Bayes Study Week3

Junmin Park

2020 10 13

Ch.7 Linear Regression

0. 데이터 불러오기

```
library('car')
```

```
## Loading required package: carData
```

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
data("Anscombe")
head(Anscombe)
```

	education <int>	income <int>	young <dbl>	urban <int>
ME	189	2824	350.7	508
NH	169	3259	345.9	564
VT	230	3072	348.5	322
MA	168	3835	335.3	846
RI	180	3549	327.1	871
CT	193	4256	341.0	774

6 rows

1. model1

```
mod_string = " model {
    for (i in 1:length(education)) {
        education[i] ~ dnorm(mu[i], prec)
        mu[i] = b0 + b[1]*income[i] + b[2]*young[i] + b[3]*urban[i]
    }

    b0 ~ dnorm(0.0, 1.0/1.0e6)
    for (i in 1:3) {
        b[i] ~ dnorm(0.0, 1.0/1.0e6)
    }

    prec ~ dgamma(1.0/2.0, 1.0*1500.0/2.0)
        ## Initial guess of variance based on overall
        ## variance of education variable. Uses low prior
        ## effective sample size. Technically, this is not
        ## a true 'prior', but it is not very informative.
    sig2 = 1.0 / prec
    sig = sqrt(sig2)
} "
```

- initialization, burn in period

```
set.seed(123)
data_jags = as.list(Anscombe)

params1 = c("b", "sig")

inits1 = function() {
  inits = list("b"=rnorm(3,0.0,100.0), "prec"=rgamma(1,1.0,1.0))
}

mod = jags.model(textConnection(mod_string), data=data_jags, inits=inits1, n.chains=3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 51
##    Unobserved stochastic nodes: 5
##    Total graph size: 422
##
## Initializing model
```

```
update(mod, 1000)
```

- simulating 3 chains

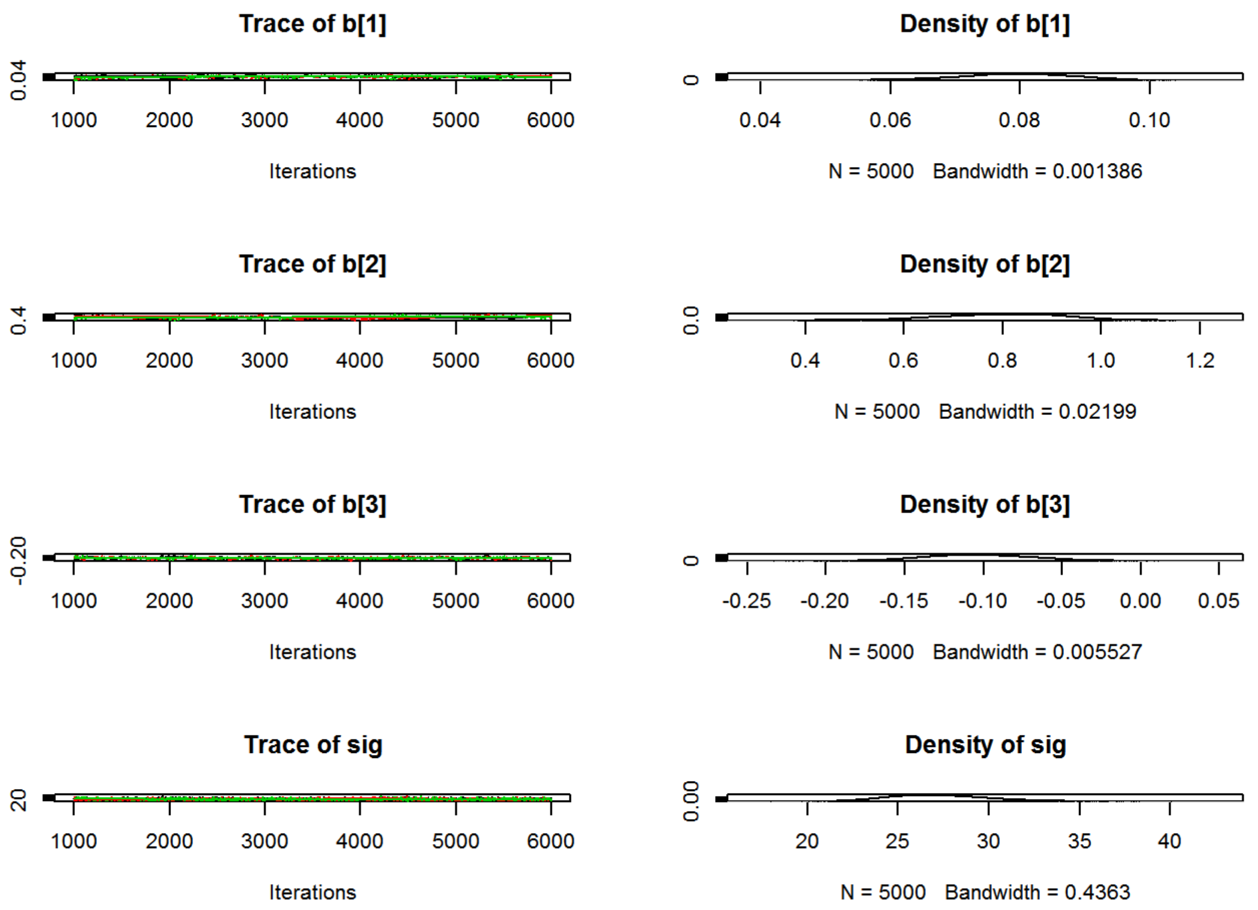
```
mod_sim = coda.samples(model=mod,
                        variable.names=params1,
                        n.iter=5000)
```

- combining 3 chains

```
mod_csim = as.mcmc(do.call(rbind, mod_sim))
```

- checking convergence

```
plot(mod_sim)
```




```
gelman.diag(mod_sim)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## b[1]       1.00       1.01
## b[2]       1.10       1.32
## b[3]       1.01       1.03
## sig        1.00       1.00
##
## Multivariate psrf
##
## 1.09
```

```
autocorr.diag(mod_sim)
```

```
##             b[1]      b[2]      b[3]        sig
## Lag 0  1.0000000 1.0000000 1.0000000 1.00000000
## Lag 1  0.9821154 0.9935722 0.9749550 0.06298275
## Lag 5  0.9176703 0.9693630 0.8833056 0.04290954
## Lag 10 0.8470672 0.9421323 0.7878870 0.05133166
## Lag 50 0.4643770 0.7621009 0.3716275 0.01884917
```

```
effectiveSize(mod_sim)
```

```
##      b[1]      b[2]      b[3]       sig
##  131.66163   46.68272  161.12754 6541.49806
```

```
summary(mod_sim)
```

```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD  Naive SE Time-series SE
## b[1]  0.07889 0.009078 7.412e-05      0.0008129
## b[2]  0.79146 0.141929 1.159e-03      0.0207238
## b[3] -0.10249 0.035681 2.913e-04      0.0028834
## sig  27.34933 2.835733 2.315e-02      0.0403126
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%    97.5%
## b[1]  0.05956  0.07324  0.07928  0.08523  0.09510
## b[2]  0.47257  0.70010  0.79936  0.89066  1.05789
## b[3] -0.17017 -0.12668 -0.10376 -0.07878 -0.02958
## sig  22.52093 25.32365 27.09379 29.09744 33.59894
```

lm 함수와 비교해 보기

```
lmod = lm(education ~ income + young + urban, data=Anscombe)
summary(lmod)
```

```
##
## Call:
## lm(formula = education ~ income + young + urban, data = Anscombe)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.240 -15.738  -1.156  15.883  51.380
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.868e+02  6.492e+01  -4.418 5.82e-05 ***
## income       8.065e-02  9.299e-03   8.674 2.56e-11 ***
## young        8.173e-01  1.598e-01   5.115 5.69e-06 ***
## urban       -1.058e-01  3.428e-02  -3.086  0.00339 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.69 on 47 degrees of freedom
## Multiple R-squared:  0.6896, Adjusted R-squared:  0.6698
## F-statistic: 34.81 on 3 and 47 DF,  p-value: 5.337e-12
```

1. model 2

not using urban variable

```
mod2_string = " model {
    for (i in 1:length(education)) {
        education[i] ~ dnorm(mu[i], prec)
        mu[i] = b0 + b[1]*income[i] + b[2]*young[i]
    }

    b0 ~ dnorm(0.0, 1.0/1.0e6)
    for (i in 1:2) {
        b[i] ~ dnorm(0.0, 1.0/1.0e6)
    }

    prec ~ dgamma(1.0/2.0, 1.0*1500.0/2.0)
        ## Initial guess of variance based on overall
        ## variance of education variable. Uses low prior
        ## effective sample size. Technically, this is not
        ## a true 'prior', but it is not very informative.
    sig2 = 1.0 / prec
    sig = sqrt(sig2)
} "
```

- initializing, bunr in period

```
params2 = c("b", "sig")

inits2 = function() {
  inits = list("b"=rnorm(2,0.0,100.0), "prec"=rgamma(1,1.0,1.0))
}

mod2 = jags.model(textConnection(mod2_string), data=data_jags, inits=inits2, n.chains=3)
```

```
## Warning in jags.model(textConnection(mod2_string), data = data_jags, inits =
## inits2, : Unused variable "urban" in data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 51
##    Unobserved stochastic nodes: 4
##    Total graph size: 320
##
## Initializing model
```

```
update(mod2, 1000)
```

- simulating 3 chains, combining 3 chanins

```
mod2_sim = coda.samples(model=mod2,
                       variable.names=params2,
                       n.iter=5000)

mod2_csim = as.mcmc(do.call(rbind, mod2_sim))
```

3. model3

using income, young, income * young

```
mod3_string = " model {
    for (i in 1:length(education)) {
        education[i] ~ dnorm(mu[i], prec)
        mu[i] = b0 + b[1]*income[i] + b[2]*young[i] + b[3]*income[i]*young[i]
    }

    b0 ~ dnorm(0.0, 1.0/1.0e6)
    for (i in 1:3) {
        b[i] ~ dnorm(0.0, 1.0/1.0e6)
    }

    prec ~ dgamma(1.0/2.0, 1.0*1500.0/2.0)
        ## Initial guess of variance based on overall
        ## variance of education variable. Uses low prior
        ## effective sample size. Technically, this is not
        ## a true 'prior', but it is not very informative.
    sig2 = 1.0 / prec
    sig = sqrt(sig2)
} "
```

- initializing, burn in

```
params3 = c("b", "sig")

inits3 = function() {
  inits = list("b"=rnorm(3,0.0,100.0), "prec"=rgamma(1,1.0,1.0))
}

mod3 = jags.model(textConnection(mod3_string), data=data_jags, inits=inits3, n.chains=3)
```

```
## Warning in jags.model(textConnection(mod3_string), data = data_jags, inits =
## inits3, : Unused variable "urban" in data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 51
##    Unobserved stochastic nodes: 5
##    Total graph size: 372
##
## Initializing model
```

```
update(mod3, 1000)
```

- simulation, combining

```
mod3_sim = coda.samples(model=mod3,
                        variable.names=params3,
                        n.iter=5000)

mod3_csim = as.mcmc(do.call(rbind, mod3_sim))
```

4. model selection

```
dic.samples(mod, n.iter=1e5)
```

```
## Mean deviance:  481
## penalty 5.268
## Penalized deviance: 486.3
```

```
dic.samples(mod2, n.iter=1e5)
```

```
## Mean deviance:  489.1
## penalty 4.023
## Penalized deviance: 493.1
```

```
dic.samples(mod3, n.iter=1e5)
```

```
## Mean deviance:  487
## penalty 5.201
## Penalized deviance: 492.2
```

1번이 dic 제일 낮다 1번 쓰는게 나을듯

prior probability

```
mean(mod_csim[,1]>0.07)
```

```
## [1] 0.8394667
```

ch.8 ANoVA

1. model1

same variance

```
data("PlantGrowth")
```

```
mod_string = " model {
    for (i in 1:length(y)) {
        y[i] ~ dnorm(mu[grp[i]], prec)
    }

    for (j in 1:3) {
        mu[j] ~ dnorm(0.0, 1.0/1.0e6)
    }

    prec ~ dgamma(5/2.0, 5*1.0/2.0)
    sig = sqrt( 1.0 / prec )
} "
```

```
set.seed(82)
str(PlantGrowth)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
##  $ group : Factor w/ 3 levels "ctrl","trt1",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
data_jags = list(y=PlantGrowth$weight,
                 grp=as.numeric(PlantGrowth$group))

params = c("mu", "sig")

inits = function() {
  inits = list("mu"=rnorm(3,0.0,100.0), "prec"=rgamma(1,1.0,1.0))
}

mod = jags.model(textConnection(mod_string), data=data_jags, inits=inits, n.chains=3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 4
##    Total graph size: 74
##
## Initializing model
```

```
update(mod, 1e3)
```

```
mod_sim = coda.samples(model=mod,
                       variable.names=params,
                       n.iter=5e3)
mod_csim = as.mcmc(do.call(rbind, mod_sim))
```

```
summary(mod_sim)
```

```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean      SD  Naive SE Time-series SE
## mu[1] 5.0311 0.22814 0.0018627      0.0018768
## mu[2] 4.6632 0.22911 0.0018707      0.0018704
## mu[3] 5.5268 0.22747 0.0018573      0.0018573
## sig   0.7128 0.09215 0.0007524      0.0008565
##
## 2. Quantiles for each variable:
##
##        2.5%    25%    50%    75%  97.5%
## mu[1] 4.576 4.8813 5.0314 5.1821 5.4787
## mu[2] 4.210 4.5125 4.6622 4.8130 5.1170
## mu[3] 5.086 5.3768 5.5252 5.6745 5.9782
## sig   0.560 0.6476 0.7022 0.7677 0.9209
```

2. model2

different variance

```
mod2_string = " model {
    for (i in 1:length(y)) {
        y[i] ~ dnorm(mu[grp[i]], prec[grp[i]])
    }

    for (j in 1:3) {
        mu[j] ~ dnorm(0.0, 1.0/1.0e6)
        prec[j] ~ dgamma(5/2.0, 5*1.0/2.0)
        sig[j] = sqrt( 1.0 / prec[j])
    }
} "
```

```
set.seed(82)
data_jags = list(y=PlantGrowth$weight,
                 grp=as.numeric(PlantGrowth$group))

params2 = c("mu", "sig")

inits2 = function() {
  inits = list("mu"=rnorm(3,0.0,100.0), "prec"=rgamma(3,1.0,1.0))
}

mod2 = jags.model(textConnection(mod2_string), data=data_jags, inits=inits2, n.chains=3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 6
##    Total graph size: 80
##
## Initializing model
```

```
update(mod, 1e3)
```

```
mod2_sim = coda.samples(model=mod2,
                       variable.names=params2,
                       n.iter=5e3)
mod2_csim = as.mcmc(do.call(rbind, mod2_sim))
```

```
summary(mod2_sim)
```

```
##
## Iterations = 1:5000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
## mu[1]  5.0335 0.2621 0.002140       0.002107
## mu[2]  4.6630 0.2996 0.002446       0.002544
## mu[3]  5.5242 0.2367 0.001933       0.001893
## sig[1] 0.8041 0.1643 0.001342       0.001427
## sig[2] 0.9250 0.1907 0.001557       0.001655
## sig[3] 0.7347 0.1521 0.001242       0.001347
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75% 97.5%
## mu[1]  4.5071 4.8691 5.0341 5.1982 5.557
## mu[2]  4.0712 4.4734 4.6630 4.8535 5.260
## mu[3]  5.0539 5.3710 5.5235 5.6768 5.989
## sig[1] 0.5589 0.6882 0.7795 0.8932 1.196
## sig[2] 0.6441 0.7890 0.8947 1.0261 1.380
## sig[3] 0.5061 0.6276 0.7111 0.8135 1.097
```

3. model selection

```
dic1 = dic.samples(mod, n.iter=1e3)
dic1
```

```
## Mean deviance:  58.96
## penalty 4.153
## Penalized deviance: 63.11
```

```
dic2 = dic.samples(mod2, n.iter=1e3)
dic2
```

```
## Mean deviance:  61.1
## penalty 5.612
## Penalized deviance: 66.71
```

ch.9 Logistic Regression

```
library("MASS")
data("OME")
```

```
any(is.na(OME))
```

```
## [1] FALSE
```

```
dat = subset(OME, OME != "N/A")
dat$OME = factor(dat$OME)
str(dat)
```

```
## 'data.frame':    712 obs. of  7 variables:
##  $ ID     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Age    : int  30 30 30 30 30 30 30 30 30 30 ...
##  $ OME    : Factor w/ 2 levels "high","low": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Loud   : int  35 35 40 40 45 45 50 50 55 55 ...
##  $ Noise  : Factor w/ 2 levels "coherent","incoherent": 1 2 1 2 1 2 1 2 1 2 ...
##  $ Correct: int  1 4 0 1 2 2 3 4 3 2 ...
##  $ Trials : int  4 5 3 1 4 2 3 4 3 2 ...
```

```
mod1_string = " model {
    for (i in 1:length(y)) {
        y[i] ~ dbin(phi[i], n[i])
        logit(phi[i]) = b[1]*Age[i] + b[2]*OMElow[i] + b[3]*Loud[i] + b[4]*Noiseincoherent[i]
    }

    for (j in 1:4) {
        b[j] ~ dnorm(0.0, 1.0/4.0^2)
    }

} "
```

```
mod_glm = glm(Correct/Trials ~ Age + OME + Loud + Noise, data=dat, weights=Trials, family="bino
mial")
X = model.matrix(mod_glm)[,-1]
data_jags = as.list(as.data.frame(X))
data_jags$y = dat$Correct
data_jags$n = dat$Trials
```

```
params = c('b')

mod1 = jags.model(textConnection(mod1_string), data=data_jags, n.chains=3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 712
##    Unobserved stochastic nodes: 4
##    Total graph size: 4373
##
## Initializing model
```

```
update(mod1, 1e3)
```

```
mod1_sim = coda.samples(model=mod1,
                        variable.names=params,
                        n.iter=5e3)
mod1_csim = as.mcmc(do.call(rbind, mod1_sim))
```

```
raftery.diag(mod1_sim)
```

```
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##       Burn-in  Total Lower bound  Dependence
##       (M)      (N)   (Nmin)       factor (I)
##  b[1] 20       21738 3746         5.80
##  b[2] 18       18442 3746         4.92
##  b[3] 22       23947 3746         6.39
##  b[4] 6        6878  3746         1.84
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##       Burn-in  Total Lower bound  Dependence
##       (M)      (N)   (Nmin)       factor (I)
##  b[1] 22       24236 3746         6.47
##  b[2] 11       12285 3746         3.28
##  b[3] 32       34644 3746         9.25
##  b[4] 7        7397  3746         1.97
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##       Burn-in  Total Lower bound  Dependence
##       (M)      (N)   (Nmin)       factor (I)
##  b[1] 14       14531 3746         3.88
##  b[2] 18       20562 3746         5.49
##  b[3] 20       20953 3746         5.59
##  b[4] 7        7534  3746         2.01
```

```
summary(mod1_sim)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD  Naive SE Time-series SE
## b[1] -0.009237 0.003087 2.521e-05       0.000116
## b[2] -0.878117 0.123991 1.012e-03       0.003768
## b[3]  0.045196 0.003901 3.185e-05       0.000181
## b[4]  1.156451 0.106030 8.657e-04       0.001433
##
## 2. Quantiles for each variable:
##
##          2.5%     25%       50%       75%    97.5%
## b[1] -0.01519 -0.0113 -0.009271 -0.007162 -0.00317
## b[2] -1.12507 -0.9598 -0.876546 -0.795205 -0.63722
## b[3]  0.03755  0.0426  0.045230  0.047815  0.05279
## b[4]  0.94964  1.0837  1.155795  1.228064  1.36649
```

```
summary(mod_glm)
```

```
##
## Call:
## glm(formula = Correct/Trials ~ Age + OME + Loud + Noise, family = "binomial",
##     data = dat, weights = Trials)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8354  -0.3389   0.4296   0.8501   2.3694
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -7.294441   0.434928 -16.772  < 2e-16 ***
## Age             0.018896   0.003767   5.016 5.28e-07 ***
## OMElow         -0.237150   0.123257  -1.924   0.0544 .
## Loud            0.171682   0.008880  19.333  < 2e-16 ***
## Noiseincoherent  1.576304   0.115236  13.679  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1431.12  on 711  degrees of freedom
## Residual deviance:  732.38  on 707  degrees of freedom
## AIC: 1262.6
##
## Number of Fisher Scoring iterations: 5
```

prediction

```
pm_coef = colMeans(mod1_csim)

v = c(60, 0, 50, 0)
pm_Xb = v %*% pm_coef
phat = 1.0 / (1.0 + exp(-pm_Xb))
phat
```

```
##           [,1]
## [1,] 0.8462635
```

```
pm_Xb = X %*% pm_coef
phat = 1.0 / (1.0 + exp(-pm_Xb))
summary(phat)
```

```
##        V1
##  Min.   :0.5373
##  1st Qu.:0.7414
##  Median :0.8296
##  Mean   :0.8088
##  3rd Qu.:0.9011
##  Max.   :0.9666
```

0.7을 treshhold로 나눈 table

```
(tab0.7 = table(phat > 0.7, (dat$Correct / dat$Trials) > 0.7))
```

```
##
##         FALSE TRUE
##   FALSE    96   31
##   TRUE    149  436
```

```
sum(diag(tab0.7)) / sum(tab0.7)
```

```
## [1] 0.747191
```