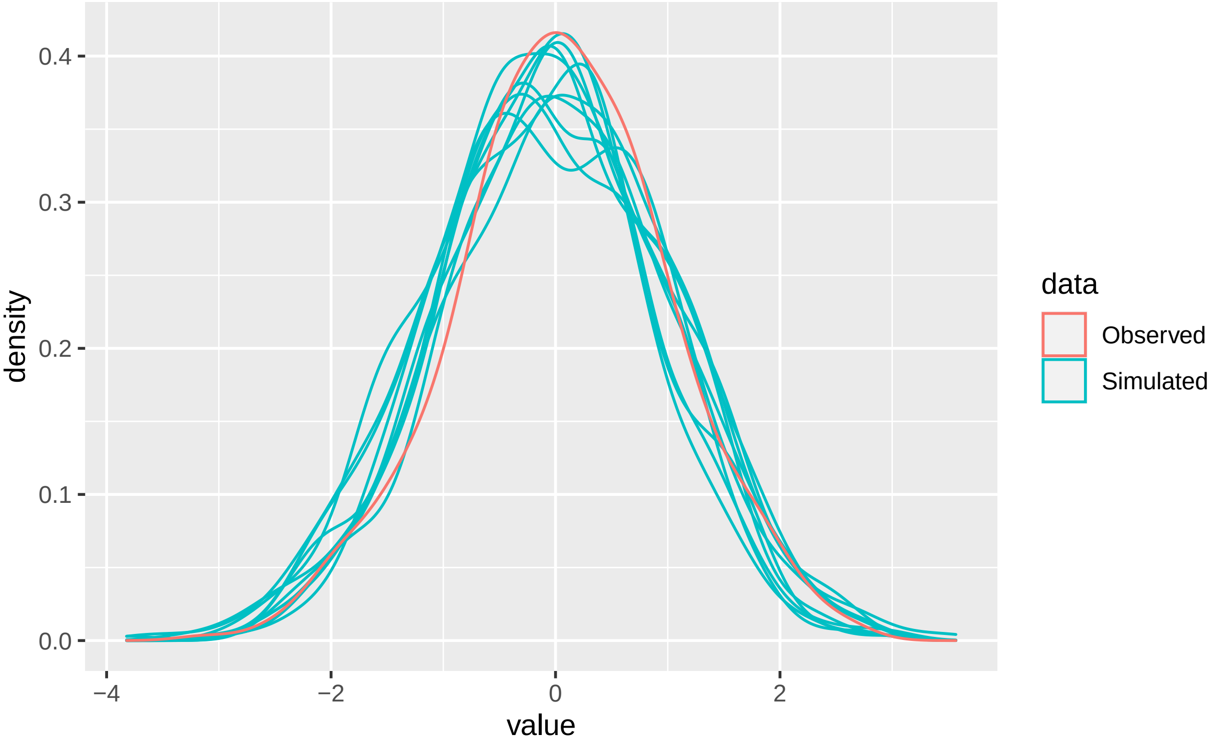
PPC and Production-Level Tables

Graham Tierney

9/7/2020

Posterior Predictive Checks

PPC visualize "typical" data distributions given your parameter estimates, then compare them to your observed distribution. See Gelman, Meng, and Stern (1996): https://www.jstor.org/stable/24306036.



Why do PPC?

- Show if your modeling assumption is good (not just if MCMC has converged)
- Check tail behavior

How to do them?

1) Take some posterior draws of your parameters (NOT the posterior means)
2) Sample from p(Y|X,parameters), one draw per observation
3) Plot densities or histograms of the simulated AND obsereved values

Example

```
n <- 500
x1 <- rnorm(n)
x2 <- rbinom(n,10,prob = .1)
y <- 3 + 2*x1 + -1*x2 + rnorm(n)*(1/rgamma(n,3/2,3/2)) #t distribution w/ df=3

jags_model <- function(){
  for(i in 1:n){
    mu[i] = inprod(beta,X[i,])
    resid[i] = Y[i]-mu[i]
    Y[i] ~ dnorm(mu[i],tau)
  }
  tau ~ dgamma(1/10,1/10)
  sigma = pow(tau,-1/2)

  for(i in 1:3){
    beta[i] ~ dnorm(0,.01)
  }
}
X <- cbind(1,x1,x2)
jags_output <- jags(data = list(Y=y,X=X,n=n),model.file = jags_model,
                    parameters.to.save = c("mu","sigma","beta","resid"))
```

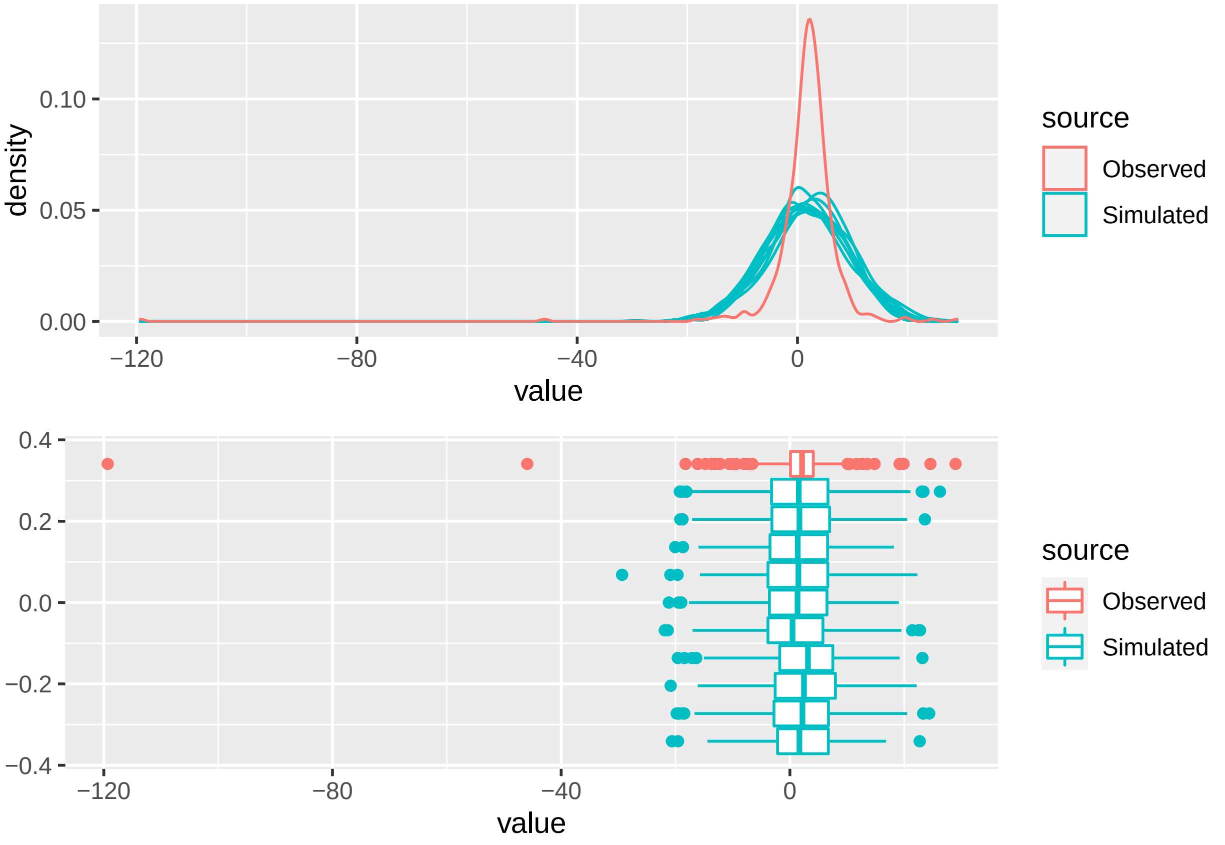
Lets see if we got reasonable point estimates for the parameters.

```
jags_output$BUGSoutput$sims.matrix[,c(str_c("beta[",1:3,"]"),"sigma")] %>%
  as_tibble() %>%
  reshape2::melt(id.vars = c()) %>%
  mutate(Variable = case_when(variable == "beta[1]" ~ "Intercept",
                              variable == "beta[2]" ~ "x1",
                              variable == "beta[3]" ~ "x2",
                              variable == "sigma" ~ "sigma")) %>%
  group_by(Variable) %>%
  summarise(Mean = mean(value),SD = sd(value),
            `2.5% Quantile` = quantile(value,.025),
            `97.5% Quantile` = quantile(value,.975),.groups = "drop") %>%
  kable(format = "latex",digits = 3) %>%
  kableExtra::kable_styling()
```

Variable	Mean	SD	2.5% Quantile	97.5% Quantile
Intercept	2.927	0.443	2.068	3.780
sigma	6.899	0.216	6.485	7.339
x1	2.248	0.311	1.628	2.861
x2	-1.256	0.323	-1.872	-0.617

```
#set # of ppc draws and randomly pick parameter draws
nppc_sims <- 10
ppc_param_draws <- sample(1:nrow(jags_output$BUGSoutput$sims.matrix),replace = F,size = nppc_sims)

#extract simulated values
mu_sims <- jags_output$BUGSoutput$sims.matrix[,c(str_c("mu[",1:n,"]"))]
sigma_sims <- jags_output$BUGSoutput$sims.matrix[,c(str_c("sigma"))]
```

```
#compute posterior predictive draws
ppc_sims <- sapply(ppc_param_draws,function(r) rnorm(n=length(y),mean = mu_sims[r,],sd = sigma_sims[r]))
ppc_plot_data <- cbind(ppc_sims,y) %>%
  as_tibble() %>%
  reshape2::melt() %>%
  mutate(source = ifelse(variable == "y","Observed","Simulated"))

ppc_density <- ppc_plot_data %>%
  ggplot(aes(x=value,group = variable,color = source)) +
  geom_density()

ppc_boxplot <- cbind(ppc_sims,y) %>%
  as_tibble() %>%
  reshape2::melt() %>%
  mutate(source = ifelse(variable == "y","Observed","Simulated")) %>%
  ggplot(aes(x=value,group = variable,color = source)) +
  geom_boxplot()

gridExtra::grid.arrange(ppc_density,ppc_boxplot)
```



We want to see that the observed plots are indistinguishable from the simulated plots. Obviously, thats not quite the case here. So lets give the data likelihood slightly fatter tails by making it a t-distribution and put a prior on the degrees of freedom.

```
jags_model_t <- function(){
  for(i in 1:n){
    mu[i] = inprod(beta,X[i,])
    resid[i] = Y[i]-mu[i]
    gamma[i] ~ dgamma(alpha/2,alpha/2)
    Y[i] ~ dnorm(mu[i],tau*gamma[i])
  }

  tau ~ dgamma(1/10,1/10)
  sigma = pow(tau,-1/2)

  alpha = 3

  for(i in 1:3){
    beta[i] ~ dnorm(0,.01)
  }
}
jags_output_t <- jags(data = list(Y=y,X=X,n=n),model.file = jags_model_t,
                    parameters.to.save = c("mu","sigma","beta","resid","alpha"))
```

Now lets check the estimated model parameters.

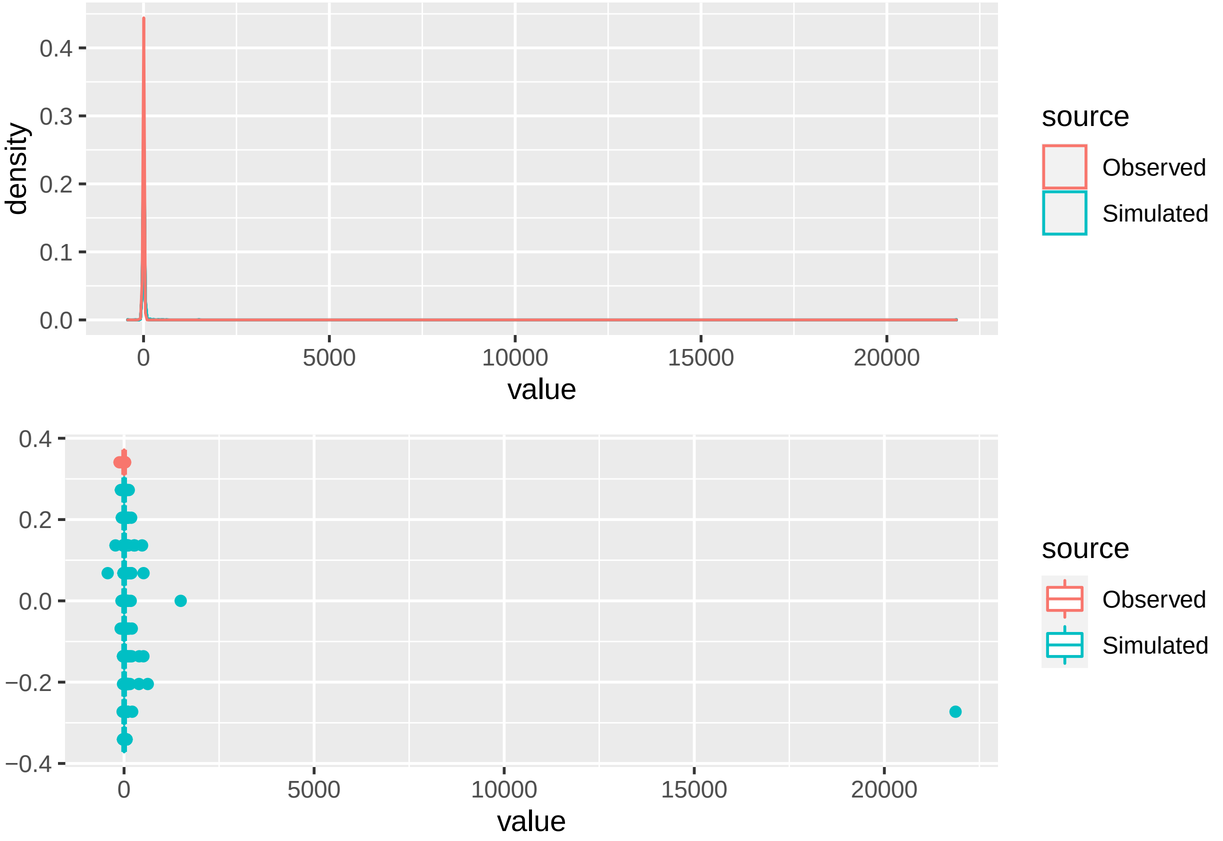
```
jags_output_t$BUGSoutput$sims.matrix[,c(str_c("beta[",1:3,"]"),"sigma","alpha")] %>%
  as_tibble() %>%
  reshape2::melt(id.vars = c()) %>%
  mutate(Variable = case_when(variable == "beta[1]" ~ "Intercept",
                              variable == "beta[2]" ~ "x1",
                              variable == "beta[3]" ~ "x2",
                              variable == "sigma" ~ "sigma",
                              variable == "alpha" ~ "alpha")) %>%
  group_by(Variable) %>%
  summarise(Mean = mean(value),SD = sd(value),
            `2.5% Quantile` = quantile(value,.025),
            `97.5% Quantile` = quantile(value,.975),.groups = "drop") %>%
  kable(format = "latex",digits = 3) %>%
  kableExtra::kable_styling()
```

Variable	Mean	SD	2.5% Quantile	97.5% Quantile
alpha	3.000	0.000	3.000	3.000
Intercept	3.123	0.110	2.909	3.336
sigma	1.390	0.073	1.256	1.538
x1	2.112	0.073	1.976	2.263
x2	-1.034	0.078	-1.183	-0.884

Now recreat the PPC checks.

```
#set # of ppc draws and randomly pick parameter draws
nppc_sims <- 10
ppc_param_draws <- sample(1:nrow(jags_output_t$BUGSoutput$sims.matrix),replace = F,size = nppc_sims)

#extract simulated values
mu_sims <- jags_output_t$BUGSoutput$sims.matrix[,c(str_c("mu[",1:n,"]"))]
sigma_sims <- jags_output_t$BUGSoutput$sims.matrix[,c(str_c("sigma"))]
alpha_sims <- jags_output_t$BUGSoutput$sims.matrix[,c(str_c("alpha"))]
```

```
#compute posterior predictive draws
ppc_sims <- sapply(ppc_param_draws,function(r) rnorm(n=length(y),
                                 mean = mu_sims[r,],
                                 sd = sigma_sims[r])/
                     rgamma(length(y),mean(alpha_sims)/2,mean(alpha_sims)/2))
ppc_plot_data <- cbind(ppc_sims,y) %>%
  as_tibble() %>%
  reshape2::melt() %>%
  mutate(source = ifelse(variable == "y","Observed","Simulated"))

ppc_density <- ppc_plot_data %>%
  ggplot(aes(x=value,group = variable,color = source)) +
  geom_density()

ppc_boxplot <- cbind(ppc_sims,y) %>%
  as_tibble() %>%
  reshape2::melt() %>%
  mutate(source = ifelse(variable == "y","Observed","Simulated")) %>%
  ggplot(aes(x=value,group = variable,color = source)) +
  geom_boxplot()

gridExtra::grid.arrange(ppc_density,ppc_boxplot)
```


density
value
source
Observed
Simulated
value
source
Observed
Simulated