Small team statistics

George G. Vega Yon
Kayla de la Haye

October 1, 2018

Simulation process

For each set of experiments, generate N teams by doing:

1. Draw a random graph of size n_i from a bernoulli distribution with parameter p_i, call it G_i.
2. Generate n_i other graphs by permuting G_i with different levels of accuracy a_{ij}
3. Generate $Y_i \sim \text{Beta}\left(\exp\left(\theta^{\text{t}} X_i\right), 1.5\right)$, where X_i is a vector of team level statistics, including $a_i = n_i^{-1}\sum_j a_{ij}$, the average level of accuracy. The resulting value Y_i will be between 0 and 1.

Once all N teams have been simulated, estimate the model using MLE

R Markdown

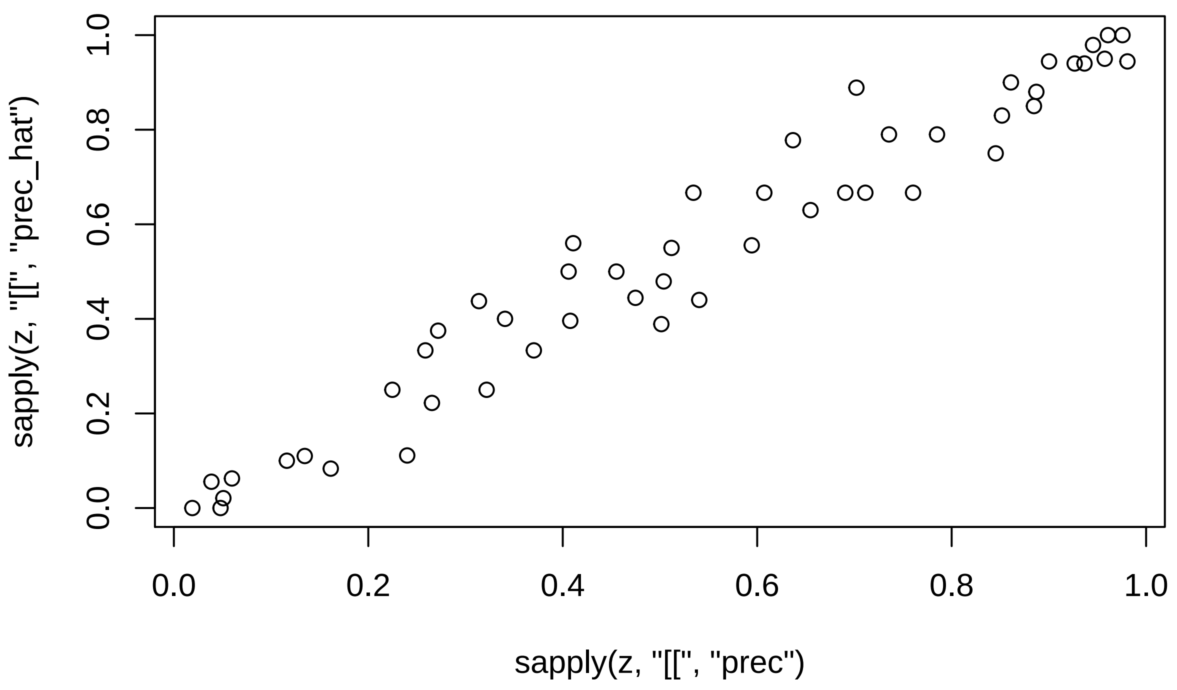
```r
library(magrittr)
library(stats4)

set.seed(65454)

source("beta_mle.R")

n_sims  <- 1e3
n_teams <- 50
n       <- replicate(n_sims, sample(c(3, 4, 5), n_teams, TRUE), simplify = FALSE)
dens    <- replicate(n_sims, runif(n_teams), simplify = FALSE)
prec    <- replicate(n_sims, runif(n_teams), simplify = FALSE)

theta   <- replicate(n_sims, rnorm(3), simplify=FALSE)
X       <- lapply(n, function(n0) {
  cbind(
    X1 = rbinom(n_teams, n0, .5)/n0,
    X2          = rnorm(n_teams)
    )
  })
```

```r
# Simple example
z <- sim_experiment(
  n     = n[[2]],
  dens  = dens[[2]],
  prec  = prec[[2]],
  X     = X[[2]],
  theta = c(-2, 1, .5)
  )

# Extracting the data
```

```r
d <- cbind(
  h = sapply(z, "[[", "prec_hat"),
  y = sapply(z, "[[", "response"),
  X = X[[1]]
)

ans <- beta_mle(d[,"y"], d[,-2])
summary(ans)
```

```
## Maximum likelihood estimation
##
## Call:
## beta_mle(Y = d[, "y"], X = d[, -2])
##
## Coefficients:
##         Estimate Std. Error
## beta -0.03923851  0.1979583
## h    -1.38747778  0.3598211
## X1   -0.25359711  0.4252156
## X2    0.08206155  0.1328339
##
## -2 log L: 78.67721
```

```r
# Correlation
plot(
  sapply(z, "[[", "prec"),
  sapply(z, "[[", "prec_hat")
)
```



```r
ans <- parallel::mcmapply(
  sim_experiment,
  n = n, dens=dens, prec=prec, X=X, theta=theta, mc.cores = 8,
  SIMPLIFY = FALSE
  )
```

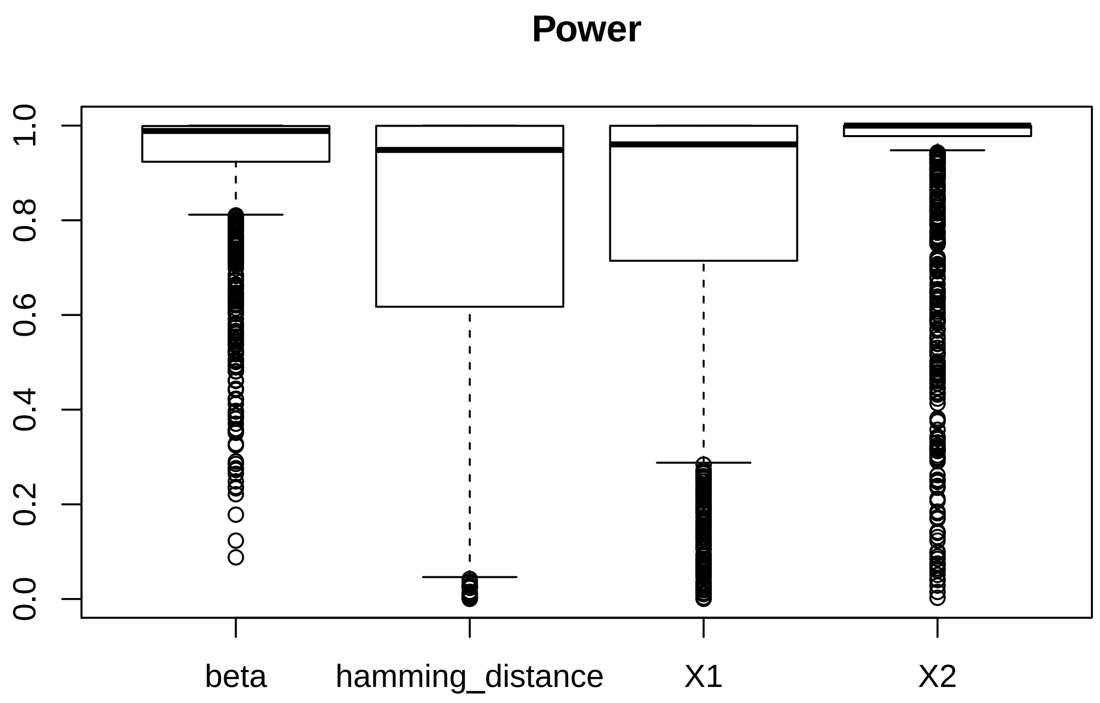
```
# Estimating models ------------------------------------------------------------
mles0 <- parallel::mcmapply(function(dat, x) {

  # Extracting the data
  d <- cbind(
    y = sapply(dat, "[[", "response"),
    hamming_distance = sapply(dat, "[[", "prec_hat"),
    X = x
  )

  beta_mle(d[,"y"], d[,-1])

}, dat = ans, x=X, mc.cores=8, SIMPLIFY=FALSE)

mles1 <- parallel::mcmapply(function(dat, x) {

  # Extracting the data
  d <- cbind(
    y = sapply(dat, "[[", "response"),
    group_size = sapply(dat, "[[", "n"),
    X = x
  )

  beta_mle(d[,"y"], d[,-1])

}, dat = ans, x=X, mc.cores=8, SIMPLIFY=FALSE)


mles2 <- parallel::mcmapply(function(dat, x) {

  # Extracting the data
  d <- cbind(
    y = sapply(dat, "[[", "response"),
    hamming_distance = sapply(dat, "[[", "prec_hat"),
    group_size = sapply(dat, "[[", "n"),
    X = x
  )

  beta_mle(d[,"y"], d[,-1])

}, dat = ans, x=X, mc.cores=8, SIMPLIFY=FALSE)
```

```
## Warning in mclapply(seq_len(n), do_one, mc.preschedule = mc.preschedule, :
## scheduled cores 8 encountered errors in user code, all values of the jobs
## will be affected
```

```
# Computing pvalues ------------------------------------------------------------

pvals0 <- lapply(mles0, function(model) {
  tryCatch(calc_pval(model@coef, model@vcov), error =function(e) NULL)
  })
```

```
## Warning in sqrt(diag(information)): NaNs produced
## Warning in sqrt(diag(information)): NaNs produced
```

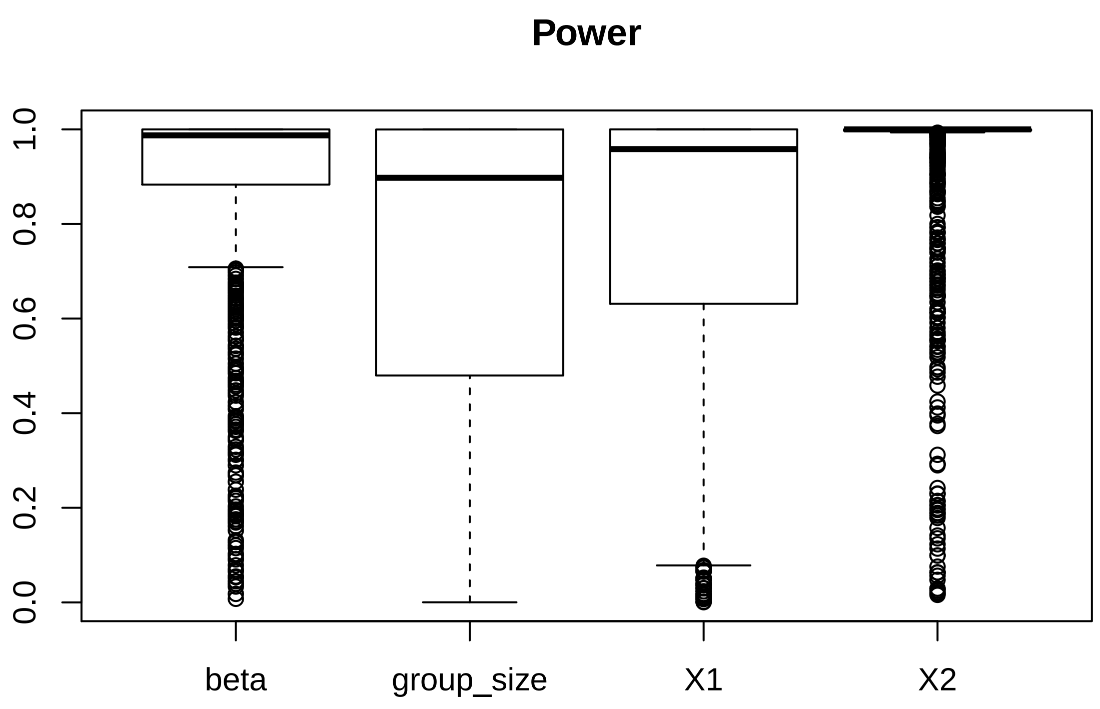
```
pvals0 <- do.call(rbind, pvals0)
boxplot(1 - pvals0, main="Power")
```



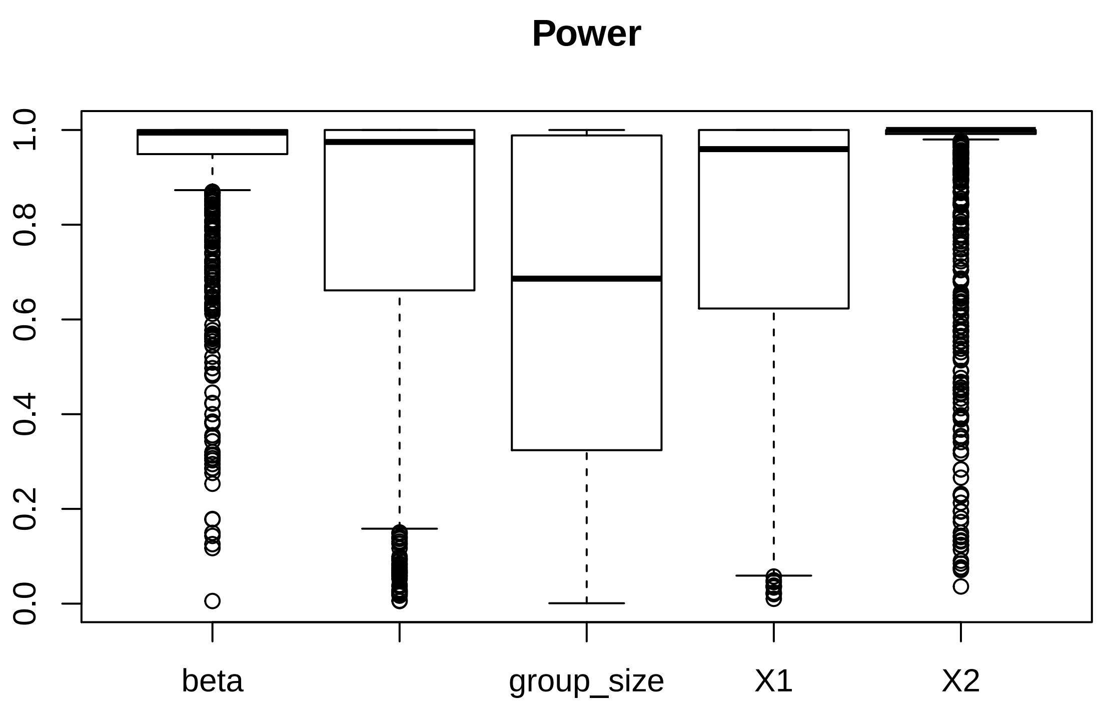
```
pvals1 <- lapply(mles1, function(model) {
  tryCatch(calc_pval(model@coef, model@vcov), error =function(e) NULL)
  })
```

```
## Warning in sqrt(diag(information)): NaNs produced

## Warning in sqrt(diag(information)): NaNs produced

## Warning in sqrt(diag(information)): NaNs produced

## Warning in sqrt(diag(information)): NaNs produced
```

```
pvals1 <- do.call(rbind, pvals1)
boxplot(1 - pvals1, main="Power")
```



```
pvals2 <- lapply(mles2, function(model) {
  tryCatch(calc_pval(model@coef, model@vcov), error =function(e) NULL)
  })
pvals2 <- do.call(rbind, pvals2)
boxplot(1 - pvals2, main="Power")
```


Power
beta
group_size
X1
X2