STATISTICAL RETHINKING WINTER 2020/2021
HOMEWORK, WEEK 5 SOLUTIONS

1. Load the data and construct the variables we'll need:

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

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

The model is straightforward. The only issue is the priors. Since I've standardized the outcome, we can use the ordinary N(0,0.5) prior from the examples in the text with standardized outcomes. Then the prior outcomes will stay largely within the possible outcome space. A bit more regularization than that wouldn't be a bad idea either.

```
m1 <- ulam(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- a[jid] + w[wid],
        a[jid] ~ dnorm(0,0.5),
        w[wid] ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ), data=dat_list , chains=4 , cores=4 )
```

Since this is your first MCMC homework, we'll spend some time inspecting the chains to ensure they worked. First, the diagnostics that `precis` provides:

```
precis( m1 , 2 )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat
a[1]  -0.27 0.20 -0.59  0.04  2259    1
a[2]   0.22 0.20 -0.10  0.51  2271    1
a[3]   0.20 0.19 -0.11  0.51  2249    1
a[4]  -0.54 0.20 -0.86 -0.22  2415    1
a[5]   0.80 0.20  0.46  1.11  2501    1
a[6]   0.48 0.19  0.17  0.78  2455    1
a[7]   0.14 0.20 -0.16  0.46  2398    1
```

```
a[8]  -0.66 0.19 -0.97 -0.36  2690    1
a[9]  -0.34 0.20 -0.66 -0.01  2120    1
w[1]   0.11 0.26 -0.30  0.53  3174    1
w[2]   0.09 0.26 -0.34  0.52  3122    1
w[3]   0.23 0.26 -0.20  0.65  3037    1
w[4]   0.46 0.25  0.07  0.86  2837    1
w[5]  -0.10 0.25 -0.51  0.30  3074    1
w[6]  -0.31 0.26 -0.74  0.11  2391    1
w[7]   0.25 0.26 -0.17  0.65  2955    1
w[8]   0.22 0.25 -0.18  0.63  2928    1
w[9]   0.07 0.27 -0.37  0.50  3293    1
w[10]  0.10 0.25 -0.32  0.50  2791    1
w[11] -0.02 0.25 -0.40  0.38  3664    1
w[12] -0.03 0.25 -0.44  0.37  2818    1
w[13] -0.09 0.26 -0.50  0.32  3396    1
w[14]  0.00 0.26 -0.41  0.42  4054    1
w[15] -0.19 0.26 -0.60  0.23  3296    1
w[16] -0.17 0.26 -0.59  0.23  3177    1
w[17] -0.12 0.25 -0.54  0.28  3427    1
w[18] -0.72 0.26 -1.14 -0.30  3020    1
w[19] -0.14 0.25 -0.53  0.26  3294    1
w[20]  0.32 0.26 -0.10  0.73  2960    1
sigma  0.85 0.05  0.77  0.93  3289    1
```

The `n_eff` values are all actually higher than the number of samples (2000), and all the `Rhat` values at exactly 1. Looks good so far. These diagnostics can mislead, however, so let's look at the trace plots too:

```
traceplot( m1 )
```

Result is in Figure 1. These pass the hairy-caterpillar-ocular-inspection-test: all the chains mix in the same region, and they move quickly through it, not getting stuck anyplace.

Now let's plot these parameters so they are easier to interpret:

```
plot( precis( m1 , 2 ) )
```

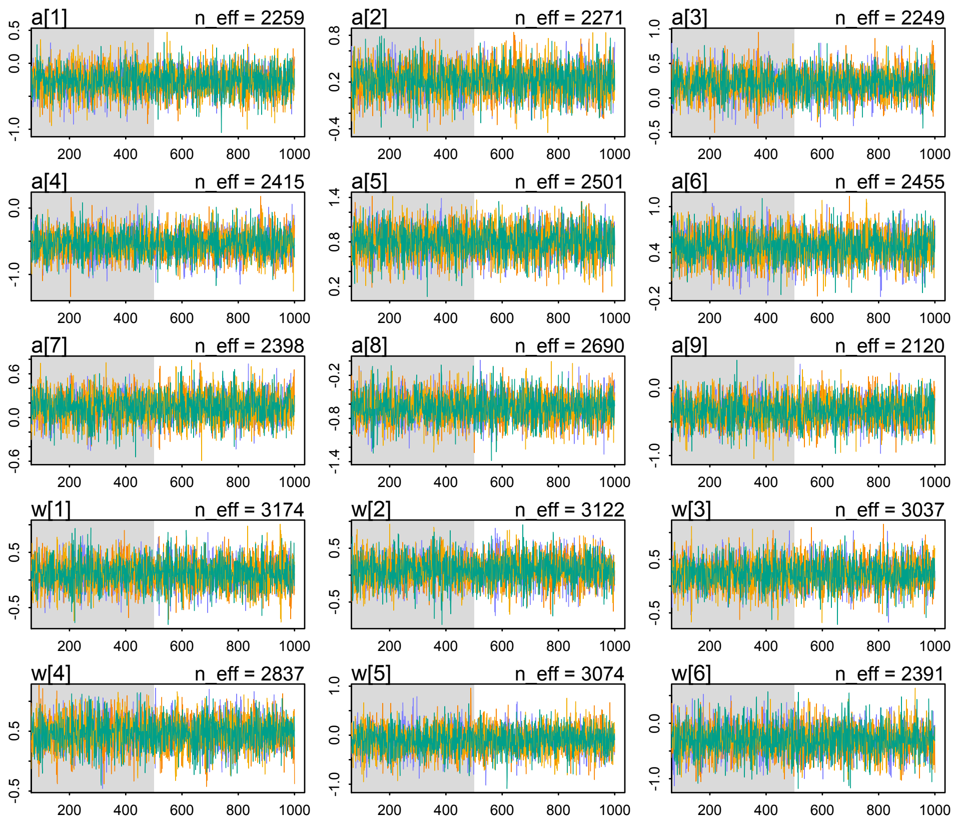
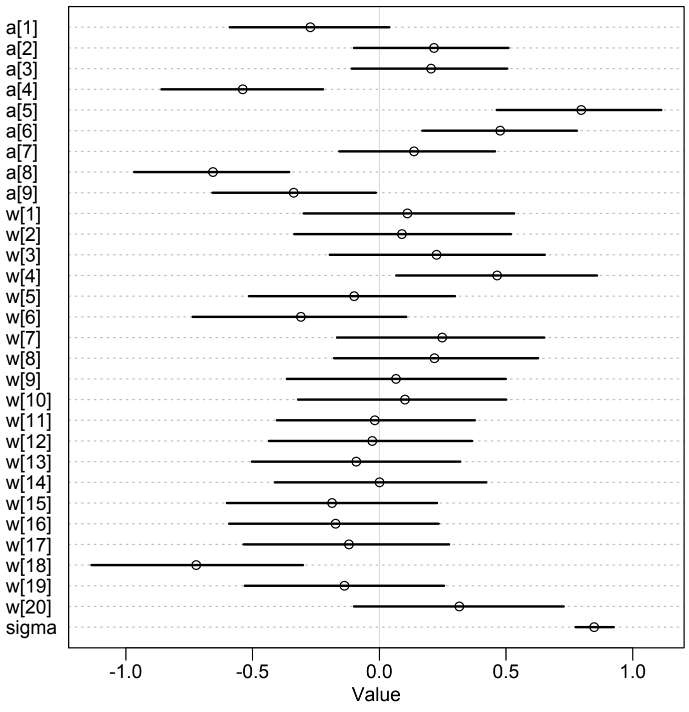


FIGURE 1. First page of trace plot output for `m1`.



The a parameters are the judges. Each represents an average deviation of the scores. So judges with lower values are harsher on average. Judges with higher values liked the wines more on average. There is some noticeable variation here. It is fairly easy to tell the judges apart.

The `w` parameters are the wines. Each represents an average score across all judges. Except for wine 18 (a New Jersey red I think), there isn't that much variation. These are good wines, after all. Overall, there is more variation from judge than from wine.

2. The easiest way to code the data is to use indicator variables. Let's look at that approach first. I'll do an index variable version next. I'll use the three indicator variables `W` (NJ wine), `J` (American NJ), and `R` (red wine).

```
dat_list2 <- list(
    S = standardize(d$score),
    W = d$wine.amer,
    J = d$judge.amer,
    R = ifelse(d$flight=="red",1L,0L)
)
```

The model structure is just a linear model with an ordinary intercept. I'll put a relatively tight prior on the intercept, since it must be near zero (centered outcome). What about the coefficients for the indicator variables? Let's pretend we haven't already seen the results from Problem 1—there aren't any big wine differences to find there. Without that cheating foresight, we should consider what the most extreme effect could be. How big could the difference between NJ and French wines be? Could it be a full standard deviation? If so, then maybe a Normal(0,0.5) prior makes sense, since they place a full standard deviation difference out in the tails of the prior. I'd personally be inclined to something even tighter, so that it regularizes more. But let's go with these wide priors, which nevertheless stay within the outcome space. It would make even more sense to put a tighter prior on the difference between red and white wines—on average they should be the no different, because judges only compare within flights. Here's the model:

```
m2a <- ulam(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- a + bW*W + bJ*J + bR*R,
        a ~ dnorm( 0 , 0.2 ),
        c(bW,bJ,bR) ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp(1)
    ), data=dat_list2 , chains=4 , cores=4 )
precis( m2a )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat
a     -0.01 0.12 -0.21  0.20  1405    1
```

```
bR      0.00 0.13 -0.22  0.21  1644    1
bJ      0.22 0.14  0.01  0.44  1605    1
bW     -0.18 0.14 -0.40  0.04  1456    1
sigma  1.00 0.05  0.92  1.09  1660    1
```

As expected, red and wines are on average the same—`bR` is right on top of zero. American judges seem to be more on average slightly more generous with ratings—`bJ` is slightly but reliably above zero. American wines have slightly lower average ratings than French wines—`bW` is mostly below zero, but not very large in absolute size.

Okay, now for an index variable version. The thing about index variables is that you can easily end up with more parameters than in an equivalent indicator variable model. But it's still the same posterior distribution. You can convert from one to the other (if the priors are also equivalent). We'll need three index variables:

```
dat_list2b <- list(
    S = standardize(d$score),
    wid = d$wine.amer + 1L,
    jid = d$judge.amer + 1L,
    fid = ifelse(d$flight=="red",1L,2L)
)
```

Now `wid` is 1 for a French wine and 2 for a NJ wine, `jid` is 1 for a French judge and 2 for an American judge, and `fid` is 1 for red and 2 for white. Those `1L` numbers are just the R way to type the number as an integer—"`1L`" is the integer 1, while "`1`" is the real number 1. We want integers for an index variable.

Now let's think about priors for the parameters that correspond to each index value. Now the question isn't how big the difference could be, but rather how far from the mean an indexed category could be. If we use Normal(0,0.5) priors, that would make a full standard deviation difference from the global mean rare. It will also match what we had above, in a crude sense. Again, I'd be tempted to something narrow, for the sake of regularization. But certainly something like Normal(0,10) is flat out silly, because it makes impossible values routine. Let's see what we get:

```
m2b <- ulam(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- w[wid] + j[jid] + f[fid],
        w[wid] ~ dnorm( 0 , 0.5 ),
        j[wid] ~ dnorm( 0 , 0.5 ),
        f[wid] ~ dnorm( 0 , 0.5 ),
        sigma ~ dexp(1)
    ), data=dat_list2b , chains=4 , cores=4 )
precis( m2b )
```

```
       mean   sd  5.5% 94.5% n_eff Rhat
w[1]   0.09 0.29 -0.38  0.54   593    1
w[2]  -0.10 0.29 -0.56  0.34   626    1
j[1]  -0.11 0.30 -0.57  0.35   446    1
j[2]   0.13 0.30 -0.33  0.61   621    1
f[1]   0.00 0.29 -0.47  0.47   840    1
f[2]   0.00 0.29 -0.48  0.47   825    1
sigma  1.00 0.05  0.92  1.09  1457    1
```

To see that this model is the same as the previous, let's compute contrasts. The contrast between American and French wines is:

```
post <- extract.samples(m2b)
diff_w <- post$w[,2] - post$w[,1]
precis( diff_w )
```

```
'data.frame': 2000 obs. of 1 variables:
        mean   sd  5.5% 94.5%   histogram
diff_w -0.19 0.15 -0.43  0.06 ▁▁▁▁▂▅▇▇▃▁▁▁
```

That's almost exactly the same mean and standard deviation as `bW` in the first model. The other contrasts match as well.

Something to notice about the two models is that the second one does sample less efficiently. The `n_eff` values are lower. This isn't a problem, but it is a consequence of the higher correlations in the posterior, a result of the redundant parameterization. If you look at the `pairs(m2b)` plot, you'll see tight correlations for each pair of index parameters of the same type. This is because really it is a difference that matters, and many combinations of two numbers can produce the same difference. But the priors keep this from ruining our inference. if you tried the same thing without priors, it would likely fall apart and return very large standard errors.

3. Again I'll show both the indicator approach and the index variable approach.

For the indicator approach, we can use the same predictor variables as before:

```
dat_list2 <- list(
    S = standardize(d$score),
    W = d$wine.amer,
    J = d$judge.amer,
    R = ifelse(d$flight=="red",1L,0L)
)
```

It's the model that is different.

```
m3 <- ulam(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- a + bW*W + bJ*J + bR*R +
              bWJ*W*J + bWR*W*R + bJR*J*R,
        a ~ dnorm(0,0.2),
        c(bW,bJ,bR) ~ dnorm(0,0.5),
        c(bWJ,bWR,bJR) ~ dnorm(0,0.25),
        sigma ~ dexp(1)
    ), data=dat_list2 , chains=4 , cores=4 )
```

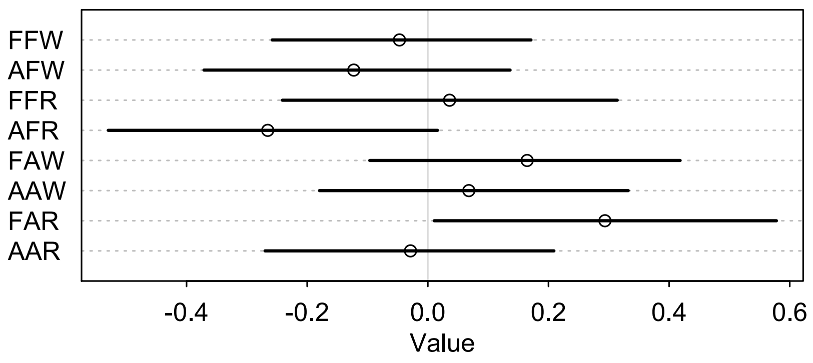
I used the same priors as before for the main effects. I used tighter priors for the interactions. Why? Because interactions represent sub-categories of data, and if we keep slicing up the sample, differences can't keep getting bigger. Again, the most important thing is not to use flat priors like Normal(0,10) that produce impossible outcomes.

```
precis(m3)
```

```
       mean   sd  5.5% 94.5% n_eff Rhat
a     -0.05 0.13 -0.26  0.17  1342    1
bR     0.08 0.18 -0.20  0.38  1406    1
bJ     0.21 0.18 -0.08  0.50  1498    1
bW    -0.08 0.17 -0.35  0.20  1249    1
bJR    0.05 0.18 -0.25  0.34  1888    1
bWR   -0.23 0.19 -0.52  0.06  1842    1
bWJ   -0.02 0.19 -0.32  0.28  1844    1
sigma  1.00 0.05  0.92  1.09  2030    1
```

Reading the parameters this way is not easy. But right away you might notice that `bW` is now close to zero and overlaps it a lot on both sides. NJ wines are no longer on average worse. So the interactions did something. Glancing at the interaction parameters, you can see that only one of them has much mass away from zero, `bWR`, the interaction between NJ wines and red flight, so red NJ wines. To get the predicted scores for red and white wines from both NJ and France, for both types of judges, we can use `link`:

```
pred_dat <- data.frame(
    W = rep( 0:1 , times=4 ),
    J = rep( 0:1 , each=4 ),
    R = rep( c(0,0,1,1) , times=2 )
)
mu <- link( m3 , data=pred_dat )
row_labels <- paste( ifelse(pred_dat$W==1,"A","F") ,
                     ifelse(pred_dat$J==1,"A","F") ,
                     ifelse(pred_dat$R==1,"R","W") , sep="" )
plot( precis( list(mu=mu) , 2 ) , labels=row_labels )
```



I've added informative labels. FFW means: French wine, French judge, White wine. So the first four rows are as judged by French judges. The last four are as judged by American judges. The two rows that jump out are the 4th and the 2nd-to-last, AFR and FAR. Those are NJ red wines as judged by French judges and French red wines as judged by American judges. French judges didn't like NJ reds so much (really only one NJ red, if you look back at Problem 1). And American judges liked French reds more. Besides these two interactions, notice that it is very hard to figure this out from the table of coefficients.

Now let's do an index version. The way to think of this is to make unique parameters for each combination. If we consider all the interactions—including a three-way interaction between nation, judge and flight—there would be 8 combinations and so 8 parameters to estimate. Let's go ahead and do that, so we can simultaneously consider the 3-way interaction.

There are several ways to go about coding this. I'm going to use a trick and make an array of parameters. An array is like a matrix, but can have more than 2 dimensions. If we make a 2-by-2-by-2 array of parameters, then there will be 8 parameters total and we can access each by just using these index variables:

```
dat_list2b <- list(
    S = standardize(d$score),
    wid = d$wine.amer + 1L,
    jid = d$judge.amer + 1L,
    fid = ifelse(d$flight=="red",1L,2L)
)
```

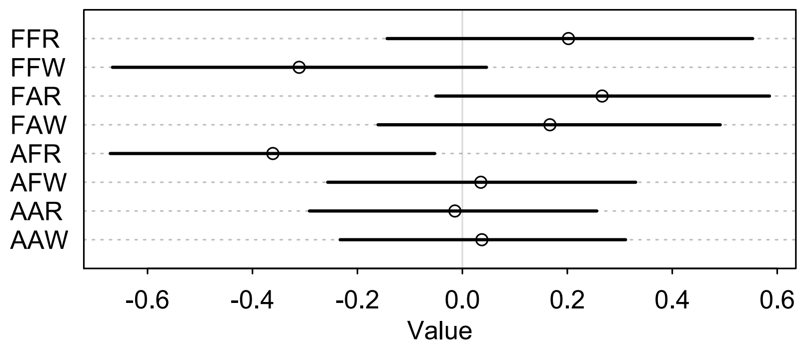
I'll build the model in raw Stan code. The way to define an array is just to use the dimensions you want after the parameter name, when you declare it in the parameters block. Like this:

```
mcode <- "
data{
    vector[180] S;
    int fid[180];
    int jid[180];
    int wid[180];
}
parameters{
    real w[2,2,2];
    real<lower=0> sigma;
}
model{
    vector[180] mu;
    sigma ~ exponential( 1 );
    for ( i in 1:2 )
        for ( j in 1:2 )
            for ( k in 1:2 )
                w[i,j,k] ~ normal( 0 , 0.5 );
    for ( i in 1:180 ) {
        mu[i] = w[wid[i], jid[i], fid[i]];
    }
    S ~ normal( mu , sigma );
}
"

m3b <- stan( model_code=mcode , data=dat_list2b , chains=4 , cores=4 )

row_labels = c("FFR","FFW","FAR","FAW","AFR","AFW","AAR","AAW" )
plot( precis( m3b , 3 , pars="w" ) , labels=row_labels )
```

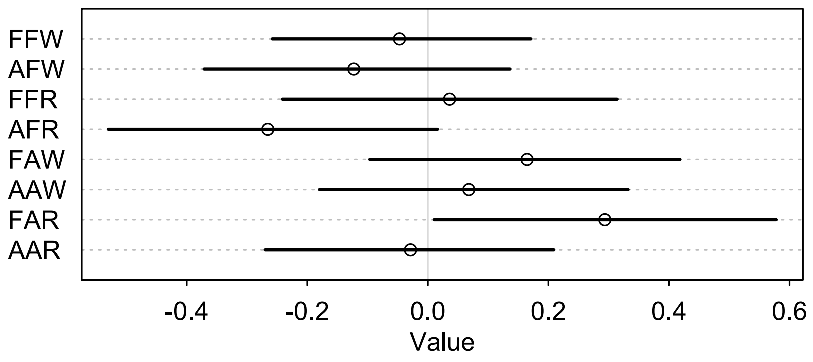
You could also do this model in `ulam` (as of rethinking 1.84+):

```
m3b <- ulam(
    alist(
        S ~ dnorm( mu , sigma ),
        mu <- w[wid,jid,fid],
        real['2,2,2']:w ~ normal(0,0.5),
        sigma ~ dexp(1)
    ), data=dat_list2b , chains=4 , cores=4 )
```

The `'2,2,2'` literal is not very elegant, so I'm likely to improve this is a later version. Either way, the result is displayed on the next page (for sake of line breaks).



The previous predictions, for comparison:



The most noticeable change is that FFW (French wines, French judges, white) have a lower expected rating in the full interaction model. There are some other minor differences as well. What has happened? The three way interaction would be, in the first model's indicator terms, when a wine is American, the judge is American, and the flight is red. In the first model, a prediction for such a wine is just a sum of parameters:

$$\mu_i = \alpha + \beta_W + \beta_J + \beta_R + \beta_{WJ} + \beta_{WR} + \beta_{JR}$$

This of course limits means that these parameters have to account for the AAR wine. In the full interaction mode, an AAR wine gets its own parameter, as does every other combination. None of the parameters get polluted by averaging over different combinations. Of course, there isn't a lot of evidence that prediction is improved much by allowing this extra parameter. The differences are small, overall. These wines are all quite good. But it is worth understand how the full interaction model gains additional flexibility. This additional flexibility typically requires some addition regularization. When we arrive at multilevel models later, you'll see how we can handle regularization more naturally inside of a model.