitative analysis with visual assessment of the posterior predictive distribution output of our final model.

[](https://i2.wp.com/www.weirddatascience.net/wp-content/uploads/2019/04/posterior_predictive.png)

Posterior predictive density plot of varying intercept, varying slope negative binomial GLM of UFO sightings.

Show posterior predictive plotting code.

bayes\_plots.r  
(Includes code to generate traceplot and posterior predictive distribution plot.)

library( tidyverse )

library( magrittr )

library( lubridate )

library( ggplot2 )

library( showtext )

library( cowplot )

library( rstan )

library( bayesplot )

library( tidybayes )

# Load UFO data

ufo\_population\_sightings <-

readRDS("work/ufo\_population\_sightings.rds")

# UFO reporting font

font\_add( "main\_font", "/usr/share/fonts/TTF/weird/Tox Typewriter.ttf")

font\_add( "bold\_font", "/usr/share/fonts/TTF/weird/Tox Typewriter.ttf")

showtext\_auto()

# Plots, posterior predictive checking, LOO.

# (Visualisations only produced for varying slope/intercept model, as a result

# of LOO checking.

# Bayesplot needs to be told which theme to use as a default.

theme\_set( theme\_weird() )

# Read the fitted model

fit\_ufo\_pop\_negbinom\_var\_intercept\_slope <-

readRDS( "work/fit\_ufo\_pop\_negbinom\_var\_intercept\_slope.rds" )

## Model checking visualisations

# Extract posterior estimates from the fit (from the generated quantities of the stan model)

counts\_pred\_negbinom\_var\_intercept\_slope <- as.matrix( fit\_ufo\_pop\_negbinom\_var\_intercept\_slope, pars = "counts\_pred" )

# First, as always, a traceplot

tp <-

traceplot(

fit\_ufo\_pop\_negbinom\_var\_intercept\_slope,

pars = c("pop\_intercept", "pop\_slope", "phi" ),

ncol=1 ) +

scale\_colour\_viridis\_d( name="Chain", direction=-1 ) +

theme\_weird()

title <-

ggdraw() +

draw\_label("Traceplot of Key Parameters", fontfamily="main\_font", colour = "#cccccc", size=20, hjust=0, vjust=1, x=0.02, y=0.88) +

draw\_label("<http://www.weirddatascience.net> | @WeirdDataSci", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=0, vjust=1, x=0.02, y=0.40)

titled\_tp <-

plot\_grid(title, tp, ncol=1, rel\_heights=c(0.1, 1)) +

theme(

panel.background = element\_rect(fill = "#222222", colour = "#222222"),

plot.background = element\_rect(fill = "#222222", colour = "#222222"),

)

save\_plot("output/traceplot.pdf",

titled\_tp,

base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )

# Posterior predictive density. (Visual representation of goodness of fit.)

gp\_ppc <-

ppc\_dens\_overlay(

y = extract2( ufo\_population\_sightings, "count" ),

yrep = counts\_pred\_negbinom\_var\_intercept\_slope ) +

theme\_weird()

title <-

ggdraw() +

draw\_label("Posterior Predictive Density Plot", fontfamily="main\_font", colour = "#cccccc", size=20, hjust=0, vjust=1, x=0.02, y=0.88) +

draw\_label("<http://www.weirddatascience.net> | @WeirdDataSci", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=0, vjust=1, x=0.02, y=0.40)

titled\_gp\_ppc <-

plot\_grid(title, gp\_ppc, ncol=1, rel\_heights=c(0.1, 1)) +

theme(

panel.background = element\_rect(fill = "#222222", colour = "#222222"),

plot.background = element\_rect(fill = "#222222", colour = "#222222"),

)

save\_plot("output/posterior\_predictive.pdf",

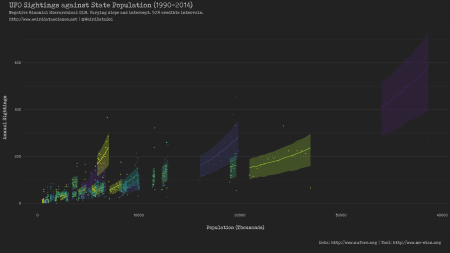
titled\_gp\_ppc,

base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )

In comparison with earlier attempts, the varying intercept and slope model visibly captures the overall shape of the distribution with terrifying ease. As our wary confidence mounts in the mindless automaton we have fashioned, we can now examine its predictive ability on our original data.

[](https://i1.wp.com/www.weirddatascience.net/wp-content/uploads/2019/04/predictive_plot.png)

Varying intercept and slope negative binomial GLM of UFO sightings against population.

Show negative binomial varying intercept and slope plot code.

population\_plot.r

library( tidyverse )

library( magrittr )

library( lubridate )

library( ggplot2 )

library( showtext )

library( cowplot )

library( rstan )

library( bayesplot )

library( tidybayes )

library( modelr )

# Load UFO data and model

ufo\_population\_sightings <-

readRDS("work/ufo\_population\_sightings.rds")

fit\_ufo\_pop\_negbinom\_var\_intercept\_slope <-

readRDS("work/fit\_ufo\_pop\_negbinom\_var\_intercept\_slope.rds")

#readRDS("work/fit\_ufo\_pop\_normal\_var\_intercept\_slope.rds")

# UFO reporting font

font\_add( "main\_font", "/usr/share/fonts/TTF/weird/Tox Typewriter.ttf")

font\_add( "bold\_font", "/usr/share/fonts/TTF/weird/Tox Typewriter.ttf")

showtext\_auto()

# Plots, posterior predictive checking, LOO

theme\_set( theme\_weird() )

## Model checking visualisations

# Extract posterior estimates from the fit (from the generated quantities of the stan model)

counts\_pred\_negbinom\_var\_intercept\_slope <- as.matrix( fit\_ufo\_pop\_negbinom\_var\_intercept\_slope, pars = "counts\_pred" )

## Create per-state predictive fit plots

# Convert fitted model (stanfit) object to a tibble

fit\_tbl <-

summary(fit\_ufo\_pop\_negbinom\_var\_intercept\_slope)$summary %>%

as.data.frame() %>%

mutate(variable = rownames(.)) %>%

select(variable, everything()) %>%

as\_tibble()

counts\_predicted <-

fit\_tbl %>%

filter( str\_detect(variable,'counts\_pred') )

ufo\_population\_sightings\_pred <-

ufo\_population\_sightings %>%

ungroup() %>%

mutate( count\_mean = counts\_predicted$mean,

lower = counts\_predicted$`25%`,

upper = counts\_predicted$`75%`)

# (Using mean and SD of fit summary)

predictive\_plot <-

ggplot( ufo\_population\_sightings\_pred ) +

geom\_point( aes( x=population, y=count, colour=state ), size=0.6, alpha=0.8 ) +

geom\_line(aes( x=population, y=count\_mean, colour=state )) +

geom\_ribbon(aes(x=population, ymin = lower, ymax = upper, fill=state), alpha = 0.25) +

labs( x="Population (Thousands)", y="Annual Sightings" ) +

scale\_fill\_viridis\_d( name="State" ) +

scale\_colour\_viridis\_d( name="State" ) +

theme(

axis.title.y = element\_text( angle=90 ),

legend.position = "none" )

# Construct full plot, with title and backdrop.

title <-

ggdraw() +

draw\_label("UFO Sightings against State Population (1990-2014)", fontfamily="main\_font", colour = "#cccccc", size=20, hjust=0, vjust=1, x=0.02, y=0.88) +

draw\_label("Negative Binomial Hierarchical GLM. Varying slope and intercept. 50% credible intervals.", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=0, vjust=1, x=0.02, y=0.48) +

draw\_label("<http://www.weirddatascience.net> | @WeirdDataSci", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=0, vjust=1, x=0.02, y=0.16)

data\_label <- ggdraw() +

draw\_label("Data: <http://www.nuforc.org> | Tool: <http://www.mc-stan.org>", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=1, x=0.98 )

predictive\_plot\_titled <-

plot\_grid(title, predictive\_plot, data\_label, ncol=1, rel\_heights=c(0.1, 1, 0.1)) +

theme(

panel.background = element\_rect(fill = "#222222", colour = "#222222"),

plot.background = element\_rect(fill = "#222222", colour = "#222222"),

)

save\_plot("output/predictive\_plot.pdf",

predictive\_plot\_titled,

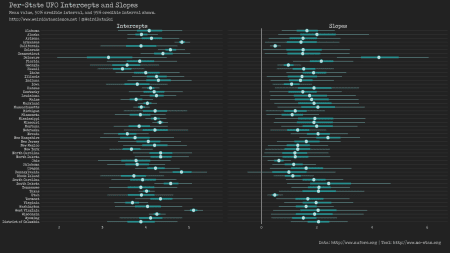
base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )

The purpose of our endeavours is to show whether or not the frequency of extraterrestrial visitations is merely a sad reflection of the number of unsuspecting humans living in each state. After seemingly endless cryptic calculations, our statistical machinery implies that there are deeper mysteries here: allowing the relationship between sightings and the underlying linear predictors to vary by state more perfectly predicts the data. There are clearly other, hidden, factors in play.

More than that, however, our final model allows us to quantify these differences. We can now retrieve from the very bowels of our inferential process the per-state distribution of paremeters for both the slope and intercept of the linear predictor.

[](https://i1.wp.com/www.weirddatascience.net/wp-content/uploads/2019/04/ufo_per-state_intercepts-slopes.png)

Varying slope and intercept negative binomial GLM parameter plot for UFO sightings model.

Show per-state intercept and slope plotting code.

slope\_intercept\_plot.r

library( tidyverse )

library( magrittr )

library( lubridate )

library( ggplot2 )

library( showtext )

library( cowplot )

library( rstan )

library( bayesplot )

library( tidybayes )

library( modelr )

# Load UFO data and model

ufo\_population\_sightings <-

readRDS("work/ufo\_population\_sightings.rds")

fit\_ufo\_pop\_negbinom\_var\_intercept\_slope <-

readRDS("work/fit\_ufo\_pop\_negbinom\_var\_intercept\_slope.rds")

#readRDS("work/fit\_ufo\_pop\_normal\_var\_intercept\_slope.rds")

# UFO reporting font

font\_add( "main\_font", "/usr/share/fonts/TTF/weird/Tox Typewriter.ttf")

font\_add( "bold\_font", "/usr/share/fonts/TTF/weird/Tox Typewriter.ttf")

showtext\_auto()

# Plots, posterior predictive checking, LOO

theme\_set( theme\_weird() )

# Use teal colour scheme

color\_scheme\_set( "teal")

## Model checking visualisations

# Extract posterior estimates from the fit (from the generated quantities of the stan model)

counts\_pred\_negbinom\_var\_intercept\_slope <- as.matrix( fit\_ufo\_pop\_negbinom\_var\_intercept\_slope, pars = "counts\_pred" )

# US state data

us\_state\_factors <-

levels( factor( ufo\_population\_sightings$state ) )

# US state names for nice plotting

# Data:

state\_code\_data <-

read\_csv( file="data/us\_states.csv" ) %>%

filter( code %in% us\_state\_factors )

# Rename variables back to state names

posterior\_intercepts <-

as.data.frame( fit\_ufo\_pop\_negbinom\_var\_intercept\_slope ) %>%

as\_tibble %>%

select(starts\_with('state\_intercept') ) %>%

rename\_all( ~us\_state\_factors ) %>%

rename\_all( ~extract2( state\_code\_data, "us\_state" ) )

# Rename variables back to state names

posterior\_slopes <-

as.data.frame( fit\_ufo\_pop\_negbinom\_var\_intercept\_slope ) %>%

as\_tibble %>%

select(starts\_with('state\_slope') ) %>%

rename\_all( ~us\_state\_factors ) %>%

rename\_all( ~extract2( state\_code\_data, "us\_state" ) )

# Posterior draws combined

posterior\_slopes\_long <-

posterior\_slopes %>%

gather( value = "slope" )

posterior\_intercepts\_long <-

posterior\_intercepts %>%

gather( value = "intercept" )

posterior\_draws\_long <-

bind\_cols( posterior\_intercepts\_long, posterior\_slopes\_long ) %>%

select( -key1 ) %>%

transmute( state = key, intercept, slope )

# Interval plots (slope and intervals)

# Plot intercept parameters for varying intercept and slope model

gp\_intercept <-

mcmc\_intervals( posterior\_intercepts ) +

ggtitle( "Intercepts" ) +

theme\_weird()

# Plot slope parameters for varying intercept and slope model

# (Remove y-axis labels as this will be aligned with the intercept plot.)

gp\_slope <-

mcmc\_intervals( posterior\_slopes ) +

ggtitle( "Slopes" ) +

theme\_weird() +

theme(

axis.text.y = element\_blank()

)

gp\_slope\_intercept <-

plot\_grid( gp\_intercept, gp\_slope, ncol=2 )

# Construct full plot, with title and backdrop.

title <-

ggdraw() +

draw\_label("Per-State UFO Intercepts and Slopes", fontfamily="main\_font", colour = "#cccccc", size=20, hjust=0, vjust=1, x=0.02, y=0.88) +

draw\_label("Mean value, 50% credible interval, and 95% credible interval shown.", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=0, vjust=1, x=0.02, y=0.48) +

draw\_label("<http://www.weirddatascience.net> | @WeirdDataSci", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=0, vjust=1, x=0.02, y=0.16)

data\_label <- ggdraw() +

draw\_label("Data: <http://www.nuforc.org> | Tool: <http://www.mc-stan.org>", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=1, x=0.98 )

data\_label <- ggdraw() +

draw\_label("Data: <http://www.nuforc.org> | Tool: <http://www.mc-stan.org>", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=1, x=0.98 )

data\_label <- ggdraw() +

draw\_label("Data: <http://www.nuforc.org> | Tool: <http://www.mc-stan.org>", fontfamily="main\_font", colour = "#cccccc", size=12, hjust=1, x=0.98 )

gp\_slope\_intercept\_titled <-

plot\_grid(title, gp\_slope\_intercept, data\_label, ncol=1, rel\_heights=c(0.1, 1, 0.1)) +

theme( panel.background = element\_rect(fill = "#222222", colour = "#222222"),

plot.background = element\_rect(fill = "#222222", colour = "#222222"))

save\_plot("output/ufo\_per-state\_intercepts-slopes.pdf",

gp\_slope\_intercept\_titled,

base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )

It is important to note that, while we are still referring to the \(\alpha\) and \(\beta\) parameters as the slope and intercept, their interpretation is more complex in a generalised linear model with a \(\log\) link function than in the simple linear model. For now, however, this diagram is sufficient to show that the horror visited on innocent lives by our interstellar visitors is not purely arbitrary, but depends at least in part on geographical location.

With this malign inferential process finally complete we will turn, in the next post, to a trembling interpretation of the model and its dark implications for our collective future.

**Model Fitting and Comparison Code Listing**

Show full model fitting and LOO-CV comparison code.

This code fits the range of models developed in this series, relying on the individual Stan source code files, and runs the LOO-CV comparisons discussed in this post.

population\_model.r

library( tidyverse )

library( magrittr )

library( ggplot2 )

library( showtext )

library( rstan )

library( tidybayes )

library( loo )

# Load UFO data

ufo\_population\_sightings <-

readRDS("work/ufo\_population\_sightings.rds")

## Simple Models

## Complete Pooling -- all states considered identical.

# Fit model of UFO sightings (Normal)

if( not( file.exists( "work/fit\_ufo\_pop\_normal.rds" ) ) ) {

message("Fitting basic Normal model.")

fit\_ufo\_pop\_normal <-

stan( file="model/population\_model\_normal.stan",

data=list(

observations = nrow( ufo\_population\_sightings ),

population = extract2( ufo\_population\_sightings, "population" ),

counts = extract2( ufo\_population\_sightings, "count" )

)

)

saveRDS( fit\_ufo\_pop\_normal, "work/fit\_ufo\_pop\_normal.rds" )

message("Basic Normal model fitted.")

} else {

fit\_ufo\_pop\_normal <- readRDS( "work/fit\_ufo\_pop\_normal.rds" )

}

# Fit model of UFO sightings (Poisson)

if( not( file.exists( "work/fit\_ufo\_pop\_poisson.rds" ) ) ) {

message("Fitting basic Poisson model.")

fit\_ufo\_pop\_poisson <-

stan( file="model/population\_model\_poisson.stan",

data=list(

observations = nrow( ufo\_population\_sightings ),

population\_raw = extract2( ufo\_population\_sightings, "population" ),

counts = extract2( ufo\_population\_sightings, "count" )

)

)

saveRDS( fit\_ufo\_pop\_poisson, "work/fit\_ufo\_pop\_poisson.rds" )

message("Basic Poisson model fitted.")

} else {

fit\_ufo\_pop\_poisson <- readRDS( "work/fit\_ufo\_pop\_poisson.rds" )

}

# Fit model of UFO sightings (Negative Binomial)

if( not( file.exists( "work/fit\_ufo\_pop\_negbinom.rds" ) ) ) {

message("Fitting basic negative binomial model.")

fit\_ufo\_pop\_negbinom <-

stan( file="model/population\_model\_negbinomial.stan",

data=list(

observations = nrow( ufo\_population\_sightings ),

population\_raw = extract2( ufo\_population\_sightings, "population" ),

counts = extract2( ufo\_population\_sightings, "count" ) )

)

saveRDS( fit\_ufo\_pop\_negbinom, "work/fit\_ufo\_pop\_negbinom.rds" )

message("Basic negative binomial model fitted.")

} else {

fit\_ufo\_pop\_negbinom <- readRDS( "work/fit\_ufo\_pop\_negbinom.rds" )

}

## Multilevel Models

## Partial Pooling (Varying Intercept)

if( not( file.exists( "work/fit\_ufo\_pop\_negbinom\_var\_intercept.rds" ) ) ) {

message("Fitting varying intercept negative binomial model.")

fit\_ufo\_pop\_negbinom\_var\_intercept <-

stan( file="model/population\_model\_negbinomial\_var\_intercept.stan",

data=list(

observations = nrow( ufo\_population\_sightings ),

population\_raw = extract2( ufo\_population\_sightings, "population" ),

counts = extract2( ufo\_population\_sightings, "count" ),

states = length( unique( ufo\_population\_sightings$state ) ),

state = as.numeric( factor( ufo\_population\_sightings$state ) )

),

chains=4, iter=2000,

control = list(max\_treedepth = 15, adapt\_delta=0.9)

)

saveRDS( fit\_ufo\_pop\_negbinom\_var\_intercept, "work/fit\_ufo\_pop\_negbinom\_var\_intercept.rds" )

message("Varying intercept negative binomial model fitted.")

} else {

fit\_ufo\_pop\_negbinom\_var\_intercept <- readRDS( "work/fit\_ufo\_pop\_negbinom\_var\_intercept.rds" )

}

### Partial Pooling (Varying intercept and slope.)

if( not( file.exists( "work/fit\_ufo\_pop\_negbinom\_var\_intercept\_slope.rds" ) ) ) {

message("Fitting varying intercept and slope negative binomial model")

fit\_ufo\_pop\_negbinom\_var\_intercept\_slope <-

stan( file="model/population\_model\_negbinomial\_var\_intercept\_slope.stan",

data=list(

observations = nrow( ufo\_population\_sightings ),

population\_raw = extract2( ufo\_population\_sightings, "population" ),

counts = extract2( ufo\_population\_sightings, "count" ),

states = length( unique( ufo\_population\_sightings$state ) ),

state = as.numeric( factor( ufo\_population\_sightings$state ) )

),

chains=4, iter=2000,

control = list(max\_treedepth = 12, adapt\_delta=0.8)

)

saveRDS( fit\_ufo\_pop\_negbinom\_var\_intercept\_slope, "work/fit\_ufo\_pop\_negbinom\_var\_intercept\_slope.rds" )

message("Varying intercept and slope negative binomial model fitted.")

} else {

fit\_ufo\_pop\_negbinom\_var\_intercept\_slope <- readRDS( "work/fit\_ufo\_pop\_negbinom\_var\_intercept\_slope.rds" )

}

# Hierarchical normal. (Linear regression)

if( not( file.exists( "work/fit\_ufo\_pop\_normal\_var\_intercept\_slope.rds" ) ) ) {

message("Fitting varying intercept and slope normal model")

fit\_ufo\_pop\_normal\_var\_intercept\_slope <-

stan( file="model/population\_model\_normal\_var\_intercept\_slope.stan",

data=list(

observations = nrow( ufo\_population\_sightings ),

population\_raw = extract2( ufo\_population\_sightings, "population" ),

counts = extract2( ufo\_population\_sightings, "count" ),

states = length( unique( ufo\_population\_sightings$state ) ),

state = as.numeric( factor( ufo\_population\_sightings$state ) )

),

chains=4, iter=2000,

control = list(max\_treedepth = 15, adapt\_delta=0.9)

)

saveRDS( fit\_ufo\_pop\_normal\_var\_intercept\_slope, "work/fit\_ufo\_pop\_normal\_var\_intercept\_slope.rds" )

message("Varying intercept and slope normal model fitted.")

} else {

fit\_ufo\_pop\_normal\_var\_intercept\_slope <- readRDS( "work/fit\_ufo\_pop\_normal\_var\_intercept\_slope.rds" )

}

## Notify by text

message("All models fit.")

# Compare models with LOO

log\_lik\_normal <- extract\_log\_lik(fit\_ufo\_pop\_normal, merge\_chains = FALSE)

r\_eff\_normal <- relative\_eff(exp(log\_lik\_normal))

loo\_normal <- loo(log\_lik\_normal, r\_eff = r\_eff\_normal, cores = 2)

log\_lik\_poisson <- extract\_log\_lik(fit\_ufo\_pop\_poisson, merge\_chains = FALSE)

r\_eff\_poisson <- relative\_eff(exp(log\_lik\_poisson))

loo\_poisson <- loo(log\_lik\_poisson, r\_eff = r\_eff\_poisson, cores = 2)

log\_lik\_negbinom <- extract\_log\_lik(fit\_ufo\_pop\_negbinom, merge\_chains = FALSE)

r\_eff\_negbinom <- relative\_eff(exp(log\_lik\_negbinom))

loo\_negbinom <- loo(log\_lik\_negbinom, r\_eff = r\_eff\_negbinom, cores = 2)# Compare models with LOO

log\_lik\_negbinom\_var\_intercept <- extract\_log\_lik(fit\_ufo\_pop\_negbinom\_var\_intercept, merge\_chains = FALSE)

r\_eff\_negbinom\_var\_intercept <- relative\_eff(exp(log\_lik\_negbinom\_var\_intercept))

loo\_negbinom\_var\_intercept <- loo(log\_lik\_negbinom\_var\_intercept, r\_eff = r\_eff\_negbinom\_var\_intercept, save\_psis = TRUE)

log\_lik\_negbinom\_var\_intercept\_slope <- extract\_log\_lik(fit\_ufo\_pop\_negbinom\_var\_intercept\_slope, merge\_chains = FALSE)

r\_eff\_negbinom\_var\_intercept\_slope <- relative\_eff(exp(log\_lik\_negbinom\_var\_intercept\_slope))

loo\_negbinom\_var\_intercept\_slope <- loo(log\_lik\_negbinom\_var\_intercept\_slope, r\_eff = r\_eff\_negbinom\_var\_intercept\_slope, save\_psis = TRUE)

normal\_poisson\_comparison <- compare( loo\_normal, loo\_poisson )

poiss\_negbinom\_comparison <- compare( loo\_poisson, loo\_negbinom )

negbinom\_negbinom\_var\_intercept\_comparison <- compare( loo\_negbinom, loo\_negbinom\_var\_intercept )

negbinom\_var\_intercept\_negbinom\_var\_intercept\_slope\_comparison <- compare( loo\_negbinom\_var\_intercept, loo\_negbinom\_var\_intercept\_slope )

saveRDS( normal\_poisson\_comparison, "work/normal\_poisson\_comparison.rds" )

saveRDS( poiss\_negbinom\_comparison, "work/poiss\_negbinom\_comparison.rds" )

saveRDS( negbinom\_negbinom\_var\_intercept\_comparison, "work/negbinom\_negbinom\_var\_intercept\_comparison.rds" )

saveRDS( negbinom\_var\_intercept\_negbinom\_var\_intercept\_slope\_comparison, "work/negbinom\_var\_intercept\_negbinom\_var\_intercept\_slope\_comparison.rds" )