Using Overdispersion to Estimate Network Size

Anna Smith (Statistics, Columbia) and Tian Zheng (Statistics, Columbia)

June 19, 2017

Motivation

In the following code, we demonstrate how overdispersion in network data can be leveraged to obtain better estimates of personal network size from standard survey questions. For a more detailed description of this issue, see Zheng et al. (2006).

Briefly, we motivate the idea behind the methodology used here with a real world application (taken from Zheng et al. 2006):

> Recently a survey was taken of Americans asking, among other things, "How many males do you know incarcerated in state or federal prison?" The mean of the responses to this question was 1.0. To a reader of this journal, that number may seem shockingly high. We would guess that you probably do not know anyone in prison. In fact, we would guess that most of your friends do not know anyone in prison either. This number may seem totally incompatible with your social world.
>
> So how was the mean of the responses 1? According to the data, 70% of the respondents reported knowing 0 people in prison. However, the responses show a wide range of variation, with almost 3% reporting that they know at least 10 prisoners. Responses to some other questions of the same format, for example, "How many people do you know named Nicole?," show much less variation.

In this analysis, we can treat this **overdispersion as a source of information**, not just as an issue that requires correction.

Simulate Data

For illustration, we simulate data from the prior, as follows:

```r
I <- 200;           ## number of surveyed individuals
K <- 32;            ## number of subgroups

mu_alpha <- 5;
mu_beta <- -5;
sigma_alpha <- 1;
sigma_beta <- 1;

alpha <- rnorm(I, mu_alpha, sigma_alpha);   ## log gregariousness of individuals
beta <- rnorm(K, mu_beta, sigma_beta);      ## log prevalence of subgroups

omega_inv <- runif(K, 0.1, 0.95);
omega <- 1 / omega_inv;

y <- array(dim = c(I, K))
for (i in 1:I) {
  for (k in 1:K) {
    xi_i_k <- exp(alpha[i] + beta[k]) / (omega[k] - 1);
    y[i,k] <- rnbinom(1,
                      size = xi_i_k,
                      prob = 1 / omega[k]);
  }
}
```

where y_{ik} records the presence of a tie between the ith individual ($i = 1, ..., I$) and the kth subgroup ($k = 1, ..., K$); α_i represents the gregariousness or expected degree of the ith person andβ_k represents the prevalence or proportion of total links that involve group k, each on the log scale; and ω_k represents the amount of overdispersion, with higher values of ω_k corresponding to more variation in the distribution of connections involving group k than would be expected under the Poisson model and $\omega_k = 1$ corresponding to the null model (no overdispersion).

Moving forward, we will only consider subgroups with variation in the simulated survey questions. That is, if *all* (simulated) survey participants do or do not know anyone from the lth subgroup, the lth subgroup is dropped from the analysis. This is a reasonable practice in real world applications as well, since in such cases, we have no information about variability or overdispersion to leverage in our estimation of personal network size.

```
var0 <- apply(y, 1, var)
var0 <- which(var0 == 0)
if (length(var0) > 0) {
  y <- y[-var0,]
  alpha <- alpha[- var0]
  I <- nrow(y)
  }
```

Specify Hyper-Parameters

Since we are simulating data here, we can use the simulated parameters to inform our choice of hyper-parameters. This is an easy way to obtain appropriate prior distributions in this setting:

```
mu_beta <- c(beta[1:12], rep(mean(beta[1:12]), K - 12))
sigma_beta <- c(rep(.01, 12), rep(10, K - 12))
```

In practice, ...

Run the Model with Stan

First, we package the simulated data for use with Stan:

```
data <- list(I = nrow(y), K = ncol(y), mu_beta = mu_beta, sigma_beta = sigma_beta, y = y)
```

and use the following Stan code to specify the model:

```
stan_model = "
data {
  int<lower=0> I;                         // respondents
  int<lower=0> K;                         // subpopulations
  vector[K] mu_beta;                      // prior mean of beta
  vector<lower=0>[K] sigma_beta;          // prior variance of beta
  int  y[I,K];                            // known by respondent i in subpopulation k
  }

parameters {
  vector[I] alpha;                        // log degree
  vector[K] beta;                         // log prevalence of group in population
  vector<lower = 0 , upper = 1>[K] inv_omega;  // ineverse overdispersion; implies the uniform prior
  real mu_alpha;                         // prior mean for alpha
  real<lower=0> sigma_alpha;             // prior scale for alpha
  }

model {
// priors
  alpha ~ normal(mu_alpha, sigma_alpha);
  beta ~ normal(mu_beta, sigma_beta);     // informative prior on beta: location and scale are identifi

// hyperpriors
  mu_alpha ~ normal(0,25);                // weakly informative (no prior in paper)
  sigma_alpha ~ normal(0,5);              // weakly informative (no prior in paper)


  for (k in 1:K) {
    real omega_k_m1;
    omega_k_m1 = inv(inv(inv_omega[k]) - 1) ;
    for (i in 1:I) {
      real xi_i_k;
      xi_i_k = omega_k_m1 * exp(alpha[i] + beta[k])  ;
      y[i,k] ~ neg_binomial(xi_i_k, omega_k_m1);
      }
    }
  }"
```

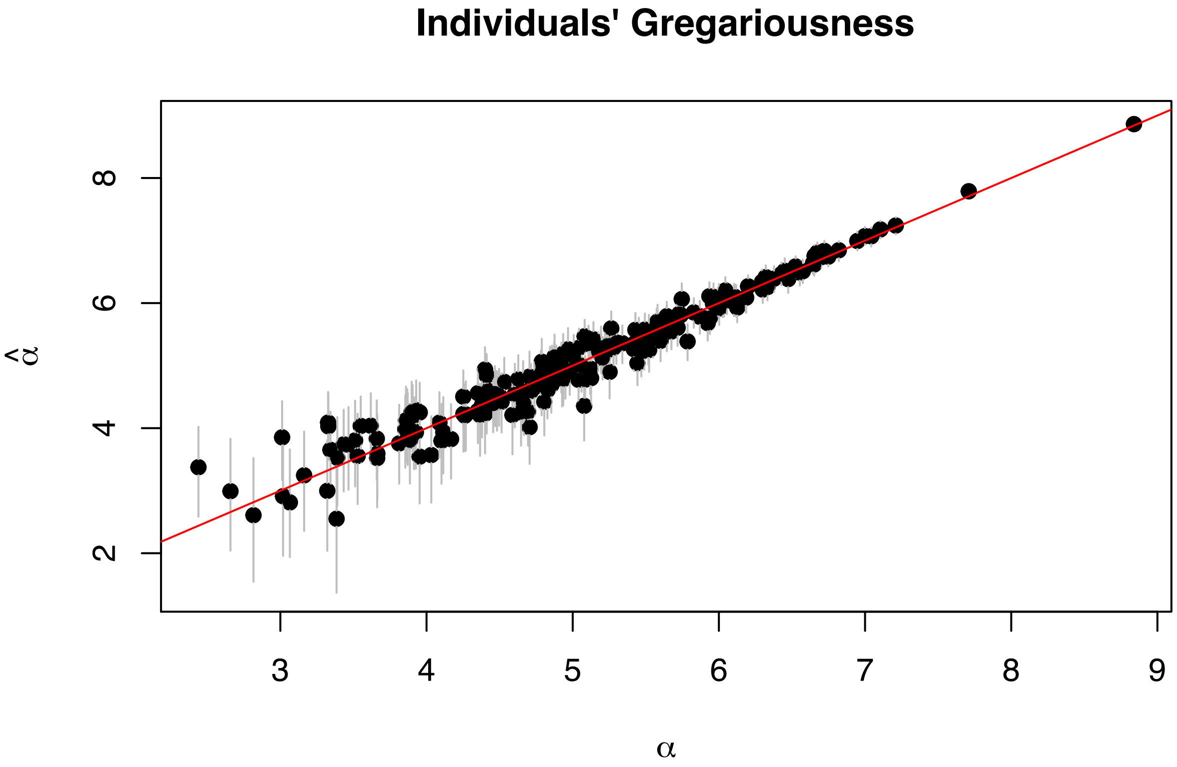
Finally, we run the Stan model in R and examine the results. The results are presented as the plots of simulated versus estimated values of model parameters.

```
library(rstan)
fit <- stan(model_code = stan_model, #file='NB_norecall.stan',
            data = data,
            # warmup = 1000, iter = 2000,  # takes ~ 40min/chain
            warmup = 50, iter=500,
            chains = 2)
```

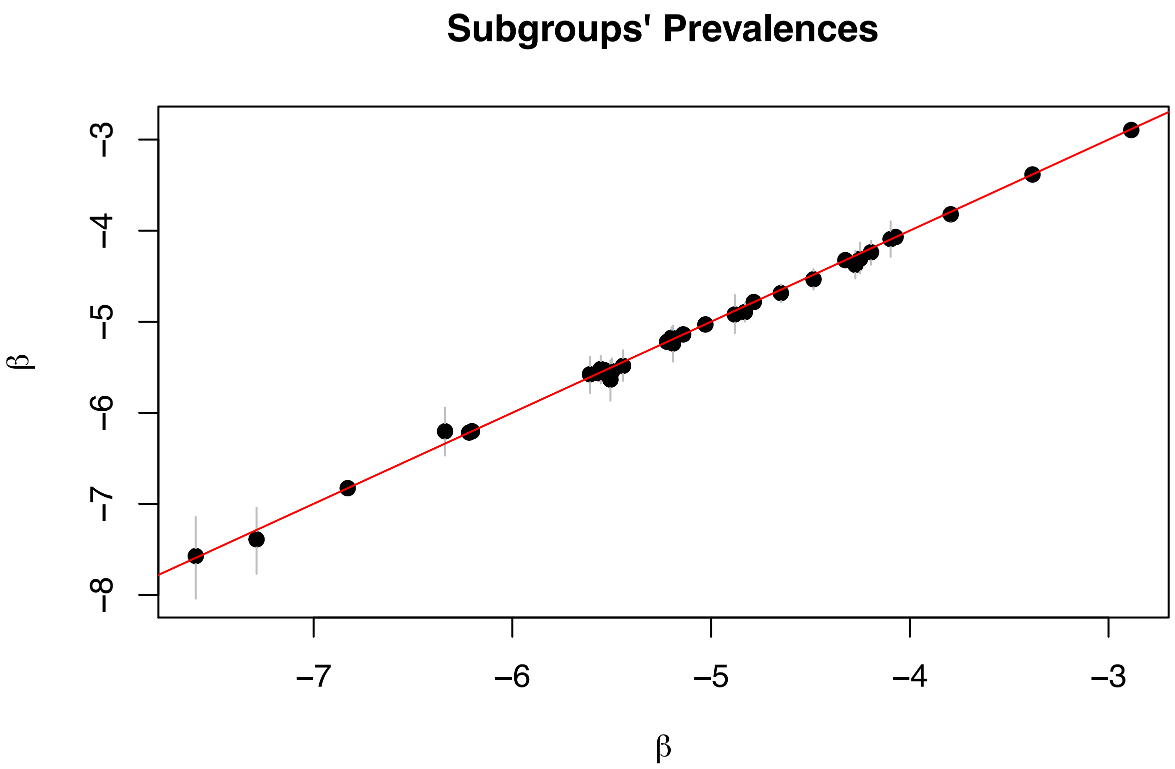
```
## In file included from file65f143a3ab8a.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boos
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boos
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boost/config/compiler/clang
## #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##           ^
## <command line>:6:9: note: previous definition is here
## #define BOOST_NO_CXX11_RVALUE_REFERENCES 1
##         ^
## In file included from file65f143a3ab8a.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/include/stan/math/rev/core
##     static void set_zero_all_adjoints() {
##                 ^
## In file included from file65f143a3ab8a.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/include/stan/math/rev/core
##     static void set_zero_all_adjoints_nested() {
##                 ^
## In file included from file65f143a3ab8a.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/include/stan/math/prim/mat
##       size_t fft_next_good_size(size_t N) {
##              ^
## In file included from file65f143a3ab8a.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boos
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boos
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boos
## In file included from /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boos
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boost/multi_array/concept_
##       typedef typename Array::index_range index_range;
##                                           ^
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boost/multi_array/concept_
##       typedef typename Array::index index;
##                                     ^
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boost/multi_array/concept_
##       typedef typename Array::index_range index_range;
##                                           ^
## /Library/Frameworks/R.framework/Versions/3.3/Resources/library/BH/include/boost/multi_array/concept_
##       typedef typename Array::index index;
##                                     ^
## 8 warnings generated.
##
## SAMPLING FOR MODEL 'cf3448a88f73a64b6b4d77baac5b9931' NOW (CHAIN 1).
## WARNING: The initial buffer, adaptation window, and terminal buffer
##          overflow the total number of warmup iterations.
##          Defaulting to a 15%/75%/10% partition,
##            init_buffer = 7
##            adapt_window = 38
##            term_buffer = 5
##
##
## Chain 1, Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 1, Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 1, Iteration:  51 / 500 [ 10%]  (Sampling)
## Chain 1, Iteration: 100 / 500 [ 20%]  (Sampling)
## Chain 1, Iteration: 150 / 500 [ 30%]  (Sampling)
## Chain 1, Iteration: 200 / 500 [ 40%]  (Sampling)
## Chain 1, Iteration: 250 / 500 [ 50%]  (Sampling)
## Chain 1, Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 1, Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 1, Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 1, Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 1, Iteration: 500 / 500 [100%]  (Sampling)
##  Elapsed Time: 12.5245 seconds (Warm-up)
##                158.76 seconds (Sampling)
##                171.284 seconds (Total)
##
##
## SAMPLING FOR MODEL 'cf3448a88f73a64b6b4d77baac5b9931' NOW (CHAIN 2).
## WARNING: The initial buffer, adaptation window, and terminal buffer
##          overflow the total number of warmup iterations.
##          Defaulting to a 15%/75%/10% partition,
##            init_buffer = 7
##            adapt_window = 38
##            term_buffer = 5
##
##
## Chain 2, Iteration:   1 / 500 [  0%]  (Warmup)
## Chain 2, Iteration:  50 / 500 [ 10%]  (Warmup)
## Chain 2, Iteration:  51 / 500 [ 10%]  (Sampling)
## Chain 2, Iteration: 100 / 500 [ 20%]  (Sampling)
## Chain 2, Iteration: 150 / 500 [ 30%]  (Sampling)
## Chain 2, Iteration: 200 / 500 [ 40%]  (Sampling)
## Chain 2, Iteration: 250 / 500 [ 50%]  (Sampling)
## Chain 2, Iteration: 300 / 500 [ 60%]  (Sampling)
## Chain 2, Iteration: 350 / 500 [ 70%]  (Sampling)
## Chain 2, Iteration: 400 / 500 [ 80%]  (Sampling)
## Chain 2, Iteration: 450 / 500 [ 90%]  (Sampling)
## Chain 2, Iteration: 500 / 500 [100%]  (Sampling)
##  Elapsed Time: 11.4715 seconds (Warm-up)
##                149.933 seconds (Sampling)
##                161.404 seconds (Total)
```

```r
sims <- extract(fit, permuted = FALSE, inc_warmup = TRUE)
check <- monitor(sims, warmup = floor(dim(sims)[1]/2),
                 probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
                 digits_summary = 1, print = F)


### extract the results and compute summaries of the posterior ###
out <- extract(fit)
alpha_post <- out $ alpha
alpha_hat <- apply(alpha_post, 2, mean)
alpha_lower <- apply(alpha_post, 2, quantile, p = .025)
alpha_upper <- apply(alpha_post, 2, quantile, p = .975)
beta_post <- out $ beta
```

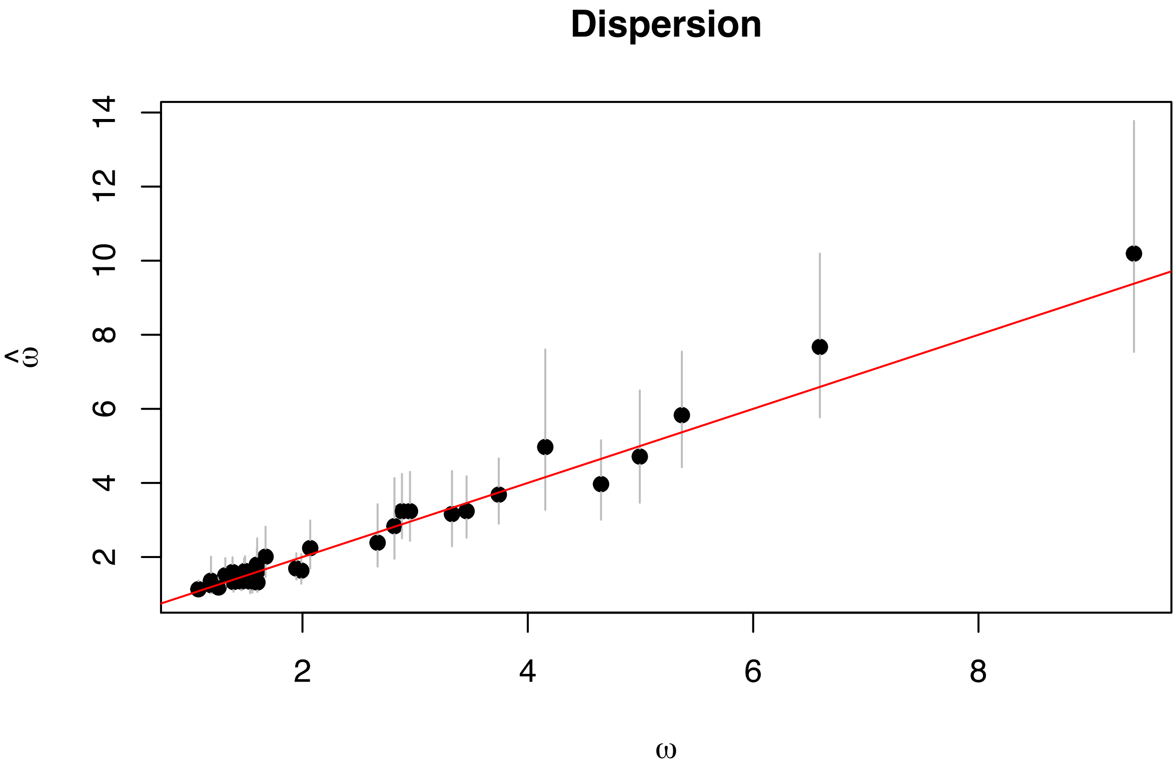
```
beta_hat <- apply(beta_post, 2, mean)
beta_lower <- apply(beta_post, 2, quantile, p = .025)
beta_upper <- apply(beta_post, 2, quantile, p = .975)
omega_post <- 1 / out $ inv_omega
omega_hat <- apply(omega_post, 2, mean)
omega_lower <- apply(omega_post, 2, quantile, p = .025)
omega_upper <- apply(omega_post, 2, quantile, p = .975)

### generate the posterior check plots ###

plot(alpha, alpha_hat, main="Individuals' Gregariousness",
     ylim=c(min(alpha_lower),max(alpha_upper)), xlab = expression(alpha),
     ylab = expression(hat(alpha)))
for (i in 1:I) lines(c(alpha[i], alpha[i]), c(alpha_lower[i], alpha_upper[i]), col="grey")
points(alpha, alpha_hat, pch=16)
abline(0, 1, col = "red")
```



```
plot(beta, beta_hat, main="Subgroups' Prevalences",
     ylim=c(min(beta_lower),max(beta_upper)), xlab = expression(beta),
     ylab = expression(hat(beta)))
for (k in 1:K) lines(c(beta[k], beta[k]), c(beta_lower[k], beta_upper[k]), col="grey")
points(beta, beta_hat, pch=16)
abline(0, 1, col = "red")
```



```
plot(omega, omega_hat,main="Dispersion",
     ylim=c(min(omega_lower),max(omega_upper)), xlab = expression(omega),
     ylab = expression(hat(omega)))
for (k in 1:K) lines(c(omega[k], omega[k]), c(omega_lower[k], omega_upper[k]), col="grey")
points(omega, omega_hat, pch=16)
abline(0, 1, col = "red")
```


Dispersion
$\hat{\omega}$
2
4
6
8
10
12
14
2
4
6
8
ω