Handling Missing Data in Health Science Research

Day 1 - Part II

2022-06-21

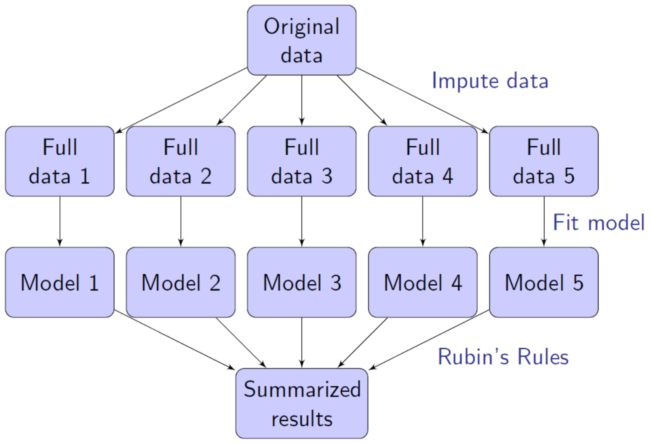
Contents

Missing data methods for cross-sectional data

- Complete-case analysis (list-wise deletion)
  - Wasteful
  - Parameter estimates might be unbiased depending on the missing data mechanism
  - Produces standard errors that are too large
- Single value imputation
  - Mean imputation
  - Regression imputation
  - These produce standard errors that are too small
- Many R packages are available

Multiple imputation

- Multiple imputation (MI) is a simulation-based method for filling in missing values using observed data
- Typical MI approach involves 3 basic steps
  1. Imputation
  2. Model fitting
  3. Summarize estimates using Rubin's Rules (Rubin, 1987)
- MI is valid when data is MCAR/MAR
- MI requires the correct specification of the **imputation model**



Prior to MI

- Decide on the **analysis model** based on the scientific question of interest
  - At this step, it is important to identify confounders, interaction terms, quadratic terms, etc. that will be included in the model
  - More on how to impute interaction terms later
- Identify the variables to include in the **imputation model**
  - The imputation model should contain any variables that will be in the analysis model, plus auxiliary variables
  - The imputation model is typically more general than the analysis model

Imputation step

- Choose m (number of imputations)
- Select the imputation method
  - **Joint modeling (JM)** (We will cover R packages amelia and jomo)
  - **Fully conditional specification (FCS)** (We will cover the R package mice)
- Perform imputation
- This will result in m full data sets in wide format
  - Each data set will be slightly different from each other

Modeling step

- Analyze each of m data with the same model
- This will result in m different parameter estimates and variance estimates
  - $\hat{\beta}^{(k)}$ and $\widehat{\text{Cov}}(\hat{\beta}^{(k)})$ for $k = 1, ..., m$

Pooling step

- Now we combine all m estimates to arrive at a single $\hat{\beta}$ and $\widehat{\text{Cov}}(\hat{\beta})$ using **Rubin's rules**
- $\hat{\beta}$ is just the average of all $\hat{\beta}^{(k)}$ for $k = 1, ..., m$

$$\hat{\beta} = \bar{\beta} = \frac{1}{m}\sum_{k=1}^{m}\hat{\beta}^{(k)}$$

- The estimated covariance for $\hat{\beta}$ is given by combining two sources of variability:
  - **Within-imputation** variability (W)
  - **Between-imputation** variability (B)

$$\widehat{\text{Cov}}(\hat{\beta}) = W + (1 + m^{-1})B,$$

  where

$$W = \frac{1}{m}\sum_{k=1}^{m}\widehat{\text{Cov}}(\hat{\beta}^{(k)})$$

  and

$$B = \frac{1}{m-1}\sum_{k=1}^{m}\left(\hat{\beta}^{(k)} - \bar{\beta}\right)\left(\hat{\beta}^{(k)} - \bar{\beta}\right)^T.$$

Joint modeling (JM)

- Specify a joint model for the entire data set, usually under multivariate normal (MVN)
- Derive the posterior predictive distribution i.e. distribution of unobserved values conditional on observed data

Fully conditional specification (FCS)

- Designed to handle variables of mixed type (continuous, categorical, count, etc.)
- Specify a conditional model for each missing variable
- Impute data on a variable-by-variable basis

van Buuren (2007) compares the two methods in greater detail

- We will use three different packages in R (Amelia, jomo, and mice) to illustrate how to perform MI on `skin_mar` data set

```r
skin_mar <- read.table("skinb_MAR.txt", header = TRUE)
## you'll need to convert your categorical variables into factor in R;
## otherwise there will be warnings
skin_mar[, c("center","skin","gender","treatment")] <-
  lapply(skin_mar[, c("center","skin","gender","treatment")], factor)
head(skin_mar)
```

```
##       ID center age skin gender exposure treatment Y
## 1 100034      1  NA    1      1        4         0 0
## 2 100045      1  68    1      0        2         0 0
## 3 100056      1  58    1      0        7         0 1
## 4 100067      1  53    1      1        3         0 0
## 5 100102      1  55    0      0        2         0 0
## 6 100113      1  59 <NA>      1       10         0 0
```

JM in R

Amelia package

- Amelia is an R package that was developed by Gary King, James Honaker, Anne Joseph, and Kenneth Scheve in 2001
- A newer version with GUI was released in 2011 (https://gking.harvard.edu/files/gking/files/amelia_jss.pdf)
- Amelia assumes that your data is jointly distributed as multivariate normal (JM)
- The imputation algorithm is based on Expectation-maximization with bootstraping (EMB)
- Vignettes
  - https://cran.r-project.org/web/packages/Amelia/vignettes/intro-mi.html
  - https://cran.r-project.org/web/packages/Amelia/vignettes/using-amelia.html

A note about imputing categorical variables using JM

In JM, all variables in the data including the missing values are assumed to follow a multivariate normal distribution. Therefore, when missing categorical variables are imputed, the imputed values may contain non-integers.

Ordinal variables

If the ordinal variable (including binary variable) that contains missing values is a predictor and if you intend to include the variable in the model as a continuous variable, then no extra step is required. We can learn more about the underlying distribution of the data if we leave the imputed values as non-integers rather than forcing them to be integers. Therefore whenever the analysis model permits, the imputed ordinal predictors should be allowed to take on continuous values. In particular, Horton (2003) recommends **not to** round the imputed binary variables at 0.5!

Nominal variables

Nominal variables don't have inherent orders in their categories so non-integer imputed values are not meaningful. In Amelia, nominal variables of p categories can be listed in the `nom =` option. Then, the function will automatically $p-1$ dummy variables and each of them will be imputed under the multivariate normal distribution with the rest of the data. The imputed values will be appropriately scaled into probabilities for each of the p possible categories, and one of the categories will be drawn. The function will reconstruct the original p-category multinomial variable and return it to the user. Note we need at least one non-missing observation from each of the p categories.

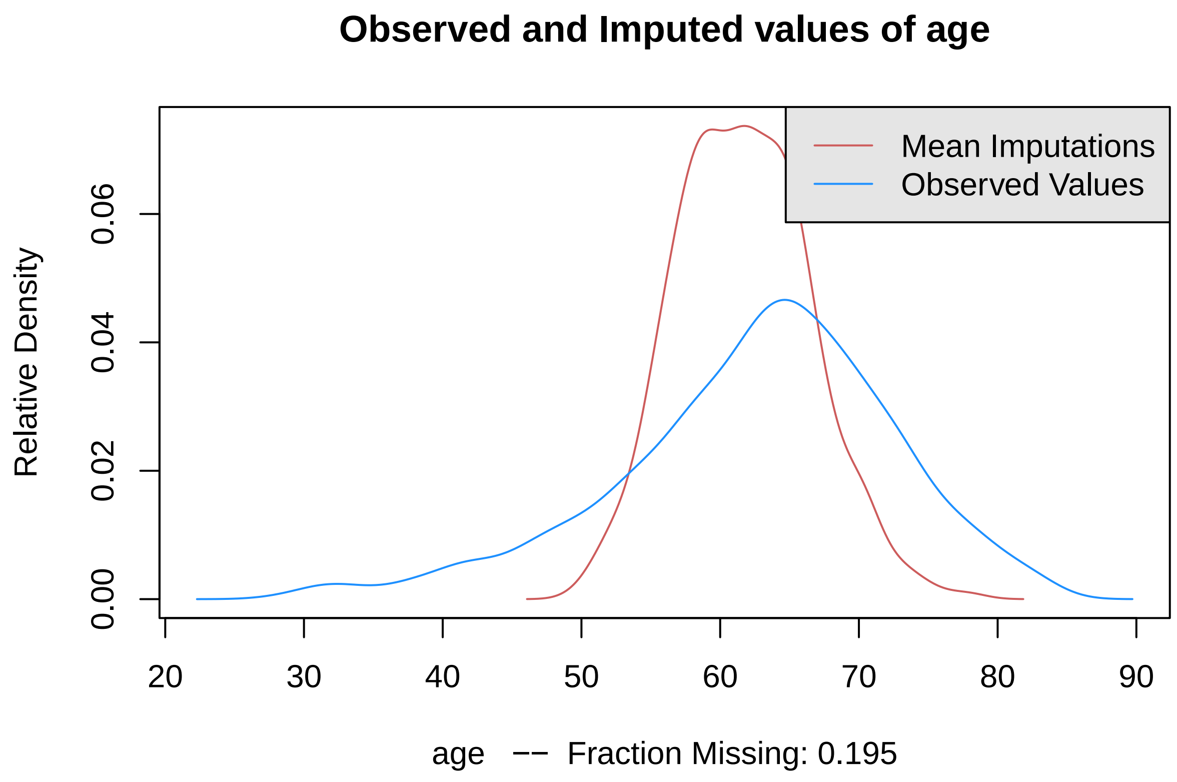
```r
library(Amelia)
library(mitml) # required for `amelia2mitml.list()` and `testEstimates()`

set.seed(100) ## set seed to ensure reporducibility

start.time <- Sys.time()
# a vector of numbers or names indicating columns in the data that are nominal variables
amelia.out <- amelia(skin_mar, m=5, noms = c("center", "gender", "treatment", "skin"),
                     idVars = "ID")
```

```
## -- Imputation 1 --
##
##   1  2  3  4  5  6  7
##
## -- Imputation 2 --
##
##   1  2  3  4  5  6  7  8
##
## -- Imputation 3 --
##
##   1  2  3  4  5  6  7  8  9 10
##
## -- Imputation 4 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
##
## -- Imputation 5 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13
```

```
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 0.260047 secs
```

```
# look at the observed and imputed values density for one variable
compare.density(amelia.out, var="age")
```



```
# save imputation output as a list and fit glm on each imputed data
implist.amelia <- amelia2mitml.list(amelia.out)
fit.amelia <- with(implist.amelia, glm(Y ~ age + treatment + gender + exposure + skin,
                                       family = poisson("log")))

# apply rubin's rule using testEstimates() from mitml
testEstimates(fit.amelia)
```

```
##
## Call:
##
## testEstimates(model = fit.amelia)
##
## Final parameter estimates and inferences obtained from 5 imputed data sets.
##
##             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
## (Intercept)   -3.279     0.753    -4.353 6.328e+00     0.004     3.879     0.839
## age            0.014     0.012     1.179 5.838e+00     0.284     4.807     0.867
## treatment1     0.057     0.095     0.602 1.090e+03     0.548     0.064     0.062
## gender1        0.482     0.118     4.089 2.311e+06     0.000     0.001     0.001
## exposure       0.128     0.007    18.107 1.449e+03     0.000     0.055     0.054
## skin1          0.358     0.118     3.041 3.626e+01     0.004     0.497     0.366
##
## Unadjusted hypothesis test as appropriate in larger samples.
```

- RIV: relative increase in variance due to nonresponse

- FMI: fraction of missing information
- More information about these definitions can be found here: https://bookdown.org/mwheymans/bookmi/measures-of-missing-data-information.html

Jomo package

- The R package Jomo was developed by Matteo Quartagno, Simon Grund, and James Carpenter in 2019
  - https://journal.r-project.org/archive/2019/RJ-2019-028/RJ-2019-028.pdf
- Jomo is based on JM imputation for clustered/multilevel data, but can be used for cross-sectional data as well
- Jomo handles binary and categorical data through **latent normal variables**

```
library(jomo)
library(mitools)

# separate data into
# Y: variables that are missing
# X: variables that are complete

Y.jomo <- skin_mar[,c("age", "skin")]
X.jomo <- skin_mar[,c("center", "gender", "exposure", "treatment", "Y")]

# specify a column of 1 if we want an intercept
X.jomo$cons <- 1

head(Y.jomo)
```

```
##   age skin
## 1  NA    1
## 2  68    1
## 3  58    1
## 4  53    1
## 5  55    0
## 6  59 <NA>
```

```
head(X.jomo)
```

```
##   center gender exposure treatment Y cons
## 1      1      1        4         0 0    1
## 2      1      0        2         0 0    1
## 3      1      0        7         0 1    1
## 4      1      1        3         0 0    1
## 5      1      0        2         0 0    1
## 6      1      1       10         0 0    1
```

```
set.seed(100) # set seed for reproducibility

start.time <- Sys.time()
jomo.out <-jomo(Y = Y.jomo, X = X.jomo, nburn = 1000, nbetween = 1000, nimp = 5)
```

```
## No clustering, using functions for single level imputation.
## Found  1 continuous outcomes and  1 categorical. Using function jomo1mix.
## ..................................................
## ..................................................
## First imputation registered.
## ..................................................
## ..................................................
## Imputation number  2 registered
## ..................................................
## ..................................................
## Imputation number  3 registered
## ..................................................
## ..................................................
## Imputation number  4 registered
## ..................................................
## ..................................................
## Imputation number  5 registered
## The posterior mean of the fixed effects estimates is:
##             center     gender    exposure   treatment          Y       cons
## age    -0.86387663 -0.9364850  0.20314336 -0.17763248  0.4795268 66.0304014
## skin.1  0.03007637  0.1017703 -0.05182371 -0.09564998 -0.1230650  0.2175744
##
## The posterior covariance matrix is:
##               age    skin.1
## age    98.6647988 0.2726279
## skin.1  0.2726279 1.0000000
```

```
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 9.070308 secs
```

```
# look at the imputed dataset
head(jomo.out[which(jomo.out$Imputation == 1),])
```

```
##           age skin center gender exposure treatment Y cons id Imputation
## 1684 80.05501    1      1      2        4         1 0    1  1          1
## 1685 68.00000    1      1      1        2         1 0    1  2          1
## 1686 58.00000    1      1      1        7         1 1    1  3          1
## 1687 53.00000    1      1      2        3         1 0    1  4          1
## 1688 55.00000    0      1      1        2         1 0    1  5          1
## 1689 59.00000    1      1      2       10         1 0    1  6          1
```

```
# get separate imputed dataset manually using imputationList() from mitools
imp.list.jomo <- imputationList(split(jomo.out, jomo.out$Imputation)[-1])
fit.jomo <- with(imp.list.jomo, glm(Y ~ age + treatment + gender + exposure + skin,
                                    family = poisson("log")))

testEstimates(fit.jomo)
```

```
##
## Call:
```

```
##
## testEstimates(model = fit.jomo)
##
## Final parameter estimates and inferences obtained from 5 imputed data sets.
##
##              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
## (Intercept)    -3.760     0.599    -6.275    14.285     0.000     1.124     0.584
## age             0.013     0.009     1.476     8.645     0.175     2.127     0.735
## treatment       0.043     0.093     0.469 11321.570     0.639     0.019     0.019
## gender          0.484     0.118     4.089 37544.852     0.000     0.010     0.010
## exposure        0.126     0.007    16.983   292.241     0.000     0.132     0.123
## skin1           0.423     0.116     3.644    45.753     0.001     0.420     0.325
##
## Unadjusted hypothesis test as appropriate in larger samples.
```

FCS in R

Fully conditional specification (FCS) was proposed as a tool for handling a mixture of missing continuous and categorical variables. In particular, van Buuren proposed to impute each missing variable using an appropriate model conditional on other observed variables. For example, binary variables can be imputed based on a logistic regression model.

Mice package

- Mice stands for "Multiple Imputation by Chained Equations" and the package was developed by Stef van Buuren (https://www.jstatsoft.org/article/view/v045i03)
- Mice imputes missing values with plausible values
- These plausible values are drawn from a distribution specifically designed for each missing variable

```
library(mice)

set.seed(100) # set seed for reproducibility

# norm for continuous data, logreg for binary data
start.time <- Sys.time()
mice.out <- mice(skin_mar[,-1], seed = 1, m = 5,
                 method=c("", "norm", "logreg" ,"", "", "", ""))
```

```
##
##  iter imp variable
##   1   1  age  skin
##   1   2  age  skin
##   1   3  age  skin
##   1   4  age  skin
##   1   5  age  skin
##   2   1  age  skin
##   2   2  age  skin
##   2   3  age  skin
##   2   4  age  skin
##   2   5  age  skin
##   3   1  age  skin
##   3   2  age  skin
##   3   3  age  skin
##   3   4  age  skin
##   3   5  age  skin
##   4   1  age  skin
##   4   2  age  skin
##   4   3  age  skin
##   4   4  age  skin
##   4   5  age  skin
##   5   1  age  skin
##   5   2  age  skin
##   5   3  age  skin
##   5   4  age  skin
##   5   5  age  skin
```

```
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 0.6077211 secs
```

```
# extract complete data from mice output
complete.data <- complete(mice.out, "all")

# check the first imputed dataset
head(complete.data$`1`)
```

```
##   center      age skin gender exposure treatment Y
## 1      1 57.80555    1      1        4         0 0
## 2      1 68.00000    1      0        2         0 0
## 3      1 58.00000    1      0        7         0 1
## 4      1 53.00000    1      1        3         0 0
## 5      1 55.00000    0      0        2         0 0
## 6      1 59.00000    0      1       10         0 0
```

```
fit.mice <- with(mice.out, glm(Y ~ age + treatment + gender + exposure + skin,
                               family = poisson("log")))

# the pool() function from mice could also calculate the pooled result
pool(fit.mice)
```

```
## Class: mipo    m = 5
##          term m     estimate         ubar            b            t dfcom
## 1 (Intercept) 5 -2.962608572 1.136465e-01 4.844027e-01 6.949297e-01  1677
## 2         age 5  0.008061149 2.386011e-05 1.065226e-04 1.516872e-04  1677
## 3  treatment1 5  0.048528773 8.415086e-03 1.104217e-04 8.547592e-03  1677
## 4     gender1 5  0.482039023 1.389722e-02 1.204191e-04 1.404172e-02  1677
## 5    exposure 5  0.126630944 4.748527e-05 1.826999e-05 6.940926e-05  1677
## 6       skin1 5  0.477060058 9.449153e-03 1.389216e-02 2.611974e-02  1677
##            df        riv     lambda        fmi
## 1    5.600096 5.11483741 0.83646335 0.87449471
## 2    5.514745 5.35735457 0.84270187 0.87964910
## 3 1500.389703 0.01574625 0.01550215 0.01681185
## 4 1588.064360 0.01039798 0.01029097 0.01153505
## 5   38.736575 0.46170070 0.31586542 0.34864887
## 6    9.663054 1.76424179 0.63823715 0.69537389
```

```r
# or use testEstimates() from mitml
testEstimates(as.mitml.result(fit.mice))
```

```
##
## Call:
##
## testEstimates(model = as.mitml.result(fit.mice))
##
## Final parameter estimates and inferences obtained from 5 imputed data sets.
##
##             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
## (Intercept)   -2.963     0.834    -3.554     5.717     0.013     5.115     0.874
## age            0.008     0.012     0.655     5.633     0.539     5.357     0.879
## treatment1     0.049     0.092     0.525 16644.714     0.600     0.016     0.016
## gender1        0.482     0.118     4.068 37770.034     0.000     0.010     0.010
## exposure       0.127     0.008    15.200    40.092     0.000     0.462     0.348
## skin1          0.477     0.162     2.952     9.820     0.015     1.764     0.695
##
## Unadjusted hypothesis test as appropriate in larger samples.
```

Methods available in mice:

```r
methods(mice)
```

```
##  [1] mice.impute.2l.bin              mice.impute.2l.lmer
##  [3] mice.impute.2l.norm             mice.impute.2l.pan
##  [5] mice.impute.2lonly.mean         mice.impute.2lonly.norm
##  [7] mice.impute.2lonly.pmm          mice.impute.cart
##  [9] mice.impute.jomoImpute          mice.impute.lasso.logreg
## [11] mice.impute.lasso.norm          mice.impute.lasso.select.logreg
## [13] mice.impute.lasso.select.norm   mice.impute.lda
## [15] mice.impute.logreg              mice.impute.logreg.boot
## [17] mice.impute.mean                mice.impute.midastouch
## [19] mice.impute.mnar.logreg         mice.impute.mnar.norm
## [21] mice.impute.norm                mice.impute.norm.boot
## [23] mice.impute.norm.nob            mice.impute.norm.predict
## [25] mice.impute.panImpute           mice.impute.passive
## [27] mice.impute.pmm                 mice.impute.polr
## [29] mice.impute.polyreg             mice.impute.quadratic
## [31] mice.impute.rf                  mice.impute.ri
## [33] mice.impute.sample              mice.mids
## [35] mice.theme
## see '?methods' for accessing help and source code
```

Imputing interaction variables using FCS

Now suppose our analysis model includes an interaction effect between two of the predictors, and one or two of the predictors contain missing values. Because the imputation model should be congenial to the analysis model, we need to include the interaction term in the imputation model as well.

Impute then transform

- Only include main effects and other variables in the imputation model
- Create the interaction variable after the imputation step
- Not generally recommended

Transform then impute

- **Active imputation**
- Includes the interaction variable in the imputation model with all other variables along with the main effects
- Assumes the interaction variable to be another independent variable
- Also referred to as "Just Another Variable" (JAV)
- The relationship between the interaction effect and the main effects are not necessarily internally consistent
- **Passive imputation**
- Passively imputes the interaction variable
- The interaction variable is used to impute other missing values but not the main effects that are used to create the interaction effect
- The relationship between the interaction effect and the main effects is preserved
- **Improved passive imputation**
- In addition to passive imputation, it includes the interaction of a main effect and the outcome as one of the predictors of the other main effect in the imputation model
- Intuition behind why this approach may reduce bias compared to the conventional passive imputation is that if there were a true interaction between two main effects in the scientific model, the relationship between one of the main effects and the outcome would vary with the other main effect
- This approach has been shown to produce the least biased estimates in many studies

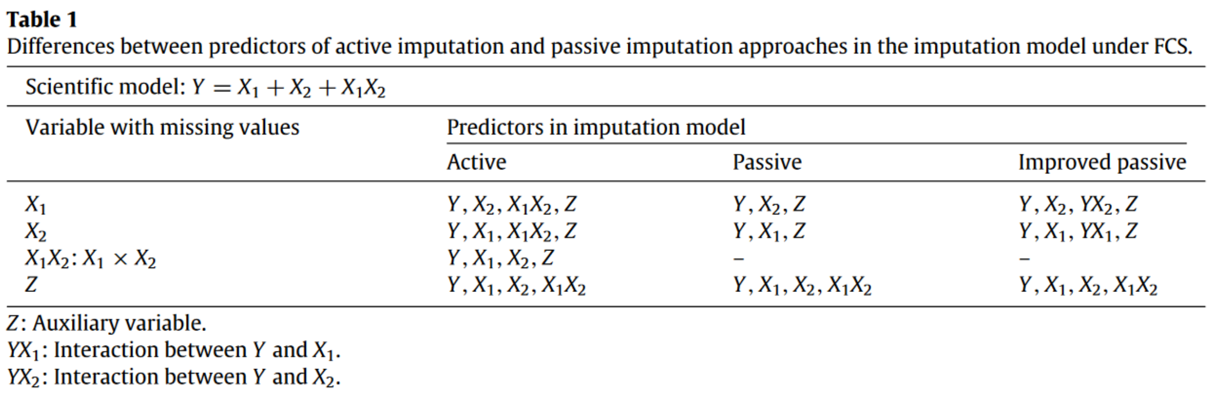


Table 1
Differences between predictors of active imputation and passive imputation approaches in the imputation model under FCS.

Scientific model: $Y = X_1 + X_2 + X_1X_2$

Variable with missing values	Predictors in imputation model		
	Active	Passive	Improved passive
X_1	Y, X_2, X_1X_2, Z	Y, X_2, Z	Y, X_2, YX_2, Z
X_2	Y, X_1, X_1X_2, Z	Y, X_1, Z	Y, X_1, YX_1, Z
X_1X_2: $X_1 \times X_2$	Y, X_1, X_2, Z	–	–
Z	Y, X_1, X_2, X_1X_2	Y, X_1, X_2, X_1X_2	Y, X_1, X_2, X_1X_2

Z: Auxiliary variable.
YX_1: Interaction between Y and X_1.
YX_2: Interaction between Y and X_2.

Figure 1: Mitani et al. (2015)

- Our new analysis model includes an interaction effect between `treatment` and `skin`

$$\log E(Y) = \beta_0 + \beta_1 \text{treatment} + \beta_2 \text{age} + \beta_3 \text{gender} + \beta_4 \text{exposure} + \beta_5 \text{skin} + \beta_6 \text{treatment} \times \text{skin}$$

- `skin` is missing in $\approx 20\%$ of the individuals
- Using the mice package, we will apply active, passive, and improved passive imputation approaches

Active imputation

```
# generate the interaction term
skin_mar$int <-
  as.numeric(as.character(skin_mar$treatment)) * as.numeric(as.character(skin_mar$skin))
head(skin_mar)
```

```
##       ID center age skin gender exposure treatment Y int
## 1 100034      1  NA    1      1        4         0 0   0
## 2 100045      1  68    1      0        2         0 0   0
## 3 100056      1  58    1      0        7         0 1   0
## 4 100067      1  53    1      1        3         0 0   0
## 5 100102      1  55    0      0        2         0 0   0
## 6 100113      1  59 <NA>      1       10         0 0  NA
```

```
### MICE Active

# select the variables to include in the imputation model
toimp <- skin_mar[c("center", "age", "skin", "gender", "exposure", "treatment", "Y", "int")]

set.seed(100) # set seed for reproducibility

# impute using fcs
# since the interaction term is also binary, we will use logistic regression to predict it
active.out <- mice(toimp, m = 5,
                   method = c("", "norm", "logreg", "", "", "", "", "logreg"),
                   print=FALSE)

# build analysis model
fit.active <- with(data = active.out,
                   glm(Y ~ age + treatment + gender + exposure + skin + int,
                                       family = poisson("log")))

# summarize results
testEstimates(as.mitml.result(fit.active))
```

```
##
## Call:
##
## testEstimates(model = as.mitml.result(fit.active))
##
## Final parameter estimates and inferences obtained from 5 imputed data sets.
##
##              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
## (Intercept)    -3.152     0.538    -5.853 1.128e+01     0.000     1.472     0.652
## age             0.016     0.008     2.110 1.194e+01     0.057     1.374     0.635
## treatment1     -0.324     0.246    -1.318 7.381e+00     0.227     2.790     0.787
## gender1         0.472     0.118     3.989 1.300e+05     0.000     0.006     0.006
## exposure        0.128     0.011    11.271 9.937e+00     0.000     1.736     0.691
## skin1          -0.171     0.195    -0.878 1.414e+01     0.395     1.136     0.586
## int             0.756     0.404     1.872 6.075e+00     0.110     4.303     0.853
##
## Unadjusted hypothesis test as appropriate in larger samples.
```

Passive imputation

```
###  MICE Passive

# Dry run to get meth and pred
ini <- mice(toimp, max = 0, print = FALSE)

# Save the methods and specify to passively impute the interactions
ini$method
```

```
##    center       age      skin    gender  exposure treatment         Y       int
##        ""     "pmm"  "logreg"        ""        ""        ""        ""     "pmm"
```

```
meth <- ini$method
meth["age"] <- "norm"
meth["skin"] <- "logreg"
meth["int"] <- "~I(treatment*skin)"
meth
```

```
##                center                   age                  skin
##                    ""                "norm"              "logreg"
##                gender              exposure             treatment
##                    ""                    ""                    ""
##                     Y                   int
##                    "" "~I(treatment*skin)"
```

```
# Remove interactions from predicting main effects
ini$predictorMatrix
```

```
##           center age skin gender exposure treatment Y int
## center         0   1    1      1        1         1 1   1
## age            1   0    1      1        1         1 1   1
## skin           1   1    0      1        1         1 1   1
## gender         1   1    1      0        1         1 1   1
## exposure       1   1    1      1        0         1 1   1
## treatment      1   1    1      1        1         0 1   1
## Y              1   1    1      1        1         1 0   1
## int            1   1    1      1        1         1 1   0
```

```
pred <- ini$predictorMatrix
pred["skin", "int"] <- 0
pred[c("skin", "age", "int"),]
```

```
##      center age skin gender exposure treatment Y int
## skin      1   1    0      1        1         1 1   0
## age       1   0    1      1        1         1 1   1
## int       1   1    1      1        1         1 1   0
```

```
set.seed(100) # set seed for reproducibility

# Impute using fcs
passive.out <- mice(toimp, m = 5, method = meth, pred = pred, print = FALSE)

# build analysis model
fit.passive <- with(data = passive.out,
                    glm(Y ~ age + treatment + gender + exposure + skin + int,
                        family = poisson("log")))

# summarize results
testEstimates(as.mitml.result(fit.passive))
```

```
##
## Call:
##
## testEstimates(model = as.mitml.result(fit.passive))
##
## Final parameter estimates and inferences obtained from 5 imputed data sets.
##
##             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
## (Intercept)   -2.635     0.442    -5.967 3.084e+02     0.000     0.129     0.120
## age            0.008     0.007     1.174 2.469e+02     0.241     0.146     0.134
## treatment1    -0.514     0.184    -2.792 1.424e+07     0.005     0.001     0.001
## gender1        0.251     0.132     1.895 7.613e+06     0.058     0.001     0.001
## exposure       0.133     0.011    12.470 5.134e+05     0.000     0.003     0.003
## skin1         -0.155     0.170    -0.911 7.984e+06     0.363     0.001     0.001
## int            1.079     0.243     4.445 2.663e+07     0.000     0.000     0.000
##
## Unadjusted hypothesis test as appropriate in larger samples.
```

Improved passive imputation

```
###  MICE Improved Passive

# generate the interaction term between the outcome and each of the main effects
skin_mar$intYtrt <- skin_mar$Y * skin_mar$treatment
skin_mar$intYskin <- skin_mar$Y * skin_mar$skin
head(skin_mar)
```

```
##       ID center age skin gender exposure treatment Y int intYtrt intYskin
## 1 100034      1  NA    1      1        4         0 0   0      NA       NA
## 2 100045      1  68    1      0        2         0 0   0      NA       NA
## 3 100056      1  58    1      0        7         0 1   0      NA       NA
## 4 100067      1  53    1      1        3         0 0   0      NA       NA
## 5 100102      1  55    0      0        2         0 0   0      NA       NA
## 6 100113      1  59 <NA>      1       10         0 0  NA      NA       NA
```

```
# select the variables to include in the imputation model
toimp <- skin_mar[c("center", "age", "skin", "gender", "exposure", "treatment",
                    "Y", "int", "intYtrt", "intYskin")]
```

```
# Dry run to get meth and pred
ini <- mice(toimp, max = 0, print = FALSE)

# Save the methods and specify to passively impute the interactions
ini$method
```

```
##    center       age      skin    gender  exposure treatment         Y       int
##        ""     "pmm"  "logreg"        ""        ""        ""        ""     "pmm"
##   intYtrt  intYskin
##        ""        ""
```

```
meth <- ini$method
meth["age"] <- "norm"
meth["skin"] <- "logreg"
meth["int"] <- "~I(treatment*skin)"
meth["intYskin"] <- "~I(Y*skin)"
meth
```

```
##                center                    age                   skin
##                    ""                 "norm"               "logreg"
##                gender               exposure              treatment
##                    ""                     ""                     ""
##                     Y                    int                intYtrt
##                    "" "~I(treatment*skin)"                     ""
##              intYskin
##          "~I(Y*skin)"
```

```
# Remove interactions from predicting main effects
ini$predictorMatrix
```

```
##           center age skin gender exposure treatment Y int intYtrt intYskin
## center         0   1    1      1        1         1 1   1       0        0
## age            1   0    1      1        1         1 1   1       0        0
## skin           1   1    0      1        1         1 1   1       0        0
## gender         1   1    1      0        1         1 1   1       0        0
## exposure       1   1    1      1        0         1 1   1       0        0
## treatment      1   1    1      1        1         0 1   1       0        0
## Y              1   1    1      1        1         1 0   1       0        0
## int            1   1    1      1        1         1 1   0       0        0
## intYtrt        0   0    0      0        0         0 0   0       0        0
## intYskin       0   0    0      0        0         0 0   0       0        0
```

```
pred <- ini$predictorMatrix
pred[c("skin"), "int"] <- 0
pred[c("skin"), "intYskin"] <- 0
pred["age", c("intYtrt", "intYskin")] <- 0
pred[c("skin", "age", "int", "intYtrt", "intYskin"),]
```

```
##          center age skin gender exposure treatment Y int intYtrt intYskin
## skin          1   1    0      1        1         1 1   0       0        0
## age           1   0    1      1        1         1 1   1       0        0
## int            1   1    1      1        1         1 1   0       0         0
## intYtrt        0   0    0      0        0         0 0   0       0         0
## intYskin       0   0    0      0        0         0 0   0       0         0
```

```r
set.seed(100) # set seed for reproducibility

# Impute using fcs
imp.passive.out <- mice(toimp, m = 5, method = meth, pred = pred, print = FALSE)

# build analysis model
fit.imp.passive <- with(data = imp.passive.out,
                        glm(Y ~ age + treatment + gender + exposure + skin + int,
                            family = poisson("log")))

# summarize results
testEstimates(as.mitml.result(fit.imp.passive))
```

```
##
## Call:
##
## testEstimates(model = as.mitml.result(fit.imp.passive))
##
## Final parameter estimates and inferences obtained from 5 imputed data sets.
##
##              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
## (Intercept)    -2.683     0.609    -4.403 1.399e+01     0.001     1.149     0.589
## age             0.009     0.009     0.907 1.246e+01     0.381     1.307     0.623
## treatment1     -0.514     0.184    -2.797 1.591e+08     0.005     0.000     0.000
## gender1         0.252     0.132     1.901 2.775e+06     0.057     0.001     0.001
## exposure        0.133     0.011    12.371 1.481e+04     0.000     0.017     0.017
## skin1          -0.157     0.170    -0.923 5.248e+06     0.356     0.001     0.001
## int             1.078     0.243     4.438 5.657e+07     0.000     0.000     0.000
##
## Unadjusted hypothesis test as appropriate in larger samples.
```

Conclusion on MI

- Although many packages exist to perform MI, the user is still responsible for navigating through various choices
- Challenges are
  - selecting the imputation method (JM vs FCS)
  - choosing a package
  - building the imputation model (choosing which variables to include, how to impute each variable for FCS)
  - selecting the number of imputations
  - dealing with variables of mixed type, especially nominal categorical variables such as race
  - dealing with derived variables (higher order terms, interaction terms, etc.)
- Sensitivity analysis is important
- Involve a statistician with expertise in missing data