Stat 343 Bayes Practice with Stan

Earthquakes

This example is taken from Chihara and Hesterberg. Here's a quote from them:

> "The Weibull distribution has been used to model the time between successive earthquakes (Hasumi et al (2009); Tiampo et al. (2008)). The data set `quakes` contains the time between earthquakes (in days) for all earthquakes of magnitude 6 or greater from 1970 through 2009 (from http://earthquake.usgs.gov/earthquakes/eqarchives/)."

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

```
library(tidyverse)
```

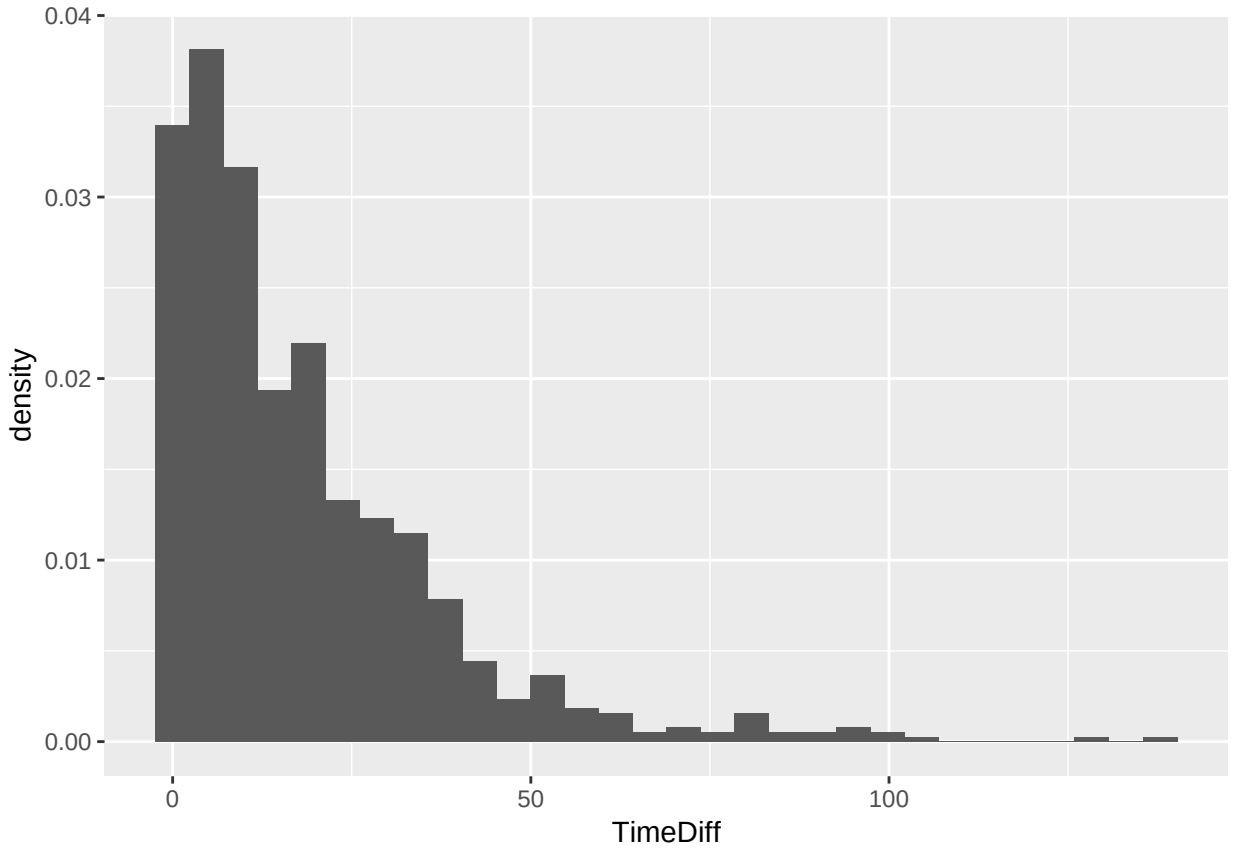
```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.2
```

```
library(rstan)
```

```
## Warning: package 'StanHeaders' was built under R version 3.5.2
```

```
rstan_options(auto_write = TRUE)

quakes <- read_csv("http://www.evanlray.com/data/chihara_hesterberg/Quakes.csv")

ggplot(data = quakes, mapping = aes(x = TimeDiff)) +
  geom_histogram(mapping = aes(y = ..density..))
```



We have previously estimated the parameters of a Weibull model for wind speeds via Maximum Likelihood Estimation; recall that we had to do this via numerical optimization. Let's fit a Weibull distribution to the earthquake timing data, but using a Bayesian approach and MCMC this time.

So, we'll use the model

$X_i \overset{\text{iid}}{\sim} \text{Weibull}(k, \lambda)$,

where X_i is the ith observed time between consecutive earthquakes.

Recall that the Weibull distribution has two parameters, the shape parameter $k > 0$ and the scale parameter $\lambda > 0$. If $X \sim \text{Weibull}(k, \lambda)$, then it has pdf

$$f(x|k, \lambda) = \frac{kx^{k-1}}{\lambda^k} e^{-(x/\lambda)^k}$$

In R, the density function can be evaluated with the `dweibull` function, which has the following arguments:

- `x`: vector of values at which to evaluate the pdf.
- `shape`, `scale`: shape and scale parameters, the latter defaulting to 1.
- `log`: logical; if TRUE, returns the log of the pdf.

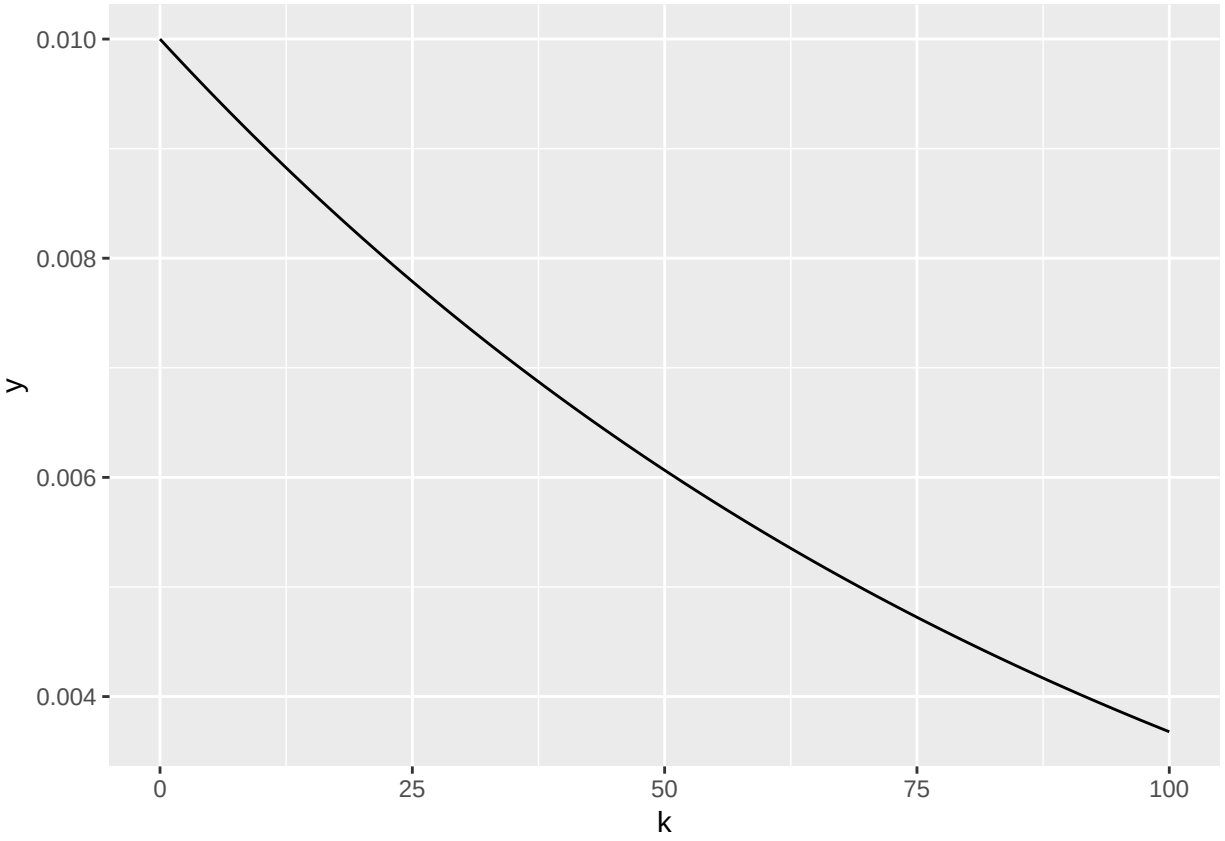
1. Set up model definition in stan

I have set up a skeleton of the stan file, included in this repository. Edit that file now to add necessary declarations and model statements for this model to the data, parameters, and model blocks. The stan function to use for the Weibull distribution is called `weibull`.

See the included `earthquakes_model.stan` file for the Stan code.

I asked you to use prior of $k \sim$ Exponential(0.01) and $\lambda \sim$ Exponential(0.01). Here is a plot of what this prior looks like.

```
ggplot(data = data.frame(k = c(0, 100)), mapping = aes(x = k)) +
  stat_function(fun = dexp, args = list(rate = 0.01))
```



2. Perform estimation

You will need to load the rstan package, set up a list with the data for the stan model, and call `stan` to compile the model and perform sampling.

```
library(rstan)

fit <- stan("earthquakes_model.stan",
  data = list(n = nrow(quakes), x = quakes$TimeDiff),
  iter = 1000,
  chains = 4)
```

```
##
## SAMPLING FOR MODEL 'earthquakes_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.57 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
```

```
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.11243 seconds (Warm-up)
## Chain 1:                0.119695 seconds (Sampling)
## Chain 1:                0.232125 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'earthquakes_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.119 seconds (Warm-up)
## Chain 2:                0.089129 seconds (Sampling)
## Chain 2:                0.208129 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'earthquakes_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
```

```
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.087893 seconds (Warm-up)
## Chain 3:                0.072761 seconds (Sampling)
## Chain 3:                0.160654 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'earthquakes_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.31 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.100589 seconds (Warm-up)
## Chain 4:                0.13217 seconds (Sampling)
## Chain 4:                0.232759 seconds (Total)
## Chain 4:
```

3. Plot results

Make some exploratory plots of the results. It would be nice to have:

- a scatterplot of the posterior samples, showing both parameters for each sample from the posterior
- histograms or density plots summarizing the marginal posterior distribution for each model parameter.

```
samples <- as.data.frame(fit)
head(samples)
```

```
##           k   lambda      lp__
## 1 0.9319509 16.73098 -3126.735
## 2 0.9347022 17.08692 -3126.315
## 3 0.9023295 17.64417 -3126.258
## 4 0.9263102 17.46059 -3125.944
## 5 0.9227347 17.75171 -3126.035
## 6 0.9554903 17.98387 -3127.035
```

4. Find posterior means and credible intervals

Obtain approximate posterior means and 95% posterior credible intervals for each model parameter.

Posterior mean and 95% CI for k:

```
samples %>%
  summarize(
    post_mean_k = mean(k),
    post_95CI_k_lower = quantile(k, probs = 0.025),
    post_95CI_k_upper = quantile(k, probs = 0.975)
  )
```

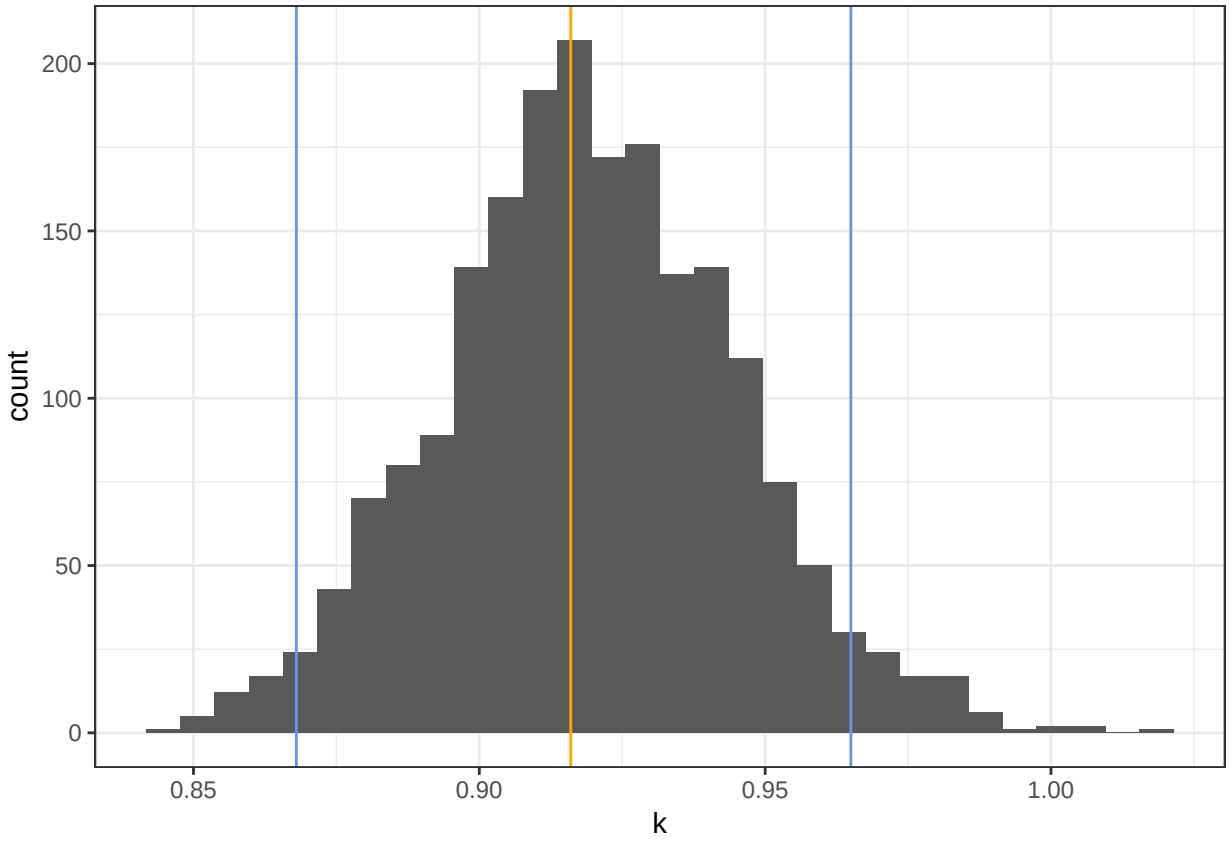
```
##   post_mean_k post_95CI_k_lower post_95CI_k_upper
## 1   0.9192687         0.8703126         0.9721171
```

Interpretation: based on the observed data, a point estimate of k is the posterior mean of 0.916; there is a 95% probability that k is between about 0.868 and 0.965.

I did not ask you to make this plot, but it's nice to just confirm how these relate to the samples from the posterior:

```
ggplot(data = samples, mapping = aes(x = k)) +
  geom_histogram() +
  geom_vline(xintercept = 0.916, color = "orange") +
  geom_vline(xintercept = 0.868, color = "cornflowerblue") +
  geom_vline(xintercept = 0.965, color = "cornflowerblue") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Posterior mean and 95% CI for λ

```
samples %>%
  summarize(
    post_mean_lambda = mean(lambda),
    post_95CI_lambda_lower = quantile(lambda, probs = 0.025),
    post_95CI_lambda_upper = quantile(lambda, probs = 0.975)
  )
```

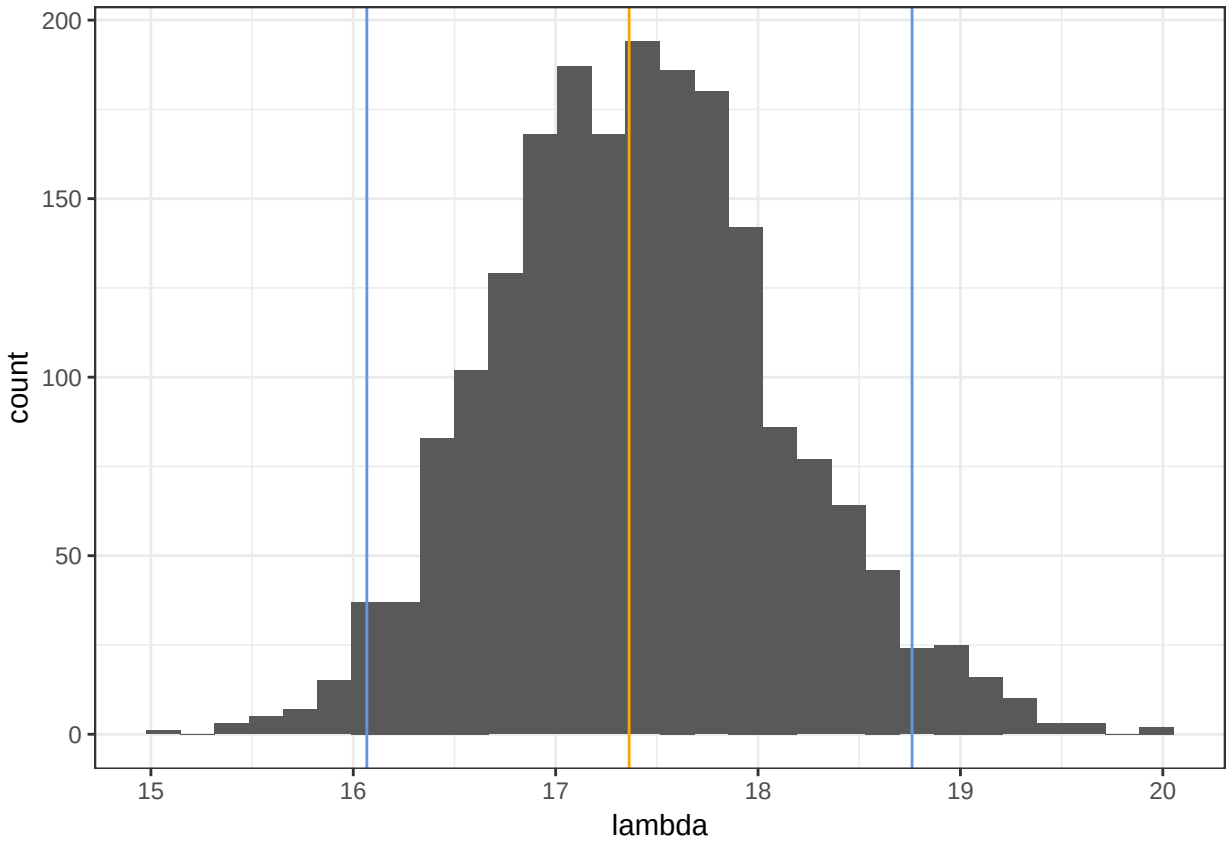
```
##   post_mean_lambda post_95CI_lambda_lower post_95CI_lambda_upper
## 1         17.40965               16.07882               18.93103
```

Interpretation: based on the observed data, a point estimate of λ is the posterior mean of 17.363; there is a 95% probability that λ is between about 16.067 and 18.761.

I did not ask you to make this plot, but it's nice to just confirm how these relate to the samples from the posterior:

```
ggplot(data = samples, mapping = aes(x = lambda)) +
  geom_histogram() +
  geom_vline(xintercept = 17.363, color = "orange") +
  geom_vline(xintercept = 16.067, color = "cornflowerblue") +
  geom_vline(xintercept = 18.761, color = "cornflowerblue") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



5. What is your effective sample size for each parameter?

```
nrow(samples)
```

```
## [1] 2000
```

```
fit
```

```
## Inference for Stan model: earthquakes_model.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##            mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## k          0.92    0.00 0.03     0.87     0.90     0.92     0.94     0.97
## lambda    17.41    0.02 0.71    16.08    16.92    17.39    17.85    18.93
## lp__   -3126.90    0.04 1.02 -3129.61 -3127.27 -3126.59 -3126.18 -3125.92
##        n_eff Rhat
## k       1155 1.00
## lambda  1153 1.00
## lp__     831 1.01
##
## Samples were drawn using NUTS(diag_e) at Thu Mar  7 11:25:52 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

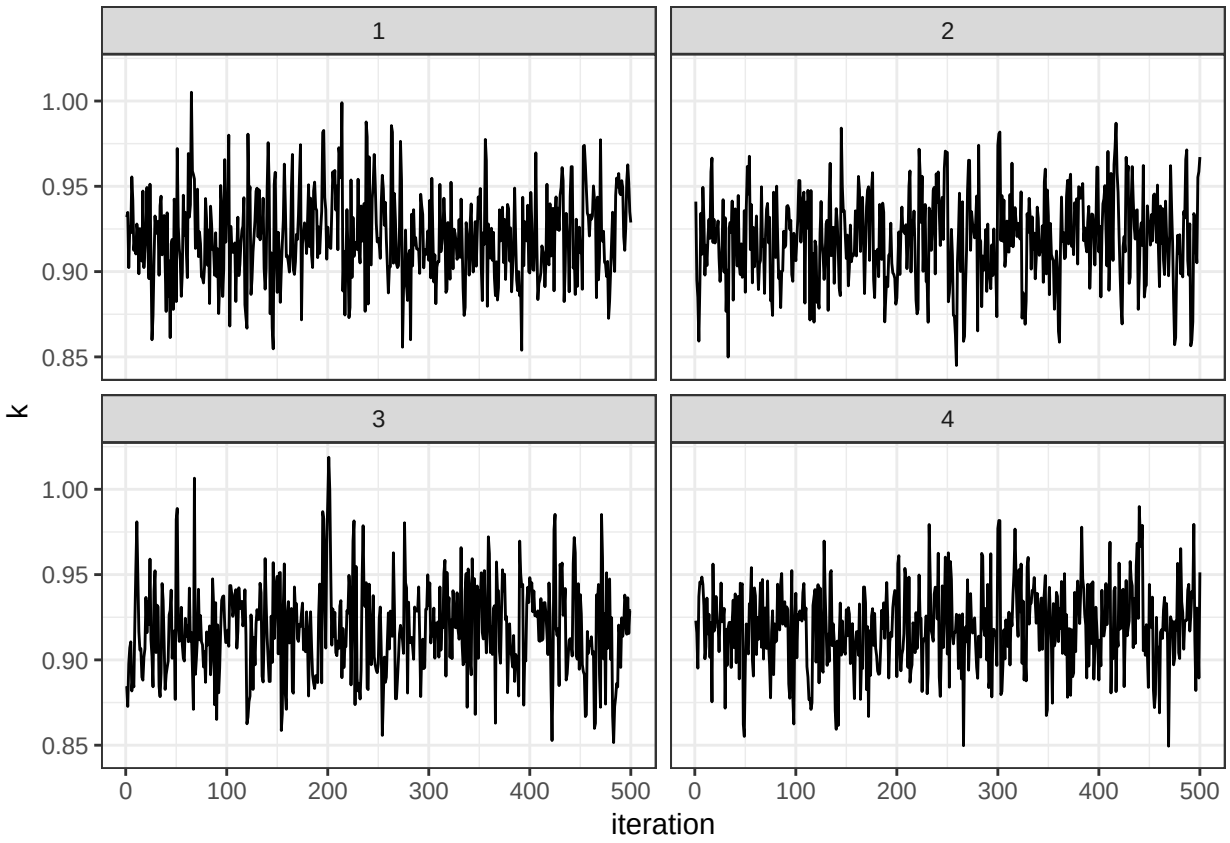
We had four Markov chains; each generated 1000 samples, but we discarded the first 500 samples from each as burn-in. Thus, we have a total of 2000 samples.

However, because of dependence in consecutive samples generated from a given Markov chain, we have an *effective* samples size less than 2000. Our 2000 dependent samples give us about as much information about what the posterior looks like as roughly 1297 independent samples for k, and about 1382 independent samples for λ.

Here's some code that I'm including to explore this more, but it's not a priority for you to dive into this in this class.

```
samples_array <- as.array(fit)
samples_by_chain <- bind_rows(
  samples_array[, 1, 1:2] %>%
    as.data.frame() %>%
    mutate(
      k_previous = lag(k, 1),
      iteration = row_number(),
      chain = 1
    ),
  samples_array[, 2, 1:2] %>%
    as.data.frame() %>%
    mutate(
      k_previous = lag(k, 1),
      iteration = row_number(),
      chain = 2
    ),
  samples_array[, 3, 1:2] %>%
    as.data.frame() %>%
    mutate(
      k_previous = lag(k, 1),
      iteration = row_number(),
      chain = 3
```

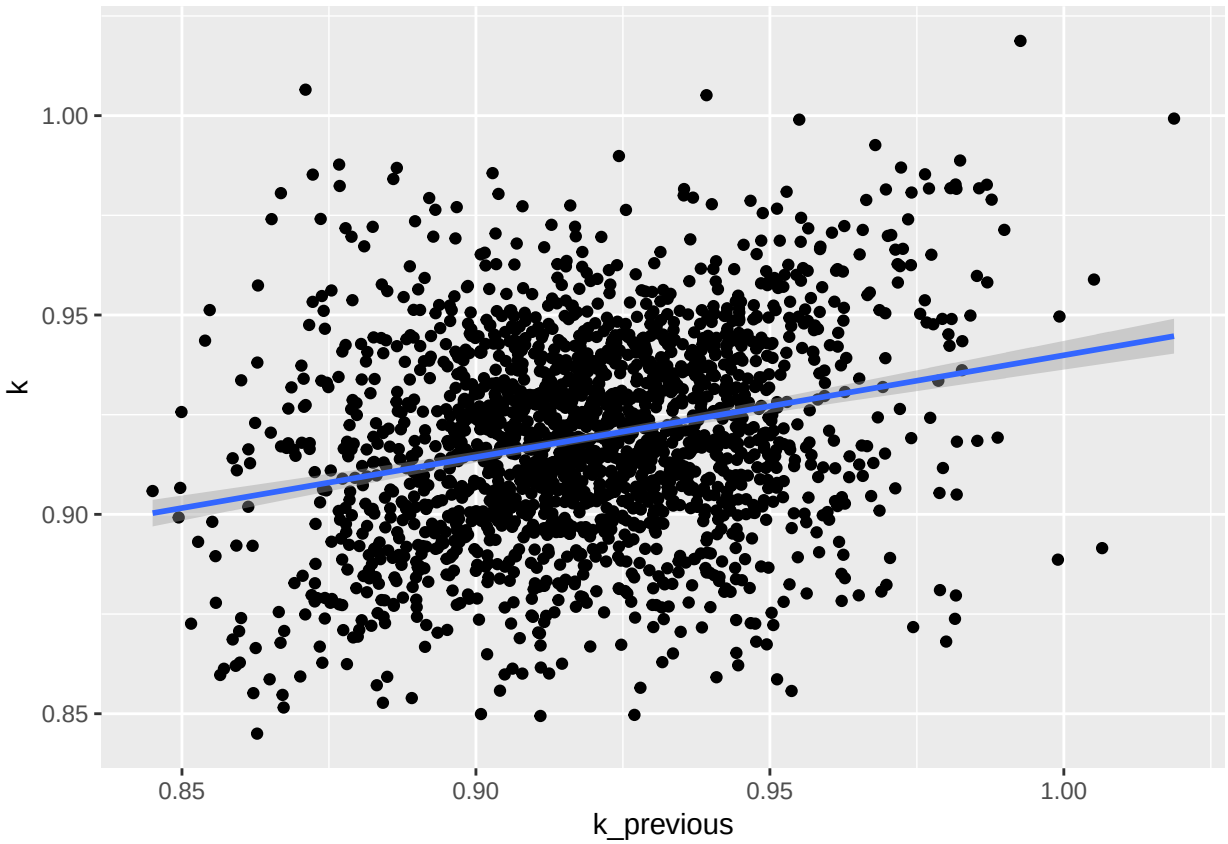
```
    ),
  samples_array[, 4, 1:2] %>%
    as.data.frame() %>%
    mutate(
      k_previous = lag(k, 1),
      iteration = row_number(),
      chain = 4
    )
)

ggplot(data = samples_by_chain, mapping = aes(x = iteration, y = k)) +
  geom_line() +
  facet_wrap( ~ chain) +
  theme_bw()
```




```
ggplot(data = samples_by_chain, mapping = aes(x = k_previous, y = k)) +
  geom_point() +
  geom_smooth(method = "lm")
```

```
## Warning: Removed 4 rows containing non-finite values (stat_smooth).
```

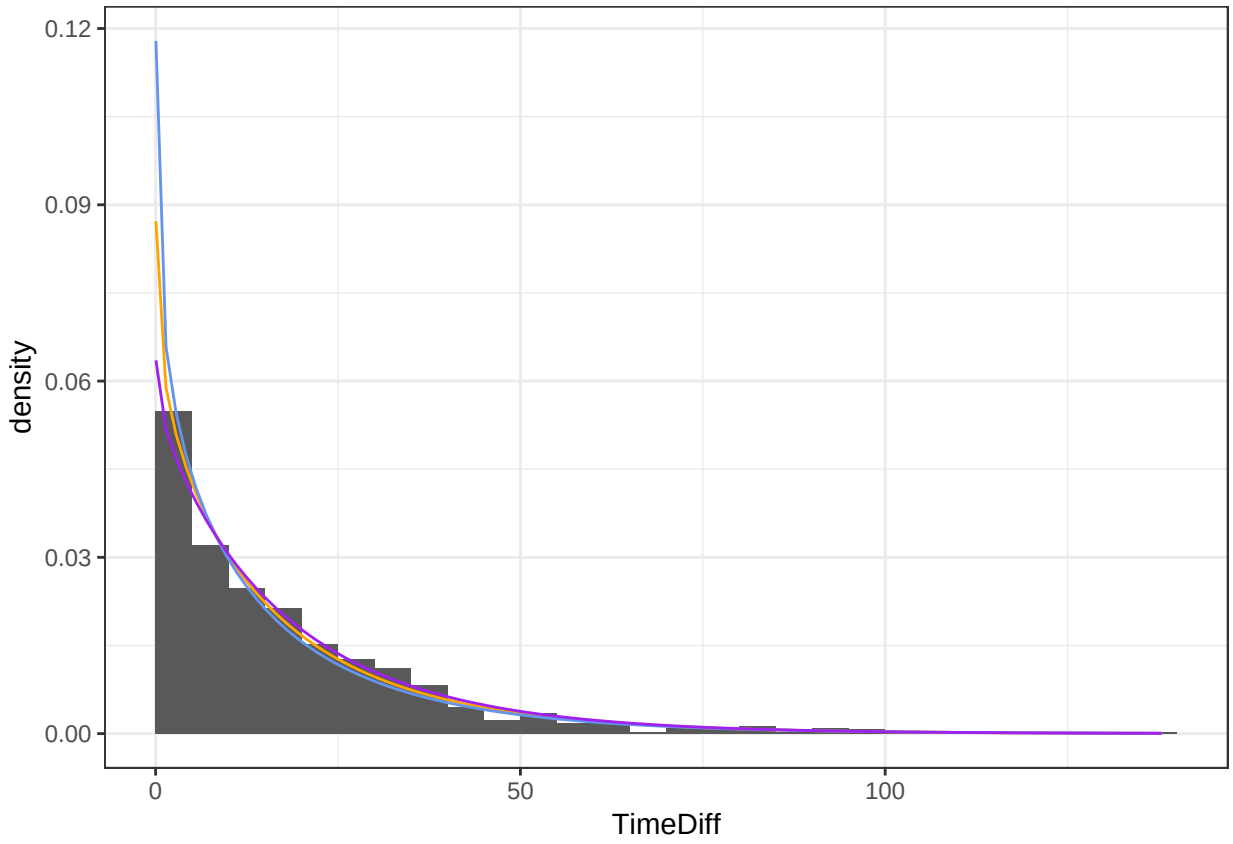
```
## Warning: Removed 4 rows containing missing values (geom_point).
```



The lack of independence between consecutive samples is indicated by the slight trend in the scatter plot above. In this plot, each point has coordinates (sampled value of k at iteration $i-1$, sampled value of k at iteration i). The slight trend indicates correlation between these sampled values; knowing the sampled value of k at iteration $i-1$ tells you a little bit about the sampled value of k at iteration i.

6. Add three new layers to the data plot below: 1) a Weibull density using the posterior mean parameter values; 2) a Weibull density using the parameter values at the lower endpoints of the 95% credible intervals; and 3) a Weibull density using the parameter values at the upper endpoints of the 95% credible intervals.

```
ggplot(data = quakes, mapping = aes(x = TimeDiff)) +
  geom_histogram(mapping = aes(y = ..density..), boundary = 0, binwidth = 5) +
  stat_function(fun = dweibull, args = list(shape = 0.916, scale = 17.363), color = "orange") +
  stat_function(fun = dweibull, args = list(shape = 0.868, scale = 16.067), color = "cornflowerblue") +
  stat_function(fun = dweibull, args = list(shape = 0.965, scale = 18.761), color = "purple") +
  theme_bw()
```


0.12
0.09
0.06
0.03
0.00
density
0
50
100
TimeDiff