Cox Model

Zachary Gager

12/13/2021

Import Libraries and External Files

```
library(survival)
library(gridExtra)
source("../SimulationFunctions.R")
source("../convert_list.R")
```

Cox Example

In our case of one time-varying covariate (the function-valued trait), the Cox Proportional Hazard Regression Model has the form

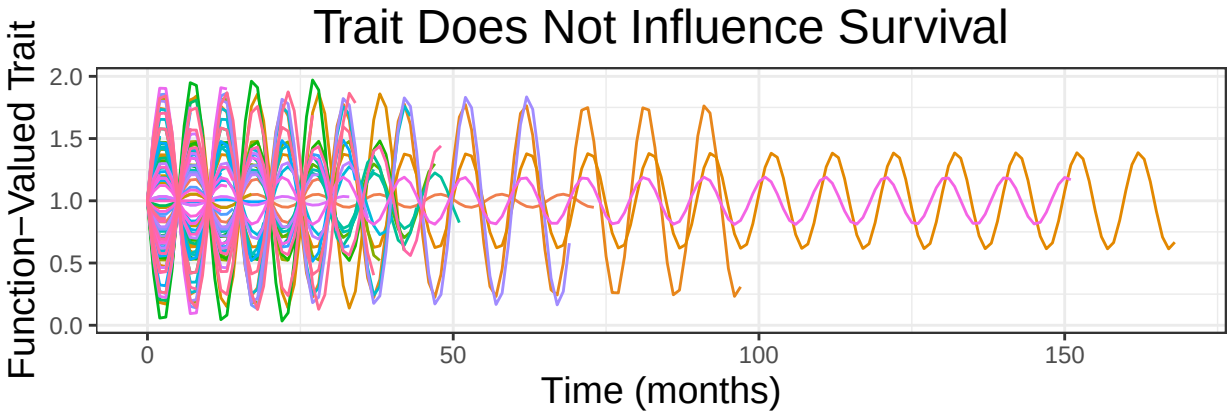
$\lambda(t) = \lambda_0(t)e^{\beta * \text{fvt}(t)}$

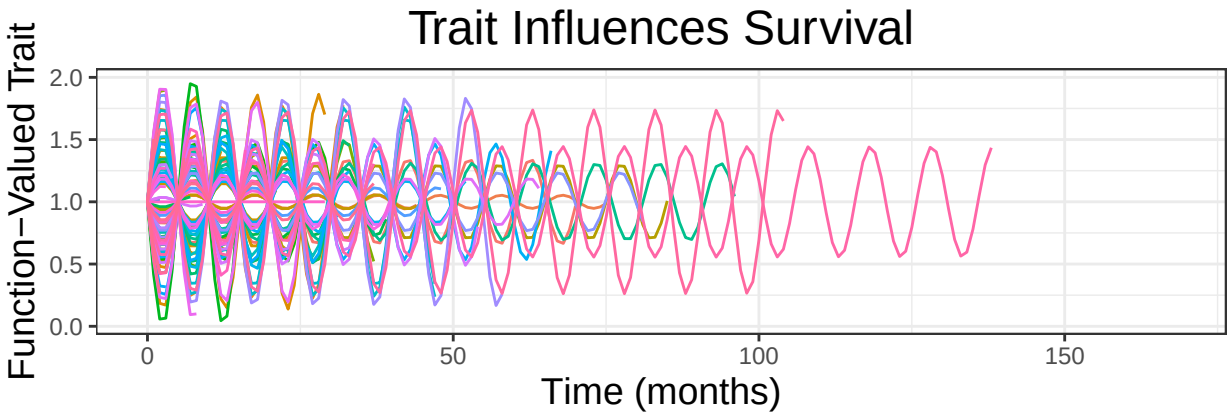
The hypothesis test is $H_0 : \beta = 0$ and $H_A : \beta \neq 0$.

The `sim_exp_sin` function simulates the data with a hazard rate with this exponential relationship, where $\lambda_0(t) =$ `lambda_fvt` and $\beta =$ `m`. Here is an example of analyzing our `sim_exp_sin` (with the default parameters and 100 organisms) with the `coxph` function. The `coxph` function gives the estimated value for β and its associated p-value for the above hypothesis test.

Simulate

```
set.seed(4)
out = sim_exp_sin(100, create_plots=T)
grid.arrange(out$plot_nat, out$plot_fvt)
```





Convert List

Currently, the data looks like this

```
out$nat[1:5, 1:5] # or out$fvt for where trait influences survival
```

```
     [,1]      [,2]      [,3]      [,4]      [,5]
[1,]    1 1.2008816 1.3253539 1.3260713        NA
[2,]    1 0.9660096 0.9451396        NA        NA
[3,]    1 0.8960364 0.8311078 0.8295925 0.8920596
[4,]    1 0.9690740 0.9498257 0.9495232 0.9682809
[5,]    1 1.0130065        NA        NA        NA
```

where the rows are organisms and the columns are time. If the organism is dead their value at that time is NA.

This is not how the `coxph` function likes the data, so we made a `convert_list` function to make it in the correct format.

```
converted_nat = convert_list(out$nat)
converted_fvt = convert_list(out$fvt)
head(converted_nat)
```

```
  time status     trait ids
1    1      0 1.2008816   1
2    2      0 1.3253539   1
3    3      1 1.3260713   1
4    1      0 0.9660096   2
5    2      1 0.9451396   2
```

```
6    1      0 0.8960364   3
```

Here the status is 1 when the organism dies. The column `ids` indicate which organism. For interval censored data, we also needed a t_0 for each data point, so we added that like so:

```
converted_nat$time0 = converted_nat$time - 1
converted_fvt$time0 = converted_fvt$time - 1
head(converted_nat)
```

```
  time status     trait ids time0
1    1      0 1.2008816   1     0
2    2      0 1.3253539   1     1
3    3      1 1.3260713   1     2
4    1      0 0.9660096   2     0
5    2      1 0.9451396   2     1
6    1      0 0.8960364   3     0
```

Now we have time intervals, where each row represents an specific organism, its status, and its function-valued trait at time `[time0, time)`

Right Censored

Using the converted lists, we ran them through `coxph` as either right or interval censored data. With right censored data, we didn't use `time0` in the model.

```
# Dataset with natural death
cox_model_right_nat = coxph(Surv(time, status) ~ trait, data=converted_nat)
summary(cox_model_right_nat)
```

```
Call:
coxph(formula = Surv(time, status) ~ trait, data = converted_nat)

  n= 2065, number of events= 100
   (2 observations deleted due to missingness)

        coef exp(coef) se(coef)     z Pr(>|z|)
trait 0.3134    1.3680   0.2891 1.084    0.278

      exp(coef) exp(-coef) lower .95 upper .95
trait     1.368      0.731    0.7762     2.411

Concordance= 0.547  (se = 0.032 )
Likelihood ratio test= 1.17  on 1 df,   p=0.3
Wald test            = 1.17  on 1 df,   p=0.3
Score (logrank) test = 1.17  on 1 df,   p=0.3
```

In the output above, the value of `coef` (0.3134) is the estimate of β, the `exp(coef)` (1.368) term is the estimate of e^{β}, and the `Pr(>|z|)` term (0.278) is the p-value of the above hypothesis test. We noticed that the natural death dataset had a p-value typically greater than 0.05.

Next is the `coxph` function on the dataset where the hazard rate was exponentially proportional to the function-valued trait.

```
# Dataset with hazard rate exponentially proportional to fvt
cox_model_right_fvt = coxph(Surv(time, status) ~ trait, data=converted_fvt)
summary(cox_model_right_fvt)
```

```
Call:
coxph(formula = Surv(time, status) ~ trait, data = converted_fvt)
```

```
  n= 2360, number of events= 100
   (4 observations deleted due to missingness)

        coef exp(coef) se(coef)     z Pr(>|z|)
trait 0.6099    1.8402   0.3156 1.933   0.0533 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      exp(coef) exp(-coef) lower .95 upper .95
trait      1.84     0.5434    0.9914     3.416

Concordance= 0.578  (se = 0.034 )
Likelihood ratio test= 3.74  on 1 df,   p=0.05
Wald test            = 3.73  on 1 df,   p=0.05
Score (logrank) test = 3.74  on 1 df,   p=0.05
```

Here, the `coef` term (0.609) is greater than 0. The actual β value we used in this simulated dataset was `m=1`. Also, the p-value here was `0.0533`, very close to 0.05. We classified the models as follows:

FVT Influences Survival: `coef > 0` and `Pr(>|z|) < 0.05`

FVT Doesn't Influence Survival: `coef < 0` or `Pr(>|z|) > 0.05`

We would classify both the above populations as **FVT Doesn't Influence Survival**; however, the actual population where FVT Influences Survival was very close to a correct classification (the p-value was just 0.0533).

Interval Censored

We just did the above but added `time0` as a variable, like so:

```
# Dataset with natural death
cox_model_interval_nat = coxph(Surv(time0, time, status) ~ trait, data=converted_nat)
summary(cox_model_interval_nat)
```

```
Call:
coxph(formula = Surv(time0, time, status) ~ trait, data = converted_nat)

  n= 2065, number of events= 100
   (2 observations deleted due to missingness)

        coef exp(coef) se(coef)     z Pr(>|z|)
trait 0.2710    1.3113   0.2687 1.009    0.313

      exp(coef) exp(-coef) lower .95 upper .95
trait     1.311     0.7626    0.7744     2.221

Concordance= 0.549  (se = 0.033 )
Likelihood ratio test= 1.02  on 1 df,   p=0.3
Wald test            = 1.02  on 1 df,   p=0.3
Score (logrank) test = 1.02  on 1 df,   p=0.3
```

The estimates did not change much for the natural death dataset

```
# Dataset with hazard rate exponentially proportional to fvt
cox_model_interval_fvt = coxph(Surv(time0, time, status) ~ trait, data=converted_fvt)
summary(cox_model_interval_fvt)
```

```
Call:
coxph(formula = Surv(time0, time, status) ~ trait, data = converted_fvt)

  n= 2360, number of events= 100
   (4 observations deleted due to missingness)

        coef exp(coef) se(coef)     z Pr(>|z|)
trait 0.5849    1.7948   0.3121 1.874   0.0609 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      exp(coef) exp(-coef) lower .95 upper .95
trait     1.795     0.5572    0.9736     3.309

Concordance= 0.588  (se = 0.036 )
Likelihood ratio test= 3.54  on 1 df,   p=0.06
Wald test            = 3.51  on 1 df,   p=0.06
Score (logrank) test = 3.54  on 1 df,   p=0.06
```

Again, the p-value was smaller for the dataset where there was a relationship, but in this case both would've been classified as **FVT Doesn't Influence Survival**.

Clustering

We can tell the `coxph` model to consider covariance amongst the same individual in a population with the `cluster` parameter.

```
# Dataset with hazard rate exponentially proportional to fvt
cox_model_interval_fvt_withcluster = coxph(Surv(time0, time, status) ~ trait,
                                           cluster=ids,
                                           data=converted_fvt)
summary(cox_model_interval_fvt_withcluster)
```

```
Call:
coxph(formula = Surv(time0, time, status) ~ trait, data = converted_fvt,
    cluster = ids)

  n= 2360, number of events= 100
   (4 observations deleted due to missingness)

        coef exp(coef) se(coef) robust se     z Pr(>|z|)
trait 0.5849    1.7948   0.3121    0.3462 1.689   0.0912 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      exp(coef) exp(-coef) lower .95 upper .95
trait     1.795     0.5572    0.9105     3.538

Concordance= 0.588  (se = 0.036 )
Likelihood ratio test= 3.54  on 1 df,   p=0.06
Wald test            = 2.85  on 1 df,   p=0.09
Score (logrank) test = 3.54  on 1 df,   p=0.06,   Robust = 3.17  p=0.07

  (Note: the likelihood ratio and score tests assume independence of
     observations within a cluster, the Wald and robust score tests do not).
```

So we tested all 4 combinations

- Right censored data with clustering by `ids`
- Right censored data without clustering by `ids`
- Interval censored data with clustering by `ids`
- Interval censored data without clustering by `ids`

The above for combinations were tested in `cox_models.R`