Lecture 15 Handout

Elizabeth Colantuoni

3/15/2021

I. Objectives

Upon completion of this session, you will be able to do the following:

- Describe imputation procedures to account for missing data
- Implement single or multiple imputation approaches using R

II. Analysis of missing data in Nepali Anthropometry Study

In this section, we will walk through both exploratory analysis of missing data patterns and implemention of missing data imputation approaches within the Nepali Anthropometry Study.

A. What are the patterns of missing data?

In this section, we will open the data and explore the patterns with respect to sex, age, alive, maternal age, breast feeding and weight.

```
# load and wrangle data
load("NepalAnthro.rdata")
d = nepal.anthro
d = d[,c("id","fuvisit","sex","age","alive","mage","bf","wt")]
d$sex = d$sex - 1
## Number of missing values for bf:
sum(is.na(d$bf))
```

```
## [1] 53
```

```
## To keep things simple, we will do a mode replacement for bf
bf.modes = tapply(d$bf,d$fuvisit,Mode,na.rm=T)
bf.modes
```

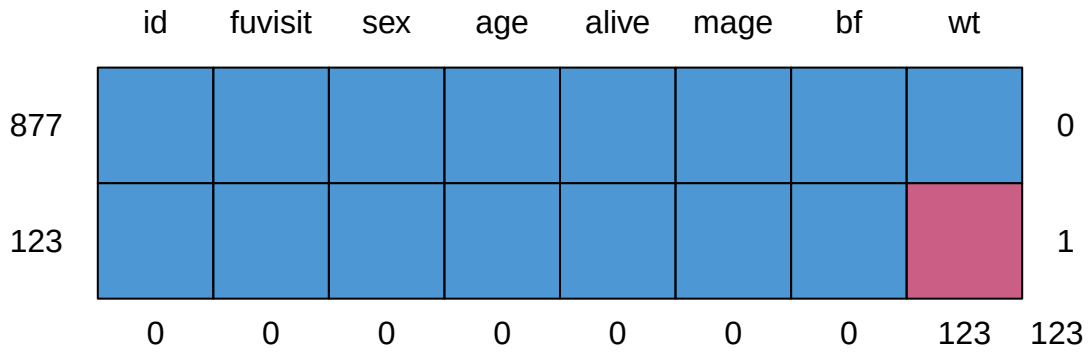
```
## 0 1 2 3 4
## 0 0 0 0 0
```

```
d$bf[is.na(d$bf)] = 0
d$bf = as.factor(d$bf)
## Sort d by follow-up visit and missing wt status
d = d[order(d$fuvisit,!is.na(d$wt)),]
```

Note there are 53 missing values for the indicator for breast feeding. To simplify our missing data analyses, we will replace these values with the mode of the observed breast feeding indicator.

The table below displays the missing data patterns with respect to each variable in the analysis dataset. Note there are 877 rows of data with no missing values and 123 rows of data with missing weight.

```
md.pattern(d)
```

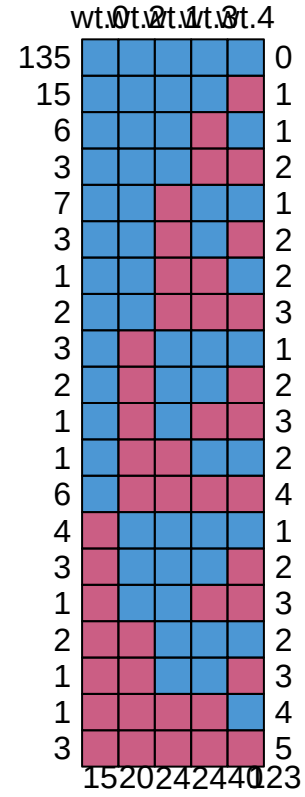



```
##     id fuvisit sex age alive mage bf  wt
## 877  1       1   1   1     1    1  1   1   0
## 123  1       1   1   1     1    1  1   0   1
##      0       0   0   0     0    0  0 123 123
```

So far, we have ignored the longitudinal structure of the data, i.e. weight information may be missing periodically over the course of the follow-up or children may have dropped out of the study.

To explore the missing weight information over time, we will reshape the data to a wide format and then look at the missing data patterns.

```
# Create a wide dataset
d.wide <- reshape(d,v.names=c("wt","age","bf"),idvar="id",timevar="fuvisit",direction="wide")
md.pattern(d.wide[,c("wt.0","wt.1","wt.2","wt.3","wt.4")])
```



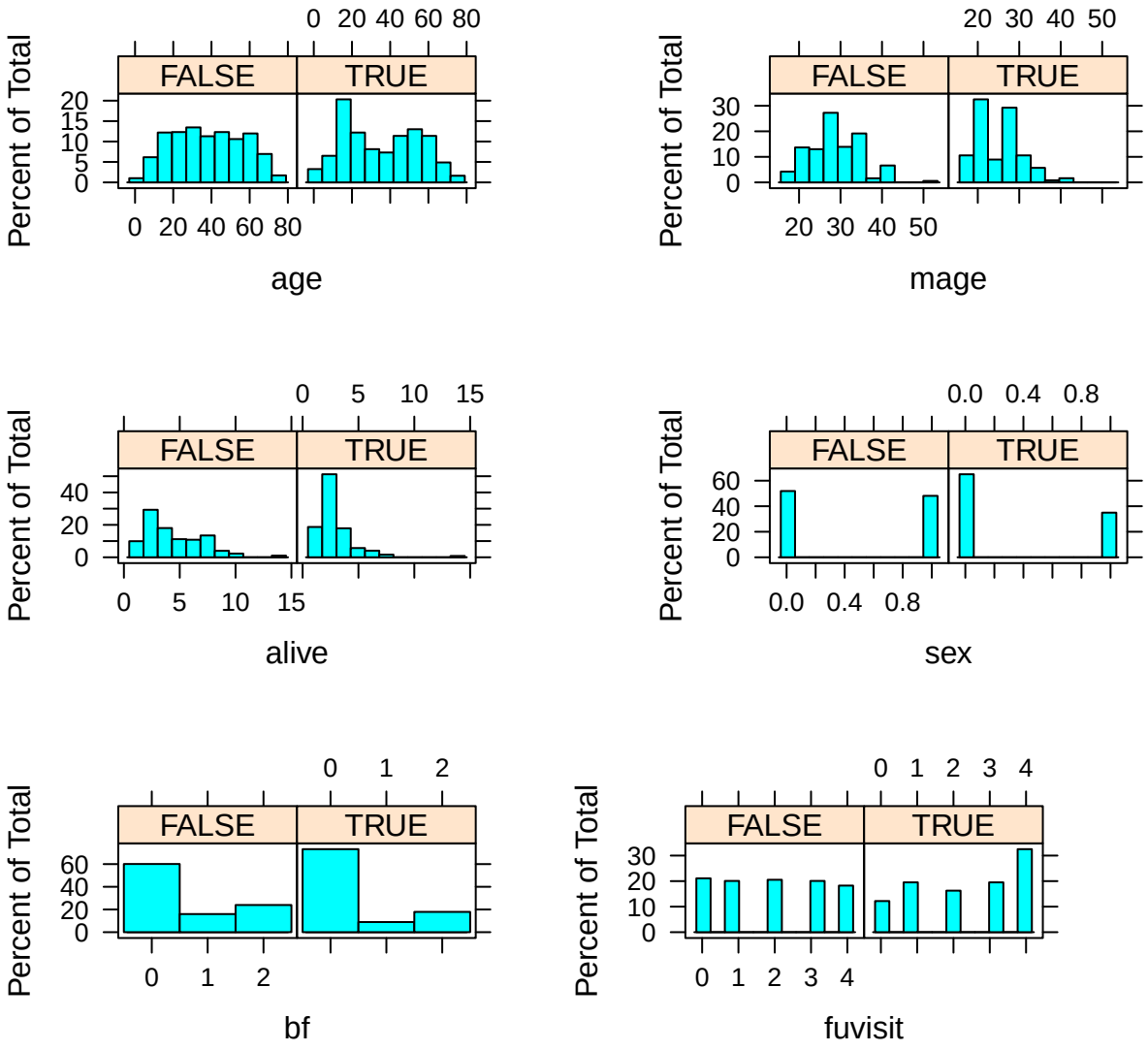

```
##     wt.0 wt.2 wt.1 wt.3 wt.4
## 135    1    1    1    1    1   0
## 15     1    1    1    1    0   1
## 6      1    1    1    0    1   1
## 3      1    1    1    0    0   2
## 7      1    1    0    1    1   1
## 3      1    1    0    1    0   2
## 1      1    1    0    0    1   2
## 2      1    1    0    0    0   3
## 3      1    0    1    1    1   1
## 2      1    0    1    1    0   2
## 1      1    0    1    0    0   3
## 1      1    0    0    1    1   2
## 6      1    0    0    0    0   4
## 4      0    1    1    1    1   1
## 3      0    1    1    1    0   2
## 1      0    1    1    0    0   3
## 2      0    0    1    1    1   2
## 1      0    0    1    1    0   3
## 1      0    0    0    0    1   4
## 3      0    0    0    0    0   5
##       15   20   24   24   40 123
```

What do you notice about the patterns of missing data? Do the missing data patterns represent drop-out? or intermittent missing data?

B. What factors correlate with missingness?

First, we create missing data indicators for weight, then we correlate the missing data indicators with available covariates.

```
R.wt = is.na(d$wt)
```



What patterns do you notice? What covariates are related to missing weight?

C. Imputation approaches

We will explore several imputation approaches using the mice function in R.

1. Single imputation approaches

We will demonstrate the simple mean replacement, predicted mean replacement and single value replacement.

```
## Simple mean replacement
imp.mean <- mice(d[,-1], method = "mean", m = 1, maxit = 1, print=F)
## Simple predicted mean replacement
imp.regmean <- mice(d[,-1], method = "norm.predict", m = 1, maxit = 1, print=F)
## Single value imputation
imp.predict <- mice(d[,-1], method = "norm.nob", m = 1, maxit = 1,seed=4321,print=F)

## Check the simple predicted mean replacement
fit = lm(wt~sex+age+mage+bf+alive+fuvisit,data=d)
check.regmean = predict(fit,d)
check.regmean[!is.na(d$wt)] = d$wt[!is.na(d$wt)]

## Compare imputed values
my.out20 = as.data.frame(
cbind(complete(imp.mean)[1:20,],complete(imp.regmean)[1:20,7],check.regmean[1:20],complete(imp.predict)
names(my.out20) = c("fuvisit","sex","age","alive","mage","bf","wt.mean","wt.regmean","wt.confirm","wt.p
my.out20
```

```
##     fuvisit sex age alive mage bf wt.mean wt.regmean wt.confirm wt.predict
## 151       0   0   1     4   26  2    11.2       6.21       6.21       6.47
## 331       0   1  19     8   43  2    11.2       8.88       8.88       9.88
## 506       0   1  23     4   29  1    11.2       8.88       8.88       7.08
## 511       0   1  54     4   29  0    11.2      13.24      13.24      12.46
## 541       0   0  36     1   19  0    11.2      10.90      10.90      11.95
## 661       0   0   2     2   20  0    11.2       6.64       6.64       7.92
## 666       0   0  51     2   20  0    11.2      12.81      12.81      12.05
## 806       0   0  23     1   18  1    11.2       8.76       8.76       9.03
## 876       0   0   0     2   21  2    11.2       5.89       5.89       6.49
## 886       0   1  12     4   25  1    11.2       7.31       7.31       7.56
## 891       0   1  37     4   25  0    11.2      10.91      10.91       9.63
## 896       0   1   4     1   22  2    11.2       6.13       6.13       2.76
## 916       0   0  14     3   26  1    11.2       7.97       7.97       8.74
## 936       0   1  12     2   24  1    11.2       7.30       7.30       7.35
## 941       0   0  46     2   24  0    11.2      12.38      12.38      14.94
## 1         0   0  41     5   35  0    12.8      12.80      12.80      12.80
## 6         0   1  57     5   35  0    14.9      14.90      14.90      14.90
## 11        0   1   8     5   35  2     7.7       7.70       7.70       7.70
## 16        0   1  35     5   35  0    12.1      12.10      12.10      12.10
## 21        0   0  49     5   35  0    14.2      14.20      14.20      14.20
```

2. Multiple imputation

Next, we will perform a multiple imputation for weight by creating $M = 5$ imputed datasets. In practice, you typically set M to some large number.

```
imp.mult = mice(d[,-1],print=F)
## Print the predictorMatrix
imp.mult$predictorMatrix
```

```
##         fuvisit sex age alive mage bf wt
## fuvisit       0   1   1     1    1  1  1
## sex           1   0   1     1    1  1  1
## age           1   1   0     1    1  1  1
## alive         1   1   1     0    1  1  1
## mage          1   1   1     1    0  1  1
## bf            1   1   1     1    1  0  1
## wt            1   1   1     1    1  1  0
```

```
## Print the complete data for the first 20 obs
complete(imp.mult,"broad")[1:20,c(1:7,14,21,28,35)]
```

```
## New names:
## * fuvisit -> fuvisit...1
## * sex -> sex...2
## * age -> age...3
## * alive -> alive...4
## * mage -> mage...5
## * ...
```

```
##     fuvisit.1 sex.1 age.1 alive.1 mage.1 bf.1 wt.1 wt.2 wt.3 wt.4 wt.5
## 151         0     0     1       4     26    2  5.2  7.4  5.8  4.1  4.1
## 331         0     1    19       8     43    2  9.8  7.5  8.1  9.1  9.1
## 506         0     1    23       4     29    1  8.7  9.1  5.1 10.0  9.6
## 511         0     1    54       4     29    0 12.4 12.1 13.0 15.4 12.1
## 541         0     0    36       1     19    0 11.4 12.0 10.2 12.0  8.9
## 661         0     0     2       2     20    0  5.8  6.6  7.2  8.2  8.2
## 666         0     0    51       2     20    0 11.7 13.3 16.5 14.5 13.8
## 806         0     0    23       1     18    1  9.2  9.7  9.7 10.3  9.7
## 876         0     0     0       2     21    2  4.1  4.6  7.0  4.6  4.6
## 886         0     1    12       4     25    1  6.7  7.9  4.7  9.1  5.2
## 891         0     1    37       4     25    0 11.3 10.8  8.5 11.3 10.6
## 896         0     1     4       1     22    2  6.8  5.2  6.6  5.2  4.1
## 916         0     0    14       3     26    1  7.1  8.3  9.0  7.7  8.7
## 936         0     1    12       2     24    1  5.9  8.8  5.8  7.5  6.7
## 941         0     0    46       2     24    0 11.9 11.9 15.5 12.6 12.3
## 1           0     0    41       5     35    0 12.8 12.8 12.8 12.8 12.8
## 6           0     1    57       5     35    0 14.9 14.9 14.9 14.9 14.9
## 11          0     1     8       5     35    2  7.7  7.7  7.7  7.7  7.7
## 16          0     1    35       5     35    0 12.1 12.1 12.1 12.1 12.1
## 21          0     0    49       5     35    0 14.2 14.2 14.2 14.2 14.2
```

4. Changing the prediction models

You can control the "prediction matrix" by telling mice what variables you want to include. Alternatively, you create your own predictorMatrix (we will do this later in the lab).

```
imp.mult$predictorMatrix
```

```
##         fuvisit sex age alive mage bf wt
## fuvisit       0   1   1     1    1  1  1
## sex           1   0   1     1    1  1  1
## age           1   1   0     1    1  1  1
## alive         1   1   1     0    1  1  1
## mage          1   1   1     1    0  1  1
## bf            1   1   1     1    1  0  1
## wt            1   1   1     1    1  1  0
```

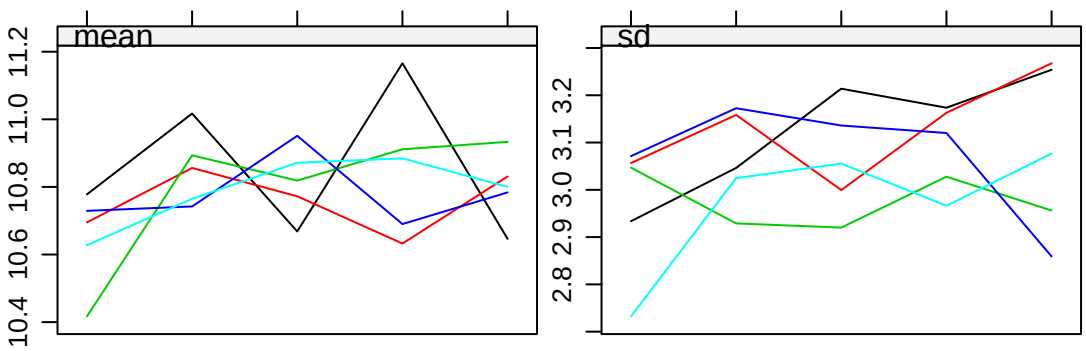
```
## Drop parity from prediction model
new.pMatrix = imp.mult$predictorMatrix
new.pMatrix[,"alive"] = 0
imp.mult2 <- mice(d[,-1],predictorMatrix = new.pMatrix,print=F)
## Or you can select variables based on a minimum level of correlation
imp.mult3 = mice(d[,-1], pred=quickpred(d[,-1], mincor=.3), print=F)
imp.mult3$predictorMatrix
```

```
##         fuvisit sex age alive mage bf wt
## fuvisit       0   0   0     0    0  0  0
## sex           0   0   0     0    0  0  0
## age           0   0   0     0    0  0  0
## alive         0   0   0     0    0  0  0
## mage          0   0   0     0    0  0  0
## bf            0   0   0     0    0  0  0
## wt            0   0   1     0    0  1  0
```

5. Summaries of the MCMC algorithm

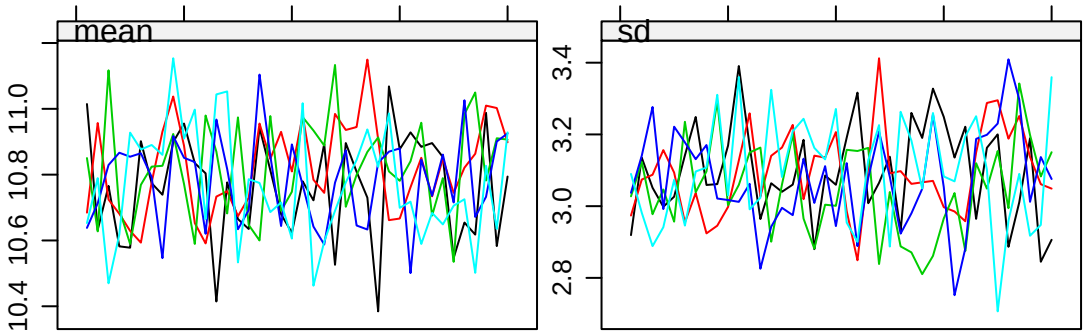
You can look to see the imputed values over the MCMC samples and compare the observed values with imputed values using special "plot" commands.

```
par(mfrow=c(1,1))
plot(imp.mult)
```



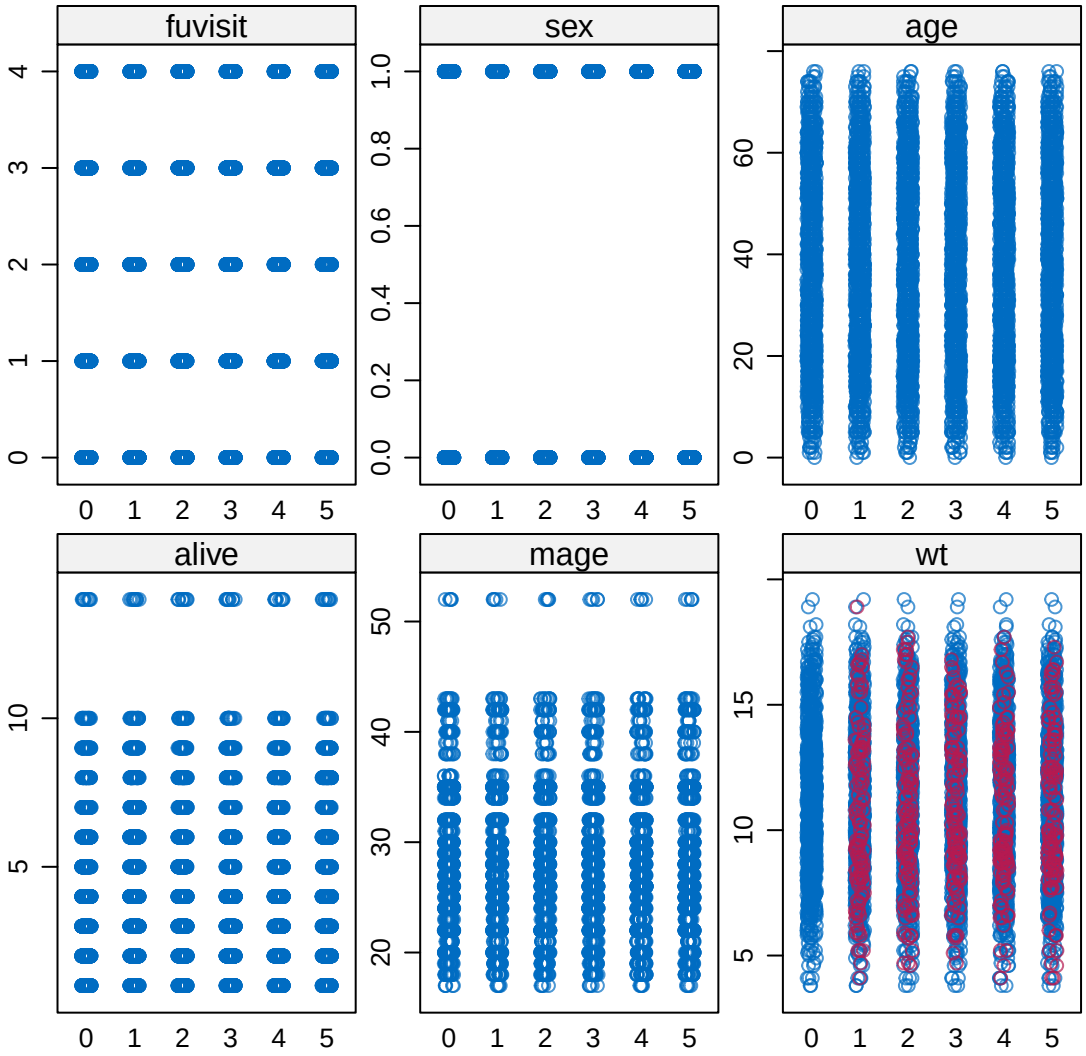
Iteration

```
## Increase the length of the mcmc chain
imp.multlonger = mice(d[,-1],maxit=40,print=F)
plot(imp.multlonger)
```



Iteration

```
## Look at strip plot; shows observed and imputed values
stripplot(imp.mult)
```


fuvisit
sex
age
alive
mage
wt

6. Fitting a model to the imputed data

Here we will compare the results of the analysis conducted on the observed/available data with the results based on the imputed datasets.

```
fit.obs = lm(wt~ns(age,3)+bf+mage+alive+sex,data=d,na.action=na.omit)

## Fit the same model to each of the imputed datasets
fit.mean = lm(wt~ns(age,3)+bf+mage+alive+sex,data=complete(imp.mean))
fit.regmean = lm(wt~ns(age,3)+bf+mage+alive+sex,data=complete(imp.regmean))
fit.predict = lm(wt~ns(age,3)+bf+mage+alive+sex,data=complete(imp.predict))
fit.mult = with(imp.mult,lm(wt~ns(age,3)+bf+mage+alive+sex))
# You can look at the fit for each of the imputed datasets
#summary(fit$analyses[[1]])
pool.mult = pool(fit.mult)
est = as.data.frame(round(cbind(fit.obs$coefficients,
                          fit.mean$coefficients,
                          fit.regmean$coefficients,
                          fit.predict$coefficients,
                          summary(pool.mult)[,2]),3))
names(est) = c("obs","mean","regmean","predicted","mult")
est
```

```
##                obs   mean regmean predicted   mult
## (Intercept)  4.282  6.224   4.566     4.721  4.533
## ns(age, 3)1  6.507  4.696   6.267     6.239  6.330
## ns(age, 3)2 12.957  9.690  12.539    12.208 12.531
## ns(age, 3)3  7.635  6.191   7.575     7.389  7.504
## bf1         -0.317 -0.701  -0.377    -0.430 -0.388
## bf2         -0.158 -0.784  -0.266    -0.385 -0.275
## mage         0.051  0.045   0.050     0.050  0.051
## alive       -0.029 -0.026  -0.028    -0.030 -0.031
## sex         -0.323 -0.280  -0.324    -0.349 -0.342
```

```
se = as.data.frame(round(cbind(summary(fit.obs)$coefficients[,2],
              summary(fit.mean)$coefficients[,2],
              summary(fit.regmean)$coefficients[,2],
              summary(fit.predict)$coefficients[,2],
              summary(pool.mult)[,3]),3))
names(se) = c("obs","mean","regmean","predicted","mult")
se
```

```
##               obs  mean regmean predicted  mult
## (Intercept) 0.403 0.410   0.334     0.356 0.542
## ns(age, 3)1 0.292 0.294   0.240     0.256 0.289
## ns(age, 3)2 0.637 0.661   0.539     0.575 0.686
## ns(age, 3)3 0.263 0.278   0.226     0.241 0.296
## bf1         0.162 0.172   0.140     0.149 0.180
## bf2         0.182 0.184   0.150     0.160 0.193
## mage        0.012 0.014   0.011     0.012 0.015
## alive       0.030 0.034   0.028     0.029 0.033
## sex         0.092 0.099   0.081     0.086 0.091
```

7. Accounting for the longitudinal structure

So far we have ignored the longitudinal structure of the data. We can incorporate the longitudinal information by doing the imputation on the data in a wide format!

```
## Create 10 imputed datasets!

## Create a predictionMatrix for the analysis
## Variables are: sex, alive, mage, wt.0,age.0,bf.0,
## wt.1,age.1,bf.1, wt.2,age.2,bf.2, wt.3,age.3,bf.3,
## wt.4,age.4,bf.4
##
## Initialize the matrix
predM = matrix(1,nrow=18,ncol=18)
for(i in 1:18) predM[i,i] = 0
## wt.0 will be predicted by
v0 = c("sex","alive","mage","age.0","bf.0")
## wt.1 will be predicted by
v1 = c("sex","alive","mage","wt.0","bf.0","age.1","bf.1")
## wt.2 will be predicted by
v2 = c("sex","alive","mage","wt.1","bf.1","age.2","bf.2")
## wt.3 will be predicted by
v3 = c("sex","alive","mage","wt.2","bf.2","age.3","bf.3")
## wt.4 will be predicted by
v4 = c("sex","alive","mage","wt.3","bf.3","age.4","bf.4")
names(d.wide)
```

```
##  [1] "id"    "sex"   "alive" "mage"  "wt.0"  "age.0" "bf.0"  "wt.1"  "age.1"
## [10] "bf.1"  "wt.2"  "age.2" "bf.2"  "wt.3"  "age.3" "bf.3"  "wt.4"  "age.4"
## [19] "bf.4"
```

```
# Define the model for wt.0, wt.1, wt.2, wt.3, wt.4
predM[4,] = ifelse(!is.na(match(names(d.wide)[-1],v0)),1,0)
predM[7,] = ifelse(!is.na(match(names(d.wide)[-1],v1)),1,0)
predM[10,] = ifelse(!is.na(match(names(d.wide)[-1],v2)),1,0)
predM[13,] = ifelse(!is.na(match(names(d.wide)[-1],v3)),1,0)
predM[16,] = ifelse(!is.na(match(names(d.wide)[-1],v4)),1,0)
row.names(predM) = names(d.wide)[-1]
colnames(predM) = names(d.wide)[-1]
## Conduct the multiple imputation
wide.imp = mice(d.wide[,-1],predictorMatrix=predM,print=F,m=10)
```

```
## Warning: Number of logged events: 4
```

```
wide.imp$loggedEvents
```

```
##   it im dep      meth   out
## 1  0  0     collinear age.1
## 2  0  0     collinear age.2
## 3  0  0     collinear age.3
## 4  0  0     collinear age.4
```

```
## Once we have the imputed dataset,
## we can look at the imputed values
## For instance, look at the set of imputed
## values for wt.0
wide.imp$imp$wt.0
```

```
##        1    2    3    4    5    6    7    8    9   10
## 151  4.1  3.8  7.4  4.6  4.9  4.6  5.9  5.9  5.9  3.8
## 331  8.7  8.5  9.3  8.5  9.1  7.0  7.3  7.3  8.6 10.1
## 506  8.2 10.0  8.6  8.7  8.2  7.8  9.1 10.0 10.1 10.1
## 511 12.4 12.3 15.4 14.9 13.0 14.2 11.7 12.7 10.6 13.0
## 541 12.0  8.1 10.3 13.2  9.2  9.2 16.7 11.8 12.0 10.6
## 661  5.9  7.0  5.9  4.1  5.8  4.1  7.6  7.0  7.2  5.1
## 666 11.7 15.4 12.7 13.9 14.3 10.6 12.8 15.5 12.8 12.5
## 806 10.6  8.7  8.7  8.8  8.7  8.6  7.8  8.5  8.8  8.2
## 876  3.8  5.2  4.6  4.9  4.9  4.9  3.8  3.8  6.8  7.4
## 886  7.2  6.7  8.0  5.1  5.2  6.8  5.2  5.9  7.3  5.8
## 891 10.4 16.7  9.2 11.8 12.0 11.2 11.2 10.5 10.4 13.1
## 896  7.4  5.2  7.4  4.1  4.1  4.6  7.4  4.6  7.4  4.9
## 916  9.3  7.0  5.9  4.7  5.4  7.6  7.0  7.6  8.8  8.8
## 936  6.8  5.9  7.3  6.6  6.7  8.4  8.0  7.2  5.8  8.0
## 941 11.6 12.4  9.9 11.8 13.8 16.3 13.8 13.9 16.2 13.8
```

```
## Reconfigure each dataset back into long format
## First create a list of the completed wide datasets
id = 1:nrow(d.wide)
all.imp.data <- as.list(1:10)
for (i in 1:10){
    all.imp.data[[i]] <- cbind(id,complete(wide.imp,action=i))
}

## Then reconfigure each data set and add a fuvisit variable
long.imp.data2 <- lapply(all.imp.data,reshape,varying=5:19,idvar="id",
                         timevar="fuvisit",times=0:4,direction="long")

long.imp.data2 <- lapply(long.imp.data2,FUN=function(u){ u[order(u$id,d$fuvisit),]})

## Run the a random intercept + slope for age model
## on each dataset
long.imp.re <- lapply(long.imp.data2, FUN=function(u){
  u$agec = u$age-6
  u$agesp24 = ifelse(u$age>24,u$age-24,0)
  u$agesp48 = ifelse(u$age>48,u$age-48,0)
  lmer(wt ~ agec+agesp24+agesp48 + (1+age|id),data=u,
       control = lmerControl(optimizer ="Nelder_Mead"))
  })

## Save the set of regression coefficients and
## variance covariance matrices
hyper.imp.parms <- lapply(long.imp.re,fixed.effects)
hyper.imp.vcov <- lapply(long.imp.re,FUN=function(u) as.matrix(vcov(u)))

## Average the estimates
my.est = matrix(unlist(hyper.imp.parms),ncol=4,byrow=T)
rubin.est = apply(my.est,2,mean)
rubin.var = apply(array(unlist(hyper.imp.vcov),c(4,4,5)),1:2,mean) + (1+1/10)*cov(my.est)

## Fit the the model to the available data
d$agec = d$age - 6
d$agesp24 = ifelse(d$age>24,d$age-24,0)
```

```
d$agesp48 = ifelse(d$age>48,d$age-48,0)
fit.available = lmer(wt ~ agec+agesp24+agesp48 + (1+age|id),data=d,
control=lmerControl(optimizer ="Nelder_Mead"))

my.out = as.data.frame(cbind(fixed.effects(fit.available),
                        sqrt(diag(vcov(fit.available))),
                          rubin.est,sqrt(diag(rubin.var))))
names(my.out) = c("est.avail","se.avail","est.imp","se.imp")
my.out
```

```
##             est.avail se.avail est.imp  se.imp
## (Intercept)    6.4544  0.12662  6.4597 0.15759
## agec           0.1661  0.00634  0.1684 0.00863
## agesp24       -0.0304  0.00865 -0.0362 0.01107
## agesp48       -0.0241  0.00734 -0.0202 0.00867
```