EP16: Missing Values in Clinical Research: Multiple Imputation

8. Analysis of Imputed Data, Pooling & More

Nicole Erler

Department of Biostatistics, Erasmus Medical Center

n.erler@erasmusmc.nl

Analysing Imputed Data

Once we have confirmed that our imputation was successful, we can move on to the **analysis of the imputed data**.

For example a logistic regression model:

```
glm(DM ~ age + gender + hypchol + BMI + smoke + alc, family = binomial())
```

Analysing Imputed Data

Once we have confirmed that our imputation was successful, we can move on to the **analysis of the imputed data**.

For example a logistic regression model:

```
glm(DM ~ age + gender + hypchol + BMI + smoke + alc, family = binomial())
```

To fit the model on each of the imputed datasets:

- no need to extract the data from the `mids` object
- instead: can use `with()`

```
mod1 <- with(imp4, glm(DM ~ age + gender + hypchol + BMI + smoke + alc,
                       family = binomial()))
```

Analysing Imputed Data

`mod1` is an object of class `mira` (Multiply Imputed Repeated Analyses).

Analysing Imputed Data

`mod1` is an object of class `mira` (Multiply Imputed Repeated Analyses).

The `mira` object has elements

- **call:** the current call
- **call1:** call that created the `mids` object
- **nmis:** vector with number of missing values per variable
- **analyses:** a list of the separate analyses

Pooling the Results

Pooled results can be obtained using `pool()` and its summary.

```
pool(mod1)$pooled
```

```
##            term m estimate     ubar        b        t dfcom     df    riv lambda    fmi
## 1   (Intercept) 5  -7.5150 1.58e-01 1.52e-03 1.60e-01  2472 2262.4 0.0115 0.0114 0.0123
## 2           age 5   0.0574 1.84e-05 1.89e-07 1.86e-05  2472 2237.6 0.0123 0.0122 0.0131
## 3  genderfemale 5  -0.3800 1.56e-02 2.09e-04 1.59e-02  2472 2109.5 0.0161 0.0158 0.0168
## 4    hypcholyes 5  -0.0204 3.17e-02 1.72e-03 3.38e-02  2472  732.2 0.0651 0.0611 0.0637
## 5           BMI 5   0.1048 8.18e-05 1.37e-06 8.34e-05  2472 1961.5 0.0201 0.0197 0.0207
## 6       smoke.L 5   0.0265 1.32e-02 1.61e-04 1.34e-02  2472 2160.9 0.0146 0.0144 0.0153
## 7       smoke.Q 5  -0.0681 1.31e-02 1.20e-04 1.32e-02  2472 2277.2 0.0110 0.0109 0.0118
## 8         alc.L 5  -0.4081 2.07e-02 9.66e-03 3.23e-02  2472   30.5 0.5595 0.3588 0.3971
## 9         alc.Q 5   0.1636 2.30e-02 1.80e-02 4.47e-02  2472   16.8 0.9414 0.4849 0.5370
## 10        alc.C 5  -0.0304 2.31e-02 4.35e-03 2.83e-02  2472  110.9 0.2264 0.1846 0.1989
## 11        alc^4 5   0.0284 2.68e-02 6.25e-03 3.43e-02  2472   80.3 0.2797 0.2185 0.2373
```

Pooling the Results

```
summary(pool(mod1), conf.int = TRUE)
```

```
##            term estimate std.error statistic     df p.value   2.5 %  97.5 %
## 1   (Intercept)  -7.5150   0.39961   -18.806 2262.4  0.0000 -8.2986 -6.7313
## 2           age   0.0574   0.00432    13.304 2237.6  0.0000  0.0490  0.0659
## 3  genderfemale  -0.3800   0.12600    -3.016 2109.5  0.0026 -0.6271 -0.1329
## 4    hypcholyes  -0.0204   0.18380    -0.111  732.2  0.9118 -0.3812  0.3405
## 5           BMI   0.1048   0.00913    11.481 1961.5  0.0000  0.0869  0.1228
## 6       smoke.L   0.0265   0.11591     0.229 2160.9  0.8190 -0.2008  0.2538
## 7       smoke.Q  -0.0681   0.11503    -0.592 2277.2  0.5542 -0.2936  0.1575
## 8         alc.L  -0.4081   0.17974    -2.270   30.5  0.0304 -0.7749 -0.0412
## 9         alc.Q   0.1636   0.21134     0.774   16.8  0.4497 -0.2828  0.6099
## 10        alc.C  -0.0304   0.16820    -0.181  110.9  0.8568 -0.3637  0.3029
## 11        alc^4   0.0284   0.18525     0.153   80.3  0.8784 -0.3402  0.3971
```

Pooling the Results

Pooling with `mice::pool()` is available for most types of models.

It extracts the model coefficients and variance-covariance matrices using `tidy()` from the package **broom**. Hence, pooling using the `pool()` function from **mice** only works for models of classes for which a method `tidy()` exists.

An alternative is offered by the package **mitools** and the function `MIcombine()`.

Functions for Pooled Results

mice provides some functions for evaluating model fit or model comparison.

Functions for Pooled Results

mice provides some functions for evaluating model fit or model comparison.

`pool.r.squared()` calculates the pooled (adjusted) R^2:

```
mod2 <- with(imp4, lm(SBP ~ DM + age + hypten))
pool.r.squared(mod2, adjusted = TRUE)
```

```
##               est     lo 95     hi 95 fmi
## adj R^2 0.3265363 0.2957747 0.3573434 NaN
```

The argument `adjusted` specifies whether the adjusted R^2 or the standard R^2 is returned.

Functions for Pooled Results

To compare nested models:

- `D1()`: multivariate Wald test
- `D3()`: likelihood-ratio test statistic

To pool test statistics when no variance-covariance matrix is available:

- `D2()`: Combining test statistics

For details, see Van Buuren (2012), Section 5.3 and Schafer (1997).

Functions for Pooled Results

Example: To test if `smoke` has a relevant contribution to the model for `DM` from above we re-fit the model without `smoke` and compare the two models:

```
mod3 <- with(imp4, glm(DM ~ age + gender + hypchol + BMI + alc,
                       family = "binomial"))
# likelihood ratio test
D3(mod1, mod3)
```

```
##    test statistic df1      df2 df.com   p.value         riv
##  1 ~~ 2    0.2559   2 65004.54    Inf 0.7742202 0.005929978
```

`anova()` allows comparison of multiple nested models

Functions for Pooled Results

The package **miceadds** extends **mice**, for example with the following functionality:

Combine χ^2 or F statistics from multiply imputed data:

```
miceadds::micombine.chisquare(dk, df, ...)
miceadds::micombine.F(values, df1, ...)
```

Functions for Pooled Results

The package **miceadds** extends **mice**, for example with the following functionality:

Combine χ^2 or F statistics from multiply imputed data:

```
miceadds::micombine.chisquare(dk, df, ...)
miceadds::micombine.F(values, df1, ...)
```

Calculate correlation or covariance of imputed data:

```
miceadds::micombine.cor(mi.res, ...)
miceadds::micombine.cov(mi.res, ...)
```

Functions for Pooled Results

```
# fit chi-square test on each imputed dataset
chisq_mira <- with(imp4, chisq.test(table(educ, smoke)))
```

Functions for Pooled Results

```r
# fit chi-square test on each imputed dataset
chisq_mira <- with(imp4, chisq.test(table(educ, smoke)))
```

```r
# extract degrees of freedom
dfs <- chisq_mira$analyses[[1]]$parameter

# extract test statistic
stat <- sapply(chisq_mira$analyses, "[[",  'statistic')
```

Functions for Pooled Results

```r
# fit chi-square test on each imputed dataset
chisq_mira <- with(imp4, chisq.test(table(educ, smoke)))
```

```r
# extract degrees of freedom
dfs <- chisq_mira$analyses[[1]]$parameter

# extract test statistic
stat <- sapply(chisq_mira$analyses, "[[",  'statistic')
```

```r
# pool the tests
miceadds::micombine.chisquare(dk = stat, df = dfs)
```

```
## Combination of Chi Square Statistics for Multiply Imputed Data
## Using 5 Imputed Data Sets
## F(8, 336475.37)=15.615     p=0
```

Functions for Pooled Results

```
miceadds::micombine.cor(imp4, variables = c('weight', 'BMI', 'creat'))
```

```
##   variable1 variable2     r    rse fisher_r fisher_rse   fmi    t       p lower95 upper95
## 1       BMI     creat 0.039 0.0217    0.039      0.022 0.151  1.8 7.0e-02 -0.0032   0.082
## 2       BMI    weight 0.865 0.0051    1.311      0.020 0.019 64.7 0.0e+00  0.8542   0.874
## 3     creat    weight 0.140 0.0209    0.141      0.021 0.115  6.6 3.3e-11  0.0992   0.181
## 4     creat       BMI 0.039 0.0217    0.039      0.022 0.151  1.8 7.0e-02 -0.0032   0.082
## 5    weight       BMI 0.865 0.0051    1.311      0.020 0.019 64.7 0.0e+00  0.8542   0.874
## 6    weight     creat 0.140 0.0209    0.141      0.021 0.115  6.6 3.3e-11  0.0992   0.181
```

Extract Imputed Data

The function `complete()` allows us to **extract the imputed data** from a `mids` object:

```
mice::complete(data, action = 1, include = FALSE)
```

- `data`: the `mids` object
- `action`:
  - `1`, …, `m` (single imputed dataset)
  - `"long"`: long format (imputed data stacked vertically)
  - `"broad"`: wide format (imputed data combined horizontally; ordered by imputation)
  - `"repeated"`: (like `"broad"`, but ordered by variable)
- `include`: include the original data?
  (if `action` is `"long"`, `"broad"` or `"repeated"`)

Combining `mids` objects

To **increase the number of imputed datasets** without re-doing the initial *m* imputations, a second set of imputations can be done and the two `mids` objects combined using `ibind()`.

```
# same syntax as before, but different seed
imp4b <- update(imp4, seed = 456)
```

```
imp4combi <- ibind(imp4, imp4b) # combine
```

```
# check the new number of imputed datasets:
imp4combi$m
```

```
## [1] 10
```

Adding variables to `mids` objects

The function `cbind.mids()` allows us to **add columns** to a `mids` object. The extra columns can either be a `data.frame`, `matrix`, `vector` or `factor` or another `mids` object.

For example data columns that should be part of the imputed data for completeness, but are not needed in the imputation.

```
# "otherdata" is a data.frame
impextra <- mice:::cbind.mids(x = imp4, y = otherdata)
```

Note:

`cbind()` just adds columns to the data, you need to make sure they are **sorted correctly** so that the rows of the new data are from the same subjects as the corresponding rows in the imputed data.

References

Schafer, Joseph L. 1997. *Analysis of Incomplete Multivariate Data*. CRC press.

Van Buuren, Stef. 2012. *Flexible Imputation of Missing Data*. Chapman & Hall/Crc Interdisciplinary Statistics. Taylor & Francis. https://stefvanbuuren.name/fimd/.