CSP Assignment ps4

Module: Computational Statistics & Probability - MADS
Professor: Dr. Gregory Wheeler
Student: Haowei Lee
Textbook: Statistical Rethinking 2nd edition
* The pdf file is produced by knitting rmd into html and export the result as pdf.

Import libraries

```
library(rethinking)
```

Question 1

Adapt the code in the book for plotting the observed data (Figure 11.4, top panel) to recreate the posterior predictions plot provided.

```
# Import the data
data(chimpanzees)
d <- chimpanzees

d$treatment <- 1 + d$prosoc_left + 2*d$condition

dat_list <- list(
  pulled_left = d$pulled_left,
  actor = d$actor,
  treatment = as.integer(d$treatment) )
```

Build the model

```
m11.4 <- ulam(
  alist(
    pulled_left ~ dbinom( 1 , p ) ,
    logit(p) <- a[actor] + b[treatment] ,
    a[actor] ~ dnorm( 0 , 1.5 ),
    b[treatment] ~ dnorm( 0 , 0.5 )
  ) , data=dat_list , chains=4 , log_lik=TRUE )
```

```
dat <- list( actor=rep(1:7,each=4) , treatment=rep(1:4,times=7) )
p_post <- link( m11.4 , data=dat )
p_mu <- apply(p_post , 2 , mean )
p_ci <- apply(p_post, 2, PI)
```

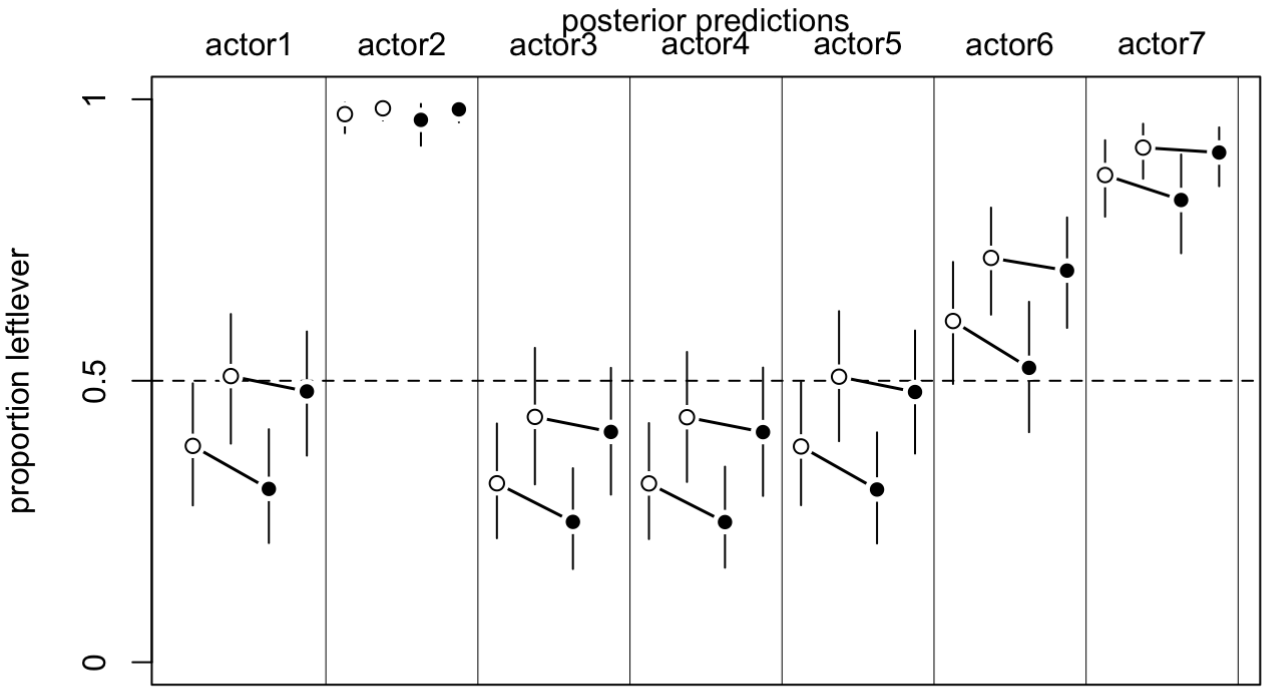
Create the plot

```
plot( NULL,xlim=c(1,28),ylim=c(0,1),xlab="",
ylab="proportion leftlever",xaxt="n",yaxt="n")
axis( 2,at=c(0,0.5,1),labels=c(0,0.5,1))
abline( h=0.5,lty=2)

for (j in 1:7)
  abline(v=(j - 1)*4 + 4.5,lwd=0.5)
for (j in 1:7)
  text((j - 1)*4 + 2.5,1.1,concat("actor",j),xpd=TRUE)
for ( j in (1:7)[-2] ) {
  lines((j - 1)*4 + c(1,3) , p_mu[(j - 1)*4 + c(1,3)] , lwd=1.5 )
  lines((j - 1)*4 + c(2,4) , p_mu[(j - 1)*4 + c(2,4)] , lwd=1.5 )
}
for ( i in 1:28 )
  lines( c(i,i), p_ci[,i] , lwd=1 )

points( 1:28 , p_mu , pch=16 , col="white" , cex=1.3 )
points( 1:28 , p_mu , pch=c(1,1,16,16) )
points( 1:28,t(p_mu),pch=16,col="white",cex=1.7)
points( 1:28,t(p_mu),pch=c(1,1,16,16),lwd=1)
yoff <-0.01

mtext( "posterior predictions\n")
```



Question 2

```
data(Wines2012)
d <- Wines2012
```

Inspect the original data

```
options(digits = 2) #set to two decimal
precis(d, depth=2)
```

```
##              mean   sd 5.5% 94.5%     histogram
## judge         NaN   NA   NA    NA
## flight        NaN   NA   NA    NA
## wine          NaN   NA   NA    NA
## score       14.20 2.66   10    18
## wine.amer    0.60 0.49    0     1
## judge.amer   0.56 0.50    0     1
```

a. Construct index variables

```
dat_list <- list(
                 S = standardize(d$score),
                 jid = as.integer(d$judge),
                 wid = as.integer(d$wine)
                )
```

b. Choose dnorm(0,0.5) to be the priors.

Because we standardize the wine score, so the values will tend to center around zero. Therefore we can choose 0.5 as our without further information.

c. Build a model considering variation among judges and wines.

```
model_Q2 <- ulam(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- j[jid] + w[wid],
        j[jid] ~ dnorm(0,0.5),
        w[wid] ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ), data=dat_list , chains=4 , cores=4 )
```

To revisit the details of our model

```
show(model_Q2)
```

```
## Hamiltonian Monte Carlo approximation
## 2000 samples from 4 chains
##
## Sampling durations (seconds):
##         warmup sample total
## chain:1   0.05   0.04  0.09
## chain:2   0.04   0.04  0.08
## chain:3   0.05   0.04  0.09
## chain:4   0.05   0.04  0.09
##
## Formula:
## S ~ dnorm(mu, sigma)
## mu <- j[jid] + w[wid]
## j[jid] ~ dnorm(0, 0.5)
## w[wid] ~ dnorm(0, 0.5)
## sigma ~ dexp(1)
```

We have 2000 samples

Inspect the result with different index

```
options(digits = 2) #set to two decimal
precis(model_Q2, depth=2)
```

```
##          mean    sd   5.5%  94.5% n_eff Rhat4
## j[1]  -0.2765 0.194 -0.584  0.028  2356     1
## j[2]   0.2078 0.201 -0.120  0.512  2315     1
## j[3]   0.2087 0.200 -0.106  0.532  2421     1
## j[4]  -0.5408 0.208 -0.875 -0.209  2572     1
## j[5]   0.7985 0.199  0.480  1.118  1921     1
## j[6]   0.4732 0.195  0.163  0.808  1779     1
## j[7]   0.1300 0.198 -0.181  0.447  2307     1
## j[8]  -0.6571 0.198 -0.971 -0.333  2057     1
## j[9]  -0.3457 0.193 -0.643 -0.030  2102     1
## w[1]   0.1229 0.263 -0.297  0.547  3132     1
## w[2]   0.0898 0.264 -0.324  0.540  3870     1
## w[3]   0.2286 0.261 -0.191  0.637  2678     1
## w[4]   0.4628 0.252  0.051  0.852  2973     1
## w[5]  -0.1041 0.262 -0.522  0.312  2975     1
## w[6]  -0.3004 0.261 -0.708  0.118  3279     1
## w[7]   0.2404 0.266 -0.186  0.659  3022     1
## w[8]   0.2262 0.262 -0.193  0.635  2488     1
## w[9]   0.0713 0.266 -0.362  0.493  2925     1
## w[10]  0.1049 0.267 -0.335  0.525  2949     1
## w[11] -0.0050 0.263 -0.441  0.419  2475     1
## w[12] -0.0202 0.256 -0.428  0.385  2926     1
## w[13] -0.0890 0.257 -0.496  0.307  2537     1
## w[14]  0.0046 0.266 -0.428  0.421  3153     1
## w[15] -0.1842 0.265 -0.624  0.231  2975     1
## w[16] -0.1625 0.256 -0.547  0.252  2500     1
## w[17] -0.1146 0.264 -0.521  0.313  3176     1
## w[18] -0.7160 0.263 -1.140 -0.299  3265     1
## w[19] -0.1317 0.268 -0.560  0.284  3103     1
## w[20]  0.3249 0.257 -0.100  0.728  2781     1
## sigma  0.8491 0.047  0.776  0.929  3073     1
```

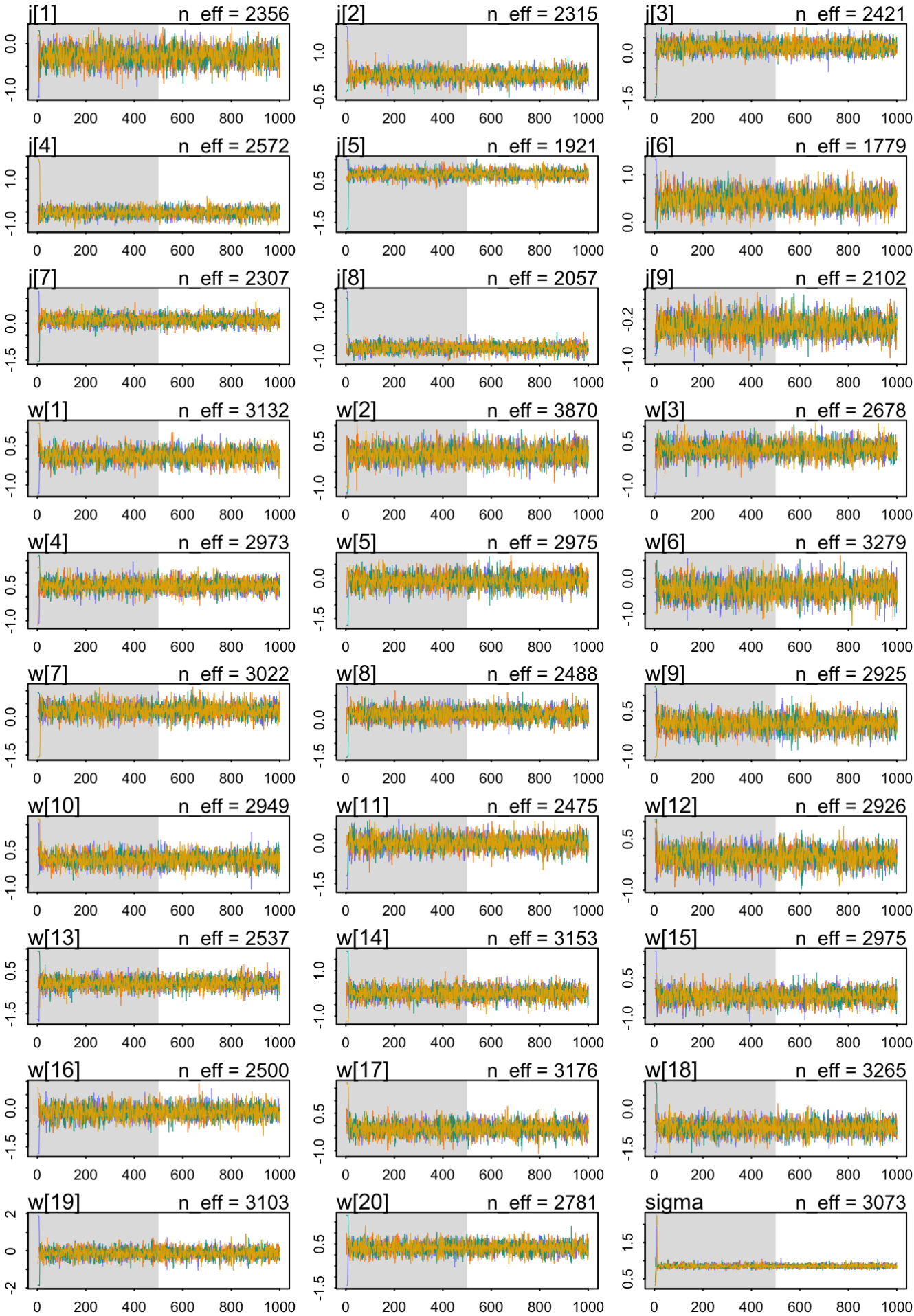
To evaluate the MCMC model built by `ulam`, we can look at two indexes: **n_eff** and **Rhat**. As defined, if **n_eff** is lower than the number of samples, the sampling is inefficient. As for **Rhat**, if the value is above 1, then the MCMC is not converging. Based on this definition of evaluation, we know that our model has good values of these indicators because our **n_eff** is more than 2000 samples. As for **Rhat**, we have 1 for all parameters. Therefore, we can say our n_eff and That are pretty good.

However, sometimes these two indicators could be misleading, so it would be good to include some visualisation into our evaluation on our Markov chain. To do that, we can use `traceplot()`

d. Check if the chains is converging.

```
traceplot(model_Q2)
```

```
## [1] 1000
## [1] 1
## [1] 1000
```



Traditionally, we observe three key aspects to evaluate our MCMC model when plotting it out. (refer to the textbook)

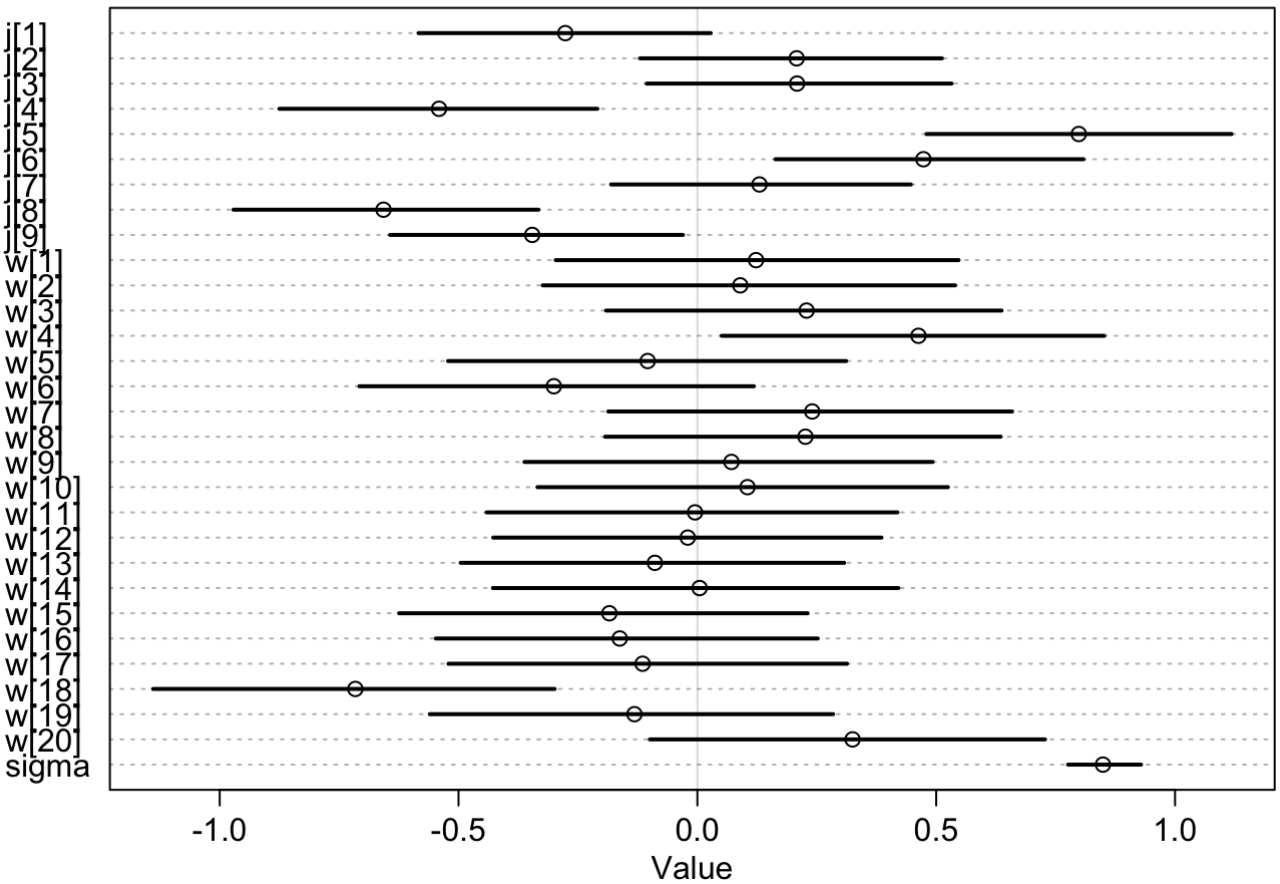
1. Stationarity
2. Good Mixing

3. Convergence

From the trace plot we have, we can see that the plots satisfy with the three evaluation components.

In addition, we can plot out the precis to observe the range of the parameters.

```
plot(precis(model_Q2, 2))
```



Questions:

- Is there variation among individual judges and individual wines?
- Are there patterns you can notice just by plotting the differences?
- Which judges have the lowest and highest ratings?
- Which wines are rated worst and which best on average?

Answers:
To answer the four questions above, we can observe either the **precis** result or the **precis plot**. (The scores are already standardised above; therefore, we use standardised scores for comparison in the following analysis)

For the first two questions, we observe the precis plot and see noticeable variations among judges. j[i] for i=1:9 are values of 9 judges. Each j[i] represent the score and variation of a judge giving the score. Lower values mean that a judge gives a lower score on average, while higher values mean that a judge gives a higher score on average. As for the wine, we can observe the 20 kinds of wine at w[i] for i=1:20, representing the average score of a specific wine among all judges. Specifically, except for w[18], whose distribution is more extreme to the left side, other wines have values condense around the middle, where the value is 0. In conclusion, the variation of judges contributes more to the variation on the score than the part of the variation of wine do.

As for the ranking aspect in terms of judges and wine, we can also obtain insight from the result of precis and its plot. It is evident that j[8], the judge indicated as 8, gives a lower rating on average and gives the lowest mean score at around -0.65. On the other hand, the highest score, at around 0.80. was provided by j[5], the judge indicated as 5. As for the wine itself, w[18] has a lower score on average and has the lowest mean score at around -0.72, while w[4] wins the highest score at around 0.46.

Question 3

standardise score and construct indicator variables

```
list_Q3 <- list(
    score_sd = standardize(d$score),
    us_wine = d$wine.amer,
    us_judge = d$judge.amer,
    red_wine = ifelse(d$flight=="red",1L,0L)  # If red wine, assign 1L (Add "L" to make
indexes integer)
)
```

As in question 2, we first standardize the wine score, so we keep using the ordinary prior 0.5 for wine, judge and redwine. As for a, the intercept, we can try smaller sigma.

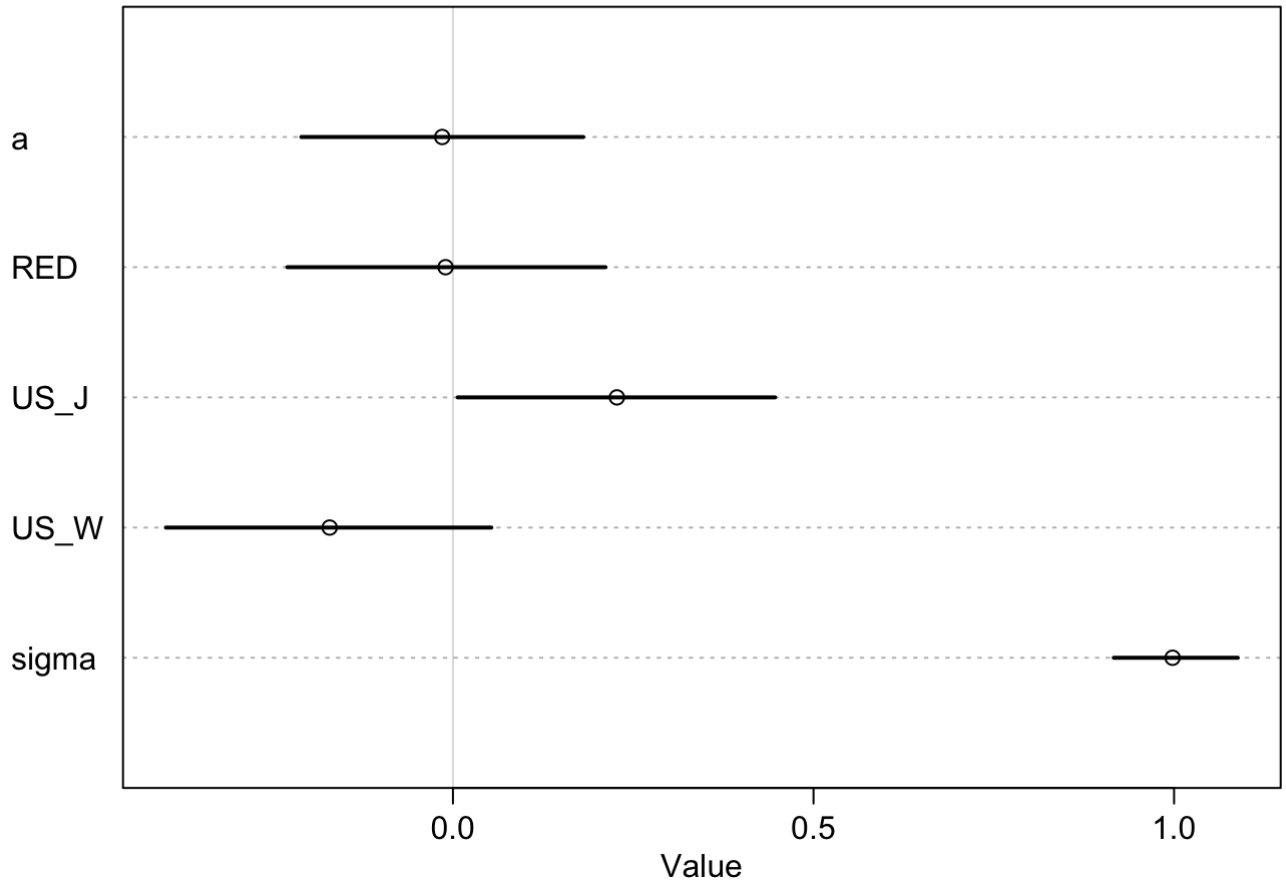
```
model_Q3 <- ulam(
    alist(
        score_sd ~ dnorm( mu , sigma ),
        mu <- a + US_W*us_wine + US_J*us_judge + RED*red_wine,
        a ~ dnorm( 0 , 0.2 ),
        c(US_W, US_J, RED) ~ dnorm( 0 , 0.5 ),  # omit the  index variables
        sigma ~ dexp(1)
    ), data=list_Q3 , chains=4 , cores=4 )
```

Inspect the result

```
options(digits = 2) #set to two decimal
precis(model_Q3, depth=2)
```

```
##         mean    sd    5.5% 94.5% n_eff Rhat4
## a     -0.015 0.121 -0.2104 0.181  1370     1
## RED   -0.010 0.139 -0.2297 0.212  1585     1
## US_J   0.227 0.140  0.0065 0.447  1498     1
## US_W  -0.171 0.141 -0.3981 0.053  1629     1
## sigma  0.998 0.054  0.9164 1.088  1900     1
```

```
plot(precis(model_Q3, 2))
```



Questions:

- What do you conclude about the differences among the wines and judges?
- Compare your results to the results from Problem 2.

Answers:

Looking at the result of precis and its plot, we can conclude that:

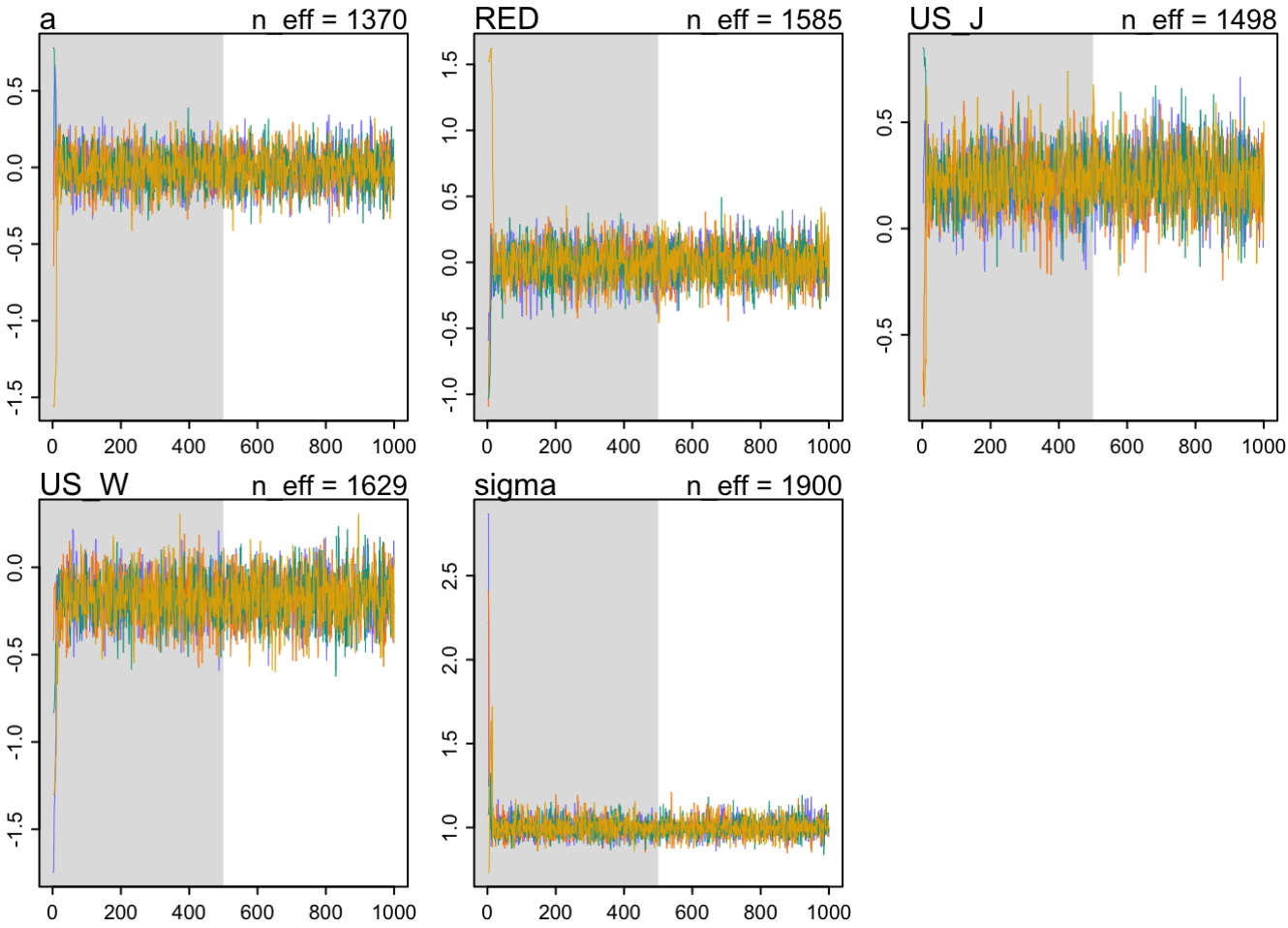
1. There is not much difference between rea and white wine based on the value of US_W coefficient is around 0.
2. American judges tend to be more generous on scoring the wine because we see that the vaulue of US_J is above 0 at around 0.23, and have positive correlation to the score.
3. As for the wine itself, American wine have lower score in general becase we see that the vaulue of US_W is below 0 at around -0.18, implying the negative correlation to the score

```
show(model_Q3)
```

```
## Hamiltonian Monte Carlo approximation
## 2000 samples from 4 chains
##
## Sampling durations (seconds):
##         warmup sample total
## chain:1   0.08   0.07  0.15
## chain:2   0.08   0.08  0.16
## chain:3   0.08   0.07  0.15
## chain:4   0.07   0.07  0.14
##
## Formula:
## score_sd ~ dnorm(mu, sigma)
## mu <- a + US_W * us_wine + US_J * us_judge + RED * red_wine
## a ~ dnorm(0, 0.2)
## c(US_W, US_J, RED) ~ dnorm(0, 0.5)
## sigma ~ dexp(1)
```

```
traceplot(model_Q3)
```

```
## [1] 1000
## [1] 1
## [1] 1000
```



Looking at the traceplot of model in question 3, we see that it also satisfy the three evaluation points: **stationarity**, **good mixing**, and **convergence**. Though the **Rhat** for every parameters are also 1 as the previous model in question 2, the **n_eff** here is below 2000. Still, the amount of those effective sample is close to 2000, so it is a reasonable model.

One interesting observation is that when we use `show()` to inspect the training process behind our MCMC model, we find that model we built in question two takes a smaller amount of time training the Markov Chain. In each chain, they all take faster than 0.1 seconds. However, the four chains of model built in question three all take longer than 0.1 seconds. The time difference might result from the number of variables we consider. In question 3, we consider three variables, while in question to we only consider two. Based on the result we have for both models, we see that the result is not significantly different. However, if we consider time, we may want to construct a more time-efficient model.