**Crossing the Line**

This post continues our series on developing statistical models to explore the arcane relationship between UFO sightings and population. The previous post is available here: [Bayes vs. the Invaders! Part One: The 37th Parallel](http://www.weirddatascience.net/index.php/2019/04/03/bayes-vs-the-invaders-part-one-the-37th-parallel/).

The simple linear model developed in the previous post is far from satisfying. It makes many unsupportable assumptions about the data and the form of the residual errors from the model. Most obviously, it relies on an underlying Gaussian (or *normal*) distribution for its understanding of the data. For our count data, some basic features of the Guassian are inappropriate.

Most notably:

* a Gaussian distribution is [continuous](https://en.wikipedia.org/wiki/Probability_distribution#Continuous_probability_distribution) whilst counts are [discrete](https://en.wikipedia.org/wiki/Probability_distribution#Discrete_probability_distribution) — you can’t have 2.3 UFO sightings in a given day;
* the Gaussian can produce negative values, which are impossible when dealing with counts — you can’t have a negative number of UFO sightings;
* the Gaussian is symmetrical around its mean value whereas count data is typically *skewed*.

Moving from the safety and comfort of basic *linear regression*, then, we will delve into the madness and chaos of *generalized linear models* that allow us to choose from a range of distributions to describe the relationship between state population and counts of UFO sightings.

**Basic Models**

We will be working in a [Bayesian](https://en.wikipedia.org/wiki/Bayesian_inference) framework, in which we assign a *prior distribution* to each parameter that allows, and requires, us to express some *prior knowledge* about the parameters of interest. These priors are the initial starting points for parameters Afrom which the model moves towards the underlying values as it learns from the data. Choice of priors can have significant effects not only on the outputs of the model, but also its ability to function effectively; as such, it is both an important, but also arcane and subtle, aspect of the Bayesian approach[1](http://www.weirddatascience.net/index.php/2019/04/08/bayes-vs-the-invaders-part-two-abnormal-distributions/#easy-footnote-bottom-1-559).

Practically speaking, a simple linear regression can be expressed in the following form:

$$y \sim \mathcal{N}(\mu, \sigma)$$

(Read as “\(y\) *is drawn from* a normal distribution with mean \(\mu\) and standard deviation \(\sigma\)”).

In the the above expression the model relies on a Gaussian, or *normal* *likelihood* (\(\mathcal{N}\)) to describe the data — making assertions regarding how we believe the underlying data was generated. The Gaussian distribution is parameterised by a *location parameter* (\(\mu\)) and a standard deviation (\(\sigma\)).

If we were uninterested in prediction, we could describe the *shape* of the distribution of counts (\(y\)) without a predictor variable. In this approach, we could specify our model by providing *priors* for \(\mu\) and \(\sigma\) that express a level of belief in their likely values:

$$\begin{eqnarray}  
y &\sim& \mathcal{N}(\mu, \sigma) \\  
\mu &\sim& \mathcal{N}(0, 1) \\  
\sigma &\sim& \mathbf{HalfCauchy}(2)  
\end{eqnarray}$$

This provides an initial belief as to the likely shape of the data that informs, via arcane computational procedures, the model of how the observed data approaches the underlying truth[2](http://www.weirddatascience.net/index.php/2019/04/08/bayes-vs-the-invaders-part-two-abnormal-distributions/#easy-footnote-bottom-2-559).

This model is less than interesting, however. It simply defines a range of possible Gaussian distributions without unveiling the horror of the underlying relationships between unsuspecting terrestrial inhabitants and anomalous events.

To construct such a model, relating a *predictor* to a *response*, we express those relationships as follows:

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

In this model, the parameters of the likelihood are now probability distributions themselves. From a traditional linear model, we now have an *intercept* (\(\alpha\)), and a *slope* (\(\beta\)) that relates the change in the predictor variable (\(x\)) to the change in the response. Each of these [*hyperparameters*](https://en.wikipedia.org/wiki/Hyperparameter) is fitted according to the observed dataset.

**A New Model**

We can now break free from the bonds of pure linear regression and consider other distributions that more naturally describe data of the form that we are considering. The awful power of GLMs is that they can use an underlying linear model, such \(\alpha + \beta x\), as parameters to a range of likelihoods beyond the Guassian. This allows the natural description of a vast and esoteric menagerie of possible data.

For count data the default likelihood is the [Poisson](https://en.wikipedia.org/wiki/Poisson_distribution) distribution, whose sole parameter is the *arrival rate* (\(\lambda\)). While somewhat restricted, as we will see, we can begin our descent into madness by fitting a Poisson-based model to our observed data.

**Stan**

To fit a model, we will use the [Stan](http://mc-stan.org/) probabilistic programming language. Stan allows us to write a program defining a stastical model which can then be fit to the data using [Markov-Chain Monte Carlo](https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo) (MCMC) methods. In effect, at a very abstract level, this approach uses a random sampling to discover the values of the parameters that best fit the observed data[3](http://www.weirddatascience.net/index.php/2019/04/08/bayes-vs-the-invaders-part-two-abnormal-distributions/#easy-footnote-bottom-3-559).

Stan lets us specify models in the form given above, along with ways to pass in and define the nature and form of the data. This code can then be called from R using the rstan package.

In this, and subsequent posts, we will be using Stan code directly as both a learning and explanatory exercise. In typical usage, however it is often more convenient to use one of two excellent R packages [brms](https://github.com/paul-buerkner/brms) or [rstanarm](https://github.com/stan-dev/rstanarm) that allow for more compact and convenient specification of models, with well-specified raw Stan code generated automatically.

**De Profundis**

In seeking to take our first steps beyond the [placid island of ignorance](http://www.hplovecraft.com/writings/texts/fiction/cc.aspx) of the Gaussian, the Poisson distribution is a first step for assessing count data. Adapting the Gaussian model above, we can propose a predictive model for the entire population of states as follows:

$$\begin{eqnarray}  
y &\sim& \mathbf{Poisson}(\lambda) \\  
\lambda &\sim& \alpha + \beta x \\  
\alpha &\sim& \mathcal{N}(0, 5) \\  
\beta &\sim& \mathcal{N}(0, 5)  
\end{eqnarray}$$

The sole parameter of the Poisson is the *arrival rate* (\(\lambda\)) that we construct here from a population-wide intercept (\(\alpha\)) and slope (\(\beta\)).

The Stan code for the above model, and associated R code to run it, is below:

Show model specification and execution code.

population\_model\_poisson.stan

data {

// Number of rows (observations)

int observations;

// Vector (real numbers) of predictor values (population of state)

vector[ observations ] population;

// Array (integer) of responses (counts)

// Cannot be less than 0

int counts[observations];

}

parameters {

// Intercept

real< lower=0 > a;

// Slope

real< lower=0 > b;

}

model {

// Priors for a and b

a ~ normal( 0, 5 );

b ~ normal( 0, 5 );

// Model linking counts to sightings

counts ~ poisson( a + population \* b );

}

generated quantities {

// Posterior predictions

vector[observations] counts\_pred;

// Log likelihood (for LOO model checking)

vector[observations] log\_lik;

// Generate a set of log likelihoods and predicted values

// for posterior analysis and model checking.

for (n in 1:observations) {

log\_lik[n] = poisson\_lpmf( counts[n] | a + population[n]\*b );

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

}

}

population\_model\_poisson.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")

# Fit model of UFO sightings (Poisson)

# As this is computationally expensive, the fitted model will be

# saved to disk, and the process only run if the saved model file

# does not already exist.

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

message("Fitting basic Poisson model.")

sms\_notify("Fitting basic Poisson model.")

fit\_ufo\_pop\_poisson <-

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

data=list(

observations = nrow( ufo\_population\_sightings ),

population = 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" )

}

With this model encoded and fit, we can now peel back the layers of the procedure to see the extent to which it has endured the horror of our data.

The MCMC algorithm that underpins Stan — specifically [Hamiltonian Monte Carlo](https://en.wikipedia.org/wiki/Hamiltonian_Monte_Carlo) (HMC) using the [No U-Turn Sampler](http://www.stat.columbia.edu/~gelman/research/unpublished/nuts.pdf) (NUTS) — attempts to find an island of stability in the space of possibilities that corresponds to the best fit to the observed data. To do so, the algorithm spawns a set of [Markov chains](https://en.wikipedia.org/wiki/Markov_chain) that explore the parameter space. If the model is appropriate, and the data coherent, the set of Markov chains end up *converging* to exploring a similar, small set of possible states.

**Validation**

When modelling via this approach, a first check of the model’s chances of having fit correctly is to examine the so-called ‘traceplot’ that shows how well the separate Markov chains ‘mix’ — that is, converge to exploring the same area of the space[4](http://www.weirddatascience.net/index.php/2019/04/08/bayes-vs-the-invaders-part-two-abnormal-distributions/). For the Poisson model above, the traceplot can be created using the bayesplot library:

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

Traceplot of Markov chains from Poisson model fitting. ([PDF Version](http://www.weirddatascience.net/wp-content/uploads/2019/04/poisson_traceplot.pdf))

Show traceplot code.

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()

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

theme\_set( theme\_weird() )

# Read the fitted model

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

# First, as always, a traceplot

tp <-

traceplot(

fit\_ufo\_pop\_poisson,

pars = c("a", "b"),

ncol=1 ) +

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

theme\_weird()

title <-

ggdraw() +

draw\_label("Traceplot of Key Model 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/poisson\_traceplot.pdf",

titled\_tp,

base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )

These traceplots exhibit the characteristic insane scribbling of well-mixed chains, often referred to — in, of course, hushed whispers — as a manifestation of a [hairy caterpillar](https://druedin.com/2016/12/26/that-hairy-caterpillar/); the separate lines representing each chain are clearly overlapping and exploring the same areas. If, by contrast, the lines were largely separated or did not show the same space, there would be reason to believe that our model had become lost and unable to find a coherent voice amongst the myriad babbling murmurs of the data.

A second check on the sanity of the modelling process is to examine the output of the model itself to show the value of the fitted parameters of interest, and some diagnostic information:

fit\_ufo\_pop\_poisson %>%

summary(pars=c("a", "b" )) %>%

extract2( "summary" )

mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff Rhat

a 5.5199115 7.162684e-03 2.701118e-01 4.99950988 5.33593127 5.51805161 5.70399617 6.05461185 1422.118 1.001581

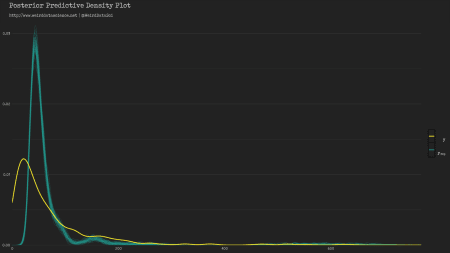
b 0.0107647 1.728273e-06 6.574476e-05 0.01063562 0.01072192 0.01076418 0.01080788 0.01089496 1447.097 1.001551

For assessment of successful model fit, the [Rhat](https://github.com/stan-dev/stan/wiki/Stan-Best-Practices) value represents the extent to which the various Markov chains exploring the space, of which there are four by default in Stan, are consistent with each other. As a rule of thumb, a value of Rhat > 1.1 indicates that the model has not converged appropriately and may require a longer set of random sampling iterations, or an improved model. Here, the valuesof Rhat are close to the ideal value of 1.

As a final step, we should examine how well our model can reproduce the shape of the original data. Models aim to be eerily lifelike parodies of the truth; in a Bayesian framework, and in the Stan language, we can build into the model the ability to draw random samples from the *posterior predictive distribution* — the set of parameters that the model has learnt from the data — to create new possible values of the outcomes based on the observed inputs. This process can be repeated many times to produce a multiplicity of possible outcomes drawn from model, which we can then visualize to see graphically how well our model fits the observed data.

In the Stan code above, this is created in the generated\_quantities block. When using more convenient libraries such as brms or rstanarm, draws from the posterior predictive distribution can be obtained more simply after the model has been fit through a range of helper functions. Here, we undertake the process manually.

We can see, then, how well the Poisson distribution, informed by our selection of priors, has shaped itself to the underlying data.

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

Posterior predictive density plot of fitted Poisson model. ([PDF Version](http://www.weirddatascience.net/wp-content/uploads/2019/04/poisson_posterior_predictive.pdf))

Show posterior predictive plot code.

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.

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

theme\_set( theme\_weird() )

# Read the fitted model

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

## Model checking visualisations

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

counts\_pred\_poisson <- as.matrix( fit\_ufo\_pop\_poisson, pars = "counts\_pred" )

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

# Sample 50 rows for overlay

counts\_pred\_sample <-

counts\_pred\_poisson[ sample( nrow( counts\_pred\_poisson ), 50 ), ]

gp\_ppc <-

ppc\_dens\_overlay(

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

yrep = counts\_pred\_sample,

alpha=0.4) +

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/poisson\_posterior\_predictive.pdf",

titled\_gp\_ppc,

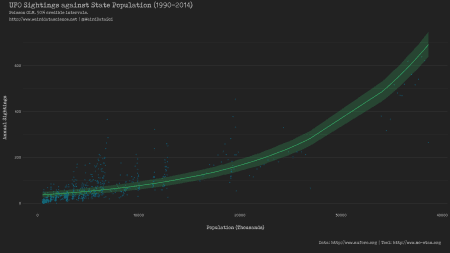
base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )

In the diagram above, the yellow line shows the densities of count values; the blue lines show a sample of twisted mockeries spawned by our [piscine](https://www.collinsdictionary.com/dictionary/french-english/poisson) approximations. As can be seen, the model has roughly captured the shape of the true distribution, but has notable dissimilarities with the original data.

To appreciate the full horror of what we have wrought, of course, we can plot the predictions of the model against the real data.

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

Global poisson GLM of UFO sightings against population. ([PDF Version](http://www.weirddatascience.net/wp-content/uploads/2019/04/poisson_predictive_plot.pdf))

Show posterior predictive plot code.

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\_poisson <-

readRDS("work/fit\_ufo\_pop\_poisson.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\_poisson <- as.matrix( fit\_ufo\_pop\_poisson, pars = "counts\_pred" )

# US state data

us\_state\_factors <-

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

## Create per-state predictive fit plots

# Convert fitted model (stanfit) object to a tibble

fit\_tbl <-

summary(fit\_ufo\_pop\_poisson)$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$`2.5%`,

upper = counts\_predicted$`97.5%`)

# (Using mean and SD of fit summary)

predictive\_plot <-

ggplot( ufo\_population\_sightings\_pred ) +

geom\_point( aes( x=population, y=count ), colour="#0b6788", size=0.6, alpha=0.8 ) +

geom\_line(aes( x=population, y=count\_mean ), colour="#3cd070" ) +

geom\_ribbon(aes(x=population, ymin = lower, ymax = upper ), alpha = 0.2, fill="#3cd070") +

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("Poisson GLM. 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/poisson\_predictive\_plot.pdf",

predictive\_plot\_titled,

base\_width = 16,

base\_height = 9,

base\_aspect\_ratio = 1.78 )