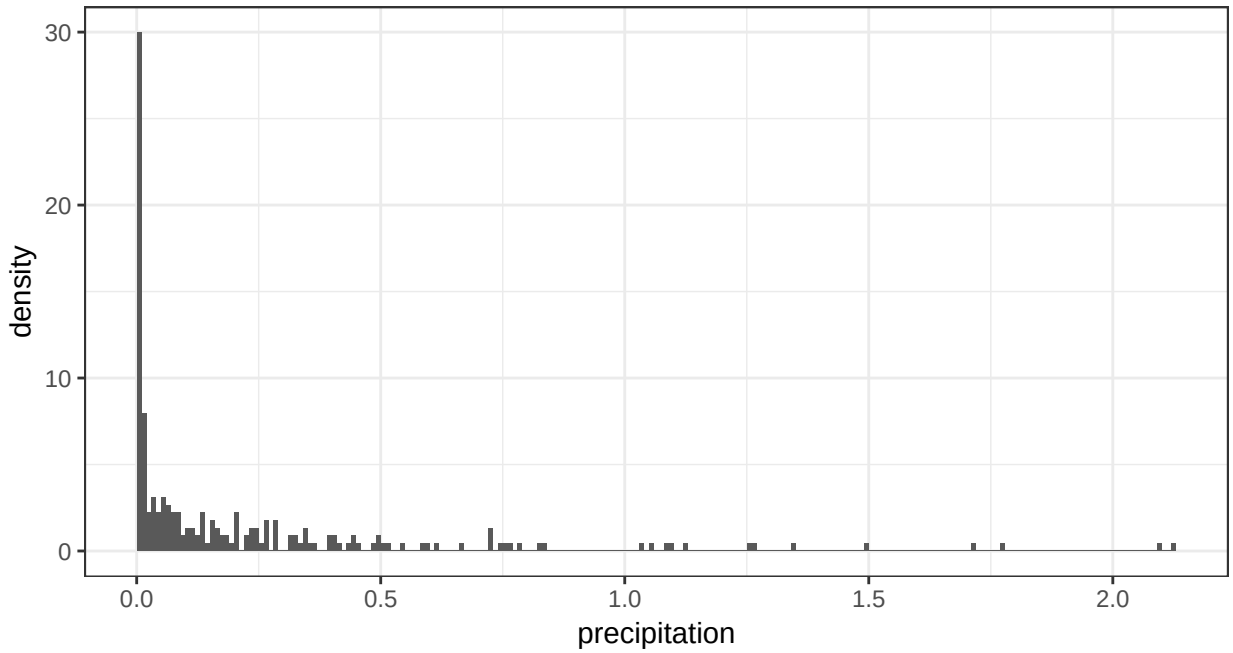
Stat 343: Numerically Maximizing the Likelihood via Stan

Rainfall in Illinois, all storms from 1960 - 1964

The code below reads in the data and makes an initial plot:



- Let's model these data with a Gamma distribution and estimate the parameters by maximum likelihood.
- We saw before that we will need to use some sort of numerical method to maximize the likelihood in order to obtain parameter estimates.

Overview of Stan

- Stan is a programming language that can be used to specify statistical models
- Most often, used for parameter estimation in Bayesian framework (coming up next!)
- Also provides methods for maximizing the (log)-likelihood
  - Newton's method
  - BFGS and L-BFGS
    - basically Newton's method, but don't require explicit calculation of the Hessian matrix of the log-likelihood
    - L-BFGS requires less memory to run, and is the default method for optimization in Stan
- Stan code is compiled to C++ code, which is in turn compiled to an executable file
- There are interfaces to interact with Stan from R, Python, MATLAB, Julia, Stata, and the command line

Model Definition in Stan

- The following code is stored in a separate file called `gamma_model.stan`.
- It defines the **data** that will be used for parameter estimation, the model **parameter**s, and the statistical **model**.

```
data {
  // sample size; a non-negative integer
  int<lower=0> n;

  // vector of n observations; rainfall amounts
  vector[n] x;
}

parameters {
  // the shape parameter for the Gamma distribution; a non-negative real number
  real<lower=0> alpha;

  // the scale parameter for the Gamma distribution; a non-negative real number
  real<lower=0> beta;
}

model {
  // each element of the vector x is modeled as following a Gamma(alpha, beta) distribution
  x ~ gamma(alpha, beta);
}
```

Performing optimization by calling Stan from R

We need to do the following tasks:

1. Read in our data
2. Set up a list with the "data" Stan needs to know about
3. Compile the Stan model definition to an executable
4. Call Stan do do the optimization
5. Extract the parameter estimates

All of the below (continuing on next page) is R code that is in a .Rmd or .R file.

```r
# Load the rstan package
library(rstan)

# Read in data
il_storms <- bind_rows(
  read_csv("http://www.evanlray.com/data/rice/Chapter%2010/illinois60.txt", col_names = FALSE),
  read_csv("http://www.evanlray.com/data/rice/Chapter%2010/illinois61.txt", col_names = FALSE),
  read_csv("http://www.evanlray.com/data/rice/Chapter%2010/illinois62.txt", col_names = FALSE),
  read_csv("http://www.evanlray.com/data/rice/Chapter%2010/illinois63.txt", col_names = FALSE),
  read_csv("http://www.evanlray.com/data/rice/Chapter%2010/illinois64.txt", col_names = FALSE)
)
names(il_storms) <- "precipitation"

# Set up list with data Stan will need to know about
stan_data <- list(
  n = nrow(il_storms),
  x = il_storms$precipitation
)
```

```
# Compile the Stan model definition to an executable.  Takes a few seconds to run.
gamma_model_compiled <- stan_model(file = "gamma_model.stan")

# Call Stan to do optimization
gamma_fit <- optimizing(gamma_model_compiled,
  data = stan_data,
  seed = 8742,
  init = "random"
)

# Here's a look at the return object, which is a list with 3 components:
#  * par is a named vector of parameter estimates
#  * value is the value of the log-likelihood at the maximum,
#    after dropping constants that don't involve the parameters
#  * return_code is 0 if everything went well in the optimization procedure,
#    otherwise an error code to be sad about
gamma_fit
```

```
## $par
##  alpha   beta
## 0.4408 1.9643
##
## $value
## [1] 185.3
##
## $return_code
## [1] 0
```

```
# Let's make a plot!
ggplot(data = il_storms, mapping = aes(x = precipitation)) +
  geom_histogram(center = 0.005, binwidth = 0.01, mapping = aes(y = ..density..)) +
  stat_function(fun = dgamma,
    args = list(shape = gamma_fit$par["alpha"], rate = gamma_fit$par["beta"]),
    color = "orange") +
  theme_bw()
```