Homework 5

Question 7E1

```
data(Wines2012)
d <- Wines2012

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

```
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 , cmdstan=TRUE )
```

```
## This is cmdstanr version 0.4.0.9000

## - Online documentation and vignettes at mc-stan.org/cmdstanr

## - CmdStan path set to: C:/Users/Public/.cmdstanr/cmdstan-2.26.1

## - Use set_cmdstan_path() to change the path

## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## 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 2 Iteration:    1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration:  200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration:  300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration:  400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration:  500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration:  501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration:  600 / 1000 [ 60%]  (Sampling)
```

```
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.4 seconds.
## Chain 2 finished in 0.4 seconds.
## Chain 3 finished in 0.4 seconds.
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 0.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.4 seconds.
## Total execution time: 16.3 seconds.
```

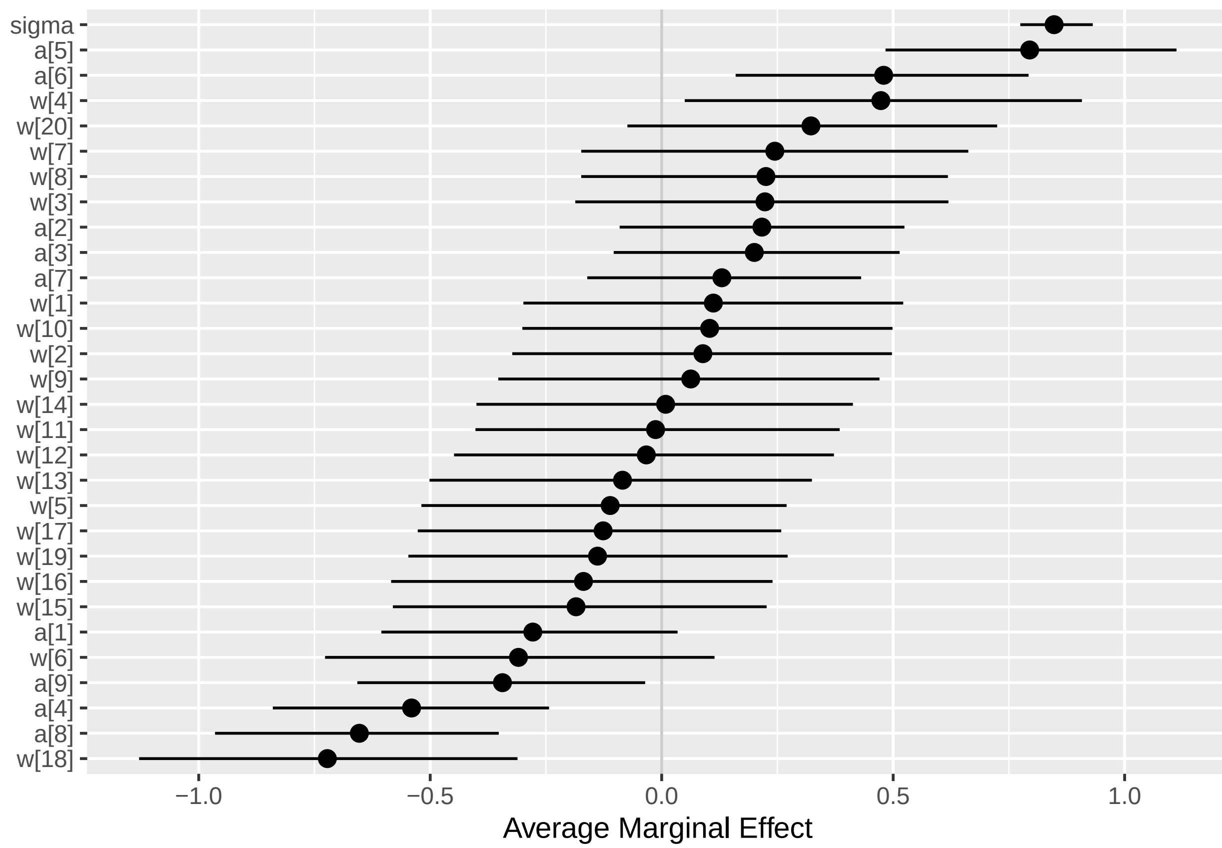
```r
coef <- precis(m1,2)
coef$name <- row.names(coef)
coef
```

```
##              mean         sd        5.5%       94.5%    n_eff     Rhat4  name
## a[1]  -0.278369941 0.19836766 -0.60551121  0.03429281 2079.139 0.9991343  a[1]
## a[2]   0.216456486 0.19622297 -0.09082728  0.52435759 2580.321 0.9998527  a[2]
## a[3]   0.200104305 0.19163527 -0.10369016  0.51390049 1821.868 0.9996002  a[3]
## a[4]  -0.540226716 0.18488197 -0.83991140 -0.24324385 1972.856 0.9991558  a[4]
## a[5]   0.794841525 0.19553095  0.48345563  1.11202335 1903.142 0.9998754  a[5]
## a[6]   0.479505035 0.19477978  0.15969242  0.79251526 2007.390 0.9987834  a[6]
## a[7]   0.130030896 0.18793104 -0.16076955  0.43075139 1935.802 1.0002912  a[7]
## a[8]  -0.653030769 0.19541151 -0.96473514 -0.35189402 2215.136 0.9998671  a[8]
## a[9]  -0.343943575 0.19326402 -0.65744228 -0.03563197 1897.402 1.0007348  a[9]
## w[1]   0.111629041 0.26026794 -0.29871574  0.52174931 2311.686 0.9997940  w[1]
## w[2]   0.088988940 0.26034975 -0.32282386  0.49735867 2741.550 0.9995385  w[2]
## w[3]   0.223028366 0.25216970 -0.18659805  0.61938039 2275.948 0.9995122  w[3]
## w[4]   0.473399408 0.26792422  0.04980580  0.90771546 2497.025 1.0000725  w[4]
```

```
## w[5]   -0.111224528 0.25303616 -0.51899946  0.26981864 2405.535 1.0026407  w[5]
## w[6]   -0.309337632 0.26901595 -0.72713600  0.11418881 2280.077 0.9997239  w[6]
## w[7]    0.244371052 0.26050774 -0.17376595  0.66235622 3115.295 0.9994760  w[7]
## w[8]    0.225154236 0.24632094 -0.17381744  0.61822779 2786.631 0.9986483  w[8]
## w[9]    0.062779795 0.25884105 -0.35297428  0.47045297 2129.454 0.9992560  w[9]
## w[10]   0.103594078 0.25329866 -0.30096901  0.49853992 2698.399 1.0001735 w[10]
## w[11]  -0.013161483 0.24385493 -0.40231350  0.38447709 2336.517 1.0000985 w[11]
## w[12]  -0.033202015 0.25659890 -0.44852738  0.37208707 2524.656 0.9993038 w[12]
## w[13]  -0.084641074 0.25924034 -0.50152080  0.32446323 2675.612 1.0000929 w[13]
## w[14]   0.008565749 0.25394390 -0.40011122  0.41305019 2831.435 0.9986902 w[14]
## w[15]  -0.185004878 0.25334865 -0.58059889  0.22674283 2429.259 0.9989091 w[15]
## w[16]  -0.168949156 0.25602021 -0.58449120  0.23952499 2670.744 1.0012947 w[16]
## w[17]  -0.126470609 0.24569248 -0.52689845  0.25829822 2231.416 1.0016168 w[17]
## w[18]  -0.722159481 0.26316874 -1.12877660 -0.31140707 2809.579 0.9998947 w[18]
## w[19]  -0.138577934 0.25534883 -0.54717580  0.27202568 3020.121 0.9990070 w[19]
## w[20]   0.322422653 0.24935615 -0.07407082  0.72463585 2653.750 0.9995331 w[20]
## sigma   0.847624633 0.05008081  0.77453753  0.93116847 2457.926 0.9994926 sigma
```

```
p <- ggplot(data = coef, aes(x = reorder(name, mean),
                             y = mean, ymin = `5.5%`, ymax = `94.5%`))

p + geom_hline(yintercept = 0, color = "gray80") +
    geom_pointrange() + coord_flip() +
    labs(x = NULL, y = "Average Marginal Effect")
```




```
d2 <- as.data.frame(dat_list) %>%
  mutate(jid=as.factor(jid)) %>%
```

```r
  mutate(wid=as.factor(wid))
head(d2)
```

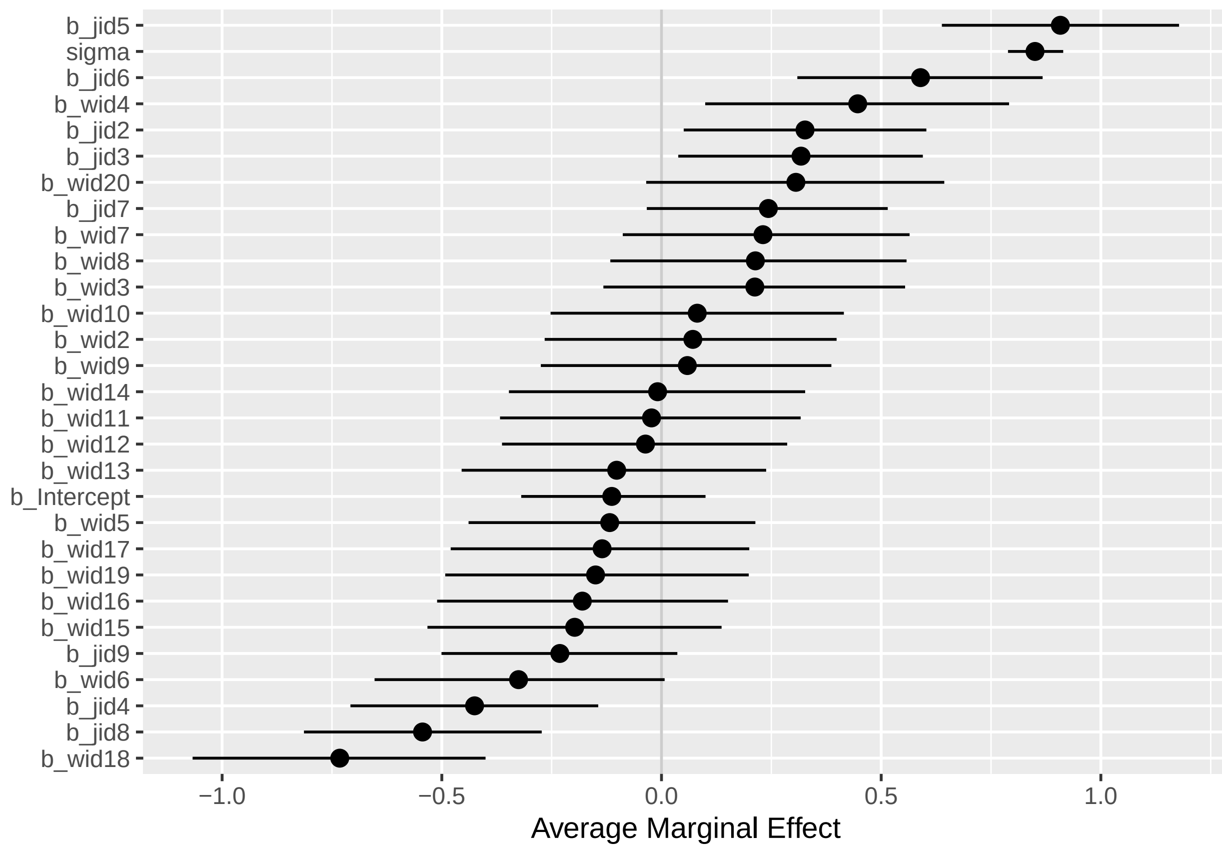
```
##              S jid wid
## 1 -1.57660412   4   1
## 2 -0.45045832   4   3
## 3 -0.07507639   4   5
## 4  0.30030555   4   7
## 5 -2.32736799   4   9
## 6 -0.45045832   4  11
```

```r
df <- as.data.frame(posterior_summary(m1, probs = c(0.1, 0.9))) %>% filter(Estimate > -10)

df$name <- row.names(df)

p <- ggplot(data = df, aes(x = reorder(name, Estimate  ),
                              y = Estimate  , ymin = Q10, ymax = Q90))

p + geom_hline(yintercept = 0, color = "gray80") +
    geom_pointrange() + coord_flip() +
    labs(x = NULL, y = "Average Marginal Effect")
```




```r
d3 <- d %>%
  mutate(wine.amer=factor(wine.amer, labels=c('French', 'American')) ) %>%
  mutate(judge.amer=factor(judge.amer, labels=c('French', 'American') )) %>%
  mutate(score=standardize(score)) %>%
  select(score, flight, wine.amer, judge.amer)
```

```r
head(d3)
```

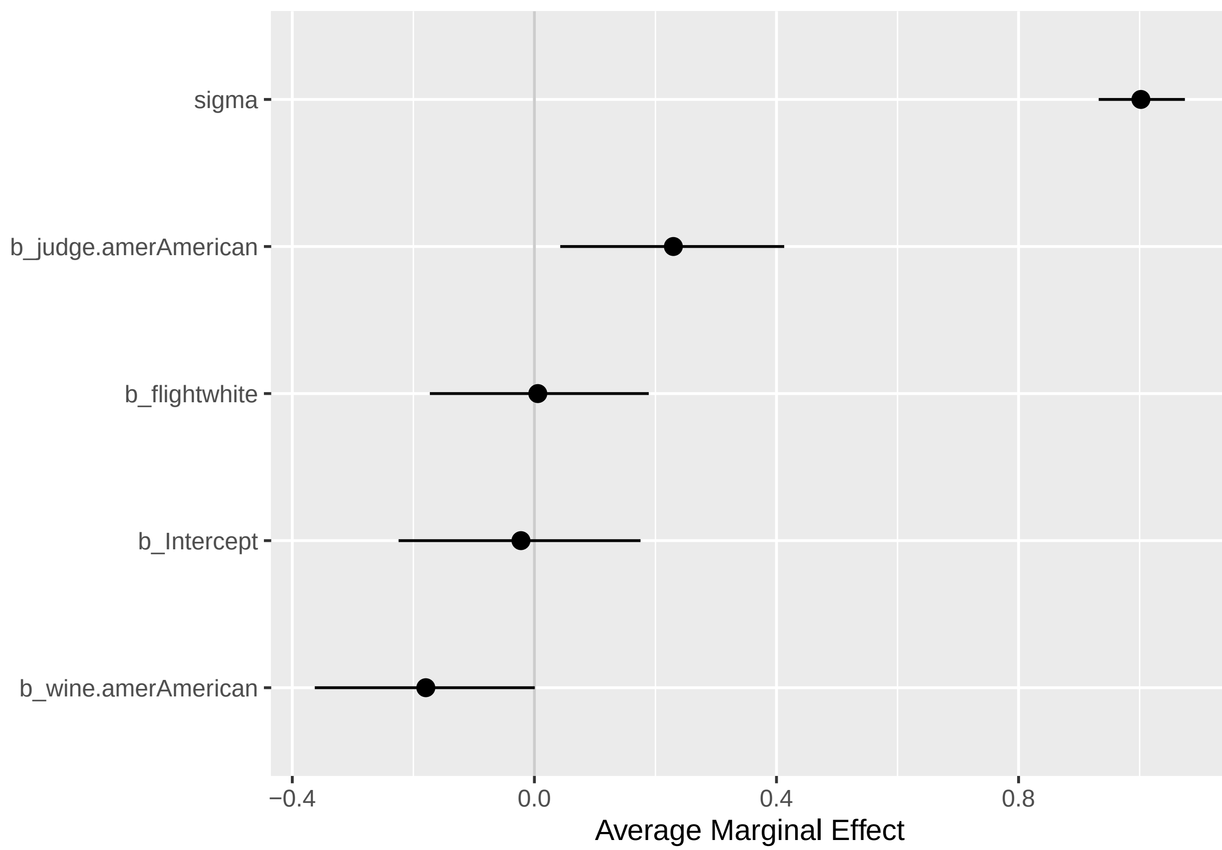
```
##         score flight wine.amer judge.amer
## 1 -1.57660412  white  American     French
## 2 -0.45045832  white  American     French
## 3 -0.07507639  white    French     French
## 4  0.30030555  white    French     French
## 5 -2.32736799  white  American     French
## 6 -0.45045832  white  American     French
```

```r
df <- as.data.frame(posterior_summary(m3, probs = c(0.1, 0.9))) %>% filter(Estimate > -10)

df$name <- row.names(df)

p <- ggplot(data = df, aes(x = reorder(name, Estimate  ),
                              y = Estimate  , ymin = Q10, ymax = Q90))

p + geom_hline(yintercept = 0, color = "gray80") +
    geom_pointrange() + coord_flip() +
    labs(x = NULL, y = "Average Marginal Effect")
```




```r
d3s <- d %>%
  mutate(S=standardize(score)) %>%
  mutate(wid= wine.amer + 1L) %>%
  mutate(jid= judge.amer + 1L) %>%
  rowwise %>%
  mutate(fid =ifelse(flight == 'white', 2L, 1L)) %>%
```

```r
  select(S, wid , jid, fid)
head(d3s)
```

```
## # A tibble: 6 x 4
## # Rowwise:
##         S   wid   jid   fid
##     <dbl> <int> <int> <int>
## 1 -1.58       2     1     2
## 2 -0.450      2     1     2
## 3 -0.0751     1     1     2
## 4  0.300      1     1     2
## 5 -2.33       2     1     2
## 6 -0.450      2     1     2
```

```r
row_labels = c("FFR", "FFW", "FAR", "FAW", "AFR", "AFW", "AAR", "AAW")

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);
}
"

m3s <- stan(model_code = mcode, data=d3s, cores=4)
```

```r
row_labels = c("FFR", "FFW", "FAR", "FAW", "AFR", "AFW", "AAR", "AAW")
```

```r
results <- precis(m3s, 3, pars='w')
row.names(results) <- row_labels
head(results)
```

```
##            mean        sd        5.5%       94.5%    n_eff     Rhat4
## FFR  0.20278341 0.2216720 -0.15009120  0.55797080 7082.610 0.9995729
## FFW -0.30805499 0.2270568 -0.67608717  0.04764032 6761.877 0.9999312
## FAR  0.26732628 0.1991237 -0.05047396  0.58758671 7459.715 0.9998276
## FAW  0.16473986 0.1970029 -0.14452314  0.47897919 7925.183 0.9991866
## AFR -0.36076719 0.1878611 -0.66209213 -0.06414612 8047.008 0.9994777
## AFW  0.03847544 0.1863493 -0.26443879  0.33234038 7081.129 0.9991524
```