Introduction to Bioequivalence

José Bayoán Santiago Calderón, PhD

2020-06-03

1 Introduction

We recommend first reading the documentation which goes over how to download and install the module.

1.1 Quick Start

We first explore an example from the `pumas_be` function documentation.

Let us load the module through

```
using Pkg
Pkg.activate(joinpath(@__DIR__, "tutorials", "bioequivalence"))
using Bioequivalence
```

The first example comes from a two sequences, two periods, two formulations crossover design (`RT|TR`) from Schütz, Labes, and Fuglsang (2014).

```
SLF2014 = Bioequivalence.testdata("SLF2014_8")
show(SLF2014)
```

```
1434×4 DataFrames.DataFrame
 Row    id      period   sequence   AUC
        Int64   Int64    Cat...       Float64

 1      1       1        TR         168.407
 2      1       2        TR         210.919
 3      2       1        TR         131.031
 4      2       2        TR         67.4314
 5      3       1        TR         151.737
 6      3       2        TR         85.1296
 7      4       1        TR         375.575
 8      4       2        TR         147.502
 9      5       1        TR         50.023
 10     5       2        TR         20.0107
⋮
 1424   995     2        RT         47.0775
 1425   996     1        RT         128.804
 1426   996     2        RT         102.543
 1427   997     1        RT         85.6709
 1428   997     2        RT         45.7204
 1429   998     1        RT         286.78
```

```
 1430   998     2        RT         73.9542
 1431   999     1        RT         63.1119
 1432   999     2        RT         13.3009
 1433   1000    1        RT         55.8031
 1434   1000    2        RT         50.0577
```

The `id` variable specifyies the identifier for each subject.

The `sequence` must be a `String` with each `Character` representing a formulation.

For example, `"RTTR"` represents `R` formulation in the first period, `T` formulation in the second and third periods, and `R` formulation in the fourth period.

The `period` variable must be integers specifying the period based on the sequence.

For example,

```
id = 1
period = 1
sequence = "TR"
endpoint = 168.407
```

means that the endpoint for subject with ID `1` in the first period when receiving the `T` formulation recorded the value 168.407.

Notice that the study design is fully characterized by the unique values of the `sequence` variable.

In order to perform the bioequivalence analysis we can pass the data to the `pumas_be` function.

```
Crossover = pumas_be(SLF2014)
show(Crossover)
```

```
Design: RT|TR

Sequences: RT|TR (2)
Periods: 1:2 (2)
Subjects per Sequence: (RT = 288, TR = 429)

Average Bioequivalence

               PE        SE       lnLB        lnUB       GMR     LB      UB


T - R  -0.0680309  0.044609  -0.141501  0.00543947  0.934232  0.868  1.0055
```

When passing only the data argument it will use the default values for all the other arguments.

For example, if the dataset had the variable indicating the subject ID as `subj` instead of `id`,

```
using DataFrames
rename!(SLF2014, "id" => "subj")
try
    pumas_be(SLF2014)
catch err
    err
end
```

```
Error: ArgumentError: data must contain id, sequence, period, and endpoint
```

In this case we can pass the variable position (i.e., `1`) or the name of the variable as a `Symbol` to the function.

```
Crossover = pumas_be(SLF2014, id = :subj)
show(Crossover)
```

```
Design: RT|TR

Sequences: RT|TR (2)
Periods: 1:2 (2)
Subjects per Sequence: (RT = 288, TR = 429)

Average Bioequivalence

               PE        SE       lnLB        lnUB       GMR     LB      UB


T - R  -0.0680309  0.044609  -0.141501  0.00543947  0.934232  0.868  1.0055
```

1.2 Examining the Results

The `BioequivalenceStudy` will show the study design as well as how many subjects were in each sequence.

For convenience it also shows the formulation and periods as well as how many there are for that design.

A coefficients table will also be showing the parameter estimates and related values for each comparison among each non-reference formulation and the reference formulation. Those will include the point estimate (`PE`), standard error (`SE`), the 90% confidence interval in the natural log scale (i.e., `lnLB`, `lnUB`), the geometric means ratio (`GMR`), and the 90% confidence intervals in the natural scale (i.e., `LB`, `UB`).

The average bioequivalence is assessed based on the lower and upper bounds of the 90% percent confidence interval. Based on the regulatory guidelines for the FDA, the confidence intervals are reported in percentage form with two decimal places. In other words, if the lower bound value is 0.85793 the percentage form would be 85.793% and be rounded down to 85.79%. If the upper bound value is 1.20522 the percentage form would be 120.522% and be rounded up to 120.53%. The Bioequivalence results will have correct percentage format in decimal form (e.g. 1.2053 for 120.53%).

Besides the bioequivalence results one can query a number of properties.

We can access the information about the design

```
show(Crossover.design)
```

```
(RT = 288, TR = 429)
```

The data used for the model

```
show(Crossover.data)
```

```
1434×5 DataFrames.DataFrame
 Row    id     sequence   period   AUC       formulation
        Cat...   Cat...       Cat...     Float64   Cat...
```

```
 1      1      TR         1        5.12638   'T'
 2      1      TR         2        5.35148   'R'
 3      2      TR         1        4.87543   'T'
 4      2      TR         2        4.21111   'R'
 5      3      TR         1        5.02215   'T'
 6      3      TR         2        4.44418   'R'
 7      4      TR         1        5.92846   'T'
 8      4      TR         2        4.99384   'R'
 9      5      TR         1        3.91248   'T'
 10     5      TR         2        2.99627   'R'
⋮
 1424   995    RT         2        3.8518    'T'
 1425   996    RT         1        4.85829   'R'
 1426   996    RT         2        4.63028   'T'
 1427   997    RT         1        4.45051   'R'
 1428   997    RT         2        3.82254   'T'
 1429   998    RT         1        5.65872   'R'
 1430   998    RT         2        4.30345   'T'
 1431   999    RT         1        4.14491   'R'
 1432   999    RT         2        2.58783   'T'
 1433   1000   RT         1        4.02183   'R'
 1434   1000   RT         2        3.91318   'T'
```

Statistics about the data such as:

how many observations did the initial data had.

```
show(Crossover.data_stats.total)
```

```
1434
```

how many observations did the analysis actually used (e.g., total - count of missing values).

```
show(Crossover.data_stats.used_for_analysis)
```

```
1434
```

as well as summary statistics of the endpoint distribution by sequence, period or formulation.

```
show(Crossover.data_stats.sequence)
```

```
2×9 DataFrames.DataFrame
 Row   sequence   mean      std        min       q25       median    q75       max       n
       Cat...     Float64   Float64    Float64   Float64   Float64   Float64   Float64   Int64

 1     RT         4.25295   0.858753   2.22401   3.77021   4.25043   4.71928   17.7519   576

 2     TR         5.44929   3.12075    2.42739   4.32815   4.8114    5.24877   20.1913   858
```

```
show(Crossover.data_stats.period)
```

```
2×9 DataFrames.DataFrame
 Row   period   mean      std       min       q25       median    q75       max       n
       Cat...   Float64   Float64   Float64   Float64   Float64   Float64   Float64   Int64

 1     1        5.32608   2.57664   3.27928   4.50087   4.89085   5.27117   20.1913   717
 2     2        4.61142   2.45832   2.22401   3.77503   4.22118   4.67236   19.9725   717
```

```
show(Crossover.data_stats.formulation)
```

```
2×9 DataFrames.DataFrame
 Row   formulation   mean      std       min       q25       median    q75       max       n
       Cat...          Float64   Float64   Float64   Float64   Float64   Float64   Float64
Int64

 1     'R'           4.93118   2.43666   2.42739   4.14032   4.52236   4.96409   19.9725
717
 2     'T'           5.00632   2.64537   2.22401   4.00392   4.67886   5.19103   20.1913
717
```

One can also query the model used for inference. The exact API for extracting values from the models will depend on the design. The most common API is documented here. For example, we can obtain the coefficient of determination (R2) through the API.

```
using StatsBase
r2(Crossover.model)
```

```
0.947066878314061
```

likewise we can obtain the `deviance` of the model through

```
deviance(Crossover.model)
```

```
490.35529438202303
```

Derived statistics for the model such as the Wald test for join-significance for the sequence, period, and formulation covariates

```
show(Crossover.model_stats.Wald)
```

```
                  Wald               Distribution  p-value

Formulation     2.32578  FDist(ν1=1.0, ν2=715.0)     0.1277
Sequence         2.3571  FDist(ν1=1.0, ν2=715.0)     0.1252
Period          266.357  FDist(ν1=1.0, ν2=715.0)     <1e-50
Subject          16.5092  FDist(ν1=715.0, ν2=715.0)   <1e-99
```

and the linear regression mean estimates per formulation

```
show(Crossover.model_stats.lsmeans)
```

```
      Mean  Standard Deviation  t-statistic    Distribution  p-value

R  4.63723            0.414669      11.183   TDist(ν=715.0)   <1e-26
T  4.5692             0.414669      11.0189  TDist(ν=715.0)   <1e-25
```

are available as showen above.

You can also run models for various endpoints easily as following

```
data = Bioequivalence.testdata("PJ2017_4_3")
show(data)
```

```
68×5 DataFrames.DataFrame
 Row   id      sequence   period   AUC       Cmax
       Int64   Cat...       Int64    Int64?    Int64?

 1     1       RTTR       1        10671     817
 2     1       RTTR       2        12772     1439
 3     1       RTTR       3        13151     1310
 4     1       RTTR       4        11206     1502
 5     2       TRRT       1        6518      1393
 6     2       TRRT       2        6068      1372
 7     2       TRRT       3        5996      1056
 8     2       TRRT       4        5844      1310
 9     3       TRRT       1        4939      1481
 10    3       TRRT       2        5728      1377
⋮
 58    15      RTTR       2        7905      1065
 59    15      RTTR       3        6550      830
 60    15      RTTR       4        7515      1247
 61    16      RTTR       1        11473     1368
 62    16      RTTR       2        9698      1281
 63    16      RTTR       3        10355     1083
 64    16      RTTR       4        10365     1418
 65    18      TRRT       1        5210      668
 66    18      TRRT       2        5120      842
 67    18      TRRT       3        5420      1176
 68    18      TRRT       4        missing   missing
```

```
auc = pumas_be(data)
cmax = pumas_be(data, endpoint = :Cmax)
output = Dict(endpoint => pumas_be(data, endpoint = endpoint) for endpoint in
setdiff(propertynames(data), (:id, :sequence, :period)))
```

```
Dict{Symbol,Bioequivalence.BioequivalenceStudy{NamedTuple{(:total, :used_for_analysis, :
formulation, :sequence, :period),Tuple{Int
64,Int64,DataFrames.DataFrame,DataFrames.DataFrame,DataFrames.DataFrame}},NamedTuple{(:
RTTR, :TRRT),Tuple{Int64,Int64}},MixedModel
s.LinearMixedModel{Float64},NamedTuple{(:Wald, :lsmeans),Tuple{StatsBase.CoefTable,
StatsBase.CoefTable}},Tuple{Float64,Float64}}}
with 2 entries:
  :AUC  => Design: RTTR|TRRT...
  :Cmax => Design: RTTR|TRRT...
```

```
show(output[:AUC])
```

```
Design: RTTR|TRRT

Sequences: RTTR|TRRT (2)
Periods: 1:4 (4)
Subjects per Sequence: (RTTR = 8, TRRT = 9)
Regulatory constant for reference-scaled estimates: 0.10
Auxiliary parameter for reference-scaled estimates: 1.11

Average Bioequivalence

              PE         SE         lnLB       lnUB      GMR      LB       UB     scLB
scUB  Varratio    VarLB    VarUB
```

```
T - R  0.0356935  0.0226361  -0.00230475  0.0736918  1.03634  0.9976  1.0765  0.8682
1.1518   1.34546  0.8641  2.1116
```

```
show(output[:Cmax])
```

```
Design: RTTR|TRRT

Sequences: RTTR|TRRT (2)
Periods: 1:4 (4)
Subjects per Sequence: (RTTR = 8, TRRT = 9)
Regulatory constant for reference-scaled estimates: 0.10
Auxiliary parameter for reference-scaled estimates: 1.11

Average Bioequivalence

               PE         SE       lnLB         lnUB       GMR      LB      UB    scLB
 scUB  Varratio   VarLB   VarUB

T - R  -0.0917895  0.0524819  -0.179889  -0.00369023  0.912297  0.8353  0.9964  0.7005
1.4274   1.25641  0.8069  1.9719
```