3.4 Practical: categorical predictor

Benjamin Rosenbaum

November 4, 2021

In this exercise, we will learn different ways to code a categorical predictor. We will also learn how to use the `generated quantities{}` block.

Suppose we have measurements of a continuous variable in 2 groups, e.g. individual body masses in 2 populations.

We want to test if both groups have a different mean.

statistical model:

$$y_{1,i} \sim \text{normal}(\mu_1, \sigma), \quad i = 1, ..., n_1$$
$$y_{2,i} \sim \text{normal}(\mu_2, \sigma), \quad i = 1, ..., n_2$$

Notice that here, we assume that both populations have the same standard deviation.

We could also model 2 separate standard deviations.

Research question: what is the mean difference $\delta = \mu_2 - \mu_1$ of both populations?

Setup

```
rm(list=ls())
library(rstan)
library(coda)
library(BayesianTools)
library(brms)

setwd("~/Nextcloud/teaching Bayes 2021")

rstan_options(auto_write = TRUE)
options(mc.cores = 4) # number of CPU cores
```

Generate data

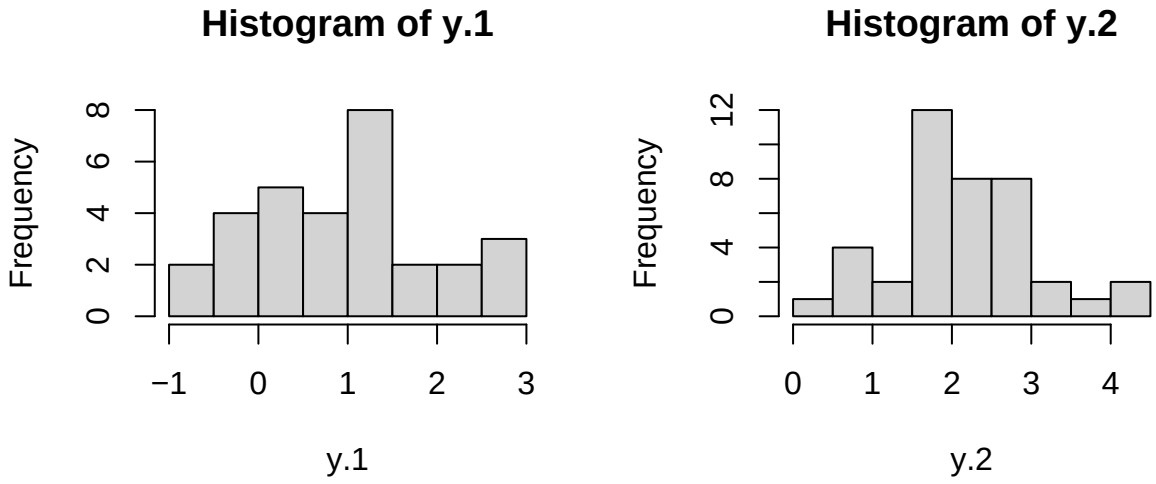
```
set.seed(123) # initiate random number generator for reproducibility

n.1 = 30
mu.1 = 1
sigma.1 = 1

n.2 = 40
mu.2 = 2
sigma.2 = 1
```

```
y.1 = rnorm(n=n.1, mean=mu.1, sd=sigma.1)
y.2 = rnorm(n=n.2, mean=mu.2, sd=sigma.2)

par(mfrow=c(1,2))
hist(y.1)
hist(y.2)
```



Stan code

We translate the statistical model above straightforward into a Stan model.

Diffence in means can be investigated after fitting. Alternatively, we can compute it in the model in the `generated quantities` block

```
data = list(n1=n.1,
            n2=n.2,
            y1=y.1,
            y2=y.2)

stan_code = '
data {
  int n1;
  vector[n1] y1;
  int n2;
  vector[n2] y2;
}
parameters {
  real mu1;
  real mu2;
  real<lower=0> sigma;
}
model {
  // priors
  mu1 ~ normal(0, 10);
  mu2 ~ normal(0, 10);
  sigma ~ normal(0, 10);
  // likelihood
  y1 ~ normal(mu1, sigma);
```

```
  y2 ~ normal(mu2, sigma);
}
generated quantities{
  real delta;
  delta = mu2-mu1;
}
'
```

```
stan_model = stan_model(model_code=stan_code)
fit.1  = sampling(stan_model, data=data)
```

```
print(fit.1, digits=3, probs=c(0.025, 0.975))
```

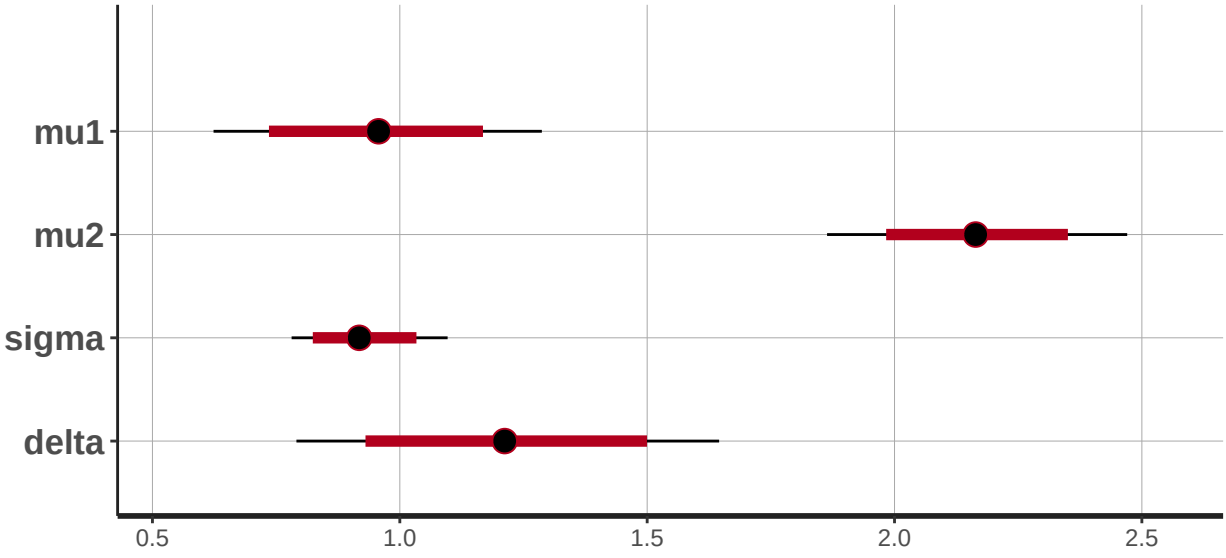
```
## Inference for Stan model: 42ab72f5377cbb91167f2b62d07c706f.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean    sd    2.5%   97.5% n_eff  Rhat
## mu1     0.955   0.003 0.171   0.624   1.287  3985 1.000
## mu2     2.166   0.003 0.150   1.864   2.470  3427 0.999
## sigma   0.924   0.001 0.081   0.781   1.097  3835 1.001
## delta   1.212   0.004 0.221   0.792   1.645  3737 1.000
## lp__  -28.919   0.028 1.274 -32.201 -27.474  2035 1.000
##
## Samples were drawn using NUTS(diag_e) at Thu Oct  7 15:06:29 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Now we can check if `delta>0`

```
plot(fit.1)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
posterior = as.matrix(fit.1)
head(posterior)
```

```
##           parameters
## iterations       mu1      mu2     sigma     delta      lp__
##       [1,] 0.9582488 2.178040 0.9952437 1.2197907 -28.00208
##       [2,] 0.9280431 2.160673 0.8245007 1.2326300 -27.95518
##       [3,] 0.8434813 1.891029 0.8071171 1.0475474 -30.83149
##       [4,] 0.9700242 1.879927 0.8988530 0.9099028 -29.36631
##       [5,] 1.1509269 2.440589 0.8962925 1.2896625 -30.00112
##       [6,] 1.1044751 2.328075 0.8563978 1.2235997 -28.75155
```

```
sum(posterior[, "delta"]>0)/nrow(posterior)
```

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

Classical data format with categorical predictor

Here we code the data differently, just one dataset with a factorial predictor `group`.

```
df = data.frame(y=c(y.1,y.2), group=c(rep(1,n.1),rep(2,n.2)))

head(df)
```

```
##           y group
## 1 0.4395244     1
## 2 0.7698225     1
## 3 2.5587083     1
## 4 1.0705084     1
## 5 1.1292877     1
## 6 2.7150650     1
```

```
data = list(y=df$y,
            group2=df$group2,
            n=nrow(df))
```

Note that the factorial group is an integer (1 or 2), so we can use it as an index in the Stan model.

The statistical model reads:

$$y_i \sim \text{ normal}(\mu_{group_i}, \sigma), \quad i = 1, \ldots, n$$

μ now is a vector of length 2.

```
stan_code = '
data {
  int n;
  real y[n];
  int group[n];
}
parameters {
  real mu[2];
  real<lower=0> sigma;
}
model {
  // priors
  mu ~ normal(0, 10);
  sigma ~ normal(0, 10);
  // likelihood
```

```
  for(i in 1:n){
    y[i] ~ normal(mu[group[i]], sigma);
  }
}
generated quantities{
  real delta;
  delta = mu[2]-mu[1];
}
'
```

```
stan_model = stan_model(model_code=stan_code)
fit.2  = sampling(stan_model, data=data)
```

```
print(fit.2, digits=3, probs=c(0.025, 0.975))
```

```
## Inference for Stan model: 7b52a24092cd58929e9e9a5af34ecf02.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean    sd    2.5%   97.5% n_eff  Rhat
## mu[1]   0.951   0.003 0.172   0.614   1.305  4174 1.000
## mu[2]   2.162   0.002 0.150   1.871   2.464  4191 0.999
## sigma   0.927   0.001 0.082   0.786   1.107  3219 1.000
## delta   1.211   0.003 0.231   0.747   1.666  4475 1.000
## lp__  -28.932   0.030 1.291 -32.196 -27.473  1833 1.001
##
## Samples were drawn using NUTS(diag_e) at Thu Oct  7 15:00:51 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Classical data format and dummy coding

Again, we code the data a little differently. We define an integer variable which indicates if the observation belongs to `group2` (=1) or not (=0).

This is known as **dummy coding** and the statistical model reads

$$y_i \sim \text{normal}(\mu + \delta \cdot group2_i, \sigma)$$

Here, μ is a single value and decribes the mean of population 1: $\mu + 0 \cdot \delta$ (`group2=0`),

δ is the "effect" of population 2 compared to population 1, the mean of population 2 is: $\mu + 1 \cdot \delta$ (`group2=1`)

If the predictor "group" has more than 2 levels (n), an additional dummy variable per level ("intercept" + dummy variables $groupj$, $j = 2, \ldots, n$) has to be coded. dummy variable $groupj = 1$ or 0 indicates if observation belongs to a group j or not, and its respective "effect" is the difference to the first group/intercept. This is tedious and only used for demonstration here, to show how `lm` and `brms` work internally.

```
df$group2=c(rep(0,n.1),rep(1,n.2)) # now group2=0, group2=1

head(df)
```

```
##           y group group2
## 1 0.4395244     1      0
## 2 0.7698225     1      0
```

```
## 3 2.5587083     1      0
## 4 1.0705084     1      0
## 5 1.1292877     1      0
## 6 2.7150650     1      0
```

```
data = list(y=df$y,
            group=df$group,
            n=nrow(df))

stan_code = '
data {
  int n;
  real y[n];
  int group2[n];
}
parameters {
  real mu;
  real delta;
  real<lower=0> sigma;
}
model {
  // priors
  mu ~ normal(0, 10);
  delta ~ normal(0, 10);
  sigma ~ normal(0, 10);
  // likelihood
  for(i in 1:n){
    y[i] ~ normal(mu + group2[i]*delta, sigma);
  }
}
'
```

```
stan_model = stan_model(model_code=stan_code)
fit.3  = sampling(stan_model, data=data)
```

```
print(fit.3, digits=3, probs=c(0.025, 0.975))
```

```
## Inference for Stan model: d3346621cd520d8d32f0d24c717a30ff.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean    sd    2.5%   97.5% n_eff  Rhat
## mu      0.952   0.004 0.169   0.617   1.271  2143 1.002
## delta   1.216   0.005 0.222   0.779   1.652  2159 1.002
## sigma   0.924   0.002 0.083   0.779   1.100  2279 1.000
## lp__  -28.919   0.031 1.272 -32.204 -27.461  1732 1.001
##
## Samples were drawn using NUTS(diag_e) at Mon Oct 11 09:56:57 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Frequentist solutions

```
t.test(y.1, y.2)
```

```
##
##  Welch Two Sample t-test
##
## data:  y.1 and y.2
## t = -5.4121, df = 57.308, p-value = 1.275e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.6598746 -0.7633823
## sample estimates:
## mean of x mean of y
## 0.9528962 2.1645247
```

```
df$group = as.factor(df$group)
summary(lm(y~group, data=df))
```

```
##
## Call:
## lm(formula = y ~ group, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9195 -0.5618 -0.0761  0.5397  2.0044
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.9529     0.1658   5.749 2.32e-07 ***
## group2        1.2116     0.2193   5.526 5.61e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9079 on 68 degrees of freedom
## Multiple R-squared:  0.3099, Adjusted R-squared:  0.2997
## F-statistic: 30.53 on 1 and 68 DF,  p-value: 5.615e-07
```

Common statistical tests are linear models!

https://lindeloev.github.io/tests-as-linear/

brms fit

The brms model uses the classical `lm()` version, which is dummy coding and equivalent to our third Stan model, where `Intercept` is the mean of group 1, and `group2`'s effect (=`delta`) is the difference in means of group 1 and group 2.

```
priors = c(prior(normal(0,10), class=b) )
```

```
fit.b1 = brm(y ~ group,
             prior = priors,
             data = df)
```

```
fit.b1
```

```
##  Family: gaussian
```

```
##   Links: mu = identity; sigma = identity
## Formula: y ~ group
##    Data: df (Number of observations: 70)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.95      0.17     0.63     1.28 1.00     4015     2910
## group2        1.21      0.22     0.78     1.65 1.00     4008     2922
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.92      0.08     0.78     1.09 1.00     4415     3244
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(conditional_effects(fit.b1),
     points=TRUE)
```