HW12

In Ericson et al. (2015), the authors propose a new heuristic model for intertemporal choice and compare it against several existing models in a large 940-subject data set using cross validation. However, in order to do this they fit the parameters of each model to 75% of the entire dataset and then use that point estimate try to predict the other 25%. However, in order to do this, they throw out all of the individual subject information! This is presumably because each subject only answered at most 25 intertemporal choice questions so there isn't enough data to estimate each subject's parameters very well. But by getting a single set of parameters for all of the subjects, the authors are totally mischaracterizing these models. There is no normative, biological or psychological reason for every member of the population to have the same parameters. The fact that the authors' favorite model has the best cross validation score could potentially be only due to the best parameters for that model having lower variance across the population. It seems like what the authors really wanted to do was estimate each subjects' parameters individually, using partial pooling to reduce the variance in those estimates, and then do cross validation. So I here take the authors' dataset, fit several models in Stan and use the recent LOO-CV approximation from Vehtari et al. (2015) to efficiently do model comparison.

First get the data into a usable form:

```r
library("rstan")
library("boot")
library("ggplot2")
library("loo")
library("rstan")
library("boot")
library("ggplot2")
setwd("~/Documents/BDA/Homework 12")
data <- read.csv("choices.csv", stringsAsFactors = FALSE)

# Drop rows with missing responses
data <- subset(data, !is.na(LaterOptionChosen))

condition.number = 0
# Select the subset of data for this condition, where 0 is pooled over all.
if (condition.number != 0) {
  data <- subset(data, Condition == condition.number)
} else {
  data <- data
}

# Add additional features required by ITCH model
data <- transform(data, XStar=(X1 + X2) / 2)
data <- transform(data, TStar=(T1 + T2) / 2)
data <- transform(data, G=scale(X2 - XStar))
data <- transform(data, R=scale((X2 - X1) / XStar))
data <- transform(data, D=scale(T2 - TStar))
data <- transform(data, T=scale((T2 - T1) / TStar))

# Add additional features required by DRIFT model
data <- transform(data, DriftD=scale(X2 - X1))
data <- transform(data, DriftR=scale((X2 - X1) / X1))
data <- transform(data, DriftI=scale((X2 / X1)^(1 / (T2 - T1)) - 1))
data <- transform(data, DriftT=scale(T2 - T1))
```

```r
# Rescale data when working with "raw" numbers
data <- transform(data, X1=X1 / max(X2))
data <- transform(data, X2=X2 / max(X2))

it_data <- data
#it_data <- data[data$Subject %in% 1:50,]

G <- drop(it_data$G)
R <- drop(it_data$R)
D <- drop(it_data$D)
T <- it_data$T
LaterOptionChosen <- it_data$LaterOptionChosen
Subject <- as.numeric(as.factor(it_data$Subject))
Condition <- it_data$Condition
n_subjects <- nlevels(as.factor(Subject))
n_data <- nrow(it_data)
conds <- rep(NA, n_subjects)
for(d in 1:n_data) {
  conds[Subject[d]] <- Condition[d]
}
n_conds <- nlevels(as.factor(conds))
X1 <- it_data$X1
X2 <- it_data$X2
T1 <- it_data$T1
T2 <- it_data$T2
epsilon <- 0.01
```

We now compare the following three models:

ITCH (Ericson et al. 2015): $P(LL) = \beta_I + \beta_G(x_2 - x_1) + \beta_R\frac{(x_2-x_1)}{x^*} + \beta_D(t_2 - t_1) + \beta_T\frac{(t_2-t_1)}{t^*}$

Exponential: $P(LL) = L(a(x_2\delta^{t_2} - x_1\delta^{t_2}))$

Hyperbolic: $P(LL) = L(a(x_2(1+kt_2)^{-1} - x_1(1+kt_1)^{-1}))$

where x_i is amount of utility (money) of prize i and t_i is the time at which prize i is received.

In order to estimate the ITCH model, we put a prior over the betas for each subject so that , $\beta_s \sim N(\mu, \sigma)$ with $\mu \sim N(0, 100)$ and $\sigma \sim \text{Unif}(0, 100)$. The exponential model has priors $a \sim N(\mu_a, \sigma_a)$, $\delta \sim \text{Beta}(\mu_d, \kappa_d)$ (all Beta distributions are parameterized with $\mu = \frac{\alpha}{\alpha+\beta}$ and $\kappa = \alpha + \beta$) and hyperpriors $\mu_d \sim \text{Beta}(1, 1)$, $\kappa_d \sim \text{Unif}(0, 100)$, $\mu_a \sim \text{Unif}(0.0001, 500)$ and $\sigma_a \sim \text{Unif}(0.001, 100)$. The hyperbolic model is the same with k parameterized the same way as δ in the exponential model. These are clearly not very informative priors but I chose them initially to be as similar as possible to the analysis in the paper where they used maximum likelihood to estimate the paramters. The ranges over δ and k for example are the same as given to the optimization function in the authors' code.

And now the models in Stan:

```stan
data {
  int<lower=0> n_subjects;
  int<lower=0> n_data;
  int<lower=0,upper=1> LaterOptionChosen[n_data];
  real G[n_data];
  real R[n_data];
  real D[n_data];
  real T[n_data];
  int Subject[n_data];
}
parameters {
  vector[5] mu;
  vector<lower=0>[5] sigma;
  vector[5] beta[n_subjects];
}
transformed parameters {
  real p[n_data];
  for (d in 1:n_data) {
    p[d] <- beta[Subject[d], 1] + beta[Subject[d], 2] * G[d] + beta[Subject[d], 3] * R[d] + beta[Subjec
  }
}
model {
  mu ~ normal(0, 100);
  sigma ~ uniform(0, 100);
  for (s in 1:n_subjects)
    beta[s] ~ normal(mu, sigma);
  for (d in 1:n_data)
    LaterOptionChosen[d] ~ bernoulli_logit(p[d]);
}
generated quantities {
  vector[n_data] log_lik;

  for (d in 1:n_data)
    log_lik[d] <- bernoulli_logit_log(LaterOptionChosen[d], p[d]);
}


data {
  int<lower=0> n_subjects;
  int<lower=0> n_data;
  int<lower=0,upper=1> LaterOptionChosen[n_data];
  real X1[n_data];
  real X2[n_data];
  real T1[n_data];
  real T2[n_data];
  int Subject[n_data];
}
parameters {
  real<lower=0, upper=1> mu_d;
  real<lower=0> kappa_d;
  real mu_a;
  real<lower=0> sigma_a;
  real<lower=0, upper=1> delta[n_subjects];
  real a[n_subjects];
}
transformed parameters {
  real p[n_data];
  real z[n_data];
  real alpha_d;
  real beta_d;
  for (d in 1:n_data) {
    z[d] <- X2[d] * delta[Subject[d]]^T2[d] - X1[d] * delta[Subject[d]]^T1[d];
    //homothetic:
```

```
    //z[d] <- log(z[d]);
    p[d] <- inv_logit(a[Subject[d]] * z[d]);
    //bounded:
    //p[d] <- epsilon * 0.5 + (1 - epsilon) * p[d];
  }
  alpha_d <- mu_d * kappa_d;
  beta_d <- kappa_d - alpha_d;
}
model {
  mu_d ~ uniform(0, 1);
  kappa_d ~ uniform(0, 100);
  mu_a ~ uniform(0.0001, 500);
  sigma_a ~ uniform(0.001, 100);
  for (s in 1:n_subjects) {
    delta[s] ~ beta(alpha_d, beta_d);
    a[s] ~ normal(mu_a, sigma_a);
  }
  for (d in 1:n_data)
    LaterOptionChosen[d] ~ bernoulli(p[d]);
}
generated quantities {
  vector[n_data] log_lik;

  for (d in 1:n_data)
    log_lik[d] <- bernoulli_log(LaterOptionChosen[d], p[d]);
}


data {
  int<lower=0> n_subjects;
  int<lower=0> n_data;
  int<lower=0,upper=1> LaterOptionChosen[n_data];
  real X1[n_data];
  real X2[n_data];
  real T1[n_data];
  real T2[n_data];
  int Subject[n_data];
}
parameters {
  real<lower=0, upper=1> mu_k;
  real<lower=0> kappa_k;
  real mu_a;
  real<lower=0> sigma_a;
  real<lower=0, upper=1> k[n_subjects];
  real a[n_subjects];
}
transformed parameters {
  real p[n_data];
  real z[n_data];
  real alpha_k;
  real beta_k;
  for (d in 1:n_data) {
    z[d] <- X2[d] * (1.0 / (1.0 + k[Subject[d]] * T2[d])) - X1[d] * (1.0 / (1.0 + k[Subject[d]] * T1[d]
    //homothetic:
    //z[d] <- log(z[d]);
    p[d] <- inv_logit(a[Subject[d]] * z[d]);
    //bounded:
    //p[d] <- epsilon * 0.5 + (1 - epsilon) * p[d];
  }
  alpha_k <- mu_k * kappa_k;
  beta_k <- kappa_k - alpha_k;
}
model {
  mu_k ~ uniform(0, 1);
  kappa_k ~ uniform(0, 100);
  mu_a ~ uniform(0.0001, 500);
  sigma_a ~ uniform(0.001, 100);
  for (s in 1:n_subjects) {
    k[s] ~ beta(alpha_k, beta_k);
    a[s] ~ normal(mu_a, sigma_a);
  }
  for (d in 1:n_data)
    LaterOptionChosen[d] ~ bernoulli(p[d]);
}
generated quantities {
  vector[n_data] log_lik;

  for (d in 1:n_data)
    log_lik[d] <- bernoulli_log(LaterOptionChosen[d], p[d]);
}
```

And now fit the models and get parameter estimates:

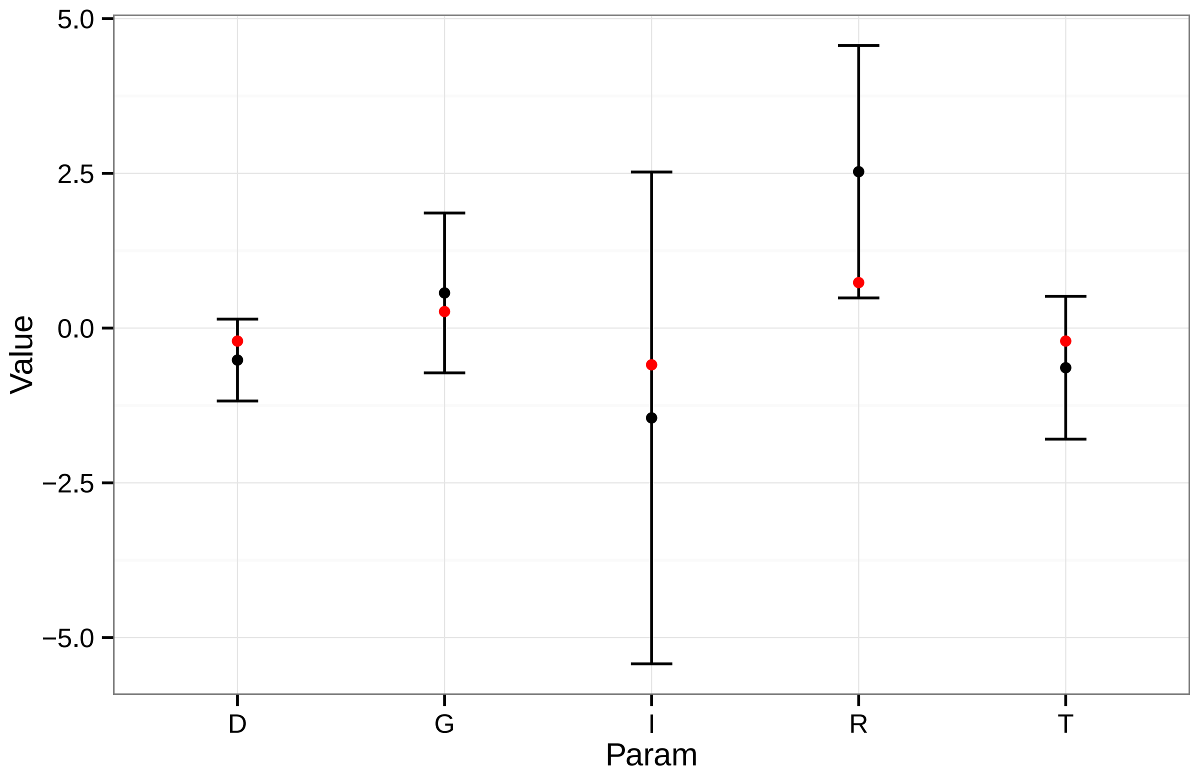
```r
fit_itch <- stan("itch.stan")
m_itch <- as.data.frame(monitor(fit_itch))
fit_exp <- stan("exp.stan")
m_exp <- as.data.frame(monitor(fit_exp))
fit_hyp <- stan("hyp.stan")
m_hyp <- as.data.frame(monitor(fit_hyp))
```

We'll compare the parameter estimates to the paremeters as estimated in the original paper in order to make sure our models are giving us something reasonable:

```r
exponential.log.likelihood.function <- function(data, theta, epsilon) {
  a <- theta[1]
  delta <- theta[2]

  z <- with(data, X2 * delta^T2 - X1 * delta^T1)
  p <- inv.logit(a * z)
  p <- epsilon * 0.5 + (1 - epsilon) * p

  l <- ifelse(data$LaterOptionChosen == 1, p, 1 - p)
  ll <- sum(log(l))

  return(ll)
}

exponential.fit.function <- function(data, epsilon) {
  # a in [0.0001, 500.00]
  # delta in [0.01, 0.99]
  f <- function (theta) {
    exponential.log.likelihood.function(data, theta, epsilon)
  }

  results <- optim(
    c(1.0, 0.5),
    f,
    method="L-BFGS-B",
    lower=c(0.0000001, 0.01),
    upper=c(500.00, 0.99),
    control=list(fnscale=-1)
  )

  if (results$convergence != 0) {
    warning("Fitting function failed to converge")
  }

  return(results$par)
}

hyperbolic.log.likelihood.function <- function(data, theta, epsilon) {
    a <- theta[1]
    k <- theta[2]

    z <- with(data, X2 * (1.0 / (1.0 + k * T2)) - X1 * (1.0 / (1.0 + k * T1)))
    p <- inv.logit(a * z)
    p <- epsilon * 0.5 + (1 - epsilon) * p

    l <- ifelse(data$LaterOptionChosen == 1, p, 1 - p)
    ll <- sum(log(l))

    return(ll)
}

hyperbolic.fit.function <- function(data, epsilon) {
    # a in [0.0001, 500.00]
    # k in [0.0001, 100.00]
    f <- function (theta) {
        hyperbolic.log.likelihood.function(data, theta, epsilon)
    }

    results <- optim(
        c(0.001, 0.5),
        f,
        method="L-BFGS-B",
        lower=c(0.0000001, 0.0001),
        upper=c(500.00, 100.00),
        control=list(fnscale=-1)
    )

    if (results$convergence != 0) {
        warning("Fitting function failed to converge")
    }

    return(results$par)
}
```

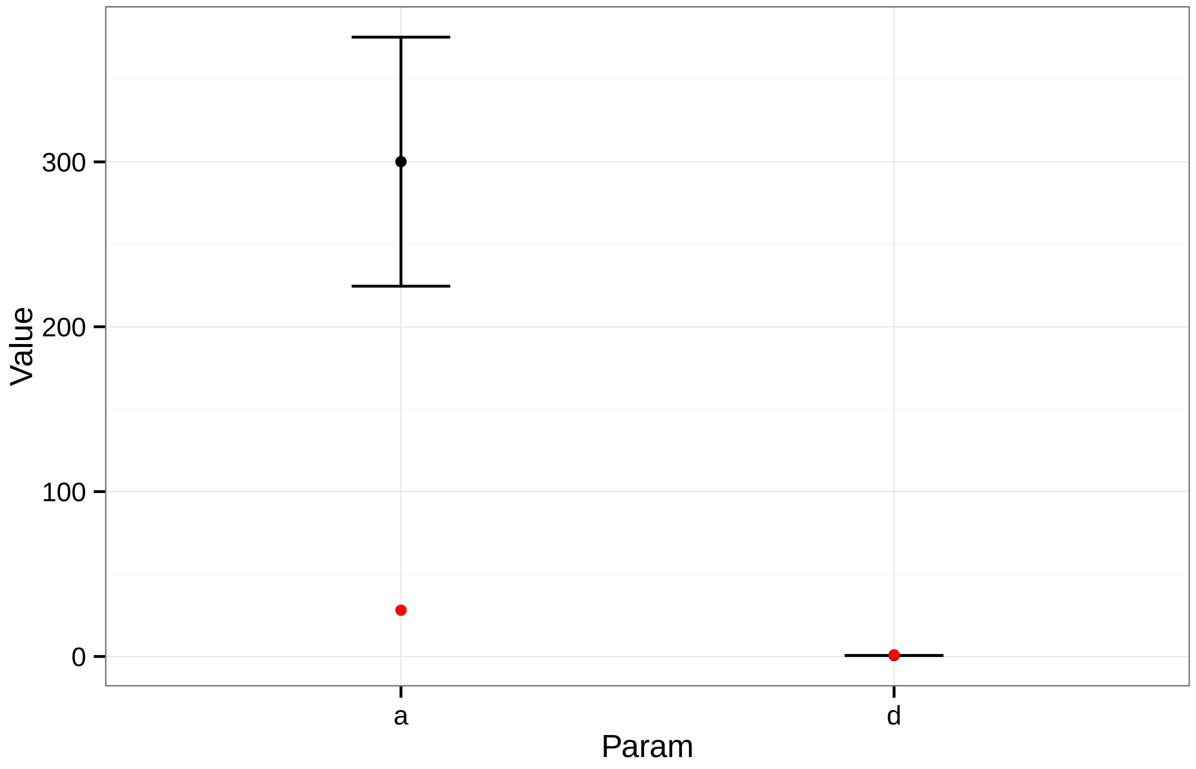
Here I comepare the parameter estimates from the paper to the mean and standard deviation of the population distribution over parameters. The black dot and bars represent the mean and spread of the population distribution from the Bayesian model for each paramter and the maximum likelihood estimate is in red. Here is the plot for the ITCH model:

```
itch_glm <- glm(LaterOptionChosen ~ G + R + D + T, data=it_data, family=binomial(link="logit"))
mu <- m_itch[0:5, c("mean")]
sigma <- m_itch[6:10, c("mean")]
mle_params <- itch_glm$coefficients
limits <- aes(ymax=bayes_params + bayes_sigma, ymin=bayes_params - bayes_sigma)
df <- data.frame(
  mle_params <- mle_params,
  bayes_params <- mu,
  bayes_sigma <- sigma,
  params <- c("I", "G", "R", "D", "T")
)
p <- ggplot(df, aes(x=params, y=bayes_params))
p + geom_point() +
  geom_errorbar(limits, width=0.2) +
  geom_point(y=mle_params, color="red") +
  xlab("Param") +
  ylab("Value") +
  ggtitle("") +
  theme_bw()
```



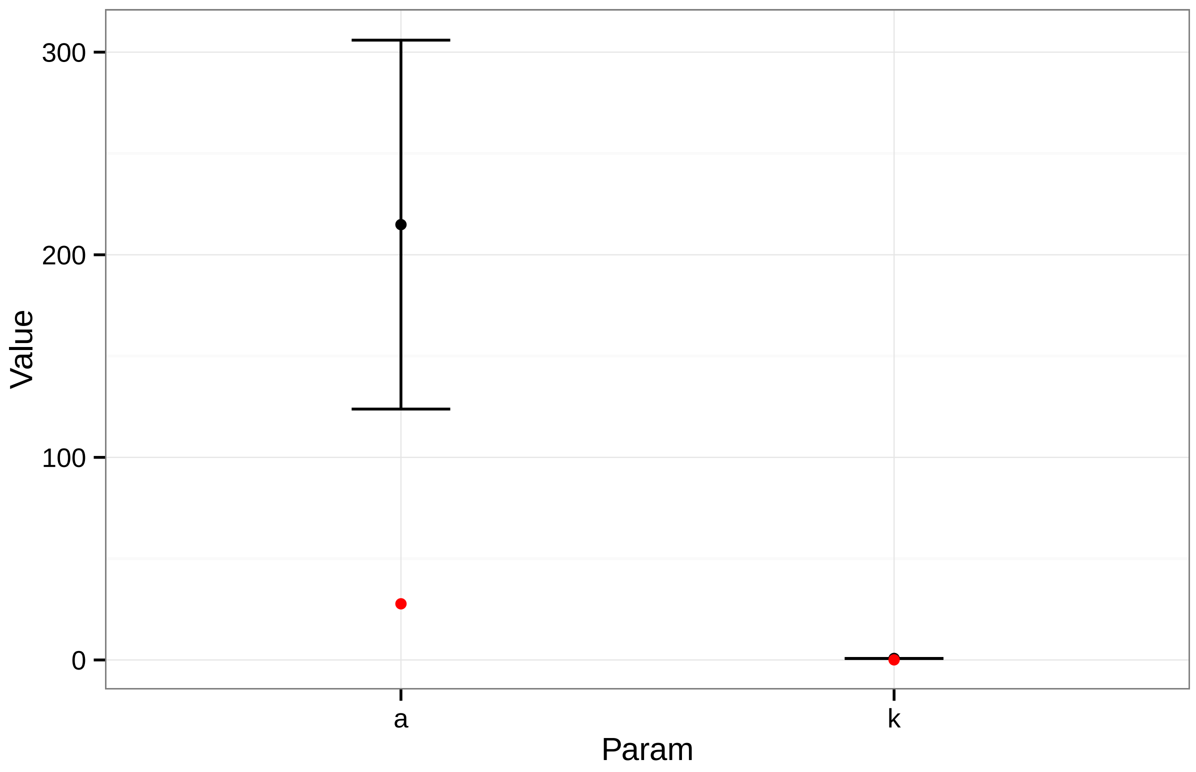
And here is the same plot for the exponential model:

```
mu <- m_exp[1, "mean"]
kappa <- m_exp[2, "mean"]
alpha <- mu * kappa;
beta <- kappa - alpha;
var <- (alpha * beta)/((alpha + beta)^2 * (alpha + beta + 1))
mle_params <- exponential.fit.function(it_data, epsilon)
bayes_params <- c(m_exp[3, "mean"], m_exp[1, "mean"])
bayes_sigma <- c(m_exp[4, "mean"], var)
limits <- aes(ymax=bayes_params + bayes_sigma, ymin=bayes_params - bayes_sigma)
df <- data.frame(
  mle_params <- mle_params,
  bayes_params <- bayes_params,
  bayes_sigma <- bayes_sigma,
  params <- c("a", "d")
)
p <- ggplot(df, aes(x=params, y=bayes_params))
p + geom_point() +
  geom_errorbar(limits, width=0.2) +
  geom_point(y=mle_params, color="red") +
  xlab("Param") +
  ylab("Value") +
  ggtitle("") +
  theme_bw()
```



And here is the same plot for the hyperbolic model:

```
mu <- m_hyp[1, "mean"]
kappa <- m_hyp[2, "mean"]
alpha <- mu * kappa;
beta <- kappa - alpha;
var <- (alpha * beta)/((alpha + beta)^2 * (alpha + beta + 1))
mle_params <- hyperbolic.fit.function(it_data, epsilon)
bayes_params <- c(m_hyp[3, "mean"], m_hyp[1, "mean"])
bayes_sigma <- c(m_hyp[4, "mean"], 1/m_hyp[2, "mean"])
limits <- aes(ymax=bayes_params + bayes_sigma, ymin=bayes_params - bayes_sigma)
df <- data.frame(
  mle_params <- mle_params,
  bayes_params <- bayes_params,
  bayes_sigma <- bayes_sigma,
  params <- c("a", "k")
)
p <- ggplot(df, aes(x=params, y=bayes_params))
p + geom_point() +
  geom_errorbar(limits, width=0.2) +
  geom_point(y=mle_params, color="red") +
  xlab("Param") +
  ylab("Value") +
  ggtitle("") +
  theme_bw()
```

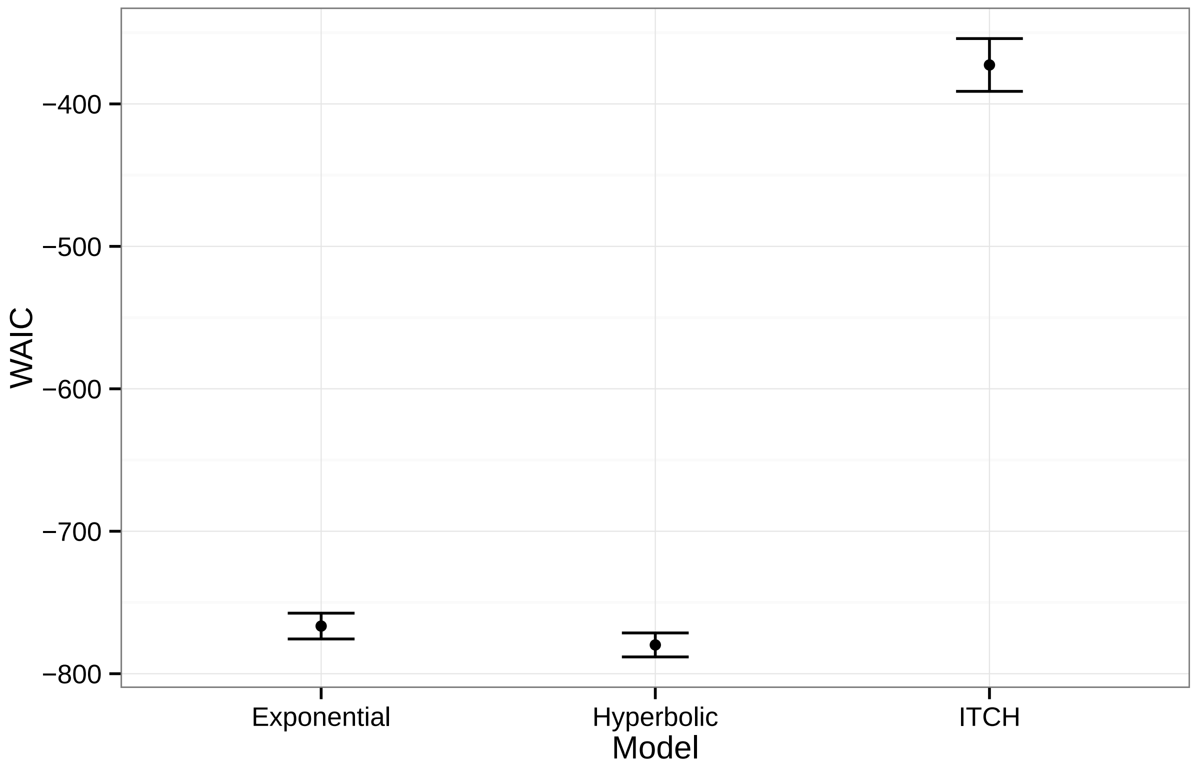


And now we get WAIC and LOO-CV scores:

```r
log_lik_itch <- extract_log_lik(fit_itch)
log_lik_exp <- extract_log_lik(fit_exp)
log_lik_hyp <- extract_log_lik(fit_hyp)
loo_exp <- loo(log_lik_exp)
loo_hyp <- loo(log_lik_hyp)
loo_itch <- loo(log_lik_itch)
waic_exp <- waic(log_lik_exp)
waic_hyp <- waic(log_lik_hyp)
waic_itch <- waic(log_lik_itch)
```

There was a small percentage of the data for which the shape parameter of the Pareto distribution for the LOO approximation was greater than 1 so I decided to just use WAIC to compare the models. Here I plot the expected log posterior densities from the WAIC:

```r
df <- data.frame(
  waics <- c(waic_exp$elpd_waic, waic_hyp$elpd_waic, waic_itch$elpd_waic),
  se_waics <- c(waic_exp$se_elpd_waic, waic_hyp$se_elpd_waic, waic_itch$se_elpd_waic),
  models <- c("Exponential", "Hyperbolic", "ITCH")
)

limits <- aes(ymax = waics + se_waics, ymin=waics - se_waics)
p <- ggplot(df, aes(x=models, y=waics))
p + geom_point() +
  geom_errorbar(limits, width=0.2) +
  xlab("Model") +
  ylab("WAIC") +
  ggtitle("") +
  theme_bw()
```



So while it seems that ITCH is still the better fit to the data at least now we can be confident that we gave the existing models a reasonable shot without requiring low population variance in parameters (which is not discussed at all by the theory).