Bayesian_HW_6_Whitson

Paul Whitson

5/19/2020

Homework 6: Robust Estimation

Data Exploration:

The following exercise will utilize the stock price of Apple, Inc. (AAPL) during the calendar year 2015:

```
library(HDInterval)
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(ggplot2)
library(coda)
```

```
##
## Attaching package: 'coda'
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
suppressWarnings(library(quantmod))
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

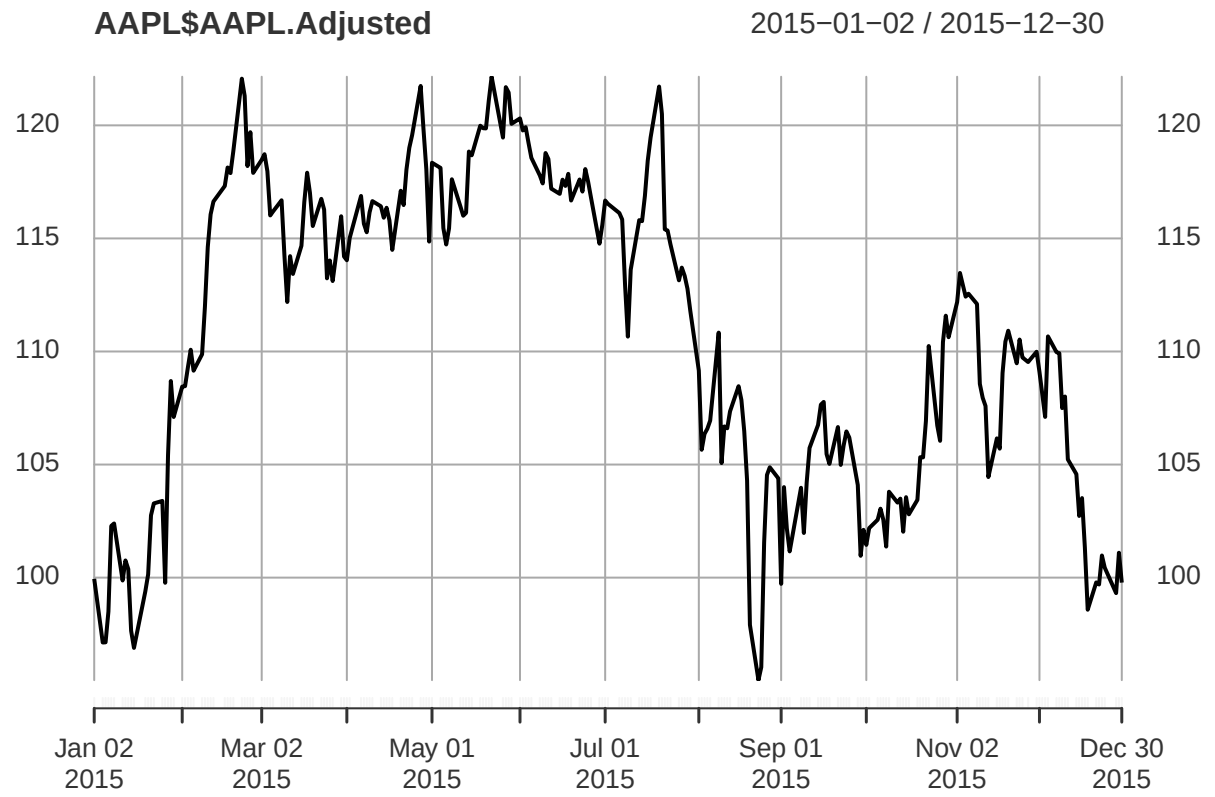
```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
getSymbols("AAPL", from="2015-1-1",to="2015-12-31") #creates an xts object
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "AAPL"
```

```
#Plot raw data: adjusted daily price:
plot(AAPL$AAPL.Adjusted)
```

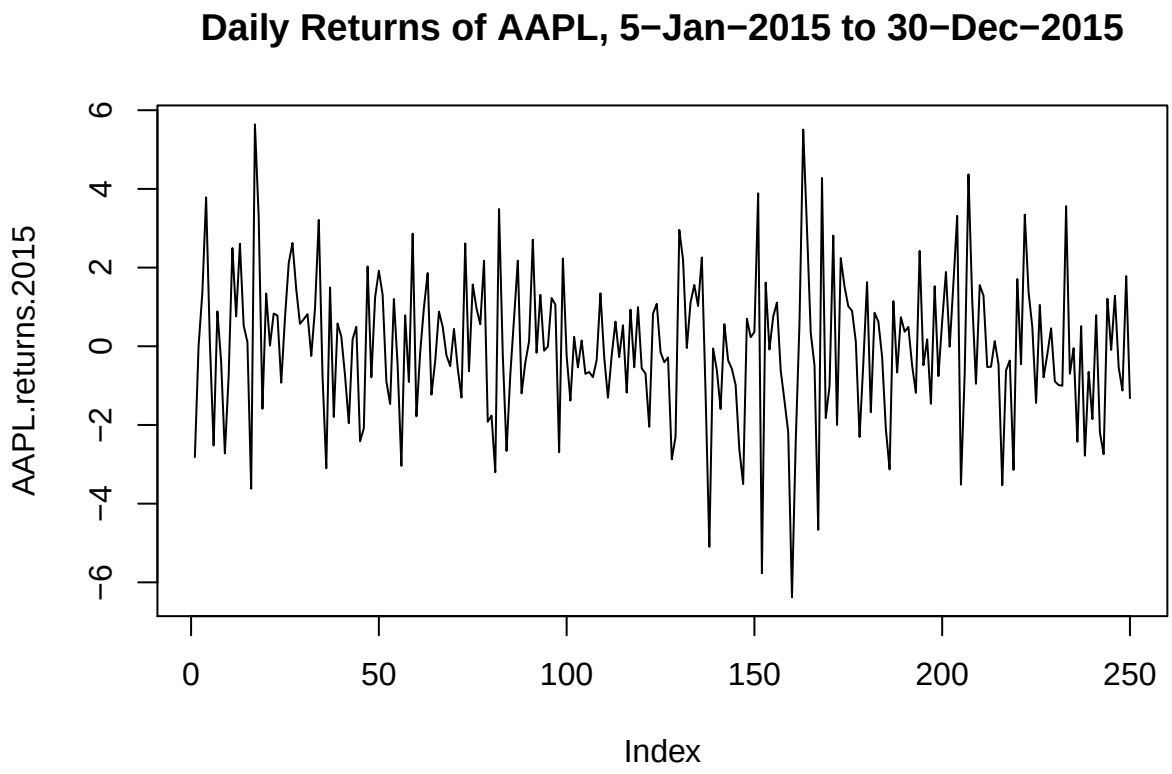


We will focus on the daily returns, defined as the change in stock price between the current day and the preceding day.

```
#Calculate daily returns:
AAPL.1.day<-as.matrix(AAPL)
head(AAPL.1.day)
```

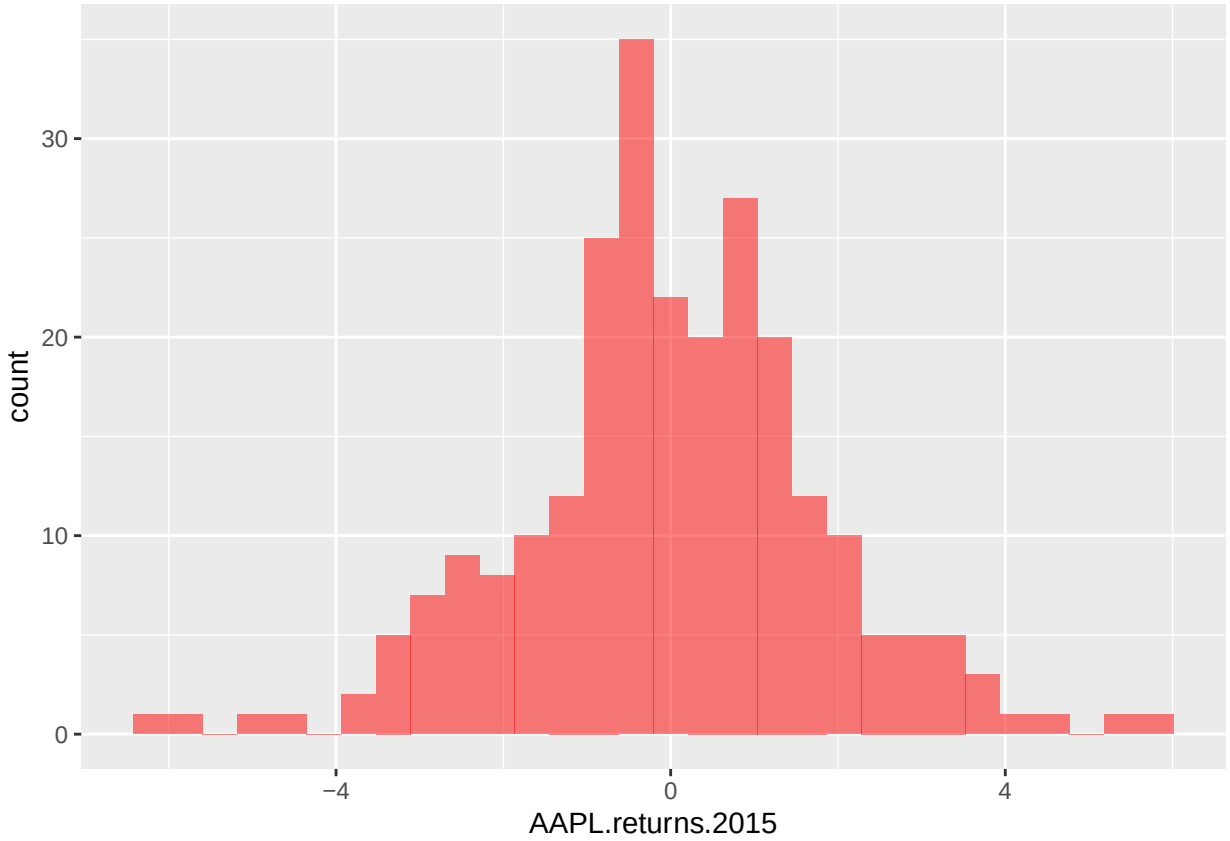
```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2015-01-02    111.39    111.44   107.35     109.33    53204600
## 2015-01-05    108.29    108.65   105.41     106.25    64285500
## 2015-01-06    106.54    107.43   104.63     106.26    65797100
## 2015-01-07    107.20    108.20   106.70     107.75    40105900
## 2015-01-08    109.23    112.15   108.70     111.89    59364500
## 2015-01-09    112.67    113.25   110.21     112.01    53699500
##            AAPL.Adjusted
## 2015-01-02      99.94589
## 2015-01-05      97.13024
## 2015-01-06      97.13942
## 2015-01-07      98.50152
## 2015-01-08     102.28619
## 2015-01-09     102.39584
```

```
AAPL.returns.2015<-diff(AAPL.1.day[,6])

#plot time series of daily returns
plot(AAPL.returns.2015, type = 'l', main = "Daily Returns of AAPL, 5-Jan-2015 to 30-Dec-2015")
```




```
#plot histogram of daily returns
ggplot(data=data.frame(AAPL.returns.2015), aes(x=AAPL.returns.2015)) + geom_histogram(fill = "red", alp
```

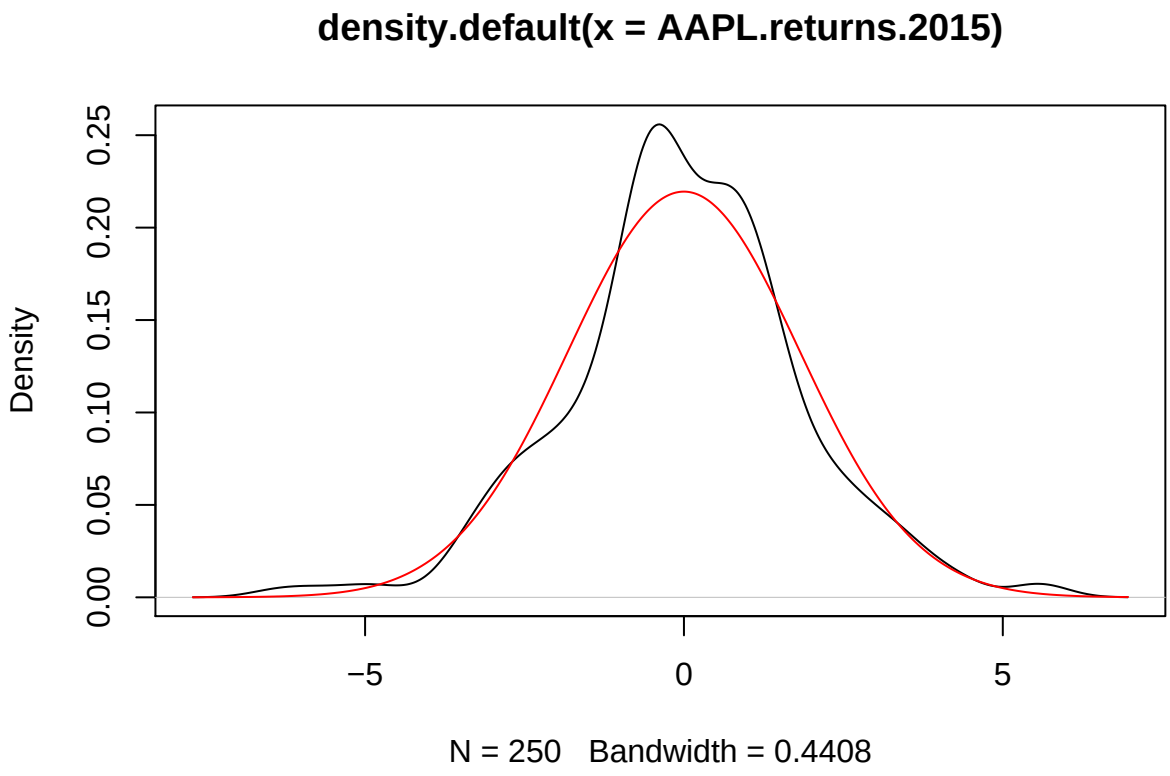
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



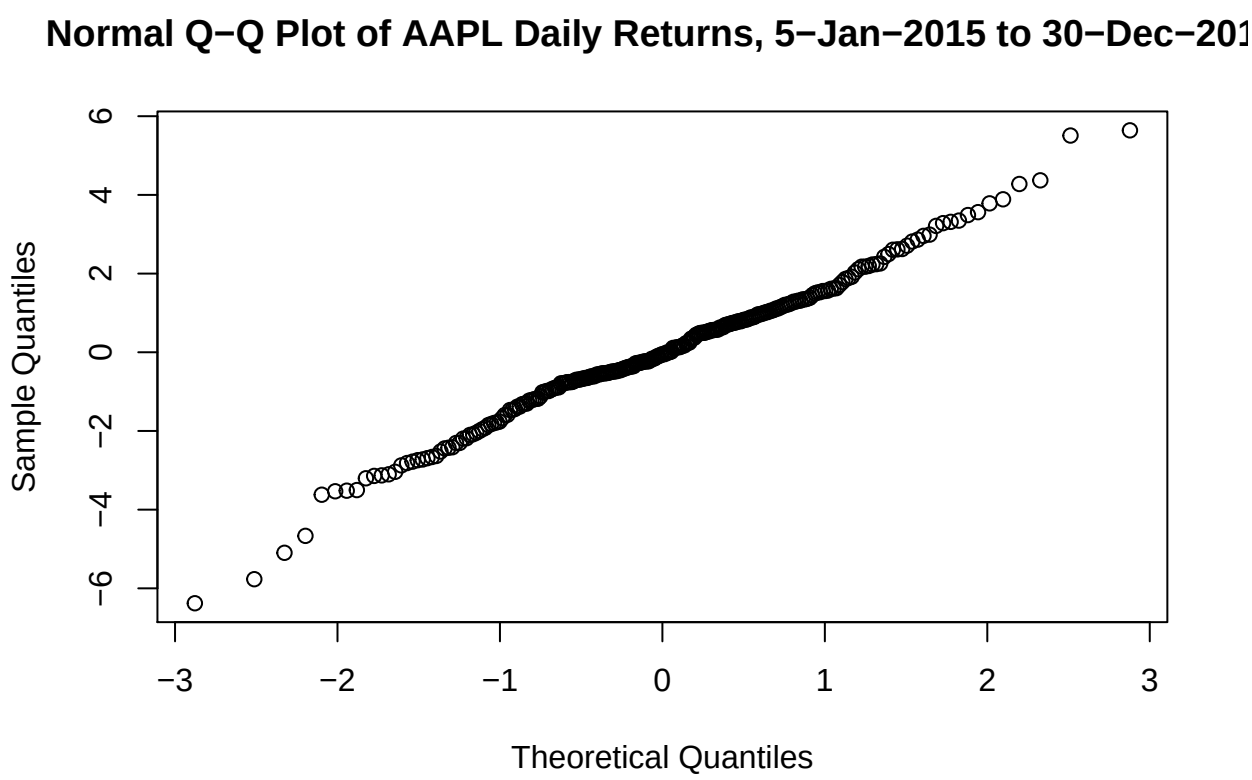
As shown in the above histogram, the daily returns appear to show some multi-modal or fat-tailed (high kurtosis) behavior.

The normality of the distribution was analyzed as follows:

```
#Compare data distribution to normal distribution:
AAPL.dens<-density(AAPL.returns.2015)
plot(AAPL.dens)
lines(AAPL.dens$x,dnorm(AAPL.dens$x,mean(AAPL.returns.2015),sd(AAPL.returns.2015)),col="red")
```




```
#Normal Probability Plot:
qqnorm(AAPL.returns.2015, main = "Normal Q-Q Plot of AAPL Daily Returns, 5-Jan-2015 to 30-Dec-2015")
```




```
#there is a difference in slope between the middle and edges of the distribution, indicating fat tails.
```

The empirical density plot (black) does appear to indicate fatter tails and a higher / narrower center compared to the standard normal distribution (red). The normal probability plot looks mostly linear, but with some change in slope at the edges compared to the center.

While these all suggest non-normality, it can be difficult to determine deviations from normality in the extreme tails when we only have ~250 data points.

Estimation of Model Parameters (Normal)

For comparison, the sample mean and standard deviation were first calculated. The sample mean is approximately 0, and the sample standard deviation is approximately 1.82.

```
#Calculate descriptive statistics:
mean(AAPL.returns.2015) #-0.000663452 is the mean daily change (very close to 0)
```

```
## [1] -0.000663452
```

```
sd(AAPL.returns.2015) #1.8176 is the normal ML estimate
```

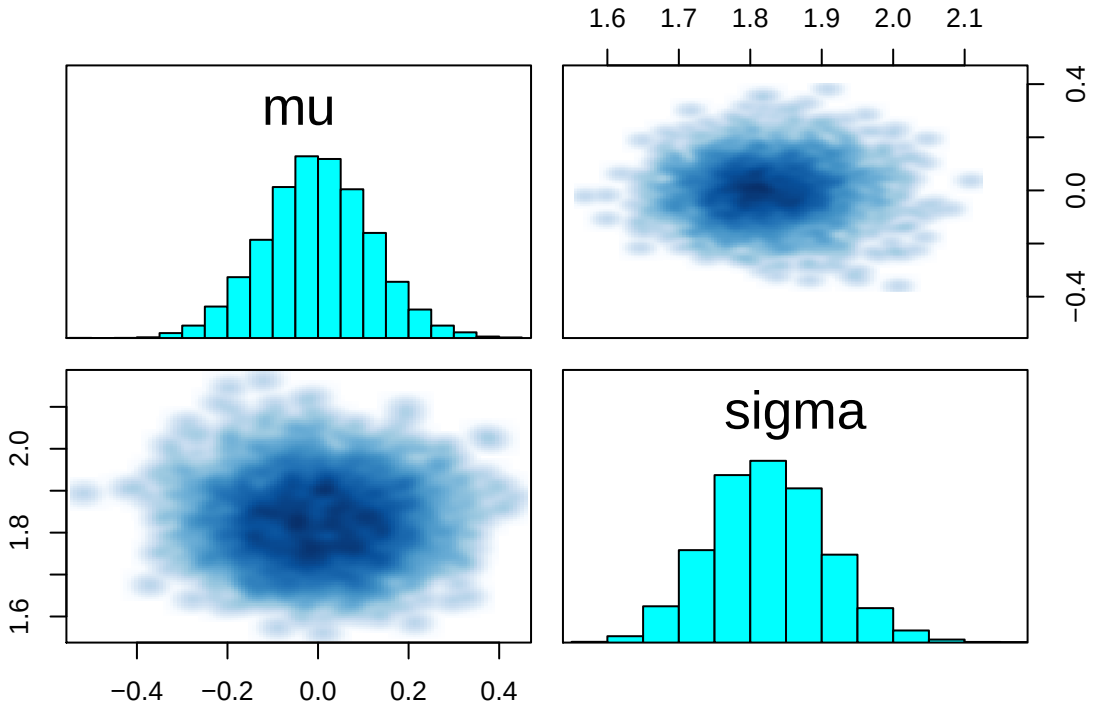
```
## [1] 1.817628
```

We will use MCMC sampling via Stan to estimate the parameters of the distribution in Stan using a normal distribution.

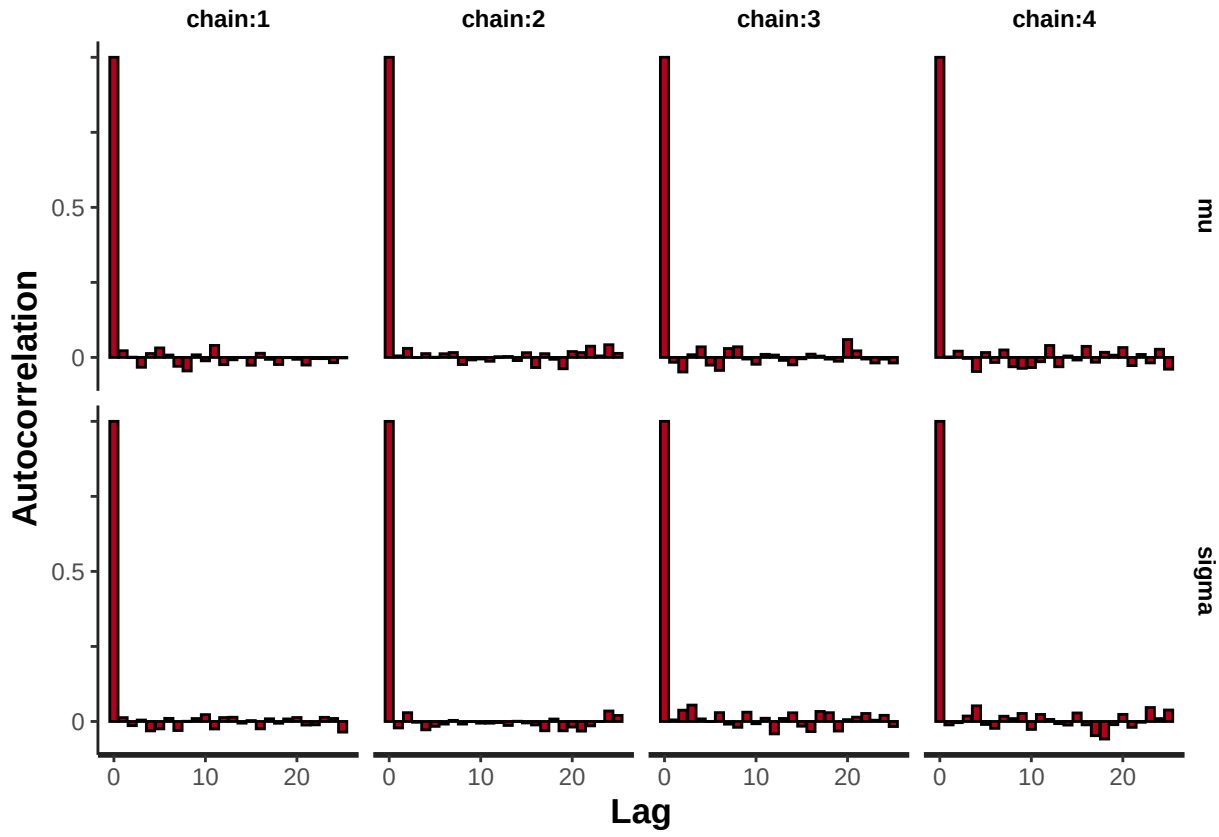
```
## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Users/Paul/Library/R/3.6/
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:88:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknou
## namespace Eigen {
## ^
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expe
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file no
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

Diagnostics were performed on the MCMC results. All 4 chains seemed to converge fairly well. Mu and sigma estimates do not appear to be correlated with each other, and with a thinning parameter of 5, there is minimal autocorrelation:

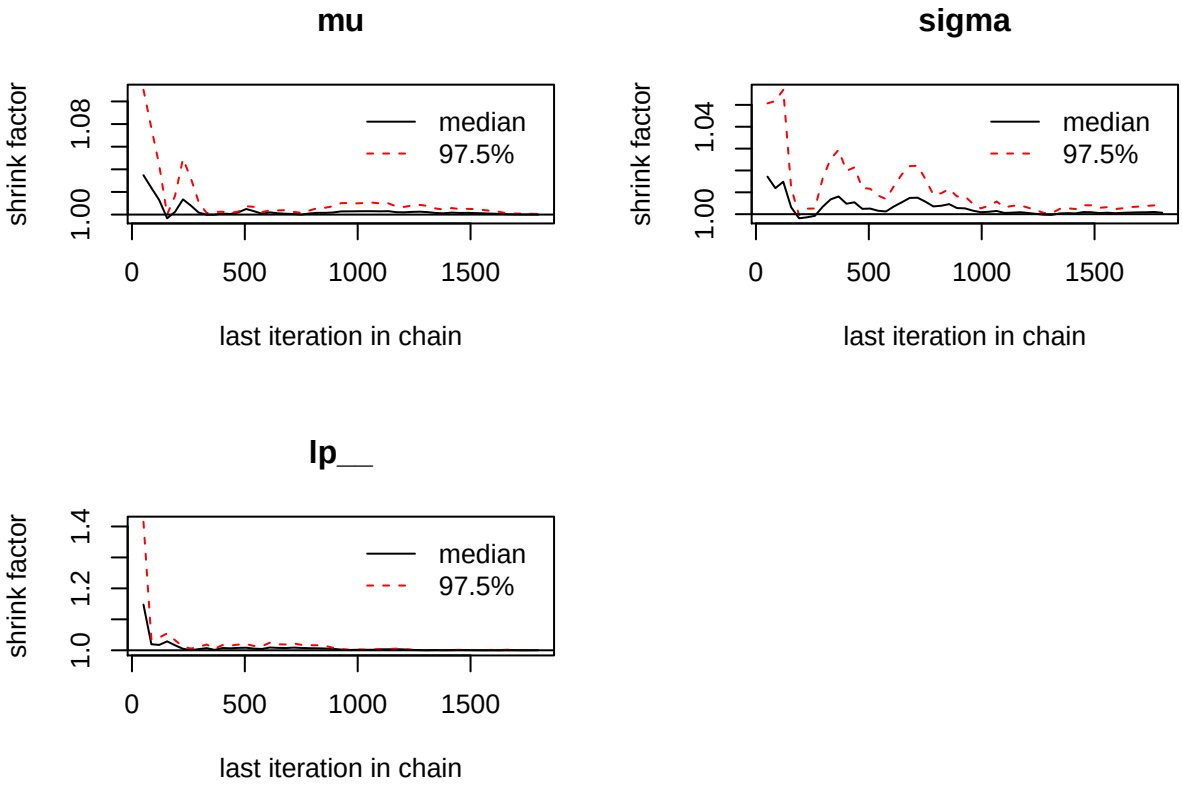
```
#traceplot(stanFit1) #all 4 chains seem to overlap
pairs(stanFit1, pars=c('mu','sigma')) #mu and sigma do not appear to be correlated with each other
```




```
stan_ac(stanFit1, separate_chains = T) #autocorrelation is minimal, now that we have implemented a thin
```



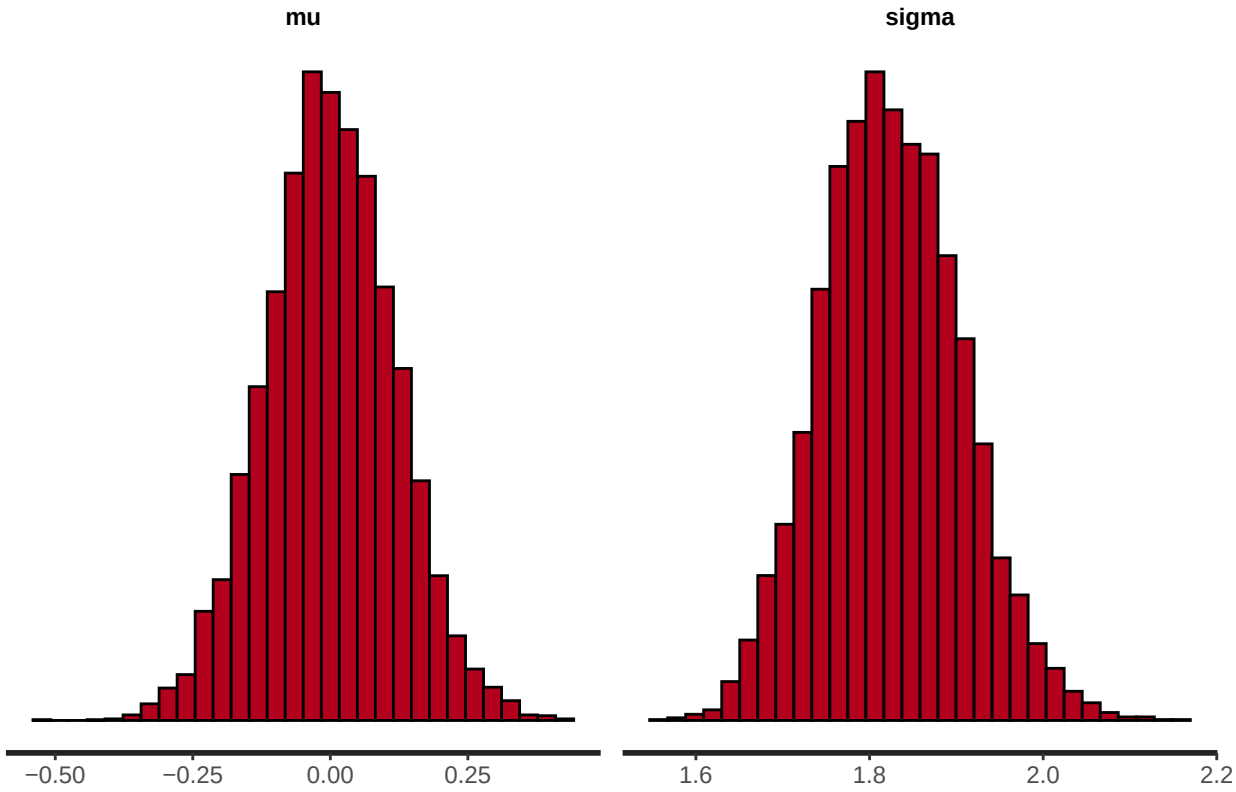

```
stanFit1_coda <- mcmc.list( lapply ( 1:ncol(stanFit1) ,
                                     function (x) { mcmc(as.array(stanFit1)[,x,]) } ))
gelman.plot(stanFit1_coda) #there is some bouncing around for the first 500 iterations, but all the cha
```



Using the simulation above assuming a Gaussian distribution, the following parameter estimates were obtained. The estimate of mu is 0.00, and the estimate of sigma is 1.83; these are consistent with the estimates calculated directly from the data.

```
stan_hist(stanFit1) #both distributions are approximately bell-shaped
```

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




```
print(stanFit1)
```

```
## Inference for Stan model: 77a7302f12306a23f9ba2a46472fa56e.
## 4 chains, each with iter=10000; warmup=1000; thin=5;
## post-warmup draws per chain=1800, total post-warmup draws=7200.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## mu       0.00    0.00 0.12   -0.23   -0.08    0.00    0.08    0.23  7148
## sigma    1.83    0.00 0.08    1.68    1.77    1.82    1.88    1.99  6830
## lp__  -274.29    0.01 1.01 -277.01 -274.69 -273.97 -273.57 -273.31  7224
##       Rhat
## mu       1
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Tue May 19 23:04:08 2020.
## 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).
```

```
hdi(stanFit1_coda, credMass = 0.95)
```

```
##               mu    sigma      lp__
## lower -0.2431847 1.668462 -276.3392
## upper  0.2188583 1.982812 -273.2856
```

```
## attr(,"credMass")
## [1] 0.95
```

Estimation of Model Parameters (Robust)

In this case, we will use a t-distribution to model the distribution of the data, as it is more robust to the presence of outliers and "fat-tailed" distributions:

```
#Model string:

modelString2 = "
data {
    int<lower=1> Ntotal;
    real y[Ntotal];
    real mean_mu;
    real sd_mu;
}
transformed data {
    real unifLo;
    real unifHi;
    real normalSigma;
    real expLambda; // new parameter for prior distribution of nu
    unifLo = sd_mu/1000;
    unifHi = sd_mu*1000;
    normalSigma = sd_mu*100;  // 100 times larger than MLE
    expLambda = 1/29.0; // set value for mean of exponential distribution of nuMinusOne at 29 (implies n
}
parameters {
    real<lower=0> nuMinusOne;
    real mu;
    real<lower=0> sigma;
}
transformed parameters {
  real<lower=0> nu;
  nu=nuMinusOne+1; // nuMinusOne can be anything >0, shift it to make nu >= 1
}
model {
    sigma ~ uniform(unifLo, unifHi); // the standard deviation of the normal distribution of the data ha
    mu ~ normal(mean_mu, normalSigma); // the mean of the normal distribution of the data has a Gaussian
    nuMinusOne ~ exponential(expLambda); // assume an exponential prior distribution of nu
    y ~ student_t(nu, mu, sigma); // in this case, we use a t-distribution with an extra parameter, nu
}
"

#Create Stan DSO object:
stanDso2 <- stan_model( model_code=modelString2)
```

```
## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Users/Paul/Library/R/3.6/I
## In file included from <built-in>:1:
```

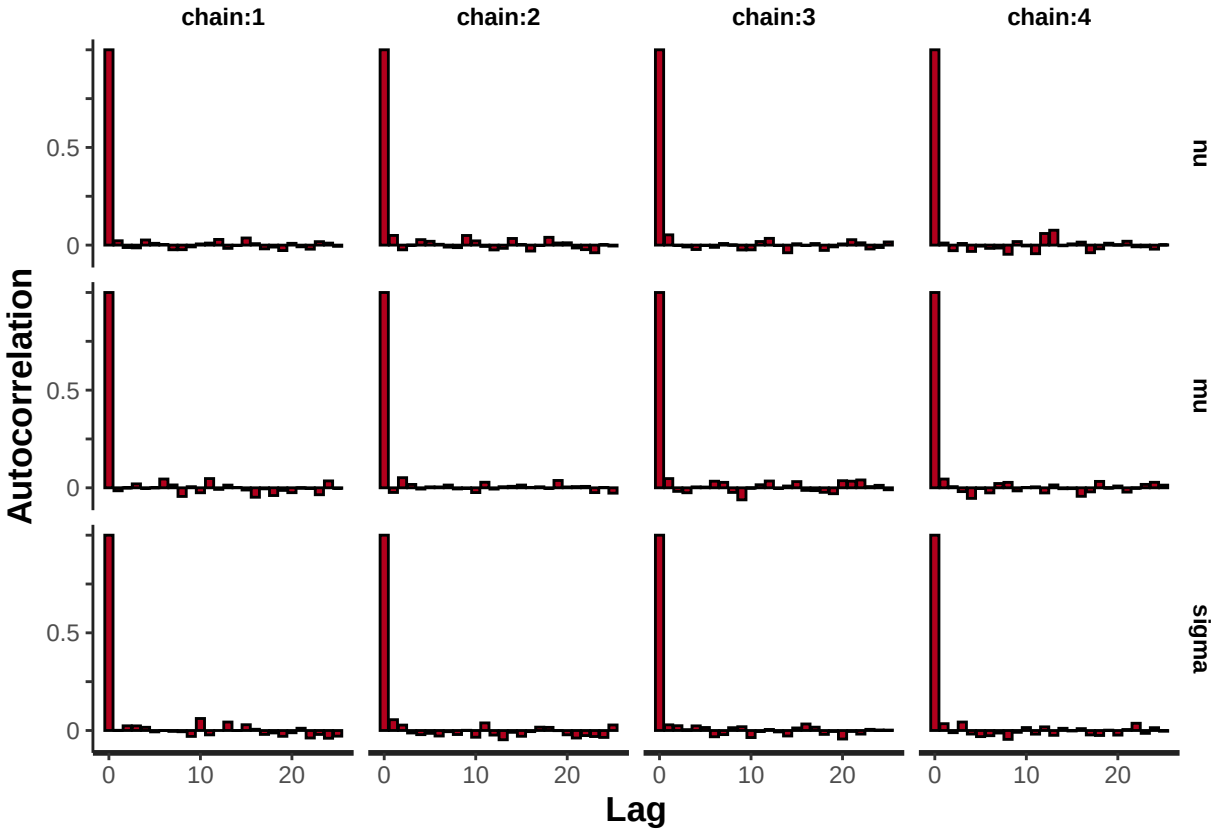
```
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:88:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknow
## namespace Eigen {
## ^
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expe
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Users/Paul/Library/R/3.6/library/StanHeaders/include/stan/math/prim/mat/fun/E
## In file included from /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Dense:1:
## /Users/Paul/Library/R/3.6/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file no
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

```
#Fit Model:
stanFit2 <- sampling( object=stanDso2 ,
                      data = dataList ,
                      pars=c('nu', 'mu', 'sigma'), #add nu
                      chains = 4,
                      cores= 4,
                      iter = 10000,
                      warmup = 1000, #in early attempts, the Gelman plots showed some divergence until
                      thin = 5 )
```

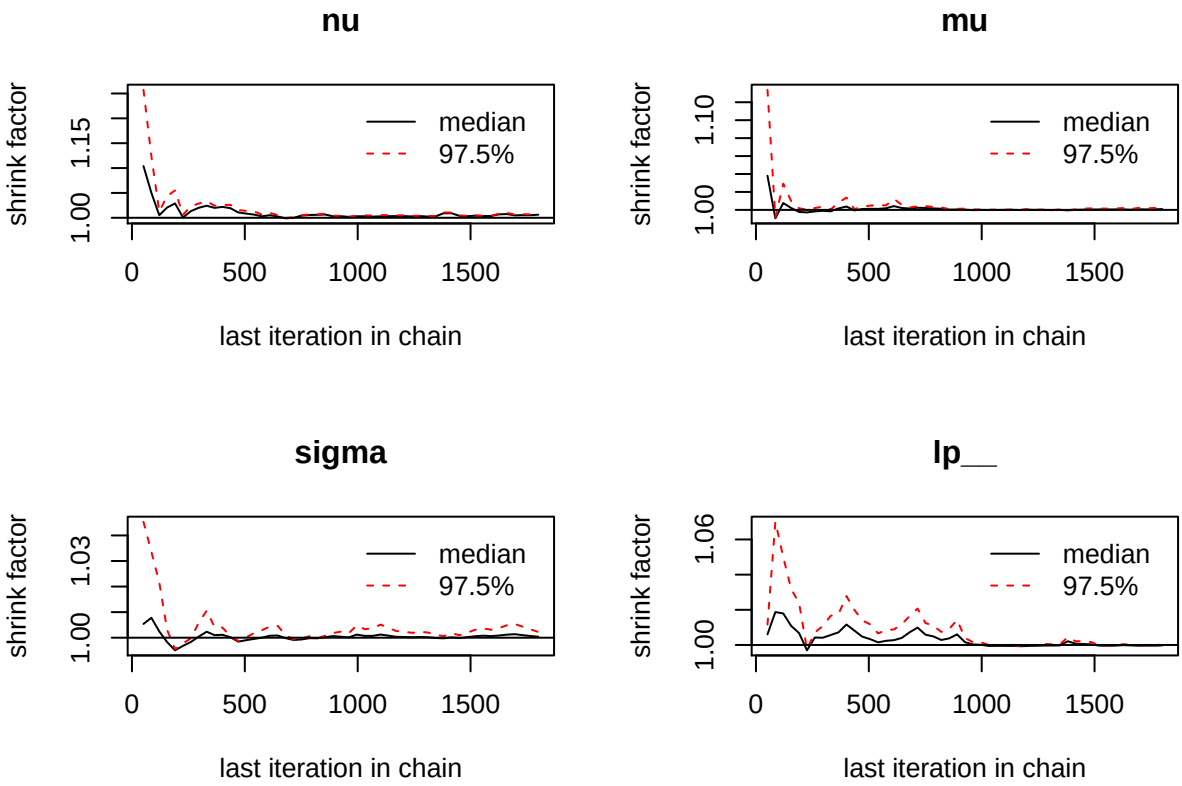
Robust Estimation: Diagnostics:

Diagnostics were performed on the sampling results. The chains seemed to overlap and converge fairly well, and there was minimal autocorrelation.

```
#traceplot(stanFit2) #all 4 chains seem to overlap
stan_ac(stanFit2, separate_chains = T) #autocorrelation is minimal, now that we have implemented a thin
```

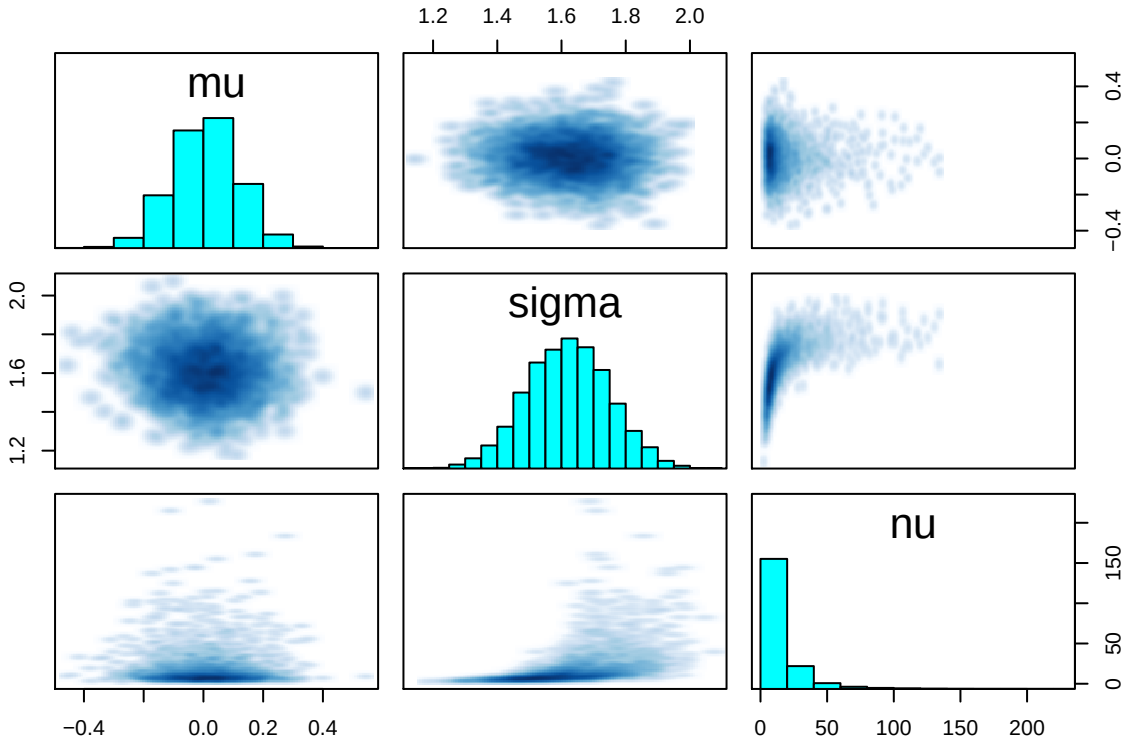



```
stanFit2_coda <- mcmc.list( lapply ( 1:ncol(stanFit2) ,
                                     function (x) { mcmc(as.array(stanFit2)[,x,]) } ))
gelman.plot(stanFit2_coda) #there is some bouncing around for the first 500 iterations, but all the cha
```

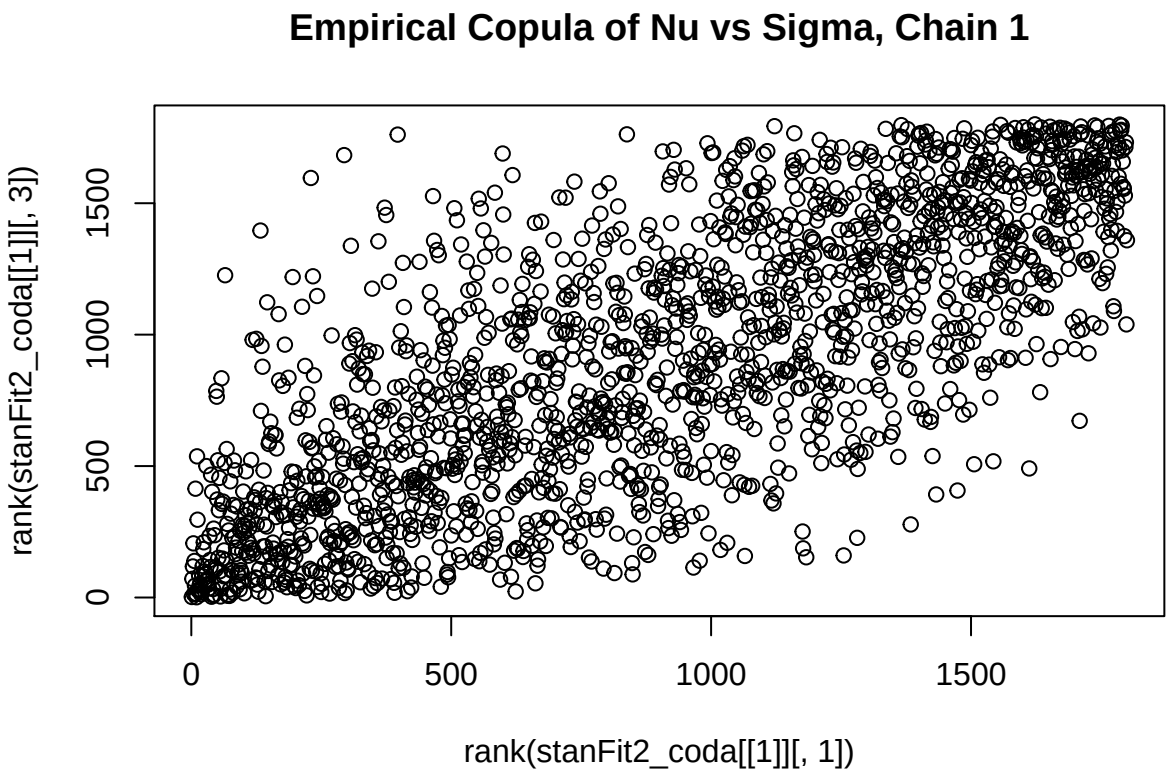


The pairs plot does not indicate correlation between mu and sigma, but does show some potential correlation with sigma and nu. The empirical copula also indicates some correlation between nu and sigma, so these estimates should be regarded with some caution.

```
pairs(stanFit2, pars=c('mu','sigma', "nu"))
```




```
plot(x = rank(stanFit2_coda[[1]][,1]),
     y = rank(stanFit2_coda[[1]][,3]),
     main = "Empirical Copula of Nu vs Sigma, Chain 1")
```



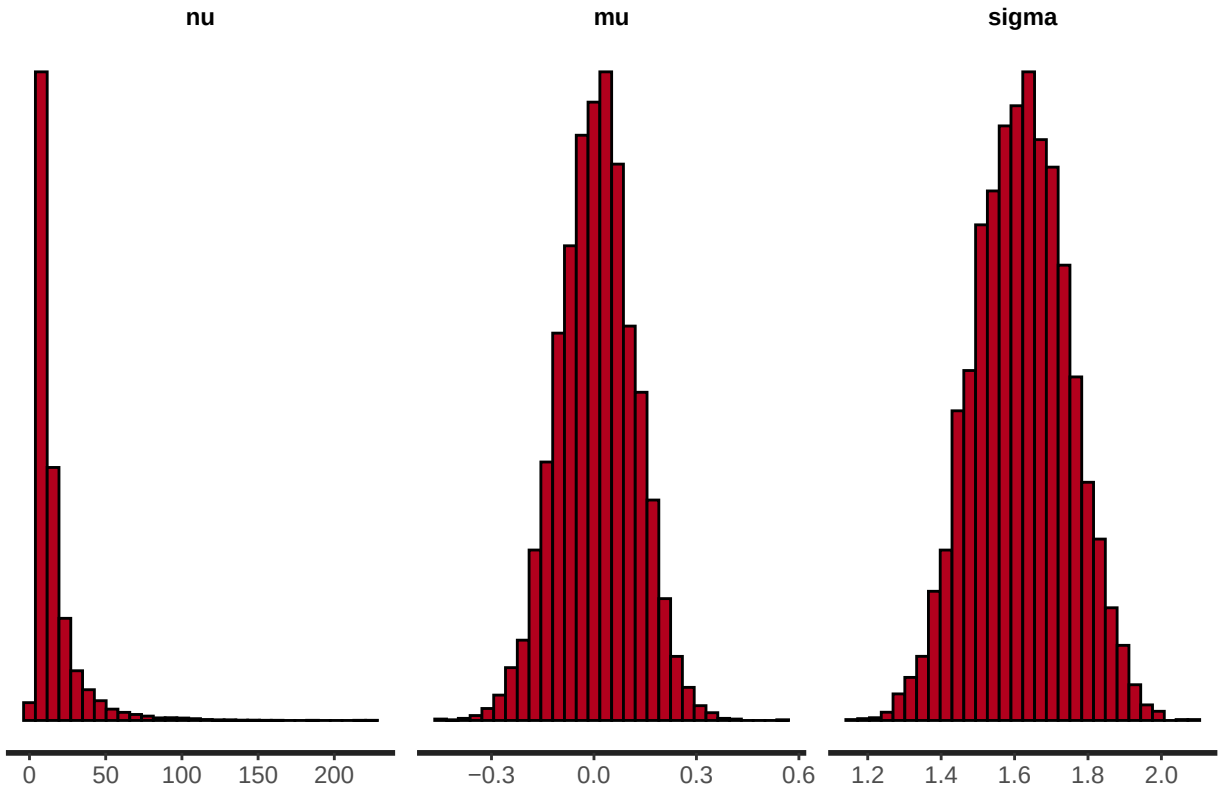
Parameter Estimates and HDI:

The following output summarizes the results of robust estimation with the t-distribution.

Nu has a lot of density at low numbers and a mean at about 15, possibly indicating non-normality in the t-distribution. However, the 95% HDI for nu ranges from 3 to 45, so it is difficult to say definitively that this distribution is non-normal.

```
stan_hist(stanFit2)
```

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




```
print(stanFit2) #estimate of standard deviation is 1.62
```

```
## Inference for Stan model: b816da9926a0ab71382faa9734546682.
## 4 chains, each with iter=10000; warmup=1000; thin=5;
## post-warmup draws per chain=1800, total post-warmup draws=7200.
##
##          mean se_mean    sd    2.5%     25%     50%     75%   97.5% n_eff
## nu      15.41    0.19 15.76    4.13    6.95   10.21   17.27   60.25  6748
## mu       0.01    0.00  0.11   -0.21   -0.07    0.01    0.08    0.22  6876
## sigma    1.62    0.00  0.13    1.37    1.53    1.62    1.71    1.87  6413
## lp__  -356.37    0.01  1.23 -359.50 -356.94 -356.08 -355.48 -354.94  6817
##       Rhat
## nu       1
## mu       1
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Tue May 19 23:04:49 2020.
## 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).
```

```
hdi(stanFit2_coda)
```

```
##             nu         mu    sigma      lp__
```

```
## lower  3.030522 -0.2063099 1.376444 -358.7649
## upper 44.195583  0.2243372 1.875638 -354.8394
## attr(,"credMass")
## [1] 0.95
```

Results and Conclusions:

The first method (normal distribution) produces a higher estimate of the standard deviation (1.83) vs the robust (t-distribution) method's estimate of 1.62. This is because the normal distribution method is more sensitive to the presence of outliers in the tails.

In this case, the distribution of the data was fairly symmetrical (potential outliers on both sides), so the estimate of mu was very similar between the two methods.