

1 Bioequivalence

Bioequivalence.jl is a package for performing bioequivalence analysis part of the Pumas (PharmaceUtical Modeling And Simulation) ecosystem for developing, simulating, fitting, and analyzing pharmaceutical models.

This document provides a how-to and technical details for performing bioequivalence analysis using the software.

1.1 How to obtain Bioequivalence.jl?

Bioequivalence.jl is covered under the [Julia Computing EULA](#) and distributed through via [JuliaPro](#) from Julia Computing Inc.

Using JuliaPro, one can install the Bioequivalence.jl package using the Julia package manager.

```
using Pkg;  
Pkg.add("Bioequivalence")
```

1.2 Getting Started

To load the package, use

```
using Bioequivalence
```

The Bioequivalence package provides two functions: `read_be` and `generate_design`.

`read_be` is an alias for `BioequivalenceStudy` which takes a table with data from a bioequivalence study and other optional arguments and returns the output of the analysis. This function tries to be quite smart and flexible. For most cases, it should work how you want it to work, but allows for users to overwrite default behavior.

The other function provided by Bioequivalence.jl is `generate_design` which allows to generate data according to some bioequivalence study design.

1.2.1 Setting up the environment

There are a few packages that can be helpful when using Bioequivalence.jl.

- CSV.jl is a package for importing text files (e.g., comma-separated values or CSV)
- DataFrames.jl is a package that provides a table representation for data
- StatsBase.jl is a package that provides a common way to query statistical models

In order to install any of these packages, one can use the package manager.

For example to install CSV.jl,

```
julia> using Pkg;  
julia> Pkg.add("CSV")
```

We can load the packages through,

```
using CSV, DataFrames, StatsBase, Bioequivalence
```

We always recommend using a local environment for each application which allows for reproducibility. For learning more about environments, we recommend checking out the [Julia documentation](#).

1.3 Designs

For each design, we will use the syntax `read_be(data, endpoint, kwargs...)`

where the keyword arguments allow to change some default behavior.

For example, every data, should be a `DataFrame` with the following variables:

- `id`: the subject ID
- `sequence`: the sequence of formulations (e.g., "RT" for reference and test)
- `period`: integers referring to the periods starting with the first period being 1
- `endpoint`: the values for the endpoint (e.g., AUC)

If the dataset has the endpoint under the variable `auc`, the syntax would be

```
output = read_be(data, :auc)
```

If the `id` variable is named `subject_id` then the syntax would follow as

```
output = read_be(data, :auc, id = :subject_id)
```

Other arguments will be explained through the examples.

The result may be queried in the following manner,

```
output.data  
output.design  
output.model  
output.result
```

The final section on additional information provides a complete summary.

This section will cover the most common arguments for each of the supported designs:

- Parallel (e.g., R|T, A|B|C)
- 2x2 (e.g., RT|TR)

- Balaam (e.g., RR|RT|TR|TT)
- Dual (e.g., RTT|TRR)
- 2S4P1 (e.g., RTTR|TRRT)
- 2S4P2 (e.g., RTRT|TRTR)
- WD3F (e.g., ABC|ACB|BAC|BCA|CAB|CBA)
- WD4F (e.g., ABCD|CADB|DCBA|BDAC)

1.3.1 Nonparametric

Nonparametric designs are included for analysis of endpoints such as Tmax.

We can load data from a bioequivalence study by using the `CSV.read` function and giving it the path to a file.

In this case, we will use some example/validation datasets included with Bioequivalence.jl.

This dataset is available from Patterson and Jones (2006).

```
PJ31 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/Nonparametric/PJ2006_3_1.tsv")),
first(PJ31, 6)
```

	id	sequence	period	Tmax
	Int64	String	Int64	Float64
1	1	RT	1	0.5
2	1	RT	2	0.5
3	2	TR	1	1.0
4	2	TR	2	1.0
5	3	TR	1	0.5
6	3	TR	2	0.5

We have now assigned the data to the variable PJ31.

The average bioequivalence analysis can be requested through `read_be`.

```
Tmax = read_be(PJ31, :Tmax)
```

Design: RT|TR

Sequences: RT|TR (2)

Periods: 1:2 (2)

Subjects per Sequence: (RT = 17, TR = 15)

Average Bioequivalence

	lnLB	lnUB	LB	UB
T - R	-0.202733	0.346574	0.816497	1.41421

The first argument is the data and the second argument is which variable should be used as the endpoint.

Variable names in `DataFrames` are symbols.

When the name of the endpoint matches some form of Tmax (case insensitive) it defaults to nonparametric.

Otherwise, it will attempt a parametric design. This can be overwritten through the `nonparametric` argument.

Nonparametric analysis uses the Wilcoxon signed rank test.

`Tmax.model`

```
1-element Array{HypothesisTests.ApproximateSignedRankTest{Float64},1}:
Approximate Wilcoxon signed rank test
```

Population details:

```
parameter of interest: Location parameter (pseudomedian)
value under h_0:      0
point estimate:       0.0
95% confidence interval: (-0.2027, 0.3466)
```

Test summary:

```
outcome with 95% confidence: fail to reject h_0
two-sided p-value:          0.8774
```

Details:

```
number of observations:      32
Wilcoxon rank-sum statistic: 182.0
rank sums:                  [182.0, 169.0]
adjustment for ties:        1848.0
normal approximation ( $\mu$ ,  $\sigma$ ): (6.5, 38.88122940443113)
```

Other endpoints can potentially be analyzed through a nonparametric method as well

```
PJ46 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/Williams/PJ2006_4_6.tsv"))
first(PJ46, 6)
```

	id	sequence	period	AUC	Cmax
	Int64	String	Int64	Float64	Float64
1	1	DCAB	1	2942.0	563.6
2	1	DCAB	2	2525.0	658.1
3	1	DCAB	3	278.0	55.6
4	1	DCAB	4	359.0	73.0
5	2	ADBC	1	484.0	108.4
6	2	ADBC	2	4190.0	818.0

```
NP = read_be(PJ46, nonparametric = true)
```

Design: ADBC|BACD|CBDA|DCAB

Sequences: ADBC|BACD|CBDA|DCAB (4)

Periods: 1:4 (4)

Subjects per Sequence: (ADBC = 7, BACD = 7, CBDA = 7, DCAB = 7)

Average Bioequivalence

	lnLB	lnUB	LB	UB
D - A	2.03582	2.11813	7.65856	8.31556
B - A	-0.0734472	0.0543133	0.929185	1.05582
C - A	2.05093	2.14861	7.7751	8.57292

1.3.2 Parallel

Consider a parallel design dataset with balance between treatment groups such as Clayton and Leslie (1981).

```
ClaytonandLeslie1981 = CSV.read(string(dirname(pathof(Bioequivalence)),
                                     "../data/Parallel/FSL2015_1.tsv"))
first(ClaytonandLeslie1981, 6)
```

	id	sequence	period	AUC
	Int64	String	Int64	Float64
1	1	T	1	2.52
2	2	T	1	8.87
3	3	T	1	0.79
4	4	T	1	1.68
5	5	T	1	6.95
6	6	T	1	1.05

The bioequivalence analysis can be requested through `BioequivalenceStudy` or its alias `read_be`.

```
# notice it defaults to the AUC endpoint
Parallel = read_be(ClaytonandLeslie1981) # Same as read_be(ClaytonandLeslie1981, :AUC)
```

Design: R|T

Sequences: R|T (2)

Periods: 1:1 (1)

Subjects per Sequence: (R = 9, T = 9)

Average Bioequivalence

	PE	SE	lnLB	lnUB	GMR	LB	UB
T - R	-0.721906	0.333328	-1.31755	-0.126258	0.485825	0.267789	0.881387

The output shows the design, statistical model, and result.

One can access the statistical model directly through

`Parallel.model`

```
1-element Array{HypothesisTests.UnequalVarianceTTest,1}:
```

```
Two sample t-test (unequal variance)
```

```
-----
Population details:
```

```
parameter of interest: Mean difference
value under h_0:      0
point estimate:       -0.7219063401460275
95% confidence interval: (-1.4507, 0.0069)
```

```
Test summary:
```

```
outcome with 95% confidence: fail to reject h_0
two-sided p-value:          0.0519
```

```
Details:
```

```
number of observations: [9,9]
t-statistic:            -2.1657564404915037
degrees of freedom:     11.633717042877814
empirical standard error: 0.3333275739825092
```

The results can be accessed through

`Parallel.result`

	Parameter	PE	SE	lnLB	lnUB	GMR	LB	UB
	String	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	T - R	-0.721906	0.333328	-1.31755	-0.126258	0.485825	0.267789	0.881387

or calling `coeftable` on the object

`coeftable(Parallel)`

	PE	SE	lnLB	lnUB	GMR	LB	UB
T - R	-0.721906	0.333328	-1.31755	-0.126258	0.485825	0.267789	0.881387

Notice that the results are validated with those reported in Fuglsang, Schütz, and Labes (2015). The Geometric Means Ratio (GMR) has a point estimate of (48.58) and a 90% confidence interval of (26.78, 88.14). These are the values reported in the study obtained with various statistical packages with the Welch correction.

1.3.3 Crossover

The most common bioequivalence design is perhaps the 2x2 crossover (RT|TR). Consider a dataset from Schütz, Labes, and Fuglsang (2014) which has been simulated with an extreme range in raw data, outliers, and imbalance between sequences and a large number of subjects.

```
SLF2014 = CSV.read(string(dirname(pathof(Bioequivalence))),
                  "../data/2S2P/SLF2014_8.tsv")
first(SLF2014, 6)
```

	id	period	sequence	AUC
	Int64	Int64	String	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

The bioequivalence analysis can be requested through `BioequivalenceStudy`

```
# notice it defaults to the AUC endpoint
Crossover = read_be(SLF2014)
```

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	0.868054	1.00545

As with other designs one can access the specific elements of the models use in the analysis.

```
loglikelihood(Crossover.model.model)
```

```
-1265.3503174162302
```

The results are obtained through,

```
Crossover.result
```

	Parameter	PE	SE	lnLB	lnUB	GMR	LB	UB
	String	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	T - R	-0.0680309	0.044609	-0.141501	0.00543947	0.934232	0.868054	1.00545

Crossover 2x2 designs are validated with Schütz, Labes, and Fuglsang (2014) and Patterson and Jones (2006). In this example, the estimate for the GMR is (93.42) with a 90% confidence interval of (86.81, 100.55).

1.3.4 Balaam

The Balaam design (RR|RT|TR|TT) is explored with a dataset from Chow and Liu (2009).

```
ChowandLiu2009 = CSV.read(string(dirname(pathof(Bioequivalence)),
                                "../data/Balaam/CL2009_9_2_1.tsv")),
first(ChowandLiu2009, 6)
```

	id	sequence	period	AUC
	Int64	String	Int64	Int64
1	1	TT	1	280
2	1	TT	2	482
3	2	TT	1	219
4	2	TT	2	161
5	3	TT	1	230
6	3	TT	2	99

The data has the same specification as other crossover studies.

The analysis follows similarly as well.

```
# notice it defaults to the AUC endpoint
Balaam = read_be(ChowandLiu2009)
```

```
Design: RR|RT|TR|TT
```

```
Sequences: RR|RT|TR|TT (4)
```

```
Periods: 1:2 (2)
```

```
Subjects per Sequence: (RR = 6, RT = 6, TR = 6, TT = 6)
```

```
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	-0.129419	0.0936591	-0.290245	0.0314075	0.878606	0.74808	1.03191	0.75401
	1.32624	6.49813	3.871	10.9082				

We can also obtain the results for the intra-subject variabilities through

```
Balaam.model. $\sigma$ 
```

```
0.2294170516947747
```

Designs with repeated measures include rescaled parameter estimates and variance analysis.

The rescaled parameter estimates are especially of interest when the reference drug is a highly variable drug (HVD) or a narrow therapeutic index (NTI) drug (Haidar et al. 2008).

The default values are:

- regulatory constant for reference-scaled estimates: 0.1
- auxiliary parameter for reference-scaled estimates: 1.11

These may be overwritten by passing the keyword arguments (see the docstring of `read_be` for more details).

1.3.5 Dual

Consider the example 4.1 in Patterson and Jones (2006)

```
PJ41 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/Dual/PJ2006_4_1.tsv"))
first(PJ41, 6)
```

	id	sequence	period	AUC	Cmax
	Int64	String	Int64	Float64	Float64
1	101	TRR	1	12.26	0.511
2	101	TRR	2	16.19	0.688
3	101	TRR	3	11.34	0.533
4	102	TRR	1	397.98	13.27
5	102	TRR	2	267.63	7.933
6	102	TRR	3	487.55	12.952

```
Dual = read_be(PJ41, :Cmax)
```

```
Design: RTT|TRR
```

```
Sequences: RTT|TRR (2)
```

```
Periods: 1:3 (3)
```

```
Subjects per Sequence: (RTT = 47, TRR = 48)
```

```
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	-0.0565121	0.0685313	-0.169803	0.0567792	0.945055	0.843831	1.05842	0.57774
	1.73088	1.23391	1.03853	1.46604				

1.3.6 2S4P Designs

For 2S4P designs both RRTT|TTRR and RTRT|TRTR are supported

```
PJ43 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/2S4P/PJ2006_4_3.tsv"))
first(PJ43, 6)
```

	id	sequence	period	AUC	Cmax
	Int64	String	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

```
Inner = read_be(PJ43)
```

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	0.997698	1.07647	0.908307
	1.10095	1.29984	0.962939	1.7546				

```
PJ44 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/2S4P/PJ2006_4_4.tsv"))
first(PJ44, 6)
```

	id	sequence	period	AUC	Cmax
	Int64	String	Int64	Float64	Float64
1	1	RTTR	1	812.6	99.85
2	1	RTTR	2	1173.7	204.09
3	1	RTTR	3	889.1	170.94
4	1	RTTR	4	620.1	112.78
5	2	TRTR	1	216.3	29.06
6	2	TRTR	2	338.0	50.48

```
Outer = read_be(PJ44, :Cmax)
```

Design: RTRT|TRTR

Sequences: RTRT|TRTR (2)

Periods: 1:4 (4)

Subjects per Sequence: (RTRT = 27, TRTR = 27)

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	0.413998	0.0745634	0.29061	0.537386	1.51285	1.33724	1.71153	0.566073	
	1.76656	0.937014	0.79558	1.10359					

1.3.7 Williams

Consider a 3 formulations Williams design

```
PJ45 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/Williams/PJ2006_4_5.tsv"))
first(PJ45, 6)
```

	id	sequence	period	AUC	Cmax
	Int64	String	Int64	Int64	Int64
1	1	SRT	1	7260	1633
2	1	SRT	2	6463	1366
3	1	SRT	3	8759	2141
4	2	RTS	1	3457	776
5	2	RTS	2	6556	2387
6	2	RTS	3	4081	1355

```
W3F = read_be(PJ45)
```

Design: RST|RTS|SRT|STR|TRS|TSR

Sequences: RST|RTS|SRT|STR|TRS|TSR (6)

Periods: 1:3 (3)

Subjects per Sequence: (RST = 9, RTS = 11, SRT = 11, STR = 10, TRS = 11, TSR = 10)

Average Bioequivalence

		PE	SE	lnLB	lnUB	GMR	LB	UB
S - R	0.341072	0.0378374	0.278325	0.403819	1.40645	1.32092	1.49753	
T - R	0.149693	0.0378182	0.086978	0.212408	1.16148	1.09087	1.23665	

Imagine for a moment that the reference formulation is actually S instead of R. One can pass such a parameter as following.

```
W3F = read_be(PJ45, reference = 'S')
```

Design: RST|RTS|SRT|STR|TRS|TSR

Sequences: RST|RTS|SRT|STR|TRS|TSR (6)

Periods: 1:3 (3)

Subjects per Sequence: (RST = 9, RTS = 11, SRT = 11, STR = 10, TRS = 11, TSR = 10)

Average Bioequivalence

		PE	SE	lnLB	lnUB	GMR	LB	UB
R - S	-0.341072	0.0378374	-0.403819	-0.278325	0.711008	0.667765	0.757051	
T - S	-0.191379	0.0380271	-0.254441	-0.128318	0.825819	0.77535	0.879574	

Williams designs for four formulations are available as well

```
PJ46 = CSV.read(string(dirname(pathof(Bioequivalence)),
                        "../data/Williams/PJ2006_4_6.tsv"))
first(PJ46, 6)
```

	id	sequence	period	AUC	Cmax
	Int64	String	Int64	Float64	Float64
1	1	DCAB	1	2942.0	563.6
2	1	DCAB	2	2525.0	658.1
3	1	DCAB	3	278.0	55.6
4	1	DCAB	4	359.0	73.0
5	2	ADBC	1	484.0	108.4
6	2	ADBC	2	4190.0	818.0

```
W4F = read_be(PJ46)
```

```
Design: ADBC|BACD|CBDA|DCAB
```

```
Sequences: ADBC|BACD|CBDA|DCAB (4)
```

```
Periods: 1:4 (4)
```

```
Subjects per Sequence: (ADBC = 7, BACD = 7, CBDA = 7, DCAB = 7)
```

Average Bioequivalence

	PE	SE	lnLB	lnUB	GMR	LB	UB
D - A	0.00468645	0.0342341	-0.0523004	0.0616733	1.0047	0.949044	1.06361
B - A	2.09353	0.0342341	2.03654	2.15052	8.11351	7.66407	8.5893
C - A	2.05738	0.0342341	2.0004	2.11437	7.82546	7.39198	8.28436

1.3.8 Additional documentation

`@doc BioequivalenceStudy`

```
BioequivalenceStudy(data::AbstractDataFrame,
                    endpoint::Union{Integer, Symbol} = :AUC,
                    w::Real = 0.1,
                    ::Real = 1.11;
                    id::Union{Integer, Symbol} = :id,
                    sequence::Union{Integer, Symbol} = :sequence,
                    period::Union{Integer, Symbol} = :period,
                    reference::Union{Nothing, Char} = nothing,
                    nonparametric::Bool = occursin(r"(?i)tmax", string(endpoint))),
                    reml::Bool = false)
```

Return a bioequivalence study

Arguments

- data: must have id, sequence, period, and an endpoint.

- id: which variable is the subject id?
- sequence: which variable is the sequence?
- period: which variable is the period?
- endpoint: which variable is the endpoint?
- w: regulatory constant for reference-scaled estimates
- : auxiliary parameter for reference-scaled estimates
- reference: which formulation is the reference?

For example, in a design with RTTR|TRRT one can specify 'R' to be the reference. By default, the reference is taken to be the first character (alphabetically).

- nonparametric: use nonparametric method for analysis of the endpoint?
- reml: if the design uses a linear mixed model should it optimize REML instead of ML?

Each bioequivalence study has the following fields:

- data: data used for the study
- design: number of subjects in each sequence
- model: statistical models used for the analysis
- result: result based on previous components

Current designs include:

- Parallel (e.g., R|T, A|B|C)
- 2x2 (e.g., RT|TR)
- Balaam (e.g., RR|RT|TR|TT)
- Dual (e.g., RTT|TRR)
- 2S4P1 (e.g., RTTR|TRRT)
- 2S4P2 (e.g., RTRT|TRTR)
- WD3F (e.g., ABC|ACB|BAC|BCA|CAB|CBA)
- WD4F (e.g., ABCD|CADB|DCBA|BDAC)

Examples,

```
julia> read_be(data) # read_be is an alias for BioequivalenceStudy, same as `Bioequival
```

```
julia> read_be(data, sequence = :Sequence) # It declares :Sequence as the sequence vari
```

```
julia> read_be(data, reml = true) # Model estimation uses REML instead of ML if applica
```

```
@doc generate_design
```

```
generate_design(design::AbstractString,  
                amt::Union{Number,AbstractVector{<:Number}},  
                formulation::AbstractVector,  
                subjects_per_sequence::Union{<:Integer,AbstractVector{<:Integer}}  
                )::DataFrame
```

Returns a DataFrame with id, sequence, period, formulation, amt, evid, cmt, and time. It can be used to quickly set up data for PuMaS, NCA, and Bioequivalence. In order to add covariates, use `join` to join the result of this function with another DataFrame with covariates.

The following designs are available:

- "Parallel" => 'A':'A' + num_formulations - 1
- "2x2" => ["RT", "TR"]
- "Balaam" => ["RR", "RT", "TR", "TT"]
- "Dual" => ["RTT", "TRR"]
- "2S4P1" => ["RTTR", "TRRT"]
- "2S4P2" => ["RTRT", "TRTR"]
- "WD3F" => ["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"]
- "WD4F" => ["ABCD", "CADB", "DCBA", "BDAC"]

Examples

```
julia> skeleton = generate_design("Parallel", 100, ["tablet", "soft", "hard"], 10)
```

```
julia> skeleton = generate_design("2S4P2", [50, 25], ["tablet", "capsule"], [10, 9])
julia> m = length(skeleton.id)
julia> data = join(skeleton,
                   DataFrame(id = 1:m,
                             wt = rand(100:200, m),
                             age = rand(25:85, m)),
                   on = :id)
```

1.4 References

Chow, Shein-Chung, and Jen-pei Liu. 2009. Design and Analysis of Bioavailability and Bioequivalence Studies. 3rd ed. Chapman & Hall/CRC Biostatistics Series 27. Boca Raton: CRC Press.

Clayton, D, and A Leslie. 1981. "The Bioavailability of Erythromycin Stearate versus Enteric-Coated Erythromycin Base When Taken Immediately before and after Food." Journal of International Medical Research 9 (6): 470-77. DOI:[10.1177/030006058100900608](https://doi.org/10.1177/030006058100900608).

Fuglsang, Anders, Helmut Schütz, and Detlew Labes. 2015. "Reference Datasets for Bioequivalence Trials in a Two-Group Parallel Design." The AAPS Journal 17 (2): 400-404. DOI:[10.1208/s12248-014-9704-6](https://doi.org/10.1208/s12248-014-9704-6).

Haidar, Sam H., Fairouz Makhoul, Donald J. Schuirmann, Terry Hyslop, Barbara Davit, Dale Conner, and Lawrence X. Yu. 2008. "Evaluation of a Scaling Approach for the Bioequivalence of Highly Variable Drugs." The AAPS Journal 10 (3): 450-54. DOI:[10.1208/s12248-008-9053-4](https://doi.org/10.1208/s12248-008-9053-4).

Patterson, Scott D, and Byron Jones. 2006. Bioequivalence and Statistics in Clinical Pharmacology. Boca Raton: Chapman & Hall/CRC. ISBN: 9781420034936.

Schütz, Helmut, Detlew Labes, and Anders Fuglsang. 2014. "Reference Datasets for 2-Treatment, 2-Sequence, 2-Period Bioequivalence Studies." The AAPS Journal 16 (6): 1292-97. DOI:[10.1208/s12248-014-9661-0](https://doi.org/10.1208/s12248-014-9661-0).