Using dejaVu for imputation of an existing study dataset

Jonathan W. Bartlett*

*Department of Mathematical Sciences, University of Bath, UK

April 27, 2021

1 Introduction

The `dejaVu` package performs multiple imputation for recurrent event data sets. The package can be used to perform multiple imputation of an existing study dataset where some patients dropped out. Imputation can be performed either assuming dropout is at random (missing at random) or assuming a specific non-random dropout mechanism (missing not at random).

The package can also be used to simulate recurrent event datasets, in order to evaluate the impact of dropout and the properties of multiple imputation based analyses. Please see the other accompanying vignette for details on this, and for further details on more advanced aspects of the package.

2 Loading the data

We demonstrate how `dejaVu` can be used to perform multiple imputation of an existing simulated dataset. The (simulated) dataset is called `simData`, and is loaded when `dejaVu` is loaded. We first load the package:

```
library(dejaVu)
```

Note: due to namespace conflicts between libraries `MASS` and `trafficlight`, `dejaVu` does will not work if `trafficlight` is loaded.

First we load and summarise the simulated dataset:

```
summary(simData)
```

```
##        z                y             fupTime
##  Min.   :0.000   Min.   :0.000   Min.   :0.002774
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.787716
##  Median :1.000   Median :0.000   Median :1.025081
##  Mean   :0.506   Mean   :0.544   Mean   :0.877520
##  3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.061419
##  Max.   :1.000   Max.   :5.000   Max.   :1.099961
```

Before doing any imputation, we fit a negative binomial model to the observed data:

```
#analyse observed data
library(MASS)
obsMod <- glm.nb(y~z+offset(log(fupTime)), data=simData)
summary(obsMod)
```

```
##
## Call:
## glm.nb(formula = y ~ z + offset(log(fupTime)), data = simData,
##     init.theta = 9.962382938, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3545  -0.8935  -0.6827   0.7467   2.6796
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.13515    0.07616  -1.775    0.076 .
## z           -0.83582    0.13415  -6.230 4.66e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(9.9624) family taken to be 1)
##
##     Null deviance: 507.85  on 499  degrees of freedom
## Residual deviance: 466.15  on 498  degrees of freedom
## AIC: 927.4
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  10.0
##           Std. Err.:  11.6
##
##  2 x log-likelihood:  -921.397
```

The estimated log rate ratio for active (`z=1`) vs. control (`z=0`) is -0.84, and is highly statisticaly significant. This observed data (likelihood based) analysis is valid if the dropout mechanism is at random. That is, if the probability of dropout at a given time is not related to the post dropout outcome conditional on the pre dropout outcome.

3 Data preparation

Before we can perform multiple imputation, we must import the data into the form `dejaVu` requires. The first step is to create a `DejaData` object containing the covariates we will use. Here the only covariate will be the patient's randomised treatment group:

```
covar.df <- data.frame(id=1:dim(simData)[1], z=simData$z)
dejaData <- MakeDejaData(covar.df, arm="z", Id="id")
```

The core data structure used by `dejaVu` package stores the actual event times for each patient, rather than just the total number of events observed. In general for certain types of model, the actual times of events may be required, rather than simply the number. However, for the negative binomial model we use here, which only uses baseline covariates (in fact, just the patient's randomised treatment group), the actual times of patients' events do not affect estimates or inferences. As such, we will use the package's helper function `expandEventCount`, which takes the number of events each patient has had observed, and their follow-up time, and creates pseudo/fake event times for each patient. These times are evenly spaced between

time zero and each patient's follow-up time. As noted above, this choice of even spacing has no impact for the models we use here:

```
eventTimes <- expandEventCount(count = simData$y, time = simData$fupTime)

## Warning in expandEventCount(count = simData$y, time = simData$fupTime):  Synthesizing fake
event times from event count data.
```

Using `expandEventCount` always issues a warning when used, in order to warn us that we should only use the fake event times created so long as our analysis and imputation method is not affected by the actual placement of the times.

Next we use the `ImportSim` function to create the data object which `dejaVu` can use:

```
obsData <-
  ImportSim(
  dejaData,
  event.times = eventTimes,
  status = "dropout",
  study.time = 1,
  censored.time = simData$fupTime,
  allow.beyond.study = T
  )
```

The `status = "dropout"` argument value specifies that the data we are importing is incomplete, in the sense that not all patients completed their planned follow-up. The `study.time` argument specifies the planned length of follow-up for the study, which for this dataset is 1 year. The `censored.time` argument is a vector containing the length of actual follow-up for the patients. The final argument, `allow.beyond.study`, is set to true here because, as would often be the case in a real study, although the planned follow-up was 1 year, some patients' actual follow-up time slightly exceeded 1 year.

4 Fitting model to observed data

Next we ask `dejaVu` to fit a negative binomal model to the observed data. This is exactly the same model as we fitted earlier, but now the returned object is of type `SingleSimFit`, which can then be passed to the package's imputation function:

```
obsFit <- Simfit(obsData,equal.dispersion=TRUE)
```

The `equal.dispersion` argument specifies whether the variance of the gamma random effects in the negative binomial model should be assumed to have the same variance in the treatment groups, or if they should be allowed to differ.

5 Imputation

We are now ready to perform multiple imputation:

```
imputed.data.sets <- Impute(fit = obsFit,impute.mechanism = weighted_j2r(trt.weight=0),
                            N=10)
```

The first argument passed is the fitted model object created earlier. The third argument specifies the number of imputations to generate. The second argument specifies the imputation method to use. At

present there is one imputation method included within `dejaVu`, the weighted jump to reference method. For patients in the placebo arm, this imputes the number of events in a patient's incomplete follow-up period using the MAR estimated placebo rate from the observed data model fit. For patients in the active arm, imputation is performed using the MAR estimated active rate if `trt.weight` is set to 1. Conversely, if `trt.weight` is set to 0, active patients' data is imputed using the MAR estimated placebo rate, which may be appropriate in situations where the active patients switch to placebo when they dropout. In the above code, we have imputed making this assumption.

`dejaVu` imputes for every patient whose observed follow-up is less than the specified study duration (here 1 year). Thus no imputation is performed for patients who have observed follow-up equal or greater than the study's planned follow-up length, as specified in the `allow.beyond.study` argument in `ImportSim`.

6 Analysing the imputed datasets

We can now analyse the imputed datasets, using the `Simfit` function:

```
fitted <- Simfit(imputed.data.sets,family="negbin")
summary(fitted)

## Summary for imputed data sets
## Treatment Effect: 0.5095203
## SE (log) treatment effect: 0.1669642
## Degrees of freedom: 38.55875
## p-value: 0.0002474157
## Adjusted d.o.f: 33.51771
## Adjusted p-value: 0.0002954297
## Average dispersion: 0.2041964
## Number of subjects dropped out per arm: 247 253
```

The `Simfit` function has fitted the negative binomial regression model to each imputed dataset, saved the estimated log rate ratios and corresponding standard errors, and combined these using Rubin's rules. By default, the same covariates are used as used in the imputation model. If desired, additional covariates can be added.