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)
```

	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
7	1	2.0	2.0	39.1933	0.0	0.0	IV
8	1	3.0	3.0	25.4495	0.0	0.0	IV
9	1	4.0	4.0	13.0165	0.0	0.0	IV
10	1	6.0	6.0	3.81448	0.0	0.0	IV
11	1	8.0	8.0	1.47339	0.0	0.0	IV
12	1	10.0	10.0	1.10532	0.0	0.0	IV
13	1	12.0	12.0	0.911367	0.0	0.0	IV
14	1	16.0	16.0	0.830115	0.0	0.0	IV
15	1	20.0	20.0	0.624201	0.0	0.0	IV
16	1	24.0	24.0	0.653632	0.0	0.0	IV
17	2	0.0	0.0	59.7702	5000.0	0.0	IV
18	2	0.05	0.05	66.354	0.0	0.0	IV
19	2	0.35	0.35	55.507	0.0	0.0	IV
20	2	0.5	0.5	59.0243	0.0	0.0	IV
21	2	0.75	0.75	55.8154	0.0	0.0	IV
22	2	1.0	1.0	53.6728	0.0	0.0	IV
23	2	2.0	2.0	38.8955	0.0	0.0	IV
24	2	3.0	3.0	30.9587	0.0	0.0	IV
25	2	4.0	4.0	24.2407	0.0	0.0	IV
26	2	6.0	6.0	15.8675	0.0	0.0	IV
27	2	8.0	8.0	10.663	0.0	0.0	IV
28	2	10.0	10.0	7.32787	0.0	0.0	IV
29	2	12.0	12.0	5.83294	0.0	0.0	IV
30	2	16.0	16.0	3.30032	0.0	0.0	IV
31	2	20.0	20.0	2.32031	0.0	0.0	IV
32	2	24.0	24.0	1.71656	0.0	0.0	IV
33	3	0.0	0.0	165.733	5000.0	0.0	IV
34	3	0.05	0.05	130.022	0.0	0.0	IV
35	3	0.35	0.35	127.35	0.0	0.0	IV
36	3	0.5	0.5	97.7563	0.0	0.0	IV
37	3	0.75	0.75	86.6491	0.0	0.0	IV
38	3	1.0	1.0	81.895	0.0	0.0	IV
39	3	2.0	2.0	35.7601	0.0	0.0	IV
40	3	3.0	3.0	22.2936	0.0	0.0	IV
41	3	4.0	4.0	12.7924	0.0	0.0	IV
42	3	6.0	6.0	6.46943	0.0	0.0	IV
43	3	8.0	8.0	4.9808	0.0	0.0	IV
44	3	10.0	10.0	3.38318	0.0	0.0	IV
45	3	12.0	12.0	3.33256	0.0	0.0	IV
46	3	16.0	16.0	2.69385	0.0	0.0	IV
47	3	20.0	20.0	2.21823	0.0	0.0	IV
48	3	24.0	24.0	2.0378	0.0	0.0	IV
49	4	0.0	0.0	133.911	5000.0	0.0	IV
50	4	0.05	0.05	123.97	0.0	0.0	IV
51	4	0.35	0.35	111.766	0.0	0.0	IV
52	4	0.5	0.5	122.325	0.0	0.0	IV

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)
```

```
257.8792837435675
```

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)
```

```
Error: UndefVarError: parse_ncadata not defined
```

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)
```

```
Error: UndefVarError: pop not defined
```

`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)
```

```
Error: UndefVarError: pop not defined
```

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

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

```
Error: UndefVarError: pop not defined
```

One can also compute AUC on a certain interval. To compute AUC on the interval $[10, \infty]$ on the first individual

```
NCA.auc(pop[1], interval=(10,Inf).*timeu)
```

```
Error: UndefVarError: pop not defined
```

Note that we need to apply the time unit to the interval for units compatibility. One can also specify multiple intervals

```
NCA.auc(pop[1], interval=[(10,Inf).*timeu, (10, 15).*timeu])
```

```
Error: UndefVarError: pop not defined
```

In many cases, the AUC commands may need to extrapolate in order to cover the desired interval. To see the percentage of extrapolation ($\frac{\text{extrapolated AUC}}{\text{Total AUC}} \cdot 100$), you can use the command:

```
NCA.auc_extrap_percent(pop[1])
```

```
Error: UndefVarError: pop not defined
```

Area under the first moment of the concentration (AUMC) is

$$\int_{t_0}^{t_1} t \cdot \text{concentration}(t)dt.$$

The interface of computing AUMC is exactly the same with AUC, and one needs to change `auc` to `aumc` for calculating AUMC or related quantities. For instance,

```
NCA.aumc_extrap_percent(pop[1])
```

```
Error: UndefVarError: pop not defined
```

```
NCA.aumc(pop[1])
```

```
Error: UndefVarError: pop not defined
```

2.2 Terminal Rate Constant (λz)

The negative slope for concentration vs time in log-linear scale is the terminal rate constant, often denoted by λz. To compute λz, one can call

```
NCA.lambdaz(pop[1])
```

```
Error: UndefVarError: pop not defined
```

To get the coefficient of determination (r^2), the adjusted coefficient of determination ($adjr^2$), the y-intercept, the first time point used, and the number of points used while computing λz, one can do:

```
NCA.lambdazr2(pop)
```

```
Error: UndefVarError: pop not defined
```

```
NCA.lambdazadjr2(pop)
```

```
Error: UndefVarError: pop not defined
```

```
NCA.lambdazintercept(pop)
```

```
Error: UndefVarError: pop not defined
```

```
NCA.lambdaztimefirst(pop)
```

```
Error: UndefVarError: pop not defined
```

```
NCA.lambdaznpoints(pop)
```

```
Error: UndefVarError: pop not defined
```

By default, λz calculation checks last 10 or less data points, one can change it by providing the keyword `threshold`, e.g.

```
NCA.lambdaz(pop[1], threshold=2)
```

```
Error: UndefVarError: pop not defined
```

One can also specify the exact data points by passing their indices

```
NCA.lambdaz(pop[1], idxs=[10, 15, 16])
```

```
Error: UndefVarError: pop not defined
```

You can also pass their time points

```
NCA.lambdaz(pop[1], slopetimes=[1,2,3].*timeu)
```

```
Error: UndefVarError: pop not defined
```

2.3 Simple functions

`T_max` is the time point at which the maximum concentration (`C_max`) is observed, and they can be computed by:

```
NCA.tmax(pop[1])
```

```
Error: UndefVarError: pop not defined
```

```
NCA.cmax(pop[1])
```

```
Error: UndefVarError: pop not defined
```

```
NCA.cmax(pop[1], interval=(20, 24).*timeu)
```

```
Error: UndefVarError: pop not defined
```

```
NCA.cmax(pop[1], interval=[(20, 24).*timeu, (10, 15).*timeu])
```

```
Error: UndefVarError: pop not defined
```

Note that `cmax` returns `C_max` and normalized `C_max` if `dose` is provided. If `dose` is provided in the `NCASubject`, that `dose` will be used by all computations where dose can be used.

`T_last` is the time of the last observed concentration value above the lower limit of quantization (LLQ), and the corresponding concentration value is (`C_last`). They can be computed by the command

```
NCA.tlast(pop[1])
```

```
Error: UndefVarError: pop not defined
```

```
NCA.clast(pop[1])
```

```
Error: UndefVarError: pop not defined
```

The half-life can be computed by:

```
NCA.thalf(pop[1])
```

```
Error: UndefVarError: pop not defined
```

One may need to interpolate or to extrapolate the concentration-time data. For example, if you wanted to interpolate the concentration at $t = 12$ using linear interpolation, you would do:

```
NCA.interpextrapconc(pop[1], 12timeu, method=:linear)
```

```
Error: UndefVarError: pop not defined
```

`method` can be `:linear`, `:linuplogdown`, or `:linlog`.

3 Plots and Summary

To generate linear and log-linear plots, one can do:

```
using Plots # load the plotting library
plot(pop)
```

```
Error: UndefVarError: pop not defined
```

to only generate the linear plot:

```
plot(pop, loglinear=false)
```

```
Error: UndefVarError: pop not defined
```

Similarly, to generate log-linear plot:

```
plot(pop, linear=false)
```

```
Error: UndefVarError: pop not defined
```

To calculate all NCA quantities, one can do

```
report = NCAReport(pop)
```

```
Error: UndefVarError: pop not defined
```

The `NCAReport` object holds all quantities, and one can call `NCA.to_dataframe` to get a `DataFrame` object.

```
NCA.to_dataframe(report)
```

```
Error: UndefVarError: report not defined
```

4 Multiple dose

The interface of doing NCA with multiple doses is the same as doing single dose NCA. To load the data with multiple doses, one can do

```
multiple_doses_file = Pumas.example_nmtran_data("nca_test_data/dapa_IV_ORAL")
mdata = CSV.read(multiple_doses_file)

timeu = u"hr"
concu = u"mg/L"
amtu  = u"mg"
mpop = parse_ncadata(mdata, time=:TIME, conc=:COBS, amt=:AMT, formulation=:FORMULATION,
    occasion=:OCC,
                                     iv="IV", timeu=timeu, concu=concu, amtu=amtu)
```

```
Error: UndefVarError: parse_ncadata not defined
```

To plot:

```
plot(mpop)
```

```
Error: UndefVarError: mpop not defined
```

To compute AUC and λz:

```
NCA.auc(mpop)
```

```
Error: UndefVarError: mpop not defined
```

To get a summary, we need to provide a reference dose. In this example, we are going to let the first dose be the reference dose.

```
rep = NCAReport(mpop, ithdose=1)
```

```
Error: UndefVarError: mpop not defined
```

```
NCA.to_dataframe(rep)
```

```
Error: UndefVarError: rep not defined
```