Measurement error and missing data

Frank Edwards

5/1/2020

What is measurement error?

- When randomly sampling a unit from a larger population, we obtain a single value
- This single value x differs by an unknown amount from the true population value, μ. This is called sampling error.
- This means that our observation is but one of many possible values that could have been observed with a random sample of the population
- Treating the observation as the **truth** is thus unwise

Divorce data again, now with measurement error!

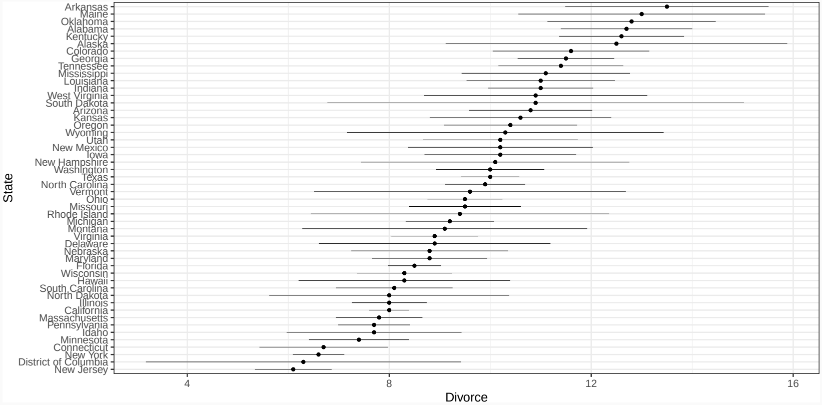
```r
library(rethinking)
data(WaffleDivorce)
d <- WaffleDivorce

head(d %>% select(Location, Divorce, Divorce.SE))
```

```
##      Location Divorce Divorce.SE
## 1     Alabama    12.7       0.79
## 2      Alaska    12.5       2.05
## 3     Arizona    10.8       0.74
## 4    Arkansas    13.5       1.22
## 5 California     8.0       0.24
## 6    Colorado    11.6       0.94
```

Visualizing the uncertainty

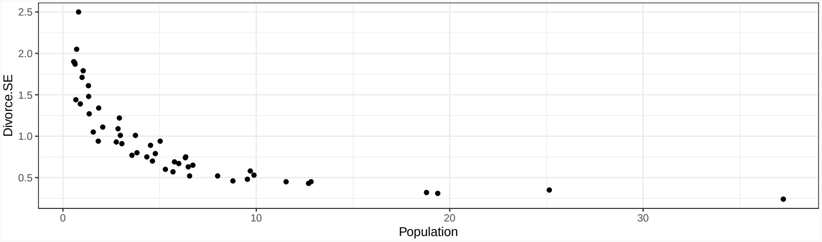
```
ggplot(d, aes(x = reorder(Location, Divorce), y = Divorce,
       ymin = Divorce - Divorce.SE * 1.65, ymax = Divorce + Divorce.SE * 1.65)) +
  coord_flip() +
  geom_pointrange(size = 0.1) +
  labs(x = "State")
```



Error is a function of population size

The American Community Survey (ACS) samples a fixed percent of the population. Places with smaller populations have fewer sampled households and larger error

```r
ggplot(d, aes(x = Population, y = Divorce.SE)) +
  geom_point()
```



What this means in practice

```r
NJ<-d %>% filter(Loc=="NJ") %>% select(Loc, Divorce, Divorce.SE)
NJ
```

```
##   Loc Divorce Divorce.SE
## 1  NJ     6.1       0.46
```

Let’s assume that NJ’s true Divorce rate is in fact 6.1 (we got very luck in sampling!). Here’s ten possible values we *could* have obtained instead of 6.1

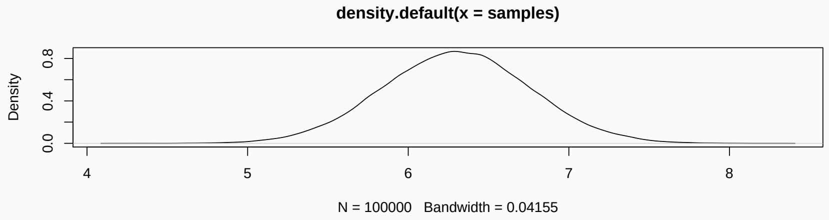
```r
rnorm(10, 6.1, 0.46)
```

```
##  [1] 5.811831 6.184476 5.715611 6.833829 6.251574 5.722585 6.324217 6.439629
##  [9] 6.364859 5.959521
```

What this means in practice

Let's say that the true divorce rate in NJ is 6.3, this is the density of possible observations with the one percent ACS sample.

```r
samples<-rnorm(1e5, 6.3, 0.46)
plot(density(samples))
```



Treating the truth as a parameter

For each observed value of a state's divorce rate based on a sample, D_{obs_i}, there is a true state divorce rate D_{true_i}. The observation is one draw from the true sampling distribution

$$D_{\mathrm{obs}_i} \sim \mathrm{Normal}(D_{\mathrm{true}_i}, D_{\mathrm{SE}_i})$$

We don't observe D_{true}, but can estimate a posterior for it from our model, and incorporate uncertainty in measurement across the model.

Defining a model

- Note that we'll now be estimating uncertainty in divorce rates as we estimate our model of divorce rates!

$$D_{\mathrm{obs}_i} \sim \mathrm{Normal}(D_{\mathrm{true}_i}, D_{\mathrm{SE}_i})$$

$$D_{\mathrm{true}_i} \sim \mathrm{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta_A A_i + \beta_M M_i$$

$$\alpha \sim \mathrm{Normal}(0, 1)$$

$$\beta_A \sim \mathrm{Normal}(0, 1)$$

$$\beta_M \sim \mathrm{Normal}(0, 1)$$

$$\sigma \sim \mathrm{Exponential}(1)$$

Estimating the model

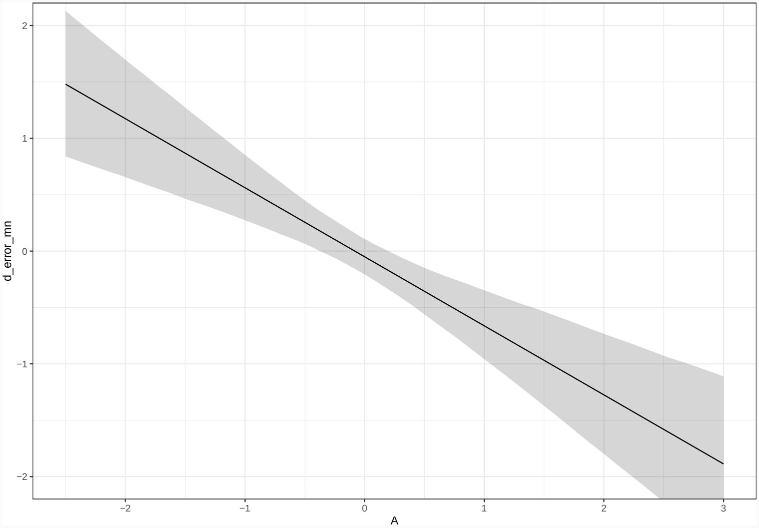
```r
d_slim<-list(
  D_obs = scale(d$Divorce), A = scale(d$MedianAgeMarriage),
  M = scale(d$Marriage), D_se = d$Divorce.SE / sd(d$Divorce),
  N = nrow(d))

m_error<-ulam(alist(
  D_obs ~ dnorm(D_true, D_se),
  vector[N]:D_true ~ dnorm(mu, sigma),
  mu<-a + bA * A + bM * M,
  a ~ dnorm(0, 0.2),
  bA ~ dnorm(0, 0.5),
  bM ~ dnorm(0, 0.5),
  sigma ~ dexp(1)
), data=d_slim, cores = 4, chains = 4)
```

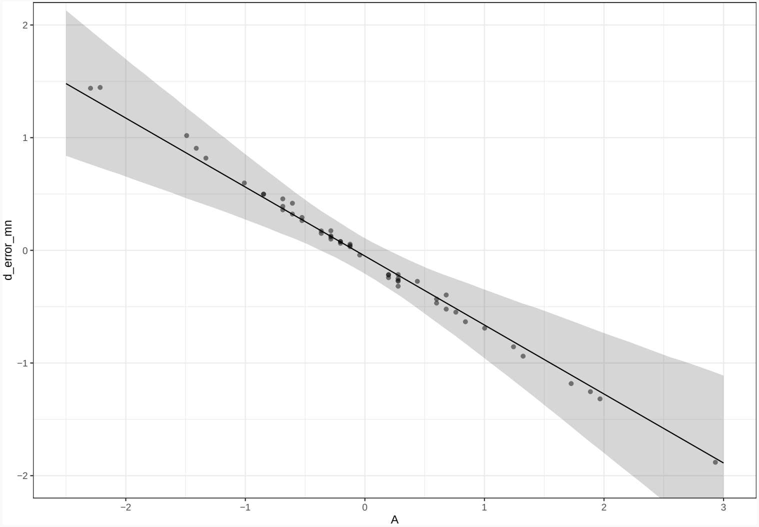
Estimate a model with no measurement error adjustment for comparison

```
m_no_error<-ulam(alist(
  D_obs ~ dnorm(mu, sigma),
  mu<-a + bA * A + bM * M,
  a ~ dnorm(0, 0.2),
  bA ~ dnorm(0, 0.5),
  bM ~ dnorm(0, 0.5),
  sigma ~ dexp(1)
), data=d_slim, cores = 4, chains = 4)
```

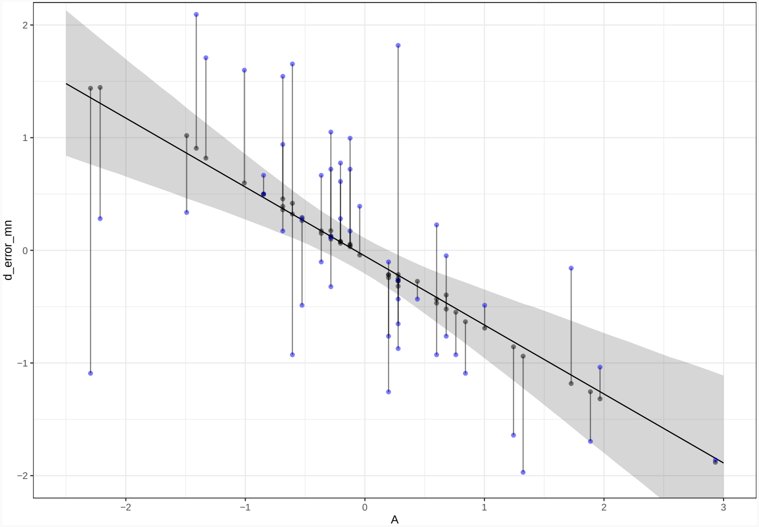
Visualize the results



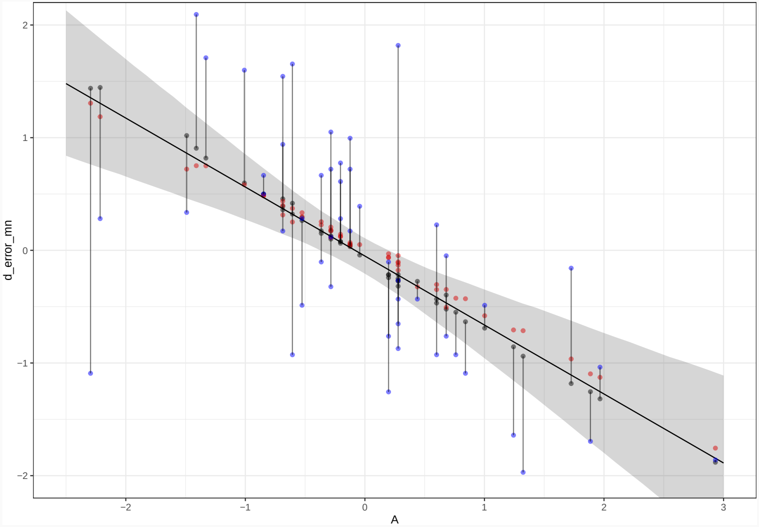
Visualize the results: error model posterior mu



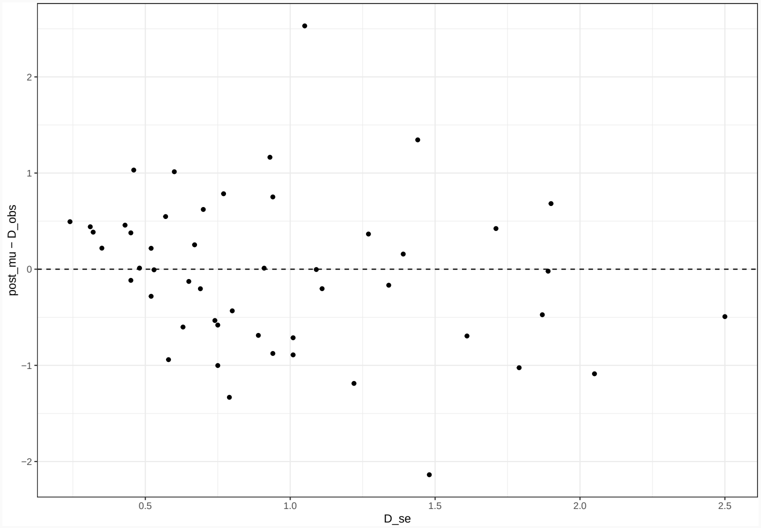
Visualize the results: observed data (blue)



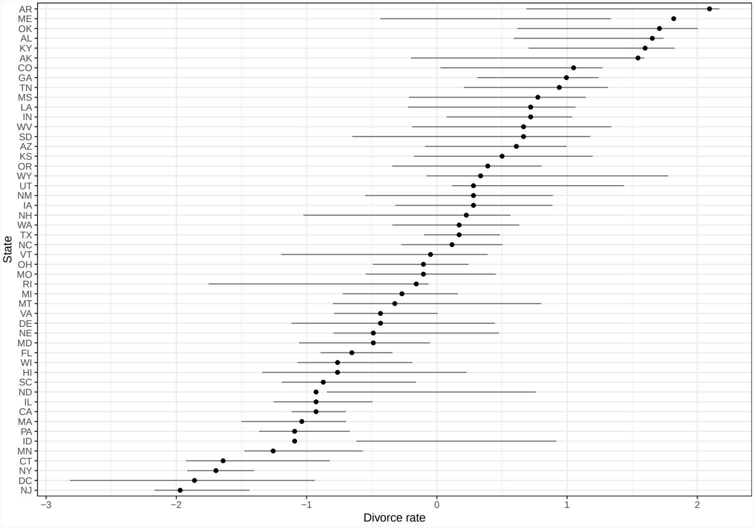
Visualize the results: model without error adjustment (red)



Visualize the results: magnitude of shrinkage



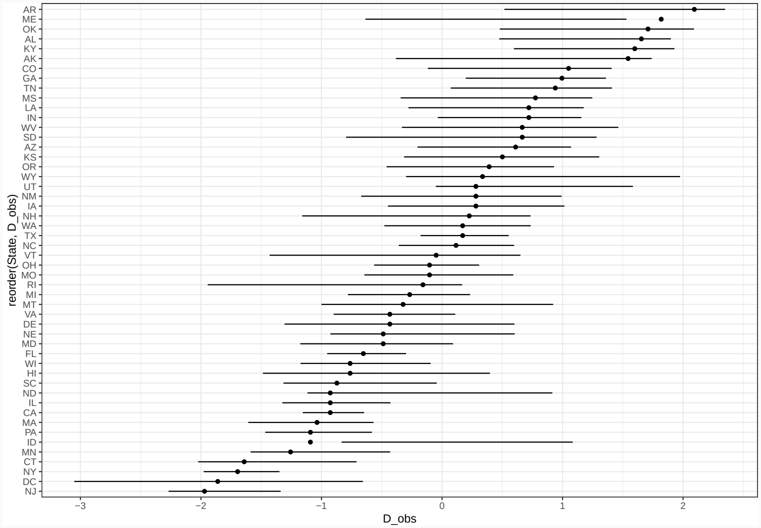
The posterior for each observation of D



Fitting this model with brms

```r
m_error_brm<-brm(D_obs|mi(D_se) ~ A + M,
                 prior = c(
                   prior(normal(0, 0.2), class = Intercept),
                   prior(normal(0, 0.5), class = b),
                   prior(exponential(1), class = sigma)),
                 data = d_slim, save_mevars = T)
```

Visualzing the brms inferences



Measurement error on both outcomes and predictors

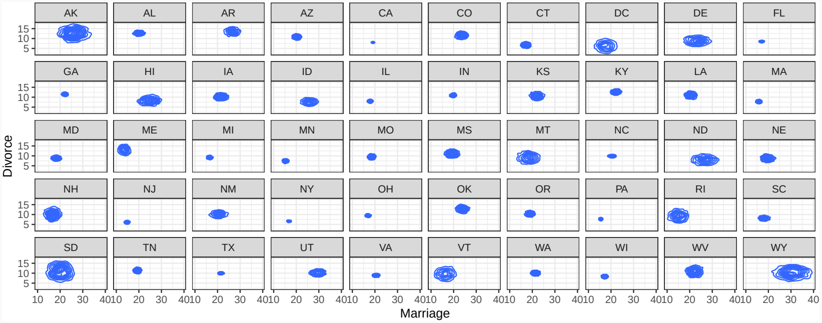
Let’s look at that data again

```r
head(d %>%
       select(Loc, Marriage, Marriage.SE,
              Divorce, Divorce.SE))
```

```
##   Loc Marriage Marriage.SE Divorce Divorce.SE
## 1  AL     20.2        1.27    12.7       0.79
## 2  AK     26.0        2.93    12.5       2.05
## 3  AZ     20.3        0.98    10.8       0.74
## 4  AR     26.4        1.70    13.5       1.22
## 5  CA     19.1        0.39     8.0       0.24
## 6  CO     23.5        1.24    11.6       0.94
```

What this means in practice

If we assume that the observed marriage and divorce values are the true value, the sampling distributions look like this



Incorporating two kinds of measurement error into our model

$$D_{\mathrm{obs}_i} \sim \mathrm{Normal}(D_{\mathrm{true}_i}, D_{\mathrm{SE}_i})$$

$$D_{\mathrm{true}_i} \sim \mathrm{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta_A A_i + \beta_M M_{\mathrm{true}_i}$$

$$M_{\mathrm{obs}_i} \sim \mathrm{Normal}(M_{\mathrm{true}_i}, M_{SE_i})$$

$$M_{\mathrm{true}_i} \sim \mathrm{Normal}(0, 1)$$

$$\alpha \sim \mathrm{Normal}(0, 1)$$

$$\beta_A \sim \mathrm{Normal}(0, 1)$$

$$\beta_M \sim \mathrm{Normal}(0, 1)$$

$$\sigma \sim \mathrm{Exponential}(1)$$

Estimating the model with ulam()

```r
d_slim$M_se<-d$Marriage.SE/sd(d$Marriage)
m_error_both<-ulam(alist(
  D_obs ~ dnorm(D_true, D_se),
  vector[N]:D_true ~ dnorm(mu, sigma),
  mu<-a + bA * A + bM * M,
  M ~ dnorm(M_true, M_se),
  vector[N]:M_true ~ dnorm(0,1),
  a ~ dnorm(0, 0.2),
  bA ~ dnorm(0, 0.5),
  bM ~ dnorm(0, 0.5),
  sigma ~ dexp(1)),
  data=d_slim, cores = 4, chains = 4)
```

Estimating the model with brm()

```r
m_error_brm_both<-brm(D_obs|mi(D_se) ~ A +
                        me(M, M_se),
                 prior = c(
                   prior(normal(0, 0.2), class = Intercept),
                   prior(normal(0, 0.5), class = b),
                   prior(exponential(1), class = sigma)),
                 data = d_slim, save_mevars = T)
```

Measurement error

- Data are often measured with error
- Sometimes we know this error - that’s great!
- Incorporate the measurement error into your model
- Bayesian models are generative: we can estimate a posterior for variables measured with error at the same time as we estimate other parameters
- Correlated errors across multiple measures can induce bias (see DAGs on page 498)

Missing data

Missing data

- Somtimes variables are missing a value

Missing data

- Somtimes variables are missing a value
- R calls these `NA`

Missing data

- Somtimes variables are missing a value
- R calls these `NA`
- There can be many reasons that a value isn't reported

Missing data

- Somtimes variables are missing a value
- R calls these `NA`
- There can be many reasons that a value isn’t reported
- Think hard about why it may have happened

Missing data

- Somtimes variables are missing a value
- R calls these `NA`
- There can be many reasons that a value isn’t reported
- Think hard about why it may have happened
- Software typically defaults to *listwise deletion*, or *complete case analysis*

Missing data

- Somtimes variables are missing a value
- R calls these `NA`
- There can be many reasons that a value isn't reported
- Think hard about why it may have happened
- Software typically defaults to *listwise deletion*, or *complete case analysis*
- At best, this discards perfectly good information in the other variables in that row. At worst, it leads to biased inference.

Imputation

- Avoid discarding data whenever possible

Imputation

- Avoid discarding data whenever possible
- Use common sense to replace missings when possible (e.g. in panel data when we have a respondent's age in one wave but not another)

Imputation

- Avoid discarding data whenever possible
- Use common sense to replace missings when possible (e.g. in panel data when we have a respondent’s age in one wave but not another)
- Avoid imputing data with a single value when we are uncertain (never replace a missing with a mean or interpolation). This overstates your certainty, and artificially narrows posterior intervals

Imputation

- Avoid discarding data whenever possible
- Use common sense to replace missings when possible (e.g. in panel data when we have a respondent's age in one wave but not another)
- Avoid imputing data with a single value when we are uncertain (never replace a missing with a mean or interpolation). This overstates your certainty, and artificially narrows posterior intervals
- Use imputation models to populate your uncertainty into the analysis.

Missing data typology

The ludicrous names that get used in the literature to describe various mechanisms for missing data

- Missing completely at random: each observation has equal probability of being missing.
- Missing at random: each observation's probability of being missing is conditional on some set of measured variables
- Missing not at random: each observation's probaility of being missing is conditional on some set of unmeasured variables

This dog ate my homework

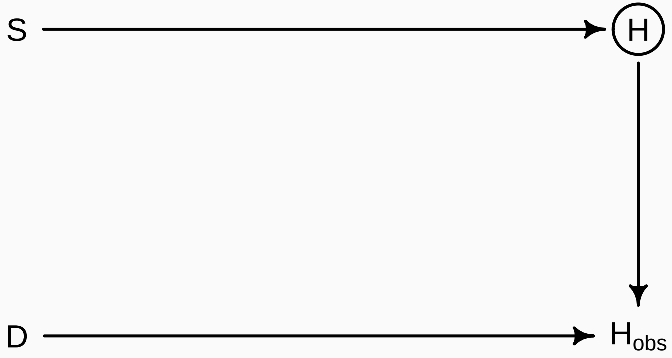
He’s still a good boy

Why do dogs make homework go missing?

Let's asssume that H is the grade a homework would have received if graded, S is how much a student studied, and D is whether a dog ate the homework. We observe H_obs, a vector of grades where some are missing.

Dogs eat homework completely at random

Nothing affects a dog's decision to eat homework, they just strike at random!



$$\begin{tikzcd}
S \arrow[r] & \boxed{H} \arrow[d] \\
D \arrow[r] & H_{obs}
\end{tikzcd}$$

Simulate random homework eating

Dogs eat homework 20 percent of the time, and strike at random

```
N<-100
S<-rnorm(N)
H<-rbinom(N, size = 10, prob = inv_logit(S))
D<-rbinom(N, 1, p = 0.2)
H_obs<-H
H_obs[D==1]<-NA
summary(H_obs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   3.000   4.000   4.692   6.000  10.000      22
```

How does the missingness affect inference? List-wise deletion

```r
library(broom)
d<-data.frame(H_obs = H_obs, S = S) %>% filter(!is.na(H_obs))
mcar_0<-brm(H_obs ~ S, data = d)
```

```r
tidy(mcar_0)
```

```
##          term    estimate std.error       lower       upper
## 1 b_Intercept    5.010457 0.1728653    4.724120    5.295409
## 2         b_S    2.367944 0.2097332    2.020992    2.718633
## 3       sigma    1.480677 0.1239770    1.290049    1.696584
## 4        lp__ -145.912882 1.2764912 -148.397932 -144.559153
```

How does the missingness affect inference? Complete data

```
d<-data.frame(H = H, S = S)
mcar_1<-brm(H ~ S, data = d)
```

```
tidy(mcar_1)
```

```
##          term    estimate std.error      lower      upper
## 1 b_Intercept    5.056265 0.1428282   4.817824   5.287313
## 2         b_S    2.308409 0.1651645   2.045640   2.589497
## 3       sigma    1.411133 0.1045346   1.251116   1.593560
## 4        lp__ -181.274366 1.2884789 -183.813197 -179.913627
```

Dogs eat homework when students study too much

Why don't you want to go play?

$$\begin{tikzcd}
S \arrow[r] \arrow[d] & \textcircled{H} \arrow[d] \\
D \arrow[r] & H_{obs}
\end{tikzcd}$$

Simulate bad dogs

```r
P<-ifelse(S>0, 0.5, 0)
D<-rbinom(N, 1, prob = P)
H_obs<-H
H_obs[D==1]<-NA
summary(H_obs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   3.000   4.000   4.273   6.000  10.000      23
```

How does the missingness affect inference? List-wise deletion

```r
d<-data.frame(H_obs = H_obs, S = S) %>% filter(!is.na(H_obs))
mar_0<-brm(H_obs ~ S, data = d)
```

```r
tidy(mar_0)
```

```
##          term    estimate std.error      lower       upper
## 1 b_Intercept    5.123431 0.1615340    4.853224    5.389861
## 2         b_S    2.393996 0.1817246    2.094337    2.696579
## 3       sigma    1.319999 0.1105789    1.147163    1.511831
## 4        lp__ -135.513357 1.2176541 -137.888184 -134.165899
```

How does the missingness affect inference? Complete data

```
d<-data.frame(H = H, S = S)
mar_1<-brm(H ~ S, data = d)
```

```
tidy(mar_1)
```

```
##          term    estimate std.error       lower       upper
## 1 b_Intercept    5.057506 0.1429910    4.825733    5.290436
## 2         b_S    2.309885 0.1660931    2.043301    2.585037
## 3       sigma    1.412079 0.1015118    1.256419    1.584946
## 4        lp__ -181.251686 1.2327416 -183.704430 -179.905489
```

Dogs hate noise, and so does homework

Noisy homes *X* make bad homework and dogs

$$\begin{tikzcd}
S \arrow[rr] \arrow[dd] & & \boxed{H} \arrow[dd] \\
& \boxed{X} \arrow[ur] \arrow[dl] & \\
D \arrow[rr] & & H_{obs}
\end{tikzcd}$$

Simulate noisy houses

```r
N<-100
S<-rnorm(N)
X<-rnorm(N)
H<-rbinom(N, size = 10, prob = inv_logit(2 + S - 2*X))
D<-ifelse(X>1, 1, 0)
H_obs<-H
H_obs[D==1]<-NA
summary(H_obs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   2.000   8.000   9.000   8.247  10.000  10.000      19
```

How does the missingness affect inference? List-wise deletion

```r
d<-data.frame(H_obs = H_obs, S = S, X = X) %>% filter(!is.na(H_obs))
mnar_0<-brm(H_obs ~ S, data = d)
```

```r
tidy(mnar_0)
```

```
##          term    estimate std.error       lower      upper
## 1 b_Intercept    8.377476 0.2212538   8.0167669   8.737188
## 2         b_S    1.153430 0.2229636   0.7812635   1.530252
## 3       sigma    1.957408 0.1588300   1.7158168   2.239891
## 4        lp__ -173.878775 1.2765405 -176.3685728 -172.527316
```

How does the missingness affect inference? Complete data

```
d<-data.frame(H = H, S = S)
mnar_1<-brm(H ~ S, data = d)
```

```
tidy(mnar_1)
```

```
##           term    estimate std.error       lower       upper
## 1 b_Intercept    7.360858 0.2998019    6.877486    7.858758
## 2         b_S    1.506460 0.3009301    1.020529    2.007309
## 3       sigma    3.011228 0.2237737    2.663287    3.404937
## 4        lp__ -256.092820 1.2324443 -258.498001 -254.743549
```

How does the missingness affect inference? Complete data with unobserved cause

```
d<-data.frame(H = H, S = S, X = X)
mnar_2<-brm(H ~ S + X, data = d)
```

```
tidy(mnar_2)
```

```
##              term    estimate std.error        lower      upper
## 1 b_Intercept    7.457159 0.1740462    7.1747142   7.743018
## 2             b_S    1.117298 0.1772999    0.8296919   1.404930
## 3             b_X   -2.304900 0.1615164   -2.5704538  -2.039863
## 4           sigma    1.695863 0.1210468    1.5112663   1.911102
## 5            lp__ -199.693354 1.3923707 -202.4171385 -198.019594
```

Very good dogs

Dogs only eat bad homework

$$\begin{tikzcd}
S \arrow[r] & \textcircled{H} \arrow[dl] \arrow[d] \\
D \arrow[r] & H_{obs}
\end{tikzcd}$$

Simulate good dogs

```
N<-100
S<-rnorm(N)
H<-rbinom(N, size = 10, prob = inv_logit(S))
D<-ifelse(H<5, 1, 0)
H_obs<-H
H_obs[D==1]<-NA
summary(H_obs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   5.000   5.000   6.000   6.738   8.000  10.000      39
```

How does the missingness affect inference? List-wise deletion

```
d<-data.frame(H_obs = H_obs, S = S) %>% filter(!is.na(H_obs))
mnar_0<-brm(H_obs ~ S, data = d)
```

```
tidy(mnar_0)
```

```
##          term   estimate std.error       lower      upper
## 1 b_Intercept   5.754663 0.2103699    5.4076133   6.105897
## 2         b_S   1.515797 0.2304014    1.1362693   1.899124
## 3       sigma   1.145530 0.1057819    0.9897332   1.332844
## 4        lp__ -99.865863 1.1903074 -102.2037173 -98.517445
```

How does the missingness affect inference? Complete data

```
d<-data.frame(H = H, S = S)
mnar_1<-brm(H ~ S, data = d)
```

```
tidy(mnar_1)
```

```
##            term    estimate std.error       lower       upper
## 1 b_Intercept    4.903829 0.1389065    4.674313    5.128918
## 2         b_S    2.278978 0.1520660    2.023544    2.531579
## 3       sigma    1.405604 0.1040351    1.245806    1.591546
## 4        lp__ -180.754545 1.2672585 -183.300258 -179.401340
```

Bayesian imputation

- Much like with measurement error, we can treat missing values as parameters to be estimated from our model.
- HMC samples continuous missing measures well
- But it can’t handle categorical measures as easily (see 15.3 for a method)
- We’re going to use brm(), but see 15.2.2 for examples using ulam()

Bayesian imputation with brm()

```r
library(mice)
data(nhanes)
summary(nhanes)
```

```
##       age            bmi             hyp             chl       
##  Min.   :1.00   Min.   :20.40   Min.   :1.000   Min.   :113.0  
##  1st Qu.:1.00   1st Qu.:22.65   1st Qu.:1.000   1st Qu.:185.0  
##  Median :2.00   Median :26.75   Median :1.000   Median :187.0  
##  Mean   :1.76   Mean   :26.56   Mean   :1.235   Mean   :191.4  
##  3rd Qu.:2.00   3rd Qu.:28.93   3rd Qu.:1.000   3rd Qu.:212.0  
##  Max.   :3.00   Max.   :35.30   Max.   :2.000   Max.   :284.0  
##                 NA's   :9       NA's   :8       NA's   :10     
```

Let’s build a model to predict BMI as a function of age and cholesterol

```r
m0<-brm(bmi ~ age + chl,
        data = nhanes, cores = 4)
```

```r
tidy(m0)
```

```
##          term    estimate  std.error       lower      upper
## 1 b_Intercept  19.41425783 4.08673940  12.77672986  26.0945251
## 2       b_age  -5.34252093 1.68150558  -8.01861983  -2.6468062
## 3       b_chl   0.08389625 0.02576278   0.04317525   0.1262215
## 4       sigma   3.45781357 0.89396538   2.35098064   5.0549389
## 5        lp__ -38.53672426 1.63118244 -41.71180431 -36.6707266
```

This model dropped the missing values in bmi and chl

- We can impute values by sampling them with HMC, giving them a likelihood and priors
- We'll obtain posteriors for each missing value
- We specify these likelihoods and priors explicitly

```r
m1 <- brm(bmi|mi() ~ age + chl,
          data = nhanes)
```

But we’ve got missingness on chl too

We need to specify a model with more than one likelihood, as we did in the measurement error model. We set (missing and non-missing) values in bmi to be a function of age and cholesterol, and missing cholesterol values to be a function of age.

```r
formula_2 <-
  bf(bmi | mi() ~ age + mi(chl)) + # BMI model
  bf(chl | mi() ~ age) # chl model
m2 <- brm(formula_2,
          data = nhanes)
```

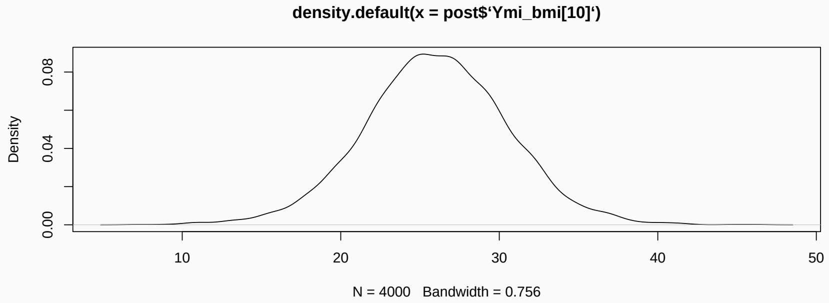
The output

```r
summary(m2)
```

```
##  Family: MV(gaussian, gaussian) 
##   Links: mu = identity; sigma = identity
##          mu = identity; sigma = identity 
## Formula: bmi | mi() ~ age + mi(chl) 
##          chl | mi() ~ age 
##    Data: nhanes (Number of observations: 25) 
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
## 
## Population-Level Effects: 
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## bmi_Intercept    21.94      3.82    14.65    29.84 1.00     1760     2101
## chl_Intercept   138.72     24.74    89.72   186.90 1.00     2654     2774
## bmi_age          -4.29      1.48    -7.17    -1.34 1.00     1274     2240
## chl_age          30.84     12.90     5.73    56.47 1.00     2707     2650
## bmi_michl         0.06      0.02     0.01     0.11 1.00     1070     1526
## 
## Family Specific Parameters: 
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma_bmi     3.31      0.77     2.15     5.08 1.00     1558     1909
## sigma_chl    40.18      7.65    28.27    57.83 1.00     2095     2085
## 
## Samples were drawn 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).
```

Visualizing the imputation

```
post<-posterior_samples(m2)
plot(density(post$‘Ymi_bmi[10]‘))
```



Bayesian imputation: pros/cons

Pros - We can allow for very complex structures in our models - We can neatly specify variable specific models for missing data - Sampling missing values using HMC as part of the model is consistent with Bayesian modeling principals

Cons - Gets technically complex quickly - Doesn't have easy solution for categorical data - Computationally intensive for big data

Multiple imputation by chained equations (MICE)

MICE is a common pseudo-Bayesian approach to missing data. It produces k predictions for each unobserved variable based on a fully conditional regression models (where each variable is a function of all others).

- We'll allow each variable in the model to be a function of all others (we can relax this)
- Then make conditional predictions for each missing value
- This results in k complete datasets
- We apply our analysis over each dataset, and pool results

For a detailed description of the method and software, See Van Buuren and Groothuis-Oudshoorn, 2011:
`http://www.jstatsoft.org/v45/i03/paper`

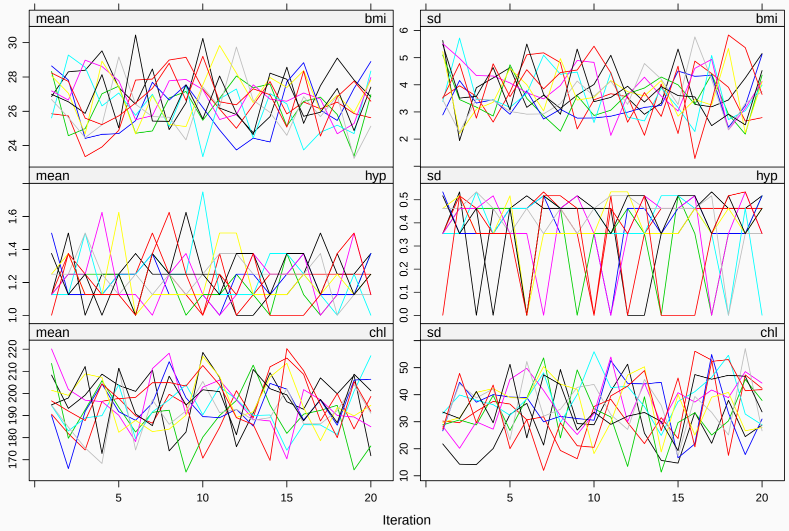
MICE in practices

It's honestly too easy

```
library(mice)
imp_nhanes<-mice(nhanes, m = 10, maxit = 20)
summary(imp_nhanes)
```

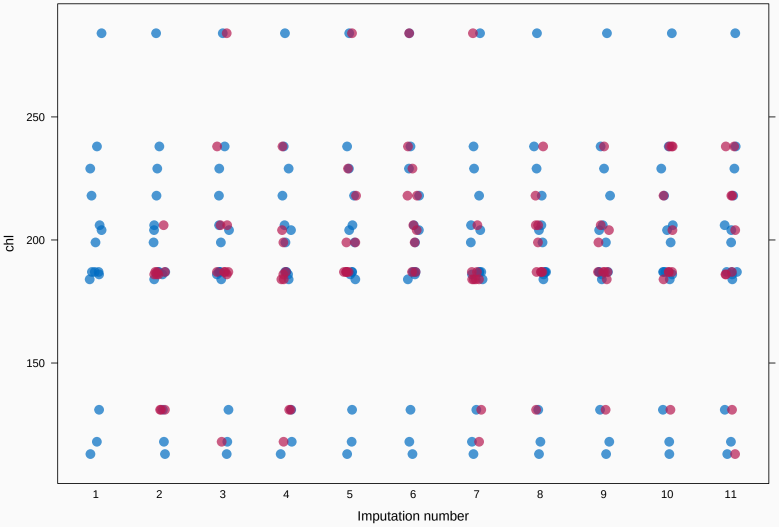
Visualizing convergence

```
plot(imp_nhanes)
```



Visualizing the imputations

```
stripplot(imp_nhanes, chl~.imp, pch=20, cex=2)
```



Changing predictors

If we've got a collinear variable or a factor with many categories, we may want to exclude it from the model. Columns indicate predictors, row indicates outcomes.

```
pred<-imp_nhanes$predictorMatrix
```

Turn off hyp as a predictor of chl

```
pred["chl", "hyp"]<-0
pred
```

```
##     age bmi hyp chl
## age   0   1   1   1
## bmi   1   0   1   1
## hyp   1   1   0   1
## chl   1   1   0   0
```

Use the new predictor matrix

```r
imp_nhanes2<-mice(nhanes, m = 10, maxit = 20,
                  predictorMatrix = pred)
summary(imp_nhanes)
```

Change imputation methods

- The default in mice for continuous measures is partial mean matching
- The algorithm constructs a regression model, samples new coefficients from the parameter distribution, then makes a prediction. It then randomly selects one proximate observation from the observed data
- This makes the imputation data appear similar to the observed
- But sometimes we may not want to to do this, or may have categorical data

Imputation methods

For a full list of methods, see

`https://stefvanbuuren.name/fimd/sec-modelform.html`

```
meth<-imp_nhanes$method
```

Let’s change chl to a linear model

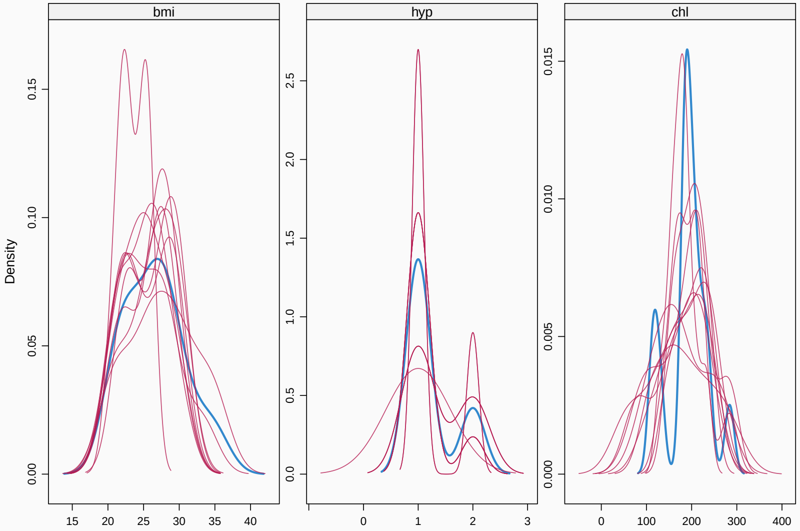
```
meth[4]<-”norm”
```

And re-run the model

```
imp_nhanes3<-mice(nhanes, m = 10, maxit = 20,
                  method = meth)
summary(imp_nhanes3)
```

Visualize imputations

```r
densityplot(imp_nhanes3)
```



Using the imputed data

After imputation, we conduct our analysis over each imputed dataset.

Typically, we'll need to separately fit each model.

For frequentist methods, either use a loop or the with() function

```r
fit <- with(imp_nhanes3, lm(bmi ~ chl))
```

Then pool your results according to Rubin's rules for combination. This averages beta parameters, and adjusts standard errors for cross-imputation variance

The results

```r
pool(fit)
```

```
## Class: mipo    m = 10
##          term  m    estimate         ubar            b            t dfcom
## 1 (Intercept) 10 21.31695134 1.113277e+01 2.129882e+00 1.347564e+01    23
## 2         chl 10  0.02587948 2.925939e-04 5.994597e-05 3.585345e-04    23
##         df       riv    lambda       fmi
## 1 16.56385 0.2104482 0.1738597 0.2583155
## 2 16.26681 0.2253655 0.1839170 0.2686309
```

With brms

In a Bayesian context, we don't have to worry about formulas for combination. We fit the model to each dataset, then pool the posterior samples from each for inference.

brms provides brm_multiple() to do this

```
brm_mi<-brm_multiple(bmi ~ age + chl,
                     data =imp_nhanes3, cores = 4)
```

Output

```r
summary(brm_mi)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: bmi ~ age + chl
##    Data: imp_nhanes3 (Number of observations: 25)
## Samples: 40 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 40000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    20.85      3.18    14.26    26.84 1.18      150      634
## age          -3.61      1.01    -5.59    -1.60 1.10      235     1224
## chl           0.06      0.02     0.02     0.10 1.19      143      685
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     3.08      0.54     2.21     4.33 1.09      261      713
##
## Samples were drawn 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).
```

The posterior samples

```r
post<-posterior_samples(brm_mi)
nrow(post)
```

```
## [1] 40000
```

Summary

- Measurement error is an important part of the data generating process
- Ignoring it presumes we have perfect measurement, can bias inference, and understates uncertainty
- Missing data is also important. List-wise deletion at best discards information (and overstates certainty), at worst biases inference
- If data is missing completely at random, or is conditional on an observed variable, we can impute the missings and recover valid inference
- If missingness is correlated with the outcome, we've got a problem

Further reading

- Further reading on mice:
  `http://www.gerkovink.com/miceVignettes/`
- Further reading on Bayesian imputation and brms:
  `https://cran.r-project.org/web/packages/brms/vignettes/brms_missings.html`

Thank you!

Thanks for a wonderful semester and for being experimental subjects as I try teaching Bayes!

Stay safe, and please be in touch: `frank.edwards@rutgers.edu`