Statistical Rethinking Winter 2020 – Homework Week 5

Alessandro Gentilini .@gmail (just auditing)

January 9, 2021

1. Consider the `data(Wines2012)` data table. These data are expert ratings of 20 different French and American wines by 9 different French and American judges. Your goal is to model `score`, the subjective rating assigned by each judge to each wine. I recommend standardizing it.

In this first problem, consider only variation among judges and wines. Construct index variables of `judge` and `wine` and then use these index variables to construct a linear regression model. Justify your priors. You should end up with 9 judge parameters and 20 wine parameters. Use `ulam` instead of quap to build this model, and be sure to check the chains for convergence. If you'd rather build the model directly in Stan or PyMC3 or Julia (Turing is a good choice!), go ahead. I just want you to use MCMC instead of quadratic approximation.

How do you interpret the variation among individual judges and individual wines?

Do you notice any patterns, just by plotting the differences?

Which judges gave the highest/lowest ratings?

Which wines were rated worst/best on average?

```
library(rethinking)
data(Wines2012)
Wines2012$score_std <- (Wines2012$score-mean(Wines2012$score))/sd(Wines2012$score)
Wines2012$w_id <- as.integer(Wines2012$wine)
Wines2012$j_id <- as.integer(Wines2012$judge)

# fix stan parser error:
names(Wines2012)[names(Wines2012) == "wine.amer"] <- "wine_amer"
names(Wines2012)[names(Wines2012) == "judge.amer"] <- "judge_amer"
```

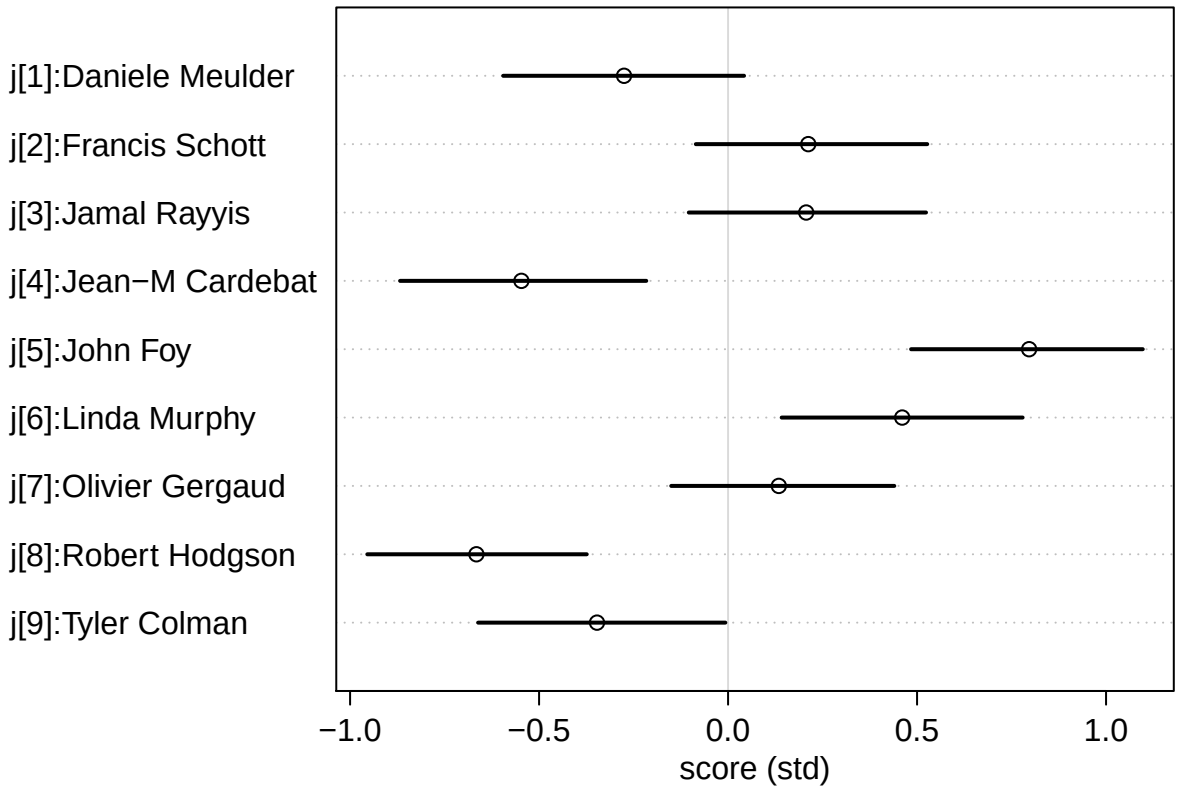
```
m1 <- ulam(
    alist(
        score_std ~ dnorm( mu , sigma ),
        mu <- w[w_id]+j[j_id],
        w[w_id] ~ dnorm( 0 , 0.5 ),
        j[j_id] ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp( 1 )
    ) , data=Wines2012, chains=1 )
```

```
##
## SAMPLING FOR MODEL '3b95caf74b1a96765826f44b98c606de' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 1: Adjust your expectations accordingly!
```

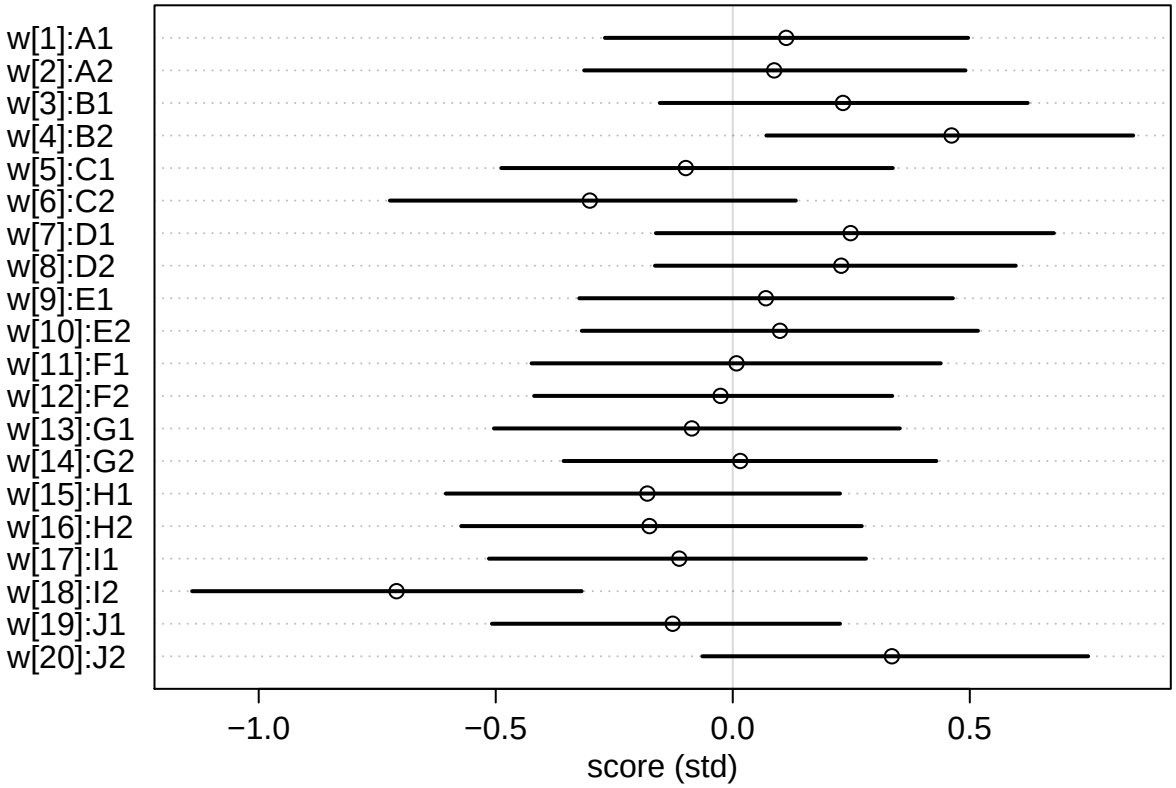
```
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.065447 seconds (Warm-up)
## Chain 1:                0.054823 seconds (Sampling)
## Chain 1:                0.12027 seconds (Total)
## Chain 1:
```

I plot the scores following the section 5.3.2 of the book:

```
labels <- paste( "j[" , 1:9 , "]:" , levels(Wines2012$judge) , sep="" )
plot( precis( m1 , depth=2 , pars="j" ) , labels=labels ,
    xlab="score (std)" )
```




```
labels <- paste( "w[" , 1:20 , "]:" , levels(Wines2012$wine) , sep="" )
plot( precis( m1 , depth=2 , pars="w" ) , labels=labels ,
    xlab="score (std)" )
```



The variation among individual judges is due to the fact that judging a wine is not a purely objective task and I think that the personal preferences (or the personal tastes) sneak in; I suspect the 20 wines are not uniformly distributed accross all the relevant dimensions in wine tasting (white-red; still-sparkling; sweet-dry; etc.) and, let's say, if there is a prevalence of sparkling wine and a judge has the sparkling wine as his favorites then maybe he will find difficult to give low scores and be impartial.

The variation among individual wines are less large: wines **are** different and so it is expected a variation among individual wine.

I do not understand what does *plotting the differences* mean.

John Foy gives the highest scores while Robert Hodgson gives the lowest scores.

B2 on average gets the highest score while I2 get the lowest score.

---

2. Now consider three features of the wines and judges:

```
(1) `flight`: Whether the wine is red or white.
(2) `wine.amer`: Indicator variable for American wines.
(3) `judge.amer`: Indicator variable for American judges.
```

Use indicator or index variables to model the influence of these features on the scores. Omit the individual judge and wine index variables from Problem 1. Do not include interaction effects yet. Again use `ulam`, justify your priors, and be sure to check the chains.

What do you conclude about the differences among the wines and judges?

Try to relate the results to the inferences in Problem 1.

```
Wines2012$f_id <- as.integer(Wines2012$flight)

Wines2012$wine_amer <- as.factor(Wines2012$wine_amer)
```

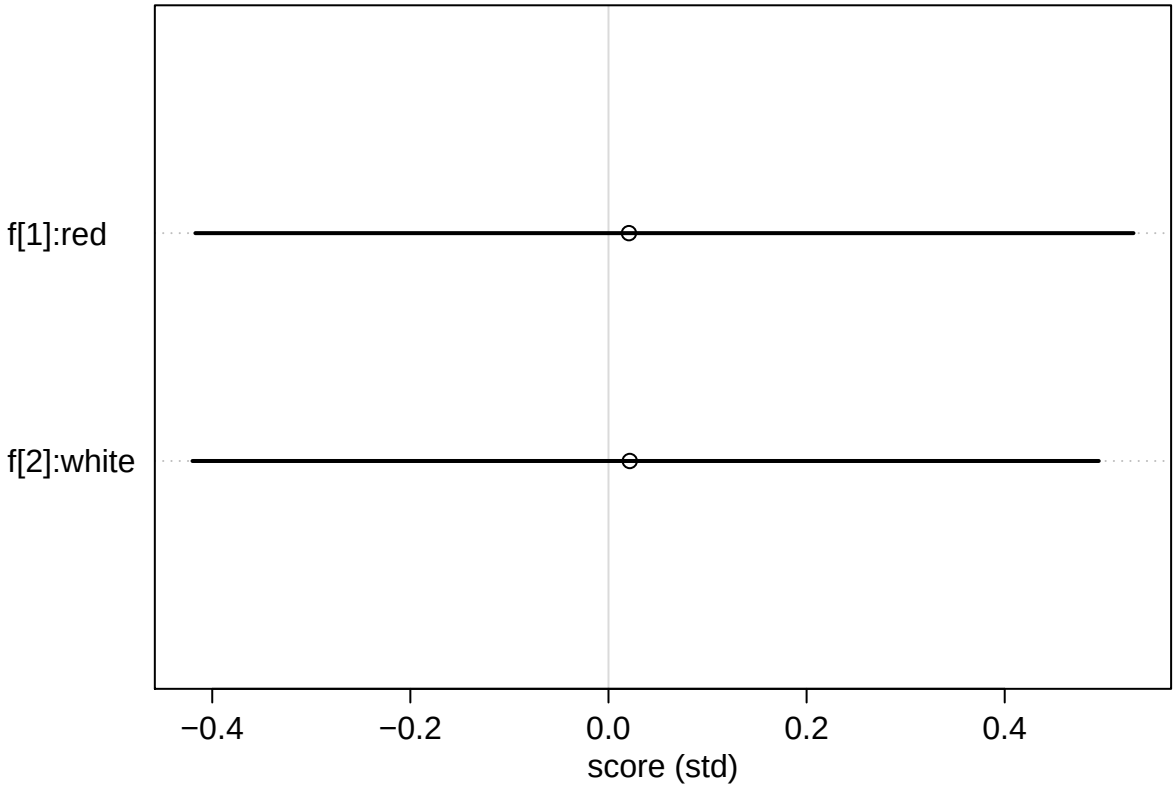
```
Wines2012$judge_amer <- as.factor(Wines2012$judge_amer)

Wines2012$wa_id <- as.integer(Wines2012$wine_amer)
Wines2012$ja_id <- as.integer(Wines2012$judge_amer)
```

```
m2 <- ulam(
    alist(
        score_std ~ dnorm( mu , sigma ),
        mu <- f[f_id]+wa[wa_id]+ja[ja_id],
        f[f_id] ~ dnorm( 0 , 0.5 ),
        wa[wa_id] ~ dnorm( 0 , 0.5 ),
        ja[ja_id] ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp( 1 )
    ) , data=Wines2012, chains=1 )
```

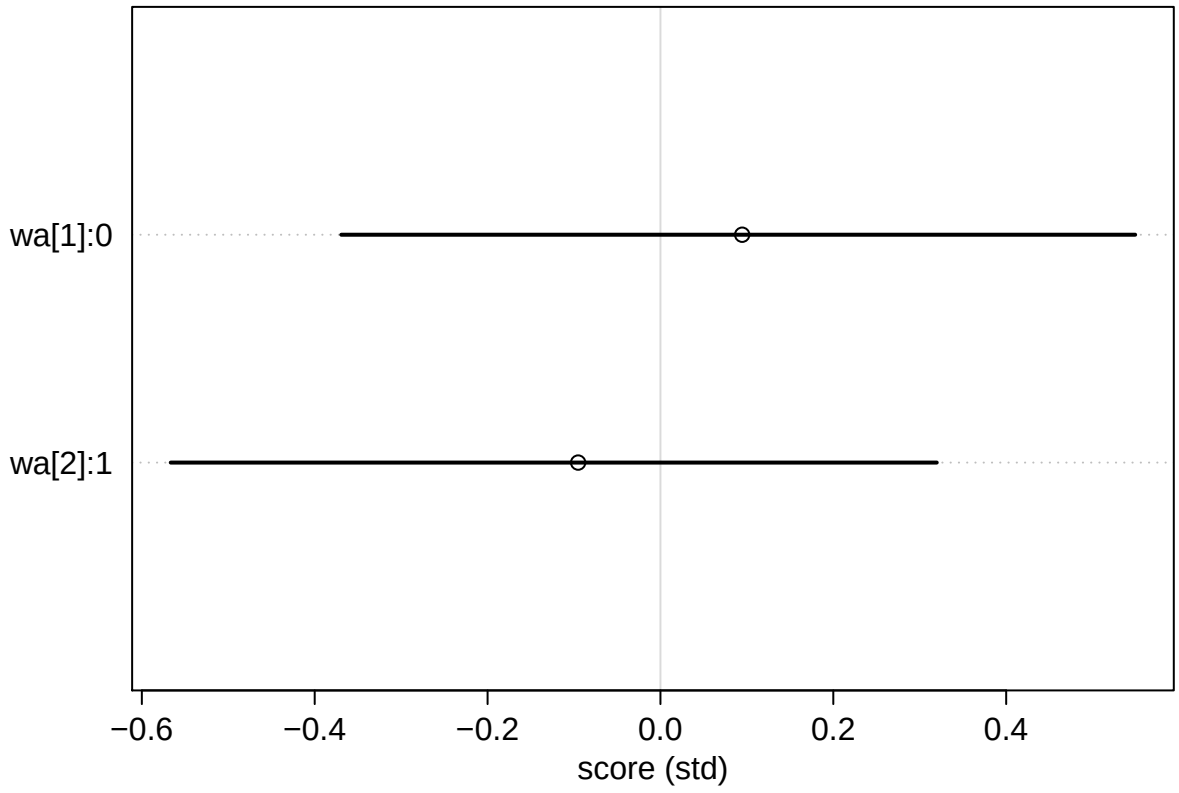
```
##
## SAMPLING FOR MODEL '6dd07826cc05e9e6c0aa7c205882449b' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.3 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.112994 seconds (Warm-up)
## Chain 1:                0.104582 seconds (Sampling)
## Chain 1:                0.217576 seconds (Total)
## Chain 1:
```

I plot the scores following the section 5.3.2 of the book:

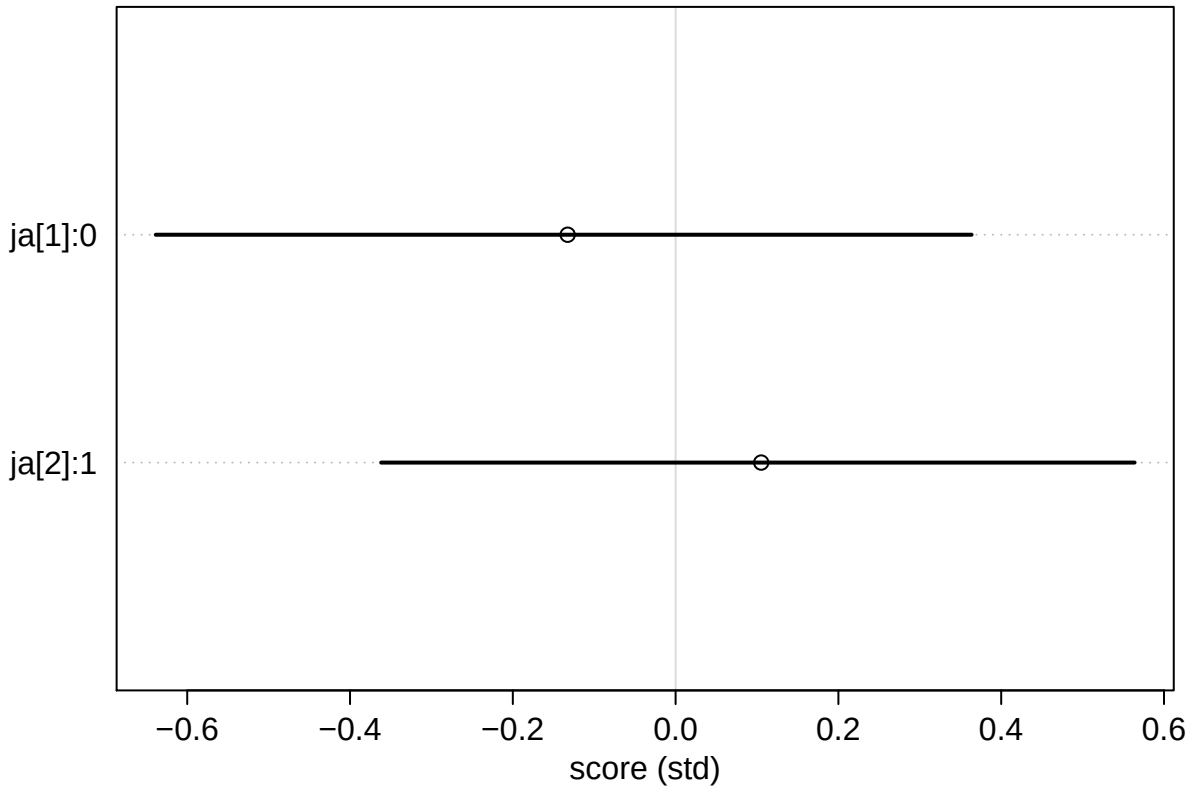
```
labels <- paste( "f[" , 1:2 , "]:" , levels(Wines2012$flight) , sep="" )
plot( precis( m2 , depth=2 , pars="f" ) , labels=labels ,
    xlab="score (std)" )
```




```
labels <- paste( "wa[" , 1:2 , "]:" , levels(Wines2012$wine_amer) , sep="" )
plot( precis( m2 , depth=2 , pars="wa" ) , labels=labels ,
    xlab="score (std)" )
```



```
labels <- paste( "ja[" , 1:2 , "]:" , levels(Wines2012$judge_amer) , sep="" )
plot( precis( m2 , depth=2 , pars="ja" ) , labels=labels ,
    xlab="score (std)" )
```



---

3. Now consider two-way interactions among the three features. You should end up with three different interaction terms in your model. These will be easier to build, if you use indicator variables. Again use `ulam`, justify your priors, and be sure to check the chains.

Explain what each interaction means.

Be sure to interpret the model's predictions on the outcome scale (`mu`, the expected score), not on the scale of individual parameters. You can use `link` to help with this, or just use your knowledge of the linear model instead.

What do you conclude about the features and the scores?

Can you relate the results of your model(s) to the individual judge and wine inferences from Problem 1?

Reset the dataset:

```
data(Wines2012)
Wines2012$score_std <- (Wines2012$score-mean(Wines2012$score))/sd(Wines2012$score)

# fix stan parser error:
names(Wines2012)[names(Wines2012) == "wine.amer"] <- "wine_amer"
names(Wines2012)[names(Wines2012) == "judge.amer"] <- "judge_amer"
Wines2012$f_id <- as.integer(Wines2012$flight)
```

```
Wines2012$wa_id <- as.integer(as.factor(Wines2012$wine_amer))
Wines2012$ja_id <- as.integer(as.factor(Wines2012$judge_amer))
```

Create the indicator variables:

```
Wines2012$is_red <- ifelse(Wines2012$flight=="red",1,0)
```

```
m3 <- ulam(
    alist(
        score_std ~ dnorm( mu , sigma ),
        mu <- f[f_id] + wa[wa_id] + ja[ja_id] + fw*is_red*wine_amer + fj*is_red*judge_amer + wj*wine_ame
        f[f_id] ~ dnorm( 0 , 0.5 ),
        wa[wa_id] ~ dnorm( 0 , 0.5 ),
        ja[ja_id] ~ dnorm( 0 , 0.5 ),
        fw ~ dnorm( 0 , 0.5 ),
        fj ~ dnorm( 0 , 0.5 ),
        wj ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp( 1 )
    ) , data=Wines2012, chains=1 )
```

```
##
## SAMPLING FOR MODEL 'a4bb86e242fa010e17913a3ff8be962d' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.61 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.575439 seconds (Warm-up)
## Chain 1:                0.502839 seconds (Sampling)
## Chain 1:                1.07828 seconds (Total)
## Chain 1:
```

```
precis(m3,depth=2)
```

```
##               mean         sd       5.5%       94.5%    n_eff     Rhat4
## f[1]   0.139085108 0.33252724 -0.3884905  0.68317972 411.9530 0.9981203
## f[2]  -0.061567660 0.30226335 -0.5489915  0.40299733 380.5640 0.9979981
## wa[1]  0.004507309 0.30807356 -0.5011267  0.51592333 298.2118 0.9985571
## wa[2]  0.047969686 0.31601651 -0.4796846  0.53865898 418.0809 0.9980090
## ja[1] -0.087743739 0.29923853 -0.5460598  0.40576226 419.3025 0.9980091
## ja[2]  0.158826980 0.30613790 -0.3400489  0.64092605 415.6532 0.9980192
```

```
## fw    -0.417896710 0.25263179 -0.8179031 -0.01784167 356.1487 0.9980666
## fj     0.060791156 0.24709589 -0.3267227  0.43370677 579.2815 0.9979999
## wj    -0.051544623 0.23556411 -0.4427515  0.29887794 587.6988 0.9979999
## sigma  0.993521014 0.05331053  0.9166649  1.08289660 547.2039 1.0018349
```