NCA Tutorial

Yingbo Ma

February 12, 2019

1 Introduction

This is an introduction to `NCA.jl`, a software for noncompartmental analysis (NCA). In this tutorial we will show how to use `NCA.jl` to analysis data.

1.1 Installation

Currently, `NCA.jl` is a submodule in `PuMaS.jl`, so you only need to install `PuMaS.jl`, and everything will be ready to go.

1.2 Getting Started

To load the package, use

```
using PuMaS.NCA
```

First, let's load the example NCA data inside `PuMaS.jl`. This data have 24 individuals, and each of them has 16 data points.

```
using PuMaS, CSV

file = PuMaS.example_nmtran_data("nca_test_data/dapa_IV")
data = CSV.read(file)
```

here is what the dataset looks like

```
first(data, 6) # take first 6 rows
```

	ID	TIME	TAD	CObs	AMT_IV	AMT_ORAL	Formulation
	Int64	Float64	Float64	Float64	Float64	Float64	String
1	1	0.0	0.0	157.021	5000.0	0.0	IV
2	1	0.05	0.05	141.892	0.0	0.0	IV
3	1	0.35	0.35	116.228	0.0	0.0	IV
4	1	0.5	0.5	109.353	0.0	0.0	IV
5	1	0.75	0.75	66.4814	0.0	0.0	IV
6	1	1.0	1.0	74.7532	0.0	0.0	IV

2 Efficient Computation of Multiple NCA Diagnostics

2.1 AUC and AUMC

We can compute the area under the curve (AUC) from the first observation time to infinity. Below we are accessing the concentration and corresponding time array for the first individual. By default, the `auc` function computes the AUC from initial time to infinity (AUCinf).

```
NCA.auc(data[:CObs][1:16], data[:TIME][1:16])
```

```
263.792662196049
```

```
NCA.auc(data[:CObs][1:16], data[:TIME][1:16], method=:linuplogdown)
```

```
257.8586273987722
```

the keyword argument `method` can be `:linear`, `:linuplogdown`, or `:linlog`, and it defaults to `:linear`. This is a simple interface, however it is not efficient if you want to compute many quantities. The recommended way is to create an `NCASubject` or an `NCAPopulation` object first and then call the respective NCA diagnostic on the data object. To parse data to an `NCAPopulation` object one can call the `parse_ncadata` function and give column names of `id`, `time`, `conc` (concentration), `amt` (dosage), `formulation`, `iv` (IV bolus name). Note that, by default, the lower limit of quantization (LLQ) is 0, and concentrations that are below LLQ (BLQ) are dropped. Also, we can add units by providing `timeu`, `concu`, and `amtu`.

```
timeu = u"hr"
concu = u"mg/L"
amtu  = u"mg"
pop = parse_ncadata(data, id=:ID, time=:TIME, conc=:CObs, amt=:AMT_IV,
    formulation=:Formulation, iv="IV",
  llq=0concu, timeu=timeu, concu=concu, amtu=amtu)
```

Here, each element of `pop` has the type `NCASubject`. It is a lazy data structure and actual computations are not performed. When we are instantiating `NCASubject`, it only performs data checking and cleaning. To calculate AUC, one can do:

```
NCA.auc(pop)
```

	id	auc
	Int64	Quantity
1	1	263.793 mg hr L^-1
2	2	323.253 mg hr L^-1
3	3	339.848 mg hr L^-1
4	4	373.361 mg hr L^-1
5	5	132.145 mg hr L^-1
6	6	303.86 mg hr L^-1
7	7	380.275 mg hr L^-1
8	8	279.126 mg hr L^-1
9	9	239.831 mg hr L^-1
10	10	260.862 mg hr L^-1
11	11	146.864 mg hr L^-1
12	12	359.489 mg hr L^-1
13	13	522.905 mg hr L^-1
14	14	262.988 mg hr L^-1
15	15	378.993 mg hr L^-1
16	16	206.926 mg hr L^-1
17	17	341.551 mg hr L^-1
18	18	195.925 mg hr L^-1
19	19	433.443 mg hr L^-1
20	20	214.27 mg hr L^-1
21	21	232.537 mg hr L^-1
22	22	471.515 mg hr L^-1
23	23	292.413 mg hr L^-1
24	24	170.305 mg hr L^-1

`AUClast` is the area under the curve from the first observation to the last observation. To compute `AUClast` on the second individual, one would do:

```
NCA.auc(pop[2], auctype=:last)
```

```
302.24594 mg hr L^-1
```

Or to compute the AUC on every individual, one would do:

```
NCA.auc(pop, auctype=:last)
```