Wulfing_HW03

Sophie Wulfing

2/21/2022

1. Recreate the model of the mean or the simple linear regression from the session_07 R script as a Bayesian model. Were the results the same as the frequentist model?

Results were not the same, as the frequentist model (black linewhere beta = .2804 and a standard deviation of .06) resulted in a slightly steeper beta1 value than the Bayesian model (Red line where b1 = .27 with a standard deviation of .08).

```
# sink("HW3_model.txt") #Had to comment this out so markdown would knit
 cat("
     model { #always start JAGS with this model line

     # Priors-all things we don't know
     beta0 ~ dnorm(0,0.01)      # precision inverse of variance. This means huge variance
     beta1 ~ dnorm(0,0.01)
     precision <- 1 / variance  #Priors are unknown. We only know mass and svl
     variance <- sigma^2
     sigma ~ dunif(0,15) #sigma sq root of variance. We're saying anywhere btwn 1 and 15 (15 would be ma
     #No prior for mew. We will calc further down. We've covered mew using priors for b0 and b1

     # Likelihood
     for(i in 1:nobs){
     mass[i] ~ dnorm(mew[i], precision) #This is the likelihood. From penguins data. Assuming it's norma

     mew[i] <- beta0 + beta1 * svl[i] #obs drawn from mew which depends on krill with some random noise

     } # i loop

     } # end of the model. Penguins will now be saved in WD
     ",fill=TRUE)
```

```
##
##      model { #always start JAGS with this model line
##
##      # Priors-all things we don't know
##      beta0 ~ dnorm(0,0.01)       # precision inverse of variance. This means huge variance
##      beta1 ~ dnorm(0,0.01)
##      precision <- 1 / variance   #Priors are unknown. We only know mass and svl
##      variance <- sigma^2
##      sigma ~ dunif(0,15) #sigma sq root of variance. We're saying anywhere btwn 1 and 15 (15 would b
##      #No prior for mew. We will calc further down. We've covered mew using priors for b0 and b1
##
##
```

```
##      # Likelihood
##      for(i in 1:nobs){
##      mass[i] ~ dnorm(mew[i], precision) #This is the likelihood. From penguins data. Assuming it's no
##
##      mew[i] <- beta0 + beta1 * svl[i] #obs drawn from mew which depends on krill with some random noi
##
##      } # i loop
##
##      } # end of the model. Penguins will now be saved in WD
##
```

```
# sink() #I had to comment this out to get markdown to knit

# Bundle data
win.data <- list(mass = mass,
                 svl = svl,
                 nobs = nrow(data))

# Function to generate starting values aka initial values. Supply init vals
inits <- function()list(beta0 = rnorm(1),
                        sigma = runif(1, 0, 15))

# Parameters to be monitored (= to estimate)
params <- c("beta0",
            "beta1",
            "sigma")

# MCMC settings
nc <- 3
ni <- 1000
nb <- 1
nt <- 1

out <- jags(win.data, inits, params, "HW3_model.txt", n.chains = nc,
            n.thin = nt, n.iter = ni, n.burnin = nb, working.directory = getwd())
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 6
##    Unobserved stochastic nodes: 3
##    Total graph size: 35
##
## Initializing model
```

```
print(out, dig =2)
```

```
## Inference for Bugs model at "HW3_model.txt", fit using jags,
##  3 chains, each with 1000 iterations (first 1 discarded)
##  n.sims = 2997 iterations saved
##          mu.vect sd.vect   2.5%   25%   50%   75% 97.5% Rhat n.eff
## beta0      -4.61    4.13 -12.47 -6.91 -4.79 -2.53  4.28 1.00  3000
```

```
## beta1       0.26    0.09   0.07  0.22  0.27  0.31  0.44 1.00  3000
## sigma       1.47    0.98   0.62  0.92  1.20  1.67  3.91 1.01   370
## deviance   19.06    4.24  14.33 16.02 17.94 20.98 30.19 1.01   330
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 8.9 and DIC = 28.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
m <- lm(mass ~ svl)
summary(m)
```

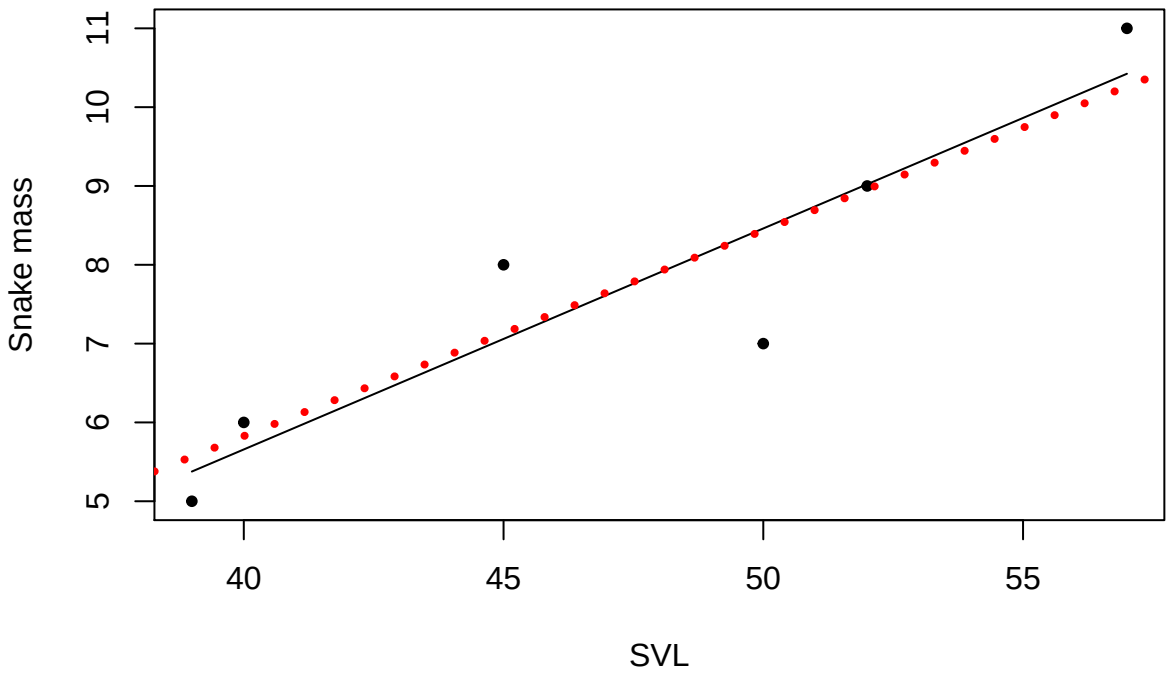
```
##
## Call:
## lm(formula = mass ~ svl)
##
## Residuals:
##       1       2       3       4       5       6
##  0.34286  0.94086 -0.37674 -1.46113 -0.02193  0.57608
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.5588     2.8565  -1.946  0.12352
## svl           0.2804     0.0600   4.673  0.00949 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9503 on 4 degrees of freedom
## Multiple R-squared:  0.8452, Adjusted R-squared:  0.8065
## F-statistic: 21.84 on 1 and 4 DF,  p-value: 0.009495
```

```
svl_p <- seq(min(svl),max(svl),length.out = length(svl))

svl_preds <- m$coefficients[1] + m$coefficients[2]*svl_p



plot(mass ~ svl,
     ylab = "Snake mass",
     xlab = "SVL",
     pch = 20)
lines(svl_preds ~ svl_p)
abline(a =out$BUGSoutput$mean$beta0,
       b = out$BUGSoutput$mean$beta1,
       lwd = 4,lty = 3, col = "red")
```


Snake mass
5
6
7
8
9
10
11
40
45
50
55
SVL

2. Complete exercise 1 on page 89 of Kery 2010. Provide the model matrix. Note: there are two correct answers: a means and an effects parameterization – either is fine.

Question: 1. Fitting a design matrix: The interaction-effects ANCOVA wasn't a useful statistical model for the toy snake data set, since six fitted parameters perfectly explain six observations and we can't estimate anymore the variability in the system. Use lm() to fit a custom-built design matrix, i.e., the design matrix of an ANCOVA with partial interaction effects, where the slopes of the mass length relationship are the same in population 1 and population 3. Build this design matrix in R, call it X, and fit the model by directly specifying X as the explanatory variable in function lm()

```
fm <- lm(mass ~ pop * svl)

model.matrix(fm)
```

```
##   (Intercept) pop2 pop3 svl pop2:svl pop3:svl
## 1           1    0    0  40        0        0
## 2           1    0    0  45        0        0
## 3           1    1    0  39       39        0
## 4           1    1    0  50       50        0
## 5           1    0    1  52        0       52
## 6           1    0    1  57        0       57
## attr(,"assign")
## [1] 0 1 1 2 3 3
## attr(,"contrasts")
## attr(,"contrasts")$pop
## [1] "contr.treatment"
```

```
#Now to make partial interaction effects
X <- model.matrix(fm)[,2:5]
X[3:4,3] <- 0
#X[2,5] <- X[2,3]

X
```

```
##   pop2 pop3 svl pop2:svl
## 1    0    0  40        0
## 2    0    0  45        0
## 3    1    0   0       39
## 4    1    0   0       50
## 5    0    1  52        0
## 6    0    1  57        0
```

```
new_m <- lm(mass~X)

summary(new_m)
```

```
## Warning in summary.lm(new_m): essentially perfect fit: summary may be unreliable
```

```
##
## Call:
## lm(formula = mass ~ X)
##
```

```
## Residuals:
##          1          2          3          4          5          6
## -1.332e-15  1.332e-15 -4.930e-32  3.451e-31  1.332e-15 -1.332e-15
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -1.000e+01  2.273e-14 -4.400e+14 1.45e-15 ***
## Xpop2        7.909e+00  2.743e-14  2.883e+14 2.21e-15 ***
## Xpop3       -1.800e+00  6.928e-15 -2.598e+14 2.45e-15 ***
## Xsvl         4.000e-01  5.329e-16  7.506e+14 8.48e-16 ***
## Xpop2:svl    1.818e-01  3.426e-16  5.308e+14 1.20e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.665e-15 on 1 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 8.216e+29 on 4 and 1 DF,  p-value: 8.274e-16
```

```
model.matrix(new_m)
```

```
##   (Intercept) Xpop2 Xpop3 Xsvl Xpop2:svl
## 1           1     0     0   40         0
## 2           1     0     0   45         0
## 3           1     1     0    0        39
## 4           1     1     0    0        50
## 5           1     0     1   52         0
## 6           1     0     1   57         0
## attr(,"assign")
## [1] 0 1 1 1 1
```

To be honest, I tried to graph this but couldn't get it to do it. Would you then take the average of the intercepts between pop1 and pop3 and then call that model the model for both of them? Or would you graph all three separately like you did in the original model still?

3. Fit a means and effects parameterization of a t-test or an ANOVA using either a) data you simulate, b) the Swiss hare data, or c) your own data. Report the results from the two methods either as a table or plot, and interpret them.

For this question, I used the Swiss hare data. I ran an ANOVA looking at the different effects that regions have on the density of hares. In the effects model, everything is being compared to the Aare region. The estimates report the DIFFERENCE between the number of hares you will expect to see at each site and that number of expected hares an the Aare region. The corresponding p values show us if these differences are stastically different from one another (we can say that Central and SW regions are the most stastically different). In the means model, each estimate is the number of expected hares in each region, without comparing the regions to eachother. The p values tell us if each expected number of hares is stastically signficant (which they all are, according to the model)

```
hares <- read.delim("HW3_hareData.txt")

model_effects <- lm(mean.density ~ region, hares)
summary(model_effects)
```

```
##
## Call:
## lm(formula = mean.density ~ region, data = hares)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9006 -2.3279 -0.4942  1.6288 16.9740
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        4.8541     0.2763  17.569  < 2e-16 ***
## regionBaselland   -1.8980     0.6034  -3.146  0.00173 **
## regionCH.Central  -2.2351     0.4438  -5.036 6.13e-07 ***
## regionCH.E        -1.1373     0.4131  -2.753  0.00606 **
## regionCH.N         0.4020     0.4712   0.853  0.39390
## regionCH.SW        4.7665     0.4938   9.653  < 2e-16 ***
## regionCH.W         0.2649     0.4304   0.616  0.53843
## regionRhone       -1.2481     0.6535  -1.910  0.05657 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.35 on 669 degrees of freedom
##   (275 observations deleted due to missingness)
## Multiple R-squared:  0.2377, Adjusted R-squared:  0.2298
## F-statistic: 29.81 on 7 and 669 DF,  p-value: < 2.2e-16
```

```
model_means <- lm(mean.density ~ region -1, hares)
summary(model_means)
```

```
##
## Call:
## lm(formula = mean.density ~ region - 1, data = hares)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -6.9006 -2.3279 -0.4942  1.6288 16.9740
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## regionAare         4.8541     0.2763  17.569  < 2e-16 ***
## regionBaselland    2.9561     0.5364   5.511 5.09e-08 ***
## regionCH.Central   2.6190     0.3474   7.540 1.54e-13 ***
## regionCH.E         3.7168     0.3071  12.104  < 2e-16 ***
## regionCH.N         5.2561     0.3817  13.769  < 2e-16 ***
## regionCH.SW        9.6206     0.4092  23.508  < 2e-16 ***
## regionCH.W         5.1190     0.3301  15.509  < 2e-16 ***
## regionRhone        3.6060     0.5922   6.090 1.91e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.35 on 669 degrees of freedom
##   (275 observations deleted due to missingness)
## Multiple R-squared:  0.7002, Adjusted R-squared:  0.6966
## F-statistic: 195.3 on 8 and 669 DF,  p-value: < 2.2e-16
```