Lab 5 (worksheet version)

Schedule for today

- We look at three ways to model regression models today
  - the index method
  - the indicator method
  - the model matrix method
- We will also see how to implement them in three ways
  - quap
  - stan
  - rjags
- We will fill in the blanks in the code together. After that you're welcome to take a look at your homework and we will have a Q&A session about the HW4
- If we have extra time, we will go over previous homeworks
- The full version will be posted tomorrow

8H5

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 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. 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?

Index Model

First we write down the model:

$$
\begin{aligned}
s_i &\sim N(\mu_i, \sigma^2) \\
\mu_i &= \alpha_{\text{JID}[i]} + w_{\text{WID}[i]} \\
\alpha_{\text{JID}[i]} &\sim N(0, 0.5^2) \\
\beta_{\text{WID}[i]} &\sim N(0, 0.5^2) \\
\sigma &\sim \text{Exp}(1)
\end{aligned}
$$

preparing data

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

```
# have a look at the ids
str(dat_list)
```

quap() code

```
m1 <- quap(
    alist(
      # likelihood: score distributed as normal, mean is mu, std is sigma

      # mean function: mean broken into two groups, judge effect(a) and wine effect(w)
      # indexed by their ids

      # priors




    ),
    # data
    data=dat_list
    )
```

rjags() code

```
m1_code <- "
  \\ the data object we read in will also be here
  data {
    D <- dim(S)
    n <- D[1]
  }
  model{
  # in JAGS we tend to specify everything iteratively
  # likelihood
   for(i in 1:n){
      # response

      # bracket notation
      # mean function

      # posterior predictive
      # ynew[i] ~ dnorm(mu[i], tau)
   }
    # priors
    # beware of the precision specification
    for (j in 1:Jid) {

    }
    for (j in 1:Wid) {

    }
    sigma ~ dexp(lambda)
    tau <- pow(sigma, -2)
  }
```

```
"
m8.5.jags <- jags.model(
  file = textConnection(m1_code),
  data = list(
    S =
    jid =
    wid =
    Jid =
    Wid =
    ma =
    sa =
    mb =
    sb =
    lambda = 1
  )
)
```

```
m8.5.samps <- coda.samples(m8.5.jags,
                           variable.names = c("a", "b"),
                           n.iter = 1e4)
m8.5.samps.df <- as.data.frame(m8.5.samps[[1]])
plot(precis(m8.3.samps.df, depth = 2))
```

view results

STAN code

```
library(rstan)
model_code <- "
  data{
    // data
    int<lower=1> N; // number of observations
    vector[N] S; // obsrevations

    // grouping factors
     // number of judge ids
     // judge ids
     // number of judge ids
     // judge ids

    // hyperparameters: feed in the values
    real ma;
    real mb;
    real sa;
    real sb;
    real lambda;
  }
  // important! for parameters you want to set their ranges
  parameters{
    vector[Jid] a; // judge effect
    vector[Wid] b; // wine effect
    real<lower=0> sigma;
```

```
  }
  model{
    // declare variable for the expected outcome
    // important! mu is a local variable
    // in stan we need to put mu in the model, instead of the parameter
    vector[N] mu;
    // priors
    a ~ normal(ma, sa);
    b ~ normal(mb, sb);
    sigma ~ exponential(lambda);
    // likelihood
    for (i in 1:N) {
      mu[i] = a[jid[i]] + b[wid[i]];
    }
    S ~ normal(mu, sigma);
  }
"
stan.out <- stan(
  model_code = model_code,
  iter = 1e4, # length of the markov chain
  data = list(
    N = length(dat_list$S),
    S = dat_list$S,
    Jid = max(dat_list$jid),
    jid = dat_list$jid,
    Wid = max(dat_list$wid),
    wid = dat_list$wid,
    ma = 0,
    sa = 0.5,
    mb = 0,
    sb = 0.5,
    lambda = 1
  )
)
print(stan.out)
stan_plot(stan.out, pars = c("a", "b"))
```

Model Matrix

Equivalently we can re-express this model in matrix form. Suppose we group the 'wine effects' and the 'judge effects' in to vectors: $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_9)^T$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_{20})^T$. And then we build two model matrix, $\mathbf{X}_{\alpha n \times 20}, \mathbf{X}_{\beta_{n \times 9}}$ that indicates the allocation of each data row:

$$\mathbf{X}_{\alpha i,j} = 1, \text{ if the i-th score is given by judge } j, \text{ o.w. } \mathbf{X}_{\alpha i,j} = 0$$

The same goes for $\mathbf{X}_{\beta}$.

Preparing the data

```
dat_x <- data.frame(
  score = standardize(d$score),
  judge = (d$judge),
```

```
  wine = (d$wine)
)
# making the model matrix
x <- model.matrix(~ -1 + judge + wine,
                  data = dat_x,
                  contrasts.arg = list(wine = contrasts(d$wine, contrasts = FALSE)))
# view the data
x[1,]
dat_x[1,]
# break into two matrices
x1 <- x[,1:9]
x2 <- x[,10:29]
```

```
m1_x <- quap(
    alist(
      # likelihood: score distributed as normal, mean is mu, std is sigma
        S ~ dnorm( mu , sigma ),
      # mean function: mean broken into two groups, judge effect(a) and wine effect(w)
      # indexed by their ids
      # quap adopts the matrix multiplication operator in R
        mu <-
      # priors
        a ~ dnorm(0,0.5),
        b ~ dnorm(0,0.5),
        sigma ~ dexp(1)
    ),
    # data
    data=list(
      x1 = x1,
      x2 = x2,
      S = standardize(d$score)
    ),
    # important: in previous code we do not specify the length of a and b
    # a trick we saw in the splines code is to use start()
    # length of vector a is just columns of Xa
    start=list(
      #a =
      #b =
    )
)
plot( precis( m1_x , 2 ) )
```

JAGS

```
m1_code <- "
  data {
    D <- dim(S)
    n <- D[1]
  }
  model{
   for(i in 1:n){
      # likelihood
      S[i] ~ dnorm(mu[i], tau)
```

```
      # posterior predictive
      # ynew[i] ~ dnorm(mu[i], tau)
   }
   # in rjags we can also use the matrix multiplication
   mu =
   # conditional mean using matrix algebra
   for (j in 1:Jid) {
     a[j] ~ dnorm(ma, pow(sa, -2))
   }
   for (j in 1:Wid) {
     b[j] ~ dnorm(mb, pow(sb, -2))
   }
   sigma ~ dexp(lambda)
   tau <- pow(sigma, -2)
}
"
m8.3.jags <- jags.model(
  file = textConnection(m1_code),
  data = list(
    S = dat_list$S,
    Jid = max(dat_list$jid),
    Wid = max(dat_list$wid),
    x1 = x1,
    x2 = x2,
    ma = 0,
    sa = 0.5,
    mb = 0,
    sb = 0.5,
    lambda = 1
  )
)
```

STAN

```
library(rstan)
model_code <- "
  data{
    // data
    int<lower=1> N; // number of observations
    real S[N]; // obsrevations

    // matrices corresponding to grouping factors
    int<lower=1> Jid; // number of judge ids
    int<lower=1> Wid; // number of wine ids
    matrix[N,Jid] x1; // model matrix for judge
     x2; // model matrix for wine

    // hyperparameters
    real ma;
    real mb;
    real sa;
    real sb;
    real lambda;
```

```
  }
  parameters{
    vector[Jid] a; // judge effect
    vector[Wid] b; // wine effect
    real<lower=0> sigma;
  }
  model{
    real mu[N];
    // priors
    a ~ normal(ma, sa);
    b ~ normal(mb, sb);
    sigma ~ exponential(lambda);
    // likelihood
    // in stan * suffices for matrix multiplication
    S ~ normal(, sigma);
  }
"
stan.out <- stan(
  model_code = model_code,
  iter = 1e4,
  data = list(
    N = length(dat_list$S),
    S = dat_list$S,
    x1 = x1,
    x2 = x2,
    Jid = max(dat_list$jid),
    Wid = max(dat_list$wid),
    ma = 0,
    sa = 0.5,
    mb = 0,
    sb = 0.5,
    lambda = 1
  )
)
print(stan.out)
stan_plot(stan.out, pars = c("a", "b"))
```

8H6

Now consider three features of the wines and judges:

- flight: Whether the wine is red or white.
- wine.amer: Indicator variable for American wines.
- 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 justify your priors. What do you conclude about the differences among the wines and judges? Try to relate the results to the inferences in the previous problem.

Indicator Model

We can build the following model:

$$
\begin{aligned}
s_i &\sim N(\mu_i, \sigma^2) \\
\mu_i &= a + \beta_W W_{amer,i} + \beta_J J_{amer,i} + \beta_R R_i \\
a &\sim N(0, 0.2^2) \\
b_W, b_J, b_R &\sim N(0, 0.5^2) \\
\sigma &\sim \text{Exp}(1)
\end{aligned}
$$

Preprocess the variables:

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

QUAP

```
m2a <- quap(alist(
  # model
  S ~ dnorm(mu , sigma),
  # linear combination of mean function
  mu <-
  # priors
  a ~ dnorm(0 , 0.2),
  c(bW, bJ, bR) ~ dnorm(0 , 0.5),
  sigma ~ dexp(1)
),
data = dat_list2)
precis(m2a)
```

JAGS

```
m2_code <- "
  data {
    D <- dim(S)
    n <- D[1]
  }
  model{
   for(i in 1:n){
      # likelihood
      S[i] ~ dnorm(mu[i], tau)

      # posterior predictive
      # ynew[i] ~ dnorm(mu[i], tau)
   }
   mu =
   a ~ dnorm(0 ,pow(0.2,-2))
   bW ~ dnorm(mb, pow(sb,-2))
   bJ ~ dnorm(mb, pow(sb,-2))
   bR ~ dnorm(mb, pow(sb,-2))
```

```
   sigma ~ dexp(lambda)
   tau <- pow(sigma, -2)
}
"
m8.3.jags <- jags.model(file = textConnection(m2_code),
                        data = list(S = dat_list$S,
                                    W = d$wine.amer,
                                    J = d$judge.amer,
                                    R = ifelse(d$flight=="red",1L,0L),
                                    mb = 0,
                                    sb = 0.5,
                                    lambda = 1))
```

STAN

```
library(rstan)
model_code <- "
  data{
    // data
    int<lower=1> N; // number of observations
    vector[N] S; // obsrevations
    vector[N] W; // wine america
    vector[N] J; // judge america
    vector[N] R; // wine color type

    // hyperparameters
    real ma;
    real mb;
    real sa;
    real sb;
    real lambda;
  }
  parameters{
    real a, bW, bJ, bR;
    real<lower=0> sigma;
  }
  model{
    real mu[N];
    // priors
    a ~ normal(ma, sa);
    bW ~ normal(mb, sb);
    bJ ~ normal(mb, sb);
    bR ~ normal(mb, sb);
    sigma ~ exponential(lambda);
    // likelihood
    S ~ normal(, sigma);
  }
"
stan.out <- stan(model_code = model_code, iter = 1e4,
                 data = list(N = length(d$score),
                             S = standardize(d$score),
                             W = d$wine.amer,
                             J = d$judge.amer,
```

```
                              R = ifelse(d$flight=="red",1L,0L),
                              ma = 0,sa = 0.2,
                              mb = 0,sb = 0.5,lambda = 1))
print(stan.out)
stan_plot(stan.out, pars = c("a","b"))
```