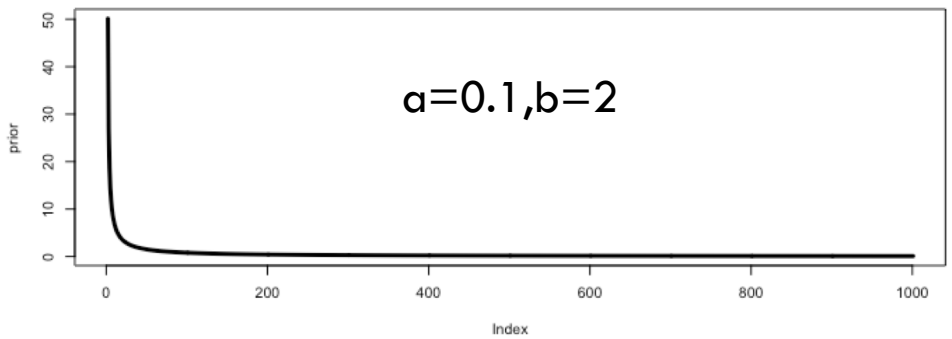
認知情報解析学演習
JAGS

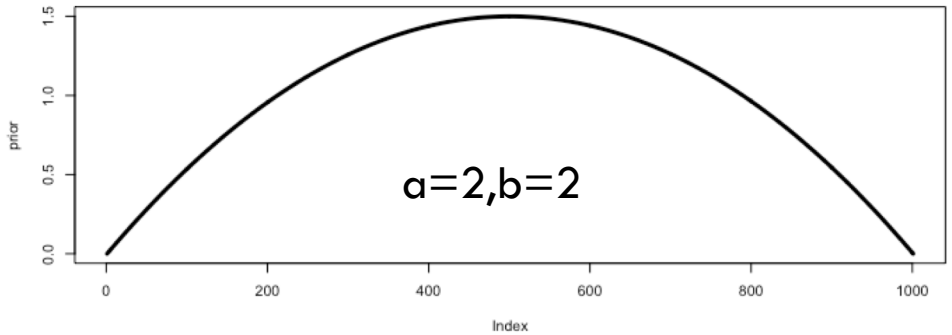
ベータ分布(PRIOR)

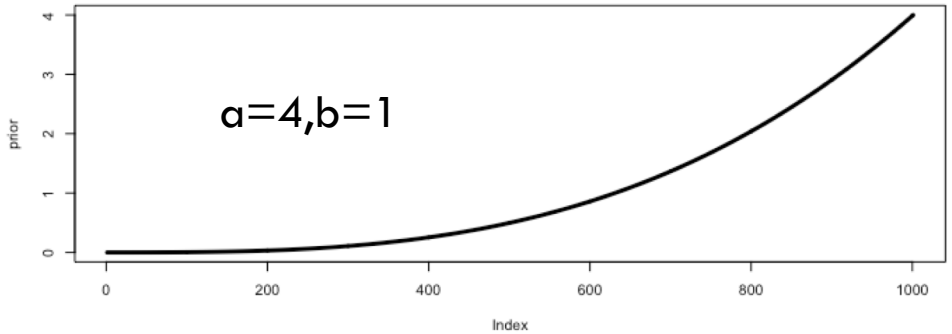
$$p(\theta|a,b) = beta(\theta|a,b)$$

$$= \frac{\theta^{(a-1)}(1-\theta)^{(b-1)}}{B(a,b)}$$

$$B(a,b) = \int_0^1 \theta^{(a-1)}(1-\theta)^{(b-1)} d\theta$$







ベータ分布

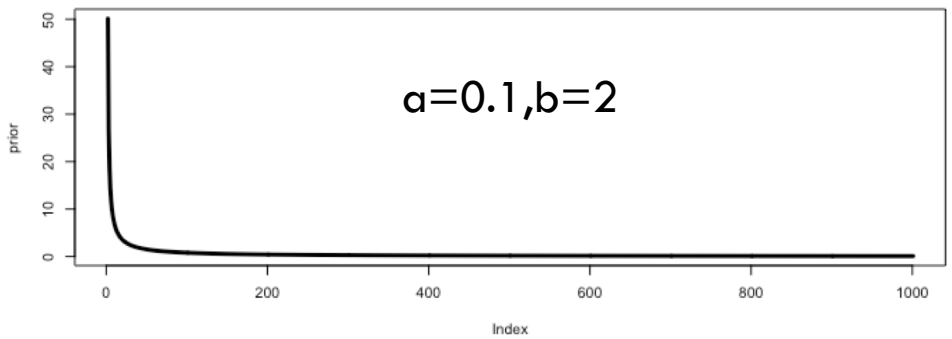
$$beta(\theta|a,b) = \frac{\theta^{(a-1)}(1-\theta)^{(b-1)}}{B(a,b)}$$

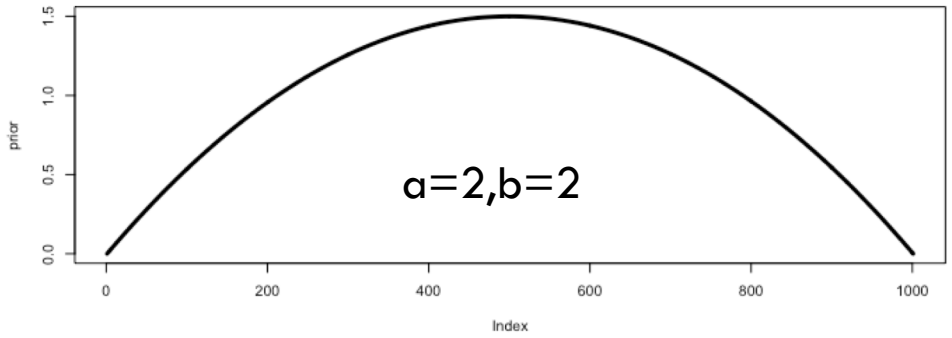
Mean: $\mu = \frac{a}{a+b}$

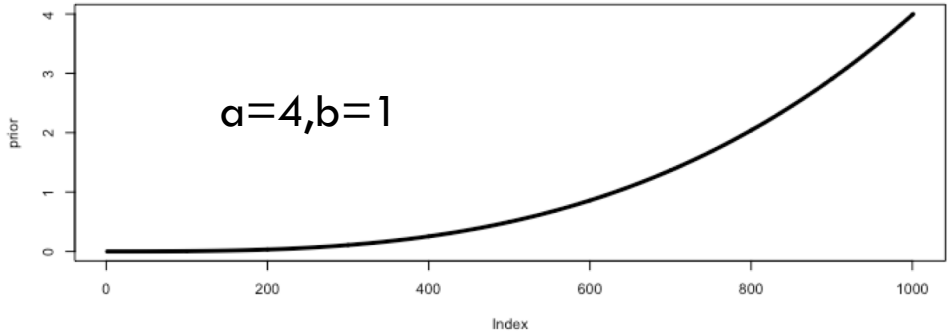
Mode: $\omega = \frac{a-1}{a+b-2}$

Concentration: $\kappa = a + b$

$$a = \mu\kappa; b = (1-\mu)\kappa$$
$$a = \omega(\kappa-2)+1; b = (1-\omega)(\kappa-2)+1$$







JAGS (JUST ANOTHER GS)

- Data はリスト形式
- Modelはテキスト（外部）ファイル
- Jags
  - n.chains: 独立したサンプルの数
  - n.adapt: サンプラーのパラメターを自動調整に使用するサンプル数
  - N.iter: iterationの回数
  - Thining: サンプルの記録周期（何個飛ばしで記録するか）
  - Update: サンプルの空焼き
  - 手続き
    - Data -> model -> jags.model -> update -> sampling

EXAMPLE

Model:

```
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta )
  }
  theta ~ dbeta( 1, 1 )
}
```

$$y_i \sim bern(\theta)$$
$$\theta \sim beta(a = 1, b = 1)$$

EXAMPLE

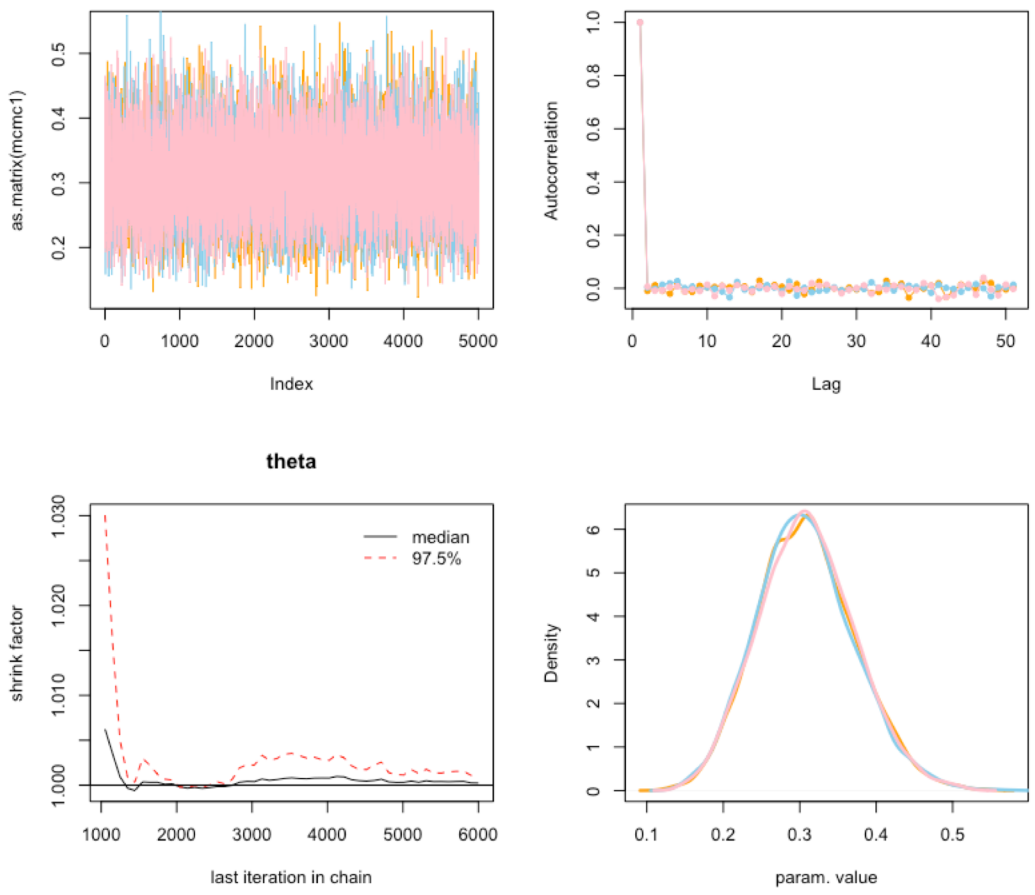
```
# data preparation
y = sample(c(rep(1,15), rep(0,35)))
Ntotal=length(dat$y)
datalist = list(y=y,Ntotal=Ntotal)

# model
txt = "
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta )
  }
  theta ~ dbeta( 1, 1 )
}"
writeLines(txt, "~/model.txt")
```

EXAMPLE

```
# jags
library(rjags)
jagsModel = jags.model(file="~/model.txt",
                       data=datalist,n.chains=3,n.adapt=500)
update(jagsModel,n.iter=1000)
codaSamples=coda.samples(jagsModel,variable.names=c("theta"),n.iter=5000)
mcmcMat<-as.matrix(codaSamples)
# checking MCMC
HDI.plot(mcmcMat)
traceplot(codaSamples)
autocorr.plot(codaSamples,type='l')
gelman.plot(codaSamples)
```

CHECKING MCMC



EXAMPLE2: 2 DIFF. THETA

```
# model 8.4
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta[s[i_data]] )
  }
  for ( i_s in 1:Nsubj) {
    theta[i_s] ~ dbeta( 2, 2 )
  }
}

# model 8.2
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta)
  }
  for ( i_s in 1:Nsubj) {
    theta ~ dbeta( 1, 1 )
  }
}
```

EXAMPLE2: 2 DIFF. THETA

```
# model 8.4
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta[s[i_data]] )
  }
  for ( i_s in 1:Nsubj) {
    theta[i_s] ~ dbeta( 2, 2 )
  }
}
```

$$y_i \sim bern(\theta_s)$$
$$\theta_s \sim beta(a = 2, b = 2)$$

```
# model 8.2
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta)
  }
  for ( i_s in 1:Nsubj) {
    theta ~ dbeta( 2, 2 )
  }
}
```

$$y_i \sim bern(\theta)$$
$$\theta \sim beta(a = 1, b = 1)$$

EXAMPLE

```
# data preparation
y1 = sample(c(rep(1,6), rep(0,2)))
y2 = sample(c(rep(1,2), rep(0,5)))
y = c(y1,y2)
s = c(rep(1,8),rep(2,7))
Ntotal=length(dat$y)
datalist = list(y = y, Ntotal = Ntotal, s = s)
```

EXAMPLE

```
# model
txt = "
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta[s[i_data]] )
  }
  for ( i_s in 1:Nsubj) {
    theta[i_s] ~ dbeta( 2, 2 )
  }
}"
writeLines(txt, "~/model.txt")
```

EXAMPLE

```
# jags
jagsModel = jags.model(file="~/model.txt",
                       data=datalist,n.chains=3,n.adapt=500)
update(jagsModel,n.iter=1000)
codaSamples=coda.samples(jagsModel,variable.names=c("theta"),n.iter=5000)
mcmcMat<-as.matrix(codaSamples)
# checking MCMC
HDI.plot(mcmcMat)
traceplot(codaSamples)
autocorr.plot(codaSamples,type='l')
gelman.plot(codaSamples)
```

ベータ分布

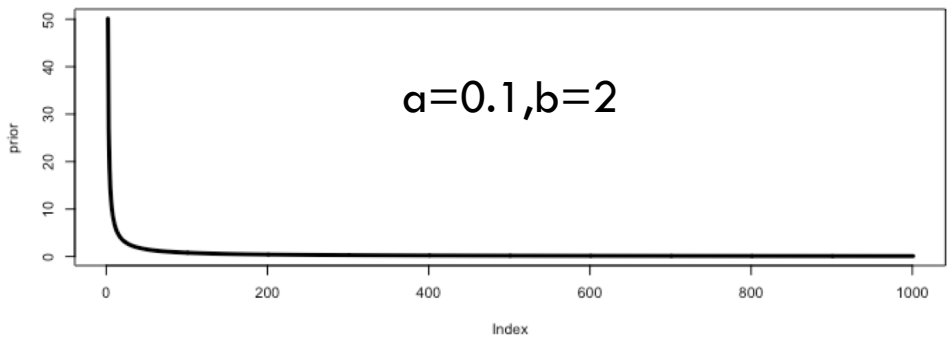
$$beta(\theta|a,b) = \frac{\theta^{(a-1)}(1-\theta)^{(b-1)}}{B(a,b)}$$

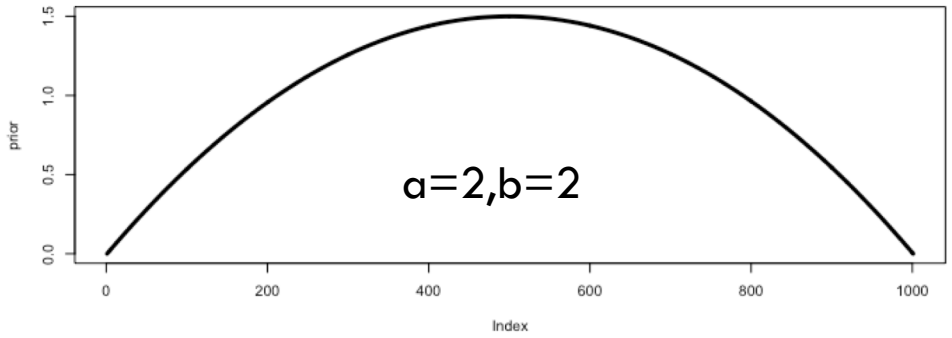
Mean: $\mu = \frac{a}{a+b}$

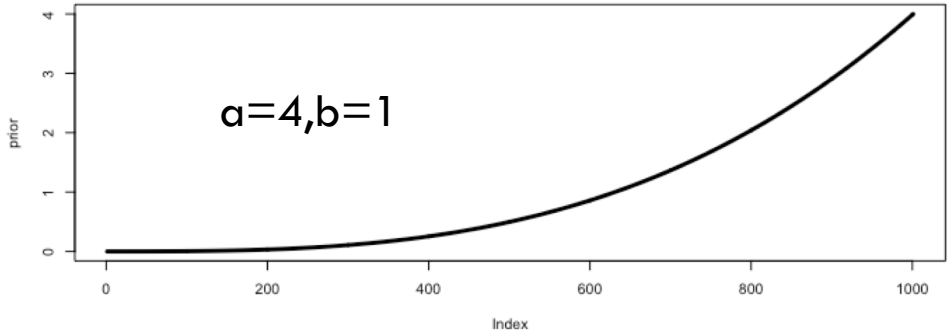
Mode: $\omega = \frac{a-1}{a+b-2}$

Concentration: $\kappa = a + b$

$$a = \mu\kappa; b = (1-\mu)\kappa$$
$$a = \omega(\kappa-2)+1; b = (1-\omega)(\kappa-2)+1$$







階層的モデル

Y~ Bern(theta)

theta ~ beta(omega(Kappa-2)+1, (1-omega)(Kappa-2)+1)

$y_i \sim bern(\theta)$

$\theta \sim beta(a, b) = beta(\omega(\kappa - 2) + 1, (1 - \omega)(\kappa - 2) + 1)$

#omega ~ beta(a,b)

GAMMA分布

$$\Gamma(k|a,b) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x)$$

$\alpha > 0$: shape; $\beta > 0$: rate

Mean: $\mu = \frac{\alpha}{\beta}$

Mode: $\omega = \frac{\alpha-1}{\beta}$

Var: $\sigma = \frac{\sqrt{\alpha}}{\beta}$

$$\alpha = \frac{\mu^2}{\sigma^2}; \beta = \frac{\mu}{\sigma^2}$$

$$\alpha = 1 + \omega\beta; \beta = \frac{\omega + \sqrt{\omega^2 + 4\sigma^2}}{2\sigma^2}$$

階層的モデル

Y~ Bern(theta)

theta ~ beta(omega(Kappa-2)+1, (1-omega)(Kappa-2)+1)

omega ~ beta(a,b)

Kappa ~ gamma(S, R)

$$
\begin{gathered}
y_i \sim bern(\theta_s) \\
\theta_s \sim beta(\omega(\kappa - 2) + 1, (1 - \omega)(\kappa - 2) + 1) \\
\omega \sim beta(a, b) \\
\kappa \sim gamma(s, r)
\end{gathered}
$$

階層的モデル

```
txt = "
model {
  for ( i_data in 1:Ntotal ) {
    y[ i_data ] ~ dbern( theta[s[i_data]] )
  }
  for ( i_s in 1:Nsubj) {
    theta[i_s] ~ dbeta( omega*(kappa-2)+1,(1-omega)*(kappa-2)+1)
  }
  omega ~ dbeta(1,1)
  kappa <- kappaMinusTwo + 2
  kappaMinusTwo ~ dgamma(0.01, 0.01)
}”
writeLines(txt, "~/model.txt")
```

階層的モデル

```
dat<-read.csv(" http://www.matsuka.info/data_folder/TherapeuticTouchData.csv")
y=dat$y
s=as.numeric(dat$s)
Ntotal=length(dat$y)
Nsubj=length(unique(s))
datalist = list(y=y,s=s,Ntotal=Ntotal,Nsubj=Nsubj)


jagsModel = jags.model(file="~/model.txt",data=datalist,n.chains=3,n.adapt=500)
update(jagsModel,n.iter=1000)
codaSamples=coda.samples(jagsModel,variable.names=c("theta","omega","kappa"),n.iter
=5000)
mcmcMat<-as.matrix(codaSamples)
```

階層的モデル

Y~ Bern(theta)
theta ~ beta(omega.c(Kappa.c-2)+1, (1-omega.c)(Kappa.c-2)+1)
omega.c ~ beta(omega(Kappa-c)+1,(1-omega)(Kappa-2)+1)
Kappa.c ~ gamma(S, R)
omega ~ beta(a,b)
Kappa ~ gamma(S,R)

$y_i \sim bern(\theta_s)$

$\theta_s \sim beta(\omega_c(\kappa_c - 2) + 1, (1 - \omega_c)(\kappa_c - 2) + 1)$

$\omega_c \sim beta(\omega(\kappa - 2) + 1, (1 - \omega)(\kappa - 2) + 1)$

$\kappa_c \sim gamma(s, r)$

$\omega \sim beta(a, b)$

$\kappa \sim gamma\ (s, r)$

階層的モデル

```
model {
 for (i_s in 1:Nsubj) {
   z[i_s] ~ dbin( theta[i_s], N[i_s])
   theta[i_s] ~ dbeta(omega[c[i_s]]*(kappa[c[i_s]]-2)+1,
      (1-omega[c[i_s]])*(kappa[c[i_s]]-2)+1)
 }
 for (i_c in 1:Ncat) {
   omega[i_c] ~ dbeta(omega0*(kappa0-2)+1,(1-omega0)*(kappa0-2)+1)
   kappa[i_c] <- kappaMinusTwo[i_c]+2
   kappaMinusTwo[i_c] ~ dgamma(0.01,0.01)
 }
 omega0 ~ dbeta(1,1)
 kappa0 <- kappaMinusTwo0+2
 kappaMinusTwo0 ~ dgamma(0.01,0.01)
}
```