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
...	...	...	...	...	...	...	...

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)
```

```
NCAPopulation (24 subjects):
  ID: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23, 24]
    concentration: mg L^-1
    time:          hr
    auc:           mg hr L^-1
    aumc:          mg hr^2 L^-1
    λz:            hr^-1
    dose:          mg
```

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	Unitful
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)
```

	id	auc
	Int64	Unitful
1	1	246.932 mg hr L^-1
2	2	302.246 mg hr L^-1
3	3	288.58 mg hr L^-1
4	4	333.804 mg hr L^-1
5	5	129.061 mg hr L^-1
6	6	291.951 mg hr L^-1
7	7	333.994 mg hr L^-1
8	8	259.967 mg hr L^-1
9	9	233.643 mg hr L^-1
10	10	242.719 mg hr L^-1
11	11	141.435 mg hr L^-1
12	12	311.005 mg hr L^-1
13	13	427.174 mg hr L^-1
14	14	246.329 mg hr L^-1
15	15	311.131 mg hr L^-1
16	16	196.672 mg hr L^-1
17	17	319.297 mg hr L^-1
18	18	185.399 mg hr L^-1
19	19	403.216 mg hr L^-1
20	20	202.64 mg hr L^-1
21	21	222.77 mg hr L^-1
22	22	364.756 mg hr L^-1
23	23	265.663 mg hr L^-1
24	24	156.806 mg hr L^-1

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)
```

```
27.824427196048966 mg hr L^-1
```

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])
```

```
2-element Array{Unitful.Quantity{Float64,M*T*L^-3,Unitful.FreeUnits{(mg, hr
, L^-1),M*T*L^-3,nothing}},1}:
 27.824427196048966 mg hr L^-1
          4.6593795 mg hr L^-1
```

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])
```

```
6.391564517256502
```

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

```
1411.6198735770822 mg hr^2 L^-1
```

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])
```

```
0.03876710923615265 hr^-1
```

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)
NCA.lambdazadjr2(pop)
NCA.lambdazintercept(pop)
NCA.lambdaztimefirst(pop)
NCA.lambdaznpoints(pop)
```

	id	lambdaznpoints
	Int64	Int64
1	1	5
2	2	3
3	3	5
4	4	6
5	5	5
6	6	10
7	7	4
8	8	4
9	9	7
10	10	4
11	11	6
12	12	5
13	13	4
14	14	6
15	15	3
16	16	6
17	17	4
18	18	5
19	19	7
20	20	3
21	21	5
22	22	3
23	23	4
24	24	3

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)
```

```
0.029877467931765923 hr^-1
```

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

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

```
0.10617388957053892 hr^-1
```

You can also pass their time points

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

```
0.5387479621404708 hr^-1
```

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

```
2-element Array{Unitful.Quantity{Float64,M*L^-3,Unitful.FreeUnits{(mg, L^-1
),M*L^-3,nothing}},1}:
 0.653632 mg L^-1
  1.10532 mg L^-1
```

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

```
0.653632 mg L^-1
```

The half-life can be computed by:

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

```
1.2865889604594312 hr
```

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)
```

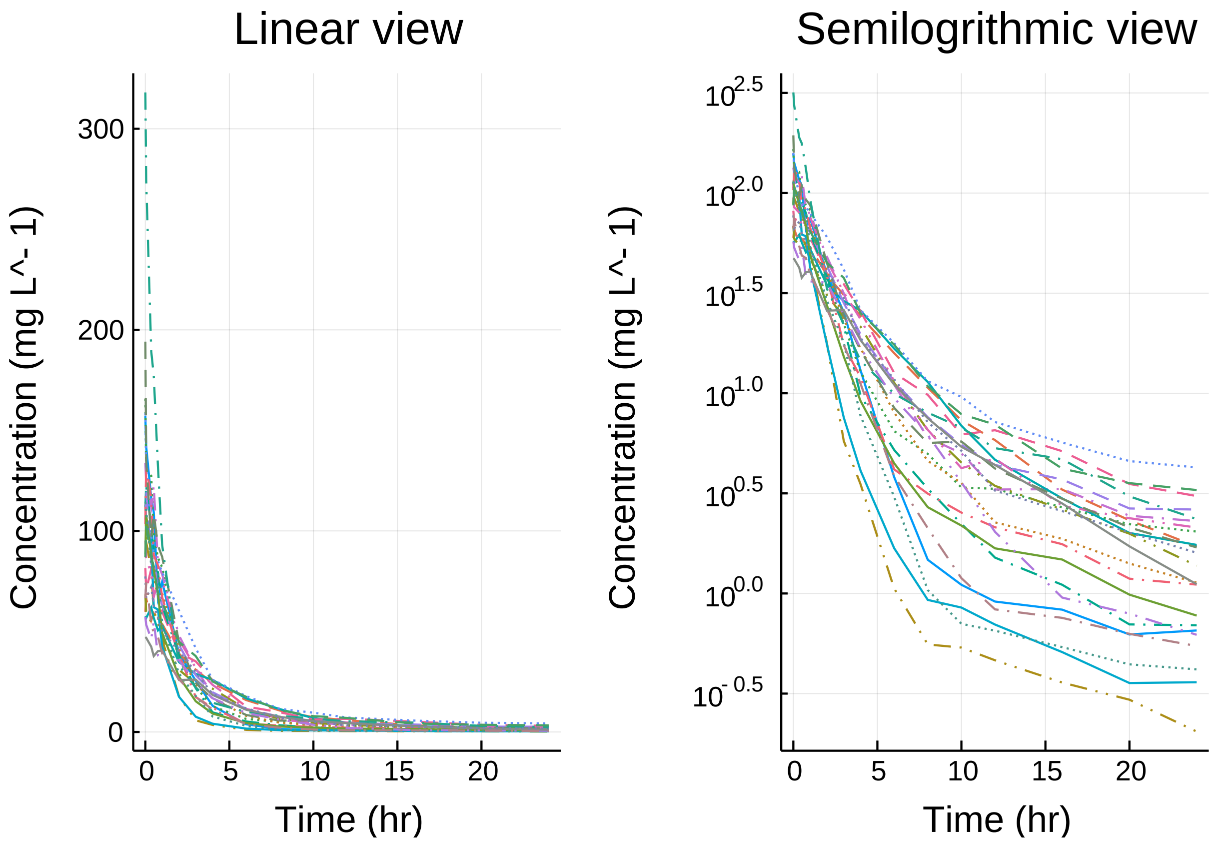
```
0.911367 mg L^-1
```

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

3 Plots and Summary

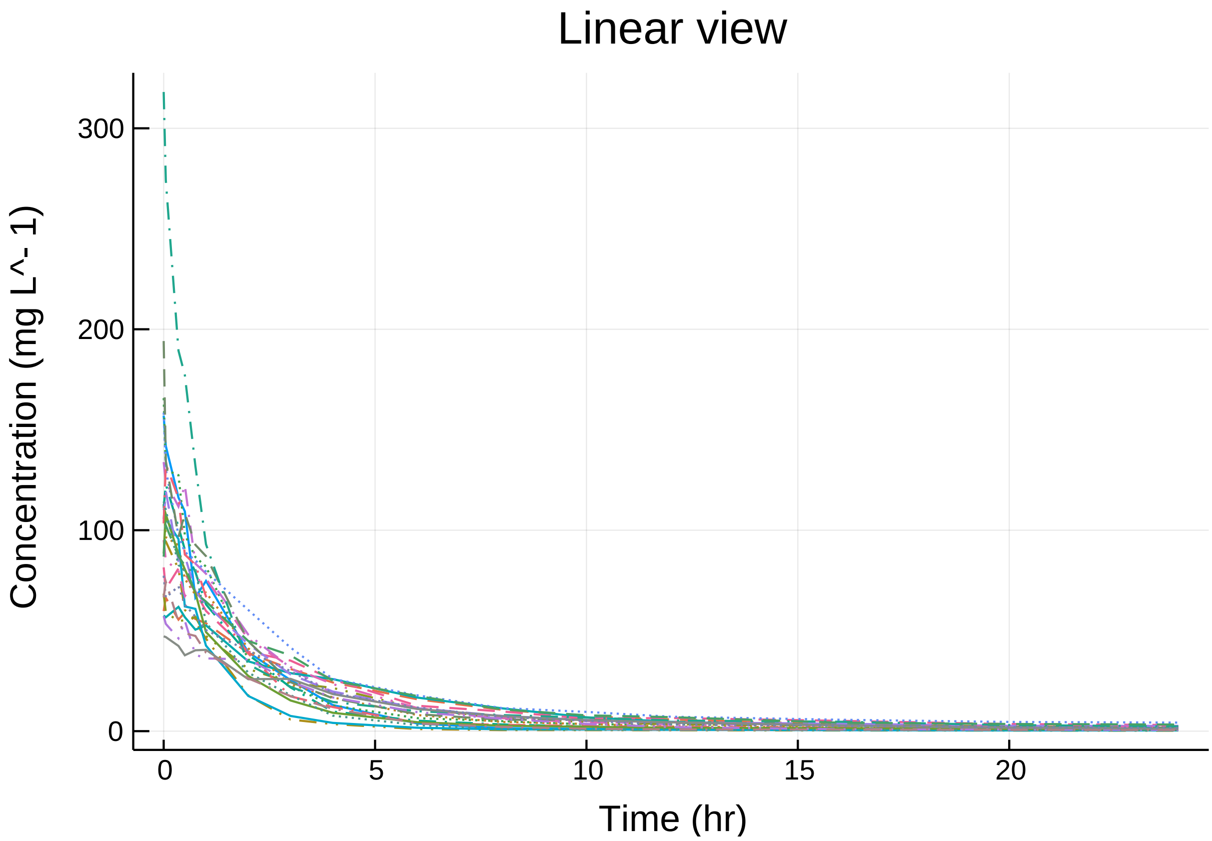
To generate linear and log-linear plots, one can do:

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



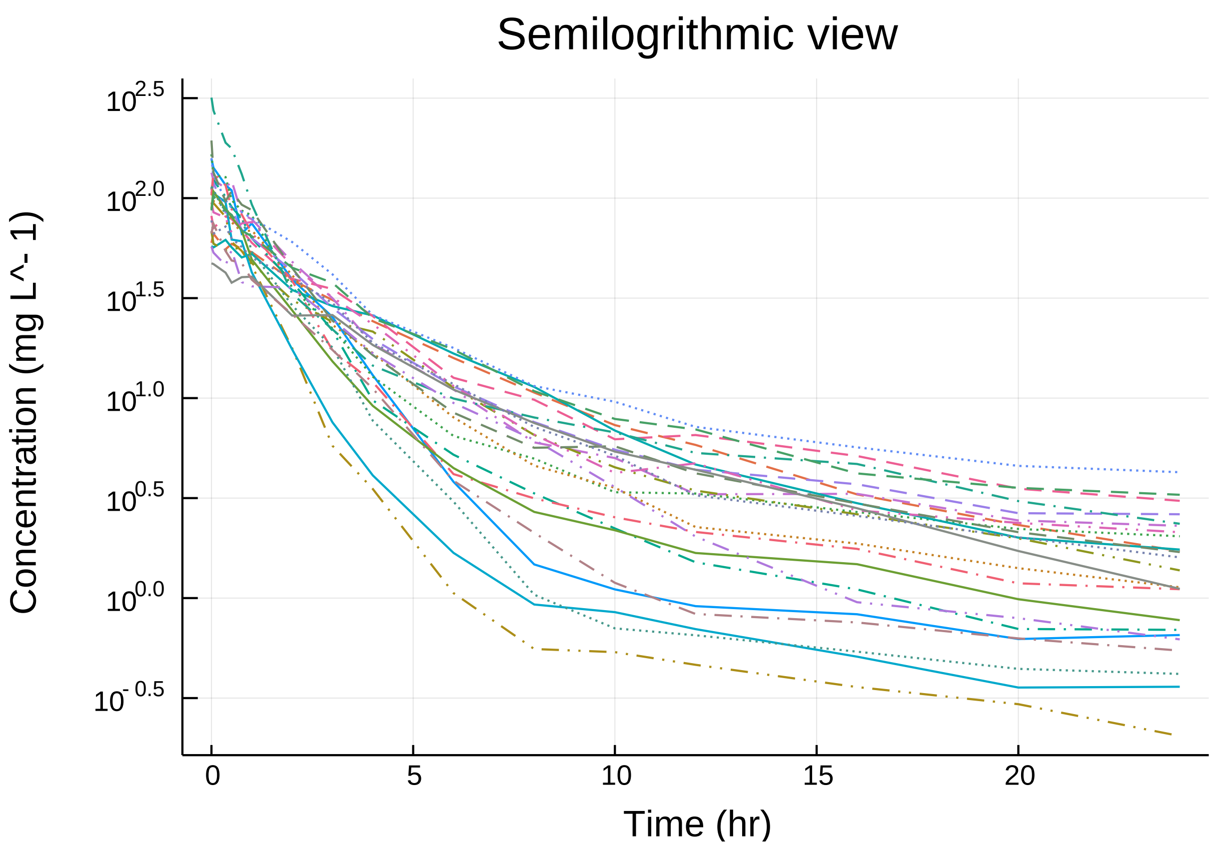
to only generate the linear plot:

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



Similarly, to generate log-linear plot:

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



To calculate all NCA quantities, one can do

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

```
NCAReport
  keys: (:lambdaz, :lambdazr2, :lambdazadjr2, :lambdazintercept, :lambdaznp
oints, :lambdaztimefirst, :cmax, :tmax, :cmin, :tmin, :c0, :clast, :tlast,
:thalf, :auc, :aumc, :auc_extrap_percent, :aumc_extrap_percent, :cl, :clf,
:vss, :vz, :tlag, :mrt, :fluctation, :accumulationindex, :swing, :bioav, :t
au, :cavg, :mat)
```

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

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

	id	lambdaz	lambdazr2	lambdazadjr2	lambdazintercept	lambdaznpoints	lambda
	Int64	Unitful	Float64	Float64	Float64	Int64	Ur
1	1	0.0387671 hr^{-1}	0.876585	0.835447	0.420709	5	10
2	2	0.0817121 hr^{-1}	0.99798	0.99596	2.49292	3	16
3	3	0.0397477 hr^{-1}	0.974958	0.96661	1.63628	5	10
4	4	0.0581041 hr^{-1}	0.875383	0.844229	2.12623	6	8.
5	5	0.0662631 hr^{-1}	0.990272	0.987029	0.0418686	5	10
6	6	0.146807 hr^{-1}	0.954561	0.948881	3.66418	10	2.
7	7	0.0662241 hr^{-1}	0.973581	0.960371	2.66532	4	12
8	8	0.0591092 hr^{-1}	0.994796	0.992194	1.54269	4	12
9	9	0.112094 hr^{-1}	0.914916	0.8979	2.03001	7	6.
10	10	0.0758397 hr^{-1}	0.994204	0.991306	2.16764	4	12
11	11	0.0663746 hr^{-1}	0.94937	0.936713	0.442621	6	8.
12	12	0.0541511 hr^{-1}	0.929973	0.90663	2.17602	5	10
13	13	0.0445008 hr^{-1}	0.955804	0.933706	2.4707	4	12
14	14	0.0664999 hr^{-1}	0.967734	0.959668	1.61075	6	8.
15	15	0.0314503 hr^{-1}	0.99291	0.985821	1.50681	3	16
16	16	0.0756395 hr^{-1}	0.980329	0.975412	1.53717	6	8.
17	17	0.0763209 hr^{-1}	0.991771	0.987657	2.32757	4	12
18	18	0.0396862 hr^{-1}	0.972902	0.963869	0.0346782	5	10
19	19	0.0778716 hr^{-1}	0.985955	0.983147	2.70697	7	6.
20	20	0.0534663 hr^{-1}	0.993466	0.986931	0.81816	3	16
21	21	0.113931 hr^{-1}	0.999478	0.999304	2.83811	5	10
22	22	0.0307751 hr^{-1}	0.959955	0.91991	1.91362	3	16
23	23	0.0597314 hr^{-1}	0.999868	0.999802	1.89977	4	12
24	24	0.0404818 hr^{-1}	0.994011	0.988021	0.36	3	16

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

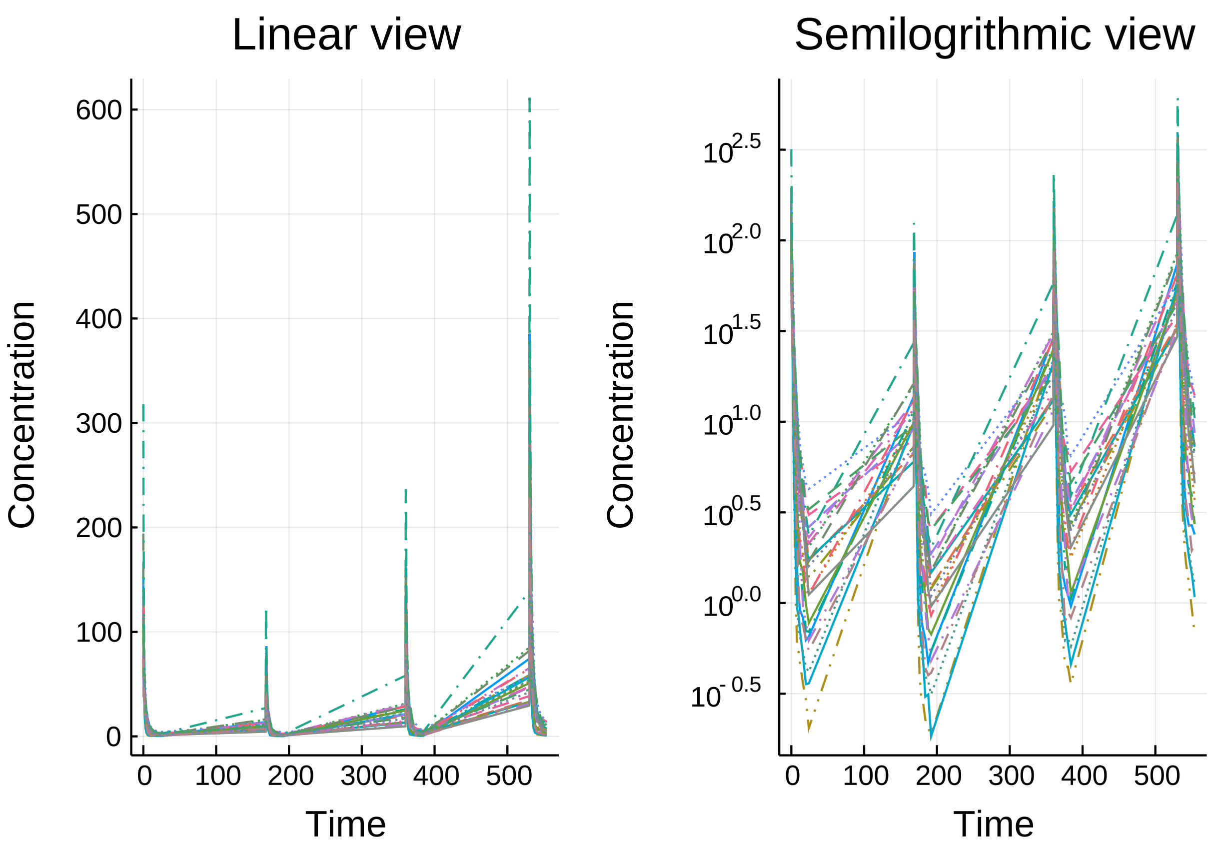
```
multiple_doses_file = PuMaS.example_nmtran_data("nca_test_data/dapa_IV_ORAL")
mdata = CSV.read(multiple_doses_file)

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)
```

```
NCAPopulation (24 subjects):
  ID: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23, 24]
    concentration: mg L^-1
    time:          hr
    auc:           mg hr L^-1
    aumc:          mg hr^2 L^-1
    λz:            hr^-1
    dose:          mg
```

To plot:

```
plot(mpop)
```



To compute AUC and λz:

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

	id	occasion	auc
	Int64	Int64	Unitful
1	1	1	263.793 mg hr L^-1
2	1	2	201.822 mg hr L^-1
3	1	3	421.23 mg hr L^-1
4	1	4	1047.38 mg hr L^-1
5	2	1	323.253 mg hr L^-1
6	2	2	252.802 mg hr L^-1
7	2	3	491.18 mg hr L^-1
8	2	4	1222.78 mg hr L^-1
9	3	1	339.848 mg hr L^-1
10	3	2	245.088 mg hr L^-1
11	3	3	481.385 mg hr L^-1
12	3	4	1187.11 mg hr L^-1
13	4	1	373.361 mg hr L^-1
14	4	2	302.196 mg hr L^-1
15	4	3	582.565 mg hr L^-1
16	4	4	1352.82 mg hr L^-1
17	5	1	132.145 mg hr L^-1
18	5	2	105.753 mg hr L^-1
19	5	3	196.432 mg hr L^-1
20	5	4	483.226 mg hr L^-1
21	6	1	303.86 mg hr L^-1
22	6	2	251.367 mg hr L^-1
23	6	3	462.525 mg hr L^-1
24	6	4	1119.53 mg hr L^-1
...	...	...	...

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)
NCA.to_dataframe(rep)
```

	id	occasion	lambdaz	lambdazr2	lambdazadjr2	lambdazintercept	lambdaznpoin
	Int64	Int64	Unitful	Float64	Float64	Float64	Int64
1	1	1	0.0387671 hr^{-1}	0.876585	0.835447	0.420709	5
2	1	2	0.192171 hr^{-1}	0.75655	0.726119	2.91033	10
3	1	3	0.0320026 hr^{-1}	0.997517	0.995033	0.727804	3
4	1	4	0.0244296 hr^{-1}	0.855536	0.783304	1.44802	4
5	2	1	0.0817121 hr^{-1}	0.99798	0.99596	2.49292	3
6	2	2	0.102657 hr^{-1}	0.9924	0.9886	2.62369	4
7	2	3	0.145337 hr^{-1}	0.944494	0.937556	4.2038	10
8	2	4	0.0919952 hr^{-1}	0.965349	0.953798	4.1059	5
9	3	1	0.0397477 hr^{-1}	0.974958	0.96661	1.63628	5
10	3	2	0.0478226 hr^{-1}	0.93242	0.909894	1.46381	5
11	3	3	0.0578421 hr^{-1}	0.946466	0.933083	2.3107	6
12	3	4	0.0621692 hr^{-1}	0.996529	0.993057	3.3814	3
13	4	1	0.0581041 hr^{-1}	0.875383	0.844229	2.12623	6
14	4	2	0.0566557 hr^{-1}	0.99282	0.985641	1.90658	3
15	4	3	0.0518813 hr^{-1}	0.944628	0.916943	2.40633	4
16	4	4	0.0815239 hr^{-1}	0.928549	0.910686	3.83982	6
17	5	1	0.0662631 hr^{-1}	0.990272	0.987029	0.0418686	5
18	5	2	0.0239926 hr^{-1}	0.984904	0.969808	-1.101	3
19	5	3	0.0664184 hr^{-1}	0.924111	0.905138	0.480106	6
20	5	4	0.0815214 hr^{-1}	0.999949	0.999898	1.55726	3
21	6	1	0.146807 hr^{-1}	0.954561	0.948881	3.66418	10
22	6	2	0.068555 hr^{-1}	0.998942	0.997884	2.03372	3
23	6	3	0.098923 hr^{-1}	0.946162	0.932703	3.31109	6
24	6	4	0.150275 hr^{-1}	0.946905	0.941005	5.1108	11
...	...	...	...	...	...	...	...