Gamma imputation tutorial (Jackson 2014)

Nikolas S. Burkoff[**], Paul Metcalfe[*], Jonathan Bartlett[*] and David Ruau[*]

[*]AstraZeneca, B&I, Advanced Analytics Centre, UK
[**]Tessella, 26 The Quadrant, Abingdon Science Park, Abingdon, OX14 3YS, UK

October 27, 2024

1 Introduction

In this vignette the `InformativeCensoring` R library is used to perform the multiple imputation (MI) approach proposed by Jackson et al. [1]. Throughout this package the method is referred to as the γ-imputation method.

2 Theory

The γ-imputation method allows the independent censoring assumption in the Cox proportional hazard model to be relaxed, allowing sensitivity analyses to be performed. This section describes the method and the reader is referred to [1] for further details, including possible extensions of the algorithm not implemented in this package.

Consider a time to event data set where subject i has time to event T_i and is censored at time C_i. Each subject has an observed time $Y_i = \min(C_i, T_i)$ and event indicator δ_i where $\delta_i = 1$ when $T_i < C_i$ and $\delta_i = 0$ otherwise. The MI procedure generates M imputed datasets where subjects who were censored now have an imputed time $T_i^m \geq Y_i$ and a new event indicator δ_i^m. The data $\{T_i^m, \delta_i^m\}$ are imputed under the assumption that at the point of censoring the hazard function *jumps* by a (potentially subject specific) constant[1] denoted by γ_i. With $\gamma_i \neq 0$, the independent censoring assumption has been relaxed, and by varying the size and magnitude of γ_i sensitivity analyses can be performed.

Once the M data sets have been imputed, standard time to event statistical analyses can be applied to each data set and the results combined using Rubin's rules [3] to produce a single overall estimate for parameters of interest.

2.1 The imputation model

The approach proposed by [1] assumes that the hazard for failure, given that censoring has not yet occurred, is equal to

$$h(t|C_i > t, Z_i, S_i) = h_{(0,S_i)}(t)\exp(\beta Z_i)$$

where Z_i are time independent covariates for subject i, S_i is the strata for subject i, $h_{(0,S_i)}(t)$ is the baseline hazard function for the strata defined by S_i and β are the regression coefficients. This model can be fitted to the observed data using partial likelihood in the standard way[2].

After censoring has occurred it is assumed

$$h(t|C_i < t, Z_i, S_i) = h_{(0,S_i)}(t)\exp(\beta Z_i)\exp(\gamma_i),$$

[1] The general framework allows γ to be general function, see [1] for further details.

[2] The package fits Cox proportional hazard models, but in principle other models could be used.

therefore if $\gamma_i > 0$ there is an elevated risk for failure after censoring and if $\gamma_i < 0$ there is a decreased risk after censoring. By allowing subject specific γ_i it is possible to take into account the reason for subject censoring into the imputation procedure, i.e. use different values of γ_i dependent on the reason for censoring. An example of this could be if a subject was censored because they left the country, which might be assumed to be unrelated to the disease, and therefore one may be willing to assume $\gamma_i = 0$.

An important point to note is that the parameter γ_i represents the change in log hazard for failure following censoring, conditional on the covariates in the imputation model. The interpretation of γ_i is thus as a conditional log hazard ratio, adjusted for whatever covariates Z_i and strata S_i have already been accounted for.

2.2 Number of imputations

In multiple imputation the user must specify the number of imputations to be generated. The validity of results do not depend on how many imputations are used. However, if a small number of imputations are used, the results may contain a non-negliglble amount of simulation (Monte-Carlo) variability. The magnitude of this variability can be assessed by re-running the analysis a number of times (without resetting the seed) with a given number of imputations. If the variability between the results of the separate runs is appreciable, then it is advised that a larger number of imputations be used. The judgment as to what level of simulation error is acceptable can usually be made by ensuring that sufficient imputations are used such that if the analysis were to be re-run with a different seed, the results would not differ to the precision with which they are to be reported.

2.3 Bootstrapping

In order to make the imputations so called ‘proper’, to generate each imputed dataset the Cox model is fitted to a bootstrap sample of the original dataset. Strata can be passed in order to perform this sampling within strata. The usual recommendation (for non-parametric bootstrap) is that the bootstrap sampling scheme should match that used to sample the original data.

2.4 Sampling event times

Given a subject that was censored at time C_i (so $\delta_i = 0$), in strata S_i and with covariates Z_i an imputed failure time ($> C_i$) is sampled from the model

$$h(t|C_i, Z_i, S_i) = \hat{h}_{j(0,S_i)}(t)\exp(\hat{\beta}_j Z_i + \gamma_i)$$

where $\hat{h}_{j(0,S_i)}$ and $\hat{\beta}_j$ are from the model fit associated with bootstrapped sample j. The precise details of the sampling procedure used can be found in [1]. In the following we assume that a subject has a maximum possible follow up period of F_i. If the imputed time is $> F_i$, then a time of F_i and event indicator $\delta_i = 0$ are imputed. Due to the nature of the sampling procedure used, imputed event times can only take the value of one of the observed event times, so if the follow up time is unknown (or until an event occurs) we recommend setting F_i as the maximum time of any subject on the trial.

2.5 Analysing the imputed data sets

Conventional Cox models can be fitted to each of the imputed data sets and can generate M estimates for a regression coefficient of interest, say β: $\{\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_M\}$, together with M estimates of their variance $\{v_1, v_2, \ldots, v_M\}$. Using Rubin's rules [3] these values can be combined into a single estimate for β, its variance and statistical significance level:

- Estimate: $\hat{\beta} = \text{mean}(\hat{\beta}_k)$
- Variance: $\hat{v} = U + (1 + 1/M)B$ where $U = \text{mean}(v_k)$ and $B = \text{var}(\hat{\beta}_k)$
- Test statistic: $t = \hat{\beta}/\sqrt{\hat{v}}$ follows a t-distribution with $\nu = (M-1)\left(1 + \frac{U}{(1+1/M)B}\right)^2$ degrees of freedom (see [4]).

3 Using the package

In this section we describe how to use the package, step by step. We first load the package:

```
library(InformativeCensoring)
```

Note, messages about masking such as 'The following object is masked from 'package:survival': cox.zph' are to be expected.

3.1 Data

In this example we use the `nwtco` data set from the `survival` package. Note that the following analyses are purely illustrative, and are intended only to demonstrate the usage of the package. We first load the data:

```
data(nwtco)

#only use first 500 subjects
nwtco <- nwtco[1:500,]
head(nwtco)

##   seqno instit histol stage study rel edrel age in.subcohort
## 1     1      2      2     1     3   0  6075  25        FALSE
## 2     2      1      1     2     3   0  4121  50        FALSE
## 3     3      2      2     1     3   0  6069   9        FALSE
## 4     4      2      1     4     3   0  6200  28         TRUE
## 5     5      2      2     2     3   0  1244  55        FALSE
## 6     6      1      1     2     3   0  2932  32        FALSE
```

3.2 Setting the seed

Since multiple imputation methods makes use of random resampling and random imputation, the results will change every time the (same) analysis is performed, unless the seed is set prior to analysis. Thus the seed should be set (to an arbitrary integer) prior to analysis, to ensure that results are reproducible.

```
#Set random number seed
set.seed(3957129)
```

3.3 Specifying γ_i

The function which performs the imputation has two arguments (`gamma.factor` and `gamma`) which determine whether imputation is performed for each subject and if it is to be performed, what the value of γ_i is. At least one of `gamma.factor` and `gamma` must be passed to the function.

In the simplest usage, the user supplies a single numeric value to the argument `gamma.factor` but nothing in the `gamma` argument. The function then imputes using $\gamma_i =$`gamma.factor`.

Alternatively, the user can supply a column name or vector of values (with length equal to the number of rows in the data frame) to the argument `gamma`, but not pass a value to the `gamma.factor` argument. In this case γ_i equals the corresponding value in the vector passed to `gamma`. If a subject's corresponding element of `gamma` is `NA`, no imputation will be performed for that subject. If a subject's corresponding element is 0, imputation will always be performed assuming non-informative censoring (conditional on the model covariates).

Lastly, the user may supply both the `gamma.factor` and `gamma` arguments. In this case γ_i is calculated according to

$$\gamma_i = gamma[i] \times gamma.factor$$

where *gamma*$[i]$ denotes the ith element of the argument `gamma`.

To illustrate, we now define a new column in the data frame, `basegamma`, as follows

```
nwtco$basegamma <- 1
nwtco$basegamma[nwtco$histol==2] <- NA
```

This definition will mean that those subjects with `histol==2` will not have imputation performed.

3.4 Generating imputations

The following code generates 10 imputed datasets using the `gammaImpute` function. Following this code we then describe each of the arguments.

```
#Create the m imputed data sets (a GammaImputedSet object)
imputed.data.sets <- gammaImpute(formula=Surv(edrel,rel)~histol + instit + strata(stage),
                                 data = nwtco, m=10,
                                 gamma="basegamma", gamma.factor=0.5, DCO.time=6209)
```

This function takes the following arguments:

- `formula`: the model formula to be used when fitting the models in order to produce baseline hazards and linear predictors. Only right-censored `Surv` objects can be used and `cluster` and `tt` terms cannot be included; however, `strata` terms can be used[3].
- `data`: a data frame to apply the multiple imputation method.
- `m`: the number of data sets to impute.
- `gamma`: either column name for the subjects' base `gamma` values (in quotes as above) or a vector of base `gamma` values, as described previously.
- `gamma.factor`: as described previously. Here we have used a value of 0.5, so that for those subjects who are being imputed (as dictated by `basegamma`), their log hazard following censoring is being increased by 0.5.
- `bootstrap.strata`: any strata to be used in the bootstrap sampling. This option is distinct from any `strata` argument used in model formula. If the argument is not used then standard resampling with replacement is used. See the `strata` argument for the `boot` function for further details.
- `DCO.time`: either column name containing the subject's data cutoff time (this is F_i in Section 2.4 and see this section for further details) or a vector of DCO.times for the subjects or a single number to be used as the DCO.time for all subjects.
- `parallel`, `ncpus`, `cl`: parallelism options, see `help(gammaImpute)` and the `parallel` package vignette for further details – note when using `parallel="multicore"` or `parallel="snow"` it is necessary to set the random number generator to type "L'Ecuyer-CMRG" using the command `RNGkind("L'Ecuyer-CMRG")` in order to ensure proper random number stream behaviour. A warning is output if this is not the case, see `parallel` package vignette for further details.

Additional arguments are passed into the Cox model fitting function `survival::coxph`. Note by default this function uses the Efron method for dealing with ties. If the Breslow method is required then use `ties="breslow"`[4].

[3] You cannot use `strata(W1) + strata(W2)` in a formula and should use `strata(W1,W2)` instead

[4] older versions of `survival` use `method="breslow"` instead.

Infinite γ: It is possible for γ_i to be infinite. If $\gamma = -\infty$ then the subject is censored at their `DCO.time`. If $\gamma_i = \infty$ (or large enough for R to view $\eta = \exp(-(\beta Z_i) - \gamma_i)$ as zero) then the package will cause imputed subjects to die instantly at their time of censoring. If you use the function multiple times with increasing values of γ_i for subjects, then at the point at which η becomes numerically indistinguishable from zero you may get a somewhat abrupt change in coefficient values.

3.5 Fitting models to imputed data

We now analyse the generate imputed datasets, and combine the results using Rubin's rules:

```
#Fit Cox models onto each imputed data set (a GammaStatList object)
imputed.fits <- ImputeStat(imputed.data.sets)

#Combine the results using Rubin's rules
summary(imputed.fits)

##                est        se        t        df     Pr(>|t|)      lo 95     hi 95
## histol 1.5329736 0.3033142 5.054078 300038587 4.324746e-07  0.9384887 2.1274585
## instit 0.1670072 0.3110995 0.536829 661395638 5.913857e-01 -0.4427366 0.7767511
```

By default, Cox models using the same formula as was used in the imputation are fitted to the imputed event times/indicators. It is possible to specify a different formula using the `formula` argument (only the right hand side of the formula is required). It is also possible to fit either Weibull or exponential models to the data by using the `method` argument. Additional arguments to `ImputeStat`, such as `ties="breslow"` are passed directly into the model fitting function `coxph` or `survreg`. The fitting procedure can be parallelized using the same `parallel`, `ncpus` and `cl` arguments as used in the `gammaImpute` function.

```
fits.weibull <- ImputeStat(imputed.data.sets,method="weibull",formula=~histol)
```

Caution is however advised in regards to fitting models that differ to that used to impute. In particular, in general the imputation model should not impose restrictions that the analysis model does not [2].

Lastly, should the user wish to see the estimated coefficients and variances from the fits to each imputed dataset, these can be viewed by:

```
imputed.fits$statistics

## $estimates
##         histol    instit
##  [1,] 1.534177 0.1680203
##  [2,] 1.534177 0.1680203
##  [3,] 1.534177 0.1680203
##  [4,] 1.534177 0.1680203
##  [5,] 1.534177 0.1680203
##  [6,] 1.534177 0.1680203
##  [7,] 1.522142 0.1578894
##  [8,] 1.534177 0.1680203
##  [9,] 1.534177 0.1680203
## [10,] 1.534177 0.1680203
##
## $vars
##           histol     instit
##  [1,] 0.09202746 0.09680869
##  [2,] 0.09202746 0.09680869
##  [3,] 0.09202746 0.09680869
##  [4,] 0.09202746 0.09680869
##  [5,] 0.09202746 0.09680869
##  [6,] 0.09202746 0.09680869
##  [7,] 0.09158853 0.09643801
##  [8,] 0.09202746 0.09680869
##  [9,] 0.09202746 0.09680869
## [10,] 0.09202746 0.09680869
```

3.6 Accessing individual imputed data sets and model fits

The `ExtractSingle` function can be used to extract out a single imputed data set. The `index` argument is an integer between 1 and M and allows the user to specify which imputed data set is to be extracted:

```
#for the third data set
imputed.data.set <- ExtractSingle(imputed.data.sets,index=3)

head(imputed.data.set$data)

##   seqno instit histol stage study rel edrel age in.subcohort basegamma
## 1     1      2      2     1     3   0  6075  25        FALSE        NA
## 2     2      1      1     2     3   0  4121  50        FALSE         1
## 3     3      2      2     1     3   0  6069   9        FALSE        NA
## 4     4      2      1     4     3   0  6200  28         TRUE         1
## 5     5      2      2     2     3   0  1244  55        FALSE        NA
## 6     6      1      1     2     3   0  2932  32        FALSE         1
##   internalDCO.time internal_gamma_val impute.time impute.event
## 1             6209                 NA        6075            0
## 2             6209                0.5        6209            0
## 3             6209                 NA        6069            0
## 4             6209                0.5        6209            0
## 5             6209                 NA        1244            0
## 6             6209                0.5        6209            0
```

Note the new columns which include `impute.time`, the event/censoring time for the imputed data set; `impute.event` the event indicator for the imputed data set; `internal_gamma_val` the value of γ used for each subject.

The `ExtractSingle` function can also be used to extract individual model fits and view residuals. As before the `index` argument is an integer between 1 and M allows the user to specify which model fit is to be extracted:

```
fifth.fit <- ExtractSingle(imputed.fits,index=5)

print(fifth.fit)

## Call:
## model.function(formula = formula, data = object$data, model = TRUE)
##
##          coef exp(coef) se(coef)     z        p
## histol 1.5342    4.6375   0.3034 5.057 4.25e-07
## instit 0.1680    1.1830   0.3111 0.540    0.589
##
## Likelihood ratio test=41.76  on 2 df, p=8.552e-10
## n= 500, number of events= 88

head(residuals(fifth.fit$model,type="schoenfeld"))
```

```
##          histol     instit
## 32   0.6213238  0.7095590
## 46   0.6340152 -0.2759473
## 47  -0.3548449 -0.2807959
## 90  -0.3561944 -0.2818638
## 121  0.6424457  0.7170601
## 124  0.6561889  0.7323994
```

Plotting Schoenfeld Residuals: The `survival` package includes the `cox.zph` function for testing the Cox proportional hazards assumption and plotting scaled Schoenfeld residuals. See `help(cox.zph)` and `help(plot.cox.zph)` for further details. The model fits from a γ-imputation procedure can also use the `cox.zph` function:

```
test.assump <- cox.zph(imputed.fits,index=3,transform="km")
print(test.assump)

##         chisq df     p
## histol   4.39  1 0.036
## instit   4.45  1 0.035
## GLOBAL   5.34  2 0.069

plot(test.assump[1],main="Scaled Schoenfeld Residuals for histol")
```

In particular, as noted by [1], the imputation model assuming informative censoring and the Cox model which is used to analyse the resulting imputed datasets cannot in general both be correctly specified. [1] suggest examining the Schoenfeld residuals both for the Cox model fitted to the observed data and from those after fitting Cox models to each imputed datasets to examine whether the proportional hazards assumption appears to be satisfied.

4 Performing sensitivity analysis

In this section we give an example showing how the package can be used to show how changing γ (specifically `gamma.factor`) affects the overall results. γ can be interpreted as the coefficient associated with a time dependent binary covariate taking the value 0 prior to censoring and 1 following being censored. As γ is increased, the coefficient associated with the covariate of censoring is increased, and one can examine how sensitive inferences are to varying γ.

Broadly speaking there are two approaches to conducting a sensitivity analysis in this context. The first is, in discussion with suitable subject matter experts, determine a range of plausible values for the sensitivity parameter γ. One then examines how the substantive results change over this range, to determine if the conclusions are robust across the defined range.

An alternative is a so called ‘tipping point analysis’. Here one increases γ until the substantive conclusions change. One can then judge whether the value of γ needed to ‘tip’ the results over are plausible or implausible. The judgement of what is thought plausible clearly requires input from clinical experts and trial specific context information.

4.1 Simulated example

To illustrate the principle, in the following we simulate a simple dataset, with a single binary covariate representing treatment allocation. For each individual we simulate a censoring time and an event time, where the hazard for the event depends on treatment allocation. Here we generate the censoring and event time entirely independently, so in fact the non-informative censoring assumption holds. Nonetheless, we use the simulated dataset to illustrate how sensitivity for the log hazard ratio comparing the treatment groups varies if we allow for informative censoring.

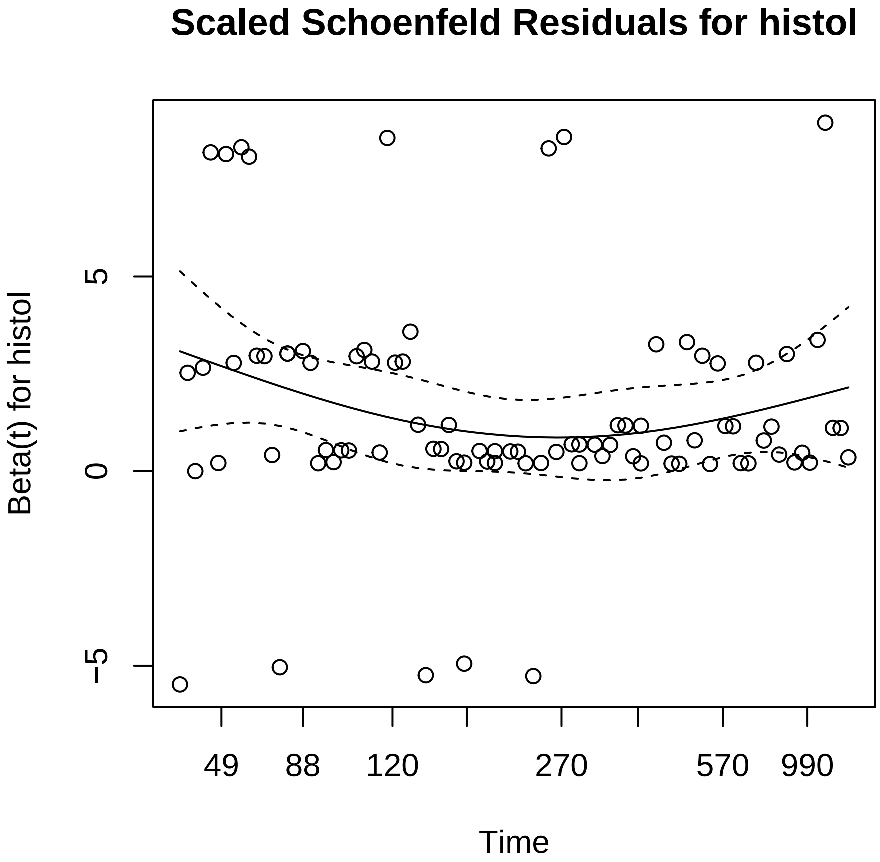

Figure 1: Scaled Schoenfeld residuals for histol for the third imputed data set and its model fit. See help(plot.cox.zph) for plotting options

First we generate the simulated dataset:

```
set.seed(6110)
#simulate a time to event dataset, with two treatment groups
#note that here we are simulating with non-informative censoring
N <- 1000
gamma.dataset <- data.frame(Id=1:N,
                            DCO.time=rep(3,N))
#Z is a treatment variable, assigned in a 1:1 ratio
gamma.dataset$Z <- c(rep(1,N*0.5),rep(0,N*0.5))

#generate censoring and event times
C <- rexp(n = N,rate = 0.3)
T <- rexp(n = N,rate=0.05*exp(gamma.dataset$Z))
gamma.dataset$Y <- pmin(T,C,3)
gamma.dataset$delta <- 1*((T < C) & (T <3))
gamma.dataset$Z <- factor(gamma.dataset$Z)
```

4.1.1 Analyses assuming noninformative censoring

Now we fit a Cox model to the observed data, with treatment as covariate. This is valid if censoring is noninformative, conditional on treatment group.

```
obsDataCox <- coxph(Surv(Y,delta) ~ Z, data=gamma.dataset)
summary(obsDataCox)

## Call:
## coxph(formula = Surv(Y, delta) ~ Z, data = gamma.dataset)
##
##   n= 1000, number of events= 163
##
##      coef exp(coef) se(coef)     z Pr(>|z|)
## Z1 0.9898    2.6907   0.1710 5.789 7.08e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##    exp(coef) exp(-coef) lower .95 upper .95
## Z1     2.691     0.3717     1.925     3.762
##
## Concordance= 0.615  (se = 0.019 )
## Likelihood ratio test= 36.89  on 1 df,   p=1e-09
## Wald test            = 33.51  on 1 df,   p=7e-09
## Score (logrank) test = 36.32  on 1 df,   p=2e-09
```

Next, we will impute event times for those subjects with $Z = 1$ and who were censored prior to the data cut-off time of 3. We first define a new variable in the data frame to select which subjects will be imputed:

```
gamma.dataset$basegamma <- NA
gamma.dataset$basegamma[(gamma.dataset$Y < 3) & (gamma.dataset$Z==1)] <- 1
```

Note that the value of the `gamma` argument for subjects who have had an event in the original data is ignored.

Now we impute in the $Z = 1$ group, assuming non-informative censoring, by setting `gamma.factor` to zero:

```
imputed.data.sets <- gammaImpute(formula=Surv(Y,delta)~Z,
                                   data = gamma.dataset, m=100, gamma="basegamma",
                                   gamma.factor=0, DCO.time="DCO.time")
fits <- ImputeStat(imputed.data.sets)
summary(fits)

##          est       se        t       df     Pr(>|t|)     lo 95    hi 95
## Z1 0.9923284 0.171484 5.786712 8864.249 7.421585e-09 0.6561801 1.328477
```

As we should expect, the estimate and standard error for the comparison of treatment groups is very similar to the estimates from fitting the Cox model to the observed data. This is because the non-informative censoring assumption made by fitting the Cox model to the observed data is the same non-informative censoring assumption being made by the imputation model.

4.1.2 Analyses assuming immediate failure after censoring in $Z = 1$

We will now analyse making the rather extreme assumption that subjects in the $Z = 1$ group who were censored prior to their data cut-off in fact fail immediately after they were censored. We first perform this analysis directly, by generating a copy of the original data frame and modifying the time and event indicator to reflect this assumption:

```
gamma.dataset.copy <- gamma.dataset
gamma.dataset.copy$Y[(gamma.dataset.copy$Z==1) & (gamma.dataset.copy$delta==0) &
                        (gamma.dataset.copy$Y<gamma.dataset.copy$DCO.time)] <-
  gamma.dataset.copy$Y[(gamma.dataset.copy$Z==1) & (gamma.dataset.copy$delta==0) &
                         (gamma.dataset.copy$Y<gamma.dataset.copy$DCO.time)]+ 0.0001
gamma.dataset.copy$delta[(gamma.dataset.copy$Z==1) & (gamma.dataset.copy$delta==0) &
                        (gamma.dataset.copy$Y<gamma.dataset.copy$DCO.time)] <- 1
```

Now we fit the Cox model to this modified dataset:

```
immFailureCox <- coxph(Surv(Y,delta) ~ Z, data=gamma.dataset.copy)
summary(immFailureCox)

## Call:
## coxph(formula = Surv(Y, delta) ~ Z, data = gamma.dataset.copy)
##
##   n= 1000, number of events= 414
##
##      coef exp(coef) se(coef)    z Pr(>|z|)
## Z1 2.1460    8.5509   0.1522 14.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##    exp(coef) exp(-coef) lower .95 upper .95
## Z1     8.551     0.1169     6.345     11.52
##
## Concordance= 0.707  (se = 0.01 )
## Likelihood ratio test= 317.8  on 1 df,   p=<2e-16
## Wald test            = 198.8  on 1 df,   p=<2e-16
## Score (logrank) test = 287.4  on 1 df,   p=<2e-16
```

The estimated log hazard ratio comparing $Z = 1$ versus $Z = 0$ is now larger. This makes sense, as we have now assumed that in the $Z = 1$ group (only) those subjects who were censored actually failed immediately after their censoring.

We now demonstrate that we can obtain the same results by using `gammaImpute`. To do this we impute the original dataset using `gamma.factor=1000`, which increases the hazard for failure after censoring by exp(1000) which R evaluates to infinity.

```
imputed.data.sets <- gammaImpute(formula=Surv(Y,delta)~Z,
                                  data = gamma.dataset, m=100, gamma="basegamma",
                                  gamma.factor=1000, DCO.time="DCO.time")
fits <- ImputeStat(imputed.data.sets)
summary(fits)

##         est        se        t  df Pr(>|t|)    lo 95   hi 95
## Z1 2.146058 0.1522129 14.09906 Inf        0 1.847726 2.44439
```

Re-assuringly, the estimate and standard error are almost identical.

4.1.3 Analyses assuming censoring at data cut-off in $Z = 0$

We will now analyse make the opposite extreme assumption, namely that those in the $Z = 1$ group who were censored prior to the data cut-off time (3) were all in fact event free at the time of data cut-off:

```
gamma.dataset.copy <- gamma.dataset
gamma.dataset.copy$Y[(gamma.dataset.copy$Z==1) & (gamma.dataset.copy$delta==0) &
                        (gamma.dataset.copy$Y<gamma.dataset.copy$DCO.time)] <- 3
```

Now we fit the Cox model to this modified dataset:

```
noFailureCox <- coxph(Surv(Y,delta) ~ Z, data=gamma.dataset.copy)
summary(noFailureCox)

## Call:
## coxph(formula = Surv(Y, delta) ~ Z, data = gamma.dataset.copy)
##
##   n= 1000, number of events= 163
##
##      coef exp(coef) se(coef)     z Pr(>|z|)
## Z1 0.5618    1.7538   0.1715 3.275  0.00106 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##    exp(coef) exp(-coef) lower .95 upper .95
## Z1     1.754     0.5702     1.253     2.455
##
## Concordance= 0.568  (se = 0.019 )
## Likelihood ratio test= 11.42  on 1 df,   p=7e-04
## Wald test            = 10.72  on 1 df,   p=0.001
## Score (logrank) test = 11  on 1 df,   p=9e-04
```

The estimated log hazard ratio comparing $Z = 1$ versus $Z = 0$ is now smaller than that obtained from analysing the original observed data. Again this makes sense, as we have now assumed that in the $Z = 1$ group those subjects who were censored prior to the data cut-off time are now censored at the data cut-off time.

As before, we now demonstrate that we can obtain the same results by using `gammaImpute`. To do this we impute the original dataset using `gamma.factor=-1000`, which increases the hazard for failure after censoring by $\exp(-1000)$ which R evaluates to zero.

```
imputed.data.sets <- gammaImpute(formula=Surv(Y,delta)~Z,
                                 data = gamma.dataset, m=100, gamma="basegamma",
                                 gamma.factor=-1000, DCO.time="DCO.time")
fits <- ImputeStat(imputed.data.sets)
summary(fits)

##          est        se        t  df    Pr(>|t|)    lo 95     hi 95
## Z1 0.5617668 0.1715479 3.274693 Inf 0.001057768 0.225539 0.8979945
```

Re-assuringly, the estimate and standard error are almost identical to those obtained from the `noFailureCox` model fit.

4.1.4 Sensitivity analysis

The preceding analyses make assumptions that would ordinarily be considered extreme, and almost certainly implausible. As discussed earlier, it may be of interest to examine how the treatment effect estimate varies as γ is varied between these extremes. We now repeatedly perform the imputation and analyse the resulting datasets, for various values of `gamma.factor` (again only imputing in those with $Z = 1$). The code below defines a function which performs the imputation for a given value of `gamma.factor`. We now use 10 imputations to reduce computation time. Through an initial analysis, we established that the treatment effect HR effectively stablizes at values of `gamma.factor` above 7 and below -7. We therefore only perform imputations for values of `gamma.factor` between -7 and 7:

```
#we now define a function which will apply the gamma imputation and return
#the model fit results. Note we will only ask for imputation in Z=1 group
#by setting
my.imputation.function <- function(gamma.factor){
  imputed.data.sets <- gammaImpute(formula=Surv(Y,delta)~Z,
                                   data = gamma.dataset, m=10, gamma="basegamma",
                                   gamma.factor=gamma.factor, DCO.time="DCO.time")
  fits <- ImputeStat(imputed.data.sets)
  return(summary(fits))
}

#set random seed
set.seed(12123)

#we will impute across the following sequence of gamma values
gamma.factor <- seq(-7,7)

#perform the imputations
answers <- lapply(gamma.factor,my.imputation.function)

#extract out the Z=1 vs Z=0 results into a matrix and add gamma column
Z1 <- do.call("rbind",lapply(answers,function(x)x["Z1",]))
Z1 <- cbind(Z1,gamma.factor)
head(Z1)

##            est        se        t           df     Pr(>|t|)     lo 95     hi 95
## [1,] 0.5617668 0.1715479 3.274693          Inf 0.0010577679 0.2255390 0.8979945
## [2,] 0.5626188 0.1715477 3.279664 122226191.19 0.0010393092 0.2263915 0.8988462
## [3,] 0.5645710 0.1715983 3.290074   4755966.60 0.0010016190 0.2282445 0.9008976
## [4,] 0.5707400 0.1714513 3.328876   2825716.38 0.0008719844 0.2347015 0.9067784
## [5,] 0.5966176 0.1714414 3.480009     96040.54 0.0005016185 0.2605944 0.9326407
## [6,] 0.6415657 0.1708002 3.756235     38639.28 0.0001727420 0.3067930 0.9763385
```

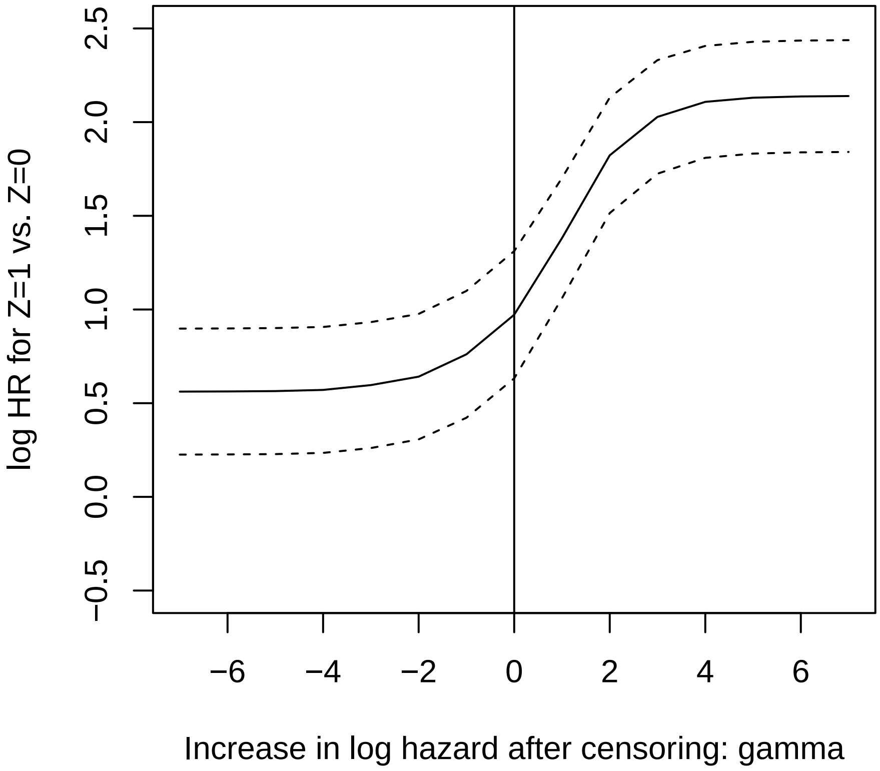

Figure 2: Estimates for log HR comparing group 1 to group 0, as a function of gamma with 95% confidence interval. Note that gamma=0 corresponds to assuming non-informative censoring.

```
##      gamma.factor
## [1,]           -7
## [2,]           -6
## [3,]           -5
## [4,]           -4
## [5,]           -3
## [6,]           -2

#ylim=c(-0.5,2)
plot(gamma.factor,Z1[,"est"],type="l",ylim=c(-0.5,2.5),ylab="log HR for Z=1 vs. Z=0",
     xlab="Increase in log hazard after censoring: gamma")
lines(gamma.factor,Z1[,"lo 95"],lty=2)
lines(gamma.factor,Z1[,"hi 95"],lty=2)
abline(v=0)
```

Figure 2 shows the estimated log HR for $Z = 1$ versus $Z = 0$ (and pointwise 95% confidence intervals) as `gamma.factor` ranges between -7 and 7. As we should expect given our earlier results, the log HR ranges from around 0.5 to just over 2 as we vary `gamma.factor`.

References

[1] Dan Jackson, Ian White, Shaun Seaman, Hannah Evans, Kathy Baisley, and James Carpenter. Relaxing the independent censoring assumption in the Cox proportional hazards model using multiple imputation. *Statistics in Medicine*, 33(27):4681–4694, 2014.

[2] Xiao-Li Meng. Multiple-Imputation Inferences with Uncongenial Sources of Input. *Statistical Science*, 9(4):538–558, 1994.

[3] Donald B Rubin. Multiple Imputation for Nonresponse in Surveys (Wiley Series in Probability and Statistics). 1987.

[4] Donald B Rubin and Nathaniel Schenker. Multiple imputation in health-are databases: An overview and some applications. *Statistics in medicine*, 10(4):585–598, 1991.